ETSI TS 126 445 V13.2.0 (2016-08)

TECHNICAL SPECIFICATION

Universal Mobile Telecommunications System (UMTS);
LTE;
Codec for Enhanced Voice Services (EVS);
Detailed algorithmic description
(3GPP TS 26.445 version 13.2.0 Release 13)

Reference

RTS/TSGS-0426445vd20

Keywords

LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or
print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any
existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the
print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

ETSI

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document is a detailed description of the signal processing algorithms of the Enhanced Voice Services coder.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1]   3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2]   3GPP TS 26.441: "Codec for Enhanced Voice Services (EVS); General Overview".

[3]   3GPP TS 26.442: "Codec for Enhanced Voice Services (EVS); ANSI C code (fixed-point)".

[4]   3GPP TS 26.444: "Codec for Enhanced Voice Services (EVS); Test Sequences".

[5]   3GPP TS 26.446: "Codec for Enhanced Voice Services (EVS); AMR-WB Backward Compatible Functions".

[6]   3GPP TS 26.447: "Codec for Enhanced Voice Services (EVS); Error Concealment of Lost Packets".

[7]   3GPP TS 26.448: "Codec for Enhanced Voice Services (EVS); Jitter Buffer Management".

[8]   3GPP TS 26.173: "ANSI-C code for the Adaptive Multi Rate - Wideband (AMR-WB) speech codec".

[9]   3GPP TS 26.190: "Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Transcoding functions".

[10]   3GPP TS 26.103: "Speech codec list for GSM and UMTS".

[11]   3GPP TS 26.290: "Extended AMR Wideband codec; Transcoding functions".

[12]   3GPP TS 26.401: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; General description".

[13]   3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction".

[14]   3GPP TS 22.105: "Services and Service capabilities".

[15]   IETF RFC 4867 (2007): "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", J. Sjoberg, M. Westerlund, A. Lakaniemi and Q. Xie.

[16]   IEEE Transactions on Acoustic, Speech and Signals Processing, Vol. 2, April, pp. 9-12 (1993): "Immittance spectral pairs (ISP) for speech encoding",Bistritz, Y., and Peller, S.

[17]   IEEE Journal on Selected Areas in Communications, Vol. 6, No. 2, February, pp. 314-323 (1988): "Transform Coding of Audio Signals Using Perceptual Noise Criteria", Johnston, J.D.

[18] IEEE Transactions on Acoustic, Speech and Signals Processing, Vol. 34, No. 6, December, pp. 1419 1426 (1986): "The computation of line spectral frequencies using Chebyshev polynomials", Kabal, P., and Ramachandran, R.P.

[19] IEEE Proceedings of International Conference on Acoustics, Speech and Signal Processing, March-April, pp. 4001-4004 (2008): "Transition mode coding for source controlled CELP codecs", Eksler, V., and Jelínek, M.

[20] 3GPP2 C.S0014-B, (2006) "Enhanced Variable Rate Codec, Speech Service 3 Option 3 and 68 for Wideband Spread Spectrum 4 Digital Systems."

[21] Speech and Audio Processing for Coding, Enhancement and Recognition, Springer (2014): Recent Speech Coding Technologies and Standards", Sinder, D., Varga, I., Krishnan, V. Rajendran, V, and Villette, S.

[22] Proceedings of EUSIPCO (2002): "LSF quantization with multiple scale lattice VQ for transmission over noisy channels", Vasilache, A. and Tabus, I.

[23] Proceedings of INTERSPEECH (2013): "Computationally efficient objective function for algebraic codebook optimization in ACELP", Bäckström, T.

[24] Proceedings of INTERSPEECH (2012): "Enumerative algebraic coding for ACELP", Bäckström, T.

[25] ITU-T Recommendation G.718 Amendment 2 (2010): "New Annex B on Superwideband scalable extension for ITU-T G.718 and corrections to main body fixed-point C-code and description text."

[26] Signal Processing, no. 82, pp 563-586 (2002): "Multiple-scale leader-lattice VQ with application to LSF quantization", Vasilache, A. and Tabus, I.

[27] IETF RFC 4566 (2006): "SDP: Session Description Protocol", M. Handley, V. Jacobson and C. Perkins.

[28] IETF RFC 3264 (2002): "An Offer/Answer Model with the Session Description Protocol (SDP)", J. Rosenberg and H. Schulzrinne.

[29] IETF RFC 5939 (2010): "Session Description Protocol (SDP) Capability Negotiation", F. Andreasen.

[30] IETF RFC 3550 (2003): "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson.

[31] IETF RFC 0768 (1980): "User Datagram Protocol", J. Postel.

[32] ITU-T Recommendation G.722 Amendment 1 (2010): "7 kHz audio-coding within 64 kbit/s: New Annex B with superwideband embedded extension."

[33] ITU-T Recommendation G.720.1 (2010): "Generic sound activity detector."

[34] American Mathematical Society, (2002): "Numerical Analysis: Mathematics of Scientific Computing", Kincaid, D. and Cheney, W.

[35] ITU-T Recommendation G.722.2 (2003): "Wideband coding of speech at around 16 kbit/s using Adaptive Multi-Rate Wideband (AMR-WB)."

[36] 3GPP TS 26.201: "Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Frame structure".

[37] 3GPP TS 26.171: "Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; General description".

[38] IETF RFC 3551 (2003): "RTP Profile for Audio and Video Conferences with Minimal Control", H. Schulzrinne and S. Casner.

[39] 3GPP TS 26.103: "Speech codec list for GSM and UMTS"

[40]   3GPP TS 26.244: "3GPP file format (3GP)".

[41]   IETF RFC 3839 (2004): "MIME Type Registrations for 3rd Generation Partnership Project (3GPP) Multimedia files", R. Castagno and D. Singer.

[42]   IEEE Transactions on Information Theory, Vol. 37, no. 6, Nov., pp. 1551-1566 (1991): "Trellis Coded Vector Quantization", T.R. Fischer, M.W. Marcellin, and M. Wang.

3   Definitions, abbreviations and mathematical expressions

3.1   Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

frame: an array of audio samples spanning 20ms time duration.

EVS codec: The EVS codec includes two operational modes: EVS Primary operational mode (EVS Primary mode) and EVS AMR-WB Inter-Operable modes (EVS AMR-WB IO mode). When using the EVS AMR-WB IO mode the speech frames are bitstream interoperable with the AMR-WB codec [9]. Frames generated by an EVS AMR-WB IO mode encoder can be decoded by an AMR-WB decoder, without the need for transcoding. Likewise, frames generated by an AMR-WB encoder can be decoded by an EVS AMR-WB IO mode decoder, without the need for transcoding.

EVS Primary mode: Includes 11 bit-rates for fixed-rate or multi-rate operation; 1 average bit-rate for variable bit-rate operation; and 1 bit-rate for SID (3GPP TS 26.441 [2]). The EVS Primary mode can encode narrowband, wideband, super-wideband and fullband signals. None of these bit-rates are interoperable with the AMR-WB codec.

EVS AMR-WB IO mode: Includes 9 codec modes and SID. All are bitstream interoperable with the AMR-WB codec (3GPP TS 26.171 [37]).

Operational mode: Used for the EVS codec to distinguish between EVS Primary mode and EVS AMR-WB IO mode.

3.2   Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

ACELP	Algebraic Code-Excited Linear Prediction
AMR	Adaptive Multi Rate (codec)
AMR-NB	Adaptive Multi Rate Narrowband (codec) – Also referred to as AMR
AMR-WB	Adaptive Multi Rate Wideband (codec)
AR	Auto-Regressive
AVQ	Algebraic Vector Quantization
CELP	Code-Excited Linear Prediction
CLDFB	Complex Low Delay Filter Bank
CMR	Codec Mode Request
CNG	Comfort Noise Generator
DCT	Discrete Cosine Transformation

DFT	Discrete Fourier Transform
DTX	Discontinuous Transmission
EFR	Enhanced Full Rate (codec)
EVS	Enhanced Voice Services
FB	Fullband
FCB	Fixed Codebook
FEC	Frame Erasure Concealment
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FT	Frame Type
GC	Generic Coding
HF	High Frequency
HP	High Pass
IIR	Infinite Impulse Response
IP	Internet Protocol
ISF	Immittance Spectral Frequency
ISP	Immittance Spectral Pair
ISPP	Interleaved Single-Pulse Permutation
JBM	Jitter Buffer Management
LD	Low Delay
LP	Linear Prediction
LPF	Low Pass Filter
LSB	Least Significant Bit
LSF	Line Spectral Frequency
LSP	Line Spectral Pair
LTP	Long-Term Prediction
MA	Moving Average
MDCT	Modified Discrete Cosine Transform
MRLVQ	Multi-Rate Lattice Vector Quantization
MSB	Most Significant Bit
MSVQ	Multi-Stage Vector Quantization
MTSI	Multimedia Telephony Service for IMS
NB	Narrowband
OL	Open-Loop
PCPRS	Path-Choose Pulse Replacement Search

PS	Packet Switched
PSTN	Public Switched Telephone Network
QMF	Quadrature Mirror Filter (See also CLDFB)
SAD	Signal Activity Detection
SDP	Session Description Protocol
SID	Silence Insertion Descriptor
SNR	Signal-to-Noise Ratio
SWB	Super Wideband
TC	Transition Coding
ToC	Table of Contents
UC	Unvoiced Coding
VC	Voiced Coding
VMR-WB	Variable Rate Multimode Wideband
VQ	Vector Quantization
WB	Wideband
WMOPS	Weighted Millions of Operations Per Second

3.3 Mathematical Expressions

For the purposes of the present document, the following conventions apply to mathematical expressions:

$\lfloor x \rfloor$ indicates the largest integer less than or equal to x: $\lfloor 1.1 \rfloor = 1$, $\lfloor 1.0 \rfloor = 1$ and $\lfloor -1.1 \rfloor = -2$;

$\lceil x \rceil$ indicates the smallest integer greater than or equal to x: $\lceil 1.1 \rceil = 2$, $\lceil 2.0 \rceil = 2$ and $\lceil -1.1 \rceil = -1$

$|x|$ indicates the absolute value of x: $|17| = 17$, $|{-}17| = 17$;

$\min(x_0,\dots x_{N-1})$ indicates the minimum of $x_0,\dots, x_{N-1}$, N being the number of components;

$\max(x_0,\dots x_{N-1})$ indicates the maximum of $x_0, \dots, x_{N-1}$;

$\mathrm{sgn}(x) \quad \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{otherwise;} \end{cases}$

$\mathbf{A}^{\mathrm{T}}$ indicates the transpose of matrix $\mathbf{A}$;

$x \bmod y$ indicates the remainder after dividing x by y: $x \bmod y = x - \left(y \lfloor x/y \rfloor \right)$;

$\mathrm{round}(x)$ is traditional rounding: $\mathrm{round}(x) = \mathrm{sgn}(x) \cdot \lfloor |x| + 0.5 \rfloor$;

$\exp(x)$ is equivalent to e^x where e is the base of the natural algorithm;

$\sum$ indicates summation;

$\prod$ indicates product;

Unless otherwise specified, $\log(x)$ denotes logarithm at base 10 throughout this Recommendation.

4 General description of the coder

4.1 Introduction

The present document is a detailed description of the signal processing algorithms of the Enhanced Voice Services coder. The detailed mapping from 20ms input blocks of audio samples in 16 bit uniform PCM format to encoded blocks of bits and from encoded blocks of bits to output blocks of reconstructed audio samples is explained. Four sampling rates are supported; 8 000, 16 000, 32 000 and 48 000 samples/s and the bit rates for the encoded bit stream of may be 5.9, 7.2, 8.0, 9.6, 13.2, 16.4, 24.4, 32.0, 48.0, 64.0, 96.0 or 128.0 kbit/s. An AMR-WB Interoperable mode is also provided which operates at bit rates for the encoded bit stream of 6.6, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s.

The procedure of this document is mandatory for implementation in all network entities and User Equipment (UE)s supporting the EVS coder.

The present document does not describe the ANSI-C code of this procedure. In the case of discrepancy between the procedure described in the present document and its ANSI-C code specifications contained in [3] the procedure defined by the [3] prevails.

4.2 Input/output sampling rate

The encoder can accept fullband (FB), superwideband (SWB), wideband (WB) or narrow-band (NB) signals sampled at 48, 32, 16 or 8 kHz. Similarly, the decoder output can be 48, 32, 16 or 8 kHz, FB, SWB, WB or NB.

4.3 Codec delay

The input signal is processed using 20 ms frames. The codec delay depends on the output sampling rate. For WB, SWB and FB output (i.e. output sampling rate > 8 kHz) the overall algorithmic delay is 32 ms. It consists of one 20 ms frame, 0.9375 ms delay of input resampling filters on the encoder-side, 8.75 ms for the encoder look-ahead, and 2.3125 ms delay of time-domain bandwidth extension on the decoder-side. For 8 kHz decoder output the decoder delay is reduced to 1.25 ms needed for resampling using a complex low-delay filterbank, resulting in 30.9375 ms overall algorithmic delay.

4.4 Coder overview

The EVS codec employs a hybrid coding scheme combining linear predictive (LP) coding techniques based upon ACELP (Algebraic Code Excited Linear Prediction), predominantly for speech signals, with a transform coding method, for generic content, as well as inactive signal coding in conjunction with VAD/DTX/CNG (Voice Activity Detection/Discontinuous Transmission/ Comfort Noise Generation) operation. The EVS codec is capable of switching between these different coding modes without artefacts.

The EVS codec supports 5.9kbps narrowband and wideband variable bit rate (VBR) operation based upon the ACELP coding paradigm which also provides the AMR-WB interoperable encoding and decoding. In addition to perceptually optimized waveform matching, the codec utilizes parametric representations of certain frequency ranges. These parametric representations constitute coded bandwidth extensions or noise filling strategies.

The decoder generates the signal parameters represented by the indices transmitted in the bit-stream. For the bandwidth extension and noise fill regions estimates from the coded regions are used in addition to the decoded parametric data to generate the signals for these frequency regions.

The description of the coding algorithm of this Specification is made in terms of bit-exact, fixed-point mathematical operations. The ANSI C code described in [3], which constitutes an integral part of this Specification, reflects this bit-exact, fixed-point descriptive approach. The mathematical descriptions of the encoder (clause 5), and decoder (clause 6), can be implemented in different ways, possibly leading to a codec implementation not complying with this Specification. Therefore, the algorithm description of the ANSI C code of [3] shall take precedence over the mathematical descriptions of clauses 5 and 6 whenever discrepancies are found. A non-exhaustive set of test signals that can be used with the ANSI C code are available described in [4].

4.4.1 Encoder overview

Figure 1 represents a high-level overview of the encoder. The signal resampling block corrects mismatches between the sampling frequency and the signal bandwidth command line parameter that is specified on the command line or through a file containing a bandwidth switching profile as explained in TS 26.442 and TS 26.443. For the case that the signal bandwidth is lower than half the input sampling frequency, the signal is decimated to a the lowest possible sampling rate out of the set of (8, 16, 32 kHz) that is larger than twice the signal bandwidth.
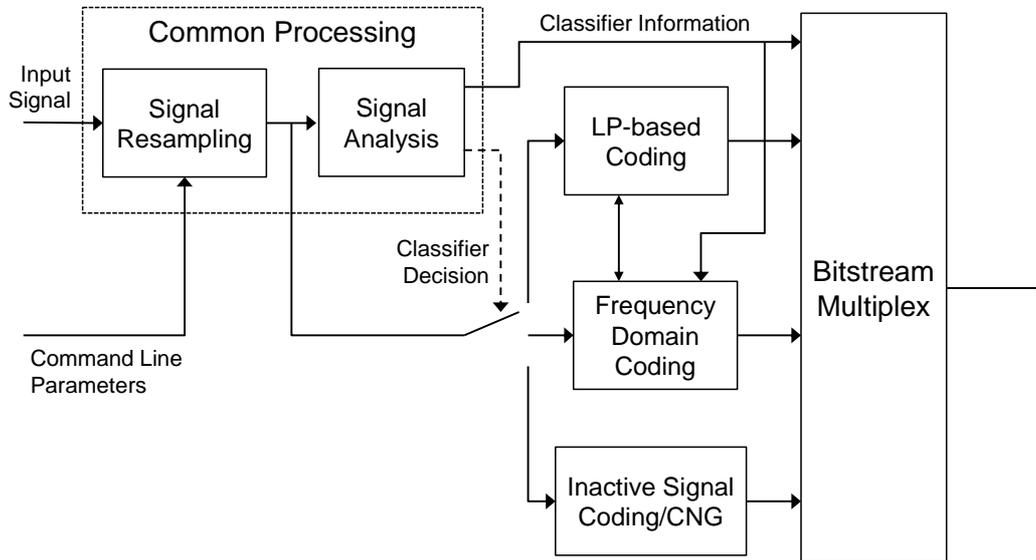
Figure 1: Encoder overview

The signal analysis determines which of three possible encoder strategies to employ: LP based coding (ACELP), frequency domain encoding and inactive coding (CNG). In some operational modes the signal analysis step includes a closed loop decision to determine which encoding method will result in the lowest distortion. Further parameters derived in the signal analysis aid the operation of these coding blocks and some of the analysis parameters, such as the coding strategy to be employed, are encoded into the bit-stream. In each of the coding blocks the signal analysis is further refined to obtain parameters relevant for the particular coding block.

The signal analysis and sub-sequent decision of the coding mode is performed independently for each 20ms frame and the switching between different modes is possible on a frame-by-frame basis. In the switching instance parameters are exchanged between the coding modes to ensure that the switching is as seamless as possible and closed-loop methods are sometimes employed in this case. In addition, switching between different bandwidths and/or bit-rates (including both the EVS Primary mode and the EVS AMR-WB IO mode) is possible on frame boundaries.

The signal analysis and all other blocks have full access to the command line parameters such as bit-rate, sampling rate, signal bandwidth, DTX activation as signalling information.

4.4.1.1 Linear Prediction Based Operation

The input signal is split into high frequency band and low frequency band paths; where the cut-off frequency between these two bands is determined from the operational mode (bandwidth and bit-rate) of the codec.

The linear-prediction coefficient estimation is performed for every 20ms frame. Within a frame, several interpolation points are established depending upon the bitrate and the optimum interpolation is transmitted to the decoder. The linear-prediction residual is further analysed and quantized using different quantization schemes dependent upon the nature of the residual. For the 5.9kbps VBR operation additional low-rate coding modes at rates conforming to the design constraints are employed.

The high-frequency portion of the signal is represented with several different parametric representations. The parameters used for this representation vary as a function of the bit-rate and the residual quantization strategy. The transmitted parameters include some or all of spectral envelope, energy information and temporal evolution information.

The LP based core can be configured so that both the linear prediction coefficients and the residual quantization are interoperable with the AMR-WB decoder. For this purpose the configuration of the LP coefficient estimation, parametric HF representation and the residual quantization is similar to those of AMR-WB. For the AMR-WB interoperable operation modes, identical codebooks to the AMR-WB quantizers are used.
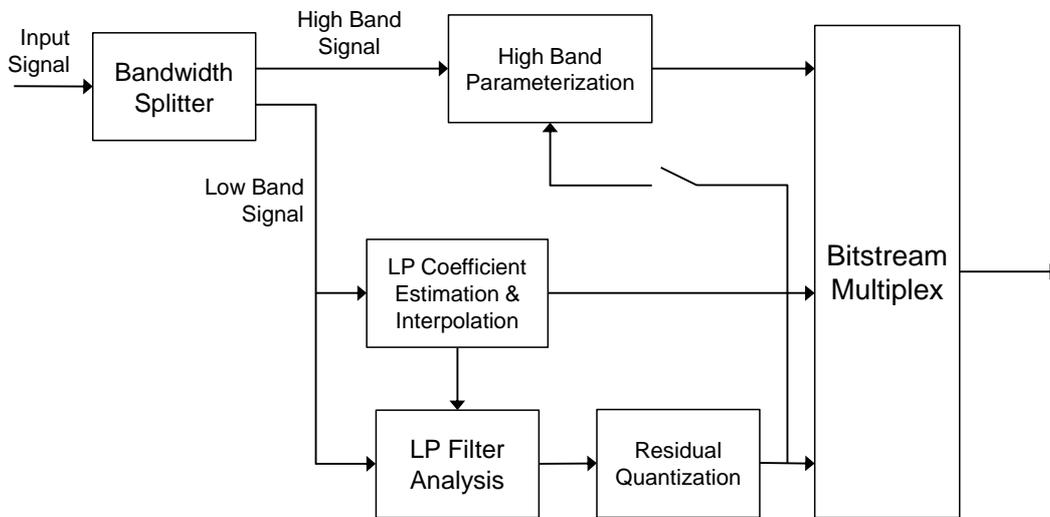
Figure 2: Linear prediction based operation including parametric HF representation

4.4.1.2   Frequency Domain Operation

For the frequency domain coding the encoding block can be envisaged as being separated into a control layer and a signal processing layer. The control layer performs signal analysis to derive several control and configuration parameters for the signal processing layer. The time-to-frequency transformation is based on the Modified Discrete Cosine Transform (MDCT) and provides adaptive time-frequency resolution. The control layer derives measures of the time distribution of the signal energy in a frame and controls the transform.

The MDCT coefficients are quantized using a variety of direct and parametric representations depending upon bit rate signal type and operating mode.
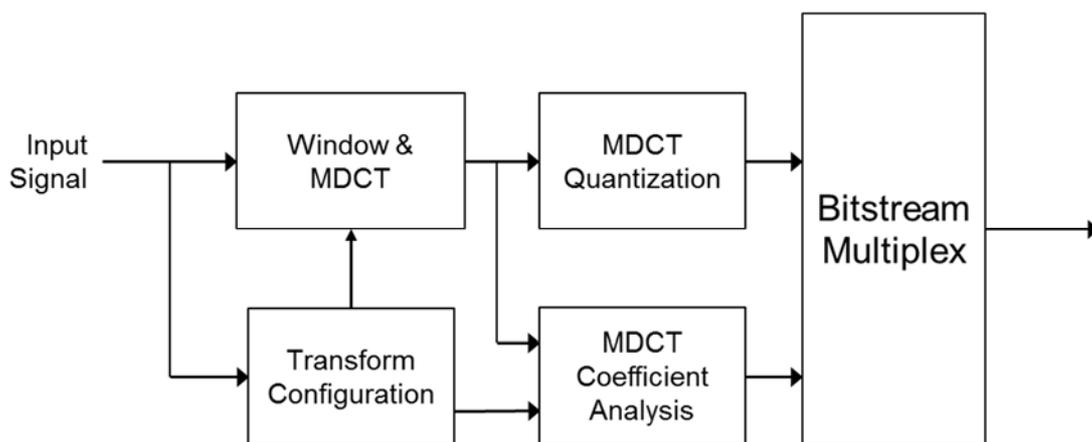
Figure 3: Frequency domain encoder

4.4.1.3   Inactive Signal coding

When the codec is operated in DTX on mode the signal classifier depicted in Figure 1 selects the discontinuous transmission (DTX) mode for frames that are determined to consist of background noise. For these frames a low-rate parametric representation of the signal is transmitted no more frequently than every 8 frames (160ms).

The low-rate parametric representation is used in the decoder for comfort noise generation (CNG) and includes parameters describing the frequency envelope of the background signal, energy parameters describing the overall energy and its time evolution.

4.4.1.4   Source Controlled VBR Coding

VBR coding describes a method that assigns different number of bits to a speech frame in the coded domain depending on the characteristics of the input speech signal [20] [21]. This method is often called source-controlled coding as well. Typically, a source-controlled coder encodes speech at different bit rates depending on how the current frame is classified, e.g., voiced, unvoiced, transient, or silence. Note that DTX operation can be combined with VBR coders in the same way as with Fixed Rate (FR) coders; the VBR operation is related to active speech segments.

The VBR solution provides narrowband and wideband coding using the bit rates 2.8, 7.2 and 8.0 kbps and produces an average bit rate at 5.9 kbps.

Due to the finer bit allocation, in comparison to Fixed Rate (FR) coding, VBR offers the advantage of a better speech quality at the same average active bit rate than FR coding at the given bit rate. The benefits of VBR can be exploited if the transmission network supports the transmission of speech frames (packets) of variable size, such as in LTE and UMTS networks.

4.4.2   Decoder overview

The decoder receives all quantized parameters and generates a synthesized signal. Thus, for the majority of encoder operations it represents the inverse of the quantized value to index operations.

For the AMR-WB interoperable operation the index lookup is performed using the AMR-WB codebooks and the decoder is configured to generate an improved synthesized signal from the AMR-WB bitstream.

4.4.2.1   Parametric Signal Representation Decoding (Bandwidth Extension)

In addition to the generation of signal components specifically represented by the transmitted indices, the decoder performs estimates of signal regions where the transmitted signal representation is incomplete, i.e. for the parametric signal representations and noise fill as well as blind bandwidth extension in some cases.

4.4.2.2   Frame loss concealment

The EVS codec includes frame loss concealment algorithms [21]. For all coding modes an extrapolation algorithm is in place that estimates the signal in a lost frame. For the LP based core this estimation operates on the last received residual and LP coefficients. For the frequency domain core in some cases the last received MDCT coefficients are extrapolated and in addition the resulting time domain signal is guaranteed to give a smooth time evolution from the last received frame into the missing frames.

Once the frame loss is recovered, i.e., the first good frame is received the codec memory is updated and frame boundary mismatches towards the last lost frame are minimized.

For situations of sustained frame loss the signal is either faded to background noise or its energy is reduced and finally muted when no reasonable extrapolation can be assumed.

The EVS codec also includes the "channel aware" mode, which may be employed for improved performance under packet loss conditions in a VoIP system. In the channel aware mode, partial copies (secondary frames) of the current speech frame are piggybacked on future speech frames (primary frames), without any increase in the total bit rate for the primary and secondary frames. If the current frame is lost, then its partial copy can be retrieved by polling the de-jitter buffer to enable faster and improved recovery from the packet loss.

4.4.3   DTX/CNG operation

The codec is equipped with a signal activity detection (SAD) algorithm for classifying each input frame as active or inactive. It supports a discontinuous transmission (DTX) operation in which the comfort noise generation (CNG) module is used to approximate and update the statistics of the background noise at a variable bit rate. Thus, the transmission rate during inactive signal periods is variable and depends on the estimated level of the background noise. However, the CNG update rate can also be fixed by means of a command line parameter.

4.4.3.1 Inactive Signal coding

When the codec is operated in DTX on mode the signal classifier depicted in Figure 1 selects the discontinuous transmission (DTX) mode for frames that are determined to consist of background noise. For these frames a low-rate parametric representation of the signal is transmitted no more frequently than every 8 frames (160ms).

The low-rate parametric representation is used in the decoder for comfort noise generation (CNG) and includes parameters describing the frequency envelope of the background signal, energy parameters describing the overall energy and its time evolution.

4.4.4 AMR-WB-interoperable option

As mentioned previously, EVS can operate in a mode which is fully interoperable with the AMR-WB codec bitstream.

4.4.5 Channel-Aware Mode

EVS offers partial redundancy [21] based error robust channel aware mode at 13.2 kbps for both wideband and super-wideband audio bandwidths.

In a VoIP system, packets arrive at the decoder with random jitters in their arrival time. Packets may also arrive out of order at the decoder. Since the decoder expects to be fed a speech packet every 20 ms to output speech samples in periodic blocks, a de-jitter buffer is required to absorb the jitter in the packet arrival time. The channel aware mode combines the presence of a de-jitter buffer with partial redundancy coding of a current frame which gets piggy backed onto a future frame. At the receiver, the de-jitter buffer is polled to check if a partial redundant copy of the current lost frame is available in any of the future frames. If present, the partial redundant information is used to synthesize the lost frame which offers significant quality improvements under low to high FER conditions. Source control is used to determine which frames of input can best be coded at a reduced frame rate (called primary frames) to accommodate the attachment of redundancy without altering the total packet size. In this way, the channel aware mode includes redundancy in a constant-bit-rate channel (13.2 kbps).

4.5 Organization of the rest of the Technical Standard

In clauses 5 and 6, detailed descriptions of the encoder and the decoder are given. Bit allocation is summarized in clause 7.

5 Functional description of the encoder

The description of the encoder is as follows. First, common pre-processing is described, and then the different elements of the encoder are described one by one. The discontinuous transmission (DTX) operation and the AMR-WB interoperable option are then given in separate subclauses, again referencing the same processing as in the default option.

5.1 Common processing

5.1.1 High-pass Filtering

The input audio signal $s_{inp}(n)$ sampled at 8, 16, 32 or 48 kHz, n being the sample index, is high-pass filtered to suppress undesired low frequency components. The transfer function of the HP filter has a cut-off frequency of 20 Hz (–3 dB) and is given by

$$H_{20Hz}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$ (1)

The coefficients of the HP filter for a given input sampling frequency are given in the table below.

Table 1: Coefficients of the 20Hz HP filer

	8 kHz	16 kHz	32 kHz	48 kHz
b_0	0.988954248067140	0.994461788958195	0.997227049904470	0.998150511190452
b_1	-1.977908496134280	-1.988923577916390	-1.994454099808940	-1.996301022380904
b_2	0.988954248067140	0.994461788958195	0.997227049904470	0.998150511190452
a_1	1.977786483776764	1.988892905899653	1.994446410541927	1.996297601769122
a_2	-0.978030508491796	-0.988954249933127	-0.994461789075954	-0.996304442992686

The input signal, filtered by the HP filter, is denoted as $s_{HP}(n)$.

5.1.2 Complex low-delay filter bank analysis

5.1.2.1 Sub-band analysis

The audio signal $s_{HP}(n)$ is decomposed into complex valued sub-bands by a complex modulated low delay filter bank (CLDFB). Depending on the input sampling rate sr_{HP}, the CLDFB generates a time-frequency matrix of 16 time slots and $L_C = sr_{HP}/800Hz$ sub-bands where the width of each sub-band is 400 Hz.

The analysis prototype w_c is an asymmetric low-pass filter with an adaptive length depending on sr_{HP}. The length of w_c is given by $L_{w_c} = 10 \cdot L_c$ meaning that the filter spans over 10 consecutive blocks for the transformation. The prototype of the LP filter has been generated for 48 kHz. For other input sampling rates, the prototype is obtained by means of interpolation so that an equivalent frequency response is achieved. Energy differences in the sub-band domain caused by different transformation lengths are compensated for by an appropriate normalization factors in the filter bank. The following figure shows the plot of the LP filter prototype w_c for sr_{HP} of 48 kHz.
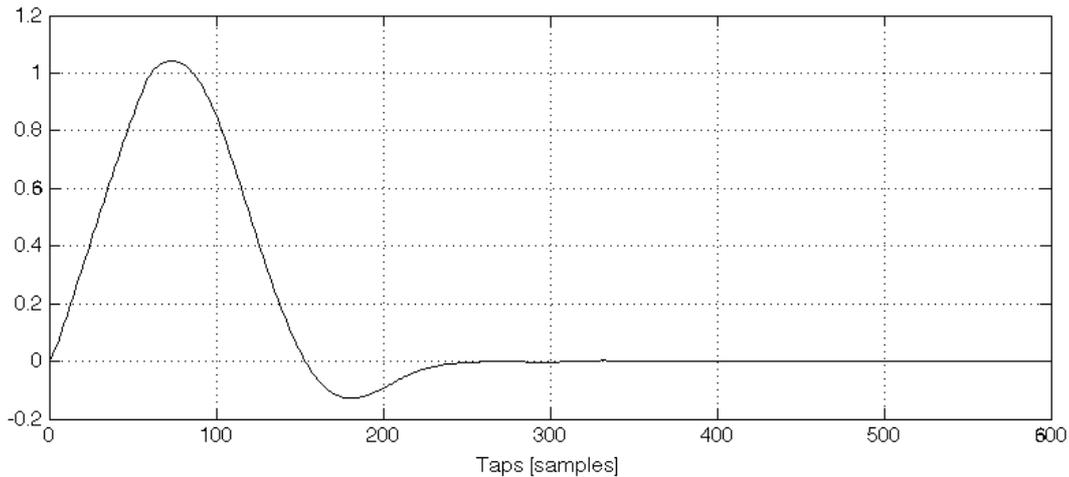
Figure 4 : Impulse response of CLDFB prototype filter with 600 taps for 48 kHz sample rate

The filter bank operation is described in a general form by the following formula:

$$X_{CR}(t,k) = \frac{80}{L_c} \cdot \sum_{n=-8L_C}^{n=2L_C-1} w_C(10L_C-1-i) \cdot s_{HP}(i+t \cdot L_C) \cos\left[\frac{\pi}{L_C}(n+n_0)\left(k+\frac{1}{2}\right)\right]$$

$$X_{CI}(t,k) = \frac{80}{L_c} \cdot \sum_{n=-8L_C}^{n=2L_C-1} w_C(10L_C-1-i) \cdot s_{HP}(i+t \cdot L_C) \sin\left[\frac{\pi}{L_C}(n+n_0)\left(k+\frac{1}{2}\right)\right] \tag{2}$$

where X_{CR} and X_{CI} are the real and the imaginary sub-band values, respectively, t is the sub-band time index with $0 \le t \le 15$, index i is defined as $i = n + 8L_C$, n_0 is the modulation offset of $n_0 = \frac{1}{2} - \frac{L_C}{2}$ and k is the sub-band index with $0 \le k \le L_C - 1$.

As the equations show, the filter bank is comparable to a complex MDCT but with a longer overlap towards the past samples. This allows for an optimized implementation of CLDFB by adopting DCT-IV and DST-IV frameworks.

5.1.2.2   Sub-band energy estimation

The energy in the CLDFB domain is determined for each time index t and frequency sub-band k by

$$E_C(t,k) = (X_{CR}(t,k))^2 + (X_{CI}(t,k))^2 \quad 0 \le t \le 15, \; 0 \le k \le L_C \tag{3}$$

Furthermore, energy per-band $\overline{E}_C(k)$ is calculated by summing up the energy values in all time slots. That is

$$\overline{E}_C(k) = \sum_{t=0}^{15} E_C(k,t) \quad 0 \le k \le L_C \tag{4}$$

In case $L_C > 20$, additional high frequency energy value E_{HF} $\overline{E}_{HF}$ is calculated for the frequency range from 8kHz to 16kHz by summing up $E_C(t,k)$ over one frame which is delayed by one time slot.

$$E_{HF} = \sum_{t=-1}^{14} \sum_{k=20}^{\min(40,L_C)} E_C(t,k) \tag{5}$$

E_{HF} is further scaled to an appropriate energy domain. In case the high bands are not active, E_{HF} is initialized to the maximum value.

5.1.3 Sample rate conversion to 12.8 kHz

The linear predictive (LP) analysis, the long-term prediction (LTP), the VAD algorithm and signal are performed at the 12.8 kHz sampling rate. The HP-filtered input signal $s_{HP}(n)$ is therefore converted from the input sampling frequency to 12.8 kHz.

5.1.3.1 Conversion of 16, 32 and 48 kHz signals to 12.8 kHz

For 16, 32 and 48 kHz signals, the sampling conversion is performed by first up-sampling the signal to 192 kHz, then filtering the output through a low-pass FIR filter $H_{6.4}(z)$ that has the cut-off frequency at 6.4 kHz. Then, the signal is down-sampled to 12.8 kHz. The filtering delay is 15 samples at 16 kHz sampling frequency which corresponds to 0.9375 ms.

The up-sampling is performed by inserting 11, 5 or 3 (for 16, 32 or 48 kHz, respectively) zero-valued samples between each 2 samples for each 20-ms frame of 320 samples (at 16 kHz sampling frequency)

$$s_{192}(n) = \begin{cases} s_{HP}(n/F_{up}), & \text{if } n/F_{up} = \lfloor n/F_{up} \rfloor \\ 0, & \text{otherwise} \end{cases}, \qquad 0 \le n < 3840 \qquad (6)$$

where $s_{192}(n)$ is the signal at 192 kHz sampling frequency and F_{up} is the up-sampling factor equal to 12 for a 16 kHz input, 6 for a 32 kHz input and 4 for a 48 kHz input. Then, the signal $s_{192}(n)$ is filtered through the LP filter $H_{6.4}(z)$ and decimated by 15 by keeping one out of 15 samples. The filter $H_{6.4}(z)$ is a 361-tap linear phase FIR filter having a cut-off frequency of 6.4 kHz in the 192 kHz up-sampled domain. The filtering and decimation can be done using the relation

$$s_{12.8}(n) = \frac{F_{up}}{15} \sum_{i=-180}^{180} s_{192}(15n+i)h_{6.4}(i), \qquad n = 0,...,255 \qquad (7)$$

where $h_{6.4}$ is the impulse response of $H_{6.4}(z)$. The operations in equations (6) and (7) can be implemented in one step by using only a part of the filter coefficients at a time with an initial phase related to the sampling instant n. That is

$$s_{12.8}(n) = \frac{F_{up}}{15} \sum_{i=-180/F_{up}+1}^{180/F_{up}} s_{HP}(\lfloor 15n/F_{up} \rfloor + i)h_{6.4}(F_{up}i - n \bmod F_{up}), \qquad n = 0,...,255 \qquad (8)$$

In case the encoder is externally forced to narrow-band processing of the input signal, the cut-off frequency of the LP filter is changed from 6.4 kHz to 4 kHz.

5.1.3.2 Conversion of 8 kHz signals to 12.8 kHz

For 8 to 12.8 kHz resampling a sharper resampling filter is beneficial. Double length low-pass FIR filter $H_{3.9}(z)$ is used in this case. The doubling of the impulse response length is compensated by a low delay resampling method. The filter $H_{3.9}(z)$ is a 241-tap linear phase FIR filter having a cut-off frequency of 3.9 kHz and is applied in the up-sampled domain which is 64 kHz. Direct FIR filtering with this filter would yield a delay of 120/64 = 1.875 ms. In order to reduce this delay to 0.9375 ms, future samples are determined at 8 kHz by adaptive linear prediction. The exact number of future samples is found based on the difference between the actual delay (1.875 ms) and the desired delay (0.9375 ms) at 8 kHz. Therefore $\lfloor (1.875 - 0.9375) * 8 \rfloor = 7$ future samples are predicted. These predicted samples are concatenated at the end of the current frame $s_{HP}(n)$ to form a support vector $s_{HPC}(n)$. Then, the sample rate conversion of $s_{HPC}(n)$ is performed in a similar way as for the other sampling rates, i.e. $s_{HPC}(n)$ is first up-sampled to 64 kHz, the output is filtered through the low-pass FIR filter $H_{3.9}(z)$ and the resulting signal is down-sampled to

12.8 kHz. The final filtering delay is aligned with that of the other resampling configurations, i.e 12 samples at 12.8 kHz sampling frequency which corresponds to 0.9375 ms.

To determine the future samples, linear prediction coefficients of order 16 are computed in the pre-emphasized domain in the following way. The last $L_{ss} = 120$ samples of the input frame $s_{HP}(n), n = 40,...,159$ at 8 kHz are windowed by an asymmetrical analysis window win_{ss_120}:

$$win_{ss\_120}(i) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{i2\pi}{223}\right) & i = 0,...,111 \\ \cos\left(\dfrac{(i-112)2\pi}{31}\right) & i = 112,119 \end{cases} \tag{9}$$

and a first order autocorrelation analysis is made on the windowed signal $s_{HPW}(n)$. The pre-emphasis coefficient μ_{ss} is obtained by

$$\mu_{ss} = r_w(1)/r_w(0) \tag{10}$$

where $r_w(0)$ and $r_w(1)$ are the autocorrelation coefficients

$$r_w(k) = \sum_{n=k}^{L_{ss}-1} s_{HPW}(n) s_{HPW}(n-k), \quad k = 0,1 \tag{11}$$

The last 120 samples of the signal $s_{HP}(n), n = 40,...,159$ are pre-emphasized using the adaptive filter

$$H_{pre-emph\_ss}(z) = 1 - \mu_{ss} z^{-1} \tag{12}$$

to obtain the pre-emphasized signal $s_{HP\_pre}(n)$ of $L_{ss} = 120$ samples. Then $s_{HP\_pre}(n), n = 0,...,119$ is windowed by the asymmetrical analysis window win_{ss_120} and a 16th order autocorrelation analysis is made on the windowed signal $s_{HP\_preW}(n)$

$$r_{pw}(k) = \sum_{n=k}^{L_{ss}-1} s_{HP\_preW}(n) s_{HP\_preW}(n-k), \quad k = 0,...,16 \tag{13}$$

These autocorrelation coefficients are lag-windowed by

$$r_{pwl}(k) = r_{pw}(k) \cdot w_{lag8k}(k), \quad k = 0,...,16 \tag{14}$$

where $w_{lag8k}(k)$ is defined as

$$w_{lag8k}(k) = \exp\left[-\frac{1}{2}\left(\frac{2\pi 60 k}{8000}\right)^2\right], \quad k = 0,...,16 \tag{15}$$

Based on the autocorrelation coefficients $r_{pwl}(k)$, the linear prediction coefficients $a_{ss}(k)$ are computed by the Levinson-Durbin algorithm. The future samples in the pre-emphasized domain $s_{HP\_pre}(n), \quad n = 120,...,126$ are predicted by zero input filtering through the $1/A_{ss}(z)$ synthesis filter

$$s_{HP\_pre}(k) = \sum_{j=1}^{16} -s_{HP\_pre}(k-j) \cdot a_{ss}(j), \quad k = 120,...,126 \tag{16}$$

Finally, the concatenated signal $s_{HP\_pre}(n)$ is de-emphasized through the filter $1/(1-\mu_{ss}z^{-1})$. Note that only the last 7 predicted samples need to be de-emphasized. These 7 de-emphasized samples are concatenated to $s_{HP}(n)$ (at positions $n = 160,\ldots,166$) to form the support vector $s_{HPC}(n)$.

The up-sampling of $s_{HPC}(n)$ is then performed by inserting 7 zero-valued samples between each 2 samples for each 20-ms frame of 160 samples (at 8 kHz sampling frequency) completed by 7 predicted future samples (167 in total)

$$s_{64}(n) = \begin{cases} s_{HPC}(n/8) & \text{if } n/8 = \lfloor n/8 \rfloor \\ 0 & \text{otherwise} \end{cases}, \qquad n = 0,\ldots,1335 \tag{17}$$

where $s_{64}(n)$ is the signal at 64 kHz sampling frequency. Then, the signal $s_{64}(n)$ is filtered through the LP filter $H_{3.9}(z)$ and decimated by 5 by keeping one out of 5 samples. The filtering and decimation can be done using the relation

$$s_{12.8}(n) = \frac{8}{5} \sum_{i=-120}^{120} s_{64}(5n+i-d_{8\_12\_64})h_{3.9}(i+120), \qquad n = 0,\ldots,255 \tag{18}$$

where $h_{3.9}$ is the impulse response of $H_{3.9}(z)$ and $d_{8\_12\_64} = 60$ assures that the index of s_{64} is never higher than the highest available index for (which is 1335). Indeed, it corresponds to the delay of this filtering at 64 kHz. To reduce complexity, the operations in equations (17) and (18) can be implemented in one step by using only a part of the filter coefficients at a time with an initial phase related to the sampling instant n. This polyphase implementation of the resampling filter is applied on the concatenated support vector. That is

$$s_{12.8}(n) = \frac{8}{5} \sum_{i=-14}^{15} s_{HPC}(\lfloor 5n/8 \rfloor + i - d_{8\_12\_8})h_{3.9}(8i - n\bmod 8 + 120), \qquad n = 0,\ldots,255 \tag{19}$$

where $d_{8\_12\_8} = 8$ is derived from the delay of this filtering at 8 kHz. It assures that the index of s_{HPC} is never higher than the highest available index (which is 166).

5.1.3.3 Conversion of input signals to 16, 25.6 and 32 kHz

If ACELP core is selected for WB, SWB or FB signals at bitrates higher than 13.2 kbps (see subclause 5.1.16), its internal sampling rate is set to 16 kHz rather than 12.8 kHz. If the input signal is sampled at 8 kHz, there is no conversion needed because for NB signals, ACELP core is always operated at 12.8 kHz. If the input signal is sampled at 16 kHz, no conversion is needed either and the input signal is only delayed by 15 samples which corresponds to 0.9375 ms. This is to keep all pre-processed signals aligned regardless of the bitrate or bandwidth. Thus, the input signal is resampled to 16 kHz only if its sampling frequency is 32 or 48 kHz.

The resampling operation is done in the same way as for the case of 12.8 kHz (see subclause 5.1.3.1), i.e. by means of FIR filtering. The coefficients of the LP filter are different but the filtering delay is still the same, i.e. 0.9375 ms.

The resampled signal is denoted $s_{16}(n)$ where $n=0,..,319$.

The input signal is converted to 25.6 kHz at 48 kbps and to 32 kHz at 96 or 128kbps but only for SWB and FB signals. The sampling conversion is again similar as in the case of 12.8 kHz with differences in LP filter coefficients. The resampled signals are denoted $s_{25.6}(n)$ and $s_{32}(n)$, respectively.

5.1.4 Pre-emphasis

A first-order high-pass filter is used to emphasize higher frequencies of the input signal and it is given by

$$H_{pre-emph}(z) = 1 - \beta_{pre-emph}z^{-1} \tag{20}$$

where $\beta_{pre-emph}$ is the pre-emphasis factor which is set to 0.68. The input signal to the pre-emphasis filter is $s_{12.8}(n)$ and the output signal is denoted $s_{pre}(n)$.

If ACELP core is selected for WB, SWB or FB signals at bitrates higher than 13.2 kbps (see subclause 5.1.16), its internal sampling rate is 16kHz rather than 12.8kHz. In this case, the pre-emphasis is re-done at the end of the pre-processing chain, on $s_{16}(n)$ with $\beta_{pre-emph} = 0.72$. The resulting signal is denoted $s_{pre16}(n)$.

If MDCT-based TCX is selected for SWB or FB high-rate LPC configurations, $\beta_{pre-emph} = 0.9$ is used as pre-emphasis factor when pre-emphasis is applied to signals at a sampling rate higher than 16kHz.

5.1.5 Spectral analysis

Spectral analysis is used in the encoder for signal activity detection (SAD) and signal classification functions. The discrete Fourier transform (DFT) is used to perform the spectral analysis and spectral energy estimation.

5.1.5.1 Windowing and DFT

The frequency analysis is done twice per frame using 256-point fast Fourier transform (FFT) with a 50% overlap. The centre of the first window is placed 96 samples past the beginning of the current frame. The centre of the second window is placed 128 samples farther, i.e., in the middle of the second subframe of the current frame. A square root of a Hanning window (which is equivalent to a sine window) is used to weight the input signal for the frequency analysis. The square root Hanning window is given by

$$w_{FFT}(n) = \sqrt{0.5 - 0.5\cos\left(\frac{2\pi n}{L_{FFT}}\right)} = \sin\left(\frac{\pi n}{L_{FFT}}\right), \qquad n = 0,...,L_{FFT} - 1 \tag{21}$$

where $L_{FFT} = 256$ is the size of FFT analysis. Note that only half of the window is computed and stored since it is symmetric (from 0 to $L_{FFT}/2$).
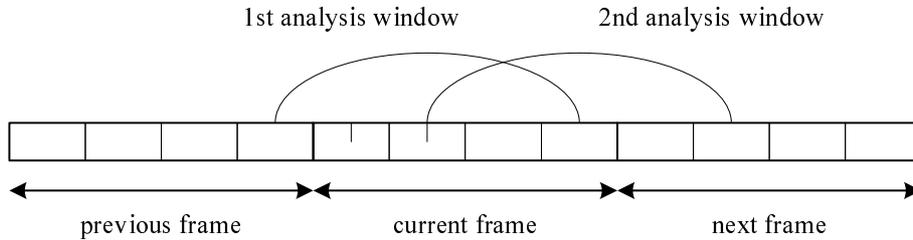
Figure 5: Relative positions of the spectral analysis windows

The windowed signal for both spectral analyses is obtained as:

$$\begin{aligned} s_{wnd}^{[0]}(n) &= w_{FFT}(n)s_{pre}(n), & n = 0, ,, L_{FFT} - 1, \\ s_{wnd}^{[1]}(n) &= w_{FFT}(n)s_{pre}(n + L_{FFT}/2), & n = 0, ,, L_{FFT} - 1, \end{aligned} \tag{22}$$

where $s_{pre}(n)$ is the pre-emphasized input signal ($s_{pre}(0)$ is the first sample in the current frame). The superscripts [0] and [1] used to denote the first and the second frequency analysis, respectively, are dropped for simplicity. An FFT is performed on both windowed signals to obtain two sets of spectral parameters per frame:

$$X(k) = \sum_{n=0}^{N-1} s_{wnd}(n) e^{-j2\pi kn/N}, \qquad k = 0,...,L_{FFT} - 1 \tag{23}$$

The output of the FFT provides the real and the imaginary parts of the spectrum denoted as $X_R(k)$, $k = 0,...,128$ and $X_I(k)$, $k = 1,...,127$. Note, that $X_R(0)$ corresponds to the spectrum at 0 Hz (DC) and $X_R(128)$ corresponds to the spectrum at 6400 Hz. The spectrum at these points is only real-valued and usually ignored in the subsequent analysis.

After the FFT analysis, the resulting spectrum is divided into critical bands [17] using the intervals having the following limits (20 bands in the frequency range 0-6400 Hz):

Table 2: Critical bands

band	f_l	f_h	M_{CB}
0	0	100	2
1	100	200	2
2	200	300	2
3	300	400	2
4	400	510	2
5	510	630	2
7	630	770	3
6	770	920	3
8	920	1080	3
9	1080	1270	4
10	1270	1480	4
11	1480	1720	5
12	1720	2000	6
13	2000	2320	6
14	2320	2700	8
15	2700	3150	9
16	3150	3700	11
17	3700	4400	14
18	4400	5300	18
19	5300	6350	21

The 256-point FFT results in a frequency resolution of 50 Hz (i.e., 6400/128 Hz). Thus, after ignoring the DC component of the spectrum, the number of frequency bins per critical band are given in the last column, denoted M_{CB}.

5.1.5.2 Energy calculations

The spectral analysis module also calculates several energy-related parameters. For example, an average energy per critical band is computed as

$$E_{CB}(i) = \frac{1}{(L_{FFT}/2)^2 M_{CB}(i)} \sum_{k=0}^{M_{CB}(i)-1} \left(X_R^2(k+j_i) + X_I^2(k+j_i) \right), \qquad i = 0,...,19 \qquad (24)$$

where $X_R(k)$ and $X_I(k)$ are, respectively, the real and the imaginary parts of the k-th frequency bin and j_i is the index of the first bin in the ith critical band given by $j_i = \{1, 3, 5, 7, 9, 11, 13, 16, 19, 22, 26, 30, 35, 41, 47, 55, 64, 75, 89, 107\}$. Furthermore, energy per frequency bin, $E_{BIN}(k)$, is calculated as

$$E_{BIN}(k) = \frac{4}{L_{FFT}^2} \left(X_R^2(k) + X_I^2(k) \right), \qquad k = 0,...,127 \qquad (25)$$

Finally, the spectral analysis module computes the average total energy for both FFT analyses in a 20 ms frame by summing the average critical band energies E_{CB}. That is, the spectrum energy for the first spectral analysis window is computed as

$$E_{frame}^{[0]} = \sum_{i=0}^{19} E_{CB}(i) \qquad (26)$$

and, similarly, the second frame energy, denoted as $E_{frame}^{[1]}$.

The total frame energy (in dB) is computed as the average of the two frame energies. That is

$$E_t = 10 \log \left(0.5 \left(E_{frame}^{[0]} + E_{frame}^{[1]} \right) \right) \tag{27}$$

The total energy per frequency bin (power spectrum) is calculated as

$$PS(k) = 0.5 \left(E_{BIN}^{[0]}(k) + E_{BIN}^{[1]}(k) \right), \qquad k = 0,...,127 \tag{28}$$

The output parameters of the spectral analysis module (both spectral analyses), that is the average energy per critical band, the energy per frequency bin and the total energy in dB, are used in several subsequent functions.

Note that, for narrow band inputs sampled at 8 kHz, after sampling conversion to 12.8 kHz, there is no content at both ends of the spectrum. Thus, the lowest critical band as well as the last three critical bands are not considered in the computation of output parameters (only bands from $i = 1,...,16$ are considered).

In addition to the absolute frame energy E_t, calculated in (27), relative energy of the frame is calculated as the difference between the total frame energy in dB and the long-term active signal energy $\bar{E}_t$. The relative frame energy is given by

$$E_{rel} = E_t - \bar{E}_{sp} \tag{29}$$

The long-term active signal energy is updated only during active frames (explained in subclause 5.1.12.5). Note that the long-term active signal energy $\bar{E}_{sp}$ is updated only after the signal activity detection module.

5.1.6   Bandwidth detection

A detection algorithm is applied to detect the actual input audio bandwidth for input sampling rates greater than 8 kHz. This bandwidth information is used to run the codec in its optimal mode, tailored for a particular bandwidth (BW) rather than for a particular input sampling frequency. For example, if the input sampling frequency is 32 kHz but there is no "energetically" meaningful spectral content above 8 kHz, the codec is operated in the WB mode. The following bandwidths/modes are used throughout the EVS codec: NB (0-4kHz), WB (0-8kHz), SWB (0-16kHz) and FB (0-20 kHz).

The detection algorithm is based on computing energies in spectral regions and comparing them to certain thresholds. The bandwidth detector operates on the CLDFB values (see subclause 5.1.2). In the AMR-WB IO mode, the bandwidth detector uses a DCT transform to determine the signal bandwidth.

5.1.6.1   Mean and maximum energy values per band

The CLDFB energy vector $\bar{E}_C$ computed per 400Hz frequency bins (see subclause 5.1.2.2), is further aggregated as described below. Each value of $E_{C4}(i)$ represents a 1600Hz band consisting of four CLDFB energy bins summed up from k_{start} to k_{stop}.

$$E_{C4}(i) = \sum_{k_{start}}^{k_{stop}} \bar{E}_C(k), \quad i = 0,..,8 \tag{30}$$

Depending on the input sampling frequency up to nine CLDFB bands are calculated using the above equation and the values are given below:

Table 3: CLDFB bands for energy calculation

i	k_{start}	k_{stop}	bandwidth in kHz	bandwidth index
0	3	6	1.2 – 2.8	NB
1	11	14	4.4 – 7.2	WB
2	14	17		
3	23	26	9.2 – 15.6	SWB
4	27	30		
5	31	34		
6	35	38		
7	42	45	16.8 – 20.0	FB
8	46	49		

The values in CLDFB bands are converted to the log domain and scaled by

$$E_{C4}(i) = \log_{10}\left[\frac{E_{C4}(i)}{f_{scl}^2 \cdot 8}\right], \qquad i = 0,..,8 \tag{31}$$

where f_{scl} is set according to the input sampling frequency as follows: 88.293854 for 8kHz, 88.300926 for 16kHz, 88.304118 for 32kHz and 88.028412 for 48kHz.

The per-band CLDFB energy is then used to calculate the mean energy values per bandwidth:

$$
\begin{aligned}
E_{NB} &= E_{C4}(0) + 1.6 \\
E_{WB} &= \frac{1}{2}\sum_{i=1}^{2} E_{C4}(i) + 1.6 \\
E_{SWB} &= \frac{1}{4}\sum_{i=3}^{6} E_{C4}(i) + 1.6 \\
E_{FB} &= \frac{1}{2}\sum_{i=7}^{8} E_{C4}(i) + 1.6
\end{aligned}
\tag{32}
$$

and the maximum energy values per bandwidth:

$$
\begin{aligned}
E_{NB,\max} &= E_{NB} + 1.6 \\
E_{WB,\max} &= \max_{i=1,2}(E_{C4}(i)) + 1.6 \\
E_{SWB,\max} &= \max_{i=3,..,6}(E_{C4}(i)) + 1.6 \\
E_{FB,\max} &= \max_{i=7,8}(E_{C4}(i)) + 1.6
\end{aligned}
\tag{33}
$$

In case of the DCT based detector, the DCT values are computed by first applying a Hanning window on the 320 samples of the input audio signal sampled at input sampling rate. Then the windowed signal is transformed to the DCT domain and finally decomposed into several bands as shown in Table 3a.

Table 3a: DCT bands for energy calculation

i	bandwidth in kHz	bandwidth index
0	1.5 – 3.0	NB
1	4.5 – 7.5	WB
2		
3	9.0 – 15.0	SWB
4		
5		
6		
7	16.5 – 19.5	FB
8		

The values in DCT bands are converted to the log domain by

$$E_{C4}(i) = \log_{10}\left[E_{C4}(i)\right], \qquad i = 0,...,8 \tag{33a}$$

and per-band and maximum energies are computed using (32) and (33) while the constant 1.6 in these equations is omitted in case of the DCT based detector.

5.1.7 Bandwidth decision

The following decision logic is identical for CLDFB and DCT versions of energy calculations, except for some constants which were adapted to get similar detection results.

The long-term CLDFB energy mean values for NB, WB and SWB are updated as follows:

$$\overline{E}_{NB} = 0.75\overline{E}_{NB}^{[-1]} + 0.25E_{NB}$$
$$\overline{E}_{WB} = 0.75\overline{E}_{WB}^{[-1]} + 0.25E_{WB} \tag{34}$$
$$\overline{E}_{SWB} = 0.75\overline{E}_{SWB}^{[-1]} + 0.25E_{SWB}$$

where the superscript [-1] has been used to denote the value of $\overline{E}_{NB}$ in the previous frame. The update takes place only if the local SAD decision is active and only if the long-term background noise level, $\overline{N}_t$, is higher than 30 dB.

The values are compared to certain thresholds also taking the current maximum values into account, which leads to increasing or decreasing counters for each bandwidth as described below in the flowchart.
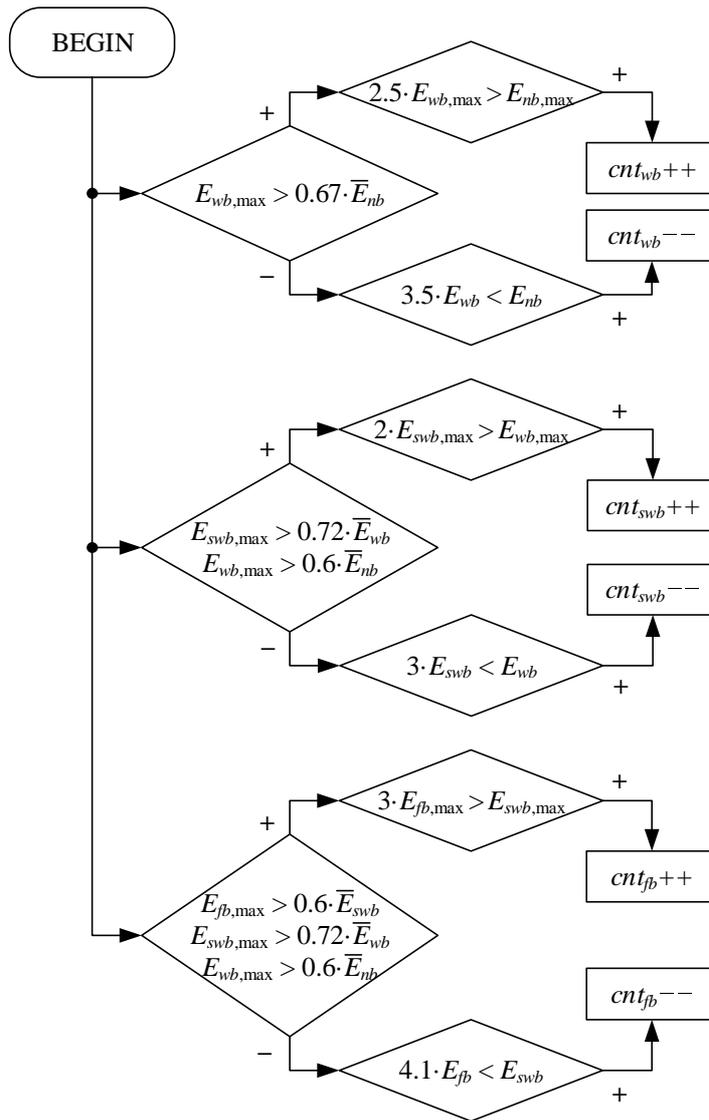
Figure 6: Increasing and decreasing of BW counters

The tests in the above diagram are performed sequentially from top to bottom. The BW counters are then used to decide the actual signal bandwidth, *BW*, according to the logic described in the following schematic diagram.
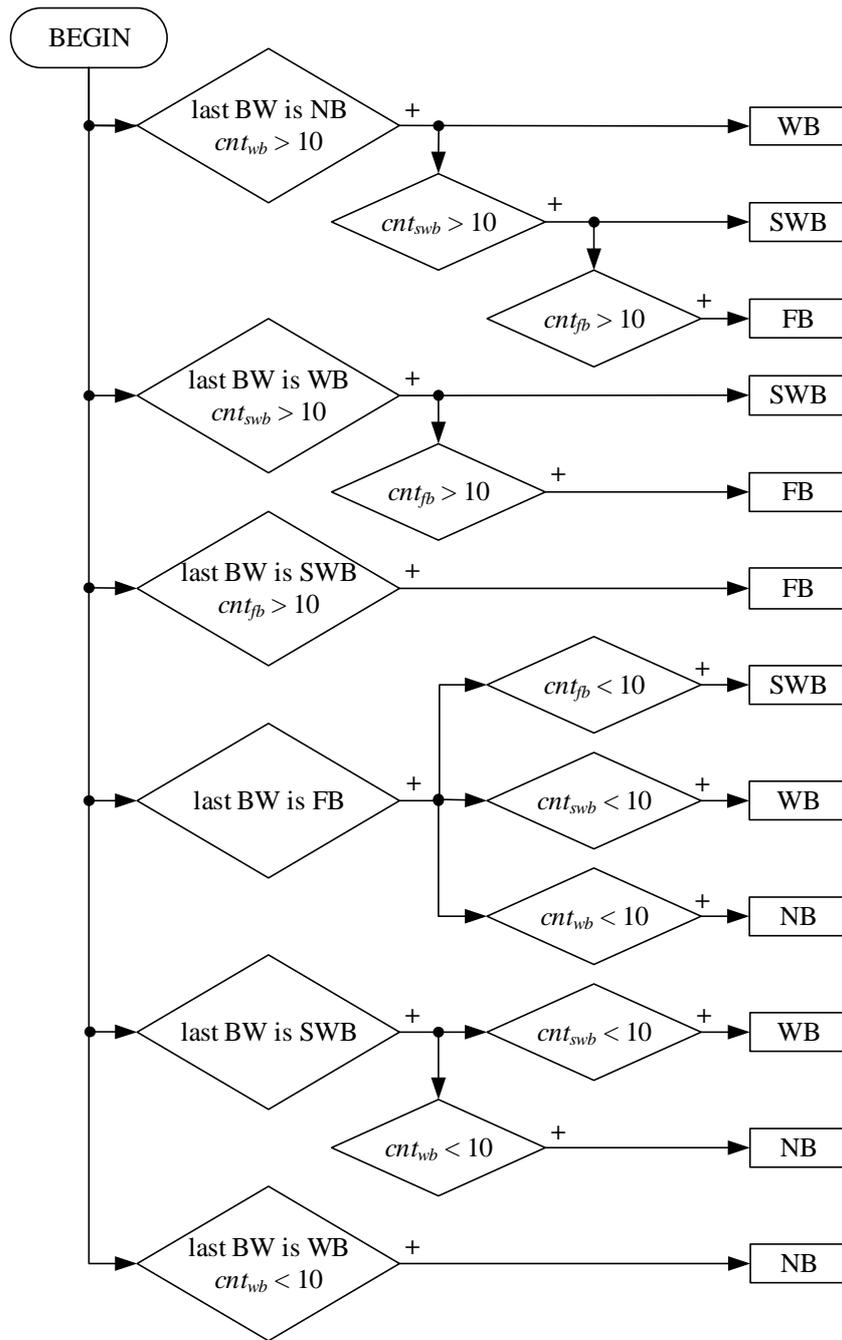
Figure 7: BW selection logic

In the above diagram, the tests are performed in a sequential order, i.e. it could happen that decision about signal bandwidth is changed several times in this logic. After every selection of a particular bandwidth, certain counters are reset to their minimal value 0 or to their maximum value 100.

Finally, the resulting bandwidth can be upper limited in case the codec performance has not been optimized for it at particular bitrate. For example, at 9.6 kbps, the codec supports coding up to SWB. Therefore, if the detected bandwidth is FB, it is overwritten to SWB at this bitrate. The following table shows the range of bitrates for which the codec performance has been optimized for each bandwidth.

Table 4: Optimization of the codec performance per bandwidth

bandwidth	bitrate range [kbps]
NB	7.2 - 24.4
WB	7.2 - 128
SWB	9.6 - 128
FB	16.4 - 128

5.1.8 Time-domain transient detection

The HP-filtered input signal $s_{HP}(n)$ including the look-ahead is input to the time-domain transient detector. The HP-filtered input signal $s_{HP}(n)$ is further high-pass filtered. The transfer function of the transient detection's HP filter is given by

$$H_{TD}(z) = 0.375 - 0.5z^{-1} + 0.125z^{-2} \qquad (35)$$

The signal, filtered by the transient detection's HP filter, is denoted as $s_{TD}(n)$. The HP-filtered signal $s_{TD}(n)$ is segmented into 8 consecutive segments of the same length. The energy of the HP-filtered signal $s_{TD}(n)$ for each segment is calculated as:

$$E_{TD}(i) = \sum_{n=0}^{L_{segment}-1} \left(s_{TD}\left(iL_{segment}+n\right)\right)^2, \qquad i = 0,...,7 \qquad (36)$$

where $L_{segment} = L_{inp}/8$ is the number of samples in 2.5 milliseconds segment at the input sampling frequency. An accumulated energy is calculated using:

$$E_{Acc} = \max(E_{TD}(i-1), 0.8125 E_{Acc}) \qquad (37)$$

A transient is detected if the energy of a segment $E_{TD}(i)$ exceeds the accumulated energy by a constant factor of 8.5 and the attack index is set to i. If no attack is detected but strong energy increase is detected in segment i, the attack index is set to i and the frame is not marked as a transient frame.

The energy change for each segment is calculated as:

$$E_{chng}(i) = \begin{cases} \dfrac{E_{TD}(i)}{E_{TD}(i-1)}, & E_{TD}(i) > E_{TD}(i-1) \\[2ex] \dfrac{E_{TD}(i-1)}{E_{TD}(i)}, & E_{TD}(i-1) > E_{TD}(i) \end{cases} \qquad (38)$$

The temporal flatness measure is calculated as:

$$TFM(N_{past}) = \frac{1}{8+N_{past}} \sum_{i=-N_{past}}^{7} E_{chng}(i) \qquad (39)$$

The maximum energy change is calculated as:

$$MEC(N_{past}, N_{new}) = \max\left(E_{chng}(-N_{past}), E_{chng}(-N_{past}+1),..., E_{chng}(N_{new}-1)\right) \qquad (40)$$

If index of $E_{chng}(i)$ or $E_{TD}(i)$ is negative then it indicates a value from the previous segment, with segment indexing relative to the current frame. N_{past} is the number of the segments from the past frames. It is equal to 0 if the temporal flatness measure is calculated for the usage in ACELP/TCX decision. If the temporal flatness measure is calculated for the TCX LTP decision then it is equal to:

$$N_{past} = 1 + \min\left(8, \left\lceil 8\frac{pitch}{L_{celp}} + 0.5 \right\rceil\right) \tag{41}$$

N_{new} is the number of segments from the current frame. It is equal to 8 for non-transient frames. For transient frames first the locations of the segments with the maximum and the minimum energy are found:

$$i_{\max} = \underset{i \in \{-N_{past},\ldots,7\}}{\arg\max} E_{TD}(i)$$
$$i_{\min} = \underset{i \in \{-N_{past},\ldots,7\}}{\arg\min} E_{TD}(i) \tag{42}$$

If $E_{TD}(i_{\min}) > 0.375 E_{TD}(i_{\max})$ then N_{new} is set to $i_{\max} - 3$, otherwise N_{new} is set to 8.

5.1.9 Linear prediction analysis

Short-term prediction or linear prediction (LP) analysis using the autocorrelation approach determines the coefficients of the synthesis filter of the CELP model. The autocorrelation of windowed speech is converted to the LP coefficients using the Levinson-Durbin algorithm. Then, the LP coefficients are transformed to the line spectral pairs (LSP) and consequently to line spectral frequencies (LSF) for quantization and interpolation purposes. The interpolated quantized and unquantized coefficients are converted back to the LP domain to construct the synthesis and weighting filters for each subframe.

5.1.9.1 LP analysis window

In case of encoding of an active signal frame, two sets of LP coefficients are estimated in each frame using a 25 ms asymmetric analysis window (320 samples at 12.8 kHz sampling rate), one for the frame-end and one for mid-frame LP analysis. A look ahead of 8.75ms (112 samples at 12.8 kHz sampling rate) is used for the frame-end autocorrelation calculation. The frame structure is shown below.
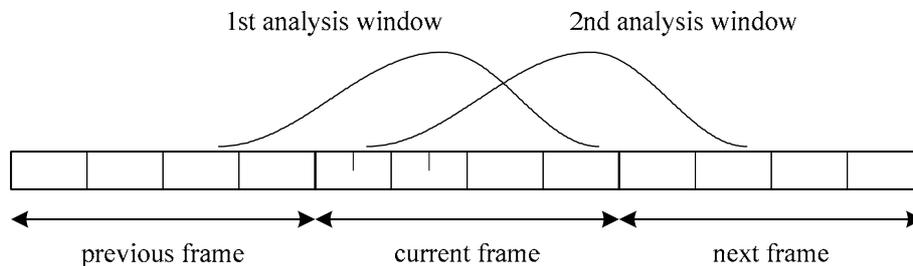
Figure 8: Relative positions and length of the LP analysis windows

The frame is divided into four sub-frames, each having a length of 5 ms, i.e., 64 samples. The windows for frame-end analysis and for mid-frame analysis are centred at the 2nd and 4th sub-frame of the current frame, respectively. An asymmetrical window with the length of 320 samples is used for windowing. The windowed signal for mid-frame is calculated as

$$s_{wmid}(n) = s_{pre}(n-80)w(n), \qquad n = 0,\ldots,319 \tag{43}$$

and the windowed signal for frame-end is calculated as

$$s_{wend}(n) = s_{pre}(n+48)w(n), \qquad n = 0,\ldots,319 \tag{44}$$

5.1.9.2 Autocorrelation computation

The autocorrelations of the windowed signal are computed by

$$r_c(k) = \sum_{n=k}^{L-1} s_{wend}(n)s_{wend}(n-k), \qquad k = 0,\ldots,16, \tag{45}$$

where L is set to 320. When $r_c(0) < 100$, $r_c(0)$ is set to 100 as well.

5.1.9.3 Adaptive lag windowing

In addition, bandwidth expansion is applied by lag windowing the autocorrelations using the following window

$$w_{lag}(i) = \exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right], \qquad i = 1,...,16, \tag{46}$$

where f_s is the sampling frequency (12800 or 16000) and the bandwidth frequency f_0 is set adaptively based on the OL pitch lag $d_{OL}^{[2]}$ in the 12.8 kHz domain and the normalized correlation $C_{norm}^{[2]}$ (i.e., pitch gain). These parameters are obtained in the OL pitch estimation module from the look-ahead part of the current or the previous frame, depending on whether the adaptive lag windowing is applied before or after the OL pitch estimation. In some special cases, $\min(d_{OL}^{[0]}, d_{OL}^{[1]})$ and $\max(C_{norm}^{[0]}, C_{norm}^{[1]})$ are used instead of $d_{OL}^{[2]}$ and $C_{norm}^{[2]}$, respectively. These situations will be described later in this specification. Note that the shorter pitch lag and/or the larger pitch gain, the stronger (heavy smoothing with larger f_0) window is used to avoid excessive resonance in the frequency domain. The longer pitch lag and/or the smaller normalized correlation, the weaker window (light smoothing with smaller f_0) is used to get more faithful representation of the spectral envelope.

Table 5: Selection of band width frequency f_0 in Hz

	$d_{OL}^{[2]} < 80$	$80 <= d_{OL}^{[2]} < 160$	$160 <= d_{OL}^{[2]}$
$0.6 < C_{norm}^{[2]}$	60	40	20
$0.3 < C_{norm}^{[2]} <= 0.6$	40	40	20
$C_{norm}^{[2]} <= 0.3$	40	20	20

The modified autocorrelation function, $r'(k)$ is calculated as

$$r'(k) = r_c(k) w_{lag}(k), \quad k = 0,...,16 \tag{47}$$

Further, $r'(0)$ is multiplied by the white noise correction factor 1.0001 which is equivalent to adding a noise floor of -40 dB.

5.1.9.4 Levinson-Durbin algorithm

The modified autocorrelation function, $r'(k)$, is used to obtain the LP filter coefficients a_k, $k = 1, ..., 16$ by solving the set of equations:

$$\sum_{k=1}^{16} a_k r'(|i-k|) = -r'(i), \qquad i = 1,...,16 \tag{48}$$

The set of equations in (48) is solved using the Levinson-Durbin algorithm. This algorithm uses the following recursion:

$$E(0) = r'(0)$$
$$\text{for } i = 1 \ldots 16$$
$$k_i = -\left[r'(i) + \sum_{j=1}^{i-1} a_j^{i-1} r'(i-j) \right] / E(i-1)$$
$$a_i^{(i)} = k_i \tag{49}$$
$$\text{for } j = 1 \ldots i-1 \qquad a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}$$
$$E(i) = \left(1 - k_i^2\right) E(i-1)$$
$$\text{end}$$

The final solution is given as $a_j = a_j^{(16)}$, $j = 1, \ldots 16$. The residual error energies (LP error energies) $E(i)$ are also used in the subsequent processing.

5.1.9.5   Conversion of LP coefficients to LSP parameters

The LP filter coefficients a_j are converted to the LSP representation [16] for quantization and interpolation purposes. For a 16th-order LP filter, the LSPs are defined as the roots of the sum and difference polynomials

$$F_1(z) = \frac{A(z) + z^{-16} A(z^{-1})}{1 + z^{-1}}$$
$$\tag{50}$$
$$F_2(z) = \frac{A(z) - z^{-16} A(z^{-1})}{1 - z^{-1}}$$

The polynomials $F_1(z)$ and $F_2(z)$ are symmetric and asymmetric, respectively. It can be proved that all roots of these polynomials lie on the unit circle and are interlaced. The polynomials $F_1(z)$ and $F_2(z)$ have each 8 conjugate roots, denoted $q_i = \cos(\omega_i)$ and called the Line Spectral Pairs (LSPs). The corresponding angular frequencies ω_i are the Line Spectral Frequencies (LSFs). The LSFs satisfy the ordering property $0 < \omega_0 < \ldots < \omega_{15} < \pi$. The coefficients of these polynomials are found by the following recursive relations:

$$f_1(0) = 1$$
$$f_2(0) = 1$$
$$\text{for } i = 1, \ldots, 8$$
$$f_1(i) = a_i + a_{16-i} - f_1(i-1) \tag{51}$$
$$f_2(i) = a_i - a_{16-i} + f_2(i-1)$$
$$\text{end}$$

where $M = 16$ is the predictor order.

The LSPs are found by evaluating the polynomials $F_1(z)$ and $F_2(z)$ at 100 points equally spaced between 0 and π and checking for sign changes. A sign change indicates the existence of a root and the sign change interval is then divided four times to track the root precisely. Considering the conjugate symmetry of the polynomials $F_1(z)$ and $F_2(z)$ and removing the linear term, it can be shown that the polynomials $F_1(z)$ and $F_2(z)$ can be written (considering $z = e^{j\omega}$) as

$$F_1'(\omega) = 2\cos 8\omega + 2 f_1(1) \cos 7\omega + \ldots + 2 f_1(7) \cos \omega + f_1(8)$$
$$F_2'(\omega) = 2\cos 8\omega + 2 f_2(1) \cos 7\omega + \ldots + 2 f_2(7) \cos \omega + f_2(8) \tag{52}$$

Considering the frequency mapping $x = \cos(\omega)$ we can define

$$T_m(\omega) = \cos(m\omega) \tag{53}$$

an mth-order Chebyshev polynomial in x [18]. The polynomials $F_1'(z)$ and $F_2'(z)$ can then be rewritten using this Chebyshev polynomial expansion as

$$F_1'(\omega) = 2T_8(x) + 2f_1(1)T_7(x) + \ldots + 2f_1(7)T_1(x) + f_1(8)$$
$$F_2'(\omega) = 2T_8(x) + 2f_2(1)T_7(x) + \ldots + 2f_2(7)T_1(x) + f_2(8)$$

(54)

Neglecting the factor of 2, which does not affect the root searching mechanism, the series to be evaluated can be generalized to

$$Y(x) = \sum_{k=0}^{7} f_k T_{8-k}(x)$$

(55)

The Chebyshev polynomials satisfy the order recursion

$$b_k(x) = 2xb_{k+1}(x) - b_{k+2}(x) + f_k$$

(56)

with initial conditions $b_8(x) = b_9(x) = 0$. This recursion can be used to calculate $b_0(x)$ and $b_2(x)$. Then, $Y(x)$ can be expressed in terms of $b_0(x)$ and $b_2(x)$

$$Y(x) = \sum_{k=0}^{7} \left[b_k(x) - 2xb_{k+1}(x) + b_{k+2}(x) \right] T_k(x) = \frac{b_0(x) - b_2(x) + f_0}{2}$$

(57)

The details about Chebyshev polynomial evaluation method can be found in [18].

In the following part of this document, the LSPs found by the described method will be denoted as q_i, $i=1,..,16$ with $q_0 = 1$.

5.1.9.6 LSP interpolation

The LP parameters for each subframe are obtained by means of interpolation between the end-frame parameters of the current frame, the mid-frame parameters of the current frame and the end-frame parameters of the previous frame. However, the LP parameters are not particularly suitable for interpolation due to stability issues. For this reason, the interpolation is done on the respective LSP parameters and then converted back to the LP domain.

Let $q_{end,i}$ denote the end-frame LSP vector of the current frame, $q_{mid,i}$ the mid-frame LSP vector of the current frame, both calculated by the method described in the previous section. Furthermore, let $q_{end-1,i}$ be the end-frame LSP vector of the previous frame. The interpolated LSP vectors for all subframes are then given by

$$q_i^{[0]} = 0.5q_{end-1,i} + 0.5q_{mid,i}$$
$$q_i^{[1]} = q_{mid,i}$$
$$q_i^{[2]} = 0.5q_{mid,i} + 0.5q_{end,i}$$
$$q_i^{[3]} = q_{end,i}$$

(58)

The same formula is used for interpolation of quantized LSPs described later in this document.

5.1.9.7 Conversion of LSP parameters to LP coefficients

Once the interpolated LSP vectors are calculated, they are converted back into LP filter coefficients for each subframe. Each LSP parameter $q_i = \cos\omega_i$ gives rise to a second order polynomial factor of the form $1 - 2\cos\omega_i z^{-1} + z^{-2}$. These can be multiplied together to form the polynomials, i.e.

$$F_1'(x) = \prod_{k=1}^{8} 2(x - x_{2k-1})$$

$$F_2'(x) = \prod_{k=1}^{8} 2(x - x_{2k})$$

(59)

By using the Chebyshev polynomial expansion defined in (55) we can apply the following recursion to find the coefficients of the polynomials:

$$
\begin{aligned}
&f_1'(0) = 1 \\
&f_1'(1) = -2q_0 \\
&\text{for } i = 2,..,8 \\
&\quad f_1'(i) = -2q_{2i-2} f_1'(i-1) + 2 f_1'(i-2) \\
&\quad \text{for } j = i-1,..,2 \quad f_1'(j) = f_1'(j) - 2q_{2i-2} f_1'(j-1) + f_1'(j-2) \\
&\quad f_1'(1) = f_1'(1) - 2q_{2i-2} \\
&\text{end}
\end{aligned}
$$

(60)

The coefficients $f_2'(i)$ are computed similarly, by replacing q_{2i-2} by q_{2i-1}, and with initial conditions $f_2'(0) = 1$ and $f_2'(1) = -2q_1$. Once the coefficients $f_1'(z)$ and $f_2'(z)$ are found, they are multiplied by $1+z^{-1}$ and $1-z^{-1}$, respectively, to form the polynomials $F_1(z)$ and $F_2(z)$. That is

$$
\begin{aligned}
f_1(i) &= f_1'(i) + f_1'(i-1), & i = 8,...,1 \\
f_2(i) &= f_2'(i) - f_2'(i-1), & i = 8,...,1
\end{aligned}
$$

(61)

Finally, the LP coefficients are found by

$$
\begin{aligned}
a_i &= 0.5 f_1(i) + 0.5 f_2(i), & i = 1,...,8 \\
a_i &= 0.5 f_1(i) - 0.5 f_2(i), & i = 8,...,16
\end{aligned}
$$

(62)

with $a_0 = 1$. This is directly derived from the equation $A(z) = (F_1'(z) + F_2'(z))/2$, and considering the fact that $F_1(z)$ and $F_2(z)$ are symmetric and asymmetric polynomials, respectively. The details of this procedure can be found in [18].

5.1.9.8   LP analysis at 16kHz

If ACELP core is selected for WB, SWB or FB signals at bitrates higher than 13.2 kbps, its internal sampling rate is set to 16 kHz rather than 12.8 kHz. In this case, the LP analysis is done at the end of the pre-processing chain on input signal resampled to 16 kHz and pre-emphasized (see subclauses 5.1.3.3 and 5.1.4). In this case, the length of the LP analysis window is 400 samples at 16 kHz, which corresponds again to 25 ms. The windowed signal for mid-frame is calculated as

$$s_{wmid}(n) = s_{pre}(n-100)w(n), n = 0,..., \quad 399$$

(63)

and the windowed signal for frame-end is calculated as

$$s_{wmid}(n) = s_{pre}(n+60)w(n), n = 0,..., \quad 399$$

(64)

The autocorrelation computation, adaptive lag windowing and the conversion of LP coefficients to LSP parameters are performed similarly as in subclauses 5.1.9.2 thru 5.1.9.5. However, the LSP interpolation is done on 5 sub-frames instead of 4 sub-frames. The interpolated LSP vectors are given by

$$q_i^{[0]} = 0.55 q_{end-1,i} + 0.45 q_{mid,i}$$

$$q_i^{[1]} = 0.15 q_{end-1,i} + 0.85 q_{mid,i}$$

$$q_i^{[2]} = 0.75 q_{mid,i} + 0.25 q_{end,i}$$

$$q_i^{[3]} = 0.35 q_{mid,i} + 0.65 q_{end,i}$$

$$q_i^{[4]} = q_{end,i}$$

The conversion of LSP parameters to LP coefficients is then performed similarly as in subclause 5.1.9.7. At the end of the LP analysis there are $M=16$ LSP parameters and a_i coefficients but the corresponding LSFs span the range of 0-8000 Hz rather than 0-6400 Hz.

The LP analysis at 25.6 kHz and 32 kHz is described later in this document.

5.1.10   Open-loop pitch analysis

The Open-Loop (OL) pitch analysis calculates three estimates of the pitch lag for each frame. This is done in order to smooth the pitch evolution contour and to simplify the pitch analysis by confining the closed-loop pitch search to a small number of lags around the open-loop estimated lags.

The OL pitch estimation is based on a perceptually weighted pre-emphasized input signal. The open-loop pitch analysis is performed on a signal decimated by two, i.e. sampled at 6.4 kHz. This is in order to reduce the computational complexity of the searching process. The decimated signal is obtained by filtering the signal through a 5th-order FIR filter with coefficients {0.13, 0.23, 0.28, 0.23, 0.13} and then down-sampling the output by 2.

The OL pitch analysis is performed three times per frame to find three estimates of the pitch lag: two in the current frame and one in the look-ahead area. The first two calculations are based on 10-ms segments of the current frame. The final (third) estimation corresponds to the look-ahead, and the length of this segment is 8.75ms.

5.1.10.1   Perceptual weighting

Perceptual weighting is performed by filtering the pre-emphasized input signal $s_{pre}(n)$ through a perceptual weighting filter, derived from the LP filter coefficients. The traditional perceptual weighting filter $W(z) = A(z/\gamma_1)/A(z/\gamma_2)$ has inherent limitations in modelling the formant structure and the required spectral tilt concurrently. The spectral tilt is pronounced in speech signals due to the wide dynamic range between low and high frequencies. This problem is eliminated by introducing the pre-emphasis filter (see subclause 5.1.4) at the input which enhances the high frequency content. The LP filter coefficients are found by means of LP analysis on the pre-emphasized signal. Subsequently, they are used to form the perceptual weighting filter. Its transfer function is the same as the LP filter transfer function but with the denominator having coefficients equal to the de-emphasis filter (inverse of the pre-emphasis filter). In this way, the weighting in formant regions is decoupled from the spectral tilt. Finally, the pre-emphasized signal is filtered through the perceptual filter to obtain a perceptually weighted signal, which is used further in the OL pitch analysis.

The perceptual weighting filter has the following form

$$W(z) = A(z/\gamma_1) H_{\text{de-emph}}(z) = \frac{A(z/\gamma_1)}{1 - \beta_{pre-emph} z^{-1}} \tag{65}$$

where

$$H_{\text{de-emph}}(z) = \frac{1}{1 - \beta_{pre-emph} z^{-1}} \tag{66}$$

and $\beta_{pre-emph} = 0.68$ and $\gamma_1 = 0.92$. Since $A(z)$ is computed based on the pre-emphasized signal $s_{pre}(n)$, the tilt of the filter $1/A(z/\gamma_1)$ is less pronounced compared to the case when $A(z)$ is computed based on the original signal. The de-emphasis is also performed on the output signal in the decoder. It can be shown that the quantization error spectrum is shaped by a filter having a transfer function $1/W(z)H_{de-emph}(z) = 1/A(z/\gamma_1)$. Thus, the spectrum of the quantization error is shaped by a filter whose transfer function is $1/A(z/\gamma_1)$, with $A(z)$ computed based on the pre-emphasized

signal. The perceptual weighting is performed on a frame-by-frame basis while the LP filter coefficients are calculated on a sub-frame basis using the principle of LSP interpolation, described in subclause 5.1.9.6. For a sub-frame size $L = 64$, the weighted speech is given by

$$s_h(n) = s_{pre}(n) + \sum_{i=1}^{16} a_i \, \gamma_1^i \, s_{pre}(n-i) + 0.68 \, s_h(n-1), \qquad n = 0, \dots, 255 \qquad (67)$$

where 0.68 is the pre-emphasis factor. Furthermore, for the open-loop pitch analysis, the computation is extended for a period of 8.75ms using the look-ahead from the future frame. This is done using the filter coefficients of the 4th subframe in the present frame. Note that this extended weighted signal is used only in the OL pitch analysis of the present frame.

If ACELP core is selected for WB, SWB or FB signals at bitrates higher than 13.2 kbps, its internal sampling rate is set to 16 kHz rather than 12.8 kHz. Nevertheless, the OL pitch analysis is done only at 12.8 kHz and the estimated OL pitch values are later resampled to 16 kHz. However, perceptually weighted input signal sampled at a16 kHz is still needed in the search of the adaptive codebook. The perceptual weighting filter at 16 kHz has the following form

$$W(z) = \frac{A(z/\gamma_2)}{1 - \beta_{pre-emph16} z^{-1}} \qquad (68)$$

where $\beta_{pre-emph16} = 0.72$ and $\gamma_2 = 0.94$. Thus, for this case, the pre-emphasis is done as follows

$$s_{h16}(n) = s_{pre16}(n) + \sum_{i=1}^{16} a_i \, \gamma_2^i \, s_{pre16}(n-i) + \beta_{pre-emph16} s_{h16}(n-1), \qquad n = 0, \dots, 319 \qquad (69)$$

The perceptual weighting at 25.6 kHz and 32 kHz is described later in this document.

5.1.10.2   Correlation function computation

The correlation function for each of the three segments (or half-frames) is obtained using correlation values computed over a first pitch delay range from 10 to 115 (which has been decimated by 2) and over a second pitch delay range from 12 to 115 (which has been decimated by 2). Both of the two delay ranges are divided into four sections: [10, 16], [17, 31], [32, 61] and [62, 115] for the first delay range and [12, 21], [22, 40], [41, 77] and [77, 115] for the second delay range, so that the two sets of four sections overlap. The first sections in the two sets, [10, 16] and [12, 21] are, however, used only under special circumstances to avoid quality degradation for pitch lags below the lowest pitch quantization limit. Due to this special use of the first sections in the sets, omitting the pitch lags 10 and 11 in the second set of sections presents no quality issues. In addition, the second set omits pitch lags between 17 and 20, when the first sections are not used. The first section is mainly used in speech segments with stable, short pitch lags and the above limits have therefore a negligible effect on the overall pitch search and quantization performance.

The autocorrelation function is first computed on a decimated weighted signal $s_{hd}(n)$ for each pitch lag value in both sets by

$$C(d) = \sum_{n=0}^{L_{sec}} s_{hd}(n) s_{hd}(n-d) \qquad (70)$$

where the summation limit L_{sec} depends on the delay section, i.e.:

$$
\begin{aligned}
L_{\text{sec}} &= 40 && for && d = 10,\ldots,16 && in\ set\ 1, \\
L_{\text{sec}} &= 40 && for && d = 17,\ldots,31 && in\ set\ 1, \\
L_{\text{sec}} &= 62 && for && d = 32,\ldots,61 && in\ set\ 1, \\
L_{\text{sec}} &= 115 && for && d = 62,\ldots,115 && in\ set\ 1, \\
L_{\text{sec}} &= 40 && for && d = 12,\ldots,21 && in\ set\ 2, \\
L_{\text{sec}} &= 50 && for && d = 22,\ldots,40 && in\ set\ 2, \\
L_{\text{sec}} &= 80 && for && d = 41,\ldots,77 && in\ set\ 2, \\
L_{\text{sec}} &= 115 && for && d = 78,\ldots,115 && in\ set\ 2
\end{aligned}
\tag{71}
$$

This will ensure that, for a given delay value, at least one pitch cycle is included in the correlation computation. The autocorrelation window is aligned with the first sample of each of the two 10-ms segments of the current frame, where the autocorrelation can thus be calculated directly according to equation (70). To maximize the usage of the look-ahead segment, the autocorrelation window in the third segment is aligned with the last available sample. In this final segment the autocorrelation function of equation (70) is computed backwards, i.e., the values of n are negative. Therefore, the computation as such is the same for all three segments, only the window alignment differs and the indexing of the signal is reversed in the last segment.

5.1.10.3 Correlation reinforcement with past pitch values

The autocorrelation function is then weighted for both pitch delay ranges to emphasize the function for delays in the neighbourhood of pitch lag parameters determined in the previous frame (extrapolated pitch lags).

The weighting function is given by a triangular window of size 27 and it is centred on the extrapolated pitch lags. The weighting function is given by

$$
w_{pn}(13+i) = w_{pn}(13-i) = 1 + \alpha_{pn}(1 - i/14), \quad i = 0,\ldots,13. \tag{72}
$$

where α_{pn} is a scaling factor based on the voicing measure from the previous frame (the normalized pitch correlation) and the pitch stability in the previous frame. During voiced segments with smooth pitch evolution, the scaling factor is updated in each frame by adding a value of $0.16\overline{R}_{xy}$ and it is upper-limited to 0.7. $\overline{R}_{xy}$ is the average of the normalized correlation in the two half frames of the previous frame and is given in equation (78). The scaling factor α_{pn} is reset to zero (no weighting) if $\overline{R}_{xy}$ is less than 0.4 or if the pitch lag evolution in the previous frame is unstable or if the relative frame energy of the previous frame is more than a certain threshold. The pitch instability is determined by testing the pitch coherence between consecutive half-frames. The pitch values of two consecutive half-frames are considered coherent if the following condition is satisfied:

$$
(max\_value < 1.4\ min\_value)\ \text{AND}\ ((max\_value - min\_value) < 14)
$$

where max_value and min_value denote the maximum and minimum of the two pitch values, respectively. The pitch evolution in the current frame is considered stable if pitch coherence is satisfied for both, the first half-frame of the current frame and the second half-frame of the previous frame as well as the first half-frame and the second half-frame of the current frame.

The extrapolated pitch lag in the first half-frame, $\tilde{d}^{[0]}$, is computed as the pitch lag from the second half-frame of the previous frame plus a pitch evolution factor f_{evol}, computed from the previous frame (described in subclause 5.1.10.7). The extrapolated pitch lag in the second half-frame, $\tilde{d}^{[1]}$, is computed as the pitch lag from the second half-frame of the previous frame plus twice the pitch evolution factor. The extrapolated pitch lag in the look ahead, $\tilde{d}^{[2]}$, is set equal to $\tilde{d}^{[1]}$. That is

$$
\begin{aligned}
\tilde{d}^{[0]} &= d^{[-1]} + f_{evol}, \\
\tilde{d}^{[1]} &= d^{[-1]} + 2f_{evol}, \\
\tilde{d}^{[2]} &= \tilde{d}^{[1]},
\end{aligned}
\tag{73}
$$

where $d^{[-1]}$ is the pitch lag in the second half-frame of the previous frame. The pitch evolution factor is obtained by averaging the pitch differences of consecutive half-frames that are determined as coherent (according to the coherence rule described above).

The autocorrelation function weighted around an extrapolated pitch lag d is given by

$$C_w(d+i) = C(d+i)w_{pn}(13+i), \quad i = -13, \ldots, 13.$$ (74)

5.1.10.4 Normalized correlation computation

After weighting the correlation function with the triangular window of equation (72) centred at the extrapolated pitch lag, the maxima of the weighted correlation function in each of the four sections (three sections, if the first section is not used) are determined. This is performed for both pitch delay ranges. Note that the first section is used only during high-pitched segments. For signals other than narrowband signals, this means that the open-loop pitch period of the second half-frame of the previous frame is lower than or equal to 34. For narrowband signals, the open-loop pitch period of the second half-frame of the previous frame needs to be lower than or equal to 24 and the scaling factor α_{pn} has to be higher than or equal to 0.1. It is further noted that the scaling factor α_{pn} is set to 0, if the previous frame were an unvoiced or a transition frame and the signal has a bandwidth higher than narrowband. In the following, the special case of three sections will not be explicitly dealt with if it arises directly from the text. The pitch delays that yield the maximum of the weighted correlation function will be denoted as $d_{\max}(k)$, where $k = 0,1,2,3$ denotes each of the four sections. Then, the original (raw) correlation function at these pitch delays (pitch lags) is normalized as

$$C_{norm}(d_{\max}(k)) = \frac{C(d_{\max}(k))}{\sqrt{\sum_{n=0}^{L_{sec}} s_h^2(n) \sum_{n=0}^{L_{sec}} s_h^2(n - d_{\max}(k))}}, \qquad k = 0,1,2,3$$ (75)

The same normalization is applied also to the weighted correlation function, $C_w(d)$, which yields $C_{wn}(d)$. It is noted that $s_h(n)$ is aligned at the first sample of the corresponding half-frame for the two half-frames of the current frame and at the last sample of the look-ahead for the look-ahead segment, where the calculation is performed backwards in order to exploit the full look-ahead as well as possible.

At this point, four candidate pitch lags, $d_{\max}(k)$, k = 0,1,2,3, have been determined for each of the three segments (two in the current frame and one in the look-ahead) in each of the two pitch delay ranges. In correspondence with these candidate pitch lags, normalized correlations (both weighted and raw) have been calculated. All remaining processing is performed using only these selected values, greatly reducing the overall complexity.

5.1.10.5 Correlation reinforcement with pitch lag multiples

In order to avoid selecting pitch multiples within each pitch delay range, the weighted normalized correlation in a lower pitch delay section is further emphasized if one of its multiples is in the neighbourhood of the pitch lag in a higher section. That is,

$$\begin{aligned} &if \left(\left| k \times d_{\max}(2) - d_{\max}(3) \right| \leq k \right) \\ &\quad if \left(\alpha_{prev} < 0.6 \vee d_{\max}(2) > 0.4 d_{prev} \right) \\ &\qquad C_{wn}(d_{\max}(2)) = \alpha_{mult} C_{wn}(d_{\max}(2)) \\ &\quad \alpha_{mult} = (\alpha_{mult})^2 \\ &if \left(\left| k \times d_{\max}(1) - d_{\max}(2) \right| \leq k \right) \\ &\quad if \left(\alpha_{prev} < 0.6 \vee d_{\max}(1) > 0.4 d_{prev} \right) \\ &\qquad C_{wn}(d_{\max}(1)) = \alpha_{mult} C_{wn}(d_{\max}(1)) \\ &\quad \alpha_{mult} = (\alpha_{mult})^2 \end{aligned}$$

where $\alpha_{mult} = 1.17$, α_{prev} is a voicing factor (normalized pitch correlation) from the previous frame, and d_{prev} is the pitch value from the second half-frame of the previous frame. In addition, when the first section is searched and the pitch multiple of the shortest-section candidate lag is larger than 20 samples, the following reinforcement is performed:

$$if \left(\left| k \times d_{\max}(0) - d_{\max}(1) \right| \le k \right)$$
$$if \left(\alpha_{prev} < 0.6 \vee d_{\max}(0) > 0.4 d_{prev} \right)$$
$$C_{wn}\left(d_{\max}(0) \right) = \alpha_{mult} C_{wn}\left(d_{\max}(0) \right)$$

Further, α_{prev}, is given by the voicing factor of the second half-frame in the previous frame if the normalized correlation in the second half-frame was stronger than in the first half-frame, or otherwise by the mean value of these two normalized correlations. In this way, if a pitch period multiple is found in a higher section, the maximum weighted correlation in the lower section is emphasized by a factor of 1.17. However, if the pitch lag in section 3 is a multiple of the pitch lag in section 2 and at the same time the pitch lag in section 2 is a multiple of the pitch lag in section 1, the maximum weighted correlation in section 1 is emphasized twice. This correlation reinforcement is, however, not applied at each section when the previous frame voicing factor, α_{prev}, is less than 0.6 and the pitch value is less than 0.4 times the previous pitch value (i.e., the pitch value does not appear to be a halved value of the previous frame pitch or larger). In this way, the emphasis of the correlation value is allowed only during clear voicing conditions or when the value can be considered to belong to the past pitch contour.

The correlation reinforcement with pitch lag multiples is independent in each of the two pitch delay ranges.

It can be seen that the "neighbourhood" is larger when the multiplication factor k is higher. This is to take into account an increasing uncertainty of the pitch period (the pitch length is estimated roughly with integer precision at a 6400 Hz sampling frequency). For the look-ahead part, the first line of the condition above relating to the highest pitch lags is modified as follows:

$$if \left(\left| k \times d_{\max}(1) - d_{\max}(2) \right| \le 2(k-1) \right)$$

Note that first section is not considered in the correlation reinforcement procedure described here, i.e., the maximum normalized correlation in the first section is never emphasized.

5.1.10.6 Initial pitch lag determination and reinforcement based on pitch coherence with other half-frames

An initial set of pitch lags is determined by searching for the maximum weighted normalized correlation in the four sections in each of the three segments or half-frames. This is done independently for both pitch delay ranges. The initial set of pitch lags is given by

$$d_{init}^{[k]} = \underset{i=0}{\overset{3}{\arg\max}} \left(C_{wn}\left(d_{\max}^{[k]}(i) \right) \right), \qquad for \, k = 0,1,2, \tag{76}$$

where the superscript $[k]$ denotes the first, the second or the third (look-ahead) half-frame.

To find the right pitch value, another level of weighting is performed on the weighted normalized correlation function, $C_{wn}(d)$, in each section of each half-frame in each pitch delay range. This weighting is based on pitch coherence of the initial set of pitch lags, $d_{init}^{[i]}$, with pitch lags $d_{init}^{[j]}(k)$, $k = 0,1,2,3$, $j \ne i$, i.e., those from the other half-frames. The weighting is further reinforced with pitch lags selected from the complementary pitch delay range, denoted as $z_{init}^{[j]}(k)$, $k = 0,1,2,3$, $j \ne i$. Further, the weighting favours section-wise stability, where a stronger weighting is applied for coherent pitch values that are from the same section of the same set as the initial pitch lag, and a slightly weaker weighting is applied for coherent pitch values that are from a different section and/or a different pitch delay range than the initial pitch lag. That is, if the initial pitch lag in a half-frame i is coherent with a pitch lag of section k in half-frame j, then the corresponding weighted normalized correlation of section k in half-frame j is further emphasized by weighting it by the value $1 + \alpha_1 \left(1 - \delta_{pit}/14 \right)$, if the initial pitch lag is also from section k in the same pitch delay range, or by $1 + \alpha_2 \left(1 - \delta_{pit}/14 \right)$, if the initial pitch lag is not from section k in the same pitch delay range. The variable δ_{pit} is the absolute difference between the two analysed pitch lags and the two weighting factors are defined as

$$\alpha_1 = 0.4\left(C_{norm}\left(d_{init}^{[i]}\right) + 0.5r_e\right),$$
$$\alpha_2 = 0.25\left(C_{norm}\left(d_{init}^{[i]}\right) + 0.5r_e\right),$$

where α_1 is upper-bounded by 0.4, α_2 is upper-bounded by 0.25 and $C_n(d)$ is the raw normalized correlation (similar to the weighted normalized correlation, defined in equation (75)). Finally, r_e is a noise correction factor added to the normalized correlation in order to compensate for its decrease in the presence of the background noise. It is defined as

$$r_e = 0.0002.4492e^{0.1596N_t} - 0.022 \qquad \text{constrained by } 0 < r_e < 0.5 \tag{77}$$

where N_t is the total background noise energy, calculated as described in subclause 5.1.11.1.

The procedure described in this subclause helps further to avoid selecting pitch multiples and insure pitch continuity in adjacent half-frames.

5.1.10.7 Pitch lag determination and parameter update

Finally, the pitch lags in each half-frame, $d^{[0]}$, $d^{[1]}$ and $d^{[2]}$, are determined. They are selected by searching the maximum of the weighted normalized correlations, corresponding to each of the four sections across both pitch delay ranges. In case of VBR operation, the normalized correlations are searched in addition to the weighted normalized correlations for a secondary evaluation. When the normalized correlation of the candidate lag is very high (lower-bounded by 0.9) and it is considered a halved value (lower-bounded by a multiplication by 0.4 and upper-bounded by a multiplication by 0.6) of the corresponding candidate identified by searching the weighted normalized correlation, the secondary pitch lag candidate is selected instead of the firstly selected one.

In total, six or eight values are thus considered in each segment or half-frame depending on whether section 0 is searched. After determining the pitch lags, the parameters needed for the next frame pitch search are updated. The average normalized correlation $\overline{R}_{xy}$ is updated by:

$$\overline{R}_{xy} = 0.5\left(C_{norm}\left(d^{[0]}\right) + C_{norm}\left(d^{[1]}\right)\right) + 0.5r_e, \quad \text{constrained by } \overline{R}_{xy} \leq 1 \tag{78}$$

Finally, the pitch evolution factor f_{evol} to be used in computing the extrapolated pitch lags in the next frame is updated. The pitch evolution factor is given by averaging the pitch differences of the consecutive half-frames that are determined as coherent. If $d^{[-1]}$ is the pitch lag in the second half of the previous frame, then pitch evolution is given by

$$
\begin{aligned}
&\delta_{pitch} = 0 \\
&cnt = 0 \\
&\text{for } i = 0 \ldots 2 \\
&\quad \text{if } d^{[i]} \text{ AND } d^{[i-1]} \text{ are coherent} \\
&\quad\quad \delta_{pitch} = \delta_{pitch} + d^{[i]} - d^{[i-1]} \\
&\quad\quad cnt = cnt + 1 \\
&\quad \text{if } cnt > 0 \\
&\quad\quad f_{evol} = d_{pitch}/cnt \\
&\quad \text{else} \\
&\quad\quad f_{evol} = 0
\end{aligned}
\tag{79}
$$

Since the search is performed on the decimated weighted signal, the determined pitch lags, $d^{[0]}$, $d^{[1]}$ and $d^{[2]}$ are multiplied by 2 to obtain the open-loop pitch lags for the three half-frames. That is

$$
\begin{aligned}
T_{OL}^{[0]} &= 2d^{[0]}, \\
T_{OL}^{[1]} &= 2d^{[1]}, \\
T_{OL}^{[2]} &= 2d^{[2]}.
\end{aligned}
\tag{80}
$$

In the following text, the following notation is used for the normalized correlations corresponding to the final pitch lags:

$$
C_{norm}^{[i]} = C_{norm}^{[i]}\left(d^{[i]}\right)
\tag{81}
$$

5.1.10.8 Correction of very short and stable open-loop pitch estimates

Usually, music harmonic signals or singing voice signals have short pitch lags and they are more stationary than normal speech signals. It is extremely important to have the correct and precise short pitch lags as incorrect pitch lags may have a serious impact upon the quality.

The very short pitch range is defined from $PIT_{MIN\_DOUBLEEXTEND} = 17$ to $PIT_{MIN} = 34$ at the sampling frequency $Fs = 12.8$ kHz. As the pitch candidate is so short, pitch detection of using time domain only or frequency domain only solution may not be reliable. In order to reliably detect short pitch value, three conditions may need to be checked: (1) in frequency domain, the energy from 0 Hz to $F_{MIN} = Fs / PIT_{MIN}$ Hz must be relatively low enough; (2) in time domain, the maximum short pitch correlation in the pitch range from $PIT_{MIN\_DOUBLEEXTEND}$ to PIT_{MIN} must be relatively high enough compared to the maximum pitch correlation in the pitch range from PIT_{MIN} to PIT_{MAX}; (3) the absolute value of the maximum normalized short pitch correlation must be high enough. These three conditions are more important; other conditions may be added such as Voice Activity Detection and Voiced Classification.

Suppose $Voicing_m$ notes the average normalized pitch correlation value of the four subframes in the current frame:

$$
Voicing_m = [\ C_{norm}^{[0]} + C_{norm}^{[1]} + C_{norm}^{[2]} + C_{norm}^{[3]}\ ] / 4
\tag{82}
$$

$C_{norm}^{[0]}$, $C_{norm}^{[1]}$, $C_{norm}^{[2]}$, $C_{norm}^{[3]}$ are the four normalized pitch correlations calculated for each subframe; for each subframe, the best pitch candidate is found in the pitch range from $P = PIT_{MIN}$ to $P = PIT_{MAX}$. The smoothed pitch correlation from previous frame to current frame is

$$
Voicing_{sm} = (3 \cdot Voicing_{sm} + Voicing)/4
\tag{83}
$$

Before the real short pitch is decided, two pre-decision conditions are checked first : (a) check if the harmonic peak is sharp enough, which is indicated by the flag $pre\_decision\_flag = harmonic\_sharp\_flag$. It is used to decide if the initial open-loop pitch is correct or not; (b) check if the maximum energy in the frequency region [0, F_{MIN}] is low enough, which is indicated by the flag $pre\_decision\_flag = LF\_lack\_flag$.

  (a) Determine base pitch frequency d_{freq} according to the initial open-loop pitch $d^{[1]}$

$$
d_{freq} = Round(L_{FFT} / d^{[1]})
\tag{84}
$$

  Then, based on the amplitude spectrum of input signal in frequency domain, determine the decision parameters which are used to confirm whether the pitch related to the base pitch frequency is accurate. The decision parameters include energy spectrum difference, average energy spectrum and the ratio of energy spectrum difference and average energy spectrum.

  Compute the energy spectrum difference and the average energy spectrum of the frequency bins around base pitch frequency d_{freq}

$$
E_{diff} = \sum_{k=1}^{2d_{freq}-1} \left(E_{BIN}\left(d_{freq}\right) - E_{BIN}(k)\right)
\tag{85}
$$

$$E_{avrg} = \sum_{k=1}^{2d_{freq}-1} E_{BIN}(k) \Big/ (2d_{freq}-1) \tag{86}$$

Compute the weighted and smoothed energy spectrum difference and average energy spectrum

$$E_{diff\_sm} = 0.2\,E_{diff} + 0.8\,E_{diff\_sm,prev} \tag{87}$$

$$E_{avrg\_sm} = 0.2\,E_{avrg} + 0.8\,E_{avrg\_sm,prev} \tag{88}$$

where E_{diff_sm} and E_{avrg_sm} are weighted and smoothed energy spectrum difference and average energy spectrum of the frequency bins around the base pitch frequency.

Compute the ratio of energy spectrum difference and average energy spectrum

$$R_{diff} = E_{diff} \Big/ E_{avrg} \tag{89}$$

Based on the decision parameters calculated above, confirm whether the initial open-loop pitch is accurate.

The harmonic sharpness flag is determined as follows:

$$
\begin{aligned}
&if\ \left(E_{diff\_sm} < -10\ AND\ E_{avrg\_sm} < 38.5\ AND\ R_{diff} < -0.8\right)\\
&\quad pre\_decision\_flag = harmonic\_sharp\_flag = 1\\
&if\ \left(E_{diff\_sm} > 10\ AND\ E_{avrg\_sm} > 83\ AND\ R_{diff} > 0.5\right)\\
&\quad pre\_decision\_flag = harmonic\_sharp\_flag = 0
\end{aligned}
\tag{90}
$$

If the above conditions are not satisfied, $pre\_decision\_flag$ remains unchanged.

(b) Assume that the maximum energy in the frequency region $[0, F_{MIN}]$ (Hz) is $Energy0$ (dB), the maximum energy in the frequency region $[F_{MIN}, 900]$ (Hz) is $Energy1$ (dB), the relative energy ratio between $Energy0$ and $Energy1$ is given by

$$Ratio = Energy1 - Energy0 \tag{91}$$

This energy ratio is weighted by multiplying an average normalized pitch correlation value $Voicing_m$,

$$Ratio = Ratio \cdot Voicing_m \tag{92}$$

Before using the $Ratio$ parameter to detect the lack of low frequency energy, it is smoothed in order to reduce the uncertainty,

$$EnergyRatio_{LF\_sm} = (15 \cdot EnergyRatio_{LF\_sm} + Ratio)/16 \tag{93}$$

where the $EnergyRatio_{LF\_sm}$ is the low frequency smoothed energy ratio. If $LF\_lack\_flag = 1$ then a lack of low frequency energy has been detected (otherwise not detected). LF_lack_flag is determined by the following procedure,

$$
\begin{aligned}
&if\ (EnergyRatio_{LF\_sm} > 35\ or\ Ratio > 50)\\
&\quad pre\_decision\_flag = LF\_lack\_flag = 1\ ;\\
&if\ (EnergyRatio_{LF\_sm} < 16)\\
&\quad pre\_decision\_flag = LF\_lack\_flag = 0\ ;
\end{aligned}
\tag{94}
$$

If the above conditions are not satisfied, $pre\_decision\_flag$ remains unchanged.

An initial very short pitch candidate T_p is found by searching a maximum normalized pitch correlation from $P = PIT_{MIN\_DOUBLEEXTEND}$ to PIT_{MIN},

$$R(T_p) = \max\{R(P), P = PIT_{MIN\_DOUBLEEXTEND}, ..., PIT_{MIN}\} \tag{95}$$

If $Voicing0$ notes the current short pitch correlation,

$$Voicing0 = R(T_p) \tag{96}$$

The smoothed short pitch correlation from previous frame to current frame is

$$Voicing0_{sm} = (3 \cdot Voicing0_{sm} + Voicing0)/4 \tag{97}$$

By using all the available parameters, the final very short pitch lag is decided with the following procedure,

$$
\begin{aligned}
&if\ ((pre\_decision\_flag = 1)\ AND\ (VAD = 1)\ AND\ (Voicing0_{sm} > 0.65)\ AND\ (Voicing0_{sm} > 0.7 \cdot Voicing_{sm})) \\
&\quad Open\_Loop\_Pitch = T_{OL}^{[0]} = T_{OL}^{[1]} = T_p \\
&\quad f_{spitch} = Stab\_short\_pitch\_flag = flag\_spitch = 1 \\
&\quad Coder\_type = VOICED
\end{aligned}
\tag{98}
$$

wherein $f_{spitch} = Stab\_short\_pitch\_flag = flag\_spitch = 1$ is a flag which forces the codec to select the time domain CELP coding algorithm for short pitch signal even if the frequency domain coding algorithm and AUDIO class is previously selected; $Coder\_type = VOICED$ is a flag which forces the coder to select VOICED class for short pitch signal.

5.1.10.9 Fractional open-loop pitch estimate for each subframe

The OL pitch is further refined by maximizing the normalized correlation function with a fractional resolution around the pitch lag values $d^{[0]}$ and $d^{[1]}$ (in the 12.8-kHz sampling domain). The fractional open-loop pitch lag is computed four times per frame, i.e., for each subframe of 64 samples. This is similar as the closed-loop pitch search, described in later in this specification. The maximum normalized correlation corresponding to the best fractional open-loop pitch lag is then used in the classification of VC frames (see subclause 5.1.13.2). The fractional open-loop pitch search is performed by first maximizing an autocorrelation function C of the perceptually weighted speech s_h for integer lags in the interval [$d^{[i]} - 7 ... d^{[i]} + 6$], where $d^{[i]} = d^{[0]}$ for the search in the first and the second subframes, and $d^{[i]} = d^{[1]}$ for the third and fourth subframes. The autocorrelation function is similar to equation (70) except that perceptually weighted speech at 12.8 kHz sampling rate is used, i.e,

$$C_{raw}(d) = \sum_{n=0}^{63} s_h(n) s_h(n-d) \tag{99}$$

In the above equation, $s_h(0)$ corresponds to the first sample in each subframe.

Let d_{int} be the integer lag maximizing $C_{raw}(d)$. The fractional open-loop pitch search is then performed by interpolating the correlation function C_{raw} and searching for its maximum in the interval $[d_{int} - 3/4 ... d_{int} + 3/4]$. The interpolation is performed with a 1/4 sample resolution using an FIR filter – a Hamming windowed sinc function truncated at ± 17. The filter has its cut-off frequency (–3 dB) at 5062 Hz and –6 dB at 5760 Hz in the 12.8 kHz domain. This means the interpolation filter exhibits a low-pass frequency response. Note that the negative fractions are not searched if d_{int} coincides with the lower end of the searched interval, i.e., if $d_{int} = d^{[i]} - 7$.

Once the best fractional pitch lag, d_{fr}, is found, the maximum normalized correlation is computed similarly to equation (75), i.e.,

$$C_{fr}(d_{fr}) = \frac{C_{raw}(d_{fr})}{\sqrt{\sum_{n=0}^{63} s_h^2(n) \sum_{n=0}^{63} s_h^2(n - d_{fr})}} \tag{100}$$

The same normalization is applied also to the weighted correlation function, $C_w(d)$, which yields $C_{wn}(d)$.

At this point, four candidate pitch lags, $d_{\max}(k)$, $k = 0,1,2,3$, have been determined for each of the three half-frames (two in the current frame and one in the look ahead) in each of the two pitch delay ranges. In correspondence with these candidate pitch lags, normalized correlations (both weighted and raw) have been calculated. All remaining processing is performed using only these selected values, greatly reducing the overall complexity.

Note that the last section (long pitch periods) in both pitch delay ranges is not searched for the look ahead part. Instead, the normalized correlation values and the corresponding pitch lags are obtained from the last section search of the second half-frame. The reason is that the summation limit in the last section is much larger than the available look ahead and also the computational complexity is reduced.

The fractional OL pitch estimation as described above is performed only for bitrates lower or equal to 24.4 kbps. For higher bitrates, the VC mode is not supported and consequently, there is no reason to estimate pitch with fractional resolution. Therefore, at higher bitrates, $d_{fr} = d^{[i]}$ where for the first and for the second subframe i=0 and for the third and the fourth subframe i=1.

5.1.11 Background noise energy estimation

The background noise energy is estimated (updated) in two stages. In the first stage, noise energy is updated only for critical bands where the current frame signal energy is less than the previously estimated background noise energy. This stage is called the downward noise energy update. In the second stage, noise energy is updated if the signal characteristics are statistically close to the model of background noise. Therefore, in the second stage, noise energy can be updated regardless of the current frame signal energy.

5.1.11.1 First stage of noise energy update

The total noise energy per frame is computed as follows:

$$N_t = 10 \log \left(\sum_{i=0}^{19} N_{CB}^{[-1]}(i) \right) \tag{101}$$

where $N_{CB}^{[-1]}(i)$ is the estimated noise energy in the ith critical band of the previous frame.

The noise energy per critical band $N_{CB}(i)$ is initialized to 0.0035 dB. The updated noise energy in the ith critical band, denoted $N_{tmp}(i)$, is computed as follows:

$$N_{tmp}(i) = 0.9 N_{CB}^{[-1]}(i) + 0.1 \left[0.25 E_{CB}^{[-1]}(i) + 0.75 \left(E_{CB}^{[0]}(i) + 0.5 E_{CB}^{[1]}(i) \right) \right] \tag{102}$$

where $E_{CB}^{[-1]}(i)$ corresponds to the energy per critical band calculated in the second spectral analysis in the previous frame, and $N_{CB}^{[-1]}(i)$ is the estimated noise energy per critical band also in the previous frame. Noise energy is then updated only in critical bands that have lower energy than the background noise energy. That is

$$N_{CB}^{[0]}(i) = N_{tmp}(i), \qquad i = 0,..,19 \text{ AND } N_{tmp}(i) < N_{CB}^{[-1]}(i) \tag{103}$$

The superscript [0] in the above equation is used to stress that it corresponds to the current frame.

Another feature used in noise estimation and SAD is an estimate of the frame to frame energy variation. The absolute energy difference between the current and the last frame is calculated,.

$$E_{tv} = \left| E_t^{[-1]} - E_t \right|. \tag{104}$$

where the superscript [-1] has been used to denote the previous frame. The frame energy variation is then used to update the feature

$$\overline{E}_{vh2} = \max\left(0.1,\, 0.98\overline{E}_{vh2}^{[-1]} + 0.02\min(3.0,\, E_{tv})\right) \tag{105}$$

Other energy features that are updated before the SAD and the second stage of the noise estimation are first initialized during the very two frames after encoder initialization. The initialization is done as follows

$$
\begin{aligned}
E_{th} &= E_t \\
E_{tl} &= E_t \\
\overline{E}_{tl} &= E_t \\
\overline{E}_t^{[-1]} &= E_t \\
\overline{E}_{vh2}^{[-1]} &= 0.0 \\
\overline{E}_{dyn}^{[-1]} &= 0.0
\end{aligned}
\tag{106}
$$

After the two frames of initialization the total frame energy is smoothed by means of LP filtering. That is:

$$\overline{E}_t = 0.80\overline{E}_t^{[-1]} + 0.20E_t \tag{107}$$

The features E_{tl} and E_{th} are envelope tracking features of the frame energy E_t and are used to create the long-term minimum energy $\overline{E}_{tl}$ and an estimate of the energy dynamics $\overline{E}_{dyn}$. That is

$$E_{th} = \min(E_{th}^{[-1]} - 0.04, E_t) \tag{108}$$

To calculate E_{tl} the following processing is applied:

$$
\begin{aligned}
&E_{tl} = E_{tl}^{[-1]} + 0.08 \\
&\text{if } c_{harm}^{[-1]} > 50 \text{ then} \\
&\quad \text{if } ini_{frame} < 150 \quad \text{AND} \quad (E_t^{[-1]} - \overline{E}_t) < 3.0 \qquad E_{tl} = E_{tl} + \min(2, 0.1(E_t^{[-1]} - E_{tl})) \\
&\quad \text{if } (E_t^{[-1]} - E_{tl}) > 30 \quad \text{AND} \quad c_{harm}^{[-1]} < 20 \qquad E_{tl} = E_{tl} + 0.2(E_t^{[-1]} - E_{tl}) \\
&\quad \text{else if } (E_t^{[-1]} - E_{tl}) > 10 \qquad\qquad\qquad\qquad E_{tl} = E_{tl} + 0.08 \\
&E_{tl} = \min(E_{tl}, E_t)
\end{aligned}
\tag{109}
$$

where $c_{harm}^{[-1]}$ is the number of frames since the last harmonic event from the previous frame. See clause 5.1.11.3.2 for details about its computation. The new value of E_{tl} is then used to update its long-term value through an AR process. That is

$$\overline{E}_{tl} = (1 - \alpha_{tl})\overline{E}_{tl}^{[-1]} + \alpha_{tl}E_{tl} \tag{110}$$

where the parameter α_{tl} is set as follows

$$\text{if } \left(c_{harm}^{[-1]} > 30 \text{ AND } E_t^{[-1]} - E_{tl} > 30\right) \text{ OR } \left(c_{harm}^{[-1]} > 30 \text{ AND } ini_{frame} < 150\right) \text{ OR } \left(\overline{E}_{tl} - E_{tl} > 30\right) \tag{111}$$
$$\alpha_{tl} = 0.03 \text{ otherwise } \alpha_{tl} = 0.02$$

The energy dynamics feature $\overline{E}_{dyn}$ is just an LP-filtered version of the difference between E_{th} and E_{tl}. That is

$$\overline{E}_{dyn} = 0.9\overline{E}_{dyn}^{[-1]} + 0.1(E_{th} - E_{tl}) \tag{112}$$

5.1.11.2 Second stage of noise energy update

In the second stage of the noise energy update, the critical bands not updated in the first stage are updated only if the current frame is inactive. However, the SAD decision obtained in clause 5.1.12, which is based on the SNR per critical band, is not used for determining whether the current frame is inactive and whether the noise energy is to be updated. Another decision is performed based on other parameters not directly dependent on the SNR per critical band. The basic parameters used for the noise update decision are:

– pitch stability

– signal non-stationarity

– normalized correlation (voicing)

– ratio between 2nd-order and 16th-order LP residual error energies

These parameters have generally low sensitivity to the noise level variations. Another set of parameters is calculated to cover harmonic (tonal) signals and, in particular, music. These parameters prevent the noise energy to be updated, when strong harmonicity or tonality is detected even when its energy is low. The parameters related to the detection of tonal signals are

– spectral diversity

– complementary non-stationarity

– HF energy

– tonal stability

The reason for not using the SAD decision for noise update is to make the noise estimation robust to rapidly changing noise levels. If the SAD decision was used for the noise update, a sudden increase in noise level would cause an increase of SNR even for inactive speech frames, preventing the noise estimator to update, which in turn would maintain the SNR high in the following frames. Consequently, the noise update would be blocked and some other logic would be needed to resume the noise adaptation.

5.1.11.2.1 Basic parameters for noise energy update

The pitch stability counter is computed as

$$pc = \left| d^{[0]} - d^{[-1]} \right| + \left| d^{[1]} - d^{[0]} \right| \tag{113}$$

where $d^{[0]}$, $d^{[1]}$ and $d^{[-1]}$ are the OL pitch lags for the first half-frame, second half-frame and the second half-frame of the pervious frame. The pitch stability is true if the value of pc is less than 12. Further, for frames with low voicing, pc is directly set to 12 to indicate pitch instability. That is

$$\text{if } (C_{norm}^{[0]} + C_{norm}^{[1]} + C_{norm}^{[2]})/3 + r_e < th_{Cpc} \text{ then } pc = 12, \tag{114}$$

where $C_{norm}^{[i]}$ are the normalized raw correlations as defined in clause 5.1.10.7 and r_e is a correction added to the normalized correlation in order to compensate for the decrease of normalized correlation in the presence of background noise, defined in clause 5.1.10.6. The voicing threshold $th_{Cpc} = 0.52$ for WB inputs, and $th_{Cpc} = 0.65$ for NB inputs.

Signal non-stationarity is analysed based on the product of ratios between the current frame energy per critical band and its long-term average per critical band. The average long-term energy per critical band is calculated as

$$\overline{E}_{CB}(i) = \alpha_e \overline{E}_{CB}^{[-1]}(i) + (1 - \alpha_e)(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i)) \text{, for } i = b_{min} \text{ to } b_{max}, \tag{115}$$

where $b_{min} = 0$ and $b_{max} = 19$ in case of WB signals, and $b_{min} = 1$ and $b_{max} = 16$ in case of NB signals. The update factor α_e is a linear function of the relative frame energy, defined in clause 5.1.5.2 and it is given as follows

$$\alpha_e = 0.064E_{rel} + 0.75 \text{, constrained by } \alpha_e \leq 0.999 \tag{116}$$

where all negative values of E_{rel} are replaced by 0. The frame non-stationarity is then given by the product of the ratios between the frame energy and its long-term average calculated in the previous frame. That is

$$nonstat = \prod_{i=b_{min}}^{b_{max}} \frac{\max(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i), \overline{E}_{CB}^{[-1]}(i)) + 1}{\min(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i), \overline{E}_{CB}^{[-1]}(i)) + 1} \tag{117}$$

The voicing factor for noise update is given by

$$voicing = (C_{norm}^{[0]} + C_{norm}^{[1]})/2 + r_e \tag{118}$$

The ratio between the LP residual energy after 2nd-order and 16th-order analysis is given by

$$resid\_ratio = \frac{E(2)}{E(16)} \tag{119}$$

where $E(2)$ and $E(16)$ are the LP residual energies after 2nd-order and 16th-order analysis, and computed in the Levinson-Durbin recursion (see clause 5.1.9.4). This ratio reflects the fact that, to represent a signal spectral envelope, a higher order of LP is generally needed for speech signal than for noise. In other words, the ratio between $E(2)$ and $E(16)$ is expected to be lower for noise than for active speech.

5.1.11.2.2 Spectral diversity

The basic parameters for noise estimation have their limitations for certain music signals, such as piano concerts or instrumental rock and pop. Spectral diversity gives information about significant spectral changes. The changes are tracked in the frequency domain in critical bands by comparing energies in the first spectral analysis of the current frame with the second spectral analysis two frames ago. The energy per critical band corresponding to the first spectral analysis of the current frame is denoted as $E_{CB}^{[0]}(i)$ and is defined in clause 5.1.5.2. Let the energy per critical band corresponding to the second spectral analysis two frames ago be denoted as $E_{CB}^{[-3]}(i)$. For all bands higher than 9, the maximum and the minimum of the two energies is found as

$$\begin{aligned} E_{max}(i) &= \max\left\{ E_{CB}^{[0]}(i), E_{CB}^{[-3]}(i) \right\} \\ E_{min}(i) &= \min\left\{ E_{CB}^{[0]}(i), E_{CB}^{[-3]}(i) \right\} \end{aligned}, \quad \text{for } i = 10,..,b_{max}, \tag{120}$$

where $b_{max} = 19$ in case of WB signals, and $b_{max} = 16$ in case of NB signals. The energy ratio is the calculated as

$$E_{rat}(i) = \frac{E_{max}(i)}{E_{min}(i)}, \quad \text{for } i = 10,..,b_{max}. \tag{121}$$

The spectral diversity is then calculated as the normalized weighted sum of the ratios in all critical bands with the weight itself being the maximum energy $E_{max}(i)$. That is

$$p_{div} = \frac{\displaystyle\sum_{i=10}^{b_{max}} E_{max}(i) E_{rat}(i)}{\displaystyle\sum_{i=10}^{b_{max}} E_{max}(i)} \tag{122}$$

The spectral diversity is used as an auxiliary parameter for the complementary non-stationarity described below.

5.1.11.2.3 Complementary non-stationarity

The complementary non-stationarity is motivated by the fact that the non-stationarity described in clause 5.1.11.2.1 and calculated in equation (117) is low when a sharp energy attack in a harmonic signal is followed by a slow energy decay. In this case, the average long-term energy per critical band, $\overline{E}_{CB}(i)$, slowly increases after the attack whereas the

current energy per critical band, $0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i)$, slowly decreases. At certain point (few frames after the attack frame) they are the same yielding only a small value of the *nonstat* parameter. This indicates to the noise estimation logic an absence of active signal which is wrong. It may lead to a false update of the background noise and consequently a collapse of the SAD algorithm.

To overcome this problem, there is an alternative calculation of the average long-term energy per critical band. It is calculated in the same way as in equation (115) but with a different factor. That is

$$\overline{F}_{CB}(i) = \beta_e \overline{F}_{CB}^{[-1]}(i) + (1 - \beta_e)(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i)), \text{ for } i = b_{\min} \text{ to } b_{\max}. \tag{123}$$

where $\overline{F}_{CB}(i)$ is initialized to 0.03. The update factor $\beta_e = \alpha_e$ and reset to 0 if $p_{div} > 5$. The complementary non-stationarity parameter is then calculated in the same way as *nonstat* but using $\overline{F}_{CB}^{[-1]}(i)$ instead of $\overline{E}_{CB}^{[-1]}(i)$. That is:

$$nonstat2 = \prod_{i=b_{\min}}^{b_{\max}} \frac{\max(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i), \overline{F}_{CB}^{[-1]}(i)) + 1}{\min(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i), \overline{F}_{CB}^{[-1]}(i)) + 1} \tag{124}$$

The complementary non-stationarity must be used by the noise estimation logic only in certain signal passages. These are characterized by the parameter $\overline{a}_{pred}$ which can be described as the average non-binary decision combined from non-stationarity and tonal stability. That is

if *nonstat* > th_{stat} OR $p_{tonal} = 1$ then $\overline{a}_{pred} = 0.99\overline{a}_{pred} + 0.01 \times 1$ otherwise $\overline{a}_{pred} = 0.99\overline{a}_{pred} + 0.01 \times 0$

where $\overline{a}_{pred}$ is in the range [0; 1] and p_{tonal} is the tonal stability described in clause 5.1.11.2.5 and defined in equation (136).

5.1.11.2.4   HF energy content

The HF energy content represents another parameter, which is used for the detection of certain noise-like musical signals such as cymbals or low-frequency drums. This parameter is calculated as

$$p_{hfE} = \frac{\sum_{i=10}^{b_{\max}} E_{CB}(i)}{9 \sum_{i=b_{\min}}^{9} E_{CB}(i)}, \text{ constrained by } p_{hfE} < 10 \tag{125}$$

but only for frames that have at least a minimal HF energy, i.e. when both the numerator and the denominator of the above equation are higher than 100. If this is not fulfilled, $p_{hfE} = 0$. Finally, the long-term value if this parameter is calculated as

$$\overline{p}_{hfE} = 0.9\overline{p}_{hfE}^{[-1]} + 0.1p_{hfE} \tag{126}$$

where $\overline{p}_{hfE}^{[-1]}$ is initialized to zero.

5.1.11.2.5   Tonal stability

The tonal stability exploits the harmonic spectral structure of certain musical signals. In the spectrum of such signals there are tones which are stable over several consecutive frames. To exploit this feature, it is necessary to track the positions and shapes of strong spectral peaks. The tonal stability is based on a correlation between the spectral peaks in the current frame and the past frame. The input to the algorithm is an average logarithmic energy spectrum, defined as

$$E_{dB}(k) = 10\log\left[0.5\left(E_{BIN}^{[0]}(k) + E_{BIN}^{[1]}(k)\right)\right], \quad k = 0, \ldots, 127, \tag{127}$$

where $E_{BIN}(k)$ is defined in clause 5.1.5.2 and the superscripts [0] and [1] denote the first and the second spectral analysis, respectively. In the following text, the term "spectrum" will refer to the average logarithmic energy spectrum, as defined by the above equation.

The tonal stability is calculated in three stages. In the first stage, indices of local minima of the spectrum are searched in a loop and stored as i_{min}. This is described by the following equation

$$i_{min} = \left(E_{dB}(i-1) > E_{dB}(i)\right) \wedge \left(E_{dB}(i) < E_{dB}(i+1)\right), \quad \forall i \;\; i = 1,\dots,126, \tag{128}$$

The index 0 is added to i_{min} if $E_{dB}(0) < E_{dB}(1)$. Consequently, the index 127 is added to i_{min}, if $E_{dB}(127) < E_{dB}(126)$. Let us denote the total number of minima found as N_{min}. The second stage consists of calculating a spectral floor and its subtraction from the spectrum. The spectral floor is a piece-wise linear function which runs through the detected local minima. Every piece between two consecutive minima $i_{min}(x)$ and $i_{min}(x+1)$ can be described by a linear function as

$$f(j) = k.(j - i_{min}(x)) + q, \quad j = i_{min}(x),\dots,i_{min}(x+1), \tag{129}$$

where k is the slope of the line and $q = E_{dB}(i_{min}(x))$. The slope is calculated by

$$k = \frac{E_{dB}(i_{min}(x+1)) - E_{dB}(i_{min}(x))}{i_{min}(x+1) - i_{min}(x)} \tag{130}$$

Thus, the spectral floor is a logical connection of all pieces. The leading bins of the spectrum up to $i_{min}(0)$ and the terminating bins of the spectrum from $i_{min}(N_{min}-1)$ are set to the spectral values themselves, i.e.

$$
\begin{aligned}
F(j) &= E_{dB}(j) & j &= 0,\dots,i_{min}(0)-1 \\
F(j) &= f(j) & j &= i_{min}(0),\dots,i_{min}(N_{min}-1)-1. \\
F(j) &= E_{dB}(j) & j &= i_{min}(N_{min}-1),\dots,127
\end{aligned} \tag{131}
$$

Finally, the spectral floor is subtracted from the spectrum by

$$E_{dB,res}(j) = E_{dB}(j) - F(j), \quad i = 0,\dots,127 \tag{132}$$

and the result is the residual spectrum. The calculation of the spectral floor and its subtraction is illustrated in the following figure.
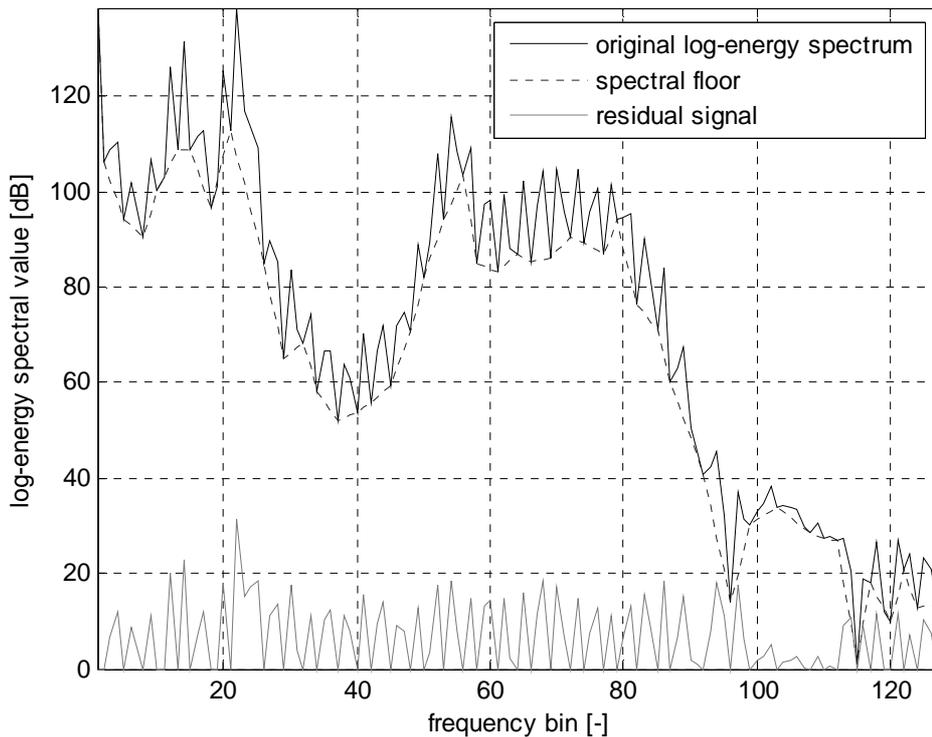
Figure 9 : Spectral floor in the tonal stability

The third stage of the tonal stability calculation is the calculation of the correlation map and the long-term correlation map. This is again a piece-wise operation. The correlation map is created on a peak-by-peak basis where each two consecutive minima delimit one peak. Let us denote the residual spectrum of the previous frame as $E_{dB,res}^{[-1]}(j)$. For every peak in the current residual spectrum, normalized correlation is calculated with the previous residual spectrum. The correlation operation takes into account all indices (bins) of that peak delimited by two consecutive minima, i.e.

$$M_{cor}(i_{\min}(x):i_{\min}(x+1)) = \frac{\left(\sum\limits_{j=i_{\min}(x)}^{i_{\min}(x+1)-1} E_{dB,res}(j)E_{dB,res}^{[-1]}(j)\right)^2}{\sum\limits_{j=i_{\min}(x)}^{i_{\min}(x+1)-1}\left(E_{dB,res}(j)\right)^2 \sum\limits_{j=i_{\min}(x)}^{i_{\min}(x+1)}\left(E_{dB,res}^{[-1]}(j)\right)^2}, \quad x=0,\dots,N_{\min}-2 \quad (133)$$

where the leading bins up to $i_{\min}(0)$ and the terminating bins from $i_{\min}(N_{\min}-1)$ are set to zero. The figure below shows a graphical representation of the correlation map.
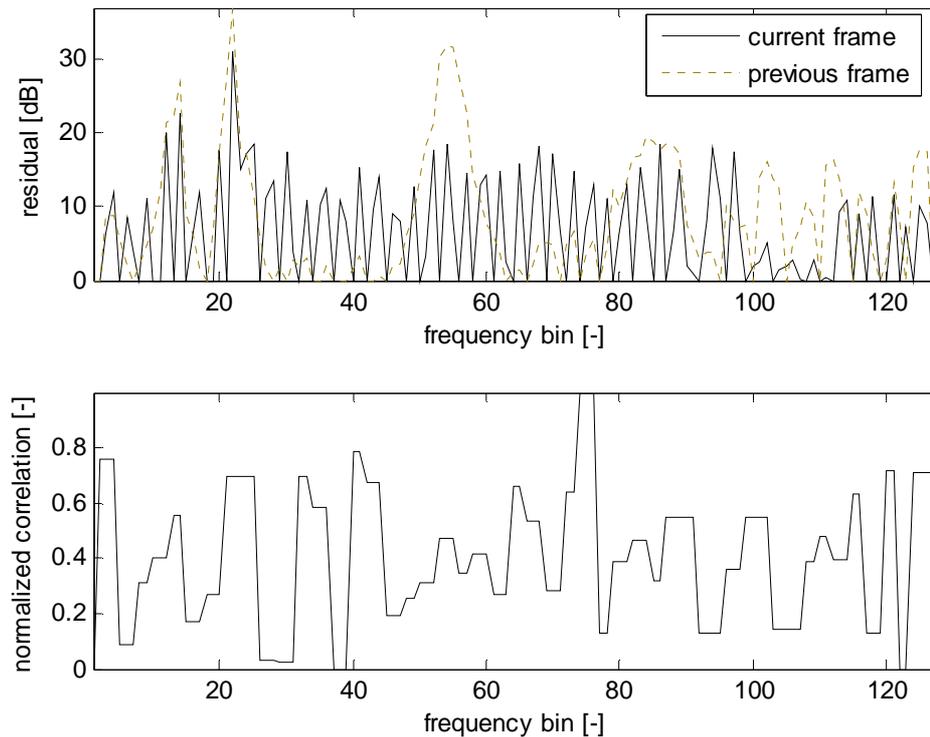
Figure 10 : Correlation map in the tonal stability calculation

The correlation map of the current frame is used to update its long-term value, which can be expressed as

$$\overline{M}_{cor}(k) = \alpha_{map}\overline{M}_{cor}(k) + (1-\alpha_{map})M_{cor}(k), \quad k = 0,\ldots,N_{SPEC}-1 \tag{134}$$

where $\alpha_{map} = 0.9$. If any value of $\overline{M}_{cor}(j)$ exceeds the threshold of 0.95, the flag f_{strong} is set to one, otherwise it is set to zero. The long-term correlation map is initialized to zero for all k. Finally, all bins of $\overline{M}_{cor}(k)$ are summed together by

$$m_{sum} = \sum_{j=0}^{127} \overline{M}_{cor}(j) \tag{135}$$

In case of NB signals, the correlation map in higher bands is very low due to missing spectral content. To overcome this deficiency, m_{sum} is multiplied by 1.53.

The decision about tonal stability is taken by subjecting m_{sum} to an adaptive threshold th_{tonal}. This threshold is initialized to 56 and it is updated in every frame by

$$\text{if } m_{sum} > 56 \text{ then } th_{tonal} = th_{tonal} - 0.2 \text{ otherwise } th_{tonal} = th_{tonal} + 0.2$$

and is upper limited by 60 and lower limited by 49. Thus, it decreases when the summed correlation map is relatively high, indicating a good tonal segment, and increases otherwise. When the threshold is lower, more frames will be classified as tonal, especially at the end of active music periods. Therefore, the adaptive threshold may be viewed as a hangover.

The p_{tonal} parameter is set to one whenever m_{sum} is higher than th_{tonal} or when the flag f_{strong} is set to one. That is:

$$\text{if } m_{sum} > th_{tonal} \text{ OR } f_{strong} = 1 \text{ then } p_{tonal} = 1 \text{ otherwise } p_{tonal} = 0 \tag{136}$$

5.1.11.2.6 High frequency dynamic range

From the residual spectrum $E_{dB,res}$ as described in equation 116, another parameter is computed. This parameter is called the high frequency dynamic D_{hf} is derived from the high band spectral dynamic of the residual spectrum and is used to set the high frequency dynamic range flag F_{hf} which is used inside the GSC to decide about the number of subframe and the bit allocation. The high frequency dynamic is compute as the average of the last 40 bin from the residual spectrum:

$$D_{hf(t)} = 0.7 \cdot D_{hf(t-1)} + 0.3 \cdot \left(\frac{1}{40} \sum_{i=L-40}^{i=L} E_{dB,res}(i) \right)$$ (137)

And the high frequency dynamic range flag is set depending on the past values and the actual high frequency dynamic as :

$$if \left(D_{hf\,(t)} < 9.6dB \,\&\, F_{hf(t-1)}) == 1 \right)$$
$$F_{hf(t)} = 0$$
$$D'_{hf} = D_{hf\,(t)}$$ (138)
$$else\,if \left(\left(D_{hf\,(t)} - D'_{hf} \right) > 4.5dB \right)$$
$$F_{hf(t)} = 1$$

Where t represents the frame at time t and D'_{hf} represents the average high frequency dynamic at when the last time the flag F_{hf} was set to 0.

5.1.11.2.7 Combined decision for background noise energy update

The noise energy update decision is controlled through the logical combination of the parameters and flags described in the previous sections. The combined decision is a state variable denoted p_{nup} which is initially set to 6, and which is decremented by 1 if an inactive frame is detected or incremented by 2 if an active frame is detected. Further, p_{nup} is bounded by 0 and 6. The following diagram shows the conditions under which the state variable p_{nup} is incremented by 2 in each frame.
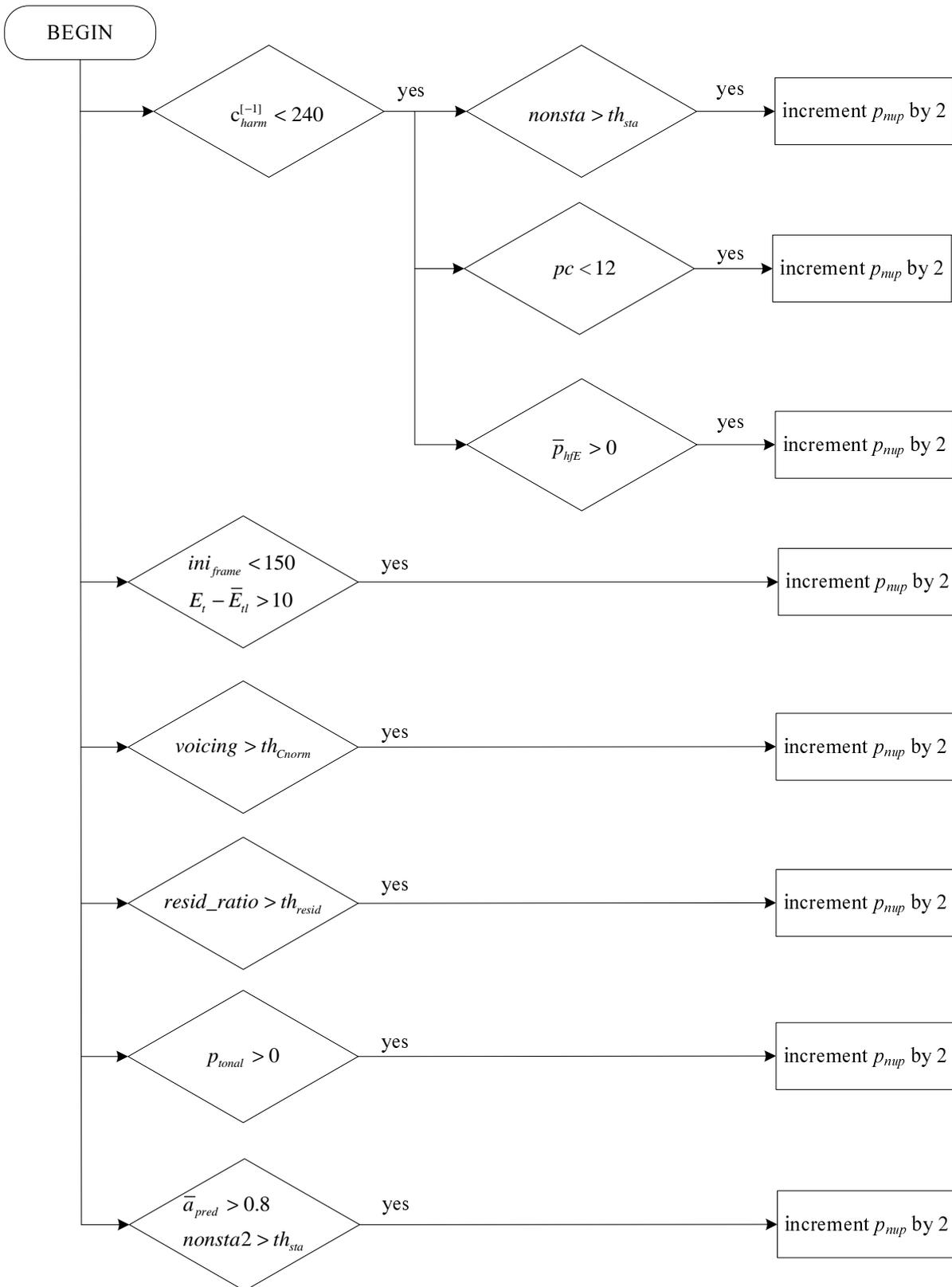
Figure 11 : Incrementing the state variable for background noise energy update

where, for WB signals, $th_{sta} = 350000$, $th_{Cnorm} = 0.85$ and $th_{resid} = 1.6$, and for NB signals, $th_{sta} = 500000$, $th_{Cnorm} = 0.7$ and $th_{resid} = 10.4$. If p_{nup} is not incremented in any of the conditions from the above diagram, it is automatically decremented by 1. Therefore, it takes at least 6 frames before p_{nup} reaches 0 which signals the subsequent logic that background noise energy can be updated. The final decision about background noise energy update is described in the subsequent section.

5.1.11.3 Energy-based parameters for noise energy update

The parameters in this section are used in addition to the p_{nup} described in the previous section to control when it is possible and safe to allow the noise estimate sub-bands to be increased according to the pre calculated noise estimate N_{tmp} calculated in equation (102).

5.1.11.3.1 Closeness to current background estimate

Similar to *nonstat* and *nonstat2* the parameter *nonstat*$_B$ represents a spectral difference. The difference is that it is the closeness/variation compared to the current background noise estimate that is measured. The calculation of the feature also differs in calculation during initialization, that is $ini_{frame} < 100$, or during normal operation. During initialization the comparison is made using a constant, $E_{MIN} = 0.0035$ which is the initialization value for the sub-band energies, as shown in

$$nonstat_B = \sum_{i=2}^{16} \left| \log(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i) + 1) - \log(E_{MIN} + 1) \right| \tag{139}$$

This is done to reduce the effect of decision errors in the background noise estimation during initialization. After the initialization period the calculation is made using the current background noise estimate of the respective sub-band, according to:

$$nonstat_B = \sum_{i=2}^{16} \left| \log(0.5E_{CB}^{[0]}(i) + 0.5E_{CB}^{[1]}(i) + 1) - \log(\overline{N}_{CB}^{[-1]}(i) + 1) \right| \tag{140}$$

It is worth noting that the calculation of *nonstat*$_B$ is not dependent of the band width as it is made over the same sub-bands regardless of the input bandwidth.

5.1.11.3.2 Features related to last correlation or harmonic event

Two related features are created which relate to the occurrence of frames where correlation or harmonic events are detected. The first is a counter, $\overline{c}_{harm}$, that keeps track of how many frames that have passed since the last frame where correlation or harmonic event has occurred. That is if a correlation or harmonic event is detected the counter is reset otherwise it is incremented by one, according to:

$$\text{if } \left((C_{norm}^{[0]} + C_{norm}^{[1]})/2 > 0.85 \text{ OR } p_{tonal} > 0 \right) \text{then } \overline{c}_{harm} = 0 \text{ else } \overline{c}_{harm} = \overline{c}_{harm} + 1 \tag{141}$$

where $C_{norm}^{[.]}$ is the normalized correlation in the first or the second half-frame and p_{tonal} is the result of the tonal detection in clause 5.1.11.2.5. If the counter $\overline{c}_{harm}$ is larger than 1 it is limited to 1 if $E_t < 15.0$ or if $ini_{frame} > 10$ AND $(E_t - \overline{E}_t) > 7.0$ is 1. Depending on the estimated short term variance of the input frame energy the current value of the counter $\overline{c}_{harm}$ can be reduced to one quarter of its value (or 1 if it was less than 4). The reduction is made for frames where $\overline{c}_{harm} > 1$ where $E_t > 30.0$ and the short therm variance estimate of the frame energy is larger than 8.0. The other feature is the long term measure of the relative occurrence of correlation or tonal frames. It is represented as a scalar value, $\overline{c}_{ev}$, which is updated using a first order AR-process with different time constants depending on if the current frame is classified as a correlation/tonal frame or not according to:

$$\overline{c}_{ev} = (1 - \alpha)\overline{c}_{ev}^{[-1]} + \alpha(c_{harm} == 0) \text{ where } \alpha = \begin{cases} 0.03 & \text{if } c_{harm} == 0 \\ 0.01 & \text{otherwise} \end{cases} \tag{142}$$

where the test, $\overline{c}_{harm} == 0$, represents a detection of a correlation/tonal event.

5.1.11.3.3 Energy-based pause detection

To improve the tracking of the background noise the energy pause detector monitors the number of frames since the frame energy got close to the long-term minimum frame energy estimate $\overline{E}_{tl}$. For inactive frames the counter is 0 or higher, $c_{bg} \geq 0$, where positive integers represent the number of frames since the start of the current pause. When active content is detected the counter is set and kept at, $c_{bg} = -1$. Initially $c_{bg} = 0$ so the detector is in a inactive state and checks for an energy increase relative the long term minimum energy tracker $\overline{E}_{tl}$ that could triggers a transition to and active state:

$$\text{if } c_{bg} \geq 0 \text{ AND } (E_t - \overline{E}_{tl}) > 5 \text{ then } c_{bg} = -1 \tag{143}$$

If the detector is in an active state the detector checks if the frame energy once again has come close to the long term minimum energy

$$\text{if } c_{bg} == -1 \text{ AND } (E_t - \overline{E}_{tl}) < 5 \text{ then } c_{bg} = 0 \tag{144}$$

The final step in the update of this parameter is to increment the counter if the detector is in an inactive state

$$\text{if } c_{bg} \geq 0 \text{ then } c_{bg} = c_{bg} + 1 \tag{145}$$

5.1.11.3.4 Long-term linear prediction efficiency

This section describes how the residual energies from the linear prediction analysis made in clause 5.1.9 can be used to create a long term feature that can be used to better determine when the input signal is active content or background noise based on the input signal alone.

The analysis provides several new features by analysing the linear prediction gain going from 0th-order to 2nd-order linear prediction and going from 2nd-order to 16th-order prediction. Starting with the 2nd order prediction that is analysed compared to the energy of the input signal, or 0th-order prediction. For a more stable long term feature the gain is calculated and limited as

$$g_{LP\_0\_2} = \max(0, \min(8, E(0)/E(2))) \tag{146}$$

where $E(0)$ is the energy of the input signal and $E(2)$ is the residual energy after the second-order linear prediction (see clause 5.1.9.4). The limited prediction gain is then filtered in two steps to create long term estimate of this gain through. The first is made using

$$\overline{g}_{LP\_0\_2} = 0.85 \overline{g}_{LP\_0\_2}^{[-1]} + 0.15 g_{LP\_0\_2} \tag{147}$$

and typically this will become either 0 or 8 depending on the type of background noise in the input once there is a segment of background only input. A second feature is then created using the difference between the first long term feature and the frame by frame limited prediction gain according to:

$$g_{ad\_0\_2} = \left| \overline{g}_{LP\_0\_2} - g_{LP\_0\_2} \right| \tag{148}$$

This will give an indication of the current frames prediction gain compared to the long term gain. This difference is used to create a second long term feature, this is done using a filter with different filter coefficient depending on if the long term difference is higher or lower than the currently estimated average difference according to

$$\overline{g}_{ad\_0\_2} = (1-\alpha)\overline{g}_{ad\_0\_2}^{[-1]} + \alpha g_{ad\_0\_2} \quad \text{where } \alpha = \begin{cases} 0.1 & \text{if } g_{ad\_0\_2} < \overline{g}_{ad\_0\_2}^{[-1]} \\ 0.2 & \text{otherwise} \end{cases} \tag{149}$$

This second long term features is then then combined with the frame difference to prevent the filtering from masking occasional high frame differences, the final parameter is the maximum of the frame and the long term version of the feature

$$g_{\max\_0\_2} = \max(\overline{g}_{ad\_0\_2}, g_{ad\_0\_2}) \tag{150}$$

The feature created using the difference between 2nd order prediction and 16th order prediction is analysed is slightly different. The first step here is also to prediction gain is calculated as

$$g_{LP\_2\_16} = \max(0, \min(8, E(2)/E(16)))$$ (151)

Where $E(2)$ represents the residual energy after a 2nd order linear prediction and $E(16)$ is the residual energy after a 16th order linear prediction, see clause 5.1.9.4. This limited prediction gain is then used for two long term estimate of this gain, one where the filter coefficient differs if the long term estimate is to be increased or not as shown in

$$\overline{g}_{LP\_2\_16} = (1-\alpha)\overline{g}_{LP\_2\_16}^{[-1]} + \alpha g_{LP\_2\_16} \quad \text{where } \alpha = \begin{cases} 0.20 & \text{if } g_{LP\_2\_16} > \overline{g}_{LP\_2\_16}^{[-1]} \\ 0.03 & \text{othrewise} \end{cases}$$ (152)

The second long term estimate uses a constant filter coefficient as, according to

$$\overline{g}_{LP2\_2\_16} = (1-\beta)\overline{g}_{LP2\_2\_16}^{[-1]} + \beta g_{LP\_2\_16} \quad \text{where } \beta = 0.02$$ (153)

For most types of background signals both will be close to 0 but since they have different responses to content where e the 16th order linear prediction is needed (typically for speech and other active content). The first $\overline{g}_{LP\_2\_16}$ will usually be higher than the second $\overline{g}_{LP2\_2\_16}$. This difference between the long term features is measured according to

$$g_{ad\_2\_16} = \overline{g}_{LP\_2\_16} - \overline{g}_{LP2\_2\_16}$$ (154)

Which is used as an input to the filter which creates the third long term feature according to

$$\overline{g}_{ad\_2\_16} = (1-\alpha)\overline{g}_{ad\_2\_16}^{[-1]} + \alpha g_{ad\_2\_16} \quad \text{where } \alpha = \begin{cases} 0.02 & \text{if } g_{ad\_2\_16} < \overline{g}_{ad\_2\_16}^{[-1]} \\ 0.05 & \text{otherwise} \end{cases}$$ (155)

Also, this filter uses different filter coefficients depending on if the third long term signal is to be increased or not. Also here the long term signal is combined with the filter input signal to prevent the filtering from masking occasional high inputs for the current frame. The final parameter is then the maximum of the frame and the long term version of the feature

$$g_{\max\_2\_16} = \max(\overline{g}_{ad\_2\_16}, g_{ad\_2\_16})$$ (156)

Note that also some of the other calculated features in this sub section are used in the combination logic for the noise estimation, $\overline{g}_{ad\_2\_16}$, $g_{\max\_2\_16}$, $g_{\max\_0\_2}$, $g_{ad\_0\_2}$, and $g_{LP\_0\_2}$

5.1.11.3.5 Additional long-term parameters used for noise estimation

Some additional parameters that processed to create long term estimates are three measures the relation of the current frames energy compared to the energy of the noise estimate. The first calculates the difference between the current frame energy and the level of the current noise estimate this is then filtered to build a long term estimate according to

$$\overline{N}_{dist} = (1-\beta)\overline{N}_{dust}^{[-1]} + \beta(E_t - \overline{N}_t) \quad \text{where } \beta = 0.03$$ (157)

Another feature estimates a long term estimate of how often the current frame energy is close to the level of the background estimate using:

$$\overline{N}_{track} = (1-\beta)\overline{N}_{track}^{[-1]} + \beta((E_t - \overline{N}_t) < 10) \quad \text{where } \beta = 0.03$$ (158)

The third estimate is a second order estimate for the number of frames that the current input has been close to the noise estimate. This is simply a counter is reset if the long term estimate $\overline{N}_{track}$ is higher than a threshold and incremented otherwise, as shown in

$$c_{Ntr} = \begin{cases} 0 & \text{if } \overline{N}_{track} \geq 0.05 \\ c_{Ntr}^{[-1]} + 1 & \text{otherwise} \end{cases}$$ (159)

The last additional features calculates an long term estimate of the difference in the current frame energy to the long term minimum energy feature, this is done by low pass filtering the calculated energy difference according to

$$\overline{E}_{tl\_dist} = (1-\beta)\overline{E}_{dust}^{[-1]} + \beta(E_t - \overline{E}_{tl}) \quad \text{where} \quad \beta = 0.03 \tag{160}$$

5.1.11.4 Decision logic for noise energy update

Already in the first step of the noise estimation (see clause 5.1.11.1), the current noise estimate has been reduced in sub-bands where the background noise energy was higher than the sub-band energy for the current frame. The decision logic described in this subsection shows how it is decided when to update and how the step size, $step_{size}$, if the update is adapted based on the earlier described features or combinations thereof.

Every frame an attempt is made to adjust the background noise estimate upwards, as it is important not to do the update in active content several conditions are evaluated in order to decide if update is possible and how large an update that should be allowed. As it is always allowed to make downwards updates it is equally important that possible updates are not prevented for extended times as this will affect the efficiency of the SAD. The noise update uses a flag to keep track of the number of prevented noise updates, c_{first_updt}, the same flag is also used to indicate that no update has taken place. The counter c_{first_updt} is initialized to the value 0 to indicate that no update has been done so far. When updates are successful it is set to 1 and for failed updates the counter is incremented by 1.

The major decision step in the noise update logic is whether an update is to be made or not and this is formed by evaluation of the following logical expression

$$f_{UPDATE} = (f_{bg\_MASK} \text{ AND } (f_{bg\_nup} \text{ OR } f_{bg\_dynamic} \text{ OR } f_{bg\_tracking} \text{ OR } f_{bg\_new})) \text{ OR } f_{bg\_int} \tag{161}$$

where f_{bg_MASK} ensures that it is safe to do an update provided that any of the four pause detectors, f_{bg_nup}, $f_{bg\_dynamic}$, $f_{bg\_tracking}$, and f_{bg_new} indicate that an update is allowed. Note that the last term in the condition f_{bg_int} is not is not combined with f_{bg_MASK} as it handles the noise estimation during initialization.

Starting with the mask which ensures that the normal updates only can occur when the current frame energy is close to the estimated long-term minimum energy, $\overline{E}_{tl}$ (see clause 5.1.11.1), is adjusted with a level dependent scaling of the estimated frame energy variations, $\overline{E}_{vh2}$, according to

$$f_{bg\_MASK} = E_t < \overline{E}_{tl} + (1.5 + 1.5(\overline{E}_t < 50))\overline{E}_{vh2} \tag{162}$$

The first pause detector f_{bg_nup} is based on the metric p_{nup} control logic described in subclause 5.1.11.2.7, when p_{nup} is 0 updates are allowed, that is

$$\text{if } p_{nup} == 0 \text{ then } f_{bg\_nup} = 1 \text{ else } f_{bg\_nup} = 0 \tag{163}$$

The second pause detector allows for updates for low energy frames if the estimated signal dynamics is high and a sufficient number of frames have passed since the last correlation event, that is

$$f_{bg\_dynamic} = \overline{E}_{dyn} > 15 \text{ AND } E_t < \overline{E}_{tl} + 2\overline{E}_{vh2} \text{ AND } \overline{c}_{harm} > 20 \tag{164}$$

The third pause detector allows updates when there are consecutive frames that are similar in energy to the current low level frames in a row,

$$f_{bg\_track} = \overline{N}_{track} > 0.9 \tag{165}$$

The last detector is itself a combination of a mask and two pause detectors and mainly uses the additional features described in subclause 5.1.11.3.4, the detector is evaluated using

$$f_{bg\_new} = f_{bg\_M1} \text{ AND } (f_{bg\_A1} \text{ OR } f_{bg\_A2}) \tag{166}$$

where f_{bg_M1} is the mask for the detector and f_{bg_A1} and f_{bg_A2} are the additional detectors. For this detector the following seven flags are first evaluated. The first flag signals that the frame energy close to background noise energy where the threshold is adapted to the estimated frame to frame energy variations, as

$$f_{enr\_bg} = E_t < \overline{E}_{tl} + (1.5 + 1.5(\overline{E}_t < 50))\overline{E}_{vh2} \tag{167}$$

The second flag signals that the high linear prediction gain with 2nd order model if for a stationary signal, as

$$f_{cns\_bg} = g_{LP\_0\_2} > 7.95 \text{ AND } nonsta < 10^3 \tag{168}$$

The third flag signals that there is a low linear prediction gain for 16th order linear prediction

$$f_{lp\_bg} = g_{max\_2\_16} < 0.1 \tag{169}$$

The fourth flag signals that the current frame has low spectral fluctuation

$$f_{ns\_bg} = nonsta < 10^5 \tag{170}$$

The fifth flag signals that the long term correlation is low

$$f_{haco\_bg} = \overline{c}_{ev} < 0.5 \tag{171}$$

The sixth flag signals low long term correlation value including the current frame

$$f_{haev\_bg} = c_{ev\_max} < 0.4 \tag{172}$$

The seventh and last flag signals a non-speech like input signal

$$f_{c2\_bg} = (g_{ad\_0\_2} > 0.5 \text{ AND } g_{LP\_0\_2} < 7.95) == 0 \tag{173}$$

Using the above flags possible to express the mask as

$$f_{bg\_M1} = f_{enr\_bg} \text{ OR } ((f_{cns\_bg} \text{ OR } f_{lp\_bg}) \text{ AND } f_{ns} \text{ AND } f_{bg\_haco} \text{ AND } f_{c2\_bg}) \tag{174}$$

The two additional detectors f_{bg_A1} and f_{bg_A2}, are also those build using sub detectors and additional conditions. Starting with f_{bg_A1} the sub detectors are:

$$f_{bg\_A1} = \begin{array}{l} (E_t < 55) \text{ AND } f_{c2\_bg} \text{ AND} \\ ((comb_{ahc} < 0.85 \text{ AND } (g_{max\_2\_16} < 0.1 \text{ AND } g_{max\_0\_2} < 0.1) \text{ OR} \\ comb_{ahc} < 0.15 \text{ OR } comb_{hcn} < 0.30) \end{array} \tag{175}$$

where the combination metrics $comb_{ahc}$ and $comb_{hcm}$ are combinations where the maximum of a number of metrics are used for the comparison

$$comb_{ahc} = \max(\max(\overline{a}_{pred}, \overline{c}_{ev}), \overline{g}_{ad\_2\_16}) \tag{176}$$

$$comb_{hcm} = \max(\max(\overline{c}_{ev}, \overline{g}_{max\_2\_16}), \overline{g}_{max\_0\_2}) \tag{177}$$

For the f_{bg_A2} sub detector

$$f_{bg\_A2} = (c_{ev\_max} < 0.4) \text{ AND } (\overline{a}_{pred} < 0.85) \text{ AND } (f_{c2\_bg} \text{ OR } f_{enr\_bg}) \tag{178}$$

where the combination metric $c_{ev\_max} = \max(\overline{c}_{ev}, c_{harm} == 0)$ is calculated as

$$c_{ev\_max} = \max(\overline{c}_{ev}, c_{harm} == 0) \tag{179}$$

The last term f_{bg_ini} in the f_{UPDATE} handles the special conditions of noise update during the initialization, which occurs during the 150 first frames after the codec start. Also the initialization flag is evaluated as a combination of two flags according to

$$f_{bg\_ini} = f_{bg\_ini\_1} \text{ AND } f_{bg\_ini\_2} \tag{180}$$

where the first flag test for initialization period and a sufficient number of frames without correlation event, according to

$$f_{bg\_ini\_1} = ini_{frame} < 150 \text{ AND } \overline{c}_{harm} > 5 \text{ AND } (E_t - \overline{E}_t) < 7 \tag{181}$$

The second flag evaluates a number of earlier calculated features against initialization specific thresholds according to

$$f_{bg\_ini\_2} = \begin{matrix} (\overline{a}_{pred} < 0.59 \text{AND } \overline{c}_{ev} < 0.23) \text{ OR} \\ \overline{a}_{pred} < 0.38 \text{ OR } \overline{c}_{ev} < 0.15 \text{ OR} \\ nonstaB < 50 \text{ OR } f_{bg\_nup} \text{ OR} \\ (E_t < 42 \text{ AND } \overline{c}_{harm} > 10 \text{ AND } \overline{c}_{ev} < 0.35 \text{ AND act}_{pred} < 0.80) \end{matrix} \tag{182}$$

Every frame an attempt is made to adjust the background noise estimate upwards, as it is important not to do the update in active content several conditions are evaluated in order to decide if update is possible and how large an update that should be allowed. At the same time it is important that possible updates are not prevented for extended times. The noise update uses a flag to keep track of the number of prevented noise updates. The same flag is also used to indicate that no update has taken place. The flag c_{first_updt} is initialized to the value 0 to indicate that no update has been done so far. When updates are successful it is set to 1 and for failed updates the counter is incremented by 1.

If the above condition f_{UPDATE} is evaluated to 0 the noise estimation only checks if the current content is music and if the following conditions

$$c_{Ntr} > 300 \text{ AND } \overline{c}_{ev} > 0.9 \text{ OR } \overline{N}_{tn} > 0 \tag{183}$$

is evaluated to 1 the sub-band noise level estimates are reduced to recover from noise updates made before or during music, the reduction is made per sub-band depending on if the current estimate is high enough

$$N_{CB}^{[0]}(i) = 0.98 N_{CB}^{[0]}(i) \quad \text{for all i where } N_{CB}^{[0]}(i) > 2E_{MIN} \tag{184}$$

And the $\overline{c}_{nup\_0}$ is updated according to the definition in equation (198) before noise estimation is terminated for this frame.

The following steps are taken when f_{UPDATE} is evaluated to 1. First the step size, $step_{size}$, is initially set to 0, before the process of determining if the noise update should be set to 1.0, 0.1, or 0.01. For the update $step_{size}$ to be 1.0 the following condition

$$(\overline{a}_{pred} < 0.85 \text{ AND } f_{bg\_Nup}) \tag{185}$$

And any of the following conditions

$$\overline{E}_{tl\_dist} < 10 \text{ OR } f_{bg\_sd\_1}) \text{ AND } \overline{N}_{dist} < 40 \text{ AND } (E_t - N_t) < 10 \tag{186}$$

$$c_{first\_updt} == 0 \text{ AND } c_{harmb} > 80 \text{ AND } \overline{c}_{p\_count} > 0.5 \tag{187}$$

$$f_{bg\_ini} \text{ AND } (f_{bg\_Nup} \text{ OR } nonstaB < 10 \text{ OR } \overline{c}_{harm} > 80) \tag{188}$$

is evaluated to 1 the step $step_{size}$ is set to 1.0 and c_{first_updt} is set to 1 before the noise estimation for the current frame is made

$$N_{CB}^{[0]}(i) = N_{tmp}(i) \quad \text{for all i} \tag{189}$$

Where $N_{tmp}(i)$ is the pre-calculated new noise estimate from subclause 5.1.11.1 and the noise estimation procedure is done for the current frame after the $\bar{c}_{nup\_0}$ in equation (198) is updated.

If the above condition failed then the $step_{size}$ is set to 0.1 if any of the four following conditions are met

$$(act_{pred} < 0.80)\text{ AND}(f_{bg\_Nup}\text{ OR }f_{PAU})\text{ AND}(\bar{c}_{haco} < 0.10) \tag{190}$$

$$(act_{pred} < 0.70)\text{ AND}(f_{bg\_Nup}\text{ OR }nonstaB < 17)\text{ AND}f_{bg\_pauj}\text{ AND}(\bar{c}_{haco} < 0.15) \tag{191}$$

$$\bar{c}_{harm} > 80\text{ AND }\bar{N}_t > 5\text{ AND }E_t < \min(1.0, \bar{E}_{tl} + 1.5\bar{E}_{vh2}) \tag{192}$$

$$\bar{c}_{harm} > 50\text{ AND }c_{first\_updt} > 30\text{ AND }f_{bg\_Nup}\text{ AND }c_{p\_count} > 0.5 \tag{193}$$

If the $step_{size}$ has been set to 0.1 is will be reduced to 0.01 if

$$f_{bg\_Nup} == 0\text{ AND }\bar{c}_{harm} < 50 \tag{194}$$

and the following condition is met

$$\bar{a}_{pred} > 0.6\text{ OR }(f_{bg\_ini} == 0\text{ AND }nonstaB > 8\text{ AND }(\bar{E}_{tl} - N_t) < 10 \tag{195}$$

If the $step_{size}$ is set to 0.1 or 0.01 the c_{first_updt} is set to 1 before the noise estimation for the current frame is made

$$N_{CB}^{[0]}(i) = N_{CB}^{[0]}(i) + updt_{step}(N_{tmp}(i) - N_{CB}^{[0]}(i)) \quad \text{for all } i \tag{196}$$

and the noise estimation procedure is done for the current frame after $\bar{c}_{nup\_0}$ is updated in equation (198).

I the conditions to set the $step_{size}$ to 0.1 failed the step size is $step_{size}$ is 0 and noise update has failed, after testing if the following condition is true

$$f_{bg\_Nup}\text{ OR }\bar{c}_{harm} > 100 \tag{197}$$

the c_{first_updt} is incremented and the noise estimation is done after the following update of $\bar{c}_{nup\_0}$

In all cases the noise estimation updates end with an update of $\bar{c}_{nup\_0}$ which is the long-term estimate of how frequent noise estimations could be possible according to

$$\bar{c}_{nup\_0} = (1-\beta)c_{nup\_0}^{[-1]} + \beta(p_{nup} == 0)\text{ where }\beta = 0.2 \tag{198}$$

where p_{nup} is calculated in clause 5.1.11.2.6.

5.1.12 Signal activity detection

In this module active signal is detected in each frame and the main flags for external use are the three flags f_{LSAD_HE}, f_{LSAD} and the combined f_{SAD}. These flags are set to one for the active signal, which is any useful signal bearing some meaningful information. Otherwise, they are set to zero indicating an inactive signal, which has no meaningful information. The inactive signal is mostly a pause or background noise. The three flags represent different trade-offs between quality and efficiency, and are used respectively by various subsequent processing modules.

The entire signal activity detection (SAD) module described in this section consists of three sub-SAD modules. Two of the modules, namely the SAD1 and the SAD2, work on the spectral analysis of the 12.8kHz sampled signal, see subclause 5.1.12.1 and 5.1.12.2 respectively for detailed descriptions. The third module, namely the SAD3, operates on the QMF sub-band filter that runs on the input sampling frequency, see subclause 5.1.12.6. A preliminary activity decision, f_{SAD}, is first obtained by combining two of the three sub-SAD modules, the SAD1 and SAD2, for input with bandwidth greater than NB, or directly from SAD1 for NB input. This preliminary decision is then further combined

with the decision output f_{SAD3} of the third sub-SAD module, the SAD3, depending upon the codec mode of operation and the input signal characteristic. The resulting decision is then feed to a DTX hangover module to produce the final output f_{SAD}.

Internally the flag f_{SAD_DTX} is used to always produce a flag with DTX hangover whether DTX is on or off. When this no longer is needed and DTX is on f_{SAD_DTX} replaces the combined f_{SAD} to reduce the number of variables used externally.

5.1.12.1   SAD1 module

The SAD1 module is a sub-band SNR based SAD with hangover that utilizes significance thresholds to reduce the amount off false detections for energy variations in the background noise. During SAD initialization period the following variables are set as follows

$$
\begin{aligned}
nb_{act\_frame} &= 3 \\
lp_{speech} &= 45.0 \\
f_{SAD\_reg\_h} &= 0 \\
f_{SAD\_reg\_l} &= 0 \\
f_{SAD\_cnt} &= 0 \\
f_{LSAD\_reg} &= 0 \\
f_{LSAD\_cnt} &= 0
\end{aligned}
\tag{199}
$$

The output of the SAD1 module is two binary flags (signal activity decisions) f_{LSAD_HE} and f_{LSAD}. The difference between them is due to the setting of parameters for the significance thresholds. The first binary decision f_{LSAD_HE} is used by the speech/music classification algorithm described in clause 5.1.13.5. The second binary decision f_{LSAD} is developed further and leads to the final SAD1 decision, f_{SAD}. Note that all decisions can be modified by the subsequent modules.

The spectral analysis described in clause 5.1.5 is performed twice per frame. Let $E_{CB}^{[0]}(i)$ and $E_{CB}^{[1]}(i)$ denote the energy per critical band for the first and second spectral analysis, respectively (as computed in clause 5.1.5.2). The average energy per critical band for the whole frame and part of the previous frame is computed as

$$
\overline{E}_{CB}(i) = 0.2\, E_{CB}^{[-1]}(i) + 0.4\, E_{CB}^{[0]}(i) + 0.4\, E_{CB}^{[1]}(i), \qquad i = b_{\min},\ldots, b_{\max}
\tag{200}
$$

where $E_{CB}^{[-1]}(i)$ denotes the energy per critical band from the second analysis of the previous frame, $b_{\min}$ and $b_{\max}$ hereafter denote respectively the minimum and the maximum critical band involved in the computation, where $b_{\min} = 1$, $b_{\max} = 16$ for NB input signals and $b_{\min} = 0$, $b_{\max} = 19$ for WB signals (see Table 2 in subclause 5.1.5.1). The signal-to-noise ratio (SNR) per critical band is then computed as

$$
SNR_{CB}(i) = \frac{\overline{E}_{CB}(i)}{N_{CB}(i)}, \qquad i = b_{\min},\ldots, b_{\max}, \text{ constrained by } SNR_{CB} \geq 1
\tag{201}
$$

where $N_{CB}(i)$ is estimated noise energy per critical band, as explained in clause 5.1.11.1. The average SNR per frame, in dB, is then computed using significance thresholds with two different settings

$$
SNR_{av\_HE} = 10\log\left(\sum_{i=b_{\min}}^{b_{\max}} \begin{cases} SNR_{CB}(i) & \text{if } SNR_{CB}(i) \geq sign_{thr\_HE} \\ min_{snr\_HE} & \text{othrewise} \end{cases} \right)
$$

$$
SNR_{av} = 10\log\left(\sum_{i=b_{\min}}^{b_{\max}} \begin{cases} SNR_{CB}(i) & \text{if } SNR_{CB}(i) \geq sign_{thr} \\ min_{snr} & \text{othrewise} \end{cases} \right)
\tag{202}
$$

where $sign_{thr\_HE}$, min_{snr_HE}, and min_{snr} are control parameters that differ between codec modes and sampling rates.

Table 6: Control parameters for the significance thresholds for different bandwidths

Bandwidth	$sign_{thr\_HE}$	min_{snr_HE}	$sign_{thr}$	min_{snr}
NB	2.65	0.05	1.75	0.25
WB	2.5	0.2	1.3	0.8
SWB	2.5	0.2	1.75	0.25

The signal activity is detected by comparing the two average SNR's per frame to a certain threshold the first is then used without hangover and the second has a hangover period added to prevent frequent switching at the end of an active speech period. The threshold is a function of the long-term SNR and the estimated frame to frame energy variations, mainly the variation in noise but without the need to identify noise frames. The initial estimate of the long-term SNR is given by

$$SNR_{LT} = \overline{E}_{sp}^{[-1]} - \overline{N}_t^{[-1]}$$

(203)

where $\overline{E}_{sp}$ is the long-term active signal energy, calculated in equation (234) and $\overline{N}_t$ is the long-term noise energy, calculated in equation (233). If this estimate is lower than the signal dynamics estimate $\overline{E}_{dyn}$ calculated in equation (112). Then the estimate is adjusted according to

$$SNR_{LT} = SNR_{LT} + 1$$
$$\text{if } SNR_{LT} > \overline{E}_{dyn}$$
$$SNR_{LT} = \overline{E}_{dyn}$$

(204)

The energy variation is the $\overline{E}_{vh2}$ calculated in equation (105) in clause 5.1.11.1.

The threshold calculation is calculated in three steps, one initial value and two sequential modifications. The initial value is calculated as

$$th_{SAD} = n_k + n_c\,SNR_{LT} + n_v(\overline{E}_{vh2} - n_{v\_ofs})$$

(205)

Where the function parameters, n_k, n_c, n_v, and n_{v_ofs} are set according to the current input bandwidth summarized in the following table

Table 7: Functional parameters for the initial th_{SAD} calculation for different bandwidths

Bandwidth	n_k	n_c	n_v	n_{v_ofs}
NB	0.1	16.0	4.00	1.15
WB and SWB	0.1	16.1	2.05	1.65

If the estimated SNR conditions are good, i.e. if $SNR_{LT} > 20$, the threshold is updated and upper limited for certain low-level NB signals. That is

$$th_{SAD} = th_{SAD} + 0.3(SNR_{LT} - 20)$$
$$\text{if } (BW = NB \text{ AND } SNR_{LT} > 40 \text{ AND } th_{SAD} > 24.1 \text{ AND } E_t < 45) \text{ then } th_{SAD} = 24.1$$

(206)

5.1.12.1.1 SNR outlier filtering

The average SNR per frame, SNR_{av}, that is estimated as shown in equation (202) is updated such that any sudden instantaneous SNR variations in certain sub-bands do not cause spurious deviations in the average SNR from the long term behaviour. A set of bands and SNRs per band are determined and accumulated based on noise characteristics as

shown in equations (209), (210). The critical band that contains the maximum average SNR is identified initially as the outlier band whose index is represented as, $i_{snr\_outlier}$, and the outlier band SNR is given by,

$$i_{snr\_outlier} = i\big|_{\max(SNR_{CB}(j))}, j = b_{\min}, \ldots, b_{\max} \tag{207}$$

$$SNR_{outlier} = SNR_{CB}(i_{snr\_outlier}) \tag{208}$$

The background noise energy is accumulated in bands $b_{\min}$ through $b_{\min+2}$ and in bands $b_{\min+3}$ through $b_{m\,ax}$.

$$N_{CB\_L} = \sum_{i=b_{\min}}^{b_{\min+2}} N_{CB}(i) \tag{209}$$

$$N_{CB\_H} = \sum_{i=b_{\min+3}}^{b_{\max}} N_{CB}(i) \tag{210}$$

The average SNR, SNR_{av}, is modified for WB and SWB signals through outlier filtering as follows,

if ($SNR_{outlier} < \mathrm{MAX}_{\mathrm{SNR\_OUTLIER\_3}}$ AND $i_{snr\_outlier} > 3$ AND $i_{snr\_outlier} < \mathrm{MAX}_{\mathrm{SNR\_OULIER\_IND}}$)
{

   if ($N_{CB\_L} > N_{CB\_H} * \mathrm{OUTLIER\_THR\_1}$ OR $SNR_{outlier} < \mathrm{MAX}_{\mathrm{SNR\_OUTLIER\_1}}$)
       $SNR_{av} = \mathrm{SNR}_{\mathrm{OUTLIER\_WGHT\_1}} * (SNR_{av} - SNR_{outlier})$
   else if ($N_{CB\_L} > N_{CB\_H} * \mathrm{OUTLIER\_THR\_2}$ OR $SNR_{outlier} < \mathrm{MAX}_{\mathrm{SNR\_OUTLIER\_2}}$) (211)
       $SNR_{av} = \mathrm{SNR}_{\mathrm{OUTLIER\_WGHT\_2}} * (SNR_{av} - SNR_{outlier})$
   else
       $SNR_{av} = \mathrm{SNR}_{\mathrm{OUTLIER\_WGHT\_3}} * (SNR_{av} - SNR_{outlier})$

}

The outlier filtering parameters used in updating the average SNR are listed in the table below.

Table 8: SNR outlier filtering parameters

Parameter	value
MAX_SNR_OUTLIER_1	10
MAX_SNR_OUTLIER_2	25
MAX_SNR_OUTLIER_3	50
SNR_OUTLIER_WGHT_1	1.0
SNR_OUTLIER_WGHT_2	1.01
SNR_OUTLIER_WGHT_3	1.02
OUTLIER_THR_1	10
OUTLIER_THR_2	6
Maximum outlier band index (MAX_SNR_OUTLIER_IND)	17
TH_CLEAN	35

Based on the outlier band estimated in equation (207), a weighting is determined as per equation (211) and applied to SNRs per band (through outlier filtering by subtracting the SNR in the outlier band) or on the average SNR. The threshold, th_{SAD}, is updated based on the outlier filtering and further statistics from background noise level variations, previous frame coder type, and the weighting of SNR per band. The threshold update is not performed when the long-term SNR, SNR_{LT} is below the clean speech threshold, TH_CLEAN = 35dB.

$$\text{if } (\, i_{snr\_outlier} <= 4 \text{ AND prev\_coder\_type} > \text{UNVOICED})$$

$$thr_{SAD} = thr_{SAD} - 1.0$$

$$SNR_{av} = 10 * \log 10 \big(SNR_{av\_lt} \big)$$

$$\text{else if} (\, SNR_{outlier} < \text{MAX}_{\text{SNR\_OUTLIER\_2}} \, \&\& \text{ prev\_coder\_type} < \text{UNVOICED}) \qquad (212)$$

$$thr_{SAD} = thr_{SAD} + \big(1 - 0.04 \, SNR_{outlier} \big)$$

$$\text{else}$$

$$thr_{SAD} = thr_{SAD} + \max \big(0, \, \big(0.6 - 0.01 \, SNR_{outlier} \big) \big)$$

where the smoothed average SNR, SNR_{av_lt}, is calculated after the SNR outlier filtering is performed in equation (211).

$$SNR_{av\_lt}^{[current]} = 0.5 \, SNR_{av\_lt}^{[prev]} + 0.5 \, SNR_{av} \qquad (213)$$

The updated threshold, th_{SAD}, as shown in equation (212) and the updated average SNR, SNR_{av}, as shown in equation (211) are used in signal activity detection logic as described in Clause 5.1.12.3.

5.1.12.2   SAD2 module

The SAD2 module is also a sub-band SNR based SAD and makes an activity decision for each frame by measuring the frame's modified segmental SNR. The output of SAD2 module is a binary flag f_{SAD2} which is set to 1 for active frame and set to 0 for inactive frame. For each frame, the SNR per critical band is first computed. The average energy per critical band for the whole frame and part of the previous frame is computed as

$$\overline{E}_{CB}(i) = \begin{cases} 0.2 \, E_{CB}^{[-1]}(i) + 0.4 \, E_{CB}^{[0]}(i) + 0.4 \, E_{CB}^{[1]}(i), & i = b_{\min}, \ldots, b_{\max} & \text{if } E_{CB}^{[0]}(i) > E_{CB}^{[1]}(i) \\ 0.2 \, E_{CB}^{[-1]}(i) + 0.3 \, E_{CB}^{[0]}(i) + 0.5 \, E_{CB}^{[1]}(i), & i = b_{\min}, \ldots, b_{\max} & \text{if } E_{CB}^{[0]}(i) \le E_{CB}^{[1]}(i) \end{cases} \qquad (214)$$

where $E_{CB}^{[-1]}(i)$ denotes the energy per critical band from the second spectral analysis of the previous frame, $E_{CB}^{[0]}(i)$ and $E_{CB}^{[1]}(i)$ denote respectively the energy per critical band for the first and second spectral analysis of the current frame, $b_{\min} = 0$, $b_{\max} = 19$. More weighting is given to the energy of the second spectral analysis for the current frame if the energy of the second spectral analysis is higher than the first spectral analysis. This is designed to improve the detection of signal onsets. The SNR per critical band is then computed as

$$SNR_{CB}(i) = \frac{\overline{E}_{CB}(i)}{N_{CB}(i)}, \qquad i = b_{\min}, \ldots, b_{\max}, \text{ constrained by } SNR_{CB} \ge 1 \qquad (215)$$

where $N_{CB}(i)$ is the estimated noise energy per critical band, as described in clause 5.1.11. The SNR per critical band is then converted to a logarithmic domain as

$$SNR_{CB\log}(i) = \log_{10} \big(SNR_{CB}(i) \big) \qquad (216)$$

The log SNR per critical band is then modified by

$$MSNR_{CB}(i) = \big(SNR_{CB\log}(i) + \alpha(i, SNR_{LT}) \big)^{\beta(SNR_{LT})}, \qquad i = b_{\min}, \ldots, b_{\max} \qquad (217)$$

where $MSNR_{CB}(i)$ is the modified SNR per critical band, $\alpha(i, SNR_{LT})$ is an offset value which is a function of the critical band and the long-term SNR of the input signal as calculated in equation (203), summation of $SNR_{CB\log}(i) + \alpha(i, SNR_{LT})$ is constrained to be not greater than 2, and $\beta(SNR_{LT})$ is an exponential factor used to re-shape the mapping function between $MSNR_{CB}(i)$ and $SNR_{CB}(i)$, $\beta(SNR_{LT})$ is also a function of the long-term SNR of the input signal. The offset value $\alpha(i, SNR_{LT})$ is determined as shown in the following table

Table 9: Determination of $\alpha(i, SNR_{LT})$

	$i<2$	$2\leq i<7$	$7\leq i<18$	$18\leq i$
$SNR_{LT}>24$	0	0	0	0
$18<SNR_{LT}\leq24$	0.1	0.2	0.2	0.2
$SNR_{LT}\leq18$	0.2	0.4	0.3	0.4

and $\beta(SNR_{LT})$ is determined as

$$\beta = \begin{cases} 6 & SNR_{LT}>18 \\ 9 & SNR_{LT}\leq18 \end{cases} \tag{218}$$

The modified segmental SNR is then computed as

$$MSSNR = \sum_{i=b_{\min}}^{b_{\max}} MSNR_{CB}(i) \tag{219}$$

and a relaxed modified segmental SNR is also computed. The procedure of calculating the relaxed modified segmental SNR is similar to the calculation of the modified segmental SNR with the only difference being that, besides $\alpha(i, SNR_{LT})$, another offset value $\gamma(i)$ is also added to the log SNR per critical band $SNR_{CB\log}(i)$ when calculating the relaxed modified SNR per critical band. The relaxed modified SNR per critical band is therefore computed as

$$RlxMSNR_{CB}(i) = \left(SNR_{CB\log}(i) + \alpha(i, SNR_{LT}) + \gamma(i)\right)^{\beta(SNR_{LT})}, \qquad i = b_{\min},\ldots,b_{\max} \tag{220}$$

where $\gamma(i)$ is a function of the critical band and is determined as

$$\gamma(i) = \begin{cases} 0.4 & i<7 \\ 0 & otherwise \end{cases} \tag{221}$$

The relaxed modified segmental SNR is used in the hangover process at a later stage of the algorithm.

A further enhancement (increase in value) is made to the modified segmental SNR if an unvoiced signal is detected. Unvoiced signal is detected if both of the two critical bands covering the highest frequency range have SNRs greater than a threshold of 5, i.e. if $SNR_{CB}(18)>5$ and $SNR_{CB}(19)>5$. In this case, the contributions of the overall modified segmental SNR from the two critical bands is boosted. The boost is performed over the two critical bands where the number of critical bands is extended from two to eight and the corresponding modified segmental SNR is re-computed over the extended bands as

$$MSSNR = \left(MSSNR + 3\cdot MSNR_{CB}(18) + 3\cdot MSNR_{CB}(19)\right)\cdot\frac{20}{26} \tag{222}$$

where multiplication by 20/26 effectively performs the mapping of the modified segmental SNR calculated on the extended scale back onto the same scale as if it were computed over the original 20 critical bands. The re-computation of modified segmental SNR is only conducted if the computed value is greater than before. If no unvoiced signal of above type is detected, N_{sigCB}, which is the number of critical bands whose SNR is greater than a threshold of 2 is determined. If $N_{sigCB}>13$, a second type unvoiced signal is detected, and if the long-term SNR of the input signal SNR_{LT} is further below a threshold of 24, the modified segmental SNR in this case is re-computed as

$$MSSNR = MSSNR + \Delta \tag{223}$$

where $\Delta = 2.5\cdot SNR_{LT} - 15.5$ and is limited to be a positive value.

The primary signal activity decision is made in SAD2 by comparing the modified segmental SNR to a decision threshold THR_{SAD2}. The decision threshold is a piece wise linear function of the long-term SNR of the input signal and is determined as

$$THR_{SAD2} = \begin{cases} MIN[2.4 \cdot SNR_{LT} - 42.2, 80] & SNR_{LT} > 24 \\ MIN[2.4 \cdot SNR_{LT} - 40.2, 80] & 18 < SNR_{LT} \le 24 \\ MAX[2.5 \cdot SNR_{LT} - 10, 1] & SNR_{LT} \le 18 \end{cases} \qquad (224)$$

If the modified segmental SNR is greater than the decision threshold, the activity flag f_{SAD2} is set to 1, and a counter of consecutive active frames, C_{act}, used by SAD2 is incremented by 1, and if the current frame is ineither a soft or a hard hangover period as described later in this subclause, the corresponding hangover period elapses by 1. Otherwise, the consecutive active frames counter C_{act} is set to 0 and the setting of f_{SAD2} is further evaluated by a hangover process.

The hangover scheme used by SAD2 consists of a soft hangover process followed by a hard hangover process. The soft hangover is designed to prevent low level voiced signals during a speech offset from being cut. When within the soft hangover period, the SAD2 is operating in an offset working state where the relaxed modified segmental SNR calculated earlier is used to compare to the decision threshold THR_{SAD2} (compared to the normal working state where the modified segmental SNR is used). If the relaxed modified segmental SNR is greater than the decision threshold, the activity flag f_{SAD2} is set to 1 and the soft hangover period elapses by 1. Otherwise, if the relaxed modified segmental SNR is not greater than the decision threshold, the soft hangover period is quit and the setting of f_{SAD2} is finally evaluated by a hard hangover process. When within the hard hangover period, the activity flag f_{SAD2} is forced to 1 and the hard hangover period elapses by 1. The soft hangover period is initialized if the number of consecutive voiced frames exceeds 3. The frame is considered a voiced frame if the pitch correlation is not low and the pitch stays relatively stable, that is, if $(C_{norm}^{[0]} + C_{norm}^{[1]} + C_{norm}^{[2]})/3 + r_e > 0.65$ and $\left((d_{OL}^{[0]} - d_{OL}^{[-1]}) + (d_{OL}^{[1]} - d_{OL}^{[0]}) + (d_{OL}^{[2]} - d_{OL}^{[1]})\right)/3 < 14$ where $C_{norm}^{[0]}, C_{norm}^{[1]}, C_{norm}^{[2]}$ are respectively the normalized pitch correlation for the second half of the previous frame, the first half of the current frame and the second half of the current frame as calculated, r_e is the noise correction factor, $d_{OL}^{[-1]}, d_{OL}^{[0]}, d_{OL}^{[1]}, d_{OL}^{[2]}$ are respectively the OL pitch lag for the second half of the previous frame, the first half of the current frame, the second half of the current frame and the look-ahead as described in subclause 5.1.10. The value to which the soft hangover period is initialized is a function of the long-term SNR of the input signal and the noise fluctuation FLU_N computed in equation (225), and is determined as

Table 10: Determination of the soft hangover period initialization length

	$SNR_{LT} > 24$	$18 < SNR_{LT} \le 24$	$SNR_{LT} \le 18$
$FLU_N < 40$	1	1	2
$FLU_N \ge 40$	1	3	4

The hard hangover period is initialized if the consecutive active frames counter C_{act} reaches a threshold of 3. The value to which the hard hangover period is initialized is also a function of the long-term SNR of the input signal and the noise fluctuation, and is determined as

Table 11: Determination of the hard hangover period initialization length

	$SNR_{LT} > 24$	$18 < SNR_{LT} \le 24$	$SNR_{LT} \le 18$
$FLU_N < 40$	1	1	2
$FLU_N \ge 40$	1	1	3

The noise fluctuation FLU_N is estimated over background frames declared as inactive by the final SAD flag of SAD2, f_{SAD2}, by measuring the moving average of the segmental SNR in the logarithm domain. The noise fluctuation is computed as

$$FLU_N = FLU_N^{[-1]} - (1-\alpha) \cdot \delta$$

$$\delta = FLU_N^{[-1]} - \sum_{i=b_{\min}}^{b_{\max}} SNR_{CB\log}(i) \tag{225}$$

where $FLU_N^{[-1]}$ denotes the noise fluctuation of the previous frame, α is the forgetting factor controlling the update rate of the moving average filter and is set to 0.99 for an increasing update (when $FLU_N > FLU_N^{[-1]}$) and 0.9992 for a decreasing update (when $FLU_N \le FLU_N^{[-1]}$). δ is constrained by $\delta \le 10$ for decreasing updates and $\delta \ge -10$ for increasing updates. To speed up the initialization of noise fluctuation, for the first 50 background frames, α is set to 0.9 for increasing updates and 0.95 for decreasing updates, δ is constrained by $\delta \le 30$ for decreasing updates and $\delta \ge -50$ for increasing updates.

5.1.12.3 Combined decision of SAD1 and SAD2 modules for WB and SWB signals

The decision of the SAD1 module is modified by the decision of the SAD2 module for WB and SWB signals.

For f_{LSAD_HE} the decision logic is direct if the average SNR per frame is larger than the SAD decision threshold and if the final SAD2 flag is set to 1. That is,

$$\text{if } SNR_{av\_HE} > th_{SAD} \text{ AND } f_{SAD2} = 1 \quad \text{then } f_{LSAD\_HE} = 1, \text{otherwise } f_{LSAD\_HE} = 0 \tag{226}$$

Likewise, for f_{LSAD} the decision logic is direct if the average SNR per frame is larger than the SAD decision threshold and if the final SAD2 flag is set to 1.That is,

$$\text{if } SNR_{av} > th_{SAD} \text{ AND } f_{SAD2} = 1 \quad \text{then } f_{LSAD} = 1, f_{SAD} = 1 \tag{227}$$

otherwise, f_{LSAD} is set to 0 and the hangover logic decides if f_{SAD} should be set to active or not.

The hangover logic works as a state machine that keeps track of the number of frames since the last active primary decision and if a sufficient number of consecutive active frames have occurred in a row to allow the final decision to remain active even if the primary decision has already gone inactive. Thus, if there has not been a sufficient number of primary decisions in a row there is no hangover addition and the final decision is set to inactive, that is f_{SAD} is set to 0 if f_{LSAD} is 0.

The hangover length depends on SNR_{LT}, initially set to 0 frames and if $15 \le SNR_{LT} < 35$ it is set to 4 frames and if $SNR_{LT} < 15$ it is set to 3 frames. The counting of hangover frames is reset only if at least 3 consecutive active speech frames ($SNR_{av} > th_{SAD}$) were present, meaning that no hangover is used if $SNR_{av} > th_{SAD}$ in only one or two adjacent frames. This is to avoid adding the hangover after short energy bursts in the acoustic signal, increasing the average data rate in the DTX operation.

5.1.12.4 Final decision of the SAD1 module for NB signals

Similarly to the WB case two primary decisions are generated but in this case there is no dependency on the SAD2 module. If $SNR_{av\_HE} > th_{SAD}$ the frame is declared as active and the primary SAD flag, f_{LSAD_HE} is set to 1. Otherwise, f_{LSAD_HE} is set to 0.

To get the final SAD decision f_{SAD} there is a difference in how the primary decision is made and how the hangover is handled compared to WB. For NB signals the hangover has a window of 8 frames after the last run of three consecutive active primary decision, that is $SNR_{av} > th_{SAD}$. During this hangover period the SAD decision is not automatically set to active; instead, the threshold th_{SAD} is decreased by 5.2 if $SNR_{LT} < 19$, and by 2 if $19 \le SNR_{LT} < 35$. The SAD decision is then made by comparing the average SNR to the corrected threshold following condition (206). Again, counting of hangover frames is reset only if at least 3 consecutive active frames were present.

5.1.12.5 Post-decision parameter update

After the final decision is formed, some SAD1-related long term-parameters are updated according to the primary and final decisions.

$$\begin{aligned} prim_{act\_quick} &= 0.90\, prim_{act\_quick} + 0.10\, f_{LSAD} \\ prim_{act\_slow} &= 0.99\, prim_{act\_slow} + 0.01\, f_{LSAD} \end{aligned} \tag{228}$$

These are then used to form a measure of the long term primary activity of f_{LSAD}

$$prim_{act} = 0.90\, prim_{act} + 0.10 \min(prim_{act\_quic}, prim_{act\_slow}) \tag{229}$$

Similar metrics are generated for the f_{LSAD_HE}

$$\begin{aligned} prim_{act\_quick\_HE} &= 0.90\, prim_{act\_quick\_HE} + 0.10\, f_{LSAD\_HE} \\ prim_{act\_slow\_HE} &= 0.99\, prim_{act\_slow\_HE} + 0.01\, f_{LSAD\_HE} \end{aligned} \tag{230}$$

These are then used to form an measure of the long term primary activity of f_{LSAD}

$$prim_{act\_HE} = 0.90\, prim_{act\_HE} + 0.10 \min(prim_{act\_quic\_HE}, prim_{act\_slow\_HE}) \tag{231}$$

To keep track of the history f_{SAD} decisions the registers $f_{SAD\_reg\_h}, f_{SAD\_reg\_l}, f_{SAD\_cnt}$ are updated so that $f_{SAD\_reg\_h}$ and $f_{SAD\_reg\_l}$ keeps track of the latest 50 frames with regard to the f_{SAD} decisions by removing the oldest decision and adding the latest and updating f_{SAD_cnt} so that it reflects the current number of active frames in the registers.

Similarly to keep track of the history for f_{LSAD} decisions the registers f_{LSAD_reg} and f_{LSAD_cnt} are updated so that f_{LSAD_reg} keeps track of the latest 16 frames with regard to the f_{LSAD} decisions by removing the oldest decision and adding the latest and updating f_{LSAD_cnt} so that it reflects the current number of active frames in the registers.

When the SAD decisions have been made, the speech music classifier decision has been made and the noise estimation for the current frame has been completed the long term estimates of active speech level, $\overline{E}_{sp}$, and long term noise level estimate $\overline{N}_t$ can be updated. During the four first frames the initialization is made for both variables using mainly the current input as follows

$$\begin{aligned} \overline{N}_t &= N_t \\ \text{if } \overline{E}_{sp}^{[-1]} &< N_t + 10 \text{ then } \overline{E}_{sp}^{[0]} = N_t + 10 \end{aligned} \tag{232}$$

Where N_t is the total sub band noise level after update for the current frame. For the long term noise level estimate the initialization uses different filter coefficient during the remainder of the 150 frames initialization as follows

$$\overline{N}_t = (1-\alpha)\overline{N}_t^{[-1]} + \alpha N_t \quad \text{where } \alpha = \begin{cases} 0.02 & \text{if } ini_{frame} < 150 \\ 0.05 & \text{otherwise} \end{cases} \tag{233}$$

For the active speech level the update after the initial four frames only occurs if f_{LSAD_HE} is 1 AND the f_{high_lpn} from the speech music classifier is 0. Where f_{high_lpn} is generated as $L_n > L_s$ AND $L_n > L_m$ based on the features L_s, L_m, L_n calculated based on equation (329) in subclause 5.1.13.6.3. If those conditions are met then the speech level estimate is updated according to

$$\overline{E}_{sp} = \begin{cases} 0.98\overline{E}_{sp}^{[-1]} + 0.02 E_t & \text{if } (\overline{E}_{sp}^{[-1]} - E_t) < 10 \\ \overline{E}_{sp}^{[-1]} - 0.05 & \text{otherwise} \end{cases} \tag{234}$$

5.1.12.6   SAD3 module

The SAD3 module is shown in Figure 12. The processing steps are described as follows:

a) Extract features of the signal according to the input QMF data. The QMF data is from CLDFB.

b) Calculate some SNR parameters according to the extracted features of the signal and make a decision of background music.

c) Make a pre-decision of SAD3 according to the features of the signal, the SNR parameters, and the output flag of the decision of background music and then output a pre-decision flag.

d) The output of SAD3 is generated through the addition of SAD3 hangover.
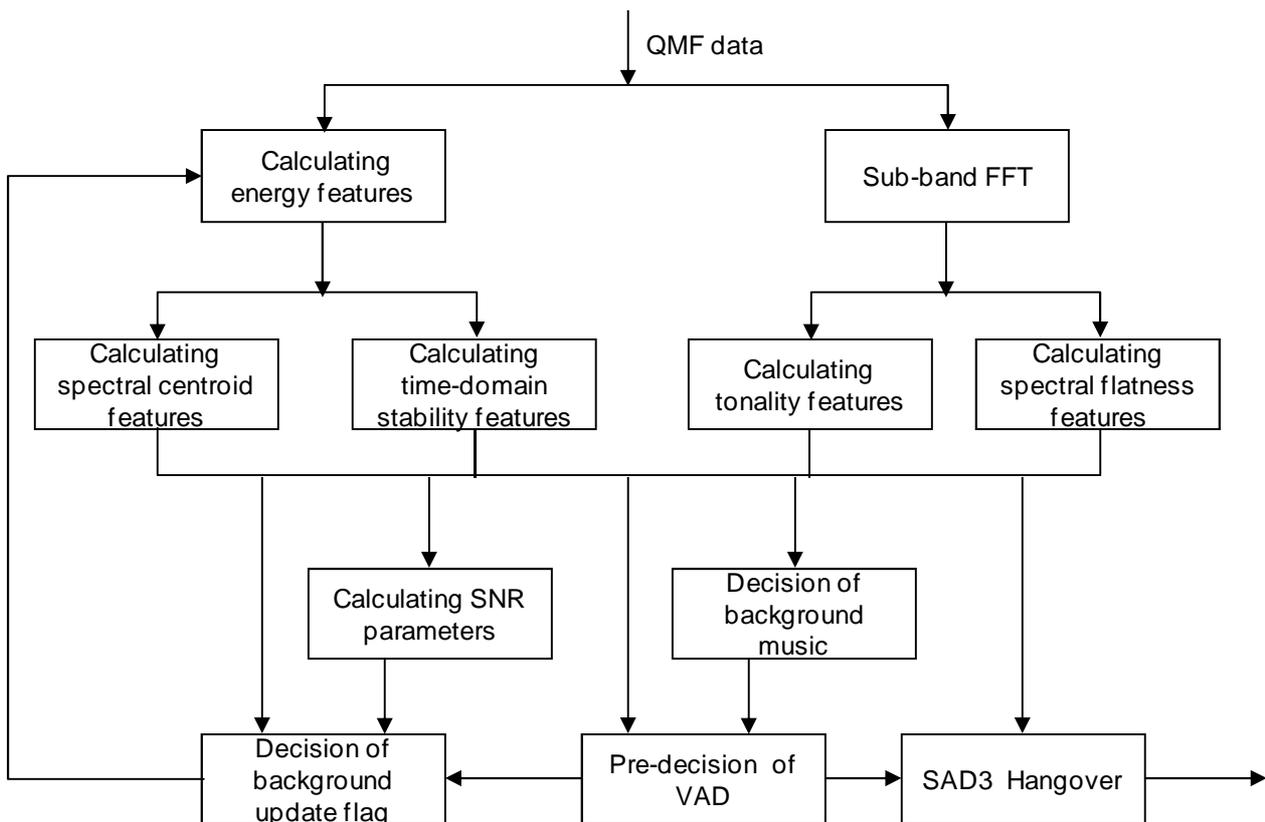
Figure 12: Block diagram of SAD3

5.1.12.6.1   Sub-band FFT

Sub-band FFT is used to obtain spectrum amplitude of signal. Let $X[k, l]$ denote the output of QMF filter applied to the lth sample in the kth sub-band. $X[k,l]$ is converted into a frequency domain representation from the time domain by FFT as follows:

$$X_{DFT}[k, j] = \sum_{l=0}^{15} X[k,l]e^{-\frac{2\pi j}{16}l} ; 0 \le k < 10, 0 \le j < 16 \qquad (235)$$

The spectrum amplitude of each sample is computed in the following steps:

Step 1: Compute the energy of $X_{DFT}[k, j]$ as follows:

$$X_{DFT\_POW}[k,j] = ((\mathrm{Re}(X_{DFT}[k,j]))^2 + (\mathrm{Im}(X_{DFT}[k,j]))^2); 0 \le k < 10, 0 \le j < 16 \tag{236}$$

where $\mathrm{Re}(X_{DFT}[k,j])$, $\mathrm{Im}(X_{DFT}[k,j])$ are the real part and imaginary part of $X_{DFT}[k,j]$, respectively.

Step 2: If k is even, the spectrum amplitude, denoted by A_{sp}, is computed by

$$A_{sp}(8k+j) = \sqrt{X_{DFT\_POW}[k,j] + X_{DFT\_POW}[k,15-j]}, \quad 0 \le k < 10, 0 \le j < 8 \tag{237}$$

If k is odd, the spectrum amplitude is computed by

$$A_{sp}(8k+7-j) = \sqrt{X_{DFT\_POW}[k,j] + X_{DFT\_POW}[k,15-j]}, \quad 0 \le k < 10, 0 \le j < 8 \tag{238}$$

5.1.12.6.2 Computation of signal features

In Pre-decision Energy Features (EF), Spectral Centroid Features (SCF), and Time-domain Stability Features (TSF) of the current frame are computed by using the sub-band signal; Spectral Flatness Features (SFF) and Tonality Features (TF) are computed by using the spectral amplitude.

5.1.12.6.2.1 Computation of EF

The energy features of the current frame are computed by using the sub-band signal. The energy of background noise of the current frame, including both the energy of background noise over individual sub-band and the energy of background noise over all sub-bands, is estimated with the updated flag of background noise, the energy features of the current frame, and the energy of background noise over all sub-bands of the previous frame. The energy of background noise of the current frame will be used to compute the SNR parameters of the next frame (see subclause 5.1.12.6.3). The energy features include the energy parameters of the current frame and the energy of background noise. The energy parameters of the frame are the weighted or non-weighted sum of energies of all sub-bands.

The frame energy is computed by:

$$E_f = \sum_{k=1}^{(L_C-2)} \overline{E}_C(k) + 0.24\overline{E}_C(0) \tag{239}$$

The energy of sub-band divided non-uniformly is computed by:

$$E_{f\_snr}(i) = \sum_{j=N_{region}(i)}^{N_{region}(i+1)-1} \overline{E}_C(j), \quad 0 \le i < N_{snr} \tag{240}$$

Where N_{region} is the sub-band division indices of $E_{f\_snr}(i)$. The sub-bands based on this kind of division are also called SNR sub-bands and are used to compute the SNR of sub-band. N_{snr} is the number of SNR sub-bands.

The energy of sub-band background noise of the current frame is computed by:

$$E_{bg\_snr}^{[0]}(i) = \alpha E_{bg\_snr}^{[-1]}(i) + (1-\alpha)E_{f\_snr}^{[0]}, \quad 0 < \alpha < 1, 0 \le i < N_{snr} \tag{241}$$

Where $E_{bg\_snr}^{[-1]}(i)$ is the energy of sub-band background noise of the previous frame.

The energy of background noise over all sub-bands is computed according to the background update flag, the energy features of the current frame and the tonality signal flag:

$$E_{f\_bg}^{[0]} = \frac{E_{f\_bg\_sum}^{[0]}}{N_{f\_bg}^{[0]}} \tag{242}$$

If certain conditions that include at least that the background update flag is 1 and the tonality signal flag f_{tonal_signal} is 0 are met, $E_{f\_bg\_sum}^{[0]}$ and $N_{f\_bg}^{[0]}$ are computed by:

$$E_{f\_bg\_sum}^{[0]} = E_{f\_bg\_sum}^{[-1]} + E_f^{[0]} \qquad (243)$$

$$N_{f\_bg}^{[0]} = N_{f\_bg}^{[-1]} + 1 \qquad (244)$$

Otherwise, $E_{f\_bg\_sum}^{[0]}$ and $N_{f\_bg}^{[0]}$ are computed by:

$$E_{f\_bg\_sum}^{[0]} = E_{f\_bg\_sum}^{[-1]} \qquad (245)$$

$$N_{f\_bg}^{[0]} = N_{f\_bg}^{[-1]} \qquad (246)$$

Where $E_{f\_bg\_sum}^{[i]}$ and $N_{f\_bg}^{[i]}$ are the sum of E_f and the counter of E_f, respectively. The superscript [-1] denotes the previsous frame and [0] denotes the current frame.

5.1.12.6.2.2 Computation of SCF

The spectral centroid features are the ratio of the weighted sum to the non-weighted sum of energies of all sub-bands or partial sub-bands, or the value is obtained by applying a smooth filter to this ratio. The spectral centroid features can be obtained in the following steps:

a) Divide the sub-bands for computing the spectral centroids as shown in Table 12.

Table 12: QMF sub-band division for computing spectral centroids

Spectral centroid feature number (i)	$N_{sc\_start}(i)$	$N_{sc\_end}(i)$
2	0	9
3	1	23

b) Compute two spectral centroid features, i.e.: the spectral centroid in the first interval and the spectral centroid in the second interval, by using the sub-band division for computing spectral centroids in Step a) and the following equation:

$$F_{SC}(i) = (\sum_{k=N_{sc\_start}(i)}^{N_{sc\_end}(i)} (k+1)\overline{E}_C(k) + \Delta_1) / \sum_{k=N_{sc\_start}(i)}^{N_{sc\_end}(i)} \overline{E}_C(k) + \Delta_2), \ 1 < i < 4 \qquad (247)$$

c) Smooth the spectral centroid in the second interval, $F_{SC}(2)$, to obtain the smoothed spectral centroid in the second interval by

$$F_{SC}^{[0]}(0) = 0.7 F_{SC}^{[-1]}(0) + 0.3 F_{SC}^{[0]}(2) \qquad (248)$$

5.1.12.6.2.3 Computation of SFF

Spectral Flatness Features are the ratio of the geometric mean to the arithmetic mean of certain spectrum amplitude, or this ratio multiplied by a factor. The spectrum amplitude A_{sp}, is smoothed as follows:

$$A_{ssp}^{[0]}(i) = 0.7 A_{ssp}^{[-1]}(i) + 0.3 A_{sp}^{[0]}(i), \ 0 \le i < N_A \qquad (249)$$

where $A_{ssp}^{[0]}(i)$ and $A_{ssp}^{[-1]}(i)$ are the smoothed spectrum amplitude of the current frame and the previous frame, respectively. N_A is the number of spectrum amplitude.

Then the the smoothed spectrum amplitude is divided for three frequency regions as shown in Table 13 and the spectral flatness features are computed for these frequency regions.

Table 13: Sub-band division for computing spectral flatness

Spectral flatness number (k)	$N_{A\_start}(k)$	$N_{A\_end}(k)$
0	5	19
1	20	39
2	40	64

The spectral flatness features are the ratio of the geometric mean to the arithmetic mean of the spectrum amplitude or the smoothed spectrum amplitude.

Let $N(k) = N_{A\_end}(k) - N_{A\_start}(k)$ be the number of the spectrum amplitudes used to compute the spectral flatness feature $F_{SF}(k)$. We have

$$F_{SF}(k) = \frac{(\prod_{n=N_{A\_start}(k)}^{N_{A\_end}(k)} A_{ssp}(n))^{1/N(k)}}{\sum_{n=N_{A\_start}(k)}^{N_{A\_end}(k)} A_{ssp}(n)) / N(k)}$$
(250)

The spectral flatness features of the current frame are further smoothed as follows:

$$F_{SSF}^{[0]}(k) = 0.85\, F_{SSF}^{[-1]}(k) + 0.15 F_{SF}^{[0]}(k)$$
(251)

Where $F_{SSF}^{[0]}(k)$ and $F_{SSF}^{[-1]}(k)$ are the smoothed spectral flatness features of the current frame and the previous frame respectively.

5.1.12.6.2.4   Computation of TSF

The time-domain stability features are the ratio of the variance of the sum of amplitudes to the expectation of the square of amplitudes, or this ratio multiplied by a factor. The time-domain stability features are computed with the energy features of the most recent N frame. Let the energy of the nth frame be $E_f^{[n]}$. The amplitude of $E_f^{[n]}$ is computed by

$$A_{t1}^{[n]} = \sqrt{E_f^{[n]}} + 0.001$$
(252)

By adding together the energy amplitudes of two adjacent frames from the current frame to the Nth previous frame, the sum of N/2 energy amplitudes is obtained as

$$A_{t2}^{[n]} = A_{t1}^{[-2n]} + A_{t1}^{[-2n-1]}$$
(253)

Where $A_{t1}^{[k]}$ is the energy amplitude of the current frame for $k = 0$ and $A_{t1}^{[k]}$ the energy amplitude of the previous frames for $k < 0$.

Then the ratio of the variance to the average energy of the N/2 recent sums is computed and the time-domain stability F_{TS} is obtained as follows:

$$F_{TS} = \sum_{n=0}^{N/2-1} (A_{t2}^{[n]} - \frac{1}{N/2} \sum_{n=0}^{N/2-1} A_{t2}^{[n]})^2 / (\sum_{n=0}^{N/2-1} (A_{t2}^{[n]})^2 + 0.0001)$$
(254)

Note that the value of N is different when computing different time-domain stabilities.

5.1.12.6.2.5 Computation of TF

The tonality features are computed with the spectrum amplitudes. More specifically, they are obtained by computing the correlation coefficient of the amplitude difference of two adjacent frames, or with a further smoothing the correlation coefficient, in the following steps:

a) Compute the amplitudes difference of two adjacent frames. If the difference is smaller than 0, set it to 0.

$$D_{sp}(i) = \begin{cases} 0 & if \ A_{sp}(i+6) < A_{sp}(i+5) \\ A_{sp}(i+6) - A_{sp}(i+5) & otherwise \end{cases}$$ (255)

b) Compute the correlation coefficient between the non-negative amplitude difference of the current frame obtained in Step a) and the non-negative amplitude difference of the previous frame to obtain the first tonality features as follows:

$$F_{TR} = \frac{\sum_{i=0}^{N} D_{sp}^{[0]}(i) \ D_{sp}^{[-1]}(i)}{\sqrt{\sum_{i=0}^{N} (D_{sp}^{[0]}(i))^2 \ (D_{sp}^{[-1]}(i))^2}}$$ (256)

where $D_{sp}^{[-1]}(i)$ is the amplitude differece of the previous frame.

Various tonality features can be computed as follows:

$$F_T(0) = F_{TR}$$
$$F_T^{[0]}(1) = 0.96 F_T^{[-1]}(1) + 0.04 F_{TR}$$ (257)
$$F_T^{[0]}(2) = 0.90 F_T^{[-1]}(2) + 0.10 F_{TR}$$

where $F_T^{[-1]}$ are tonality features of the previous frame.

5.1.12.6.3 Computation of SNR parameters

The SNR parameters of the current frame are computed with the background energy estimated from the previous frame, the energy parameters and the energy of the SNR sub-bands of the current frame.

The total SNR of all sub-band is computed by:

$$SNR_t = \log_2((E_f^{[0]} + 0.0001)/(E_{f\_bg}^{[-1]}))$$ (258)

The average SNR of all sub-bands is computed by:

$$SNR_{flux} = (\sum_{i=0}^{N-1} SNR_t(i))/N$$ (259)

where N is number of the most recent frames and $SNR_t(i)$ is SNR_t of the ith frame.

The frequency-domain SNR is computed by:

$$SNR_f = (\sum_{i=0}^{N_{snr}-1} SNR_{sub}(i))/N_{snr}$$ (260)

where N_{snr} is the number of SNR sub-band and $SNR_{sub}(i)$ it the SNR of each sub-band by:

$$T(i) = \log_2((E_{f\_snr}^{[0]}(i) + 0.0001)/(E_{bg\_snr}^{[-1]}(i) + 0.0001))$$
$$SNR_{sub}(i) = \begin{cases} 0 & if \ T(i) < -0.1 \\ T(i) & otherwise \end{cases}$$ (261)

The first long-time SNR is computed by:

$$SNR_{lt\_org} = \log_{10}(E^{[-1]}_{lt\_active} / E^{[-1]}_{lt\_inactive}) \tag{262}$$

The computation method of $E^{[-1]}_{lt\_active}$ and $E^{[-1]}_{lt\_inactive}$ can be found in subclause 5.1.12.6.6.

The second long-time SNR is obtained by accordingly adjusting SNR_{lt_org} as follows:

$$SNR_{lt} = T_{lt\_snr} + 0.4(0.4T_{lt\_snr} + 0.1)F_{tnSC0} \tag{263}$$

where:

$$F_{tnSC0} = \begin{cases} 0 & F^{[-1]}_{nSC0} < 1.4 \\ F^{[-1]}_{nSC0} - 1.4 & 1.4 \leq F^{[-1]}_{nSC0} \leq 2.2 \\ 0.8 & F^{[-1]}_{nSC0} > 2.2 \end{cases} \tag{264}$$

where F_{nSC0} is the long-time background spectral centroid. If the current frame is active frame and the background-update flag is 1, the long-time background spectral centroid of the current frame is updated as follows:

$$F^{[0]}_{nSC0} = \alpha F^{[-1]}_{nSC0} + (1-\alpha)F^{[0]}_{SC}(0), \quad \alpha \in (0,1) \tag{265}$$

where $F^{[-1]}_{nSC0}$ is the long-time background spectral centroid of the previous frame.

The initial long-time frequency-domain SNR of the current frame $SNR^{[0]}_l$ is computed by:

$$SNR^{[0]}_l = S^{[-1]}_{sp\_snr\_f} / N^{[-1]}_{sp\_snr\_f} - S^{[-1]}_{ns\_snr\_f} / N^{[-1]}_{ns\_snr\_f} \tag{266}$$

where $S_{sp\_snr\_f}$ and $N_{sp\_snr\_f}$ are respectively the frequency-domain SNR SNR_f accumulator and frequency-domain SNR SNR_f counter when the current frame is pre-decided as active sound, and $S_{ns\_snr\_f}$ and $N_{ns\_snr\_f}$ are respectively SNR_f accumulator and SNR_f counter when the current frame is pre-decided as inactive sound. The superscript [-1] denotes the previsous frame. The details of computation can be found in Steps e) and i) of subclause 5.1.12.6.6.

The smoothed average long-time SNR is computed by:

$$SNR^{[0]}_{lf\_smooth} = 0.9SNR^{[-1]}_{lf\_smooth} + 0.1SNR^{[0]}_l \tag{267}$$

The long-time frequency-domain SNR is computed by:

$$SNR_{lf} = \begin{cases} 0 & T_{lf\_snr} < 0 \\ T_{lf\_snr} & 0 \leq T_{lf\_snr} \leq \text{MAX\_LF\_SNR} \\ \text{MAX\_LF\_SNR} & T_{lf\_snr} > \text{MAX\_LF\_SNR} \end{cases} \tag{268}$$

$$T_{lf\_snr} = 0.12(SNR^{[0]}_l - 3)$$

where MAX_LF_SNR is the maximum of SNR_{lf} .

5.1.12.6.4 Decision of background music

With the energy features, F_T , F_{TS} , F_{SSF} , and F_{SC} of the current frame, the tonality signal flag of the current frame is computed and used to determine whether the current frame is tonal signal. If it is tonal signal, the current frame is music and the following procedure is carried out:

a) Suppose the current frame is non-tonal signal and a flag f_{tonal_frame} is used to indicate whether the current frame is tonal. If $f_{tonal\_frame} = 1$, the current frame is tonal. If $f_{tonal\_frame} = 0$, the current frame is non-tonal.

b) If $F_T(0) > 0.6$ or its smoothed value $F_T(1)$ is greater than 0.86., go to Step c). Otherwise, go to Step d).

c) Verify the following three conditions:

(1) The time-domain stability feature $F_{TS}(5)$ is smaller than 0.072;

(2) The spectral centroid feature $F_{SC}(0)$ is greater than 1.2;

(3) One of three spectral flatness features is smaller than its threshold,
$F_{SSF}(0) < 0.76$ OR $F_{SSF}(1) < 0.88$ OR $F_{SSF}(2) < 0.96$.

If all the above conditions are met, the current frame is considered as a tonal frame and the flag f_{tonal_frame} is set to 1. Then go to Step d).

d) Update the tonal level feature l_{tonal} according to the flag f_{tonal_frame}. The initial value of l_{tonal} is set in the region [0, 1] when the active-sound detector begins to work.

$$l_{tonal}^{[0]} = \begin{cases} 0.975 l_{tonal}^{[-1]} + 0.025 & \text{if } f_{tonal\_frame} \text{ is } 1 \\ 0.997 l_{tonal}^{[-1]} & \text{otherwise} \end{cases} \tag{269}$$

Where $l_{tonal}^{[0]}$ and $l_{tonal}^{[-1]}$ are respectively the tonal level of the current frame and the previous frame.

e) Determine whether the current frame is tonal according to the updated $l_{tonal}^{[0]}$ and set the tonality signal flag f_{tonal_signal}.

If $l_{tonal}^{[0]}$ is greater than 0.5, the current frame is determined as tonal signal. Otherwise, the current frame is determined as non-tonal signal.

$$f_{tonal\_signal} = \begin{cases} 1 & \text{if } l_{tonal}^{[0]} > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{270}$$

5.1.12.6.5　Decision of background update flag

The background update flag is used to indicate whether the energy of background noise is updated and its value is 1 or 0. When this flag is 1, the energy of background noise is updated. Otherwise, it is not updated.

The initial background update flag of the current frame is computed by using the energy features, the spectral centroid features, the time-domain stability features, the spectral flatness features, and the tonality features of the current frame. The initial background update flag is updated with the VAD decision, the tonality features, the SNR parameters, the tonality signal flag, and the time-domain stability features of the current frame to obtain the final background update flag. With the obtained background update flag, background noise is detected.

First, suppose the current frame is background noise. If any one of the following conditions is met, the current frame is not noise signal.

a) The time-domain stability $F_{TS}(5) > 0.12$;

b) The spectral centroid $F_{SC}(0) > 4.0$ and the time-domain stability $F_{TS}(5) > 0.04$;

c) The tonality feature $F_T(1) > 0.5$ and the time-domain stability $F_{TS}(5) > 0.1$;

d) The spectral flatness of each sub-band or the average obtained by smoothing the spectral flatness is smaller than its specified threshold, $F_{SSF}(0) < 0.8$ OR $F_{SSF}(1) < 0.78$ OR $F_{SSF}(2) < 0.8$;

e) The energy of the current frame E_f is greater than a specified threshold: $E_f^{[0]} > 32 E_{sf}^{[-1]}$, where $E_{sf}^{[-1]}$ is the long time smoothed energy of the previous frame and $E_{sf}^{[k]}$ of kth frame is computed : $E_{sf}^{[k]} = 0.95 E_{sf}^{[k-1]} + 0.05 E_f^{[k]}$;

f) The tonality features F_T are greater than their corresponding thresholds: $F_T(1) > 0.60$ **OR** $F_T(0) > 0.86$;

g) The initial background update flag can be obtained in Steps a) - f). The initial background update flag is then updated. When the SNR parameters, the tonality features, and the time-domain stability features are smaller than their corresponding thresholds : $SNR_f < 0.3$ **AND** $SNR_t < 1.2$ **AND** $F_T(1) < 0.5$ **AND** $F_{TS}(3) < 0.1$ and both the combined f_{SAD} and f_{tonal_signal} are set to 0, the background update flag is updated to 1.

5.1.12.6.6 SAD3 Pre-decision

The SAD3 decision f_{SAD3} is computed with the tonality signal flag, the SNR parameters, the spectral centroid features, and the frame energy. Then, the SAD3 decision is further combined with f_{SAD} which is from SAD1 and SAD2 decision to make a final decision of SAD. The final decision of SAD is made in the following steps:

a) Obtain the second long-time SNR SNR_{lt} by computing the ratio of the average energy of long-time active frames to the average energy of long-time background noise for the previous frame;

b) Compute the average of SNR_t for a number of recent frames to obtain SNR_{flux} ;

c) Compute the SNR threshold for making SAD3 decision, denoted by T_{snr_f} , with the spectral centroid features F_{SC} , the second long-time SNR SNR_{lt} , the long-time frequency-domain SNR SNR_{lf} , the number of continuous active frames $N_{continuous\_sp1}$, and the number of previous continuous noise frames $N_{continuous\_ns}$. Set the initial value of T_{snr_f} to SNR_{lt} . First, adjust T_{snr_f} with the spectral centroid features, if the spectral centroids are located in the different regions, an appropriate offset may be added to T_{snr_f} . Then, T_{snr_f} is further adjusted according to $N_{continuous\_sp1}$, $N_{continuous\_ns}$ and SNR_{flux} , and SNR_{lf} . When $N_{continuous\_sp1}$ is greater than its threshold, the SNR threshold is appropriately decreased. When $N_{continuous\_ns}$ is greater than its threshold, the SNR threshold is appropriately increased. If SNR_{lt} is greater than a specified threshold, the SNR threshold may be accordingly adjusted.

d) Make an initial VAD decision with the SAD3 decision threshold T_{snr_f} and the SNR parameters such as SNR_f and SNR_t of the current frame. First f_{SAD3} is set to 0. If $SNR_f > T_{snr\_f}$, or $SNR_t > 4.0$, f_{SAD3} is set to 1. The initial VAD decision can be used to compute the average energy of long-time active frames E_{lt_active} . The value of E_{lt_active} is used to make SAD3 decision for the next frame.

$$E_{lt\_active}^{[0]} = E_{fg\_sum}^{[0]} / N_{fg}^{[0]} \tag{271}$$

where $E_{fg\_sum}^{[0]}$ and $N_{fg}^{[0]}$ is computed by:

$$E_{fg\_sum}^{[0]} = \begin{cases} E_{fg\_sum}^{[-1]} + E_f^{[0]} & if \ f_{SAD3} = 1 \\ E_{fg\_sum}^{[-1]} & otherwise \end{cases} \tag{272}$$

$$N_{fg}^{[0]} = \begin{cases} N_{fg}^{[-1]} + 1 & if \ f_{SAD3} = 1 \\ N_{fg}^{[-1]} & otherwise \end{cases} \tag{273}$$

e) Update the initial SAD3 decision according to the tonality signal flag, the average SNR of all sub-bands, the spectral centroids, and the second long-time SNR. If the tonality signal flag f_{tonal_signal} is 1, f_{SAD3} is set to 1. The parameters $S_{sp\_snr\_f}$ and $N_{ns\_snr\_f}$ are updated by:

$$S^{[0]}_{sp\_snr\_f} = \begin{cases} S^{[-1]}_{sp\_snr\_f} + SNR^{[0]}_f & if \ f_{SAD3}=1 \ and \ SNR^{[0]}_f > (S^{[-1]}_{sp\_snr\_f} / N^{[-1]}_{sp\_snr\_f}) \\ S^{[-1]}_{sp\_snr\_f} & otherwise \end{cases} \quad (274)$$

$$N^{[0]}_{sp\_snr\_f} = \begin{cases} N^{[-1]}_{sp\_snr\_f} + 1 & if \ f_{SAD3}=1 \ and \ SNR^{[0]}_f > (S^{[-1]}_{sp\_snr\_f} / N^{[-1]}_{sp\_snr\_f}) \\ N^{[-1]}_{sp\_snr\_f} & otherwise \end{cases} \quad (275)$$

If $SNR_{flux} > (B + SNR_{lt} *A)$, where A and B are two constants. f_{SAD3} is set to 1. If any one of the following conditions is met:

condition 1: $SNR_{flux} > 2.1 + 0.24 SNR_{lt}$

condition 2: $SNR_{flux} > T_1 \ and \ F_{SC}(3) > T_2 \ and \ SNR_{lt\_org} < T_3$

f_{SAD3} is to 1. Where T_1, T_2 and T_3 are the thresholds.

f) Update the number of hangover frames for active sound according to the decision result, the long-time SNR, and the average SNR of all sub-bands for several previous frames, and the SNR parameters and the SAD3 decision for the current frame; See subclause 5.1.12.6.7 for details;

g) Add the active-sound hangover according to the decision result and the number of hangover frames for active sound of the current frame to make the SAD3 decision. See subclause 5.1.12.6.7 for details;

h) Make a combined decision with f_{SAD} and the f_{SAD3} decision, the output flag of the combined decision is namely combined f_{SAD}. See subclause 5.1.12.7;

i) After Steps g) and h), the average energy of long-time background noise, denoted by $E_{lt\_inactive}$, can be computed with the SAD decisions combined f_{SAD} and f_{SAD}. $E_{lt\_inactive}$ is used to make the SAD decision for the next frame. If both combined f_{SAD} and f_{SAD} are 0, $S_{ns\_snr\_f}$, $N_{ns\_snr\_f}$ are updated and $E_{lt\_inactive}$ is computed as follows:

$$S^{[0]}_{ns\_snr\_f} = \begin{cases} S^{[-1]}_{ns\_snr\_f} + SNR^{[0]}_f & if \ f_{SAD} = 0 \ and \ the \ combined \ f_{SAD} = 0 \\ S^{[-1]}_{ns\_snr\_f} & otherwise \end{cases} \quad (276)$$

$$N^{[0]}_{ns\_snr\_f} = \begin{cases} N^{[-1]}_{ns\_snr\_f} + 1 & if \ f_{SAD} = 0 \ and \ the \ combined \ f_{SAD} = 0 \\ N^{[-1]}_{ns\_snr\_f} & otherwise \end{cases} \quad (277)$$

$$E^{[0]}_{lt\_inactive} = E^{[0]}_{bg\_sum} / N^{[0]}_{bg} \quad (278)$$

where $E^{[0]}_{bg\_sum}$ and $N^{[0]}_{bg}$ is computed by:

$$E^{[0]}_{bg\_sum} = \begin{cases} E^{[-1]}_{bg\_sum} + E^{[0]}_f & if \ f_{SAD} = 0 \ and \ the \ combined \ f_{SAD} = 0 \\ E^{[-1]}_{bg\_sum} & otherwise \end{cases} \quad (279)$$

$$N^{[0]}_{bg} = \begin{cases} N^{[-1]}_{bg} + 1 & if \ f_{SAD} = 0 \ and \ the \ combined \ f_{SAD} = 0 \\ N^{[-1]}_{bg} & otherwise \end{cases} \quad (280)$$

The functions of the Pre-decision module are described in Steps a) - e) in this subclause.

5.1.12.6.7   SAD3 Hangover

The long-time SNR and the average SNR of all sub-bands are computed with the sub-band signal (See subclause 5.1.12.6.2.1 and 5.1.12.6.3). The current number of hangover frames for active sound is updated according to the SAD3 decision of several previous frames, SNR_{lt_org}, and SNR_{flux} and the SNR parameters and the SAD3 decision of the current frame. The precondition for updating the current number of hangover frames for active sound is that the flag of active sound indicates that the current frame is active sound. If both the number of previous continuous active frames $N_{continuous\_sp2}$<8 and SNR_{flux}<4.0, the curent number of hangover frames for active sound is updated by subtracting $N_{continuous\_sp2}$ from the minimum number of continuous active frames. Suppose the minimum number of continuous active frames is 8. The updated number of hangover frames for active sound, denoted by N_{hang}, is computed as follows:

$$N_{hang} = 8 - N_{continuous\_sp2} \tag{281}$$

Otherwise, if both $SNR_{flux} > 0.9$ and $N_{continuous\_sp2} > 50$, the number of hangover frames for active sound is set according to the value of SNR_{lt}. Otherwise, this number of hangover frames is not updated.

$N_{continuous\_sp2}$ is set to 0 for the first frame. When the current frame is the second frame and the subsequent frames, $N_{continuous\_sp2}$ is updated according to the previous combined f_{SAD} as follows:

If the previous combined f_{SAD} is 1, $N_{continuous\_sp2}$ is increase by 1;

If the previous combined f_{SAD} is 0, $N_{continuous\_sp2}$ is set equal to 0.

5.1.12.7   Final SAD decision

The feature parameters mentioned above divide into two categories. The first feature category includes the number of continuous active frames $N_{continuous\_sp2}$, the average total SNR of all sub-bands SNR_{flux}, and the tonality signal flag f_{tonal_signal}. SNR_{flux} is the average of SNR over all sub-bands for a predetermined number of frames. The second feature category includes the flag of noise type, SNR_{lf_smooth} in a predetermined period of time, the number of continuous noise frames, frequency-domain SNR.

First, the parameters in the first and second feature categories and f_{SAD} and f_{SAD3} are obtained. The first and second feature categories are used for the SAD detection.

The combined decision is made in the following steps:

a) Compute the energy of background noise over all sub-bands for the previous frame with the background update flag, the energy parameters, and the tonality signal flag of the previous frame and the energy of background noise over all sub-bands of the previous 2 frames. Computing the background update flag is described in subclause 5.1.12.6.5.

b) Compute the above-mentioned SNR_{flux} with the energy of background noise over all sub-bands of the previous frame and the energy parameters of the current frame.

c) Determine the flag of noise type according to the above-mentioned parameters SNR_{lt_org} and SNR_{lf_smooth}. First, the noise type is set to non-silence. Then, when SNR_{lt_org} is greater than the first preset threshold and SNR_{lf_smooth} is greater than the second preset threshold, the flag of noise type is set to silence.

Then, the first and second feature categories, f_{SAD3} and f_{SAD} are used for active-sound detection in order to make the combined decision of SAD.

When the input sampling frequency is 16 kHz and 32 kHz, the decision procedure is carried out as follows:

a) Select f_{SAD3} as the initial value of the combined f_{SAD} ;

b) If the noise type is silence, and the frequency-domain SNR SNR_f is greater than 0.2 and the combined f_{SAD} set 0, f_{SAD} is selected as the output of the SAD, combined f_{SAD} . Otherwise, go to Step c).

c) If the smoothed long-time frequency-domain average SNR is smaller than 10.5 or the noise type is not silence, go to Step d). Otherwise, the initial value of the combined f_{SAD} in Step a) is still selected as the decision result of the SAD;

d) If any one of the following conditions is met, the result of a logical operation **OR** of f_{SAD3} and f_{SAD} is used as the output of the SAD. Otherwise, go to Step e):

Condition 1: The average total SNR of all sub-bands is greater than the first threshold, e.g. 2.2;

Condition 2: The average total SNR of all sub-bands is greater than the second threshold, e.g. 1.5 and the number of continuous active frames is greater than 40;

Condition 3: The tonality signal flag is set to 1.

e) When the input sampling frequency is 32 kHz: If the noise type is silence, f_{SAD} is selected as the output of the SAD and the decision procedure is completed. Otherwise, the initial value of the combined f_{SAD} in Step a) is still selected as the decision result of the SAD. When the input sampling frequency is 16 kHz: f_{SAD} is selected as the output of the SAD and the decision procedure is completed.

When the input sampling frequency is neither 16 kHz nor 32 kHz, the procedure of the combined decision is performed as follows:

a) Select f_{SAD3} as the initial value of the combined f_{SAD} ;

b) If the noise type is silence, go to Step c). Otherwise, go to Step d);

c) If the smoothed long-time frequency-domain average SNR is greater than 12.5 and f_{tonal_signal} =0, the combined f_{SAD} is set to f_{SAD} . Otherwise, the initial value of combined f_{SAD} in Step a) is selected as the decision result of the SAD;

d) If any one of the following conditions is met, the result of a logical operation **OR** of f_{SAD3} and f_{SAD} is used as the output of the final SAD, combined f_{SAD} . Otherwise, the initial value of combined f_{SAD} in Step a) is selected as the decision result of the SAD;

Condition 1: The average total SNR of all sub-bands is greater than 2.0;

Condition 2: The average total SNR of all sub-bands is greater than 1.5 and the number of continuous active frames is greater than 30;

Condition 3: The tonality signal flag is set to 1.

After the combined f_{SAD} is obtained by using the above-mentioned method, it needs to be modified as follows:

a) Compute the number of background-noise updates, N_{update_count} according to the background update flag, specifically:

When the current frame is indicated as background noise by the background update flag and N_{update_count} is smaller than 1000, N_{update_count} increases by 1. Note that N_{update_count} is set to zero at the initialization of the codec.

b) Compute number of modified frames for active sound , N_{md_frames} according to the SAD3 decision f_{SAD3}, the number of background-noise updates N_{update_count}, and the number of hangover frames for active sound N_{hang}, specifically:

When the current frame is indicated as active sound by f_{SAD3} and N_{update_count} is smaller than 12, N_{md_frames} is selected as $max(20, N_{hang})$.

c) Compute the final decision of SAD for the current frame according to the number of modified frames for active sound N_{md_frames} and the combined f_{SAD}, specifically:

When the current frame is indicated as inactive sound by the combined f_{SAD} and N_{md_frames} is greater than 0, the final decision of SAD for the current frame, the combined f_{SAD} is modifiedas active sound and N_{md_frames} decreases by 1.

5.1.12.8   DTX hangover addition

For better DTX performance a version f_{SAD_DTX} of the combined f_{SAD} is generated through the addition of hangover. In this case there are two concurrent hangover logics that can extend the f_{SAD_DTX} active period. One is for DTX in general and one specifically to add additional DTX hangover in the case of music.

During the SAD initialization period the following variables are set as follows

$$
\begin{aligned}
ho_{cnt} &= 0 \\
ho_{cnt\_dtx} &= 0 \\
ho_{cnt\_music} &= 0
\end{aligned}
\tag{282}
$$

The general DTX hangover works in the same way as the SAD1 hangover the main difference is in the hangover length. Also here the initial DTX hangover length depends on SNR_{LT}, initially the hangover is set to 2 frames and if the current input bandwidth is NB and $SNR_{LT} < 16$ or $prim_{act\_HE} > 0.95$ it is set to 3, then follows a number of steps that may modify this start value. The modification depends on other input signal characteristics and codec mode.

The first two modifications increases the hangover length if there has already been a high activity, additional activity after a long burst has little effect on the total activity but can better cover short pauses. If there has been 12 or more active frames from the primary detector in SAD1 during the 16 last frames, that is $f_{LSAD\_cnt} > 12$, the allowed number of hangover frames is increased with 2 frames. Similarly if there has been 40 or more active frames for the final decision of SAD1 during the last 50 frames , that is $f_{SAD\_cnt} > 40$, the allowed number of hangover frames is increased with 5 frames. At this point the allowed number of hangover frames may have been increased with 7 frames over the initial value, and to limit the total number of hangover frames it is therefore limited to $HANGOVER\_LONG - 1$. Another condition for limiting the hangover addition is if the primary activity becomes low there are different limits for different codec conditions, for AMR_WB_IO core the limit is 2, the same limit is also used for high SNR $SNR_{LT} > 25$ for WB or SWB input in other conditions the limit is 3 frames. The condition for applying the limit is if the primary activity $f_{LSAD\_cnt} < 7$ or if the $SNR_{LT} > 16$ and $prim_{act\_HE} < 0.85$.

The DTX hangover can also be reduced if the final decision from SAD3 already includes a long hangover.

According to the noise type in SAD3, the decrement of the DTX hangover is set as shown in Table 14.

Table 14: Setting of the decrement of the DTX hangover

Bandwidth	Silence-type noise	Non-silence-type noise
NB	0	1
WB	2	3
SWB	2	1

As for the hangover in SAD1 the counting of DTX hangover frames is reset only if at least 3 consecutive active speech frames ($f_{SAD} == 1$) or if the SAD1 final decision has been active for over 45 of the 50 latest frames.

For the music hangover to start counting music hangover frames $prim_{act\_HE} > 0.98$ AND $E_t > 40$ AND $f_{LSAD\_cnt} > 14$ AND $f_{SAD\_cnt} > 48$ at which point the $f_{SAD} == 1$ for the next 15 frames or until hangover is terminated by the hangover termination logic, which can be triggered by the flag f_{hoT} which is described below.

The DTX hangover and the hangover described in subclause 5.1.12.3 when decisions from SAD1 and SAD2 are combined may be early terminated. The early hangover termination helps to increase the system capacity by saving unnecessary hangover frames. At each hangover frame, the comfort noise which will be produced at the decoder side is estimated at the encoder side, assuming if the current hangover frame would be encoded as the first SID frame after active burst. If the estimated comfort noise is found close to the noise characteristic maintained in the local CNG module in the encoder side, then no more hangover frame is considered needed and the hangover is terminated. Otherwise, hangover keeps on as long as the initial hangover length is not reached.

Specifically, the energy and the LSP spectrum of the comfort noise which will be produced at the decoder side are estimated at the encoder side. The energy of the current frame excitation is calculated

$$E_{sid} = \frac{1}{L} \sum_{n=0}^{L-1} r(n)^2 \qquad (283)$$

which is then converted to log domain

$$E'_{sid} = \log_2 E_{sid} \qquad (284)$$

where $r(n)$ is the LP excitation of the current frame calculated in subclause 5.6.2.1.5, L is the frame length, E'_{sid} is limited to non-negative value. An age weighted average energy, $enr_{hist-ave-weighted}$, is calculated from hangover frames except the current frame in the same way as described in sub-clause 6.7.2.1.2. The $enr_{hist-ave-weighted}$, together with the energy of the current frame excitation E_{sid} are used to compute the estimated excitation energy for the comfort noise, E_{est}.

$$E_{est} = \alpha \cdot enr_{hist-ave-weighted} + (1-\alpha) \cdot E_{sid} \qquad (285)$$

where α is a smoothing factor, $\alpha = 0.8$ if m, the number of hangover frames used for $enr_{hist-ave-weighted}$ calculation is less than 3, otherwise, $\alpha = 0.95$. The estimated excitation energy for the comfort noise is then converted to log domain

$$E'_{est} = \log_2 E_{est} \qquad (286)$$

where E'_{est} is bounded to non-negative value. An average LSP vector, $ho_{lsp-ave-weighted}$, is calculated over the same hangover frames where the age weighted average energy $enr_{hist-ave-weighted}$ is calculated in the same way as described in sub-clause 6.7.2.1.2. The $ho_{lsp-ave-weighted}$, together with the end-frame LSP vector of the current frame are used to compute the estimated LSP vector for the comfort noise, q_{est}.

$$q_{est} = 0.8 \cdot ho_{lsp-ave-weighted} + 0.2 \cdot q_{end} \qquad (287)$$

A set of energy and LSP difference parameters are calculated. The difference between the current frame log excitation energy and the log hangover average excitation energy is calculated.

$$dE_{s2h} = \left| E'_{sid} - \log_2 (enr_{hist-ave-weighted}) \right| \qquad (288)$$

The difference between the current frame end-frame LSP vector and the hangover average LSP vector is calculated.

$$dq_{s2h} = \sum_{k=0}^{M-1} \left| q_{end}(k) - ho_{lsp-ave-weighted}(k) \right| \qquad (289)$$

where M is the order of LP filter. The difference between the estimated log excitation energy for the comfort noise and the current log excitation energy for the comfort noise kept in the local CNG module is calculated.

$$dE = \left| E'_{est} - \log_2 E_{CN} \right| \qquad (290)$$

where E_{CN} is the comfort noise excitation energy kept in the local CNG module as calculated in subclause 5.6.2.1.6. The difference between the estimated LSP vector for the comfort noise and the current LSP vector for the comfort noise kept in the local CNG module is calculated.

$$dq = \sum_{k=0}^{M-1} \left| q_{est}(k) - \bar{\bar{q}}(k) \right| \qquad (291)$$

where $\bar{\bar{q}}$ is the comfort noise LSP vector kept in the local CNG module as calculated in subclause 5.6.2.1.4. The maximum difference per LSP element between the estimated LSP vector for the comfort noise and the current LSP vector for the comfort noise kept in the local CNG module is calculated.

$$dq_{max} = \max\left(\left| q_{est}(k) - \bar{\bar{q}}(k) \right|, \qquad k = 0,1,...,M-1 \right) \qquad (292)$$

The hangover termination flag f_{hoT} is set to 1 if $dq < 0.4$ and $dE < 1.4$ and $dq_{s2h} < 0.4$ and $dE_{s2h} < 1.2$ and $dq_{max} < 0.1$ when operating in VBR mode, or if $dq < 0.4$ and $dE < 0.8$ and $dq_{s2h} < 0.4$ and $dE_{s2h} < 0.8$ and $dq_{max} < 0.1$ when operating in non-VBR mode. Otherwise f_{hoT} is set to 0. A f_{hoT} set to 1 means the current frame can be encoded as a SID frame even it is still in the hangover period. For safety reason of prevent CNG on short pauses between speech utterances, the actual encoding of SID frame is delayed by one frame.

5.1.13 Coding mode determination

To get the maximum encoding performance, the LP-based core uses a signal classification algorithm with six distinct coding modes tailored for each class of signal, namely the Unvoiced Coding (UC) mode, Voiced Coding (VC) mode, Transition Coding (TC) mode, Audio Coding (AC) mode, Inactive Coding (IC) mode and Generic Coding (GC) mode. The signal classification algorithm uses several parameters, some of them being optimized separately for NB and WB inputs.

Figure 13 shows a simplified high-level diagram of the signal classification procedure. In the first step, the SAD decision is queried whether the current frame is active or inactive. In case of inactive frame, IC mode is selected and the procedure is terminated. In the IC mode the inactive signal is encoded either in the transform domain by means of the AVQ technology or in the time/transform domain by means of the GSC technology, described below. In case of active frames, the speech/music classification algorithm is run to decide whether the current frame shall be coded with the AC mode. The AC mode, has been specifically designed to efficiently encode generic audio signals, particularly music. It uses a hybrid encoding technique, called the Generic Signal audio Coder (GSC) which combines both, LP-based coder operated in the time domain and a transform-domain coder. If the frame is not classified as "audio", the classification algorithm continues with selecting unvoiced frames to be encoded with the UC mode. The UC mode is designed to encode unvoiced frames. In the UC mode, the adaptive codebook is not used and the excitation is composed of two vectors selected from a linear Gaussian codebook.

If the frame is not classified as unvoiced, then detection of stable voiced frames is applied. Quasi-periodic segments are encoded with the VC mode. VC selection is conditioned by a smooth pitch evolution. It uses ACELP technology, but given that the pitch evolution is smooth throughout the frame, more bits are assigned to the algebraic codebook than in the GC mode.

The TC mode has been designed to enhance the codec performance in the presence of frame erasures by limiting the usage of past information [19]. To minimize at the same time its impact on a clean channel performance, it is used only on the most critical frames from a frame erasure point of view – these are voiced frames following voiced onsets.

If a frame is not classified in one of the above coding modes, it is likely to contain a non-stationary speech segment and is encoded using a generic ACELP model (GC).
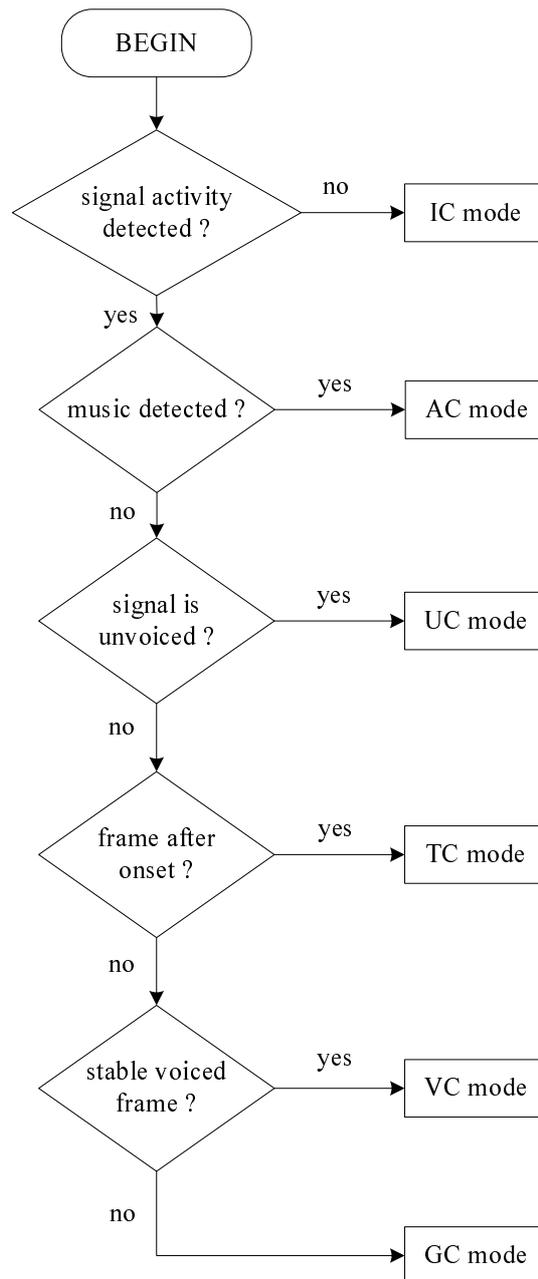
Figure 13: High-level diagram of the coding mode determination procedure

The selection of the coding modes is not uniform across the bitrates and input signal bandwidth. These differences will be described in detail in the subsequent sections. The classification algorithm starts with setting the current mode to GC.

5.1.13.1 Unvoiced signal classification

The unvoiced parts of the signal are characterized by a missing periodic component. The classification of unvoiced frames exploits the following parameters:

– voicing measures

– spectral tilt measures

– sudden energy increase from a low level to detect plosives

– total frame energy difference

– energy decrease after spike

5.1.13.1.1 Voicing measure

The normalized correlation, used to determine the voicing measure, is computed as part of the OL pitch searching module described in clause 5.1.10. The average normalized correlation is then calculated as

$$\overline{R}_{xy3} = \frac{1}{3}\left(C_{norm}^{[0]} + C_{norm}^{[1]} + C_{norm}^{[2]}\right) \tag{293}$$

where $C_{norm}^{[i]}$ is defined in subclause 5.1.11.3.2.

5.1.13.1.2 Spectral tilt

The spectral tilt parameter contains information about frequency distribution of energy. The spectral tilt is estimated in the frequency domain as a ratio between the energy concentrated in low frequencies and the energy concentrated in high frequencies, and is computed twice per frame.

The energy in high frequencies is computed as the average of the energies in the last two critical bands

$$\overline{E}_h = 0.5\left(E_{CB}\left(b_{max} - 1\right) + E_{CB}\left(b_{max}\right)\right) \tag{294}$$

where $E_{CB}(i)$ are the critical band energies, computed in subclause 5.1.5.2 and b_{max} is the maximum useful critical band ($b_{max} = 19$ for WB inputs and $b_{max} = 16$ for NB inputs).

The energy in low frequencies is computed as the average of the energies in the first 10 critical bands for WB signals and in 9 critical bands for NB signals. The middle critical bands have been excluded from the computation to improve the discrimination between frames with high-energy concentration in low frequencies (generally voiced) and with high-energy concentration in high frequencies (generally unvoiced). In between, the energy content is not informative for any of the classes and increases the decision uncertainty.

The energy in low frequencies is computed differently for voiced half-frames with short pitch periods and for other inputs. For voiced female speech segments, the harmonic structure of the spectrum is exploited to increase the voiced-unvoiced discrimination. These frame are characterized by the following the condition

$$0.5\left(C_{norm}^{[0]} + C_{norm}^{[1]}\right) + r_e > 0.6 \quad \text{AND} \quad T_{OL}^{[2]} < 128 \tag{295}$$

where $C_{norm}^{[i]}$ are computed as defined in subclause 5.1.10.4 and the noise correction factor r_e as defined in subclause 5.1.10.6. For these frames, $\overline{E}_l$ is computed bin-wise and only frequency bins sufficiently close to the speech harmonics are taken into account in the summation. That is

$$\overline{E}_l = \frac{1}{C}\sum_{k=K_{min}}^{25} E_{BIN}\left(k\right)w_h\left(k\right) \tag{296}$$

where K_{min} is the first bin ($K_{min} = 1$ for WB inputs and $K_{min} = 3$ for NB inputs) and $E_{BIN}(k)$ are the bin energies, as defined in subclause 5.1.5.2, in the first 25 frequency bins (the DC component is not considered). Note that these 25 bins correspond to the first 10 critical bands and that the first 2 bins not included in the case of NB input constitute the first critical band. In the summation above, only the terms related to the bins close to the pitch harmonics are considered. So $w_h\left(k\right)$ is set to 1, if the distance between the nearest harmonics is not larger than a certain frequency threshold (50 Hz) and is set to 0 otherwise. The counter C is the number of the non-zero terms in the summation. In other words, only bins closer than 50 Hz to the nearest harmonics are taken into account. Thus, only high-energy terms will be included in the sum if the structure is harmonic at low frequencies. On the other hand, if the structure is not harmonic, the selection of the terms will be random and the sum will be smaller. Thus, even unvoiced sounds with high energy content in low frequencies can be detected. This processing cannot be done for longer pitch periods, as the frequency resolution is not sufficient. For frames not satisfying the condition (295), the low frequency energy is computed per critical band as

$$\bar{E}_l = \frac{1}{10}\sum_{k=0}^{9} E_{CB}(k),$$

$$\bar{E}_l = \frac{1}{9}\sum_{k=1}^{9} E_{CB}(k),$$

(297)

for WB and NB inputs, respectively. The resulting low- and high-frequency energies are obtained by subtracting the estimated noise energy from the values $\bar{E}_l$ and $\bar{E}_h$ calculated above. That is

$$E_h = \bar{E}_h - N_h$$

(298)

$$E_l = \bar{E}_l - N_l$$

(299)

where N_h is the average noise energy in critical bands 18 and 19 for WB inputs, and 15 and 16 for NB inputs and N_l is the average noise energy in the first 10 critical bands for WB input and in the critical bands 1-9 for NB inputs. They are computed similarly as in the two equations above. The estimated noise energies have been integrated into the tilt computation to account for the presence of background noise.

Finally, the spectral tilt is given by

$$e_{tilt} = \frac{E_l}{E_h}$$

(300)

For NB signals, the missing bands are compensated by multiplying e_{tilt} by 6. Note that the spectral tilt computation is performed twice per frame to obtain $e_{tilt}^{[0]}$ and $e_{tilt}^{[1]}$, corresponding to both spectral analyses per frame. The average spectral tilt used in unvoiced frame classification is given by

$$\bar{e}_{tilt} = \frac{1}{3}\left(e_{tilt}^{[-1]} + e_{tilt}^{[0]} + e_{tilt}^{[1]}\right)$$

(301)

where $e_{tilt}^{[-1]}$ is the tilt in the second half of the previous frame.

5.1.13.1.3 Sudden energy increase from a low energy level

The maximum energy increase E_d from a low signal level is evaluated on eight short-time segments having the length of 32 samples. The energy increase is then computed as the ratio of two consecutive segments provided that the first segment energy was sufficiently low. For better resolution of the energy analysis, a second pass is computed where the segmentation is done with a 16 sample offset. The short-time maximum energies are computed as

$$E_{1,st}^{[j]} = \max_{i=0}^{31}\left(s_{pre}^2(i+32j)\right), \quad j = -1,\ldots,7,$$

(302)

where $j = -1$ corresponds to the last segment of the previous frame, and $j = 0, \ldots 7$ corresponds to the current frame. The second set of maximum energies is computed by shifting the speech indices in equation (302) by 16 samples. That is

$$E_{2,st}^{[j]} = \max_{i=0}^{31}\left(s_{pre}^2(i+32j-16)\right), \quad j = 0,\ldots,8,$$

(303)

$$E_{2,st}^{[j]} = \max_{i=0}^{31}\left(s_{pre}^2(i+32j+16)\right), \quad j = -1,\ldots,7,$$

(304)

The maximum energy variation E_d is computed as follows:

$$E_d = \max_{x,j} \left\{ \frac{E_{x,st}^{[j]}}{E_{x,st}^{[j-1]} + 1} \right\}, \quad for\ j = 0,\ldots,7,\ x = 1,\ldots,2 \tag{305}$$

5.1.13.1.4 Total frame energy difference

The classification of unvoiced frames is further improved by taking into account the difference of total frame energy. This difference is calculated as

$$dE_t = E_t - E_t^{[-1]}$$

where E_t is the total frame energy calculated in subclause 5.1.5.2 and $E_t^{[-1]}$ is the total frame energy in the previous frame.

5.1.13.1.5 Energy decrease after spike

The detection of energy decrease after a spike prevents the UC mode on significant temporal events followed by relatively rapid energy decay. Typical examples of such signals are castanets.

The detection of the energy decrease is triggered by detecting a sudden energy increase from a low level as described in subclause 5.1.13.1.3. The maximum energy variation E_d must be higher than 30 dB. Further, for NB inputs the mean correlation must not be too high, i.e. the condition $\overline{R}_{xy3} + r_e < 0.68$ must be satisfied too.

The energy decrease after a spike is searched within 10 overlapped short-time segments (of both sets of energies) following the detected maximum energy variation. Let's call $j_{\max}$ the index j for which E_d was found, and the corresponding set of energies $x_{\max}$. If $x_{\max} = 1$, then the searched interval is $j = j_{\max},\ldots,j_{\max} + 4$ for both sets. If $x_{\max} = 2$, then the searched interval is $j = j_{\max},\ldots,j_{\max} + 4$ for the 2nd set, but $j = j_{\max} + 1,\ldots,j_{\max} + 5$ for the 1st set of energies.

The energy decrease after a spike is then searched as follows. As the energy can further increase beyond the segment $[x_{\max}, j_{\max}]$ for which E_d was found, the energy increase is tracked beyond that segment to find the last segment with energy still monotonically increasing. Let's denote the energy of that segment $E_{d,\max}$. Starting from that segment until the end of the searched interval, the minimum energy $E_{d,\min}$ is then determined. The detection of an energy decrease after spike is based on the ratio of the maximum and minimum energies

$$dE_2 = \frac{E_{d,\max}}{E_{d,\min} + 10^{-5}} \tag{306}$$

This ratio is then compared to a threshold of 21 dB for NB inputs and 30 dB for other inputs.

The detection of energy decrease after a spike further uses a hysteresis in the sense that UC is prevented not only in the frame where dE_2 is above the threshold ($dE_{2,hyst} = 0$), but also in the next frame ($dE_{2,hyst} = 1$). In subsequent frames ($dE_{2,hyst} = 2$), the hysteresis is reset ($dE_{2,hyst} = -1$) only if the following condition is met:

$$\left(dE_t > 5\right) \quad OR \quad \left[\left(E_{rel} > -13\right) AND \left(\overline{R}_{xy3} + r_e < 0.695\right)\right]. \tag{307}$$

Given that the searched interval of overlapped segments is always 10, it can happen that the detection cannot be completed in the current frame if a rapid energy increase happens towards the frame end. In that case, the detection is completed in the next frame, however, as far as the hysteresis logic is concerned, the detection of energy decrease after a spike still pertains to the current frame.

5.1.13.1.6 Decision about UC mode

To classify frames for encoding with UC mode, several conditions need to be met. As the UC mode does not use the adaptive codebook and no long-term prediction is thus exploited, it is necessary to make sure that only frames without

periodic content are coded with this mode. The decision logic is somewhat different for WB and NB inputs and is described for both cases separately.

For WB inputs, all of the following conditions need to be satisfied to select the UC mode for the current frame.

1. Normalized correlation is low:

$$\overline{R}_{xy3} + r_e < 0.695$$

2. Energy is concentrated in high frequencies.

$$\left(e_{tilt}^{[0]} < 6.2\right) \text{AND} \left(E_h^{[0]} > 0.0035\right)$$
$$\left(e_{tilt}^{[1]} < 6.2\right) \text{AND} \left(E_h^{[1]} > 0.0035\right)$$

3. The current frame is not in a segment following voiced offset:

$$c_{raw}^{[-1]} = \text{UC} \quad \text{OR} \quad \left[\left(e_{tilt}^{[0]} < 2.4\right) \text{AND} \left(E_h^{[0]} > 0.0035\right) \text{AND} \left(C_{norm}^{[0]} + r_e < 0.74\right)\right]$$

where $c_{raw}^{[-1]}$ is the raw coding mode selected in the previous frame (described later in this document).

4. There is no sudden energy increase:

$$E_d^{[0]} \le 30 \quad \text{AND} \quad E_d^{[-1]} \le 30$$

5. The current frame is not in a decaying segment following sharp energy spike:

$$dE_{2,hyst} < 0$$

For NB inputs, the following conditions need to be satisfied to classify the frame for NB UC coding.

1. Normalized correlation is low:

$$\overline{R}_{xy3} + r_e < 0.68 \quad \text{AND} \quad C_{norm}^{[2]} + r_e < 0.79$$

2. Energy is concentrated in high frequencies.

$$\left(e_{tilt}^{[0]} < 10\right) \text{AND} \left(E_h^{[0]} > 0.0035\right)$$
$$\left(e_{tilt}^{[1]} < 9.5\right) \text{AND} \left(E_h^{[1]} > 0.0035\right)$$

3. The current frame is not in a segment following voiced offset:

$$CT_{raw}^{[-1]} = \text{UC} \quad \text{OR} \quad \left[\left(e_{tilt}^{[0]} < 9.8\right) \text{AND} \left(E_h^{[0]} > 0.0035\right) \text{AND} \left(C_{norm}^{[0]} + r_e < 0.76\right)\right]$$

where $CT_{raw}^{[-1]}$ is the raw coding mode selected in the previous frame (described later in this document).

4. There is no sudden energy increase:

$$E_d^{[0]} \le 29 \quad \text{AND} \quad E_d^{[-1]} \le 29$$

5. The current frame is not in a decaying segment following sharp energy spike:

$$dE_{2,hyst} < 0$$

5.1.13.2 Stable voiced signal classification

The second step in the signal classification algorithm is the selection of stable voiced frames, i.e. frames with high periodicity and smooth pitch contour. The classification is mainly based on the results of the fractional open-loop pitch search described in section 5.1.10.9. As the fractional open-loop pitch search is done in a similar way as the closed-loop pitch search, it is assumed that if the open-loop search gives a smooth pitch contour within predefined limits, the optimal closed-loop pitch search would give similar results and limited quantization range can then be used. The frames are classified into the VC mode if the fractional open-loop pitch analysis yields a smooth contour of pitch evolution over all four subframes. The pitch smoothness condition is satisfied if $d_{fr}^{[i+1]} - d_{fr}^{[i]} < 3$, for i = 0, 1, 2, where $d_{fr}^{[i]}$ is the fractional open-loop pitch lag found in subframe *i* (see section 5.1.10.9 for more details). Furthermore, in order to select VC mode for the current frame the maximum normalized correlation C_{fr} must be greater than 0.605 in each of the four subframes. Finally, the spectral tilt $\overline{e}_{tilt}$ must be higher than 4.0.

The decision about VC mode is further improved for frames with stable short pitch evolution and high correlation (e.g. female or child voices or opera voices). Pitch smoothness is again satisfied if $d_{fr}^{[i+1]} - d_{fr}^{[i]} < 3$, for i = 0, 1, 2. High correlation is achieved in frames for which the mean value of the normalized correlation in all four subframes is higher than 0.95 and the mean value of the smoothed normalized correlation is higher than 0.97. That is

$$C_{fr} = \frac{1}{4} \sum_{i=0}^{3} C_{fr}^{[i]} > 0.95 \qquad (308)$$

The smoothing of the normalized correlation is done as follows

$$\overline{C}_{fr}^{[0]} = 0.75 \overline{C}_{fr}^{[-1]} + 0.25 C_{fr} \qquad (309)$$

Finally, VC mode is also selected in frames for which the flag $f_{spitch} = Stab\_short\_pitch\_flag = flag\_spitch$ has been previously set to 1 in the module described in sub-clause 5.1.10.8. Further, when the signal has very high pitch correlation, f_{spitch} is also set to 1 so that the VC mode is maintained to avoid selecting Audio Coding (AC) mode later, as follows,

```
If ( fspitch =1 or
        (dpit1 <= 3 AND dpit2 <= 3 AND dpit3 <= 3 AND Voicingm > 0.95 AND Voicingsm > 0.97) or
        (AC_old=0 AND Voicingsm > 0.97))
{
    VC = 1;
    fspitch =1
}
```

wherein $dpit1 = \left| T_{OL}^{[0]} - T_{OL}^{[1]} \right|$, $dpit2 = \left| T_{OL}^{[1]} - T_{OL}^{[2]} \right|$, $dpit3 = \left| T_{OL}^{[2]} - T_{OL}^{[3]} \right|$, $Voicing_m$ and $Voicing_{sm}$ are defined in subclause 5.1.10.8, and *AC_old=0* means the previous Audio Coding mode is not selected.

The decision taken so far (i.e. after UC and VC mode selection) is called the "raw" coding mode, denoted c_{raw}. The value of this variable from the current frame and from the previous frame is used in other parts of the codec.

5.1.13.3 Signal classification for FEC

This subclause describes the refinement of the signal classification algorithm described in the previous section in order to improve the codec's performance for noisy channels. The classification used to select UC and VC frames cannot be directly used in the FEC as the purpose of the classification is not the same. Instead, better performance could be achieved by tuning both classification aspects separately.

The basic idea behind using a different signal classification approach for FEC is the fact that the ideal concealment strategy is different for quasi-stationary speech segments and for speech segments with rapidly changing characteristics. Whereas the best processing of erased frames in non-stationary speech segments can be summarized as a rapid drop of energy, in the case of quasi-stationary signal, the speech-encoding parameters do not vary dramatically and can be kept practically unchanged during several adjacent erased frames before being damped. Also, the optimal method for a signal recovery following an erased block of frames varies with the classification of the speech signal.

Furthermore, this special classification information is also used to select frames to be encoded with the TC mode (see subclause 5.1.13.4).

To distinguish the signal classification algorithm for FEC from the signal classification algorithm for coding mode determination (described earlier in subclauses 5.1.13.1 and 5.1.13.2), we will refer here to "signal class" rather than "coding mode" and denote it c_{FEC}.

5.1.13.3.1 Signal classes for FEC

The frame classification is done with the consideration of the concealment and recovery strategy in mind. In other words, any frame is classified in such a way that the concealment can be optimal if the following frame is missing, and that the recovery can be optimal if the previous frame was lost. Some of the classes used in the FEC do not need to be transmitted, as they can be deduced without ambiguity at the decoder. Here, five distinct classes are used and defined as follows:

- INACTIVE CLASS comprises all inactive frames. Note, that this class is used only in the decoder.

- UNVOICED CLASS comprises all unvoiced speech frames and all frames without active speech. A voiced offset frame can also be classified as UNVOICED CLASS if its end tends to be unvoiced and the concealment designed for unvoiced frames can be used for the following frame in case it is lost.

- UNVOICED TRANSITION CLASS comprises unvoiced frames with a possible voiced onset at the end. The onset is however still too short or not built well enough to use the concealment designed for voiced frames. The UNVOICED TRANSITION CLASS can only follow a frame classified as UNVOICED CLASS or UNVOICED TRANSITION CLASS.

- VOICED TRANSITION CLASS comprises voiced frames with relatively weak voiced characteristics. Those are typically voiced frames with rapidly changing characteristics (transitions between vowels) or voiced offsets lasting the whole frame. The VOICED TRANSITION CLASS can only follow a frame classified as VOICED TRANSITION CLASS, VOICED CLASS or ONSET CLASS.

- VOICED CLASS comprises voiced frames with stable characteristics. This class can only follow a frame classified as VOICED TRANSITION CLASS, VOICED CLASS or ONSET CLASS.

- ONSET CLASS comprises all voiced frames with stable characteristics following a frame classified as UNVOICED CLASS or UNVOICED TRANSITION CLASS. Frames classified as ONSET CLASS correspond to voiced onset frames where the onset is already sufficiently built for the use of the concealment designed for lost voiced frames. The concealment techniques used for frame erasures following the ONSET CLASS are the same as those following the VOICED CLASS. The difference is in the recovery strategy.

- AUDIO CLASS comprises all frames with harmonic or tonal content, especially music. Note that this class is used only in the decoder.

5.1.13.3.2 Signal classification parameters

The following parameters are used for the classification at the encoder: normalized correlation, $\bar{R}_{xy2}$, spectral tilt measure, $\bar{e}_{tilt,dB}$, pitch stability counter, pc, relative frame energy, E_{rel}, and zero crossing counter, zc. The computation of these parameters which are used to classify the signal is explained below.

The normalized correlation, used to determine the voicing measure, is computed as part of the OL pitch analysis module described in subclause 5.1.10. The average correlation $\bar{R}_{xy2}$ is defined as

$$\bar{R}_{xy2} = \frac{1}{2}\left(C_{norm}^{[1]} + C_{norm}^{[2]}\right) \tag{310}$$

where $C_{norm}^{[1]}$ and $C_{norm}^{[2]}$ are the normalized correlation of the second half-frame and the look ahead, respectively.

The spectral tilt measure, $\bar{e}_{tilt,dB}$, is computed as the average (in dB) of both frame tilt estimates, as described in subclause 5.1.13.1.2. That is

$$\overline{e}_{tilt,dB} = 10\log\left(\max\left\{ e_{tilt}^{[0]} . e_{tilt}^{[1]} , 1 \right\} \right) \tag{311}$$

The pitch stability counter, pc, assesses the variation of the pitch period. It is computed as follows:

$$pc = \left| d^{[1]} - d^{[0]} \right| + \left| d^{[2]} - d^{[1]} \right| \tag{312}$$

where the values $d^{[0]}$, $d^{[1]}$ and $d^{[2]}$ correspond to the three OL pitch estimates evaluated in each frame (see subclause 5.1.10).

The last parameter is the zero-crossing rate, zc, computed on the second half of the current speech frame and the look-ahead. Here, the zero-crossing counter, zc, counts the number of times the signal sign changes from positive to negative during that interval. The zero-crossing rate is calculated as follows

$$zc = -\frac{1}{4} \sum_{n=112}^{368} \text{sign}\left[s_{pre}(n).s_{pre}(n-1) \right] \tag{313}$$

where the function sign[.] returns +1 if the value is positive or -1 it is negative.

5.1.13.3.3 Classification procedure

The classification parameters are used to define a function of merit, f_m. For that purpose, the classification parameters are first scaled between 0 and 1 so that each parameter translates to 0 for an unvoiced signal and to 1 for a voiced signal. Each parameter, p_x, is scaled by a linear function as follows:

$$p_x^s = k_x p_x + c_x \tag{314}$$

and clipped between 0 and 1 (except for the relative energy which is clipped between 0.5 and 1). The function coefficients, k_x and c_x, have been found experimentally for each of the parameters so that the signal distortion due to the concealment and recovery techniques used in the presence of frame erasures is minimal. The function coefficients used in the scaling process are summarized in the following table.

Table 15: Coefficients of the scaling function for FEC signal classification

parameter	description	k_x	c_x
$\overline{R}_{xy2}$	normalized correlation	2.857	-1.286
$\overline{e}_{tilt,dB}$	spectral tilt	0.04167	0
pc	pitch stability counter	-0.07143	1.857
E_r	relative frame energy	0.05	0.45
zc	zero-crossing counter	-0.04	2.4

The function of merit has been defined as

$$f_m = \frac{1}{6}\left(2\overline{R}_{xy2}^s + \overline{e}_{tilt,dB}^s + pc^s + E_r^s + zc^s \right) \tag{315}$$

where the superscript s indicates the scaled version of the parameters. The classification is then done using the function of merit, f_m, and following the rules summarized in the following table.

Table 16: Rules for FEC signal classification

previous class	rule	selected class
VOICED CLASS ONSET CLASS VOICED TRANSITION CLASS	$f_m \geq 0.66$	VOICED CLASS
	$0.49 \leq f_m < 0.66$	VOICED TRANSITION CLASS
	$f_m < 0.49$	UNVOICED CLASS
UNVOICED CLASS UNVOICED TRANSITION CLASS	$f_m > 0.63$	ONSET CLASS
	$0.63 \geq f_m > 0.585$	UNVOICED TRANSITION CLASS
	$f_m \leq 0.585$	UNVOICED CLASS

For the purpose of FEC signal classification, all inactive speech frames, unvoiced speech frames and frames with very low energy are directly classified as UNVOICED CLASS. This is done by checking the following condition

$$f_{LSAD} = 0 \quad \text{OR} \quad c_{raw} = \text{UC} \quad \text{OR} \quad E_r < -6 \tag{316}$$

which has precedence over the rules defined in the above table.

5.1.13.4   Transient signal classification

As a compromise between the clean-channel performance of the codec and its robustness to channel errors, the use of the TC mode is limited only to a single frame following voiced onsets and to transitions between two different voiced segments. Voiced onsets and transitions are the most problematic parts from the frame erasure point of view. Therefore, the frame after the voiced onset and voiced transitions must be as robust as possible. If the transition/onset frame is lost, the following frame is encoded using the TC mode, without the use of the past excitation signal, and the error propagation is broken.

The TC mode is selected according to the counter of frames from the last detected onset/transition i_{TC}. The onset/transition detection logic is described by the state machine in the following diagram.
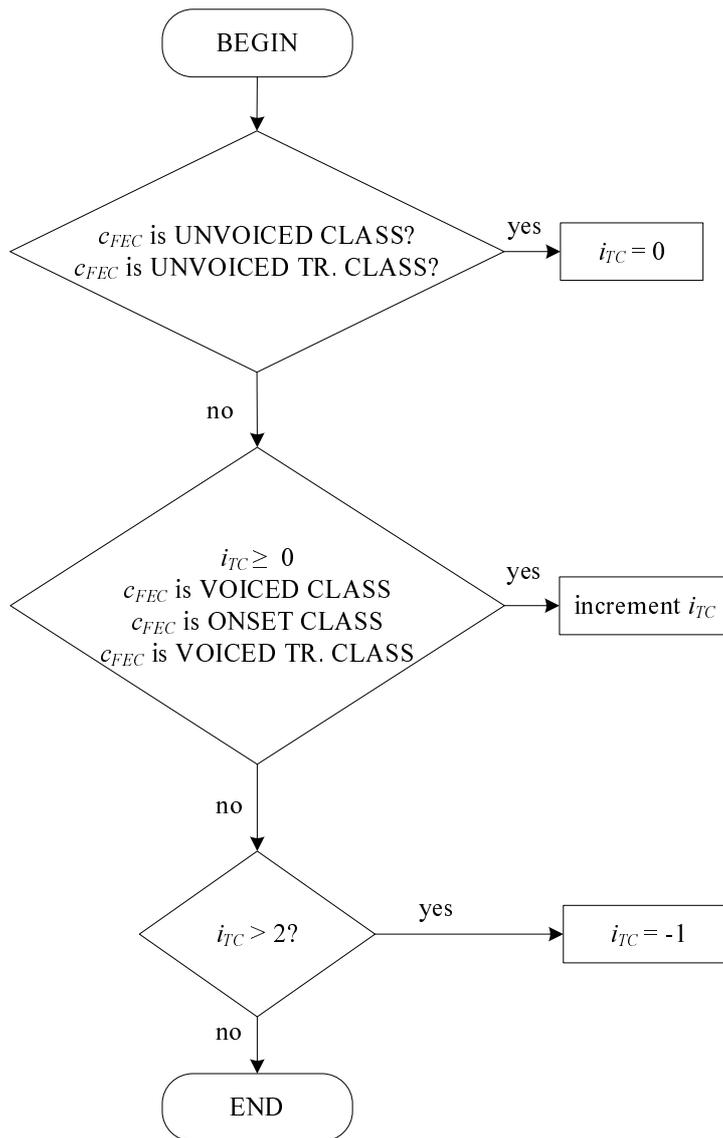
Figure 14 : TC onset/transition state machine

In the above logic, i_{TC} is set to 0 for all inactive frames, resp. frames for which the FEC signal class is either UNVOICED CLASS or UNVOICED TRANSITION CLASS. When the first onset frame is encountered i_{TC} is set to 1. This is again determined by the FEC signal class. The onset/transition frame is always coded with the GC mode, i.e. the coding mode is set to GC in this frame. The next active frame after the onset frame increases i_{TC} to 2 which means that there is a transition and TC mode is selected. The counter is set to -1 if none of the above situations happens waiting for the next inactive frame. Naturally, the GC/TC mode is selected by the above logic only when the current frame is an active frame, i.e. when $f_{LSAD} > 0$.

5.1.13.5 Modification of coding mode in special cases

In some special situations, the decision about coding mode is further modified. This is e.g. due to unsuitability of the selected coding mode at particular bitrate or due to signal characteristics which make the selected mode inappropriate. For example, the UC mode is supported only up to 9.6 kbps after which it is replaced by the GC mode. The reason is that for bitrates higher than 9.6 kbps the GC mode already has enough bits to fully represent the random content of an unvoiced signal. The UC mode is also replaced by the GC mode at 9.6 kbps if the counter of previous AC frames is bigger than 0. The counter of AC frames is initialized to the value of 200, incremented by 10 in every AC frame and decremented by 1 in other frames but not in IC frames. The counter is upper limited by 1000 and lower limited by 0.

At 32 and 64 kbps, only GC and TC modes are employed, i.e. if the coding mode has been previously set to UC or VC, it is overwritten to GC. Finally, for certain low-level signals, it could happen that the gain quantizer in the NB VC mode

goes out of its dynamic range. Therefore, the coding mode is changed to GC mode if the relative frame energy $E_r < -10$ dB but only at 8.0 kbps and lower bitrates.

The coding mode is also changed to the TC mode in case of mode switching. If, in the previous frame, 16 kHz ACELP core was used but the current frame uses 12.8 kHz ACELP core, it is better to prevent potential switching artefacts resulting from signal discontinuity and incorrect memory. This modification is done only for frames other than VC, i.e. if $c_{raw} \neq VC$, where c_{raw} is the raw coding mode described in subclause 5.1.13.2. Further, the modification takes place only for active frames in case of DTX operation.

The coding mode is overridden to the TC mode if MDCT-based core was used in the last frame but the current frame is encoded with an LP-based core. Finally, the coding mode is changed to the TC mode if the EVS codec is operated in the DTX mode and if the last frame was a SID frame encoded with the FD_CNG technology.

5.1.13.6 Speech/music classification

Music signals, in general, are more complex than speech and conform less to any known LP-based model. Therefore, it is of interest to distinguish music signals (generic audio signals) from speech signals. Speech/music classification then allows using a different coding approach to such signals. This new approach has been called the Generic audio Signal Coding mode (GSC), or the Audio Coding (AC) mode.

The speech/music classification is done in two stages. The first stage of the speech/music classifier is based on the Gaussian Mixture Model (GMM) and performs the best statistically based discrimination of speech from generic audio. The second stage has been optimized directly for the GSC mode. In other words, the classification in the second stage is done in such a way that the selected frames are suitable for the AC mode. Each speech/music classifier stage yields its own binary decision, f_{SM1} and f_{SM2} which is either 1 (music) or 0 (speech or background noise). The speech decision and the background noise decision have been grouped together only for the purposes of the speech/music classification. The selection of the IC mode for inactive signals incl. background noise is done later in the codec and described in subclause 5.1.13.5.7.

The decisions of the first and the second stage are refined and corrected for some specific cases in the subsequent modules, described below. The final decision about the AC mode is done based on f_{SM1} and f_{SM2} but the two flags are also used for the selection of the coder technology which is described in subclause 5.1.16.

5.1.13.6.1 First stage of the speech/music classifier

The GMM model has been trained on a large database of speech and music signals covering several male and female speakers, multiple languages and various genres of instrumental and vocal music. The statistical model uses a vector of 12 unique features, all normalized to a unit interval and derived from the basic parameters that have been calculated in the pre-processing part of the encoder. There are three statistical models inside the GMM: speech, music and noise. The statistical model of the background noise has been added to improve the SAD algorithm described in subclause 5.1.12. Each statistical model is represented by a mixture of six normal (Gaussian) distributions, determined by their relative weight, mean and full covariance matrix. The speech/music classifier exploits the following characteristics of the input signal:

- OL pitch

- normalized correlation

- spectral envelope (LSPs)

- tonal stability

- signal non-stationarity

- LP residual error

- spectral difference

- spectral stationarity
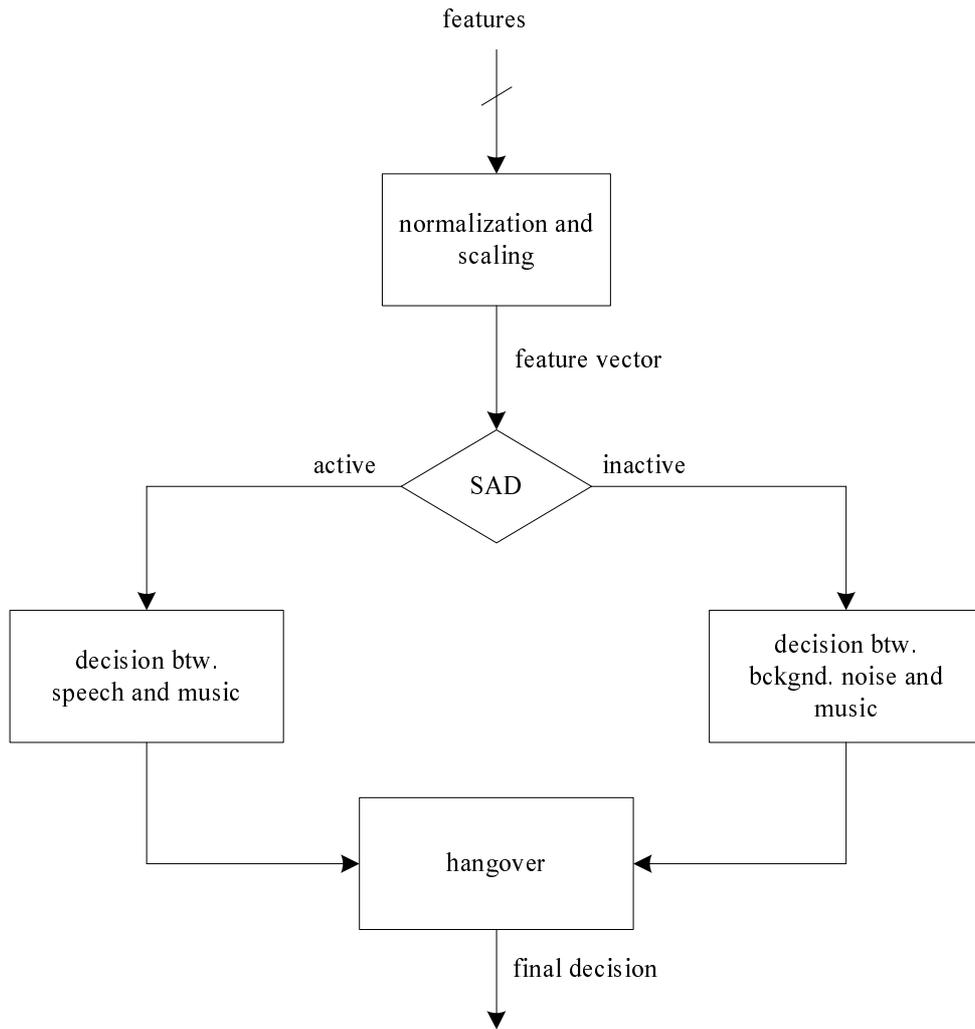
Figure 15 : Schematic diagram of the first stage of the speech/music classifier

The OL pitch feature is calculated as the average of the three OL pitch estimates, i.e.

$$FV_0 = \frac{1}{3}\left(T_{OL}^{[0]} + T_{OL}^{[1]} + T_{OL}^{[2]}\right) \tag{317}$$

where $T_{OL}^{[i]}$ are computed as in subclause 5.1.10.7. In onset/transition frames and in the TC frame after, it is better to use only the OL pitch estimate of the second analysis window, i.e. $FV_0 = T_{OL}^{[2]}$.

The normalized correlation feature FV_1 used by the speech/music classifier is the same one as used in the unvoiced signal classification. See subclause 5.1.13.1.1. for the details of its computation. In onset/transition frames and in the TC frame after, it is better to use only the correlation value of the second analysis window, i.e. $FV_1 = C_{norm}^{[2]}$.

There are five LSF parameters used as features inside the first stage of the speech/music classifier. These are calculated as follows

$$FV_{i+1} = \arccos(q_{end,i}) + \arccos(q_{end,i}^{[-1]}), \quad i = 1,..,5 \tag{318}$$

Another feature used by the speech/music classifier is the correlation map which is calculated as part of the tonal stability measure in subclause 5.1.11.2.5. However, for the purposes of speech/music classification, it is not the long-term correlation map which is summed but rather the correlation map of the current frame. The reason is to limit the impact of past information on the speech/music decision in the current frame. That is

$$FV_7 = \sum_{j=0}^{127} M_{cor}(j) + \sum_{j=0}^{127} M_{cor}^{[-1]}(j) \tag{319}$$

In case of NB signals, the value of FV_7 is multiplied by 1.53.

Signal non-stationarity is also used in the speech/music classifier but its calculation is slightly different than in the case of background noise estimation described in subclause 5.1.11.2.1. Firstly, the current log-energy per band is defined as

$$E_{CB2}(i) = \log(0.5 E_{CB}^{[0]}(i) + 0.5 E_{CB}^{[1]}(i)), \quad i=2,..,16 \tag{320}$$

Then,

$$FV_8 = \sum_{i=2}^{16} \left| E_{CB2}(i) - E_{CB2}^{[-1]}(i) \right| \tag{321}$$

The LP residual log-energy ratio is calculated as

$$FV_9 = \log\left(\frac{E(13)}{E(1)}\right) + \log\left(\frac{E^{[-1]}(13)}{E^{[-1]}(1)}\right) \tag{322}$$

where the superscript [-1] denotes values from the previous frame. In case of NB signals, the statistical distribution of FV_9 is significantly different than in case of WB signals. Thus, for NB signals $FV_9 = -1.647$.

For the last two features, the power spectrum must be normalized as follows:

$$PS_n(k) = \frac{PS(k)}{\sum_{k=3}^{69} PS(k)}, \quad k=3,..,69 \tag{323}$$

and difference spectrum calculated as follows:

$$dPS_n(k) = PS_n(k) - PS_n^{[-1]}(k), \quad k=3,..,69 \tag{324}$$

Then we calculate the spectral difference as the sum of $dPS_n(k)$ in the log domain. That is

$$FV_{10} = \log\left(\sum_{k=3}^{69} dPS_n(k)\right) + \log\left(\sum_{k=3}^{69} dPS_n^{[-1]}(k)\right) \tag{325}$$

The spectral non-stationarity is calculated as the product of ratios between power spectrum and the difference spectrum. This is done as follows

$$FV_{11} = \log \sum_{k=3}^{69} \frac{\max\left(PS_n(k), PS_n^{[-1]}(k)\right)}{dPS_n(k)} \tag{326}$$

5.1.13.6.2 Scaling of features in the first stage of the speech/music classifier

The feature vector FV_i, $i=0,..,11$ is scaled in such a way that all its values are approximately in the range [0;1]. This is done as

$$FV_s(i) = sfa_i FV_i + sfb_i \quad i=0,..,11 \tag{327}$$

where the scaling factors sfa_i and sfb_i have been found on a large training database. The scaling factors are defined in the following table.

Table 17: Scaling factors for feature vector in the speech/music classifier

i	WB		NB	
	sfa_i	sfb_i	sfa_i	sfb_i
0	0.048	-0.0952	0.0041	0
1	1.0002	0	0.8572	0.1020
2	0.6226	-0.0695	0.6739	-0.1000
3	0.5497	-0.1265	0.6257	-0.1678
4	0.4963	-0.2230	0.5495	-0.2380
5	0.5049	-0.4103	0.5793	-0.4646
6	0.5069	-0.5717	0.2502	0
7	0.0041	0	0.0041	0
8	0.0022	-0.0029	0.0020	0
9	0.0630	1.0015f	0.0630	1.0015
10	0.0684	0.9103f	0.0598	0.8967
11	0.1159	-0.2931	0.0631	0

5.1.13.6.3 Log-probability and decision smoothing

The multivariate Gaussian probability distribution is defined as

$$p(\mathbf{FV}) = \frac{1}{\sqrt{(2\pi)^k |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{FV} - \mathbf{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{FV} - \mathbf{\mu})\right) \tag{328}$$

where $\mathbf{FV}$ is the feature vector, $\mathbf{\mu}$ is the vector of means and $\mathbf{\Sigma}$ is the variance matrix. As stated before, the dimension of the feature vector is $k=12$. The means and the variance matrix are found by the training process of the Gaussian Mixture Model (GMM). This training is done by means of the EM (Expectation-Maximization) algorithm . The speech/music classifier is trained with a mixture of 6 Gaussian distributions. The log-likelihood of each mixture is defined as

$$L_i = -\frac{1}{2}\log(2\pi)^{12} + \log w_i + \log|\mathbf{\Sigma}_i| - \frac{1}{2}(\mathbf{FV} - \mathbf{\mu}_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{FV} - \mathbf{\mu}_i) \quad i=1,..,6 \tag{329}$$

where w_i is the weight of each mixture. The term $\log w_i + \log|\mathbf{\Sigma}_i|$ is calculated in advance and stored in the form of a look-up table. The probability over the complete set of 6 Gaussians is then calculated in the following way:

$$p_{all} = \sum_{i=1}^{6} \exp\left[\log w_i + \log|\mathbf{\Sigma}_i| - \frac{1}{2}(\mathbf{FV} - \mathbf{\mu}_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{FV} - \mathbf{\mu}_i)\right] \tag{330}$$

and the log-likelihood over the complete set as

$$L_{all} = \log(p_{all}) - \frac{1}{2}\log(2\pi)^{12} \tag{331}$$

Since there are three trained classes in the GMM model (speech, music and noise), three log-likelihood values are obtained by the above estimation process: L_s, L_m, L_n. Only L_s and L_m are used in the subsequent logic. L_n is used in the SAD algorithm to improve detection accuracy. Therefore, in case of inactive signal when f_{LSAD_HE} is 0

$$L_s = 1.2 L_n \tag{332}$$

The difference of log-likelihood is calculated as

$$dL_{sm} = L_m - L_s \tag{333}$$

which can be directly interpreted as a speech/music decision without hangover. This decision has low dynamic range and fluctuates a lot around zero, especially for mixed signals. To improve the detection accuracy, the decision is smoothed by means of AR filtering

$$wdL_{sm} = \gamma_{comb} dL_{sm} + (1 - \gamma_{comb})wdL_{sm}^{[-1]} \tag{334}$$

where the superscript [-1] denotes the previous frame and γ_{comb} is the filtering factor which is adaptively set on a frame-by-frame basis. The filtering factor is in the range [0;1] and is based on two measures (weighting factors). The first weighting factor γ_E is related to the relative frame energy and the second weighting factor γ_{drop} is designed to emphasize rapid negative changes of dL_{sm}.

The energy-based weight γ_E is calculated as follows

$$\gamma_E = 1 + \frac{E_r}{15}, \quad \text{conditioned by } 0.01 < \gamma_E < 1 \tag{335}$$

where E_r is relative frame energy. The result of the addition means that the weight has values close to 0.01 in low-energy segments and close to 1 in high energy, or more important, segments. Therefore, the smoothed decision follows the current decision more closely if the signal energy is relatively high and leads to past information being disregarded more readily. On the other hand, if the signal energy is low, the smoothed decision puts more emphasis on previous decisions rather than the current one. This logic is motivated by the observation that discrimination between speech and music is more difficult when the SNR of the signal is low.

The second weighting factor γ_{drop} is designed to track sudden transitions from music to speech. This situation happens only in frames where $dL_{sm} < 0$ and at the same time $dL_{sm} < dL_{sm}^{[-1]}$. In these frames

$$\begin{aligned}
\overline{w}_{drop} &= -dL_{sm} &&\text{,if } dL_{sm}^{[-1]} > 0 \\
\overline{w}_{drop} &= \overline{w}_{drop}^{[-1]} + (dL_{sm}^{[-1]} - dL_{sm}) &&\text{,otherwise}
\end{aligned} \tag{336}$$

where the parameter $\overline{w}_{drop}$ is a quantitative measure of sudden falls, or drops, in the value of dL_{sm}. This parameter is set to 0 in all frames that do not fulfil the previous condition. In the first frame when dL_{sm} falls below 0, it is set equal to its negative value and it is incremented when dL_{sm} continues to fall in consecutive frames. In the first frame when dL_{sm} stops decreasing it is reset back to 0. Thus, $\overline{w}_{drop}$ is positive only during frames of falling dL_{sm} and the bigger the fall the bigger its value. The weighting factor γ_{drop} is then calculated as

$$\gamma_{drop} = \frac{\overline{w}_{drop}}{20}, \quad \text{conditioned by } 0.1 < \gamma_{drop} < 1 \tag{337}$$

The filtering factor is then calculated by from the product of both weights, i.e.

$$\gamma_{comb} = \gamma_E \cdot \gamma_{drop}, \quad \text{conditioned by } 0.01 < \gamma_{comb} \tag{338}$$

5.1.13.6.4 State machine and final speech/music decision

The state machine is an event-based decision system in which the state of the speech/music classifier is changed from INACTIVE to ACTIVE and vice-versa. There are two intermediate states: ENTRY and INSTABLE. The classifier must always go through the ENTRY state in order to be ACTIVE. During the ENTRY period, there is no relevant past information that could be exploited by the algorithm for hangover additional described later in this section. When the classifier is in the ACTIVE state but the energy of the input signal goes down up to the point when it is almost equal to the estimated background noise energy level, the classifier is in an UNSTABLE state. Finally, when SAD goes to 0, the classifier is in INACTIVE state. The following state-flow diagram shows the transitions between the states.
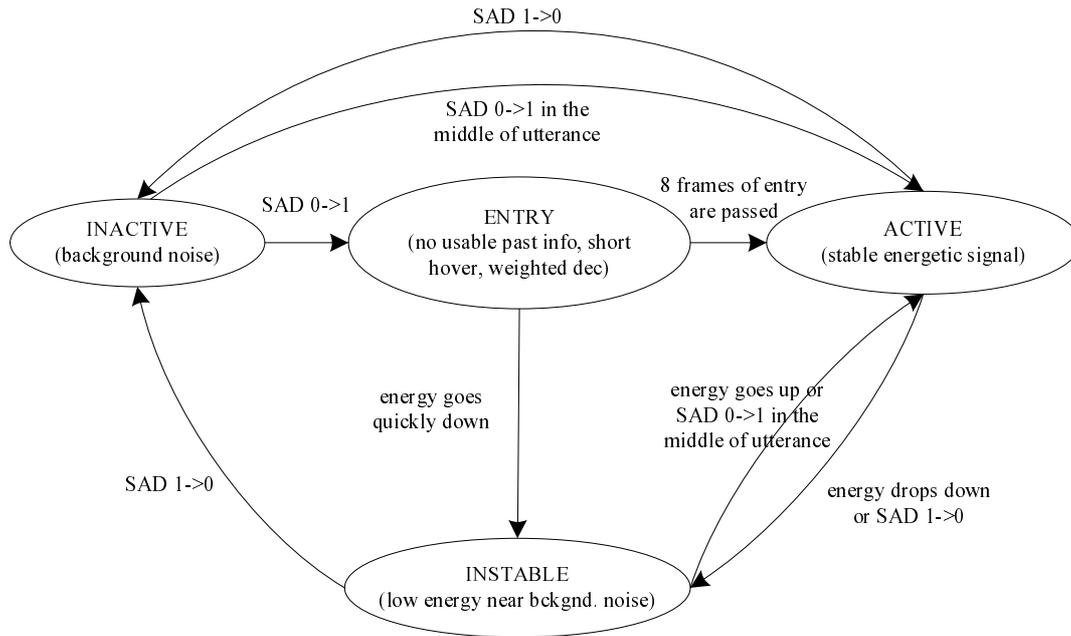
Figure 16 : State machine for the first stage of the speech/music classifier

The conditions for changing the states are described in from of decision tree in figure 17. The processing starts at the top-left corner and stops at bottom-right corner. The counter of inactive states c_{inact} is initialized to 0 and the state variable sm_{state} is initialized to -8. The state variable stays within the range [-8;+8] where the value of -8 means INACTIVE state and the value of +8 means ACTIVE STATE. If $0 < sm_{state} < 8$ the classifier is in ENTRY state and if $-8 < sm_{state} \leq 0$ the classifier is in INSTABLE state.

If the speech/music classifier is in INACTIVE state, i.e. if $sm_{state} = -8$ then the smoothed decision is automatically set to 0, i.e. $wdL_{sm} = 0$.

The final decision of the speech/music classifier is binary and it is characterized by the flag f_{SM}. The flag is set to 0 if $f_{LSAD\_HE} = 0$ and the classifier is in INACTIVE STATE, i.e. when $sm_{state} = -8$. If there is a transition from the ACTIVE state to the INACTIVE or INSTABLE state, characterized by $sm_{state} \leq 0$, the flag retains its value from the previous frame. If the classifier is in ENTRY state, characterized by $0 < sm_{state} < 8$, the flag is set according to weighted average of past non-binary decisions. This is done as follows

$$dec = \sum_{j=1}^{8} g_{entry}(sm_{state}, j)dL_{sm}^{[j-1]} \tag{339}$$

where the weighting coefficients $g_{entry}(i, j)$ are given in the following table:

Table 18: Weighting coefficients for the ENTRY period of the speech/music classifier

$g_{entry}(i, j)$	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	0.6	0.4	0	0	0	0	0	0
3	0.47	0.33	0.2	0	0	0	0	0
4	0.4	0.3	0.2	0.1	0	0	0	0
5	0.3	0.25	0.2	0.15	0.1	0	0	0
6	0.233	0.207	0.18	0.153	0.127	0.1	0	0
7	0.235	0.205	0.174	0.143	0.112	0.081	0.05	0
8	0.2	0.179	0.157	0.136	0.114	0.093	0.071	0.05
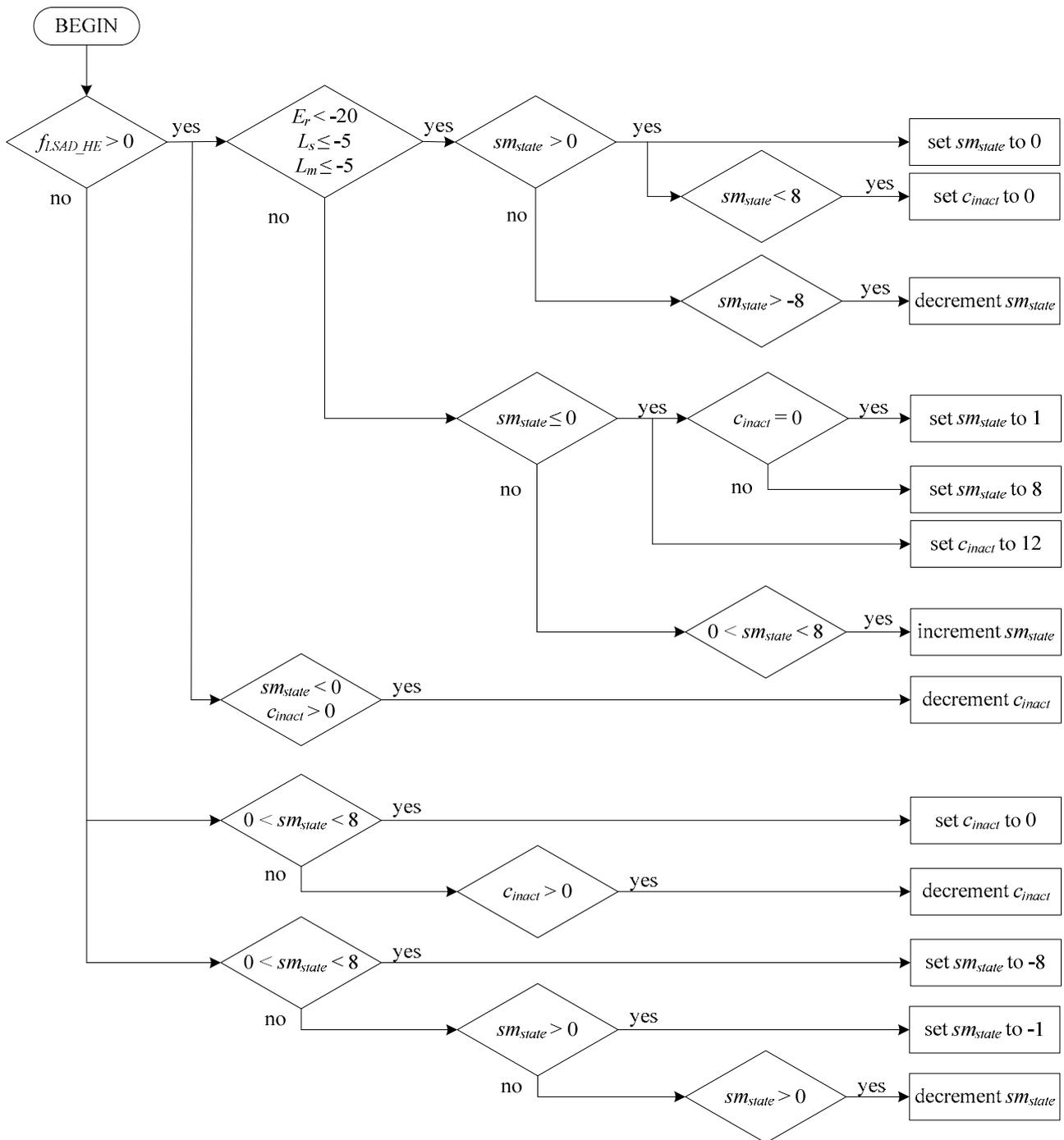
Figure 17 : Decision tree for transitions between INACTIVE and ACTIVE states of the speech/music classifier

The flag f_{SM} is set to 1 when $dec > 2$. If the classifier is in a stable ACTIVE state, the flag retains its value from the previous frame unless one of the following two situations happens. If $wdL_{sm} > 0$ but the decisions in the previous three frames were all "speech", i.e. $f_{SM}^{[i]} = 0$ for i=-1,-2,-3, there is a transition from speech to music and f_{SM} is set to 1. If $wdL_{sm} < 0$ but the decision in the previous frame was "music", i.e. $f_{SM}^{[-1]} = 1$, there is a transition from music to speech and f_{SM} is set to 0.

The speech/music decision obtained by the algorithm described so far will be denoted f_{SM1} in the following text to distinguish it from the second stage of the speech/music classifier in which the decision will be denoted f_{SM2}.

5.1.13.6.5 Improvement of the classification for mixed and music content

The speech/music decision f_{SM1} obtained above is further refined with the goal of improving the classification rate on music and mixed content. A set of feature parameters are extracted from the input signal and buffered. Statistical analysis is performed on each feature parameter buffer and a binary speech/music decision $f_{SM'}$ is obtained using a tree-based classification. During the processing the value '1' indicates music and the value '0' indicates non-music. As a result of this refinement, the earlier speech/music decision f_{SM1} may be adjusted from '0' to '1' if $f_{SM'}$ has a final value of '1' in the situation that the f_{SM1} and $f_{SM'}$ are not aligned with each other.

The feature parameters used to form the feature parameter buffers include a spectral energy fluctuation parameter, $flux$, a tilt parameter of the LP analysis residual energies, $tilt_R$, a high-band spectral peakiness parameter, pk_h, a parameter of correlation map sum, cor_{sum}, a voicing parameter, vm, and finally three tonal parameters NT, $NT2$ and NT_l. Since music is assumed to only existing during high SNR active regions of the input signal, the classification refinement is only applied for active frames and when the long-term SNR is high. So, if the SAD flag f_{LSAD} indicates that the current frame is an inactive frame or the long-term SNR SNR_{LT} is below a threshold of 25, i.e. $if(f_{LSAD}==0 \| SNR_{LT}<25)$, then the classification refinement is terminated without executing fully. In the early termination case, the speech/music decision f_{SM1} is kept unchanged and two long-term speech/music decisions $LTf_{SM'}$, $LTf'_{SM'}$, as will be described later in this subclause, are both initialized to 0.5.

Before computing the various feature parameters, percussive music is first detected. Percussive music is characterized by temporal spike-like signals. First the log maximum amplitude of the current frame is found as

$$Am_{\max} = 20\log_2 \max(|s(i)|), \qquad i = 0,1,...,255 \tag{340}$$

where $s(i)$ is the time-domain input frame. The difference between the log maximum amplitude and its moving average from the previous frame is calculated

$$D_{ltA} = Am_{\max} - \overline{Am_{\max}}^{[-1]} \tag{341}$$

where the superscript [-1] denotes the value from the previous frame. $\overline{Am_{\max}}$ is updated at each frame after the calculation of D_{ltA}, if both the normalized pitch correlations of the current frame $C_{norm}^{[0]}$ and $C_{norm}^{[1]}$ as calculated in defined in subclause 5.1.11.3.2 are greater than 0.9 as

$$\overline{Am_{\max}} = \alpha \cdot \overline{Am_{\max}}^{[-1]} + (1-\alpha) \cdot Am_{\max} \tag{342}$$

where the value α is the forgetting factor and is set to 0.75 for increasing updates ($Am_{\max} > \overline{Am_{\max}}^{[-1]}$) and 0.995 for decreasing updates ($Am_{\max} < \overline{Am_{\max}}^{[-1]}$). The D_{ltA}, the total frame energy E_t calculated in subclause 5.1.5.2, the normalized pitch correlation $C_{norm}^{[i]}$ defined in subclause 5.1.11.3.2 and the long-term active signal energy $\overline{E}_t$ are used to identify the temporal spike-like signals. First, certain energy relationship between several past frames is checked. If $E_t^{[-2]} - E_t^{[-3]} > 6$ and $E_t^{[-2]} > E_t^{[-1]}$ and $E_t^{[-2]} - E_t^{[0]} > 3$ and $E_t^{[-2]} - \overline{E}_t > 3$ and $E_t^{[-1]} > E_t^{[0]}$, where the superscript $[-i]$ denotes the i-th frame in the past, the energy envelope of temporal spike-like signal is considered found. Then if the voicing is low, that is if $0.5C_{norm}^{[0]} + 0.25C_{norm}^{[1]} + 0.25C_{norm}^{[2]} < 0.75$, the percussive music flag f_{pc} is set to 1 indicating the detection of spike-like signal, if the the normalized pitch correlations for the second half of the previous frame, the first half of the current frame and the second half of the current frame, $C_{norm}^{[0]}$, $C_{norm}^{[1]}$ and $C_{norm}^{[2]}$ are all less than 0.75 and the D_{ltA} is greater than 10, or if simply the long-term speech/music decision $LTf_{SM'}$ is greater than 0.8.

Besides the detection of percussive music, sound attacks are also detected using $E_t^{[i]}$, $\overline{E}_t$, $LTf_{SM'}$ and $D_{ltA}^{[-1]}$, where $D_{ltA}^{[-1]}$ denotes the D_{ltA} of the previous frame. If $E_t^{[0]} - E_t^{[-1]} > 6$ and $E_t^{[0]} - \overline{E}_t > 5$ and $LTf_{SM'} > 0.9$ and $D_{ltA}^{[-1]} > 5$, sound attack is detected and the attack flag f_{att} is set to 3. The attack flag f_{att} is decremeted by 1 in each frame afterthe calculation and buffering of the spectral energy fluctuation parameter $flux$ which is claculated from the log

energy spectrum of the current frame as follows: Firstly, all local peaks and valleys in the log spectrum $E_{dB}(k), k = 0,1,...126$, as calculated in equation (127) are identified. A value of $E_{dB}(k)$ is considered as a local peak if $E_{dB}(k) > E_{dB}(k-1)$ and $E_{dB}(k) > E_{dB}(k+1)$. A value of $E_{dB}(k)$ is considered as a local valley if $E_{dB}(k) < E_{dB}(k-1)$ and $E_{dB}(k) < E_{dB}(k+1)$. Besides, the first local valley is found as the $E_{dB}(k)$ with $E_{dB}(k) < E_{dB}(k+1)$ and $E_{dB}(k) \leq \forall E_{dB}(i), i = 0,...,k-1$, the last local valley is found as the $E_{dB}(k)$ with $E_{dB}(k) < E_{dB}(k-1)$ and $E_{dB}(k) \leq \forall E_{dB}(i), i = k+1,...,126$. For each local peak, its peak to valley distance is calculated as

$$p2v(i) = E_p(i) - E_{vl}(i) + E_p(i) - E_{vh}(i), \qquad i = 0,1,...,m-1 \qquad (343)$$

where $p2v(i)$ denotes the peak to valley distance of the i-th local peak, $E_p(i)$ denotes the log energy of the i-th local peak and $E_{vl}(i)$, $E_{vh}(i)$ denote the respect log energy of the local valleys adjacent to the i-th local peak at the lower frequency side and the higher frequency side, m denotes the number of local peaks. An array called peak to valley distance map is then obtained as

$$MAP_{p2v}(k) = \begin{cases} p2v(i) & if \ k = idx_P(i) \\ 0 & otherwise \end{cases}, \qquad k = 0,1,...,126 \qquad (344)$$

where $MAP_{p2v}(k)$ denotes the peak to valley distance map, $idx_P(i)$ denotes the index (or the location) of the i-th local peak in the log spectrum $E_{dB}(k)$. The spectral energy fluctuation parameter $flux$ is defined as the average energy deviation between the current frame spectrum and the spectrum two frames ago at locations of the local spectral peaks $idx_P(i)$. The $flux$ is computed as

$$flux = \frac{1}{m} \cdot \sum_{k=0, MAP_{p2v}(k) \neq 0}^{127} \left| E_{dB}^{[0]}(k) - E_{dB}^{[-2]}(k) \right| \qquad (345)$$

where $E_{dB}^{[0]}(k)$ and $E_{dB}^{[-2]}(k)$ denote respectively the log energy spectrum of the current frame and the log energy spectrum of the frame two frames ago, m denotes the number of local peaks. If $m = 0$, $flux$ is set to 5. The computed $flux$ is stored into a buffer $BUF_{flux}(i), i = 0,1...59$ of 60 frames if there is no sound attack in the past 3 frames (including the current frame), that is if $f_{att} \leq 0$. Moreover, if the long-term speech/music decision $LTf_{SM'}$ is greater than 0.8 meaning a strong music signal in previous classifications, then the value of $flux$ is upper limited to 20 before it is stored into the $BUF_{flux}(i)$. The $flux$ buffer $BUF_{flux}(i)$ is altered at every first active frame after an inactive segment (flagged by f_{LSAD}) that all values in the buffer excluding the one just calculated and stored for the current frame are changed to negative values.

The effective portion of the buffer $BUF_{flux}(i)$ is determined in each frame after the calculation and buffering of the parameter $flux$. The effective portion is defined as the portion in the $flux$ buffer $BUF_{flux}(i)$ which contains continuous non-negative values starting from the value of the latest frame. If percussive music is detected, that is if the percussive music flag f_{pc} is set to 1, each value in the effective portion of the $flux$ buffer $BUF_{flux}(i)$ is initialized to 5.

The tilt parameter of the LP analysis residual energies $tilt_R$ is calculated as

$$tilt_R = \frac{\sum_{i=1}^{15} E(i) \cdot E(i+1)}{\sum_{i=1}^{15} E(i) \cdot E(i)} \qquad (346)$$

where $E(i)$ is the LP error energies computed by the Levinson-Durbin algorithm. The computed $tilt_R$ is stored into a buffer $BUF_{tilt}(i), i = 0,1...59$ of 60 frames.

The high-band spectral peakiness parameter pk_h reflects an overall tonality of the current frame at its higher frequency band and is calculated from the peak to valley distance map $MAP_{p2v}(k)$ as

$$pk_h = \sum_{k=64}^{126} MAP_{p2v}(k) \tag{347}$$

The calculated pk_h is stored into a buffer $BUF_{pk_h}(i), i = 0,1...59$ of 60 frames.

The three tonal parameters NT, $NT2$ and NT_l are also calculated from the peak to valley distance map $MAP_{p2v}(k)$. NT denotes the first number of harmonics found from the spectrum of the current frame. NT is calculated as

$$NT = \sum_{k=0}^{126} \left(MAP_{p2v}(k) > 55\right) \tag{348}$$

$NT2$ denotes the second number of harmonics also found from the spectrum of the current frame. $NT2$ is defined more strictly than NT and is calculated as

$$NT2 = \sum_{k=0}^{126} \left(MAP_{p2v}(k) > 80\right) \tag{349}$$

NT_l denotes the number of harmonics found only at the low frequency band of the current frame's spectrum and is calculated as

$$NT_l = \sum_{k=0}^{64} \left(MAP_{p2v}(k) > 80\right) \tag{350}$$

The calculated values of NT, $NT2$ and NT_l are stored into their respective buffers $BUF_{NT}(i)$, $BUF_{NT2}(i)$ and $BUF_{NT_l}(i)$ all of 60 frames.

The sum of correlation map m_{sum} as calculated by

$$m_{sum} = \sum_{j=0}^{127} M_{cor}(j) \tag{351}$$

is also stored into a buffer $BUF_{m_{sum}}(i), i = 0,1...59$ of 60 frames, where $M_{cor}(j)$ is the correlation map calculated in subclause 5.1.11.2.5.

The voicing parameter vm is defined as the difference of log-likelihood between speech class and music class as calculated in subclause 5.1.13.6.3. The vm is calculated as

$$vm = L_s - L_m \tag{352}$$

where L_s, L_m are the log-likelihood of speech class and the log-likelihood of music class respectively. vm is stored into a buffer $BUF_{vm}(i), i = 0,1...9$ of 10 frames.

The speech/music decision $f_{SM'}$ is obtained through a tree-based classification. The $f_{SM'}$ is first initialized as a hysteresis of the long-term speech/music decision $LTf_{SM'}$ from the previous frame, i.e.

$$f_{SM'} = \begin{cases} 1 & if \ LTf_{SM'}^{[-1]} > 0.5 \\ 0 & otherwise \end{cases} \tag{353}$$

where the superscript [-1] denotes the value from the previous frame. Then, the $f_{SM'}$ can be altered through successive classifications. Let LEN denotes the length of the effective portion in $BUF_{flux}(i)$. Depending on the actual value of

LEN, different classification procedures are followed. If $LEN \le 5$, insufficient data is considered in the feature parameter buffers. The classification is terminated and the initialized $f_{SM'}$ is used as the final $f_{SM'}$. If $5 < LEN < 10$, the respective mean values M_{pk_h}, $M_{m_{sum}}$ and M_{NT} are calculated from $BUF_{pk_h}(i)$, $BUF_{cor_{sum}}(i)$ and $BUF_{NT}(i)$ over the effective portion and the variance V_{tilt}, calculated over the effective portion from $BUF_{tilt}(i)$ is also obtained. In addition, the number of positive values N_{vm6} among the 6 latest values in $BUF_{vm}(i)$ is counted. The speech/music decision $f_{SM'}$ is then set to 1 if $N_{vm6} < 4$ and any of the following conditions is fulfilled; $M_{pk_h} > 1100$ or $M_{m_{sum}} > 100$ or $V_{tilt} < 0.00008$ or $M_{NT} > 27$. Otherwise, if $LEN > 10$, the feature buffers are first analysed over the portion POR_{10} containing the latest 10 values. The mean values $M_{flux_{10}}$, $M_{pk_{h10}}$ and $M_{m_{sum10}}$ are calculated from $BUF_{flux}(i)$, $BUF_{pk_h}(i)$ and $BUF_{m_{sum}}(i)$ over POR_{10} and for the same portion the variance $V_{tilt_{10}}$ is also calculated from $BUF_{tilt}(i)$. Besides, the mean value of $BUF_{flux}(i)$ M_{flux_5}, over a shorter portion of the latest 5 frames is also calculated. The N_{vm} is found as the number of positive values in $BUF_{vm}(i)$. The speech/music decision $f_{SM'}$ is determined without the need to analyse any longer portion if strong speech or music characteristics are found within POR_{10}, that is, the $f_{SM'}$ and $LTf_{SM'}$ are both set to 1 if $N_{vm} < 3$ and $M_{flux_5} < 15$ and any of the following conditions is fulfilled: $M_{flux_{10}} < 8.5$ or $M_{pk_{h10}} > 1050$ or $M_{m_{sum10}} > 100$ or $V_{tilt_{10}} < 0.001 \&\& M_{flux_{10}} < 12$. The $f_{SM'}$ and $LTf_{SM'}$ are both set to 0 if any of the following conditions is fulfilled: $M_{flux_{10}} > 16$ or $M_{flux_5} > 19$ or $M_{flux_{10}} > 15 \&\& N_{vm} > 2$ or $BUF_{flux}(59) \ge 20 \&\& vm > 0$. If no class is determined for $f_{SM'}$ over the POR_{10} values, the $f_{SM'}$ is determined iteratively over portions starting from POR_{10} until the whole effective portion is reached. For each iteration, the respective mean values M_{flux}, M_{pk_h} and $M_{m_{sum}}$ are calculated from $BUF_{flux}(i)$, $BUF_{pk_h}(i)$ and $BUF_{cor_{sum}}(i)$ over the portion under analysis and for the same portion the variance V_{tilt} is also calculated from $BUF_{tilt}(i)$. The mean value $M_{flux_{10}}$ is calculated from $BUF_{flux}(i)$ over POR_{10}, and the number of positive values in $BUF_{vm}(i)$, N_{vm}, is also counted. The value of $f_{SM'}$ is set to 1 if $N_{vm} < 3$ and any of the following conditions is fulfilled: $M_{flux} < 12 + 0.05 \cdot (LEN - 10) \&\& M_{flux_{10}} < 15$ or $V_{tilt} < 0.0001 + 0.000018 \cdot (LEN - 10)$ or $M_{pk_h} > 1050 - 5 \cdot (LEN - 10)$ or $M_{m_{sum}} > 95 - 0.3 \cdot (LEN - 10)$. If through the above iteration procedure the $f_{SM'}$ is not set and if the effective portion reaches the maximum of 60 frames, a final speech/music discrimination is made from $BUF_{NT}(i)$, $BUF_{NT_l}(i)$ and $BUF_{NT2}(i)$. The mean value M_{NT} of the $BUF_{NT}(i)$, the sum value S_{NT_l} of the $BUF_{NT_l}(i)$, and the sum value M_{NT2} of the $BUF_{NT2}(i)$ are calculated over the whole buffers. A low frequency tonal ratio R_{NT_l} is calculated as

$$R_{NT_l} = \frac{S_{NT_l}}{S_{NT2}} \tag{354}$$

The $f_{SM'}$ is set to 1 if $M_{NT} > 18$ or $R_{NT_l} < 0.2$. Otherwise, if $M_{NT} < 1$, the $f_{SM'}$ is set to 0.

If LEN is greater than 30, then both the two long-term speech/music decisions $LTf_{SM'}$ and $LTf'_{SM'}$ are updated at each frame with $f_{SM'}$ as

$$LTf_{SM'} = 0.97 \cdot LTf_{SM'}^{[-1]} + (1 - 0.97) \cdot f_{SM'} \tag{355}$$

$$LTf'_{SM'} = 0.97 \cdot LTf'^{[-1]}_{SM'} + (1 - 0.97) \cdot f_{SM'} \tag{356}$$

where the superscript [-1] denotes the value from the previous frame. If the total frame energy E_t calculated in subclause 5.1.5.2 is greater than 1.5 and $BUF_{NT2}(59)$ is less than 2 and the raw coding mode CT_{raw} is either UNVOICED or INACTIVE, then an unvoiced counter CT_{uv} initialized to 300 at the first frame is updated by

$$CT_{uv} = CT_{uv} - 8 \tag{357}$$

Otherwise, CT_{uv} is incremented by 1. The value of CT_{uv} is bounded between [0, 300]. The CT_{uv} is further smoothed by an AR filtering as

$$LT_{CT_{uv}} = 0.9 \cdot LT_{CT_{uv}}^{[-1]} + 0.1 \cdot CT_{uv} \tag{358}$$

where $LT_{CT_{uv}}$ is the smoothed CT_{uv}, the superscript [-1] denotes the value from the previous frame. If $f_{SM'}$ is set to 1 in any previous stage, the flag f_{SM1} is overridden by $f_{SM'}$ unless the long-term speech/music decision $LTf'_{SM'}$ as calculated in equation (356) is close to speech and the smoothed unvoiced counter $LT_{CT_{uv}}$ exhibits strong unvoiced characteristic, that is, the f_{SM1} is set to 1 if $f_{SM'} = 1$ and $LTf'_{SM'} \geq 0.2$ *or* $LT_{CT_{uv}} \geq 200$.

5.1.13.6.6   Second stage of the speech/music classifier

The second stage of the speech/music classifier has been designed and optimized for the GSC technology. Not all frames classified as music in the first stage can be encoded directly with the GSC technology due to its inherent limitations. Therefore, in the second stage of the speech/music classifier, a subset of frames that have been previously classified as "music", i.e. for which $f_{SM1} = 1$, are reverted to speech and encoded with one of the CELP modes. The decision in the second stage of the speech/music classifier is denoted f_{SM2}. The second stage is run only for WB, SWB and FB signals, not for NB. The reason for this limitation is purely due to the fact that GSC technology is only applied at bandwidths higher than NB.

The second stage of the speech/music classifier starts with signal stability estimation which is based on frame-to-frame difference of the total energy of the input signal. That is

$$dE_t(i) = E_t^{[-i]} - E_t^{[-i-1]}, \quad \text{for } i = 1,..,40 \tag{359}$$

Then, the mean energy difference is calculated as

$$\overline{dE_t} = \frac{1}{40} \sum_{i=1}^{40} dE_t(i) \tag{360}$$

i.e. over the period of the last 40 frames. Then, the statistical deviation of the delta-energy values around this mean is calculated as

$$E_{dev} = \sqrt{\frac{1}{15} \sum_{i=1}^{15} \left(dE_t(i) - \overline{dE_t}\right)^2} \tag{361}$$

i.e. over the period of the last 15 frames.

After signal stability estimation, correlation variance is calculated as follows. First, mean correlation is estimated over the period of the last 10 frames. This is done as

$$\overline{C}_{norm2} = \frac{1}{10} \sum_{i=1}^{10} C_{norm2}^{[-i]} \tag{362}$$

where $C_{norm2} = 0.5 C_{norm}^{[0]} + 0.5 C_{norm}^{[1]} + r_e$ and the superscript [-1] is used to denote past frames. Then, the correlation variance is defined as

$$E_{dev} = \frac{1}{10} \sum_{i=1}^{10} \left(C_{norm2}^{[-i]} - \overline{C}_{norm2}\right)^2 \tag{363}$$

In order to discriminate highly-correlated stable frames, long-term correlation is calculated as

$$\tilde{C}_{norm2} = 0.9 \tilde{C}_{norm2}^{[-1]} + 0.1 C_{norm2}^{[-1]} \tag{364}$$

The flag f_{hscor} is set to 1 if $\tilde{C}_{norm2}^{[-1]} > 0.8$ and at the same time $E_{dev} < 0.0005$.

In the next step, attacks are detected in the inputs signal. This is done by dividing the current frame of the input signal into 32 segments where each segment has the length of 8 samples. Then, energy is calculated in each segment as

$$E_{seg}(k) = \sum_{i=0}^{7} s_{pre}^2(8k+i), \quad \text{for } k = 0,..,31 \tag{365}$$

The segment with the maximum energy is then found by

$$k_{att} = \max_{k}\left(E_{seg}(k)\right) \tag{366}$$

and this is the position of the candidate attack. In all active frames where $f_{LSAD} > 0$ and for which the coding mode was set to GC, the following logic is executed to eliminate false attacks, i.e. attacks that are not sufficiently strong. First, the mean energy in the first 3 sub-frames is calculated as

$$E_{SF3} = \frac{1}{24} \sum_{k=0}^{23} E_{seg}(k) \tag{367}$$

and the mean energy after the detected candidate attack is defined

$$E_{after} = \frac{1}{32 - k_{att}} \sum_{k=k_{att}}^{31} E_{seg}(k) \tag{368}$$

and the ratio of these two energies is compared to a certain threshold. That is

$$\text{if } \frac{E_{after}}{E_{SF3}} < 8 \text{ then } k_{att} = 0 \tag{369}$$

Thus, the candidate attack position is set to 0 if the attack is not sufficiently strong. Further, if the FEC class of the last frame was VOICED CLASS and if $E_{after}/E_{SF3} < 20$ then k_{att} is also set to 0.

To further reduce the number of falsely detected attacks, the segment with maximum energy is compared to other segments. This comparison is done regardless of the selected coding mode and f_{LSAD}.

$$\text{if } \frac{E_{seg}(k_{att})}{E_{seg}(k)} < 1.3 \text{ then } k_{att} = 0 \quad \text{for } k = 2,..,21, k \neq k_{att} \tag{370}$$

Thus, if the energy in any of the above defined segments, other than k_{att}, is close to that of the candidate attack, the attack is eliminated by setting k_{att} to 0.

Initially the speech/music decision in the second stage is set equal to the speech/music decision from the first stage, i.e. $f_{SM2} = f_{SM1}$. In case the decision is "music", it could be reverted to "speech" in the following situations.

The decision is reverted from music to speech for highly correlated stable signals with higher pitch period. These signals are characterized by

$$\text{if } f_{hscor} > 0 \text{ AND } T_{OL}^{[0]} > 130 \text{ then } f_{SM2} = 0 \tag{371}$$

Further, if the above condition is fulfilled and the selected coding mode was TC, it is changed to GC. This is to avoid any transition artefacts during stable harmonic signal.

In case there is an energetic event characterized by $dE_t(1) > 4.5$ and at the same time $dE_t(1) - dE_t(2) > 10$ it could mean that an attack has occurred in the input signal and the following logic takes place. If, in this situation, the counter of frames from the last detected onset/transition i_{TC}, described in subclause 5.1.13.4, has been set to 1 the attack is confirmed and the decision is changed to speech, i.e. $f_{SM2} = 0$. Also, the coding mode is changed to TC. Otherwise, if

there has been an attack found by the attack tracking algorithm described above, and the position of this attack is beyond the end of the third sub-frame, the decision is also changed to speech and the coding mode is changed to TC. That is

$$\text{if } k_{att} \geq 24 \text{ then } f_{SM2} = 0 \tag{372}$$

Furthermore, an attack flag f_{att} is set to 1 if the detected attack is located after the first quarter of the first sub-frame, i.e. when $k_{att} \geq 4$. This flag is later used by the GSC technology. Finally, the attack flag f_{att} is set to 1 in all active frames ($f_{LSAD} = 1$) that have been selected for GC coding and for which the decision in the first stage of the speech/music classifier was "speech". However, it is restricted only to frames in which the attack is located in the fourth subframe. In this case, the coding mode is also changed to TC for better representation of the attack.

As previously described, if f_{spitch} =flag_spitch=1, VC mode is maintained and AC mode is set to 0; that is,

```
if ( f_spitch =1  and  sampling rate = 16kHz and  bit rate < 13.2kbps )
{
    f_SM2 =0;
}
```

5.1.13.6.7 Context-based improvement of the classification for stable tonal signals

By using context-based improvement of the classification, an error in the classification in the previous stage can be corrected. If the current frame has been provisionally classified as "speech", the classification result can be corrected to "music", and vice versa. To determine a possible error in the current frame, the values of 8 consecutive frames including the current frame are considered for some features.

Figure 18 shows the multiple coding mode signal classification method. If the current frame has been provisionally classified as "speech" after the first- and second-stage classification, then the frame is encoded using the CELP-based coding. On the other hand, if the current frame is initially classified as "music" after the first- and second-stage classification, then the frame is further analysed for fine-classification of "speech" or "music" to select either the GSC-based coding or MDCT-based transform coding, respectively. The parameters used to perform the fine-classification in multiple coding mode selection include:

- Tonality

- Voicing

- Modified correlation

- Pitch gain, and

- Pitch difference

The tonality in the sub-bands of 0-1kHz, 1-2 kHz, and 2-4 kHz are estimated as $tonality1$, $tonality2$, and $tonality3$ as follows:

$$tonality1 = \frac{\max_{k=[0,1,...19]}\big(PS(k)\big)}{\sum_{k=0}^{19} PS(k)}$$

$$tonality2 = \frac{\max_{k=[20,21,...39]}\big(PS(k)\big)}{\sum_{k=20}^{39} PS(k)}$$

$$tonality3 = \frac{\max\limits_{k=[40,41,...79]}(PS(k))}{\sum\limits_{k=40}^{79} PS(k)}$$

where PS is the power spectrum. The maximum tonality, $tonality$, is estimated as,

$$tonality = \max(tonality1, tonality2, tonality3)$$

The voicing feature, $voicing$, is the same one as used in the unvoiced signal classification. See equation (237) in subclause 5.1.13.1.1. for the details of its computation. The voicing feature from the first analysis window is used, i.e.

$$voicing = C_{norm}^{[0]}$$

The modified correlation, old_corr, is the normalized correlation from the previous frame.

The pitch gain, $lowrate\_pitchGain$, is the smoothed closed-loop pitch gain estimated from the previous frame, i.e.,

$$lowrate\_pitchGain = 0.9 * lowrate\_pitchGain + 0.1 * gain\_pit$$

where $gain\_pit$ is the ACB gain in each of the sub-frames from the previous frame.

The pitch deviation is estimated as the sum of pitch differences between the current frame open-loop pitch, T_{op}^{n} and the open loop pitch in the previous three frames, $T_{op}^{n-1}, T_{op}^{n-2}, T_{op}^{n-3}$

$$pitch\_diff = \sum_{i=1}^{3} \left| T_{op}^{n-i} - T_{op}^{n} \right|$$

The features, $voicing$, old_corr, and $tonality$ are smoothed to minimize spurious instantaneous variations as follows:

$$lt\_voicing = \alpha_{1c} * lt\_voicing + (1 - \alpha_{1c}) * voicing$$

$$lt\_corr = \alpha_{2c} * lt\_corr + (1 - \alpha_{2c}) * old\_corr$$

$$lt\_tonality = \alpha_{3c} * lt\_tonality + (1 - \alpha_{3c}) * tonality$$

where α_{1c} and α_{2c} are 0.1 in active frames (i.e., SAD = 1), and 0.7 in background noise and inactive frames. Similarly, α_{3c} is 0.1 in active frames and 0.5 in inactive frames.
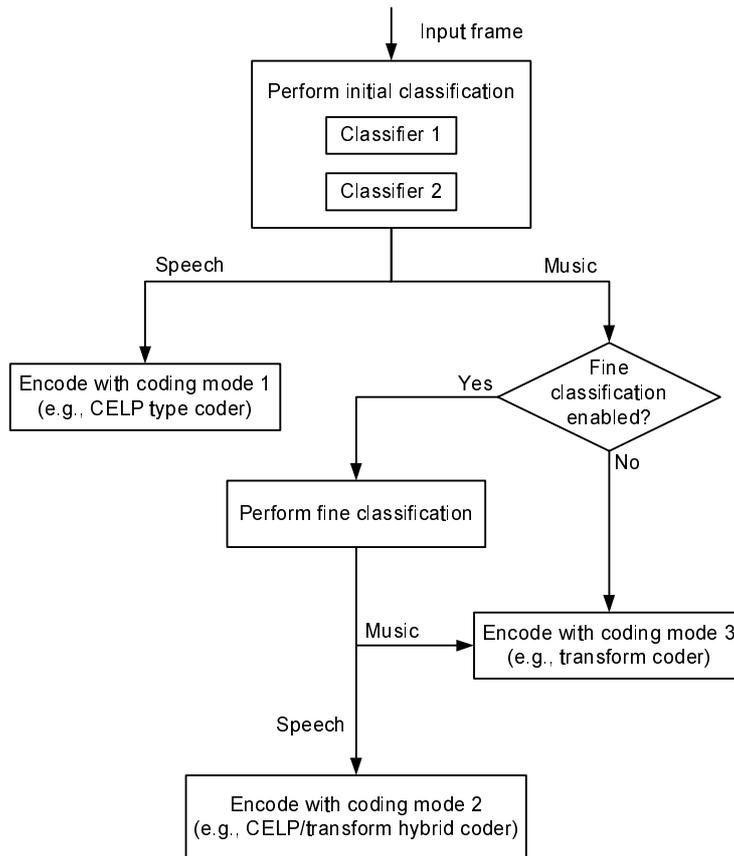
Figure 18 : Multiple coding mode signal classification

The following condition is evaluated to select the GSC or MDCT based coding,

$$\text{if } f_{SM1} == TRUE \text{ AND } \min(tonality1, tonality2, tonality3) > 50.0 \text{ AND}$$

$$\sum_{i=1,2} tonalityi > 200.0 \text{ AND} \sum_{i=2,3} tonalityi > 200.0 \text{ AND} \sum_{i=1,3} tonalityi > 200.0 \text{ AND } lt\_tonality < 20000.0 \text{ AND}$$

$$\left(\begin{array}{l} \left(lt\_tonality > 1000.0 \text{ AND } \max(lt\_voicing, voicing) > 0.99 \right) \\ \text{OR } \left(lt\_tonality > 1500.0 \text{ AND } lt\_corr \right) > 0.99 \\ \text{OR } \left(lt\_tonality > 3000.0 \text{ AND } lt\_lowrate\_pitchGain \right) > 0.96 \\ \text{OR } \left(lt\_pitch\_diff > 0 \text{ AND } lt\_lowrate\_pitchGain \right) > 0.89 \end{array} \right)$$

then $f_{SM1} = FALSE$, $f_{SM2} = FALSE$

A hangover logic is used to prevent frequent switching between coding modes of GSC and MDCT-based coding. A hangover period of 6 frames is used. The coding mode is further modified as per below.

Figure 19 shows two independent state machines, which are defined in the context-based classifier, SPEECH_STATE and MUSIC_STATE. Each state machine has two states . In each state a hangover of 6 frames is used to prevent frequent transitions. If there is a change of decision within a given state, the hangover in each state is set to 6, and the hangover is then reduced by 1 for each subsequent frame. A state change can only occur once the hangover has been reduced to zero. The following six features are used in the context-based classifier (the superscript [-*i*] is used below to denote the past frames).

The tonality in the region of 1~2 kHz, ton_2 is defined as

$$ton_2 = 0.2 * \log_{10} \left[\sqrt{\frac{1}{8} \sum_{i=0}^{7} \left\{ tonality2^{[-i]} \right\}^2} \right] \tag{373}$$

The tonality in the region of 2~4 kHz, ton_3 is defined as

$$ton_3 = 0.2 * \log_{10}\left[\sqrt{\frac{1}{8}\sum_{i=0}^{7}\left\{tonality3^{[-i]}\right\}^2}\right] \tag{374}$$

The long-term tonality in the low band, ton_{LT} is defined as

$$ton_{LT} = 0.2 * \log_{10}\left[lt\_tonality\right] \tag{375}$$

The difference between the tonality in 1~2 kHz band and the tonality in 2~4 kHz band is defined as

$$d_{ft} = 0.2 * \left\{\log_{10}(tonality2(n)) - \log_{10}(tonality3(n))\right\} \tag{376}$$

The linear prediction error LP_{err} is defined as

$$LP_{err} = \sqrt{\frac{1}{8}\sum_{i=0}^{7}\left\{FV_s^{[-i]}(9)\right\}^2} \tag{377}$$

where $FV_s(9)$ has been defined in equation (327).

The difference between the scaled voicing feature $FV_s(1)$ defined equation (327) and the scaled correlation map feature $FV_s(7)$ defined in equation (327) is defined as

$$d_{vcor} = \max(FV_s(1) - FV_s(7), 0) \tag{378}$$

The following two independent state machines are used to correct errors in the previous stages of the speech/music classification. The are two state machines are called SPEECH_STATE and MUSIC_STATE. There are also two hangover variables denoted $hang_{sp}$ and $hang_{mus}$ which are initialized to the value of 6 frames. The following four conditions are evaluated to determine the transition of one state to another.

Condition A is defined as

if $d_{vcor} > 0.4$ AND $d_{ft} < 0.1$ AND $FV_s(1) > (2*FV_s(7) + 0.12)$ AND $ton_2 < d_{vcor}$ AND

$ton_3 < d_{vcor}$ AND $ton_{LT} < d_{vcor}$ AND $FV_s(7) < d_{vcor}$ AND $FV_s(1) > d_{vcor}$ AND $FV_s(1) > 0.76$ (379)

then $f_A = 1$

Condition B is then defined as

if $d_{vcor} < 0.4$ then $f_B = 1$ (380)

Condition C is defined as

if $0.26 < ton_2 < 0.54$ AND $ton_3 > 0.22$ AND $0.26 < ton_{LT} < 0.54$ AND $LP_{err} > 0.5$

then $f_C = 1$ (381)

and finally condition D is defined as

if $ton_2 < 0.34$ AND $ton_3 < 0.26$ AND $0.26 < ton_{LT} < 0.45$ then $f_D = 1$ (382)
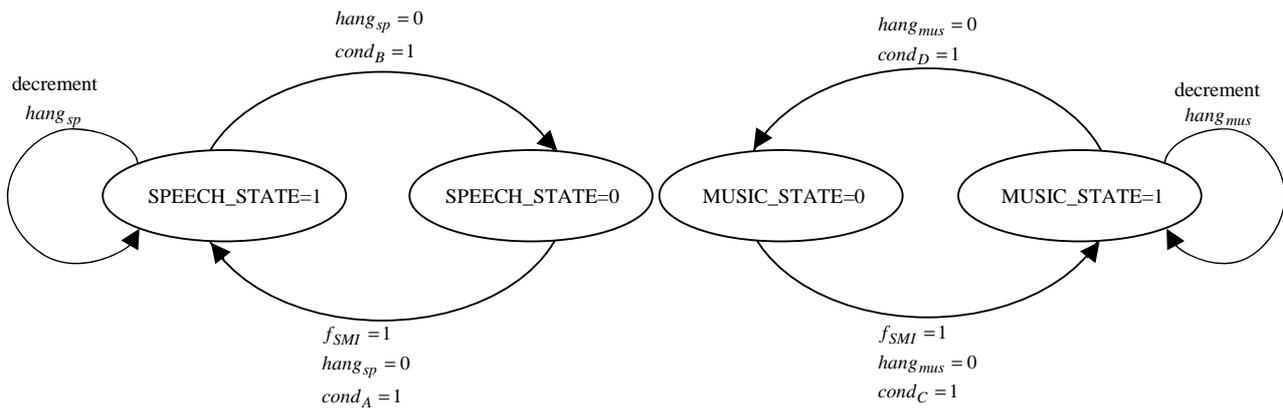
Figure 19 : State machines for context-based speech/music correction

The decisions from the speech/music classifier, f_{SM1} and f_{SM2} are changed to 0 ("speech") if f_{SM1} was previously set to 1 ("music") and if the context-based classifier is in SPEECH_STATE. Similarly, the decisions from the speech/music classifier, f_{SM1} and f_{SM2} are changed to 1 ("music") if f_{SM1} was previously set to 0 ("speech") and if the context-based classifier is in MUSIC_STATE.

5.1.13.6.8   Detection of sparse spectral content

At 13.2kbps, the coding of music signal benefits from combining the advantages of MDCT and GSC technologies. For frames classified as music after the context-based improvement, coding mode producing better quality is selected between MDCT and GSC based on an analysis of signal spectral sparseness and linear prediction efficiency (depending on the input bandwidth).

For each active frame, the sum of the log energy spectrum $PS_{\log}(k), k = 0,1,...,127$ is calculated to determine the spectral sparseness analysis.

$$S_{PS_{\log}} = \sum_{k=0}^{127} PS_{\log}(k) \tag{383}$$

Then the log energy spectrum $PS_{\log}(k)$ is sorted in descending order of magnitude. Each element in the sorted log energy spectrum $PS_{\log\downarrow}(k)$ is accumulated one by one along in descending order until the accumulated value exceeds 75% of the $S_{PS_{\log}}$. The index (or the position), I_{spa}, of the last element added into the accumulation can be regarded as a kind of representation of the spectral sparseness of the frame and is stored into a sparseness buffer $BUF_{spa}(k)$ of 8 frames.

If the input bandwidth is WB, some parameters dedicated to WB are calculated, including the mean of the sparseness buffer, the long-term smoothed sparseness, the high-band log energy sum, the high-band high sparseness flag, the low-band high sparseness flag, the linear prediction efficiency and the voicing metric. For other input bandwidths the above parameters are not calculated. The mean of the sparseness buffer is obtained as

$$M_{spa} = \frac{1}{8} \cdot \sum_{k=0}^{7} BUF_{spa}(k) \tag{384}$$

Then the long-term smoothed sparseness LT_{spa} is calculated as

$$LT_{spa} = 0.75 \cdot LT_{spa}^{[-1]} + 0.25 \cdot M_{4\min} \tag{385}$$

where $LT_{spa}^{[-1]}$ denotes the long-term smoothed sparseness in the previous frame, $M_{4\min}$ denotes the average of the four smallest values in the sparseness buffer $BUF_{spa}(k)$. The reason of using $M_{4\min}$ is to reduce the possible negative

impact to the LT_{spa} from interfering frames. The long-term smoothed sparseness LT_{spa} is initialized to the value of I_{spa} and the sparseness buffer $BUF_{spa}(k)$ is also initialized to the value of I_{spa} for all its elements if the current frame is the first active frame after a pause. The high-band log energy sum is calculated over $PS_{\log}(k), k = 80,81,...,127$

$$S_{PS_{\log}h} = \sum_{k=80}^{127} PS_{\log}(k) \qquad (386)$$

To obtain the high-band high sparseness flag, the high-band log energy spectrum $PS_{\log}(k), k = 80,81,...,127$ is first sorted in descending order. The ratio of the sum of the first 5 elements (or the 5 largest values) of the sorted high-band log energy spectrum to the high-band log energy sum is calculated

$$R_{sph} = \frac{\sum_{k=0}^{4} PS_{\log h\downarrow}(k)}{S_{PS_{\log}h}} \qquad (387)$$

where $PS_{\log h\downarrow}(k), k = 0,1,...,47$ is the sorted high-band log energy spectrum. The ratio R_{sph} can be regarded as a kind of representation of the high-band spectral sparseness of the frame and is stored into a high-band sparseness buffer $BUF_{spah}(k)$. The mean of the buffer $BUF_{spah}(k)$ is calculated. If the mean is greater than 0.2, the high-band high sparseness flag f_{spah} is set to 1 indicating a high sparseness of the high-band spectrum, otherwise set to 0. Similarly, to obtain the low-band high sparseness flag, the low-band log energy spectrum $PS_{\log}(k), k = 0,1,...,59$ is sorted in descending order and the ratio of the sum of the first 5 elements (or the 5 largest values) of the sorted low-band log energy spectrum to the low-band log energy sum is calculated

$$R_{spl} = \frac{\sum_{k=0}^{4} PS_{\log l\downarrow}(k)}{\sum_{k=0}^{59} PS_{\log}(k)} \qquad (388)$$

where $PS_{\log l\downarrow}(k), k = 0,1,...,47$ is the sorted low-band log energy spectrum. The ratio R_{spl} can be regarded as a kind of representation of the low-band sparseness of the frame. If the ratio is greater than 0.18, the low-band high sparseness flag f_{spal} is set to 1 indicating a high sparseness of the low-band spectrum, otherwise set to 0. The LP residual log-energy ratio of the current frame $epsP$, as calculated in subclause 5.1.13.5.1 which is shown again below

$$epsP = \log\left(\frac{E^{[-1]}(13)}{E^{[-1]}(1)}\right) \qquad (389)$$

is stored into a LP residual log-energy ratio buffer $BUF_{epsP}(k)$ of 8 frames. The mean of the buffer $BUF_{epsP}(k)$, M_{epsP}, is calculated and used to represent the short-term linear prediction efficiency at the current frame. The lower the M_{epsP} is the higher the short-term linear prediction efficiency is. The scaled normalized correlation $FV_S(1)$ as calculated in subclause 5.1.13.5.2 is stored into a voicing buffer $BUF_v(k)$ of 8 frames. The mean of the buffer $BUF_v(k)$, M_v, is calculated and used to represent the voicing metric at the current frame.

Decision on which coding mode to use (MDCT or GSC) is made for each frame previously classified as music, that is, for each frame where f_{SM1} is set to 1. GSC coding mode is selected by setting f_{SM2} to 1 and changing f_{SM1} to 0. GSC coding mode is selected for frame with extremely non-sparse spectrum, that is, when I_{spa} is greater than 90. In this case, the GSC hangover flag $f_{H_{GSC}}$ is also set to 1 meaning that a soft hangover period will be applied. Otherwise, if $f_{H_{GSC}}$ is set to 1, the current frame is in a soft hangover period where the determination of extremely non-sparse

spectrum is slightly relaxed, that is, GSC coding mode is selected if I_{spa} is greater than 85. If in above case, I_{spa} is not greater than 85, GSC coding mode is still selected if I_{spa} of the current frame is deviating from the average I_{spa} of its adjacent GSC coded frames by less than 7. A maximum of 7 frames are used for the averaging. The selection between MDCT coding mode and GSC coding mode ends here if the input bandwidth is SWB. For WB input bandwidth, one more step is applied. In this case, GSC coding mode is also selected if the various sparseness measures calculated all do not exhibit strong sparseness characteristics and the linear prediction efficiency is assumed high. Specifically, GSC coding mode is selected if $M_{epsP} < -1.3$ and $M_v > 0.85$ and $I_{spa} > 50$ and $LT_{spa} > 60$ and f_{spal} is set to 0 and $S_{PS_{\log}h} \leq 0.15 * S_{PS_{\log}}$ or if condition $S_{PS_{\log}h} \leq 0.15 * S_{PS_{\log}}$ is not met but f_{spah} is not set to 1. In above case, the GSC hangover flag $f_{H_{GSC}}$ is also set to 1. The flag $f_{H_{GSC}}$ is set to 0 if GSC coding mode is not selected through the whole procedure described above.

5.1.13.6.9 Decision about AC mode

The decisions in the first and in the second stage of the speech/music classifier, refined and corrected by the modules described so far are used to determine the usage of the AC mode. As mentioned before, in the AC mode GSC technology is used to encode the input signal. The decision about the AC mode is done always but the GSC technology is used only at certain bitrates. This is described in subclause 5.1.16.

Before making decision about the AC mode, the speech/music classification results are overridden for certain noisy speech signals. If the level of the background noise is higher than 12dB, i.e. when $\overline{N}_t > 12$, then $f_{SM1} = f_{SM2} = 0$. This is a protection against mis-classification of active noisy speech signals.

For certain unvoiced SWB signals, GSC technology is preferred over the UC or GC mode which would normally be selected. In order to override the selection of the coding mode, there is a flag denoted f_{UV_SWB} which is set to 1 under the following condition

$$\text{if } f_{SAD} = 1 \text{ AND } \overline{N}_t > 12 \text{ AND } f_{SM1} = 0 \text{ AND } BW \geq SWB \text{ AND } c_{raw} = UC \text{ then } f_{UV\_SWB} = 1 \quad (390)$$

where c_{raw} is the raw coding mode, described in subclause 5.1.13.2.

The AC mode is selected if $f_{SM2} = 1$ or if $f_{UV\_SWB} = 1$.

5.1.13.6.10 Decision about IC mode

The IC mode has been designed and optimized for inactive signals which are basically the background noise. Two encoding technologies are used for the encoding of these frames, the GSC and the AVQ. The GSC technology is used at bitrates below 32kbps and the AVQ is used otherwise. The selection of the IC mode at bitrates below 32kbps are conditioned by $f_{SAD} = 1$ whereas for higher bitrates, the condition is changed to $f_{LSAD} = 1$.

The TC mode and the AC mode are not used at 9.6, 16.4 and 24.4 kbps. Thus, at these bitrates, the coding mode is changed to the GC mode if it was previously set to the TC or AC mode. Furthermore, the selection of the IC mode at the previously mentioned bitrates is conditioned only by $f_{SAD} = 1$.

5.1.14 Coder technology selection

Multiple coding technologies are employed within the EVS codec, based on one of the following two generic principles for speech and audio coding, the LP-based (analysis-by-synthesis) approach and the transform-domain (MDCT) approach. There is no clearly defined borderline between the two approaches in the context of this codec. The LP-based coder is essentially based on the CELP technology, optimized and tuned specifically for each bitrate. The transform-domain approach is adopted by the HQ MDCT technology. There are also two hybrid schemes in which both approaches are combined, the GSC technology and the TCX technology. The selection of the coder technology depends on the actual bitrate, the bandwidth, speech/music classification, the selected coding mode and other parameters. The following table shows the allocation of technologies based on bitrate, bandwidth and content.

Table 19: Allocation of coder technologies per bitrate, bandwidth and content

bitrate	7.2	8	9.6	13.2	16.4	24.4	32	48	64
NB									
speech	ACELP	ACELP	ACELP	ACELP	ACELP	ACELP			
audio	HQ MDCT	HQ MDCT	TCX	TCX/HQ MDCT	TCX/HQ MDCT	TCX			
noise	GSC	GSC	TCX	GSC	TCX	TCX			
WB									
speech	ACELP	ACELP	ACELP	ACELP	ACELP	ACELP	ACELP	TCX	ACELP
audio	GSC	GSC	TCX	GSC/TCX/HQ MDCT	TCX/HQ MDCT	TCX	HQ MDCT	TCX	HQ MDCT
noise	GSC	GSC	TCX	GSC	TCX	TCX	ACELP	TCX	ACELP
SWB									
speech				ACELP	ACELP	ACELP	ACELP	TCX	ACELP
audio				GSC/TCX/HQ MDCT	TCX/HQ MDCT	TCX/HQ MDCT	TCX/HQ MDCT	TCX	HQ MDCT
noise				GSC	TCX	TCX	ACELP	TCX	ACELP
FB									
speech					ACELP	ACELP	ACELP	TCX	ACELP
audio					TCX	TCX/HQ MDCT	TCX/HQ MDCT	TCX	HQ MDCT
noise					TCX	TCX	ACELP	TCX	ACELP

The TCX technology is used for any content at bitrates higher than 64 kbps.

At 9.6kbps, 16.4kbps and 24.4kbps a specific technology selector is used to select either ACELP or an MDCT-based technology (HQ MDCT or TCX). This selector is described in clause 5.1.14.1.

At all other bitrates, the division into "speech", "audio" and background "noise" is based on the decision of the SAD and on the decision of the speech/music classifier.

The decision between the TCX technology and the HQ MDCT technology is done adaptively on a frame-by-frame basis. There are two selectors, one for 13.2 and 16.4 kbps and the second for 24.4 and 32 kbps. There is no adaptive selection beyond these bitrates as shown in the above table. These two selectors are described in detail in the subclauses 5.1.14.2 and 5.1.14.3.

5.1.14.1 ACELP/MDCT-based technology selection at 9.6kbps, 16.4 and 24.4 kbps

At 9.6kbps, 16.4kbps and 24.4kbps the decision to choose either ACELP or an MDCT-based technology is not based on the decision of the speech/music classifier as it is done for other bitrates, but on a specific technology selector described below.

The technology selector is based on two estimates of the segmental SNR, one estimate corresponding to the transform-based technology (described in subclause 5.1.14.1.1), another estimate corresponding to the ACELP technology (described in subclause 5.1.14.1.2). Based on these two estimates and on a hysteresis mechanism, a decision is taken (described in subclause 5.1.14.1.3).

5.1.14.1.1 Segmental SNR estimation of the MDCT-based technology

The segmental SNR estimation of the TCX technology is based on a simplified TCX encoder. The input audio signal is first filtered using a LTP filter, then windowed and transformed using a MDCT, the MDCT spectrum is then shaped using weighted LPC, a global gain is then estimated, and finally the segmental SNR is derived from the global gain. All these steps are described in detail in the following clauses.

5.1.14.1.1.1 Long term prediction (LTP) filtering

The LTP filter parameters (pitch lag and gain) are first estimated. The LTP parameters are not only used for filtering the audio input signal for estimating the segmental SNR of the transform-based technology. The LTP parameters are also encoded into the bitstream in case the TCX coding mode is selected, such that the TCX LTP postfilter described in subclause 6.9.2.2 can use them. Note that the LTP filter parameter estimation is also performed at 48kbps, 96kbps and 128kbps even though the parameters are not used to filter the audio input signal in this case.

A pitch lag with fractional sample resolution is determined, using the open-loop pitch lag $T_{OL}^{[1]}$ and an interpolated autocorrelation. The LTP pitch lag has a minimum value of $p_{\min}$, a maximum value of $p_{\max}$ and a fractional pitch resolution p_{res}. Additionally, two thresholds p_{fr1} and p_{fr2} are used. If the pitch lag is less than p_{fr2}, the full fractional precision p_{res} is used. If the pitch lag is greater than p_{fr1}, no fractional lag is used. For pitch lags in between, half of the fractional precision p_{res} is used. These parameters depend on the bitrate and are given in the table below.

Table 20: LTP parameters vs bitrate

Bitrate	Bandwidth	LTP sampling rate	LTP frame length N_{LTP}	p_{res}	$p_{\min}$	$p_{\max}$	p_{fr1}	p_{fr2}
9.6 kbps	NB, WB, SWB	12.8kHz	256	4	29	231	154	121
16.4-24.4 kbps	NB	12.8kHz	256	4	29	231	154	121
16.4-24.4 kbps	WB, SWB, FB	16kHz	320	6	36	289	165	36
48 kbps	WB, SWB, FB	25.6kHz	512	4	58	463	164	58
96-128 kbps	WB, SWB, FB	32kHz	640	6	72	577	75	72

First the parameter δ is initialized depending on the fractional pitch resolution:

$$\delta = \begin{cases} 8 & ,if\ p_{res} = 6 \\ 16 & ,if\ p_{res} = 4 \end{cases}. \tag{391}$$

Then the search range for the pitch lag is determined as follows:

$$t_{\min} = \min\left(\max\left(T_{OL}^{[1]} - \frac{\delta}{2}, p_{\min} \right), p_{\max} - \delta + 1 \right)$$
$$t_{\max} = \max\left(\min\left(T_{OL}^{[1]} + \frac{\delta}{2} - 1, p_{\max} \right), p_{\min} + \delta - 1 \right). \tag{392}$$

For the search range, autocorrelation of the weighted input signal s_h (including the look-ahead part) is computed (note that at 48kbps, 96kbps and 128kbps, the weighted input signal is not available, so the non-weighted input signal is used instead), extended by 4 additional samples in both directions required for subsequent interpolation filtering:

$$C(d) = \sum_{i=0}^{N_{TCX}-1} s_h(i)s_h(i-d), d = (t_{\min} - 4)..(t_{\max} + 4). \tag{393}$$

Within the search range, the index and value of the maximum correlation are determined:

$$C_{LTP} = C(t_{\min})$$
$$d_{LTP} = t_{\min}$$

$$for\ d = t_{\min}..t_{\max}$$
$$if\ (C(d) > C_{LTP})\ then$$
$$C_{LTP} = C(d)$$
$$d_{LTP} = d \tag{394}$$

The maximum correlation value is normalized as follows:

$$C_{LTP,norm} = \frac{C_{LTP}}{\sqrt{\left(\sum_{i=0}^{N_{TCX}-1} s_h^2(i)\right)\left(\sum_{i=0}^{N_{TCX}-1} s_h^2(i-d_{max})\right)+0.1}} \, . \tag{395}$$

The fractional precision of the transmitted pitch lag is determined by the initial pitch lag d_{LTP}, the maximum fractional resolution p_{res}, and the thresholds p_{fr1} and p_{fr2}.

$$s = \begin{cases} 1 & , \quad if \ d_{LTP} < p_{fr2} \\ 2 & , \quad if \ (d_{LTP} \geq p_{fr2}) \wedge (d_{LTP} < p_{fr1}) \\ p_{res} & , \quad if \ d_{LTP} \geq p_{fr1} \end{cases} \tag{396}$$

For determining fractional pitch lag the autocorrelation C is interpolated around the maximum value by FIR filtering:

$$h_{int} = \begin{cases} inter6\_1 & , \quad if \ p_{res} = 6 \\ inter4\_1 & , \quad if \ p_{res} = 4 \end{cases} \, . \tag{397}$$

$$C_{LTP,int}(f) = \sum_{i=0}^{3} h_{int}(ip_{res}+f)C(d_{LTP}-i) + h_{int}((i+1)p_{res}-f)C(d_{LTP}+i+1). \tag{398}$$

$$f = is \quad , \quad i = \left(-\frac{p_{res}}{s}+1\right)...\left(\frac{p_{res}}{s}-1\right). \tag{399}$$

The integer and fractional parts of the refined pitch lag (d_{LTP} and f_{LTP}) are then determined by searching the maximum of the interpolated correlation:

$$\begin{aligned} i_{min} &= \begin{cases} 0 & , \quad if \ d_{LTP} = t_{min} \\ -\dfrac{p_{res}}{s}+1 & , \quad else \end{cases} \\ C_{max} &= C_{LTP,int}(i_{min}s) \\ f_{LTP} &= i_{min}s \\[10pt] for \ i &= i_{min}...\left(\frac{p_{res}}{s}-1\right) \\ &\quad if \ (C_{LTP,int}(is) > C_{max}) \ then \\ &\qquad C_{max} = C_{LTP,int}(is) \\ &\qquad f_{LTP} = is \\[10pt] if \ (f_{LTP} &< 0) \ then \\ &\quad f_{LTP} = f_{LTP} + p_{res} \\ &\quad d_{LTP} = d_{LTP} - 1 \end{aligned} \tag{400}$$

For transmission in the bitstream, the pitch lag is encoded to an integer index $I_{LTP,lag}$ (that can be encoded with 9 bits) as follows:

$$I_{LTP,lag} = \begin{cases} (d_{LTP} - p_{\min})p_{res} + f_{LTP} & , \ if \ d_{LTP} < p_{fr2} \\ (d_{LTP} - p_{fr2})\dfrac{p_{res}}{2} + (p_{fr2} - p_{\min})p_{res} + \dfrac{f_{LTP}}{2} & , \ if \ (d_{LTP} \ge p_{fr2}) \wedge (d_{LTP} < p_{fr1}) \\ (d_{LTP} - p_{fr1}) + (p_{fr1} - p_{fr2})\dfrac{p_{res}}{2} + (p_{fr2} - p_{\min})p_{res} & , \ if \ d_{LTP} \ge p_{fr1} \end{cases} .(401)$$

The decision if LTP is activated is taken according to the following condition:

$$if \ \big((bitrate < 48kbps) \wedge (mode = TCX20) \wedge \big(C_{LTP,norm}C_{LTP,norm}^{(prev)} > 0.25\big) \wedge (TFM_{LTP} < 3.5)\big) \vee$$
$$\big((bitrate \ge 48kbps) \wedge (mode = TCX10) \wedge \big(\max\big(C_{LTP,norm}, C_{LTP,norm}^{(prev)}\big) > 0.5\big) \wedge (MEC_{LTP} < 3.5)\big) \vee$$
$$\big((bitrate \ge 48kbps) \wedge (C_{LTP,norm} > 0.44) \wedge (L_{celp}(1.2 - C_{LTP,norm}) < d_{LTP})\big) \vee \qquad (402)$$
$$\big((bitrate \ge 48kbps) \wedge (mode = TCX20) \wedge (C_{LTP,norm} > 0.44) \wedge ((TFM_{LTP} < 6) \vee ((TFM_{LTP} < 7) \wedge (MEC_{LTP} < 22))))\big)$$
then
$$LTP_{on} = 1$$
else
$$LTP_{on} = 0$$

with the temporal flatness measure TFM_{LTP} and the maximum energy change MEC_{LTP} are computed as described in clause 5.1.8.

If LTP is activated, the predicted signal s_{pred} is computed from the input signal s (including the lookahead part) by interpolating the past input signal using a polyphase FIR filter. The polyphase index of the filter is determined by the fractional pitch lag:

$$i = \begin{cases} -d_{LTP} - 1 & , \ if \ f_{LTP} = 0 \\ -d_{LTP} - 2 & , \ else \end{cases}$$
$$f = \begin{cases} 0 & , \ if \ f_{LTP} = 0 \\ p_{res} - f_{LTP} & , \ else \end{cases} . \qquad (403)$$

$$h_{int} = \begin{cases} inter6\_2tcx2(f) & , \ if \ p_{res} = 6 \\ inter4\_2tcx2(f) & , \ if \ p_{res} = 4 \end{cases} . \qquad (404)$$

$$s_{pred}(n) = \sum_{j=0}^{3} h_{int}(j)s(i + j + n), n = 0...N_{TCX} - 1 . \qquad (405)$$

The LTP gain $\tilde{g}_{LTP}$ is computed from the input and predicted signals:

$$\tilde{g}_{LTP} = \frac{\displaystyle\sum_{j=0}^{N_{TCX}-1} s(n)s_{pred}(n)}{\displaystyle\sum_{j=0}^{N_{TCX}-1} (s_{pred}(n))^2} . \qquad (406)$$

For transmission in the bitstream, the gain is quantized to an integer index $I_{LTP,gain}$ (that can be encoded with 2 bits)

$$I_{LTP,gain} = \min\big(\lfloor 4\tilde{g}_{LTP} - 0.5 \rfloor, 3\big) . \qquad (407)$$

The quantized gain g_{LTP} is computed as:

$$g_{LTP} = 0.15625\big(I_{LTP,gain} + 1\big) . \qquad (408)$$

If the quantized gain is less than zero, LTP is deactivated:

$$if \left(I_{LTP,gain} < 0\right) then \, LTP_{on} = 0 \,. \tag{409}$$

If LTP is not active, the LTP parameters are set as follows:

$$
\begin{aligned}
&if \, \left(LTP_{on} = 0\right) then \\
&\quad d_{LTP} = N_{TCX} \\
&\quad f_{LTP} = 0 \\
&\quad g_{LTP} = 0 \\
&\quad s_{pred}(n) = 0 \,, \quad n = 0...N_{TCX} - 1
\end{aligned}
\tag{410}
$$

The LTP filtered signal is then computed, except at 48kbps, 96kbps and 128kbps. The LTP filtered signal is computed by multiplying the predicted signal with the LTP gain and subtracting it from the input signal. To smooth parameter changes, a zero input response is added for a 5ms transition period. If LTP was not active in the previous frame, a linear fade-in is applied to the gain over a 5ms transition period.

If LTP was active in the previous frame, the zero input response z is computed:

$$z_{pred}(n) = \sum_{j=0}^{3} h_{int}(j) s(j+n) \,, \quad n = -m...-1 \,. \tag{411}$$

$$z(n) = s_{LTP}(n) - s(n) + g_{LTP} z_{pred}(n) \,, \quad n = -m...-1 \,. \tag{412}$$

The zero input response is then computed by LP synthesis filtering with zero input, and applying a linear fade-out to the second half of the transition region:

$$z(n) = -\sum_{j=1}^{m} a(j) z(n-j), n = 0...63 \,. \tag{413}$$

$$z(n) = \min\left(\frac{64-n}{32}, 1\right) z(n), n = 0...63 \,. \tag{414}$$

$$z(n) = 0, n = 64...N_{TCX} - 1 \,. \tag{415}$$

with the LP coefficients are obtained by converting the mid-frame LSP vector of the current frame $q_{mid,i}$ using the algorithm described in subclause 5.1.9.7. Finally the LTP filtered signal is computed:

$$s_{LTP}(n) = \begin{cases} s(n) - \min\left(\dfrac{n}{64}, 1\right) g_{LTP} s_{pred}(n) & , if \, g_{LTP}^{(prev)} = 0 \\ s(n) - g_{LTP} s_{pred}(n) + z(n) & , if \, g_{LTP}^{(prev)} \neq 0 \end{cases} , n = 0...N_{TCX} - 1 \,. \tag{416}$$

5.1.14.1.1.2 Windowing and MDCT

The LTP filtered signal s_{LTP} is windowed using a sine-based window whose shape depends on the previous mode. If the past frame was encoded with a MDCT-based coding mode, the window is defined as

$$w(n) = 0 \,, for \, n = -\frac{N-L}{2},...,-1 \,. \tag{417}$$

$$w(n) = \sin\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2L}\right], for \, n = 0,..., L-1 \,. \tag{418}$$

$$w(n) = 1, for \, n = L,..., N-1 \,. \tag{419}$$

$$w(n) = \sin\left[\left(n + \frac{1}{2} - N + L\right)\frac{\pi}{2L}\right], \text{ for } n = N,...,N+L-1 \ . \tag{420}$$

$$w(n) = 0 \ , \text{ for } n = N+L,..., \frac{3N+L}{2} \ . \tag{421}$$

If the past frame was encoded with the ACELP coding mode, the window is defined as

$$w(n) = 0 \ , \text{ for } n = -\frac{7N}{8} + \frac{L}{2},...,-1 \ . \tag{422}$$

$$w(n) = 1 \ , \text{ for } n = 0,...,N-1 \ . \tag{423}$$

$$w(n) = \sin\left[\left(n + \frac{1}{2} - N + L\right)\frac{\pi}{2L}\right], \text{ for } n = N,...,N+L-1 \ . \tag{424}$$

$$w(n) = 0 \ , \text{ for } n = N+L,..., \frac{5N}{8} + \frac{L}{2} \ . \tag{425}$$

with $L = 112$, $N = 256$ at 12.8kHz, and $L = 140$, $N = 320$ at 16kHz. The total length of the window is $2N$ (40ms) when the past frame was encoded with a MDCT-based coding mode and $5N/2$ (50ms) when the past frame was encoded with the ACELP coding mode.

The windowed LTP-filtered signal is transformed with a MDCT using time domain aliasing (TDA) and a discrete cosine transform (DCT) IV as described in subclause 5.3.2.2, producing the MDCT coefficients $X(i)$ with $i = 0,...,L_{TCX} - 1$, L_{TCX} is N when the past frame was encoded with a MDCT-based coding mode and L_{TCX} is $5N/4$ when the past frame was encoded with the ACELP coding mode.

5.1.14.1.1.3 MDCT spectrum shaping

The mid-frame LSP vector of the current frame $q_{mid,i}$ is converted into LP filter coefficients $A_{TCX}(z)$ using the algorithm described in clause 5.1.9.7. The LP filter coefficients $A_{TCX}(z)$ are then weighted as described in clause 5.1.10.1, producing weighted LP filter coefficients $\tilde{A}_{TCX}(z) = A_{TCX}(z/\gamma_1)$ with $\gamma_1 = 0.92$ at 12.8kHz and $\gamma_1 = 0.94$ at 16kHz. The weighted LP filter coefficients are then transformed into the frequency domain as described in subclause 5.3.3.2.3.2. The obtained LPC gains are finally applied to the MDCT coefficients as described in subclause 5.3.3.2.3.3, producing the LPC shaped MDCT coefficients $\hat{X}(i)$.

When the encoded bandwidth is NB, the MDCT coefficients corresponding to the frequencies above 4kHz are set to zeros: $\hat{X}(i) = 0$, $i = 0,..., \frac{5L_{TCX}}{8} - 1$.

5.1.14.1.1.4 Global gain estimation

A global gain g_{TCX} is estimated similarly to the first step described in subclause 5.3.3.2.8.1.1. The energy of each block of 4 coefficients is first computed:

$$E[k] = 9 + 10\log_{10}\left(0.01 + \sum_{i=0}^{3} \hat{X}^2[4k+i]\right). \tag{426}$$

A bisection search is performed with a final resolution of 0.125dB:

Initialization: Set *fac = offset = 128* and *target = 500 if NB, target = 850 otherwise*
Iteration: Do the following block of operations 10 times
1- *fac=fac/2*
2- *offset = offset – fac*

2- $ener = \sum_{i=0}^{L_{TCX}/4-1} a[i]$, where $a[i] = \begin{cases} E[k] - offset & \text{if } E[k] - offset > 0.3 \\ 0 & \text{otherwise} \end{cases}$

3- *if(ener>target) then offset=offset+fac*

If *offset<=32, then offset= -128.*
The gain is then given by:

$$g_{TCX} = 10^{offset/20}$$

5.1.14.1.1.5   Segmental SNR estimation of the MDCT-based technology

The estimated TCX SNR in one subframe is given by

$$snr_{TCX} = \frac{\sum_{n=0}^{63} [s_h(n)]^2 + 10^{-6}}{\dfrac{64 g_{TCX} g_{TCX} \sqrt{2}}{12 L_{TCX}}} . \tag{427}$$

Finally, the estimated segmental SNR of the whole encoded TCX frame $ssnr_{TCX}$ is obtained by converting the per-subframe SNRs snr_{TCX} into dB and averaging them over all subframes.

5.1.14.1.2   Segmental SNR estimation of the ACELP technology

The segmental SNR estimation of the ACELP technology is based on the estimated SNR of the adaptive-codebook and the estimated SNR of the innovative-codebook. This is described in detail in the following clauses.

5.1.14.1.2.1   SNR estimation of the adaptive-codebook

An integer pitch-lag per subframe d_{int} is derived from the refined open-loop pitch lags d_{fr} (see clause 5.1.10.9).

When the sampling-rate is 12.8kHz, the number of subframe is four, and the integer pitch lags are simply equal to the refined open-loop pitch lags rounded to the nearest integer.
When the sampling-rate is 16kHz, the number of subframe is five. The refined open-loop pitch lags are first scaled by a factor of 1.25, then they are rounded to the nearest integer and finally the four obtained integer pitch lags are mapped to the five subframes. The first integer pitch-lag is mapped to the first subframe, the second integer pitch-lag is mapped to the second subframe, the third integer pitch-lag is mapped to the third and fourth subframes, and the fourth integer pitch-lag is mapped to the fifth subframe.

A gain is then computed for each subframe

$$g = \frac{\sum_{n=0}^{63} s_h(n) s_h(n - d_{int})}{\sum_{n=0}^{63} s_h(n - d_{int}) s_h(n - d_{int}) + 10^{-6}} . \tag{428}$$

The estimated SNR of the adaptive-codebook is then computed for each subframe

$$snr_{ada} = \frac{\sum_{n=0}^{63} [s_h(n)]^2 + 10^{-6}}{\sum_{n=0}^{63} [s_h(n) - g s_h(n - d_{int})]^2 + 10^{-6}} . \tag{429}$$

5.1.14.1.2.2 SNR estimation of the innovative-codebook

The estimated SNR of the innovative-codebook snr_{ino} is assumed to be a constant, which depends on the encoded bandwidth and on the bitrate. $snr_{ino} = 0.15$ at 9.6kbps NB, $snr_{ino} = 0.059$ at 9.6kbps WB and $snr_{ino} = 0.092$ at 16.4 and 24.4kbps WB and SWB.

5.1.14.1.2.3 Segmental SNR estimation of ACELP

The estimated SNR of one ACELP encoded subframe is then computed by combining the adaptive-codebook SNR and the innovative-codebook SNR.

$$snr_{ace} = snr_{ada} snr_{ino}. \tag{430}$$

Finally, the estimated segmental SNR of the whole encoded ACELP frame $ssnr_{ace}$ is obtained by converting the per-subframe SNRs snr_{ace} into dB and averaging them over all subframes.

5.1.14.1.3 Hysteresis and final decision

The ACELP technology is selected if

$$ssnr_{ace} + dssnr > ssnr_{TCX} . \tag{431}$$

otherwise the MDCT-based technology is selected.

$dssnr$ adds hysteresis in the decision, in order to avoid switching back and forth too often between the two coding technologies. $dssnr$ is computed as described below ($dssnr$ is 0 by default). Further, in 12.8 kHz core (i.e., 9.6 kbps and 13.2 kbps), the dssnr is updated as shown in equation (433a).

$$
\begin{aligned}
&if\left(ssnr_{ace} > ssnr_{TCX}\right) \wedge \\
&\left(ssnr_{ace} < ssnr_{TCX} + 2\right) \wedge \\
&\left(\left(TFM^{(prev)} + TFM < 3.25\right) \vee \left(stabfac = 1\right)\right) \wedge . \\
&\left(num\_acelp\_frames \le 6\right) \\
&then \\
&dssnr = -2
\end{aligned}
\tag{432}
$$

$$
\begin{aligned}
&if\left(ssnr_{ace} < ssnr_{TCX}\right) \wedge \\
&\left(ssnr_{ace} > ssnr_{TCX} - 2\right) \wedge \\
&\left(TFM^{(prev)} + TFM > 3.25\right) \wedge . \\
&\left(num\_acelp\_frames \ge 6\right) \\
&then \\
&dssnr = 2
\end{aligned}
\tag{433}
$$

$$
\begin{aligned}
&if\left(sr\_core = 12800\right) \wedge \left(offset < 74\right) \wedge \left(nonstat > 5\right) \wedge \\
&\left(ssnr_{ace} >= ssnr_{TCX} - 4\right) \wedge \left(num\_acelp\_frames >= 1\right) \wedge \\
&\left(\begin{aligned}&\left(\left(L_s > L_m\right) \wedge \left(mean(voicing) > 0.3\right)\right) \vee \\ &\left(\left(num\_acelp\_frames >= 6\right) \wedge \left(L_s > L_m - 1.5\right)\right)\end{aligned}\right) \wedge \\
&\left(f_{SM} = 0\right) \wedge \left(vad\_flag\right) \\
&then \\
&dssnr = 4
\end{aligned}
\tag{433a}
$$

where $offset$ is described in clause 5.1.14.1.1.4, and L_s, L_m, f_{SM} are described in clause 5.1.13.6, and $nonstat$ is described in 5.1.11.2.1.

$$
\begin{aligned}
&if\left(SNR_{LT} < 35\right) \\
&then \\
&\quad if\left((vad\_flag = 1) \vee (dtx\_on = 1)\right) \\
&\quad then \\
&\quad\quad dssnr = dssnr + 2 \\
&\quad else \\
&\quad\quad dssnr = dssnr - 2
\end{aligned}
\tag{434}
$$

with *TFM* is the temporal flatness measure described in clause 5.1.8, *stabfac* is a stability factor described in subclause 6.1.1.3.2 but using the unquantized LSF parameters estimated at 12.8kHz, num_acelp_frames is the number of consecutive previous ACELP frames (if the previous frame was not ACELP, $num\_acelp\_frames = 0$), SNR_{LT} is the long-term SNR as described in clause 5.1.12, vad_flag is the SAD decision as described in clause 5.1.12, and dtx_on indicates whether DTX is enabled or not.

5.1.14.2 TCX/HQ MDCT technology selection at 13.2 and 16.4 kbps

The selection between TCX and HQ MDCT (Low Rate HQ) technology at 13.2 kbps (NB, WB and SWB) and 16.4 kbps (WB and SWB) is done on a frame-by-frame basis and is based on the following measures.

– Voicing measures

– Spectral noise floor

– SAD decision

– High-band energy

– High-band sparseness (with hysteresis)

The boundaries of frequency bands for the purposes of the TCX/HQ technology selection is set according to the following table.

Table 21: Boundaries of frequency bands for TCX/HQ MDCT (Low Rate HQ) selection

Band width	Low band CLDFB b_{LCLDFB}	High band CLDFB b_{HCLDFB}	Low band FFT b_{LFFT}	High band FFT b_{HFFT}
NB	8	10	L_{FFT} /4	L_{FFT} *5/16
WB	12	20	L_{FFT} *3/8	L_{FFT} /2
SWB	16	40	L_{FFT} /2 for sparseness, L_{FFT} *3/8 otherwise	L_{FFT} /2

Voicing measure V is defined as the average of pitch gain $C_{norm}^{[0]}$ of the former half-frame and $C_{norm}^{[1]}$ of the latter half-frame defined in (81),

$$
V = \frac{1}{2}\left(C_{norm}^{[0]} + C_{norm}^{[1]}\right).
\tag{435}
$$

Sparseness measure S is defined as

$$
S = 1. - 2 \cdot p / b_{LFFT} ,
\tag{436}
$$

where p is a number of bins which attain following condition within low band:

$$
E_i > \max(E_{i-1}, E_{i+1}, 3.0, \log(10) \cdot (E_{tot} - \text{MDCT\_SW\_SIG\_PEAK\_THR})) ,
\tag{437}
$$

where E_{tot} is an averaged energy of all spectrum bands.

High energy measure E_{High} is defined in terms of CLDFB energy as

$$E_{High} = \log_{10}\left(\left(\frac{1}{(b_{HCLDFB} - b_{LCLDFB})}\right)\sum_{j=b_{LQMF}}^{b_{HQMF}-1} E_j + 0.0001\right). \tag{438}$$

Flag indicating the sparseness for high bands, f_{H_SPARSE} = TRUE when

$$N_{peak} \leq (b_{HFFT} - b_{LFFT}) \cdot \text{HI\_SPARSE\_THR}, \tag{439}$$

where N_{peak} is a number of FFT bins within b_{LFFT} and $b_{HFFT} - 1$ which attain

$$E_j \geq \text{E}_{tot} + \log(10) \cdot \text{MDCT\_SW\_SIG\_LINE\_THR}. \tag{440}$$

Otherwise, f_{H_SPARSE} =FALSE.

Flag indicating the sparseness for high bands with hysteresis, $f_{H\_SPARSE\_HYS}$ = TRUE when

$$N_{peak} \leq ((b_{HFFT} - b_{LFFT}) \cdot \text{HI\_SPARSE\_THR/HYST\_FAC}). \tag{441}$$

Otherwise, $f_{H\_SPARSE\_HYS}$ =FALSE.

Additionally, f_{H_SPARSE} is set TRUE when following is satisfied:

$$prev\_f_{\text{H\_SPARSE}} > 0 \,\&\&\, f_{\text{H\_SPARSE}} \,\&\&\, (\min(C_{norm}^{[0]}, C_{norm}^{[1]}, C_{norm}^{[2]}) > \text{VOICING\_THR}). \tag{442}$$

E_{floor} is the averaged energy only for the local minima of the spectrum. With the notation of 5.1.11.2.5, it is defined as:

$$E_{floor} = \frac{1}{N_{\min}} \sum_{i=0}^{N_{\min}-1} E_{dB}(i_{\min}(i)). \tag{443}$$

Correlation map sum, m_{sum} is defined in 5.1.11.2.5.

Indication of possible switching, f_{Switch} =TRUE when previous core was not Transform coding, or followings are satisfied.

$$\begin{aligned}
&(\,prev\_E_{HIGH} <= \text{HI\_ENER\_LO\_THR}\,)// \\
&(\,E_{HIGH} <= \text{HI\_ENER\_LO\_THR}\,)// \\
&(\,prev\_core == HQ \,\&\&\,(\,total\_brate == 13200\,// \\
&\quad f_{H\_SPAESE} \,\&\&\, prev\_f_{H\_SPARSE} \geq 0 \,\&\&\, prev\_f_{H\_SPARSE} \leq 1\,))// \\
&(\,prev\_core == TCX \,\&\&\,(!\, f_{H\_SPARSE} \,\&\&\, prev\_f_{H\_SPARSE} > 0\,))
\end{aligned} \tag{444}$$

where $prev\_f_{HIGH}$ and $prev\_f_{H\_SPARSE}$ are f_{HIGH} and f_{H_SPARSE} at the previous frames. Note that $prev\_f_{H\_SPARSE}$ is integer from -1 to 2, while others are all Boolean.

Indication of preference for TCX, f_{TCX} = TRUE when followings are satisfied:

$$\begin{aligned}
&(E_{tot} - E_{floor} \geq \text{SIG\_HI\_LEVEL\_THR}) \,\&\& \\
&((m_{sum} \geq \text{COR\_THR}) \,\| \\
&\quad (V \geq \text{VOICING\_THR}) \,\| (S \geq \text{SPARSENESS\_THR})) \,\&\& \\
&((E_{High} \leq \text{HI\_ENER\_LO\_THR}) \,\| (f_{H\_Sparse}))
\end{aligned} \tag{445}$$

Indication of preference for HQ MDCT, f_{HQ} = TRUE when followings are satisfied:

$$(E_{tot} - E_{floor} < \text{SIG\_LO\_LEVEL\_THR})\|$$
$$((m_{sum} < \text{COR\_THR} \cdot \text{HYST\_FAC}) \& \&$$
$$(V < \text{VOICING\_THR} \cdot \text{HYST\_FAC}) \& \& \qquad , \qquad (446)$$
$$(S < \text{SPARSENESS\_THR} \cdot \text{HYST\_FAC}))\|$$
$$(total\_brate == 13200 \& \& ! f_{TCX} \& \& transient\_frame)$$

where *transient_frame* is the output of the time-domain transient detector (see 5.1.8). For 16.4 kbps, f_{TCX} is set to FALSE and f_{HQ} to TRUE when *transient_frame* is detected.

Based on the above definitions and thresholds listed in the table below, switching between HQ and MDCT based TCX is carried out as follows. Switching between HQ and TCX can only occur when f_{Switch} is TRUE. In this case, TCX is used if f_{TCX} is TRUE, or otherwise HQ is used if f_{TCX} is TRUE. In any other case, the same kind of transform coding is applied as in the previous frame. If the previous frame was not coded by transform coding, HQ is used for the low rate (13.2 kbps) and TCX for the high rate (16.4 kbps).

In case input signal is noisy speech (*noisy_speech_flag*==TRUE && *vadflag*== FALSE) , transition from TCX to HQ is prohibited at 16.4 kbps.

$prev\_f_{H\_SPARSE}$ is reset to 0 if f_{H_SPARSE} is FALSE, otherwise it is incremented by one (with a maximum allowed value of 2)

$prev\_E_{HIGH}$ and $prev\_f_{H\_SPARSE}$ are reset to FALSE and -1, respectively, upon encoder initialization or when a non-transform-coded frame is encountered.

Table 22: List of thresholds used in TCX/HQ MDCT (Low Rate HQ) selection

Parameter	Meaning	13.2 kbps	16.4 kbps
SIG_LO_LEVEL_THR	Low level signal	22.5	23.5
SIG_HI_LEVEL_THR	High level signal	28.0	19.0
COR_THR	correlation	80.0	62.5
VOICING_THR	voicing	0.6	0.4
SPARSENESS_THR	sparseness	0.65	0.4
HI_ENER_LO_THR	High energy low limit	9.5	12.5
HYST_FAC	Hysteresis control	0.8	0.8
MDCT_SW_SIG_LINE_THR	Significant Spectrum	2.85	2.85
MDCT_SW_SIG_LINE_THR	Significant peak	36.0	36.0

5.1.14.3 TCX/HQ MDCT technology selection at 24.4 and 32 kbps

The decision between using the TCX technology or the HQ MDCT (high rate HQ) technology at 24.4 kbps and 32 kbps for SWB signals is based on the average energy values and peak-to-average ratios of different sub-bands, furthermore, the average energy values and peak-to-average ratios are calculated by the CLDFB band energy analysis $\overline{E}_C(k)$, spectral analysis $X(k)$ and the bit-rate.

First, the average energy of three CLDFB sub-bands: 0~3.2kHz, 3.2~6.4kHz and 6.4~9.6kHz $E_{gain}(i)$, $i = 0,1,2$ are calculated according to

$$E_{gain}(i) = \sum_{k=0}^{7} \overline{E}_C(k + 8i) / 8, \qquad i = 0,1,2 \qquad (447)$$

Second, the spectral peak $peak(i)$ and spectral average $avrg(i)$, $i = 0,1$ of the FFT sub-bands: 1~2.6kHz and 4.8~6.4kHz are calculated according to

$$peak(0) = \max(X(20), X(21),..., X(20+31))$$
$$avrg(0) = \sum_{k=0}^{31} X(20+k)$$
$$peak(1) = \max(X(96), X(97),..., X(96+31))$$
$$avrg(1) = \sum_{k=0}^{31} X(96+k)$$

(448)

At 24.4kbps, the CLDFB sub-band (4.8~9.6kHz) average energy $E_{gain}(3)$, and the CLDFB sub-band (400-3.2kHz) average energy $E_{gain}(4)$ are also calculated according to

$$E_{gain}(3) = \sum_{k=0}^{11} \overline{E}_C(k+12)/12$$
$$E_{gain}(4) = 8 \cdot (E_{gain}(0) - \overline{E}_C(0)/8)/7$$

(449)

The peak energy $peak_{E1}$ and average energy $avrg_{E1}$ of the CLDFB sub-band (8~10kHz) are also calculated according to

$$peak_{E1} = \max(\overline{E}_C(20), \overline{E}_C(21),..., \overline{E}_C(20+4))$$
$$avrg_{E1} = \sum_{k=0}^{4} \overline{E}_C(20+k)$$

(450)

To identify the MDCT coding mode, three conditions are identified:

Condition I:

$$\text{if } E_{gain}(3) > 0.8 \cdot E_{gain}(4) \text{ AND}$$
$$2.56 \cdot peak(0) \cdot avrg(1) > peak(1) \cdot avrg(0) \text{ AND}$$
$$peak(0) \cdot avrg(1) < 5.12 \cdot peak(1) \cdot avrg(0))$$
$$\text{then } ConditionI = 1$$
$$\text{else } ConditionI = 0$$

(451)

Condition II:

$$\text{if } E_{gain}(3) > 0.4 \cdot E_{gain}(4) \text{ AND } 32 \cdot peak(1) < 1.5 \cdot avrg(1) \text{ AND } 5 \cdot peak_{E1} < 1.5 \cdot avrg_{E1}$$
$$\text{then } ConditionII = 1 \text{ else } ConditionII = 0$$

(452)

Condition III:

$$\text{if } ((2.56 \cdot peak(0) \cdot avrg(1) < peak(1) \cdot avrg(0) \text{ AND } 32 \cdot peak(1) > 1.5 \cdot avrg(1)) \text{ OR}$$
$$(peak(0) \cdot avrg(1) > 2.56 \cdot peak(1) \cdot avrg(0) \text{ AND } 32 \cdot peak(1) < 1.5 \cdot avrg(1)))$$
$$\text{then } ConditionIII = 1$$
$$\text{else } ConditionIII = 0$$

(453)

The primary classifier decision D_{MDCT1} at 24.4kbps is formed according to

$$\text{if } (conditionI \text{ OR } conditionII \text{ OR } conditionIII)$$
$$\text{then } D_{MDCT1} = 3, \text{ i.e. select HQ-MDCT}$$
$$\text{else } D_{MDCT1} = 1, \text{ i.e. select TCX}$$

(454)

At 32kbps, further spectral analysis is needed. First, a noise-floor envelope $X_{nf}(k)$ and a peak envelope $X_p(k)$ are calculated as

$$X_{nf}(0) = |X(0)|^2$$
$$X_{nf}(k) = \alpha_{nf} X_{nf}(k-1) + (1-\alpha_{nf})|X(k)|^2, \qquad k=1,...,L_{FFT}-1$$

(455)

and

$$X_p(0) = |X(0)|^2$$
$$X_p(k) = \alpha_p X_p(k-1) + (1-\alpha_p)|X(k)|^2, \qquad k=1,...,L_{FFT}-1$$

(456)

respectively, where the smoothing factors α_{nf} and α_p depend on the instantaneous magnitude spectrum

$$\alpha_{nf} = \begin{cases} 0.9578, & |X(k)|^2 > X_{nf}(k-1) \\ 0.6472, & |X(k)|^2 \le X_{nf}(k-1) \end{cases}$$

(457)

$$\alpha_p = \begin{cases} 0.4223, & |X(k)|^2 > X_p(k-1) \\ 0.8029, & |X(k)|^2 \le X_p(k-1) \end{cases}$$

(458)

The noise-floor energy E_{nf} and the peak envelope energy E_p are formed by averaging the noise-floor and peak envelopes, respectively. That is,

$$E_{nf} = \frac{1}{L_{FFT}} \sum_{k=0}^{L_{FFT}-1} X_{nf}(k)$$

(459)

$$E_p = \frac{1}{L_{FFT}} \sum_{k=0}^{L_{FFT}-1} X_p(k)$$

(460)

Spectral peaks are identified in two steps. First, all k for which $|X(k)|^2 > 0.64 \cdot X_p(k)$ holds true are marked as peak candidates. Second, for each sequence of consecutive k, the largest spectral magnitude is kept as a peak representative for that sequence. Peak sparseness measure S is formed by averaging the peak distances among the peak representatives, with $S=0$ if less than 2 peaks are identified. Two decision variables are formed

$$isclean = E_p/E_{nf} > 147.87276$$
$$issparse = S > 12$$

(461)

The peak energy $peak_{E2}$ and average energy $avrg_{E2}$ of the CLDFB sub-band at 10~12 kHz are calculated according to

$$peak_{E2} = \max(\overline{E}_C(25), \overline{E}_C(26),...,\overline{E}_C(25+4))$$
$$avrg_{E2} = \sum_{k=0}^{4} \overline{E}_C(25+k)$$

(462)

Three conditions are then checked.

Condition I:

$$\text{if } (E_{gain}(2) > 1.2 \cdot E_{gain}(1)) \text{ then } ConditionI = 1 \text{ else } ConditionI = 0$$

(463)

Condition II:

$$\text{if } E_{gain}(2) > 0.8 \cdot E_{gain}(1) \text{ AND } 5 \cdot peak_{E2} > 2.0 \cdot avrg_{E2} \text{ then } ConditionII = 1 \text{ else } ConditionII = 0$$ (464)

Condition III:

$$\text{if } 2.25 \cdot peak(0) \cdot avrg(1) < peak(1) \cdot avrg(0) \text{ OR } peak(0) \cdot avrg(1) > 2.25 \cdot peak(1) \cdot avrg(0))$$
$$\text{then } ConditionIII = 1 \text{ else } ConditionIII = 0$$

(465)

The primary classifier decision D_{MDCT1} at 32kbps is formed according to

$$if \; ((isclean \oplus issparse) \; OR \; conditionI \; OR \; conditionII \; OR \; conditionIII)$$
$$then \; D_{MDCT1} = 3, \; i.e. \; select \; HQ \; MDCT \; technnology \qquad (466)$$
$$else \; D_{MDCT1} = 1 \; i.e. \; select \; TCX \; technology$$

To increase the classifier stability for both 24.4kbps and 32kbps, the primary classifier decision D_{MDCT1} is low-pass filtered from frame to frame.

$$\overline{D}_{MDCT1} = 0.2 D_{MDCT1} + 0.8 \overline{D}_{MDCT1}^{[-1]} \qquad (467)$$

Finally, hysteresis is applied such that the classifier decision from the previous frame is only changed if the decision passes the switching range $[1.1, 1.6]$

$$if \; (E_{gain}^{[-1]}(0) > 0.5 \cdot E_{gain}(0) \; AND \; E_{gain}^{[-1]}(0) < 2.0 \cdot E_{gain}(0)) \; AND$$
$$(E_{gain}^{[-1]}(1) > 0.5 \cdot E_{gain}(1) \; AND \; E_{gain}^{[-1]}(1) < 2.0 \cdot E_{gain}(1))$$
$$then \; D_{MDCT}^{final} = D_{MDCT}^{[-1]} \qquad (468)$$
$$else \; if \; \overline{D}_{MDCT1} > D_{MDCT}^{[-1]} \; \&\& \overline{D}_{MDCT1} > 1.6 \; then \; D_{MDCT}^{final} = 3$$
$$else \; if \; \overline{D}_{MDCT1} < 1.1 \; then \; D_{MDCT}^{final} = 1$$

If none of these conditions are met, the previous classifier is kept, i.e. $D_{MDCT}^{final} = D_{MDCT}^{[-1]}$ .and the buffers are updated as follows

$$\overline{D}_{MDCT1}^{[-1]} = \overline{D}_{MDCT1}$$
$$D_{MDCT}^{[-1]} = D_{MDCT}^{final}$$
$$E_{gain}^{[-1]}(1) = \cdot E_{gain}(1) \qquad (469)$$
$$E_{gain}^{[-1]}(0) = \cdot E_{gain}(0)$$

5.1.14.4 TD/Multi-mode FD BWE technology selection at 13.2 kbps and 32 kbps

The input WB or SWB signal is divided into low band signal and high band signal (wideband input) or super higher band signal (super wideband input). Firstly, the low band signal is classified based on the characteristics of the low band signal and coded by the LP-based approach or the transform-domain approach.

The selection between TD BWE and multi-mode FD BWE technology of super higher band signal or high band signal at 13.2 kbps (WB and SWB) and 32 kbps (SWB) is performed based on the characteristic of the input signal and coding modes of the low band signal. Except for MDCT mode, if the input signal is classified as music signal, the high band signal or the super higher band signal is encoded by multi-mode FD BWE;if the input signal is classified as speech signal, the high band signal or the super higher band signal is encoded by TD BWE. In the case that the low band signal is classified as IC mode, the high band signal or the super higher band signal is also encoded by multi-mode FD BWE.

If the decision in the first stage of the speech/music classifier $f_{SM1} = 1$, i.e. the input signal is classified as music signal, or the decision in the first stage of the speech/music classifier $f_{SM1} = 0$ and the decision in the second stage of the speech/music classifier $f_{SM2} = 1$, or the low band signal is classified as IC mode, the high band or the super higher band signal is encoded by multi-mode FD BWE, otherwise, the high band or super higher band signal is encoded by TD BWE. It is noted that, when the flag of the super wideband noisy speech $f_{UV\_SWB} = 1$, the super higher band is encoded by TD BWE. It is the same TD/multi-mode FD BWE technology selection for FB inputs.

5.2 LP-based Coding

In general terms, speech dominated content is encoded using Analysis-by-Synthesis Linear Prediction (LP) paradigm. At some low bitrates configurations, the LP-based coding is used also for generic audio. On the other hand, LP prediction is not used above 64 kb/s. The LP-based coding consists in encoding the LP excitation signal and the speech spectral envelope, represented by the LP filter coefficients. Depending on the particular characteristics of a speech frames, different flavours of the excitation coding are used to encode voiced or unvoiced speech frames, audio frames, inactive frames etc.

The internal sampling rate of the LP-based coding is rather independent of the input signal sampling rate. Instead, it depends on the encoded bitrate to optimize coding efficiency. In the EVS, there are two different internal sampling rates used – 12.8 kHz is used up to 13.2 kb/s inclusively, and 16 kHz sampling rate is used for higher bitrates. It means that up to 13.2 kb/s, the LP-based encoding covers first 6.4 kHz of the input signal while from 16.4 kb/s and up the LP-based encoding covers 8 kHz of the input. For NB signals, the sampling rate is always 12.8 kHz.

For other than NB signals, the upper bandwidth (not covered with the LP-based coding) is then encoded using bandwidth extension (BWE) technologies, ranging from blind BWE at the lowest bitrates, parametric BWEs optimized to different content at higher bitrates, up to full encoding of the upper bandwidth spectrum at the highest bitrate (64 kb/s).

The basic block for the LP excitation coding is a subframe. The size of the subframe in samples is independent of the internal sampling rate. It equals to 64 samples. It means that at 12.8 kHz internal sampling rate, EVS uses 4 subframes of 5 ms while at 16 kHz internal sampling rate, EVS uses 5 subframes of 4 ms.

5.2.1 Perceptual weighting

The encoding parameters, such as adaptive codebook delay and gain, algebraic codebook index and gain are searched by minimizing the error between the input signal and the synthesized signal in a perceptually weighted domain. Perceptual weighting is performed by filtering the signal through a perceptual weighting filter, derived from the LP filter coefficients. The perceptually is similar to the weighting also used in open-loop pitch analysis. However, an adaptive perceptual weighting is used in case of LP-based excitation coding.

The traditional perceptual weighting filter $W(z) = A(z/\gamma_1)/A(z/\gamma_2)$ has inherent limitations in modelling the formant structure and the required spectral tilt concurrently. The spectral tilt is more pronounced in wideband signals due to the wide dynamic range between low and high frequencies. A solution to this problem is to introduce a pre-emphasis filter at the input and enhance the high frequency content in case of wideband signals. The LP filter coefficients are then found by means of LP analysis on the pre-emphasized signal. Subsequently, they are used to form a perceptual weighting filter. Its transfer function is the same as the LP filter transfer function but with the denominator having fixed coefficients (similar to the pre-emphasis filter). In this way, the weighting in formant regions is decoupled from the spectral tilt as shown below. Finally, the pre emphasized signal is filtered through the perceptual filter to obtain a perceptually weighted signal, which is used further.

The perceptual weighting filter has the following form

$$W(z) = A(z/\gamma_1)H_{\text{de-emph}}(z) = A(z/\gamma_1)/\left(1 - \beta_1 z^{-1}\right) \tag{470}$$

where

$$H_{\text{de-emph}}(z) = \frac{1}{(1 - \beta_1 z^{-1})} \tag{471}$$

and β_1 is equal to 0.68.

Because $A(z)$ is computed based on the pre-emphasized signal $s_{pre}(n)$, the tilt of the filter $1/A(z/\gamma_1)$ is less pronounced compared to the case when $A(z)$ is computed based on the original signal (as the pre-emphasized signal itself exhibits less spectral tilt than the original wideband signal). Since de-emphasis is performed in the decoder, it can be shown that the quantization error spectrum is shaped by a filter having a transfer function $1/W(z)H_{de-emph}(z) = 1/A(z/\gamma_1)$. Thus, the spectrum of the quantization error is shaped by a filter whose transfer

function is $1/A(z/\gamma_1)$, with $A(z)$ computed based on the pre-emphasized signal. The perceptual weighting is performed on a frame basis while the LP filter coefficients are calculated on a subframe basis using the principle of LSP interpolation, described in subclause 5.1.9.6. For a subframe of size $L = 64$, the weighted speech is given by

$$s_h(n) = s_{pre}(n) + \sum_{i=1}^{16} a_i \, \gamma_1^i \, s_{pre}(n-i) + \beta_1 \, s_h(n-1), \qquad n = 0, \ldots, L-1 \qquad (472)$$

5.2.2 LP filter coding and interpolation

5.2.2.1 LSF quantization

5.2.2.1.1 LSF weighting function

For frame-end LSF quantization, the weighting given by equation (481) is defined by combining the magnitude weighting, frequency weighting, IHM and squared IHM.

As shown in figure 19, since the spectral analysis and LP analysis use similar temporal sections, the FFT spectrum of the second analysis window can be reused to find the best weighting function for the frame-end LSF quantizer.

Relative positions of the spectral analysis windows

Relative positions and length of the LP analysis windows
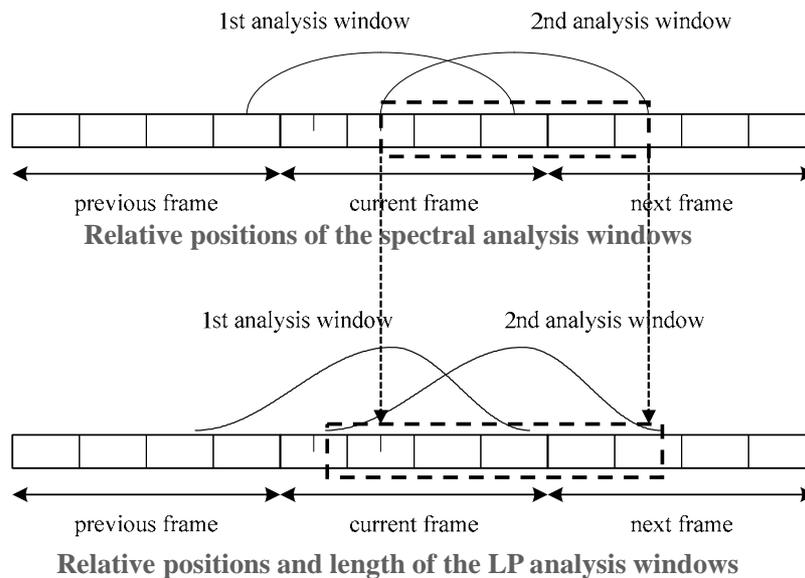
Figure 19: LSF weighting computation with FFT spectrum

Figure 20 is a block diagram of a spectral analysis module that determines a weighting function. The spectral analysis computation is performed by a pre-processing module and the output is a linear scale spectrum magnitude which is obtained by FFT.
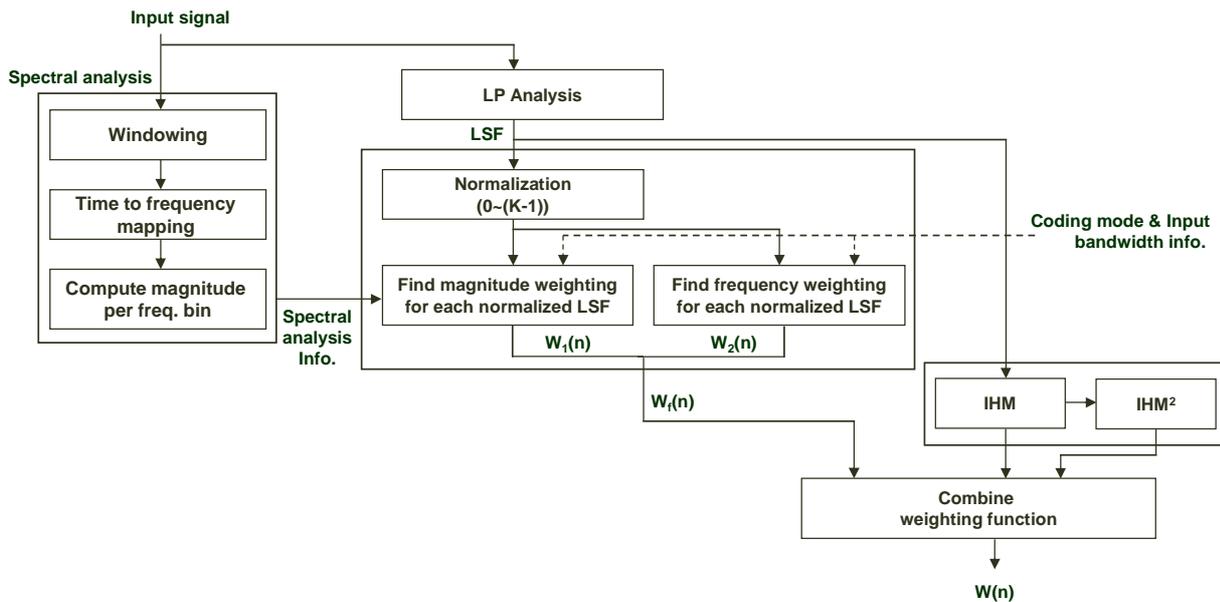
Figure 20: Block diagram of LSF weighting computation

In the Normalization block, the LSFs are normalized to a range of 0 to K -1. The LSFs generally span the range of 0 to π. For a 12.8 kHz internal sampling frequency, K is 128 and for a 16 kHz internal sampling frequency, K is 160.

The Find magnitude weighting for each normalized LSF block determines the magnitude weighting function $W_1(n)$ using the spectrum analysis information and the normalized LSF.

The magnitude weighting function is determined using the magnitude of the spectral bins corresponding to the frequency of the normalized LSFs and the additional two magnitudes of the neighbouring spectral bins (+1 and -1 of the spectral bin corresponding to the frequency of the normalized LSFs) around the spectral bin.

The spectral magnitude is obtained by a 128-point FFT and its bandwidth corresponds to the range of 0 to 6400 Hz. If the internal sampling frequency is16 kHz, the number of spectral magnitudes is extended to 160. Because the spectrum magnitude for the range of 6400 to 8000 Hz is missing, the spectrum magnitude for this range will be generated by the input spectrum. More specifically, the average value of the last 32 spectrum magnitudes which correspond to the bandwidth of 4800 to 6400 Hz are repeated to fill in the missing spectrum.

The final magnitude function determines the weighting function of each magnitude associated with a spectral envelope by extracting the maximum magnitude among the three spectral bins.

$$W_1(n) = \left(\sqrt{w_f(n) - Min} \right) + 2 \text{ ,for } n=0,...,M\text{-}1 \tag{473}$$

where Min is the minimum value of $w_f(n)$ and

$$w_f(n) = 10 \log\left(E_{\max}(n) \right), \text{ for } n=0,...,M\text{-}1 \tag{474}$$

where $M = 16$ and the $E_{\max}(n)$ is the maximum magnitude among the three spectral bins for each LSF.

In the Find frequency weighting for each normalized LSF block, the frequency weighting function $W_2(n)$ is determined by using frequency information from the normalized LSF.

The function determines the weighting function of each frequency using the predetermined weighting graph which is selected by using the input bandwidth and coding mode. There are two predetermined weighting graphs, as shown in figure 21, which are determined by perceptual characteristics such as Bark scale and a formant distribution of the input signal.

The function corresponding to graph (a) in figure 21 is as follows.

$$W_2(n) = \begin{cases} 0.5 + \dfrac{\sin\left(\dfrac{\pi \cdot f_n(n)}{12}\right)}{2}, & for\ f_n(n) = [0,5], \\[4mm] \dfrac{1}{\left(\dfrac{(f_n(k)-6)}{121}+1\right)}, & for\ f_n(n) = [6,127]\ for\ WB\ and\ [6,159]\ for\ WB16kHz. \end{cases} \tag{475}$$

The function corresponding to graph (b) in figure 21 is as follows.

$$W_2(n) = \begin{cases} 0.5 + \dfrac{\sin\left(\dfrac{\pi \cdot f_n(n)}{12}\right)}{2}, & for\ f_n(n) = [0,5], \\[4mm] 1.0, & for\ f_n(n) = [6,20], \\[4mm] \dfrac{1}{\left(\dfrac{3(f_n(k)-20)}{107}+1\right)}, & for\ f_n(n) = [21,127]\ for\ WB\ and\ [21,159]\ for\ WB16kHz. \end{cases} \tag{476}$$
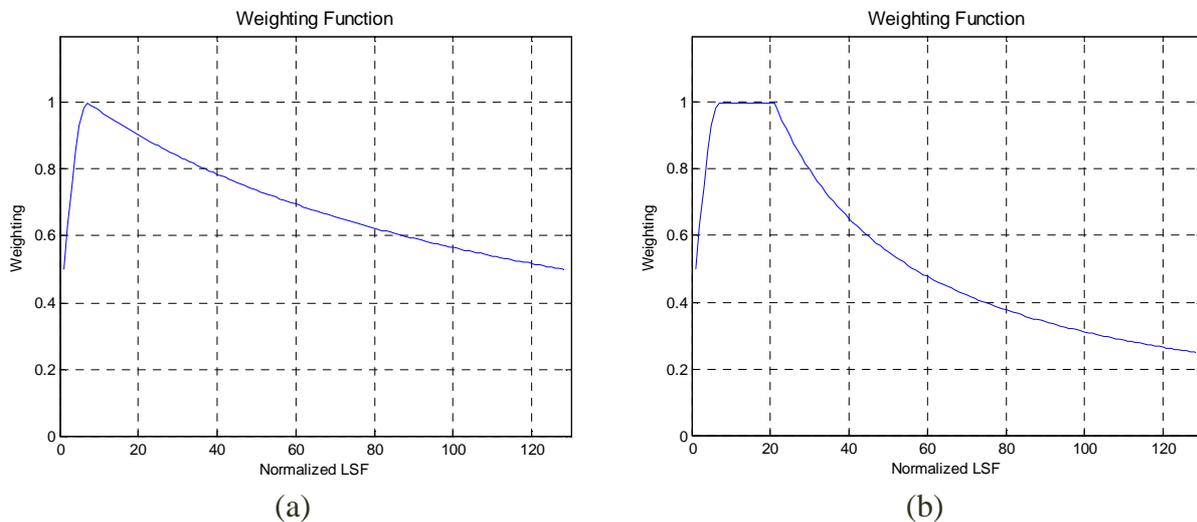
Figure 21: Frequency weighting functions

Next, the FFT weighting function $W_f(n)$ is determined by combining the magnitude weighting function and the frequency weighting function. Computing the FFT weighting function $W_f(n)$ for frame-end LSF quantization is performed as follows:

$$W_f(n) = W_1(n) \cdot W_2(n)\ ,\ n=0,...,M\text{-}1 \tag{477}$$

The FFT weighting function uses different types of frequency and magnitude weighting functions depending on frequency bandwidth (NB, WB or WB16 kHz) and coding modes (UC or others such as VC, GC, AC, IC and TC).

Along with the FFT weightings W_f, another weighting function called the inverse harmonic mean (IHM) is computed and defined as:

$$W_{IHM}(n) = \frac{1}{lsf_n - lsf_{n-1}} + \frac{1}{lsf_{n+1} - lsf_n}\ ,\ n=0,...,M\text{-}1 \tag{478}$$

The LSFs lsf_n are normalized between 0 and π, where the first and the last weighting coefficients are calculated with this pseudo LSFs $lsf_0 = 0$ and $lsf_M = \pi$. M is the order 16 of the LP model.

IHM approximates the spectral sensitivity of LSFs by measuring how close adjacent LSFs come. If two LSF parameters are close together the signal spectrum has a peak near that frequency. Hence a LSF that is close to one of its neighbours has a high scalar sensitivity and should be given a high weight. The sensitivity of close neighbours LSF is even enhanced by computing the squared of IHM:

$$W_{IHM^2}(n) = W_{IHM}(n) \cdot W_{IHM}(n), \quad n=0,...,M\text{-}1 \tag{479}$$

The three set of weightings, W_f, W_{IHM}, and W_{IHM^2} are gathered into an M by 4 matrix as follows:

$$E = \begin{bmatrix} 1 & W_{IHM}(0) & W_{IHM^2}(0) & W_f(0) \\ 1 & W_{IHM}(1) & W_{IHM^2}(1) & W_f(1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & W_{IHM}(M-1) & W_{IHM^2}(M-1) & W_f(M-1) \end{bmatrix} \tag{480}$$

The set of weightings are combined linearly by multiplying the matrix E by a constant column vector P of dimension M:

$$W = E \cdot P \tag{481}$$

The vector P is different for NB, WB/SWB at internal sampling rate 12.8 kHz and WB/SWB at internal sampling rate 16 kHz. The vectors P are derived off-line over a training data by minimizing the distance of the linear combination W and the weightings derived mathematically based on Gardner and Rao method, weightings near-optimal but too complex for being computed on-line compared to an heuristic approach.

5.2.2.1.2 Bit allocation

The frame-end LSF quantization codebooks and bit allocations depend on the selected coding mode. In addition, different codebooks are used for NB, WB and WB 16kHz modes. This means there is a separate, optimized codebook for each coding mode and for each input bandwidth. In NB mode the LSF vectors are in the range of 0-6400Hz although the input signal has content only up to 4kHz. The WB mode corresponds to the mode where the LSF parameters are estimated in the 0-6400Hz range. The WB2 mode corresponds to the mode where the LSF parameters are estimated in the 0-8000Hz range and it is used in general for the higher bitrates.

Table 23 shows the bit allocation for frame-end LSF quantization for each coding mode.

Table 23: Bit allocation for LSF vectors

ACELP core bitrate (kbps)	Inactive	Unvoiced	Voiced	Generic	Transition	Audio	
3.6	0	27	16	22	0	0	
7.2	22	37	31	29	31	22	
8.0	22	40	36	33	34	22	
9.6	31	31	31	31	0	0	
13.20	31	0	38	38	40	31	
16.40	31	0	31	31	0	31	
24.40	31	0	31	31	0	31	
32.00	41	0	0	41	41	0	
64.00	41	0	0	41	41	0	

5.2.2.1.3 Predictor allocation

There are three possible cases. In safety net only the mean removed LSF vectors are quantized with the multi stage quantizer. In MA predictive quantization the MA prediction error is quantized with the MSVQ. In switched safety net /AR predictive there is a selection between quantizing the mean removed LSF vector and the AR prediction error. Table 24 specifies for each coding type and each bandwidth which quantization scheme is used. The values in the table indicate safety net (0), MA prediction (1), and AR prediction combined with safety net (2). The value "-1" indicates that the corresponding mode is not used. The coding modes that employ switched safety net/ AR prediction use one bit to signal which one of the two variants is used.

Table 24: Predictive mode type for LSF quantizer

	Inactive	Unvoiced	Voiced	Generic	Transition	Audio
Narrowband	1	1	2	2	0	2
Wideband <9.6kbps	1	1	2	2	0	2
Wideband 16kHz	1	-1	2	1	0	1
Wideband >=9.6kbps	1	1	2	1	0	1

The predictor values are optimized for all quantizer modes. For a given coding mode and bandwidth, all bitrates use the same predictor values. In general LSF values for voiced speech are considered quite stable over several consecutive frames. Consequently the corresponding AR predictor has the highest coefficient values. Other AR predictor coefficients are slightly lower. For the MA predictor the same value of 1/3 is used everywhere. The value is significantly lower than for AR coefficients since the quantization error starts oscillating over time if the MA coefficient is too large. The value is experimentally chosen to provide reasonable prediction efficiency, stability and good error recovery.

5.2.2.1.4 LSF quantizer structure

A safety net, predictive or switched safety-net predictive multi-stage vector quantizer (MSVQ) is used to quantize the full length frame-end LSF vector for all modes except voiced mode at 16 kHz internal sampling frequency. The last stage of the MSVQ is a multiple scale lattice vector quantizer (MSLVQ) [22]. For each coding mode number of 1 to 4 unstructured VQ stages are used followed by a MSLVQ stage. The number of stages, number of bits per each stage and the codebook names for each coding mode are detailed in table 25. The codebook names are mentioned to illustrate how some of the codebooks are reused between modes.

Table 25: Optimized codebooks and their bit allocation for LSF quantizers

Coding mode	Bits VQ safety net	Bits in VQ stages – safety net	Codebooks	Bits VQ predictive mode	Bits in VQ stages predictive mode	Codebooks
Inactive NB	-	-	-	5	5	IAA_MA1
Unvoiced NB	-	-	-	8	4+4	UVD_MA1 UVD_MA2
Voiced NB	8	4+4	SVNB_SN1 SVNB_SN2	6	3+3	GESVNB_AR1 GESVNB_AR2
Generic NB	9	5+4	GETRNB_SN1 GETRNB_SN2	6	3+3	GESVNB_AR1 GESVNB_AR2
Transition NB	9	5+4	GETRNB_SN1 GETRNB_SN2	-	-	-
Audio NB	4	4	AUNB_SN1	0	0	-
Inactive WB	-	-	-	5	5	IAA_MA1
Unvoiced WB	-	-	-	12	4+4+4	UVD_MA1 UVD_MA2 UVWB_MA3
Voiced WB	8	4+4	SVWB_SN1 SVWB_SN2	6	3+3	GESVWB_AR1 GESVWB_AR2
Generic WB	9	5+4	GETRWB_SN1 GETRWB_SN2	6	3+3	GESVWB_AR1 GESVWB_AR2
Transition WB	9	5+4	GETRWB_SN1 GETRWB_SN2	-	-	-
Audio WB	4	4	AUWB_SN1	0	0	-
Inactive WB2	-	-	-	5	5	IAA_MA1
Unvoiced WB2	-	-	-	-	-	-
Voiced WB2	-	-	BC-TCVQ	-	-	BC-TCVQ
Generic WB2	-	-	-	5	5	GEWB2_MA1
Transition WB2	8	4+4	TRWB2_SN1 TRWB2_SN2	-	-	-
Audio WB2	-	-	-	5	5	AUWB2_MA1
CNG	4	4	CNG_SN1	-	-	-
Generic WB >= 9.6kbps	-	-	-	5	5	GEWB_MA1

The WB2 voiced mode is using BC-TCVQ technology detailed in subclause 5.2.2.1.5.

Overall the optimized VQ codebooks use 14,368 kBytes and the MSLVQ parameters use 9.304 kBytes, including CNG mode.

The remaining LSF quantizer bits are used for the MSLVQ stage. The quantization in all the stages is done such that it minimizes a weighted Euclidean distortion. The calculation of the weights is detailed in subclause 5.2.2.2.1. The search in the multi-stage quantizer is done such that at most 2 candidates are kept per stage. For each candidate obtained in the search in the unstructured optimized VQ, a residual LSF vector is formed by subtracting from the LSF vector the codevectors obtained in each unstructured VQ stage. If there is one optimized VQ stage two residual LSF vectors are obtained, if there are two optimized VQ stages, 4 candidates are obtained and so on.

Each residual LSF vector is split into two 8-dimensional sub vectors. Each sub vector is coded as follows. The lattice codebook obtained through the reunion of three D_8^+ lattice truncations differently scaled. Each lattice truncation has a different number of leader classes. The leader classes contained in the lattice truncations are given in table 26.

Table 26: Lattice leader class vectors

Leader class index	Leader class vector	Leader class index	Leader class vector
0	1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	25	3.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0
1	0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	26	3.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0
2	1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0,	27	1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 0.5, 0.5
3	2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	28	2.5, 1.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5
4	1.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	29	2.5, 2.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
5	1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0	30	3.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
6	2.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0	31	2.0, 2.0, 2.0, 1.0, 1.0, 1.0, 1.0, 0.0
7	1.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	32	2.0, 2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0
8	1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0	33	3.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
9	2.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0	34	3.0, 2.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0
10	2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	35	4.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
11	1.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5	36	1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 0.5
12	2.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	37	2.5, 1.5, 1.5, 1.5, 1.5, 0.5, 0.5, 0.5
13	2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0	38	2.5, 2.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5
14	2.0, 2.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0	39	3.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
15	3.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0	40	2.0, 2.0, 2.0, 2.0, 1.0, 1.0, 0.0, 0.0
16	1.5, 1.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5	41	3.0, 2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0
17	2.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5	42	3.0, 2.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0
18	2.0, 2.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0	43	3.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
19	2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0	44	4.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0
20	3.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0	45	1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 1.5
21	1.5, 1.5, 1.5, 1.5, 1.5, 0.5, 0.5, 0.5	46	2.5, 1.5, 1.5, 1.5, 1.5, 1.5, 0.5, 0.5
22	2.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5	47	2.5, 2.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5
23	2.0, 2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0	48	3.5, 1.5, 1.5, 0.5, 0.5, 0.5, 0.5, 0.5
24	2.0, 2.0, 2.0, 1.0, 1.0, 0.0, 0.0, 0.0	49	

Given the bitrate available for the lattice codebook, the codebook is thus defined by a set of three integers representing the number of leader vectors for each truncation and three positive real number representing the scale for each lattice truncation. For instance a multiple scale lattice structure is defined by the number of leaders (20, 14, 5, 16, 10, 0) and the scales (1.057, 1.794, 2.896, 1.154, 1.860, 0.0). It means that the first subvector is quantized with a structure having three lattice truncations having 20, 14, and 5 leader classes respectively, which are scaled with the scales 1.057, 1.794, 2.896, respectively. The second subvector has only two truncations having 16 and 10 leader classes respectively. The truncations are ordered such that, for each subvector, their number of leader classes is descendingly ordered.

The difference in number of bits between the total number of bits for LSF end encoding, the prediction bit if needed and the number of bits used for unstructured VQ is used for the MSLVQ stage. The quantization in all the stages is done such that it minimizes a weighted Euclidean distortion. The calculation of the weights is detailed in subclause 5.2.2.1.1.

Suppose x is the current LSF 8-dimensional sub vector and w its corresponding weight vector. The vector x is normalized, i.e. component wise multiplied with the inverse of the off line estimated standard deviation. The resulting vector x is further sorted in descending order based on the absolute value of its components and the weights vector is arranged following the same order. Let x' be the vector of descendingly sorted absolute values of x and w' the correspondingly sorted weights vector. The weighted distance to the best codevector of each leader class corresponds to:

$$\left\| x' - s_j l_k \right\|_{w'}^2 = \sum_{i=1}^{8} x'^2(i)w'(i) - 2s_j \sum_{i=1}^{8} w'(i)x'(i)l_k(i) + s_j^2 \sum_{i=1}^{8} w'(i)l_k^2(i) \qquad (482)$$

where l_k is the leader vector corresponding to class k and s_j is the scale of the truncation j. Each lattice codebook has at most 3 truncations with their corresponding scales. Each truncation has a given number of leader vector classes. The sum of cardinalities of the classes for the truncations forming the codebook for the first LSF subvector and for the second subvector are within the number of bits for the considered operating point given by the overall bitrate and bandwidth. Computing in the transformed input space the second and the third terms from equation (482) directly gives a relative measure of goodness for the best codevector from the leader class k and truncation j which may be considered as a potential codevector for the truncation j and the leader class k.

$$d_{kj} = -2s_j \sum_{i=1}^{8} w'(i)x'(i)l_k(i) + s_j^2 \sum_{i=1}^{8} w'(i)l_k^2(i) \tag{483}$$

The part of equation (483) that is independent of the scale is calculated only once for all the leader classes from the first truncation, which is the one having the highest number of leader classes. When adding the last term to the first sum of equation (483) the product $x'(8)l_k(8)$ is considered with negative sign if the parity constraint of the leader k is not obeyed by the signs of the vector x. The contribution of the scale values is considered only afterwards in order to obtain the value d_{kj}. The leader class vector k and the truncation j having the smallest d_{kj} correspond to the codevector of the current input vector. The inverse permutation of the sorting operation on the input vector applied on the winning leader vector l_k gives the lattice codevector after applying also the corresponding signs. If the parity of leader vector l_k is 0 the signs are identical to the signs of the input vector. If the parity is 1 the signs are similar to the signs of the input vector with the constraint that the number of negative components is even. If the parity is -1 the signs are similar with signs of the input vector with the constraint that the number of negative components is odd. The final codevector is obtained after multiplication with the scale s_j and with the inverse of the component-wise off-line computed standard deviation. The standard deviations are individually estimated for each coding mode and bandwidth.

The candidate quantized LSF vectors are obtained by adding each lattice quantized residual to the corresponding candidates from the upper stages. The obtained candidates are increasingly sorted. For each sorted candidate the weighted Euclidean distortion with respect to the original LSF vector is calculated. The candidate that minimizes this distortion is selected as codevector to be encoded. The indexes corresponding to the first unstructured optimized VQ codebooks together with the index in the lattice codebook are written in the bitstream. The index for the lattice codebook is obtained as described in subclause 5.2.2.1.4.2.

For the CNG mode, using a total of 29 bits for the LSF quantization, the multiple scale lattice codebook structure is specific to each of the 16 codevectors obtained in the first stage. In addition based on the value of the last component of the 16 dimensional LSF vector only part of the first stage codebook is searched. If the last component of x is larger than 6350 then the search is done only for the first 6 codevectors of the first stage and the LSF vector corresponds to internal sampling frequency of 16kHz, otherwise the search is performed within the last 10 codevectors of the first stage.

5.2.2.1.4.1 Selection between safety net and predictive mode

For the modes where switched safety-net prediction is allowed the selection between the two is done as follows. For frame error concealment reasons safety net is imposed, and variable $force\_sf$ set to 1, under the following conditions:

- first three ACELP frames after an HQ frame

- in voiced class signals, if the frame erasure mode LSF estimate of the next frame based on the current frame is at a distance from the current frame LSF vector larger than 0.25. The distance, or stability factor, is calculated as:

$$sf = 1.25 - \frac{256}{400000} \frac{D}{frame\_len} \tag{484}$$

where frame_len is the frame length of the current frame and D is the Euclidean distance between the current frame LSF vector and the FER estimate for the next frame. In this case $force\_sf$ calculated at the current frame is stored in memory for use at the subsequent frame, thereby forcing the safety net decision for the subsequent frame when $force\_sf$ is equal to 1.

- some cases of rate switching

Safety net usage is decided by the following code line:

```
if ( force_sf || Err[0] < abs_threshold || Err[0]*(*streaklimit) < 1.05 * Err[1])
```

Thus the safety net mode is selected if force_sf is enabled or if for the quantized safety net codevector the quantization distortion (weighted Euclidean distance) is smaller than abs_threshold of 41000 for NB or 45000 for WB frames. For these relatively low error values the quantization is already transparent to original LSF values and it makes sense from the error recovery point of view to use safety-net as often as possible. Finally the safety net quantized error is compared to the predictively quantized error, with scaling of 1.05 to prefer safety net usage as well using *streaklimit multiplying

factor that is adaptive to the number of consecutive predictive frames. The *streaklimit factor gets smaller, when the streak of continuous predictive frames gets longer. This is done in order to restrict the very long usage streaks of predictive frames for frame-erasure concealment reasons. For voiced speech longer predictive streaks are allowed than for other speech types. In voiced mode streak limiting starts after 6 frames, in other modes after 3 frames.

5.2.2.1.4.2 Indexing of the lattice codevector

The indexes of each one of the two multiple scale lattice codevectors is composed of the following entities:

- scale indexes j_1, j_2 for the two 8-dimensional subvectors

- leader class index, k_1, k_2 for the two 8-dimensional subvectors

- leader permutation index, I_{l1}, I_{l2} unsigned permutation index

- sign index with parity constraint, I_{s1}, I_{s2}

- scale offset $O_s(j_i), i = 1,2$ the number of codevectors corresponding to the truncations with smaller scale indexes

- leader offset $O_l(k_i), i = 1,2$ the number of codevectors corresponding to leader classes with smaller leader indexes

- $\pi_0(k_i)$, i=1,2 cardinality of unsigned leader class, i.e. number of unsigned permutations in the class, shown in table 27.

- N_2 is the number of codevectors for the second subvector

The index for each subvector is calculated using

$$I_i = O_s(j_i) + O_l(k_i) + (I_{si}\pi_0(k_i) + I_{li}) \; for \; i = 1,2 \tag{485}$$

The indexes I_{li} and I_{si} are obtained using the position encoding based on counting the binomial coefficients and the sign encoding described in [26].

Table 27: Cardinality of unsigned leader vector permutations

Leader vector index	π_0	Leader vector index	π_0	Leader vector index	π_0
0	28	17	56	34	1120
1	1	18	420	35	8
2	70	19	56	36	8
3	8	20	280	37	280
4	8	21	56	38	168
5	28	22	168	39	56
6	168	23	28	40	420
7	28	24	560	41	336
8	1	25	168	42	840
9	280	26	336	43	28
10	28	27	28	44	168
11	56	28	280	45	1
12	8	29	28	46	168
13	56	30	8	47	420
14	420	31	280	48	168
15	56	32	70		
16	70	33	8		

The binomial encoding used for calculating I_{l1} and I_{l2} uses the fact that the cardinality of an unsigned leader class with distinct values $v_0, \ldots, v_{n-1}$, each having the number of occurrences $k_0, \ldots, k_{n-1}$ is given by:

$$\begin{pmatrix} S \\ k_0...k_{n-1} \end{pmatrix} = \begin{pmatrix} S \\ k_0 \end{pmatrix}\begin{pmatrix} S-k_0 \\ k_1 \end{pmatrix}...\begin{pmatrix} S-\sum_{i=0}^{n-2}k_i \\ k_{n-1} \end{pmatrix}. \tag{486}$$

The distinct values for each leader class vector and the number of each value in each leader class vector are given in the following table:

Table 28: Leader vector distinct values, their number of occurrences, and leader vector parities

Leader class index	Distinct values	Number of occurrences	Parity	Leader class index	Distinct values	Number of occurrences	Parity
0	1, 0,	2,6	0	25	3.0, 1.0, 0.0	1,5,2	0
1	0.5	8	1	26	3.0, 2.0, 1.0, 0.0	1,1,1,5	0
2	1, 0	4,4	0	27	1.5, 0.5	6,2	1
3	2, 0	1,7	0	28	2.5, 1.5, 0.5,	1,3,4	-1
4	1.5, 0.5	1,7	-1	29	2.5, 0.5	2,6	1
5	1.0, 0.0	6,2	0	30	3.5, 0.5,	1,7	-1
6	2.0, 1.0, 0.0,	1,2,5	0	31	2.0, 1.0, 0.0	3,4,1	0
7	1.5, 0.5	2,6	1	32	2.0, 0.0	4,4	0
8	1.0	8	0	33	3.0, 1.0	1,7	0
9	2.0, 1.0, 0.0	1,4,3	0	34	3.0, 2.0, 1.0, 0.0	1,1,3,3	0
10	2.0, 0.0	2,6	0	35	4.0, 0.0,	1,7	0
11	1.5, 0.5	3,5	-1	36	1.5, 0.5	7,1	-1
12	2.5, 0.5	1,7	1	37	2.5, 1.5, 0.5	1,4,3	1
13	2.0, 1.0, 0.0	1,6,1	0	38	2.5, 1.5, 0.5	2,1,5	-1
14	2.0, 1.0, 0.0	2,2,4	0	39	3.5, 1.5, 0.5	1,1,6	1
15	3.0, 1.0, 0.0	1,1,6	0	40	2.0, 1.0, 0.0	4,2,2	0
16	1.5, 0.5	4,4	1	41	3.0, 2.0, 1.0, 0.0	1,1,5,1	0
17	2.5, 1.5, 0.5	1,1,6	-1	42	3.0, 2.0, 1.0, 0.0	1,2,1,4	0
18	2.0, 1.0, 0	2,4,2	0	43	3.0, 0.0	2,6	0
19	2.0, 0.0	3,5	0	44	4.0, 1.0, 0.0	1,2,5	0
20	3.0, 1.0, 0.0,	1,3,4	0	45	1.5	8	1
21	1.5, 0.5	5,3	-1	46	2.5, 1.5, 0.5	1,5,2	-1
22	2.5, 1.5, 0.5	1,2,5	1	47	2.5, 1.5, 0.5	2,2,4	1
23	2.0, 1.0	2,6	0	48	3.5, 1.5, 0.5	1,2,5	-1
24	2.0, 1.0, 0.0	3,2,3	0	49			

The index for the two multiple scale lattice codevectors corresponding to the two residual LSF subvectors are combined in a single index, I, which is written in the bitstream.

$$I = N_2 I_1 + I_2. \tag{487}$$

5.2.2.1.5 LSFQ for voiced coding mode at 16 kHz internal sampling frequency : BC-TCVQ

5.2.2.1.5.1 Block-constrained trellis coded vector quantization (BC-TCVQ)

The VC mode operating at 16 kHz internal sampling frequency has two decoding rates: 31 bits per frame and 40 bits per frame. The VC mode is quantized by a 16-state and 8 stage block-constrained trellis coded vector quantization (BC-TCVQ) scheme.

Trellis coded vector quantization (TCVQ) [42] generalizes trellis coded quantization (TCQ) to allow vector codebooks and branch labels. The main feature of TCVQ is the partitioning of an expanded set of VQ symbols into subsets and the labelling of the trellis branches with these subsets. TCVQ is based on a rate-1/2 convolutional code, which has $N = 2^v$ trellis states and two branches entering/leaving each trellis state. Given a block of m source vectors, the Viterbi algorithm (VA) is used to find the minimum distortion path. This encoding procedure allows the best trellis path to begin in any of N initial states and end in any of N terminal states. In TCVQ, the codebook has $2^{(R+\tilde{R})L}$ vector codewords. $\tilde{R}$ is referred to as "codebook expansion factor" (in bits per dimension) since the codebook has $2^{\tilde{R}L}$ times as many codewords as a nominal rate-R VQ. The encoding is accomplished in the following two steps.

Step 1. For each input vector, find the closest codeword and corresponding distortion in each subset.

Step 2. Let the branch metric for a branch labelled with subset S be the distortion found in step 1 and use the VA to find the minimum distortion path through the trellis.

BC-TCVQ is a low-complexity approach that requires exactly one bit per source sample to specify the trellis path. Figure 22 shows the concept of 'block constrained' and illustrates the search process of the Viterbi algorithm with a 4-state and 8 stages trellis structure, which selects '00' and '10' as initial states. When the initial state is '00', the terminal state is selected to be one of '00' or '01' and when the initial state is '10', the terminal state is selected to be one of '10' or '11'. As an example, the survival path from the initial stage with state '00' to the stages ($NS - \log_2 4$) with state '00' is shown by a dotted line. In this case, the only two possible trellis paths for the last two stages are toward states '00' and '01'. This example uses one bit for the initial state and one bit for the terminal state. If the terminal state is decided, the path information for the last two stages is not needed.
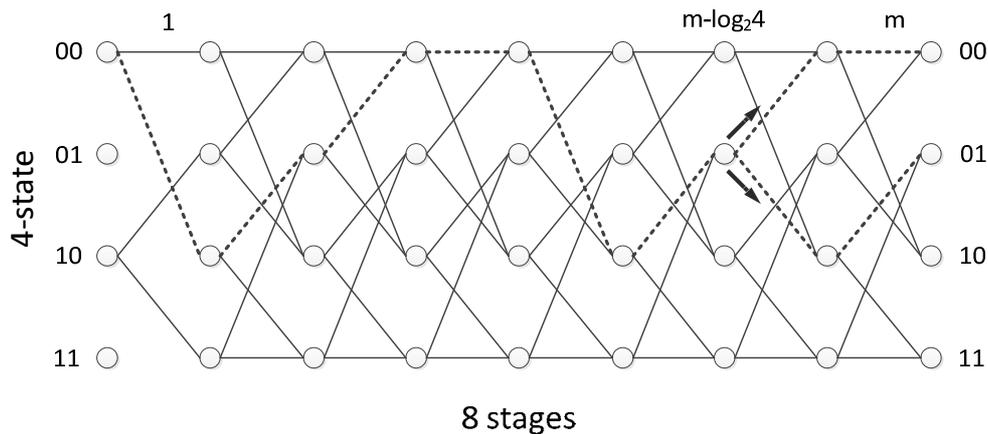
Figure 22: Block constrained concept in 4-state and 8 stages trellis structure for BC-TCVQ encoding

For any $0 \le k \le \upsilon$, consider a BC-TCVQ structure that allows 2^k initial trellis states and exactly $2^{\upsilon-k}$ terminal trellis states for each allowed initial trellis state. A single VA encoding, starting from the allowed initial trellis states, proceeds in the normal way up to the vector stage $m - k$. It takes k bits to specify the initial state, and $m - k$ bits to specify the path to vector stage $m - k$. A unique terminating path, possibly dependent on the initial trellis state, is pre-specified for each trellis state at vector stage $m - k$ through vector stage m. Regardless of the value of k, the encoding complexity is only a single VA search of the trellis, and exactly m bits are required to specify an initial trellis state and a path through the trellis.

The BC-TCVQ for VC mode at a 16kHz internal sampling frequency utilizes 16-state ($N =16$) and 8-stage ($NS =8$) TCVQ with 2-dimensional ($L =2$) vector. LSF subvectors with two elements are allocated to each stage. Table 29 shows the initial states and terminal states for 16-state BC-TCVQ. In this case the parameters k and v are 2 and 4, respectively. Four bits are used for both the initial state and terminal state.

Table 29: Initial state and terminal state for 16-state BC-TCVQ

Initial state	Terminal state
0	0, 1, 2, 3
4	4, 5, 6, 7
8	8, 9, 10, 11
12	12, 13, 14, 15

5.2.2.1.5.2   Bit Allocations and codebook size for BC-TCVQ

The bit allocations for the LSF quantizer at 31 and 40 bits/frame are summarized in tables 30 and 31.

Table 30: Bit allocation for the LSF quantizer at 31 bits/frame

Parameters		Bit allocation
BC-TCVQ	Path information (Initial states + path + final states)	2+4+2
	Subset codewords	4 bits $\times 2$ (Stages 1 to 2) 3 bits $\times 2$ (Stages 3 to 4) 2 bits $\times 4$ (Stages 5 to 8)
Scheme selection		1
Total		31

Table 31: Bit allocation for the LSF quantizer at 40 bits/frame

Parameters		Bit allocations
BC-TCVQ	Path information (Initial states + path + final states)	2+4+2
	Subset codewords	4 bits $\times 2$ (Stages 1 to 2) 3 bits $\times 2$ (Stages 3 to 4) 2 bits $\times 4$ (Stages 5 to 8)
SVQ	Subset codewords	5 (1st vector with dim.=8) 4 (2nd vector with dim.=8)
Scheme selection		1
Total		40

Figures 23 and 24 show the LSF quantizer at 31 and 40 bits/frame, respectively. The 1st and 2nd BC-TCVQ use the same bit allocation but different codebook entries. The 3rd and 4th SVQ use the same bit allocation and codebooks. The 31 bit LSF quantizer uses BC-TCVQ and the 40 bit LSF quantizer uses both BC-TCVQ and SVQ.

The following table summarizes the codebook size for BC-TCVQ and SVQ. The overall codebook size is 2,432 words. In addition, there are several tables for BC-TCVQ such as intra-prediction coefficients (56 words), scale information (32 words) and branch information (192 words). The total codebook size is 2,712 words.

Table 32: Codebook size for BC-TCVQ and SVQ

	1st stage	2nd stage	3rd stage	4th stage	5th stage	6th stage	7th stage	8th stage	Total per frame
Bits for BC-TCVQ subcodebook	4	4	3	3	2	2	2	2	
Scalars for Predictive	256	256	128	128	64	64	64	64	1,024
Scalars for Safety-net	256	256	128	128	64	64	64	64	1,024
Bits for SVQ subcodebook	5				4				
Scalars	256				128				384
Total									2,432

5.2.2.1.5.3   Quantization scheme selection

The quantization scheme for the VC mode consists of Safety-net and Predictive schemes. The quantization scheme is selected in an open-loop manner as shown in the figures 23 and 24. The scheme selection is done by calculating the prediction error of unquantized LSFs.

The prediction error (E_k) of the k th frame is obtained from the inter-frame prediction contribution $p_k(i)$, the weighting function $w_{end}(i)$, and a mean-removed unquantized LSF $z_k(i)$ as

$$E_k = \sum_{i=0}^{M-1} w_{end}(i)\big(z_k(i) - p_k(i)\big)^2 \qquad (488)$$

where

$$p_k(i) = \rho(i)\hat{z}_{k-1}(i) \text{ , for i=0,\ldots,M} \qquad (489)$$

and $\rho(i)$ is the selected AR prediction coefficients for VC mode and $\hat{z}_{k-1}(i)$ is the mean-removed quantized LSF of the previous frame and M is the LPC order.

When E_k is bigger than a threshold, it implies the tendency of the current frame to be non-stationary. Then the safety-net scheme is a better choice. Otherwise the predictive scheme is selected. In addition, the streak limit (streaklimit) prevents the consecutive selection of the predictive scheme.

The quantization scheme selection is shown by the following pseudo-code.

```
If  Ek > streaklimit * op_loop_thr
    safety_net = 1;
else
    safety_net = 0;
```

where E_k is the prediction error of the kth frame and the open-loop threshold (op_loop_thr) is 3,784,536.3.

If the safety-net flag (safety_net) is set to 1, the safety-net scheme is selected, and if the safety-net flag (safety_net) is set to 0, the predictive scheme is selected. The scheme selection is encoded using a single bit.

5.2.2.1.5.4 31 bit LSF quantization by the predictive BC-TCVQ with safety-net

Figure 23 shows the predictive BC-TCVQ with safety-net for an encoding rate of 31 bits.
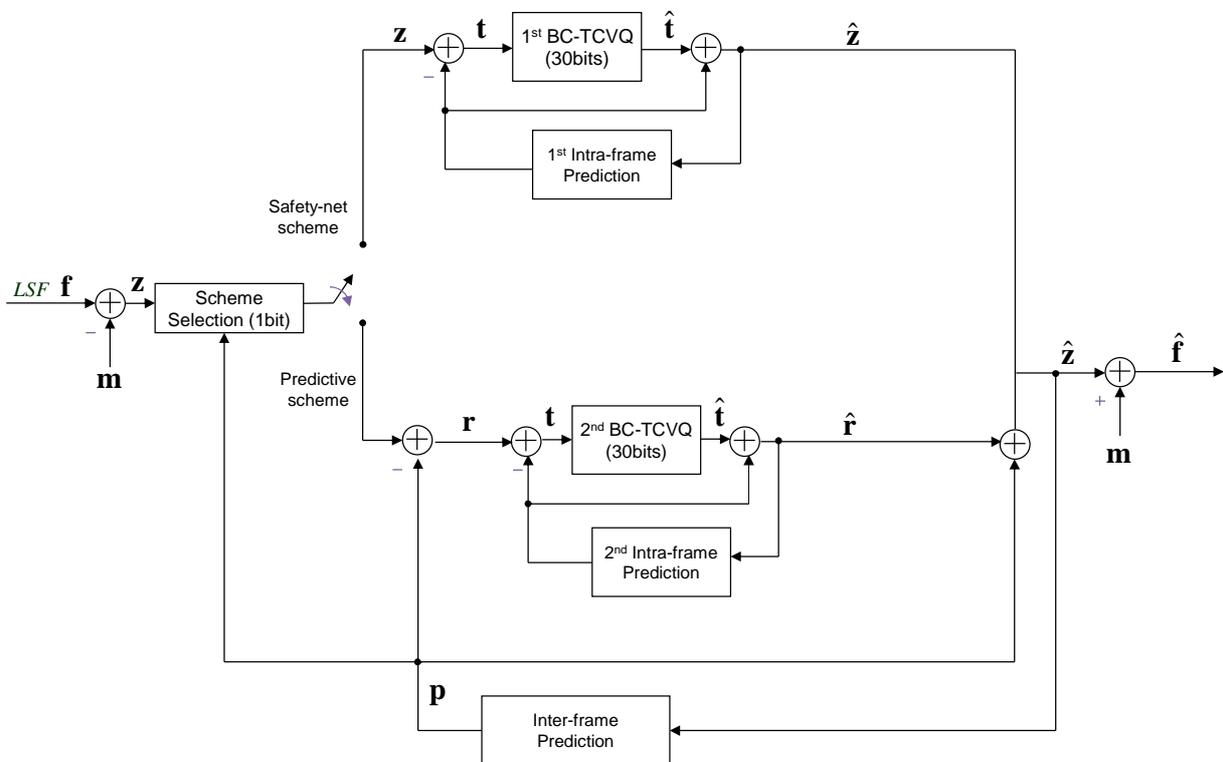
Figure 23: Block diagram of the predictive BC-TCVQ with safety-net for an encoding rate of 31bits/frame

The operation of the 31 bit LSF quantizer is described as follows. If the safety-net scheme is selected, the mean-removed LSF vector, $z_k(i)$, is quantized by the 1st BC-TCVQ and 1st intra-frame prediction with 30 bits. If the

predictive scheme is selected, the prediction error, $r_k(i)$, which is the difference between the mean-removed LSF vector $z_k(i)$ and the prediction vector $p_k(i)$ is quantized by the 2nd BC-TCVQ and 2nd intra-frame prediction with 30 bits.

An optimal index for each stage of BC-TCVQ is obtained by searching for an index which minimizes $E_{werr}(p)$ of equation (490).

$$E_{werr}(p) = \sum_{i=0}^{1} w_{end}(2(j-1)+i)\left(t_k'(2(j-1)+i) - c_j^p(i)\right)^2 \text{, for } p=1,..., P_j \text{ and } j=1,..., M/2 \quad (490)$$

where P_j is the number of codevectors in the j th sub-codebook, c_j^p is the pth codevector of j th the subcodebook, $w_{end}(i)$ is a weighting function, and $t_k' = [t_k^t(0), t_k^t(1),..., t_k^t(M/2-1)]$.

Intra-frame correlation typically remains in the inter-frame AR prediction error vectors. The presence of significant intra-frame correlation motivates the introduction of an intra-predictive coding scheme for the AR prediction error vector, as shown in figure 23, in order to increase the coding gain. The intra-frame prediction uses the quantized elements of the previous stage. The difference between $z_k(i)$ and its prediction is then quantized. The prediction is formed for each trellis node using the output codevectors specified by the survivor path associated with the particular node.

The prediction coefficients used for the intra-frame prediction is predefined by the codebook training process. The prediction coefficients are two-by-two matrices for the 2-dimensional vector. The intra-frame prediction process of BC-TCVQ is as follows. The prediction residual vector, $t_k(i)$, which is the input of the 1st BC-TCVQ, is computed as

$$t_k(0) = z_k(0)$$

$$t_k(i) = z_k(i) - \tilde{z}_k(i) \text{, for } i=1,..., M/2-1 \quad (491)$$

where

$$\tilde{z}_k(i) = \mathbf{A}_i \hat{z}_k(i-1) \text{, for } i=1,..., M/2-1 \quad (492)$$

where $\tilde{z}_k(i)$ is the estimation of $z_k(i)$, $\hat{z}_k(i-1)$ is the quantized vector of $z_k(i-1)$, and $\mathbf{A}_i$ is the prediction matrix with 2×2 which is computed as

$$\mathbf{A}_i = \mathbf{R}_{01}^i [\mathbf{R}_{11}^i]^{-1} \text{, for } i=1,..., M/2-1, \quad (493)$$

where

$$\mathbf{R}_{01}^i = [\mathbf{z}(i)\mathbf{z}^t(i-1)] \text{ and } \mathbf{R}_{11}^i = \lfloor \mathbf{z}(i-1)\mathbf{z}^t(i-1) \rfloor \quad (494)$$

and M is the LPC order.

Then

$$\hat{z}_k(i) = \hat{t}_k(i) + \tilde{z}_k(i) \text{, for } i=0,..., M/2-1. \quad (495)$$

The prediction residual, $t_k(i)$, is quantized by the 1st BC-TCVQ. The 1st BC-TCVQ and the 1st intra-frame prediction are repeated to quantize $z_k(i)$. Table 33 represents the designed prediction coefficients $\mathbf{A}_i$ for the BC-TCVQ in the safety-net scheme.

Table 33: Intra-frame prediction coefficients for the BC-TCVQ in the safety-net scheme

Coefficient Number	Coefficient Value
$\mathbf{A}_1$	$\begin{bmatrix} -0.452324 & 0.808759 \\ -0.524298 & 0.305544 \end{bmatrix}$
$\mathbf{A}_2$	$\begin{bmatrix} 0.009663 & 0.606028 \\ -0.013208 & 0.421115 \end{bmatrix}$
$\mathbf{A}_3$	$\begin{bmatrix} 0.144877 & 0.673495 \\ 0.080963 & 0.580317 \end{bmatrix}$
$\mathbf{A}_4$	$\begin{bmatrix} 0.208225 & 0.633144 \\ 0.215958 & 0.584520 \end{bmatrix}$
$\mathbf{A}_5$	$\begin{bmatrix} 0.050822 & 0.767842 \\ 0.076879 & 0.416693 \end{bmatrix}$
$\mathbf{A}_6$	$\begin{bmatrix} 0.005058 & 0.550614 \\ -0.006786 & 0.296984 \end{bmatrix}$
$\mathbf{A}_7$	$\begin{bmatrix} -0.023860 & 0.611144 \\ -0.162706 & 0.576228 \end{bmatrix}$

For the predictive scheme, $r_k(i)$ is quantized by the 2nd BC-TCVQ and the 2nd intra-frame prediction. An optimal index for each stage of BC-TCVQ is obtained by searching for an index which minimizes $E_{werr}(p)$ in equation (490).

The intra-frame prediction uses the same process with different prediction coefficients as that of the safety-net scheme. Then

$$\hat{r}_k(i) = \hat{t}_k(i) + \tilde{r}_k(i) \text{, for i=0,…,M/2-1.} \tag{496}$$

The prediction residual, $t_k(i)$, is quantized by the 2nd BC-TCVQ. The 2nd BC-TCVQ and the 2nd intra-frame prediction are repeated to quantize. Table 34 represents the designed prediction coefficients $\mathbf{A}_i$ for the BC-TCVQ in the predictive scheme.

Table 34: Intra-frame prediction coefficients for the BC-TCVQ in the predictive scheme

Coefficient Number	Coefficient Value
$\mathbf{A}_1$	$\begin{bmatrix} -0.292479 & 0.676331 \\ -0.422648 & 0.217490 \end{bmatrix}$
$\mathbf{A}_2$	$\begin{bmatrix} 0.048957 & 0.500476 \\ 0.087301 & 0.287286 \end{bmatrix}$
$\mathbf{A}_3$	$\begin{bmatrix} 0.199481 & 0.502784 \\ 0.106762 & 0.420907 \end{bmatrix}$
$\mathbf{A}_4$	$\begin{bmatrix} 0.240459 & 0.440504 \\ 0.214255 & 0.396496 \end{bmatrix}$
$\mathbf{A}_5$	$\begin{bmatrix} 0.193161 & 0.494850 \\ 0.158690 & 0.306771 \end{bmatrix}$
$\mathbf{A}_6$	$\begin{bmatrix} 0.093435 & 0.370662 \\ 0.065526 & 0.148231 \end{bmatrix}$
$\mathbf{A}_7$	$\begin{bmatrix} 0.037417 & 0.336906 \\ -0.024246 & 0.187298 \end{bmatrix}$

5.2.2.1.5.5   40 bit LSF quantization using the predictive BC-TCVQ/SVQ with safety-net

Figure 24 shows the predictive BC-TCVQ/split-VQ(SVQ) with safety-net for an encoding rate of 40 bits. Both 31 bit LSF quantizer and 40 bit LSF quantizer use the same codebook for BC-TCVQ.
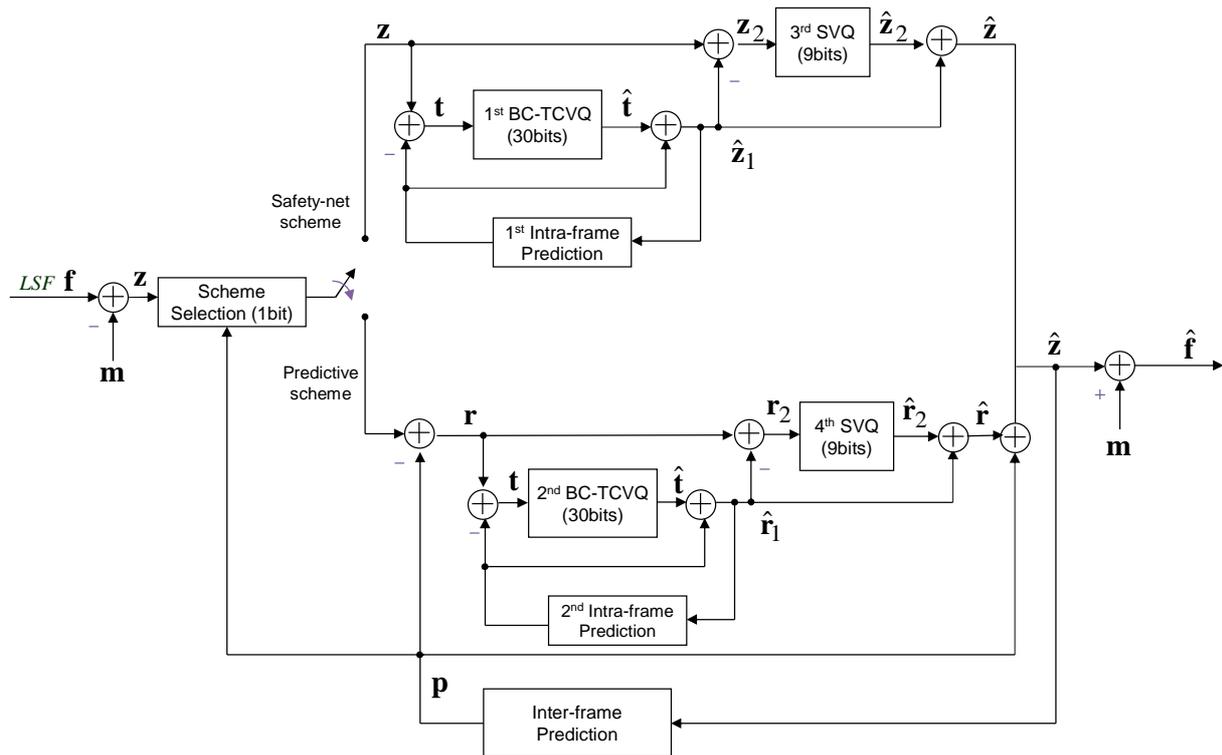
Figure 24: Block diagram of the predictive BC-TCVQ/SVQ with safety-net for an encoding rate of 40 bits/frame

In the LSF quantization for an encoding rate of 40 bit/frame, the difference between the mean-removed LSF and its BC-TCVQ output is quantized by the 3rd and 4th SVQ, as shown in figure 24. The scheme selection, 1st and 2nd BC-TCVQ, and 1st and 2nd intra-frame prediction blocks of the 40 bit LSF quantizer are exactly same as those of the 31 bit LSF quantizer. Both LSF quantizers use same codebooks for the BC-TCVQ.

If the current coding mode in the scheme selection block is selected as the predictive scheme, the prediction error $\mathbf{r}$ is derived by subtracting $\mathbf{p}$ from the mean-removed LSF $\mathbf{z}$. It is quantized by the 2nd BC-TCVQ and the 2nd intra-frame prediction. The residual signal $\mathbf{r}_2$ is obtained by subtracting $\hat{\mathbf{r}}_1$ from $\mathbf{r}$. The residual signal $\mathbf{r}_2$ is then split into two sub-vectors of dimensions 8 and 8, and is quantized using the 4th SVQ. Since the low band is perceptually more important than the high band, five bits are allocated to the 1st 8-dimensional VQ and four bits are allocated to the 2nd 8-dimensional VQ. $\mathbf{r}_2$ is quantized by the 4th SVQ to produce $\hat{\mathbf{r}}_2$. $\hat{\mathbf{r}}$ is then obtained by adding $\hat{\mathbf{r}}_2$ to $\hat{\mathbf{r}}_1$. Finally the predictive scheme output $\hat{\mathbf{z}}$ is derived by adding $\mathbf{p}$ to $\hat{\mathbf{r}}$.

If the current coding mode is selected as the safety-net scheme, the mean-removed LSF $\mathbf{z}$ is quantized by the 1st BC-TCVQ and the 1st intra-frame prediction. The residual signal $\mathbf{z}_2$ is extracted by subtracting $\hat{\mathbf{z}}_1$ from $\mathbf{z}$, and it is quantized by the 3rd SVQ to produce $\hat{\mathbf{z}}_2$. The 3rd SVQ is exactly same as the 4th SVQ. That is, both SVQ quantizers use same codebooks. Because the input distribution of the 3rd SVQ is different from that of the 4th SVQ, scaling factors are used to compensate the difference. Scaling factors are computed by considering the distribution of both residual signals $\mathbf{z}_2$ and $\mathbf{r}_2$. To minimize the computational complexity in in an actual implementation, the input signal $\mathbf{z}_2$ of the 3rd SVQ is divided by the scaling factor, and the resulting signal is quantized by the 3rd SVQ. The quantized signal $\hat{\mathbf{z}}_2$ of the 3rd SVQ is obtained by multiplying the quantized output with the scaling factor. Table 35 shows the scaling factors for the quantization and de-quantization. Finally, the quantized mean-removed LSF $\hat{\mathbf{z}}$ is derived by adding $\hat{\mathbf{z}}_2$ to $\hat{\mathbf{z}}_1$.

Table 35: Scaling factor for the SVQ

Dimension	0	1	2	3	4	5	6	7
Inverse scale factor for quantization	0.5462	0.5434	0.5553	0.5742	0.5800	0.5725	0.6209	0.6062
Scale factor for de-quantization	1.8307	1.8404	1.8009	1.7416	1.7240	1.7467	1.6106	1.6497
Dimension	8	9	10	11	12	13	14	15
Inverse scale factor for quantization	0.6369	0.6432	0.6351	0.6173	0.6397	0.6562	0.6331	0.6404
Scale factor for de-quantization	1.5702	1.5548	1.5745	1.6199	1.5633	1.5239	1.5796	1.5615

5.2.2.1.6 Mid-frame LSF quantizer

For a more accurate representation of the spectral envelope during signal transitions, the encoder quantizes mid-frame LSF coefficients. In contrast to the frame-end LSF vector, the mid-frame LSF vector is not quantized directly. Instead, a weighting factor is searched in a codebook to calculate a weighted average between the quantized LSF vectors of the current and the previous frames. Only 2-6 bits are required depending on the bitrate and the coding mode (see Table 35a).

Table 35a: Bit allocation in mid-frame LSF quantization

Bitrate [bps]	IC	UC	VC	GC	TC	AC
7200	2	5	4	5	5	2
8000	2	5	4	5	5	2
9600	2	5	4	5	0	0
13200	2	0	5	5	5	2
16400	4	0	5	5	0	0
24400	5	0	5	5	0	0
32000	5	0	0	5	5	5
64000	5	0	0	5	5	5

Before searching the codebook, the unquantized mid-frame LSF vector is weighted with the LSF weighting function defined in Equation (481). For simplicity, the following description will be provided by using LSP vectors instead of LSF vectors. These two vectors are related by the following simple relation $q(k) = \cos(\omega(k))$ where $q(k)$ is the kth LSP coefficient and $\omega(k)$ is kth LSF coefficient. The mid-frame LSP weighting can be expressed using the following formula

$$q_{wmid}(k) = W(k)q_{mid}(k), \text{ for } k=0,\dots,M\text{-}1. \tag{496a}$$

where $q_{mid}(k)$ is the kth unquantized LSP coefficient and $W(k)$ is kth weighting factor of the function defined in Equation (481). Note, that this is not the weighting factor which is quantized. This weighting is based on the FFT spectrum where more weight is put on perceptually important part of the spectrum and less weight elsewhere.

The weighting factor to be quantized is a vector of size M that is searched in a closed-loop fashion such that the error between the quantized mid-frame LSP coefficients and this weighted representation is minimized in a mean-square sense. That is

$$E_{mid} = \sum_{k=0}^{M-1}\left[q_{wmid}(k) - \left[(1 - f_{mid}(k))q_{wend}^{[-1]}(k) + f_{mid}(k)q_{wend}(k)\right]\right]^2 \tag{496b}$$

where $q_{wend}(k)$ is kth quantized weighted end-frame LSP coefficient and f_{mid} is the mid-frame weighting vector taken from the codebook. To save computation complexity, both operations are combined. That is

$$E_{mid} = \sum_{k=0}^{M-1} w_f(k) \Big[q_{mid}(k) - \big[(1 - f_{mid}(k)) q_{end}^{[-1]}(k) + f_{mid}(k) q_{end}(k) \big] \Big]^2 \qquad (496c)$$

Once the winning weighting factor is found, the quantized LSP vector is reordered to maintain a stable LP filter. After the quantization, the end-frame and the mid-frame LSF vectors are used to determine the quantized LP parameters in each subframe. This is done in the same way as for unquantized LP parameters (see Equation (58) in Clause 5.1.96).

5.2.3 Excitation coding

The excitation signal coding depends on the coding mode. In general it can be stated that in the absence of DTX/CNG operation, the excitation signal is coded per subframes of 64 samples. This means that it is encoded four times per frame in case of 12.8 kHz internal sampling rate and five times per frame in case of 16 kHz internal sampling rate. The exception is the GSC coding where longer subframes can be used to encode some components of the excitation signal, especially at lower bitrates.

The excitation coding will be described in the following subclauses, separately for each coding mode. The description of excitation coding starts with the GC and VC modes. For the UC, TC, and GSC modes, it will be described in subsequent subclauses with references to this subclause.

5.2.3.1 Excitation coding in the GC, VC and high rate IC/UC modes

The GC,VC and high rate IC/UC modes are very similar and are described together. The VC mode is used in stable voiced segments where the pitch is evolving smoothly within an allowed range as described in subclause 5.1.13.2. Thus, the major difference between the VC and GC modes is that more bits are assigned to the algebraic codebook and less to the adaptive codebook in case of the VC mode as the pitch is not allowed to evolve rapidly in the VC mode. The high-rate IC and UC modes are similar and are used for signalling inactive frames where only a background noise is detected, and unvoiced frames, respectively. The two modes differ from GC mode mainly by their specific gain coding codebook. The GC mode is then used in frames not assigned to a specific coding mode during the signal classification procedure and is aimed at coding generic speech and audio frames. The principle of excitation coding is shown in a schematic diagram in figure 25. The individual blocks and operations are described in detail in the following subclauses.
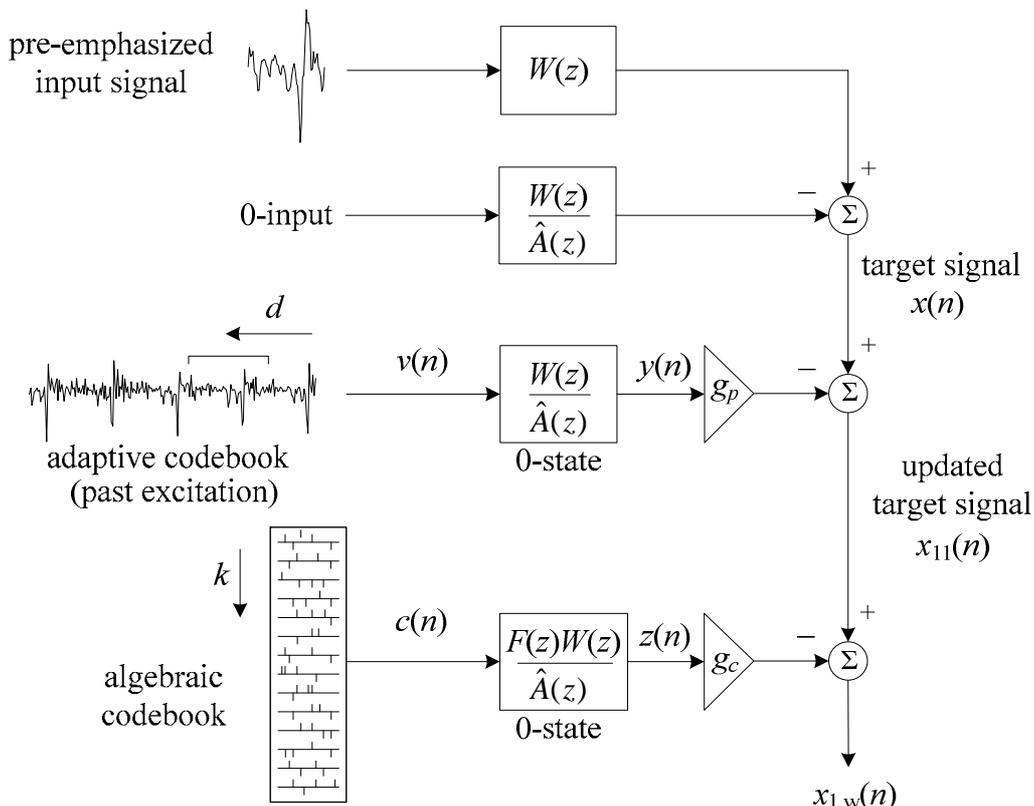
Figure 25: Schematic diagram of the excitation coding in GC and VC mode

5.2.3.1.1   Computation of the LP residual signal

To keep the processing flow similar for all coding modes, the LP residual signal is computed for the whole frame in the first processed subframe of each frame, as this is needed in the TC mode. For each subframe, the LP residual is given by

$$r(n) = s_{pre}(n) + \sum_{i=1}^{16} \hat{a}_i s_{pre}(n-i), \qquad n = 0, \ldots, 63 \tag{497}$$

where $s_{pre}(n)$ is the pre-emphasized input signal, defined in subclause 5.1.4 and $\hat{a}_i$ are the quantized LP filter coefficients, described in subclause 5.2.2.1.

In DTX operation the computed LP residual signal $r(n)$ is attenuated by multiplying an attenuation factor att for all input bandwidths except NB. The attenuation factor is calculated as

$$att = \frac{1}{1 + \dfrac{flr}{6} burstho_{cnt}} \tag{497a}$$

where $burstho_{cnt}$ as determined in subclause 5.6.2.1.1 is upper limited by HO_HIST_SIZE, $flr = 0.6$ if the bandwidth is not WB or the latest bitrate used for actively encoded frames R_{latest_active} is larger than 16.4 kbps. Otherwise flr is determined from a hangover attenuation table as defined in Table 35b. flr is only updated in the first SID frame after an active signal period if two criteria are both fulfilled. The first criterion is satisfied if AMR-WB IO mode is used or the bandwidth=WB. The second criterion is met if the number of consecutive active frames in the latest active signal segment was at least $MIN\_ACT\_CNG\_UPD = 20$ number of frames or if the current SID is the very first encoded SID frame. The attenuation factor att is finally lower limited to flr.

Table 35b: Attenuation floor

Latest active bitrate [kbps]	flr
$R_{latest\_active} \leq 7.2$	0.5370318
$7.2 < R_{latest\_active} \leq 8.0$	0.6165950
$8.0 < R_{latest\_active} \leq 9.6$	0.6839116
$9.6 < R_{latest\_active} \leq 13.2$	0.7079458
$13.2 < R_{latest\_active} \leq 16.4$	0.7079458

5.2.3.1.2   Target signal computation

The target signal for adaptive codebook search is usually computed by subtracting a zero-input response of the weighted synthesis filter $W(z)H(z) = A(z/\gamma_1)H_{de-emph}(z)/\hat{A}(z)$ from the weighted pre-emphasized input signal. This is performed on a subframe basis. An equivalent procedure for computing the target signal, which is used in this codec, is filtering of the residual signal, $r(n)$, through the combination of the synthesis filter $H(z) = 1/\hat{A}(z)$ and the weighting filter $W(z) = A(z/\gamma_1)H_{de-emph}(z)$. After determining the excitation signal for a given subframe, the initial states of these filters are updated by filtering the difference between the LP residual signal and the excitation signal. The memory update of these filters is explained in subclause 5.2.3.1.8. The residual signal, $r(n)$, which is needed for finding the target vector, is also used in the adaptive codebook search to extend the past excitation buffer. This simplifies the adaptive codebook search procedure for delays less than the subframe size of 64 as will be explained in the next subclause. The target signal in a given subframe is denoted as $x(n)$.

5.2.3.1.3 Impulse response computation

The impulse response, $h(n)$, of the weighted synthesis filter

$$W(z)H(z) = A(z/\gamma_1)H_{de-emph}(z)/\hat{A}(z) \tag{498}$$

is computed for each subframe. Note that $h(n)$ is not the impulse response of the filter $H(z)$, but of the filter $W(z)H(z)$. In the equation above, $\hat{A}(z)$, is the quantized LP filter, the coefficients of which are $\hat{a}_i$ (see subclause 5.2.2.1). This impulse response is needed for the search of adaptive and algebraic codebooks. The impulse response $h(n)$ is computed by filtering the vector of coefficients of the filter $A(z/\gamma_1)$, extended by zeros, through the two filters: $1/\hat{A}(z)$ and $H_{de-emph}(z)$.

5.2.3.1.4 Adaptive codebook

5.2.3.1.4.1 Adaptive codebook search

The adaptive codebook search consists of performing a closed-loop pitch search, and then computing the adaptive codevector, $v(n)$, by interpolating the past excitation at the selected fractional pitch lag. The adaptive codebook parameters (or pitch parameters) are the closed-loop pitch, T_{CL}, and the pitch gain, g_p (adaptive codebook gain), calculated for each subframe. In the search stage, the excitation signal is extended by the LP residual signal to simplify the closed-loop search. The adaptive codebook search is performed on a subframe basis. The bit allocation is different for the different modes.

In the first and third subframes of a GC, UC or IC frame, the fractional pitch lag is searched with a resolution in the range [34, 91½], and with integer sample resolution in the range [92, 231] depending on the bit-rate and coding mode. Closed-loop pitch analysis is performed around the open-loop pitch estimates. Always bounded by the minimum and maximum pitch period limits, the range [$d^{[0]}$–8, $d^{[0]}$+7] is searched in the first subframe, while the range [$d^{[1]}$–8, $d^{[1]}$+7] is searched in the third subframe. The pitch period quantization limits are summarized in table 36.

Table 36: Pitch period quantization limits

Rates (kbps)	Sampling rate of the limits (kHz)	IC/UC	VC	GC
7.2	12.8	n.a.	[17; 231]	[34; 231]
8.0	12.8	n.a.	[17; 231]	[20; 231]
9.6	12.8	n.a.	[29; 231]	[29; 231]
13.2	12.8	n.a.	[17; 231]	[20; 231]
16.4	16	n.a.	[36; 289]	[36; 289]
24.4	16	n.a.	[36; 289]	[36; 289]
32	16	[21; 289]	n.a.	[21; 289]
64	16	[21; 289]	n.a.	[21; 289]

For the second and fourth subframes, a pitch resolution depending on the bit-rate and coding mode is used and the closed-loop pitch analysis is performed around the closed-loop pitch estimates, selected in the preceding (first or third) subframe. If the closed-loop pitch fraction in the preceding subframe is 0, the pitch is searched in the range [T_I–8, T_I+7½], where $T_I = \lfloor T_{CL}^{[p]} \rfloor$ is the integer part of the fractional pitch lag of the preceding subframe (p is either 0, to denote the first subframe, or 3 to denote the third subframe). If the fraction of the pitch in the previous subframe is $\geq 1/2$, the pitch is searched in the range [T_I–7, T_I+8½]. The pitch delay is encoded as follows. In the first and third subframe, absolute values of the closed-loop pitch lags are encoded. In the third and fourth subframe, only relative values with respect to the absolute ones are encoded.

In the VC mode, the closed-loop pitch lag is encoded absolutely in the first subframe and relatively in the following 3 subframes. If the fraction of the closed-loop pitch of the preceding subframe is 0, the pitch is searched in the interval [T_I–4, T_I+3½]. If the fraction of the closed-loop pitch lag in the preceding subframe is $\geq 1/2$, the pitch is searched in the range [T_I–3, T_I+4½].

The closed-loop pitch search is performed by minimizing a mean-squared weighted error between the target signal and the past filtered excitation (past excitation, convolved with $h(n)$). This is achieved by maximizing the following correlation

$$C_{CL} = \frac{\sum_{n=0}^{63} x(n)y_k(n)}{\sqrt{\sum_{n=0}^{63} y_k(n)y_k(n)}} \tag{499}$$

where $x(n)$ is the target signal and $y_k(n)$ is the past filtered excitation at delay k. Note that negative indices refer to the past signal. Note also that the search range is limited around the open loop pitch lags, as explained earlier. The convolution of the past excitation signal with $h(n)$ is computed only for the first delay in the searched range. For other delays, it is updated using the recursive relation

$$y_k(n) = y_{k-1}(n-1) + u(-k)h(n), \qquad n = 0, \ldots, 63 \tag{500}$$

where $u(n)$, $n = -(231+17), \ldots, 63$, is the excitation buffer. Note that in the search stage, the samples $u(n)$, $n = 0, \ldots, 63$, are unknown and they are needed for pitch delays less than 64. To simplify the search, the LP residual signal, $r(n)$, is copied to $u(n)$ for $n = 0, \ldots, 63$, in order to make the relation in equation (500) valid for all delays. If the optimum integer pitch lag is in the range [34, 91], the fractions around that integer value are tested. The fractional pitch search is performed by interpolating the normalized correlation of equation (499) and searching for its maximum. The interpolation is performed using an FIR filter for interpolating the term in equation (499) using a Hamming windowed sinc function truncated at ± 17. The filter has its cut off frequency (–3 dB) at 5050 Hz and –6 dB at 5760 Hz in the down-sampled domain, which means that the interpolation filter exhibits low-pass frequency response. Note that the fraction is not searched if the selected best integer pitch coincides with the lower end of the searched interval.

Once the fraction is determined, the initial adaptive codevector, $v'(n)$, is computed by interpolating the past excitation signal $u_k(n)$ at the given phase (fraction). In the following text, the fractional pitch lags (not the fractions) in all subframes will be denoted as $d_{fr}^{[i]}$, where the index $i = 0,1,2,3$ denotes the subframe.

In order to enhance the coding performance, a low-pass filter can be applied to the adaptive codevector. This is important since the periodicity doesn't necessarily extend over the whole spectrum. The low pass filter is of the form $b_{LPF}(z) = a + b \cdot z^{-1} + a \cdot z^{-2}$. Thus, the adaptive codevector is given by

$$v(n) = \sum_{i=-1}^{1} b_{LPF}(i+1)v'(n+i), \qquad n = 0, \ldots, 63 \tag{501}$$

where $b_{LPF} = \{0.18, 0.64, 0.18\}$ for $sr_{celp} = 12800$ for rates at and above 32kbps and $b_{LPF} = \{0.21, 0.48, 0.21\}$ otherwise.

An adaptive selection is possible by sending 1 bit per sub-frame. There are then two possibilities to generate the excitation, the adaptive codebook $v(n)$, $v(n) = v'(n)$ in the first path, or its low pass-filtered version as described above in the second path. The path which results in minimum energy of the target signal $x(n)$ is selected for the filtered adaptive codebook vector.

Alternatively, the first or the second path can be used without any adaptive selection. Table 37 summarizes the strategy for the different combinations.

Table 37: Adaptive codebook filtering configuration

Rates (kbps)	IC/UC	VC	GC
7.2	n.a.	Non-filtered	LP filtered
8.0	n.a.	Non-filtered	LP filtered
9.6	n.a.	Non-filtered	LP filtered
13.2	n.a.	Adaptive selection	Adaptive selection
16.4	LP-filtered	Adaptive selection	LP filtered
24.4	LP-filtered	Adaptive selection	LP filtered
32	n.a.	n.a.	Adaptive selection
64	n.a.	n.a.	Adaptive selection

5.2.3.1.4.2   Computation of adaptive codevector gain

The adaptive codevector gain (pitch gain) is then found by

$$g_p = \frac{\sum_{n=0}^{63} x(n)y(n)}{\sum_{n=0}^{63} y(n)y(n)}, \qquad \text{constrained by } 0 \le g_p \le 1.2 \tag{502}$$

where $y(n) = v(n)*h(n)$ is the filtered adaptive codevector (zero-state response of $W(z)H(z)$ to $v(n)$).

To avoid instability in case of channel errors, g_p is limited by 0.95, if the pitch gains of the previous subframes have been close to 1 and the LP filters of the previous subframes have been close to being unstable (highly resonant).

The instability elimination method tests two conditions: resonance condition using the LP spectral parameters (minimum distance between adjacent LSFs), and gain condition by testing for high valued pitch gains in the previous frames. The method works as follows. First, a minimum distance between adjacent LSFs is computed as…

At 9.6, 16.4 and 24.4 kbps, the gain is further constrained. It is done for helping the recovery after the loss of a previous frame.

$$g_p = \min(g_p, 0.8 \sqrt{\frac{\sum_{n=0}^{63} x(n)y(n)}{\sum_{n=0}^{63} y(n)y(n)}}) \tag{503}$$

5.2.3.1.5   Algebraic codebook

5.2.3.1.5.1   Adaptive pre-filter

An important feature of this codebook is that it is a dynamic codebook, whereby the algebraic codevectors are filtered through an adaptive pre-filter $F(z)$. The transfer function of the adaptive pre-filter varies in time in relation to parameters representative of spectral characteristics of the signal. The pre-filter is used to shape the frequency characteristics of the excitation signal to damp frequencies perceptually annoying to the human ear. Here, a pre-filter relevant to WB signals is used which consists of two parts: a periodicity enhancement part $1/(1 - 0.85z^{-T})$ and a tilt part $(1 - \beta_1 z^{-1})$. That is,

$$F^{(0)}(z) = \frac{1 - \beta_1 z^{-1}}{1 - 0.85z^{-T}} \tag{504}$$

The periodicity enhancement part of the filter colours the spectrum by damping inter-harmonic frequencies, which are annoying to the human ear in case of voiced signals. T is the integer part of the closed-loop pitch lag in a given

subframe (representing the fine spectral structure of the speech signal) rounded to the ceiling, i.e., $\left| d_{fr}^{[i]} \right|$, where i denotes the subframe.

The factor β_1 of the tilt part of the pre-filter is related to the voicing of the previous subframe. At 16.4 and 24.4 kbps it is bounded by [0.28, 0.56] and it computed as

$$\beta_1 = 0.28 + \frac{0.28 E_v^{[-1]}}{E_v^{[-1]} + E_c^{[-1]}} \tag{505}$$

Otherwise it is bounded by [0.0, 0.5] and is given by

$$\beta_1 = \frac{0.5 E_v^{[-1]}}{E_v^{[-1]} + E_c^{[-1]}} \tag{506}$$

where $E_v^{[-1]}$ and $E_c^{[-1]}$ are the energies of the scaled pitch codevector and the scaled algebraic codevector of the previous subframe, respectively. The role of the tilt part is to reduce the excitation energy at low frequencies in case of voiced frames.

Depending on bitrates, coding mode and the estimated level of background noise, the adaptive pre-filter also includes a filter based on the spectral envelope, which colours the spectrum by damping frequencies between the formant regions. The final form of the adaptive pre filter $F(z)$ is given by

$$F(z) = F^{(0)}(z) \frac{\hat{A}(z/\eta_1)}{\hat{A}(z/\eta_2)} \tag{507}$$

where $\eta_1 = 0.75$ and $\eta_2 = 0.9$ if $sr_{celp} = 12800$ Hz and $\eta_1 = 0.8$ and $\eta_2 = 0.92$ if $sr_{celp} = 16000$ Hz.

The codebook search is performed in the algebraic domain by combining the pre-filter, $F(z)$, with the weighted synthesis filter prior to the codebook search. Thus, the impulse response $h(n)$ of the weighted synthesis filter must be modified to include the pre-filter $F(z)$. That is, $h(n) \leftarrow h(n) * f(n)$, where $f(n)$ is the impulse response of the pre-filter.

5.2.3.1.5.2 Overview of Algebraic codebooks used in EVS

Depending on the bitrate and rendered bandwidth, algebraic codebooks of different sizes are used in the EVS codec. The following tables summarize the codebooks used in each subframe at different bitrates of the EVS codec

Table 38: NB Algebraic codebook configurations (bits/subframe)

Rate (kbps)	IC	UC	VC	GC
7.2	n.a.	n.a.	12/12/12/20	12/12/12/20
8.0	n.a.	n.a.	12/20/12/20	12/20/12/20
9.6	30/32/32/32	30/32/32/32	28/28/28/28	24/26/24/26
13.2	n.a.	n.a.	36/43/36/43	36/36/36/43
16.4	56/58/56/58	56/58/56/58	56/56/56/58	55/56/55/56
24.4	96/98/96/98	96/98/96/98	96/96/96/98	94/96/96/96

Table 39: WB Algebraic codebook configurations (bits/subframe)

Rate (kbps)	IC	UC	VC	GC	VC-FEC	GC-FEC	GSC
7.2	n.a.	n.a.	12/12/12/20	12/12/12/20	n.a.	n.a.	n.a.
8.0	n.a.	n.a.	12/20/12/20	12/20/12/20	n.a.	n.a.	n.a.
9.6	28/28/28/28	28/28/28/28	26/26/26/28	20/26/24/24	n.a.	n.a.	n.a.
13.2	n.a.	n.a.	28/36/36/36 (TD BWE) 36/36/36/43 (FD BWE)	28/36/28/36 (TD BWE) 36/36/36/36 (FD BWE)	n.a.	n.a.	n.a.
16.4	43/43/43/43/43	43/43/43/43/43	40/43/43/43/43	40/43/40/43/43	n.a.	n.a.	n.a.
24.4	75/75/75/75/75	75/75/75/75/75	73/75/73/75/75	73/73/73/75/73	73/73/73/73/75	70/75/73/73/73	n.a.
32	12/12/12/12/12	n.a.	n.a.	36/36/36/36/36	n.a.	n.a.	n.a.
64	12/12/12/12/12	n.a.	n.a.	36/36/36/36/36	n.a.	n.a.	n.a.

Table 40: SWB Algebraic codebook configurations (bits/subframe)

Rate (kbps)	IC	UC	VC	GC	VC-FEC	GC-FEC	GSC
9.6	24/26/24/26	24/26/24/26	20/26/24/24	20/20/20/20	n.a.	n.a.	n.a.
13.2	n.a.	n.a.	28/36/28/36	28/28/28/36	n.a.	n.a.	n.a.
16.4	36/36/36/36/36	36/36/36/36/36	34/36/36/36/36	34/36/34/36/36	n.a.	n.a.	n.a.
24.4	62/65/62/65/62	62/65/62/65/62	62/62/62/65/62	62/62/62/62/62	62/62/62/62/62	61/61/62/61/62	n.a.
32	12/12/12/12/12	n.a.	n.a.	36/28/28/36/36	n.a.	n.a.	n.a.
64	12/12/12/12/12	n.a.	n.a.	36/36/36/36/36	n.a.	n.a.	n.a.

Table 41: FB Algebraic codebook configurations (bits/subframe)

Rate (kbps)	IC	UC	VC	GC	VC-FEC	GC-FEC
16.4	36/36/36/36/36	36/36/36/36/36	34/36/36/34/36	34/34/36/34/36	n.a.	n.a.
24.4	62/62/65/62/65	62/62/65/62/65	62/62/62/62/62	62/62/62/62/62	61/62/61/62/62	61/61/61/61/61
32	12/12/12/12/12	n.a.	n.a.	36/28/28/36/36	n.a.	n.a.
64	12/12/12/12/12	n.a.	n.a.	36/36/36/36/36	n.a.	n.a.

VC-FEC and GC-FEC are specific configurations for which 4 bits are reserved to transmit LPC-based information exploited by the decoder in case of error of the previous frame.

5.2.3.1.5.3   Codebook structure and pulse indexing of the 7-bit codebook

In the 7-bit codebook, the algebraic vector contains only 1 non-zero pulse at one of 64 positions. The pulse position is encoded with 6 bits and the sign of the pulse is encoded with 1 bit. This gives a total of 7 bits for the algebraic code. The sign index here is set to 1 for positive signs and 0 for negative signs.

5.2.3.1.5.4   Codebook structure and pulse indexing of the 12-bit codebook

In the 12-bit codebook, the algebraic vector contains only 2 non-zero pulses. The 64 positions in a subframe are divided into 2 tracks, where each track contains one pulse, as shown in table 42.

Table 42: Potential positions of individual pulses in the 12-bit algebraic codebook

Track	Pulse	Positions
1	0	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62
2	1	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63

Each pulse position in one track is encoded with 5 bits and the sign of the pulse in the track is encoded with 1 bit. This gives a total of 12 bits for the algebraic code. The sign index here is set to 0 for positive signs and 1 for negative signs.

The index of the signed pulse is given by

$$I = p + s \cdot 2^M \tag{508}$$

where p is the position index, s is the sign index, and $M = 5$ is the number of bits per track. For example, a pulse at position 31 has a position index of $31/2 = 15$ and it belongs to the track with index 1 (second track).

5.2.3.1.5.5 Codebook structure and pulse indexing of the 20-bit and larger codebooks

In the 20-bit or larger codebooks, the codevector contains 4 non-zero pulses. All pulses can have the amplitudes +1 or –1. The 64 positions in a subframe are divided into 4 tracks, where each track contains one pulse, as shown in table 43.

Table 43: Potential positions of individual pulses in the 20-bit algebraic codebook

Track	Pulse	Positions
1	0	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	1	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	2	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	3	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

Each pulse position in one track is encoded with 4 bits and the sign of the pulse in the track is encoded with 1 bit. This gives a total of 20 bits for the algebraic code.

5.2.3.1.5.6 Pulse indexing of the algebraic codebook

The objective is to enumerate all possible constellations of pulses in a vector c which corresponds to one track of length L within a sub-frame. That is, vector c has signed integer values such that its norm-1 is $|x|_1 = p$, whereby we say that c contains p pulses.

We can then partition the vector c into two parts, $c = [c_1, c_2]$ such that the partitions are of length L_1 and $L_2 = L - L_1$ and contain p_1 and $p_2 = p - p_1$ pulses respectively. The number of different constellations for the original vector c can then be determined by the recursive formulae:

$$\begin{cases} f(p,L) = \sum_{p_1=0}^{p} f(p_1, L_1) f(p - p_1, L_2), & p > 0, L > 1 \\ f(p,1) = 2, & p > 1 \\ f(0,L) = 1 & L > 0. \end{cases} \tag{509}$$

For computational efficiency, the values of this function can be pre-calculated and placed in a table.

Above equation gives the number of possible states for given p and L. We can then enumerate a specific state, where c_1 and c_2 have π_1 and $\pi_2 = p - \pi_1$ pulses respectively. The number of states that have less pulses than π_1 in partition c_1 is

$$s(c_1, c_2) = \sum_{p_1=0}^{\pi_1 - 1} f(p_1, L_1) f(p - p_1, L - L_1). \tag{510}$$

We can then define that overall state has $s(c) \geq s(c_1, c_2)$, whereby the overall state can be encoded with the recursion

$$s(c) = s(c_1, c_2) + s(c_1) + f(p_1, L_1) s(c_2), \tag{511}$$

where the boundary conditions are

$$s(c) = \begin{cases} 0, & L=1, c(1) \geq 0 \\ 1, & L=1, c(1) < 0. \end{cases} \tag{512}$$

The state can be decoded by the algorithm

1. Set $p_1 := 1$ and choose partitioning length $L_1 \geq 1$ and $L_2 \geq 1$.

2. Calculate $s(c_1, c_2)$ with p_1.

3. If $s(c) < s(c_1, c_2)$ then $\pi_1 = p_1 - 1$. Otherwise, set $p_1 := p_1 + 1$ and go to 2.

The states of the partitions $s(c_2)$ and $s(c_1)$ can then be calculated from the integer and reminder parts of the fraction $\dfrac{s(c) - s(c_1, c_2)}{f(p_1, L_1)}$. We can then recursively determine the state of each position in the vector c until a partition has $L_k = 1$, whereby

$$c_k(1) = \begin{cases} +\pi_k, & \text{for } s(c_k) = 0 \\ -\pi_k, & \text{for } s(c_k) = 1. \end{cases} \tag{513}$$

Observe that both the number of states $f(p, L)$ as well as the state $s(c)$ are integer numbers which can become larger than 32 bits. We must therefore employ arithmetic operations which support long integers throughout the algorithm.

5.2.3.1.5.7 Pulse indexing of the 43-bit codebook

The joint indexing encoding procedure of three pulses on two tracks is described as follows:

For 3 pulses on a track, the occurrence probability of 3 different pulse positions on a track is the highest, and the occurrence probability of 2 different pulse positions on a track is the second highest, and even the pulses have a higher occurrence probability on the left position of the track than on the right position of the track because the algebraic codebooks need to compensate for the boundary leap of adaptive codebooks between two neighbour sub-frame. So the case of the first pulse with lower position order will be encoded with a smaller index value and the case of more different pulse positions with higher occurrence probability will also be encoded with a smaller index value. The rule is same in case of more than 3 pulses on a track. This rule can be used to save bit in the multi-track joint indexing encoding.

1. Firstly, the pulse information for each track is indexed as follows: (here we suppose that $Q(Q=3)$ pulses are assigned for each track, and the total quantity of positions on the track is M)

   1) Analyse the statistics about the positions of the Q pulses to be encoded on a track and obtain pulse distribution on the track, it includes: quantity (namely $pos\_num = N$) of pulse positions with pulses in it, the pulse position distribution which includes pulse position vector: $P(N) = \{p(0), p(1), ..., p(N-1)\}$, N is the quantity of pulse positions , $p(i)$ is the ith pulse positions with pulse in it on the track, and quantity of pulses in each pulse position with pulse in it which includes pulse number vector $SU(N) = \{su(0), su(1), ..., su(N-1)\}$, $su(0) + su(1) + \cdots + su(N-1) = Q$, where Q is the number of pulses per track, $su(i)$ is the number of pulses in position $p(i)$, and pulse sign vector $S(N) = \{s(0), s(1), ..., s(N-1)\}$, $s(i)$ is the ith sign in position $p(i)$. If there are pulses having the same positions (pulses with the same positions have the same signs), they are merged into one pulse and the number of pulses for each pulse position as well as the pulse sign is saved. Pulse position are sorted in ascend order, the pulse sign is also adjusted based on the order of pulse position.

   2) Compute the offset index $I_1(N)$ according to the quantity of pulse positions, the offset index is saved in a table and used in both encoder and decoder sides. Each offset index in the table indicate a unique number of pulse positions in the track, in case $Q = N$, the offset index only indicate a pulse distribution of pulse positions on the track $P(N)$, in case $Q > N$, the offset index indicate many $SU(N)$ which have a same pulse distribution of pulse positions on the track $P(N)$.

   3) Compute the pulse- position index $I_2(N)$ according to the pulse distribution of pulse positions on the track $(0 \leq I_2(N) < C_M^N)$. The $I_2(N)$ only indicate a pulse distribution of pulse positions on the track $P(N)$ among all the pulse distribution of $I_1(N)$. Permuting serial numbers of the positions $P(N) = \{p(0), p(1), ..., p(N-1)\}$ and all possible values of $P(N)$ are ordered from a smaller value to a greater value

$p(0) < p(1) < \cdots < p(N-1)$, N refers to the quantity of positions with pulses in it, M is the total quantity of positions on the track. Compute $I_2(N)$ by using the permutation method as follows:

$$I_2(N) = C_M^N - C_{M-p(0)}^N + \sum_{n=1}^{N-1}[C_{M-p(n-1)-1}^{N-n} - C_{M-p(n)}^{N-n}], \ 0 \le I_2(N) < C_M^N, \ 0 \le p(0) < p(1) < \cdots < p(N-1) < 15 \ (514)$$

wherein $p(n)$ represents a position serial number of an nth position that has pulses on it, $n \in [0, N-1]$, $p(0) \in [0, M-N]$, $p(n) \in [p(n-1)+1, M-N+n]$, $p(0) < p(1) < \cdots < p(N-1)$. For 43bit mode, 3 pulses on a track, $M = 16$, $0 < N \le 3$.

$$I_2(N) = C_{16}^3 - C_{16-p(0)}^3 + C_{15-p(0)}^2 - C_{16-p(1)}^2 + C_{15-p(1)}^1 - C_{16-p(2)}^1 \tag{515}$$

Compute the pulse-number index $I_3(N)$ according to the quantity of pulses in each pulse position as follows: $I_3(N)$ is determined according to $SU(N)$ which represents the quantity of pulses in each position with pulses. In order to determine correspondence between $SU(N)$ and $I_3(N)$ through algebraic calculation, a calculation method of the third index $I_3(N)$ is provided below:

For a track, situations that a track with N pulse positions and Q pulses are mapped to situations that a N positions track have $Q-N$ pulses, where Q represents the total number of pulses that are required to be encoded and on the track. For example, in the condition of 6-pulse 4-position ($Q = 6$, $N = 4$) situations, $SU(N)$ is {1, 2, 1, 2}, 1 is subtracted from the number of pulses in each position (because each position has at least one pulse) to obtain {0, 1, 0, 1}, that is, information of $SU(N)$ is mapped to a 2-pulse 4-position ($Q = 2$, $M = 4$) encoding situation. Figure 26 gives an example of the mapping for $I_3(N)$.
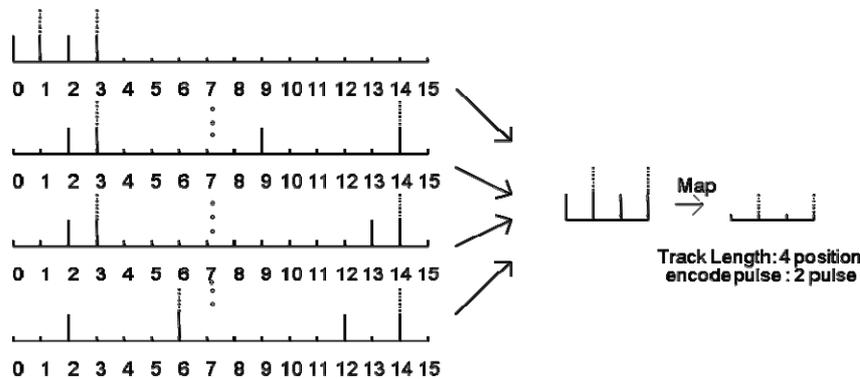
Figure 26: Example of mapping for $I_3(N)$

According to set order, all possible distribution situations of $Q-N$ pulses on N positions are arrayed, and an arrayed serial number is used as the index $I_3(N)$ indicating the number of pulses on a position that has pulse. A calculation formula reflecting the foregoing calculation method is:

$$I3(N) = C_{PPT}^{\Delta N} - C_{PPT-q(0)}^{\Delta N} + \sum_{h=1}^{\Delta N-1}[C_{PPT-h-q(h-1)}^{\Delta N-h} - C_{PPT-h-q(h)}^{\Delta N-h}] \tag{516}$$

wherein $\Delta N = Q - N$, $PPT = Q - 1$, $q(h)$ represents a position serial number of an $(h+1)^{th}$ pulse, $h \in [0, \Delta N - 1]$, $q(h) \in [0, N-1]$, $q(0) \le q(1) \le ... \le q(\Delta N - 1)$, and $\sum$ indicates summation.

Compute the pulse-sign index $I_4(N)$ based on the N pulse sign information.

The pulse sign represented by $s(i)$ may be a positive value or a negative value. A simple coding mode is generally applied, $s(i) = 0$ represents a positive pulse and $s(i) = 1$ represents a negative pulse.

Generate the global index I. Combine the indices $I_1(N)$, $I_2(N)$, $I_3(N)$ and $I_4(N)$ to get the global index I as follows :

$$I = I_1(N) + I_{23} \times 2^N + I_4(N), 1 \le N \le M, I \in [0, W-1] \tag{517}$$

$$I_{23} = I_3(N) \times C_M^N + I_2(N) \tag{518}$$

Here W is the upper range of I which is also the number of total permutations of Q pulses.

2. Combine the index of the two 3-pulse tracks together which is encoded as in step 1, suppose the indexes of the two tracks are Ind_1 and Ind_2 respectively, $Ind_1 \in [0, W_1 - 1]$ and $Ind_2 \in [0, W_2 - 1]$, then the *Joint_index* is as below:

$$Joint\_index = Ind_1 * W_2 + Ind_2 \tag{519}$$

3. Encode the joint index *Joint_index*. (Suppose encode with 25 bits). In order to reduce the number of bits used for pulse indexing, a threshold *THR* is set at 3611648 for 3-pulses, according to the the pulse number, combination of the occurrence probability and the number of bits that may be saved. If the joint index *Joint_index* is smaller than *THR*, 24 bits will be used to encode the joint index *Joint_index*. If the joint index *Joint_index* is bigger than or equal to *THR*, *THR* will be added into the joint index *Joint_index* and 25 bits will be used to encode the joint index *Joint_index*. This procedure is described as below:

```
If ( Joint_index < THR )
{
     Joint_index is encoded with 24 bits.
}
Else
{
     Joint_index = Joint_index + THR

     Joint_index is encoded with 25 bits.
}
```

For two pulses on the other track, the index for each track is encoded just as pulse indexing of the 20-bit codebook, but there is no joint indexing procedure, then the index for each track is transmitted one by one.

5.2.3.1.5.8   Multi-track joint coding of pulse indexing

The codebook for more than three pulses on a track have idle space in difference ratio, joint encoding for more than two tracks may enable idle codebook spaces in single-track encoding to be combined, and once combined idle spaces are sufficient, one actual encoding bit may be reduced. If several encoding indexes are directly combined, the final encoding length may be very large, or even may exceed the bit width (such as 64 bits) generally used for operating, so a general solution is to split each encoding index into two part and only all the high part is combined together in order to avoid directly combining.

The method is described as follows: the value range of the original index Ind_t is divided into several intervals by a factor SLF_t, correspondingly the original index Ind_t is split into two indexes Ind_{t0} and Ind_{t1} by the factor SLF_t, the length of each interval is not greater than SLF_t, SLF_t is a positive integer, Ind_{t0} denotes a serial number of an interval to which Ind_t belongs, and Ind_{t1} denotes a serial number of Ind_t in the interval to which Ind_t belongs (apparently, $Ind_{t1} \le SLF_t$), and: $Ind_t \le Ind_{t0} \times SLF_t + Ind_{t1}$;

The most economical case of splitting is performed as following:

$Ind_{t0} = \text{Int}(Ind_t / SLF_t)$, where $\text{Int}()$ denotes rounding down to an integer, and

$Ind_{t1} = Ind_t \% SLF_t$, where $\%$ denotes taking a remainder.

If a combined index needs to achieve better effect of saving encoding bits, it is needed to select a split index that retains the space characteristics of Ind_t as much as possible, and therefore, for the track t providing a split index to participate in combination,

if $SLF_t = 2^{K_t}$, it is appropriate to select Ind_{t0} to participate in combination, and

if $SLF_t = \text{Int}(Ind_{t\max} / 2^{K_t})$, it is appropriate to select Ind_{t1} to participate in combination.
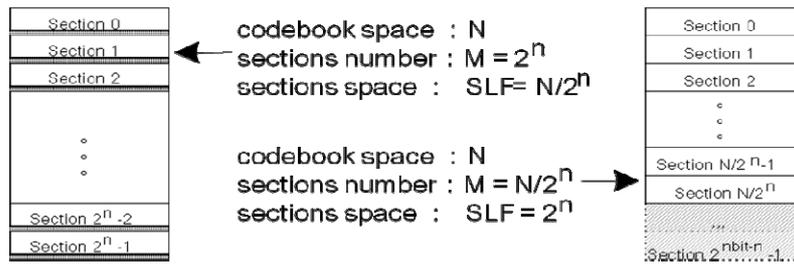
Figure 27: The split factor selection and the corresponding codebook space section

Each track may adopt different SLF_t, according to the pulse number on it

Table 44: the parameters for multi-track joint coding

pulse	bits	Codebook space		Hi Bit			effective ratio	re-back bits		
		Dec	Hex	value	bits	range		8bit	16bit	24bit
1	5	32								
2	9	512	200	1	1	2	1.00	0		
3	13	5472	1560	A8	8	172	0.6875	3		
4	16	44032	AC00	AC	9	345	0.67578125	1	9	
5	19	285088	459A0	8B	8	140	0.546875		5	
6	21	1549824	17A600	BD	8	190	0.7421875		3	
7	23	7288544	6F36E0	DE	8	223	0.875		1	9
8	25	30316544	1CE9800	1CE	9	463	0.904297			8
9	27	113461024	6C34720	6C3	11	1732	0.845704			8

Multi-track joint coding processing is described as following:

Calculate an encoding index Ind_t of each track, (subscript t denotes the tth track), split Ind_t into two split indexes hi_t and $track_t\_low$ according to a set factor SLF_t combine a split index hi_t of each track to generate a combined index hi_{SLPt}. The combined index hi_{SLPt} is split into recombined indexes h_t according to the re-back bits length, and each recombined index h_t and an un-combined split index $track_t\_low$ of a corresponding track are respectively combined, then obtain the final recombined index $final\_index_t$ with fixed length 8,16 or 24 bits.

For 4 track in a sub-frame, the algebraic codebook 94bit(8777)~108bit(9999) use 24 bits mode joint en/decoding，the algebraic codebook 62bit(4444)~92bit(7777) use 16 bits mode joint en/decoding，the algebraic codebook 40bit(3222)~61bit(4443) use 8 bits mode joint en/decoding.

All the encoding steps are described as following:

1) Get the parameter from table 44 according to the pulse number of each track, include the index $bits_t$, $Hi\_Bit\_bits_t$, $Hi\_Bit\_range_t$, $re-back\_bits_t$. And get the 8/16/24 mode also according to the pulse number of all track.

2) The index Ind_t of $track_t$ is split into hi_t and $track_t\_low$, the SLF_t is $2^{(bits_t - Hi\_Bit\_bits_t)}$, and length of hi_t is $Hi\_Bit\_bits_t$, length of $track_t\_low$ is $(bits_t - Hi\_Bit\_bits_t)$,

3) Combine the hi_0 and hi_1 into hi_{SLP0} as following:

$$hi_{SLP0} = hi_0 \times Hi\_Bit\_range_1 + hi_1 \tag{520}$$

4) Split the low part of hi_{SLP0} and get the h_0, and the length of h_0 is $re-back\_bits_0$, which get from table 44 in step 1, the h_0 and $track_0\_low$ are combine into a $final\_index_0$ with the length of 8,16 or 24 bits.

5) The high part hi_{SLP0H} of hi_{SLP0} continue combining with the next hi_2 as following:

$$hi_{SLP1} = hi_{SLP0H} \times Hi\_Bit\_range_2 + hi_2 \tag{521}$$

6) Split the low part of hi_{SLP1} and get the h_1, and the length of h_1 is $re-back\_bits_1$, which get from table 44 in step 1, the h_1 and $track_1\_low$ are combine into a $final\_index_1$ with the length of 8,16 or 24 bits.

7) The high part hi_{SLP1H} of hi_{SLP1} continue combining with the next hi_3 as following:

$$hi_{SLP2} = hi_{SLP1H} \times Hi\_Bit\_range_3 + hi_3 \qquad (522)$$

8) Split the low part of hi_{SLP2} and get the h_2, and the length of h_2 is $re-back\_bits_2$, which get from table 44 in step 1, the h_2 and $track_2\_low$ are combine into a $final\_index_2$ with the length of 8,16 or 24 bits.

9) The high part hi_{SLP2H} of hi_{SLP1} is split into two parts. The low part of hi_{SLP2H} is used as the h_3, and the length of h_3 is $re-back\_bits_3$ which is obtained from table 44 in step 1, the h_3 and $track_3\_low$ are combined into a $final\_index_3$ with the length of 8,16 or 24 bits.

10) Finally, the high part $track\_hi$ of hi_{SLP2H} together with $final\_index_0$, $final\_index_1$, $final\_index_2$ and $final\_index_3$ are the outputs of multi-track joint coding and stored into the stream in 16 bits unit.
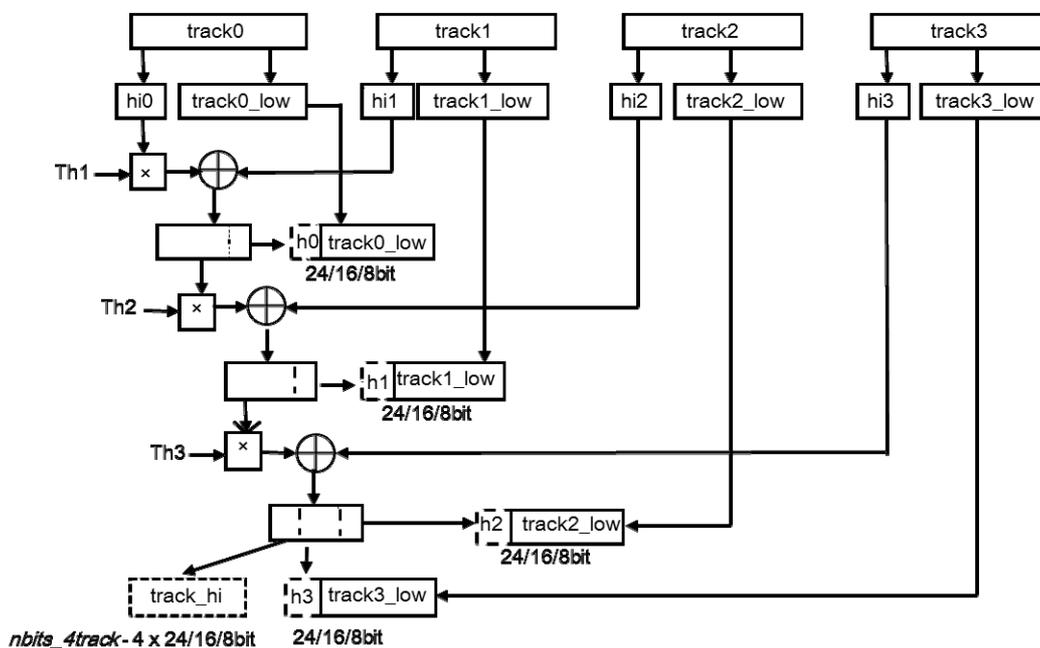
Figure 28: Schematic diagram of 4-track joint coding

5.2.3.1.5.9   The search criterion at lower bitrates

The algebraic codebook is searched by minimizing the error between an updated target signal and a scaled filtered algebraic codevector. The updated target signal is given by

$$x_{11}(n) = x(n) - g_p y(n), \qquad n = 0, \ldots, 63 \qquad (523)$$

where $y(n) = v(n) * h(n)$ is the filtered adaptive codevector and g_p is the unquantized adaptive codebook gain. Thus, the updated target signal is obtained by subtracting the adaptive contribution from the initial target signal, $x(n)$.

Let a matrix $\mathbf{H}$ be defined as a lower triangular Toeplitz convolution matrix with the main diagonal $h(0)$ and lower diagonals $h(1), \ldots, h(63)$, and $\mathbf{d} = \mathbf{H}^T \mathbf{x}_{11}$ (also known as the backward filtered target vector) be the correlation between the updated signal $x_{11}(n)$ and the impulse response $h(n)$. Furthermore, let $\boldsymbol{\Phi} = \mathbf{H}^T \mathbf{H}$ be the matrix of correlations of $h(n)$. Here, $h(n)$ is the impulse response of the combination of the synthesis filter, the weighting filter and the pre-filter $F(z)$ which includes a long-term filter.

The elements of the vector $d(n)$ are computed by

$$d(n) = \sum_{i=n}^{63} x_{11}(i)h(i-n), \qquad n = 0, \ldots, 63 \tag{524}$$

and the elements of the symmetric matrix $\mathbf{\Phi}$ are computed by

$$\varphi(i, j) = \sum_{n=j}^{63} h(n-i)h(n-j), \qquad i = 0, \ldots, 63, \quad j = i, \ldots, 63 \tag{525}$$

Let c_k the k-th algebraic codevector. The algebraic codebook is searched by maximizing the following criterion:

$$Q_k = \frac{\left(\mathbf{x}_{11}^T \mathbf{H} \mathbf{c}_k\right)^2}{\mathbf{c}_k^T \mathbf{H}^T \mathbf{H} \mathbf{c}_k} = \frac{\left(\mathbf{d}^T \mathbf{c}_k\right)^2}{\mathbf{c}_k^T \mathbf{\Phi} \mathbf{c}_k} = \frac{(R_k)^2}{E_k} \tag{526}$$

The vector $d(n)$ and the matrix $\mathbf{\Phi}$ are usually computed prior to the codebook search.

The algebraic structure of the codebooks allows for very fast search procedures since the algebraic codevector, The algebraic structure of the codebooks allows for very fast search procedures since the algebraic codevector, $c_k(n)$, contains only a few non-zero pulses. The correlation in the numerator of equation (526) is given by

$$R = \sum_{i=0}^{N_p - 1} s_i d(m_i) \tag{527}$$

where m_i is the position of the i-th pulse, s_i is its amplitude (sign), and N_p is the number of pulses. The energy in the denominator of equation (526) is given by

$$E = \sum_{i=0}^{N_p - 1} \varphi(m_i, m_i) + 2 \sum_{i=0}^{N_p - 1} \sum_{j=i+1}^{N_p - 1} s_i s_j \varphi(m_i, m_j) \tag{528}$$

For saving the search load along with a better search result in the 12-bit codebook, the pulse amplitudes are predetermined based on a high-pass filtered $d(n)$. The high-pass filter is a three-tap MA (moving-average)-type filter, and its filter coefficients are { -0.35, 1.0, -0.35 }. The sign of a pulse in a position n is set to negative when the high-pass filtered $d(n)$ is negative, otherwise the sign is set to positive. To simplify the search, $d(n)$ and $\Phi(k,h)$ are modified to incorporate the predetermined signs.

5.2.3.1.5.10 The search criterion at higher bitrates

The following search criterion is used for bit rates at and above 16.4 kbps. It allows limiting the increase of complexity for high number of pulses.

Let N be the sub-frame length, and let matrices $\mathbf{H}$ and $\mathbf{C}$, respectively, denote the $N \times N$ lower triangular Toeplitz convolution matrix and the $(N + K - 1) \times N$ full-size convolution matrix, both defined for the filter $h(n)$. Here, $h(n)$ is the length K impulse response of the combination of the synthesis filter, the weighting filter and the pre-filter $F(z)$ which includes a long-term filter. The target residual is $\mathrm{q} = \mathrm{H}^{-1}\mathbf{x}_{11}$ and $\mathbf{\Phi} = \mathbf{C}^T\mathbf{C}$ is the autocorrelation matrix of filter $h(n)$.

The elements of the autocorrelation matrix can be calculated by

$$\mathbf{\Phi}(k, h) = \phi(k - h) = \sum_{n=0}^{K} h(n)h(n - k + h), \qquad n = 0, \ldots, N - 1 \tag{529}$$

and the target residual by

$$q(n) = x_{11}(n) - \sum_{k=1}^{n} h(k)q(n-k), \quad n = 0, \dots, N-1.$$ (530)

The final target is then $\mathbf{d} = \mathbf{Rq}$ which can be calculated by

$$d(n) = \sum_{k=0}^{N-1} \phi\big(|n-k|\big)q(k), \quad n = 0, \dots, N-1.$$ (531)

Let c_k be the kth algebraic codevector. The algebraic codebook is searched by maximizing the following criterion:

$$Q_k = \frac{\left(\mathbf{q}^T \mathbf{\Phi c_k}\right)^2}{\mathbf{c_k}^T \mathbf{\Phi c_k}} = \frac{\left(\mathbf{d}^T \mathbf{c_k}\right)^2}{\mathbf{c_k}^T \mathbf{\Phi c_k}} = \frac{R_k^2}{E_k}$$ (532)

5.2.3.1.6 Combined algebraic codebook

In general the computational complexity of the algebraic codebook increases with the codebook size. In order to keep the complexity reasonable while providing better performance and scalability at high EVS ACELP bit-rates, an efficient combined algebraic codebook structure is employed. The combined algebraic codebook combines usually a frequency-domain coding in a first stage followed by a time-domain ACELP codebook in a second stage.

The frequency-domain coding of the first stage, denoted as a pre-quantizer in figure 29, uses a Discrete Cosine Transform (DCT) as the frequency representation and an Algebraic Vector Quantizer (AVQ) (see subclause 5.2.3.1.6.9) to quantize the frequency-domain coefficients of the DCT. The pre-quantizer parameters are set at the encoder in such a way that the ACELP codebook (second stage of the combined algebraic codebook) is applied to an excitation residual with more regular spectral dynamics than the pitch residual.
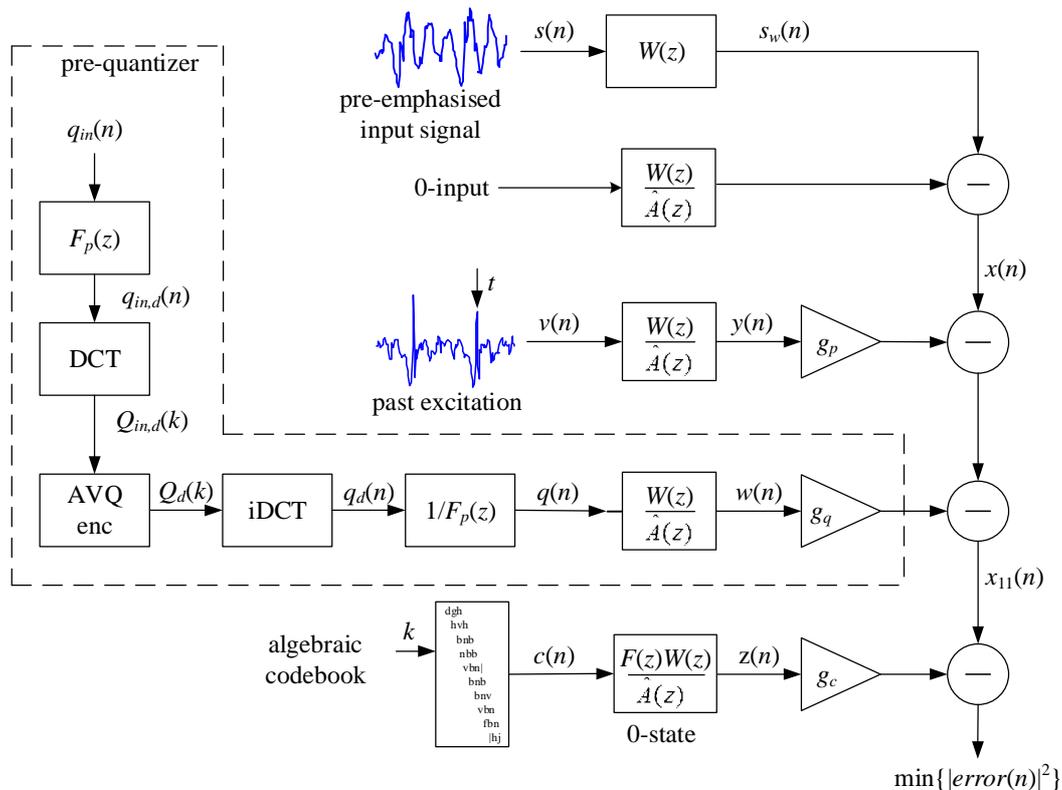
Figure 29: Schematic diagram of the ACELP encoder using a combined algebraic codebook in GC mode at high bit-rates

At the encoder, the first stage, or pre-quantizer, operates as follows. In a given subframe (aligned to the subframe of the ACELP codebook in the second stage) the excitation residual $q_{in}(n)$ after applying the adaptive codebook is computed as

$$q_{in}(n) = r(n) - g_p \cdot v(n) \tag{533}$$

where $r(n)$ is the target vector in residual domain. Further, $v(n)$ is the adaptive codevector and g_p the adaptive codevector gain.

The excitation residual $q_{in}(n)$ after applying the adaptive codebook is de-emphasized with a filter $F_p(z)$. A difference equation for such a de-emphasis filter $F_p(z)$ is given by

$$q_{in,d}(n) = q_{in}(n) + \alpha \cdot q_{in,d}(n-1) \tag{534}$$

where $q_{in,d}(n)$ is the de-emphasized residual and coefficient $\alpha = 0.3$ controls the level of de-emphasis.

Further a DCT is applied to the de-emphasized excitation residual $q_{in,d}(n)$ using a rectangular non-overlapping window. Depending on the bit rate, all blocks or only some blocks of DCT coefficients $Q_{in,d}(k)$ usually corresponding to lower frequencies are quantized using the AVQ encoder. The other (not quantized) DCT coefficients $Q_d(k)$ are set to 0 (not quantized). To obtain the excitation residual for the second (ACELP) stage of the combined algebraic codebook, the quantized DCT coefficients $Q_d(k)$ are inverse transformed, and then a pre-emphasis filter $1/F_p(z)$ is applied to obtain the time-domain contribution from the pre-quantizer $q(n)$. The pre-emphasis filter has the inverse transfer function of the de-emphasis filter $F_p(z)$.

5.2.3.1.6.1 Quantization

The AVQ encoder produces quantized transform-domain DCT coefficients $Q_d(k)$. The indices of the quantized and coded DCT coefficients from the AVQ encoder are transmitted as a pre-quantizer parameters to the decoder.

In every sub-frame, a bit-budget allocated to the AVQ is composed as a sum of a fixed bit-budget and a floating number of bits. Depending on the used AVQ sub-quantizers of the encoder, the AVQ usually does not consume all of the allocated bits, leaving a variable number of bits available in each sub-frame. These bits are floating bits employed in the following sub-frame. The floating number of bits is equal to 0 in the first sub-frame and the floating bits resulting from the AVQ in the last sub-frame in a given frame remain unused when coding WB signals or are re-used in coding of upper band (see subclause 5.2.6.3).

5.2.3.1.6.2 Computation of pre-quantizer gain

Once the pre-quantizer contribution is computed, the pre-quantizer gain is obtained as

$$g_q = \frac{\sum_{k=0}^{N-1} Q_{in,d}(k) Q_d(k)}{\sum_{k=0}^{N-1} Q_d(k) Q_d(k)} \tag{535}$$

where $Q_{in,d}(k)$ are the AVQ input frequency coefficients and $Q_d(k)$ the AVQ output (quantized) frequency coefficients where $k = 0, ..., K-1$ is the transform-domain coefficient index and $K = 64$ being the number of DCT transform coefficients.

5.2.3.1.6.3 Quantization of pre-quantizer gain

The pre-quantizer gain g_q is quantized as follows. First, the gain is normalized by the predicted innovation energy E_{pred} as follows:

$$g_{q,norm} = \frac{g_q}{E_{pred}} \tag{536}$$

where the predicted innovation energy E_{pred} is obtained as described in subclause 5.2.3.1.7.1.

Then the normalized gain $g_{q,norm}$ is quantized by a scalar quantizer in a logarithmic domain and finally de-normalized resulting in a quantized pre-quantizer gain. Specifically 6-bit scalar quantizer is used whereby the quantization levels are uniformly distributed in the log domain. The index of the quantized pre-quantizer gain is transmitted as a pre-quantizer parameter to the decoder.

5.2.3.1.6.4 Refinement of target vector

The pre-quantizer contribution $q(n)$ is used to refine the original target vector for adaptive codebook search $x(n)$ as

$$x_{updt}(n) = x(n) - g_p \cdot w(n) , \qquad (537)$$

and to refine the adaptive codebook gain using equation (502) with $x_{updt}(n)$ used instead of $x(n)$. When the pre-quantizer is used, the computation of the target vector for algebraic codebook search $x_{11}(n)$ is done using

$$x_{11}(n) = x(n) - g_q \cdot w(n) - g_{p,updt} \cdot y(n) \qquad (538)$$

where $w(n)$ is the filtered pre-quantizer contribution, i.e. the zero-state response of the weighted synthesis filter to the pre-quantizer contribution $q(n)$.

Similarly, the target vector in residual domain $r(n)$ is updated for the algebraic codebook search (the second-stage of the combined algebraic codebook) as

$$r_{updt}(n) = r(n) - g_q \cdot q(n) - g_{p,updt} \cdot v(n) . \qquad (539)$$

5.2.3.1.6.5 Combined algebraic codebook in GC mode

In the EVS codec, the combined algebraic codebook structure as from figure 29 is used at bit-rates of 32 kbps and 64 kbps. In both cases the algebraic codebook search uses 36-bit codebooks and the rest of the bit-budget is employed by the AVQ to quantize the pre-quantizer coefficients.

At 32 kbps, the available fixed bit-budget for the AVQ (116, 115, 115, 115, 155 bits for every of five subframes) is sometimes too low to properly encode all input signal frames. Consequently in GC mode at 32kbps, the DCT and iDCT stages of pre-quantizer computation are omitted when the input signal is not classified as a harmonic one. The classification is based on a harmonicity counter $harm\_acelp$ updated every frame in the pre-processing module. If in a given frame the harmonicity counter $harm\_acelp \le 2$ the frame is classified as non-harmonic and the AVQ is applied directly on the time-domain signal $q_{in,d}(n)$ and similarly producing directly the time-domain signal $q_d(n)$ in figure 29.

5.2.3.1.6.6 Combined algebraic codebook in TC mode

The combined algebraic codebook structure is used also in TC mode at 32kbps and 64kbps. In this mode the algebraic codebook from figure 29 is replaced by glottal shape codebook but the structure of the pre-quantizer remains the same as in the GC mode. In TC mode @32kbps, the DCT and iDCT stages of the pre-quantizer are always employed.
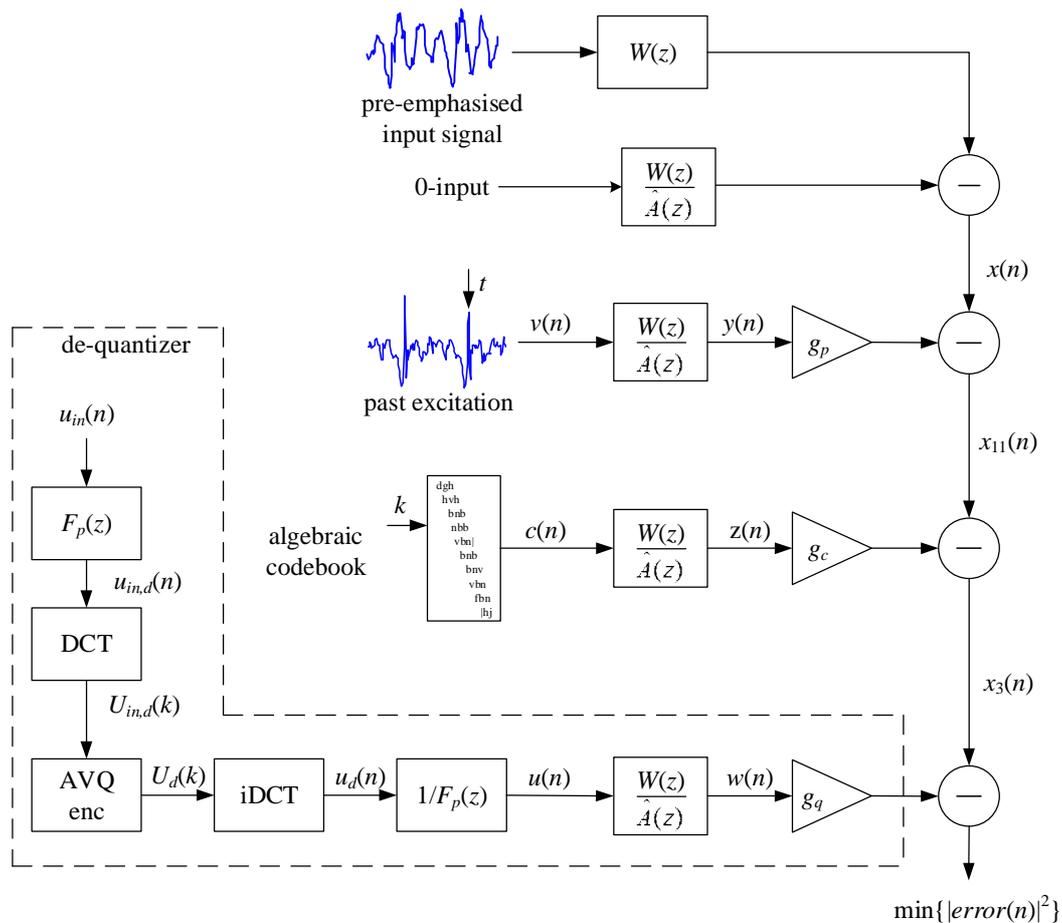
Figure 30: Schematic diagram of the ACELP encoder using a combined algebraic codebook in IC mode at high bit-rates

5.2.3.1.6.7   Combined algebraic codebook in IC mode

Depending on the input signal characteristics, the ACELP encoder using a combined algebraic codebook from figure 29 is further adaptively changed. Specifically in coding of inactive speech segments, the order of the combined algebraic codebook stages is changed. I.e. the modified combined algebraic codebook combines a time-domain ACELP codebook in a first stage followed by a frequency-domain de-quantizer coding in a second stage as shown in figure 30. The first stage algebraic codebook employs very small codebooks, specifically 12 bits per subframe.

At the encoder, the de-quantizer in IC mode operates as follows. In a given subframe, the target signal $x_3(n)$ after subtracting the scaled filtered adaptive excitation and the scaled filtered algebraic excitation is computed as

$$x_3(n) = x(n) - g_p \cdot y(n) - g_c \cdot z(n).$$ (540)

The target signal in speech domain $x_3(n)$ is filtered through the inverse of the weighted synthesis filter with zero states resulting in the target in residual domain $u_{in}(n)$.

Similarly to the combined algebraic codebook in GC mode, the signal $u_{in}(n)$ is first de-emphasized with a filter $F_p(z)$ to enhance the low frequencies. A DCT is applied to the de-emphasized signal $u_{in,d}(n)$ using rectangular non-overlapping window. Usually all blocks of DCT coefficients $U_{in,d}(k)$ are quantized using the AVQ encoder. The quantized DCT coefficients $U_d(k)$ in some bands can be however set to zero.

The quantized DCT coefficients $U_d(k)$ are further inverse transformed using iDCT, and then a pre-emphasis filter $1/F_p(z)$ is applied to obtain the time-domain contribution from the frequency-domain quantizer $u(n)$ where the pre-emphasis filter has the inverse transfer function of the de-emphasis filter $F_p(z)$.

5.2.3.1.6.8 Computation and quantization of de-quantizer gain

Once the de-quantizer contribution is computed, the de-quantizer gain is obtained as

$$g_q = \frac{\sum_{k=0}^{N-1} U_{in,d}(k) U_d(k)}{\sum_{k=0}^{N-1} U_d(k) U_d(k)} \tag{541}$$

where $U_{in,d}(k)$ are the AVQ input transform-domain coefficients and $U_d(k)$ are the AVQ output (quantized) transform-domain coefficients.

The de-quantizer gain g_q is quantized using the normalization by the algebraic codebook gain g_c. Specifically a 6-bit scalar quantizer is used whereby the quantization levels are uniformly distributed in the linear domain. The indice of the quantized de-quantizer gain g_q is transmitted as a de-quantizer parameter to the decoder.

When coding the inactive signal segments the adaptive codebook excitation contribution is limited to avoid a strong periodicity in the synthesis. In practice a limiter is applied in the adaptive codebook search to constrain the adaptive codebook gain by $0 \leq g_p \leq 0.65$.

5.2.3.1.6.9 AVQ quantization with split multi-rate lattice VQ

Prior to the AVQ quantization, the time domain or transform-domain 64 coefficients, here denoted as $S'(k)$, are split into 8 consecutive sub-bands of 8 coefficients each. The sub-bands are quantized with an 8-dimensional multi-rate algebraic vector quantizer. The AVQ codebooks are subsets of the Gosset lattice, referred to as the RE_8 lattice.

5.2.3.1.6.9.1 Multi-rate AVQ with the Gosset Lattice RE_8

5.2.3.1.6.9.1.1 Gosset Lattice RE_8

The Gosset lattice RE_8 is defined as the following union:

$$RE_8 = 2D_8 \cup \{2D_8 + (1,1,1,1,1,1,1,1)\} \tag{542}$$

where D_8 is the 8-dimensional lattice composed of all points with integers components with the constraint that the sum of the 8 components is even. The lattice $2D_8$ is simply the D_8 lattice scaled by 2. This implies that the sum of the components of a lattice point in $2D_8$ is an integer multiple of 4. Therefore, the 8 components of a RE_8 lattice point have the same parity (either all even or all odd) and their sum is a multiple of 4.

All points in the lattice RE_8 lie on concentric spheres of radius $\sqrt{8n_j}$, n_j being the codebook number in sub-band j.

Each lattice point on a given sphere can be generated by permuting the coordinates of reference points called "leaders". There are very few leaders on a sphere compared to the total number of lattice points which lie on the sphere.

5.2.3.1.6.9.1.2 Multi-rate codebooks in Gosset Lattice $RE8$

To form a vector codebook at a given rate, only lattice points inside a sphere in 8 dimensions of a given radius are taken. Codebooks of different bit rates can be constructed by including only spheres up to a given radius. Multi-rate codebooks are formed by taking subsets of lattice points inside spheres of different radii.

5.2.3.1.6.9.1.2.1 Base codebooks

First, base codebooks are designed. A base codebook contains all lattice points from a given set of spheres up to a number n_j. Four base codebooks, noted Q_0, Q_2, Q_3, and Q_4, are used. There are 36 non-null absolute leaders plus the zero leader (the origin): Table 46 gives the list of these leaders and indicates to which codebook a leader belongs.

Q_0, Q_2, Q_3, and Q_4 are constructed with respectively 0, 8, 12, and 16 bits. Hence codebook Q_{n_j} requires $4n_j$ bits to index any point in that codebook.

5.2.3.1.6.9.1.2.2 Voronoi extensions

From a base codebook C_{AVQ} (i.e. a codebook containing all lattice points from a given set of spheres up to a number n_j), an extended codebook can be generated by multiplying the elements of C_{AVQ} by a factor M_j^v, and adding a second-stage codebook called the Voronoi extension. This construction is given by

$$\mathbf{c}_j = M_j^v \cdot \mathbf{z}_j + \mathbf{v}_j \tag{543}$$

where M_j^v is the scaling factor, $\mathbf{z}_j$ is a point in a base codebook C_{AVQ} and $\mathbf{v}_j$ is a point in the Voronoi extension. The extension is computed in such a way that any point $\mathbf{c}_j$ from equation (543) is also a lattice point in RE_8. The scaling factor M_j^v is a power of 2 ($M_j^v = 2^{r_j^v}$), where r_j^v is called the Voronoi extension order.

Such extended codebooks include lattice points that extend further out from the origin than the base codebook. When a given lattice point $\mathbf{c}_j$ is not included in a base codebook C_{AVQ} (Q_0, Q_2, Q_3 or Q_4), the so-called Voronoi extension is applied, using the Q_3 or Q_4 base codebook part.

Giving the available bit-budget in particular layers, the maximum Voronoi extension order is $r_j^v = 2$. Therefore, for Q_3 or Q_4, two extension orders are used: $r_j^v = 1$ or 2 ($M_j^v = 2$ or 4).

When $r_j^v = 0$, there is no Voronoi extension, and only a base codebook is used.

5.2.3.1.6.9.1.2.3 Codebook rates

There are 8 codebooks: the first 4 are base codebooks without Voronoi extension and the last four with Voronoi extension. The codebook number n_j is encoded as a unary code with n_j "1" bits and a terminating "0". Table 45 gives for each of the 8 codebooks, its base codebook, its Voronoi extension order ($r_j^v = 0$ indicates that there is not Voronoi extension), and its unary code.

Table 45: Multi-rate codebooks in RE_8 lattice

Codebook number n_j	Base Codebook	Voronoi extension order r_j^v	Unary code for n_j
0	Q_0	0	0
2	Q_2	0	10
3	Q_3	0	110
4	Q_4	0	1110
5	Q_3	1	11110
6	Q_4	1	111110
7	Q_3	2	1111110
8	Q_4	2	11111110

For the base codebook Q_0, ($n_j = 0$), there is only one point in the codebook and 1 bit is used to transmit the unary code corresponding to n_j.

For the other three base codebooks Q_{n_j} ($n_j = 2, 3,$ or 4) without Voronoi extension:

- n_j bits are used to transmit the unary code corresponding to $n_{j,}$

- $4n_j$ bits are required to index a point in Q_{n_j}

- thus $5n_j$ bits are used in total.

For codebooks with Voronoi extension ($n_j > 4$):

- n_j bits are used to transmit the unary code corresponding to the base codebook number Q_3 (respectively Q_4) if n_j is even (respectively odd) and the Voronoi extension order r_j^v is 1 if $n_j < 7$, or 2 otherwise),

- 12 bits (respectively 16 bits) are required to index the point $\mathbf{z}_j$ in the base codebook Q_3 (respectively Q_4)

- $8r_j^v$ bits are required to index the 8-dimensional point $\mathbf{v}_j$ in the Voronoi extension of order r_j^v

- thus, $5n_j$ bits are used in total.

In the codebook number encoding, a simple bit overflow check is performed: in case when the last AVQ coded sub-band of the spectrum $S'(k)$ is quantized, $n_j > 0$ and only $5n_j - 1$ bits are available for the quantization, the terminating "0" in the codebook number coding is not encoded. At the decoder, the same bit overflow check enables the right decoding of the codebook number in this sub-band.

5.2.3.1.6.9.2   Quantization with RE_8 lattice

In lattice quantization, the operation of finding the nearest neighbour of the input spectrum $S'(k)$ among all codebook points is reduced to a few simple operations, involving rounding the components of spectrum $S'(k)$ and verifying a few constraints. Hence, no exhaustive search is carried out as in stochastic quantization, which uses stored tables. Once the best lattice codebook point is determined, further calculations are also necessary to compute the index that will be sent to the decoder. The larger the components of the input spectrum $S'(k)$, the more bits will be required to encode the index of its nearest neighbour in the lattice codebook. Hence, to remain within a pre-defined bit-budget, a gain-shape approach has to be used, where the input spectrum is first scaled down by the AVQ gain, then each 8-dimensional block of spectrum coefficients is quantized in the lattice and finally scaled up again to produce the quantized spectrum.

5.2.3.1.6.9.2.1   AVQ gain estimation

Prior to the quantization (nearest neighbour search and indexation of the nearest neighbour), the input spectrum has to be scaled down to ensure that the total bit consumption will remain within the available bit-budget.

A first estimation of the total bit-budget *nbits* without scaling (i.e. with an AVQ gain equals to 1) is performed:

$$nbits = \sum_j R_j \tag{544}$$

where R_j is a first estimate of the bit budget to encode the sub-band j given by:

$$R_j = 5\log_2\left(\frac{E_j}{2}\right) \tag{545}$$

with E_i being the energy (with a lower limit set to 2) of each sub-band $\mathbf{S}'(8j)$:

$$E_j = \max\left(2, \sum_{i=0}^{7}[S'(8j+i)]^2\right) \tag{546}$$

This gain estimation is performed in an iterative procedure described below.

Let NB_BITS be the number of bits available for the quantization process and NB_SBANDS the number of 8-dimensional sub-bands to be quantized:

Initialization:

   $fac = 128$,

   $offset = 0$,

   $nbits_{max} = 0.95$ (NB_BITS − NB_SBANDS)

for $i = 1{:}10$

      $offset = offset + fac$

$$nbits = \sum_{j=1}^{\text{NB\_SBANDS}} \max\left(0, R_j - offset\right)$$

      if $nbits \leq nbits_{max}$, then

      $offset = offset - fac$

      $fac = fac / 2$

After the 10th iteration, the AVQ gain is equal to $10\exp\left(0.1 \, offset \cdot \log_{10}(2)\right)$ and is used to obtain the scaled spectrum $S'_{norm}(k)$:

$$S'_{norm}(k) = S'(k) / \left[10\exp\left(0.1 \cdot offset \cdot \log_{10}(2)\right)\right] \tag{547}$$

5.2.3.1.6.9.2.2 Nearest neighbour search

The search of the nearest neighbour in the lattice RE_8 is equivalent to searching for the nearest neighbour in the lattice $2D_8$ and for the nearest neighbour in the lattice $2D_8 + (1,1,1,1,1,1,1,1)$, and finally selecting among those two lattice points the closest to $\mathbf{S}'_{norm}(8j)$) as its quantized version $\hat{\mathbf{S}}'(8j)$.

Based on the definition of RE_8 , the following fast algorithm is used to search the nearest neighbour of an 8-dimensional sub-band $\mathbf{S}'_{norm}(8j)$ among all lattice points in RE_8 :

Search of the nearest neighbour $\mathbf{y}_{1j}$ in $2D_8$ of $\mathbf{S}'_{norm}(8j)$:

   Compute $\mathbf{z}_j = 0.5 \cdot \mathbf{S}'_{norm}(8j)$.

   Round each component of $\mathbf{z}_j$ to the nearest integer to generate $\mathbf{z}'_j$.

   Compute $\mathbf{y}_{1j} = 2\mathbf{z}'_j$.

   Calculate the sum S of the 8 components of $\mathbf{y}_{1j}$.

   if S is not an integer multiple of 4, then modify its Ith component as follows:

$$y_{1j}(I) = \begin{cases} y_{1j}(I) - 2, & \text{if } z_j(I) - y_{1j}(I) < 0, \\ y_{1j}(I) + 2, & \text{otherwise.} \end{cases}$$

      where $I = \arg\left(\max\left(\left|z_j(i) - y_{1j}(i)\right|\right)\right)$

Search of the nearest neighbour $\mathbf{y}_{2j}$ in $2D_8 + (1,1,1,1,1,1,1,1)$ of $\mathbf{S}'_{norm}(8j)$:

   Compute $\mathbf{z}_j = 0.5 \cdot \left(\mathbf{S}'_{norm}(8j) - \mathbf{1.0}\right)$ where $\mathbf{1.0}$ denotes an 8-dimensional vector with all ones.

   Round each component of $\mathbf{z}_j$ to the nearest integer to generate $\mathbf{z}'_j$.

   Compute $\mathbf{y}_{2j} = 2\mathbf{z}'_j$.

   Calculate the sum S of the 8 components of $\mathbf{y}_{2j}$.

   if S is not an integer multiple of 4 then modify its Ith component as follows:

$$y_{2j}(I) = \begin{cases} y_{2j}(I) - 2, & \text{if } z_j(I) - y_{2j}(I) < 0, \\ y_{2j}(I) + 2, & \text{otherwise.} \end{cases}$$

      where $I = \arg\left(\max\left(\left|z_j(i) - y_{2j}(i)\right|\right)\right)$

   Compute $\mathbf{y}_{2j} = \mathbf{y}_{2j} + \mathbf{1.0}$.

Select between $\mathbf{y}_{1j}$ and $\mathbf{y}_{2j}$ as the closest point $\hat{\mathbf{S}}'(8j)$ in RE_8 to $\mathbf{S}'_{norm}(8j)$:

$$\hat{\mathbf{S}}'(8j) = \begin{cases} \mathbf{y}_{1j}, & \text{if } e_{1j} > e_{2j}, \\ \mathbf{y}_{2j}, & \text{otherwise.} \end{cases}$$

where $e_{1j} = \left(\mathbf{S}'_{norm}(8j) - \mathbf{y}_{1j}\right)^2$ and $e_{2j} = \left(\mathbf{S}'_{norm}(8j) - \mathbf{y}_{2j}\right)^2$.

5.2.3.1.6.9.3 Indexation

The quantized scaled sub-band $\hat{\mathbf{S}}'(8j)$ of $\mathbf{S}'_{norm}(8j)$) is a point $\mathbf{c}_j$ in a RE_8 lattice codebook, an index for each $\mathbf{c}_j$ has to be computed and later inserted into the bitstream.

This index is actually composed of three parts:

1) a codebook number n_j;

2) a vector index I_j, which uniquely identifies a lattice vector in a base codebook C_{AVQ};

3) and if $n_j > 4$, an 8-dimensional Voronoi extension index $\mathbf{I}_j^v$ that is used to extend the base codebook when the selected point in the lattice is not in a base codebook C_{AVQ}.

The calculation of an index for a given point $\mathbf{c}_j$ in the RE_8 lattice is performed as follows:

First, it is verified whether $\mathbf{c}_j$ is in a base codebook C_{AVQ} by identifying its sphere and its leader:

– if $\mathbf{c}_j$ is in a base codebook, the index used to encode $\mathbf{c}_j$ is thus the codebook number n_j plus the index I_j of the lattice point $\mathbf{c}_j$ in Q_{n_j}.

Otherwise, the parameters of the Voronoi extension (see equation (543)) have to determined: the scaling factor M_v, the base codebook C_{AVQ} (Q_3 or Q_4), the point $\mathbf{z}_j$ in this base codebook, and the point $\mathbf{v}_j$ in the Voronoi extension.

Then, the index used to encode is composed of the codebook number n_j ($n_j > 4$) plus the index I_j of the lattice point $\mathbf{z}_j$ in the base codebook C_{AVQ} (Q_3 or Q_4), and the index $\mathbf{I}_j^v$ of $\mathbf{v}_j$ in the Voronoi extension.

5.2.3.1.6.9.3.1 Indexing a codebook number

As explained in subclause 5.2.3.1.6.9.1.2.3 – Codebook rates, the codebook index n_j is unary encoded with n_j bits except for $n_j = 0$ that is coded with one bit (see table 45).

5.2.3.1.6.9.3.2 Indexing of codevector in base codebook

The index I_j indicates the rank of codevector $\mathbf{z}_j$ in j-th sub-band, i.e., the permutation to be applied to a specific leader to obtain $\mathbf{z}_j$. The index computation is done in several steps, as follows:

1) The input codevector $\mathbf{z}_j$ is decomposed into a sign vector $\mathbf{s}_0$ and an absolute vector $\mathbf{y}_0$ following a two-path procedure.

2) The sign vector is encoded, the associated index $bits_{sign}(\mathbf{s}_0)$ and the number of non-zero components in $\mathbf{z}_j$ are obtained. More details are given in subsequent subclauses.

3) The absolute vector is encoded using a multi-level permutation-based index encoding method, and the associated index $rank(\mathbf{y}_0)$ is obtained.

4) The absolute vector index $rank(\mathbf{y}_0)$ and the sign index $bits_{sign}(\mathbf{s}_0)$ are added together in order to obtain the input vector rank: $rank(\mathbf{z}_j)$.

$$rank(\mathbf{z}_j) = rank(\mathbf{y}_0) \cdot 2^{sign_{nb}(\mathbf{s}_0)} + bits_{sign}(\mathbf{s}_0) \tag{548}$$

5) Finally, the offset $lead_{offset}(\mathbf{z}_j)$ is added to the rank. The index I_j is obtained by

$$I_j = lead_{offset}(\mathbf{z}_j) + rank(\mathbf{z}_j) \tag{549}$$

The indexing of codevector in base codebook is done in two steps. First the sign vector is encoded.

The number of bits required for encoding the sign vector elements is equal to the number of non-zero elements in the codevector. "1" represents a negative sign and "0" a positive sign. As lattice RE_8 quantization is used, the sum of all the elements in a codevector is an integer multiple of 4. If there is any change of sign in the non-zero element, the sum may not be a multiple of 4 anymore, in that case, the last element sign in the sign vector will be omitted. For example, the sign vector of the input vector $(-1, -1, 1, 1, 1, 1, -1, -1)$ in leader 1 (see table 46) has seven bits and its value is 0x1100001.

In the second step the absolute vector and its position vector is encoded

The encoding method for the absolute vector works as follows. The absolute vector is first decomposed into ML_{max} levels. The highest-level vector L_0 is the original absolute vector. The n value for L_n is initialized to zero. Then:

1) First the intermediate absolute value vector of L_1 is obtained by removing the most frequent element as given in the decomposition order column of table 46 from the original absolute vector of L_0. Sequentially the remaining elements are built into a new absolute vector for L_1; it has a position order related to the level L_0 original absolute vector. All position values q_i of the remainder elements are used to build a position vector $(q_0, q_1, q_2, ..., q_{m_1-1})$ of L_1.

The relationship between the original absolute vector of L_0 and the new absolute vector of L_1 is that: the original absolute vector of L_0 is the upper-level vector of the new absolute vector of L_1, and the new absolute vector of L_1 is the lower-level vector of the original absolute vector of L_0. The relationship between any two neighbour level absolute vector is the same. The detail relationship is described as following:

The position vector	(	1,			4,		6	)			
		↑			↑		↑				
The element position	0	1	2	3	4	5	6	7			
The original absolute vector of L_0 (upper level vector)	(0	2,	0,	0,	4,	0,	6,	0)			
One type of the element "0" is removed	(0	2,	0,	0,	4,	0,	6,	0)			
		↓			↓		↓				
The new absolute vector of L_1 (lower level vector)	(	2,			4,		6	)			
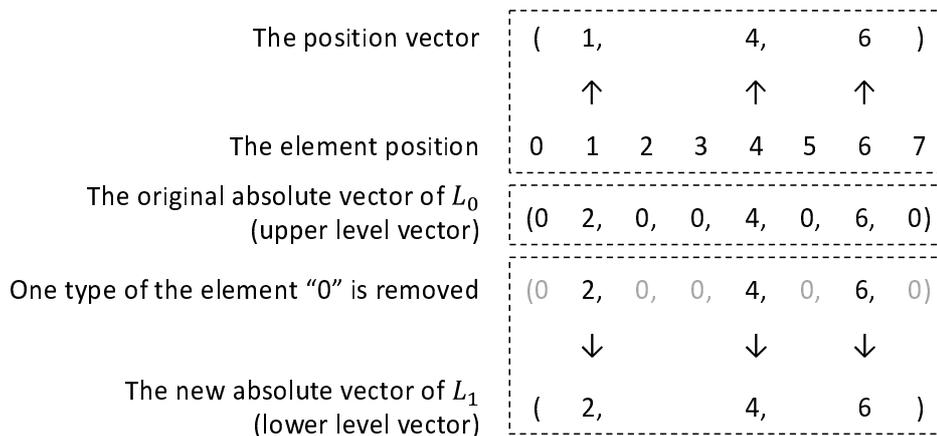
Figure 31: Example processing of first level for $K_a = 20$.

2) Then the position vector $(q_0, q_1, q_2, ..., q_{m_1-1})$ of the new absolute vector of L_1 related to the original absolute vector of L_0 is indexed based on a permutation and combination function, the indexing result being called the middle index $I_{mid,1}$. For the new absolute vector in L_1, the position vector indexing is computed as follows:

$$I_{mid,1} = C_{m_0}^{m_1} - C_{m_0-q_0}^{m_1} + \sum_{i=1}^{i<m_1} \left(C_{m_0-q_{i-1}}^{m_1-i} - C_{m_0-q_i}^{m_1-i} \right) \tag{550}$$

$$I_{final} = I_{final} \cdot C_{m_0}^{m_1} + I_{mid,1} \tag{551}$$

where I_{final} is initialized to zero before the first step at the beginning of the procedure, m_0 is the dimension of the original absolute vector of L_0, m_1 is the dimension of the new absolute vector of L_1.

If there is more than one type of element in the new absolute vector, the new absolute vector, named the upper-level vector, will be encoded using the multi-level permutation-based index encoding method as following step:

3) Increment the n value. At level L_n, $0 < L_n < ML_{max}$, the intermediate absolute value vector is obtained by removing the most frequent element as given in the decomposition order column of table 46 from the upper-level vector. Sequentially the remaining elements are built into a new absolute vector for the current level; it has a position order related to the level L_{n-1} absolute vector. All position values of the remainder elements are used to build a position vector.

4) The position vector of the current lower-level vector related to its upper-level vector is indexed based on a permutation and combination function, the indexing result being called the middle index $I_{mid,n}$. For the absolute vector in the current lower level, the position vector indexing is computed as follows:

$$I_{mid,n} = C_{m_{n-1}}^{m_n} - C_{m_{n-1}-q_0}^{m_n} + \sum_{i=1}^{i<m_n} \left(C_{m_{n-1}-q_{i-1}}^{m_n-i} - C_{m_{n-1}-q_i}^{m_n-i} \right) \tag{552}$$

$$I_{final} = I_{final} \cdot C_{m_{n-1}}^{m_n} + I_{mid,n} \tag{553}$$

The elements $q_0, q_1, q_2...$ are the element values in the L_n level position vector ranged from left to right according to their level, m_{n-1} is the dimension of the upper-level absolute vector, m_n is the dimension of the current-level absolute vector, C_q^m represents the permutation and combination formula $C_q^m = \dfrac{q!}{m!(p-q)!}$, $q,m = \{1,2,3,4,8\}$, and $q > m$. All the values for C_q^m can be stored in a simple table in order to avoid calculation of factorials. The L_{n-1} level final-index, I_{final}, is multiplied by the possible index value number, $C_{m_{n-1}}^{m_n}$, in the current level and is added to the index, $I_{mid,n}$, in the current level, to obtain the final index, I_{final}, of the current level.

5) Repeat steps 3 and 4 until there is only one type of element left in the current absolute vector. The I_{final} for the lowest level is the rank of the absolute vector called $rank(\mathbf{y}_0)$. Table 46 is a sample extracted from the 36 leader table case. The leaders are indexed by K_a. The decomposition order corresponds to the level order. The decomposition order column gives the order in which the element will be removed from the higher level. The last column gives the three class parameters, the first one is the number of sign bits, S_n, the second one is the number of decomposition levels and equals the number of element types in the leader, V_c, from the third one to the last one they represent the absolute vector dimension in each lower level except the highest level, m_1, m_2, m_3 (note that the dimension for the highest level is eight, but is not listed in table 46).

Table 46: List of leaders in base codebooks Q_{n_j} with their decomposition order and set parameter of multi-level permutation-based encoding

K_a	Leader	Decomposition order	S_n, V_c, m_1, m_2, m_3	Q_0	Q_2	Q_3	Q_4
	{0,0,0,0,0,0,0,0}			X			
0	{1,1,1,1,1,1,1,1}	{1}	{7,1}		X	X	
1	{2,2,0,0,0,0,0,0}	{0,2}	{2,2,2}		X	X	
2	{2,2,2,2,0,0,0,0}	{0,2}	{4,2,4}			X	
3	{3,1,1,1,1,1,1,1}	{1,3}	{7,2,1}			X	
4	{4,0,0,0,0,0,0,0}	{0,4}	{1,2,1}		X	X	
5	{2,2,2,2,2,2,0,0}	{2,0}	{6,2,2}				X
6	{3,3,1,1,1,1,1,1}	{1,3}	{7,2,2}				X
7	{4,2,2,0,0,0,0,0}	{0,2,4}	{3,3,3,1}			X	
8	{2,2,2,2,2,2,2,2}	{2}	{8,1}				X
9	{3,3,3,1,1,1,1,1}	{1,3}	{7,2,3}				X
10	{4,2,2,2,2,0,0,0}	{2,0,4}	{5,3,4,1}				X
11	{4,4,0,0,0,0,0,0}	{0,4}	{2,2,2}			X	
12	{5,1,1,1,1,1,1,1}	{1,5}	{7,2,1}				X
13	{3,3,3,3,1,1,1,1}	{1,3}	{7,2,4}				X
14	{4,2,2,2,2,2,2,0}	{2,0,4}	{7,3,2,1}				X
15	{4,4,2,2,0,0,0,0}	{0,2,4}	{4,3,4,2}				X
16	{5,3,1,1,1,1,1,1}	{1,3,5}	{7,3,2,1}				X
17	{ 6,2,0,0,0,0,0,0}	{0,2,6}	{2,3,2,1}			X	
18	{ 4,4,4,0,0,0,0,0}	{0,4}	{3,2,3}				X
19	{ 6,2,2,2,0,0,0,0}	{0,2,6}	{4,3,4,1}				X
20	{ 6,4,2,0,0,0,0,0}	{0,2,4,6}	{3,4,3,2,1}				X
21	{ 7,1,1,1,1,1,1,1}	{1,7}	{7,2,1}				X
22	{ 8,0,0,0,0,0,0,0}	{0,8}	{1,2,1}				X
23	{6,6,0,0,0,0,0,0}	{0,6}	{2,2,2}				X
24	{8,2,2,0,0,0,0,0}	{0,2,8}	{3,3,3,1}				X
25	{8,4,0,0,0,0,0,0}	{0,4, 8}	{2,3,2,1}				X
26	{9,1,1,1,1,1,1,1}	{1,9}	{7,2,1}				X
27	{10,2,0,0,0,0,0,0}	{0,2,10}	{2,3,2,1}				X
28	{8,8,0,0,0,0,0,0}	{0,8}	{2,2,2}				X
29	{10,6,0,0,0,0,0,0}	{0,6,10}	{2,3,2,1}				X
30	{12,0,0,0,0,0,0,0}	{0,12}	{1,2,1}				X
31	{12,4,0,0,0,0,0,0}	{0,4,12}	{2,3,2,1}				X
32	{10,10,0,0,0,0,0,0}	{0,10}	{2,2,2}				X
33	{14,2,0,0,0,0,0,0}	{0,2,14}	{2,3,2,1}				X
34	{12,8,0,0,0,0,0,0}	{0,8,12}	{2,3,2,1}				X
35	{16,0,0,0,0,0,0,0}	{0,16}	{1,2,1}				X

The last value of the decomposition order for the leader $K_a = 20$ is stored separately because this leader is the only one with 4 different values, the second dimension of the decomposition order being thus reduced from 4 to 3.

Figure 32 gives an encoding example for the leader $K_a = 20$.
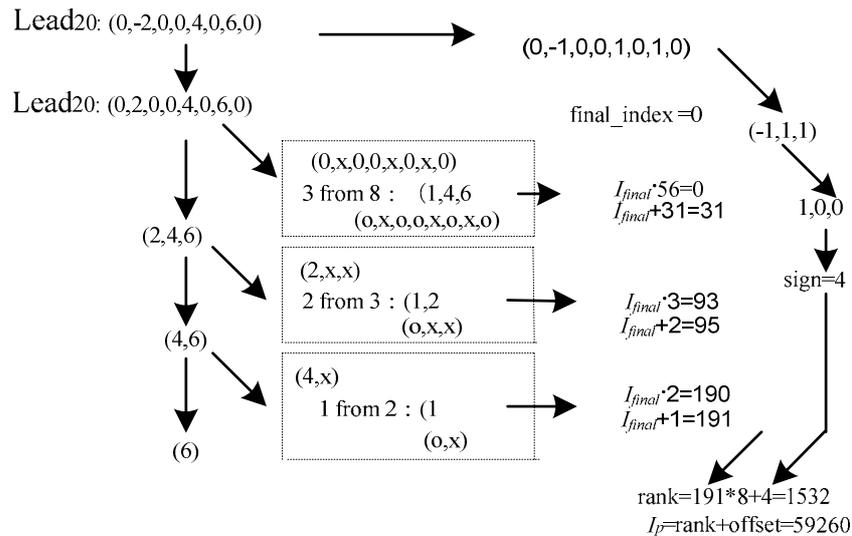
Figure 32: Example processing for $K_a = 20$.

For example, in case the input vector is {0,–2,0,0,4,0,6,0}, the absolute input vector will be {0,2,0,0,4,0,6,0}, its associated leader can be found for $K_a = 20$ K_a. The set of decomposition order is {0,2,4,6}. For the highest level L_0, element "0" is removed first from the absolute vector. The first level L_1 absolute vector is {2,4,6}, its position vector is {1,4,6}. The second element which will be removed is "2", the second level L_2 absolute vector is {4,6}, its position vector is {1,2}. The third element which will be removed is "4", the third level L_3 absolute vector is {6}, its position vector is {1}.

The absolute vectors that have only two different values, out of which the most frequent is zero, are treated separately in a less complex procedure combining the encoding of the position vector with the sign encoding. These vectors have generally higher probability of occurrence. Example of such vectors are those derived for instance from the leaders: (2,2,0,0,0,0,0,0), (2,2,2,2,0,0,0,0). For these vectors there is a single level for the creation of the index and the first level remaining elements are the non-null components which are the significant elements for the sign encoding. The determination of the remaining elements and the creation of the sign index can be done thus in a single loop.

5.2.3.1.6.9.4 Voronoi extension determination and indexing

If the nearest neighbour $\mathbf{c}_j$ is not in the base codebook, then the Voronoi extension has to be determined through the following steps.

(a)   Set the Voronoi extension order $r_j^v = 1$ and the scaling factor $M_j^v = 2^{r_j^v}$.

(b)   Compute the Voronoi index $\mathbf{I}_j^v$ of the lattice point $\mathbf{c}_j$ that depends on the extension order r_j^v and the scaling

factor M_j^v. The Voronoi index is computed via component-wise modulo operations such that $\mathbf{I}_j^v$ depends only

on the relative position of $\mathbf{c}_j$ in a scaled and translated Voronoi region:

$$\mathbf{I}_j^v = \mathrm{mod}_{M_j^v}(\mathbf{c}_j \mathbf{G}^{-1}) \tag{554}$$

where $\mathbf{G}$ is the RE_8 generator matrix. Hence, the Voronoi index $\mathbf{I}_j^v$ is a vector of integers with each component

in $[0, M_j^v - 1]$.

(c)   Compute the Voronoi codevector $\mathbf{v}_j$ from the Voronoi index $\mathbf{I}_j^v$. The Voronoi codevector is obtained as

$$\mathbf{v}_j = \mathbf{y}_{1j} - M_j^v \cdot \hat{\mathbf{y}}_{2j} \tag{555}$$

where $\hat{\mathbf{y}}_{2j}$ is the nearest neighbour of $\mathbf{y}_{2j}$ in infinite RE_8 (see subclause 5.2.3.1.6.9.2.2 for search details) and $\mathbf{y}_{1j}$ and $\mathbf{y}_{2j}$ are defined as

$$\mathbf{y}_{1j} = \mathbf{I}_j^v \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{556}$$

and

$$\mathbf{y}_{2j} = \left(\mathbf{y}_{1j} - [2,0,0,0,0,0,0,0] \right) \big/ M_j^v \tag{557}$$

(d)   Compute the difference vector $\mathbf{w}_j = \mathbf{c}_j - \mathbf{v}_j$. This difference vector $\mathbf{w}_j$ always belongs to the scaled lattice $M_j^v \cdot RE_8$. Compute $\mathbf{z}_j = \mathbf{w}_j \big/ M_j^v$, i.e. apply the inverse scaling to the difference vector $\mathbf{w}_j$. The codevector $\mathbf{z}_j$ belongs to the lattice RE_8 since $\mathbf{w}_j$ belongs to $M_j^v \cdot RE_8$ lattice.

(e)   Verify whether $\mathbf{z}_j$ is in the base codebook C_{AVQ} (i.e. in Q_3 or Q_4).

If $\mathbf{z}_j$ is not in C, increment the extension order r_j^v by 1, multiply the scaling factor M_j^v by 2, and go back to sub-step (b).

Otherwise, if $\mathbf{z}_j$ is in C, then the Voronoi extension order r_j^v has been found and the scaling factor $M_j^v = 2^{r_j^v}$ is sufficiently large to encode the index of $\mathbf{c}_j$.

5.2.3.1.6.9.3   Insertion of AVQ parameters into the bitstream

The parameters of the AVQ in each sub-band j consist of the codebook number n_j, the vector index in base codebook I_j and the 8-dimensional Voronoi index $\mathbf{I}_j^v$. The codebook numbers n_j are in the set of integers $\{0, 2, 3, 4, 5, 6, 7, 8\}$ and the size of its unary code representation is n_j bits with the exception of Q_0 that requires 1 bit and a possible overflow in the last AVQ coded sub-band. The size of each index I_j and $\mathbf{I}_j^v$ is given by $4n_j$ bits and $8r_j^v$ bits, respectively.

The AVQ parameters n_j, I_j, $\mathbf{I}_j^v$, are written sequentially in groups corresponding to the same sub-band into the corresponding bitstream as

$$[(n_0 I_0 \mathbf{I}_0^v)(n_1 I_1 \mathbf{I}_1^v)...] \,. \tag{558}$$

Note that if the lattice point in the block j is in the base codebook C, the Voronoi extension is not searched and consequently the index $\mathbf{I}_j^v$ is not written into the bitstream in this group.

The actual bit-budget needed to encode AVQ parameters in current frame varies from sub-frame to sub-frame. The difference of bits between the allocated bits and actually spent bits are unused bits that can be employed in the subsequent sub-frame or high-rate higher band coding.

5.2.3.1.7 Gain quantization

5.2.3.1.7.1 Memory-less quantization of the gains

The adaptive codebook gain (pitch gain) and the algebraic codebook gain are quantized jointly in each subframe, using a 5-bit vector quantizer. While the adaptive codebook gain is quantized directly, the algebraic codebook gain is quantized indirectly, using a predicted energy of algebraic codevector. Note that, in this case, the prediction does not use any past information which limits the effect of frame-erasure propagation.

First, energy of residual signal in dB is calculated in each subframe as

$$E_r^{[i]} = 10\log\left(\frac{1}{64}\sum_{i=0}^{63} r^2(i)\right), \qquad for\ i = 0, 1, 2, 3, \tag{559}$$

where i denotes the subframe and $r(n)$ is the residual signal, defined in subclause 5.2.3.1.1. Then, average residual signal energy is calculated for the whole frame as

$$\overline{E}_r = 0.25\sum_{i=0}^{3} E_r^{[i]} \tag{560}$$

which is further modified by subtracting an estimate of the adaptive codebook contribution. That is

$$E_i = \overline{E}_r - 0.5\left(C_{norm}^{[0]} + C_{norm}^{[1]}\right) \tag{561}$$

where $C_{norm}^{[0]}$ and $C_{norm}^{[1]}$, are as defined in subclause 5.1.10.4, are the normalized correlations of the first and the second half-frames, respectively. The result of equation (561), E_i, serves as a prediction of the algebraic codevector energy and is quantized with 3 bits once per frame. The quantized value of the predicted algebraic codevector energy is defined as

$$k_{ind} = \min_{k=0}^{7}\left|E_i - E_{book}(k)\right|$$
$$\hat{E}_i = E_{book}(k_{ind}) \tag{562}$$

where $E_{book}(k), k = 0,\ldots,2^n$ is the n-bit codebook for the predicted algebraic codevector energy and k_{ind} is the index minimizing the criterion above. The bit allocation n is bit-rate and mode dependant and is given in Table 47

Table 47: Predictor energy codebook bit allocation

Rate (kbps)	VC	GC	TC	IC/UC
7.2	n.a.	n.a.	4	n.a.
8	n.a.	n.a.	4	n.a.
9.6	3	3	n.a.	n.a.
13.2	5	4	4	n.a.
16.4	3	3	n.a.	3
24.4	3	3	n.a.	3
32	n.a.	5	5	5
64	n.a.	5	5	5

Now, let E_c denote the algebraic codebook excitation energy in dB in a given subframe, which is given by

$$E_c = 10\log\left(\frac{1}{64}\sum_{n=0}^{63} c^2(n)\right) \tag{563}$$

In the equation above, $c(n)$ is the filtered algebraic codevector, found in subclause 5.2.3.1.5.

Using the predicted algebraic codevector energy and the calculated algebraic codebook excitation energy, we may estimate the algebraic codebook gain as

$$g'_c = 10^{0.05\left(\hat{E}_i - E_c\right)} \tag{564}$$

A correction factor between the true algebraic codebook gain, g_c, and the estimated one, g'_c, is given by

$$\gamma = {g_c}\big/{g'_c} \tag{565}$$

The pitch gain, g_p, and correction factor γ are jointly vector-quantized using a n-bit codebook, where n is dependent on the bit-rate and coding mode as shown in Table 48

Table 48: Gain codebook bit allocation per subframe

Rate (kbps)	VC	GC	UC/IC
7.2	7/6/6/6	6/6/6/6	n.a.
8	8/7/6/6	8/7/6/6	n.a.
9.6	5/5/5/5	5/5/5/5	n.a.
13.2	6/6/6/6	6/6/6/6	n.a.
16.4	7/7/7/7/7	7/7/7/7/7	6/6/6/6/6
24.4	7/7/7/7/7	7/7/7/7/7	6/6/6/6/6
32	6/6/6/6/6	6/6/6/6/6	6/6/6/6/6
64	12/12/12/12/12	12/12/12/12/12	6/6/6/6/6

The gain codebook search is performed by minimizing a mean-squared weighted error between the original and the reconstructed signal, which is given by

$$E = \mathbf{x}^T\mathbf{x} + g_p^2\mathbf{y}^T\mathbf{y} + g_c^2\mathbf{z}^T\mathbf{z} - 2g_p\mathbf{x}^T\mathbf{y} - 2g_c\mathbf{x}^T\mathbf{z} + 2g_pg_c\mathbf{y}^T\mathbf{z} \tag{566}$$

where $\mathbf{x}$ is the target vector, $\mathbf{y}$ is the filtered adaptive codevector, and $\mathbf{z}$ is the filtered algebraic codevector. The quantized value of the pitch gain is denoted as $\hat{g}_p$ and the quantized value of the algebraic codebook gain is denoted as $\hat{g}_c = \hat{\gamma} g'_c$, where $\hat{\gamma}$ is the quantized value of the factor γ.

Furthermore, if pitch gain clipping is detected (as described in subclause 5.2.3.1.4.2), the last 13 entries in the codebook are skipped in the quantization procedure since the pitch gain in these entries is higher than 1.

5.2.3.1.7.2 Memory-less joint gain coding at lowest bit-rates

For the lowest bitrates of 7.2 and 8.0 kbps, slightly different memory-less joint gain coding scheme is used. This is due to the fact that there are not enough bits to cover the dynamic range of the target vector for algebraic search.

In the first subframe of the current frame, the estimated (predicted) gain of algebraic codebook is given by

$$g_{c0}^{[0]} = 10^{a_0 + a_1 CT - \log_{10}\left(\sqrt{E_c}\right)} \tag{567}$$

where CT is a signal classification parameter (the coding mode), selected for the current frame in the pre-processing part, and E_c is the energy of the filtered algebraic codevector, calculated in equation (563). The inner term inside the logarithm corresponds to the gain of innovation vector. The constants a_0 and a_1 are found by means of MSE minimization on a large signal database. The only parameter in the equation above is the coding mode CT which is constant for all subframes of the current frame. The superscript [0] denotes the first subframe of the current frame. The estimation process for the first subframe is schematically depicted in the figure below.
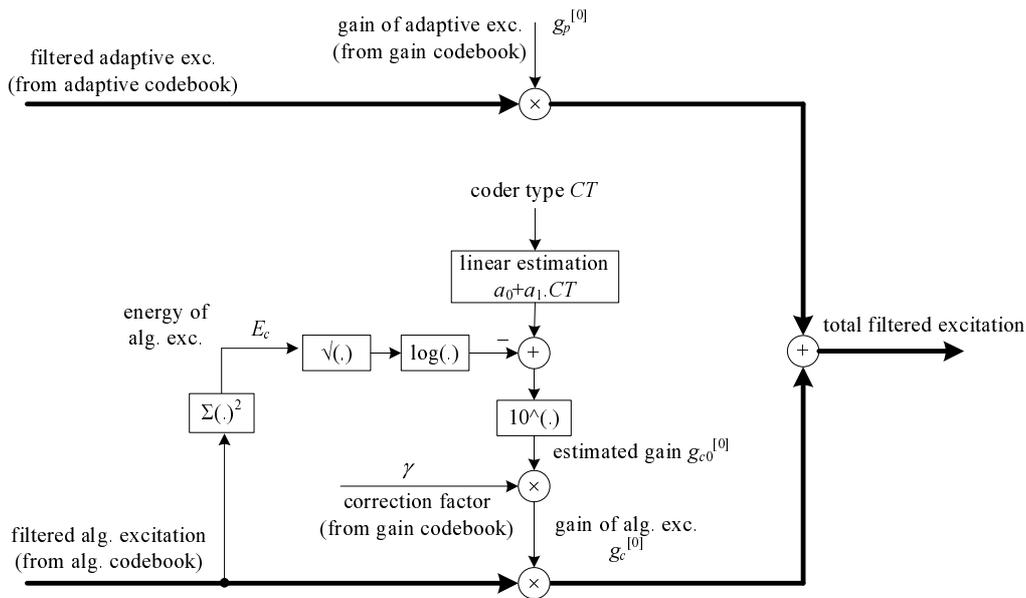
Figure 33: Schematic description of the calculation process of algebraic gain in the first subframe

All subframes following the first subframes use slightly different estimation scheme. The difference is in the fact that in these subframes, the quantized gains of both the adaptive and the algebraic codebook from previous subframe(s) are used as auxiliary estimation parameters to increase the efficiency. The estimated value of the algebraic codebook gain in kth subframe, $k>0$ is given by

$$g_{c0}^{[k]} = 10^{b_0+b_1 CT+\sum_{i=1}^{k} b_{i+1} \log_{10}(g_c^{[i-1]})+\sum_{i=1}^{k} b_{k+1+i} g_p^{[i-1]}}$$

(568)

where k=1,2,3. Note, that the terms in the first and in the second sum of the exponent, there are quantized gains of algebraic and adaptive excitation of previous subframes, respectively. Note that the term including the gain of innovation vector $\log_{10}(\sqrt{E_c})$ is not subtracted. The reason is in the use of the quantized values of past algebraic codebook gains which are already close enough to the optimal gain and thus it is not necessary to subtract this gain again. The estimation constants $b_0,\ldots,b_{2k+1}$ are found again through MSE minimization on a large signal database. The gain estimation process for the second and the following subframes is schematically depicted in the figure below.
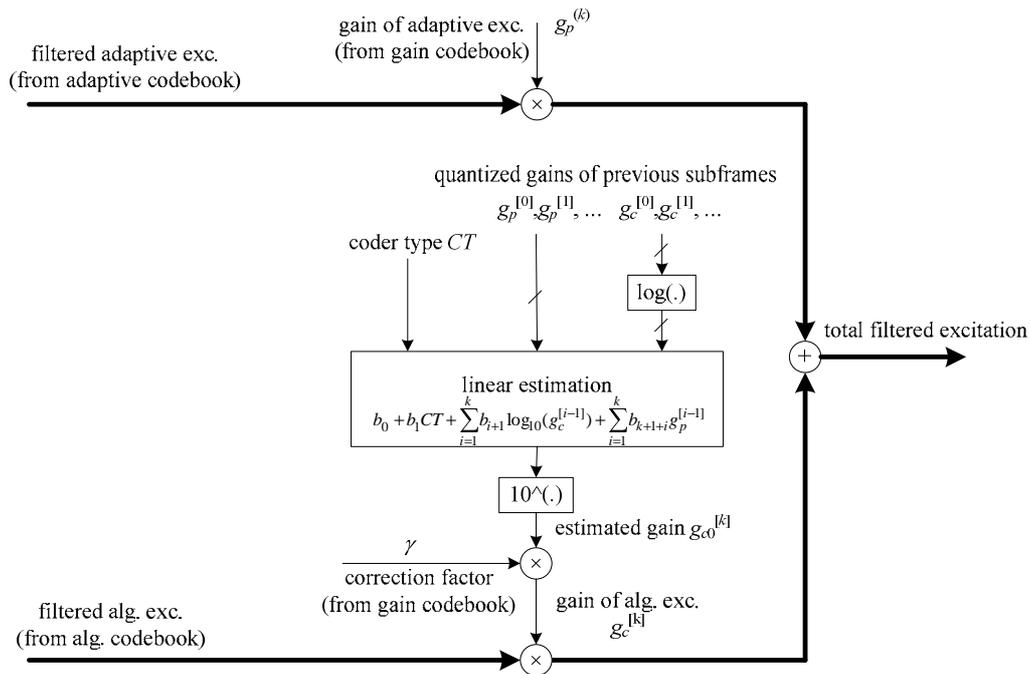
Figure 34: Schematic description of the calculation process of algebraic gain in the following subframes

The gain quantization is done both at the encoder and at the decoder by searching the gain codebook and evaluating the MMSE between the target signal and the filtered adaptive codeword. In each subframe, the codebook is searched completely, i.e. for $q=0,..,Q-1$ where Q is the number of codebook entries. It is possible to limit the searching range in case $\hat{g}_p$ is mandated to lie below certain threshold. To allow reducing the search range, the codebook entries are sorted in ascending order according to the value of $\hat{g}_p$.

The gain quantization is performed by calculating the following MMSE criterion for each codebook entry

$$E = c_0\hat{g}_p^2 + c_1\hat{g}_p + c_2[\hat{\gamma}\hat{g}_{c0}]^2 + c_3\hat{\gamma}\hat{g}_{c0} + c_4\hat{g}_p\hat{\gamma}\hat{g}_{c0} + c_5 \tag{569}$$

where the constants c_0, c_1, c_2, c_3, c_4 and c_5 are calculated as

$$c_0 = \mathbf{y}^T\mathbf{y}, \quad c_1 = \mathbf{x}^T\mathbf{y}, \quad c_2 = \mathbf{z}^T\mathbf{z}, \quad c_3 = \mathbf{x}^T\mathbf{z}, \quad c_4 = \mathbf{y}^T\mathbf{z}, \quad c_5 = \mathbf{x}^T\mathbf{x} \tag{570}$$

in which $x(i)$ is the target signal, $y(i)$ is the filtered adaptive excitation signal and $z(i)$ is the filtered algebraic excitation signal. The codevector leading to the lowest energy is chosen as the winning codevector and its entries correspond to the quantized values of g_p and γ.

Before the gain quantization process it is assumed that both the filtered adaptive and innovation codewords are already known. The gain quantization at the encoder is performed by searching the designed gain codebook in the MMSE sense. Each entry in the gain codebook consists of two values: the quantized gain of the adaptive part and the correction factor γ for the algebraic part of the excitation. The estimation of the algebraic gain excitation is done beforehand and the resulting g_{c0} is used to multiply the correction factor selected from the codebook. In each subframe the gain codebook is searched completely, i.e. for $q=0,..,Q-1$. It is possible to limit the search range if the quantized gain of the adaptive part of the excitation is mandated to be below certain threshold. To allow for reducing the search range, the codebook entries are sorted in ascending order according to the value of g_p. The gain quantization process is schematically depicted in the figure below.

filtered adapt. exc.
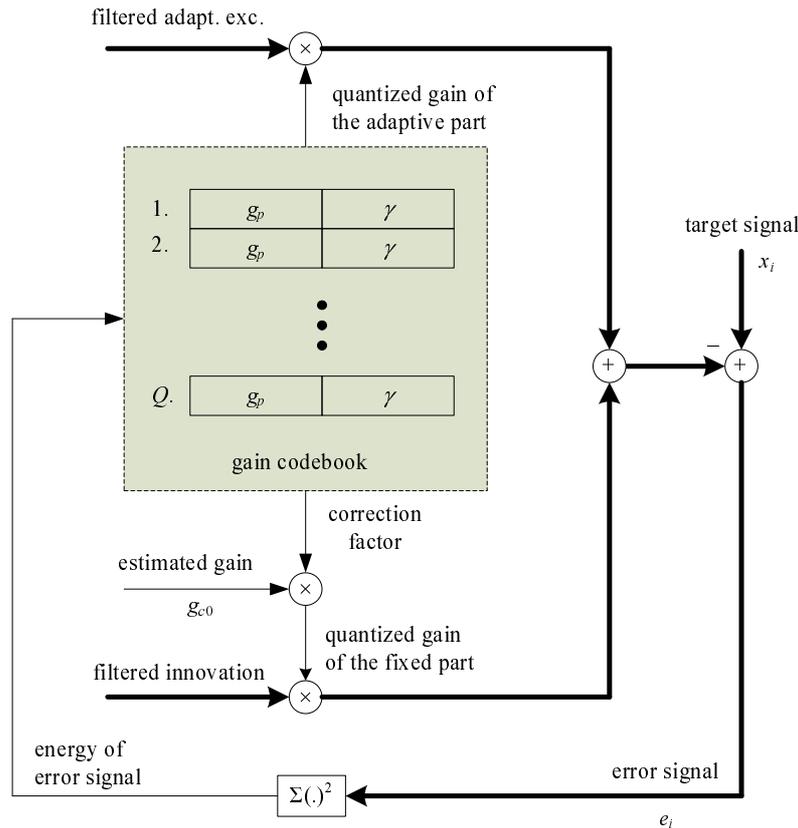
Figure 35: Schematic diagram of the gain quantization process in the encoder

The gain quantization is performed by minimizing the energy of the error signal $e(i)$ The error energy is given by

$$E = \mathbf{e}^T \mathbf{e} = (\mathbf{x} - g_p \mathbf{y} - g_c \mathbf{z})^T (\mathbf{x} - g_p \mathbf{y} - g_c \mathbf{z}) \tag{571}$$

By replacing g_c by γg_{c0} we obtain

$$E = c_5 + g_p^2 c_0 - 2g_p c_1 + \gamma^2 g_{c0}^2 c_2 - 2\gamma g_{c0} c_3 + 2g_p \gamma g_{c0} c_4 \tag{572}$$

The constants c_0, c_1, c_2, c_3, c_4 and c_5 and the estimated gain g_{c0} are computed before the search of the gain codebook. The error energy E is calculated for each codebook entry. The codevector [g_p ; γ] leading to the lowest error energy is selected as the winning codevector and its entries correspond to the quantized values of g_p and γ. The quantized value of the fixed codebook gain is then calculated as

$$g_c = g_{c0} \cdot \gamma \tag{573}$$

In the decoder, the received index is used to retrieve the values of the quantized gain of the adaptive excitation and the quantized correction factor of the estimated gain of the algebraic excitation. The estimated gain for the algebraic part of the excitation is done in the same way as in the encoder.

5.2.3.1.7.3   Scalar gain coding at highest bit-rates

At the bit-rate of 64kbps and at the last subframe of TC7 and $TC_{16}5$ (see later in subclause 5.2.3.2.2), the adaptive codebook gain (pitch gain) and the algebraic codebook gain are quantized using a scalar quantizers. The adaptive codebook gain is quantized using a uniform scalar quantizer according to MMSE criterion in the range between [0; 1.22]. In contrast the quantized algebraic codebook gain is obtained as a product of a correction factor γ and the estimated algebraic codebook gain g'_c, see equation 565, where the correction factor is quantized in log domain in the range between [0.02; 5.0].

At 64 kbps, both the adaptive codebook gain and the algebraic codebook gain are quantized by means of 6 bits each. In the last subframe of TC configurations TC7 and $TC_{16}5$, they are quantized by means of 6-8 bits depending on the bit-rate.

5.2.3.1.8 Update of filter memories

An update of the states of the synthesis and weighting filters is needed in order to compute the target signal in the next subframe.

After the two gains have been quantized, the excitation signal, $u'(n)$, in the present subframe is found by

$$u'(n) = \hat{g}_p\, v(n) + \hat{g}_c\, c(n), \qquad n = 0, \ldots, 63 \tag{574}$$

where $\hat{g}_p$ and $\hat{g}_c$ are the quantized adaptive and algebraic codebook gains, respectively, $v(n)$ is the adaptive codevector (interpolated, low-pass filtered past excitation), and $c(n)$ is the algebraic codevector (including pre-filtering). The states of the filters can be updated by filtering the signal $r(n) - u'(n)$ (difference between the residual signal and the excitation signal) through the filters $1/\hat{A}(z)$ and $A(z/\gamma_1) H_{de-emph}(z)$ and saving the states of the filters. This would require 3 stages of filtering. A simpler approach, which requires only one filtering, is as follows. The local synthesis signal (without excitation post-processing) from layer 1, $\hat{s}_1(n)$ is computed by filtering the excitation signal through $1/\hat{A}(z)$. The output of the filter due to the input $r(n) - u'(n)$ is equivalent to $x_1(n) = s(n) - \hat{s}_1(n)$. So, the states of the synthesis filter $1/\hat{A}(z)$ are given by $x_1(n), n = 48, \ldots, 63$.

The updating of the states of the filter $A(z/\gamma_1) H_{de-emph}(z)$ can be done by filtering the error signal $x_1(n)$ through this filter to find the perceptually weighted error $x_{1,w}(n)$. However, the signal $x_{1,w}(n)$ can be equivalently found by

$$x_{1,w}(n) = x(n) - \hat{g}_p\, y(n) - \hat{g}_c\, z(n) \tag{575}$$

where $x(n)$ is the adaptive codebook search target signal, $y(n)$ is the filtered adaptive codebook vector, and $z(n)$ is the filtered algebraic codebook vector. Since the signals $x(n)$, $y(n)$, and $z(n)$ are available, the states of the weighting filter are updated by computing $x_{1,w}(n)$ as in equation 575 for $n = 48, \ldots, 63$. This saves two stages of filtering.

5.2.3.2 Excitation coding in TC mode

The principle of excitation coding in TC mode is shown on a schematic diagram in Figure 36. The individual blocks and operations are described in detail in the following clauses.

5.2.3.2.1 Glottal pulse codebook search

The TC mode improves the robustness of the codec to frame erasures. It also encodes frames with an outdated past excitation buffer, e.g. after switching from HQ core frame.

The TC mode in the current frame is selected based on the classification algorithm described in subclause 5.1.13. The increased robustness, or the excitation building when the past excitation is outdated, is achieved by replacing the adaptive codebook (inter-frame long-term prediction) with a codebook of glottal impulse shapes (glottal-shape codebook) [19], which is independent from past excitation. The glottal-shape codebook consists of quantized normalized shapes of the truncated glottal impulses placed at specific positions. The codebook search consists of both the selection of the best shape and the best position.

To select the best codevector, the mean-squared error between the target signal, $x(n)$ (the same target signal as used for the adaptive codebook search described in subclause 5.2.3.1.2), and the contribution signal, $y(n)$, is minimized for all candidate glottal-shape codevectors. The glottal-shape codebook search has been designed in a similar way as the algebraic codebook search, described in subclause 5.2.3.1.5.9. In this approach, each glottal shape is represented as an impulse response of a shaping filter $G(z)$. This impulse response can be integrated in the impulse response of the

weighted synthesis filter $W(z)H(z)$ prior to the search of the optimum impulse position. The searched codevectors can then be represented by vectors containing only one non-zero element corresponding to candidate impulse positions, and they can be searched very efficiently. Once selected, the position codevector is convolved with the impulse response of the shaping filter. This procedure needs to be repeated for all the candidate shapes and the best shape-position combination will form the excitation signal.
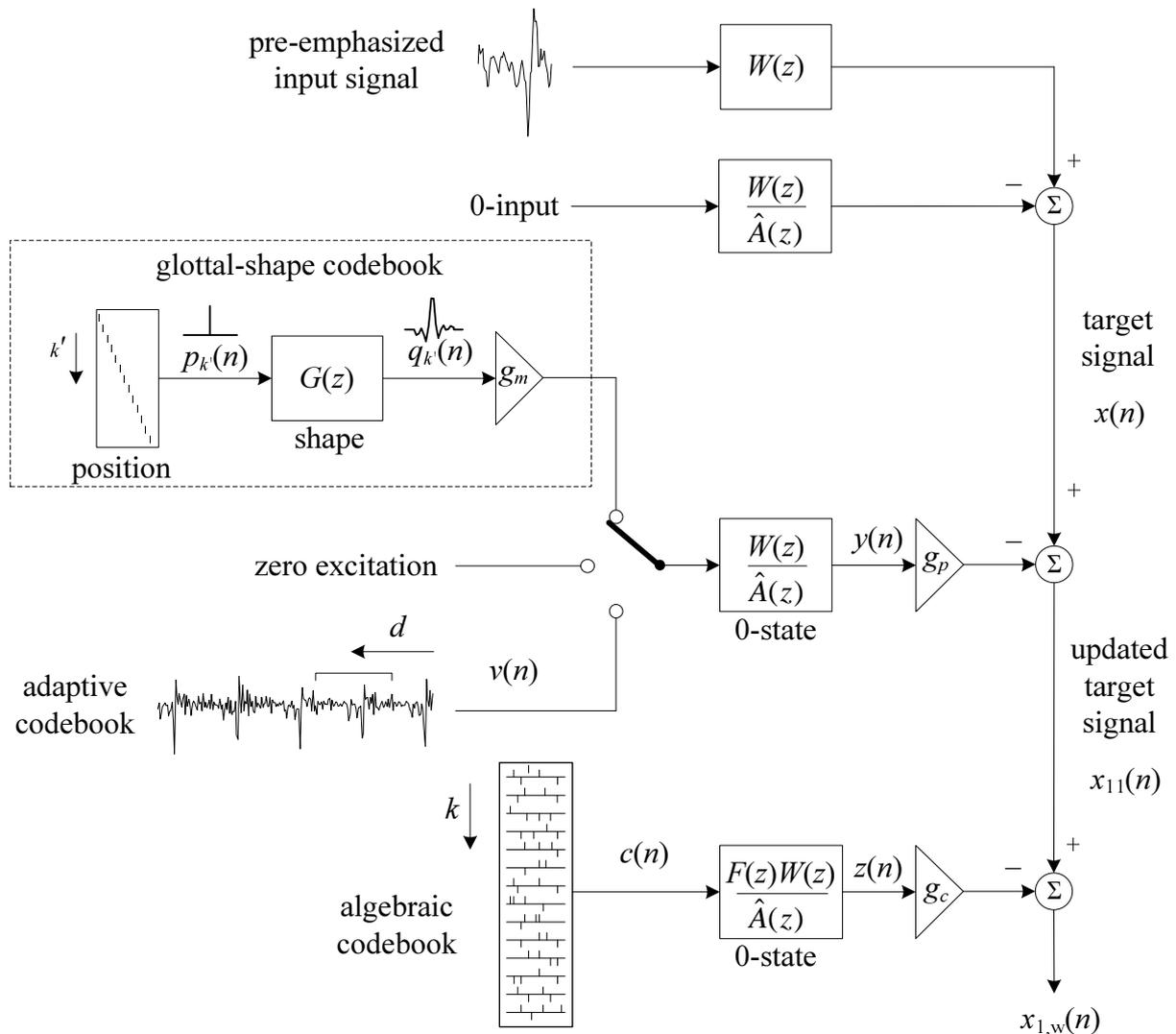
Figure 36: Schematic diagram of the excitation coding in TC mode

In the following, all vectors are supposed to be column vectors. Let $\mathbf{p}_{k'}$ be a position codevector with one non-zero element at a position k', and $\mathbf{q}_{k'}$ the corresponding glottal-shape codevector with index k' representing the centre of the glottal shape. Index k' is chosen from the range [0, 63], where 64 is the subframe length. Note that, due to the non-causal nature of the shaping filter, its impulse response is truncated for positions in the beginning and at the end of the subframe. The glottal shape codevector $\mathbf{q}_{k'}$ can be expressed in a matrix form as $\mathbf{q}_{k'} = \mathbf{G}\mathbf{p}_{k'}$, where $\mathbf{G}$ is a Toeplitz matrix representing the glottal impulse shape. Similarly to the algebraic codebook search, we can write

$$\Im_{k'} = \frac{\left(\mathbf{x}^T\mathbf{y}\right)^2}{\mathbf{y}^T\mathbf{y}} = \frac{\left(\mathbf{x}^T\mathbf{H}\mathbf{q}_{k'}\right)^2}{\mathbf{q}_{k'}^T\mathbf{H}^T\mathbf{H}\mathbf{q}_{k'}} = \frac{\left(\mathbf{x}^T\mathbf{H}\mathbf{G}\mathbf{p}_{k'}\right)^2}{\mathbf{p}_{k'}^T\mathbf{G}^T\mathbf{H}^T\mathbf{H}\mathbf{G}\mathbf{p}_{k'}} = \frac{\left(\mathbf{x}^T\mathbf{Z}\mathbf{p}_{k'}\right)^2}{\mathbf{p}_{k'}^T\mathbf{Z}^T\mathbf{Z}\mathbf{p}_{k'}} = \frac{\left(\mathbf{d}_g^T\mathbf{p}_{k'}\right)^2}{\mathbf{p}_{k'}^T\mathbf{\Phi}_g\mathbf{p}_{k'}} \tag{576}$$

where $\mathbf{H}$ is a lower triangular Toeplitz convolution matrix of the weighted synthesis filter. The rows of a convolution matrix $\mathbf{Z}^T$ correspond to the filtered shifted version of the glottal impulse shape or its truncated representation.

Because of the fact that the position codevector $\mathbf{p}_{k'}$ has only one non-zero sample, the computation of the criterion (576) is very simple and can be expressed as

$$\Im_{k'} = \frac{\left(d_g(k')\right)^2}{\Phi_g(k',k')} \tag{577}$$

As it can be seen from criterion (577), only the diagonal of the correlation matrix Φ_g from criterion (576) needs to be computed.

The codebook consists of 8 prototype glottal impulse shapes of length $L = 17$ samples placed at all subframe positions. Note that, since L is shorter than the subframe length, the remaining samples in the subframe are set to zero.

In general, the coding efficiency of the glottal-shape codebook is lower than the efficiency of the long-term prediction, and more bits are generally needed to assure good synthesized speech quality.

However, the glottal-shape codebook does not need to be used in all subframes. First, there is no reason to use this codebook in subframes that do not contain any significant glottal impulse in the residual signal. Second, the glottal-shape codebook search is important only in the first pitch period in a frame. The following pitch periods can be encoded using the more efficient standard adaptive codebook search as it does not use the excitation of the past frame anymore. To satisfy the constant bit-rate requirement, the glottal-shape codebook is used in the EVS codec only in one of the four subframes in a frame. This leads to a highly structured coding mode where the bit allocation is dependent on the position of the first glottal impulse and the pitch period. The subframe where the glottal-shape codebook is used is chosen as the subframe with the maximum sample in the residual signal in the range $\left[0, T_{op} + 2\right]$, where T_{op} is the open-loop pitch period estimated over the first half of the frame. The other subframes are processed as described in subclause 5.2.3.2.2.

Criterion (577) is typically used in the algebraic codebook search by pre-computing the backward filtered target vector $\mathbf{d}_g$ and the correlation matrix Φ_g. Given the non-causal nature of the filter $G(z)$, the matrix $\mathbf{Z}$ is not triangular and Toeplitz anymore, and this approach cannot be efficiently applied for the first $L_{1/2} = 8$ positions in the glottal-shape codebook search.

Let $\mathbf{z}_{k'}$ be the $(k'+1)$th row of the matrix $\mathbf{Z}^T$, where $\mathbf{Z}^T = \mathbf{G}^T \mathbf{H}^T$ is computed in two steps to minimize the computational complexity. In the first step, the first $L_{1/2} + 1$ rows of this matrix $\mathbf{Z}^T$ are calculated that correspond to the positions k' from the range $\left[0, L_{1/2}\right]$. In the second step, the criterion (577) is used in a similar way as in the algebraic codebook search for the remaining part of $\mathbf{Z}^T$ (the last $64 - L_{1/2} - 1$ rows of the matrix $\mathbf{Z}^T$).

In the first step, the convolution between the glottal-shape codebook entry for position $k' = 0$ and the impulse response $h(n)$ is first computed using

$$z_0(n) = \sum_{i=0}^{n} g(n-i)h(i) \qquad for\ n = 0, \ldots, 63 \tag{578}$$

where we take advantage of the fact that the filter $G(z)$ has only $L_{1/2} + 1$ non-zero coefficients.

Next, the convolution $z_1(n)$ between the glottal-shape codebook entry for the position $k' = 1$ and the impulse response $h(n)$ is computed, reusing the values of $z_0(n)$. For the following rows, the recursion is reused, resulting in

$$\begin{aligned} z_{k'}(0) &= g(-k')h(0), \\ z_{k'}(n) &= z_{k'-1}(n-1) + g(-k')h(n) \qquad for\ n = 1, \ldots, 63 \end{aligned} \tag{579}$$

The recursion (579) is repeated for all $k' \le L_{1/2}$.

Now, the criterion (577) can be computed for all positions k' from the range $\left[0, L_{1/2}\right]$ in the form

$$\Im_{k'} = \frac{\left(\sum\limits_{i=0}^{63} z_{k'}(i).x_1(i) \right)^2}{\sum\limits_{i=0}^{63} z_{k'}(i).z_{k'}(i)} \tag{580}$$

In the second step, we take advantage of the fact that rows $L_{1/2}+1,\ldots,63$ of the matrix $\mathbf{Z}^T$ are built using the coefficients of the convolution that are already computed as described by recursion (579) for $k'=L_{1/2}$. That is, each row corresponds to the previous row shifted to the right by 1 with a zero added at the beginning

$$\begin{aligned} z_{k'}(0) &= 0, \\ z_{k'}(n) &= z_{k'-1}(n-1) \qquad for \; n=1,\ldots,63 \end{aligned} \tag{581}$$

and this is repeated for k' from the range $\left[L_{1/2}+1,\ldots,63 \right]$.

Next, the target vector $\mathbf{d}_g$ and the diagonal of the matrix $\mathbf{\Phi}_g$ need to be computed. First, we evaluate the numerator and the denominator of the criterion (577) for the last position $k'=63$

$$d_g(63) = \sum\limits_{i=0}^{L_{1/2}} x\left(63 - L_{1/2} + i\right) z_{L_{1/2}}(i) \tag{582}$$

and

$$\Phi_g(63,63) = \sum\limits_{i=0}^{L_{1/2}} z_{L_{1/2}}(i)\, z_{L_{1/2}}(i) \tag{583}$$

For the remaining positions, the numerator is computed using equation (582), but with the summation index changed. In the computation of the denominator, some of the previously computed values can be reused. For example, for the position $k'=62$, the denominator of criterion (577) is computed using

$$\Phi_g(62,62) = \Phi_g(63,63) + z_{L_{1/2}}\left(L_{1/2}+1\right)\, z_{L_{1/2}}\left(L_{1/2}+1\right) \tag{584}$$

Similarly, we can continue to compute the numerator and the denominator of criterion (577) for all positions $k' > L_{1/2}$.

The search continues using the previously described procedure for all other glottal impulse shapes and the codevector corresponding to the best combination of glottal-shape and position is selected. To maintain the complexity low, the computation described above is further reduced by limiting the position search to ±4 samples around the maximum absolute value of the residual signal.

The last parameter to be determined is the gain of the glottal-shape codebook excitation. The gain is quantized in two steps. First, a roughly quantized gain of the glottal-shape codevector, g_m, is found. Then, after both the first-stage contribution (glottal-shape codevector) and the second-stage contribution (algebraic codevector) of the excitation signal are found, the gain g_p of the first-stage contribution signal is jointly quantized with the second-stage contribution gain, g_c. This is done using the memory-less gain vector quantization, as described in subclause 5.2.3.1.7.1. The found glottal shape codevector $q_{k'}(n)$ is thus the position codevector $p_{k'}(n)$ filtered through the shaping filter $G(z)$ that represents the best found glottal shape. When scaling the glottal-shape codevector $q_{k'}(n)$ with the signed quantized gain $\hat{g}_m$, we finally obtain the first stage excitation codevector, $v(n)$.

The glottal-shape gain g_m is quantized using a quantization table as follows. First, an unquantized gain in the current glottal-shape subframe is found as

$$g_m = \frac{\sum\limits_{n=0}^{N-1} x(n).y(n)}{0.01 + \sum\limits_{n=0}^{N-1} y(n).y(n)} \tag{585}$$

where N is the subframe length, $x(n)$ is the target signal and $y(n)$ is the glottal-shape codevector $q_{k'}(n)$ filtered through the weighted synthesis filter $W(z)/\hat{A}(z)$. Further, the sign of the glottal-shape gain is set to 0 if $g_m < 0$ and 1 otherwise, and written to the bitstream. Finally, the glottal-shape gain quantization index I_{gm} is found as the maximum value of I_{gm} that satisfies $\mathbf{t}(I_{gm}) < |g_m|$, where $\mathbf{t}$ is the glottal-shape gain quantization table of dimension 8. The signed quantized glottal-shape gain $\hat{g}_m$ is thus found as $\hat{g}_m = \mathbf{t}(I_{gm})$ and its value is quantized using 4 bits (1 bit for sign, 3 bits for the value).

It should be noted that the closed-loop pitch period, T_0, does not need to be transmitted anymore in a subframe which uses the glottal-shape codebook search with the exception of subframes containing more than one glottal impulse, i.e., when $(k' + T_0) < 64 + L_{1/2}$. There are situations where the pitch period of the input signal is shorter than the subframe length and, in this case, we have to transmit its value. Given the pitch period length limitations and the subframe length, a subframe cannot contain more than two impulses. In the situation that the glottal-shape codevector contains two impulses, an adaptive codebook search is used in a part of the subframe. The first T_0 samples of the glottal-shape codevector $q_{k'}(n)$ are built using the glottal-shape codebook search and then the other samples in the subframe are built using the adaptive search as shown in Figure 37.
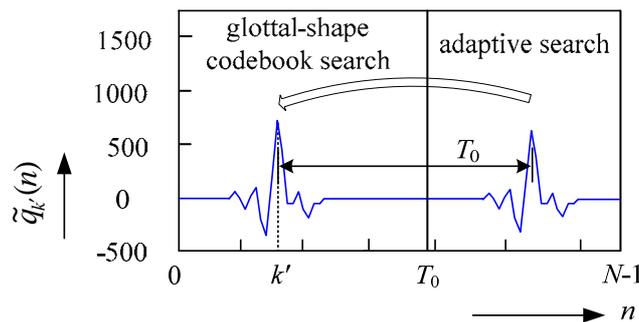
Figure 37: Glottal-shape codevector with two impulses construction

The described procedure is used even if the second glottal impulse appears in one of the first $L_{1/2}$ positions of the next subframe. In this situation, only a few samples (less than $L_{1/2} + 1$) of the glottal shape are used at the end of the current subframe. This approach has a limitation because the pitch period value transmitted in these situations is limited to $T_0 < N$, if it is bigger, it is not transmitted.

In order to enhance the coding performance, a low-pass filter is applied to the first stage excitation signal $v(n)$. In all subframes after the glottal-shape codebook subframe, the low-pass filtered first stage excitation is found as described in subclause 5.2.3.1.4.2.

5.2.3.2.2   TC frame configurations

5.2.3.2.2.1   TC frame configurations at 12.8 kHz internal sampling

At bit-rates with 12.8 kHz internal sampling rate the glottal-shape codebook is used in one out of four subframes. The other subframes in a TC frame (not encoded with the use of the glottal-shape codebook) are processed as follows. If the subframe with glottal-shape codebook search is not the first subframe in the frame, the excitation signal in preceding subframes is encoded using the algebraic CELP codebook only, this means that the first stage contribution signal is zero. If the glottal-shape codebook subframe is not the last subframe in the frame, the following subframes are

processed by the standard CELP coding (i.e., using the adaptive and the algebraic codebook search). Thus, the first stage excitation signal is the scaled glottal-shape codevector, the adaptive codevector or the zero codevector.

In order to further increase encoding efficiency and to optimize bit allocation, different processing is used in particular subframes of a TC frame dependent on the pitch period. When the first subframe is chosen as a TC subframe, the subframe with the 2nd glottal impulse in the LP residual signal is determined. This determination is based on the pitch period value and the following four situations then can appear. In the first situation, the 2nd glottal impulse is in the 1st subframe, and the 2nd, 3rd and 4th subframes are processed using the standard CELP coding (adaptive and algebraic codebook search). In the second situation, the 2nd glottal impulse is in the 2nd subframe, and the 2nd, 3rd and 4th subframes are processed using the standard CELP coding again. In the third case, the 2nd glottal impulse is in the 3rd subframe. The 2nd subframe is processed using algebraic codebook search only as there is no glottal impulse in the 2nd subframe of the LP residual signal to be searched for using the adaptive codebook. The 3rd and 4th subframes are processed using the standard CELP coding. In the last (fourth) case, the 2nd glottal impulse is in the 4th subframe (or in the next frame), the 2nd and 3rd subframes are processed using the algebraic codebook search only, and the 4th subframe is processed using the standard CELP coding. Table 49 shows all possible coding configurations in the EVS codec at 12.8 kHz internal sampling rate.

The TC configuration is transmitted in the bit-stream using a Huffman-style coding and its bit sequence is show in the Table 49 in the column bitstream.

Table 49: TC configurations used in the EVS codec at 12.8 kHz internal sampling rate

Coding configuration	Bitstream	Positions of the first (and the second, if relevant) glottal impulse(s) in the frame	Type of codebook used (GS = glottal-shape, Ada = adaptive, Alg = algebraic)			
			1st subfr.	2nd subfr.	3rd subfr.	4th subfr.
TC1	1		GS + Alg	Ada + Alg	Ada + Alg	Ada + Alg
TC2	0101		GS + Alg	Ada + Alg	Ada + Alg	Ada + Alg
TC3	0100		GS + Alg	Alg	Ada + Alg	Ada + Alg
TC4	011		GS + Alg	Alg	Alg	Ada + Alg
TC5	001		Alg	GS + Alg	Ada + Alg	Ada + Alg
TC6	0001		Alg	Alg	GS + Alg	Ada + Alg
TC7	0000		Alg	Alg	Alg	GS + Alg

5.2.3.2.2.2 TC frame configurations at 16 kHz internal sampling

At bit-rates with 16 kHz internal sampling rate the glottal-shape codebook is used in one out of five subframes. If the subframe with glottal-shape codebook search is not the first subframe in the frame, the excitation signal in preceding subframes is encoded using the algebraic CELP codebook only. If the glottal-shape codebook subframe is not the last subframe in the frame, the following subframes are processed by the standard CELP coding.

As the bit-rates with 16 kHz internal sampling rate are with high bit-budget, the number of TC configurations is reduced compared to the 12.8 kHz internal sampling rate. Table 50 shows all possible coding configurations in the EVS codec at 16 kHz internal sampling rate

Table 50: TC configurations used in the EVS codec at 16 kHz internal sampling rate

Coding configuration	Bitstream	Positions of the first glottal impulse in the frame	Type of codebook used (GS = glottal-shape, Ada = adaptive, Alg = algebraic)				
			1st subfr.	2nd subfr.	3rd subfr.	4th subfr.	5th subfr.
$TC_{16}1$	00		GS + Alg	Ada + Alg	Ada + Alg	Ada + Alg	Ada + Alg
$TC_{16}2$	01		Alg	GS + Alg	Ada + Alg	Ada + Alg	Ada + Alg
$TC_{16}3$	10		Alg	Alg	GS + Alg	Ada + Alg	Ada + Alg
$TC_{16}4$	110		Alg	Alg	Alg	GS + Alg	Ada + Alg
$TC_{16}5$	111		Alg	Alg	Alg	Alg	GS + Alg

5.2.3.2.4 Pitch period and gain coding in the TC mode

When using the TC, it is not necessary to transmit the pitch period for certain subframes. Further, it is not necessary to transmit both pitch gain, g_p, and the algebraic codebook gain, g_c, for subframes where there is no important glottal impulse, and only the algebraic codebook contribution is computed (the first stage excitation is the zero vector).

In subframes, where the glottal-shape, or adaptive, search is used, the first stage excitation gain (pitch gain), g_p, and the second stage excitation gain (algebraic gain), g_c, are quantized at bit-rates ≤ 32 kbps using the memory-less vector gain quantization described in subclause 5.2.3.1.7.1. At 64 kbps bit-rate, gains are scalar quantized as described in subclause 5.2.3.1.7.3. In glottal-shape subframes, the first stage gain, g_p, is found in the same manner as described in subclause 5.2.3.1.4.2.

When only an algebraic gain is quantized in the current frame (the first stage excitation is the zero vector), the following scalar quantization process is used. First, an optimal algebraic gain in the current subframe is found as

$$g_c = \frac{\sum_{n=0}^{N-1} x(n).z(n)}{0.01 + \sum_{n=0}^{N-1} z(n).z(n)} \tag{586}$$

where N is the subframe length, $x(n)$ is the target signal and $z(n)$ is the algebraic codevector $c(n)$ filtered through the weighted synthesis filter $W(z)/\hat{A}(z)$ with the pre-filter $F(z)$. The predictive algebraic energy calculated once per frame is employed as described in subclause 5.2.3.1.7. Further the algebraic codebook gain, g_c', and the correction factor, γ, are given by equations (564) and (565), respectively. Finally, the correction factor quantization index, I_γ, is found as the maximum value of I_γ that satisfies

$$\hat{g}_c < \left(\frac{\mathbf{t}(I_\gamma + 1) + \mathbf{t}(I_\gamma)}{2}\right).g_c' \tag{587}$$

where $\mathbf{t}$ is the algebraic gain quantization table of dimension 8. The correction factor γ is quantized with 3 bits using the quantization table $\mathbf{t}$ and the quantized algebraic gain is obtained by

$$\hat{g}_c = \mathbf{t}(I_\gamma). g_c' \tag{588}$$

The following is a list of all TC configurations corresponding to Table 49 and Table 50.

Configuration TC1

In this configuration, two first glottal impulses appear in the first subframe that is processed using the glottal-shape codebook search. This means that the pitch period value in the 1st subframe can have the maximum value less than the subframe length, i.e., $34 < T_0 < N$. Here, T_0 is the closed-loop pitch period and N the subframe length. With the ½ sample resolution it can be coded with 6 bits. The pitch periods in the next subframes are found using – depending on the bit-rate – a 5- or 6-bit delta search with a fractional resolution.

Configuration TC2

When configuration TC2 is used, the first subframe is processed using the glottal-shape codebook search. The pitch period is not needed and all following subframes are processed using the adaptive codebook search. Because we know that the 2nd subframe contains the second glottal impulse, the pitch period maximum value holds $T_0 < 2.N - 1$. This maximum value can be further reduced thanks to the knowledge of the glottal impulse position value k'. The pitch period value in the 2nd subframe is then coded using 7 bits with a fractional resolution in the whole range of $[\min(N - k', 34), 2.N - 1 - k']$. In the 3rd and 4th subframes, a delta search using 6 bits is used with a fractional resolution.

Configuration TC3

When configuration TC3 is used, the first subframe is processed using the glottal-shape codebook search with no use of the pitch value again. But because the 2nd subframe of the LP residual signal contains no glottal impulse and the adaptive search is useless, the first stage contribution signal is replaced by zeros in the 2nd subframe. The adaptive codebook parameters (T_0 and g_p) are not transmitted in the 2nd subframe. The first stage contribution signal in the 3rd subframe is constructed using the adaptive codebook search with the pitch period maximum value $(3.N - 1 - k')$ and the minimum value $(2.N - 1 - k')$, thus only a 7-bit coding of the pitch value with fractional resolution in all range is needed. The 4th subframe is processed using the adaptive search with – depending on the bit-rate – a 5- or 6-bit delta search coding of the pitch period value.

In the 2nd subframe, only the algebraic codebook gain, gc, is transmitted. Consequently, only 3 bits are needed for gain quantization in this subframe as described at the beginning of this subclause.

Configuration TC4

When configuration TC4 is used, the first subframe is processed using the glottal-shape codebook search. Again, the pitch period does not need to be transmitted. But because the LP residual signal contains no glottal impulse in the 2nd and also in the 3rd subframe, the adaptive search is useless for both these subframes. Again, the first stage excitation signal in these subframes is replaced by zeros. The pitch period value is transmitted only in the 4th subframe by means of 7 bits and its minimum value is $(3.N - k')$. The maximum value of the pitch period is limited by the $T_{\max} = 231$ value only. It does not matter if the second glottal impulse will appear in the 4th subframe or not (the second glottal impulse can be present in the next frame if $k' + T_{\max} \geq N$).

Note that the absolute value of the pitch period is necessary at the decoder for the frame concealment; therefore, it is transmitted also in the situation when the second glottal impulse appears in the next frame. When a frame m preceding the TC frame $m+1$ is missing, the correct knowledge of the pitch period value from the frames $m-1$ and $m+1$ helps to reconstruct the missing part of the synthesis signal in the frame m successfully.

The algebraic codebook gain, g_c, is quantized with 3 bits in the 2nd subframe and 3rd subframe.

Configuration TC5

When the first glottal impulse appears in the 2nd subframe, the pitch period is transmitted only for the 3rd and 4th subframe. 3rd subframe, the pitch value is coded using 9-bit absolute search while in the 4th subframe using – depending on the bit-rate – 5- or 6- bits delta search. In this case, only algebraic codebook parameters are transmitted in the 1st subframe (with the algebraic codebook gain, g_c, quantized with 3 bits).

Configuration TC6

When the first glottal impulse appears in the 3rd subframe, the pitch period does not need to be transmitted for the TC technique. In this case, only algebraic codebook parameters are transmitted in the 1st and 2nd subframe with the algebraic codebook gain, g_c, quantized with 3 bits in both subframes. Nevertheless, the pitch period is

transmitted in the 4th subframe by means of 9 bit absolute search coding for the reason of better frame erasure concealment in the frame after the TC frame. Also, the pitch period is transmitted for the 3rd subframe by means of 5 bit absolute search coding although it is not usually necessary.

Configuration TC7

When the first glottal impulse appears in the 4th subframe, the pitch period value information is not usually used in this subframe. However, its value is necessary for the frame concealment at the decoder (this value is used for the missing frame reconstruction when the frame preceding or following the TC frame is missing) or in case of strong onsets at the frame-end and very short pitch period. Thus, the pitch value is transmitted only in the 4th subframe by means of 9-bit absolute search coding and only algebraic codebook parameters are transmitted in the first three subframes (the gain pitch, g_p, is not essential). The algebraic codebook gain, g_c, is quantized with 3 bits in the 1st, 2nd and 3rd subframes. The scalar gain quantization is employed only at the 4th subframe in this configuration to encode the gain pitch and the algebraic codebook gain.

Configuration TC$_{16}$1

In this configuration, one or two first glottal impulses appear in the first subframe that is processed using the glottal-shape codebook search. This means that the pitch period value in the 1st subframe can have the maximum value less than the subframe length N, i.e., $42 < T_0 < N$ and it is coded with 6 bits. Then the pitch period in the 2nd subframe is found using 8-bit absolute search on the interval $42 < T_0 < 2*N$. Finally the pitch period in the 3rd subframe is coded using 10-bit absolute search and in the 4th and 5th subframe using 6-bit delta search.

The gain pitch and the algebraic codebook gain are coded in all subframes using 6-bit VQ at 32 kbps resp. 12-bit SQ at 64 kbps.

Configuration TC$_{16}$2

The first glottal impulse appears in the 2nd subframe and the pitch period is transmitted for the 3rd, 4th and 5th subframe. In the 3rd subframe, the pitch value is coded using 10-bit absolute search while in the 4th and 5th subframe using 6- bits delta search. The pitch period is transmitted also in the 2nd subframe by means of 6 bits and serves in case when two first glottal impulses appears in the second subframe.

The gain pitch and the algebraic codebook gain are coded in the 2nd, 3rd, 4th and 5th subframe using 6-bit VQ at 32 kbps resp. 12-bit SQ at 64 kbps. The algebraic codebook gain, g_c, is quantized in the 1st subframe with 3 bits at 32 kbps resp. 6 bits at 64 kbps

Configuration TC$_{16}$3

The first glottal impulse appears in the 3rd subframe. In this case, only algebraic codebook parameters are transmitted in the 1st and 2nd subframe with the algebraic codebook gain, g_c, quantized in both subframes with 3 bits at 32 kbps resp. 6 bits at 64 kbps. Then the pitch period is coded by means of 10-bit absolute search in the 3rd subframe and by means of 6-bit delta search in the 4th and 5th subframe.

The gain pitch and the algebraic codebook gain are coded in the 3rd, 4th and 5th subframe using 6-bit VQ at 32 kbps resp. 12-bit SQ at 64 kbps.

Configuration TC$_{16}$4

The first glottal impulse appears in the 4th subframe. In this case, only algebraic codebook parameters are transmitted in the 1st, 2nd and 3rd subframe with the algebraic codebook gain, g_c, quantized in all these subframes with 3 bits at 32kbps resp. 6 bits at 64kbps. Then the pitch period is coded by means of 10-bit absolute search in the 4th subframe and by means of 6-bit delta search in the 5th subframe.

The gain pitch and the algebraic codebook gain are coded in the 4th and 5th subframe using 6-bit VQ at 32 kbps resp. 12-bit SQ at 64 kbps.

Configuration TC$_{16}$5

When the first glottal impulse appears in the 5th subframe, the pitch period value information is not usually used in this subframe. However, its value is necessary for the frame concealment at the decoder (this value is used for the missing frame reconstruction when the frame preceding or following the TC frame is missing) or in case of strong onsets at the frame-end and very short pitch period. Thus, the pitch value is transmitted only in the 5th subframe by means of 10-bit

absolute search coding and only algebraic codebook parameters are transmitted in the first four subframes. The algebraic codebook gain, g_c, is quantized in the 1st, 2nd and 3rd subframes with 3 bits at 32kbps and 6 bits at 64 kbps. The gain pitch and the algebraic codebook gain are coded only in 5th subframe using a scalar gain quantizer.

5.2.3.2.5 Update of filter memories

In TC mode, the memories of the synthesis and weighting filter are updated as described in subclause 5.2.3.1.8. Note that signals in equation (574) are the first stage excitation signal $v(n)$ (i.e., the glottal-shape codevector, the low-pass filtered adaptive codevector, or the zero codevector) and the algebraic codevector $c(n)$ (including pre-filtering).

5.2.3.3 Excitation coding in UC mode at low rates

The principle of excitation coding in UC mode is shown in a schematic diagram in Figure 38. The individual operations are described in detail in the following clauses.
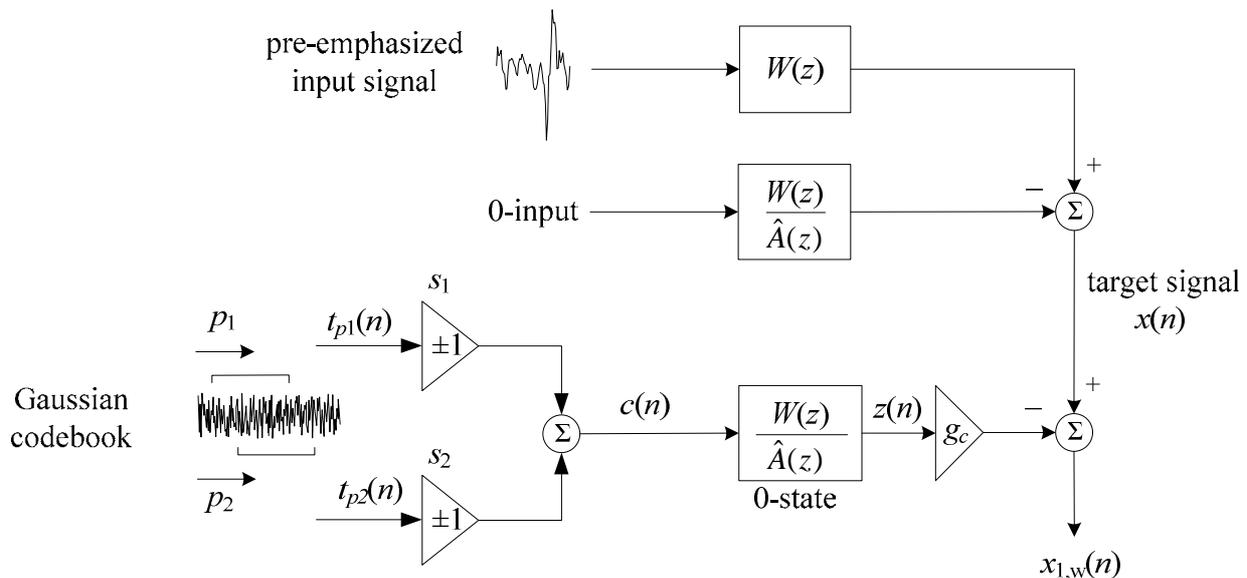
Figure 38: Schematic diagram of the excitation coding in UC mode

5.2.3.3.1 Structure of the Gaussian codebook

In UC mode, a Gaussian codebook is used for representing the excitation. To simplify the search and reduce the codebook memory requirement, an efficient structure is used whereby the excitation codevector is derived by the addition of 2 signed vectors taken from a table containing 64 Gaussian vectors of dimension 64 (the subframe size). Let $\mathbf{t}_i$ denote the i th 64-dimensional Gaussian vector in the table. Then, a codevector is constructed by

$$c = s_1 \mathbf{t}_{p_1} + s_2 \mathbf{t}_{p_2} \tag{589}$$

where s_1 and s_2 are the signs, equal to –1 or 1, and p_1 and p_2 are the indices of the Gaussian vectors from the table. In order to reduce the table memory, a shift-by-2 table is used, thus only $64 + 63 \times 2 = 190$ values are needed to represent the 64 vectors of dimension 64.

To encode the codebook index, one has to encode 2 signs, s_1 and s_2, and two indices, p_1 and p_2. The values of p_1 and p_2 are in the range $[0,63]$, so they need 6 bits each, and the signs need 1 bit each. However, 1 bit can be saved since the order of the vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ is not important. For example, choosing $\mathbf{v}_{16}$ as the first vector and $\mathbf{v}_{25}$ as the second vector is equivalent to choosing $\mathbf{v}_{25}$ as the first vector and $\mathbf{v}_{16}$ as the second vector. Thus, similar to the case of encoding two pulses in a track, only one bit is needed for both signs. The ordering of the vector indices is such that the other sign information can be easily deduced. This gives a total of 13 bits. To better explain this procedure, let us assume that the two vectors have the indices p_1 and p_2 with sign indices σ_1 and σ_2, respectively ($\sigma = 0$ if the sign is positive and $\sigma = 1$ if the sign is negative). The codevector index is given by

$$I = \sigma_1 + 2 \times (p_1 \times 6 + p_2) \tag{590}$$

If $p_1 \leq p_2$ then $\sigma_1 = \sigma_2$; otherwise σ_2 is different from σ_1. Thus, when constructing the codeword (index of codevector), if the two signs are equal then the smaller index is assigned to p_1 and the larger index to p_2, otherwise the larger index is assigned to p_1 and the smaller index to p_2.

5.2.3.3.2 Correction of the Gaussian codebook spectral tilt

In UC mode, the Gaussian codebook spectral tilt is corrected by a modification factor, which is encoded using 3 bits per subframe. First, the tilt of the target vector $x(n)$ is computed as

$$e_{tilt}^{xx} = \frac{\sum_{i=1}^{63} x(i)x(i-1)}{\sum_{i=1}^{63} [x(i)]^2} \tag{591}$$

and the tilt of the filtered Gaussian codebook $y(n)$ is computed as

$$e_{tilt}^{yy} = \frac{\sum_{i=1}^{189} y(i)y(i-1)}{\sum_{i=1}^{189} [y(i)]^2} \tag{592}$$

The filtered Gaussian codebook, $y(n)$, is the initial Gaussian codebook, $t(n)$, convolved with the weighted filter, $h(n)$. Note that vector $t(n)$ represents the whole codebook, i.e., $n = 0, \ldots, 189$.

The spectral tilt modification factor is found by

$$\delta = \frac{1 - e_{tilt}^{yy} \cdot e_{tilt}^{xx}}{2 \cdot e_{tilt}^{yy} + e_{tilt}^{xx}} \tag{593}$$

and the integer quantization index is found by

$$I_\delta = \lfloor 16\,\delta \rfloor \tag{594}$$

where the operator $\lfloor . \rfloor$ returns the integer part of a floating point number. The integer quantization index is limited to [0, 7].

Finally, the quantized spectral tilt modification factor is used to adapt the tilt of the initial Gaussian codebook. That is

$$\begin{aligned} t'(n) &= t(n) & \text{for } n = 0, \\ t'(n) &= \frac{t(n) - \hat{\delta}.t(n-1)}{1 + \hat{\delta}^2} & \text{for } n = 1, \ldots, 189 \end{aligned} \tag{595}$$

where the quantized spectral tilt modification factor is found as

$$\hat{\delta} = \frac{I_\delta}{16} \tag{596}$$

In the following, the adapted Gaussian codebook $t'(n), n = 0, \ldots, 189$, is searched to obtain the best two codevectors and signs which form the final codevector, $c(n)$, of dimension 64. In the following, we assume $t(n) \leftarrow t'(n)$.

5.2.3.3.3 Search of the Gaussian codebook

The goal of the search procedure is to find the indices p_1 and p_2 of the two best random vectors and their corresponding signs, s_1 and s_2. This is achieved by maximizing the following search criterion

$$Q = \frac{\left(\mathbf{x}^T \mathbf{z}\right)^2}{\mathbf{z}^T \mathbf{z}} = \frac{\left(\mathbf{x}^T \mathbf{H} \mathbf{c}\right)^2}{\mathbf{z}^T \mathbf{z}} = \frac{\left(\mathbf{d}^T \mathbf{c}\right)^2}{\mathbf{z}^T \mathbf{z}} \tag{597}$$

where $x(n)$ is the target vector and $\mathbf{z} = \mathbf{Hc}$ is the filtered final codevector. Note that in the numerator of the search criterion, the dot product between $x(n)$ and $z(n)$, $n = 0,\dots,63$, is equivalent to the dot product between $d(n)$ and $c(n)$, where $\mathbf{d} = \mathbf{H}^T \mathbf{x}$ is the backward filtered target vector which is also the correlation between $d(n)$ and the impulse response $h(n)$. The elements of the vector $d(n)$ are found by

$$d(n) = x(n) * h(-n) = \sum_{i=n}^{63} x(i) h(i-n) \qquad \text{for } n = 1,\dots,63 \tag{598}$$

Since $d(n)$ is independent of the codevector $c(n)$, it is computed only once, which simplifies the computation of the numerator for different codevectors.

After computing the vector $d(n)$, a predetermination process is used to identify K out of the 64 random vectors in the random table, so that the search process is then confined to those K vectors. The predetermination is performed by testing the numerator of the search criterion Q_k for the K vectors which have the largest absolute dot product (or squared dot product) between $d(n)$ and $v_i(n)$, $i = 0,\dots,63$. That is, the dot products χ_i that are given by

$$\chi_i = \sum_{n=0}^{63} d(n) v_i(n) \tag{599}$$

are computed for all random vectors $v_i(n)$ and the indices of the K vectors which result in the K largest values of $|\chi_i|$ are retained. These indices are stored in the index vector m_i, $i = 0,\dots,K-1$. To further simplify the search, the sign information corresponding to each predetermined vector is also preset. The sign corresponding to each predetermined vector is given by the sign of χ_i for that vector. These preset signs are stored in the sign vector s_i, $i = 0,\dots,K-1$.

The codebook search is now confined to the pre-determined K vectors with their corresponding signs. Here, the value $K = 8$ is used, thus the search is reduced to finding the best 2 vectors among 8 random vectors instead of finding them among 64 random vectors. This reduces the number of tested vector combinations from $64 \times 65 / 2 = 2080$ to $8 \times 9 / 2 = 36$.

Once the most promising K vectors and their corresponding signs are predetermined, the search proceeds with the selection of 2 vectors among those K vectors which maximize the search criterion Q.

We first start by computing and storing the filtered vectors $\mathbf{w}_j$, $j = 0,\dots,K-1$ corresponding to the K predetermined vectors. This can be performed by convolving the predetermined vectors with the impulse response of the weighted synthesis filter, $h(n)$. The sign information is also included in the filtered vectors. That is

$$w_j(n) = s_j \sum_{i=0}^{n} t_{m_j}(i) h(n-i) \qquad \text{for } n = 0,\dots,63, \ j = 0,\dots,K-1 \tag{600}$$

We then compute the energy of each filtered pre-determined vector as

$$\varepsilon_j = \mathbf{w}_j^T \mathbf{w}_j = \sum_{n=0}^{63} w_j^2(n) \qquad \text{for } j = 0,\dots,K-1 \tag{601}$$

and its dot product with the target vector

$$\rho_j = \mathbf{x}^T \mathbf{w}_j = \sum_{n=0}^{63} w_j(n)\, x(n) \qquad\qquad for \ j = 0,\dots,K-1 \tag{602}$$

Note that ρ_j and ε_j correspond to the numerator and denominator of the search criterion due to each predetermined vector. The search proceeds now with the selection of 2 vectors among the K predetermined vectors by maximizing the search criterion Q. Note that the final codevector is given in equation (589).

The filtered codevector $z(n)$ is given by

$$\mathbf{z} = \mathbf{Hc} = s_1 \mathbf{Ht}_{p_1} + s_2 \mathbf{Ht}_{p_2} = \mathbf{w}_{p_1} + \mathbf{w}_{p_2} \tag{603}$$

Note that the predetermined signs are included in the filtered predetermined vectors $\mathbf{w}_j$. The search criterion in equation (597) can be expressed as

$$Q = \frac{\left(\mathbf{x}^T \mathbf{z}\right)^2}{\mathbf{z}^T \mathbf{z}} = \frac{\left(\mathbf{x}^T \mathbf{w}_{p_1} + \mathbf{x}^T \mathbf{w}_{p_2}\right)^2}{\left(\mathbf{w}_{p_1} + \mathbf{w}_{p_2}\right)^T \left(\mathbf{w}_{p_1} + \mathbf{w}_{p_2}\right)} = \frac{\left(\rho_{p_1} + \rho_{p_2}\right)^2}{\varepsilon_{p_1} + \varepsilon_{p_2} + 2\mathbf{w}_{p_1}^T \mathbf{w}_{p_2}} \tag{604}$$

The vectors $\mathbf{w}_j$ and the values of ρ_j and ε_j are computed before starting the codebook search. The search is performed in two nested loops for all possible positions p_1 and p_2 that maximize the search criterion Q. Only the dot products between the different vectors $\mathbf{w}_j$ need to be computed inside the loop.

At the end of the two nested loops, the optimum vector indices p_1 and p_2 will be known. The two indices and the corresponding signs are then encoded as described above. The gain of the final Gaussian codevector is computed based on a combination of waveform matching and energy matching. The gain is given by

$$g_c = 0.6 g_w + 0.4 g_e \tag{605}$$

where g_w is the gain that matches the waveforms of the vectors $x(n)$ and $z(n)$ and is given by $g_w = \mathbf{x}^T \mathbf{z} / \mathbf{z}^T \mathbf{z}$ and g_e is the gain that matches the energies of the vectors $x(n)$ and $z(n)$ and is given by $g_e = \sqrt{\mathbf{x}^T \mathbf{z} / \mathbf{z}^T \mathbf{z}}$. Here, $x(n)$ is the target vector and $z(n)$ is the filtered codevector $c(n)$, $n = 0,\dots,63$.

5.2.3.3.4 Quantization of the Gaussian codevector gain

In UC mode, the adaptive codebook is not used and only the Gaussian codevector gain needs to be quantized. The Gaussian codevector gain in dB is given by

$$g_c^{dB} = 20\log(g_c) \tag{606}$$

is uniformly quantized between $\Gamma_{\min}$ and $\Gamma_{\max}$ with the step size given by

$$\delta = \frac{\left(\Gamma_{\max} - \Gamma_{\min}\right)}{L} \tag{607}$$

where L is the number of quantization levels. The quantization index k is given by the integer part of

$$k = \left\lfloor \frac{g_c^{dB} - \Gamma_{\min}}{\delta} + 0.5 \right\rfloor \tag{608}$$

Finally, the quantized gain in dB is given by

$$\hat{g}_c^{dB} = k \times \delta + \Gamma_{\min} \tag{609}$$

and the quantized gain is given by

$$\hat{g}_c = 10^{\hat{g}_c^{dB}/20} \tag{610}$$

In every subframe, 7 bits are used to quantize the gain. Thus, $L = 128$ and the quantization step is $\delta = 1.71875$ dB with the quantization boundaries $\Gamma_{min} = -30$ and $\Gamma_{max} = 190$. The quantized gain, $\hat{g}_c$, is finally used to form the total excitation in the UC mode by multiplying each sample of the codevector, $c(n)$, by $\hat{g}_c$.

5.2.3.3.5 Other parameters in UC mode

In UC mode, the SAD and noisiness parameters are encoded to modify the excitation vector in stationary inactive segments. The noisiness parameter is required for an anti-swirling technique used in the decoder for enhancing the background noise representation during inactive speech.

The noisiness parameter is defined as the ratio between low- and high-order LP residual variances:

$$v = \frac{\sigma_{e,2}^2}{\sigma_{e,16}^2} \tag{611}$$

where $\sigma_{e,2}^2$ and $\sigma_{e,16}^2$ denote the LP residual variances for second-order and 16th-order LP filters, respectively. The LP residual variances are readily obtained as a by-product of the Levinson-Durbin procedure, described in subclause 5.1.9.4.

The noisiness parameter is normalized to the interval [0, 1] within which it is linearly quantized with 32 levels. That is

$$\hat{v} = \left\lfloor \mu.(v-1) \middle/ 32 \right\rfloor \tag{612}$$

where μ is a normalization factor, which is different for WB and NB signals. For WB signals, $\mu = 2$, otherwise $\mu = 0.5$.

5.2.3.3.6 Update of filter memories

In UC mode, the memories of the synthesis and weighting filter are updated as described in subclause 5.2.3.1.8. Note that the excitation component $v(n)$ is missing in equation (574) for UC mode.

5.2.3.4 Excitation coding in IC and UC modes at 9.6 kbps

At 9.6 kbps, the IC and UC modes are coded with a hybrid coding embedding two stages of innovative codebooks, the algebraic pulse codebook and a Gaussian noise-like excitation. Since the long term prediction gain is expected to be very low for such frames, the adaptive codebook is not used. The principle is depicted in figure 39
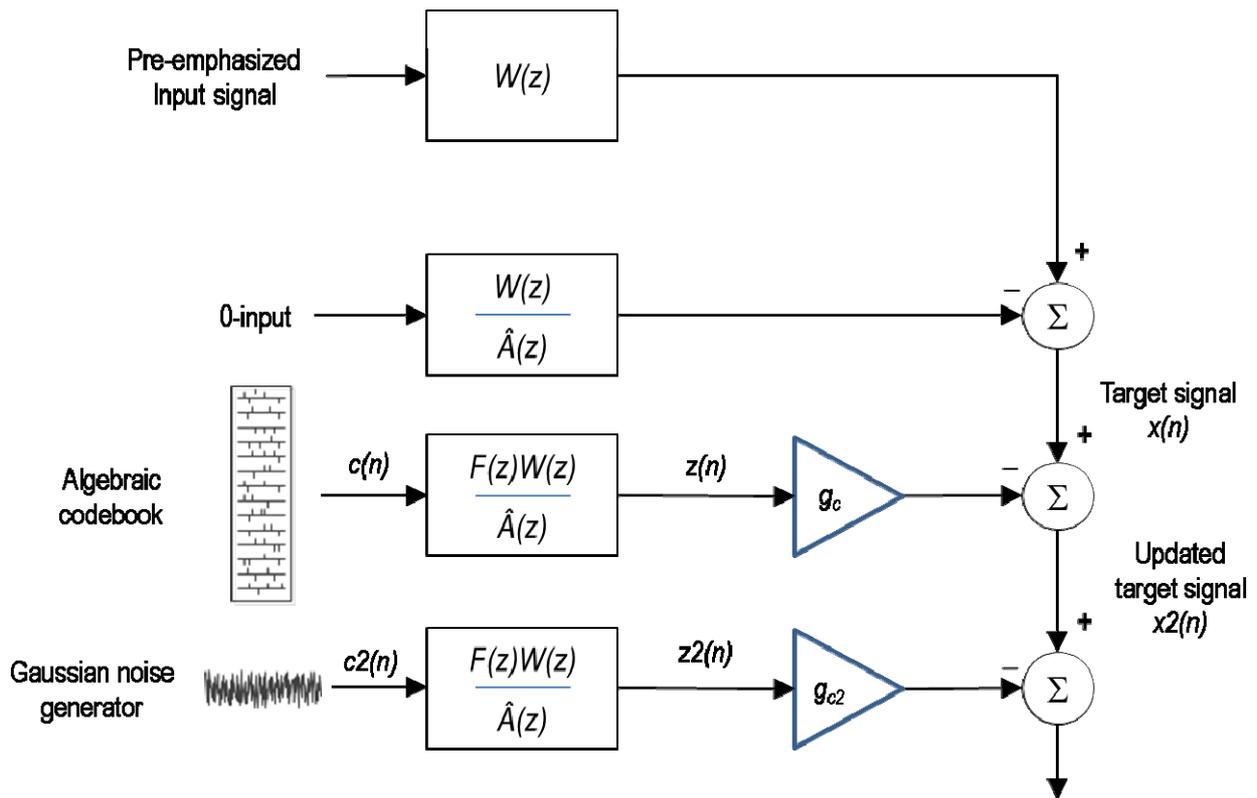
Figure 39: Schematic diagram of the excitation coding in UC and IC modes at 9.6 kbps

5.2.3.4.1 Algebraic codebook

5.2.3.4.1.1 Adaptive pre-filter

For UC mode, the adaptive pre-filter is performed similarly as in subclause 5.2.3.1.5.1. Additional the pre-filter $F(z)$ is amended with a phase scrambling filter as follows:

$$F(z) = F(z) \cdot (\frac{0.7 + z^{-1}}{1 + 0.7z^{-1}}) \tag{613}$$

For NB IC mode, the filter is designed as follows:

$$F(z) = F^{(0)}(z) \frac{\hat{A}(z/\eta_1)}{\hat{A}(z/\eta_2)} \tag{614}$$

where $F^{(0)}$ is defined in subclause 5.2.3.1.5.1 with $T = 64$, β_1 is also defined in subclause 5.2.3.1.5.1, and $\eta_1 = 0.75$ and $\eta_2 = 0.9$.

For WB IC mode, $F(z)$ is defined as:

$$F(z) = \frac{(1 - (a \cdot \beta_1 + b \cdot \gamma)z^{-1}}{1 - 0.85z^{-T}} \tag{615}$$

where $T = 64$, $a = 0.5$, $b = 0.25$ and γ represents the tilt of following filter:

$$F^{(1)}(z) = \frac{\hat{A}(z/\eta_1)}{\hat{A}(z/\eta_2)} \tag{616}$$

The tilt is computed as:

$$\gamma = -\sum_{n=1}^{63} \frac{f^{(1)}(n-1)f^{(1)}(n)}{(f^{(1)})^2(n)}$$

(617)

β_1 is bounded by [0.25 0.5] and given is by:

$$\beta_1 = 0.25 + \frac{0.25 E_v^{[-1]}}{E_v^{[-1]} + E_c^{[-1]}}$$

(618)

where $E_v^{[-1]}$ and $E_c^{[-1]}$ are the energies of the scaled pitch codevector and the scaled algebraic codevector of the previous subframe, respectively.

5.2.3.4.2 Gaussian noise generation

The Gaussian noise excitation is a second excitation added to the first innovative excitation from the algebraic codebook. This second contribution is only computed and added in WB.

The Gaussian noise excitation is produced by calling three times a random generator with a uniform distribution between -1 and +1. It follows the Central Limit Theorem.

$$c2(n) = \sum_{i=0}^{2} rand(n)$$

(619)

The Gaussian noisy excitation $c2(n)$ is spectrally shaped by applying the pre-filter $F(z)$ defined in subclause 5.2.3.4.1.1.

5.2.3.4.3 Gain coding

For NB only one gain has to be quantized, the gain of the algebraic codebook g_c. It is quantized using a 6-bit quantizer.

For WB, the two gains g_c and g_{c2} are quantized jointly in each subframe, using a 7-bit vector quantizer.

In both cases the optimal algebraic codeword gain is computed as follows:

$$g_c = \frac{\sum_{n=0}^{N-1} x(n).z(n)}{\sum_{n=0}^{63} z(n).z(n)}$$

(620)

In the equation above, $z(n)$ is the algebraic codevector $c(n)$ filtered through the weighted synthesis filter $W(z)/\hat{A}(z)$ with the pre-filter $F(z)$.is the filtered algebraic codevector.

The algebraic codebook excitation energy in dB, E_c, is also computed as follows:

$$E_c = 10\log\left(\frac{1}{64}\sum_{n=0}^{63} c^2(n)\right)$$

(621)

5.2.3.4.3.1 Innovative codebook gain coding (NB)

The algebraic codevector gain in dB is given by

$$g_c^{dB} = 20\log(g_c)$$

(622)

is uniformly quantized between -30 dB and 90dB with the step size of 1.9dB. The quantization index k is given by the integer part of

$$k = \left\lfloor \frac{g_c^{dB} + E_c + 30}{1.9} + 0.5 \right\rfloor \tag{623}$$

Finally, the quantized gain in dB is given by

$$\hat{g}_c^{dB} = k \times 1.9 - 30 \tag{624}$$

and the quantized gain is given by

$$\hat{g}_c = 10^{(\hat{g}_c^{dB} - E_c)/20} \tag{625}$$

5.2.3.4.3.2   Joint gain coding (WB)

The algebraic codebook gain is quantized indirectly, using a predicted energy of algebraic codevector. The energy of residual signal in dB is calculated.

Then, average residual signal energy is calculated for the whole frame and serves as a prediction of the algebraic codevector energy. It is quantized on 4 bits once per frame. The quantized value of the predicted algebraic codevector energy is defined as

$$k_{ind} = \min_{k=0}^{15} \left| \overline{E}r - E_{book}(k) \right|$$
$$\hat{E}_i = E_{book}(k_{ind}) \tag{626}$$

where $E_{book}(k), k = 0,\ldots,15$ is the 4-bit codebook for the predicted algebraic codevector energy and k_{ind} is the index minimizing the criterion above.

Using the predicted algebraic codevector energy, we may estimate the algebraic codebook gain as

$$g'_c = 10^{0.05(\hat{E}_i - E_c)} \tag{627}$$

A correction factor between the true algebraic codebook gain, g_c, and the estimated one, g'_c, is given by

$$\gamma = g_c \Big/ g'_c \tag{628}$$

The correction factor γ is uniformly quantized on 5 bits between -20 dB and 20dB with the step size of 1.25dB. The quantization index k is given by the integer part of

$$k = \left\lfloor \frac{\gamma + 20}{1.25} + 0.5 \right\rfloor \tag{629}$$

Finally, the quantized gain in dB is given by

$$\hat{g}_c^{dB} = k \times 1.25 - 20 - E_c + \hat{E}_i \tag{630}$$

and the quantized gain is given by

$$\hat{g}_c = 10^{\hat{g}_c^{dB}/20} \tag{631}$$

The gain of Gaussian noise excitation is quantized on 2 bits. Unlike the algebraic codeword gain, the Gaussian noise excitation gain is optimized in order to minimize the energy mismatch between the target signal and reconstructed signal. The following criterion is minimized:

$$\min_{\hat{g}_{c2}} \left| C \cdot \sum_{n=0}^{63} x^2(n) - \sum_{n=0}^{63} (\hat{g}_c \cdot z(n) + \hat{g}_{c2} \cdot z2(n))^2 \right| \tag{632}$$

where C is an attenuation factor set to 1 for clean speech, where high dynamic of energy is perceptually important and set to 0.8 for noisy speech where the noise excitation is made more conservative for avoiding fluctuation in the output energy between unvoiced and non-unvoiced frames

The quantized gain is expressed as follows where the index $k2$ of the optimal gain is sent on 2 bits:

$$\hat{g}_{c2} = (0.25k2 + 0.25) \cdot \hat{g}_c \cdot \frac{\displaystyle\sum_{n=0}^{63} \sqrt{c^2(n)}}{\displaystyle\sum_{n=0}^{63} \sqrt{c2^2(n)}} \tag{633}$$

5.2.3.4.4 Memory update

The update of the filter memories is performed as described in subclause 5.2.3.1.8 except that there is no adaptive codebook contribution. The Gaussian noise excitation is not taken into account for the update and for computing the next subframe signal target.

5.2.3.5 Excitation coding in GSC mode

In the GSC mode, the excitation is encoded using mixed time-domain/frequency-domain coding technique. This mode is aimed at encoding generic audio signals at low bit rates without introducing more delay than the ACELP structure requires. The GSC mode is used only at 12.8 kHz internal sampling rate, and the excitation could be encoded with 4 subframes, 2 subframes, or 1 subframe per frame depending on the bit rate or the signal type.

Figure 40 is a schematic block diagram showing the general concept of coding the excitation in the GSC mode. The speech/music selector is used to choose between coding the excitation signal in the GSC mode or the other ACELP modes described above. The selector mainly consists of the speech music classification (as described in subclause 5.1.13.5), where GSC is used in case music signals are detected. A further detector (as described in subclause 5.1.13.5.3) is used to verify if a detected music contains a temporal attack. In such a case the time domain transient coding mode is used to code only the attack.

When encoding in the GSC mode, the time-domain excitation contribution is first computed. In case of 4 subframes, the time-domain excitation consists of both adaptive codebook and fixed codebook as in ordinary ACELP. In case 1 or 2 subframes are used, the time-domain excitation consists only of the adaptive codebook contribution. Then the time-domain contribution and residual signal are both converted to the transform domain (using DCT). The transform-domain signals are used to determine a cut-off frequency (the upper band still containing significant pitch contribution). The time-domain excitation contribution is then filtered by removing the frequency content above the cut-off frequency. The filtered time-domain contribution in the frequency domain is subtracted from the frequency-domain residual signal, and difference signal is quantized in the frequency domain using PVQ. The quantized difference signal is then added to the filtered transformed time-domain excitation contribution, and the resulting signal is converted back to the time domain to obtain the total excitation signal.

The GSC mode is used for encoding audio signals at 7.2, and 13.2 kbit/s for WB inputs. It is also used to encode unvoiced active speech and some audio signal at 13.2 kbit/s in case of SWB inputs. Further, the GSC mode is used to encode inactive signals in case of NB, WB, SWB, and FB signals (in case DTX is off) at 7.2, 8 and 13.2kbit/s.
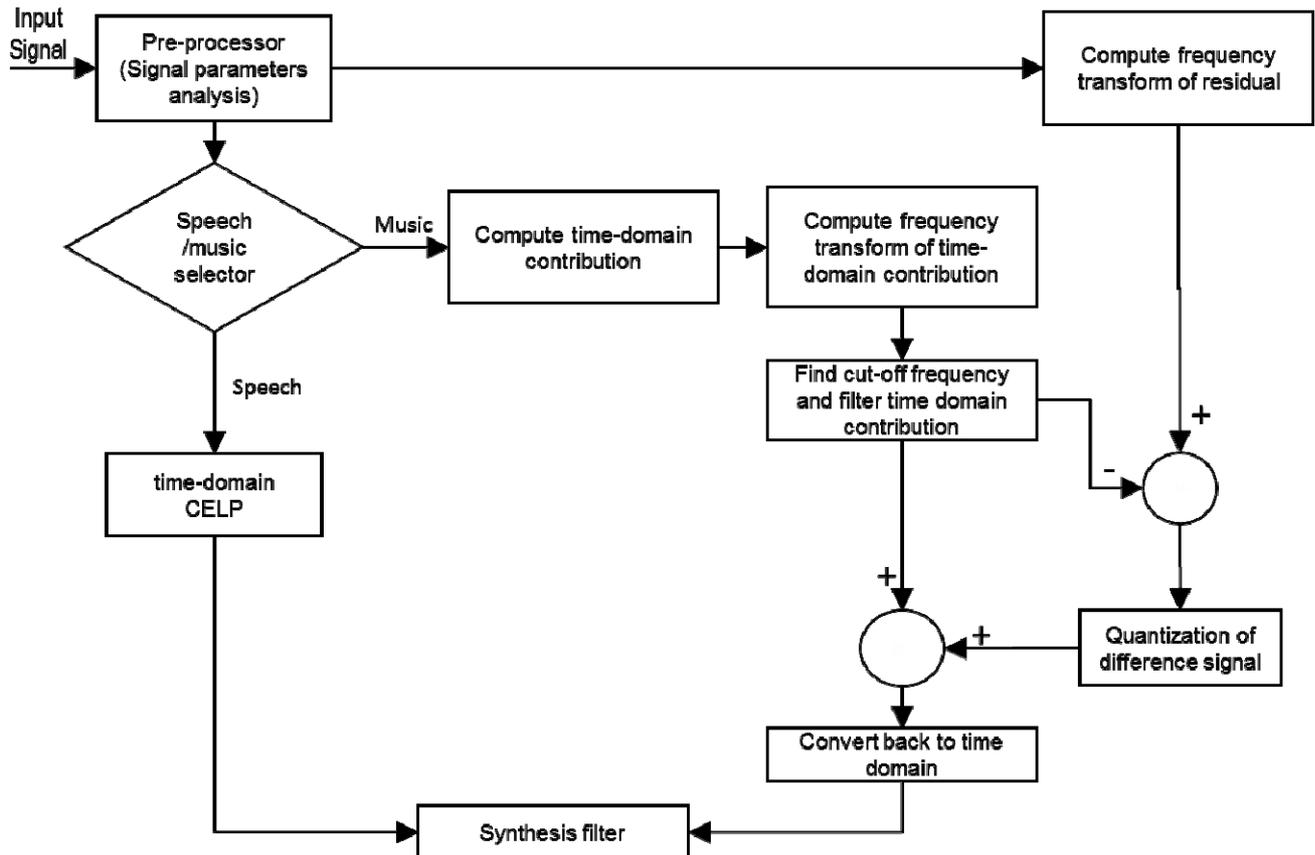
Figure 40: GSC encoder overview

5.2.3.5.1 Determining the subframe length

The subframe length, or number of subframes per frame, is determined depending on the bit rate and nature of encoded signal. In case of SWB unvoiced mode at 13.2 kbit.s, 4 subframes are used. For NB and WB signals at 13.2 kbit/s, 2 subframes are used in case of inactive signals or when the high frequency dynamic range flag F_{hf} is 0, where F_{hf} is an indicator when set to 1 is indicates the presence of high frequency spectral correlation and is computed in subclause 5.1.11.2.6. Otherwise 1 subframe is used (audio signal where long-time support is needed to get better frequency resolution).

For the bit rates of 7.2 and 8 kbit/s, 1 subframe is always used (NB, WB, and SWB audio signals and inactive speech signals). The number of subframe information is encoded at 13.2 kbps with 1 bit.

5.2.3.5.2 Computing time-domain excitation contribution

For the SWB unvoiced mode at 13.2 kbit/s, 4 subframes are used and the excitation is computed similar to ACELP Generic coding mode using both adaptive and fixed codebooks (see subclause 5.2.3.1). The signal is encoded using GENERIC coding type at 7.2 kbit/s, and the remaining bits are used to encode the frequency domain contribution. The target signal for FCB search is computed without low-pass filtering of ACB excitation.

In other modes where 1 or 2 subframes are used the time-domain excitation contributions consists only of the adaptive codebook contribution. This is determined using ordinary closed-loop pitch search as in subclause 5.2.3.1.4.1.

When only 1 or 2 subframe are used, for example at 7.2 and 8 kbit/s rates, the adaptive codebook excitation and pitch gain quantization use the AUDIO coding type. The pitch found is quantized using 10 bits for the first subframe and 6 bit for the following subframe, if any. In these case, he pitch gain is quantized using a 4 bits vector quantizer.

The total excitation is finally constructed based on both ACB and FCB at 13.2 UC mode or only ACB contribution for other modes using a total between 14 and 24 bits in case of 1 or 2 subframe up to 106 bits for the 4 subframe case..

5.2.3.5.3 Frequency transform of residual and time-domain excitation contribution

In the frequency-domain coding of the mixed time-domain / frequency-domain GSC mode, the residual signal and the time-domain excitation contribution are transformed to frequency domain. The time-to-frequency transform is performed using a 256-point Type IV discrete cosine transform (DCT_{IV}) giving a resolution of 25 Hz at the internal sampling frequency of 12.8 kHz.

The DCT_{IV}, $y(k)$, of a signal $x(n)$ of length L is defined by the following equation:

$$y(k) = \sum_{n=0}^{L-1} x(n) \cos\left[\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{L}\right], \quad k = 0,\dots,L-1 \tag{634}$$

Here $L = 256$ and $x(n)$ refers to either residual signal $r(n)$ or time-domain excitation contribution $u(n)$ with DCT output $y(k)$ corresponding to frequency transformed signals $f_r(k)$ and $f_u(k)$, respectively.

5.2.3.5.3.1 eDCT for DCT_{IV}

The efficient eDCT is built upon a discrete cosine transform type IV (DCT_{IV}) but the eDCT requires less storage and has lower complexity.

The DCT_{IV} formula in above subclause can be rewritten as:

$$\begin{cases} y(2q) = \text{Re}\{\overline{Z}(q)\} \\ y(L-1-2q) = -\text{Im}\{\overline{Z}(q)\} \end{cases}, \quad q = 0,1,\dots,L/2-1 \tag{635}$$

where the values $\overline{Z}(q)$ are given by

$$\overline{Z}(q) = W_{8L}^{-3} \cdot W_{4L}^{2q+1} \sum_{p=0}^{L/2-1} \left\{ z(p) \cdot W_{4L}^{2p+1} \right\} W_{L/2}^{pq}, \quad q = 0,1,\dots,L/2-1 \tag{636}$$

and $z(p) = x(2p) + jx(L-1-2p)$, $p = 0,1,\dots,L/2-1$, and $W_N = e^{-j\frac{2\pi}{N}} = \cos(\frac{2\pi}{N}) - j\sin(\frac{2\pi}{N})$.

Hence, the eDCT is computed using a Fast Fourier Transform (FFT) of $L/2$ points on the pre-rotated data $z(p)$:

A complex DFT with length $L/2$ is applied to the rotated data $z(p)$:

$$r(k) = \sum_{k=0}^{L/2-1} z(p) W_{L/2}^{pk} \quad\quad k = 0,\cdots,L/2-1 \tag{637}$$

Here, when $L/2 = 320 \text{ or } 160$, a simple power-2 DFT is not suitable, so it is implemented with the following low complexity 2-dimensional ($L/2 = P \times Q$) DFT, where $P = 64 \, (L/2 = 320) \text{ or } 32 \, (L/2 = 160)$ and $Q = 5$ are coprime factors.

To reduce complexity, an address table is introduced. It can be calculated by:

$$I(n_1, n_2) = (K_1 \cdot n_1 + K_2 \cdot n_2) \bmod L/2 \quad n_1 = 0,\cdots,P-1, \quad n_2 = 0,\cdots,Q-1 \tag{638}$$

where K_1, K_2 are coprime and satisfy the condition $(K_1 \cdot K_2) \bmod (L/2) = 0$.

Here, $K_1 = 65, K_2 = 256 \, (P = 64) \text{ or } 96 \, (P = 32)$. The address table I is stored for low complexity 2-dimentional DFT, and is used to indicate which samples are used for P-point DFT or Q-point DFT following:

a) Applying P-point DFT to $z(p)$ for Q times based on the address table I.

The input data to the i-th ($i = 0,\dots,Q-1$) P-point DFT is found by seeking their addresses stored in the address table I. For the i-th P-point DFT, the addresses of the input data are the P continuous elements starting from the $i \cdot P$'s element in table I. For every time of P-point DFT, the resulting data need to be applied a circular shift with a step of c_1, where c_1 is the re-ordered index, which satisfies $(c_1 \cdot ((K_1^2/Q) \bmod P)) \bmod P = 1$.

The output of step a) is:

$$w(k) = DFT\_P(z(I + iP))_{c_1} \qquad i = 0, \cdots, Q-1 \tag{639}$$

For the i-th ($i = 0,..., Q-1$) P-point DFT, the addresses of the input data are the P continuous elements starting from $I[Pi]$, and the results are circular shifted with $c_1 = 5$. Here is an example of circular shift, the original vector is $Z = [z_0\ z_1\ z_2\ z_3\ z_4]$, the new vector with 2 circular shift is $Z = [z_0\ z_2\ z_4\ z_1\ z_3]$.

b) Applying Q-point DFT to $w(k)$ for P times based on the address table I.

The input data to the i-th ($i = 0,..., P-1$) Q-point DFT is found by seeking their addresses stored in the address table I. For the i-th Q-point DFT, the addresses of the input data are the Q elements starting from the i-th element in table I, each of which separated by a step of P. For every time of Q-point DFT, the resulting data need to be applied a circular shift with a step of c_2. c_2 is the re-ordered index, which satisfies

$(c_2 \cdot ((K_2^2 / P)\,\mathrm{mod}\,Q))\,\mathrm{mod}\,Q = 1$.

The output of step b) is:

$$r(k) = DFT\_Q(w(I + i))_{c_2} \qquad i = 0, \cdots, P-1 \tag{640}$$

For the i-th ($i = 0,..., P-1$) Q-point DFT, the addresses of the input data are the Q continuous elements starting from $I[i]$, each of which is separated by a step of P, and the results are circular shifted with $c_2 = 4$ ($P = 64$) or 2 ($P = 32$). Finally, the coefficients are output according to the stored address corresponding to the address table I.

5.2.3.5.4 Computing energy dynamics of transformed residual and quantization of noise level

The DCT of the residual is divided into 16 bands (0 to 15) of length 16 bins. For bands 7 to 14, the energy dynamic per band is computed as the square of the maximum value divided by the average value per band, scaled by a factor 10. Then the average value D_{av} over the 8 band (from 7 to 14) is computed.

A long-term dynamic D_{lt} is updated as

$$D_{lt} = \begin{cases} 0.2D_{lt} + 0.8D_{av}, & when\ D_{av} > D_{lt} \\ 0.6D_{lt} + 0.4D_{av}, & otherwise \end{cases} \tag{641}$$

D_{lt} is quantized with 8 levels in the rage 50-82 (50, 54, 58, 62, 66, 70, 74, 78) with quantization index i_D from 0 to 7.

The noise level is computed as $N_{lev} = 15 - i_D$

For the bit rates of 7.2 and 8 kbit/s, the noise level is low limited to 12. Thus only values 12, 13, 14, 15 are permitted and N_{lev} is quantized with 2 bits). For UC SWB mode at 13.2 kbit/s N_{lev} is set to 15, otherwise N_{lev} is quantized with 3 bits (values 8 to 15).

5.2.3.5.6 Find and encode the cut-off frequency

The cut-off frequency consists of the last band with significant pitch contribution (the frequency after which coding improvement brought by the time-domain excitation contribution becomes too low to be valuable). Finding the cut-off frequency starts by computing the normalized cross-correlation for each frequency band between the frequency-transformed LP residual $f_r(k)$ and the frequency-transformed time-domain excitation contribution $f_u(k)$. The 256-sample DCT spectrum is divided into the 16 bands with the following number of frequency bins per band

$$B_b(i) = \{8,8,16,16,16,16,16,16,16,16,16,16,16,16,16,32\} \tag{642}$$

with cumulative frequency bins per band

$$C_{Bb}(i) = \{0,8,16,32,48,64,80,96,112,128,144,160,176,192,208,224\} \tag{643}$$

The last frequency L_f included in each of the 16 frequency bands are defined in Hz as:

$$L_f(i) = \{175,375,775,1175,1575,1975,2375,2775,3175,3575,3975,4375,4775,5175,5575,6375\} \tag{644}$$

The normalized correlation per band is defined as

$$C_c(i) = \frac{\sum_{j=C_{Bb}(i)}^{C_{Bb}(i)+B_b(i)} f_r(i)f_u(i)}{\sqrt{S_{f_r}^{'}(i)S_{f_u}^{'}(i)}} \tag{645}$$

Where $S_{f_r}^{'}(i) = \sum_{j=C_{Bb}(i)}^{C_{Bb}(i)+B_b(i)} f_r(i)f_r(i)$ and $S_{f_u}^{'}(i) = \sum_{j=C_{Bb}(i)}^{C_{Bb}(i)+B_b(i)} f_u(i)f_u(i)$.

The cross-correlation vector is then smoothed between the different frequency bands using the following relation

$$\overline{C}_{c2}(i) = \begin{cases} 2 \cdot (\min(0.5, \alpha \cdot C_c(0) + \delta \cdot C_c(1)) - 0.5), & for \ i = 0 \\ 2 \cdot (\min(0.5, \alpha \cdot C_c(i) + \beta \cdot C_c(i+1) + \beta \cdot C_c(i-1)) - 0.5), & for \ 1 \leq i < 12 \end{cases} \tag{646}$$

where $\alpha = 0.95$, $\delta = (1-\alpha)$ and $\beta = \delta/2$

The average of the smoothed cross-correlation vector is computed over the first 13 bands (representing 5575 Hz). It is then limited to a minimum value of 0.5 normalised between 0 and 1.

A first estimate of the cut-off frequency is obtained by finding the last frequency of a frequency band L_f which minimizes the difference between the last frequency of a frequency band L_f and the normalized average $\overline{C}_{c2}$ of the smoothed cross-correlation vector multiplied by the width of the spectrum of the input sound signal. That is

$$i_{\min} = \min(L_f(i) - \overline{C}_{c2}(F_s/2)), \qquad for \ 0 \leq i < 12 \tag{647}$$

and the first estimate of the cut-off frequency is given by

$$f_{tc1} = L_f(i_{\min}) \tag{648}$$

where $F_s = 12800$ Hz.

At 7.2 and 8 kbit/s, where the normalized average $\overline{C}_{c2}$ is never really high, or to artificially increase the value of f_{tc_1} to give a little more weight to the time domain contribution, the value of $\overline{C}_{c_2}$ is upscaled with a factor of 2.

The 8th pitch harmonic is computed from the minimum or lowest pitch lag value of the time-domain excitation contribution of all sub-frames, and the frequency band containing the 8th harmonic is determined. The final cut-off frequency is given by the higher value between the first estimate of the cut-off frequency f_{tc1} and the last frequency of the frequency band in which the 8th harmonic is located $L_f(i_8)$.

The cut-off frequency is quantized with a maximum of 4 bits using the values {0, 1175, 1575, 1975, 2775, 3175, 3575, 3975, 4375, 4775, 5175, 5575, 6375}.

Some hangover is added to stabilize the decision and prevent the cut-off frequency to switch between 0 (meaning no temporal contribution) and something else too often. First for the temporal contribution to be allowed, the average normalized correlation C_{norm2} and the long-term correlation $\tilde{C}_{norm2}$ as computed in subclause 5.1.13.5.3, the long term average pitch gain of the GSC temporal contribution G_{plt} and the last value of the cut-off frequency are compared to some threshold to decide if it is allowed to remove all the temporal contribution (cut-off frequency would be 0). In addition a hangover logic is used to diminish any undesired switching to a complete frequency model where the cut-off frequency would be 0.

For the lowest bitrate, 7.2 and 8.0 kbit/s, only 1 bit is used to send the cut-off frequency information when the coding mode is INACTIVE otherwise, the cut-off frequency is considered as greater than 0 (meaning the temporal contribution is used) and the length of the contribution is deduced from the pitch information. At 13 kbps, 4 bits are used to send the cut-off frequency allowing all the possible cut-off frequency values.

Once the cut-off frequency is determined, the transform of the time-domain excitation contribution is filtered in the frequency domain by zeroing the frequency bins situated above the cut-off frequency supplemented with a smooth transition region. The transition region is situated above the cut-off frequency and below the zeroed bins, and it allows for a smooth spectral transition between the unchanged spectrum below f_{tc} and the zeroed bins in higher frequencies.

5.2.3.5.7 Band energy computation and quantization

The filtered time-domain contribution in the frequency domain $f_u'(k)$ is subtracted from the frequency-domain residual signal $f_r(k)$, and the resulting difference signal in the frequency domain $f_d(k)$ is quantized with the PVQ. Before the quantization is done, some gains per frequency band B_b, as defined above, are computed and quantized using a split VQ. First the gain per band on the difference signal G_{bd} is computed as :

$$
G_{bd}(i) = \begin{cases}
\log_{10}\left(\sqrt{2.0 \cdot \left(\displaystyle\sum_{k=C_{Bb}(i)}^{k=C_{Bb}(i)+B_b(i)} f_d(k)\right)}\right), & \text{for } 0 \le i < 2 \\[4ex]
\log_{10}\left(\sqrt{\displaystyle\sum_{k=C_{Bb}(i)}^{k=C_{Bb}(i)+B_b(i)} f_d(k)}\right), & \text{for } 2 \le i < 15 \\[4ex]
\log_{10}\left(\sqrt{\left.\displaystyle\sum_{k=C_{Bb}(i)}^{k=C_{Bb}(i)+B_b(i)} f_d(k)\right/2}\right), & \text{for } i = 15
\end{cases}
\tag{649}
$$

where C_{Bb} and B_b are defined in subclause 5.2.3.5.6.

In case of NB content, only the first 10 bands are quantized using a split VQ. For other bandwidth, the number of band quantized depends on the bitrate. At low bit rate only 12 bands are quantized, being the band 0 to 8 plus the bands 10, 12 and 14. The band 9, 11, 13 and 15 being interpolated based on the quantized bands 8, 10, 12, and 14. The codebook used for the vector quantization are different depending of the bitrate and the bandwidth of the input signal giving a total 4 different set of codebooks.

In all cases, prior to the vector quantitation of the bands, the average gain of all the bands is subtract from the bands and vector quantized as well using 6 bits. In total between 21 and 26 bits are used to get the gain per band quantized $\hat{G}_{bd}$ depending of the bitrate.

5.2.3.5.8 PVQ Bit allocation

The PVQ is a coding technic that is flexible in its bit allocation. To decide where bits should be allocated inside the difference spectrum to quantize, some parameters are analyse as the bitrate, the cut-off frequency f_{tc1}, the noise level N_{lev}, the coding mode (INACTIVE, AUDIO or active UC), the bit budget available and the bandwidth.

First, only a subset of bands will be sent to the PVQ for quantization. The minimum number of band is 5 out of 16. To determine the number of band, a first criteria is the bit rate, a second criteria is the cut-off frequency and another criteria is noise level. When the number of band is decided, a minimum amount of bit is spread over the number of band decided with an emphasis on the low frequencies. If some bits remain after the minimum bit allocation, then the remaining bits are split among the bands. When the number of bands and its bit allocation are found, the bands are picked from the initial spectrum of the difference signal $f_d(k)$ based on the quantized gain of this band. The 5 first bands are always sent to the PVQ, the choice of the other bands on the energy associated to that band and the high frequency flag indicator F_{hf}.

5.2.3.5.9 Quantization of difference signal

Once the bit allocation the number of band to quantize and their position in the spectrum is defined, a new vector is concatenated containing all the chosen bands. The values are then passed to the PVQ for quantization to obtain the quantized difference spectrum $\hat{f}_d(k)$. The PVQ quantization scheme is described in subclause 5.3.4.2.7.

5.2.3.5.10 Spectral dynamic and noise filling

After the quantization by the PVQ, some band are empty and many more bins are zeroed due to the low inherent to the GSC technology available. To make the frequency model as robust as possible on speech like content, the spectral dynamic is revised and some noise filling is added to the difference spectrum.

For INACTIVE content below 13.2 kbit/s, the quantized spectrum above 1.6kHz is multiplied by a factor of 0.15. For INACTIVE content at 13.2 kHz, the quantized spectrum above 2.0kHz is multiplied by a factor of 0.25. Otherwise the scaling factor for the spectral dynamic S_{fsd} and the frequency bin where the scaling of the spectral dynamic F_{fsd} is applied is computed as follow:

$$S_{fsd} = \frac{(14 - N_{lev})}{10} - 0.4 \qquad (650)$$

and

$$F_{fsd} = 112 + (14 - N_{lev}) * 16 \qquad (651)$$

Furthermore, for frequencies above 3.2 kHz, the spectral dynamic is limited to an amplitude of ±1 for bitrate below 13.2 kbit/s and to ± 1.5 otherwise.

This scaling is then applied to the quantized difference spectrum $\hat{f}_d(k)$, to obtain its scaled version $\hat{f}_{ds}(k)$.

A noise filling is then applied to the whole difference spectrum. The noise level added is based on the bitrate, the coding mode and the spectral dynamic N_{lev} to obtain the scaled difference spectrum with noise $\hat{f}_{dsn}(k)$ on which the gain will be applied on.

5.2.3.5.11 Quantized gain addition, temporal and frequency contributions combination

Once dynamic of the quantized difference spectrum has been scaled and the noise fill has be performed, the gain of each bands is computed exactly as in subclause 5.2.3.5.7 to get gain of the quantized spectrum G_{bd2}. The gain per band to apply $G_a(i)$ consists as:

$$G_a(i) = 10^{\left(\hat{G}_{db}(i) - G_{db2}(i)\right)} \qquad (652)$$

This gain is applied to both the scaled difference spectrum with noise $\hat{f}_{dsn}(k)$ and the scaled difference spectrum $\hat{f}_{ds}(k)$ and both vectors are added to the temporal contribution to get two different spectral representation of the quantized excitation in the frequency domain, one with noise fill $f_u"(i)$ and the other without $f_{u2}"(i)$ as shown below.

$$f_u"(i) = f_u'(i) + \hat{f}_{dsn}(i) \cdot G_a(k)\Big|_{i=C_{Bb}(k)+B_b(k),\, 0 \le k < 16} \qquad (653)$$

and

$$f_{u2}"(i) = f_u'(i) + \hat{f}_{ds}(i) \cdot G_a(k)\Big|_{i=C_{Bb}(k)+B_b(k),\, 0 \le k < 16} \qquad (654)$$

5.2.3.5.12 Specifics for wideband 8kbps

The available bits are allocated to the bands of the frequency excitation signal according to the bit allocation algorithm as described in subclause 5.2.3.5.8, where the frequency excitation signal is the output of DCT_{IV} as described in subclause 5.2.3.5.3. If the index I_{high_bit} of the highest frequency band with bit allocation is more than a given threshold, the bit allocation for the frequency excitation bands will be adjusted: Decrease the number of the allocated bits of the bands with more bits, and increase the number of the allocated bits of the band I_{high_bit} and the bands near to I_{high_bit}. And then, encode the frequency excitation signal with the allocated bits, where the given threshold is determined by the available bits and the resolution of the frequency excitation signal.

The details are described as follows:

1) Allocate most of the available bits to the 5 lower frequency bands by the pre-determined bit allocation table;

2) Allocate the remaining bits to those bands excluding the lower frequency 5 bands which have the largest band energy, if there are remaining bits after the first step;

3) Search the index I_{high_bit} of the highest frequency band with bit allocation.

If the index I_{high_bit} of the highest frequency band with bit allocation is more than a given threshold $Tr_1 = 7$, the bit allocation for the frequency excitation bands will be adjusted:

1) Allocate bits to some more bands whose index is above I_{high_bit}. The number of the newly bit allocated bands N_{extr} is determined by the noise level N_{lev} and the coding mode.

2) For the newly bit allocated bands, allocate 5 bits to each band. If the number of the newly bit allocated bands $N_{extr} \leq 2$, allocate 1 more bit to each band whose index starts from 4 to $I_{high\_bit} + N_{extr}$.

3) The total number of newly allocated bits R_{extr} is $5 \cdot N_{extr} + I_{high\_bit} + N_{extr} - 4$. R_{extr} is obtained by decreasing the number of the bits allocated to the 4 lower frequency bands.

Otherwise, the original number of allocated bits to each band is not changed.

Finally, quantize and encode the frequency excitation signal according to the allocated bits.

Then, reconstruct the frequency excitation signal based on the quantized parameters. The reconstructed frequency excitation signal is corresponding to the decoded frequency excitation signal in decoder.

For the reconstructed frequency excitation signal, if the index I_{last_bin} of the highest frequency band with bit allocation is more than a given threshold, or there is the temporal contribution in the reconstructed frequency excitation signal, the frequency excitation signal above $C_{Bb}(I'_{high\_bit})$ will be reconstructed by the reconstructed frequency excitation signal; otherwise, the frequency excitation signal above $C_{Bb}(I'_{high\_bit})$ will be reconstructed by noise filling.

The detailed descriptions are as follows:

When the coding mode of previous frame is AC mode, if the last sub-band index of bit allocation I_{last_bin} is larger than $Tr_2 = 8$ or there is the temporal contribution in the reconstructed frequency excitation signal, the BWE flag F_{BWE} is set to 1. It should be noted that the BWE flag F_{BWE} is initialized to 0 and calculated for every frame. I_{last_bin} is then refined by:

$$I_{last\_bin} = \begin{cases} \max(I_{last\_bin}, 10) & if \ F_{BWE} = 1 \\ \max(I_{last\_bin}, 15) & otherwise \end{cases} \quad (655)$$

If $F_{BWE} = 1$, the frequency excitation signal below $C_{Bb}(I_{last\_bin})$ will be reconstructed as described in subclause 5.2.3.5.11, and the frequency excitation signal above $C_{Bb}(I_{last\_bin})$ will be reconstructed as follows:

$$\hat{f}_d(255-k) = f_{u2}''(159-k) \qquad 0 \le k < 255 - \left(16 \cdot I_{last\_bin} - 1\right) \tag{656}$$

And then the frequency excitation signal $\hat{f}_d(255-k)$ $0 \le k < 255 - \left(16 \cdot I_{last\_bin} - 1\right)$ is scaled by the quantized gains to obtain the scaled frequency excitation signal $f_{u2}''(255-k)$ $0 \le k < 255 - \left(16 \cdot I_{last\_bin} - 1\right)$.

When the energy ratio between the current frame and the previous frame is in the range (0.5, 2), for any band with index range is [4, 9], if the band is bit allocated in the current frame or in the previous frame, the coefficients in the band are smoothed by weighting the coefficients of the current frame and the previous frame.

For the scaled frequency excitation signal above $C_{Bb}(I_{last\_bin})$, estimate the position of the formant by LSF parameters. If the magnitudes of the coefficients near to the formant are larger than a threshold, the magnitudes are decreased to improve the perceptual quality.

Otherwise, If $F_{BWE} = 0$, the un-quantized coefficients, i.e. un-decoded coefficients at the decoder side will be reconstructed by noise filling as described in subclause 5.2.3.5.10.

5.2.3.5.13 Inverse DCT

After the gain has been applied and the combination in the frequency domain done, both frequency representations of the coded excitation are convert back to time domain using the exact same DCT as in subclause 5.2.3.5.3. The inverse transform is performed to get the quantized excitation $\hat{u}(n)$ which is the temporal representation of $f_u''(i)$ and $\hat{u}_2(n)$ which is the temporal representation of $f_{u2}''(i)$. $\hat{u}(n)$ will be used to update the TDBWE while $\hat{u}_2(n)$ is used to update the internal CELP state as the adaptive codebook memory.

5.2.3.5.14 Remove pre-echo in case of onset detection

Compute the energy of the excitation $\hat{u}(n)$ over each 4 samples using a 4-sample sliding window, and find the more energetic section to determine a possible attack (onset). If the attack is larger than the previous frame energy plus 6 dB, the algorithm finds the energy before the attack (excluding the section where the attack has been detected) and it scales it to the level of the previous frame energy plus 6dB.

5.2.4 Bass post-filter gain quantization

At 16.4 and 24.4 kbps, the bass post-filter gain is quantized on 2 bits. First the signal is reconstructed as defined in subclause 6.1.3. The bass post-filter is applied and the enhancement signal, $r(n)$ is computed as described in subclause 6.1.4.2.

The residual signal is then low pass filtered in time domain by a convolution with the impulse response $h_{lp}(n)$, impulse response simulating the filtering done in CDLFB at the decoder side. Moreover the signal is adjusted by an estimated gain α corresponding to the attenuation factor of the anti-harmonic components. It is estimated as follows;

$$s_e(n) = -\alpha \cdot r(n) * h_{lp}(n) \tag{657}$$

The optimal gain adjustment is computed as:

$$\alpha_\gamma = -\frac{\sum_{n=0}^{63}(s(n)-\hat{s}(n)) \cdot s_e(n)}{\sum_{n=0}^{63}(s(n)-\hat{s}(n))^2} \tag{658}$$

The optimal gain adjustment is quantized on 2 bits as follows:

$$k = \min(3, \max(0, \lfloor 2 \cdot \alpha_\gamma + 0.5 \rfloor)) \tag{659}$$

The quantization is adjusted in case the delta SNR provided by the estimated gain is detected as positive:

$$\Delta snr_\alpha = \sum_{i=0}^{L_{subf}} \log_{10} \frac{\sum_{n=0}^{63}(s(n+64i)-\hat{s}(n+64i))^2}{\sum_{n=0}^{63}(s(n+64i)-\hat{s}(n+64i)+s_e(n))^2} \tag{660}$$

The index k is then modified in the following way:

$$k = \max(1,k) \ \ if \ \ \Delta snr_\alpha > 0 \tag{661}$$

The again adjustment is decoded as follows

$$\hat{\alpha}_\gamma = \max(0.5 \cdot k, 0.125) \cdot \alpha \tag{662}$$

5.2.5 Source Controlled VBR Coding

5.2.5.1 Principles of VBR Coding

VBR coding [20] [21] describes a method that assigns different number of bits to a speech frame in the coded domain depending on the characteristics of the input speech signal. This method is often called source-controlled coding as well. Typically, a source-controlled coder encodes speech at different bit rates depending on how the current frame is classified, e.g., voiced, unvoiced, transient, or silence. Note that DTX operation can be combined with VBR coders in the same way as with Fixed Rate (FR) coders; the VBR operation is related to active speech segments.

The speech signal contains a varying amount of information across time, due to the way the human speech production system operates. Stationary voiced and unvoiced segments are good candidates to be encoded at lower bit-rates with minimal impact to voice quality. Transient speech contains information which is normally not well correlated to the past signal, and therefore hard to predict from the past. As a result, they are typically encoded at the higher bit-rates. Therefore it is reasonable to quantize each of the types of signals using only the necessary amount of bits, which has to be varied for maximal efficiency while at the same time minimizing the impact to voice quality.

The VBR solution provides narrowband and wideband coding using the bit rates 2.8, 7.2 and 8.0 kbps and produces an average bit rate at 5.9 kbps.

The Average Data Rate (ADR) control mechanism in subclause 5.2.5.5 relies on properties of the human speech production system, which do not apply well across different types of music signals. In such cases, the ADR for VBR mode starts approaching the most frequently used peak rate of 7.2 kbps. Due to the finer bit allocation, in comparison to Fixed Rate (FR) coding, VBR offers the advantage of a better speech quality at the same average active bit rate than FR coding at the given bit rate. The benefits of VBR can be exploited if the transmission network supports the transmission of speech frames (packets) of variable size, such as in LTE and UMTS networks.

5.2.5.2 EVS VBR Encoder Coding Modes and Bit-Rates

The signal classification algorithm described in subclause 5.1.13 forms the basis for coding mode selection in the VBR encoder. In addition to the coding modes described in subclause 5.1.13, the VBR encoder introduces two low bit-rate (2.8 kbps) coding modes called PPP (Prototype Pitch Period) for voiced speech and NELP (Noise Excited Linear Prediction) for unvoiced speech. The Transition Coding (TC) mode uses 8 kbps and all other coding modes operate at 7.2 kbps. The VBR mode targets an average bit-rate of 5.9 kbps by adjusting the proportion of the low bit-rate (2.8 kbps) and high bit-rate (7.2, 8 kbps) frames for optimal voice quality.

Next, we describe the VBR specific algorithmic modules and the average rate control mechanism.

5.2.5.3 Prototype-Pitch-Period (PPP) Encoding

5.2.5.3.1 PPP Algorithm

It was the perceptual importance of the periodicity in voiced speech that motivated the development of the (Prototype Pitch Period) PPP coding technique. PPP exploits the fact that pitch-cycle waveforms in a voiced segment do not change quickly in a frame. This suggests that we do not have to transmit every pitch-cycle to the decoder; instead, we could transmit just a representative prototype pitch period. At the decoder, the non-transmitted pitch-cycle waveforms could then be derived by means of interpolation. In this way, very low bit-rate coding can be achieved while maintaining high quality reconstructed voiced speech. Figure 41 illustrates an example of how the pitch-cycle waveforms are extracted and interpolated. We will refer these pitch-cycle waveforms as Prototype Pitch Periods (abbreviated as PPPs).
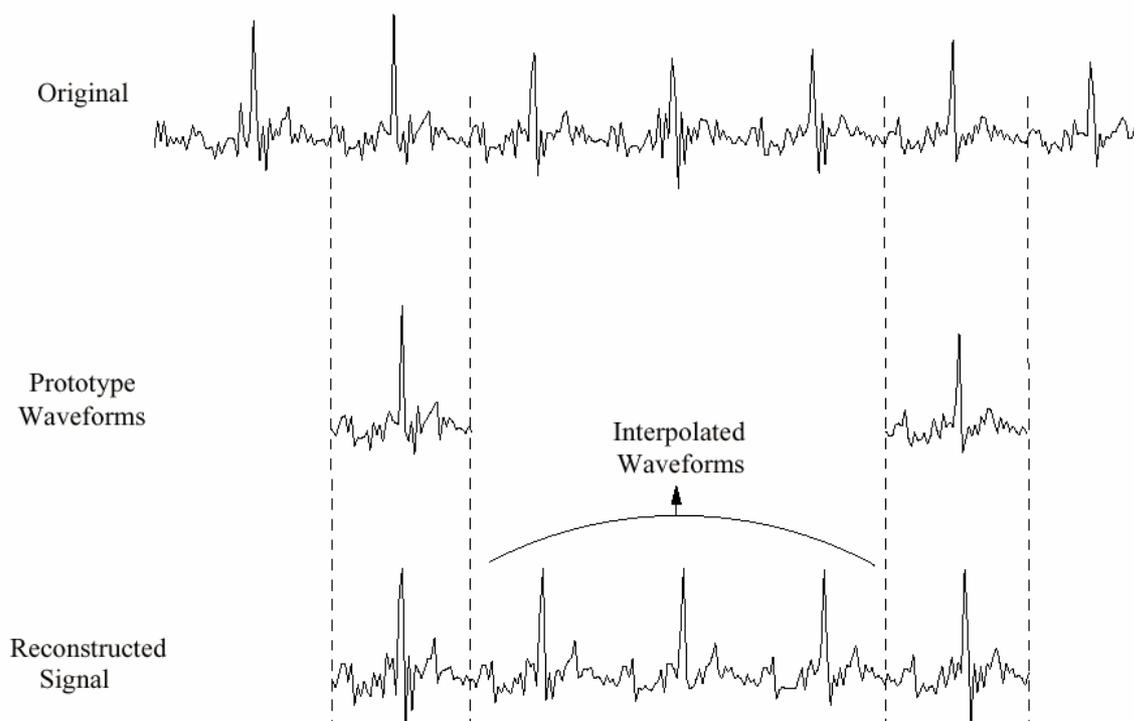
Figure 41: Illustration of principles of PPP coding [20]

The quantization of PPP is carried out in frequency domain. Hence the time-domain signal is converted to a discrete-Fourier series (DFS), whose amplitudes and phases are independently quantized.

Figure 42 presents a high-level schematic diagram of PPP coding scheme. Front-end processing including LPC analysis is same for this scheme. The LSP quantization scheme for the PPP mode is the same as that of the Voiced Coding (VC) mode.
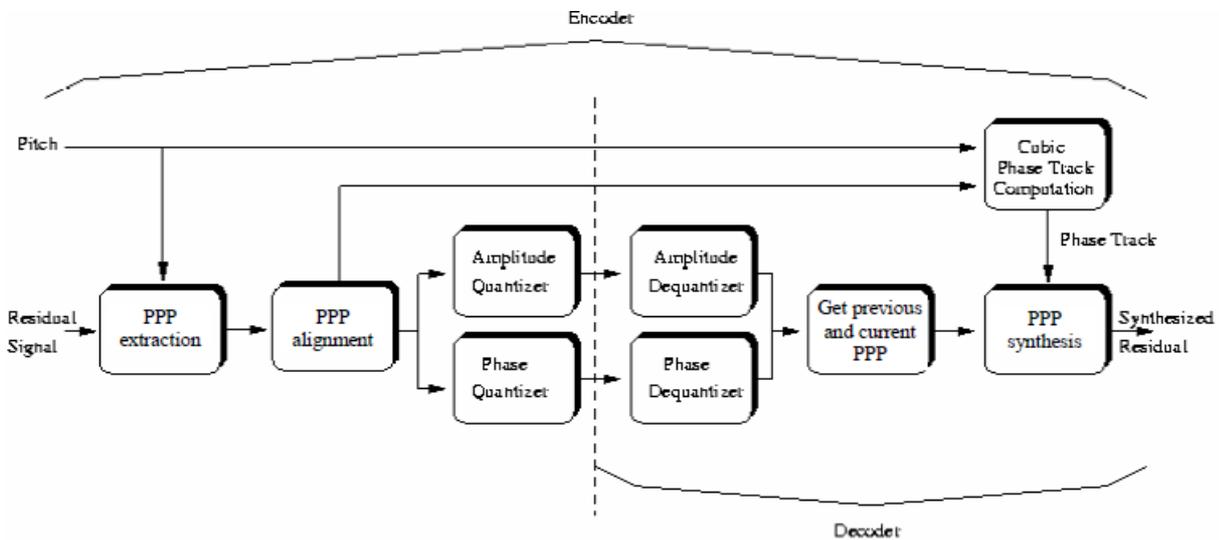
Figure 42: Block diagram of PPP coding scheme [20]

After computing the residual signal, a PPP is extracted from the residual signal on a frame-wise basis. The length of the PPP is determined by the lag estimate supplied by the pitch estimator. Special attention needs to be paid to the boundaries of the PPP during the extraction process. Since each PPP will undergo circular rotation in the alignment process (as will be seen later), the energy around the PPP boundaries needs to be minimized to prevent discontinuities after circular rotation (where the left side of the PPP meets the right side). Such minimization can be accomplished by slightly jittering the location of the extraction. After the extraction, the PPP of the current frame needs to be time-aligned with the PPP extracted from the previous frame. Specifically, the current PPP is circularly shifted until it has the maximum cross-correlation with the previous PPP. The alignment process serves two purposes: Firstly, it facilitates PPP quantization especially in low-bit-rate predictive quantization schemes, and secondly it facilitates the construction of the whole frame of excitation in the synthesis procedure which will be discussed next. The aligned PPPs are then quantized and transmitted to the decoder. To create a full-frame of excitation from the current and previous PPPs, we need to compute an instantaneous phase track which is designed carefully so as to achieve maximum time-synchrony with the original residual signal. For this purpose, a cubic phase track can be employed along with four boundary conditions:

    (1) the initial lag value,

    (2) the initial phase offset,

    (3) the final lag value and

    (4) the final phase offset.

Having created the entire frame of quantized residual, the decoder concludes its operation with LPC synthesis filter and memory updates.

5.2.5.3.2   Amplitude Quantization

The DFS amplitude is first normalized by two scaling factors at the encoder – one for the low band and one for the high band. The two resulting scaling factors are vector-quantized in the logarithmic domain and transmitted to the decoder. The normalized spectrum is non-uniformly-downsampled/averaged to transform a variable dimension vector (pitch lag dependent) to a fixed dimension vector. The Equivalent Rectangular Bandwidth (ERB) auditory scale is used for the downsampling/averaging process which helps to model the frequency-dependent frequency resolution of the human auditory system and removes perceptually irrelevant information in the spectra. The downsampled spectrum is split into a low band and high band spectra, each of which is separately quantized. At the decoder, the low and the high band

spectra are first reconstructed from the bit-stream transmitted from the encoder. The two spectra are then combined and sent to a non-uniform spectral upsampler. Afterwards, the scaling factors recovered from the bit-stream are used to denormalize the upsampled spectrum to reconstruct the quantized spectrum. Both the scaling factors and the spectra are quantized differentially to ensure minimal bit consumption.

5.2.5.3.3 Phase Quantization

The phase spectrum of the current PPP can be readily derived by combining the phase spectrum of the previous PPP and the contributions from a simplified pitch contour derived from the previous and current frame pitch lags. Finally, the number of samples needed to align the pitch pulse of the quantized PPP with that of the original residual signal is computed and sent to the decoder. This helps with closed loop pitch search in the subsequent ACELP frame.

5.2.5.4 Noise-Excited-Linear-Prediction (NELP) Encoding

The objective of NELP coding is to accurately capture unvoiced segments of speech using a minimal number of bits per frame. Front-end processing including LPC analysis is same for this scheme. The LSP quantization scheme for the NELP mode is the same as that of the Unvoiced Coding (UC) mode. The resulting residual signal is then divided into smaller sub-frames whose gains are computed and quantized. The quantized gains are applied to a randomly generated sparse excitation which is then shaped by a set of bandpass filters. The spectral characteristics of the shaped excitation are analyzed and compared to the spectral characteristics of the original residual signal. Based on this analysis, a filter is chosen to further shape the spectral characteristics of the excitation to achieve optimal performance.

5.2.5.5 Average Data Rate (ADR) Control for the EVS VBR mode

This section describes an ADR control mechanism which ensures the compliance of ADR requirements under a wide variety of language and noise type mix. The average rate control is done by a combination of threshold changes and the change of the high rate and low rate frame selection pattern. A set of thresholds referred to as bump-up thresholds play a key role in the rate control algorithm. When coding PPP frames, the encoder runs a set of checks to verify whether the given frame is suited for PPP mode of coding. If the set of checks fail the PPP coding module decides the given frame is not suitable for PPP coding and the frame is coded as a high rate frame (H-frame). This rejection is referred to as a "bump-up". There are two types of bump-ups used in the PPP coding module.

1. Open loop bump-ups: Bump-up is performed before the prototype pitch period wave form is coded. For example if the pitch lag difference between the current and previous frame is more than a certain threshold, the PPP coding module decides that the current frame is not suitable for PPP coding.

2. Closed-loop bump-ups: This is done after the prototype pitch period waveform is coded. For example if the energy ratio of the prototype pitch waveform before and after the quantization is not within a certain set of thresholds, the PPP coding module abandons the PPP mode of coding and subsequently that frame is coded using high rate coding. In general the close loop bump-ups make sure the PPP coding is in good quality.

The PPP coding module decides the current speech frame as clean speech or noisy speech by comparing the current frame's SNR against a threshold. This threshold is referred to as the clean and noisy speech threshold (TH) which is localized to the PPP coding module. If the SNR of the current frame is more than TH then the current frame is classified as a clean speech frame or as a noisy speech frame otherwise. For each of the two cases (clean and noisy) there are two sets of bump up thresholds, resulting four sets of bump up thresholds collectively. For clean speech the two bump up threshold sets consist of a strict set T_{CL1} (a strict set of bump up thresholds resulting more bump ups) and a relaxed set T_{CL2} (a relaxed set of bump up thresholds resulting less bump ups). Similarly there are two similar bump up threshold sets available for noisy speech T_{N1} and T_{N2}. Clean speech bump up threshold sets are more stricter compared to corresponding noisy speech bump up threshold sets (T_{CL1} creates more bump ups compared to T_{N1} and similarly T_{CL2} creates more bump ups compared to T_{N2}).

ADR Control mechanism is introduced based on multiple steps depending on long term ADR, short term ADR (ADR during last 600 frames) and the target rate. Following rate control mechanisms are picked based on the long term and short term average rates to achieve the desired average rate.

a. Change the threshold (TH) which classifies the speech as clean or noisy. Increasing TH classifies more frames as noisy speech and reduces the number frames classified as clean. At this point T_{CL1} and T_{N1} threshold sets are used for bump ups. However by classifying more frames as noisy speech, more frames will use the threshold set T_{N1} thus lesser number of bump ups will occur.

b. Use a low rate (L) and high rate (H) frame pattern which generates more low rate frames. For example we can set the default pattern to LHH and change the pattern to LLH to obtain more L frames, which reduces the ADR.

c. Use relaxed PPP bump up threshold sets (T_{CL2} and T_{N2}) for both clean and noisy speech. This reduces the number of bump ups thus more L frames are generated. In item (a) we increased the threshold (TH) which classifies the speech frames as clean or noisy without relaxing the corresponding PPP bump up thresholds (used T_{CL1} and T_{N1}).

d. Impact the open loop voicing and un-voicing decisions to reduce the rate by increasing PPP and NELP frames.

e. Apart from rate reduction mechanisms, the algorithm utilizes a speech quality improvement strategy if the global rate is less than the target rate by a specific margin. To achieve that a percentage of the L frames are sent to H frames to improve the speech quality.

The objective of the ADR control algorithm is to keep the average data rate of the VBR mode at 5.9 kbps and not to exceed it by 5%. The target rate is set by the algorithm (e.g. 5.9 kbps) and short term and long term average rates are computed to control different actions given in items (a) through (e) above.

The average rate during last N frames or the short term average rate ($R_{Last\,N\,frames} =$ average of last 600 active frames) is used to compute the long term average rate as follows. $R_{LT}(n)$ is updated after every N active frames. The value α is selected as 0.98.

$$R_{LT}(n) = \alpha.R_{LT}(n-1) + (1-\alpha).R_{Last\,N\,frames} \qquad (663)$$

The rate control is done in multiple steps. If the long term average rate R_{LT} is larger than the target rate, the clean and noise decision threshold TH (in above (a)) is increased in steps. If the R_{LT} cannot be reduced with the maximum TH value, the encoder will use the LLH pattern to generate more quarter rate PPP frames. If the R_{LT} is not reduced below the target rate, bump up thresholds are relaxed (item (c) above). Finally the open loop voiced and unvoiced decisions are relaxed such that the number of PPP and NELP frames will be increased to control the average rate.

Once the R_{LT} is reduced below the target rate, the rate control mechanisms are relaxed gradually. First the open loop decision thresholds are restored to the values before the aggressive rate control. The next step is to revert to the lesser aggressive bump up thresholds (T_{CL1} and T_{N1}). If the long term average rate is still well under the target rate, frame selection pattern LLH is reverted back to LHH and then the noise decision threshold TH is gradually reduced to its default value.

If the long term average rate R_{LT} is well under control and below a minimum target rate $R_{target} - \Delta_{tol2}$ speech quality improvement algorithm in (e) is exercised by converting some of the low rate PPP frames (L frames) into high rate H frames.

Figure 43 shows the flow chart of the average rate control based on this method. Table 51 contains the summary of the terms used in the flowchart in figure 43.

Table 51: Summary of the terms used in the flowchart

Symbol	Description
$LLH_{\text{mod} e}$	Set to 1 if the LLH pattern is used. Set to 0 if LHH pattern is used
$TH_{\max}$	Maximum value for the clean and noisy speech decision threshold
$Relax\_BMP\_TH$	Set to 1 if the relaxed bump up thresholds is used. Set to 0 otherwise
R_{TL}	Long term average data rate
$R_{t \arg et}$	Target average date rate
Δ_{tol1}	Rate tolerance 1. (e.g. set to 0.1 kbps for a target rate of 6.1 kbps target rate)
Δ_{tol2}	Rate tolerance 1. (e.g. set to 0.05 kbps for a target rate of 6.1 kbps target rate)
Δ_{th1}	Set size to increase the TH (value for the clean and noisy speech decision threshold)
Δ_{th2}	Set size to decrease the TH (value for the clean and noisy speech decision threshold)
$Relax\_OL$	Set to 1 if open loop voicing/unvoicing decisions are changed to have more PPP and NELP frames. Set to 0 otherwise.
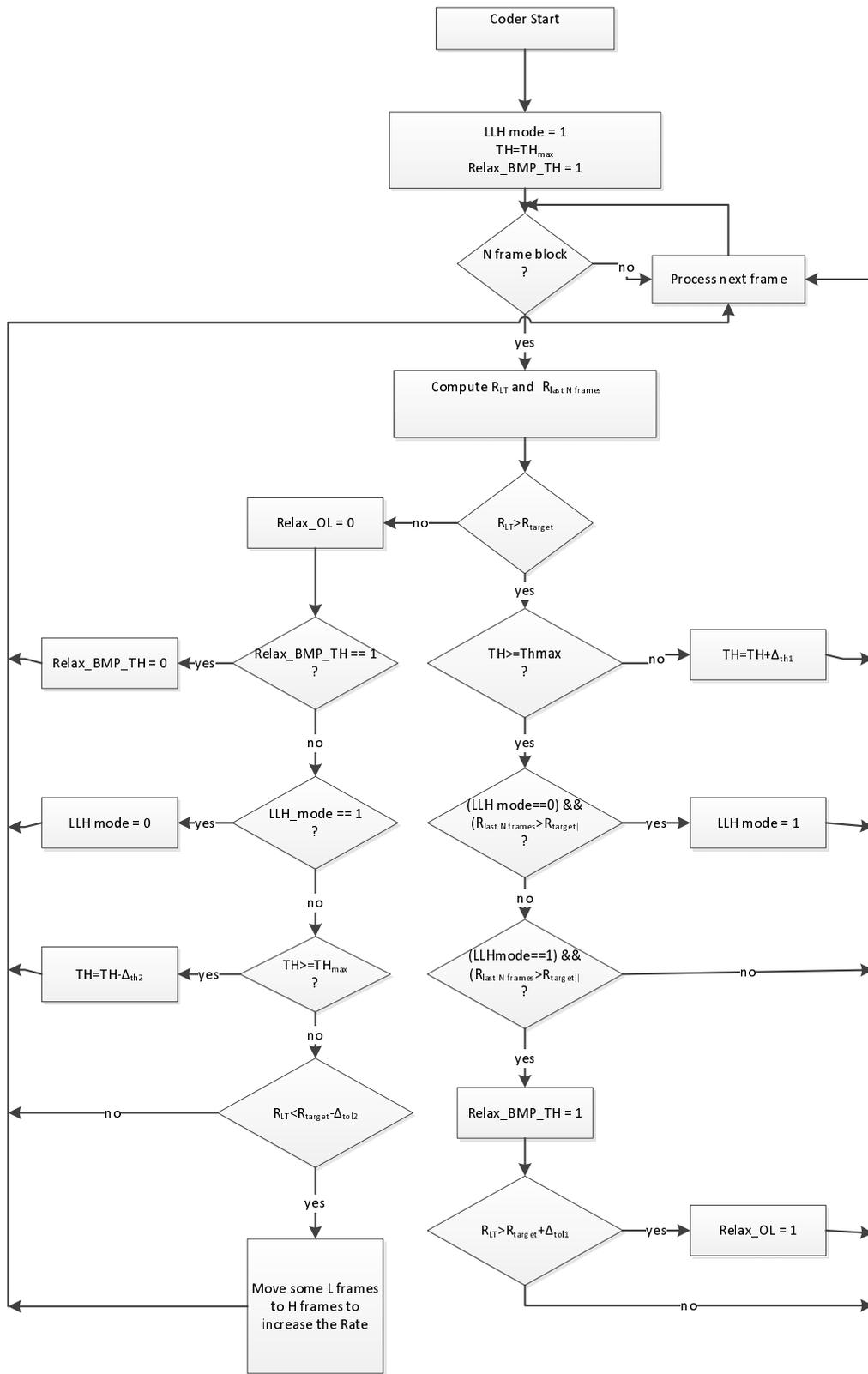
Figure 43: Average rate control in Variable Bit-Rate Coding

5.2.6 Coding of upper band for LP-based Coding Modes

5.2.6.1 Bandwidth extension in time domain

The time-domain bandwidth extension (TBE) module codes the signal content beyond the range of frequencies that are coded by the low band core encoder. The inaccuracies in the representation of the spectral and the temporal information content at higher frequencies in a speech signal are masked more easily than contents at lower frequencies. Consequently, TBE manages to encode the spectral regions beyond what is coded by the core encoder in speech signals with far fewer bits than what is used by the core encoder.

The TBE algorithm is used for coding the high band of clean and noisy speech signals when the low band is coded using the ACELP core. The input time domain signal of current frame is divided into two parts: the low band signal and the super higher band (SHB) signal for SWB signal or the higher band (HB) signal for WB signal. While the SWB TBE encoding of upper band (6.4 to 14.4 kHz or 8 to 16 kHz) is supported at bitrates 9.6 kbps, 13.2 kbps, 16.4 kbps, 24.4 kbps, and 32 kbps; the WB TBE encoding of upper band (6 to 8 kHz) is supported at 9.6 kbps and 13.2 kbps. Specifically, at 13.2 kbps and 32 kbps the SWB TBE algorithm is employed as the bandwidth extension algorithm when the speech/music classifier determines that the current frame of signal is active speech or when GSC coding is selected to encode super wideband noisy speech. Similarly, at 9.6 kbps, 16.4 kbps and 24.4 kbps the TBE algorithm is used to perform bandwidth extension for all ACELP frames. The ACELP/MDCT core coder selection at 9.6 kbps, 16.4 kbps and 24.4 kbps is based on an open-loop decision as described in subclause 5.1.14.1.

In coding the higher frequency bands that extend beyond the narrowband frequency range, bandwidth extension techniques exploit the inherent relationship between the signal structures in these bands. Since the fine signal structure in the higher bands are closely related to that in the lower band, explicit coding of the fine structure of the high band is avoided. Instead, the fine structure is extrapolated from the low band. The high level architecture of the TBE encoder is shown in figure 44 below. The front end processing to generate the high band target signal in figure 44, is replaced by a simple flip-and-decimate-by-4 operation in the WB TBE framework.
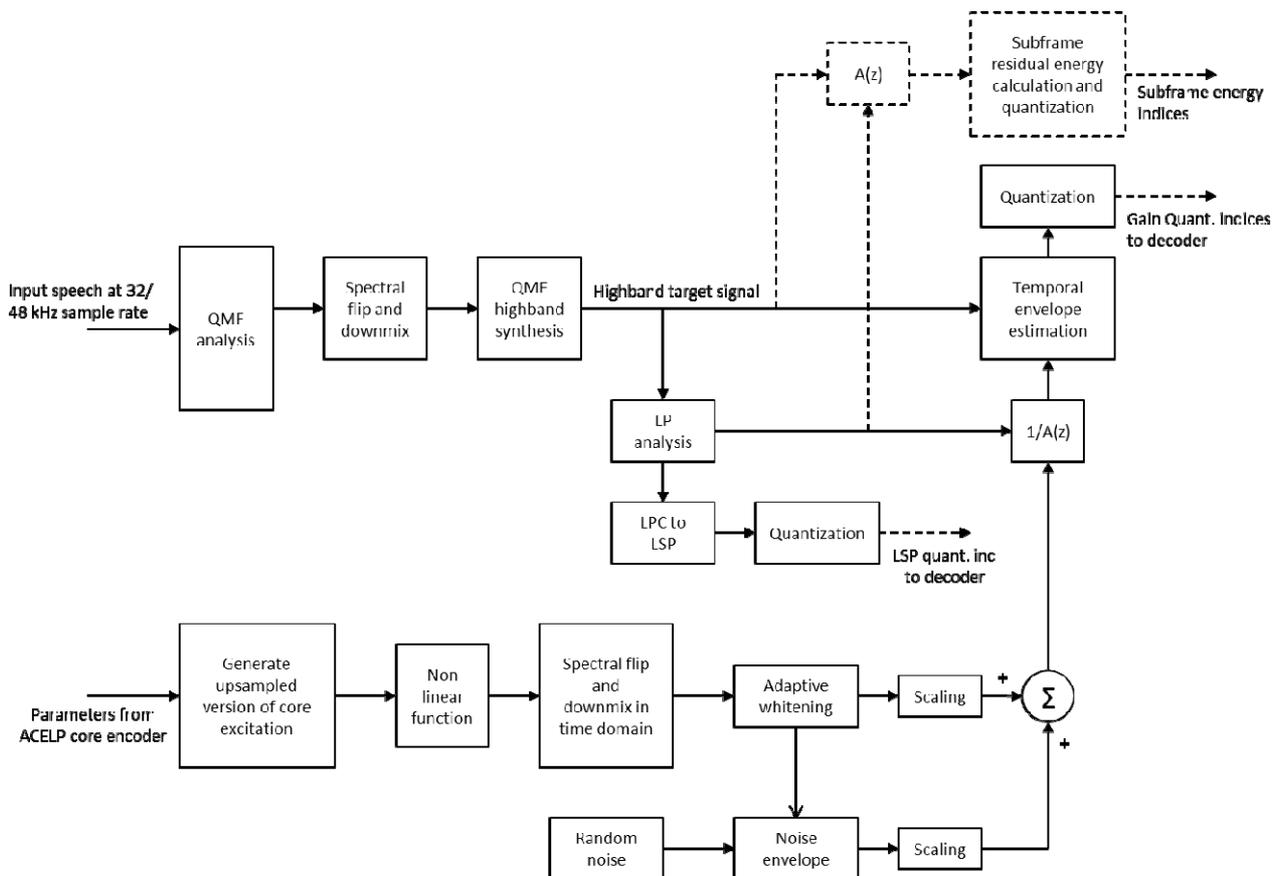
Figure 44: Time Domain Bandwidth Extension Encoder. Blocks in dashed lines are activated only at higher bit rates (24.4 kb/s or higher)

5.2.6.1.1 High band target signal generation

The input speech signal, $s_{inp}(n)$, that is sampled at either 32 kHz or 48 kHz sampling frequency, is processed through the QMF analysis filter bank. This processing step is performed as part of the common processing step as described in subclause 5.1. The output of the QMF analysis filter bank, $X_{CR}(t,k)$ and $X_{CI}(t,k)$ are the real and imaginary sub-band values, respectively, t is the sub-band time index with $0 \le t \le 15$ and k is sub-band frequency band index with $0 \le k \le L_C$. The number of sub-bands, $L_C = F_s / 800Hz$, where F_s is the sample rate of the input signal, $s_{inp}(n)$. For example, for a SWB 32 kHz sampled input, $L_c = 40$, and for a FB 48 kHz sampled input, $L_c = 60$ sub-bands.

The spectral flip and down mix module extracts the high band components from the input speech signal. The high band target signal contains the high frequency components of the input signal that are to be represented by the time domain bandwidth extension encoder. The frequency range of the high band depends on the coding bandwidth of the low band ACELP core. When the low band ACELP core codes up to a maximum bandwidth of 6.4 kHz, then the high band target signal, $S_{HB}(n)$, contains input signal components in the 6.4 -- 14.4 kHz band. When the low band ACELP core codes up to a maximum bandwidth of 8 kHz, then the high band target signal contains input signal components in the 8 -- 16 kHz band. In the high band target signal, $S_{HB}(n)$, the input signal components are arranged in a flipped and down-mixed format such that the 0 frequency value in the SWB high band target signal corresponds to the maximum frequency in the above mentioned bandwidth. The process of deriving the SWB high band target signal is shown pictorially in figure 45.
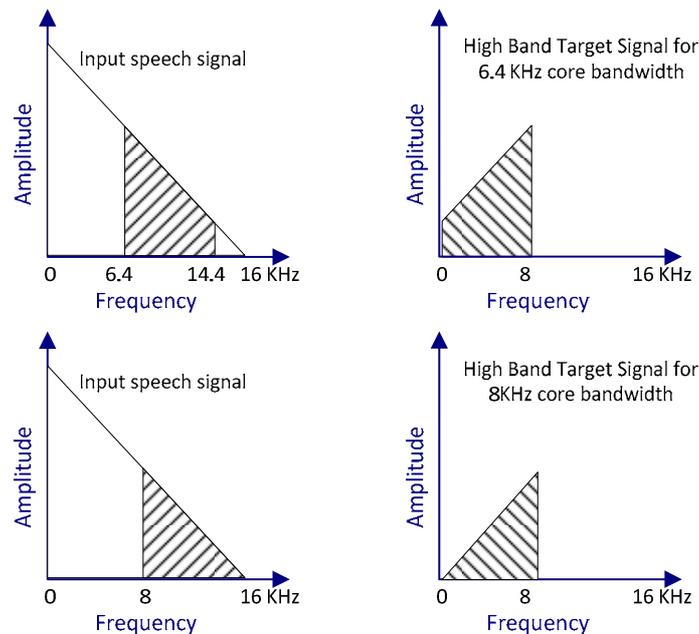
Figure 45: Generation of high band target signal for SWB TBE

Similarly, the WB high band target signal containing signal components from 6 to 8 kHz is generated as shown in figure 46. In particular, a simple flip-and-decimate-by-4 operation is performed to extract the 6 to 8 kHz high band from the 16 kHz sampled WB input signal. In case of fixed point operation, the input signal is scaled dynamically based on the spectral tilt of the input signal prior to performing the flip-and-decimate-by-4 operation. This adjustment of the scaling is done such that for signals with low energy in the high band (indicated by having a spectral tilt >= 0.95), no headroom is provided prior to decimation to avoid any loss of precision after decimation, and for signals with relatively higher energy in the high band (indicated by having a spectral tilt < 0.95), a headroom of 3 bits is provided prior to decimation to avoid any saturation in the decimation operation.
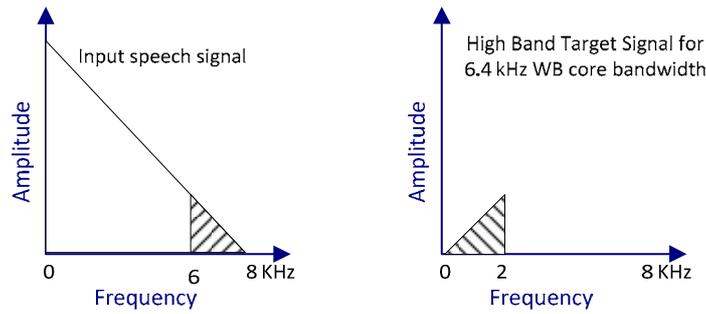
Figure 46: Generation of high band target signal for WB TBE

The process of deriving the SWB high band target signal using the complex low-delay filter bank (CLDFB) analysis is described below. The real, $X_{CR}(t,k)$, and imaginary, $X_{CI}(t,k)$, CLDFB coefficients are first flipped as follows:

$$\hat{X}_{CR}(t,k) = -(-1)^{tM} X_{CR}(t,M-k)$$
$$\hat{X}_{CI}(t,k) = (-1)^{tM} X_{CI}(t,M-k) \qquad for\ 0 \le t < L_C,\ 0 \le k < M-m \qquad (664)$$

The values m and M are dependent on the maximum bandwidth coded by the ACELP core as shown in table 52. The flipped coefficients, $\hat{X}_{CR}(t,k)$ and $\hat{X}_{CI}(t,k)$ are used to generate the high band target signal using the CLDFB synthesis as described in subclause 6.9.3.

Table 52: Maximum and minimum frequencies for spectral flipping and downmix

ACELP core		m	M	High band target signal band
Low band core sample rate	**Low band coded bandwidth**			
@12.8 kHz core	6.4 kHz	14	34	6.4-14.4 kHz
@ 16 kHz core	8 kHz	19	39	8-16 kHz
@12.8 kHz core	6.4 kHz	-	-	6-8 kHz (WB)

5.2.6.1.2 TBE LP analysis

TBE linear prediction analysis is performed on a 33.75ms high band signal that includes the current 20ms SHB target frame with 5ms of past samples and 8.75ms of look ahead samples. The SWB high band target signal $S_{HB}(n)$ is generated using the CLDFB synthesis filter bank described in subclause 6.9.3. Twenty milliseconds of the high band target signal is used to populate the shb_old_speech buffer from sample 220 to sample 539 as shown in figure 47 . Both the WB TBE and SWB TBE LP analysis follow similar steps as described below except that the buffer lengths in SWB TBE and WB TBE are different reflecting the effective upper band coding bandwidth. Certain steps that are relevant at low bit rates and only for WB TBE and not for SWB TBE LP analysis are specified where applicable.
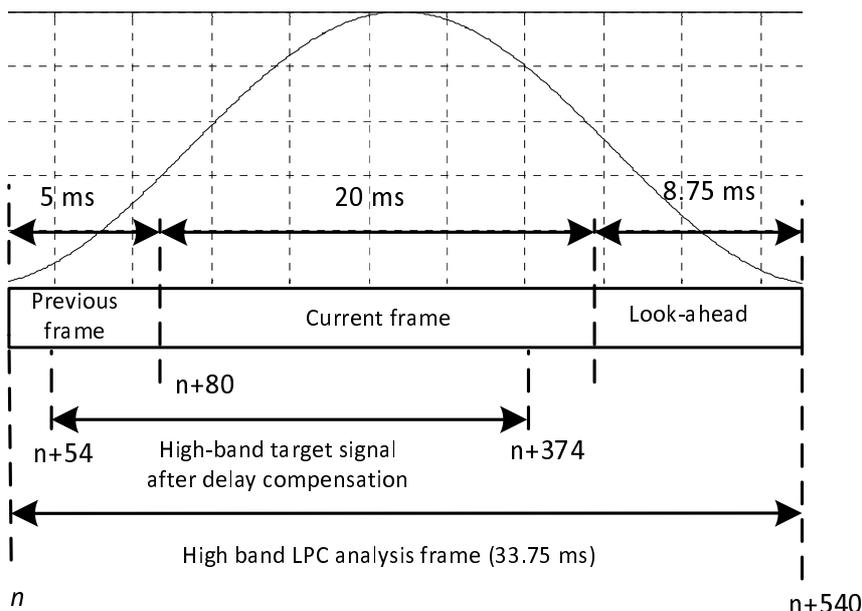
Figure 47: SWB Highband target signal buffers

The first 220 samples in the shb_old_speech buffer are filled from the memory shb_old_speech_mem. The shb_old_speech_mem is updated as

$$\text{shb\_old\_speech\_mem}[n] = \text{shb\_old\_speech}[n+320] \quad \textit{for } 0 \le n < 219 \tag{665}$$

For linear prediction analysis, a 540 sample LPC analysis frame is derived from the shb_old_speech buffer. A single analysis window $W_{SHB}(n), 0 \le n < 539$, is used to calculate 10 auto correlation coefficients. The window function is as shown in figure 48
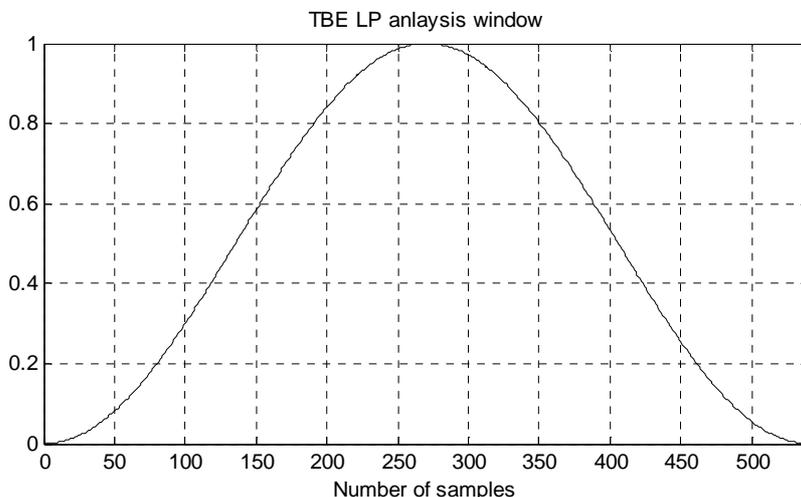
Figure 48: SWB TBE analysis window

The autocorrelation coefficients $r_C^{SHB}(k)$ is calculated according to

$$r_C^{SHB}(k) = \sum_{n=k}^{L-1} s_w^{SHB}(n) \times s_w^{SHB}(n-k), \ k = 0, \ldots, 10 \tag{666}$$

A bandwidth expansion is applied to the autocorrelation coefficients by multiplying the coefficients by the expansion function:

$$\hat{r}_C^{SHB}(k) = r_C^{SHB}(k) \times wac(k), \ k = 0, \ldots, 10 \tag{667}$$

The bandwidth expanded autocorrelation coefficients are used to obtain LP filter coefficients, $a_k^{SHB}, k = 1, \ldots, 11,$ by solving the following set of equations using the Levinson-Durbin algorithm.

$$\sum_{k=1}^{10} a_k^{SHB} \times \hat{r}_C^{SHB}(i-k) = -\hat{r}_C^{SHB}(i), \ i = 1, \ldots 11 \tag{668}$$

It should be noted that $a_1^{SHB} = 1$

5.2.6.1.3 Quantization of linear prediction parameters

First a spectral bandwidth expansion operation is performed on the LPC coefficients by multiplying a_k^{SHB} with bandwidth expansion weights bew_k^{SHB}

$$\hat{a}_k^{SHB} = a_k^{SHB} \times bew_k^{SHB}, \ k = 2, \ldots 11 \tag{669}$$

The bew_k^{SHB} coefficients are given in table 53:

Table 53: LPC bandwidth expansion coefficients

k									
2	3	4	5	6	7	8	9	10	11
0.975	0.950625	0.926859375	0.903687891	0.881095693	0.859068301	0.837591593f	0.816651804	0.796235509	0.776329621

The bandwidth expansion of LP coefficients is performed in WB TBE LP analysis and low bitrate SWB TBE analysis for bit rates below 13.2 kbps. The bandwidth expanded LP coefficients $\hat{a}_k^{SHB}, k = 1, \ldots, 11$ are converted to line spectral frequencies $\rho_k^{SHB}, k = 1, \ldots, 10$

The LSP weights w_k^{SHB} are calculated from the line spectral frequencies ρ_k^{SHB}

$$w_k^{SHB} = 250 \times fac \times \sqrt{\frac{(2/\pi)^2}{\delta_1(k) \times \delta_2(k)}} \tag{670}$$

And based on the spacing between adjacent LSF parameters, δ_1 and δ_2

$$\delta_1(k) = \begin{cases} \rho_k^{SHB} & k = 1 \\ \rho_k^{SHB} - \rho_{k-1}^{SHB} & otherwise \end{cases} \tag{671}$$

$$\delta_2(k) = \begin{cases} \rho_{k+1}^{SHB} - \rho_k^{SHB} & k \neq 10 \\ 0.5 - \rho_k^{SHB} & k = 10 \end{cases} \tag{672}$$

where $fac = 1.1 \ if \ k = 4 \ or \ 5$ and $fac = 1.0$, otherwise.

5.2.6.1.3.1 LSF quantization

WB TBE LSF quantization uses a single stage vector quantizer that utilizes 2 bits and 8 bits for LSF quantization, respectively, at 9.6 kbps and at 13.2 kbps. For low bit rate SWB LSF quantization at 9.6 kbps and primary frame encoding in channel aware mode at 13.2 kbps (see subclause 5.8.3.1), a simple 8-bit 10-dimensional single stage vector quantizer is used. In SWB TBE encoding, for bit rates at and above 13.2 kbps, the 10 dimensional LSF vector $\rho_k^{SHB}, \ k = 1, \ldots, 10$ is encoded in a SQ and extrapolation procedure. The low-frequency half (first five LSF coeffcents $\rho_k^{SHB}, \ k = 1, \ldots, 5$) of the LSF vector are scalar quantized with the following number of bits {4, 4, 3, 3, 3}.

That is the first two coefficents are quantized with four bits each, with the CB from table 54, while the remaining three coefficients are quantized with three bits each with reconstruction values in table 55.

Table 54: Four bit CB for LSF quantization

Index	Value
0000	0.01798018
0001	0.02359377
0010	0.02790103
0011	0.03181538
0100	0.03579450
0101	0.03974377
0110	0.04364637
0111	0.04754591
1000	0.05181858
1001	0.05624165
1010	0.06022101
1011	0.06419064
1100	0.06889389
1101	0.07539274
1110	0.08504436
1111	0.10014875

Table 55: Three bit CB for LSF quantization

Index	Value
000	0.02070812
001	0.02978384
010	0.03800822
011	0.04548685
100	0.05307309
101	0.06137543
110	0.07216742
111	0.09013262

The output of the SQ is five quantized coefficients $\hat{\rho}_k^{SHB}$, $k = 1,\dots,5$, which are also used as an input to extrapolation of the remaining five coefficients.

The high-frequency half (last five LSF coefficents ρ_k^{SHB}, $k = 6,\dots,10$) are calculated as weighted averaging of the quantized low-frequency part of the LSF vector and selected optimal grid points. First the already quantized coefficents are flipped arround a quantized mirroring frequency, which separates the low-frequency part from the high-frequency part of the LSF vector. Then the the fipped coefficients are adjusted by an optimal frequency grid codebook, which is obtained in a closed-loop search procedure.

The transmitted mirroring frequency $\hat{m}$ is obtained as quantized difference between the first extrapolated and last quantized coefficient, added to the last coded position

$$\hat{m} = Q\left(\rho_6^{SHB} - \hat{\rho}_5^{SHB}\right) + \hat{\rho}_5^{SHB} \tag{673}$$

The quantization is performed with two bit SQ with reconstruction points {0.01436178, 0.02111641, 0.02735687, 0.03712105}.

The quanized five low-frequency LSF coefficents $\hat{\rho}_k^{SHB}$, $k = 1,\dots,5$ are flipped around the mirroring frequency $\hat{m}$, to form a set of flipped coefficients $\breve{\rho}_k^{SHB}$

$$\breve{\rho}_k^{SHB} = 2\hat{m} - \hat{\rho}_{6-k}^{SHB}, \qquad k = 1,\ldots,5 \tag{674}$$

Then the flipped coefficients are further scaled according to

$$\tilde{\rho}_k^{SHB} = \begin{cases} \dfrac{\left(\breve{\rho}_k^{SHB} - \breve{\rho}_1^{SHB}\right)\left(0.5 - \hat{m}\right)}{\hat{m}} + \breve{\rho}_1^{SHB} & if \quad \hat{m} > 0.25 \\ \breve{\rho}_k^{SHB} & otherwise \end{cases} \tag{675}$$

The flip and scaled coefficients $\tilde{\rho}_k^{SHB}$ are adjusted with one of four grid vectors. The pre-stored set of four grid vectors $g_i, \quad i = 1,\ldots,4$ from table 56, is first re-scaled to fit into the interval between the last quantized LSF $\hat{\rho}_5^{SHB}$ and maximum grid point value 0.5

$$\tilde{g}_{i,k} = g_{i,k}\left(0.5 - \hat{\rho}_5^{SHB}\right) + \hat{\rho}_5^{SHB} \quad i = 1,\ldots,4 \quad k = 1,\ldots,5 \tag{676}$$

Table 56: CB with LSF grid points

grid 1	grid 2	grid 3	grid 4
0.15998503	0.15614473	0.14185823	0.15416561
0.31215086	0.30697672	0.26648724	0.27238427
0.47349756	0.45619822	0.39740108	0.39376780
0.66540429	0.62493785	0.55685745	0.59287916
0.84043882	0.77798001	0.74688616	0.86613986

Then smoothed coefficients $\overline{\rho}_{i,k}^{SHB}$ are obtained by weighed averaging of the re-scaled grid sets $\tilde{g}_{i,k}$ and re-scaled flipped coefficients $\tilde{\rho}_k^{SHB}$

$$\overline{\rho}_{i,k}^{SHB} = (1 - \lambda_k)\tilde{\rho}_k^{SHB} + \lambda_k \tilde{g}_{i,ki} \quad i = 1,\ldots,4 \quad k = 1,\ldots,5 \tag{677}$$

where λ_k are pre-defined weights $\lambda = \{0.2, \quad 0.35, \quad 0.5, \quad 0.75, \quad 0.8\}$

Different grid vectors g_i form different sets of smoothed coefficients $\overline{\rho}_{i,k}^{SHB}$. The optimal grid is selected as the one that produces smoothed coefficients $\overline{\rho}_{i,k}^{SHB}$ closest, in the mean-squared-error sense, to the actual high-frequency coefficients $\rho_k^{SHB}, \quad k = 1,\ldots,5$

$$i^{opt} = \arg\min_i \left\{ \sum_{k=1}^{5} \left(\overline{\rho}_{i,k}^{SHB} - \rho_k^{SHB}\right)^2 \right\} \tag{678}$$

The CB indices for the five low-frequency coefficients, mirroring frequency index, and the index of the optimal grid are transmitted to the decoder.

The quantized LSF parameters $\hat{\rho}_k^{SHB}$ are checked for inter-LSF spacing. First, the spacing between adjacent LSFs is determined

$$\delta_1(k) = \begin{cases} \rho_0^{SHB} & k = 0 \\ 0.5 - \rho_{k-1}^{SHB} & k = 10 \\ \rho_k^{SHB} - \rho_{k-1}^{SHB} & otherwise \end{cases} \tag{679}$$

If $\delta_1(k) < 0.0234952$, then the quantized LSFs are adjusted as follows

If $k = 0$, then
$$\hat{\rho}_k^{SHB} = \hat{\rho}_k^{SHB} - \delta_1(k) + 0.0234952 \,.$$
If $k = 10$, then

$$\hat{\rho}_{k-1}^{SHB} = \hat{\rho}_{k-1}^{SHB} + \delta_1(k) - 0.0234952 \ .$$

`Otherwise,`

$$\hat{\rho}_{k-1}^{SHB} = \hat{\rho}_{k-1}^{SHB} + 0.5 \times (\delta_1(k) - 0.0234952)$$

$$\hat{\rho}_{k}^{SHB} = \hat{\rho}_{k-1}^{SHB} - 0.5 \times (\delta_1(k) - 0.0234952)$$

5.2.6.1.4 Interpolation of LSF coefficients

The quantized LSF parameters $\hat{\rho}_k^{SHB}$ are converted into LSPs $\hat{\sigma}_k^{SHB}$

$$\hat{\sigma}_k^{SHB} = \cos\left(2\pi \times \hat{\rho}_k^{SHB}\right) \tag{680}$$

The interpolation between the LSPs of the current frame, $\hat{\sigma}_k^{SHB}$, and that of the previous frame $\hat{\sigma}prev_k^{SHB}$ is performed to a set of 4 interpolated LSPs, $\tilde{\sigma}_k^{SHB}(j) for\ j = 1,\ldots 4$ as follows:

$$\tilde{\sigma}_k^{SHB}(j) = IC_1(j) \times \hat{\sigma}prev_k^{SHB} + IC_2(j) \times \hat{\sigma}_k^{SHB}\ for\ k = 1,\ldots 10\ and\ j = 1,..,4 \tag{681}$$

The interpolated LSPs are then converted back to LSFs. This process gives 4 sets of interpolated LSFs $\tilde{\rho}_k^{SHB}(j) for\ j = 1,\ldots 4$. Further the LSFs are converted into LP coefficients $\tilde{a}_k^{SHB}(j), k = 1,\ldots,11\ and\ j = 1,..4$. Note $\tilde{a}_1^{SHB}(j) = 1$ for all j. The conversion from the interpolated LSFs to LP coefficients is done similar to the process employed in the core coding modules.

5.2.6.1.4.1 LSP Interpolation at 13.2 kbps and 16.4 kbps

LSP interpolation is employed to smooth the SWB TBE LSP parameters at 13.2 kbps and 16.4 kbps. When the bit rate of operation is 13.2 kbps or 16.4 kbps, the similarity of signal characteristics of the current frame and the previous frame are analysed to determine whether they meet the Preset Modification Conditions (PMCs) or not. The PMCs denote if the PMCs are met, then a first set of correction weights are determined from the Linear Spectral Frequency (LSF) differences of the current frame and the LSF differences of the previous frame. Otherwise a second set of correction weights are set with constant values that lie in the range 0.0 to 1.0. The LSPs are then interpolated using the appropriate set of correction weights. The PMCs are used to determine whether the signal characteristic of the current frame and the previous frame is close or not.

The PMCs are determined by first converting the linear prediction coefficients (LPCs) to reflection coefficients (RCs) as follows using a backwards Levinson Durbin recursion. For a given N-th order LPC vector $LPC_N = [1, a_{N1}^{SHB}, \ldots, a_{NN}^{SHB}]$ the Nth reflection coefficient value is derived using the equality $refl(N) = -a_{NN}^{SHB}$, it is then possible to calculate the lower order LPC vectors $LPC_{N-1}, \ldots, LPC_1$ using the following recursion.

$$refl(p) = a_{pp}^{SHB}{}_{pp}$$

$$F = 1 - refl(p)^2, \ p = N, N-1, \ldots, 2 \tag{682}$$

$$a_{p-1,m}^{SHB} = \frac{a_{p,m}^{SHB}}{F} - \frac{refl(p) a_{p,p-m}^{SHB}}{F}, 1 <= m < p$$

which yields the reflection coefficient vector $[refl(1), refl(2), \ldots refl(N)]$.

A spectral tilt parameter *tilt _ para* is then calculated from the first reflection coefficient *refl*(1) as follows:

$$tilt\_para = 6.6956 * (1.0 + refl(1)) * (1.0 + refl(1)) - 3.8714 * (1.0 + refl(1)) + 1.3041 \tag{683}$$

The PMCs are then determined from the spectral tilts of the current frame and the previous frame along with the coding types of the current and the previous frames. In the case that the current frame is a transition frame, the PMCs are by default not met.

In order to determine that the current frame is not a transition frame requires the following steps:

1) Determining whether there is a change from a fricative frame to a non-fricative frame. Specifically, if the tilt of the previous frame is greater than a tilt threshold value of 5.0 and the coder type of the frame is a transient, or the tilt of the previous frame is greater than a tilt threshold value of 5.0 and the tilt of the current frame is smaller than a tilt threshold value of 1.0 then the frame is a transition frame.

2) Determining whether there is a change from a non-fricative frame to a fricative frame. Specifically, if the tilt of the previous frame is smaller than a tilt threshold value of 3.0 and the coder type of the previous frame is equal to one of the four types of VOICED, GENERIC, TRANSITION or AUDIO, and the tilt of the current frame is greater than a tilt threshold value of 5.0 then again the frame is a transition frame

3) The current frame is not a transition frame if both 1) and 2) are not met, and then the PMCs are met.

When the PMCs are met, the first set of correction weights are defined as follows

$$
w_1[k] = \begin{cases} \rho_{k,new\_diff}^{SHB} / \rho_{k,old\_diff}^{SHB}, & \rho_{k,new\_diff}^{SHB} < \rho_{k,old\_diff}^{SHB} \\ \\ \rho_{k,old\_diff}^{SHB} / \rho_{k,new\_diff}^{SHB}, & \rho_{k,new\_diff}^{SHB} >= \rho_{k,old\_diff}^{SHB} \end{cases}, \quad k = 0,...,9 \qquad (684)
$$

where the $\rho_{k,new\_diff}^{SHB}, k = 0,...,9$ and $\rho_{k,old\_diff}^{SHB}, k = 0,...,9$ are defined as

$$
\rho_{k,new\_diff}^{SHB} = \begin{cases} 0.5, & k = 0 \\ \rho_{k,new}^{SHB} - \rho_{k-1,new}^{SHB}, & k = 1,...,8 \\ 0.5, & k = 9 \end{cases} \qquad (685)
$$

$$
\rho_{k,old\_diff}^{SHB} = \begin{cases} 0.5, & k = 0 \\ \rho_{k,old}^{SHB} - \rho_{k-1,old}^{SHB}, & k = 1,...,8 \\ 0.5, & k = 9 \end{cases} \qquad (686)
$$

where the $\rho_{k,new}^{SHB}, k = 0,...,9$ are the LSFs difference of the current frame and the $\rho_{k,old}^{SHB}, k = 0,...,9$ are the LSFs difference of the previous frame.

When the PMCs are not met, the second set of correction weights is used and these are set to 0.5 as follows:

$$
w_2(k) = 0.5 \qquad k = 0,...,9 \qquad (687)
$$

$$
w(k) = \begin{cases} w_1(k), & PMCs\ are met \\ w_{2(k)}, & otherwise \end{cases}, \quad k = 0,...,9 \qquad (688)
$$

The correction weights $w(k), k = 0,...,9$ are finally used in the interpolation between the LSPs of the current frame and the LSPs of the previous frame, it is described as follows:

$$
\hat{\sigma}_k^{SHB} = (1 - w(k)) * \hat{\sigma} prev_k^{SHB} + w(j) * \hat{\sigma} curr_k^{SHB}, \quad k = 0,...,9 \qquad (689)
$$

where $\hat{\sigma} prev_k^{SHB}, k = 0,...,9$ are the LSPs of the previous frame, $\hat{\sigma} curr_k^{SHB}, k = 0,...,9$ are the LSPs of the current frame, $\hat{\sigma}^{SHB}, k = 0,...,9$ are the interpolated LSPs. The interpolated LSPs are used to encode the current frame.

The interpolated LSPs are then converted back to LSFs，further the LSFs are converted into LP coefficients.

5.2.6.1.4.2 LSP Interpolation at 24.4 kbps and 32 kbps

The interpolation between the LSPs of the current frame, $\hat{\sigma}_k^{SHB}$, and that of the previous frame $\hat{\sigma} prev_k^{SHB}$ is performed to yield a set of 4 interpolated LSPs, $\tilde{\sigma}_k^{SHB}(j) for\ j = 1,...4$ as follows:

$$\tilde{\sigma}_k^{SHB}(j) = IC_1(j) \times \hat{\sigma}prev_k^{SHB} + IC_2(j) \times \hat{\sigma}_k^{SHB} \: for\, k = 1,\ldots 10\, and\: j = 1,..,4 \tag{690}$$

Table 57: LSP interpolation factors IC1 and IC2 over four sets

LSP set, j	IC1	IC2
Set 1	0.7	0.3
Set 2	0.4	0.6
Set 3	0.1	0.9
Set 4	0	1.0

The interpolated LSPs are then converted back to LSFs. This process gives 4 sets of interpolated LSFs $\tilde{\rho}_k^{SHB}(j) for\: j = 1,\ldots 4$. Further the LSFs are converted into LP coefficients $\tilde{a}_k^{SHB}(j), k = 1,\ldots,11 and\: j = 1,..4$. Note $\tilde{a}_1^{SHB}(j) = 1$ for all *j*.

5.2.6.1.5 Target and residual energy calculation and quantization

At 24.4 kb/s and 32 kb/s, energy parameters from the high band target frame and an LPC residual calculated from the target signal and the interpolated LP coefficients, $\tilde{a}_1^{SHB}(j)$, are calculated and transmitted to the decoder.

For calculation of the energy parameters, the signal in the high band target frame (see figure 47), denoted as $S_{HB}^{Tar}(n)$ is used. The 4 energy parameters are calculated according to:

$$\vartheta(j) = 0.003125 \times \sum_{n=(j-1)\times 80}^{j \times 80-1} \left[S_{HB}^{Tar}(n) \right]^2 \tag{691}$$

The sum of energy parameters, $\sum_{j=0}^{3} \vartheta(j)$, is quantized using 6 bits in SWB TBE at 24.4 kbps and 32 kbps. A residual signal $r_{HB}^{Tar}(n)$ is calculated from $S_{HB}^{Tar}(n)$ using the interpolated LP coefficients $\tilde{a}_1^{SHB}(j)$. For j=1, …4 and n=0,…79,

$$r_{HB}^{Tar}(n + (j-1) \times 80) = \sum_{n=(j-1)\times 80}^{j \times 80-1} \left(S_{HB}^{Tar}(n) - \left[\sum_{k=1}^{10} \tilde{a}_k^{SHB}(j) \times S_{HB}^{Tar}(n-k) \right] \right) \tag{692}$$

The residual signal is then used to calculate the residual energy parameters, $\vartheta_{res}(j), j = 1,\ldots 5$

$$\vartheta_{res}(j) = \sum_{n=(j-1)\times 64}^{j \times 64-1} \left[r_{HB}^{Tar}(n) \right]^2 \tag{693}$$

The residual energy parameters are then normalized. First the maximum residual energy parameter is calculated $\vartheta max_{res} = \max(\vartheta_{res}(j))$. Then $\vartheta_{res}(j)$ is normalized to get $\hat{\vartheta}_{res}(j)$ as per

$$\hat{\vartheta}_{res}(j) = \sqrt{\frac{\vartheta_{res}(j)}{\vartheta max_{res}}} \tag{694}$$

Each $\hat{\vartheta}_{res}(j)$ is then scalar quantized using a 3 bit uniform quantizer with the lowest quantization point as 0.125 and uniform quantization steps of 0.125.

5.2.6.1.6 Generation of the upsampled version of the lowband excitation

An upsampled version of the low band excitation signal is derived from the ACELP core as show in figure 49 below.
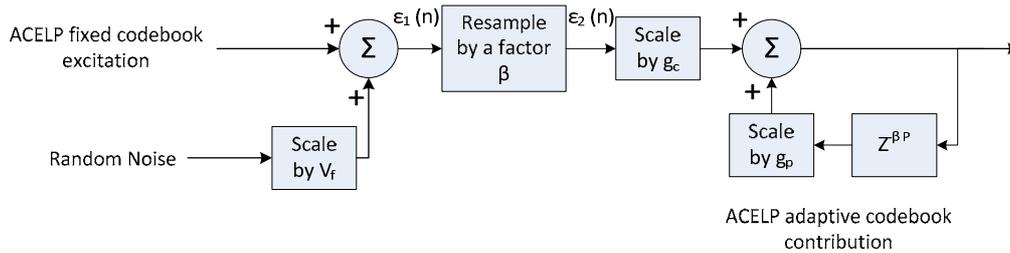
Figure 49: Generating the upsampled version of the lowband excitation

For each ACELP core coding subframe, i, a random noise scaled by a factor voice factor, Vf_i is first added to the fixed codebook excitation that is generated by the ACELP core encoder. The voice factor is determined using the subframe maximum normalized correlation parameter, β_i, that is derived during the ACELP encoding. First the β_i factors are combined to generate Vf_i.

$$Vf_i = 0.34 + 0.5 \times \beta_i + (0.5 - 0.34) \times \beta_i^2 \tag{695}$$

Vf_i calculated above is limited to a maximum of 1 and a minimum of 0.

$\varepsilon_1(n) = code(n) + Vf_i \times random(n) \, for \, 0 \leq n \leq 256$ if the ACELP core encodes a maximum of 6.4 KHz or $0 \leq n \leq 320$ if the ACELP core encodes a maximum bandwidth of 8 KHz.

The resampled output is scaled by the ACELP fixed codebook gain and added to a delayed version of itself.

$$\varepsilon(n) = g_c \times \varepsilon_2(n) + g_p \times \varepsilon(n - \beta P) \tag{696}$$

where g_c is the subframe ACELP fixed codebook gain, g_p is the subframe ACELP adaptive codebook gain and P is the open loop pitch lag.

5.2.6.1.7   Non-Linear Excitation Generation

The excitation signal $\varepsilon(n)$ is processed through a non-linear function in order to extend the pitch harmonics in the low band signal into the high band. The non-linear processing is applied to a frame of $\varepsilon(n)$ in two stages; the first stage works on the first half subframe (160 samples) of $\varepsilon(n)$ and the second stage works on the second half subframe. The non-linear processing steps for the two stages are described below. In the first stage $n_1 = 0, n_2 = 160$, and in the second stage, $n_1 = 160, n_2 = 320$.

First, the maximum amplitude sample ε_{max} and its location relative to the first sample in the stage i_{max} are determined.

$$\varepsilon_{max} = \max(|\varepsilon(n)|) \, and \, i_{max} = argmax(|\varepsilon(n)|) - n_1 \, for \, n_1 \leq n < n_2 \tag{697}$$

Based on the value of ε_{max}, the scale factor sf is determined.

$$sf = \begin{cases} \dfrac{0.67}{\varepsilon_{max}} & if \, \varepsilon_{max} > 1 \\ 0.67 & if \, \varepsilon_{max} \leq 1 \end{cases} \tag{698}$$

The scale factor sf and the previous scale factor parameter from the memory sf_{prev} are then used to determine the parameter scale step ss.

$$ss = \begin{cases} 1 & if \, sf_{prev} \leq 0 \\ e^{\left(\frac{1}{i_{max}}\right) * \log\left(\frac{sf}{sf_{prev}}\right)} & otherwise \end{cases} \tag{699}$$

If $sf_{prev} \leq 0$, then $sf_{prev} = ss$

The output of the non-linear processing $\varepsilon_{NL}(n)$ is derived as per

$$\varepsilon_{NL}(n) = \begin{cases} \varepsilon^2(n) \times sf_{prev} & if\ \varepsilon(n) \geq 0 \\ -\varepsilon^2(n) \times sf_{prev} & if\ \varepsilon(n) < 0 \end{cases} \quad for\ n_1 \leq n < n_2 \tag{700}$$

If the current frame is VOICED frame and sum of voice factors over the subframes is less than a threshold, ths, then the sign reversal when $\varepsilon(n) < 0$ is not performed. The threshold, $ths = 0.70$ when there are five subframes per frame and $ths = 0.78$ when there are four subframes per frame.

The previous scale factor parameter is updated recursively for all $j < i_{max}$ according to

```
for(j=n₁; j< n₂; j++)
    if(j<i_max)
```
$$sf_{prev} = sf_{prev} \times ss$$
```
    end
end
```

5.2.6.1.8 Spectral flip of non-linear excitation in time domain

The non-linear excitation $\varepsilon_{NL}(n)$ is spectrally flipped so that the high band portion of the excitation is modulated down to the low frequency region. This spectral flip is accomplished in time domain

$$\varepsilon_{flipped}(n) = (-1)^n \varepsilon_{NL}(n), for\ n = 0,\ldots 319 \tag{701}$$

5.2.6.1.9 Down-sample using all-pass filters

$\varepsilon_{flipped}(n)$ is then decimated using a pair of all pass filters to obtain an 8 kHz bandwidth (16 kHz sampled) excitation signal $\varepsilon_{16k}(n)$. This is done by filtering the even samples of $\varepsilon_{flipped}(n)$ by an all pass filter whose transfer function is given by

$$H_{d,1} = \left(\frac{a_{0,1} + z^{-1}}{1 + a_{0,1}z^{-1}}\right)\left(\frac{a_{1,1} + z^{-1}}{1 + a_{1,1}z^{-1}}\right)\left(\frac{a_{2,1} + z^{-1}}{1 + a_{1,1}z^{-1}}\right) \tag{702}$$

And the odd samples of $\varepsilon_{flipped}(n)$ by an all pass filter whose transfer function is given by

$$H_{d,2} = \left(\frac{a_{0,2} + z^{-1}}{1 + a_{0,2}z^{-1}}\right)\left(\frac{a_{1,2} + z^{-1}}{1 + a_{1,2}z^{-1}}\right)\left(\frac{a_{2,2} + z^{-1}}{1 + a_{1,2}z^{-1}}\right) \tag{703}$$

The 16 kHz sampled excitation signal $\varepsilon_{16k}(n)\ for\ n = 0,\ldots 159$ are obtained by averaging the outputs of the above filter.

These filter coefficients are specified in below.

Table 58: All-pass filter coefficients for decimation by a factor of 2

	All pass coefficients
$a_{0,1}$	0.06056541924291
$a_{1,1}$	0.42943401549235
$a_{2,1}$	0.80873048306552
$a_{0,2}$	0.22063024829630
$a_{1,2}$	0.63593943961708
$a_{2,2}$	0.94151583095682

5.2.6.1.10 Adaptive spectral whitening

Due to the nonlinear processing applied to obtain the excitation signal $\varepsilon_{16k}(n)$, the spectrum of this excitation is no longer flat. In order to flatten the spectrum of the excitation signal $\varepsilon_{16k}(n)$, 4th order linear prediction coefficients are estimated from $\varepsilon_{16k}(n)$. The spectrum of $\varepsilon_{16k}(n)$ is then flattened by inverse filtering $\varepsilon_{16k}(n)$ using the linear prediction filter.

The first step in the adaptive whitening process is to estimate the autocorrelation of the excitation signal

$$r_{exc}^{SHB}(k) = \sum_{n=k}^{L-1} \varepsilon_{16k}(n) \times \varepsilon_{16k}(n-k), \, k = 0,\ldots,4 \, and \, n = 0,\ldots 159 \tag{704}$$

A bandwidth expansion is applied to the autocorrelation coefficients by multiplying the coefficients by the expansion function:

$$\hat{r}_{exc}^{SHB}(k) = r_{exc}^{SHB}(k) \times wac(k), \, k = 0,\ldots,4 \tag{705}$$

The bandwidth expanded autocorrelation coefficients are used to obtain LP filter coefficients, $a_1^{WHT}, k = 1,\ldots,5$ by solving the following set of equations using the Levinson-Durbin algorithm as described in section.

$$\sum_{k=1}^{4} a_k^{WHT} \times \hat{r}_{exc}^{SHB}(i-k) = -\hat{r}_{exc}^{SHB}(i), \, i = 1,\ldots 4 \tag{706}$$

It must be noted that $a_1^{WHT} = 1$.

The whitened excitation signal $\varepsilon_{WHT}(n)$ is obtained from $\varepsilon_{16k}(n)$ by inverse filtering

$$\varepsilon_{WHT}(n) = \varepsilon_{16k}(n) - \sum_{k=1}^{4} a_k^{WHT} \varepsilon_{16k}(n-k) \tag{707}$$

4 samples of $\varepsilon_{16k}(n)$ from the previous frame are used as memory for the above filtering operation.

For bit rates 24.4 kb/s and 32 kb/s, the whitened excitation is further modulated (in a two-stage gain shape modulation) by the normalized residual energy parameter $\hat{\vartheta}_{res}(j)$. In other words, for bitrates 24.4 kb/s and 32 kb/s,

$$\varepsilon_{WHT}((j-1) \times 64 + n) = \varepsilon_{WHT}((j-1) \times 64 + n) \times \hat{\vartheta}_{res}(j)$$
$$for \, (j-1) \times 64 \le n < j \times 64 \, and \, j = 1,\ldots 5 \tag{708}$$

5.2.6.1.11 Envelope modulated noise mixing

To the whitened excitation, a random noise vector whose amplitude has been modulated by the envelope of the whitened excitation is mixed using a mixing ratio that is dependent on the extent of voicing in the low band.

First, $eabs(n) = |\varepsilon_{WHT}(n)|$ is calculated and then the envelope of the envelope of the whitened excitation signal $\varepsilon_{WHT}(n)$ is calculated by smoothing $eabs(n)$

$$envNE(n) = \alpha_1 \times eabs(n) + \alpha_2 \times envNE(n-1) \tag{709}$$

In SWB mode, the factors α_1 and α_2 are calculated using the voicing factors, Vf_i for subframes $i = 1,\ldots 4$, which calculated by parameters which determined from the low band ACELP encoder. The voicing factors denote voicing extend of the high band signal since the fine signal structure in the higher bands are closely related to that in the lower band. The average of the 4 voicing factors, $Vf = 0.25 \times \sum_{i=1,..4} Vf_i$, is calculated and modified as

$Vf = 1.09875 - 0.49875 \times Vf$. This is then confined to values between 0.6 and 0.999. Then α_1 and α_2 are estimated as

$$\alpha_1 = 1 - Vf \tag{710}$$

$$\alpha_2 = -Vf \tag{711}$$

However, for bit rates 16.4 kb/s and 24.4 kb/s and if TBE was not used in the previous frame, α_1 and α_2 are set to

$$\alpha_1 = 0.2 \tag{712}$$

$$\alpha_2 = -0.8 \tag{713}$$

and for $n = 0$, $envNE(n-1)$ is substituted by an approximated value $envNE_{prev,approx}$ as

$$envNE_{prev,approx} = \frac{1}{20} \sum_{n=0}^{19} eabs(n) \tag{714}$$

In WB mode, the factors α_1 and α_2 are initialized to $\alpha_1 = 0.05$ and $\alpha_2 = -0.96$. However, if the bitrate is 9.6kb/s, they are reset to $\alpha_1 = 0.2$ and $\alpha_2 = -0.8$ if the low band coder type is voiced or $Vf > 0.35$, or to $\alpha_1 = 0.01$ and $\alpha_2 = -0.99$ if the low band coder type is unvoiced or $Vf < 0.2$. In SWB mode, for bit rates 24.4 kbps and 32 kbps, the mix factors are estimated based on both the low band voice factor, Vf_i, and the high band closed-loop estimation, where

$$Vf_i = Vf_i * HB\_adjust\_fac, \; i = 1,2,3,4,5$$

The HB_adjust_fac mix factor is estimated based on the high band residual signal, $r_{HB}^{Tar}(n)$, transformed low band excitation, $\varepsilon_{wht}(n)$, and the modulated white noise excitation $rn_{wht}(n)$.

A vector of random numbers, $rnd(n)$ of length 160 is then modulated by $envNE(n)$ to generate $rn_{wht}(n)$ as

$$rn_{wht}(n) = rnd(n) \times envNE(n) \tag{715}$$

The whitened excitation is then de-emphasized with $\mu = 0.68$ which is the pre-emphasised effect since the used spectrum is flipped.

$$rn_{wht}(n) = rn_{wht}(n) + \mu \times rn_{wht}(n-1) \tag{716}$$

If the lowband coder type is unvoiced, the excitation $rn_{wht}(n)$ is first rescaled to match the energy level of the whitened excitation $\varepsilon_{WHT}(n)$

$$rn_{wht}(n) = scale \times rn_{wht}(n) \tag{717}$$

where

$$scale = \sqrt{\frac{\sum_{i=0}^{319} (\varepsilon_{wht}(n))^2}{\sum_{i=0}^{319} (rn_{wht}(n))^2}} \tag{718}$$

And then pre-emphasised with $\mu = 0.68$ to generate the final excitation which is the de-emphasised effect since the used spectrum is flipped.

$$\varepsilon_1(n) = rn_{wht}(n) - \mu \times rn_{wht}(n-1) \tag{719}$$

If the lowband coder type was not un-voiced, the final excitation is calculated as

$$\varepsilon_1(n) = \alpha_1 \times \varepsilon_{wht}(n) + \alpha_2 \times rn_{wht}(n) \tag{720}$$

For bit rates less than 24.4 kb/s, the mixing parameters α_1 and α_2 are estimated for other low band coder types as,

$$\alpha_1 = (Vf_i)^{1/4} \tag{721}$$

$$\alpha_2 = \sqrt{\frac{\left(\sum_{i=0}^{319}(\varepsilon_{wht}(n))^2\right) \times \left(1 - \sqrt{Vf_i}\right)}{\sum_{i=0}^{319}(rn_{wht}(n))^2}} \tag{722}$$

For bit rates 24.4 kb/s and 32 kb/s, the mixing parameters α_1, α_2 are estimated for other low band coder types as follows:

$$\alpha_1 = \sqrt{(Vf_i) \times (1.0 - 0.15 \times fac_{formant})} \tag{723}$$

$$\alpha_2 = \sqrt{\frac{\sum_{i=0}^{319}(\varepsilon_{wht}(n))^2 \times (1.0 - (Vf_i) \times (1.0 - 0.15 \times fac_{formant}))}{\sum_{i=0}^{319}(rn_{wht}(n))^2}} \tag{723a}$$

where the parameter $fac_{formant}$ is defined in equation (731).

$\varepsilon_1(n)$ is then de-emphasised to generate the final excitation.

5.2.6.1.12 Spectral shaping of the noise added excitation

The excitation signal $\varepsilon_1(n)$ is then put through the high band LPC synthesis filter that is derived from the quantized LPC coefficients (see subclause 5.2.4.1.3).

For bitrates below 24.4 kb/s, a single LPC synthesis filter $\tilde{a}_k^{SHB}, k = 1, \ldots 11$ is used and the shaped excitation signal $s\varepsilon(n)$ is generated as

$$s\varepsilon(n) = \varepsilon_1(n) - \sum_{k=1}^{11} \tilde{a}_k^{SHB} \times s\varepsilon(n-k) \tag{724}$$

For bitrates at and above 24.4 kb/s the LPC synthesis filter is applied to the excitation signal $\varepsilon_1(n)$ in four subframes based on

$$s\varepsilon(n + (j-1) \times 80) = \varepsilon_1(n + (j-1) \times 80) - \sum_{k=1}^{11} \tilde{a}_k^{SHB}(j) \times s\varepsilon(n + (j-1) \times 80 - k) \tag{725}$$
$$for\ j = 1, \ldots, 4$$

In particular, for bit rates at and above 24.4 kbps, first a memory-less synthesis is performed (with past LP filter memories set to zero) and the energy of the synthesized high band is matched to that of the original signal. In the subsequent step, the scaled or energy compensated excitation signal as shown below, $\varepsilon_1(n)$, is used to perform synthsesis in the second step.

$$\varepsilon_1(n) = \varepsilon_1(n) \left(\frac{\sum_{j=0}^{3} \vartheta(j)}{memoryless\ filtered\ shb\ synthesis\ energy} \right)$$

5.2.6.1.13 Post processing of the shaped excitation

The shaped excitation $s\varepsilon(n)$ is the synthesized high band signal which is generated by passing the excitation signal $\varepsilon_1(n)$ through the LPC synthesis filter. The excitation signal $\varepsilon_1(n)$ is determined by the low band model parameters

and the coefficients of the LPC synthesis filter are determined by the high band model parameters. A short-term post-filter is applied to the synthesized high band signal to obtain a short-term post-filtered signal. Comparing with the shape of the spectral envelope of the synthesized high frequency band signal, the shape of the short-term post-filtered signal is closer to the shape of the spectral envelope of the high-frequency band signal. The short-term post-filter is a pole-zero filter, the coefficients of the pole-zero filter are set by the set of high band model parameters. It is described as follows:

$$H_f(z) = \frac{H_n(Z)}{H_d(Z)} = \frac{1 - \alpha_1\beta z^{-1} - \alpha_2\beta^2 z^{-2} - \ldots - \alpha_M\beta^M z^{-M}}{1 - \alpha_1\gamma z^{-1} - \alpha_2\gamma^2 z^{-2} - \ldots - \alpha_M\gamma^M z^{-M}}, \qquad 0 < \beta < \gamma < 1 \quad M = 11 \qquad (726)$$

$$\hat{H}_f(z) = H_f(z)/g_f \qquad (727)$$

$H_f(z)$ is derived from the quantized LPC coefficients (see subclause 5.2.6.1.3) with the factors β and γ controlling the degree of the short-term post-filtering. The factors β and γ are calculated according to:

$$\begin{cases} \beta = 0.65 - 0.15 * fac_{formant} \\ \gamma = 0.65 + 0.15 * fac_{formant} \end{cases} \qquad (728)$$

where $fac_{formant}$ is parameter that jointly controls envelope shape and excitation noisiness. It is based on the spectral tilt, determined by the LPC coefficient a_2^{SHB}:

$$fac_{formant} = 0.5 * \left| a_2^{SHB} \right| + 0.5 * fac_{formant}^{past} \qquad (729)$$

where $fac_{formant}^{past}$ is the past value of $fac_{formant}$.

$$fac_{formant} = \max(\min(fac_{formant} - 1, 1.0), 0.0) \qquad (730)$$

$$fac_{formant} = 1.0 - 0.5 * fac_{formant} \qquad (731)$$

The gain term g_f is calculated from the truncated impulse response $h_f(n)$ of the filter $H_f(z)$ and is given by:

$$g_f = \sum_{n=0}^{19} |h_f(n)| \qquad (732)$$

The shaped excitation $s\varepsilon(n)$ is divided into four subframes $s\varepsilon(n + (j-1)*80), j = 1,2,3,4$, and each subframe $s\varepsilon(n + (j-1)*80), j = 1,2,3,4$ is filtered through $H_n(z)/g_f$, and then filtered with the synthesis filter $1/H_d(z)$ to produce the short-term post-filtered signal $s\varepsilon r(n + (j-1)*80), j = 1,2,3,4$.

After filtering the synthesized high band signal using the pole-zero filter, filter $H_t(z)$ then compensates for the tilt in the pole-zero filter $H_f(z)$ and is given by:

$$H_t(z) = 1 - \mu z^{-1} \qquad (733)$$

$$\hat{H}_t(z) = H_t(z)/g_t \qquad (734)$$

where μ is set to default constant value or adaptively calculated according to the high band coding parameters and the synthesized high band signal. The calculation process is as follows: $\mu = \gamma_t k_1^{'}$ is a tilt factor, with $k_1^{'}$ being the first reflection coefficient calculated from $h_f(n)$ by:

$$k_1' = -\frac{r_h(1)}{r_h(0)} \quad \text{and} \quad r_h(i) = \sum_{j=0}^{19-i} h_f(j)h_f(j+i) \tag{735}$$

A gain term $g_t = 1 - |\gamma_t k_1'|$ is applied to compensate for the decreasing effect of g_f in $\hat{H}_f(z)$. It has been shown that the product filter $\hat{H}_f(z)\hat{H}_t(z)$ has a gain close to unity. γ_t is set according to the sign of k_1'. If k_1' is positive, $\gamma_t = 0.65$, otherwise, $\gamma_t = 0.85$.

The short-term post-filtered signal $s\varepsilon r(n + (j-1)*80), j = 1,2,3,4$ is passed through the tilt compensation filter $\hat{H}_t(z)$ resulting in the post-filtered speech signal $s\varepsilon r'(n + (j-1)*80), j = 1,2,3,4$

Adaptive Gain Control (AGC) is applied to compensate for any gain difference between the synthesized speech signal $s\varepsilon(n + (j-1)*80), j = 1,2,3,4$ and the post-filtered signal $s\varepsilon r'(n + (j-1)*80), j = 1,2,3,4$. The gain scaling factor γ_{sc} for each subframe is calculated by:

$$\gamma_{sc}(j) = \sqrt{\frac{\sum_{n=0}^{79} (s\varepsilon(i*80+n))^2}{\sum_{n=0}^{79} (s\varepsilon r'(n))^2}} \tag{736}$$

and the post processed shaped excitation $s\varepsilon'(n + (j-1)*80), j = 1,2,3,4$ is given by:

$$s\hat{\varepsilon}(n + (j-1)*80) = \beta_{sc}(n)s\varepsilon r'(n + (j-1)*80), \quad n = 0,...,79; j = 1,2,3,4 \tag{737}$$

where $\beta_{sc}(n)$ is updated in sample-by-sample basis and given by:

$$\beta_{sc}(n) = \alpha\beta_{sc}(n-1) + (1-\alpha)\gamma_{sc} \tag{738}$$

and where α is an AGC factor with value of 0.85.

In order to smooth the evolution of the post-processed spectrally shaped highband excitation signal across frame boundaries, the look-ahead and the overlap samples are scaled based on the ratio of the current frame's energy in the overlap region and the previous frame's energy in the overlap region. The scale factor computation is performed as shown in equation (1579) in subclause 6.1.5.1.12.

The tenth-order LPC synthesis performed as described according to subclause 5.2.6.1.12 uses a memory of ten samples, thus there is at least an energy propagation over ten samples from the previous frame into the current frame. When calculating the energy scaling to be applied to the current frame, the first 10 samples of the current frame are considered as a part of previous frame energy. If the voicing factor Vf_0 is greater than 0.75, the numerator in equation (1579) is attenuated by 0.25. The spectrally shaped high band signal is then modified by the scale factor as shown in equation (1580) in Clause 6.1.5.1.12.

5.2.6.1.14 Estimation of temporal gain shape parameters

There are different initialization estimation of temporal gain shape for the WB mode and the SWB mode.

5.2.6.1.14.1 Initialization estimation of temporal gain shape for WB mode

The subframe energies in the target signal and the shaped excitation signal are calculated as follows:

Asymmetric windows are applied to the first and the eighth subframes,

$$\varphi_{Tar(1)} = \sum_{n=0}^{4} (S_{HB}^{Tar}(n)*swin(2*n+2))^2 + \sum_{n=5}^{9} (S_{HB}^{Tar}(n)*1)^2 + \sum_{n=10}^{19} (S_{HB}^{Tar}(n)*swin(2*10-n-1))^2$$

(739)

$$\varphi_{ShE(1)} = \sum_{n=0}^{4} (s\hat{\varepsilon}(n)*swin(2*n+2))^2 + \sum_{n=5}^{9} (s\hat{\varepsilon}(n)*1)^2 + \sum_{n=10}^{19} (s\hat{\varepsilon}(n)*swin(2*10-n-1))^2$$

$$\varphi_{Tar(8)} = \sum_{n=0}^{9} (S_{HB}^{Tar}(7*10+n)*swin(n+1))^2 + \sum_{n=10}^{14} (S_{HB}^{Tar}(7*10+n)*swin(3*10-2*n-2))^2$$

(740)

$$\varphi_{ShE(8)} = \sum_{n=0}^{9} (s\hat{\varepsilon}(7*10+n)*swin(n+1))^2 + \sum_{n=10}^{14} (s\hat{\varepsilon}(7*10+n)*swin(3*10-2*n-2))^2$$

And symmetric windows are applied to the subframes from the second to the seventh.

$$\varphi_{Tar(j)} = \sum_{n=0}^{9} (S_{HB}^{Tar}(j*10+n)*swin(n+1))^2 + \sum_{n=10}^{19} (S_{HB}^{Tar}(j*10+n)*swin(2*10-n-1))^2, \quad j=2,...,7$$

(741)

(742)

$$\varphi_{ShE(j)} = \sum_{n=0}^{9} (s\hat{\varepsilon}(j*10+n)*swin(n+1))^2 + \sum_{n=10}^{19} (s\hat{\varepsilon}(j*10+n)*swin(2*10-n-1))^2, \quad j=2,...,7$$

Four high band gain shapes are then calculated by combining pairs of subframe energies as follows

$$gs(j) = \sqrt{\frac{\varphi_{Tar(2*(j-1)+1)} + \varphi_{Tar(2*(j-1)+2)}}{\varphi_{ShE(2*(j-1)+1)} + \varphi_{ShE(2*(j-1)+2)}}}, \quad j=1,2,3,4$$

(743)

The asymmetric windows are determined by the number of look head samples and the number of subframes. The asymmetric window and the symmetric windows are described as follows, the number of look head samples is 5, and the length of the non-symmetrical window and the symmetrical window are both 20.
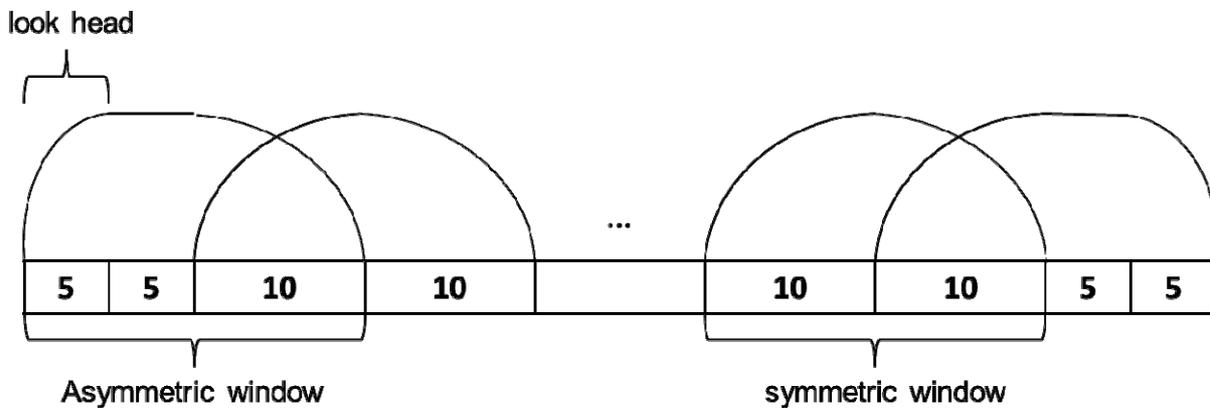
Figure 50: Asymmetric window and symmetric window

The variable *swin(n)* for $n = 0,...,10$ is tabulated in table 59 below:

Table 59: Window for highband gain shape calculation

Index	Value
0	0.0
1	0.15643448
2	0.30901700
3	0.45399052
4	0.58778524
5	0.70710677
6	0.80901700
7	0.89100653
8	0.95105654
9	0.98768836
10	1.0

5.2.6.1.14.2   Initialization estimation of temporal gain shape for SWB mode

The high band target frame (see subclause 5.2.6.1.1), $S_{HB}^{Tar}(n)$ and the shaped excitation $s\varepsilon(n)$ are used to calculate 4 temporal gain shape parameters, $gs(j)$, for $j = 1, \ldots 4$. The gain shape parameters are calculated using an overlap of 20 samples from the previous frame to avoid transition artifacts during the reconstruction at the decoder.

$$gs(j) = \sqrt{\frac{\varphi_{Tar}(j)}{\varphi_{ShE}(j)}} \; for \; j = 1, \ldots 4 \tag{744}$$

where the subframe energies in the target high band signal and the shaped excitation signal are calculated as

$$\varphi_{Tar}(j) = \sum_{n=0}^{99} \left(S_{HB}^{Tar}(n + (j-1) \times 80 - 20) \right)^2 \times swin(n), for \; j = 1, \ldots 4 \tag{745}$$

and

$$\varphi_{ShE}(j) = \sum_{n=0}^{99} \left(s\varepsilon(n + (j-1) \times 80 - 20) \right)^2 \times swin(n), for \; j = 1, \ldots 4 \tag{746}$$

The window function $swin(n)$ is a window signal is given by

$$swin(n) = \begin{cases} win(n) & n = 0 \ldots 20 \\ 1.0 & n = 21..79 \\ win(100 - n) & n = 80..99 \end{cases} \tag{747}$$

The variable $win(n)$ for $n = 0, \ldots 20$ is tabulated in table 60 below.

Table 60: Window for highband gain shape calculation

Index	Value
0	0.0
1	0.006156
2	0.024472
3	0.054497
4	0.095492
5	0.146447
6	0.206107
7	0.273005
8	0.345492
9	0.421783
10	0.5
11	0.578217
12	0.654508
13	0.726995
14	0.793893
15	0.853553
16	0.904508
17	0.945503
18	0.975528
19	0.993844
20	1.0

5.2.6.1.14.3 Additional processing of temporal gain shape

Additionally, the gain shape values are normalized such that

$$gs(j) = \frac{gs(j)}{\sqrt{\sum_{k=1}^{4} gs(j)}} \tag{748}$$

The 4th subframe gain shape value from the last subframe gs_{prev} is used to smooth the Gain Shape parameter evolution. A variable fb is calculated as

$$fb = 0.4 \bigg/ \left[1 + 0.5 \times log\left(\frac{gs(1)}{gs_{prev}}\right) + \sum_{j=2}^{4} log\left(\frac{gs(j)}{gs(j-1)}\right) \right] \tag{749}$$

If the sum of the voicing factors $\sum_{i=0}^{4} Vf_i > 0.8$, then the gain shape parameters are smoothed as follows:

$$gs(1) = fb \times gs_{prev} + (1 - fb) \times gs(1) \tag{750}$$

$$gs(j) = fb \times gs(j-1) + (1 - fb) \times gs(j), \text{ for } j = 2,\ldots4 \tag{751}$$

The gain shape parameters vector corresponding to a given frame are transformed into the log domain and vector quantized. The quantized gain shape parameters are denoted $gs_q(j)$ for $j = 1,\ldots4$

5.2.6.1.15 Estimation of frame gain parameters

In addition to the gain shape parameter, an overall frame gain parameter GF is calculated. First the spectrally shaped excitation is scaled by the gain shape parameters.

$$s\varepsilon_{scaled}(n-20) = \sum_{j=1}^{4} gs_q(j) \times swin_1(n-(j-1) \times 80) \times s\varepsilon(n-20) \text{ for } n = 0, \dots 359 \tag{752}$$

Samples with negative indices are obtained from the previous frame and

$$swin_1(n) = \begin{cases} swin(n) & 0 \le n < 100 \\ 0 & \text{otherwise} \end{cases} \tag{753}$$

The energies of the $s\varepsilon_{scaled}$ and S_{HB}^{Tar} for the entire duration of the frame and the overlaps is calculated using a window:

$$\varphi_{Tar} = \sum_{n=0}^{360} \left(S_{HB}^{Tar}(n-20) \right)^2 \times swin_1(n) \tag{754}$$

$$\varphi_{ShE} = \sum_{n=0}^{360} \left(s\varepsilon(n-20) \right)^2 \times swin_1(n) \tag{755}$$

where the negative samples are obtained from previous frames, and the window function $swin_1(n)$ is given by

$$swin_1(n) = \begin{cases} win(n) & 0 \le n < 20 \\ 1.0 & 20 \le n < 339 \\ win(360-n) & 340 \le n < 359 \end{cases} \tag{756}$$

where $win(n)$ is defined in table 60.

If the high band target energy as calculated in Equation (754) is saturated, then the target signal, S_{HB}^{Tar}, is scaled based on the number of subframes that are saturated (from clause 5.2.6.1.14.2). Then the high band target energy is recalculated using the scaled target signal using Equation (754). Subsequently, the gain frame GF in Equation (757) is compensated for the scaling performed on the target signal.

The overall frame gain parameter GF is calculated as

$$GF = \sqrt{\frac{\varphi_{Tar}}{\varphi_{ShE}}} \tag{757}$$

The parameter GF is attenuated if the quantization of the gain shape parameter is poor. First the power-off-the-peak value for unquantized and quantized gain shape parameters is calculated:

$$\pi_{unq} = \sum_{\substack{i=1 \\ i \ne argmax\{gs(j), \text{ for } j=1,..4\}}}^{4} gs(j)^2 \tag{758}$$

$$\pi_q = \sum_{\substack{i=1 \\ i \ne argmax\{gs_q(j), \text{ for } j=1,..4\}}}^{4} gs_q(j)^2 \tag{759}$$

If $\pi_q > 2.0 \times \pi_{unq}$, then the gain shape parameter quantization is deemed poorly quantized and the gain frame parameter is attenuated as follows in order to avoid perceptually annoying artifacts in the reconstructed speech signal at the decoder.

$$GF = GF \times \sqrt{\frac{2 \times \pi_{unq}}{\pi_q}} \tag{760}$$

Further, the *GF* parameter is smoothed if the current frame is determined to be similar in spectral characteristics to the previous frames. To determine similarity in spectral characteristics, the fast and slow evolution rates, *FER* and *SER* , and the minimum LSP spacing δ_{min} are determined as follows:

$$SER = \sum_{k=1}^{10} \left(\rho_k^{SHB} - lsp_{se}(k) \right)^2 \text{ and } FER = \sum_{k=1}^{10} \left(\rho_k^{SHB} - lsp_{fe}(k) \right)^2 \tag{761}$$

where $lsp_{se}(k) = 0.7 \times lsp_{se}(k-1) + 0.3 \times \rho_{k-1}^{SHB}$ and $lsp_{se}(k) = 0.3 \times lsp_{fe}(k-1) + 0.7 \times \rho_{k-1}^{SHB}$ and $\delta_{min} = \min(\delta_1(k))$

and $\delta_1(k) = \rho_k^{SHB}$ when k=1 and $\delta_1(k) = \rho_k^{SHB} - \rho_{k-1}^{SHB}$ otherwise.

A smoothed version of δ_{min} using the δ_{min} values from the previous frames is also determined:

$$\delta_{smoothed} = 0.4 \times \delta_{min} + 0.3 \times \delta_{min}\left(z^{-1}\right) + 0.2 \times \delta_{min}\left(z^{-2}\right) + 0.1 \times \delta_{min}\left(z^{-3}\right) \tag{762}$$

If the minimum LSP spacing and smoothed LSP spacing satisy a spacing critierion (e.g., $\delta_{min} < 0.008$, $\delta_{smoothed} < 0.005$) indicating an artifact generating condition, the high band target is filtered using the high band LP to attenuate the coding artifacts before estimating the gain shape and gain frame parameters.

If the *SER* and the *FER* values are smaller than 0.001, then the gain frame parameter *GF* is smoothed between the previous and the current frame.

$$GF = 0.5 \times \left(GF + GF_{prev} \right) \tag{763}$$

Also, if $\delta_{smoothed} < 0.0024$, then an additional attenuation of the *GF* is performed: $GF = \left(GF\right)^{0.8}$. Also, if the current frame's $GF > 3.0 \times GF_{prev}$, then *GF* is modified as per

$$GF = GF \times \left(0.8 + 0.5 \times fb \right) \tag{764}$$

where *fb* is calculated as described in subclause 5.2.6.1.15.

For the WB mode and if the bitrate is 9.6 kb/s, the gain frame parameter *GF* is further adapted, based on the average of the 4 voicing factors, $Vf = 0.25 \times \sum_{i=1,..4} Vf_i$, as $GF = 0.5 \times GF$, if the low band coder type is voiced, and as $GF = 0.75 \times GF$, if the low band coder type is not voiced, but $Vf \geq 0.35$.

The gain frame parameter *GF* is scalar quantized in the log domain using 5 bits. The lowest quantization point is chosen to be -1.0 and the quantization steps are set to be 0.15.

5.2.6.1.16   Estimation of TEC/TFA envelope parameters

The Temporal Envelope Coding (TEC) and the Temporal Flatness Adjuster (TFA) shape the temporal envelope of the high frequency band signal generated by the TBE. They are enabled as the post processing of the TBE at 16.4 and 24.4 kbps with the output sampling frequencies higher than 8000 Hz.

The TEC is for getting better shapes of onsets at the decoder by using the temporal envelope of the low frequency band and the transmitted information on the temporal envelope of the high frequency band. The information indicates the two different shapes, one is "steep onset" and the other is "gentle onset". The TEC has two modes for accommodating those shapes of the onsets, "no smoothing mode" is for steep onsets and "smoothing mode" is for gentle onsets. Thus, the transmitted information represents the mode of TEC to be used at the decoder as well as the shape of the onset. At the decoder, the information is used to calculate the temporal envelope of the high frequency band.

The TFA flattens the temporal envelope of the high frequency band according to the transmitted information on the flatness of the temporal envelope of the input signal.

5.2.6.1.16.1   Estimation of TEC parameters

The TEC parameter *tec_flag* to be transmitted to the decoder is estimated as follows, which is the information on the temporal envelope of the high frequency band and indicates the activation and the used mode of the TEC at the decoder.

The temporal envelope of the low frequency band of the input signal is defined as the mean of the temporal envelopes of three sub-bands in the low frequency band (sub-low-frequency-bands) in the CLDFB domain. The temporal envelope of the m-th sub-low-frequency-band $env_{ls}{}^m(i)$ is calculated by

$$env_{ls}{}^m(i) = 10\log_{10}\left(\frac{1}{(k_{u,m} - k_{l,m} + 1)}\sum_{k=k_{l,m}}^{k_{u,m}} E_C(i,k)\right) \quad \text{for } i = 0,...,15 \tag{765}$$

where $E_C(i,k)$ is the energy in the CLDFB domain described in subclause 5.1.2.2 and, $k_{l,m}$ and $k_{u,m}$ are the lower and upper limits of the CLDFB sub-band of the m-th sub-low-frequency-band. And then, the temporal envelope of the low frequency band $env_l(i)$ is calculated by

$$env_l(i) = \frac{1}{3}\sum_{m=0}^{2} env_{ls}{}^m(i) \quad \text{for } i = 0,...,15 \tag{766}$$

The temporal envelope of the high frequency band of the input signal $env_h(i)$ is calculated in the CLDFB domain.

$$env_h(i) = 10\log_{10}\left(\frac{1}{(k_u - k_x + 1)}\sum_{k=k_x}^{k_u} E_C(i,k)\right) \quad \text{for } i = 0,...,15 \tag{767}$$

where $E_C(i,k)$ is the energy in the CLDFB domain described in subclause 5.1.2.2 and, k_x and k_u are the lower and upper limits of the CLDFB sub-band in the frequency range of the TBE.

The shape of the onset is detected by these temporal envelopes of the high and low frequency bands. A steep onset is detected and the no smoothing mode is selected when $tec\_mode = 1$. And, a gentle onset is detected and the smoothing mode is selected when $tec\_mode = 0$.

$$tec\_mode = \begin{cases} 1, & v_h > 800 \text{ and } v_l > 720 \text{ and } \sum_{i=0}^{15}(env_h - env_l) < 100 \\ 0, & otherwise \end{cases} \tag{768}$$

where v_h and v_l are the variances of e_h and e_l respectively.

When $tec\_mode = 1$, the maximum values of the temporal envelopes of the high and low frequency bands and their positions are detected:

$$max_{envh} = env_h(max\_pos_{envh}) = \max_{i=0,...,15}(env_h(i)) \tag{769}$$

$$max_{envl} = env_l(max\_pos_{envl}) = \max_{i=0,...,15}(env_l(i)) \tag{770}$$

And then, the local minimum and maximum values of the temporal envelope of the high frequency band at the position i, are detected and the difference between the local maximum and minimum values is calculated:

$$\begin{cases} lmax_{envh}(i) = \max_{j=i-2,...,i}(env_h(j)) \\ lmin_{envh}(i) = \max_{j=i-2,...,i}(env_h(j)) \end{cases}, \quad \text{for } i = 0,...,15 \tag{771}$$

$$diff_{l\max\_l\min}(i) = lmax_{envh}(i) - lmin_{envh}(i), \quad \text{for } i = 0,...,15 \tag{772}$$

The maximum of the difference $diff_{l\max\_l\min}(i)$ and its position are detected:

$$max_{diff\_l\max\_l\min} = diff_{l\max\_l\min}(max\_pos_{diff\_l\max\_l\min}) = \max_{i=0,...,15}(diff_{l\max\_l\min}(i)) \tag{773}$$

Using the parameters above, tec_flag is set as

$$tec\_flag = \begin{cases} 2 & \begin{array}{l} \text{abs}(max\_pos_{envh} - max\_pos_{envl}) < 2 \text{ and } max_{diff\_l\max\_l\min} > 20 \text{ and} \\ \text{abs}(max\_pos_{diff\_l\max\_l\min} - max\_pos_{envh}) < 2 \end{array} \\ 0 & otherwise \end{cases}$$ (774)

When $tec\_mode = 0$, the smoothed temporal envelope of the low frequency band is calculated

$$env_{l,sm}(i) = 1.5894 \times \sum_{j=0}^{5} sc(j) \cdot env_l(i-j) \quad \text{for } i = 0,...,15$$ (775)

where

$$sc = [0.3662 \quad 0.1078 \quad 0.1194 \quad 0.1289 \quad 0.1365 \quad 0.1412]$$ (776)

And then, the correlation coefficient $corr_{lsm,h}$ between $e_{l,sm}(i)$ and $e_h(i)$ is calculated. Further, the ratio between the variances of $e_{l,sm}(i)$ and $e_h(i)$ is calculated:

$$rvar_{lsm,h} = \frac{v_h}{v_{l,sm} + \varepsilon_0}$$ (777)

Using these parameters, tec_flag is set as

$$tec\_flag = \begin{cases} 1 & corr_{lsm,h} \geq 0.8795 \text{ and } 0.3694 < rvar_{lsm,h} < 2.0288 \\ 0 & otherwise \end{cases}$$ (778)

5.2.6.1.16.2   Estimation of TFA parameters

The parameter tfa_flag to be transmitted to the decoder is estimated as follows, which represents the flatness of the temporal envelope of the high frequency band as well as the activation of the TFA at the decoder.

The flatness measure fl_{shbTar} of the temporal envelope of the signal in the high band target frame $S_{HB}^{Tar}(n)$ is calculated as:

$$fl_{shbTar} = \frac{\left(\prod_{i=0}^{15} env_{shbTar}(i)\right)^{\frac{1}{16}}}{\varepsilon_0 + \frac{1}{16}\sum_{i=0}^{15} env_{shbTar}(i)}$$ (779)

where $env_{shbTar}(i) = \sum_{n=20i}^{20(i+1)-1} S_{HB}^{Tar}(n)^2$ $i = 0,1,\ldots,15$ and where $\varepsilon_0 = 10^{-12}$.

In addition to the flatness measure, the mean of the pitch lags m_{pitlag} and the mean of the open-loop pitch gains of the half frames $m_{pitgain}$ are calculated. Finally, depending to the last core mode, the parameter tfa_flag is estimated by the parameters above:

in the case that the last core mode is not TCX20,

$$tfa\_flag = \begin{cases} 1 & fl_{shbTar} > 0.7 \text{ and } m_{pitlag} > 100 \text{ and } m_{pitgain} > 0.7 \\ 0 & otherwise \end{cases}$$ (780)

in the case that the last core mode is TCX20,

$$tfa\_flag = \begin{cases} 1 & fl_{shbTar} > 0.5 \text{ and } m_{pitgain} < 0.7 \\ 0 & otherwise \end{cases} \tag{781}$$

5.2.6.1.16.3 Set the transmitted parameter for TEC and TFA

The TEC and TFA parameters are jointly coded by 2 bits and then transmitted to the decoder together with the other TBE information:

$$tec\_tfa\_bits = \begin{cases} 00 & tfa\_flag = 0 \text{ and } tec\_flag = 0 \\ 01 & tfa\_flag = 1 \\ 10 & tfa\_flag = 0 \text{ and } tec\_flag = 1 \\ 11 & tfa\_flag = 0 \text{ and } tec\_flag = 2 \end{cases} \tag{782}$$

5.2.6.1.17 Estimation of full-band frame energy parameters

The full-band TBE algorithm is used for coding the full band. Since the fine signal structure in the full bands are closely related to that in the high band and low band, the full band of the signal are coded by using only 4 bits together with the information from the low band and the high band. A synthesized full band signal is obtained by coding the high band and predicting spectrum from the high band, the synthesized full band signal is then de-emphasized with μ_{FB} which is determined by the characteristic factors derived from coding the low band signal. A band passed full band signal is obtained by band-pass filtering the input signal. The energy ratio is calculated by comparing the energy calculated from the de-emphasized synthesized full band signal with the energy calculated from the band passed full band signal. Finally the parameters including the characteristic factors, high band coding information and the energy ratio are transmitted to the decoder.

The synthesized full band signal is obtained as follow: A vector of random numbers $rnd(n), n = 0,...319$ of length 320 passes through the LPC synthesis filter, the spectrum of $rnd(n), n = 0,...319$ is used as the predicted spectrum from the high band and the coefficients of the LPC synthesis filter are derived from the quantized LPC coefficients (see subclause 5.2.6.1.3).

$$s\varepsilon_{FB}(n) = rnd(n) - \sum_{k=1}^{11} \tilde{a}_k^{SHB} \times s\varepsilon_{FB}(n-k), n = 0,...,319 \tag{783}$$

Then the synthesized full band signal $s\varepsilon_{FB}(n), n = 0,...319$ is de-emphasized as follows:

The spectrum of $s\varepsilon_{FB}(n), n = 0,...319$ are moving corrected described as follows:

$$s\hat{\varepsilon}_{FB}(n) = s\varepsilon_{FB}(n) \times \cos\_fb\_exc(\mod(n/32)) \tag{784}$$

The variables $\cos\_fb\_exc(n), for n = 0,...31$ are the parameters of spectrum moving correction tabulated in Table 61 below:

Table 61: Parameters of spectrum moving correction

Index	Value	Index	Value
0	9.536743164062500e-007	16	-9.536743164062500e-007
1	9.353497034680913e-007	17	-9.353497034680913e-007
2	8.810801546133007e-007	18	-8.810801546133007e-007
3	7.929511980364623e-007	19	-7.929511411930434e-007
4	7.929511980364623e-007	20	-6.743495077898842e-007
5	5.298330165715015e-007	21	-5.298329597280826e-007
6	3.649553264040151e-007	22	-3.649552411388868e-007
7	1.860525884467279e-007	23	-1.860525173924543e-007
8	0.000000000000000e+000	24	0.000000000000000e+000
9	-1.860526737118562e-007	25	1.860527589769845e-007
10	-3.649554116691434e-007	26	3.649554969342717e-007
11	-5.298331302583392e-007	27	5.298331871017581e-007
12	-6.743496214767220e-007	28	6.743496783201408e-007
13	-7.929512548798812e-007	29	7.929513117233000e-007
14	-8.810802114567196e-007	32	8.810802683001384e-007
15	-9.353497603115102e-007	31	9.353497603115102e-007

And then flip the spectrum of $s\hat{\varepsilon}_{FB}(n), n = 0,...319$ to get $s\varepsilon'_{FB}(n), n = 0,...319$

$$s\varepsilon'_{FB}(n) = (-1)^{n+1} * s\hat{\varepsilon}_{FB}(n), n = 0,...,319 \qquad (785)$$

The signal $s\varepsilon'_{FB}(n), n = 0,...319$ is then de-emphasized with μ_{FB} described as follows:

$$s\varepsilon''_{FB}(n) = s\varepsilon'_{FB}(n) - \mu_{FB} \times s\varepsilon'_{FB}(n-1) \qquad (786)$$

The de-emphasized factor μ_{FB} is determined by the characteristic factors of the signal such as "voicing factors", "spectral tilt", "short-term average energy", "short-term average zero crossing rate". The calculation of de-emphasized factor μ_{FB} using the voicing factors is described as follows:

$$\mu_{FB} = \max((0.68 - Vf^3), 0.48) \qquad (787)$$

$$Vf = \begin{cases} 0.25 * \sum_{i=1}^{i=4} Vf_i, rate = 13.2kbps \\ 0.2 * \sum_{i=1}^{5} Vf_i, rate = 16.4kbps, 24.4kbps, 32kbps \end{cases} \qquad (788)$$

where Vf_i is the voicing factor of the ith subframe.

The signal $s\varepsilon''_{FB}(n), n = 0,...,319$ is modulated by the gain shape values $gs(j), j = 0,1,2,3$ (see subclause 5.2.6.1.14) and the overall frame gain parameter GF (see subclause 5.2.6.1.15).

$$s\hat{\varepsilon}_{FB}(n) = s\varepsilon''_{FB}(n) * GF * gs(16 * n / 320), n = 0,...,319 \qquad (789)$$

The energy of $s\hat{\varepsilon}_{FB}(n), n = 0,...319$ is described as follows:

$$\varphi_{syn} = \sum_{n=0}^{319} s\hat{\varepsilon}_{FB}(n) * s\hat{\varepsilon}_{FB}(n) \qquad (790)$$

The original input signal $s(n), n = 0,...959$ pass through the band-pass filter to get the band passed full band signal $\hat{s}(n), n = 0,...959$, and then calculate the energy φ_{ori} of $\hat{s}(n), n = 0,...959$ from 16 kHz to 20 kHz. The energy ratio is calculated as follows:

$$\varphi_{ratio} = \sqrt{\frac{\varphi_{ori}}{\varphi_{syn}}} \tag{791}$$

$$\varphi_{ratio\_q} = \log_2 \varphi_{ratio} \tag{792}$$

$$\hat{\varphi}_{ratio\_q} = \max(0, \min(15, \varphi_{ratio\_q})) \tag{793}$$

The energy ratio $\hat{\varphi}_{ratio\_q}$ is then transmitted to the decoder using 4 bits.

5.2.6.2 Multi-mode FD Bandwidth Extension Coding

The input signal of current frame is divided into two parts: the low band signal and the super higher band (SHB) signal for SWB signal or the higher band (HB) signal for WB signal. The low band signal of the input signal is coded by LP coding modes, and the SHB or HB signal of the input signal is coded by the multi-mode FD bandwidth extension (BWE) algorithm. A classification decision process of the SHB or HB signal of input signal is first performed. Then, the multi-mode BWE algorithm for LP-based coding modes uses a combination of adaptive spectral envelope and time envelope coding for super wideband extension, and spectral envelope coding for wideband extension, according to the result of the classification decision process. The coded bitstream of low band signal, the adaptive coded bitstream of SHB or HB signal as well as the result of the classification decision process of current input signal are output. Table 62 describes the multi-mode FD BWE at the different bitrates of operation. Theoretically, the delay of the synthesized output signal can be determined adaptively in the range of $[\max(D_1, D_2), \ D_1 + D_2]$ according to the delay of the core coding algorithm D_1 and the delay of the multi-mode bandwidth extension algorithm D_2. To achieve lowest delay for bandwidth extension coding, the super higher band (SHB) signal is delayed by $D_1 - D_2$. Here D_1 is larger than D_2. Then, the achieved lowest delay of the multi-mode bandwidth extension is D_1 in encoder side.

Table 62: Multi-mode FD BWE at different bitrates

Bitrate [kbps]	Bandwidth	Multi-mode FD BWE
7.2, 8	WB	Blind, HARMONIC/NORMAL
13.2	WB	Guided, HARMONIC/NORMAL
	SWB	Guided, TRANSIENT/HARMONIC/NORMAL/NOISE
32	SWB/FB	Guided, TRANSIENT/HARMONIC/NORMAL/NOISE

5.2.6.2.1 SWB/FB Multi-mode FD Bandwidth Extension

For frames declared as TRANSIENT (TS) frames or as non-TRANSIENT, a bit budget of 31 bits is allocated to the SWB Multi-mode FD Bandwidth Extension. If the super higher band (SHB) signal of the input in the previous frame or in the current frame is detected as TRANSIENT, then the current frame is also classified as TRANSIENT. Non-TRANSIENT frames can be further classified as HARMONIC (HM), NORMAL (NM) or NOISE (NS) depending upon the frequency fluctuation that is detected. Two bits of the bit budget are allocated to the signal class. In case of a TRANSIENT frame, the remaining 29 bits are allocated to encode four spectral envelopes and four time envelopes. For other cases, i.e. non-TRANSIENT frames, the remaining 29 bits are allocated to encode fourteen spectral envelopes, and no time envelope is encoded. For FD BWE encoding, the 320 MDCT coefficients of the SHB signal, $X_{M\_SHB}(k), k = 0,…,319$, are coded. In the case of the FB mode, the encoding algorithm from 8kHz to 15.5kHz is the same as the SWB mode. To encode 15.5kHz to 20kHz, the spectral energies of from 11 kHz to 15.5 kHz and from 15.5 kHz to 20 kHz are calculated, and then the ratio of the two energies is coded using 4 bits after being quantized.

5.2.6.2.1.1 Windowing and time-to-frequency transformation

The input high-pass filtered signal $s_{HP}(n)$ is delayed by $D_1 - D_2$ samples and windowed to obtain the windowed input signal $\tilde{x}(n)$ as shown in figure 51, where $D_1 - D_2$ is equal to:

$$D_1 - D_2 = \begin{cases} 102 & 13.2kbps \\ 98 & 32kbps \end{cases} \tag{794}$$
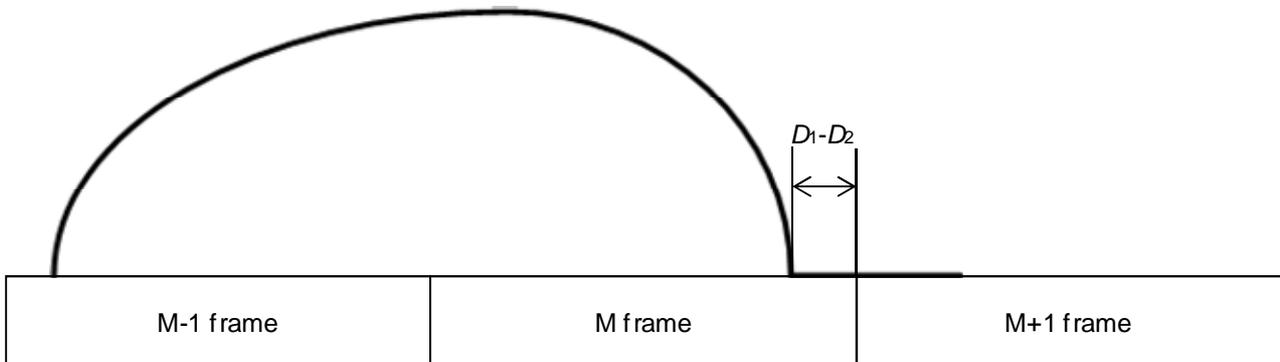
Figure 51: Windowing of the input high-pass filtered signal

A 640-point length MDCT on top of $\tilde{x}(n)$, $n = 0,1,\ldots,639$ is used for SWB FD BWE. Refer to subclause 5.3.2.

5.2.6.2.1.2 Transient detection

The input time-domain SHB signal $s_{SHB}(n)$ of current frame, sampled at 16kHz, is first high-pass filtered; the high-pass filter serves as a precaution against low frequency components adversely affecting the processing. A first order IIR filter is used, and it is given by:

$$H_{HP}(z) = \frac{0.7446(1 - z^{-1})}{1 - 0.4931z^{-1}} \tag{795}$$

The output of the high-pass filter $s_{SHB\_HP}(n)$ is obtained according to:

$$s_{SHB\_HP}(n) = 0.4931s_{HP}(n-1) + 0.7446s_{SHB}(n) - 0.7446s_{SHB}(n-1) \quad for\ n = 0,\ldots,L-1 \tag{796}$$

where $L = 320$ denotes the 20ms frame length at 16kHz. The high-pass filtered signal $s_{SHB\_HP}(n)$ is divided into four sub-frames; each corresponding to 5 ms or 80 samples.

The energy of each sub-frame, $E^{[m]}$, is computed according to:

$$E^{[m]} = \sum_{n=mL/4}^{(m+1)L/4-1} \left(s_{SHB\_HP}(n)\right)^2 \qquad for\ m = 0,\ldots,3 \tag{797}$$

For each sub-frame, the signal's long term energy, $E_{LT}^{[m]}$, is updated according to the following equation:

$$E_{LT}^{[m]} = (1 - \alpha)E_{LT}^{[m-1]} + \alpha E^{[m]} \qquad for\ m = 0,\ldots,3 \tag{798}$$

In the above equation, the forgetting factor α is set to 0.25, and the convention is that for the first sub-frame, $E_{LT}^{[-1]} = E_{LT}^{[3]}$ from the previous frame. It should be noted that when the current frame and the previous frame apply different BWE algorithms, or the core configurations are different, then the signals' long term energy $E_{LT}^{[-1]}$ is calculated as $E_{LT}^{[-1]} = \sum_{n=0}^{79} \left(s_{HP}(n)\right)^2$.

The memory state of the high-pass filter, $H_{HP}(z)$ and $E_{LT}^{[3]}$ are saved for the next frame's processing. For each sub-frame m, a comparison between the short term energy $E^{[m]}$ and the short term energy of previous sub-frame $E^{[m-1]}$ or the long term energy $E_{LT}^{[m-1]}$ is performed to detect whether the current frame is TRANSIENT or not. A transient is detected whenever the energy ratio is above a certain threshold which is larger than 1. Formally, a transient is detected whenever:

$$E^{[m]} > \rho E_{LT}^{[m-1]} \tag{799}$$

where ρ is the energy ratio threshold and is set to $\rho = 10$ for INACTIVE frames and $\rho = 13.5$ otherwise.

It should be noted that if the previous frame did not use SWB Multi-mode FD BWE then the current frame is not classified as a transient frame. In general, the time-frequency transform is applied on a 40ms frame; therefore, a transient affects two consecutive frames. To overcome this, a hangover for a detected transient is applied. A transient detected at a certain frame also triggers a transient in the next frame.

The output of the transient detector is a flag, denoted *IsTransient*. The flag is set to the logical value *TRUE* if a transient is detected or *FALSE* otherwise.

In addition, the parameters of spectral tilt and frame class for the current frame are also used for further refinement of the transient decision.

The spectral tilt of the low frequency signal $s_{12.8\_16}(n)$ is calculated by:

$$tilt\_bwe = r_1 / \sqrt{r_0} \tag{800}$$

where, r_0 and r_1 are calculated by:

$$r_0 = \sum_{n=0}^{255} s_{12.8\_16}(n) \cdot s_{12.8\_16}(n) \tag{801}$$

r_1 is initialized to $\left| s_{12.8\_16}(1) - s_{12.8\_16}(0) \right|$, and if
$\left(s_{12.8\_16}(n) - s_{12.8\_16}(n-1) \right) \cdot \left(s_{12.8\_16}(n-1) - s_{12.8\_16}(n-2) \right) < 0$, $2 \le n < 256$, r_1 is adjusted by adding
$\left| s_{12.8\_16}(n) - s_{12.8\_16}(n-1) \right|$.

The spectral tilt and class of the current frame are checked against threshold values, and the high frequency TRANSIENT signal classification of the current frame is adjusted. Formally, when *IsTransient* is *TRUE*, and $tilt > 8$, the *IsTransient* signal classification is set to *FALSE*, and also the hangover is set to 0.

Another flag, *IsTransient_LF*, is used to indicate whether a transient is present in the low frequency signal $s_{LF}^{[m]}$ of the current frame. It is calculated as follows: When the condition $E_{LF}^{[m]} > 5.5 E_{LF}^{[m-1]}$, $m = 0,...,3$ is satisfied, then the *IsTransient_LF* flag is set to logical *TRUE;* and it is set to *FALSE* otherwise.

Here, $E_{LF}^{[m]} = \sum_{n=64m}^{64m+63} \left(s_{LP}(n) \right)^2$, $m = 0,...,3$, and the convention is that for the first sub-frame, $E_{LT}^{[-1]} = E_{LT}^{[3]}$ from the previous frame.

5.2.6.2.1.3   Frequency domain classification and coding

In frames that are classified as containing transients, the value of F_{class} is set (F_{class} =TRANSIENT). For frames without transients, a frequency sharpness parameter is computed to reflect the spectral fluctuation of the frequency coefficients in the super high band signal and those frames are categorized in one of three classes:

   a) HARMONIC: when frequency sharpness is high.

   b) NOISE: when frequency sharpness is low.

c) NORMAL: when frequency sharpness is moderate.

The 288 MDCT coefficients in the 6400-13600 Hz frequency range, $X_{M\_C}(k), k = 0,...,287$ are split into nine sharpness bands (32 coefficients per band). The frequency sharpness, $\kappa(j)$, is then defined as the ratio of the peak magnitude to the average magnitude in a sharpness band j

$$\kappa(j) = \begin{cases} \dfrac{31 . A_{sharp}(j)}{\sum_{k=32j}^{32j+31}\left|X_{M\_C}(k) - A_{sharp}(j)\right|}, & if \ \sum_{k=32j}^{32j+31}\left|X_{M\_C}(k)\right| \neq 0 \\ 0, & otherwise \end{cases} \quad j = 0,...,8 \quad (802)$$

where the maximum magnitude of spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is given by

$$A_{sharp}(j) = \max_{k=32j,...32j+31}\left|X_{M\_C}(k)\right| \qquad j = 0,...,8 \qquad (803)$$

Then, six parameters are calculated; the global gain, $g_{glob} = \sum_{k=0}^{319}\left(X_{M\_SHB}(k)\right)^2$, the

average $A_{av} = \dfrac{1}{288}\sum_{k=0}^{287}\left|X_{M\_C}(k)\right|$, the $S_{var} = \sum_{j=0}^{8}\left|\dfrac{1}{32}\sum_{k=32j}^{32j+31}\left|X_{M\_C}(k)\right| - A_{av}\right|$, and three further sharpness

parameters are determined; the maximum sharpness, κ_{max}, the sharpness band counter, c_{sharp}, and the noise band counter, c_{noise}.

The maximum sharpness, κ_{max}, in all sharpness bands is computed as:

$$\kappa_{max} = \max_{j=0,...8}\left(A_{sharp}(j)\right) \qquad (804)$$

The counter c_{sharp} is computed from the nine frequency sharpness parameters, $\kappa(j)$, and from the nine maximum magnitudes, $A_{sharp}(j)$, as follows: Initialized to zero, c_{sharp} is incremented by one for each j, $j = 0,...,8$, if $\kappa(j) > 4.5$ and $A_{sharp}(j) > 8$.

The counter c_{noise} is computed from the nine frequency sharpness parameters $\kappa(j)$ as follows: Initialized to zero, c_{noise} is incremented by one for each j, $j = 0,...,8$ if $\kappa(j)$ is less than 3.

The class of non-TRANSIENT frames is determined from the three sharpness parameters, κ_{max}, c_{sharp} and c_{noise} and four other parameters; the previous frame saved class, $F_{class}^{[-1]}$, A_{av}, S_{var}, and the ratio of the global gains in the current and previous frames.

The threshold of the number of harmonics i_{sharp} and the threshold of the maximum sharpness l_{sharp} are set according to the signal class of previous frame $F_{class}^{[-1]}$ and the mode of previous extension layer M_{last_extl}. When the bandwidth is changed i.e., $M_{last\_extl} \neq SWB\_BWE$, the i_{sharp} and l_{sharp} will be decreased for harmonic mode and increased for other signal mode:

$$i_{sharp} = \begin{cases} \begin{cases} 4, & if \ F_{class}^{[-1]} = HARMONIC \\ 8, & else \ if \ F_{class}^{[-1]} = TRANSIENT \\ 6, & otherwise \end{cases} & if \ M_{last\_extl} = SWB\_BWE \\ \\ \begin{cases} 2, & if \ F_{class}^{[-1]} = HARMONIC \\ 8, & otherwise \end{cases} & otherwise \end{cases} \qquad (805)$$

$$l_{sharp} = \begin{cases} 10 & if \ M_{last\_extl} = SWB\_BWE \\ 5, & else \ if \ F_{class}^{[-1]} = HARMONIC \\ 20, & otherwise \end{cases} \quad (806)$$

- If $c_{sharp} > i_{sharp}$, $0.5 < g_{glob}/g_{glob}^{[-1]} < 1.8$ and $\kappa_{max} > l_{sharp}$, the current frame is classified as HARMONIC frame (F_{class} = HARMONIC) and the counter for signal class c_{class} is incremented by one when c_{class} is less than twelve. Otherwise, c_{class} is decremented by one when c_{class} is larger than zero.

- Then, if $c_{class} \geq 2$, the current frame is also classified as HARMONIC frame (F_{class} = HARMONIC).

- For other cases, depending on the noise counter c_{noise}, A_{av}, S_{var}, and the spectral tilt *tilt_bwe*, the current frame is classified as NORMAL or NOISE:

  if $c_{noise} > 4$, $S_{var} < 4.8 \cdot A_{av}$ and *tilt_bwe* < 5, the current frame is classified as NOISE frame (F_{class} = NOISE), otherwise, the current frame is classified as NORMAL frame (F_{class} = NORMAL).

Two bits are transmitted for SHB signal class coding. Table 63 gives the coded bits for each class.

Table 63: SHB signal class coding

Signal class F_{class}	Coded bits
NOISE	00
TRANSIENT	01
NORMAL	10
HARMONIC	11

The signal class F_{class} of the current frame is preserved as $F_{class}^{[-1]}$ for the next frame.

5.2.6.2.1.4   Sub-band division

The 320 MDCT coefficients ($k_{offset} = 6$ at 13.2kbps in the 6150-14150 Hz frequency range, $k_{offset} = 80$ at 32kbps in the 8000-16000 Hz frequency range) are either split into four sub-bands for TRANSIENT frames or fourteen sub-bands for non-TRANSIENT frames. Table 64 and table 65 define the sub-band boundaries and sizes for TRANSIENT frames and non-TRANSIENT frames respectively. The j-th sub-band comprises coefficients $X_{M\_SHB}(k)$ where $b_{swb}(j) \leq k < b_{swb}(j+1)$.

Table 64: Sub-band boundaries and number of coefficients per sub-band in TRANSIENT frames

j	$b_{swb}(j)$	$N_{swbcf}(j)$
0	0	76
1	76	76
2	152	84
3	236	84
4	320	-

Table 65: Sub-band boundaries and number of coefficients per sub-band in Non-TRANSIENT frames

j	$b_{swb}(j)$	$N_{swbcf}(j)$
0	0	16
1	16	24
2	40	16
3	56	24
4	80	16
5	96	24
6	120	16
7	136	24
8	160	24
9	184	24
10	208	24
11	232	24
12	256	32
13	288	32
14	320	-

5.2.6.2.1.5   Spectral envelope calculation and quantization

The spectral envelope, or the energy of each band, is computed as follows:

$$f_{rms}(j) = 10\log\left(\frac{1}{N_{swbcf}(j)}\sum_{k=b_{swb}(j)}^{b_{swb}(j+1)-1}\left(X_{M\_SHB}(k)\right)^2 + \varepsilon_{rms}\right) \quad for \ \ j = 0,...,13 \tag{807}$$

If the current frame is a Non-TRANSIENT frame, energy control is performed to prevent too much noise being applied to the generated spectrum. The energy control adjusts the spectral energy of each band, depending on the different characteristics between the original high frequency spectrum and the base excitation spectrum.

First a spectral copy is created by mapping the frequencies, depending upon the bandwidth, the ACELP coding modes and the FD BWE mode as defined in table 66.

Table 66: Frequency mapping to generate base excitation spectrum in FD BWE

Mode, Bit-rate, ACELP coding modes	l	$St_{src,FD}(l)$	$End_{src,FD}(l)$	$St_{dst,FD}(l)$	$End_{dst,FD}(l)$
WB @ 7.2, 8kbps All LP-based modes except for AUDIO	0	160	239	240	319
WB @ 7.2, 8kbps AUDIO WB @ 13.2kbps All	0	80	159	240	319
SWB @ 13.2kbps NORMAL, NOISE	0	112	239	246	373
	1	112	239	374	501
	2	176	239	502	565
SWB @ 13.2kbps HARMONIC	0	0	239	246	485
	1	128	207	486	565
SWB @ 32kbps NORMAL, NOISE	0	112	239	320	447
	1	112	239	448	575
	2	176	239	576	639
SWB @ 32kbps HARMONIC	0	0	239	320	559
	1	128	207	560	639

To generate the base excitation spectrum, the spectral copy is normalized by the sum of its absolute spectral components; the window size used depends on the signal characteristics.

$$X_{base\_M\_SHB}(k) = \frac{X_{cp\_SHB}(k)}{\sum_{l=k-\lfloor nlen/2 \rfloor}^{k+\lfloor (nlen-1)/2 \rfloor} \left| X_{cp\_SHB}(l) \right|} \qquad for \ k = 240 + k_{offset},...,559 + k_{offset} \qquad (808)$$

The tonality measures used for the energy control are then calculated:

$$Ton_{ec}(j) = \max \left(0.0001, \min \left(10\log \left(\frac{\frac{1}{N_{swbcf}(j)} \sum_{k=b_{swb}(j)}^{b_{swb}(j+1)-1} \left| X_M(k) \right|}{N_{swbcf}(j) \sqrt{\prod_{k=b_{swb}(j)}^{b_{swb}(j+1)-1} \left| X_M(k) \right|}} \right), 5.993 \right) \right) \quad for \ j = 0,...,13 \qquad (809)$$

The ratio between the tonality of the original high frequency spectrum ($Ton_{ec\_org}(j)$) and the tonality of the base excitation spectrum ($Ton_{ec\_sim}(j)$) is then calculated as follows:

$$ec\_fac_{FD}(j) = \begin{cases} \max \left(ec_{gam\_FD}, \frac{Ton_{ec\_sim}(j)}{Ton_{ec\_org}(j)} \right), & if \ Ton_{ec\_sim}(j) < 0.75 * Ton_{ec\_org}(j) \\ 1.0, & otherwise \end{cases}$$
$$for \ j = 0,...,13 \qquad (810)$$

where ec_{gam_FD} is 0.35.

The envelope control factor $ec\_fac_{FD}(j)$ is then applied to the envelope $\bar{f}_{rms\_SHB}(j)$:

$$f_{rms\_SHB}(j) = f_{rms}(j) * ec\_fac_{FD}(j) \qquad for \ j = 0,...,13 \qquad (811)$$

Next the spectral envelope is adjusted by subtracting the mean vectors f_{rms_mean} which are shown in table 67.

$$f'_{rms\_SHB}(j) = f_{rms\_SHB}(j) - f_{rms\_mean}(j) \qquad for \ j = 0,...,13 \qquad (812)$$

Table 67: Mean vectors in FD BWE

j	TRANSIENT	Non-TRANSIENT
0	27.23	28.62
1	23.81	28.96
2	23.87	28.05
3	19.51	27.97
4	-	26.91
5	-	26.82
6	-	26.35
7	-	25.98
8	-	24.94
9	-	24.03
10	-	22.94
11	-	22.14
12	-	21.23
13	-	20.40

If the current frame is a TRANSIENT frame, the following smoothing processes are applied before the envelope quantization.

– If *IsTransient _ LF* is *TRUE*, the coder type is INACTIVE and the transient hangover is equal to one, the flag is set to 1, and the time envelope is adjusted as follows:

$$\bar{f}_{rms}(j) = 0.05 f'_{rms}(j) \qquad for \ j = 0,...,3 \qquad (813)$$

– Otherwise, the adjustment is as follows:

$$\bar{f}_{rms}(j) = \begin{cases} f'_{rms}(j) & for \ j = 0,1 \\ 0.1 f'_{rms}(j) & for \ j = 2 \\ 0.05 f'_{rms}(j) & for \ j = 3 \end{cases} \qquad (814)$$

If the current frame is a Non-TRANSIENT frame,

$$\bar{f}_{rms\_SHB}(j) = f'_{rms\_SHB}(j) \qquad for \ j = 0,...,13 \qquad (815)$$

If the current frame is a TRANSIENT frame, the mean squared error (MSE) criterion is used for the search of the VQ, in a Non-TRANSIENT frame, the weighted mean squared error (WMSE) is used. The weighting serves to emphasise the lower frequency bands and is calculated by two methods; one is a deterministic weighting based solely on the frequency, and the other is a weighting that is calculated based upon the envelope. The first frequency weighting $w_{1\_SHB}(j)$ is defined in table 68 and the second frequency weighting $w_{SHB}(j)$ is calculated as follows:

$$w_{SHB}(j) = w_{1\_SHB}(j) * \left(\frac{f_{rms\_SHB}(j) - \min_{i=0,...,13}(f_{rms\_SHB}(i))}{\max_{i=0,...,13}(f_{rms\_SHB}(i)) - \min_{i=0,...,13}(f_{rms\_SHB}(i))} + 1 \right) \quad for \ j = 0,...,13 \quad (816)$$

Table 68: Frequency weighting $w_{1\_SHB}(j)$

j	Non-TRANSIENT
0	1.0
1	0.97826087
2	0.957446809
3	0.9375
4	0.918367347
5	0.9
6	0.882352941
7	0.865384615
8	0.849056604
9	0.833333333
10	0.818181818
11	0.803571429
12	0.789473684
13	0.775862069

The SWB spectral envelope is quantized with a multi-stage split VQ using envelope interpolation as in figure 52. In the first stage, two or three candidates' (N_{cand} , three in TRANSIENT frame, two in Non-TRANSIENT frame) indices are chosen using the error minimization criterion. The set of candidates with the least quantization error, taking into account all quantization steps, is then selected and the selected indices transmitted.
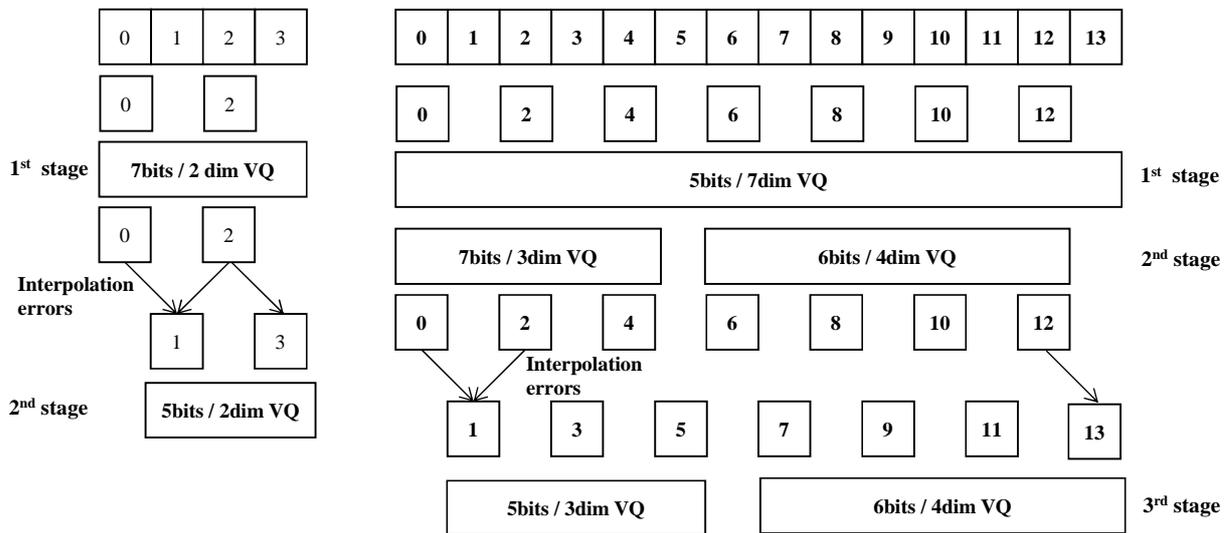
Figure 52: Envelope VQ in a TRANSIENT frame and a Non-TRANSIENT frame

Again during the first stage, values in even positions are selected and quantized using VQ with 5 bits for Non-TRANSIENT frames and 7 bits for TRANSIENT frames.

$$env_1(j) = \bar{f}_{rms\_SWB}(2*j) \qquad \begin{aligned} &for \ j = 0,...,6 \ \ \text{(Non - TRANSIENT)} \\ &for \ j = 0,1 \ \ \text{(TRANSIENT}\end{aligned} \qquad (817)$$

$$env_1(j) = \bar{f}_{rms\_SWB}(2*j) \qquad \begin{array}{l} for\ j = 0,...,6 \quad (Non\text{-}TRANSIENT) \\ for\ j = 0,1 \quad (TRANSIENT \end{array} \tag{818}$$

In Non-TRANSIENT frames, the candidate indices from the first stage VQ are defined as $idx_{0\_env_1}, idx_{1\_env_1}$. The quantization error err_1 is calculated and the error is split into err_{21} and err_{22} and quantized using 7 bits and 6 bits respectively, as follows:

$$err_1(j) = \bar{f}_{rms\_SHB}(2*j) - \hat{e}nv_1(j) \qquad for\ j = 0,...,6 \tag{819}$$

then;

$$\begin{array}{ll} err_{21}(j) = \bar{f}_{rms\_SHB}(2*j) - \hat{e}nv_1(j) & for\ j = 0,1,2 \\ err_{22}(j) = \bar{f}_{rms\_SHB}(2*(j+3)) - \hat{e}nv_1(j+3) & for\ j = 0,1,2,3 \end{array} \tag{820}$$

The candidate indices from the second stage VQ are defined as $idx_{0\_err_{21}}, idx_{1\_err_{21}}$ and $idx_{0\_err_{22}}, idx_{1\_err_{22}}$.

The two quantized values $\hat{e}rr_{21}, \hat{e}rr_{22}$ are then combined:

$$\hat{e}rr_2(j) = \begin{cases} \hat{e}rr_{21}(j) & for\ j = 0,1,2 \\ \hat{e}rr_{22}(j-3) & for\ j = 3,4,5,6 \end{cases} \tag{821}$$

At odd positions, an interpolation using boundary values is applied for intra-frame prediction and the predicted error is calculated:

$$Ierr_3(j) = \begin{cases} \bar{f}_{rms\_SHB}(2*j+1) - \dfrac{(\hat{e}nv_1(j) + \hat{e}rr_2(j) + \hat{e}nv_1(j+1) + \hat{e}rr_2(j+1))}{2} & for\ j = 0,...,5 \\ \bar{f}_{rms\_SHB}(2*j+1) - (\hat{e}nv_1(j) + \hat{e}rr_2(j)) & for\ j = 6 \end{cases} \tag{822}$$

The errors are then split into $Ierr_{31}$ and $Ierr_{32}$ and quantized using 5 bits and 6 bits respectively.

$$\begin{array}{ll} Ierr_{31}(j) = Ierr_3(j) & for\ j = 0,1,2 \\ Ierr_{32}(j) = Ierr_3(j-3) & for\ j = 3,4,5,6 \end{array} \tag{823}$$

The candidate indices from the third stage VQ are defined as $idx_{0,Ierr_{32}}, idx_{1,Ierr_{32}}$. In a TRANSIENT frame only 2 stages of quantization are applied. At odd positions, an interpolation using boundary values is applied for intra-frame prediction and the predicted error is calculated and quantized

$$Ierr_2(j) = \begin{cases} \bar{f}_{rms\_SHB}(2*j+1) - \dfrac{(\hat{e}nv_1(j) + \hat{e}nv_1(j+1))}{2} & for\ j = 0 \\ \bar{f}_{rms\_SHB}(2*j+1) - \hat{e}nv_1(j) & for\ j = 1 \end{cases} \tag{824}$$

The candidate indices from the second stage VQ are defined as $idx_{0,Ierr_2}, idx_{1,Ierr_2}, idx_{2,Ierr_2}$.

The final selected set of indices $\{idx_{env_1}, idx_{err_{21}}, idx_{err_{21}}, idx_{Ierr_{31}}, idx_{Ierr_{32}}\}$ for a Non-Transient frame, or $\{idx_{env_1}, idx_{Ierr_2}\}$ for a Transient frame are then transmitted.

5.2.6.2.1.6 Time envelope calculation and encoding

In case of TRANSIENT frames, ie, the super higher band (SHB) signal of the input in the previous frame is detected as TRANSIENT and the super higher band (SHB) signal of the input in the current frame is detected as NON-TRANSIENT, or the super higher band (SHB) signal of the input in the current frame is detected as TRANSIENT, the time envelope is also calculated. The time envelope, which represents the temporal energy of the SHB signal, is computed as a set of root mean square (RMS) calculations from each 80 samples of time-domain SHB signal. This results in four time envelope coefficients per frame.

$$t_{rms}(j) = \sqrt{\frac{1}{80}\sum\nolimits_{n=0}^{79}\left(s_{SHB}(80j+n)\right)^2} \qquad j=0,\ldots,3 \tag{825}$$

The time envelope is firstly adjusted by the attenuated value which represents the energy attenuation of the WB signal, and the attenuated value is calculated by the original WB signal $s_{ori}(n)$ and local synthesized WB signal $s_{syn}(n)$:

$$R = \sqrt{\sum\nolimits_{n=0}^{255}\left(s_{syn}(n)\right)^2 \Big/ \sum\nolimits_{n=0}^{255}\left(s_{ori}(n)\right)^2} \tag{826}$$

$$t'_{rms}(j) = \begin{cases} 1.5\cdot R\cdot t_{rms}(j) & if\ R<0.5 \\ t_{rms}(j) & else\ if\ R>1, \qquad j=0,\ldots,3 \\ R\cdot t_{rms}(j) & otherwise \end{cases} \tag{827}$$

In order to highlight the characteristics of the transient signal, the time envelope of the transient signals is modified. A reference sub-frame is first selected from the sub-frames of the input transient signal, which has the maximal amplitude value of envelope compared with values of the envelopes of the rest sub-frames. Then the time envelope of the reference sub-frame is increased whilst at the same time the envelopes of the sub-frames before and after the reference sub-frame are decreased. Then, the adjusted time envelopes of the SHB signal of the current frame are quantized and coded into the bitstream. In order to get better transient effect, in sub-frames before and after the reference sub-frame, the difference between the decreased time envelope and the maximum time envelope is greater than a preset threshold.

– If the sub-frame index idx_{tenv} which is defined as $t'_{rms}(idx_{tenv}) > 5t'_{rms}(idx_{tenv}-1)$ is less than 4, the time envelope is adjusted by:

$$t''_{rms}(j) = \begin{cases} 0.5\cdot t'_{rms}(j) & for\ j<idx_{tenv\_max} \\ 1.00\cdot 5t'_{rms}(j) & for\ j=idx_{tenv\_max} \\ 0.9\cdot t'_{rms}(j) & for\ j>idx_{tenv\_max} \end{cases} \tag{828}$$

where $idx_{tenv\_max} = \arg\max\limits_{j=0,\ldots,3}(t'_{rms}(j))$ is the sub-frame index with the maximum time envelope . It should be noted when the sub-frame index $j>idx_{tenv\_max}$,and when the condition

$\dfrac{1}{3-idx_{tenv\_max}}\sum\nolimits_{k=idx_{tenv\_max}+1}^{3} t'_{rms}(k) < t''_{rms}(idx_{tenv\_max})$ is satisfied, the adjustment of time envelope of sub-

frame $j,\ j>idx_{tenv\_max}$ can be performed.

– For other cases, to obtain the time envelopes of the SHB signal of the current frame used to encode, the adjustment is as follows:

$$\begin{cases} t''_{rms\_temp}(j-1) = 0.5(t'_{rms}(j-1)+t'_{rms}(j)), & if\ t'_{rms}(j-1)>t'_{rms}(j) \\ t''_{rms\_temp}(j) = 0.5(t'_{rms}(j-1)+t'_{rms}(j)), & if\ t'_{rms}(j-1)\leq t'_{rms}(j) \end{cases} \quad for\ j=1,2,3 \tag{829}$$

and

$$t''_{rms}(j) = 0.9t''_{rms\_temp}(j) \qquad for\ j=0,\ldots,3 \tag{830}$$

In addition, when *IsTransient_LF* is *TRUE*, the coder type is INACTIVE and the transient hangover is equal to one, the flag is then set to 1, and the time envelope is adjusted as follows

$$t'''_{rms}(j) = \begin{cases} 0.5t''_{rms}(j) & if\ flag=1 \\ t''_{rms}(j) & if\ flag=0 \end{cases} \quad for\ j=0,\ldots,3 \tag{831}$$

Finally, the values $\bar{t}_{rms}(j) = \log_2(\varepsilon_{rms}+t'''_{rms}(j)), 0\leq j\leq 3$ are further bounded in the range [0,...,15]:
$0\leq\bar{t}_{rms}(j)\leq 15, 0\leq j\leq 3$.

The adjusted time envelopes $\bar{t}_{rms}(j)$ are rounded and quantized with four bits using uniform scalar quantization in the case of TRANSIENT frames.

5.2.6.2.1.7   Bit allocation for FD BWE

Table 69 illustrates the BWE bit allocation for TRANSIENT and Non-TRANSIENT frames.

Table 69: SWB FD BWE bit allocation

Signal class	Signal class bits (F_{class})	Time envelope ($\bar{t}_{rms}(j)$)	Spectral envelope ($f_{rms}(j)$)	Total bits
TRANSIENT	2	16 (=4×4)	13 (=7+6)	31
NON-TRANSIENT	2	0	29 (=5+7+6+5+6)	31

5.2.6.2.2   WB Multi-mode FD Bandwidth Extension

At 13.2kbps, for frame declared as HARMONIC (HM) frame or NORMAL (NM), a bit budget of 6 bits is allocated to the WB Multi-mode FD Bandwidth Extension. One bit is allocated to the signal class and five bits are allocated to encode two spectral envelopes which are calculated by the 80 MDCT coefficients of the higher band (HB) signal, $X_{M\_WB}(k), k = 240,\ldots,319$.

At 7.2kbps or 8kbps, it is blind BWE and no bit budget is allocated. In this case, a two-stage blind BWE is used. In the first stage, the high band frequency generation is the same as the BWE in AMR-WB [9], described in subclauses 6.3.1, 6.3.2.2 and 6.3.3 of [9], and it is added to the ACELP core synthesis. Then the second stage BWE is generated as described in the following sub-clauses, and it is added to the core synthesis with the first stage BWE. At 5.9 kbps VBR coding and CNG coding up to 8.0 kbps, the BWE is also blind with no bit budget allocated, but only the first stage BWE is used.

5.2.6.2.2.1   Windowing and time-to-frequency transformation

The input high-pass filtered signal $s_{HP}(n)$ is delayed by $D_1 - D_2$ samples and windowed to obtain the windowed input signal $\tilde{x}(n)$ as shown in figure 51, where $D_1 - D_2 = 36$.

320-point length MDCT on top of $\tilde{x}(n)$, $n = 0,1,\ldots,319$ is used for WB FD BWE. Refer to subclause 5.3.2.

5.2.6.2.2.2   Frequency domain classification and coding

At 13.2kbps, frequency domain classification is performed. A frequency sharpness parameter is computed to reflect the spectral fluctuation of the frequency coefficients in the higher band signal and those frames are categorized in one of two classes:

a)   HARMONIC: when frequency sharpness is high.

b)   NORMAL: when frequency sharpness is moderate.

The 96 MDCT coefficients in the 5600-8000 Hz frequency range $X_{M\_WB}(k), k = 224,\ldots,319$ are split into three sharpness bands (32 coefficients per band). The frequency sharpness, $\kappa(j)$, is then defined as the ratio of the peak magnitude to the average magnitude in a sharpness band j

$$\kappa(j) = \begin{cases} \dfrac{31 \cdot A_{sharp}(j)}{\sum_{k=224+32j}^{224+32j+31} |X_{M\_C}(k) - A_{sharp}(j)|}, & if \ \sum_{k=224+32j}^{224+32j+31} |X_{M\_C}(k)| \neq A_{sharp}(j) \\ 0, & otherwise \end{cases} \quad j = 0,1,2 \quad (832)$$

where the maximum magnitude of spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is given by

$$A_{sharp}(j) = \max_{k=224+32j,\ldots,224+32j+31} |X_{M\_C}(k)| \qquad j = 0,1,2 \qquad (833)$$

Then, another two sharpness parameters are determined: the maximum sharpness, $\kappa_{\max}$, and the sharpness band counter, c_{sharp}.

The maximum sharpness, $\kappa_{\max}$, in all sharpness bands is computed as:

$$\kappa_{\max} = \max_{j=0,1,2} \left(\kappa(j) \right) \tag{834}$$

The counter c_{sharp} is computed from the three frequency sharpness parameters, $\kappa(j), j = 0,1,2$, and from the three maximum magnitudes, $A_{sharp}(j)$, as follows: Initialized to zero, c_{sharp} is incremented by one for each j if $\kappa(j) > 4.5$ and $A_{sharp}(j) > 8$.

The class of HARMONIC frame is determined from these three sharpness parameters, $\kappa_{\max}$, c_{sharp} and the class of the previous frame, $F_{class}^{[-1]}$.

The threshold of the number of harmonics i_{sharp} and the threshold of the maximum sharpness l_{sharp} are set according to the class of the previous frame $F_{class}^{[-1]}$ and the mode of previous extension layer M_{last_extl}:

$$i_{sharp} = \begin{cases} 1, & if\ F_{class}^{[-1]} = HARMONIC \\ 2, & otherwise \end{cases} \tag{835}$$

$$l_{sharp} = \begin{cases} 10, & if\ M_{last\_extl} = WB\_BWE \\ 5, & else\ if\ F_{class}^{[-1]} = HARMONIC \\ 20, & otherwise \end{cases} \tag{836}$$

Initialize the signal class of the current frame as NORMAL frame (F_{class} = NORMAL).

 – If $c_{sharp} > i_{sharp}$ and $\kappa_{\max} > l_{sharp}$, the current frame is classified as HARMONIC frame (F_{class} = HARMONIC) and the counter for signal class c_{class} is initialized to 0, and incremented by one when c_{class} is less than twelve. Otherwise, c_{class} is decremented by one when c_{class} is larger than zero.

 – If the counter for signal class c_{class} is not less than 2, the current frame is also classified as HARMONIC.

One bit is transmitted for HB signal class coding. Table 70 gives the coded bit for each class.

Table 70: HB signal class coding

Signal class F_{class}	Coded bit
NORMAL	0
HARMONIC	1

The signal class F_{class} of the current frame is preserved as $F_{class}^{[-1]}$ for the next frame.

5.2.6.2.2.3 Spectral envelope calculation and quantization

The spectral envelopes are computed from each 40 samples of frequency-domain HB signal. This results in two spectral envelopes per frame.

$$f_{rms}(j) = \frac{1}{40} \sum_{k=240+40j}^{240+40j+39} \left(X_{M\_WB}(k) \right)^2 \qquad for\ j = 0,1 \tag{837}$$

Then envelope control is performed to prevent too much noise being applied to the reconstructed spectrum. First a spectral copy is created by mapping the frequencies, depending upon the bandwidth, the ACELP coding modes and the FD BWE mode as defined in table 66.

To generate the base excitationexcitation spectrum, the spectral copy is normalized by the sum of its absolute spectral components. The parameter of adaptive normalization length L_{norm} is calculated depending on the original WB MDCT coefficients:

– The 256 WB MDCT coefficients in the 0-6400 Hz frequency range, $\hat{X}_{M\_WB}(k), k = 0,...,255$ are split into 16 sharpness bands (16 coefficients per band). In sharpness band j, if $19A_{sharp}(j) > 4\sum_{k=16j}^{16j+15}\left|\hat{X}_{M\_WB}(k)\right|$ and $A_{sharp}(j) > 10$, the counter C_{band} is incremented by one.

where $j = 0,\ldots,15$, and the maximum magnitude of the spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is:

$$A_{sharp}(j) = \max_{k=16j,\ldots16j+15}\left|\hat{X}_{M\_WB}(k)\right| \qquad j = 0,\ldots,15 \qquad (838)$$

Parameter C_{band} is initialized to 0 and calculated for every frame.

– Then the normalization length L_{norm} is obtained:

$$L_{norm} = \left\lfloor 0.5L_{norm\_pre} + 0.5L_{norm\_cur}\right\rfloor \qquad (839)$$

where the current normalization length L_{norm_cur} is calculated depending on the HB signal class:

$$L_{norm\_cur} = \begin{cases} \left\lfloor 8 + 0.5C_{band}\right\rfloor & if\ mode = NORMAL \\ \max\left(\left\lfloor 32 + 2C_{band}\right\rfloor, 24\right) & if\ mode = HARMONIC \end{cases} \qquad (840)$$

and the current normalization length is preserved as L_{norm_pre} for the next frame.

Then the base excitation spectrum is obtained by:

$$X_{base\_M\_WB}(k) = \frac{X_{M\_WB\_copy}(k)}{f_{rms\_norm}(k)} \qquad (841)$$

where, $X_{M\_WB\_copy}(k),\ 240 \le k < 320$ are the spectral copy coefficients, and the normalized envelope is calculated by:

$$f_{rms\_norm}(k) = \begin{cases} \sum_{j=k-\left\lfloor L_{norm}/2\right\rfloor}^{k+\left\lfloor (L_{norm}-1)/2\right\rfloor}\left|X_{M\_WB\_copy}(j)\right| & 240 \le k < 319 - \left\lfloor L_{norm}/2\right\rfloor \\ \sum_{j=k-\left\lfloor L_{norm}/2\right\rfloor}^{319}\left|X_{M\_WB\_copy}(j)\right| & 319 - \left\lfloor L_{norm}/2\right\rfloor \le k < 320 \end{cases} \qquad (842)$$

The tonality measures used for the energy control are then calculated:

$$Ton_{ec}(j) = \max\left(0.0001, \min\left(10\log\left(\frac{\frac{1}{40}\sum_{k=240+40j}^{240+40j+39}\left|X_M(k)\right|}{\sqrt[40]{\prod_{k=240+40j}^{240+40j+39}\left|X_M(k)\right|}}\right), 5.993\right)\right) \quad for\ j = 0,1 \qquad (843)$$

The ratio between the tonality of the original high frequency spectrum ($Ton_{ec\_org}(j)$) and the tonality of the base excitation spectrum ($Ton_{ec\_base}(j)$) is then calculated as follows:

Void (844)

$$ec\_fac_{FD}(j) = \begin{cases} \max\left(ec_{gam\_FD}, \dfrac{Ton_{ec\_base}(j)}{Ton_{ec\_org}(j)} \right), & if\ Ton_{ec\_base}(j) < 0.75 * Ton_{ec\_org}(j) \\ 1.0, & otherwise \end{cases} \quad for\ j = 0,1$$ (845)

where ec_{gam_FD} is 0.35.

The envelope control factor $ec\_fac_{FD}(j)$ is then applied to the envelope $f_{rms}(j)$:

$$f_{rms\_ec}(j) = f_{rms}(j) * ec\_fac_{FD}(j) \qquad for\ j = 0,1$$ (846)

Then the spectral envelope of log-domain is obtained by:

$$\bar{f}_{rms}(j) = \log_2\left(f_{rms\_ec}(j) \big/ 40 + \varepsilon_{rms} \right) \qquad for\ j = 0,1$$ (847)

Finally, the spectral envelopes $\bar{f}_{rms}(j),\ j = 0,1$ are quantized with a 64-dimentional array $codebook\_HB(j), 0 \le j < 64$ described in table 71.

The distance between the spectral envelopes and the codebook is calculated by:

$$dis(j) = \left(\bar{f}_{rms}(0) - codebook\_HB(2j) \right)^2 + \left(\bar{f}_{rms}(1) - codebook\_HB(2j+1) \right)^2 \qquad 0 \le j < 32$$ (848)

and the index $k_{ind} = \arg \min_{0 \le j < 32} \left(dis(j) \right)$ is encoded with 5 bits.

Table 71: Codebook $codebook\_HB(j)$

j	**0**	**1**	**2**	**3**	**4**	**5**
$codebook\_HB(j)$	1.1606680	0.6594560	-4.9874350	-5.1700310	10.230799	-0.0125740
j	6	7	8	9	10	11
$codebook\_HB(j)$	10.605126	9.7910260	-0.3739880	-0.6027910	6.2753817	0.3307670
j	12	13	14	15	16	17
$codebook\_HB(j)$	9.4537100	8.8558020	2.9320890	2.1643160	3.1332030	2.9710870
j	18	19	20	21	22	23
$codebook\_HB(j)$	8.061906	-0.5905290	15.754963	5.0496380	17.227070	18.329395
j	24	25	26	27	28	29
$codebook\_HB(j)$	-2.4710190	-3.1725330	-1.4136470	-1.9457110	15.147771	14.506490
j	30	31	32	33	34	35
$codebook\_HB(j)$	11.358370	11.714662	9.4275510	-0.1223030	7.0970160	-1.5805260
j	36	37	38	39	40	41
$codebook\_HB(j)$	12.498663	3.1614850	10.349261	1.5185040	5.3809850	-1.7341900
j	42	43	44	45	46	47
$codebook\_HB(j)$	1.1224600	-2.2397020	12.362551	12.133788	4.2788690	-1.7729040
j	48	49	50	51	52	53
$codebook\_HB(j)$	6.1577130	5.4971410	3.3243130	-2.5710470	19.097071	9.3576920
j	54	55	56	57	58	59
$codebook\_HB(j)$	7.6509204	7.4404626	0.5055090	-3.7073090	18.584702	11.302494
j	60	61	62	63		
$codebook\_HB(j)$	18.706564	18.308905	23.010420	22.915377		

5.2.6.2.2.4   Bit allocation for FD BWE

Table 72 illustrates the WB BWE bit allocation.

Table 72: WB FD BWE bit allocation

Signal class	Signal class bit (F_{class})	Spectral envelope ($\bar{f}_{rms}(j)$)	Total bits
HARMONIC	1	5	6
NORMAL	1	5	6

5.2.6.3   Coding of upper band at 64 kb/s

The SWB, resp. FB, signal at 64 kbps bit-rate is coded in two bands. The lower band that covers 0-8kHz is coded using the LP-based coding at 16 kHz internal sampling rate as described earlier and the upper band that extends the coded band-width up to 16 kHz, resp. 20 kHz, is coded using a high-rate upper band coding. The same upper band coding with a fixed 16 kbps bit-budget is used in all GC, TC and IC frames. This bit-budget can be eventually increased with unused bits coming from the AVQ within the combined algebraic codebook.

The upper band coding is mostly done in the MDCT domain and has two modes: normal and transient. While normal mode is used in most of generic and voiced frames, the use of transient mode minimizes the pre-echo and post-echo in frames where the signal at the frame beginning is significantly different from the signal at the frame end, e.g. onsets. Detection of transient frames is done in time domain using a detector described in .

First the input signal, filtered by the HP filter and sampled at 32 or 48 kHz, $s_{HP}(n)$ is transformed using MDCT and OLA function. In normal mode, the whole frame is transformed at once while in transient mode the frame is divided into four sub-frames and thus four sets of spectral coefficients $X(k)$ are present. In both modes only spectrum coefficients between 7.6 kHz and 14.4 kHz are encoded. While the spectrum between 7.6 kHz and 14.4 kHz is divided in normal mode into four bands of 1.7 kHz width each, it is divided into two bands of 3.4 kHz each in transient mode.

The other frequency coefficients are zeroed. Consequently spectral coefficients $X(k)$, $k = 304,...,575$, are encoded in normal mode frames and spectral coefficients $X(k)$, $k = 76,...,143$, are encoded in transient mode frames.

5.2.6.3.1   Coding in normal mode

In normal mode frames, the global gain g_{global} is computed on the spectrum 7.6 kHz – 14.4 kHz as follows

$$g_{global} = \frac{\sqrt{\sum_{k=304}^{575} X_M(k) \cdot X_M(k)}}{272} \tag{849}$$

and quantized using a 5-bit log gain quantizer at the range of [3.0; 500.0].

The quantized global gain $\hat{g}_{global}$ is further used to normalize the spectrum $X_M(k)$ resulting in a normalized spectrum by the quantized global gain.

$$X_{M\_norm}(k_{start} + k) = X_M(k_{start} + k) \big/ \hat{g}_{global} \qquad 0 \le k < 272 \tag{850}$$

where k_{start} is the start frequency bin of spectrum reconstruction and $k_{start} = 304$ for normal frames.

Then the spectrum envelope is computed in four bands which results in 68 spectral coefficients per band.

$$f_{env\_band}(j) = \sqrt{\frac{1}{68} \sum_{k=68j}^{68j+67} \left(X_{M\_norm}(k_{start} + k)\right)^2} \qquad 0 \le j < 4 \tag{851}$$

The spectrum envelope is quantized using two two-dimensional VQs by means of 6 bits codebook $CB_{6bits}(k)$ and 5 bits codebook $CB_{5bits}(k)$ in table 73 and table 74, respectively

Table 73: 6 bits spectral envelope VQ codebook

k	$CB_{6bits}(k)$		k	$CB_{6bits}(k)$		k	$CB_{6bits}(k)$	
0	0.044983	0.0417	22	8.919388	9.762914	44	15.26931	21.53914
1	0.524276	0.469365	23	11.29932	11.7639	45	16.98352	24.69959
2	0.671757	0.605513	24	11.78222	5.879754	46	19.59173	22.68968
3	0.983501	0.855093	25	14.05046	9.665228	47	20.1462	25.88847
4	1.227874	1.1322	26	11.20153	9.001128	48	17.79742	19.45312
5	1.672212	1.432704	27	14.43475	13.23657	49	21.29062	20.18658
6	2.548211	2.361091	28	14.33726	3.904411	50	24.09732	19.08672
7	3.196961	3.306999	29	20.07105	4.335061	51	23.61309	22.54586
8	2.580753	5.217478	30	18.10581	8.223599	52	23.68201	16.32824
9	4.207751	7.243802	31	22.35229	9.603263	53	26.88655	19.40244
10	3.517157	1.738487	32	7.242756	16.56449	54	26.00977	15.63221
11	4.381567	2.753657	33	11.77753	19.16765	55	28.93993	16.24062
12	4.758266	4.696094	34	11.1218	15.45598	56	25.09448	12.36642
13	6.827988	6.106459	35	14.56358	17.35957	57	27.71338	13.26328
14	4.450459	10.13121	36	17.82122	11.89472	58	28.33095	10.32926
15	7.256045	12.48804	37	17.46603	15.29606	59	30.63283	12.85128
16	6.70872	1.953339	38	21.33696	13.45518	60	25.2738	6.138124
17	6.60403	3.69956	39	20.54434	17.12537	61	29.19534	7.222413
18	10.61273	2.537916	40	9.056358	22.33831	62	32.17132	5.019567
19	9.387467	4.241173	41	11.23842	28.83252	63	31.979	9.473855
20	7.119045	8.281485	42	13.26273	25.14338			
21	9.062854	7.086526	43	16.24356	28.25685			

Table 74: 5 bits spectral envelope VQ codebook

k	$CB_{5bits}(k)$		k	$CB_{5bits}(k)$	
0	0.512539	0.472507	16	13.67263	5.457414
1	1.338963	1.108591	17	16.47199	3.917684
2	2.544041	1.759765	18	20.91033	6.43281
3	3.124053	3.045299	19	25.45733	8.61722
4	4.892713	3.721097	20	16.4107	7.574456
5	4.010297	5.750862	21	18.57439	10.2915
6	5.111215	2.164709	22	22.08876	12.51216
7	6.667518	3.893404	23	21.17053	17.20871
8	8.454117	2.75143	24	5.276107	9.62247
9	11.12357	3.518174	25	9.093585	11.27469
10	6.622948	5.960704	26	11.94566	15.53814
11	8.562429	5.003579	27	16.55041	15.04656
12	8.919363	7.784057	28	6.358148	17.5474
13	10.75904	5.959438	29	13.31662	21.76552
14	12.44919	8.359519	30	7.646096	26.10672
15	13.67701	11.23058	31	2.451297	31.9331

The quantized spectrum envelope $\hat{f}_{env\_band}(j)$ is used to further normalize the spectrum resulting in spectrum normalized per bands.

$$X_{M\_norm}(k_{start}+68\cdot j+k) = X_{M\_norm}(k_{start}+68\cdot j+k)\big/\hat{f}_{env\_band}(j) \qquad 0 \le j < 4,\, 0 \le k < 68 \quad (852)$$

Afterwards, the number of the bands to be quantized is calculated according to the total bits and the saturated threshold, and the bands are selected according to the quantized spectrum envelopes. Once the bands are selected, the first stage encoding is processed by means of AVQ. If there are at least 14 remaining bits after the first stage encoding and the first stage quantized spectrum is non-zero, a second stage encoding is employed also by means of AVQ.

Calculate the number of the bands to be quantized according to total bits R_{tot} and the saturated threshold $Thr1 = 400$, and select the bands to be quantized according to the quantized envelopes as follows:

– If $R_{tot} > 400$, the selected bands are by:

$$X_{M\_temp}(k) = X_{M\_norm}(k_{start} + k) \quad 0 \le k < 272 \tag{853}$$

– Otherwise, the selected bands are by:

$$\begin{cases} X_{M\_temp}(k) = X_{M\_norm}(k_{start} + k) & 0 \le k < k_{ind1} \\ X_{M\_temp}(k + k_{ind1}) = X_{M\_norm}(k_{start} + k_{ind2} + k) & 0 \le k < 272 - k_{ind2} \end{cases} \tag{854}$$

where the k_{ind1} and k_{ind2} are set as follows:

$$k_{ind1} = 64 \cdot pos_{en\_\min} + 8 \cdot \lfloor pos_{en\_\min} / 2 \rfloor \tag{855}$$

$$k_{ind2} = 64 \cdot (pos_{en\_\min} + 1) + 8 \cdot \lfloor (pos_{en\_\min} + 1) / 2 \rfloor \tag{856}$$

where $pos_{en\_\min}$ is the sub-band index with the minimum envelope , and it is calculated by:

$$pos_{en\_\min} = \arg\min_{j=0,\dots,3}(\hat{f}_{env\_band}(j)) \tag{857}$$

The number of the sub-bands N_{sv} is obtained according to the number of the total bits R_{tot} and the saturated threshold $Thr1 = 400$ as follows:

$$N_{sv} = \begin{cases} 34 & if \ R_{tot} > 400 \\ (272 + K_{ind1} - K_{ind2})/8 & otherwise \end{cases} \tag{858}$$

Quantize the normalized coefficients of the selected bands (N_{sv} sub-bands) by AVQ to obtain the quantized normalized coefficients.

The envelope of the spectrum between 14.4 kHz and 16 kHz in SWB is predicted by:

$$f_{env\_non\_cod} = 0.5 \cdot \hat{f}_{env\_band}(pos_{\min}) \tag{859}$$

And the envelope of the spectrum between 14.4 kHz and 20 kHz in FB is calculated and quantized as follows:

$$f_{env\_non\_cod} = \sqrt{\frac{1}{L_{non\_cod}} \sum_{k=0}^{L_{non\_cod}-1} (X_M(k_{end} + k))^2 + \varepsilon} \tag{860}$$

where k_{end} is the start frequency bin of spectrum reconstruction and $k_{end} = 576$ for normal frames. L_{non_cod} is the width of the MDCT coefficients between 14.4 kHz and 20 kHz in FB, and is set by: $L_{non\_cod} = 224$.

The ratio of the envelopes R_{env} is refined:

$$R_{env} = f_{env\_non\_cod} \Big/ \left(\hat{g}_{global} \cdot \hat{f}_{env\_band}(pos_{\min}) \right) \tag{861}$$

And the index of attenuation factor I_{att} is obtained according to the ratio of the envelopes, and the envelope of the spectrum between 14.4 kHz and 20 kHz in FB is finally obtained according to the attenuation factor I_{att}:

$$
I_{att} = \begin{cases} 1 & if \ R_{env} < 0.5 \\ 2 & else \ if \ R_{env} > 1.2 \\ 3 & else \ if \ R_{env} > 0.8 \\ 0 & otherwise \end{cases} \tag{862}
$$

$$
f_{env\_non\_cod} = \begin{cases} 0.25 \cdot \hat{f}_{env\_band}(pos_{\min}) & if \ I_{att} = 1 \\ 1.2 \cdot \hat{f}_{env\_band}(pos_{\min}) & else \ if \ I_{att} = 2 \\ 0.8 \cdot \hat{f}_{env\_band}(pos_{\min}) & else \ if \ I_{att} = 3 \\ 0.5 \cdot \hat{f}_{env\_band}(pos_{\min}) & otherwise \end{cases} \tag{863}
$$

Then the index I_{att} is encoded with 2bits.

 If the number of the remaining bits R_{rem} after the first stage encoding is larger than 14, then the second stage encoding is needed. Select N_{sv2} sub-bands from the first stage selected N_{sv} sub-bands to perform the second stage encoding, and the number of the sub-bands N_{sv2} is calculated according to the number of remaining bits R_{rem} and the saturated threshold $Thr2 = 12$.

The input coefficients $X_{M\_spec}(k), \ 0 \le k < 8 \cdot N_{sv}$ of the second stage encoding are obtained by reordering the differences between the original normalized coefficients $X_{M\_temp}(k)$ and the quantized normalized coefficients $\hat{X}_{norm}(k)$ as follows:

First, in the sub-bands j with the AVQ codebook index $nq(j) = 0$,

$$
X_{M\_spec}(8 \cdot k + l) = X_{M\_temp}(8 \cdot j + l), \ 0 \le l < 8, \ 0 \le j < N_{sv} \tag{864}
$$

where the index k is initialized to 0, and is incremented by 1 when $nq(j) = 0$.

Then, in the sub-bands $j = 0$, if $nq(j) != 0$, $X_{M\_spec}(8 \cdot k + l) = X_{M\_temp}(8 \cdot j + l) - \hat{X}_{norm}(8 \cdot j + l), \ 0 \le l < 8$, and the index k is incremented by 1.

The number of the sub-bands N_{sv2} is calculated according to the number of the remaining bits R_{rem} and the saturated threshold $Thr2 = 12$:

$$
N_{sv2} = \begin{cases} 33 & if \ R_{rem} > 396 \\ R_{rem}/12 & otherwise \end{cases} \tag{865}
$$

The coefficients of the first N_{sv2} sub-bands in $X_{M\_spec}(k), \ 0 \le k < 8 \cdot N_{sv}$ are selected to perform the second stage encoding as follows:

The global gain of second stage encoding $g_{global2}$ is calculated and quantized as $\hat{g}_{global2}$.

$$
g_{global2} = 16 \cdot \sqrt{\frac{1}{8 \cdot N_{sv2}} \sum_{k=0}^{8 \cdot N_{sv2}} \left(X_{M\_spec}(k) \right)^2 + 0.001} \tag{866}
$$

and the spectrum is normalized with $\hat{g}_{global2}$ as follows:

$$
\tilde{X}_{M\_spec}(k) = \frac{1}{0.0625 \cdot \hat{g}_{global2}} X_{M\_spec}(k) \ \ 0 \le k < 8 \cdot N_{sv2} \tag{867}
$$

Then the normalized spectrum is quantized by AVQ.

5.2.6.3.2 Coding in transient mode

In transient mode frames, a similar procedure as in normal mode frames is employed and the following descriptions focus on the differences.

The total bit-budget is divided by four in order to obtain a bit-budget for every sub-frame and the encoding is performed four times (once for every sub-frame). In sub-frame j, $0 \le j < 4$, the global gain g_{global} is computed on the spectrum 7.6 kHz – 14.4 kHz using equation (849) for $k = 76, ..., 143$ and quantized using a 5-bit log gain quantizer at the range of $[3.0, 500.0]$.

$$g_{global} = \sqrt{\frac{1}{68} \sum_{k=0}^{67} \left(X_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) \right)^2 + 0.001} \tag{868}$$

where k_{start} is the start frequency bin of spectrum reconstruction and $k_{start} = 76$ for transient frames. L_{fr} is the frame length. $L_{fr} = 640$ for SWB signal and $L_{fr} = 960$ for FB signal

The spectrum envelope is computed in two bands for each sub-frame, thus each band contains 34 spectral coefficients.

Then, the spectral envelope of normalized spectrum is calculated:

$$f_{env\_band}(i, j) = \sqrt{\frac{1}{34} \sum_{k=34i}^{34i+33} \left(X_{M\_norm}(k_{start} + j \cdot L_{fr}/4 + k) \right)^2 + 0.001} \quad i = 0,1 \quad j = 0,1,2,3 \tag{869}$$

where,

$$X_{M\_norm}(k_{start} + j \cdot L_{fr}/4 + k) = \frac{1}{\hat{g}_{global}} \cdot X_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) \quad 0 \le k < 68 \tag{870}$$

The total 8 spectral envelopes in 4 sub-frame are divided into 4 two-dimensional vectors, i.e., $f_{env\_band}(i, j)$, $i = 0,1$, $j = 0,1,2,3$ in each sub-frame are combined as a vector, and quantized using two-dimensional VQs. For the first vector, the spectral envelopes in the first sub-frame $f_{env\_band}(i, 0)$ are quantized using one two-dimensional VQ by means of 4 bits codebook $CB_{4bits}(I_{env}(j))$ defined in table 75.

Table 75: 4 bit spectral envelope VQ codebook

$I_{env}(j)$	$CB_{4bits}(I_{env}(j))$	
0	0.799219	0.677609
1	1.754571	1.215689
2	2.846222	2.017775
3	4.379336	1.975914
4	5.935472	2.945818
5	3.938621	4.220399
6	8.080808	2.632276
7	7.579771	4.986835
8	4.956485	10.36366
9	7.739148	8.652471
10	9.238397	7.051655
11	10.205707	5.619638
12	10.645117	4.374648
13	11.66018	3.474015
14	10.845836	2.664596
15	11.724073	1.637023

The index of this VQs is noted as $I_{env}(0)$. This 4 bits codebook can be divided into two 3 bits codebook. The first 3 bits codebook is $0 \leq I_{env}(j) < 8$, and the second is $8 \leq I_{env}(j) < 15$ in the table 75. Then the first quantized vector is determined one of the 3 bits codebook according to the $I_{env}(0)$.

If $I_{env}(0) < 8$, the first 3 bits codebook is selected as new codebook;

if $I_{env}(0) \geq 8$, the second 3 bits codebook is selected as new codebook.

Then the following three vectors are quantized by using the 3 bits codebook determined before.

The quantized spectral envelope $\hat{f}_{env\_band}(i)$ is applied to the spectrum:

$$X_{M\_norm}(k_{start} + j \cdot L_{fr}/4 + k) = \frac{1}{\hat{f}_{env\_band}(i,j)} \cdot X_{M\_norm}(k_{start} + j \cdot L_{fr}/4 + k) \qquad (871)$$

where $i = 0,1$, $j = 0,1,2,3$ and $34 \cdot i \leq k < 34 \cdot (i+1)$.

Then, the normalized spectrum is quantized by AVQ.

If $M_{extl} = SWB\_BWE\_HIGHRATE$, the envelope of the spectrum between 14.4 kHz and 16 kHz in SWB is predicted by:

$$f_{env\_non\_cod}(j) = \hat{f}_{env\_band}(1,j) \qquad j = 0,1,2,3 \qquad (872)$$

If $M_{extl} = FB\_BWE\_HIGHRATE$, the envelope of the spectrum between 14.4 kHz and 20 kHz in FB is calculated and quantized as follows:

$$f_{env\_non\_cod}(j) = \sqrt{\frac{1}{L_{non\_cod}} \sum_{k=0}^{L_{non\_cod}-1} \left(X_{M\_spec}\left(k_{end} + j \cdot L_{fr}/4 + k\right)\right)^2 + \varepsilon} \qquad (873)$$

where, k_{end} is the start frequency bin of spectrum reconstruction and $k_{end} = 144$ for transient frames. L_{non_cod} is the width of the MDCT coefficients between 14.4 kHz and 20 kHz in FB, and is set by $L_{non\_cod} = 56$.

The ratio of the envelopes $R_{env}(j)$ is refined as:

$$R_{env}(j) = f_{env\_non\_cod}(j) \Big/ \left(\hat{g}_{global} \cdot \hat{f}_{env\_band}(1, j) \right) \qquad j = 0,1,2,3 \qquad (874)$$

And the index of attenuation factor $I_{att}(j)$ is obtained according to the ratio of the envelopes, and the envelope of the spectrum between 14.4 kHz and 20 kHz in FB is finally obtained according to the attenuation factor $I_{att}(j)$:

$$I_{att}(j) = \begin{cases} 1 & \textit{if } \ f_{env\_non\_cod}(j) < 0.1 \\ 2 & \textit{else if } \ f_{env\_non\_cod}(j) < 0.3 \\ 3 & \textit{else if } \ f_{env\_non\_cod}(j) < 0.5 \\ 0 & \textit{otherwise} \end{cases} \qquad (875)$$

$$f_{env\_non\_cod}(j) = \begin{cases} 0.1 \cdot \hat{f}_{env\_band}(1, j) & \textit{if } \ I_{att}(j) = 1 \\ 0.3 \cdot \hat{f}_{env\_band}(1, j) & \textit{else if } \ I_{att}(j) = 2 \\ 0.5 \cdot \hat{f}_{env\_band}(1, j) & \textit{else if } \ I_{att}(j) = 3 \\ \hat{f}_{env\_band}(1, j) & \textit{otherwise} \end{cases} \qquad (876)$$

where $j = 0,1,2,3$. Then the index I_{att} is encoded with 2bits.

Finally, the unused AVQ bits from the current sub-frame are employed in the subsequent sub-frame within the same frame.

5.3 MDCT Coding Mode

5.3.1 General description

The decision algorithm for selecting the MDCT Coding mode, instead of CELP, is described in subclause 5.1.16. In general, the MDCT coding mode is selected whenever a signal is not speech or the bit rate for encoding is high enough for the transform coder. It always operates on full input signal independent of sample rate and selected audio bandwidth.

The EVS codec supports two major MDCT coding modes, the MDCT based TCX (TCX) and the High Quality MDCT coder (HQ). Both operate with different configurations depending on the bit rate for encoding. For some operating points, the MDCT coders are switched for optimized efficiency. The two MDCT selector algorithms are described in subclauses 5.1.14.2 and 5.1.14.3. The following table describes the used mode configurations depending on bit rate and bandwidth.

Table 76: Overview of supported MDCT coding modes

Bitrate (kbps)	Bandwidth	MDCT Classifier	MDCT mode
7.2, 8	NB		HQ low-rate
9.6	NB, WB,SWB		TCX low-rate
13.2	NB, WB, SWB	MDCT Selector	Switching HQ low-rate and TCX mid-rate
16.4	NB, WB, SWB FB		
24.4	NB, WB		TCX mid-rate
32	WB		HQ high-rate
24.4	SWB,FB	MDCT Selector II	Switching HQ high-rate and TCX mid-rate
32	SWB,FB		
48	WB,SWB,FB		TCX high-rate
64	WB,SWB,FB		HQ high-rate
96,128	WB,SWB,FB		TCX high-rate

The next subclauses describe the common time-to-frequency transformation for both major MDCT modes. In the following the TCX and the HQ coder are described including their modules and configurations.

5.3.2 Time-to-frequency transformations

5.3.2.1 Transform sizes and MDCT configurations

The MDCT encoder operates with a frame of length L at the input sampling frequency f_s.

5.3.2.2 Long block transformation (ALDO window)

The window used in long block transformation is an asymmetrical low delay optimized (ALDO) window. This ALDO window is stored in ROM in two versions, one at 48 kHz and another at 25.6 kHz respectively; the ALDO window at other input sampling frequencies (namely 8, 12.8, 16, 32 kHz) is obtained by on-the-fly decimation in the folding process described in subclause 5.3.2.2.1.

The ALDO window has a time support of 40 ms and its definition is the same at 48 and 25.6 kHz. To simplify notations the sampling frequency (48 or 25.6 kHz) associated with the ALDO window is not included in the following text. This window, $h_a(n)$, $n = 0,\ldots,2L-1$ of length $2L$, is structured in 4 segments:

$$h_a(n) = \begin{cases} w_1(n), & n = 0,\ldots,L-L_z-1 \\ w_2(n-L+L_z), & n = L-L_z,\ldots,L+L_z-1 \\ w_3(n-L-L_z), & n = L+L_z,\ldots,2L-L_z-1 \\ w_4(n-2L+L_z), & n = 2L-L_z-1,\ldots,2L-1 \end{cases} \tag{877}$$

where L is the frame length (20 ms) and L_z is the length of the last segment with a weight of 0 (5.625 ms). The different segments consist of an increasing segment $w_1(n)$, a constant segment $w_2(n)$ with a weight of 1, a decreasing segment $w_3(n)$, a constant segment $w_4(n)$ with a weight of 0, as illustrated in figure 53. Note that in practice the parts corresponding to weights of 1 and 0 are not stored explicitly.

Figure 53 is also useful to illustrate the time alignment of input signal frames at the MDCT encoder. As the fourth segment of the ALDO window has by definition a weight of 0, the frame of new input samples, $x(n)$, $n = 0,\ldots,L-1$, is time-aligned in such a way that its end coincides with the end of the third ALDO window segment $w_3(n)$; this alignment allows to save L_z samples of lookahead (5.625 ms); the previous frame of input signal, $x_{OLD}(n)$, $n = 0,\ldots,L-1$, is partially weighted by the first ALDO window segment, $w_1(n)$.
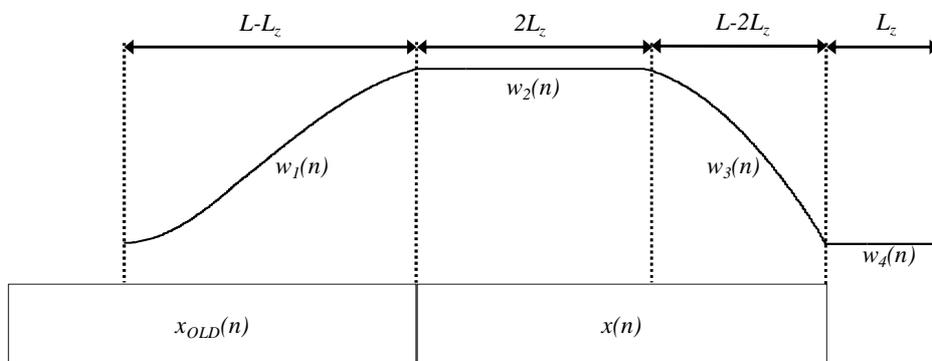
Figure 53: ALDO window

The ALDO window was obtained at 48 and 25.6 kHz by a two-step process: first an initial window $h_{init}(n)$, $n = 0,\ldots,2L-1$ was obtained, then this initial window $h_{init}(n)$ was regularized in order to ensure perfect reconstruction. The initial window $h_{init}(n)$ is defined in 4 segments with an increasing segment $W_1(n)$, a constant segment $W_2(n)$ with a weight of 1 identical to $w_2(n)$, a decreasing segment $W_3(n)$, a constant segment $W_4(n)$ with a weight of 0 identical to $w_4(n)$. The ALDO window is therefore defined by spelling out the initial window $h_{init}(n)$ and the regularization term $\Delta(n)$.

ALDO window at 48 kHz

At 48 kHz, the 20 ms frame has $L = 960$ samples. The first segment $w_1(n)$ of the ALDO window is given by

$$w_1(n) = \frac{W_1(n)}{\Delta(n)}, \quad n = 0,\ldots,L-L_z \tag{878}$$

where the first segment of the initial window is

$$W_1(n) = \sin\left(\frac{\pi}{2}\frac{n+1}{R_1}\right)^{C_1} \tag{879}$$

and the regularization factor is

$$\Delta(n) = \sqrt{h_{init}(n)h_{init}(2L-n-1) + h_{init}(n+L)h_{init}(L-n-1)}, \tag{880}$$

$R_1 = L - L_z$ is the length of the first segment (690 samples at 48 kHz), $C_1 = 1.946594238281250$ is a constant. The second segment of the ALDO window is $w_2(n) = 1, n = 0, \ldots, 2L_z - 1$. The third segment $w_3(n)$ of the ALDO window is given by

$$w_3(n) = \frac{W_3(n)}{\Delta(n)}, n = 0, \ldots, L - 2L_z \tag{881}$$

where the third segment of the initial window is

$$W_3(n) = \cos\left(\frac{\pi}{2} \frac{n}{R_2}\right)^{C_2} \tag{882}$$

and the regularization factor is

$$\Delta(n) = \sqrt{h_{init}(n)h_{init}(2L - n - 1) + h_{init}(n + L)h_{init}(L - n - 1)} , \tag{883}$$

$R_2 = L - 2L_z$ is the length of the first segment (420 samples at 48 kHz), $C_2 = 0.942810058593750$ is a constant. The fourth segment of the ALDO window is $w_4(n) = 0, n = 0, \ldots, L_z - 1$ (270 sample at 48 kHz).

ALDO window at 25.6 kHz

The ALDO window at 25.6 kHz is defined as the ALDO window at 48 kHz, except the following parameters are used:

- $L = 512$

- $L_z = 144$

- $C_1 = 1.941650390625$

- $C_2 = 0.943359375$

It follows that $R_1 = 368$ and $R_2 = 224$.

ALDO window at other sampling frequencies

The ALDO window at 48 kHz is used to derive on the fly the ALDO window at 8, 16, and 32 kHz. Similarly, the ALDO window at 25.6 kHz is used to derive on the fly the ALDO window at 12.8 kHz. Details on this on-the-fly decimation are provided in subclause 5.3.2.2.1.

5.3.2.2.1 Folding and on-the-fly window decimation

The folding operations and window decimation (when applicable) are combined in the same process. To achieve perfect reconstruction ALDO windows the 48 and 25.6 kHz are irregularly decimated.

The folding process is illustrated in figure 54. The MDCT folding is performed by dividing the 40 ms time support of the ALDO window in 4 sections delimited by dotted vertical lines.
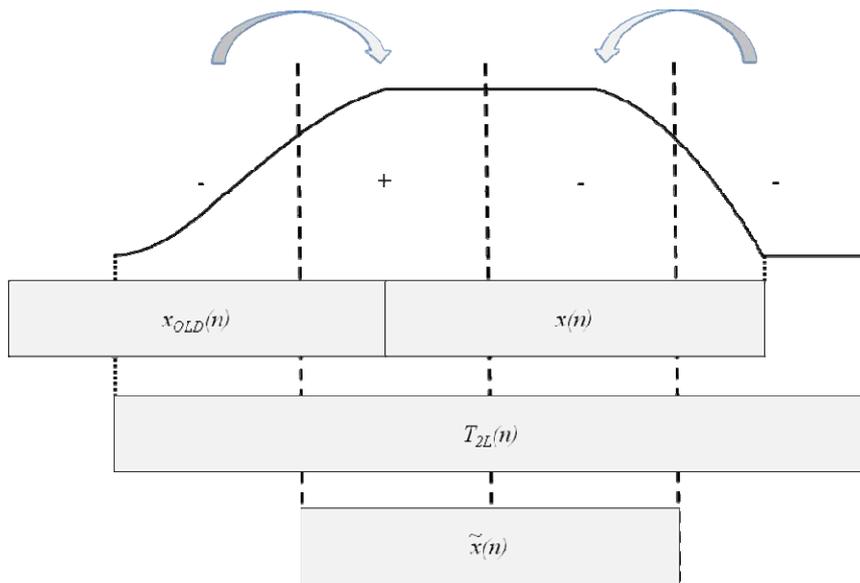
Figure 54: Folding with ALDO window.

For each frame of new input samples, a block T_{2L} of length $2L$ is folded into a block $\tilde{x}(n)$, $n = 0, \ldots, L-1$. The ALDO window at 48 or 25.6 kHz is defined at a sampling frequency corresponding to two frames of length N at 48 or 25.6 kHz ($N \geq L$). The ratio between N and L is called the decimation factor.

The folded frame is given for $n = 0, \ldots, L/2 - 1$ by:

$$
\begin{aligned}
\tilde{x}(n) = &-T_{2L}\left(\frac{3L}{2} - n - 1\right) h_a\left(\left[\frac{3N}{2} - \frac{(n+1)N}{L}\right] + d\right) \\
&- T_{2L}\left(\frac{3L}{2} + n\right) h_a\left(\left[\frac{3N}{2} - 1 + \frac{(n+1)N}{L}\right] - d\right)
\end{aligned}
\tag{884}
$$

$$
\tilde{x}\left(\frac{L}{2} + n\right) = T_{2L}(n) h_a\left(\left[n\frac{N}{L}\right] + d\right) - T_{2L}(L - n - 1) h_a\left(\left[N - 1 - n\frac{N}{L}\right] - d\right),
\tag{885}
$$

where the ratio N/L is the decimation factor and d is an offset that depends on the decimation factor. The ALDO window used for decimation, decimation factor and offset are given in table 77.

Table 77: ALDO window used for decimation, decimation factor and offset parameters.

MDCT core sampling frequency (kHz)	ALDO window used for decimation (kHz)	Decimation factor $d_f = N/L$	Offset d
8	48	6	2
12.8	25.6	2	0
16	48	3	1
25.6	25.6	1	0
32	48	3	1
48	48	1	0

The 32 kHz sampling frequency is a special case, as the ratio between 48 and 32 kHz is not integer. For this 32 kHz case, in order to achieve perfect reconstruction, the ALDO window $h_a(n)$ decimated from 48 to 16 kHz is applied to samples with even indices, the samples with odd indices are weighted by a complementary window h_{comp}. For the 32 kHz case, the folded frame is given for $n = 0, \ldots, L/4 - 1$ by:

$$\tilde{x}(2n) = -T_{2L}\left(\frac{3L}{2} - 2n - 1\right)h_{comp}\left(\frac{3N_{16}}{2} - n - 1\right) - T_{2L}\left(\frac{3L}{2} + 2n\right)h_{comp}\left(\frac{3N_{16}}{2} + n\right), \tag{886}$$

$$\begin{aligned}\tilde{x}(2n+1) &= -T_{2L}\left(\frac{3L}{2} - (2n+1) - 1\right)h_a\left(\left(\frac{3N_{16}}{2} - n - 1\right)d_f + d\right) \\ &\quad - T_{2L}\left(\frac{3L}{2} + 2n + 1\right)h_a\left(\left(\frac{3N_{16}}{2} + n + 1\right)d_f - d - 1\right)\end{aligned}, \tag{887}$$

$$\tilde{x}\left(\frac{L}{2} + 2n\right) = T_{2L}(2n)h_a\left(nd_f + d\right) - T_{2L}(L - 2n - 1)h_a\left((N_{16} - n)d_f - 1 - d\right), \tag{888}$$

$$\tilde{x}\left(\frac{L}{2} + 2n + 1\right) = T_{2L}(2n+1)h_{comp}(n) - T_{2L}(L - (2n+1) - 1)h_{comp}(N_{16} - n - 1), \tag{889}$$

where N_{16} is the frame length at 16kHz, L is the length of MDCT core frame at 32kHz, $d_f = 3$ is the decimation factor and $d = 1$ is the offset.

The complementary window $h_{comp}(n)$ of size $2N_{16}$ is stored in ROM. To obtain the window $h_{comp}(n)$, the ALDO window at 48 kHz is decimated by 3 (with an offset of 1), the resulting decimated window at 16 kHz is considered as if it was obtained from a 32 kHz window decimated by 2 (without any offset) so that half of the samples of the 32 kHz window is known; the other half is obtained doing a linear interpolation between known samples. The combination of the 16 kHz window obtained by decimation and $h_{comp}(n)$ is such that perfect reconstruction is ensured for the overall 32 kHz window.

5.3.2.2.2 eDCT

The $\tilde{x}(n)$, $n = 0, 1, \ldots, L-1$ are transformed to frequency domain by an eDCT, which is built upon a discrete cosine transform type IV (DCT$_{IV}$) but the eDCT requires less storage and has lower complexity.

The original L-point DCT$_{IV}$ formula is:

$$X_M(k) = \sum_{n=0}^{L-1}\tilde{x}(n)\cos\left[\left(n + \frac{1}{2}\right)\left(k + \frac{1}{2}\right)\frac{\pi}{L}\right], \quad k = 0, 1, \ldots, L-1 \tag{890}$$

where $\tilde{x}(n)$ is the windowed input signal of the current audio frame and $X_M(k)$ is the k-th DCT spectral component. This formula can be rewritten as:

$$\begin{cases} X_M(2q) = \text{Re}\{\overline{Z}(q)\} \\ X_M(L - 1 - 2q) = -\text{Im}\{\overline{Z}(q)\} \end{cases}, \quad q = 0, 1, \ldots, L/2 - 1 \tag{891}$$

where the values $\overline{Z}(q)$ are given by

$$\overline{Z}(q) = W_{8L}^{-3} \cdot W_{4L}^{2q+1} \sum_{p=0}^{L/2-1}\left\{z(p) \cdot W_{4L}^{2p+1}\right\}W_{L/2}^{pq}, \quad q = 0, 1, \ldots, L/2 - 1 \tag{892}$$

and $z(p) = \tilde{x}(2p) + j\tilde{x}(L - 1 - 2p)$, $p = 0, 1, \ldots, L/2 - 1$, and $W_N = e^{-j\frac{2\pi}{N}} = \cos(\frac{2\pi}{N}) - j\sin(\frac{2\pi}{N})$.

Hence, the eDCT is computed using the following steps which lead to less storage and lower complexity:

  1) Pre-processing

Apply the twiddle factors to the time domain data $\tilde{x}(n)$, so as to obtain twiddled signal $z(p)$:

$$z(p) = \tilde{x}(2p) + j \cdot \tilde{x}(L-1-2p), \quad p = 0,1,\ldots,L/2-1 \tag{893}$$

Pre-rotate the twiddled signal $z(p)$ by using a symmetric rotation factor:

$$a \cdot W_{4L}^{2p+1}, \quad p = 0,1,\ldots,L/2-1$$

where W_{4L}^{2p+1} in the rotation factor may also be expressed in the following form:

$$W_{4L}^{2p+1} = \cos\frac{2\pi(2p+1)}{4L} - j\sin\frac{2\pi(2p+1)}{4L}, \quad p = 0,1,\ldots,L/2-1 \tag{894}$$

which satisfies conditions of $\cos\left(2\pi(2(L/2 -1-q)+1)/4L\right) = \cos\left(2\pi(L - (2q+1))/4L\right) = \sin\left(2\pi(2q+1)/4L\right)$,

$q = 0,1,\ldots,L/4-1$ and $\sin\left(2\pi(2(L/2 -1-q)+1)/4L\right) = \sin\left(2\pi(L - (2q+1))/4L\right) = \cos\left(2\pi(2q+1)/4L\right)$,

$q = 0,1,\ldots,L/4-1$, and therefore, in the implementation, only one data table of cosine values covering L/2-

points $\sqrt[4]{2/L} \cdot 1/\left(1+(3\pi/4L)^2\right) \cdot \cos\left(2\pi(2p+1)/4L\right)$, $p = 0,1,\ldots,L/2-1$ needs to be stored, and

$a = \sqrt[4]{2/L} \cdot 1/\left(1+(3\pi/4L)^2\right)$ is a constant..

2) Fast Fourier Transform

Perform a Fast Fourier Transform (FFT) of L/2 points on the pre-rotated data $z(p)$.

3) Perform an in-place fixed rotate compensation

The FFT data $\overline{Z}(q)$, $q = 0,1,\ldots,L/2-1$ of $z(p)$ is rotated in-place by multiplying with a fixed rotate compensation

factor $1 + j\dfrac{3\pi}{4L}$:

$$\overline{Z}(q) = \overline{Z}(q) \cdot (1 + j\frac{3\pi}{4L}), \quad q = 0,1,\ldots,L/2-1 \tag{895}$$

4) Post-processing

The data $\overline{Z}(q)$, $q = 0,1,\ldots,L/2-1$ is finally post-rotation processed with a symmetric rotation factor:

$$b \cdot W_{4L}^{2q+1}, \quad q = 0,1,\ldots,L/2-1,$$

where W_{4L}^{2q+1} in the rotation factor may also be expressed in the following form:

$$W_{4L}^{2q+1} = \cos\frac{2\pi(2q+1)}{4L} - j\sin\frac{2\pi(2q+1)}{4L}, \quad q = 0,1,\ldots,L/2-1 \tag{896}$$

In the specific implementation, only one cosine data table of L/2- values needs to be stored i.e.

$\sqrt[4]{2/L} \cdot 1/\left(1+(3\pi/4L)^2\right) \cdot \cos\left(2\pi(2q+1)/4L\right)$, $q = 0,1,\ldots,L/2-1$ which is the same as the pre-processing, and

$b = \sqrt[4]{2/L} \cdot 1/\left(1+(3\pi/4L)^2\right)$ is a constant.

5) Obtain frequency domain data

Then the real parts of the post- rotated data are expressed as $X_M(2q)$, $q = 0,1,\ldots,L/2-1$, which represent the odd number frequency bins of the frequency domain data; and the frequency reversed imaginary parts of the post-rotated data are expressed as $X_M(L-1-2q)$, $q = 0,1,\ldots,L/2-1$, which represent the even number frequency bins of the frequency domain data.

$$\begin{cases} X_M(2q) = \mathrm{Re}\{\overline{Z}(q)\} \\ X_M(L-1-2q) = -\mathrm{Im}\{\overline{Z}(q)\} \end{cases}, \quad q = 0,1,\ldots,L/2-1 \tag{897}$$

where $\overline{Z}(q) = (1+j3\pi/4L) \cdot W_{4L}^{2q+1} \sum_{p=0}^{L/2-1} \left\{ z(p) \cdot W_{4L}^{2p+1} \right\} W_{L/2}^{pq}$, $q = 0,1,\ldots,L/2-1$.

5.3.2.3 Transient location dependent overlap and transform length

Beside ALDO window, 3 more overlap shapes are used for windowing:

- FULL: 8.75 milliseconds overlap described in subclause 5.3.2.5, used for transition from transform with long ALDO window to short transform

- HALF: 3.75 milliseconds symmetric sine overlap

- MINIMAL: 1.25 milliseconds symmetric sine overlap

For long MDCT based TCX (TCX20) transformation, that is not after ACELP, 9 combinations of one overlap shape (ALDO, HALF, MINIMAL) on the left side of the window and another overlap shape (ALDO, HALF, MINIMAL) on the right side of the window is possible. For HQ MDCT 4 combinations of one overlap shape (ALDO, FULL, HALF, MINIMAL) on the left side of the window and ALDO on the right side of the window is possible.

For short MDCT based TCX transformation (TCX10 or TCX5), 7 of the 9 possible combinations of one symmetric overlap shape (FULL, HALF, MINIMAL) on the left side of the window and another symmetric overlap shape (FULL, HALF, MINIMAL) on the right side of the window are used (see table 80).

The overlap length and the transform length of the TCX are dependent on the existence of a transient and its location and are chosen so that a transient is mostly contained in only one window as shown in table 78.

Table 78: Coding of the overlap and the transform length based on the transient position

Attack index	Overlap with the first window of the following frame	Short/Long Transform decision (binary coded) 0 – Long, 1 - Short	Binary code for the overlap width	Overlap code
none	ALDO	0	0	00
-2	FULL	1	0	10
-1	FULL	1	0	10
0	FULL	1	0	10
1	FULL	1	0	10
2	MINIMAL	1	10	110
3	HALF	1	11	111
4	HALF	1	11	111
5	MINIMAL	1	10	110
6	MINIMAL	0	10	010
7	HALF	0	11	011

If the transient detector described in the subclause 5.1.8 does not detect a transient, but there is increase of energy at high frequencies in the current frame relative to the previous frame ($E_{HF}(i)$ is the high frequency energy for the current frame and $E_{HF}(i-1)$ is the high frequency energy for the previous frame, as defined in subclause 5.1.2.2):

$$E_{HF}(i) > 39 E_{HF}(i-1) \tag{898}$$

then the short transform is chosen and overlap is dependent on the attack index set in the transient detector or half overlap is chosen if the transient detector didn't set attack index. The transient detector described in the subclause 5.1.8 basically returns the index of the last attack with the restriction that if there are multiple transients then MINIMAL overlap is preferred over HALF overlap which is preferred over FULL overlap. If an attack at position 2 or 6 is not strong enough then HALF overlap is chosen instead of the MINIMAL overlap.

For bitrates below 48 kbps the short transform length is not allowed and thus the attack index lower than 6 would trigger ALDO window and the overlap code 00.

If the previous frame was coded using ACELP a specific configuration is used as described in subclause 5.4.2.2.

Depending on the right overlap (current overlap code) and the left overlap (previous overlap code) in the current frame the configuration of the transform lengths in the current frame is chosen as presented in table 79.

Table 79: Transform lengths decision table

Previous overlap code / Current overlap code	00/10	111/011	110/010
00	TCX20	TCX20	TCX20
10	2xTCX5, TCX10	2xTCX5, TCX10	2xTCX5, TCX10
111	2xTCX5, TCX10	4xTCX5	4xTCX5
110	2xTCX5, TCX10	4xTCX5	4xTCX5
010	TCX20	TCX20	TCX20
011	TCX20	TCX20	TCX20

If the current frame is classified as the transition frame (current overlap code is 10, 111 or 110) the detailed configuration of the transform and overlap lengths is presented in table 80.

Table 80: Transform and overlap length configuration for transient frame

Prev. code / Curr. code	00/10	111/011	110/010
10	FULL,TCX5,MIN.,TCX5, MIN.,TCX10,FULL	HALF,TCX5,MIN.,TCX5, HALF,TCX10,FULL	MIN.,TCX5,MIN.,TCX5, MIN.,TCX10,FULL
111	FULL,TCX10,HALF,TCX5, MIN.,TCX5,HALF	HALF,TCX5,MIN.,TCX5,HALF, TCX5,MIN.,TCX5,HALF	MIN.,TCX5,MIN.,TCX5,MIN., TCX5,MIN.,TCX5,HALF
110	FULL,TCX10,MIN.,TCX5, MIN.,TCX5,MIN.	HALF,TCX5,MIN.,TCX5,MIN., TCX5,MIN.,TCX5,MIN.	MIN.,TCX5,MIN.,TCX5,MIN., TCX5,MIN.,TCX5,MIN.

In table 79 and table 80, the value 00/10 for previous overlap code indicates that the same configuration is used if the previous overlap code is 00 as well as when it is 10. The meaning of values 111/011 and 110/010 follows the same logic.

As example "HALF, TCX5, MIN., TCX5, HALF., TCX10, FULL" from Table 80 indicates that first TCX5 has HALF overlap on the left side and MINIMAL overlap on the right side, followed by second TCX5 with MINIMAL overlap on the left side and HALF overlap on the right side, followed by TCX10 with HALF overlap on the left side and FULL overlap on the right side. The TCX10 with HALF overlap on the left side and FULL overlap on the right is one example of the window described in subclause 5.3.2.5.2.

5.3.2.4 Short block transformation

5.3.2.4.1 Short window transform in TDA domain

5.3.2.4.1.1 Transient detection
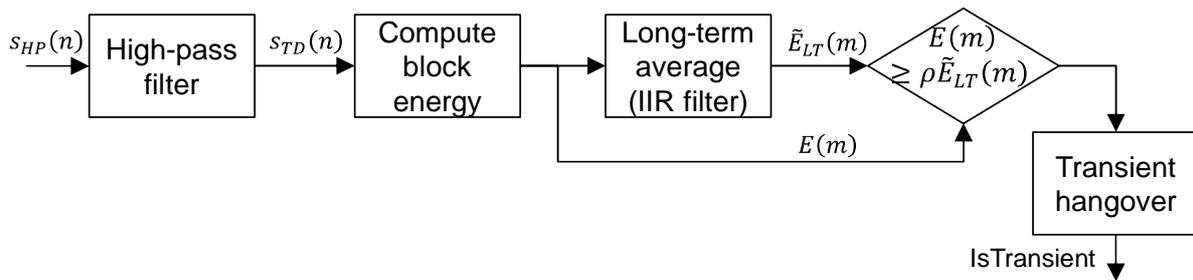
Figure 55: Transient detection algorithm

The block diagram of the transient detection algorithm is shown in figure 55. The input signal $s_{HP}(n)$ is first high-pass filtered; the high-pass filter serves as a precaution against undesired low-frequency components. A first order IIR filter is used, and it is given by:

$$H_{TD}(z) = \frac{0.7466(1 - z^{-1})}{1 - 0.4931 z^{-1}} \tag{899}$$

The same filter is used for all sampling frequencies.

The output of the high-pass filter $s_{TD}(n)$ is obtained according to:

$$s_{TD}(n) = 0.4931\, s_{TD}(n-1) + 0.7466\, s_{HP}(n) - 0.7466\, s_{HP}(n-1), \qquad n = 0, \ldots, L-1 \tag{900}$$

where L denotes the 20-ms frame length. The high-pass filtered signal $x_{TD}(n)$ is sectioned into four sub-frames, each corresponding to 5 ms, $L/4$ samples each.

The energy of each sub-frame, $E_{TD}(m)$, is computed according to:

$$E_{TD}(m) = \frac{1}{L/4} \sum_{n=mL/4}^{(m+1)L/4-1} s_{TD}(n)^2, \qquad m = 0, \ldots, 3 \tag{901}$$

The signal's long-term energy corresponding to each sub-frame, $\tilde{E}_{TD}(m)$, is updated according to the following equation:

$$\tilde{E}_{TD}(m) = (1 - \alpha)\tilde{E}_{TD}(m-1) + \alpha\, E_{TD}(m), \quad m = 0, \ldots, 3 \tag{902}$$

In the above equation, the forgetting factor α is set to 0.25, and the convention that $\tilde{E}_{TD}(-1) = \tilde{E}_{TD}(3)$ from the previous frame is used. The high-pass filter, $H_{TD}(z)$, states as well as $E_{TD}(3)$ are saved for processing in the next frame.

During switching, when core has changed or extension layer has changed, then $E_{TD}(3)$ is initialized with the energy of $s_{TD}(n)$.

For each sub-frame m, a comparison between the short-term energy $E_{TD}(m)$ and the long-term energy $\tilde{E}_{TD}(m)$ is performed. A transient is detected whenever the energy ratio is above a certain threshold. Formally, a transient is detected whenever:

$$E_{TD}(m) \geq \rho \times \tilde{E}_{TD}(m) \tag{903}$$

where ρ is the energy ratio threshold and is set according to table 81.

Table 81: Threshold table for transient detector

Extension layer	Condition	Threshold
Generic Mode	Inactive	10
	Not Inactive	13.5
Otherwise	SWB, $rate \leq 16.4\,\mathrm{kb/s}$	
	Otherwise	6

For the first coded frame with extension layer using the Generic mode then the transient detection, after Equation 903, is set to 0.

In general, the time-frequency transform is applied on a 40-ms frame; therefore, a transient will affect two consecutive frames. To overcome this, a hangover for detected transient is used. A transient detected at a certain frame will also trigger a transient at the next frame.

The output of the transient detector is a flag, denoted *IsTransient*. The flag is set to the logical value *TRUE* if a transient is detected, or *FALSE* otherwise.

If there is a change in extension layer, then the hangover and filter memories are reset before the start of the transient detector.
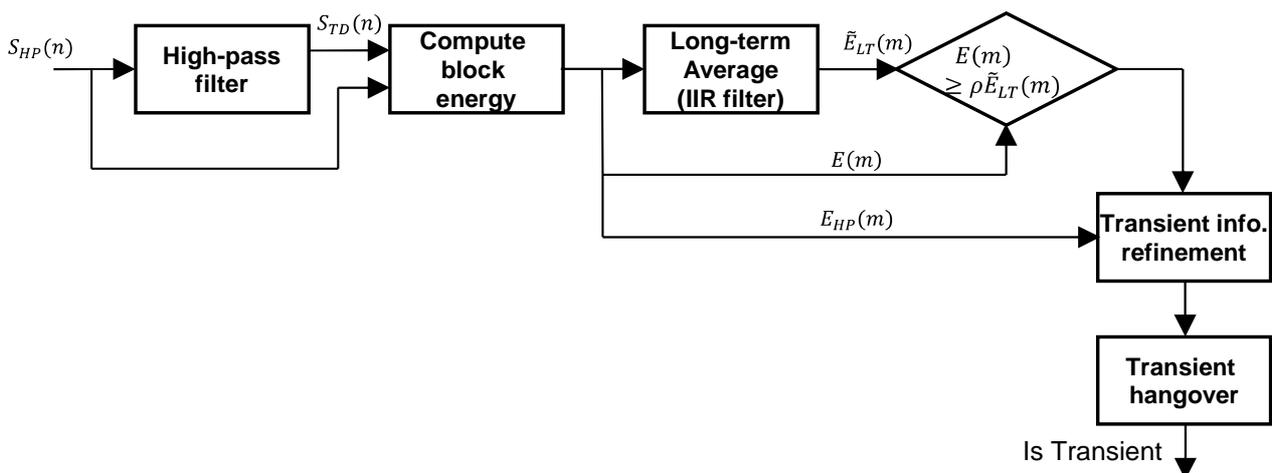
Figure 56: Transient detection algorithm for NB

The block diagram of the transient detection algorithm for NB is shown in figure 56. Most of algorithm is same as that of WB and SWB. In the *Compute block energy* block, the energy of each sub-frame for $S_{HP}(n)$, $E_{HP}(m)$, is computed according to:

$$E_{HP}(m+1) = \frac{1}{L/4} \sum_{n=mL/4}^{(m+1)L/4-1} s_{HP}(n)^2, \qquad m=0,\ldots,3 \qquad (904)$$

where the $E_{HP}(0)$ is the energy of the 4th sub-frame in the previous frame.

The *Transient information refinement* block performs an additional verification and checks whether the current frame that has been determined as a transient frame is truly a transient frame. This is to prevent a transient determination error occurring due to the high-pass filtering block removing energy in a low frequency. The operation of the *Transient information refinement* block is described for the case where the current frame is detected to be transient.
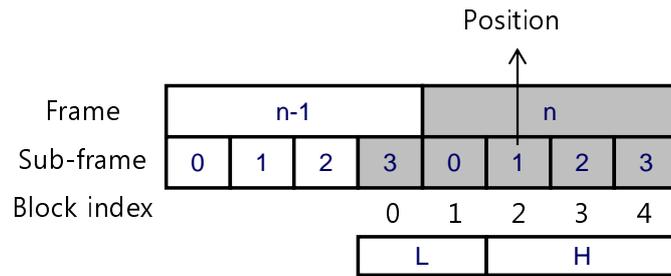
Figure 57: Transient information refinement for NB

The position which is detected as a transient is one of 4 sub-frames in frame *n*. The number of blocks included in the region L and the number of blocks included in the second region H will vary depending which of these 4 sub-frames is detected as the transient, as represented in the figure 57.

First, the ratio of the average of the short-term energy in the region H (E_high) to the average of the short-term energy in the region L (E_low) is calculated. Region H includes the block which was previously detected to be transient and blocks existing thereafter. Region L includes the blocks prior to region H.

Next, the ratio of the short-term energy of the frame *n* before the high pass filtering (E_in) to the short-term energy of the frame *n* after the high pass filtering (E_out) is calculated.

If these ratios meet the following conditions, frame *n* is determined to the normal frame instead of a transient frame.

```
if( ((E_high/E_low<2.0f)&&(E_high/E_low>0.7f)) && ((E_in/E_out)>Thres) )
     IsTransient = 0;
```

5.3.2.4.1.2   Short window transform

When the flag IsTransient is set then the transform is switched from a long transform to a series of short transforms as is depicted in figure 58. This subclause describes the short transform.
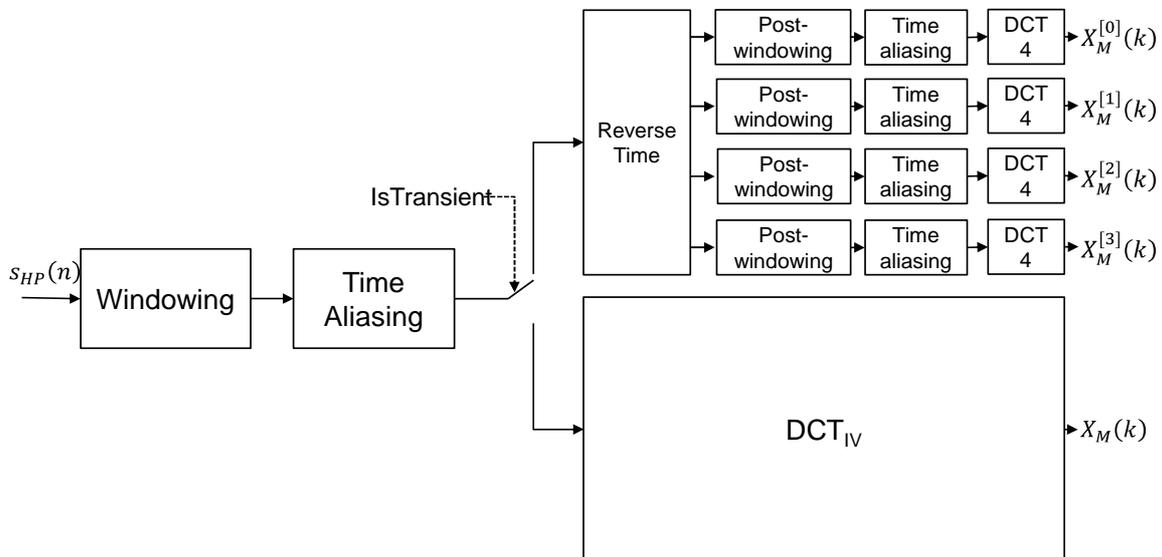
Figure 58: Adaptive time-frequency transform

Windowing and time-domain aliasing are done as described in subclause 5.3.2.2. A reordering of the time-domain aliased signal is performed. In order to keep the temporal coherence of the input signal, the output of the time-domain aliasing operation needs to be reordered before further processing. The ordering operation is necessary, without ordering the basis functions of the resulting filter-bank will have an incoherent time and frequency responses. The reordering is done according to equation 905, and consists of shuffling the upper and lower half of the TDA output signal $x_{TDA}(n)$.

$$\tilde{x}_{TDA}(n) = x_{TDA}(L-1-n), \quad n = 0,\ldots,L-1 \tag{905}$$

This reordering is only conceptual and in reality no computations are involved. Higher time resolution is obtained by zero-padding the signal $\tilde{x}_{TDA}(n)$ and dividing the resulting signal into four overlapped equal length sub-frames. The amount of zero-padding is equal to $L/8$ on each side of the signal. The segments are 50% overlapped and each segment has a length equal to $L/2$. The two inner segments are post-windowed using a sine window of length $L/2$. The windows for outer segments are constructed using half a sine window.

This operation is depicted in figure 59; each resulting post-windowed segment is further processed by applying the modified discrete cosine transform (MDCT) (i.e., time aliasing + DCT$_{IV}$). The output of the MDCT for each segment represents the signal spectrum at different time instants, thus, in case of transients, a higher time resolution is used.

The length of the output of each of the four MDCTs is half the length of the input segment, i.e., $L/4$, therefore this operation does not add additional redundancy.
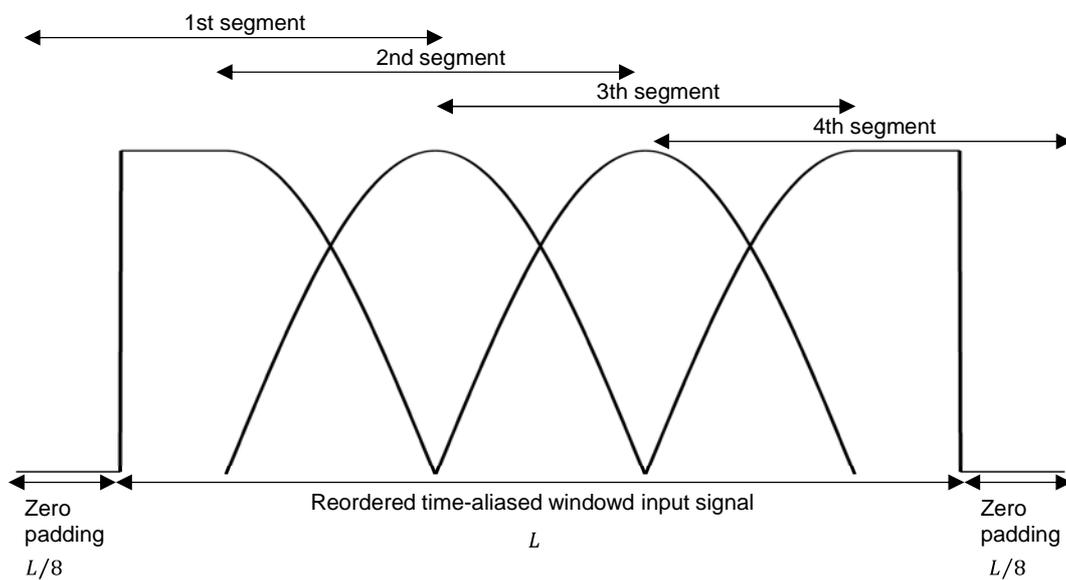
Figure 59: Higher time resolution, division into four segments

5.3.2.4.1.3   Interleaving

After the short transform described in subclause 5.3.2.4.1.2, then the coefficients of the equivalent four 5-ms transforms are interleaved. Table 82 shows the band lengths used for interleaving. Coefficients of interleave band length are read from the first transform, $X_M^{[0]}(k)$, then the second transform, $X_M^{[1]}(k)$, and so on until the fourth transform. This is then repeated for all interleave bands. The interleaved bands are written to the vector $X_M(k)$.

Table 82: Band widths for transient mode interleaving

Band	0	1	2	3	4	5	6	7	8
WB	16	16	16	16	16				
SWB	16	16	16	16	24	24	24	24	
FB	16	16	16	16	24	24	24	32	32

5.3.2.4.2   Short window transform for MDCT based TCX

After choosing the transform and overlap length configuration for transient frame as described in subclause 5.3.2.3 each sub-frame (TCX5 or TCX10) is windowed and transformed using the MDCT, which is implemented using TDA and DCT$_{IV}$.

After the TNS described in subclause 5.3.3.2.2, the MDCT bins of 2 TCX5 sub-frames are interleaved:

$$X_M(2k) = X_M^1(k),$$

$$X_M(2k+1) = X_M^2(k), \quad k = 0,...,\frac{L}{4}-1 \tag{906}$$

5.3.2.5 Special window transitions

The overlap mode FULL is used for transitions between long ALDO windows and short symmetric windows (HALF, MINIMAL) for short block configurations (TCX10, TCX5) described in subclause 5.3.2.3. The transition window is derived from the ALDO and sine window overlaps.

5.3.2.5.1 ALDO to short transition

The left overlap of the transition window has a length of 8.75ms (FULL overlap) and is derived from the ALDO window. The left slope of the ALDO analysis window (long slope) is first shortened to 8.75ms by removing the first 5.625ms. Then the first 1.25ms of the remaining slope are multiplied with the 1.25ms MINIMAL overlap slope to smooth the edge. The resulting 8.75ms slope is depicted in figure 60.

For the right part of the transition window HALF or MINIMAL overlap is used.
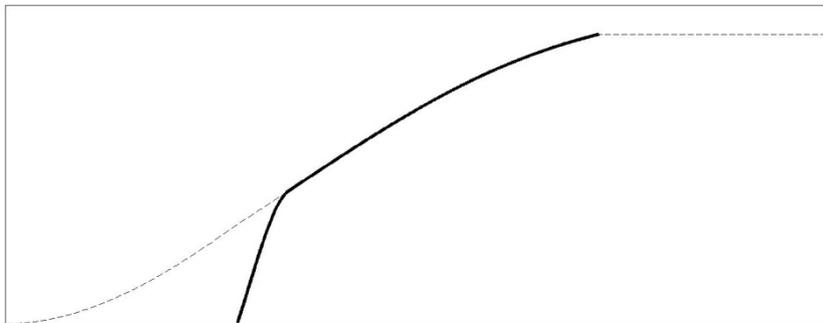
Figure 60: Window shape transition ADLO to Short

5.3.2.5.2 Short to ALDO transition

HALF or MINIMAL overlap is used for the left part of the transition window.

For the right part of the transition window the right slope of the ALDO analysis window (short slope) is used, which has a length of 8.75ms (see figure 61).
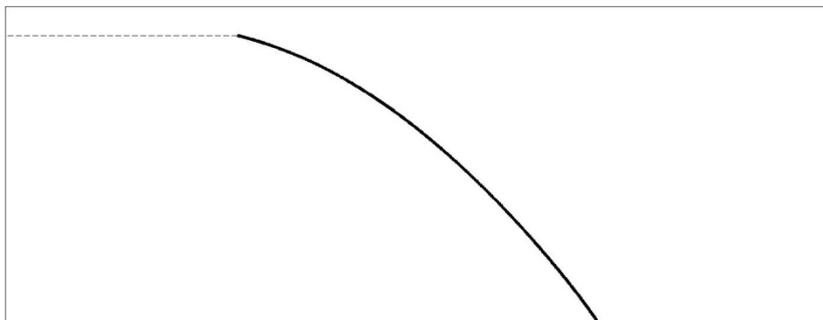
Figure 61: Window shape transition Short to ALDO

5.3.2.6 Modified Discrete Sine Transform

The MDST is computed in the same way as the MDCT defined in the previous subclauses, with two exceptions:

- The TDAC equation is described by:

$$\tilde{\mathrm{x}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & J_{L/2} & -I_{L/2} \\ I_{L/2} & J_{L/2} & 0 & 0 \end{bmatrix} x_w \tag{907}$$

- The DCT$_{\text{IV}}$ given by equation (890) is replaced by a DST$_{\text{IV}}$:

$$X_S(k) = \sum_{n=0}^{L-1} \tilde{x}(n) \sin\left[\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{L}\right], \quad k = 0,\ldots,L-1 \tag{908}$$

The MDST transformation always follows the MDCT parameters such as transform size or windowing.

5.3.3 MDCT based TCX

5.3.3.1 General description

5.3.3.1.1 High level overview

The MDCT based TCX (TCX) mode codes the MDCT spectrum using an LPC scheme for noise shaping. The LPC parameters are estimated in time domain and applied in the MDCT domain. The basic mode consists of a 20ms MDCT transformation (TCX20). For higher rates, smaller transform sizes for 10ms (TCX10) and 5ms (TCX5) are supported as well. The TCX mode provides several coding tools such as adaptive low frequency emphasis, temporal noise shaping, intelligent gap filling or noise filling to improve coding efficiency. The spectral data is finally quantized by a uniform quantizer with dead-zone and noiseless codec by an arithmetic coder module with several modes.

For the MDCT based TCX, the total length of the MDCT spectrum $L_{TCX}^{(full)}$ depends on the input sampling rate, the TCX mode and the coding mode of the previous frame. Some of the coding tools used in TCX only work on the lower MDCT bins corresponding to the CELP frequency range, as LP analysis is performed on the CELP sampling rate. This number of bins is designated by $L_{TCX}^{(celp)}$. The maximum number of bins $L_{TCX}^{(bw)}$ encoded by TCX is determined by the coding bandwidth. If $L_{TCX}^{(full)}$ exceeds $L_{TCX}^{(bw)}$, the remaining bins are filled with zeroes.

The following table defines $L_{TCX}^{(full)}$, $L_{TCX}^{(celp)}$ and $L_{TCX}^{(bw)}$:

Table 83: Overview frame sizes of MDCT-based TCX

TCX mode		TCX20		TCX10/TCX5
Previous coding mode		MDCT	CELP	
$L_{TCX}^{(full)}$		$sr_{inp}/50$	$\frac{5}{4}\cdot sr_{inp}/50$	$sr_{inp}/100$
$L_{TCX}^{(celp)}$		$sr_{celp}/50$	$\frac{5}{4}\cdot sr_{celp}/50$	$sr_{celp}/100$
$L_{TCX}^{(bw)}$	NB	160	200	80
	WB	320	400	160
	SWB	640	800	320
	FB	960	1200	480

5.3.3.1.2 Rate dependent configuration

The MDCT-based TCX contains different setups depending on the bit rate for encoding. In general, the setup consists of three configurations: Low-rate, mid-rate and high-rate configuration. The following table lists the main tools used in the various configurations.

Table 84: Overview configurations MDCT-based TCX

Bitrate [kbps]	Bandwidth	Specific configurations
9.6 (Low-rate)	NB,WB,SWB	low-rate LPC, envelope based arithmetic coder, ALFE 1, TCX20
13.2 - 32 (Mid-rate)	NB, WB, SWB,FB	mid-rate LPC, context based arithmetic coder, ALFE 2, TCX20
48 – 128 (High-rate)	WB, SWB,FB	high-rate LPC, context based arithmetic coder, ALFE 1, TCX20/10/5

One additional bit rate dependent tool is temporal noise shaping (TNS). TNS is activated only for bit rate 24.4 kbps and higher.

5.3.3.2 General encoding procedure

5.3.3.2.1 LPC parameter calculation

5.3.3.2.1.1 Low rate LPC

MDCT based TCX relies on smoothed LPC spectral envelope. Analysis methods are identical to that of ACELP. Quantization of LSF is also common to ACELP except for 9.6 kbps (NB/WB/SWB). Quantization of weighted LSF and conversion of LSF used for 9.6 kbps (NB/WB/SWB) are described in the following subclauses.

5.3.3.2.1.1.1 Quantization of weighted LSF

For MDCT based TCX at 9.6 kbps (NB/WB), envelope based arithmetic coding and low rate quantization of weighted LSF are used. After normal LPC analysis and interpolation in LSP domain, perceptual weighting is applied to the prediction coefficients to get weighted prediction coefficients $\tilde{a}_i$ as

$$\tilde{a}(i) = a(i)\gamma^i, \quad (i = 1,...m) . \tag{909}$$

Weighted prediction coefficients $\tilde{a}(i)$ are converted to LSF vector $\mathbf{q}_\gamma = \left(q_\gamma(0), q_\gamma(1), \cdots, q_\gamma(m-1)\right)^T$ and the LSF vector is quantized with two stage or three stage VQ. These VQ codebooks are used in two ways.

First and primary use of the quantization is intended to represent the weighted LPC envelope to shape the MDCT coefficients with MA inter frame prediction. Secondary use is to tell the estimate of envelope to the arithmetic coding without depending on MA prediction.

In case of primary encoding, input weighted LSF vector $\mathbf{q}_\gamma$ is subtracted by the mean vector $\mathbf{m}$ and the MA predicted contribution vector $\mathbf{q}_{MA}$ to generate the input of two stage VQ $\mathbf{q}_{VQ}$, .

$$\mathbf{q}_{VQ} = \mathbf{q}_\gamma - \mathbf{m} - \mathbf{q}_{MA} . \tag{910}$$

For the first stage VQ, 5 bits for 16 samples are allocated. For the second stage VQ, 4 bits are assigned to the lower 6 LSF parameters and 4 bits for the higher 10 parameters. A quantization criterion is to minimize the weighted distortion between input, $\mathbf{q}_{VQ}$ and output of quantized LSF vectors, $\hat{\mathbf{q}}_{VQ}$. The reconstructed LSF vector after adding mean vector and the MA predicted contribution vector, $\hat{\mathbf{q}}_\gamma (= \hat{\mathbf{q}}_{VQ} + \mathbf{m} + \mathbf{q}_{MA})$ can be converted to the weighted envelope representing $W_{MA}(z) = A_{MA}(z/\gamma)$ and used for noise shaping of MDCT domain transform coefficients.

In the secondary encoding and decoding, the same LSF vector from the VQ table at the primary encoding, $\hat{\mathbf{q}}_{VQ}$ is retrieved and mean vector is added to reconstruct weighted LSF vector without MA prediction $\hat{\mathbf{q}}'_\gamma (= \hat{\mathbf{q}}_{VQ} + \mathbf{m})$. This can inform the shape of envelope for the envelope base arithmetic coding even when the decoder cannot get the information from the previous frame.

Envelope shape could be distorted due to the lack of MA prediction and a third stage VQ is used to compensate the distortion only when necessary. Compensation is preferable when large spectral envelope or large distortion in spectral envelope is expected. In order to determine the necessity of the third stage VQ, the first and the second lower values of

$\hat{\mathbf{q}}'_\gamma$, namely, $\hat{q}'_\gamma(0)$ and $\hat{q}'_\gamma(1)$ are checked. If $\hat{q}'_\gamma(0)$ or $(\hat{q}'_\gamma(1) - \hat{q}'_\gamma(0))$ have smaller values than the threshold, namely expected envelope values and distortion are larger than threshold, the third VQ with 2 bits is applied to $\hat{q}'_\gamma(0)$ and $\hat{q}'_\gamma(1)$ for the purpose of corrections. The criteria of the selection of the retrieved vector at the third stage VQ, $\hat{\mathbf{q}}_{3VQ}$ is to minimize the weighted distortion between input $\mathbf{q}_\gamma$ and a final reconstructed LSF vector without MA prediction $\hat{\mathbf{q}}''_\gamma (= \hat{\mathbf{q}}_{3VQ} + \hat{\mathbf{q}}_{VQ} + \mathbf{m})$. If both $\hat{q}'_\gamma(0)$ and $(\hat{q}'_\gamma(1) - \hat{q}'_\gamma(0))$ values are equal to or larger than threshold, the third VQ is skipped and $\hat{\mathbf{q}}'_\gamma$ is used as a final reconstruction LSF vector.

Reconstructed LSF vector $\hat{\mathbf{q}}'_\gamma$ is corresponding to the weighted envelope $A_{NOMA}(z/\gamma)$ without depending on MA prediction. In envelope based arithmetic coding, unweighted LSF vector without depending on MA prediction $\hat{\mathbf{q}}'_1$ is also necessary to estimate the MDCT envelope. This can be obtained by low-complex direct matrix conversion described in the next subclause.

5.3.3.2.1.1.2 Direct conversion of LSF

For the interpolation of LSF to the LSF in the possible ACELP at the next frame, the reconstructed LSF vector $\hat{\mathbf{q}}_\gamma$ with MA prediction needs to be converted to unweighted domain LSF vector $\hat{\mathbf{q}}_1$. Similarly, reconstructed LSF vector without MA prediction $\hat{\mathbf{q}}'_\gamma$ needs to be converted to get unweighted domain LSF $\hat{\mathbf{q}}'_1$ to assist the envelope base arithmetic coding.

In order to get unweighted LSF $\mathbf{q}_1 = (q_1(0), q_1(1), \cdots, q_1(m-1))^T$ corresponding to $\gamma = 1.0$, weighted LSF $\mathbf{q}_\gamma = (q_\gamma(0), q_\gamma(1), \cdots, q_\gamma(m-1))^T$ is converted by using matrix K' and equally spaced LSF $\mathbf{q}_0 = (\pi/(m+1), 2\pi/(m+1), \cdots, m\pi/(m+1))^T$ as follows.

$$\mathbf{q}_1 = (1-\gamma)K'(\mathbf{q}_\gamma - \mathbf{q}_0) + \mathbf{q}_\gamma \tag{911}$$

$$\begin{pmatrix} q_1(0) \\ q_1(1) \\ \vdots \\ \vdots \\ q_1(m-1) \end{pmatrix} = K \begin{pmatrix} q_\gamma(0) - \dfrac{\pi}{m+1} \\ q_\gamma(1) - \dfrac{2\pi}{m+1} \\ \vdots \\ \vdots \\ q_\gamma(m-1) - \dfrac{m\pi}{m+1} \end{pmatrix} + \begin{pmatrix} q_\gamma(0) \\ q_\gamma(1) \\ \vdots \\ \vdots \\ q_\gamma(m-1) \end{pmatrix} \tag{912}$$

$K(=(1-\gamma)K')$ has non-zero elements only in the diagonal position and the adjacent samples as follows and the non-zero elements are shown in table 85.

$$K = \begin{pmatrix} g_{1,1} & g_{2,1} & & & & 0 \\ g_{2,1} & g_{2,2} & g_{2,3} & & & \\ & g_{3,2} & g_{3,3} & g_{3,4} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \\ 0 & & & & g_{m,m-1} & g_{m,m} \end{pmatrix} \tag{913}$$

Table 85: Non-zero elements of conversion matrices

i	from $\gamma = 0.92$ to 1.0 (12.8 kHz sample)			from $\gamma = 0.94$ to 1.0 (16 kHz sample)		
	$g_{i,i-1}$	$g_{i,i}$	$g_{i,i+1}$	$g_{i,i-1}$	$g_{i,i}$	$g_{i,i+1}$
1	-	1.19764	-0.59173	-	0.78925	-0.38537
2	-0.91173	1.79182	-0.80921	-0.57154	1.19486	-0.54136
3	-0.51779	1.44703	-0.81871	-0.33642	0.99096	-0.56792
4	-0.44862	1.36777	-0.75103	-0.29856	0.93785	-0.51255
5	-0.4515	1.30719	-0.7422	-0.29716	0.89303	-0.50509
6	-0.43157	1.21326	-0.68538	-0.28264	0.81530	-0.46020
7	-0.43606	1.21317	-0.69131	-0.27926	0.80997	-0.46378
8	-0.392	1.04941	-0.58674	-0.25334	0.69596	-0.38969
9	-0.45208	1.10009	-0.59175	-0.29656	0.72916	-0.38888
10	-0.42553	0.99725	-0.49992	-0.27488	0.65949	-0.32999
11	-0.50168	1.07575	-0.51401	-0.32630	0.70913	-0.33659
12	-0.498	1.06563	-0.50592	-0.33069	0.70668	-0.33105
13	-0.53101	1.16372	-0.58033	-0.35437	0.77582	-0.38003
14	-0.48744	1.07596	-0.52531	-0.31771	0.70752	-0.34216
15	-0.51899	1.04998	-0.49495	-0.35066	0.70177	-0.31664
16	-0.4773	0.90959	-	-0.33404	0.62528	-

5.3.3.2.1.2   Mid-rate LPC

For mid-rate bitrates (between 13.2 and 32 kbps) the LP analysis method is the same as for ACELP. The quantizer used is also the same but operated in AUDIO mode only (see subclause 5.2.2.1.2). However the interpolation of quantized LSF coefficients is different. Here each interpolated LSF coefficient i is a weighted sum of the quantized LSF coefficients referring to the current and previous frame:

$$LSF_{\text{int}erpol}(i) = 0.875 \cdot LSF_{curr}(i) + 0.125 \cdot LSF_{prev}(i), \quad i = 0..15 \tag{914}$$

These interpolated coefficients are converted back to LPC domain in the same way as done for ACELP frames.

5.3.3.2.1.3   High-rate LPC

This subclause describes the LPC analysis and quantization scheme for high-rate bitrates (48 kbps and higher). The description in the following subclauses refers to the quantization of either a single set of LPC parameters for the whole frame or two sets if the frame is divided into sub-frames. As described in subclause 5.3.2.3 a frame can be subdivided into TCX5 or TCX10 sub-frames. In this case the LPC quantizer however treats two adjacent TCX5 sub-frames as a single TCX10 sub-frame.

The LP coefficients are obtained by calculating autocorrelation on a windowed and pre-emphasized time domain signal with $\beta_{pre-emph}$ set to 0.9 for SWB and to 0.72 for WB and the input signal to the pre-emphasis filter is $s_{16}(n)$ for WB, $s_{25.6}(n)$ for 48 kbps SWB and $s_{32}(n)$ for 96 kbps and 128 kbps SWB (see subclause 5.1.4). The window used here is the window used for the TCX MDCT transformation (described in subclause 5.3.2.3) with the only exception, the ALDO windows will be replaced by FULL overlap windows (see subclause 5.3.2.5). After autocorrelation adaptive lag-windowing (see subclause 5.1.9.3) is applied to the autocorrelation coefficients which are finally converted to LP coefficients by Levinson recursion. The sample rate used for adaptive lag-windowing is the TCX sample rate (25.6 or 32 kHz) and the information of OL pitch estimation of current frame is used. For quantization the LP coefficients are converted to LSF domain.

The first set of LSF coefficients is always quantized without inter-frame dependency. In case two sets of LSF coefficients are quantized the second set may be quantized with or without dependency to the first set. This dependency (explained below) is signalled with one bit.

5.3.3.2.1.3.1   General Principle

The general principle for the quantization of a given LPC filter is represented in figure 62. A first-stage approximation is computed, and then subtracted from the input LSF vector to produce a residual LSF vector.

A LSF weighting function is derived from the first-stage approximation and applied to the residual LSF vector. The purpose and calculation of the weighting function are described in subclause 5.3.3.2.1.2.3.

The resulting weighted residual LSF vector is finally fed to the algebraic VQ encoder described in subclause 5.3.3.2.1.2.4.
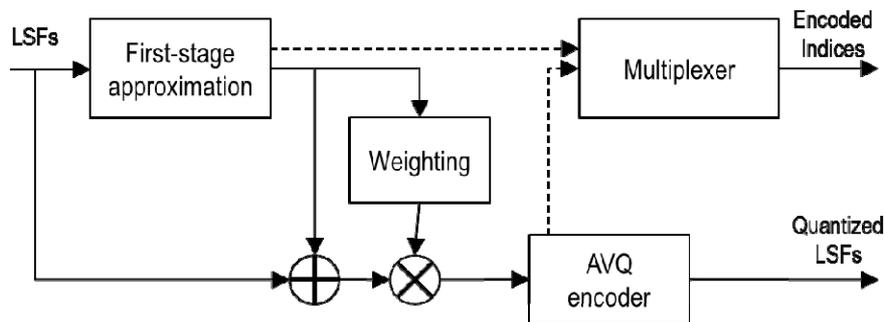
Figure 62: Principle of weighted algebraic LPC quantizer

5.3.3.2.1.3.2   First-stage approximation

The first stage approximation is a 16-dimensional, 8-bits stochastic vector quantizer applied to the input LSF vector. The codebook search uses a weighted Euclidian distance in which each component of the squared difference between the input LSF vector and the codebook entry is multiplied by:

$$w_1(i) = \frac{1}{d_i} + \frac{1}{d_{i+1}}, \quad i = 0..15 \tag{915}$$

with:

$$
\begin{aligned}
d_0 &= LSF(0) \\
d_{16} &= SF/2 - LSF(15) \\
d_i &= LSF(i) - LSF(i-1), i = 1..15
\end{aligned}
\tag{916}
$$

where LSF is the input LSF vector to be quantized and SF is the internal sampling frequency of the TCX-based codec (25600 for 48 kbps and 32000 for 96 kbps and above).

The difference between the input LSF vector and the first-stage approximation is called the residual LSF vector.

5.3.3.2.1.3.3   LSF weighting

Initially a weighted sum of the residual LSF vector is compared to a threshold:

$$\sum_i \left(w_2(i) \left(LSF(i) - \overline{LSF(i)} \right) \right)^2 < 8 \tag{917}$$

where $\overline{LSF(i)}$ are the first stage approximation of the LSF coefficients and $w_2(i)$ are the weights according to equation (918) with the factor k being 65. If the above relation is true, a special codebook is selected and no further quantization done. Otherwise the residual LSF vector is quantized with stochastic vector quantization.

The principle of stochastic vector quantization is to search a codebook of vectors for the nearest neighbour (in terms of Euclidian distance) of the vector to be quantized. When quantizing LPC filters in the LSF (line spectral frequency)

domain, a weighted Euclidian distance is generally used, each component of the vector being weighted differently depending on its value and of the value of the other components. The purpose of this weighting is to make the minimization of the Euclidian distance behave as close as possible to a minimization of the spectral distortion.

Unlike a stochastic quantizer, a uniform algebraic vector quantizer does not perform an exhaustive search of the codebook. It is therefore very complex to introduce a weighting function in the distance computation.

The solution used here is to warp the residual LSF vector (i.e. the difference between the input LSF vector and a first stage approximation) using a weighting function computed from the first-stage LSF approximation. By warping we mean applying different weights to the components of the LSF residual vector. Because the first-stage LSF approximation is also available at the decoder, the inverse weighting factors can also be computed at the decoder and the inverse warping can be applied to the quantized residual LSF vector. Warping the residual LSF vector according to a model that minimizes the spectral distortion is especially useful when the quantizer is uniform.

The weights applied to the components of the residual LSF vector are:

$$w_2(i) = \frac{SF}{2 \cdot k \cdot 16\sqrt{d_i \cdot d_{i+1}}}, \quad i = 0..15 \tag{918}$$

with

$$
\begin{aligned}
d_0 &= LSF(0) \\
d_{16} &= SF/2 - LSF(15) \\
d_i &= LSF(i) - LSF(i-1), \, i = 1..15
\end{aligned}
\tag{919}
$$

where SF is the internal sampling frequency of the TCX-based codec (25600 for 48 kbps and 32000 for 96 kbps and above). The factor k is always 60 for the first set of LSF coefficients.

5.3.3.2.1.3.4 Algebraic vector quantizer

The algebraic VQ used for quantizing the refinement is described in subclause 5.2.3.1.6.9.

5.3.3.2.1.3.5 Quantization of second set of LSF

In case a second set of LSF coefficients needs to be quantized, the quantization scheme for the second set may be dependent on the first set or not. Two approaches are tried and the one with the lower bit consumption will be chosen. The first approach is identical to the quantization of the first set. In the second approach the first stage approximation described in subclause 5.3.3.2.1.3.2 is replaced by the quantized first set of LSF. The factor k for calculating the weights is changed to 63. Finally the bit consumption of both approaches is compared and the one yielding the lower amount of bits is chosen and signalled with one bit.

5.3.3.2.2 Temporal Noise Shaping

Temporal Noise Shaping (TNS) is used to control the temporal shape of the quantization noise within each window of the transform.

If TNS is active in this encoder, up to two filters per MDCT-spectrum will be applied. The steps in the TNS encoding are described below. TNS is always calculated on a per subwindow basis, so in case of an 4 TCX5 window sequence these steps have to be applied once for each of the 4 subwindows.

Table 86: TNS configurations

Bitrate [kbps]	Bandwidth	Number of TNS filters for TCX20 and start and stop frequency	Number of TNS filters for TCX10 and start and stop frequency	Number of TNS filters for TCX5 and start and stop frequency
24.4, 32	SWB	1 (600Hz-16000Hz)	TCX10 not used	TCX5 not used
48,96,128	WB	1 (600Hz-8000Hz)	2 (800Hz-4400Hz, 4400-8000Hz)	1 (800Hz-8000Hz)
48,96,128	SWB	2 (600Hz-4500Hz, 4500-16000Hz)	2 (800Hz-8400Hz, 8400-16000Hz)	1 (800Hz-16000Hz)
24.4,32	FB	1 (600Hz-20000Hz)	TCX10 not used	TCX5 not used
48,96,128	FB	2 (600Hz-4500Hz, 4500-20000Hz)	2 (800Hz-10400Hz, 10400-20000Hz)	1 (800Hz-20000Hz)

The number of filters for each configuration and the start and the stop frequency of each filter are given in table 86.

The MDCT bins of 2 TCX5 sub-frames are rearranged prior to TNS filtering:

$$
\begin{aligned}
X_M(k) &= X_M^1(k), \quad k = 0,...7 \\
X_M(k+8) &= X_M^2(k), \quad k = 0,...7 \\
X_M(k+16) &= X_M^1(k+8), \quad k = 0,...,\frac{L}{4}-9 \\
X_M\left(k+\frac{L}{4}+8\right) &= X_M^2(k+8), \quad k = 0,...,\frac{L}{4}-9
\end{aligned}
\tag{920}
$$

For such rearanged and concatanated 2 TCX5 frames, special TNS configuration with 2 filters is used, which effectively functions as one filter per TCX5. The rearrangement is reverted after the filtering described below, to again have 2 separated TCX5 subwindows.

5.3.3.2.2.1 TNS detection

The spectral coefficients $X_M(k)$ between the start and stop frequency are divided into $n_{tns} = 3$ equal consecutive portions and for each portion the normalized autocorrelation function is calculated. The values of the autocorrelation function of all portions are then summed up and lag windowed. An exception is the first filter for 48/96/128 kbps SWB from 600 Hz to 4500 Hz which has only one portion ($n_{tns} = 1$).

The next step is an LPC calculation using the Levinson-Durbin algorithm (defined in the subclause 5.1.9.4). The filter oder d_{tns} is limited to 8 and to quarter of the number of the bins that a TNS filter covers.. As a result so called PARCOR or reflection coefficients $rq_i = k_{i+1}$ (where k_i is defined in subclause 5.1.9.4 and $i = 0,..., d_{tns} - 1$) and the prediction gain $\eta_{tns} = n_{tns}/E(d_{tns})$ are available, where $E(d_{tns})$ is the residual error energy as defined in subclause 5.1.9.4.

The TNS parcor coefficients will be quantized with a resolution of 4 bits.

The filter order is reduced such that the last PARCOR coefficient is non-zero.

TNS filter will be used only if the prediction gain is greater than a given threshold, which is dependent on the filter and varies between 1.35 and 1.85, or if the average of the squared filter coefficients is greater than a given threshold, which is dependent on the filter and varies between 0.03 and 0.075. For configurations with 2 filters per spectrum, each filter can be independently disabled or enabled.

5.3.3.2.2.2 TNS filtering

The spectral coefficients $X_M(k)$ will be replaced by the spectral coefficients filtered with the TNS filter. In the following text $X_M(k)$ refers to the TNS filtered or to the non-filtered spectral coefficients, depending on the configuration, where the configurations where TNS filtered spectral coefficients are used are listed in table 86. The filtering is done with the help of a so called lattice filter, no conversion from parcor coefficients rq to linear prediction coefficients is required.

5.3.3.2.2.3 Coding of the TNS parameters

The TNS filter order and the quantized parcor coefficients are coded using Huffman code. Which Huffman code will be used for the filter order depends on the frame configuration (TCX20/TCX10/TCX5) and on the bandwidth (SWB,WB). Which Huffman code will be used for a parcor coefficient depends on the frame configuration (TCX20/TCX10/TCX5), on the bandwidth (SWB,WB) and on the parcor coeffcient's index.

5.3.3.2.3 LPC shaping in MDCT domain

5.3.3.2.3.1 General Principle

LPC shaping is performed in the MDCT domain by applying gain factors computed from weighted quantized LP filter coefficients to the MDCT spectrum. The input sampling rate sr_{inp}, on which the MDCT transform is based, can be

higher than the CELP sampling rate sr_{celp}, for which LP coefficients are computed. Therefore LPC shaping gains can

only be computed for the part of the MDCT spectrum corresponding to the CELP frequency range. For the remaining part of the spectrum (if any) the shaping gain of the highest frequency band is used.

5.3.3.2.3.2 Computation of LPC shaping gains

To compute the 64 LPC shaping gains the weighted LP filter coefficients $\tilde{a}$ are first transformed into the frequency domain using an oddly stacked DFT of length 128:

$$X_{LPC}(b) = \sum_{i=0}^{16} \tilde{a}(i) e^{-j\frac{\pi}{128}(2b+1)i} \tag{921}$$

The LPC shaping gains g_{LPC} are then computed as the reciprocal absolute values of X_{LPC}:

$$g_{LPC}(b) = \frac{1}{\left|X_{LPC}(b)\right|} \quad , \quad b = 0...63 \tag{922}$$

5.3.3.2.3.3 Applying LPC shaping gains to MDCT spectrum

The MDCT coefficients X_M corresponding to the CELP frequency range are grouped into 64 sub-bands. The coefficients of each sub-band are multiplied by the reciprocal of the corresponding LPC shaping gain to obtain the shaped spectrum $\tilde{X}_M$. If the number of MDCT bins corresponding to the CELP frequency range $L_{TCX}^{(celp)}$ is not a multiple of 64, the width of sub-bands varies by one bin as defined by the following pseudo-code:

$$w = \left\lfloor L_{TCX}^{(celp)} / 64 \right\rfloor, \quad r = L_{TCX}^{(celp)} - 64w$$

```
if r = 0 then
```
$$s = 1 \, , \quad w_1 = w_2 = w$$
```
else if r ≤ 32 then
```
$$s = \left\lfloor 64/r \right\rfloor, \quad w_1 = w \, , \quad w_2 = w+1$$
```
else
```
$$s = \left\lfloor 64/(64-r) \right\rfloor, \quad w_1 = w+1 \, , \quad w_2 = w$$

$$i = 0$$
```
for  j = 0,...,63
{
if  j mod s ≠ 0 then
```
$$w = w_1$$
```
else
```
$$w = w_2$$

```
for  l = 0,..., min(w, L_TCX^(celp) - i) - 1
{
```
$$\tilde{X}_M(i) = X_M(i) / g_{LPC}(j)$$
$$i = i+1$$

```
}
}
```

The remaining MDCT coefficients above the CELP frequency range (if any) are multiplied by the reciprocal of the last LPC shaping gain:

$$\tilde{X}_M(i) = \frac{X_M(i)}{g_{LPC}(63)} \quad , \quad i = L_{TCX}^{(celp)}...L_{TCX}^{(bw)} - 1 \tag{923}$$

5.3.3.2.4 Adaptive low frequency emphasis

5.3.3.2.4.1 General Principle

The purpose of the adaptive low-frequency emphasis and de-emphasis (ALFE) processes is to improve the subjective performance of the frequency-domain TCX codec at low frequencies. To this end, the low-frequency MDCT spectral lines are amplified prior to quantization in the encoder, thereby increasing their quantization SNR, and this boosting is undone prior to the inverse MDCT process in the internal and external decoders to prevent amplification artifacts.

There are two different ALFE algorithms which are selected consistently in encoder and decoder based on the choice of arithmetic coding algorithm and bit-rate. ALFE algorithm 1 is used at 9.6 kbps (envelope based arithmetic coder) and at 48 kbps and above (context based arithmetic coder). ALFE algorithm 2 is used from 13.2 up to incl. 32 kbps. In the encoder, the ALFE operates on the spectral lines in vector x[] directly before (algorithm 1) or after (algorithm 2) every MDCT quantization, which runs multiple times inside a rate-loop in case of the context based arithmetic coder (see subclause 5.3.3.2.8.1).

5.3.3.2.4.2 Adaptive emphasis algorithm 1

ALFE algorithm 1 operates based on the LPC frequency-band gains, lpcGains[]. First, the minimum and maximum of the first nine gains – the low-frequency (LF) gains – are found using comparison operations executed within a loop over the gain indices 0 to 8.

Then, if the ratio between the minimum and maximum exceeds a threshold of 1/32, a gradual boosting of the lowest lines in x is performed such that the first line (DC) is amplified by $(32\,\text{min/max})^{0.25}$ and the 33rd line is not amplified:

```
tmp = 32 * min
if ((max < tmp) && (max > 0))
{
fac = tmp = pow(tmp / max, 1/128)
for (i = 31; i >= 0; i--)
{   /* gradual boosting of lowest 32 lines */
x[i] *= fac
fac  *= tmp
}
}
```

5.3.3.2.4.3 Adaptive emphasis algorithm 2

ALFE algorithm 2, unlike algorithm 1, does not operate based on transmitted LPC gains but is signaled by means of modifications to the quantized low-frequency (LF) MDCT lines. The procedure is divided into five consecutive steps:

- Step 1: first find first magnitude maximum at index i_max in lower spectral quarter ($k = 0 ... L_{TCX}^{(bw)} / 4$) utilizing $\text{invGain} = 2/g_{TCX}$ and modifying the maximum: xq[i_max] += (xq[i_max] < 0) ? -2 : 2

- Step 2: then compress value range of all x[i] up to i_max by requantizing all lines at $k = 0 ... i\_max–1$ as in the subclause describing the quantization, but utilizing invGain instead of g_{TCX} as the global gain factor.

- Step 3: find first magnitude maximum below i_max ($k = 0 ... L_{TCX}^{(bw)} / 4$) which is half as high if i_max > –1 using $\text{invGain} = 4/g_{TCX}$ and modifying the maximum: xq[i_max] += (xq[i_max] < 0) ? -2 : 2

- Step 4: re-compress and quantize all `x[i]` up to the half-height `i_max` found in the previous step, as in step 2

- Step 5: finish and always compress two lines at the latest `i_max` found, i.e. at $k =$ `i_max+1`, `i_max+2`, again utilizing `invGain` $= 2/g_{TCX}$ if the initial `i_max` found in step 1 is greater than –1, or using `invGain` $= 4/g_{TCX}$ otherwise. All `i_max` are initialized to –1. For details please see `AdaptLowFreqEmph()` in tcx_utils_enc.c.

5.3.3.2.5 Spectrum noise measure in power spectrum

For guidance of quantization in the TXC encoding process, a noise measure between 0 (tonal) and 1 (noise-like) is determined for each MDCT spectral line above a specified frequency based on the current transform's power spectrum. The power spectrum $X_P(k)$ is computed from the MDCT coefficients $X_M(k)$ and the MDST $X_S(k)$ coefficients on the same time-domain signal segment and with the same windowing operation:

$$X_P(k) = X_M^2(k) + X_S^2(k) \quad for\ k = 0..L_{TCX}^{(bw)} - 1 \tag{924}$$

Each noise measure in *noiseFlags*(k) is then calculated as follows. First, if the transform length changed (e.g. after a TCX transition transform following an ACELP frame) or if the previous frame did not use TCX20 coding (e.g. in case a shorter transform length was used in the last frame), all *noiseFlags*(k) up to $L_{TCX}^{(bw)} - 1$ are reset to zero. The noise measure start line k_{start} is initialized according to the following table 87.

Table 87: Initialization table of k_{start} in noise measure

Bitrate (kbps)	9.6	13.2	16.4	24.4	32	48	96	128
bw= NB, WB	66	128	200	320	320	320	320	320
bw=SWB,FB	44	96	160	320	320	256	640	640

For ACELP to TCX transitions, k_{start} is scaled by 1.25. Then, if the noise measure start line k_{start} is less than $L_{TCX}^{(bw)} - 6$, the *noiseFlags*(k) at and above k_{start} are derived recursively from running sums of power spectral lines:

$$s(k) = \sum_{i=k-7}^{k+7} X_P(i), \quad c(k) = \sum_{i=k-1}^{k+1} X_P(i) \tag{925}$$

$$noiseFlags(k) = \begin{cases} 1\ if\ s(k) \geq (1.75 - 0.5 \cdot noiseFlags(k)) \cdot c(k) \\ 0\ otherwise \end{cases} \quad for\ k_{start}...L_{TCX}^{(bw)} - 8 \tag{926}$$

Furthermore, every time *noiseFlags*(k) is given the value zero in the above loop, the variable *lastTone* is set to k. The upper 7 lines are treated separately since $s(k)$ cannot be updated any more ($c(k)$, however, is computed as above):

$$noiseFlags(k) = \begin{cases} 1\ if\ s(L_{TCX}^{(bw)} - 8) \geq (1.75 - 0.5 \cdot noiseFlags(k)) \cdot c(k) \\ 0\ otherwise \end{cases} \quad for\ L_{TCX}^{(bw)} - 7...L_{TCX}^{(bw)} - 2 \tag{927}$$

The uppermost line at $k = L_{TCX}^{(bw)} - 1$ is defined as being noise-like, hence $noiseFlags(L_{TCX}^{(bw)} - 1) = 1$. Finally, if the above variable *lastTone* (which was initialized to zero) is greater than zero, then $noiseFlags(lastTone + 1) = 0$. Note that this procedure is only carried out in TCX20, not in other TCX modes ($noiseFlags(k) = 0\ for\ k = 0..L_{TCX}^{(bw)} - 1$).

5.3.3.2.6 Low pass factor detector

A low pass factor c_{lpf} is determined based on the power spectrum for all bitrates below 32.0 kbps. Therefore, the power spectrum $X_P(k)$ is compared iteratively against a threshold t_{lpf} for all $k = L_{TCX}^{(bw)} - 1...L_{TCX}^{(bw)} / 2$, where $t_{lpf} = 32.0$ for

regular MDCT windows and $t_{lpf} = 64.0$ for ACELP to MDCT transition windows. The iteration stops as soon as $X_P(k) > t_{lpf}$.

The low pass factor c_{lpf} determines as $c_{lpf} = 0.3 \cdot c_{lpf,prev} + 0.7 \cdot (k+1)/L_{TCX}^{(celp)}$, where $c_{lpf,prev}$ is the last determined low pass factor. At encoder startup, $c_{lpf,prev}$ is set to 1.0. The low pass factor c_{lpf} is used to determine the noise filling stop bin (see subclause 5.3.3.2.10.2).

5.3.3.2.7   Uniform quantizer with adaptive dead-zone

For uniform quantization of the MDCT spectrum $\tilde{X}_M$ after or before ALFE (depending on the applied emphasis algorithm, see subclause 5.3.3.2.4.1), the coefficients are first divided by the global gain g_{TCX} (see subclause 5.3.3.2.8.1.1), which controls the step-size of quantization. The results are then rounded toward zero with a rounding offset which is adapted for each coefficient based on the coefficient's magnitude (relative to g_{TCX}) and tonality (as defined by *noiseFlags*(k) in subclause 5.3.3.2.5). For high-frequency spectral lines with low tonality and magnitude, a rounding offset of zero is used, whereas for all other spectral lines, an offset of 0.375 is employed. More specifically, the following algorithm is executed.

Starting from the highest coded MDCT coefficient at index $k = L_{TCX}^{(bw)} - 1$, we set $\tilde{X}_M(k) = 0$ and decrement k by 1 as long as condition *noiseFlags*(k) > 0 and $\left| \tilde{X}_M(k) \right| / g_{TCX} < 1$ evaluates to true. Then downward from the first line at index $k' \geq 0$ where this condition is not met (which is guaranteed since *noiseFlags*(0) = 0), rounding toward zero with a rounding offset of 0.375 and limiting of the resulting integer values to the range –32768 to 32767 is performed:

$$
\hat{X}_M(k) = \begin{cases} \min\left(\left\lfloor \dfrac{\tilde{X}_M(k)}{g_{TCX}} + 0.375 \right\rfloor, 32767 \right) & , \quad \tilde{X}_M(k) > 0 \\[4mm] \max\left(\left\lceil \dfrac{\tilde{X}_M(k)}{g_{TCX}} - 0.375 \right\rceil, -32768 \right) & , \quad \tilde{X}_M(k) \leq 0 \end{cases} \tag{928}
$$

with $k = 0..k'$. Finally, all quantized coefficients of $\hat{X}_M(k)$ at and above $k = L_{TCX}^{(bw)}$ are set to zero.

5.3.3.2.8   Arithmetic coder

The quantized spectral coefficients are noiselessly coded by an entropy coding and more particularly by an arithmetic coding.

The arithmetic coding uses 14 bits precision probabilities for computing its code. The alphabet probability distribution can be derived in different ways. At low rates, it is derived from the LPC envelope, while at high rates it is derived from the past context. In both cases, a harmonic model can be added for refining the probability model.

The following pseudo-code describes the arithmetic encoding routine, which is used for coding any symbol associated with a probability model. The probability model is represented by a cumulative frequency table *cum_freq[]*. The derivation of the probability model is described in the following subclauses.

```
/* global varibles */
low
high
bits_to_follow

ar_encode(symbol, cum_freq[])
{
    if (ari_first_symbol()) {
    low  = 0;
    high = 65535;
    bits_to_follow = 0;
    }
    range = high-low+1;

    if (symbol > 0) {
    high  = low + ((range*cum_freq[symbol-1])>>14) - 1;
    }
```

```
low += ((range*cum_freq[symbol-1])>>14) - 1;

for (;;) {
if (high < 32768 ) {
write_bit(0);
while ( bits_to_follow ) {
write_bit(1);
bits_to_follow--;
}
}
else if (low >= 32768 ) {
write_bit(1)
while ( bits_to_follow ) {
write_bit(0);
bits_to_follow--;
}
low  -= 32768;
high -= 32768;
}
else if ( (low >= 16384) && (high < 49152) ) {
bits_to_follow += 1;
low  -= 16384;
high -= 16384;
}
else break;

low  += low;
high += high+1;
}

if (ari_last_symbol()) /* flush bits */
if ( low < 16384 ) {
write_bit(0);
while ( bits_to_follow > 0) {
write_bit(1);
bits_to_follow--;
}
} else {
write_bit(1);
while ( bits_to_follow > 0) {
write_bit(0);
bits_to_follow--;
}
}
}
}
```

The helper functions *ari_first_symbol()* and *ari_last_symbol()* detect the first symbol and the last symbol of the generated codeword respectively.

5.3.3.2.8.1   Context based arithmetic codec

5.3.3.2.8.1.1   Global gain estimator

The estimation of the global gain g_{TCX} for the TCX frame is performed in two iterative steps. The first estimate considers a SNR gain of 6dB per sample per bit from SQ. The second estimate refines the estimate by taking into account the entropy coding.

The energy of each block of 4 coefficients is first computed:

$$E[k] = \sum_{i=0}^{4} \hat{X}^2[4.k+i] \tag{929}$$

A bisection search is performed with a final resolution of 0.125dB:

Initialization: Set *fac = offset = 12.8* and *target = 0.15(target_bits – L/16)*
Iteration: Do the following block of operations 10 times
1- *fac=fac/2*
2- *offset = offset – fac*
2- $ener = \sum_{i=0}^{L/4} a[i]$, where $a[i] = \begin{cases} E[k] - offset & \text{if } E[k] - offset > 0.3 \\ 0 & \text{otherwise} \end{cases}$

3- if(ener>target) then offset=offset+fac

The first estimate of gain is then given by:

$$g_{TCX} = 10^{0.45 + offset/2}$$ (930)

5.3.3.2.8.1.2 Rate-loop for constant bit rate and global gain

In order to set the best gain g_{TCX} within the constraints of $used\_bits \leq t\arg et\_bits$, convergence process of g_{TCX} and $used\_bits$ is carried out by using following valuables and constants:

W_{Lb} and W_{Ub} denote weights corresponding to the lower bound the upper bound,

g_{Lb} and g_{Ub} denote gain corresponding to the lower bound the upper bound, and

Lb_found and Ub_found denote flags indicating g_{Lb} and g_{Ub} is found, respectively.

μ and η are variables with $\mu = \max(1, 2.3 - 0.0025 * t\arg et\_bits)$ and $\eta = 1/\mu$.

λ and ν are constants, set as 10 and 0.96.

After the initial estimate of bit consumption by arithmetic coding, *stop* is set 0 when $t\arg et\_bits$ is larger than $used\_bits$, while *stop* is set as $used\_bits$ when $used\_bits$ is larger than $t\arg et\_bits$.

If *stop* is larger than 0, that means $used\_bits$ is larger than $t\arg et\_bits$,

g_{TCX} needs to be modified to be larger than the previous one and Lb_found is set as TRUE, g_{Lb} is set as the previous g_{TCX}. W_{Lb} is set as

$$W_{Lb} = stop - t\arg et\_bits + \lambda,$$ (931)

When Ub_found was set, that means $used\_bits$ was smaller than $t\arg et\_bits$, g_{TCX} is updated as an interpolated value between upper bound and lower bound. ,

$$g_{TCX} = (g_{Lb} \cdot W_{Ub} + g_{Ub} \cdot W_{Lb})/(W_{Ub} + W_{Lb}),$$ (932)

Otherwise, that means Ub_found is FALSE, gain is amplified as

$$g_{TCX} = g_{TCX} \cdot (1 + \mu \cdot ((stop/\nu)/t\arg et\_bits - 1)),$$ (933)

with larger amplification ratio when the ratio of $used\_bits (= stop)$ and $t\arg et\_bits$ is larger to accelerate to attain g_{Ub}.

If *stop* equals to 0, that means $used\_bits$ is smaller than $t\arg et\_bits$,

g_{TCX} should be smaller than the previous one and Ub_found is set as 1, Ub is set as the previous g_{TCX} and W_{Ub} is set as

$$W_{Ub} = t\arg et\_bits - used\_bits + \lambda,$$ (934)

If Lb_found has been already set, gain is calculated as

$$g_{TCX} = (g_{Lb} \cdot W_{Ub} + g_{Ub} \cdot W_{Lb})/(W_{Ub} + W_{Lb}),$$ (935)

otherwise, in order to accelerate to lower band gain g_{Lb}, gain is reduced as,

$$g_{TCX} = g_{TCX} \cdot (1 - \eta \cdot (1 - (used\_bits \cdot \nu)/t\arg et\_bits)),$$ (936)

with larger reduction rates of gain when the ratio of $used\_bits$ and $target\_bits$ is small.

After above correction of gain, quantization is performed and estimation of $used\_bits$ by arithmetic coding is obtained. As a result, $stop$ is set 0 when $target\_bits$ is larger than $used\_bits$, and is set as $used\_bits$ when it is larger than $target\_bits$. If the loop count is less than 4, either lower bound setting process or upper bound setting process is carried out at the next loop depending on the value $stop$. If the loop count is 4, the final gain g_{TCX} and the quantized MDCT sequence $X_{QMDCT}(k)$ are obtained.

5.3.3.2.8.1.3 Probability model derivation and coding

The quantized spectral coefficients X are noiselessly encoded starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient. They are encoded by groups of two coefficients a and b gathering in a so-called 2-tuple {a,b}.

Each 2-tuple {a,b} is split into three parts namely, MSB, LSB and the sign. The sign is coded independently from the magnitude using uniform probability distribution. The magnitude itself is further divided in two parts, the two most significant bits (MSBs) and the remaining least significant bitplanes (LSBs, if applicable). The 2-tuples for which the magnitude of the two spectral coefficients is lower or equal to 3 are coded directly by the MSB coding. Otherwise, an escape symbol is transmitted first for signalling any additional bit plane.

The relation between 2-tuple, the individual spectral values *a* and *b* of a 2-tuple, the most significant bit planes *m* and the remaining least significant bit planes, *r,* are illustrated in the example in figure 63. In this example three escape symbols are sent prior to the actual value m, indicating three transmitted least significant bit planes
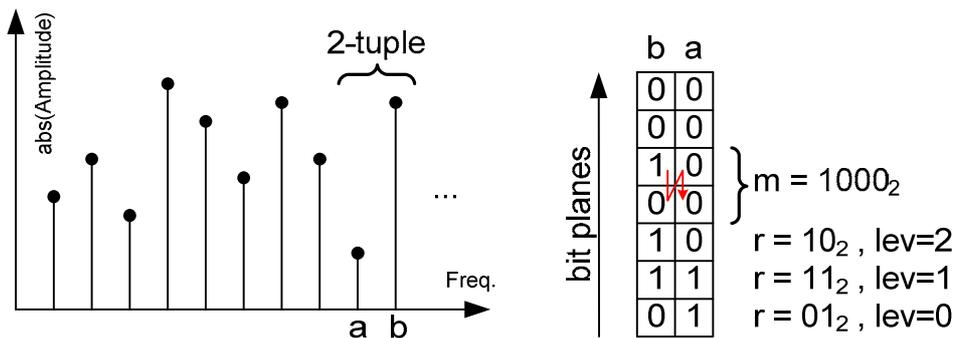
Figure 63: Example of a coded pair (2-tuple) of spectral values *a* and *b* and their representation as *m* and *r*.

The probability model is derived from the past context. The past context is translated on a 12 bits-wise index and maps with the lookup table *ari_context_lookup []* to one of the 64 available probability models stored in *ari_cf_m[]*.

The past context is derived from two 2-tuples already coded within the same frame. The context can be derived from the direct neighbourhood or located further in the past frequencies. Separate contexts are maintained for the peak regions (coefficients belonging to the harmonic peaks) and other (non-peak) regions according to the harmonic model. If no harmonic model is used, only the other (non-peak) region context is used.

The zeroed spectral values lying in the tail of spectrum are not transmitted. It is achieved by transmitting the index of last non-zeroed 2-tuple. If harmonic model is used, the tail of the spectrum is defined as the tail of spectrum consisting of the peak region coefficients, followed by the other (non-peak) region coefficients, as this definition tends to increase the number of trailing zeros and thus improves coding efficiency. The number of samples to encode is computed as follows:

$$lastnz = 2(\max_{0 \le k < L/2}\{(X[ip[2k]] + X[ip[2k+1]]) > 0\}) + 2 \tag{937}$$

The following data are written into the bitstream with the following order:

1- *lastnz/2-1* is coded on $\left\lceil \log_2(\frac{L}{2}) \right\rceil$ bits.

2- The entropy-coded MSBs along with escape symbols.

3- The signs with 1 bit-wise code-words

4- The residual quantization bits described in section when the bit budget is not fully used.

5- The LSBs are written backwardly from the end of the bitstream buffer.

The following pseudo-code describes how the context is derived and how the bitstream data for the MSBs, signs and LSBs are computed. The input arguments are the quantized spectral coefficients *X[]*, the size of the considered spectrum *L*, the bit budget *target_bits*, the harmonic model parameters (*pi, hi*), and the index of the last non zeroed symbol *lastnz*.

```
ari_context_encode(X[], L,target_bits,pi[],hi[],lastnz)
{
   c[0]=c[1]=p1=p2=0;
   for (k=0; k<lastnz; k+=2) {
   ari_copy_states();

   (a1_i,p1,idx1) = get_next_coeff(pi,hi,lastnz);
   (b1_i,p2,idx2) = get_next_coeff(pi,hi,lastnz);

   t=get_context(idx1,idx2,c,p1,p2);
   esc_nb = lev1 = 0;
   a = a1 = abs(X[a1_i]);
   b = b1 = abs(X[b1_i]);

   /* sign encoding*/
   if(a1>0) save_bit(X[a1_i]>0?0:1);
   if(b1>0) save_bit(X[b1_i]>0?0:1);

   /* MSB encoding */
   while (a1 > 3 || b1 > 3) {
   pki = ari_context_lookup[t+1024*esc_nb];

   /* write escape codeword */
   ari_encode(17, ari_cf_m[pki]);

   a1>>=1; b1 >>=1; lev1++;
   esc_nb = min(lev1,3);
   }
   pki = ari_context_lookup[t+1024*esc_nb];
   ari_encode(a1+4*b1, ari_cf_m[pki]);

   /* LSB encoding */
   for(lev=0;lev<lev1;lev++){
   write_bit_end((a>>lev)&1);
   write_bit_end((b>>lev)&1);
   }

   /*check budget*/
   if(nbbits>target_bits){
   ari_restore_states();
   break;
   }

   c=update_context(a,b,a1,b1,c,p1,p2);
   }
   write_sign_bits();
}
```

The helper functions *ari_save_states()* and *ari_restore_states()* are used for saving and restoring the arithmetic coder states respectively. It allows cancelling the encoding of the last symbols if it violates the bit budget. Moreover and in case of bit budget overflow, it is able to fill the remaining bits with zeros till reaching the end of the bit budget or till processing *lastnz* samples in the spectrum.

The other helper functions are described in the following subclauses.

5.3.3.2.8.1.4 Get next coefficient

```
(a,p,idx) = get_next_coeff(pi, hi, lastnz)
If ((ii[0] ≥ lastnz – min(#pi, lastnz)) or
(ii[1] < min(#pi, lastnz) and pi[ii[1]] < hi[ii[0]])) then
{
p=1
idx=ii[1]
a=pi[ii[1]]
}
else
{
p=0
idx=ii[0] + #pi
a=hi[ii[0]]
}
ii[p]=ii[p] + 1
```

The ii[0] and ii[1] counters are initialized to 0 at the beginning of *ari_context_encode()* (and *ari_context_decode()* in the decoder).

5.3.3.2.8.1.5 Context update

The context is updated as described by the following pseudo-code. It consists of the concatenation of two 4 bit-wise context elements.

```
if ( p1 ≠ p2 )
{
if ( mod(idx1,2) == 1 )
{
```
$$t = 1 + 2\lfloor a/2 \rfloor \cdot (1 + \lfloor a/4 \rfloor)$$
```
If ( t > 13 )
```
$$t = 12 + \min(1 + \lfloor a/8 \rfloor, 3)$$
$$c[p1] = 2^4 \cdot (c[p1] \wedge 15) + t$$
```
}
if ( mod(idx2,2) == 1 )
{
```
$$t = 1 + 2\lfloor b/2 \rfloor (1 + \lfloor b/4 \rfloor)$$
```
if ( t > 13 )
```
$$t = 12 + \min(1 + \lfloor b/8 \rfloor, 3)$$
$$c[p2] = 2^4 \cdot (c[p2] \wedge 15) + t$$
```
}
}
else
{
```
$$c[p1 \vee p2] = 16 \cdot (c[p1 \vee p2] \wedge 15)$$
```
if ( esc_nb < 2 )
```
$$c[p1 \vee p2] = c[p1 \vee p2] + 1 + (a1 + b1) \cdot (esc\_nb + 1)$$
```
else
```
$$c[p1 \vee p2] = c[p1 \vee p2] + 12 + esc\_nb$$
```
}
```

5.3.3.2.8.1.6 Get context

The final context is amended in two ways:

$$t = c[p1 \vee p2]$$
```
if min(idx1,idx2) > L/2 then
```
$$t = t + 256$$
```
if target_bits > 400 then
```

$$t = t + 512$$

The context t is an index from 0 to 1023.

5.3.3.2.8.1.7   Bit consumption estimation

The bit consumption estimation of the context-based arithmetic coder is needed for the rate-loop optimization of the quantization. The estimation is done by computing the bit requirement without calling the arithmetic coder. The generated bits can be accurately estimated by:

```
cum_freq= arith_cf_m[pki]+m
proba*= cum_freq[0]- cum_freq[1]
nlz=norm_l(proba)   /*get the number of leading zero */
nbits=nlz
proba>>=14
```

where *proba* is an integer initialized to 16384 and *m* is a MSB symbol.

5.3.3.2.8.1.8   Harmonic model

For both context and envelope based arithmetic coding, a harmonic model is used for more efficient coding of frames with harmonic content. The model is disabled if any of the following conditions apply:
- The bit-rate is not one of 9.6, 13.2, 16.4, 24.4, 32, 48 kbps.
- The previous frame was coded by ACELP.
- Envelope based arithmetic coding is used and the coder type is neither Voiced nor Generic.
- The single-bit harmonic model flag in the bit-stream in set to zero.
When the model is enabled, the frequency domain interval of harmonics is a key parameter and is commonly analysed and encoded for both flavours of arithmetic coders.

5.3.3.2.8.1.8.1   Encoding of Interval of harmonics

When pitch lag and gain are used for the post processing, the lag parameter is utilized for representing the interval of harmonics in the frequency domain. Otherwise, normal representation of interval is applied.

5.3.3.2.8.1.8.1.1   Encoding interval depending on time domain pitch lag

If integer part of pitch lag in time domain d_{int} is less than the frame size of MDCT L_{TCX}, frequency domain interval unit (between harmonic peaks corresponding to the pitch lag) T_{UNIT} with 7 bit fractional accuracy is given by

$$T_{UNIT} = \frac{(2 \cdot L_{TCX} \cdot res\_max) \cdot 2^7}{(d_{int} \cdot res\_max + d_{fr})} \tag{938}$$

where d_{fr} denotes the fractional part of pitch lag in time domain, res_max denotes the max number of allowable fractional values whose values are either 4 or 6 depending on the conditions.

Since T_{UNIT} has limited range, the actual interval between harmonic peaks in the frequency domain is coded relatively to T_{UNIT} using the bits specified in table 88. Among candidate of multiplication factors, $Ratio()$ given in the table 89 or table 90, the multiplication number is selected that gives the most suitable harmonic interval of MDCT domain transform coefficients.

$$Index_T = (T_{UNIT} + 2^6) / 2^7 - 2 \tag{939}$$

$$T_{MDCT} = \lfloor 4 \cdot T_{UNIT} \cdot Ratio(Index_{Bandwidth}, Index_T, Index_{MUL}) \rfloor / 4 \tag{940}$$

Table 88: Number of bits for specifying the multiplier depending on $Index_T$

$Index_T$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NB:	5	4	4	4	4	4	4	3	3	3	3	2	2	2	2	2
WB:	5	5	5	5	5	5	4	4	4	4	4	4	4	2	2	2

Table 89: Candidates of multiplier in the order of $Index_{MUL}$ depending on $Index_T$ (NB)

$Index_T$																
0	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	19	20	21	22	23	24	25	26	27	28	30	32	34	36	38	40
1	0.5	1	2	3	4	5	6	7	8	9	10	12	16	20	24	30
2	2	3	4	5	6	7	8	9	10	12	14	16	18	20	24	30
3	2	3	4	5	6	7	8	9	10	12	14	16	18	20	24	30
4	2	3	4	5	6	7	8	9	10	12	14	16	18	20	24	30
5	1	2	2.5	3	4	5	6	7	8	9	10	12	14	16	18	20
6	1	1.5	2	2.5	3	3.5	4	4.5	5	6	7	8	9	10	12	16
7	1	2	3	4	5	6	8	10	-	-	-	-	-	-	-	-
8	1	2	3	4	5	6	8	10	-	-	-	-	-	-	-	-
9	1	1.5	2	3	4	5	6	8	-	-	-	-	-	-	-	-
10	1	2	2.5	3	4	5	6	8	-	-	-	-	-	-	-	-
11	1	2	3	4	-	-	-	-	-	-	-	-	-	-	-	-
12	1	2	4	6	-	-	-	-	-	-	-	-	-	-	-	-
13	1	2	3	4	-	-	-	-	-	-	-	-	-	-	-	-
14	1	1.5	2	4	-	-	-	-	-	-	-	-	-	-	-	-
15	1	1.5	2	3	-	-	-	-	-	-	-	-	-	-	-	-
16	0.5	1	2	3	-	-	-	-	-	-	-	-	-	-	-	-

Table 90: Candidates of multiplier in the order of depending on $Index_T$ (WB)

$Index_T$																
0	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	19	20	21	22	23	24	25	26	27	28	30	32	34	36	38	40
1	1	2	3	4	5	6	7	8	9	10	12	14	16	18	20	22
	24	26	28	30	32	34	36	38	40	44	48	54	60	68	78	80
2	1.5	2	2.5	3	4	5	6	7	8	9	10	12	14	16	18	20
	22	24	26	28	30	32	34	36	38	40	42	44	48	52	54	68
3	1	1.5	2	2.5	3	4	5	6	7	8	9	10	11	12	13	14
	15	16	18	20	22	24	26	28	30	32	34	36	40	44	48	54
4	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	9
	10	11	12	13	14	15	16	18	20	22	24	26	28	34	40	41
5	1	1.5	2	2.5	3	3.5	4	4.5	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22.5	24	25	27	28	30	35
6	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	7	8	9	10
7	1	2	2.5	3	4	5	6	7	8	9	10	12	15	16	18	27
8	1	1.5	2	2.5	3	3.5	4	5	6	8	10	15	18	22	24	26
9	1	1.5	2	2.5	3	3.5	4	5	6	8	10	12	13	14	18	21
10	0.5	1	1.5	2	2.5	3	4	5	6	8	9	11	12	13.5	16	20
11	0.5	1	1.5	2	2.5	3	4	5	6	7	8	10	11	12	14	20
12	0.5	1	1.5	2	2.5	3	4	4.5	6	7.5	9	10	12	14	15	18
13	0.5	1	1.25	1.5	1.75	2	2.5	3	3.5	4	4.5	5	6	8	9	14
14	0.5	1	2	4	-	-	-	-	-	-	-	-	-	-	-	-
15	1	1.5	2	4	-	-	-	-	-	-	-	-	-	-	-	-
16	1	2	3	4	-	-	-	-	-	-	-	-	-	-	-	-

5.3.3.2.8.1.8.1.2 Encoding interval without depending on time domain pitch lag

When pitch lag and gain in the time domain is not used or the pitch gain is less than or equals to 0.46, normal encoding of the interval with un-equal resolution is used.

Unit interval of spectral peaks T_{UNIT} is coded as

$$T_{UNIT} = index + base \cdot 2^{\mathrm{Re}s} - bias , \qquad (941)$$

and actual interval T_{MDCT} is represented with fractional resolution of $\mathrm{Re}s$ as

$$T_{MDCT} = T_{UNIT} / 2^{\text{Re} s} . \tag{942}$$

Each paramter is shown in table 91, where "small size" means when frame size is smaller than 256 of the target bit rates is less than or equal to 150.

Table 91: Un-equal resolution for coding of (0<= index < 256)

	Res	*base*	*bias*
$index < 16$	3	6	0
$16 \le index < 80$	4	8	16
$80 \le index < 208$	3	12	80
"small size" or $208 \le index < 224$	1	28	208
$224 \le index < 256$	0	188	224

5.3.3.2.8.1.8.2 Void

5.3.3.2.8.1.8.3 Search for interval of harmonics

In search of the best interval of harmonics, encoder tries to find the index which can maximize the weighted sum E_{PERIOD} of the peak part of absolute MDCT coefficients. $E_{ABSM}(k)$ denotes sum of 3 samples of absolute value of MDCT domain transform coefficients as

$$E_{ABSM}(k) = \sum_{j=0}^{2} abs(X_M(k+j-1)) \tag{943}$$

$$E_{PERIOD}(T_{MDCT}) = (\frac{1}{num\_peak}) \sum_{n=1}^{num\_peak} E_{ABSM}(\lfloor n \cdot T_{MDCT} \rfloor)((3n-2)/255)^{0.3} \tag{944}$$

where *num_peak* is the maximum number that $\lfloor n \cdot T_{MDCT} \rfloor$ reaches the limit of samples in the frequency domain.

In case interval does not rely on the pitch lag in time domain, hierarchical search is used to save computational cost. If the index of the interval is less than 80, periodicity is checked by a coarse step of 4. After getting the best interval, finer periodicity is searched around the best interval from -2 to +2. If index is equal to or larger than 80, periodicity is searched for each index.

5.3.3.2.8.1.8.4 Decision of harmonic model

At the initial estimation, number of used bits without harmonic model, $used\_bits$, and one with harmonic model, $used\_bits_{hm}$ is obtained and the indicator of consumed bits $indicator_B$ is defined as

$$indicator_B = B_{no\_hm} - B_{hm} , \tag{945}$$

$$B_{no\_hm} = \max(stop, used\_bits) , \tag{946}$$

$$B_{hm} = \max(stop_{hm}, used\_bits_{hm}) + Index\_bits_{hm} , \tag{947}$$

where $Index\_bits_{hm}$ denotes the additional bits for model parameters of periodic harmonic structure, and *stop* and $stop_{hm}$ indicate the consumed bits when they are larger than the target bits. Thus, the larger $indicator_B$, the more preferable to use harmonic model. Relative periodicity $indicator_{hm}$ is defined as the normalized sum of absolute values for peak regions of the shaped MDCT coefficients as

$$indicator_{hm} = L_M \cdot E_{PERIOD}(T_{MDCT\_max}) / \sum_{n=1}^{L_M} E_{ABSM}(n) , \tag{948}$$

where T_{MDCT_max} is the harmonic interval that attain the max value of E_{PERIOD}. When the score of periodicity of this frame is larger than the threshold as

$$if\left((indicator_B > 2) \| ((abs(indicator_B) \leq 2) \& \&(indicator_{hm} > 2.6)\right), \tag{949}$$

this frame is considered to be coded by the harmonic model. The shaped MDCT coefficients divided by gain g_{TCX} are quantized to produce a sequence of integer values of MDCT coefficients, $\hat{X}_{TCX\_hm}$, and compressed by arithmetic coding with harmonic model. This process needs iterative convergence process (rate loop) to get g_{TCX} and $\hat{X}_{TCX\_hm}$ with consumed bits B_{hm}. At the end of convergence, in order to validate harmonic model, the consumed bits B_{no_hm} by arithmetic coding with normal (non-harmonic) model for $\hat{X}_{TCX\_hm}$ is additionally calculated and compared with B_{hm}. If B_{hm} is larger than B_{no_hm}, arithmetic coding of $\hat{X}_{TCX\_hm}$ is revert to use normal model. $B_{hm}-B_{no\_hm}$ can be used for residual quantization for further enhancements. Otherwise, harmonic model is used in arithmetic coding.

In contrast, if the indicator of periodicity of this frame is smaller than or the same as the threshold, quantization and arithmetic coding are carried out assuming the normal model to produce a sequence of integer values of the shaped MDCT coefficients, $\hat{X}_{TCX\_no\_hm}$ with consumed bits B_{no_hm}. After convergence of rate loop, consumed bits B_{hm} by arithmetic coding with harmonic model for $\hat{X}_{TCX\_no\_hm}$ is calculated. If B_{no_hm} is larger than B_{hm}, arithmetic coding of $\hat{X}_{TCX\_nohm}$ is switched to use harmonic model. Otherwise, normal model is used in arithmetic coding.

5.3.3.2.8.1.9 Use of harmonic information in Context based arithmetic coding

For context based arithmetic coding, all regions are classified into two categories. One is peak part and consists of 3 consecutive samples centered at U^{th} (U is a positive integer up to the limit) peak of harmonic peak of τ_U,

$$\tau_U = \lfloor U \cdot T_{MDCT} \rfloor. \tag{950}$$

The other samples belong to normal or valley part. Harmonic peak part can be specified by the interval of harmonics and integer multiples of the interval. Arithmetic coding uses different contexts for peak and valley regions.
For ease of description and implementation, the harmonic model uses the following index sequences:

$$pi = \left(i \in [0..L_M - 1] : \exists U : \tau_U - 1 \leq i \leq \tau_U + 1\right), \tag{951}$$

$$hi = \left(i \in [0..L_M - 1] : i \notin pi\right), \tag{952}$$

$$ip = \left(pi, hi\right), \text{ the concatenation of } pi \text{ and } hi. \tag{953}$$

In case of disabled harmonic model, these sequences are $pi = (\ )$, and $hi = ip = (0, \ldots, L_M - 1)$.

5.3.3.2.8.2 Envelope based arithmetic coder

In the MDCT domain, spectral lines are weighted with the perceptual model $W(z)$ such that each line can be quantized with the same accuracy. The variance of individual spectral lines follow the shape of the linear predictor $A^{-1}(z)$ weighted by the perceptual model, whereby the weighted shape is $S(z) = W(z)A^{-1}(z)$. $W(z)$ is calculated by transforming $\hat{\mathbf{q}}'_\gamma$ to frequency domain LPC gains as detailed in subclauses 5.3.3.2.4.1 and 5.3.3.2.4.2. $A^{-1}(z)$ is derived from $\hat{\mathbf{q}}'_1$ after conversion to direct-form coefficients, and applying tilt compensation $1 - \chi z^{-1}$, and finally transforming to frequency domain LPC gains. All other frequency-shaping tools, as well as the contribution from the harmonic model, shall be also included in this envelope shape $S(z)$. Observe that this gives only the relative variances of spectral lines, while the overall envelope has arbitrary scaling, whereby we must begin by scaling the envelope.

5.3.3.2.8.2.1 Envelope scaling

We will assume that spectral lines x_k are zero-mean and distributed according to the Laplace-distribution, whereby the probability distribution function is

$$f(x_k) = \frac{1}{2b_k} \exp\left(-\frac{|x_k|}{b_k}\right) \tag{954}$$

The entropy and thus the bit-consumption of such a spectral line is $bits_k = 1 + \log_2 2eb_k$. However, this formula assumes that the sign is encoded also for those spectral lines which are quantized to zero. To compensate for this discrepancy, we use instead the approximation

$$bits_k = \log_2\left(2eb_k + 0.15 + \frac{0.035}{b_k}\right), \tag{955}$$

which is accurate for $b_k \geq 0.08$. We will assume that the bit-consumption of lines with $b_k \leq 0.08$ is $bits_k = \log_2(1.0224)$ which matches the bit-consumption at $b_k = 0.08$. For large $b_k > 255$ we use the true entropy $bits_k = \log_2(2eb_k)$ for simplicity.

The variance of spectral lines is then $\sigma_k^2 = 2b_k^2$. If s_k^2 is the k th element of the power of the envelope shape $|S(z)|^2$ then s_k^2 describes the relative energy of spectral lines such that $\gamma^2 \sigma_k^2 = b_k^2$ where γ is scaling coefficient. In other words, s_k^2 describes only the shape of the spectrum without any meaningful magnitude and γ is used to scale that shape to obtain the actual variance σ_k^2.

Our objective is that when we encode all lines of the spectrum with an arithmetic coder, then the bit-consumption matches a pre-defined level B, that is, $B = \sum_{k=0}^{N-1} bits_k$. We can then use a bi-section algorithm to determine the appropriate scaling factor γ such that the target bit-rate B is reached.

Once the envelope shape b_k has been scaled such that the expected bit-consumption of signals matching that shape yield the target bit-rate, we can proceed to quantizing the spectral lines.

5.3.3.2.8.2.2 Quantization rate loop

Assume that x_k is quantized to an integer $\hat{x}_k$ such that the quantization interval is $[\hat{x}_k - 0.5, \hat{x}_k + 0.5]$ then the probability of a spectral line occurring in that interval is for $|\hat{x}_k| \geq 1$

$$p(\hat{x}_k) = \left(\exp\left(-\frac{|\hat{x}_k| - 0.5}{b_k}\right) - \exp\left(-\frac{|\hat{x}_k| + 0.5}{b_k}\right)\right) = \left(1 - \exp\left(-\frac{1}{b_k}\right)\right)\exp\left(-\frac{|\hat{x}_k| - 0.5}{b_k}\right). \tag{956}$$

and for $|\hat{x}_k| = 0$

$$p(\hat{x}_k) = \left(1 - \exp\left(-\frac{0.5}{b_k}\right)\right). \tag{957}$$

It follows that the bit-consumption for these two cases is in the ideal case

$$\begin{cases} 1 - \dfrac{0.5}{b_k}\log_2 e - \log_2\left(1 - \exp\left(-\dfrac{1}{b_k}\right)\right) + \dfrac{|\hat{x}_k|}{b_k}\log_2 e, & \hat{x}_k \neq 0 \\[2ex] \log_2\left(1 - \exp\left(-\dfrac{0.5}{b_k}\right)\right), & \hat{x}_k = 0. \end{cases} \tag{958}$$

By pre-computing the terms $\log_2\left(1 - \exp\left(-\frac{1}{b_k}\right)\right)$ and $\log_2\left(1 - \exp\left(-\frac{0.5}{b_k}\right)\right)$, we can efficiently calculate the bit-consumption of the whole spectrum.

The rate-loop can then be applied with a bi-section search, where we adjust the scaling of the spectral lines by a factor ρ, and calculate the bit-consumption of the spectrum ρx_k, until we are sufficiently close to the desired bit-rate. Note that the above ideal-case values for the bit-consumption do not necessarily perfectly coincide with the final bit-consumption, since the arithmetic codec works with a finite-precision approximation. This rate-loop thus relies on an approximation of the bit-consumption, but with the benefit of a computationally efficient implementation.

When the optimal scaling σ has been determined, the spectrum can be encoded with a standard arithmetic coder. A spectral line which is quantized to a value $\hat{x}_k \neq 0$ is encoded to the interval

$$\left[\exp\left(-\frac{|\hat{x}_k|-0.5}{b_k}\right),\ \exp\left(-\frac{|\hat{x}_k|+0.5}{b_k}\right)\right] \tag{959}$$

and $\hat{x}_k = 0$ is encoded onto the interval

$$\left[1,\ \exp\left(-\frac{|\hat{x}_k|+0.5}{b_k}\right)\right]. \tag{960}$$

The sign of $x_k \neq 0$ will be encoded with one further bit.

Observe that the arithmetic coder must operate with a fixed-point implementation such that the above intervals are bit-exact across all platforms. Therefore all inputs to the arithmetic coder, including the linear predictive model and the weighting filter, must be implemented in fixed-point throughout the system

5.3.3.2.8.2.3 Probability model derivation and coding

When the optimal scaling σ has been determined, the spectrum can be encoded with a standard arithmetic coder. A spectral line which is quantized to a value $\hat{x}_k \neq 0$ is encoded to the interval

$$\left[\exp\left(-\frac{|\hat{x}_k|-0.5}{b_k}\right),\ \exp\left(-\frac{|\hat{x}_k|+0.5}{b_k}\right)\right] \tag{961}$$

and $\hat{x}_k = 0$ is encoded onto the interval

$$\left[1,\ \exp\left(-\frac{|\hat{x}_k|+0.5}{b_k}\right)\right]. \tag{962}$$

The sign of $x_k \neq 0$ will be encoded with one further bit.

5.3.3.2.8.2.4 Harmonic model in envelope based arithmetic coding

In case of envelope base arithmetic coding, harmonic model can be used to enhance the arithmetic coding. The similar search procedure as in the context based arithmetic coding is used for estimating the interval between harmonics in the MDCT domain. However, the harmonic model is used in combination of the LPC envelope as shown in figure 64. The shape of the envelope is rendered according to the information of the harmonic analysis.

Harmonic shape at k in the frequency data sample is defined as

$$Q(k) = h \cdot \exp\left(-\frac{(k-\tau)^2}{2\sigma^2}\right), \tag{963}$$

when $\tau\text{-}4 \leq k \leq \tau + 4$, otherwise $Q(k) = 1.0$, where τ denotes center position of Uth harmonics.

$$\tau = \lfloor U \cdot T_{MDCT} \rfloor \tag{964}$$

h and σ are height and width of each harmonics depending on the unit interval as shown,

$$h = 2.8\left(1.125 - \exp\left(-0.07 \cdot T_{MDCT} / 2^{\text{Res}}\right)\right) \tag{965}$$

$$\sigma = 0.5\left(2.6 - \exp\left(-0.05 \cdot T_{MDCT} / 2^{\text{Res}}\right)\right) \tag{966}$$

Height and width get larger when interval gets larger.

The spectral envelope $S(k)$ is modified by the harmonic shape $Q(k)$ at k as

$$S(k) = S(k) \cdot (1 + g_{harm} \cdot Q(k)), \tag{967}$$

where gain for the harmonic components g_{harm} is always set as 0.75 for Generic mode, and g_{harm} is selected from {0.6, 1.4, 4.5, 10.0} that minimizes E_{norm} for Voiced mode using 2 bits,

$$E_{ABSres} = \sum_{k=0}^{L_M-1} (|X_M(k)|/S(k)), \qquad (968)$$

$$E_{norm} = \sum_{k=0}^{L_M-1} (|X_M(k)|/S(k)/E_{ABSres})^4. \qquad (969)$$

Harmonic envelope combined with LPC envelope

$T_{MDCT}/2^{Res}$

0   Frequency [Hz]   6400
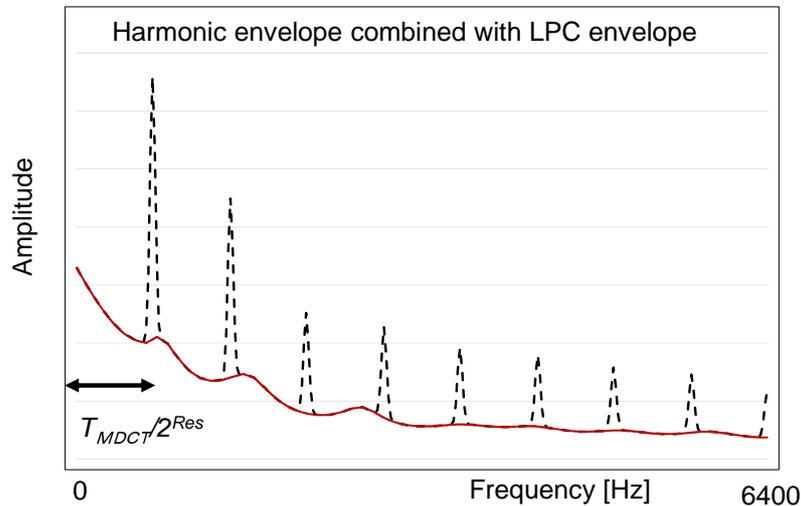
Figure 64: Example of harmonic envelope combined with LPC envelope used in envelope based arithmetic coding.

5.3.3.2.9 Global gain coding

5.3.3.2.9.1 Optimizing global gain

The optimum global gain g_{opt} is computed from the quantized and unquantized MDCT coefficients. For bit rates up to 32 kbps, the adaptive low frequency de-emphasis (see subclause 6.2.2.3.2) is applied to the quantized MDCT coefficients before this step. In case the computation results in an optimum gain less than or equal to zero, the global gain g_{TCX} determined before (by estimate and rate loop) is used.

$$g'_{opt} = \frac{\sum_{k=0}^{L_{TCX}^{(bw)}-1} X_M(k)\hat{X}_M(k)}{\sum_{k=0}^{L_{TCX}^{(bw)}-1} \left(\hat{X}_M(k)\right)^2} \qquad (970)$$

$$g_{opt} = \begin{cases} g'_{opt} & , \quad if \ g'_{opt} \geq 0 \\ g_{TCX} & , \quad if \ g'_{opt} < 0 \end{cases} \qquad (971)$$

5.3.3.2.9.2 Quantization of global gain

For transmission to the decoder the optimum global gain g_{opt} is quantized to a 7 bit index $I_{TCX,gain}$:

$$I_{TCX,gain} = \left\lfloor 28\log_{10}\left(\sqrt{L_{TCX}^{(bw)}\big/160} \ \ g_{opt} \right) + 0.5 \right\rfloor \qquad (972)$$

The dequantized global gain $\hat{g}_{TCX}$ is obtained as defined in subclause 6.2.2.3.3).

5.3.3.2.9.3   Residual coding

The residual quantization is a refinement quantization layer refining the first SQ stage. It exploits eventual unused bits *target_bits-nbbits*, where *nbbits* is the number of bits consumed by the entropy coder. The residual quantization adopts a greedy strategy and no entropy coding in order to stop the coding whenever the bit-stream reaches the desired size.

The residual quantization can refine the first quantization by two means. The first mean is the refinement of the global gain quantization. The global gain refinement is only done for rates at and above 13.2kbps. At most three additional bits is allocated to it. The quantized gain $\hat{g}_{TCX}$ is refined sequentially starting from *n=0* and incrementing *n* by one after each following iteration:

$if\,(\,g_{opt} < \hat{g}_{TCX}\,)\ then$

  $write\_bit(0)$

  $\hat{g}_{TCX} = \hat{g}_{TCX} \cdot 10^{-2^{-n-2}/28}$

$else\ then$

  $write\_bit(1)$

  $\hat{g}_{TCX} = \hat{g}_{TCX} \cdot 10^{2^{-n-2}/28}$

The second mean of refinement consists of re-quantizing the quantized spectrum line per line. First, the non-zeroed quantized lines are processed with a 1 bit residual quantizer:

$if\,(X[k] < \hat{X}[k])\ then$

  $write\_bit(0)$

$else\ then$

  $write\_bit(1)$

Finally, if bits remain, the zeroed lines are considered and quantized with on 3 levels. The rounding offset of the SQ with deadzone was taken into account in the residual quantizer design:

$fac\_z = (1 - 0.375) \cdot 0.33$

$if(\ |X[k]| < fac\_z \cdot \hat{X}[k])\ then$

  $write\_bit(0)$

$else\ then$

  $write\_bit(1)$

  $write\_bit((1 + \mathrm{sgn}(X[k]))/2)$

5.3.3.2.10   Noise Filling

On the decoder side noise filling is applied to fill gaps in the MDCT spectrum where coefficients have been quantized to zero. Noise filling inserts pseudo-random noise into the gaps, starting at bin $k_{NFstart}$ up to bin $k_{NFstop} - 1$. To control the amount of noise inserted in the decoder, a noise factor is computed on encoder side and transmitted to the decoder.

5.3.3.2.10.1   Noise Filling Tilt

To compensate for LPC tilt, a tilt compensation factor is computed. For bitrates below 13.2 kbps the tilt compensation is computed from the direct form quantized LP coefficients $\hat{a}$, while for higher bitrates a constant value is used:

$$t'_{NF} = \begin{cases} 0.5625 & ,\ if\ bitrate \geq 13200 \\ \min\left(1, \dfrac{\displaystyle\sum_{i=0}^{15} \hat{a}(i+1)\hat{a}(i)}{\displaystyle\sum_{i=0}^{15} (\hat{a}(i))^2} + 0.09375\right) & ,\ if\ bitrate < 13200 \end{cases} \qquad (973)$$

$$t_{NF} = \max\left(0.375, t'_{NF}\right)^{\frac{1}{L_{TCX}^{(celp)}}} \tag{974}$$

5.3.3.2.10.2 Noise Filling Start and Stop Bins

The noise filling start and stop bins are computed as follows:

$$k_{NFstart} = \begin{cases} L_{TCX}^{(celp)} \Big/ 6 & , \; \text{if } bitrate \geq 13200 \\ L_{TCX}^{(celp)} \Big/ 8 & , \; \text{if } bitrate < 13200 \end{cases} \tag{975}$$

$$k_{NFstop} = \begin{cases} t(0) & \text{if IGF is used} \\ L_{TCX}^{(bw)} & \text{else} \end{cases}$$

$$k_{NFstop,LP} = \begin{cases} \min(t(0), round(c_{lpf} \cdot L_{TCX}^{(celp)})) & , \text{if IGF is used} \\ \min(L_{TCX}^{(bw)}, round(c_{lpf} \cdot L_{TCX}^{(celp)})) & , \text{else} \end{cases} \tag{976}$$

5.3.3.2.10.3 Noise Transition Width

At each side of a noise filling segment a transition fadeout is applied to the inserted noise. The width of the transitions (number of bins) is defined as:

$$w_{NF} = \begin{cases} 8 & , \; \text{if } bitrate < 48000 \\ 4 + \lfloor 12.8 \cdot g_{LTP} \rfloor & , \; \text{if } (bitrate \geq 48000) \wedge TCX\,20 \wedge (HM = 0 \vee previous = ACELP) \\ 4 + \lfloor 12.8 \cdot \max(g_{LTP}, 0.3125) \rfloor & , \; \text{if } (bitrate \geq 48000) \wedge TCX\,20 \wedge (HM \neq 0 \wedge previous \neq ACELP) \\ 3 & , \; \text{if } (bitrate \geq 48000) \wedge TCX\,10 \end{cases} \tag{977}$$

where *HM* denotes that the harmonic model is used for the arithmetic codec and *previous* denotes the previous codec mode.

5.3.3.2.10.4 Computation of Noise Segments

The noise filling segments are determined, which are the segments of successive bins of the MDCT spectrum between $k_{NFstart}$ and $k_{NFstop,LP}$ for which all coefficients are quantized to zero. The segments are determined as defined by the following pseudo-code:

$k = k_{NFstart}$

$while \ (k > k_{NFstart} \ / \ 2) \ and \ (\hat{X}_M(k) = 0) \ do \ k = k - 1$

$k = k + 1$

$k'_{NFstart} = k$

$j = 0$

$while \ (k < k_{NFstop,LP}) \{$

   $while \ (k < k_{NFstop,LP}) \ and \ (\hat{X}_M(k) \neq 0) \ do \ k = k + 1$

   $k_{NF0}(j) = k$

   $while \ (k < k_{NFstop,LP}) \ and \ (\hat{X}_M(k) = 0) \ do \ k = k + 1$

   $k_{NF1}(j) = k$

   $if \ (k_{NF0}(j) < k_{NFstop,LP}) \ then \ j = j + 1$

$\}$

$n_{NF} = j$

where $k_{NF0}(j)$ and $k_{NF1}(j)$ are the start and stop bins of noise filling segment j, and n_{NF} is the number of segments.

5.3.3.2.10.5 Computation of Noise Factor

The noise factor is computed from the unquantized MDCT coefficients of the bins for which noise filling is applied.

If the noise transition width w_{NF} is 3 or less bins, an attenuation factor is computed based on the energy of even and odd MDCT bins:

$$E_{NFeven} = \sum_{i=0}^{\left\lceil \frac{k_{NFstop,LP}}{2} \right\rceil - \left\lfloor \frac{k'_{NFstart}}{2} \right\rfloor - 1} \left(X_M \left(2\left\lfloor \frac{k'_{NFstart}}{2} \right\rfloor + 2i \right) \right)^2 \tag{978}$$

$$E_{NFodd} = \sum_{i=0}^{\left\lceil \frac{k_{NFstop,LP}}{2} \right\rceil - \left\lfloor \frac{k'_{NFstart}}{2} \right\rfloor - 1} \left(X_M \left(2\left\lfloor \frac{k'_{NFstart}}{2} \right\rfloor + 2i + 1 \right) \right)^2 \tag{979}$$

$$f_{NFatt} = \begin{cases} \sqrt{\dfrac{2\min(E_{even}, E_{odd})}{E_{even} + E_{odd}}} & , \ if \ w_{NF} \leq 3 \\ 1 & , \ if \ w_{NF} > 3 \end{cases} \tag{980}$$

For each segment an error value is computed from the unquantized MDCT coefficients, applying global gain, tilt compensation and transitions:

$$E'_{NF}(j) = \frac{1}{g_{TCX}} \sum_{i=k_{NF0}}^{k_{NF1}-1} \left(|X_M(i)| \frac{\min(i - k_{NF0}(j) + 1, w_{NF})}{w_{NF}} \frac{\min(k_{NF1}(j) - i, w_{NF})}{w_{NF}} \left(\frac{1}{t_{NF}} \right)^i \right) \tag{981}$$

A weight for each segment is computed based on the width of the segment:

$$e_{NF}(j) = \begin{cases} k_{NF1}(j) - k_{NF0}(j) - w_{NF} + 1 & , & (w_{NF} \leq 3) \wedge (k_{NF1}(j) - k_{NF0}(j) > 2w_{NF} - 4) \\ \dfrac{0.28125}{w_{NF}}(k_{NF1}(j) - k_{NF0}(j))^2 & , & (w_{NF} \leq 3) \wedge (k_{NF1}(j) - k_{NF0}(j) \leq 2w_{NF} - 4) \\ k_{NF1}(j) - k_{NF0}(j) - 7 & , & (w_{NF} > 3) \wedge (k_{NF1}(j) - k_{NF0}(j) > 12) \\ 0.03515625(k_{NF1}(j) - k_{NF0}(j))^2 & , & (w_{NF} > 3) \wedge (k_{NF1}(j) - k_{NF0}(j) \leq 12) \end{cases} \quad (982)$$

The noise factor is then computed as follows:

$$f_{NF} = \begin{cases} f_{att} \dfrac{\sum\limits_{i=0}^{n_{NF}-1} E'_{NF}(i)}{\sum\limits_{i=0}^{n_{NF}-1} e_{NF}(i)} & , & if \quad \sum\limits_{i=0}^{n_{NF}-1} e_{NF}(i) > 0 \\ \\ 0 & , & else \end{cases} \quad (983)$$

5.3.3.2.10.6 Quantization of Noise Factor

For transmission the noise factor is quantized to obtain a 3 bit index:

$$I_{NF} = \min\left(\lfloor 10.75 f_{NF} + 0.5 \rfloor, 7\right) \quad (984)$$

5.3.3.2.11 Intelligent Gap Filling

The *Intelligent Gap Filling* (IGF) tool is an enhanced noise filling technique to fill gaps (regions of zero values) in spectra. These gaps may occur due to coarse quantization in the encoding process where large portions of a given spectrum might be set to zero to meet bit constraints. However, with the IGF tool these missing signal portions are reconstructed on the receiver side (RX) with parametric information calculated on the transmission side (TX). IGF is used only if TCX mode is active.

See table 92 below for all IGF operating points:

Table 92: IGF application modes

Bitrate	Mode
9.6 kbps	WB
9.6 kbps	SWB
13.2 kbps	SWB
16.4 kbps	SWB
24.4 kbps	SWB
32.2 kbps	SWB
48.0 kbps	SWB
16.4 kbps	FB
24.4 kbps	FB
32.0 kbps	FB
48.0 kbps	FB
96.0 kbps	FB
128.0 kbps	FB

On transmission side, IGF calculates levels on scale factor bands, using a complex or real valued TCX spectrum. Additionally spectral whitening indices are calculated using a spectral flatness measurement and a crest-factor. An arithmetic coder is used for noiseless coding and efficient transmission to receiver (RX) side.

5.3.3.2.11.1 IGF helper functions

5.3.3.2.11.1.1 Mapping values with the transition factor

If there is a transition from CELP to TCX coding ($isCelpToTCX = true$) or a TCX 10 frame is signalled ($isTCX10 = true$), the TCX frame length may change. In case of frame length change, all values which are related to the frame length are mapped with the function tF:

$$tF: \mathrm{N} \times \mathrm{P} \to \mathrm{N},$$

$$tF(n, f) := \begin{cases} \left\lfloor nf + \dfrac{1}{2} \right\rfloor, & \text{if } \left\lfloor nf + \dfrac{1}{2} \right\rfloor \text{ is even} \\[4mm] \left\lfloor nf + \dfrac{1}{2} \right\rfloor + 1, & \text{if } \left\lfloor nf + \dfrac{1}{2} \right\rfloor \text{ is odd} \end{cases} \tag{985}$$

where n is a natural number, for example a scale factor band offset, and f is a transition factor, see table 97.

5.3.3.2.11.1.2 TCX power spectrum

The power spectrum $P \in \mathrm{P}^n$ of the current TCX frame is calculated with:

$$P(sb) := R(sb)^2 + I(sb)^2, \ sb = 0,1,2,\ldots,n-1 \tag{986}$$

where n is the actual TCX window length, $R \in \mathrm{P}^n$ is the vector containing the real valued part (cos-transformed) of the current TCX spectrum, and $I \in \mathrm{P}^n$ is the vector containing the imaginary (sin-transformed) part of the current TCX spectrum.

5.3.3.2.11.1.3 The spectral flatness measurement function SFM

Let $P \in \mathrm{P}^n$ be the TCX power spectrum as calculated according to subclause 5.3.3.2.11.1.2 and b the start line and e the stop line of the SFM measurement range.

The SFM function, applied with IGF, is defined with:

$$SFM: \mathrm{P}^n \times \mathrm{N} \times \mathrm{N} \to \mathrm{P},$$

$$SFM(P, b, e) := 2^{\left(\frac{1}{2} + p\right)} \left(\frac{1}{e-b} \left(1 + \sum_{sb=b}^{e-1} P(sb) \right) \right)^{-1}, \tag{987}$$

where n is the actual TCX window length and p is defined with:

$$p := \frac{1}{e-b} \sum_{sb=b}^{e-1} \left\lfloor max\left(0, log_2\left(P(sb)\right)\right) \right\rfloor \tag{988}$$

5.3.3.2.11.1.4 The crest factor function $CREST$

Let $P \in \mathrm{P}^n$ be the TCX power spectrum as calculated according to subclause 5.3.3.2.11.1.2 and b the start line and e the stop line of the crest factor measurement range.

The $CREST$ function, applied with IGF, is defined with:

$$CREST: \mathrm{P}^n \times \mathrm{N} \times \mathrm{N} \to \mathrm{P},$$

$$CREST(P, b, e) = max\left(1, E_{max} \left(\frac{1}{e-b} \sum_{sb=b}^{e-1} \left\lfloor max\left(0, \log_2\left(P(sb)\right)\right) \right\rfloor^2 \right)^{-\frac{1}{2}} \right), \tag{989}$$

where n is the actual TCX window length and E_{max} is defined with:

$$E_{max} := \left\lfloor \max_{sb \in [b,e[\subset N} \left(0, \log_2\left(P(sb)\right)\right) \right\rfloor.$$
(990)

5.3.3.2.11.1.5 The mapping function hT

The hT mapping function is defined with:

$$hT : P \times N \to (0,1,2),$$

$$hT(s,k) = \begin{cases} 0 & \text{for} & s \le ThM_k \\ 1 & \text{for} & ThM_k < s \le ThS_k, \\ 2 & \text{for} & s > ThS_k \end{cases}$$
(991)

where s is a calculated spectral flatness value and k is the noise band in scope. For threshold values ThM_k, ThS_k refer to table 93 below.

Table 93: Thresholds for whitening for nT, ThM and ThS

Bitrate	Mode	nT	ThM	ThS
9.6 kbps	WB	2	0.36, 0.36	1.41, 1.41
9.6 kbps	SWB	3	0.84, 0.89, 0.89	1.30, 1.25, 1.25
13.2 kbps	SWB	2	0.84, 0.89	1.30, 1.25
16.4 kbps	SWB	3	0.83, 0.89, 0.89	1.31, 1.19, 1.19
24.4 kbps	SWB	3	0.81, 0.85, 0.85	1.35, 1.23, 1.23
32.2 kbps	SWB	3	0.91, 0.85, 0.85	1.34, 1.35, 1.35
48.0 kbps	SWB	1	1.15	1.19
16.4 kbps	FB	3	0.63, 0.27, 0.36	1.53, 1.32, 0.67
24.4 kbps	FB	4	0.78, 0.31, 0.34, 0.34	1.49, 1.38, 0.65, 0.65
32.0 kbps	FB	4	0.78, 0.31, 0.34, 0.34	1.49, 1.38, 0.65, 0.65
48.0 kbps	FB	1	0.80	1.0
96.0 kbps	FB	1	0	2.82
128.0 kbps	FB	1	0	2.82

5.3.3.2.11.1.6 Void

5.3.3.2.11.1.7 IGF scale factor tables

IGF scale factor tables are available for all modes where IGF is applied.

Table 94: Scale factor band offset table

Bitrate	Mode	Number of bands (nB)	Scale factor band offsets (t[0],t[1],…,t[nB])
9.6 kbps	WB	3	164, 186, 242, 320
9.6 kbps	SWB	3	200, 322, 444, 566
13.2 kbps	SWB	6	256, 288, 328, 376, 432, 496, 566
16.4 kbps	SWB	7	256, 288, 328, 376, 432, 496, 576, 640
24.4 kbps	SWB	8	256, 284, 318, 358, 402, 450, 508, 576, 640
32.2 kbps	SWB	8	256, 284, 318, 358, 402, 450, 508, 576, 640
48.0 kbps	SWB	3	512, 534, 576, 640
16.4 kbps	FB	9	256, 288, 328, 376, 432, 496, 576, 640, 720, 800
24.4 kbps	FB	10	256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800
32.0 kbps	FB	10	256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800
48.0 kbps	FB	4	512, 584, 656, 728, 800
96.0 kbps	FB	2	640, 720, 800
128.0 kbps	FB	2	640, 720, 800

The table 94 above refers to the TCX 20 window length and a transition factor 1.00.

For all window lengths apply the following remapping

$$t(k) := tF(t(k), f), \ k = 0,1,2,\ldots,nB \tag{992}$$

where tF is the transition factor mapping function described in subclause 5.3.3.2.11.1.1.

5.3.3.2.11.1.8   The mapping function m

Table 95: IGF minimal source subband, $minSb$

Bitrate	mode	$minSb$
9.6 kbps	WB	30
9.6 kbps	SWB	32
13.2 kbps	SWB	32
16.4 kbps	SWB	32
24.4 kbps	SWB	32
32.2 kbps	SWB	32
48.0 kbps	SWB	64
16.4 kbps	FB	32
24.4 kbps	FB	32
32.0 kbps	FB	32
48.0 kbps	FB	64
96.0 kbps	FB	64
128.0 kbps	FB	64

For every mode a mapping function is defined in order to access source lines from a given target line in IGF range.

Table 96: Mapping functions for every mode

Bitrate	Mode	nT	mapping Function
9.6 kbps	WB	2	$m2a$
9.6 kbps	SWB	3	$m3a$
13.2 1kbps	SWB	2	$m2b$
16.4 kbps	SWB	3	$m3b$
24.4 kbps	SWB	3	$m3c$
32.2 kbps	SWB	3	$m3c$
48.0 kbps	SWB	1	$m1$
16.4 kbps	FB	3	$m3d$
24.4 kbps	FB	4	$m4$
32.0 kbps	FB	4	$m4$
48.0 kbps	FB	1	$m1$
96.0 kbps	FB	1	$m1$
128.0 kbps	FB	1	$m1$

The mapping function $m1$ is defined with:

$$m1(x) := minSb + 2t(0) - t(nB) + (x - t(0)), \quad \text{for} \ \ t(0) \le x < t(nB) \tag{993}$$

The mapping function $m2a$ is defined with:

$$m2a(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(2) \\ minSb + (x - t(2)) & \text{for} \quad t(2) \le x < t(nB) \end{cases} \tag{994}$$

The mapping function $m2b$ is defined with:

$$m2b(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(4) \\ minSb + tF(32, f) + (x - t(4)) & \text{for} \quad t(2) \le x < t(nB) \end{cases} \tag{995}$$

The mapping function $m3a$ is defined with:

$$m3a(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(1) \\ minSb + tF(32, f) + (x - t(1)) & \text{for} \quad t(1) \le x < t(2) \\ minSb + tF(46, f) + (x - t(2)) & \text{for} \quad t(2) \le x < t(nB) \end{cases} \quad (996)$$

The mapping function $m3b$ is defined with:

$$m3b(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(4) \\ minSb + tF(48, f) + (x - t(4)) & \text{for} \quad t(4) \le x < t(6) \\ minSb + tF(64, f) + (x - t(6)) & \text{for} \quad t(6) \le x < t(nB) \end{cases} \quad (997)$$

The mapping function $m3c$ is defined with:

$$m3c(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(4) \\ minSb + tF(32, f) + (x - t(4)) & \text{for} \quad t(4) \le x < t(7) \\ minSb + tF(64, f) + (x - t(7)) & \text{for} \quad t(7) \le x < t(nB) \end{cases} \quad (998)$$

The mapping function $m3d$ is defined with:

$$m3d(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(4) \\ minSb + t(x - t(4)) & \text{for} \quad t(4) \le x < t(7) \\ minSb + (x - t(7)) & \text{for} \quad t(7) \le x < t(nB) \end{cases} \quad (999)$$

The mapping function $m4$ is defined with:

$$m4(x) := \begin{cases} minSb + (x - t(0)) & \text{for} \quad t(0) \le x < t(4) \\ minSb + tF(32, f) + (x - t(4)) & \text{for} \quad t(4) \le x < t(6) \\ minSb + (x - t(6)) & \text{for} \quad t(0) \le x < t(9) \\ minSb + (t(9) - t(8)) + (x - t(9)) & \text{for} \quad t(9) \le x < t(nB) \end{cases} \quad (1000)$$

The value f is the appropriate transition factor, see table 97 and tF is described in subclause 5.3.3.2.11.1.1.

Please note, that all values $t(0), t(1), \ldots, t(nB)$ shall be already mapped with the function tF, as described in subclause 5.3.3.2.11.1.1. Values for nB are defined in table 94.

The here described mapping functions will be referenced in the text as "mapping function m" assuming, that the proper function for the current mode is selected.

5.3.3.2.11.2 IGF input elements (TX)

The IGF encoder module expects the following vectors and flags as an input:

R : vector with real part of the current TCX spectrum X_M

I : vector with imaginary part of the current TCX spectrum X_S

P : vector with values of the TCX power spectrum X_P

$isTransient$: flag, signalling if the current frame contains a transient, see subclause 5.3.2.4.1.1

$isTCX10$: flag, signalling a TCX 10 frame

$isTCX20$: flag, signalling a TCX 20 frame

$isCelpToTCX$: flag, signalling CELP to TCX transition; generate flag by test whether last frame was CELP

$isIndepFla\,g$: flag, signalling that the current frame is independent from the previous frame

Listed in table 97, the following combinations signalled through flags $isTCX10$, $isTCX20$ and $isCelpToTCX$ are allowed with IGF:

Table 97: TCX transitions, transition factor f, window length n

Bitrate / Mode	$isTCX10$	$isTCX20$	$isCelpToTCX$	Transition factor f	Window length n
9.6 kbps / WB	false	true	false	1.00	320
	false	true	true	1.25	400
9.6 kbps / SWB	false	true	false	1.00	640
	false	true	true	1.25	800
13.2 kbps / SWB	false	true	false	1.00	640
	false	true	true	1.25	800
16.4 kbps / SWB	false	true	false	1.00	640
	false	true	true	1.25	800
24.4 kbps / SWB	false	true	false	1.00	640
	false	true	true	1.25	800
32.0 kbps / SWB	false	true	false	1.00	640
	false	true	true	1.25	800
48.0 kbps / SWB	false	true	false	1.00	640
	false	true	true	1.00	640
	true	false	false	0.50	320
16.4 kbps / FB	false	true	false	1.00	960
	false	true	true	1.25	1200
24.4 kbps / FB	false	true	false	1.00	960
	false	true	true	1.25	1200
32.0 kbps / FB	false	true	false	1.00	960
	false	true	true	1.25	1200
48.0 kbps / FB	false	true	false	1.00	960
	false	true	true	1.00	960
	true	false	false	0.50	480
96.0 kbps / FB	false	true	false	1.00	960
	false	true	true	1.00	960
	true	false	false	0.50	480
128.0 kbps / FB	false	true	false	1.00	960
	false	true	true	1.00	960
	true	false	false	0.50	480

5.3.3.2.11.3 IGF functions on transmission (TX) side

All function declaration assumes that input elements are provided by a frame by frame basis. The only exceptions are two consecutive TCX 10 frames, where the second frame is encoded dependent on the first frame.

5.3.3.2.11.4 IGF scale factor calculation

This subclause describes how the IGF scale factor vector $g(k)$, $k = 0,1,\ldots,nB-1$ is calculated on transmission (TX) side.

5.3.3.2.11.4.1 Complex valued calculation

In case the TCX power spectrum P is available the IGF scale factor values g are calculated using P:

$$E(k)_{cplx,\ target} := \sqrt{\frac{1}{t(k+1)-t(k)} \sum_{tb=t_k}^{t(k+1)-1} P(tb)}, \; k = 0,1,\ldots,nB-1, \tag{1001}$$

and let $m : N \to N$ be the mapping function which maps the IGF target range into the IGF source range described in subclause 5.3.3.2.11.1.8, calculate:

$$E(k)_{cplx,\ source} := \sqrt{\frac{1}{t(k+1)-t(k)}\sum_{tb=t_k}^{t(k+1)-1}P(m(tb))},\ k=0,1,\ldots,nB-1, \tag{1002}$$

$$E(k)_{real,\ source} := \sqrt{\frac{1}{t(k+1)-t(k)}\sum_{tb=t_k}^{t(k+1)-1}R(m(tb))^2},\ k=0,1,\ldots,nB-1, \tag{1003}$$

where $t(0), t(1), \ldots, t(nB)$ shall be already mapped with the function tF, see subclause 5.3.3.2.11.1.1, and nB are the number of IGF scale factor bands, see table 94.

Calculate $g(k)$ with:

$$g(k) := \left\lfloor \frac{1}{2} + 4\log_2\left(max\left(\frac{9}{10}, 16\left(\frac{E(k)_{cplx,target}}{E(k)_{cplx,source}} \right) E(k)_{real,source} \right) \right) \right\rfloor,\ k=0,1,\ldots,nB-1 \tag{1004}$$

and limit $g(k)$ to the range $[0,91] \subset Z$ with

$$g(k) = max(0, g(k)), \tag{1005}$$

The values $g(k),\ k=0,1,\ldots,nB-1,$ will be transmitted to the receiver (RX) side after further lossless compression with an arithmetic coder described in subclause 5.3.3.2.11.8.

5.3.3.2.11.4.2   Real valued calculation

If the TCX power spectrum is not available calculate:

$$E(k)_{real} := \sqrt{\frac{1}{t(k+1)-t(k)}\sum_{tb=t(k)}^{t(k+1)-1}R(tb)^2},\ k=0,1,\ldots,nB-1 \tag{1006}$$

where $t(0), t(1), \ldots, t(nB)$ shall be already mapped with the function tF, see subclause 5.3.3.2.11.1.1, and nB are the number of bands, see table 94.

Calculate $g(k)$ with:

$$g(k) := \left\lfloor \frac{1}{2} + 4\log_2\left(max\left(\frac{9}{10}, 16 E(k)_{real} \right) \right) \right\rfloor,\ k=0,1,\ldots,nB-1 \tag{1007}$$

and limit $g(k)$ to the range $[0,91] \subset Z$ with

$$\begin{aligned} g(k) &= max(0, g(k)), \\ g(k) &= min(91, g(k)). \end{aligned} \tag{1008}$$

The values $g(k),\ k=0,1,\ldots,nB-1,$ will be transmitted to the receiver (RX) side after further lossless compression with an arithmetic coder described in subclause 5.3.3.2.11.8.

5.3.3.2.11.5   IGF tonal mask

In order to determine which spectral components should be transmitted with the core coder, a tonal mask is calculated. Therefore all significant spectral content is identified whereas content that is well suited for parametric coding through IGF is quantized to zero.

5.3.3.2.11.5.1 IGF tonal mask calculation

In case the TCX power spectrum P is not available, all spectral content above $t(0)$ is deleted:

$$R(tb) := 0, \; t(0) \le tb < t(nB) \tag{1009}$$

where R is the real valued TCX spectrum after applying TNS and n is the current TCX window length.

In case the TCX power spectrum P is available, calculate:

$$E_{HP} = \frac{1}{2t(0)} \sum_{i=0}^{t(0)-1} i\, P(i) \tag{1010}$$

where $t(0)$ is the first spectral line in IGF range.

Given E_{HP}, apply the following algorithm:

```
Initialize last and next :
```

$$last := R(t(0)-1)$$

$$next := \begin{cases} 0 & \text{if } P(t(0)-1) < E_{HP} \\ R(t(0)) & \text{else} \end{cases}$$

```
for (i = t(0); i < t(nB)-1 ; i++) {
    if ( P(i) < E_HP ) {
```
$$last := R(i)$$
$$R(i) := next$$
$$next := 0$$
```
    } else if( P(i) ≥ E_HP ) {
```
$$R(i-1) := last$$
$$last := R(i)$$
$$next := R(i+1)$$
```
    }
}
if  P(t(nB-1)) < E_HP , set  R(t(nB)-1) := 0
```

5.3.3.2.11.6 IGF spectral flatness calculation

Table 98: Number of tiles nT and tile width wT

Bitrate	Mode	nT	wT
9.6 kbps	WB	2	$t(2)-t(0), t(nB)-t(2)$
9.6 kbps	SWB	3	$t(1)-t(0), t(2)-t(1), t(nB)-t(2)$
13.2 kbps	SWB	2	$t(4)-t(0), t(nB)-t(4)$
16.4 kbps	SWB	3	$t(4)-t(0), t(6)-t(4), t(nB)-t(6)$
24.4 kbps	SWB	3	$t(4)-t(0), t(7)-t(4), t(nB)-t(7)$
32.2 kbps	SWB	3	$t(4)-t(0), t(7)-t(4), t(nB)-t(7)$
48.0 kbps	SWB	1	$t(nB)-t(0)$
16.4 kbps	FB	3	$t(4)-t(0), t(7)-t(4), t(nB)-t(7)$
24.4 kbps	FB	4	$t(4)-t(0), t(6)-t(4), t(9)-t(6), t(nB)-t(9)$
32.0 kbps	FB	4	$t(4)-t(0), t(6)-t(4), t(9)-t(6), t(nB)-t(9)$
48.0 kbps	FB	1	$t(nB)-t(0)$
96.0 kbps	FB	1	$t(nB)-t(0)$
128.0 kbps	FB	1	$t(nB)-t(0)$

For the IGF spectral flatness calculation two static arrays, *prevFIR* and *prevIIR* , both of size nT are needed to hold filter-states over frames. Additionally a static flag *wasTransient* is needed to save the information of the input flag *isTransient* from the previous frame.

5.3.3.2.11.6.1 Resetting filter states

The vectors *prevFIR* and *prevIIR* are both static arrays of size nT in the IGF module and both arrays are initialised with zeroes:

$$\left. \begin{array}{l} prevFIR(k) := 0 \\ prevIIR(k) := 0 \end{array} \right\} \text{ for } k = 0,1,...,nT-1 \tag{1011}$$

This initialisation shall be done

- with codec start up

- with any bitrate switch

- with any codec type switch

- with a transition from CELP to TCX, e.g. *isCelpToTCX = true*

- if the current frame has transient properties, e.g. *isTransient = true*

5.3.3.2.11.6.2 Resetting current whitening levels

The vector *currWLevel* shall be initialised with zero for all tiles,

$$currWLevel(k) = 0, \ k = 0,1,\ldots,nT-1 \tag{1012}$$

- with codec start up

- with any bitrate switch

- with any codec type switch

- with a transition from CELP to TCX, e.g. *isCelpToTCX = true*

5.3.3.2.11.6.3 Calculation of spectral flatness indices

The following steps 1) to 4) shall be executed consecutive:

1) Update previous level buffers and initialize current levels:

$$prevWLevel(k) := currWLevel(k), \ k = 0,1,\dots,nT-1$$
$$currWLevel(k) := 0, \ k = 0,1,\dots,nT-1$$

(1013)

In case *prevIsTransient* or *isTransien t* is true, apply

$$currWLevel(k) = 1, \quad k = 0,1,\dots,nT-1$$

(1014)

else, if the power spectrum P is available, calculate

$$tmp(k) := \frac{SFM(P,e(k),e(k+1))}{CREST(P,e(k),e(k+1))}, \ k = 0,1,\dots,nT-1$$

(1015)

with

$$e(k) := \begin{cases} t(0) & k = 0 \\ e(k-1)+wT(k) & k = 1,\dots,nT-1 \end{cases}$$

(1016)

where SFM is a spectral flatness measurement function, described in subclause 5.3.3.2.11.1.3 and $CREST$ is a crest-factor function described in subclause 5.3.3.2.11.1.4.

Calculate:

$$s(k) := \min\left(2.7, tmp(k)+ prevFIR(k)+\frac{1}{2}\,prevIIR(k)\right)$$

(1017)

After calculation of the vector $s(k)$, the filter states are updated with:

$$prevFIR(k) = tmp(k), \ k = 0,1,\dots,nT-1$$
$$prevIIR(k) = s(k), \ k = 0,1,\dots,nT-1$$
$$prevIsTransient = isTransient$$

(1018)

2) A mapping function $hT : N \times P \to N$ is applied to the calculated values to obtain a whitening level index vector *currWLevel* The mapping function $hT : N \times P \to N$ is described in subclause 5.3.3.2.11.1.5.

$$currWLevel(k) = hT(s(k),k), \ k = 0,1,\dots,nT-1$$

(1019)

3) With selected modes, see table 99, apply the following final mapping:

$$currWLeve(nT-1) := currWLeve(nT-2)$$

(1020)

Table 99: modes for step 4) mapping

Bitrate	mode	mapping
9.6 kbps	WB	apply
9.6 kbps	SWB	apply
13.2 kbps	SWB	NOP
16.4 kbps	SWB	apply
24.4 kbps	SWB	apply
32.2 kbps	SWB	apply
48.0 kbps	SWB	NOP
16.4 kbps	FB	apply
24.4 kbps	FB	apply
32.0 kbps	FB	apply
48.0 kbps	FB	NOP
96.0 kbps	FB	NOP
128.0 kbps	FB	NOP

After executing step 4) the whitening level index vector $currWLevel$ is ready for transmission.

5.3.3.2.11.6.4 Coding of IGF whitening levels

IGF whitening levels, defined in the vector $currWLevel$, are transmitted using 1 or 2 bits per tile. The exact number of total bits required depends on the actual values contained in $currWLevel$ and the value of the $isIndep$ flag. The detailed processing is described in the pseudo code below:

```
isSame  = 1;
nTiles = nT ;
k       = 0;
if ( isIndep) {
  isSame = 0;
} else {
  for (k = 0; k < nTiles ; k++) {
    if ( currWLevel(k) != prevWLevel(k)) {
      isSame = 0;
      break;
    }
  }
}

if ( isSame ) {
  write_bit(1);
} else {
  if (! isIndep ) {
    write_bit(0);
  }
  encode_whitening_level( currWLevel(0) );
  for (k = 1; k < nTiles ; k++) {
    isSame = 1;
    if ( currWLevel(k) != currWLevel(k-1) ) {
      isSame = 0;
      break;
    }
  }
  if (! isSame ) {
    write_bit(1);
    for (k = 1; k < nTiles ; k++) {
      encode_whitening_level( currWLevel(k) );
    }
  } else {
    write_bit(0);
  }
}
```

wherein the vector *prevWLevel* contains the whitening levels from the previous frame and the function `encode_whitening_level` takes care of the actual mapping of the whitening level $currWLevel(k)$ to a binary code. The function is implemented according to the pseudo code below:

```
if ( currWLevel(k)==1) {
   write_bit(0);
} else {
   write_bit(1);
   if ( currWLevel(k)==0) {
      write_bit(0);
   } else {
      write_bit(1);
   }
}
```

5.3.3.2.11.7 IGF temporal flatness indicator

The temporal envelope of the reconstructed signal by the IGF is flattened on the receiver (RX) side according to the transmitted information on the temporal envelope flatness, which is an IGF flatness indicator.

The temporal flatness is measured as the linear prediction gain in the frequency domain. Firstly, the linear prediction of the real part of the current TCX spectrum is performed and then the prediction gain η_{igf} is calculated:

$$\eta_{igf} = \frac{1}{\prod_{i=1}^{8}\left(1-k_i^2\right)} \tag{1021}$$

where $k_i = i$-th PARCOR coefficient obtained by the linear prediction.

From the prediction gain η_{igf} and the prediction gain η_{tns} described in subclause 5.3.3.2.2.3, the IGF temporal flatness indicator flag *isIgfTemFlat* is defined as

$$isIgfTemFlat = \begin{cases} 1 & \eta_{igf} < 1.15 \text{ and } \eta_{tns} < 1.15 \\ 0 & otherwise \end{cases} \tag{1022}$$

5.3.3.2.11.8 IGF noiseless coding

The IGF scale factor vector g is noiseless encoded with an arithmetic coder in order to write an efficient representation of the vector to the bit stream.

The module uses the common raw arithmetic encoder functions from the infrastructure, which are provided by the core encoder. The functions used are $ari\_encode\_14bits\_sign(bit)$, which encodes the value bit, $ari\_encode\_14bits\_ext(value, cumulativeFrequencyTable)$, which encodes $value$ from an alphabet of 27 symbols ($SYMBOLS\_IN\_TABLE$) using the cumulative frequency table $cumulativeFrequencyTable$, $ari\_start\_encoding\_14bits()$, which initializes the arithmetic encoder, and $ari\_finish\_encoding\_14bits()$, which finalizes the arithmetic encoder.

5.3.3.2.11.8.1 IGF independency flag

The internal state of the arithmetic encoder is reset in case the *isIndepFlag* flag has the value *true*. This flag may be set to *false* only in modes where TCX10 windows (see table 97) are used for the second frame of two consecutive TCX 10 frames.

5.3.3.2.11.8.2 IGF all-Zero flag

The IGF all-Zero flag signals that all of the IGF scale factors are zero:

$$allZero = \begin{cases} 1 & \text{if } g(k)=0, \text{for all } 0 \le k < nB \\ 0 & else \end{cases} \tag{1023}$$

The *allZero* flag is written to the bit stream first. In case the flag is *true* , the encoder state is reset and no further data is written to the bit stream, otherwise the arithmetic coded scale factor vector g follows in the bit stream.

5.3.3.2.11.8.3   IGF arithmetic encoding helper functions

5.3.3.2.11.8.3.1   The reset function

The arithmetic encoder states consist of $t \in \{0,1\}$, and the *prev* vector, which represents the value of the vector g preserved from the previous frame. When encoding the vector g , the value 0 for t means that there is no previous frame available, therefore *prev* is undefined and not used. The value 1 for t means that there is a previous frame available therefore *prev* has valid data and it is used, this being the case only in modes where TCX10 windows (see table 97) are used for the second frame of two consecutive TCX 10 frames. For resetting the arithmetic encoder state, it is enough to set $t = 0$.

If a frame has *isIndepFlag* set, the encoder state is reset before encoding the scale factor vector g . Note that the combination $t = 0$ and *isIndepFlag = false* is valid, and may happen for the second frame of two consecutive TCX 10 frames, when the first frame had $allZero=1$. In this particular case, the frame uses no context information from the previous frame (the *prev* vector), because $t = 0$, and it is actually encoded as an independent frame.

5.3.3.2.11.8.3.2   The arith_encode_bits function

The *arith _ encode _ bits* function encodes an unsigned integer x , of length *nBits* bits, by writing one bit at a time.

```
arith_encode_bits(x, nBits)
{
  for (i = nBits - 1; i >= 0; --i) {
    bit = (x >> i) & 1;
    ari_encode_14bits_sign(bit);
  }
}
```

5.3.3.2.11.8.3.2   The save and restore encoder state functions

Saving the encoder state is achieved using the function *iisIGFSCFEncoderSaveContextState* , which copies t and *prev* vector into *tSave* and *prevSave* vector, respectively. Restoring the encoder state is done using the complementary function *iisIGFSCFEncoderRestoreContextState* , which copies back *tSave* and *prevSave* vector into t and *prev* vector, respectively.

5.3.3.2.11.8.4   IGF arithmetic encoding

Please note that the arithmetic encoder should be capable of counting bits only, e.g., performing arithmetic encoding without writing bits to the bit stream. If the arithmetic encoder is called with a counting request, by using the parameter *doRealEncoding* set to *false* , the internal state of the arithmetic encoder shall be saved before the call to the top level function *iisIGFSCFEncoderEncode* and restored and after the call, by the caller. In this particular case, the bits internally generated by the arithmetic encoder are not written to the bit stream.

The *arith _ encode _ residual* function encodes the integer valued prediction residual x , using the cumulative frequency table *cumulativeFrequencyTable* , and the table offset *tableOffset* . The table offset *tableOffset* is used to adjust the value x before encoding, in order to minimize the total probability that a very small or a very large value will be encoded using escape coding, which slightly is less efficient. The values which are between $MIN \_ ENC \_ SEPARATE = -12$ and $MAX \_ ENC \_ SEPARATE = 12$, inclusive, are encoded directly using the cumulative frequency table *cumulativeFrequencyTable* , and an alphabet size of $SYMBOLS\_ IN \_ TABLE = 27$.

For the above alphabet of SYMBOLS_IN_TABLE symbols, the values 0 and $SYMBOLS\_ IN \_ TABLE - 1$ are reserved as escape codes to indicate that a value is too small or too large to fit in the default interval. In these cases, the value *extra* indicates the position of the value in one of the tails of the distribution. The value *extra* is encoded using 4 bits if it is in the range $\{0,\ldots,14\}$, or using 4 bits with value 15 followed by extra 6 bits if it is in the range $\{15,\ldots,15+62\}$, or using 4 bits with value 15 followed by extra 6 bits with value 63 followed by extra 7 bits if it is larger or equal than

$15 + 63$. The last of the three cases is mainly useful to avoid the rare situation where a purposely constructed artificial signal may produce an unexpectedly large residual value condition in the encoder.

```
arith_encode_residual(x, cumulativeFrequencyTable, tableOffset)
{
  x += tableOffset;
  if ((x >= MIN_ENC_SEPARATE) && (x <= MAX_ENC_SEPARATE)) {
    ari_encode_14bits_ext((x - MIN_ENC_SEPARATE) + 1, cumulativeFrequencyTable);

    return;
  } else if (x < MIN_ENC_SEPARATE) {
    extra = (MIN_ENC_SEPARATE - 1) - x;
    ari_encode_14bits_ext(0, cumulativeFrequencyTable);
  } else { /* x > MAX_ENC_SEPARATE */
    extra = x - (MAX_ENC_SEPARATE + 1);
    ari_encode_14bits_ext(SYMBOLS_IN_TABLE - 1, cumulativeFrequencyTable);
  }

  if (extra < 15) {
    arith_encode_bits(extra, 4);
  } else { /* extra >= 15 */
    arith_encode_bits(15, 4);
    extra -= 15;

    if (extra < 63) {
      arith_encode_bits(extra, 6);
    } else { /* extra >= 63 */
      arith_encode_bits(63, 6);
      extra -= 63;
      arith_encode_bits(extra, 7);
    }
  }
}
```

The function $encode\_sfe\_vector$ encodes the scale factor vector g, which consists of nB integer values. The value t and the $prev$ vector, which constitute the encoder state, are used as additional parameters for the function. Note that the top level function $iisIGFSCFEncoderEncode$ must call the common arithmetic encoder initialization function $ari\_start\_encoding\_14bits$ before calling the function $encode\_sfe\_vector$, and also call the arithmetic encoder finalization function $ari\_done\_encoding\_14bits$ afterwards.

The function $quant\_ctx$ is used to quantize a context value ctx, by limiting it to $\{-3,\ldots,3\}$, and it is defined as:

```
quant_ctx(ctx)
{
  if (abs(ctx) <= 3) {
    return ctx;
  } else if (ctx > 3) {
    return 3;
  } else { /* ctx < -3 */
    return -3;
  }
}
```

The definitions of the symbolic names indicated in the comments from the pseudo code, used for computing the context values, are listed in the following table 100:

Table 100: Definition of symbolic names

the previous frame (when available)	the current frame
$a = prev[f]$	$x = g[f]$ (the value to be coded)
$c = prev[f-1]$	$b = g[f-1]$ (when available)
	$e = g[f-2]$ (when available)

```
encode_sfe_vector(t, prev, g, nB)
  for (f = 0; f < nB; f++) {
    if (t == 0) {
      if (f == 0) {
        ari_encode_14bits_ext(g[f] >> 2, cf_se00);
```

```
        arith_encode_bits(g[f] & 3, 2); /* LSBs as 2 bit raw */
      }
      else if (f == 1) {
        pred = g[f - 1]; /* pred = b */
        arith_encode_residual(g[f] - pred, cf_se01, cf_off_se01);
      } else { /* f >= 2 */
        pred = g[f - 1]; /* pred = b */
        ctx = quant_ctx(g[f - 1] - g[f - 2]); /* Q(b - e) */
        arith_encode_residual(g[f] - pred, cf_se02[CTX_OFFSET + ctx)],
              cf_off_se02[IGF_CTX_OFFSET + ctx]);
      }
    }
    else { /* t == 1 */
      if (f == 0) {
        pred = prev[f]; /* pred = a */
        arith_encode_residual(x[f] - pred, cf_se10, cf_off_se10);
      } else { /* (t == 1) && (f >= 1) */
        pred = prev[f] + g[f - 1] - prev[f - 1]; /* pred = a + b - c */
        ctx_f = quant_ctx(prev[f] - prev[f - 1]); /* Q(a - c) */
        ctx_t = quant_ctx(g[f - 1] - prev[f - 1]); /* Q(b - c) */
        arith_encode_residual(g[f] - pred,
              cf_se11[CTX_OFFSET + ctx_t][CTX_OFFSET + ctx_f)],
              cf_off_se11[CTX_OFFSET + ctx_t][CTX_OFFSET + ctx_f]);
      }
    }
  }
}
```

There are five cases in the above function, depending on the value of t and also on the position f of a value in the vector g :

- when $t = 0$ and $f = 0$, the first scalefactor of an independent frame is coded, by splitting it into the most significant bits which are coded using the cumulative frequency table cf_se00, and the least two significant bits coded directly.

- when $t = 0$ and $f = 1$, the second scale factor of an independent frame is coded (as a prediction residual) using the cumulative frequency table cf_se01.

- when $t = 0$ and $f \geq 2$, the third and following scale factors of an independent frame are coded (as prediction residuals) using the cumulative frequency table $cf\_se02[CTX\_OFFSET + ctx]$, determined by the quantized context value ctx.

- when $t = 1$ and $f = 0$, the first scalefactor of a dependent frame is coded (as a prediction residual) using the cumulative frequency table cf_se10.

- when $t = 1$ and $f \geq 1$, the second and following scale factors of a dependent frame are coded (as prediction residuals) using the cumulative frequency table $cf\_se11[CTX\_OFFSET + ctx\_t][CTX\_OFFSET + ctx\_f]$, determined by the quantized context values ctx_t and ctx_f.

Please note that the predefined cumulative frequency tables cf_se01, cf_se02, and the table offsets cf_off_se01, cf_off_se02 depend on the current operating point and implicitly on the bitrate, and are selected from the set of available options during initialization of the encoder for each given operating point. The cumulative frequency table cf_se00 is common for all operating points, and cumulative frequency tables cf_se10 and cf_se11, and the corresponding table offsets cf_off_se10 and cf_off_se11 are also common, but they are used only for operating points corresponding to bitrates larger than or equal to 48 kbps, in case of dependent TCX 10 frames (when $t = 1$).

5.3.3.2.11.9 IGF bit stream writer

The arithmetic coded IGF scale factors, the IGF whitening levels and the IGF temporal flatness indicator are consecutively transmitted to the decoder side via bit stream. The coding of the IGF scale factors is described in subclause 5.3.3.2.11.8.4. The IGF whitening levels are encoded as presented in subclause 5.3.3.2.11.6.4. Finally the IGF temporal flatness indicator flag, represented as one bit, is written to the bit stream.

In case of a TCX20 frame, i.e. ($isTCX20 = true$), and no counting request is signalled to the bit stream writer, the output of the bit stream writer is fed directly to the bit stream. In case of a TCX10 frame ($isTCX10 = true$), where two sub-frames are coded dependently within one 20ms frame, the output of the bit stream writer for each sub-frame is written to a temporary buffer, resulting in a bit stream containing the output of the bit stream writer for the individual sub-frames. The content of this temporary buffer is finally written to the bit stream.

5.3.3.2.12   Memory updates

5.3.3.2.12.1   Internal decoder

Subsequent to the MDCT based TCX encoding, a simplified decoding at sr_{celp} is performed to enable and update the filter memories for CELP coding. Based on the quantized MDCT spectral coefficients, the following decoding steps are performed:

- Adaptive low frequency deemphasis (subclause 6.2.2.3.2)

- Global gain decoding (subclause 6.2.2.3.3)

- Residual dequantizer (subclause 6.2.2.3.4)

- Formant enhancement (subclause 6.2.2.3.5)

- Noise filling (subclause 6.2.2.3.6)

- Application of global gain and LPC shaping in MDCT domain (subclause 6.2.2.3.7)

- Inverse window grouping (subclause 6.2.2.3.9)

- Temporal noise shaping (subclause 6.2.2.3.10)

The resulting MDCT spectrum is transformed to a time-domain signal at sr_{celp} using the corresponding frequency-to-time transformation, as described in subclause 6.2.4, resulting in the synthesized TCX decoder output $\hat{s}_{TCX,Enc}(n)$.

5.3.3.2.12.2   Update of filter memories

The updating of the states of the filter $A(z/\gamma_1)H_{de-emph}(z)$ is performed by applying the perceptually weighted LPC coefficients on the synthesized TCX decoder output $\hat{s}_{TCX,Enc}(n)$ and subtract this from the perceptually weighted speech signal $s_h(n)$. The LPC coefficients a_i correspond to the 4th subframe for $sr_{celp} = 12.8kHz$ and to the 5th subframe for $sr_{celp} = 16.0kHz$

$$x_{1,w}(n) = s_h(n) - \hat{s}_{TCX,Enc}(n) + \sum_{i=1}^{16} a_i \, \gamma_1^i \, \hat{s}_{TCX,Enc}(n-i), \qquad n = L_{TCX}^{(celp)} - 1 \qquad (1024)$$

The pre-emphasis-filter $H_{pre-emph}(z) = 1 - \beta z^{-1}$ is applied to $\hat{s}_{TCX,Enc}(n)$ for updating the synthesis buffers used in the target signal computation to calculate the residual signal $r(n)$ (subclause 5.2.3.1.2), and for LPC synthesis filter states used in the CELP encoder. For updating the filter states of LP residual signal computation, the pre-emphasized synthesis buffer is filtered by the quantized LPC coefficients (see subclause 5.2.3.1.1).

5.3.3.2.13   Global Gain Adjuster

For bitrates less than 13.2 kbps the optimum global gain g_{opt} is partially recomputed after noise filling and formant enhancement have been applied in the internal decoder:

$$g_{opt} = g_{opt} \sqrt{\frac{\sum_{k=0}^{L_{TCX}^{(bw)}-1} \left(\hat{X}_M(k)\right)^2}{\sum_{k=0}^{L_{TCX}^{(bw)}-1} \left(\hat{X}'_M(k)\right)^2}} \qquad (1025)$$

The new global gain g_{opt} is quantized again for transmission in the bit stream to an index I_{opt}, replacing the previously computed one.

5.3.4   High Quality MDCT coder (HQ)

5.3.4.1   Low-rate HQ coder

The structure of the Low-rate HQ MDCT core coder is presented in figure 65. Encoding is performed in the MDCT domain. Based on the input signal bandwidth, operational bitrates and signal characteristics the coding modes are decided. For example, an input signal with the sampling frequency of 32 kHz or 48 kHz with operational bitrates of 13.2 kbps and 16.4 kbps will be encoded using any one of the three alternative modes: Transient mode, Normal mode, and Harmonic mode. For an input signal with sampling frequency of 8 kHz or 16 kHz, the tonality estimation block is not used, so the input signal is either encoded under Transient or Normal mode. The mode classification decision is described in subclause 5.3.4.1.1. The following table 101 summarizes the supported modes per bit rate and bandwidth. Mode information (one bit for Transient/non-Transient, and one bit for Normal/Harmonic) is encoded and transmitted to the decoder side.

Table 101: Supported modes for low-rate HQ coder

Bitrate [kbps]	Bandwidth	Supported modes
7.2, 8	NB	Normal, Transient
13.2, 16.4	NB, WB	Normal, Transient
	SWB	Normal, Transient, Harmonic

Based on the mode, the obtained spectral coefficients are grouped into bands of unequal lengths. The energy of each band is estimated and the resulting spectral envelope consisting of the energies of all bands is quantized and encoded using Huffman coding. The quantized energies are used as input for bit allocation. The spectral coefficients are quantized using TCQ and USQ and encoded based on the allocated bits for each frequency band. The encoded spectral coefficients are adjusted using the quantized energies. The level of the most significant spectral coefficients is adjusted using an estimated gain which is coded and transmitted to the decoder. The spectral bands which are not quantized in the HF region are identified and coded with relatively few bits using the quantized spectral coefficients information.

The parameters transmitted from encoder to decoder are the mode selection, energy envelope information, the quanitzed spectral coefficients, LF and the HF parameters.
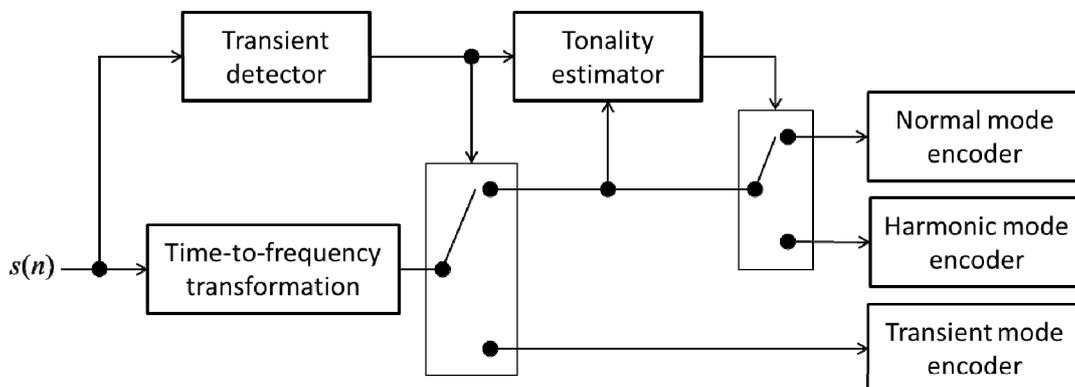
Figure 65: Structural block diagram of the Low-rate HQ coder

5.3.4.1.1 Tonality Estimation

Non transient mode is referred as Normal mode for NB and WB inputs, whereas for SWB and FB inputs at 13.2 and 16.4 kbps, Non-transient signals are further classified as Normal and Harmonic mode. The detailed description for Normal and Harmonic mode classification is as follows,

448 MDCT coefficients in the 2400-13600 Hz frequency range are used in order to perform this classification. 14 sub-bands are split, which is equally 32 coefficients per sub-band. The frequency sharpness $Sharp(j)$ is defined as the ratio between the peak and the average magnitudes in each sharpness band:

$$Sharp(j) = \begin{cases} \dfrac{32 \cdot X_{max}(j)}{\displaystyle\sum_{k=32j}^{32j+31} |X_M(k)|}, & if \quad \displaystyle\sum_{k=32j}^{32j+31} |X_M(k)| \neq 0 \\ 0, & otherwise \end{cases} \qquad j = 3, \cdots, 16 \qquad (1026)$$

where the peak magnitude of spectral coefficients in a sharpness band, denoted $X_{max}(j)$, is:

$$X_{max}(j) = \max_{k=32j, \cdots, 32j+31} |X_M(k)| \qquad j = 3, \cdots, 16 \qquad (1027)$$

The counter c_{sharp_l} denotes the number of sub-bands which have harmonic characteristics corresponding to the first 5 sub-bands. c_{sharp_l} is initialized to zero and increased by one if $Sharp(j) > 4.5$, $j = 3, \cdots, 7$. The counter c_{sharp_h} denotes the number of sub-band which have harmonic characteristics corresponding to the remaining 9 sub-bands. c_{sharp_h} is initialized to zero and increased by one if $Sharp(j) > 3.6$ and $X_{max}(j) > 10$, $j = 8, \cdots 16$.

The mode counters *mode_count* and *mode_count1* are introduced for harmonic mode detection and are initialized to zero. *mode_count* is increased by one and *mode_count1* is decreased by one if $c_{sharp\_l} + c_{sharp\_h} \geq 10$ and $c_{sharp\_h} > 5$. Otherwise, the mode counter *mode_count* is decreased by one and *mode_count1* is increased by one. *mode_count* and *mode_count1* are constrained to only hold values between 0 and 8. The calculations of *mode_count* and *mode_count1* are summarized as follows,

$$\begin{cases} \begin{cases} mode\_count = mode\_count + 1, & if \quad mode\_count < 8 \\ mode\_count1 = mode\_count1 - 1, & if \quad mode\_count1 > 0 \end{cases} & if \ c_{sharp\_l} + c_{sharp\_h} \geq 10 \ and \ c_{sharp\_h} > 5 \\ \begin{cases} mode\_count = mode\_count - 1, & if \quad mode\_count > 0 \\ mode\_count1 = mode\_count1 + 1, & if \quad mode\_count1 < 8 \end{cases} & otherwise \end{cases} \qquad (1028)$$

The current frame mode is classified as harmonic if $mode\_count1 < 5$, $c_{sharp\_l} + c_{sharp\_h} \geq 10$ and $c_{sharp\_h} > 5$; or if $mode\_count \geq 5$.

5.3.4.1.2 Grouping of spectral coefficients

The spectral coefficients are divided into bands of variable lengths; the total number of bands varies depending on the signal bandwidth (NB, WB, SWB, and FB), operational bitrates and the classifier. Tables 103 to 108 describe the band structure for different signal bandwidths and operational bitrates. The NB band structure is used for NB signals, the WB band structure is used for WB signals, and the SWB band structure is used for SWB and FB signals.

The total number of bands and the corresponding bandwidth used for NB, WB and SWB is presented in table 102. The band structure and the number of bands for FB are same as SWB for 13.2 and 16.4 kbps.

In case of the Transient mode, the coefficients of the equivalent four 5-ms transforms are consecutively joined, and the bandwidth varies based on signal bandwidth. table 103, 104, 105 shows the detailed band structure for the NB, WB, and SWB (and FB) Transient frames based on the operational bitrates. In each table, b denotes the index of the band, $k_{width}(b)$ is the corresponding band length, and $k_{start}(b)$ and $k_{end}(b)$ denote the start and end index of the spectral coefficients forming the band.

Table 102: Number of bands and its corresponding bandwidth

	Bitrate (kbps)	Nbands		Bandwidth
		Transient	Normal/Harmonic	
NB	7.2, 8	16 (4*4)	13	160
	13.2	20(4*5)	19	
WB	13.2	28(4*7)	18	320
	16.4		20	
SWB, FB	13.2 ,16.4	32(4*8)	22	568
			24	640

Table 103: Band structure for NB Transient frames

	Transient mode					
	7.2,8 kbps			13.2 kbps		
b	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$	$k_{width}(b)$	$k_{end}(b)$	$k_{end}(b)$
0	6	0	5	6	0	5
1	8	6	13	7	6	12
2	11	14	24	7	13	19
3	15	25	39	9	20	28
4	6	40	45	11	29	39
5	8	46	53	6	40	45
6	11	54	64	7	46	52
7	15	65	79	7	53	59
8	6	80	85	9	60	68
9	8	86	93	11	69	79
10	11	94	104	6	80	85
11	15	105	119	7	86	92
12	6	120	125	7	93	99
13	8	126	133	9	100	108
14	11	134	144	11	109	119
15	15	145	159	6	120	125
16	-	-	-	7	126	132
17	-	-	-	7	133	139
18	-	-	-	9	140	148
19	-	-	-	11	149	159

Table 104: Band structure for WB Transient frames

b	Transient mode 13.2, 16.4 kbps		
	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$
0	6	0	5
1	7	6	12
2	8	13	20
3	10	21	30
4	12	31	42
5	16	43	58
6	21	59	79
7	6	80	85
8	7	86	92
9	8	93	100
10	10	101	110
11	12	111	122
12	16	123	138
13	21	139	159
14	6	160	165
15	7	166	172
16	8	173	180
17	10	181	190
18	12	191	202
19	16	203	218
20	21	219	239
21	6	240	245
22	7	246	252
23	8	253	260
24	10	261	270
25	12	271	282
26	16	283	298
27	21	299	319

Table 105: Band structure for SWB, FB Transient frames

	Transient mode					
	13.2 kbps			16.4 kbps		
b	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$
0	7	0	6	8	0	7
1	8	7	14	9	15	8
2	10	15	24	11	25	17
3	11	25	35	13	36	28
4	15	36	50	17	51	41
5	21	51	71	23	71	58
6	29	72	100	32	99	81
7	41	101	141	47	140	113
8	7	142	148	8	160	167
9	8	149	156	9	168	176
10	10	157	166	11	177	187
11	11	167	177	13	188	200
12	15	178	192	17	201	217
13	21	193	213	23	218	240
14	29	214	242	32	241	272
15	41	243	283	47	273	319
16	7	284	290	8	320	327
17	8	291	298	9	328	336
18	10	299	308	11	337	347
19	11	309	319	13	348	360
20	15	320	334	17	361	377
21	21	335	355	23	378	400
22	29	356	384	32	401	432
23	41	385	425	47	433	479
24	7	426	432	8	480	487
25	8	433	440	9	488	496
26	10	441	450	11	497	507
27	11	451	461	13	508	520
28	15	462	476	17	521	537
29	21	477	497	23	538	560
30	29	498	526	32	561	592
31	41	527	567	47	593	639

In the Normal mode of operation, the bands have different sizes that increase with increasing frequency. This subdivision allows a consistent representation of the spectrum which closely resembles that of the human ear. Higher frequency resolution is used for low frequencies, while lower frequency resolution is used for high frequencies. The detailed allocation of spectral coefficients to bands for NB, WB, SWB and FB signals are presented in tables 106, 107, 108

Table 106: Band structure for NB Normal mode frames

	Normal mode					
	7.2,8 kbps			**13.2 kbps**		
b	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$
0	6	0	5	6	0	5
1	6	6	11	6	6	11
2	6	12	17	6	12	17
3	6	18	23	6	18	23
4	7	24	30	6	24	29
5	8	31	38	6	30	35
6	9	39	47	7	36	42
7	10	48	57	7	43	49
8	13	58	70	8	50	57
9	15	71	85	8	58	65
10	19	86	104	9	66	74
11	24	105	128	10	75	84
12	31	129	159	11	85	95
13	-	-	-	13	96	108
14	-	-	-	15	109	123
15	-	-	-	17	124	140
16	-	-	-	19	141	159

Table 107: Band structure for WB Normal mode frames

	Normal mode					
	13.2 kbps			**16.4 kbps**		
b	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$
0	6	0	5	6	0	5
1	6	6	11	6	6	11
2	6	12	17	6	12	17
3	6	18	23	6	18	23
4	6	24	29	6	24	29
5	7	30	36	6	30	35
6	7	37	43	7	36	42
7	8	44	51	8	43	50
8	10	52	61	8	51	58
9	11	62	72	9	59	67
10	13	73	85	11	68	78
11	16	86	101	12	79	90
12	19	102	120	14	91	104
13	24	121	144	17	105	121
14	30	145	174	20	122	141
15	37	175	211	23	142	164
16	47	212	258	28	165	192
17	61	259	319	34	193	226
18	-	-	-	42	227	268
19	-	-	-	51	269	319

Table 108: Band structure for SWB and FB Normal and Harmonic frames

b	\multicolumn{3}{c}{13.2 kbps}	\multicolumn{3}{c}{16.4 kbps}
	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$	$k_{width}(b)$	$k_{start}(b)$	$k_{end}(b)$
0	6	0	5	6	0	5
1	6	6	11	6	6	11
2	6	12	17	6	12	17
3	6	18	23	6	18	23
4	6	24	29	6	24	29
5	6	30	35	6	30	35
6	7	36	42	7	36	42
7	8	43	50	7	43	49
8	9	51	59	8	50	57
9	10	60	69	9	58	66
10	11	70	80	10	67	76
11	13	81	93	11	77	87
12	16	94	109	13	88	100
13	19	110	128	15	101	115
14	23	129	151	18	116	133
15	28	152	179	21	134	154
16	34	180	213	26	155	180
17	42	214	255	32	181	212
18	55	256	310	39	213	251
19	68	311	378	48	252	299
20	84	379	462	59	300	358
21	105	463	567	74	359	432
22	-	-	-	92	433	524
23	-	-	-	115	525	639

Normal and Harmonic mode

5.3.4.1.3   Energy Envelope coding

Energy envelope coding module is applied for all types of signal i.e., from NB, WB, SWB and FB for various bitrates as described in table 101. In this module, the spectrum energy of a band is computed the differential indices of scalar quantized band energies are encoded using either a Large symbol coding method or a Small symbol coding method. The coding method is selected according to the range required to represent all the differential indices and the bit consumption. The encoding of band energies is detailed below.

The spectrum energy of a band, $E_M(b)$ is computed as follows:

$$E_M(b) = \log_2 \left(\sum_{k=k_{start}(b)}^{k=k_{end}(b)-1} X_M(k)^2 + Epsilon \right), \quad b = 0,...,N_{bands}-1 \tag{1029}$$

In case of transient mode, the energies to be quantized are first reordered such that energy corresponding to even sub-frame index $m = 0, 2$ are in frequency-increasing order while the energy of odd sub-frame index $m = 1, 3$ are in frequency decreasing order which allows for an efficient differential energy encoding

In each frame, the energies are scalar quantized with a uniform scalar quantizer *qint*. The value of *qint* varies and it is selected based on table 109.The index of the quantized energy, $I_M(b)$, can easily be obtained as:

$$I_M(b) = round(\frac{E_M(b)}{q_{int}}), \quad b = 0,...,N_{bands}-1 \tag{1030}$$

Table 109: Scalar quantizer values for NB, WB, SWB, FB modes

BW	Mode	7.2 kbps	8 kbps	13.2 kbps	16.4kbps
NB	Transient	1.8	2.2	1.4	-
	Normal	1	1	0.8	-
WB	Transient	-	-	1.8	1.8
	Normal	-	-	0.8	0.8
SWB, FB	Transient	-	-	3	1.2
	Normal, Harmonic	-	-	0.6	0.6

The quantized indices of the band energies are differentially coded by computing

$$\Delta I_M(0) = I_M(0) - round(\frac{I_{ref}}{q_{int}})$$
$$\Delta I_M(b) = I_M(b) - I_M(b-1), \quad b = 1,...,N_{bands} - 1 \tag{1031}$$

where lowest frequency band quantized index is differentially coded using a reference band energy $I_{ref} = 24$.

The differential indices $\Delta I_M(b)$ are constrained into the range of [–256, 255]. This is performed by first adjusting the negative differential indices and then adjusting the positive differential indices as follows:

$$
\begin{aligned}
&if \ \Delta I_M(b) < -256 \\
&\Delta I_M(b) = -256 \\
&end \\
&if \ \Delta I_M(b) > 255 \\
&\Delta I_M(b) = 255 \\
&end \qquad\qquad\qquad b = 0,...,N_{bands} - 1
\end{aligned}
\tag{1032}
$$

The constrained differential indices are used for selecting the more efficient mode from either the Small symbol coding method or the Large symbol coding method. The method selection between the Small symbol coding mode and the Large symbol coding mode is described in subclause 5.3.4.1.3.2.

Using the coding method information obtained from subclause 5.3.4.1.3.2 differential indices are coded using the respective modes as indicated below.

A flag bit DENG_CMODE which was obtained from subclause 5.3.4.1.3.2 used to indicate the type of encoding method between the Small symbol coding method and the Large symbol coding method and transmitted as side information to the decoder. The flag DENG_CMODE is set to 1 when the Small coding method is used and it is set to zero for the Large symbol coding method and it is described in table 110.

If the flag DENG_CMODE is set to 1, in the Small symbol coding the band energies are either coded by resized or context based Huffman coding. A flag bit LC_MODE is used to indicate the mode selection between resized or context based Huffman coding and it is transmitted as side information to the decoder. The mode selection between resized and context based coding is done based on the estimated number of bits consumed by the respective coding modes. The resized Huffman coding described in subclause 5.3.4.1.3.3.2 is used for coding the differential indices when LC_MODE is set to 1 and LC_MODE is set to 0 for coding the differential indices using context based coding which is described in subclause 5.3.4.1.3.3.1.

5.3.4.1.3.1 Reconstruction of quantized energies

The quantized differential indices are reconstructed according to

$$
\begin{aligned}
I'_M(0) &= \Delta I_M(0) + I_{ref} \\
I'_M(b) &= \Delta I_M(b) + I'_M(b-1) \quad b = 1,..,N_{bands} - 1
\end{aligned}
\tag{1033}
$$

where $I_{ref} = 24$ is the reference band energy The final resulting reconstructed quantized energies are obtained as follows

$$\hat{I}_M(b) = I'_M(b)q_{int}, \quad b = 0,...,N_{bands} - 1 \tag{1034}$$

where the value of *qint* varies and it is selected based on table 109.

If band energies are quantized under transient mode i.e., *IsTransient* is True, the energies are reordered back to original.

5.3.4.1.3.2 Energy envelope coding mode selection

The differential quantization indices are encoded by one of two coding methods. The coding method is selected according to the range required to represent all the differential indices and the bit consumption. The range of the large symbol coding method enables it to represent larger number of bits. The large symbol coding method consists of a scale

mode and a pulse mode. The small symbol coding method uses an upper bit coding method which consists of a context based Huffman coding mode and a re-sized Huffman coding mode as well as bit packing for the lower bit.

Table 110: Low-rate HQ envelope coding modes

Coding Method index *DENG_CMODE* (1bit)	Coding Method	Coding Mode index (1bit)	Description
0	Large symbol method	0	Pulse mode
		1	Scale mode
1	Small symbol method	0	Context based Huffman coding mode
		1	Re-sized Huffman coding mode

If at least one of the differential quantization indices in all the bands of a frame cannot be represented in [-32, 31]([-46,17] for the first index), the large symbol coding method is always used. If this larger range is not required the bits consumption for both large and small symbol coding modes are compared and the coding mode with the least bits is selected. The corresponding coding method information is transmitted for each frame.

The Small symbol coding method and the large symbol method are used for estimating the bits consumption for coding the differential indices which are obtained in equation (1031). The detailed descriptions of the respective coding modes are given in the following subclause 5.3.4.1.3.3.

The number of estimated bits hcode1 and flag LCmode information obtained from subclause 5.3.4.1.3.3. is used for selecting the best mode with the Large symbol coding method bits ULbits obtained from subclause 5.3.4.1.3.4, as shown below.

$$If\ ULbits < hcode_1\ OR\ hcode_1 = -1$$
$$DENG\_CMODE = 0$$
$$else$$
$$DENG\_CMODE = 1$$

5.3.4.1.3.3   Small symbol coding method

If *IsTransient* is *True*,

In this module, the quantized indices obtained from equation (1030) are constrained into the range of [–15, 16] by calculating the differences as in equation (1031) and constraining the range. This is performed by first adjusting the negative differential indices and then adjusting the positive differential indices as follows:

1. Compute the differential indices for lowest frequency band using reference band energy as in equation (1031)

2. For b = 0, if $\Delta I_M(0) > 16, I_M(0) = I_{ref} + 16$ and $\Delta I_M(0) < -15, I_M(0) = I_{ref} - 15$

3. For the rest of the bands, compute the differential indices defined in equation (1031) in order from the highest-frequency band to the lowest-frequency band.

4. If $\Delta I_M(b) < -15, I_M(b-1) = I_M(b) + 15, b = N_{bands} - 1, ...1$

5. Re-compute the differential indices in order from the lowest-frequency sub-vector from band b=1 to the highest-frequency sub-vector.

6. If $\Delta I_M(b) > 16, \Delta I_M(b) = 16\ and\ I_M(b) = I_M(b-1) + 16, b = 1, ...N_{bands} - 1$

7. The adjusted differential indices in the range [0, 31] are obtained by adding an offset of 15 to $\Delta I_M(b)$.

Context based Huffman coding mode is used for estimating the bit consumption, if the range of differential indices lies in between [10, 22] resized Huffman coding [b-Huffman] mode is enabled for estimating the bits consumption. The Huffman codes for the differential indices for resized Huffman coding mode when *IsTransient* is *True* are given in table 111. In the table 111, H_i denotes the index of the Huffman code, H_c is the Huffman code corresponding to index H_i and H_b denotes the bits required for representing the Huffman code corresponding to index H_i.

Table 111: Huffman code for Transient frames

Hᵢ	H_c	H_b	Hᵢ	H_c	H_b	Hᵢ	H_c	H_b	Hᵢ	H_c	H_b
0	0	0	8	0	0	16	11	2	24	0	0
1	0	0	9	0	0	17	0010	4	25	0	0
2	0	0	10	0	0	18	011010	6	26	0	0
3	0	0	11	1111010	7	19	00111010	8	27	0	0
4	0	0	12	01010	5	20	010111010	9	28	0	0
5	0	0	13	110	3	21	110111010	9	29	0	0
6	0	0	14	01	2	22	0	0	30	0	0
7	0	0	15	00	2	23	0	0	31	0	0

The estimated bits for context based coding which was obtained from subclause 5.3.4.1.3.3.1 is represented as $hcode_1$, while for resized coding it is represented as $hcode_2$, a best coding mode is selected based on

$$
\begin{aligned}
&\text{If } hcode_1 >= hcode_2 \text{ } AND \text{ } hcode_2 \text{ } != 0 \\
&\quad LC \bmod e = 1 \\
&\quad hcode_1 = hcode_2 \\
&\text{end}
\end{aligned}
$$

If *IsTransient* is *False*,

The differential quantization indices are adjusted to have positive values by adding 46 in the first band and 32 in the other bands. The differential quantization indices are split into 5 upper bits and 1 lower bit. The 5 upper bits are encoded by either a context based Huffman coding mode described in subclause 5.3.4.1.3.3.1 or a re-sized Huffman coding mode described in subclause 5.3.4.1.3.3.2 and the 1 lower bit is packed.

In more detail, the context based coding mode or the resized Huffman coding is used for estimating the bits. The differential indices which are obtained in equation (1031) $\Delta I_M(b)$ are constrained into the range of [0, 63] by adding an offset of 32 to $\Delta I_M(b)$ for $b = 1,….,N_{bands}$-1 and for $b=0$ an offset of 46 is used for $\Delta I_M(0)$. If the constrained differential indices exceed [0 63] when *IsTransient* is *False* and [0 31] when *IsTransient* is *True*, $hcode_1$ is set to -1 and the differential indices are coded using the Large symbol coding method.

The least significant bit is extracted from the constrained differential indices $\Delta I_M(b)$ for $b = 0,….,N_{bands}$-1 using

$$
\begin{aligned}
R_{LSB} &= \Delta I_M(b) \text{ } AND \text{ } 1 (\text{Binary operation}) \\
\Delta I_M(b) &= \frac{\Delta I_M(b)}{2}
\end{aligned}
\tag{1035}
$$

The updated differential indices are used for estimating the bits consumed by the two different coding modes. Based on the estimated bits obtained from context based coding, $hcode_1$, and resized Huffman coding mode, $hcode_3$, the best coding mode is selected as shown below.

$$
\begin{aligned}
&\text{If } hcode_1 >= hcode_3 \text{ } AND \text{ } hcode_3 != 0 \\
&\quad LC \bmod e = 1 \\
&\quad hcode_1 = hcode_3 \\
&\text{end}
\end{aligned}
$$

The differential indices $\Delta I_M(0)$ is usually transmitted as is for both *IsTransient* is *True* or *False*, i.e., 5 bits per norm index is required and this bits are updated to the estimated bits $hcode_1$ which was shown below.

$$
hcode_1 = hcode_1 + 5
\tag{1036}
$$

The least significant bit extracted from the constrained differential indices is transmitted as is, if the Small symbol coding method is selected and $hcode_1$ is updated as shown below.

$$
hcode_1 = hcode_1 + bands
\tag{1037}
$$

By default flag LCmode is set to 0 and represents context based coding, while if flag LCmode is reset to 1, it indicates resized Huffman coding mode consumes less bits compared to context based coding, and the estimated bits $hcode_1$ is used for selecting the best mode with the Large symbol coding method bits ULbits.

5.3.4.1.3.3.1 Context based Huffman coding mode

If the this coding mode is selected for the current frame, the context based Huffman coding is applied to the adjusted differential indices. These indices are encoded using a context model which corresponds to the adjusted differential index in a previous band. The first band must be handled separately, so the context for encoding $\Delta I_M(0)$ is adjusted by subtracting $offset_{HQ,norm}(=3)$

$$ctx_{HQ,norm}(\Delta I_M(b)) = \begin{cases} I_M(0) - offset_{HQ,norm} & for\ b = 0 \\ \Delta I_M(b-1) & for\ b = 1,\ldots,N_{bands}-2 \end{cases} \qquad (1038)$$

There are three groups depending on the $ctx_{HQ,norm}(\Delta I_M(b))$ and two probability models as shown in table 112, $group_0$ and $group_2$ share the Huffman tables depending on the probability model.

Table 112: The groups and the probability models

Group index	Lower bound	Upper bound	Probability model
0	-	12	0
1	13	17	1
2	18	-	0

A Huffman table for $group_0$ and $group_2$ is defined in table 113 and a Huffman table for $group_1$ is defined in table 114.

If $ctx_{HQ,norm}(\Delta I_M(b))$ is located in $group_2$, $\Delta I_M(b)$ is reversed to $31 - \Delta I_M(b)$ and then the reversed value is encoded.

Table 113: Huffman coefficient table for the group0 Huffman coding (group0,group2)

Index	Code	Index	Code	Index	Code	Index	Code
0	11100111101111	8	11100110	16	000	24	111011
1	11100111101110	9	1110100	17	010	25	0110011
2	1110011110110	10	0110010	18	110	26	1110010
3	111001111010	11	100100	19	0111	27	1110101
4	1001010001	12	01101	20	1111	28	1001011
5	1110011111	13	1000	21	10011	29	10010101
6	111001110	14	101	22	011000	30	1001010000
7	100101001	15	001	23	111000	31	11100111100

Table 114: Huffman coefficient table for the context based Huffman coding (group1)

Index	Code	Index	Code	Index	Code	Index	Code
0	0010000100110	8	001000101	16	11	24	0010000101
1	001000100110	9	00100110	17	100	25	00100001000
2	00100111010	10	0010010	18	1011	26	001000010010
3	00100010010	11	101000	19	10101	27	00100111011
4	00100010001	12	00101	20	101001	28	001000100111
5	00100010000	13	0011	21	00100000	29	00100001001110
6	0010011100	14	000	22	00100011	30	001000010011110
7	001001111	15	01	23	001000011	31	001000010011111

5.3.4.1.3.3.2 Resized Huffman coding mode

Resized Huffman coding is applied to the adjusted differential indices obtained from equation (1035). In this method, the span of the differential indices is reduced while being able to perfectly reconstruct the differential indices. Based on the newly modified differential indices obtained from equation (1040), number of bits consumed for coding the new differential indices $\Delta I'_M(b)$ is estimated as shown below.

$$hcode_3 = \sum_{b=1}^{N_{bands}-1} H_b(\Delta I'_M(b)) \tag{1039}$$

5.3.4.1.3.3.3 Differential Indices Modification

The modification of differential indices is done according to the value of the differential index for the preceding sub band and a threshold. Equation (1040) is used for modifying the span of differential indices.

$$
\begin{aligned}
&\text{if } \Delta I_M(b\text{-}1) > 17, \\
&\quad \Delta I'_M(b) = \Delta I_M(b) + \min(\Delta I_M(b\text{-}1) - 17, 3) \\
&\text{end} \\
&\text{if } \Delta I_M(b\text{-}1) < (13), \\
&\quad \Delta I'_M(b) = \Delta I_M(b) + \max(\Delta I_M(b\text{-}1) - 13, -3) \\
&\text{end} \qquad\qquad\qquad\qquad b = 2, ..., N_{bands} - 1
\end{aligned}
\tag{1040}
$$

where,

$\Delta I_M(b)$ means the differential index for band b

$\Delta I_M(b-1)$ means the differential index for band $b-1$

$\Delta I'_M(b)$ means the new differential index for band b

Based on the new differential indices obtained from equation (1040), resized Huffman coding is applied, if any of the new differential indices lies outside [0 31] range, resized Huffman coding is not used for coding the differential indices.

The range of the new differential indices for Huffman coding is identified as shown below.

$$Range = [Range_{Min}, Range_{Max}] = [Min(\Delta I'_M(b)), Max(\Delta I'_M(b))] \tag{1041}$$

where $b = 1, ... N_{bands}$-1

Based on the range obtained from equation (1041), range difference is calculated as shown below.

$$Range_{Diff} = Max(15 - Range_{Min}, Range_{Max} - 15) \tag{1042}$$

Resized Huffman coding is used for coding the new differential indices if the $Range_{Diff}$ values lies below 11, otherwise resized Huffman coding is not used. The Huffman codes and its corresponding bits consumption for coding of the new differential indices are given in table 115.

Table 115: Huffman code for Non-Transient frames

H_i	H_c	H_b	H_i	H_c	H_b	H_i	H_c	H_b	H_i	H_c	H_b
0	0	0	8	001111111	9	16	10	2	24	1011111111	10
1	0	0	9	00111111	8	17	101	3	25	1111111111	11
2	0	0	10	0011111	7	18	1011	4	26	0	0
3	0	0	11	001111	6	19	10111	5	27	0	0
4	0	0	12	00111	5	20	101111	6	28	0	0
5	01111111111	11	13	0011	4	21	1011111	7	29	0	0
6	0111111111	10	14	001	3	22	10111111	8	30	0	0
7	0011111111	10	15	00	2	23	101111111	9	31	0	0

5.3.4.1.3.4 Large symbol coding method

If the large symbol coding method is used then either the pulse mode or the scale mode is selected to encode the differential quantization indices. The pulse mode is adequate when no differential quantization index is over [-4,3]. If this range is exceeded, the pulse mode cannot be used, and instead the scale mode is always used. Additionally, if the first quantization differential index is over [-64,63], the scale mode is always used. In the large symbol coding method a Huffman coding with 8 symbols shown in table 116 is used.

Table 116: Huffman coefficient table in the large symbol coding method

Index	Code
-4	0001011
-3	00011
-2	001
-1	01
0	1
1	0000
2	000100
3	0001010

5.3.4.1.3.4.1 Pulse mode

In the pulse mode, there are two indicators; an indicator ind_{I_0} to show whether the first index is transmitted separately and an indicator ind_{pls} to show if there is a differential quantization index exceeding the range [-4,3].

If the first index is within [-4,3], ind_{I_0} is set to 0 and the first index is then encoded by the Huffman coding defined in table 116 with the other indices. Otherwise, ind_{I_0} is set to 1 and the first index is then packed using 7 bits after adding 64.

If a pulse exists in the current frame, ind_{pls} is set to 1 and the pulse position pls_{pos} and amplitude pls_{amp} are transmitted using 5 bits and 7 bits respectively. All the other indices are then encoded by the Huffman coding in table 116. If no pulse exists, all indices are encoded by the Huffman coding in table 116.

Table 117: bit allocation for the scale mode

	cmd_0	cmd_1	ind_{I_0}	ind_{pls}	$\Delta I_M(0)$	pls_{pos}	pls_{amp}	Huffman bits
bits	1	1	1	1	7	5	7	-

5.3.4.1.3.4.2 Scale mode

In the scale mode, all the indices are split into 3 upper bits and a few lower bits depending on the minimum and maximum of all the indices. The 3 upper bits are encoded by the Huffman coding in table 116 and the lower bits are packed. The number of lower bits is defined as bit_{shift}. The bit_{shift} is calculated to make all the differential quantization indices fit within the range [-4,3] by scaling down the indices, and is represented by three bits.

5.3.4.1.4 MDCT coefficients quantization

5.3.4.1.4.1 Normal Mode

5.3.4.1.4.1.1 Overview

The figure below shows the overview of the normal mode encoder.
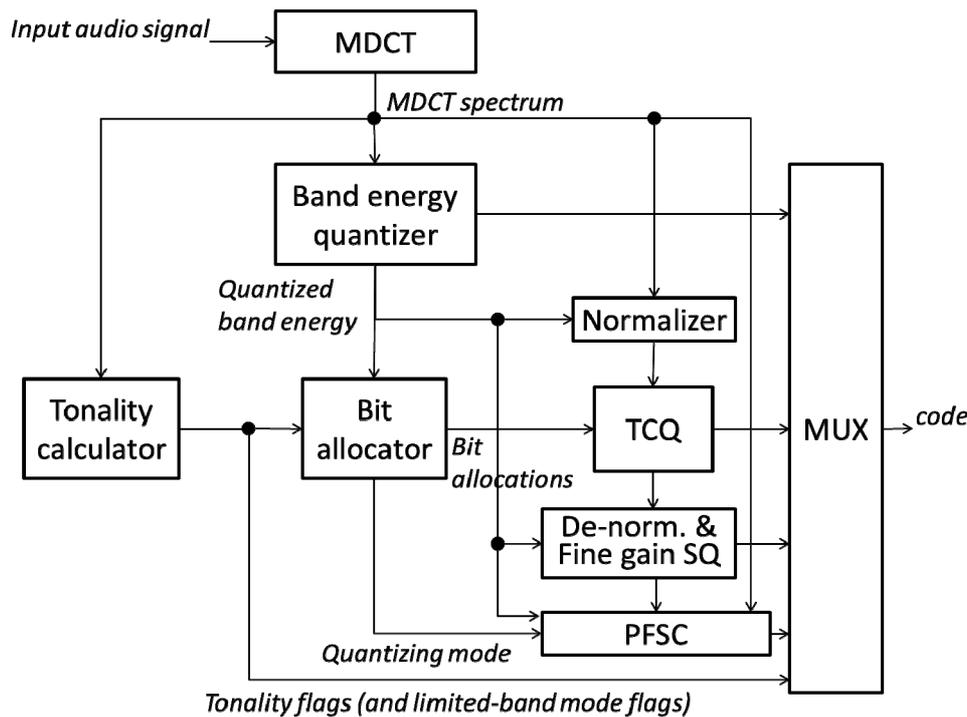
Figure 66: Block diagram of the Normal mode encoder overview

5.3.4.1.4.1.2   Energy envelope coding

Details are described in subclause 5.3.4.1.3.

5.3.4.1.4.1.3   Tonality flag calculation

Tonality flags are calculated for the high bands $b = N_{bands} - h_b$, .., $N_{bands}-1$ as described in table 118. For example, for the last (highest) five bands, i.e. b=17 to 21 for 13.2 kbps and b=19 to 23 for 16.4kbps in table 108, and the last three bands b=15 to 17 for 13.2 kbps and b=17 to 19 for 16.4kbps in case of WB as in table 107 and last two bands b=15 to 16 for 13.2 kbps for NB inputs as in table 106, peak-to-average ratios are calculated and compared with a threshold. Flags indicating whether the peak-to-average ratios are greater than the threshold are sent to the decoder side using one bit per band. If the peak-to-average ratio is greater than the threshold, the tonality flag is set to "1", otherwise it is set to "0". For the SWB and FB case, a limited-band mode flag is further sent to the decoder side as described in subclause 5.3.4.1.4.4.2, if the tonality flag is set to "1".

Table 118: Total number of high bands for tonality calculation

Bandwidth	Bitrate (kbps)	High bands , h_b
NB	13.2	2
WB	13.2,16.4	3
SWB, FB	13.2,16.4	5

5.3.4.1.4.1.4   Bit allocation

5.3.4.1.4.1.4.1   Bit allocation overview

In the Normal Mode, band spectra are encoded by either Trellis Coding Quantization (TCQ) or Pitch Filtering Spectrum Coding (PFSC) with assigned bits for the bands. TCQ is used for encoding peaky/tonal spectrum bands for NB, WB, SWB and FB, while PFSC is mainly applied for encoding other spectrum bands in a high-frequency region other than NB. This switching principle is analogous to the coding mode switching between Generic mode and Sinusoidal mode in G.718 Annex B [25].

Bit-allocation process is performed in the following manner. Firstly, bands encoded using PFSC are identified based on the tonality flags among the four highest bands in case of SWB ,FB and peak-to-average ratio is calculated for last band in the WB and necessary bits (1 or 2 bits) are allocated to each of the identified bands. Secondly, remaining bits are allocated to other bands based on perceptual importance. When there is any band whose assigned bit results is zero in

the four bands for SWB and FB signals, such band is re-identified as a PFSC encoding band and the bit allocations are re-calculated.
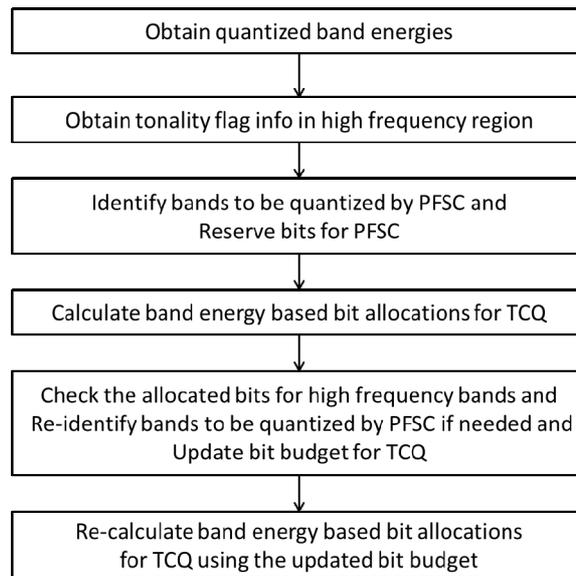
Figure 67: A flowchart of bit allocation processing for LR-HQ Normal mode.

5.3.4.1.4.1.4.2   The adjustment of quantized energy envelope prior to bit allocation

For the Non-Transient mode of NB (the bit rate is less than or equal to 13.2kbps) and WB cases, in order to make the inter-frame reconstruction more continuous and allocate more bits to perceptual important bands, the quantized energy envelopes of the highest h_b bands are adjusted prior to bit allocation. Some of the quantized energy envelopes of the high frequency bands and low frequency bands are adjusted. Then perform the bit allocation to bands according to the adjusted energy envelopes. Finally, the coefficients of the bit allocated bands are quantized and written to the bit-stream.

To adjust the quantized energy envelope of high frequency bands at NB, the following steps are performed:

a) Two bits are encoded to indicate whether the highest two bands of the previous frame is encoded. Initialize a band boundary L_{bands} to 6 and initialize adjustment factors α, β respectively:

$$\alpha = \begin{cases} 0.85 & for\ 7.2/8.0/9.6kbps \\ 0.95 & for\ 13.2kbps \end{cases}$$

$$\beta = \begin{cases} 1.05 & for\ 7.2/8.0kbps \\ 1.1 & for\ 9.6kbps \\ 1.25 & for\ 13.2kbps \end{cases} \tag{1043}$$

b) For each band, calculate the magnitude envelope:

$$Env_{av}(b) = \sqrt{\frac{\hat{E}_M(b)}{k_{width}(b)}}, \qquad b = 0,1,\cdots,N_{bands}-1 \tag{1044}$$

where $k_{width}(b)$ denotes the bandwidth of each band, and $\hat{E}_M(b)$ is computed as follows:

$$\hat{E}_M(b) = 2^{\hat{I}_M(b)} \tag{1045}$$

where $\hat{I}_M(b)$ denotes the quantized energy envelope of each band.

c) Then, calculate the sum of the differences between the consecutive two magnitude envelopes and the sum of the magnitude envelopes for high frequency bands:

$$Env_{av}^{\text{var}} = \sum_{b=L_{bands}}^{N_{bands}-1} \left| Env_{av}(b) - Env_{av}(b-1) \right| \tag{1046}$$

$$Env_{av}^{sumH} = \sum_{b=L_{bands}}^{N_{bands}-1} Env_{av}(b) \tag{1047}$$

d) Search the peak of magnitude envelopes and calculate the sum of the magnitude envelopes for low frequency bands:

$$Env_{av}^{peakL} = \max(Env_{av}(b)), \qquad b = 0,1,\cdots,L_{bands}-1 \tag{1048}$$

$$Env_{av}^{sumL} = \sum_{b=0}^{L_{bands}-1} Env_{av}(b) \tag{1049}$$

e) Initialize an adjustment factor $fac(b)$ for each band to 1, and then adjust the quantized energy envelopes of high frequency bands as follows:

1) if $\quad ((N_{bands}-L_{bands})*Env_{av}^{peakL} < 2.2*Env_{av}^{sumH}) AND ((N_{bands}-L_{bands})*Env_{av}^{peakL} > 0.5*Env_{av}^{sumH})$ is satisfied at the bit-rates below 13.2kbps or

$L_{bands}*Env_{av}^{peakL} < 4*Env_{av}^{sumL} AND$

$((N_{bands}-L_{bands})*Env_{av}^{peakL} < 2.2*Env_{av}^{sumH}) AND ((N_{bands}-L_{bands})*Env_{av}^{peakL} > 0.5*Env_{av}^{sumH})$ is satisfied at 13.2kbps,

the quantized energy envelopes of the last $N_{bands}-L_{bands}$ bands are adjusted as follows:

$$\hat{I}'_M(b) = \hat{I}_M(b)*(\min(\max(0.4 + \frac{Env_{av}^{sumH}}{(N_{bands}-L_{bands})*Env_{av}^{peakL}},0.5),\alpha)), \quad if\ Env_{av}(b) < 1.5*Env_{av}^{sumH} \tag{1050}$$

2) Otherwise, the adjustment factors of the last 2 bands are calculated:

$$fac(b) = 1 + \min(\min(\frac{0.1*Env_{av}^{sumH}}{(N_{bands}-L_{bands})*Env_{av}(b)},0.2)*\frac{Env_{av}^{\text{var}}}{Env_{av}^{sumH}},0.4), \qquad if\ T_{flag}(b)=1 \tag{1051}$$

Then, the adjustment factors of the last 2 bands are updated further according to $C_{flag}(b)$:

$$fac(b) = \begin{cases} fac(b)*(\min(\max(\dfrac{Env_{av}^{sumH}}{(N_{bands}-L_{bands})*Env_{av}(b)},1),\beta)), & if\ C_{flag}(b)=1 \\[4mm] fac(b)*(\min(\max(\dfrac{(N_{bands}-L_{bands})*Env_{av}(b)}{Env_{av}^{sumH}},0.85),1)), & otherwise \end{cases} \tag{1052}$$

Finally, the quantized energy envelopes of the last 2 bands are adjusted: $\hat{I}'_M(b) = \hat{I}_M(b)*fac(b)$,

where $T_{flag}(b)$ is the tonality flag which is calculated in subclause 5.3.4.1.4.1.3, i.e. the classification mode of the band, and $C_{flag}(b)$ is obtained from the previous frame and indicates whether the bits are allocated to the highest two bands of the previous frame or not. If $C_{flag}(b)$ is equal to 1, the band b of the previous frame is allocated bits; Otherwise, if $C_{flag}(b)$ is equal to 0, the -band b of the previous frame is not allocated bits -. After bit allocation, the flag $C_{flag}(b)$ of the current frame is preserved for the next frame.

To adjust the quantized energy envelops of low frequency bands at NB, select L_{bands} low frequency bands from N_{bands} bands, and the following steps are performed:

a) Initialize L_{bands} to 3. For each band, calculate the magnitude envelope:

$$Env_{av}(b) = \sqrt{\frac{\hat{E}_M(b)}{k_{width}(b)}}, \qquad b = 0,1,\cdots,N_{bands}-1 \qquad (1053)$$

b) Then, calculate the sum of the magnitude envelopes for high frequency bands:

$$Env_{av}^{sumH} = \sum_{b=L_{bands}}^{N_{bands}-1} Env_{av}(b) \qquad (1054)$$

the variable N_{bands} is assigned to different value for different bit rates, specifically it is assigned to 13,14,15,17 for 7.2kbps, 8kbps, 9.6kbps and 13.2kbps respectively.

c) Search the peak of magnitude envelopes and calculate the sum of the magnitude envelops for low frequency bands:

$$Env_{av}^{peakL} = \max(Env_{av}(b)), \qquad b = 0,1,\cdots,L_{bands}-1 \qquad (1055)$$

$$Env_{av}^{sumL} = \sum_{b=0}^{L_{bands}-1} Env_{av}(b) \qquad (1056)$$

d) Initialize a flag SEC_{flag} to 0. SEC_{flag} is a flag to indicate whether second stage bit allocation algorithm is used in TCQ module. When SEC_{flag} is equal to 0, second stage bit allocation algorithm will be used. When SEC_{flag} is equal to 1, the second stage bit allocation algorithm will not be used. The flag is utilized in subclause 5.3.4.1.4.1.5.1.2.

e) Determining whether to modify the energy envelops of the three low frequency bands according to their energy characteristics and spectral characteristics. The energy characteristics denote the ratio between the energy of three low frequency bands and the energy of the other bands which are determined by Env_{av}^{sumH} and Env_{av}^{sumL}. The spectral characteristics denote the degree of spectrum fluctuation which are determined by Env_{av}^{peakL}.

f) If the following conditions are satisfied,

$$(\frac{1}{6} < \frac{Env_{av}^{sumL}}{Env_{av}^{sumH}} < \frac{1}{1.5}) AND(\frac{Env_{av}^{peakL}}{Env_{av}^{sumL}} > \frac{1}{0.575 * L_{bands}})$$, the flag SEC_{flag} is set to 1, and the quantized energy

envelops of the three low frequency bands are adjusted as follows:

$$\hat{I}'_M(b) = \hat{I}_M(b) * \min(\frac{0.575 * Env_{av}^{peakL} * L_{bands}}{Env_{av}^{sumL}},1.2), \qquad b = [0, L_{bands}-1] \qquad (1057)$$

To adjust the quantized energy envelopes of high frequency bands at WB, the following steps are performed:

a) Encode two bits to indicate whether the highest two bands of the previous frame is encoded. Define two band boundaries L_{bands} and H_{bands}. If the bit rate is 13.2kbps, set L_{bands} and H_{bands} to 8 and 15, respectively; Otherwise, set L_{bands} and H_{bands} to 8 and 16, respectively. The bandwidths of low frequency bands and high frequency bands are obtained as follows:

$$bw_L = \sum_{b=L_{bands}}^{H_{bands}-1} k_{width}(b)$$

$$bw_H = \sum_{b=H_{bands}}^{N_{bands}-1} k_{width}(b)$$

(1058)

b) For each band, calculate the magnitude envelope:

$$Env_{av}(b) = \sqrt{\frac{\hat{E}_M(b)}{k_{width}(b)}}, \qquad b = 0,1,\cdots,N_{bands}-1$$

(1059)

Then, calculate the sum of the differences between the consecutive 2 magnitude envelopes and the sum of the magnitude envelopes for part of the high frequency bands:

$$Env_{av}^{var} = \sum_{b=L_{bands}}^{N_{bands}-P_{bands}-1} \left| Env_{av}(b) - Env_{av}(b-1) \right|$$

(1060)

$$Env_{av}^{sumH} = \sum_{b=L_{bands}}^{N_{bands}-P_{bands}-1} Env_{av}(b)$$

(1061)

c) The energies of low frequency bands and high frequency bands are computed as follows:

$$ener_L = \sum_{b=L_{bands}}^{H_{bands}-1} \hat{E}_M(b)$$

$$ener_H = \sum_{b=H_{bands}}^{N_{bands}-1} \hat{E}_M(b)$$

(1062)

d) Obtain adjustment factors $fac(b)$ for the highest h_b bands according to the tonality flag $T_{flag}(b)$ and the energies of low frequency bands and high frequency bands:

$$fac(b) = \begin{cases} 1+\min(\min(\dfrac{2*(N_{bands}-h_b-L_{bands})*Env_{av}(b)}{Env_{av}^{sumH}},0.2)*\dfrac{Env_{av}^{var}}{Env_{av}^{sumH}},0.4), & if\ T_{flag}(b)=1\ OR\ \dfrac{ener_H}{bw_H} > \dfrac{ener_L}{bw_L} \\ 1, & otherwise \end{cases}$$

(1063)

and for the highest $h_b - 1$ bands, update the adjustment factors $fac(b)$ according to $C_{flag}(b)$:

$$fac(b) = \begin{cases} fac(b)*\min(\max(\dfrac{Env_{av}^{sumH}}{(N_{bands}-h_b-L_{bands})*Env_{av}(b)},1),1.25), & if\ C_{flag}(b)=1 \\ fac(b)*\min(\max(\dfrac{(N_{bands}-h_b-L_{bands})*Env_{av}(b)}{Env_{av}^{sumH}},0.85),1), & otherwise \end{cases}$$

(1064)

and then adjust the quantized energy envelopes of the highest h_b bands $\hat{I}'_M(b) = \hat{I}_M(b)*fac(b)$.

To adjust the quantized energy envelopes of low frequency bands at WB, the following steps are performed:

a) Initialize a low frequency band boundary L_{bands} to 6. For each band, calculate the magnitude envelope:

$$Env_{av}(b) = \sqrt{\frac{\hat{E}_M(b)}{k_{width}(b)}}, \qquad b = 0,1,\cdots,N_{bands}-1 \qquad (1065)$$

b) Then, calculate the sum of the magnitude envelopes for high frequency bands:

$$Env_{av}^{sumH} = \sum_{b=L_{bands}}^{N_{bands}-1} Env_{av}(b) \qquad (1066)$$

the variable N_{bands} is assigned to different value for different bit rate, specifically it is assigned to 18, 20 for 13.2kbps and 16.4kbps respectively.

c) Search the peak of magnitude envelopes and calculate the sum of the magnitude envelops for low frequency bands:

$$Env_{av}^{peakL} = \max(Env_{av}(b)), \qquad b = 0,1,\cdots,L_{bands}-1 \qquad (1067)$$

$$Env_{av}^{sumL} = \sum_{b=0}^{L_{bands}-1} Env_{av}(b) \qquad (1068)$$

d) Determining whether to modify the quantized energy envelops of the six low frequency band according to their energy characteristics and spectral characteristics. The energy characteristics denote the ratio between the energy of six low frequency bands and the energy of the other bands which are determined by Env_{av}^{sumH} and Env_{av}^{sumL}; The spectral characteristics denote the degree of spectrum fluctuation which are determined by Env_{av}^{peakL}.

If the conditions $(Env_{av}^{sumH} > 2*Env_{av}^{sumL})AND(Env_{av}^{sumH} < 6*Env_{av}^{sumL})AND(0.5*Env_{av}^{peakL}*L_{bands} > Env_{av}^{sumL})$ or $(Env_{av}^{sumL} > 2*Env_{av}^{sumH})AND(0.5*Env_{av}^{peakL}*L_{bands} > Env_{av}^{sumL})AND(Env_{av}^{sumL} < 1.2*Env_{av}^{peakL})$ are satisfied, the flag $SEC_{flag} = 1$, and the quantized energy envelops of the six low frequency bands are adjusted as follows:

$$\hat{I}'_M(b) = \hat{I}_M(b)*\min(\frac{0.5*Env_{av}^{peakL}*L_{bands}}{Env_{av}^{sumL}},1.2), \qquad b = [0, \ L_{bands}-1] \qquad (1069)$$

Finally, the adjusted quantized energy envelopes $\hat{I}'_M$ and the initial quantized envelopes $\hat{I}_M(b), b = [L_{bands}, N_{bands}-1]$ are used in the first bit allocation module.

5.3.4.1.4.1.4.3   Bit allocation for PFSC

Based on the tonality flags, whether TCQ is used for encoding the band in the high frequency region is determined. When the tonality flag is set to "0", such band is excluded from target bands of the TCQ, i.e. no bit is assigned to the band for TCQ.

In the Normal Mode, the four bands in the high-frequency region for SWB/FB and one band for WB are assumed to be quantized by TCQ in default operation. However, when the tonality flag is set to "0", such band is quantized with the PFSC scheme using a similar procedure with the "sub-band search" (called as "band search" in this specification) in [25], whereas for WB cases such band is filled with noise. This means the peaky/tonal bands are encoded by the TCQ while the PFSC scheme is used for other bands.

In Figure 66, whether the PFSC is used for quantizing the high-frequency bands is indicated by 'Quantizing mode'.

In the PFSC scheme, the band search is based on a pitch filter based prediction ($\hat{X}(i) = \hat{X}(i-T)$, where T is a pitch coefficient, and low-frequency spectrum is used as its filter state (filter memory)), and the pitch coefficient (i.e. lag information as a filter parameter) is encoded. The lag information is encoded with 2 bits or 1 bit. Lower two bands among the four bands are encoded with 2 bits, while higher two bands among the four are encoded with 1 bit. When the PFSC is selected for encoding some of the four bands, necessary bits for encoding those bands are reserved and assigned for the bands before starting the bit-allocation process for TCQ encoding bands.

5.3.4.1.4.1.4.4   Bit allocation for TCQ

5.3.4.1.4.1.4.4.1   Allocating bits for fine gain adjustment

In the TCQ based MDCT coefficients quantization, fine gain adjustment is applied to several bands whose energies are larger than the others. The number of those bands is configured based on encoding bit-rate and signal bandwidth as shown in table 119.

Table 119: Bits reserved for fine gain adjustments

	7.2 kbit/s	8.0 kbit/s	13.2 kbit/s	16.4 kbit/s
NB	2 bands, $R_{FG}=2$	2 bands, $R_{FG}=2$	4 bands, $R_{FG}=4$	2 bands, $R_{FG}=2$
WB	-	-	6 bands, $R_{FG}=6$	6 bands, $R_{FG}=6$
SWB	-	-	4 bands, $R_{FG}=4$	4 bands, $R_{FG}=4$

One-bit scalar quantization is used for the fine gain adjustment. Therefore the number of reserved bits R_{FG} for NB, WB, SWB, and FB is shown in the table 119. The bits for the TCQ are obtained by subtracting the fine gain bits, and expressed by $\psi = R - R_{FG} - e_{bits}$, where ψ is the available bit budget for spectrum coding, e_{bits} is the bits consumed for quantizing the band energies which is obtained from subclause 5.3.4.1.3, and R is the total bits. It should be noted that the mode bits for switching Transient/non-Transient and Normal/Harmonic are included in the available bit budget. Therefore the necessary bits (1 or 2) are subtracted from the available bit budget in the following subclauses, i.e. $\psi = \psi - 1$ (NB and WB cases) or $\psi = \psi - 2$ (SWB and FB cases).

5.3.4.1.4.1.4.4.2   Limited-band mode

For the SWB case, bandwidth, $k_{width}(b)$, is more than 50 bins for the last four bands (i.e. b=18 to 21 in 13.2 kbps and b=20 to 23 in 16.4 kbps). At low bit-rates, such wide bandwidth would result in insufficient quantization performance. Therefore a limited-band mode is introduced for achieving efficient encoding. In the limited-band mode, only the vicinity of a perceptually important spectrum in each band is targeted to be quantized. This is realized by the following principle. A bandwidth is adaptively shortened if the maximum amplitude spectrum frequency falls into a range of frequencies around the maximum amplitude spectrum frequency in the previous frame. The difference between the maximum amplitude spectrum frequencies for the current and previous frame is calculated. When it is smaller than a threshold, the limited-band mode is used for such band. The frequency position of the maximum amplitude spectrum is searched for each of the four bands. The position for the previous frame is stored in a memory, which was searched using the quantized MDCT spectrum in the previous frame. The position for the current frame is searched using the MDCT spectrum calculated from the input signal in the current frame. When the limited-band mode is selected, the targeted band is limited to the vicinity of the maximum amplitude spectrum frequency of the previous frame. The range of the vicinity is 15 or 31 spectrum bins depending on coding bit-rate and band index. One bit is used for indicating whether the limited band mode is used when tonality flag is set to "1".

The limited-band mode can be used only in the case where the corresponding previous band was quantized by TCQ.

5.3.4.1.4.1.4.4.3   Final bit allocation for TCQ and PFSC

Band widths and quantized band energies are used for bit allocation. The band widths are basically configured by Table 108. When the limited-band mode is selected, corresponding band widths are shortened. Bit allocation for TCQ is as follows. Firstly, bits are distributed according to the quantized band energy. Secondly, if there is a band whose assigned bits are less than a minimum number of bits, no bit is assigned to the band and the assigned bits will be re-allocated to the other bands. Thirdly, if there is a noise-like band whose assigned bits are not sufficient in comparison with its bandwidth, no bit is assigned to the band and the assigned bits will be re-allocated to the other bands. Furthermore, assigned bits are compared with predefined band-based threshold, and no bit is re-assigned if the assigned bits are less than the threshold for the band.

In case no bit is assigned to any band among the highest four bands for a SWB and FB signals, PFSC is used for quantizing such band. This case can happen when the tonality flag is set to "1" but assigned bit results in zero. In this case, necessary bits for the PFSC encoding for the band is extracted from the bit budget for TCQ, ψ. Therefore ψ is updated by subtracting the necessary bits, and the bit-allocation is re-calculated.

5.3.4.1.4.1.5   Fine structure encoding

5.3.4.1.4.1.5.1   Trellis Coding Quantization (TCQ)

5.3.4.1.4.1.5.1.1   Joint USQ and TCQ

Trellis Coded Quantization (TCQ) quantizes the fine structure of normalized spectrum, selecting the Important Spectral Components (ISCs). The information for the selected ISCs in each band is coded as the position, number, sign and magnitude of the ISCs. The magnitude information is quantized by the joint Uniform Scalar Quantization (USQ) and TCQ with arithmetic coding, while the information on position, number and sign is coded by arithmetic coding. A block diagram of the fine structure encoding using TCQ is depicted by the figure 68.
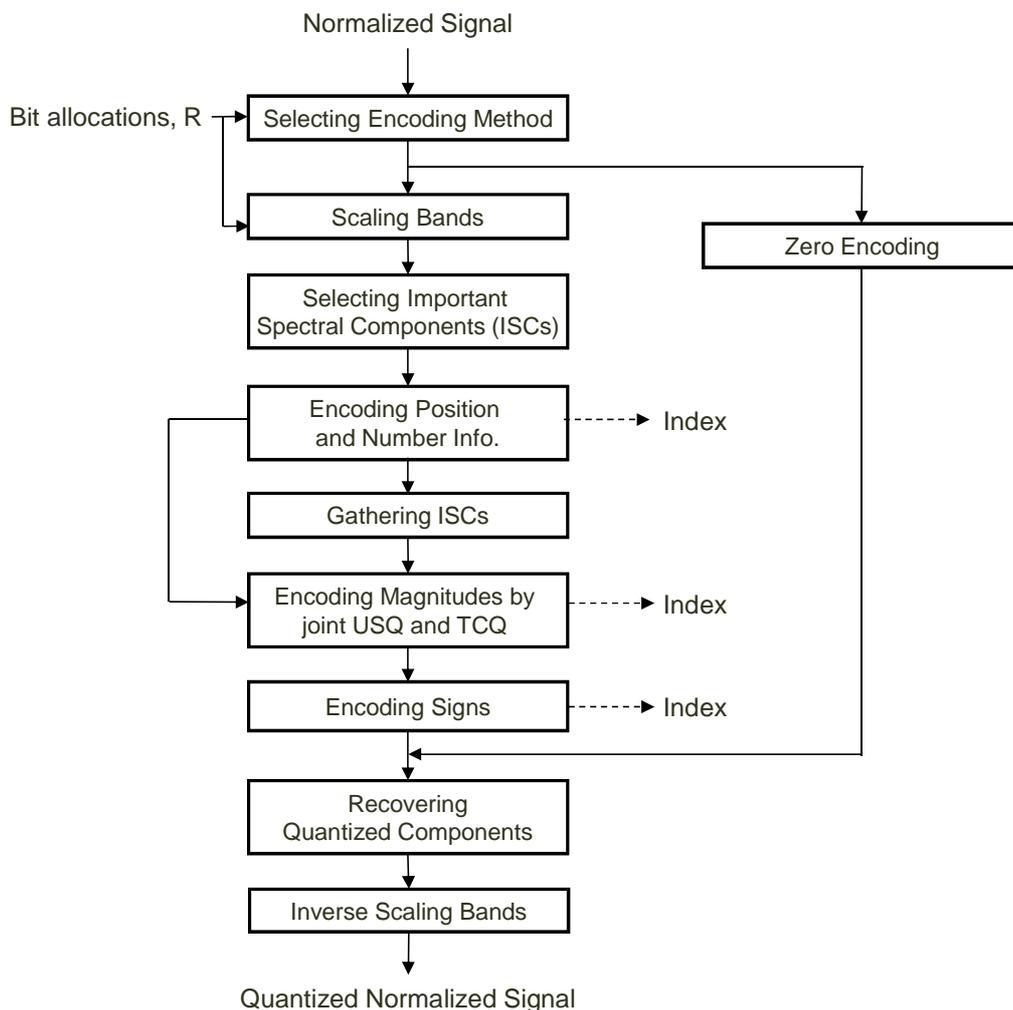
Figure 68: Block diagram of fine structure encoding using TCQ

The encoding method is selected at the Selecting Encoding Method block by the bit allocation and the information for each band. If a bit allocated for a band is zero, all the samples in that band are coded to zero by the zero encoding block. Otherwise, each band is quantized by the selected quantizer.

The quantizer selection information selects the most efficient quantizer between the USQ and TCQ by considering the input signal characteristics, i.e. the bit allocation and the length of each band. If the average number of bits for each sample in a band is greater or equal to 0.75, the band is of high importance and USQ is used; for all other bands TCQ is used. The quantizer selection flag, USQ_TCQ[i], is set to 1 when USQ is used.

The Scaling Bands block performs scaling at each band to control the bit rate, using the bit allocations and the normalized spectrum for each band. The scaling is done by considering the average bit allocation for each spectral component in the band. If the average bit allocation is bigger than the number specified by the bit allocation, then more scaling is done.

The detailed scaling process is as follows. First the estimation of the number of pulses for the current band is obtained using the length and the bit allocation information for each band. Then the number of nonzero positions is obtained by the following equation, which is based on the probabilities.

$$pNZP(i) = 2^{i-b} \, C_n^i C_{m-1}^{i-1}, \quad i \in \{1, \ldots, \min(m, n)\} \tag{1070}$$

where b is the number of bits and calculated as:

$$b = \log_2 \left(\sum_{i=1}^{\min(m,n)} 2^i \, \frac{n!}{(n-i)!\, i!} \frac{(m-1)!}{(i-1)!\,(m-i)!} \right) \tag{1071}$$

and the number of required bits for the positions is estimated as:

$$b_{nzp} = \log_2 \big(pNZP(i) \big) \tag{1072}$$

where n is the band length, m is the number of pulses, i is number of non-zero positions which have an ISC, and b is the number of bits required for the given size of band and number of pulses. Finally the number of pulses is selected by the b value which is the closest to the value of the allocated bits for the band.

The initial scaling factor is decided by the estimation of the number of pulses and the absolute value of the input signal. The input signal for each band is scaled by this factor. If the estimated number of pulses is not the same as the summation of the number of pulses for the quantized signal after scaling, a pulse redistribution process will be performed using the updated scaling coefficient. The redistribution process is as follows: if the number of selected pulses is smaller than the number of estimated pulses, the scaling coefficient will be decreased, otherwise the scaling coefficient will be increased.

The distortion function for the TCQ is sum of squared distance of each quantized and un-quantized value in each band. It is similar to the Euclidean distance but avoids the square root, since only the relative magnitudes of the values are needed rather than the exact distance.

$$d^2 = \sum_{i=1}^{n} (p_i - q_i)^2 \tag{1073}$$

where p_i is actual value and q_i is quantized value.

For the USQ module the Euclidean distance is used to determine the best quantized values. To minimize the computational complexity, the modified equation including the scaling factor is used, which is done by calculating the d_1 as:

$$d_1 = \sqrt{\sum_{i=1}^{n} (p_i - g_1 q_i)^2} \tag{1074}$$

If the number of pulses per band does not match the required value, it is necessary to add or delete some pulses while preserving the minimal metric. This procedure is done in an iterative manner by adding or deleting a single pulse, and then repeating until the number of pulses reaches the required value.

To add or delete one pulse it is necessary to calculate n values of distortions in order to select best one For example, the distortion value j corresponds to adding pulse at j-th position in a band:

$$d_2^j = \sqrt{\sum_{i=1}^{n} (p_i - g_2 \hat{q}_i)^2}, \, j = 1 \ldots n \tag{1075}$$

To avoid the full calculation of this formula n times, the following derivation can be used:

$$d_2^j = \sqrt{\sum_{i=1}^{n}(p_i - g_2\hat{q}_i)^2} = \sum_{i=1}^{n}p_i^2 - 2g_2\sum_{i=1}^{n}p_i\sum_{i=1}^{n}\hat{q}_i + g_2^2\sum_{i=1}^{n}\hat{q}_i^2 = \left\{\sum_{i=1}^{n}\hat{q}_i = \sum_{i=1}^{n}q_i + 1\right\},$$

$$\left\{\sum_{i=1}^{n}\hat{q}_i^2 = \sum_{i\in\{1...n\},i\neq j}q_i^2 + (q_j+1)^2 = \sum_{i=1}^{n}q_i^2 + 2q_j + 1\right\} = \tag{1076}$$

$$= \sum_{i=1}^{n}p_i^2 - 2g_2\left(\sum_{i=1}^{n}q_ip_i + p_j\right) + g_2^2\left(\sum_{i=1}^{n}q_i^2 + 2q_j + 1\right), \quad j = 1...nn$$

where values $\sum_{i=1}^{n}q_i^2, \sum_{i=1}^{n}q_ip_i, \sum_{i=1}^{n}p_i^2$ can be calculated just once, n is the band length (number of coefficients in a band), p is the input signal of the quantizer , q is the quantized signal, and g is the scaling factor. Finally the position j, which minimized the distortion d, is selected and q_j is updated.

To control the bit rate, an appropriate ISC has to be selected using the scaled spectral coefficients at the Selecting Important Spectral Components (ISCs) block. The spectral component to quantize is selected by using the bit allocations for each band. This selection can have various combinations which depend on the distribution and variance of the spectral component. The actual non-zero position is then calculated. The non-zero position is obtained by analyzing the amount of scaling and redistribution operations. The non-zero position will be the ISC.

If the number of pulses is not controlled by the scaling and the redistribution operations, the selected pulse will be quantized by TCQ and the surplus adjusted with the results of the quantization. Hence, if the number of non-zero position is greater than 1, and not equal to the estimated number of the pulse, and the quantizer selection information indicates the TCQ, the surplus redistribution process will be used. If this condition is satisfied, the surplus will be redistributed by the TCQ quantization operation in advance. If the real number of the pulses from the TCQ quantization is smaller than the estimated number of pulses which was derived for each band in advance, the scale factor will be multiplied by 1.1. However, if the number of pulses from the TCQ quantization is bigger than the estimated number of pulses, the scale factor will be multiplied by 0.9.

The final selected non-zero position is called by ISC whose information is encoded at the Encoding Position Info block. The information consists of the number of the selected ISC and the non-zero positions. In order to enhance the efficiency of the information encoding arithmetic coding is used.

Given a stream of symbols and their probabilities, the arithmetic coder produces a space efficient bit-stream to represent these symbols and, given the bit stream and the probabilities, the arithmetic decoder reverses the process.

In the arithmetic coding algorithm two 16 bits integers are taken as the numerators of fractions called "low" and "high". These fractions have a common denominator equal to $2^{16}-1$, such that all the fractions fall in the range [0,1) . These integers define a range so that a single number stores all the symbols of the message. This number is saved bit by bit to the bit stream during the arithmetic coding process. To avoid precision loss, at each step of coding process renormalization is performed when difference between 'low' and 'high' is less than 0.5.

In the decoder, 16 bits will first be read and stored in the arithmetic decoder accumulator. The decoder can then replicate the coding process, but instead of encoding the values, it uses the bit-stream to produce symbols. These symbols will be reconstructed correctly, if their probabilities are the same in both the encoder and decoder.

In the Gathering ISCs block, the new buffer is constructed by the selected ISCs, as shown in figure 69. The zero-band and the position which is not selected are both excluded from this buffer.

In the Joint USQ and TCQ Coding block, the magnitudes of the gathered ISCs are quantized by the joint USQ and TCQ, and the quantized information is additionally coded by arithmetic coding. In order to enhance the efficiency of the arithmetic coding, the non-zero position and the number of ISCs is utilized for the arithmetic coding. The joint USQ and TCQ have two types of coding methods. One is TCQ and USQ with 2nd bit allocation for the NB and WB, and the other one is the LSB TCQ for USQ for the SWB and FB. These methods are described in subclause 5.3.4.1.4.1.5.1.2 TCQ and USQ with second bit allocation and subclause 5.3.4.1.4.1.5.1.3 LSB TCQ for USQ.

Joint USQ and TCQ Encoding + Lossless encoding (band by band)
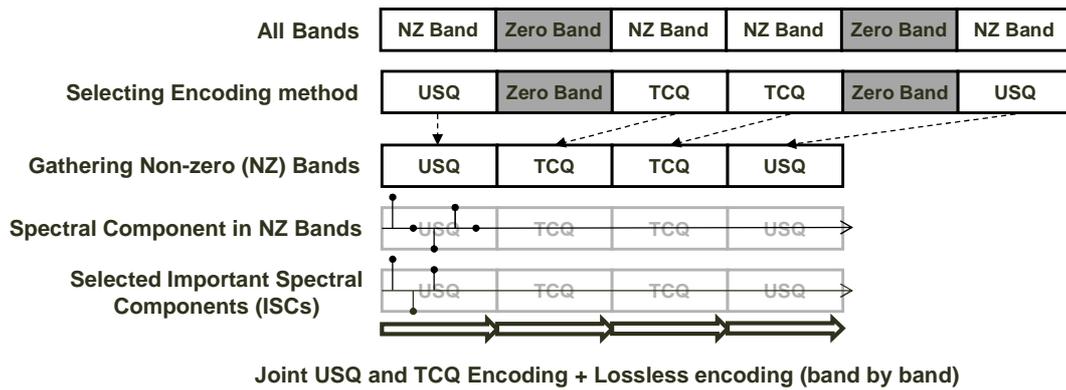
Figure 69: Concept for the gathering ISCs

In the Encode Signs block, the sign information of the selected ISC is coded by the arithmetic coding. In order to recover the quantized components, the position, sign and magnitude information is added to the quantized components to recover the real components at the Recovering Quantized Coefficients block. In this block, the zero is allocated to the zero positions.

In the Inverse Scaling Bands block, the inverse scaling of the quantized components is performed. The inverse scaling factor can be extracted by using the scaling factor in the Scaling Bands block. The inverse scaled signal is same level as that of the normalized input signal and the signal is the output of the TCQ quantization.

5.3.4.1.4.1.5.1.2 TCQ and USQ with second bit allocation

The following information is needed for TCQ and USQ with second bit allocation:

(a) Determine the total number of bits which will be allocated to the corresponding bands to be processed in current frame in subclause 5.3.4.1.4.1.4.4.1.

(b) In order to obtain the number of bits for each band by first bit allocation, implement the first bit allocation to the bands based on the total number of bits to be allocated in subclause 5.3.4.1.4.1.4.4.3.

(c) Based on the number of bits for each band by first bit allocation, the first detailed scaling process in subclause 5.3.4.1.4.1.5.1.1 is implemented on each band which is allocated bits by first bit allocation. Then, the total number of redundant bits of the current frame and the number of pulses of each band are obtained.

The general quantization and coding scheme of the TCQ and USQ with second bit allocation consists of several main blocks: quantizer decision, TCQ quantizer, USQ quantizer, lossless coder, and second bit allocation. In the quantizer decision module the quantization mode of the current band is selected by using the results of the Selecting Encoding Method block. Then the selected quantizer quantizes the current band, and the lossless encoder based on the arithmetic coding transmits the data to the bit stream. The second bit allocation block distributes surplus bits from the previously coded bands. The second bit allocation procedure detects two bands that will be encoded separately.
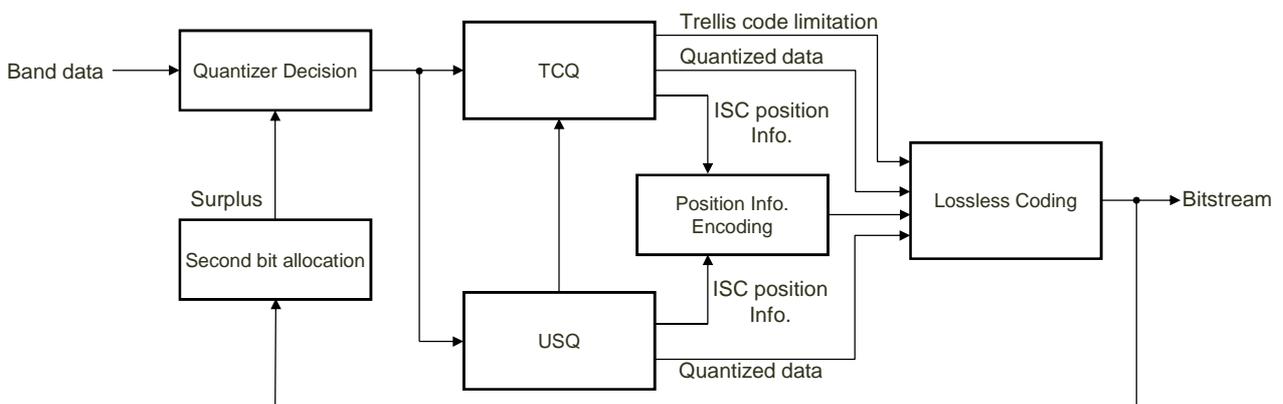
Figure 70: Block diagram of TCQ and USQ encoding with second bit allocation

The second bit allocation procedure analyzes two second bit allocation parameters. The two second bit allocation parameters include the total number of redundant bits and the characteristics of each band. Then, based on the second bit allocation parameters, select the two bands that will be allocated bits once again and quantized last from the bands to be processed in current frame. The characteristics of each band include the harmonic characteristics and the bit allocation state of each band. The tonality flags for the bands of current frame which are calculated in subclause 5.3.4.1.4.1.3 represent the harmonic characteristics of each band, and whether the highest two bands of the previous frame is quantized represents the bit allocation state of each band. If the tonality flag is equal to one, the signal type of corresponding band is harmonic. Otherwise, the signal type of corresponding band is not harmonic. If the tonality flags are not all zero or any one of the highest two bands of the previous frame is quantized, the two bands to be quantized last will be finally selected in the highest few bands. Otherwise, they will be selected in other bands than the highest few ones.

The first of these two bands is selected using the analysis of the average number of bits per bin in the band by the first bit allocation:

For each band, calculate the average number of bits per bin as follows:

$$R_{av}(b) = \frac{R(b)}{k_{width}(b)}, \quad b = 0,1,\cdots,N_{bands}-1 \tag{1077}$$

where $R(b)$ is the number of bits allocated to each band by the first bit allocation based on the total number of bits in the subclause 5.3.4.1.4.1.4.4.3, and $k_{width}(b)$ is the bandwidth of each band.

In order to determine the first band, the average number of bits per bin of each band will be compared. And the bands should satisfy the following conditions:

$$((b > N_{bands} - P_{bands}) \; AND \; (T_{flag}(b) = 1)) \;\; OR \;\; ((b > N_{bands} - 2) \; AND \; (C_{flag}(b) = 1)) \tag{1078}$$

where $T_{flag}(b)$ is the tonality flag, and $C_{flag}(b)$ indicates whether the highest two bands of the previous frame is quantized. If $C_{flag}(b)$ is equal to 1, the band b of the previous frame is quantized, else if $C_{flag}(b)$ is equal to 0, the band b of the previous frame is not quantized.

If the average number of bits per bin in band b is the least one in the bands which satisfy the above conditions, the band b will be determined as the first band for a second bit allocation.

When there are no bands satisfy the above conditions or the encoded bandwidth is NB, the first band for a second bit allocation will be selected during the bands which satisfy the following conditions:

$$(b < N_{bands} - P_{bands}) \; AND \; (R(b) > 0) \tag{1079}$$

If the average number of bits per bin in band b is the least one in the bands which satisfy the above conditions, the band b will be determined as the first band for a second bit allocation.

For the first band b for a second bit allocation, if $b = 0$ or $b = N_{bands} - 1$, then the band $b+1$ or the band $b-1$ will be selected as the second band for a second bit allocation. Otherwise, for the band $b+1$ and the band $b-1$, if the average number of bits per bin of the band $b+1$ is less than the band $b-1$, the band $b+1$ will be selected as the second band for a second bit allocation, if not, the band $b-1$ will be selected as the second band for a second bit allocation.

Thus implement the second bit allocation on the two selected bands to be allocated bits once again. When the total number of redundant bits is replenished, the redundant bits are allocated to the two bands selected above, with the first one being allocated the majority, or all, of the surplus. The number of bits allocated to the two bands is obtained respectively and the proportion of the surplus allocated to each band is shown in the following:

When the encoded bandwidth is NB:

$$if\ (R_{surplus} > 8)$$
$$R_{\sec}(0) = 0.75 * R_{surplus}$$
$$R_{\sec}(1) = 0.25 * R_{surplus}$$
$$else \qquad\qquad (1080)$$
$$R_{\sec}(0) = R_{surplus}$$
$$R_{\sec}(1) = 0$$
$$end$$

When the encoded bandwidth is WB:

$$if\ (R_{surplus} > 12)$$
$$R_{\sec}(0) = 0.75 * R_{surplus}$$
$$R_{\sec}(1) = 0.25 * R_{surplus}$$
$$else \qquad\qquad (1081)$$
$$R_{\sec}(0) = R_{surplus}$$
$$R_{\sec}(1) = 0$$
$$end$$

where $R_{surplus}$ denotes the bit surplus, $R_{\sec}(0)$ and $R_{\sec}(1)$ denotes the number of bits allocated to the first and the second band for a second bit allocation respectively.

At last, based on the number of bits allocated to the two selected bands by the first bit allocation and the number of bits allocated to the two selected bands by the second bit allocation, implement the second detailed scaling process on each of the two selected bands same as the first detailed scaling process in subclause 5.3.4.1.4.1.5.1.1. So, the number of pulses for each of the two selected bands is obtained again.

In the TCQ quantization module a trellis is used with 8 states, 4-coset (subsets) with 2 zero levels. The quantization indexes are derived from the TCQ codebook, which consists of the branch information of a trellis state (path information) and the information for quantization level allocated to the selected coset (subset). The quantization indexes are always positive integers and are coded by arithmetic coding. The detailed magnitude encoding is as follows.
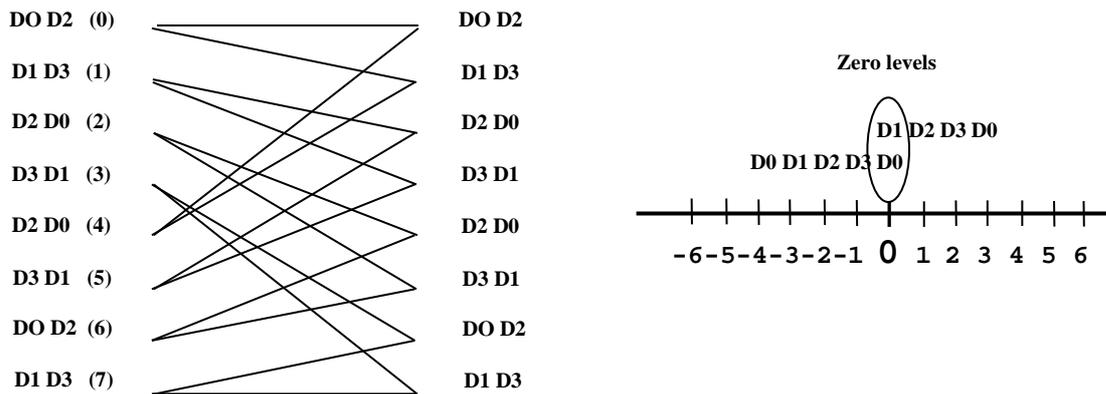
Figure 71: Trellis structure with 8 states 4-coset with 2 zero levels

Each quantized band starts from state (0), because in other cases it is required to transmit two additional bits per band. Then, by using the Viterbi algorithm, the optimal path through trellis is selected, in order to have best possible SNR when comparing the original and quantized sequences. To calculate the SNR, it is necessary to have quantization scale before starting TCQ quantization, so the band is initially quantized by USQ and the optimal scale is calculated. After the lossless encoding the bit surplus is accumulated.

The accumulated surplus is used as an additional resource for the encoding of the last two bands determined by the second bit allocation procedure. Depending on the coding mode and surplus value, the surplus is shared between these two bands in different proportions. Then the encoding of these two bands is performed in the same way as described above.

The magnitude coding based on binary arithmetic coding is carried out as follows. First the probability of a symbol is calculated as

$$\hat{p}_1 = \begin{cases} \dfrac{\hat{i}-1}{\hat{m}-j-1}, & if \ (j-1) \in M_s \\ 0, & otherwise \end{cases} n \qquad (1082)$$

$$\hat{p}_0 = 1 - \hat{p}_1 \qquad (1083)$$

where $\hat{i}$ is the number of magnitudes left to transmit in the band, $\hat{m}$ is the number of pulses left to transmit in the band, j is the current coded pulse in magnitude and M_s is the set of existing magnitudes at trellis state s.

Each magnitude pulse is encoded by using probabilities $\hat{p}_1$ and $\hat{p}_0$, where $\hat{p}_1$ corresponds to last pulse in magnitude and $\hat{p}_0$ to all other pulses.

The magnitude pulse probabilities are modified after this calculation with respect to the trellis code limitation. This information is determined by the trellis structure and is available to both the encoder and decoder. Thus any magnitude values that are impossible are encoded using zero probability and hence do not require any bits.

The encoding algorithm was modified to save complexity for bands with a large number of pulses. The idea is to introduce a non-binary arithmetic coder, where the probabilities of a coded symbol are calculated as the mutual probability of the binary symbols for the current magnitude. To avoid very low probabilities and hence loss of precision, escape symbols are utilized. After an escape symbol transmitted, the rest of the magnitude pulses transmitted are started from initial probabilities condition.

The location coding is carried out based on same algorithm as used for the magnitude coding and uses the same complexity reduction technique. The signs and LSB path vector are transmitted as is, because the distributions are random.

5.3.4.1.4.1.5.1.3   LSB TCQ for USQ

The aim of the LSB TCQ for USQ is to use the advantages of both quantizers (USQ and TCQ) in one scheme and exclude the path limitation from the TCQ. Conceptually the LSB coding of the quantized data is shown in figure 72.
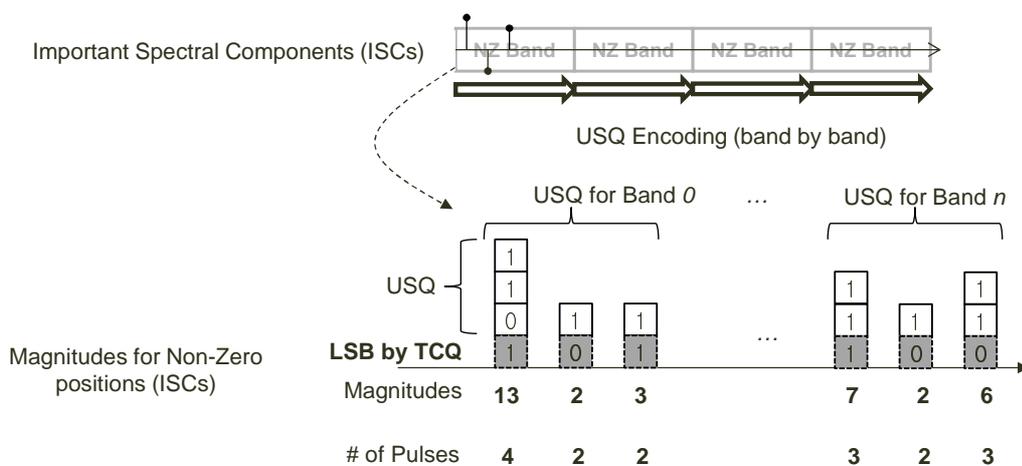
Figure 72: Concept of LSB coding

Each quantized value that is greater than one contains an LSB which can be zero or one. The sequence of LSBs can then be quantized by TCQ to find the best match between that sequence and the available trellis paths. In terms of the SNR

criteria, it does not matter where the error occurs. Thus at the cost of some errors in the quantized sequence, the length of the sequence is decreased.

The encoding of the spectral data is done by the TCQ and USQ quantizers and lossless coding based on the binary arithmetic coder. Before processing the norms, the extraction and spectrum normalization are done. The combined quantizer scheme is presented at figure 73.
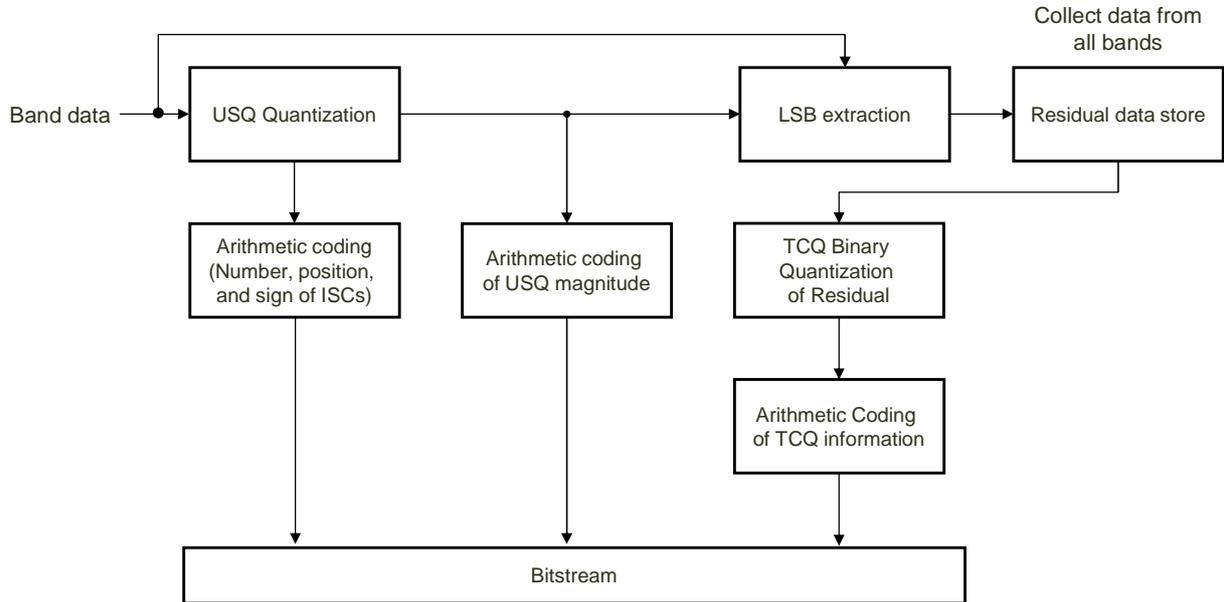
Figure 73: Block diagram for LSB TCQ for USQ encoding

The spectral data in each band are quantized by the USQ quantizer with the number on bits $R[]$ determined by the bit allocation module. In order to fit the bit requirement, the number of bits which will be used for TCQ data is extracted from each non-zero band evenly, and then the bands are quantized by the USQ. The quantization procedure is the same as described above for the TCQ and USQ algorithm. All bands that have non-zero data after quantization are collected as the difference between the quantized and un-quantized data and are called the residual. If some frequencies are quantized as zero in a nonzero band, they are not included into residual.
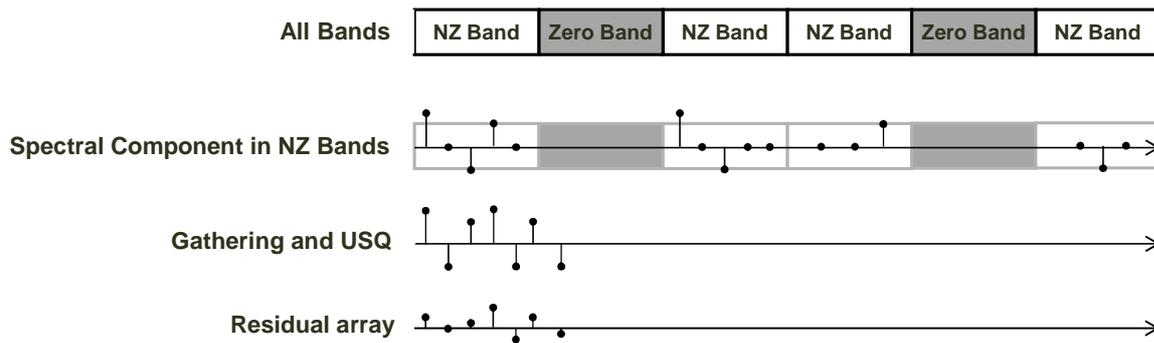
Figure 74: Concept for constructing the residual array

The residual array is quantized by TCQ with code rate ½ known $(7,5)_8$ code:
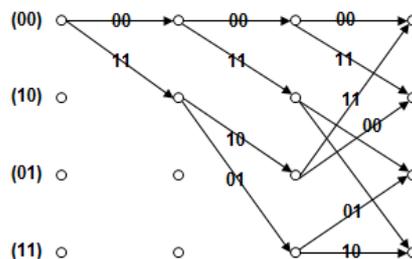
Quantization using TCQ is performed for the first $2 \cdot TCQ\_AMP$ magnitudes. After quantization the path metrics are checked and the best one selected. For lossless coding the data for the best trellis path is stored in separate array while the trace back procedure is performed. The constant TCQ_AMP was defined as 10, which allows up to 20 magnitudes per frame to be encoded.

The trellis path data is encoded by the arithmetic encoder as equi-probable symbols. The path data is binary sequence which is encoded using an arithmetic encoder with a uniform probability model.

The quantized spectral data produced by the USQ are encoded by the same method as described in subclause 5.3.4.1.4.1.5.1.2

The quantized MDCT coefficients are de-normalized using the quantized band energies.

Finally, as described in subclause 5.3.4.1.4.1.4.3.1, quantization of fine gain adjustment is performed on the dominant bands. The inner product between target MDCT coefficients and the de-normalized quantized MDCT coefficients is calculated, and fine gain adjustment factor is calculated by dividing the inner product by the energy of the de-normalized quantized MDCT coefficients. The fine gain adjustment factor is quantized by a 1-bit scalar quantizer.

5.3.4.1.4.1.5.2 Noise-filling for 0-bit assigned sub-bands

Decode the MDCT coefficients of each sub-band from the received bit stream. In order to reconstruct the un-decoded MDCT coefficients, the sub-bands are classified into the bit allocation saturated sub-bands and the bit allocation un-saturated sub-bands according to the average number of bit allocation to a coefficient in a sub-band. The noise filling module is applied to the un-saturated sub-bands. Firstly the noise gain for noise filling of each sub-band is calculated. Then, fill the appropriate noise into the un-decoded MDCT coefficients of each sub-band, and the un-decoded MDCT coefficients are reconstructed. At last, the whole frequency domain signal is obtained based on the decoded MDCT coefficients and the reconstructed MDCT coefficients.

5.3.4.1.4.1.5.2.1 Parameters calculation for noise gain

Initialize a critical number of the sub-band N_{cb} to N_{bands}. If the encoded bandwidth is SWB and the encoding mode is harmonic or normal, N_{cb} is updated to 19 at 16.4kbps and N_{cb} is updated to 17 at 13.2kbps.

Calculate the average number of allocated bits for each coefficient in each sub-band:

$$R_{av}(b) = \frac{R(b)}{k_{width}(b)}, \qquad b = 0,1,\cdots,N_{bands}-1 \tag{1084}$$

where $R(b)$ denotes the number of bits allocated to each sub-band, and $k_{width}(b)$ denotes the bandwidth of each sub-band.

Calculate the average envelope of each sub-band:

$$Env_{av}(b) = \sqrt{\frac{\hat{I}_M(b)}{k_{width}(b)}}, \qquad b = 0,1,\cdots,N_{bands}-1 \tag{1085}$$

where $\hat{I}_M(b)$ denotes the quantized energy of each sub-band.

The maximum magnitude of the decoded coefficients in each sub-band $\hat{X}_{\max}(b)$ and the number of the non-zero decoded coefficients in each sub-band $C_{nz}(b)$ are determined by the decoded coefficients.

Suppose $npulse(b)$ denotes the number of the encoded pulses of each sub-band and $npulse(b) \neq 0$ means the average number of bit allocation to a coefficient in a sub-band is larger than 0. And the energy difference $E_{diff}(b)$ is obtained as follows:

$$E_{diff}(b) = \hat{I}_M(b) - \sum_{k=k_{start}(b)}^{k_{end}(b)} \tilde{X}_M(k)^2 \tag{1086}$$

If $npulse(b) \neq 0$, compute the initial noise gain of each sub-band $\hat{I}'_{nf}(b)$ by the energy difference $E_{diff}(b)$ and sub-band bandwidth $k_{width}(b)$:

$$\hat{I}'_{nf}(b) = \begin{cases} \sqrt{\dfrac{E_{diff}(b)}{k_{width}(b)}}, & E_{diff}(b) > 0 \\ 0, & E_{diff}(b) \leq 0 \end{cases} \tag{1087}$$

Otherwise, $\hat{I}'_{nf}(b) = Env_{av}(b)$.

And, if $npulse(b) \neq 0$, compute the parameter of harmonic character $\hat{X}_{max}(b)/Env_{av}(b)$. The parameter of harmonic character $\hat{X}_{max}(b)/Env_{av}(b)$ is used to calculate the noise gain, then fill noise to the not decoded coefficients of bit allocation un-saturated sub-band.

5.3.4.1.4.1.5.2.2   Noise gain calculation

If the average number of bit allocation to a coefficient in a sub-band $R_{av}(b)$ is not less than 0.8, the bit allocation of the sub-band is defined as saturated; otherwise, the bit allocation of the sub-band is defined as un-saturated.

When the bit allocation of sub-band b is un-saturated and $\hat{I}'_{nf}(b) > 0$, based on the envelope and decoded coefficients the noise gain $\hat{I}_{nf}(b)$ is calculated as follows:

(a) First estimate the overall noise factors fac under different conditions:

(a1) When $npulse(b) \neq 0$ and the encoded bandwidth is SWB:

If the encoding mode is harmonic and $b \leq N_{cb}$:

$$fac = \frac{6*(1.5 - R_{av}(b))*Env_{av}(b)*\hat{I}'_{nf}(b)}{\hat{X}_{max}(b)^2} \tag{1088}$$

If the encoding mode is harmonic and $b > N_{cb}$:

$$fac = \frac{1.5*\hat{I}'_{nf}(b)}{\hat{X}_{max}(b)} \tag{1089}$$

If the encoding mode is normal and $b \leq N_{cb}$:

$$fac = \frac{5*(1.5 - R_{av}(b))*\hat{I}'_{nf}(b)}{\hat{X}_{max}(b)} \tag{1090}$$

If the encoding mode is normal and $b > N_{cb}$:

$$fac = \frac{4*\hat{I}'_{nf}(b)}{\hat{X}_{max}(b)} \tag{1091}$$

If the encoding mode is transient:

$$fac = 1.1 \tag{1092}$$

(a2) When $npulse(b) \neq 0$ and the encoded bandwidth is WB or NB:

Initialize fac as:

$$fac = \frac{20 * \min(1, (1.5 - R_{av}(b))) * Env_{av}(b) * \hat{I}'_{nf}(b)}{\hat{X}_{max}(b)^2}$$ (1093)

While $b > 1$ and $b < N_{bands} - 1$, fac is updated as follows:

$$if\,((C_{nz}(b+1) = 0) \,\&\&(Env_{av}(b) > 0.5 * Env_{av}(b+1)) \,\&\&(Env_{av}(b) < 2 * Env_{av}(b-1)))$$

$$fac = 1.5 * \frac{Env_{av}(b+1)}{Env_{av}(b)}$$

$$else\,if\,((C_{nz}(b-1) = 0) \,\&\&(\hat{X}_{max}(b) > 2 * Env_{av}(b)))$$ (1094)

$$fac = \frac{Env_{av}(b-1)}{Env_{av}(b)}$$

end

While $b \geq N_{bands} - N_{hf\_bands}$ and the encoded bandwidth is WB, fac is updated as follows. Where N_{hf_bands} is a number of sub-bands at the highest frequency which have a flag $f_{harmonic}(b)$ to indicate whether the sub-band is harmonic or not. If $f_{harmonic}(b)$ is equal to 1, the sub-band b have harmonic character; otherwise, the sub-band b do not have harmonic character.

$$if\,((\frac{ener_H}{bw_H} > \frac{ener_L}{bw_L}) \,\&\&(\hat{X}_{max}(b) < 2 * Env_{av}(b)))$$

$$fac = fac * (2 - \frac{\hat{I}'_{nf}(b)}{\hat{X}_{max}(b)})$$

end (1095)

$$if\,(f_{harmonic}(b) = 0)$$

$$fac = fac * (\min(1.25 * \frac{Env_{av}(b)}{\hat{I}'_{nf}(b) * fac}, 1.5))$$

end

where $ener_H$ and $ener_L$ denote the energy of high frequency bands and low frequency bands, respectively. bw_H and bw_H denote the numbers of correspond high frequency coefficients and correspond low frequency coefficients respectively.

(a3) When $npulse(b) = 0$, fac is determined as follows:

If the encoded bandwidth is SWB and the encoding mode is harmonic $fac = 0.8$; Otherwise, $fac = 1.1$.

(b) Based on the overall noise factor fac, the noise gain is calculated:

$$\hat{I}_{nf}(b) = fac * \hat{I}'_{nf}(b)$$ (1096)

When the sub-band b is a bit allocation saturated sub-band or $\hat{I}'_{nf}(b)$ is less than zero, the noise gain $\hat{I}_{nf}(b)$ is set to zero.

(c) In order to smooth the noise gain between the current frame and the previous frame, a sub-band boundary N_{pos} is defined. If the encoded bandwidth is SWB, $N_{pos} = N_{bands} - 1$. Otherwise, $N_{pos} = \max(N_{pulse}, N_{pulse}^{[-1]})$. Here N_{pulse} denotes the last sub-band with encoded pulses in current frame, $N_{pulse}^{[-1]}$ denotes the last sub-band with encoded pulses in previous frame.

For the sub-band b, if $0 < b \leq N_{pos}$, then the noise gain $\hat{I}_{nf}(b)$ is smoothed as follows:

When b is less than $N_{bands} - 1$:

$$if\,((Env_{av}(b) > 0.5 * Env_{av}^{[-1]}(b) \,\&\&\, Env_{av}(b) < 2 * Env_{av}^{[-1]}(b)) \,\|$$

$$((Env_{av}(b) + Env_{av}(b-1) + Env_{av}(b+1)) > 0.5 * (Env_{av}^{[-1]}(b) + Env_{av}^{[-1]}(b-1) + Env_{av}^{[-1]}(b+1)) \,\&\&$$

$$(Env_{av}(b) + Env_{av}(b-1) + Env_{av}(b+1)) < 2 * (Env_{av}^{[-1]}(b) + Env_{av}^{[-1]}(b-1) + Env_{av}^{[-1]}(b+1))))$$

$$if\,(\hat{I}_{nf}(b) > \hat{I}_{nf}^{[-1]}(b))$$

$$\hat{I}_{nf}(b) = 0.2 * \hat{I}_{nf}(b) + 0.8 * \hat{I}_{nf}^{[-1]}(b) \tag{1097}$$

$$else$$

$$\hat{I}_{nf}(b) = 0.6 * \hat{I}_{nf}(b) + 0.4 * \hat{I}_{nf}^{[-1]}(b)$$

$$end$$

$$end$$

When b is equal to $N_{bands} - 1$:

$$if\,((Env_{av}(b) > 0.5 * Env_{av}^{[-1]}(b) \,\&\&\, Env_{av}(b) < 2 * Env_{av}^{[-1]}(b)) \,\|$$

$$((Env_{av}(b) + Env_{av}(b-1)) > 0.5 * (Env_{av}^{[-1]}(b) + Env_{av}^{[-1]}(b-1)) \,\&\&$$

$$(Env_{av}(b) + Env_{av}(b-1)) < 2 * (Env_{av}^{[-1]}(b) + Env_{av}^{[-1]}(b-1))))$$

$$if\,(\hat{I}_{nf}(b) > \hat{I}_{nf}^{[-1]}(b))$$

$$\hat{I}_{nf}(b) = 0.2 * \hat{I}_{nf}(b) + 0.8 * \hat{I}_{nf}^{[-1]}(b) \tag{1098}$$

$$else$$

$$\hat{I}_{nf}(b) = 0.6 * \hat{I}_{nf}(b) + 0.4 * \hat{I}_{nf}^{[-1]}(b)$$

$$end$$

$$end$$

where $Env_{av}^{[-1]}(b)$ denotes the average envelope of the previous frame, $\hat{I}_{nf}^{[-1]}(b)$ denotes the noise gain of the previous frame.

5.3.4.1.4.1.5.2.3 Noise filling

First, if the encoded bandwidth is not SWB and $f_{harmonic}(b) = 0$, $b = [N_{hf\_bands}, N_{bands} - 1]$, modify the decoded coefficients:

$$\tilde{X}_M'(k) = 0.8 * \tilde{X}_M(k), \quad k = [k_{start}(N_{bands} - N_{hf\_bands}), k_{end}(N_{bands} - 1)] \tag{1099}$$

For sub-band b, if the conditions: bit allocation is unsaturated, the encoded bandwidth is not SWB, $b = N_{pulse}$, $b < N_{bands} - N_{hf\_bands}$ are all satisfied, the decoded coefficients is filled noise as follows:

Initialize a counter c_j to zero. Based on the smoothed noise gain, calculate a noise filling factor:

$$f_{smooth} = \max(\frac{\hat{I}_{nf}'(b)}{\hat{I}_{nf}(b)}, 1) \tag{1100}$$

If the decoded coefficient $\tilde{X}_M(k)$ is equal to zero,

$$\tilde{X}_M'(k) = \frac{f_{smooth} - (f_{smooth} - 1) * c_j}{k_{width}(b)} * \hat{I}_{nf}(b) * Rand \tag{1101}$$

where $Rand$ is generated by random noise, c_j is increased by 1.

Otherwise, the noise filling of the decoded coefficients is described as follows:

$$\tilde{X}'_M(k) = \tilde{X}_M(k) + \hat{I}_{nf}(b) * Rand \tag{1102}$$

Finally, update the memories as follows:

$$Env_{av}^{[-1]}(b) = Env_{av}(b) , \hat{I}_{nf}^{[-1]}(b) = \hat{I}_{nf}(b) , N_{pulse}^{[-1]} = N_{pulse} \tag{1103}$$

5.3.4.1.4.1.5.3   Pitch filtering spectrum coding (PFSC)

5.3.4.1.4.1.5.3.1   PFSC overview

Pitch filtering spectrum coding is applied only for the SWB and FB signals.

PFSC utilizes the TCQ-quantized and noise-filled low-frequency MDCT coefficients sequence to encode high frequency MDCT coefficients. Which band is encoded by PFSC is determined through the bit-allocation process described in the previous subclauses. In the PFSC, the four highest bands are the high-frequency bands, and the other bands lower than the high-frequency bands are the low-frequency bands. For example, in Table 108, the low-frequency bands are $b=0$ to 17, and the high-frequency bands are $b=18$ to 21, for the 13.2 kbps.

For each band, the most similar match with the selected similarity criteria is searched from the TCQ-quantized and envelope normalized low-frequency content. The most similar match is scaled with a scaling factor calculated using the quantized band energy to obtain the synthesized high frequency content.

5.3.4.1.4.1.5.3.2   Envelope normalization

Similarly to G.718 Annex B, quantized low-frequency content is normalized with its envelope. However, the TCQ-quantized low-frequency content is a sparse pulse sequence, and a normalization process is therefore used to flatten the low-frequency content.

The low-frequency content is normalized by dividing it by the maximum amplitude value in each sub-band. Here, the sub-band configuration is special and used only for this normalization. Each sub-band consists of 12 MDCT coefficients. By performing this process, each sub-band will have the same maximum amplitude value, and the low-frequency content can therefore be converted to an MDCT coefficient sequence whose spectral characteristic is flat and smoothed.

The envelope normalization (or smoothing) process is separately performed on the TCQ-quantized low-frequency content and the filled low-frequency noise content, respectively. And the amplitude of the normalized noise content is adaptively scaled according to the sparseness of the TCQ-quantized low-frequency content. The sparseness is calculated by dividing the number of non-zero spectrum in the TCQ-quantized low-frequency content by the bandwidth of the low-frequency content. A threshold is calculated using the sparseness and used for removing low amplitude TCQ-quantized low-frequency content and for scaling the maximum amplitude of the noise content. The normalized TCQ-quantized low-frequency content is further modified so that its non-zero content has larger amplitude than the maximum amplitude of the normalized noise content thus dynamic range of the normalized TCQ-quantized low-frequency content is modified for better matching with a targeted high-frequency spectrum. Finally, the scaled low-frequency noise content and the modified TCQ-quantized low-frequency content are added for generating envelope normalized spectrum (MDCT coefficients). If the generated spectrum becomes zero, such spectrum component is replaced with a randomly generated noise whose maximum amplitude is limited to the half of the maximum amplitude of the scaled noise component.

A block diagram of this envelope normalization is shown in Figure 98 of subclause 6.2.3.1.3.1.4.3.3. In Figure 98, all processing blocks except 'Lag info. decoding' are common to the both sides of encoder and decoder.

5.3.4.1.4.1.5.3.3   Band search

5.3.4.1.4.1.5.3.3.1   Selection of representative MDCT coefficients

Similarly to G.718 Annex B, a band search approach is used. The last four bands (i.e. $b=18$ to 21 in 13.2 kbps and $b=20$ to 23 in 16.4 kbps) are then subject to be encoded with PFSC. As shown in the table 108, the widths of the bands are 55, 68, 84, and 105 for 13.2 kbps, and 59, 74, 92, and 115 for 16.4 kbps.

To reduce computational load for calculating correlation, only limited number of input (target) MDCT coefficients are selected as representative MDCT coefficients and used for correlation calculation. The selection of the MDCT coefficients is performed by amplitude threshold process, i.e. an MDCT coefficient is selected if its absolute value is greater than a threshold. The threshold is determined using the average and standard deviation of the absolute values of the MDCT coefficients in a subjected high-frequency band.

$$thr_i = avg_i + \sigma_i \times \beta \tag{1104}$$

Here thr_i is the initial threshold for the i-th high-frequency band, avg_i is the average of the absolute values of the MDCT coefficients in the i-th high-frequency band, σ_i is the standard deviation of the absolute values of the MDCT coefficients in the i-th high-frequency band, and β is a factor for controlling the selected number of the MDCT coefficients. β is chosen so that a calculated threshold becomes higher than a threshold which is expected to be appropriate for selecting the limited number of the MDCT coefficients.

If the number of selected MDCT coefficients is less than a pre-determined number, the threshold is updated by the following equation, and an additional selection process is performed.

$$thr_i' = thr_i \times \left(-\frac{a-b}{Nmx} \times Ntmp_i + a \right) \tag{1105}$$

Here, a is the weakest attenuation factor and b is the strongest attenuation factor, and $1.0 \geq a > b > 0.0$. N_{mx} is the pre-determined number of the MDCT coefficients to be selected in the end, and $Ntmp_i$ is remaining number of MDCT coefficients to be selected. thr_i' is the updated threshold. By using this equation, the threshold is calculated according to the number of non-selected MDCT coefficients, i.e. the larger the number of non-selected MDCT coefficients is, the lower the threshold become. The above equation is equivalent to the following equation. $Ncnt_i$ is the number of already selected MDCT coefficients.

$$thr_i' = thr_i \times \left(\frac{a-b}{Nmx} \times Ncnt_i + b \right) \tag{1106}$$

This threshold update is performed twice using a different set of a and b unless the number of selected MDCT coefficients does not reach the pre-determined number.

The selected MDCT coefficients (target MDCT coefficients for the band search) are stored in a memory as their frequency positions and used for the band search process.

5.3.4.1.4.1.5.3.3.2 Matching process

Once the representative MDCT coefficients are selected, a matching process is performed by calculating the correlation between the representative MDCT coefficients and normalized low-frequency MDCT coefficients derived from the envelope normalized MDCT coefficients calculated in subclause 5.3.4.1.4.1.5.3.2. Since the correlation is calculated only using the selected MDCT coefficients, required computational complexity can be saved.

The task of the matching process is to find k' which maximizes $S(k')$.

$$S(k') = \frac{corr(k')^2}{Ene(k')}, \quad k'=0,...,Nlag^j - 1 \tag{1107}$$

where $corr(k')$, $Ene(k')$ and $Nlag^j$ denote the following.

$corr(k')$: correlation between representative MDCT coefficients and normalized low-frequency MDCT coefficients for the k'-th lag candidate,

$Ene(k')$: energy of normalized low-frequency MDCT coefficients for the k'-th lag candidate

$Nlag^j$: number of lag candidates for the j-th band.

$corr(k')$ and $Ene(k')$ are calculated by the following equations.

$$corr(k') = \sum_{k=0}^{k=Ncnt^j-1} X^j(Idx^j[k])\tilde{X}(k^j + lag^j[k'] + Idx^j[k]) \qquad (1108)$$

$$Ene(k') = \sum_{k=0}^{k=Ncnt^j-1} \tilde{X}(k^j + lag^j[k'] + Idx^j[k])^2 \qquad (1109)$$

$Ncnt^j$, $Idx^j[k]$, $lag^j[k']$ and k^j denote the followings.

$Ncnt^j$: selected number of representative MDCT coefficients in the *j*-th band,

$Idx^j[k]$: frequency position of the *k*-th representative MDCT coefficient in the *j*-th band,

$lag^j[k']$: the *k'*-th lag candidate for the *j*-th band,

k^j : starting frequency position of normalized low-frequency MDCT coefficients for the *j*-th band.

The lag candidates are defined as the frequency positions of non-zero normalized low-frequency spectrum. Therefore $lag^j[k']$ means the *k'*-th non-zero normalized low-frequency MDCT coefficients frequency position in the *j*-th band search range. The *j*-th sub-band search range is started at k^j, which is defined as offsets from the zero frequency point of the low-frequency spectrum. $k^j = \{0, 0, 64, 64\}$

By using this lag candidate representation, even when the bit budget for the lag information is small, actual lag search range can be wide, and it enables to guarantee to generate candidate spectra which have at least one non-zero spectrum.

Figure 76 shows a conceptual block diagram of the matching process. For the lag search, only the TCQ quantized component is used as the low-frequency MDCT coefficients while both of the TCQ-quantized and noise-filled low-frequency MDCT coefficients are used for generating high-frequency spectrum and calculating the scaling factors.

The best *k'*, which maximizes *S(k')*, is packed into the bit stream as an encoded parameter of the best lag candidate.
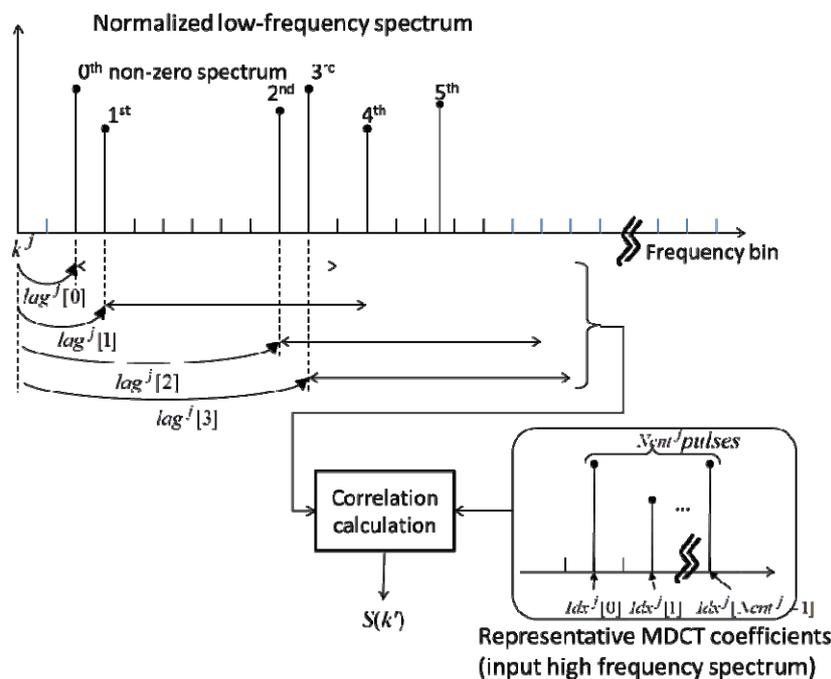
Figure 76: Conceptual block diagram of the matching process

5.3.4.1.4.1.5.3.3.3   Scaling and noise smoothing

Once the best match has been found through the matching process, scaling factors are calculated for the searched bands using the quantized band energies. Each scaling factor is calculated as the square root of the quotient of each quantized

band energy divided by its corresponding band enegy of a generated high-frequency spectrum. The band energy of the generated high-frequency spectrum is equal to $Ene(k')$ for the selected k' and is calculated according to equation (1109). The calculated scaling factors are attenuated by the scaling factor of 0.9.

Inter-frame smoothing process is applied on the generated high-frequency noise components. The generated MDCT coefficients whose amplitudes are below a threshold calculated using the sparseness of the TCQ-quantized low-frequency components are targeted for the smoothing process. When the energy of the targeted components in the previous frame is sufficiently less than current band energy, relatively strong smoothing is applied, while relatively weak smoothing is applied in other cases (i.e. current band energy is not sufficiently larger than the noise energy in the previous frame). The smoothing is performed on the energy of the targeted (noise) components.Then the smoothened energy of the noise components is divided by the energy of the noise components for calculating the final scaling factor in the smoothing process. The final scaling factor is applied to the noise components to obtain smoothened MDCT coefficients for the noise components.

Local decoding at the encoder side is necessary for switching between non-MDCT-based modes.

5.3.4.1.4.2 Transient mode

5.3.4.1.4.2.1 Energy envelope coding

The energy envelope coding is performed as described in subclause 5.3.4.1.3.

5.3.4.1.4.2.2 Bit allocation

The bit allocation to bands based on quantized band energies is performed as described in subclause 5.3.4.1.4.1.4.4. However, for transient frames the Limited band mode described in subclause 5.3.4.1.4.1.4.4.2 is not used. For SWB, no bits are allocated at the 6th and 7th bands.

5.3.4.1.4.2.3 Fine structure encoding

5.3.4.1.4.2.3.1 Trellis Coding Quantization (TCQ)

The spectral coefficient quantization is done as is described in subclause 5.3.4.1.4.1.5.1.

5.3.4.1.4.2.4 Noise-filling for 0-bit assigned bands

The noise filling is done as is described in subclause 5.3.4.1.4.1.5.2.

5.3.4.1.4.3 Harmonic Mode

5.3.4.1.4.3.1 Energy envelope coding

Overall framework of the Harmonic mode is the same as the Normal mode as shown in Figure 66, but the PFSC block is called as PFSC-based gap filling in the Harmonic mode.

The energy envelope coding is performed as described in subclause 5.3.4.1.3.

5.3.4.1.4.3.2 Bit allocation

5.3.4.1.4.3.2.1 Allocating bits for fine gain adjustment

Bit-allocation process is performed in the following manner when the signal is classified as harmonic. Firstly, two bits are reserved for transmitting the noise factor information (cf. Sec. 5.3.4.1.4.3.3.3.6) followed by four bits are allocated for performing gap filling using PFSC based approach. Then some bits are reserved for applying fine gain quantization to the band energies that are larger than the others.

The bit budget used for the spectrum quantization is obtained using

$$\psi = R - R_{FG} - e_{bits} - 6 \tag{1110}$$

where e_{bits} is the bits consumed for quantizing the band energies which is obtained from subclause 5.3.4.1.3, ψ is the available bit budget for spectrum quantization. R_{FG} is the number of bits reserved for fine gain quantization and described in table 119. For SWB and FB at the bit rates 13.2 and 16.4 kbps, $R_{FG} = 4$.

5.3.4.1.4.3.2.2   Adaptive bit allocation to bands

Based on the available bit budget ψ , bits are allocated to the bands using the adaptive bit allocation. The adaptive bit-allocation scheme uses the quantized band energies $\hat{I}_M(b)$, $\quad b=0,....,N_{bands}-1$ to allocate the available bits in a frame among the bands. In this scheme, bits are allocated by adaptively grouping the band energies into a number of groups; each group is allocated variable number of bits based on characteristics between the groups followed by bits allocation to bands within groups based on the available bit budget to the groups. Figure 77 illustrates the overview of the adaptive bit allocation. The detailed procedure is described in the following subclause.
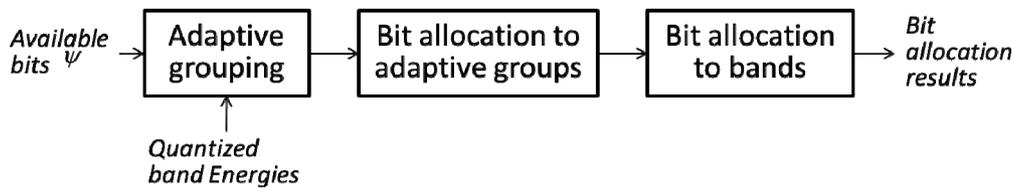
Figure 77: Overview of the Adaptive bit allocation scheme

5.3.4.1.4.3.2.2.1   Bits allocation to groups.

The adaptive bit-allocation scheme uses the quantized band energies, $\hat{I}_M(b)$, $\quad b=0,....,N_{bands}-1$ to allocate the available bits ψ in a frame among the bands.

First, the bit-allocation vector entries, i.e., bit allocation of each bands in bits per sample, are set to one, and followed by temporary storage of band energies in a buffer. This is done according to:

$$R(b) \leftarrow 1$$
$$I_{M_{temp}}(b) \leftarrow \hat{I}_M(b), \quad b=0,....,N_{bands}-1 \tag{1111}$$

In order to allocate bits to bands using band energies, firstly bands are adaptively grouped and each group contains variable number of bands, the grouping is to separate the dominant frequency band from non-dominant frequency band. In total, four groups are formed, while the widths of the first three groups are adaptively varied and the last group has a fixed width which has last four bands. The maximum variable width $G_{i_{Max}}$, $i=0,1,2$ of each adaptive group is thresholded and it is given in table 120

Table 120: Maximum widths of the groups

	13.2kbps	16.4kbps
$G_{0_{Max}}$	7	7
$G_{1_{Max}}$	9	10
$G_{2_{Max}}$	2	3

The grouping of bands is done by identifying the dominant frequency vectors (the frequency bands with large and local maximum energy factor values in the spectrum). Using the dominant frequency band as center, include the descending slope at both sides to one group; the group width is adaptively determined corresponding to the input signal characteristic. If the dominant frequency band is at the edge, only one side of the descending slope would be included in the group. In order to reduce the complexity sensitive ness of human ear is considered when searching

Once width of each group $G_W(g)$, $g=0,1,2,3$ is decided, bits are allocated to each group and it is done in the following steps.

a) Calculate the energy of each group according to

$$G_E(g) = \sum_{b=G_{Start}(g)}^{b=G_{end}(g)} I_{M_{temp}}(b), \quad g=0,1,2,3 \tag{1112}$$

where $G_{Start}(g), G_{end}(g) \quad g = 0,1,,2,3$ is the start and end of each group and width of each group is defined according to

$$G_W(g) = G_{end}(g) - G_{Start}(g), +1 \quad g = 0,1,,2,3. \tag{1113}$$

b) Allocate bits to the last group which has a fixed width based on the following steps.

   i. Calculate average number of bits that can be allocated for a group

$$G_{B_{avg}} = \frac{\psi}{4} \tag{1114}$$

   ii. Calculate mean for the group energies according to

$$G_{E_{avg}} = \frac{1}{4} \sum_{g=0}^{3} G_E(g) \tag{1115}$$

   iii. Calculate energy difference for last group according to

$$G_{E_{dif}} = \frac{G_E(3)}{4} - G_{E_{avg}} \tag{1116}$$

   iv. Calculate group energy ratio between the last group and average group energy of the remaining groups according to

$$G_{E_R} = \frac{G_E(3)}{G_{E_{avg,Adap}}},$$
$$G_{E_{avg,Adap}} = \frac{1}{3} \sum_{g=0}^{2} G_E(g) \tag{1117}$$

   v. Calculate the energy difference between maximum energy of a subvector in the last group and with average sub vector energy of the rest of the groups.

$$E_{diff_{avg}} = b_{max} - E_{avg1}, \tag{1118}$$

where b_{max} is the maximum energy in the last group according to equation (1119) and E_{avg1} is the average energy of the rest of the groups band and it is calculated according to .

$$b_{max} = \max_{g_3 = G_{start}(3),...G_{end}(3)} (I_{M_{temp}}(g_3)) \tag{1119}$$

$$E_{avg1} = \left. \sum_{b=0}^{N_{bands}-4-1} I_{M_{temp}}(b) \middle/ N_{bands} - 4 \right. \tag{1120}$$

   vi. Allocate bits to the last group according to

$$B_G(3) = \begin{cases} \min(\dfrac{G_{B_{avg}}}{4} + bw_1 G_{E_{diff}}, 10), & G_{E_R} > 0.8 \, and \, E_{diff_{avg}} > 8 \\ 0, & \text{otherwise} \end{cases} \tag{1121}$$

where bw_1 is the bit allocation weight, which takes value based on

$$bw_1 = \begin{cases} 0.4, & G_{E_{diff}} < 0 \\ 0.2, & \text{otherwise} \end{cases}$$

c) Once bits are allocated to last group, remaining bits are calculated according to

$$\psi \leftarrow \psi - B_G(3) \tag{1122}$$

d) Based on the remaining bits obtained from c), bits are allocated to the rest of the groups using following steps.

   i. Calculate the band energy difference between bands.

$$I_{M_{diff}}(b) = I_{M_{temp}}(b) - I_{M_{temp}}(b-1), \quad b = 1,.....,N_{bands} - 5 \tag{1123}$$

   ii. Identify the maximum band energy difference according to

$$b_{diff_{max}} = \max_{b=1,..,N_{bands}-5} (I_{M_{diff}}(b)) \tag{1124}$$

iii. Identify the group which has maximum band energy difference $b_{diff_{max}}$ and allocate more bits to the group .

If $b_{diff_{max}}$ belongs to group $g=1$, bits allocation to the groups will be according to

$$B_G(1) = Scale_1 \psi \frac{G_E(1)}{3G_{E_{avg,Adap}}},$$

$$B_G(2) = Scale_2 \psi \frac{G_E(2)}{3G_{E_{avg,Adap}}}, \qquad (1125)$$

$$B_G(0) = \psi - B_G(1) - B_G(2)$$

If $b_{diff_{max}}$ belongs to group $g=0$ or group $g=2$, bits allocation to the groups will be according to

$$B_G(0) = Scale_1 \psi \frac{G_E(0)}{3G_{E_{avg,Adap}}},$$

$$B_G(2) = Scale_2 \psi \frac{G_E(2)}{3G_{E_{avg,Adap}}}, \qquad (1126)$$

$$B_G(1) = \psi - B_G(0) - B_G(2)$$

where *Scale₁* and *Scale₂* values are based on table 121

Table 121: Scale factors for groups

		13.2 kbps	16.4 kbps
$b_{diff_{max}} \in Group\ 0\ or\ 1$	$Scale_1$	1.05	1.1
	$Scale_2$	0.97	0.92
$b_{diff_{max}} \in Group\ 2$	$Scale_1$	0.92	0.97
	$Scale_2$	1.0	1

Bits are allocated to individual bands in a group based on the available bit budget for the groups $B_G(g), g = 0,1,2,3$ and it is done according to subclause 5.3.4.1.4.3.2.2.2. If 0 bits are available to any of the groups $B_G(g), g = 0,1,2,3$ the corresponding bands in the group are allocated 0 bits.

5.3.4.1.4.3.2.2.2 Bits allocation to bands in a group

The bits allocation to individual bands in a group is applied using the steps given below

a) Calculate sum of band energies in a group

$$G_E(g) = \sum_{b=G_{Start}(g)}^{b=G_{end}(g)} I_{M_{temp}}(b), \quad g = 0,1,2,3 \qquad (1127)$$

b) Calculate Mean energy for the group

$$I_{M_{avg}}(g) = \frac{G_E(g)}{G_W(g)} \qquad (1128)$$

c) Set *scale₃* value according to

$$Scale_3 = \begin{cases} 0.35, & E_{diff\ count} > 0 \\ 0.6, & \text{otherwise} \end{cases} \qquad (1129)$$

where $Scale_3$ is a bitallocation weight constant which is used for controlling the amount of bits being allocated to the bands.

$E_{diff\ count}$ is the number of bands whose difference between the Mean energy for the group and band energy of the sub band exceeds 5 dB.

d) For each iteration in a group,

i. Calculate Mean energy for the group

$$I_{M_{avg}}(g) = \frac{G_E(g)}{G_W(g)} \qquad (1130)$$

ii. Allocate bits to the individual subvectros in a group according to .

$$R(b) = B_G(g) \frac{I_{M_{temp}}(b)}{G_E} + scale_3(I_{M_{temp}}(b) - I_{M_{avg}}(g)), \quad b = G_{Start}(g), G_{end}(g) \text{ and } g = 0,1,2,3 \quad (1131)$$

iii. Once the bits are allocated to indvidual bands in a group g, identify the band which has least energy in the group and verify whether bits allocated to the identfied band has under allocation using a threshold, if the band has under allocation. update the sum of energies using

$$R_{\min} = \min_{b = G_{Start}(g), G_{end}(g)} (R(b))$$

$$if \ R_{\min} < thr$$

$$R(G_{\min}) = 0,$$

$$G_E(g) \leftarrow G_E(g) - I_{M_{temp}}(G_{\min}),$$

$$I_{M_{temp}}(G_{\min}) = 0,$$

$$G_W(g) \leftarrow G_W(g) - 1$$

$$end$$

where G_{min} is the index corresponding to minimum energy R_{min} of a band in the group g

iv. Repeat Step g, until there is no under allocation.

The process described in subclause 5.3.4.1.4.3.2.2.2, is used for allocating bits to individual bands for groups $g=0,1,2,3$.

5.3.4.1.4.3.3 Fine structure encoding

5.3.4.1.4.3.3.1 Trellis Coding Quantization (TCQ)

The spectral coefficient quantization is done as is described in subclause 5.3.4.1.4.1.5.1.

5.3.4.1.4.3.3.2 Noise-filling for 0-bit assigned bands:

This subclause gives a technical overview of the noise filling processing. Noise filling consists of two algorithms. The first algorithm described in this subclause 5.3.4.1.4.1.5.2 fills the gaps in the quantized spectrum where coefficients have been quantized to zero when the bit allocation clause allocates non zero bits to the bands and fills the un quantized bands up to the transition frequency f_t where the transition frequency f_t is estimated based on the received bit allocation. The second algorithm is described as follows

5.3.4.1.4.3.3.3 PFSC-based gap filling

5.3.4.1.4.3.3.3.1 Overview

This subclause is only applied to SWB and FB input signals. The spectral coefficients which belong to bands which are assigned zero bits from the bit-allocation clause are not quantized. This means that not all transform coefficients are transmitted to the decoder. From the noise filled quantized spectrum, the gaps in the high frequency region $b = N_{bands} - 4,.., N_{bands} - 1$ which has zero bit allocation are identified and a predicted spectrum information is generated and the missing frequency bands (gaps) are filled with the predicted spectrum information.

In order to perform gap filling, the most similar match with the selected similarity criteria is searched from the coded and envelope normalized content and encodes the lag index $LagIndex^i$ parameter followed by encoding of the noise factor. The lag index $LagIndex^i$ and noise factor parameters are used in the decoder for generating the predicted spectrum, which will be used for filling the gaps. Figure 78 illustrates the overview of PFSC based gap filling for Harmonic mode.
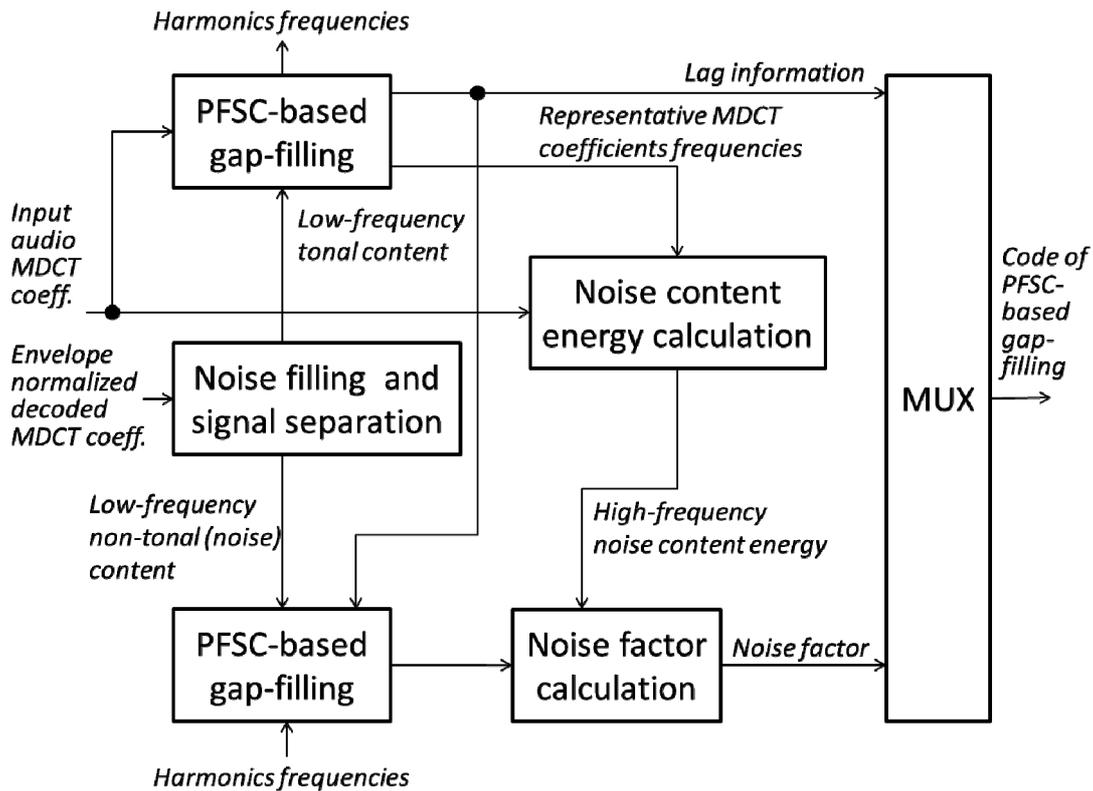
Figure 78: Overview of the Harmonic mode PFSC based gap filling

5.3.4.1.4.3.3.3.2 Envelope Normalization

Low frequency content is extracted from the quantized noise filled spectrum, which is stored temporarily in a buffer and supplies the buffer information to noise separator, which separates noise $N_M(k)$ and quantized content $\hat{X}_M(k)$. The envelope of the quantized and noise contents is normalized before further processing. The normalization is performed in logarithmic domain and it is according to

Peak to average information is calculated in log domain for the quantized content $\hat{X}_M(k)$.

$$P_{avg} = 10\log(\frac{\hat{X}_{\max}}{\hat{X}_{avg}})$$ (1132)

where $\hat{X}_{\max}$ is the maximum absolute amplitude of the quantized spectrum and $\hat{X}_{avg}$ is the mean of the absolute amplitude of the quantized spectrum.

The quantized content $\hat{X}_M(k)$ is divided into N_{sv} sub-vectors and the total number of the sub-vectors is based on the table 122. Each sub-vector $SV_l(i)$ consists of eight spectral coefficients: This sub-vector division is only available for envelope normalization.

$$SV_l(i) = \hat{X}_M(8l+i), \quad i = 0,...,7$$ (1133)

The arithmetic means of the 1st and 2nd half of each sub-band $M_a^1(l)$ and $M_a^2(l)$ are calculated as follows:

$$M_a^1(l) = \frac{\sum_{i=0}^{3}\left|SV_l(i)\right|}{4} \qquad l = 0,...,N_{sv}-1$$ (1134)

$$M_a^2(k) = \frac{\sum_{i=4}^{7} |SV_l(i)|}{4} \qquad l = 0,..., N_{sv} - 1 \tag{1135}$$

Next, the geometric mean of $M_a^1(l)$ and $M_a^2(l)$ in logarithmic domain of each sub-band, $L(l)$ is obtained as:

$$L(l) = 10 \cdot \log_{10} \left(\left| M_a^1(l) \cdot M_a^2(l) + 1e - 15 \right| \right) \quad l = 0,..., N_{sv} - 1 \tag{1136}$$

Here, the square root calculation in linear domain is performed by multiplying 0.5.

The $L(l)$, $l = 0, ..., N_{sv}$, is smoothed using moving average method with a span of seven samples. The envelope smoothed values $\tilde{L}(l)$ are obtained as:

$$\tilde{L}(l) = \begin{cases} L(l), & l = 0 \\ \dfrac{L(l-1) + L(l) + L(l+1)}{3}, & l = 1 \\ \dfrac{L(l-2) + ... + L(l+2)}{5}, & l = 2 \\ \dfrac{L(l-3) + ... + L(l+3)}{7}, & l = 3,..., N_{sv} - 4 \\ \dfrac{L(l-2) + ... + L(l+2)}{5}, & l = N_{sv} - 3 \\ \dfrac{L(l-1) + L(l) + L(l+1)}{3}, & l = N_{sv} - 2 \\ L(l), & l = N_{sv} - 1 \end{cases} \tag{1137}$$

The smoothed envelope $\tilde{L}(k)$ is transformed from logarithmic domain to linear domain as follows:

$$\tilde{L}'(l) = 10^{(-1.0) \cdot 0.05 \cdot (\tilde{L}(l))} \quad l = 0,..., N_{sv} - 1 \tag{1138}$$

The quantized contents are normalized by multiplying the smoothed envelope $\tilde{L}'(k)$ in order to obtain the normalized contents:

$$\tilde{X}_M(k) = \hat{X}_M(k) \cdot \tilde{L}'(l) \quad k = 8l,...,8l + 7 \quad l = 0,..., N_{sv} - 1 \tag{1139}$$

Table 122: Number of sub-vectors for energy normalization

Bitrate in kbps	N_{sv}
13.2	32
16.4	38

From the normalized content, thresholds are calculated according to

$$\begin{aligned} clip_{\max} &= \tilde{X}_{avg} clip_{coef} \\ clip_{\min} &= 0.25 \tilde{X}_{avg} \end{aligned} \tag{1140}$$

where $clip_{coef}$ value is according to.

$$clip_{coef} = \begin{cases} 2.5, & P_{avg} - 16 < 0 \\ P_{avg} - 13.5, & otherwise \end{cases} \tag{1141}$$

$\tilde{X}_{avg}$ is calculated according to

$$\tilde{X}_{avg} = \frac{1}{8N_{sv}} \sum_{k=0}^{8N_{sv}-1} \left| \tilde{X}_M(k) \right| \tag{1142}$$

Absolute values of the normalized coefficients $\left| \tilde{X}_M \right|$ are compared to the thresholds $clip_{max}$ and $clip_{min}$, which are adaptively calculated by equation (1140). The normalized coefficients with absolute amplitudes above $clip_{max}$ are clipped to $clip_{max}$ to avoid excessive spectral peaks, and the normalized coefficients with absolute amplitudes below $clip_{min}$ are suppressed to 0 to enhance the harmonic structure, according to

$$
\begin{aligned}
&for\, k = 0\, to\, 8N_{sv} - 1 \\
&\quad if\, \left| \tilde{X}_M(k) \right| > clip_{max} \\
&\quad\quad if\, \tilde{X}_M(k) < 0 \\
&\quad\quad\quad \tilde{X}_M(k) = -clip_{max} \\
&\quad\quad else \\
&\quad\quad\quad \tilde{X}_M(k) = clip_{max} \\
&\quad\quad end \\
&\quad\quad if\, \left| \tilde{X}_M(k) \right| < clip_{min} \\
&\quad\quad\quad \tilde{X}_M(k) = 0 \\
&\quad\quad end \\
&\quad end \\
&end
\end{aligned}
$$

Above process i.e., normalization and clipping of the normalized contents is also applied for normalizing the low frequency noise contents and the obtained normalized noise contents is represented as $\tilde{N}_M(k)$ the normalized noise contents are further adjusted according to

$$\tilde{N}_M(k) \leftarrow \tilde{N}_M(k)N_{ratio} \quad k = 0,...,8N_{sv} - 1 \tag{1143}$$

where N_{ratio} is the sparseness ratio and it is calculated according to

$$N_{ratio} = 1 - \frac{pul_{count}}{8N_{sv} - 1} \tag{1144}$$

pul_{count} is the number of non-zero spectrum in the TCQ-quantized low-frequency content.

The normalized low frequency quantized noise filled spectrum is obtained according to

$$\tilde{X}'_M(k) = \tilde{X}_M(k) + \tilde{N}_M(k) \tag{1145}$$

5.3.4.1.4.3.3.3.3   Band Search

The high-frequency region of $X_M(k)$ with width hf_w is divided into four bands as follows

$$M_i(k) = X_M(k + hf_{start} + d^i) \quad k = 0,..., SB_{width}^i - 1 \tag{1146}$$

where the starting position of high frequency is represented as hf_{start}, high frequency width is represented as hf_{width} and its corresponding values are according to table 123. The values of band widths SB_{width}^i corresponding to index i and offsets d^i are according to tables 124 and 125.

Table 123: High frequency width and starting position

Bitrate in kbps	hf_{start}	hf_{width}
13.2	256	312
16.4	300	340

Table 124: High frequency band width and offsets for 13.2 kbps

Index i	SB^i_{width}	d^i
0	56	0
1	100	56
2	100	156
3	56	256

Table 125: High frequency band width and offsets for 16.4 kbps

Index i	SB^i_{width}	d^i
0	60	0
1	110	60
2	110	170
3	60	280

For sub-band $M_i(k)$, $i=0,1$ there is a search band $\widetilde{X}^i_M(k) = \widetilde{X}_M(k), \quad k = k^i,...,k^i_{end}$ which defines from where the best match is searched.

The start position k^i and the width w^i of each sub-band are set as follows:

$$k^i = \begin{cases} 120 - \dfrac{lags^i}{2}, & i = 0 \\ 210 + \dfrac{lags^i}{2}, & i = 1 \end{cases} \tag{1147}$$

$$k^i_{end} = \begin{cases} k^i + w^i, & i = 0 \\ k^i - w^i, & i = 1 \end{cases}$$

$$and \ w^i = \begin{cases} SB^i_{width} + \dfrac{lags^i}{2}, & i = 0 \\ SB^i_{width} + \dfrac{lags^i}{2}, & i = 1 \end{cases} \tag{1148}$$

where $lags^i, i = 0,1$ is the number of search positions and it is calculated according to

$$lags^i = 2^{bits^i_{hf}} \tag{1149}$$

where the number of bits $bits^i_{hf}, i = 0,1$ is reserved for the band search for index *i*.

The best match is searched for every band *i* ,in order to reduce computational load for calculating correlation, only limited number of input spectral coefficients are selected and used for the calculation of correlation. The selection of the MDCT coefficients is described in subclause 5.3.4.1.4.3.3.3.1.

Once the input spectral coefficients are selected, searching process is performed by calculating the correlation between the input spectral coefficients and normalized low-frequency coefficients derived from the envelope normalized coefficients calculated in subclause 5.3.4.1.4.3.3.3.2. Since the correlation is calculated only using the selected coefficients, required computational complexity can be reduced.

The best match is searched for every band *i* as follows. The correlation for index *k'* is computed as

$$corr(k') = \sum_{j=0}^{j=N_{cnt}^i-1} M_i(Idx^i[j]) \tilde{X}_M^i(k'+Idx^i[j]) \quad k'=N_1,..N_2 \tag{1150}$$

where $corr(k')$, N_{cnt}^i, $Idx^i[k]$ and N_1, N_2 denote the following

$corr(k')$ is the correlation between input spectral coefficients $M_i(k)$ and search band $\tilde{X}_M^i(k)$ for the *k'-th* lag candidate,

N_{cnt}^i selected number of representative spectral coefficients in the *i*-th band. $Idx^i[k]$: frequency position of the *k*-th representative spectral coefficient in the *i*-th sub-band,

N_1, N_2 : is the start and end position of search lag and its values are calculated according to

$$N_1 = \max(0, BestIdx_{past}^i - \frac{lags^i}{2})$$

$$N_2 = \begin{cases} lags^i - 1, & BestIdx_{past}^i < 0 \\ \min(lags^i - 1, BestIdx_{past}^i + \frac{lags^i}{2}) & otherwise \end{cases} \tag{1151}$$

where $BestIdx_{past}^i$ is the previous frame best match position for band *i*

similarly, the energy for index *k'* is obtained as

$$Ene(k') = \sum_{j=0}^{j=N_{cnt}^i-1} \tilde{X}_M^i(k'+Idx^i[j])^2 \quad k'=N_1,..N_2 \tag{1152}$$

where $Ene(k')$ is the energy of the search band $\tilde{X}_M^i(k)$ for the k'-th lag candidate

The actual similarity measure used is of the form

$$S'(k') = \left| \frac{corr(k')}{\sqrt{Ene(k')}} \right| \tag{1153}$$

The task is to find *k'* which maximizes *S'(k')*. The complexity of the implementation can be reduced by squaring *S'(k')*, which removes the absolute value signs as well as square root from the denominator as equation (1107). The best match is now searched efficiently for band *i* using

$$BestIdx^i = 0$$
$$lagCorr = 0$$
$$lagEnergy = 1e30$$
for $k' = N1$ to $N2 - 1$
  if $Ene(k') > 0$
   if $lagCorr^2 Ene(k') < corr(k')^2 lagEnergy$
    $BestIdx^i = k'$
    $lagCorr = corr(k')$
    $lagEnergy = Ene(k')$
   end
  end
 end

where $BestIdx^i$ is the best match position for band i and if there is no spectral coefficients in the search band then $BestIdx^i$ value takes according to

$$BestIdx^i = \begin{cases} 0 & lagCorr^2 = 0 \ AND \ BestIdx^i_{past} < 0 \\ BestIdx^i_{past} & lagCorr^2 = 0 \end{cases} \qquad (1154)$$

The best match index $BestIdx^j$ is packed into the bit stream as the parameter $LagIndex^i$. Once the best match index $BestIdx^i$ is obtained, past frame best match index $BestIdx^i_{past}$ is updated with current frame best match index according to

$$BestIdx^i_{past} = BestIdx^i \qquad (1155)$$

5.3.4.1.4.3.3.3.4 Structure analysis for Harmonics

This subclause gives a technical overview of the structure analysis for harmonics which is applied for SWB and FB Harmonic signals. The purpose of this subclause is to estimate the harmonics from the quantized spectrum and the estimated harmonic is used for generating the new tonal spectrum for the HF region at the decoder, where the HF region is calculated using the frequency transition f_t which is estimated based on the received bit allocation.

The first step of the structure analysis is to extract a portion from the quantized spectrum. A portion from 2 kHz to 6.4 kHz for 13.2 kbps and from 2- 7.5 kHz for 16.4 kbps is used for structure analysis.

From the extracted portion, structure for harmonics is analysed. The detail procedure of this subclause comprises the following steps:

1) Grouping of spectral coefficients into a number of blocks of equal length; in total 16 and 19 blocks (N_{block}) are formed for 13.2 and 16.4. Kbps with a length of 16 coefficients per block.

2) Extract the spectral peak magnitude $X_{max}(j)$ and their corresponding positions $pos_{max}(j)$ within each block; as shown

$$X_{max}(j) = \max_{k=16j,\cdots,16j+15} |\hat{X}_M(k)| \qquad j = 5,\cdots,N_{block}-1$$
$$pos_{max}(j) = k \qquad k \in \max_{k=16j,\cdots,16j+15} |\hat{X}_M(k)| \qquad (1156)$$

3) From the extracted spectral peaks identify the spectral peaks whose spacing lie closely and discard the spectral peak from analysis which is not perceptually important.

4) Split the extracted portion into regions with cut-off at 140, 200, 16N_{block}-1 spectral coefficients and count the peaks in each region according to

$$c_{peak\_l} = c_{peak\_l} + 1, if \ pos_{max}(j) < 140$$
$$c_{peak\_m} = c_{peak\_m} + 1, if \ 140 <= pos_{max}(j) < 200 \qquad (1157)$$
$$c_{peak\_h} = c_{peak\_h} + 1, if \ 200 <= pos_{max}(j) < (16N_{block}-1)$$

Before the process counter $c_{peak\_l}, c_{peak\_m}, c_{peak\_h}$ is set to zero.

5) Calculate the spacing between the identified peak positions and identify the minimum and maximum spacing between spectral peaks.

$$Spacing_{peak}(j) = Pos_{max}(j+1) - Pos_{max}(j) \qquad j \in [0, N_{block}-1]$$
$$Spacing_{Min} = \min_{j=5,\dots N_{block}-1} (Spacing_{peak}(j)) \qquad (1158)$$
$$Spacing_{Max} = \max_{j=5,\dots N_{block}-1} (Spacing_{peak}(j))$$

6) Calculate sum of spacing according to

$$for \; j = 5 \, to \, N_{block} - 1$$

$$if \; Spacing_{peak}(j) > 0$$

$$if \; Spacing_{peak}(j) < (Spacing_{min} + 10)$$

$$Spacing_{peak\_s1} = Spacing_{peak\_sum1} + Spacing_{peak}(j),$$

$$c_{spacing\_peak\_s1} = c_{spacing\_peak\_s1} + 1$$

$$else \, if \; Spacing_{peak}(j) < (Spacing_{min} + 20)$$

$$Spacing_{peak\_s2} = Spacing_{peak\_sum2} + Spacing_{peak}(j),$$

$$c_{spacing\_peak\_s2} = c_{spacing\_peak\_s2} + 1$$

$$else$$

$$Spacing_{peak\_s3} = Spacing_{peak\_sum3} + Spacing_{peak}(j),$$

$$c_{spacing\_peak\_s3} = c_{spacing\_peak\_s3} + 1$$

$$end$$

$$end$$

$$end$$

where $Spacing_{peak\_s1}, Spacing_{peak\_s2}, Spacing_{peak\_s3}$ is the sum of spacing and the counter $c_{spacing\_peak\_s1}, c_{spacing\_peak\_s2}, c_{spacing\_peak\_s3}$ increments when the respective conditions are satisfied.

7) Estimate the harmonic frequency based on the spacing between the identified peak positions according to the following.

$$if \; c_{spacing\_peak\_s1} \, != 0$$

$$Est_{freq\_1} = \frac{Spacing_{peak\_s1}}{c_{spacing\_peak\_s1}}$$

$$end$$

$$if \; c_{spacing\_peak\_s2} > 1$$

$$Est_{freq\_2} = \frac{Spacing_{peak\_s2}}{c_{spacing\_peak\_s2}}$$

$$else \, if \; ((c_{spacing\_peak\_s2} < 2 \, AND \, (c_{spacing\_peak\_s3} \, != 0 \, OR \, c_{spacing\_peak\_s1} > 1)) \, OR$$

$$((c_{peak\_l} \, != 0 \, OR \, c_{peak\_m}) \, AND \, c_{spacing\_peak\_s2} = 0 \, AND \, c_{spacing\_peak\_s3} = 0))$$

$$Est_{freq\_2} = Est_{freq\_1}$$

$$else$$

$$Est_{freq\_2} = 2 Est_{freq\_1}$$

$$end$$

where $Est_{freq\_1}, Est_{freq\_2}$ are the estimated harmonic frequencies, which can be used for generating the missing bands.

In order to improve the stability of present frame, previous frame estimated frequency Est_{freq_2} information is used.

if $(c_{peak\_l} < 2 \, AND \, c_{peak\_m} < 2 \, AND \, c_{peak\_m} < 2) \, OR \, Spacing_{min} \, >= 80$ and the calculated estimated frequencies $Est_{freq\_1}, Est_{freq\_2}$ has a zero value it indicates that there is no harmonics extracted from _0_ to N_{block} _-1_.usually this happens when decoded spectrum is sparse. For this case, Est_{freq_2} is set to previous frame estimated frequencies if it not equal to zero or it is set to a default value of 80. In the encoder, estimated harmonic Est_{freq_2} is used for generating the noise spectrum for the HF region. As the estimated harmonic Est_{freq_2} is determined using the quantized spectrum, there is no need to transmit the Est_{freq_2} parameter to the decoder. Under rate switching conditions this structure analysis for harmonics is also used for bitrates above 16.4 kbps for extracting the harmonics.

5.3.4.1.4.3.3.3.5 Noise filling for the predicted spectrum

First high-frequency region $\tilde{H}_M(k)$ with width hf_w is divided into four bands $\tilde{H}_M^i(k)$, $i = 0,...,3$ with same band configuration described in subclause 5.3.4.1.4.3.3.3

From the envelope normalized noise spectrum $\tilde{N}_M(k)$, a desired portion of noise spectrum is extracted. The start position k^i and the end k_{end}^i of each desired portion in the normalized noise spectrum $\tilde{N}_M(k)$, $k = k^i,...k_{end}^i$ are set as follows:

$$k^i = \begin{cases} 120 - \dfrac{lags^i}{2} + BestIndex^i, & i = 0 \\ 210 + \dfrac{lags^i}{2} - BestIndex^i, & i = 1 \end{cases} \tag{1159}$$

$$k_{end}^i = \begin{cases} k^i + w^i, & i = 0 \\ k^i - w^i, & i = 1 \end{cases}$$

$$and \ w^i = \begin{cases} SB_{width}^i, & i = 0 \\ SB_{width}^i, & i = 1 \end{cases} \tag{1160}$$

here $lags^i$ is the number of search positions and it is according to equation (1149), lag index value $BestIdx^i$ is obtained from subclause 5.3.4.1.4.3.3.3.

Estimate the position of harmonics for band $i = 0, 1$ in the predicted spectrum, according to

$$
\begin{aligned}
&j = 0 \\
&pk_{pos}(j) = hf_{start} \\
&while \ pos < (hf_{start} + SB_{width}^0 + SB_{width}^1) \\
&\quad j = j + 1 \\
&\quad pos = hf_{start} + jEst_{freq\_2} \\
&\quad pk_{pos}(j) = pos \\
&end \\
&res_{tot} = j
\end{aligned}
\tag{1161}
$$

where res_{tot} represents pulse resolution for predicted spectrum and Est_{freq_2} is obtained from subclause 5.3.4.1.4.3.3.3.4.

Fill the bands $\tilde{H}_M^i(k)$, $i=0,1$ using extracted noise from the normalized spectrum $\tilde{N}_M(k)$, $k = k^i,...k^i + w^i$ except in the positions obtained in equation (1161) and fill the remaining bands $\tilde{H}_M^i(k)$, $i=2,3$ by copying the information obtained from the lower bands $i=0,1$ reversely.

5.3.4.1.4.3.3.3.6 Noise factor

On the decoder side a predicted spectrum $\tilde{H}_M(k)$ is generated for the high frequency region by using the envelope normalized noise-filled quantized spectrum $\tilde{X}_M(k), \tilde{N}_M(k)$, which is obtained from subclause 5.3.4.1.4.3.3.3.5. To control the amount of noise inserted in the predicted spectrum at the decoder, a noise factor is computed on the encoder side and quantized using a 2-bit scalar quantizer. The procedure for noise factor calculation is described in this subclause.

From the input spectral coefficients, extract the high-frequency region $H_M(k)$ according to

$$H_M(k) = X_M(k) \quad k = hf_{start},...,hf_{width} - 1 \tag{1162}$$

where the starting position of high frequency is represented as hf_{start} and high frequency width is represented as hf_{width}. The corresponding values of hf_{start}, hf_{width} is same as in table 123.

Select the tonal components in the $H_M(k)$ according to pulse resolution $Idx[j]$, $j = 0, ..., N_{cnt} - 1$ obtained in subclause 5.3.4.1.4.3.3.3.3. The selection of the spectral coefficients is based on subclause 5.3.4.1.4.3.3.3.1.

Calculate energy of the selected coefficients according to

$$Ene_S = \sum_{j=0}^{j=N_{cnt}-1} H_M(Idx[j])^2$$

Calculate noise factor using $\tilde{H}_M(k)$ and $H_M(k)$ according to

$$N_{fac} = \log_{10}(\frac{Ene_H - Ene_S}{Ene_{\tilde{H}}}) \tag{1163}$$

where $Ene_H, Ene_{\tilde{H}}$ is the energies obtained using $H_M(k)$, $\tilde{H}_M(k)$ according to

$$Ene_H = \sum_{k=0}^{k=hf_{width}-1} H_M(k)^2$$

$$Ene_{\tilde{H}} = \sum_{k=0}^{k=hf_{width}-1} \tilde{H}_M(k)^2 \tag{1164}$$

The calculated noise factor N_{fac} is encoded using a two bit scalar quantizer.

5.3.4.2 High-rate HQ coder

Table 126: High rate HQ supported modes

Bitrate [kbps]	Bandwidth	Supported modes
32	WB	Normal, Transient
24.4, 32	SWB	Transient, Harmonic, HVQ, Generic
24.4, 32	FB	Transient, Generic
64	SWB, FB	Normal, Transient

The high level structure of the high-rate HQ coder is in figure 79. The different operating modes of the high-rate HQ coder are outlined in table 126, the high-rate HQ is used at WB, SWB, and FB, at bit-rates 24.4, 32, and 64 kb/s. The coding is done in the MDCT domain. There are 5 different modes: Transient mode handles transient signals by using shorter transforms, Harmonic mode handles harmonic signals and HVQ takes care of strongly harmonic signals. Normal and Generic mode are used for all other signals.
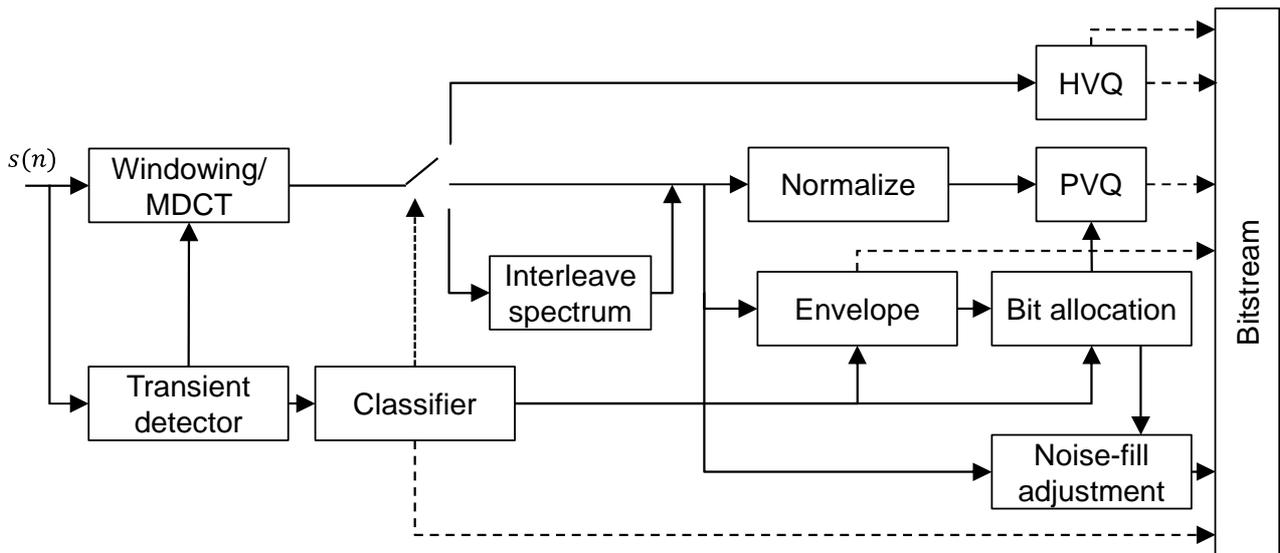
Figure 79: High level structure of the high-rate HQ encoder

The whole frequency spectrum is divided into bands, there are 3 different band structures used: Default, harmonic, and wideband, see tables 127, 128, and 129. Harmonic band structure is used for HVQ and Harmonic mode, wideband band structure is used for WB. Otherwise the default band structure is used.

Table 127: Normal band structure used in high rate HQ

b	0	1	2	3	4	5	6	7	8	9	10	11	12
$k_{start}(b)$	0	8	16	24	32	40	48	56	64	72	80	88	96
$k_{end}(b)$	7	15	23	31	39	47	55	63	71	79	87	95	103
b	13	14	15	16	17	18	19	20	21	22	23	24	25
$k_{start}(b)$	104	112	120	128	144	160	176	192	208	224	240	256	280
$k_{end}(b)$	111	119	127	143	159	175	191	207	223	239	255	279	303
b	26	27	28	29	30	31	32	33	34	35	36	37	38
$k_{start}(b)$	304	328	352	376	400	424	448	472	496	520	554	576	608
$k_{end}(b)$	327	351	375	399	423	447	471	495	519	553	575	607	639
b	39	40	41	42	43								
$k_{start}(b)$	640	672	704	736	768								
$k_{end}(b)$	671	703	735	767	799								

Table 128: Wideband band structure used in high rate HQ

b	0	1	2	3	4	5	6	7	8	9	10	11	12
$k_{start}(b)$	0	8	16	24	32	40	48	56	64	72	80	88	96
$k_{end}(b)$	7	15	23	31	39	47	55	63	71	79	87	95	103
b	13	14	15	16	17	18	19	20	21	22	23	24	25
$k_{start}(b)$	104	112	120	128	144	160	176	192	208	224	240	256	288
$k_{end}(b)$	111	119	127	143	159	175	191	207	223	239	255	287	319

Table 129: Harmonic band structure used in high rate HQ

b	0	1	2	3	4	5	6	7	8	9	10	11	12
$k_{start}(b)$	0	8	16	24	32	40	48	56	64	72	80	88	96
$k_{end}(b)$	7	15	23	31	39	47	55	63	71	79	87	95	103
b	13	14	15	16	17	18	19	20	21	22	23	24	25
$k_{start}(b)$	104	112	120	128	144	160	176	192	208	224	256	288	320
$k_{end}(b)$	111	119	127	143	159	175	191	207	223	255	287	319	367
b	26	27	28	29	30	31	32	33	34	35	36	37	38
$k_{start}(b)$	368	416	464	512	576								
$k_{end}(b)$	415	463	511	575	639								

The number of bands used in WB is 26, and for FB it is 44. For SWB 39 bands are used, except if Harmonic or HVQ is used. For Harmonic mode 31 bands are used. The HVQ will use bands 21-30 for 24.4 kb/s, or 24-30 for 32 kb/s.

Based on the different modes, the low frequency band signal is encoded with the different bandwidths. In addition, the envelopes of the higher frequency band are encoded differently according to the different modes.

Then, the mode information, the indices of the low frequency band signal and the envelopes of the higher frequency band signal are written to the bitstream.

5.3.4.2.1 Normal Mode

5.3.4.2.1.1 Envelope calculation and quantization

The norm or spectrum energy $E_M(b)$ of a band b is defined as the root-mean-square (*rms*) value of the band and computed as follows:

$$E_M(b) = \sqrt{\frac{1}{L_M(b)} \sum_{k=k_{start}(b)}^{k_{end}(b)} X_M(k)^2}, \quad b = 0,\ldots,N_{bands}-1 \tag{1165}$$

where $L_M(b) = k_{end}(b) - k_{start(b)} + 1$ is the length of the band b.

Table 130: Envelope quantization table

Index	Code	Index	Code	Index	Code	Index	Code
0	$2^{17.0}$	10	$2^{12.0}$	20	$2^{7.0}$	30	$2^{2.0}$
1	$2^{16.5}$	11	$2^{11.5}$	21	$2^{6.5}$	31	$2^{1.5}$
2	$2^{16.0}$	12	$2^{11.0}$	22	$2^{6.0}$	32	$2^{1.0}$
3	$2^{15.5}$	13	$2^{10.5}$	23	$2^{5.5}$	33	$2^{0.5}$
4	2^{15}	14	$2^{10.0}$	24	$2^{5.0}$	34	$2^{0.0}$
5	$2^{14.5}$	15	$2^{9.5}$	25	$2^{4.5}$	35	$2^{-0.5}$
6	$2^{14.0}$	16	$2^{9.0}$	26	$2^{4.0}$	36	$2^{-1.0}$
7	$2^{13.5}$	17	$2^{8.5}$	27	$2^{3.5}$	37	$2^{-1.5}$
8	$2^{13.0}$	18	$2^{8.0}$	28	$2^{3.0}$	38	$2^{-2.0}$
9	$2^{12.5}$	19	$2^{7.5}$	29	$2^{2.5}$	39	$2^{-2.5}$

In each frame, the norms are scalar quantized with a uniform logarithmic scalar quantizer with 40 steps of 3 dB.

The index of the quantized norm, $I_M(b)$, can easily be obtained as:

$$I_M(b) = 34 - 2\log_2[E_M(b)], \quad b = 0,\ldots,N_{bands} - 1 \tag{1166}$$

and is saturated such that it is limited to the range of [0,31] for 0 and [0, 39] for the others.

The quantization index of the lowest-frequency band, i.e., $I_M(0)$, is directly transmitted to the decoder. The quantization indices of the norms in the remaining $N_{bands} - 1$ bands are differentially coded by computing

$$\Delta I_M(b) = I_M(b+1) - I_M(b), \quad b = 0,\ldots,N_{bands} - 2 \tag{1167}$$

The differential indices $\Delta I_M(b)$ are constrained into the range of [–15, 16]. This is performed by first adjusting negative differential indices and then adjusting positive differential indices as follows:

1) Compute the differential indices defined in equation (1167) in order from the highest-frequency band to the lowest-frequency.

2) *if* $\Delta I_M(b) < -15$, *then* $I_M(b) = I_M(b+1)$, $b = N_{bands} - 2,\ldots,0$

3) Recompute the differential indices $\Delta I_M(b)$ in order from the lowest-frequency band to the highest-frequency.

4) *if* $\Delta I_M(b) > 16$, *then* $\Delta I_M(b) = 16$ *and* $I_M(b+1) = I_M(b) + 16$, $b = 0,\ldots,N_{bands} - 2$

5) The adjusted differential indices in the range [0, 31] are obtained by adding an offset of 15 to $\Delta I_M(b)$.

5.3.4.2.1.2 Envelope coding

The first index of the quantized norm $I_M(0)$ is transmitted by packing the bits directly using 5 bits. The adjusted quantization differential indices are coded by one of four high-rate HQ norm coding modes shown in table 131. The mode requiring the least bits is selected and used for high-rate HQ norm coding. This HQ norm coding mode information is signalled with 2 bits. The final resulting reconstructed quantized norms are denoted by $\hat{I}_M(b)$, $b = 0,\ldots,N_{bands} - 1$.

Table 131: High-rate HQ norm coding modes

Mode index	Description
0	Context based Huffman coding
1	Resized Huffman coding
2	Normal Huffman coding
3	Bit-packing

5.3.4.2.1.2.1   Context based Huffman coding mode

If this coding mode is selected for the current frame, the context based Huffman coding is applied to the quantization differential indices described in subclause 5.3.4.1.3.3.1. In this case equation (1038) shall be replaced with

$$ctx_{HQ,norm}\left(\Delta I_M(b)\right) = \begin{cases} I_M(0) - offset_{HQ,norm} & for\ b = 0 \\ \Delta I_M(b-1) & for\ b = 1,\ldots,N_{bands} - 2 \end{cases} \tag{1168}$$

When the bit-packing mode is selected the adjusted differential rates are packed directly with 5 bits.

5.3.4.2.1.2.2   Re-sized Huffman coding mode

If this coding mode is selected for the current frame, the Re-sized Huffman coding is applied to the adjusted differential indices as same as the low-rate HQ coder. Details are described in subclause 5.3.4.1.3.3.2. The Huffman codes for the differential indices are given in table 115. When the bit-packing mode is selected the adjusted differential rates are packed directly with 5 bits.

5.3.4.2.1.2.3   Normal Huffman coding and bit-packing mode

If this coding mode is selected for the current frame, the Normal Huffman coding is then applied to the adjusted differential indices. The Huffman codes for the differential indices are given in table 132.

Table 132: Huffman coefficient table

Index	Code	Index	Code	Index	Code	Index	Code
0	0011010	8	001100	16	000	24	0011110
1	0111010	9	011100	17	010	25	0111110
2	1011010	10	101100	18	1010	26	1011110
3	1111010	11	111100	19	1110	27	1111110
4	0011011	12	0010	20	001110	28	0011111
5	0111011	13	0110	21	011110	29	0111111
6	1011011	14	100	22	101110	30	1011111
7	1111011	15	110	23	111110	31	1111111

When the bit-packing mode is selected the adjusted differential rates are packed directly with 5 bits.

5.3.4.2.1.3   Bit allocation

5.3.4.2.1.3.1   Envelope adjustment before bit allocation

In order to account for psycho-acoustical weighting and masking effects, the quantized norms are adjusted prior to bit allocation. The algorithm consists of mapping the quantized norms by using spectral weighting functions. This algorithm is only used at FB.

First, the quantized norms are mapped to the spectral domain. This is equivalent to copying the quantized norms in case of stationary signals and averaging the time-dependent quantized norms of the four spectra in case of transients. This is performed according to:

$$I_{map}(b) = \begin{cases} \dfrac{1}{4}\displaystyle\sum_{k=0}^{3} I_M(4\lfloor b/4 \rfloor + k), & \text{if } IsTransient = TRUE \\ I_M(b), & \text{otherwise} \end{cases} \tag{1169}$$

The obtained spectrum is afterwards mapped to a function which is similar to the ear's output of auditory filters; this gives a representation of the psycho-acoustical importance of the input signal. This operation is performed according to the following linear operation:

$$I_{map}(p) = \frac{1}{H_{map}(p)} \sum_{b \in J_{map}(p)} I_{map}(b) + T_{map}(p), \qquad p = 0,\ldots,N_{map}-1 \tag{1170}$$

where the constants $H_{map}(p)$, $T_{map}(p)$ and the summation interval $J_{map}(p)$ are given in table 133. $N_{map} = 17$ when the bit-rate is 24.4 kb/s, $N_{map} = 18$ at 32 kb/s, and $N_{map} = 20$ at 64 kb/s.

Table 133: Spectrum mapping variables

p	$J_{map}(p)$	$H_{map}(p)$	$T_{map}(p)$	$A_{map}(p)$
0	0	1	3	8
1	1	1	3	6
2	2	1	3	3
3	3	1	3	3
4	4	1	3	3
5	5	1	3	3
6	6	1	3	3
7	7	1	3	3
8	8	1	3	3
9	9	1	3	3
10	10, 11	2	4	3
11	12, 13	2	4	3
12	14, 15	2	4	3
13	16, 17	2	5	3
14	18, 19	2	5	3
15	20, 21, 22, 23	4	6	3
16	24, 25, 26	3	6	4
17	27, 28, 29	3	6	5
18	30, 31, 32, 33, 34	5	7	7
19	35, 36, 37, 38, 39, 40, 41, 42, 43	9	8	11

The mapped spectrum is forward-smoothed according to the following:

$$I_{map}(p) = \max(I_{map}(p),\, I_{map}(p-1)-4), \qquad p = 1\ldots,N_{map} \tag{1171}$$

and the resulting in-place function is backward-smoothed according to:

$$I_{map}(p) = \max(I_{map}(p),\, I_{map}(p+1)-8), \qquad p = N_{map}-2,\ldots,0 \tag{1172}$$

After the smoothing operation, the resulting function is thresholded in order to take into account an average level of absolute threshold of hearing. Thresholding and renormalization are performed according to:

$$I_{map}(p) = T(p) - \max(I_{map}(p),\, A_{map}(p)), \quad p = 0,\ldots,N_{map}-1 \tag{1173}$$

where $A_{map}(p)$ is given by table 133 and represents a pseudo-threshold of hearing.

The resulting function is further adaptively mapped, companded or expanded depending on the dynamic range of the spectrum, to the range of $[-0,\ldots,3]$ according to the following linear mapping:

$$I_{map}(p) = 4 \cdot \frac{I_{map}(p) - \min(I_{map}(p))}{\max(I_{map}(p)) - \min(I_{map}(p))}, \qquad p = 0,\ldots,N_{map}-1 \tag{1174}$$

The resulting spectrum is mapped back to bands according to:

$$I_{adj}(b) = I_{map}(p), \quad \text{such that } b \in J_{map}(p), \ b = 0,\ldots N_{bands}-1 \tag{1175}$$

If the variable *IsTransient* is set to *TRUE*, i.e., transient mode, the resulting spectrum is further smoothed according to:

$$I_{adj}(b) = \begin{cases} \dfrac{1}{4}\displaystyle\sum_{k=0}^{3} I_{adj}(4\lfloor b/4 \rfloor + k), & \text{if } IsTransient = TRUE \\[2mm] I_{adj}(b), & \text{otherwise} \end{cases} \tag{1176}$$

Finally, the norms used for bit allocation are computed as:

$$\hat{I}_M(b) = \hat{I}_M(b) + I_{adj}(b) \tag{1177}$$

5.3.4.2.1.3.2 Envelope based bit allocation

5.3.4.2.1.3.2.1 Wideband bit allocation of non-transient mode at 24.4/32kbps

A group based bit allocation scheme has been introduced for wideband signal coding at 24.4kbps and 32kbps to avoid zero bit allocations to some of the sub-bands when they may be perceptually important.

5.3.4.2.1.3.2.1.1 Group based bit allocation

The sub-bands are divided into three groups. Firstly, the initial number of allocated bits to each group is determined according to the sum or the average of the norms in each group. Based on the initial number of allocated bits to each group, the second stage group based bit allocation is performed according to the characteristics and the energy information of the signal.

To achieve the group based bit allocation, firstly the average of the quantized norms in the index range $b = [10,17]$ is calculated,

$$I_{av} = \sum_{b=10}^{17} I_N^q(b)/8 \tag{1178}$$

For the first 4 sub-bands, if the norm $I_N^q(b)$ for bit allocation is less than I_{av}, then the norm $I_N^q(b)$ is set to I_{av}.

As a first step in the bit allocation, 1 bit is allocated to code each sub-band, and then the sub-bands are divided into 3 groups, i.e. $[0,\ldots, 15]$, $[16,\ldots, 23]$, $[24,\ldots, 25]$.

Then calculate the following parameters which indicate the characteristics and the energy information of the signal for group based bit allocation.

The factors $f_{norm\_modi}(0) = 2, f_{norm\_modi}(1) = 3$ are initialized. These factors are then employed to adjust the norms, and then the average of the norms in each group is calculated along with the sum of the averages.

$$I_{av}^G(g) = \sum_{b=g_{start}(g)}^{g_{end}(g)} I_N^q(b)/N_G(g), \qquad g = 0,\cdots,2 \tag{1179}$$

$$I_{av\_sum}^G = \sum_{g=0}^{2} I_{av}^G(g), \tag{1180}$$

Table 134: Spectrum mapping group structure

g	$g_{start}(g)$	$g_{end}(g)$	$N_G(g)$
0	0	15	16
1	16	23	8
2	24	25	2

The differences of the consecutive averages and calculated:

$$I_{av\_diff}^G(g) = I_{av}^G(g) - I_{av}^G(g+1), \qquad g = 0,1 \tag{1181}$$

If $I_{av\_diff}^G(0)$ is greater than or equal to 6, or $I_{av\_diff}^G(1)$ is greater than or equal to 3.75, then the number of bits allocated to each group is described as follows,

$$\begin{cases} R_G(0) = \left\lfloor R_{tot}' * I_{av}^G(0) / I_{av\_sum}^G \right\rfloor \\ R_G(1) = \left\lfloor 0.92 * R_{tot}' * I_{av}^G(1) / I_{av\_sum}^G \right\rfloor \\ R_G(2) = R_{tot}' - R_G(0) - R_G(1) \end{cases} \tag{1182}$$

where R_{tot}' is the total available bit budget. Otherwise, the bit allocation for each group is detailed as follows:

Initialize the number of bits allocated to each group using the following equation,

$$R_G(g) = \left\lfloor R_{tot}' * I_{av}^G(g) / I_{av\_sum}^G \right\rfloor, \qquad g = 0,1,2 \tag{1183}$$

If the average envelope of the third group $I_{av}^G(2)$ is more than 12, then adjust the number of bits of each group further:

$$\begin{aligned} &if\ (R_G(0) > 288)\{ \\ &\quad R_{sav} = R_G(0) - 288; \\ &\quad R_G(0) = 288; \\ &\} \\ &if\ (R_G(1) > 256)\{ \\ &\quad R_{sav} = R_{sav} + R_G(1) - 256; \\ &\quad R_G(1) = 256; \\ &\} \\ &if\ (R_G(2) > 96)\{ \\ &\quad R_{sav} = R_{sav} + R_G(2) - 96; \\ &\quad R_G(2) = 96; \\ &\} \end{aligned} \tag{1184}$$

where R_{sav} is the number of the saved bits and initialized to zero. If $R_{sav} > 0$, the saved bits are re-allocated to each group according to:

$$\begin{cases} R_G(0) = R_G(0) + R_{sav}/2, & if\ R_G(0) != 288 \\ R_G(1) = R_G(1) + R_{sav}/2, & if\ R_G(0) = 288 \\ R_G(1) = R_{tot}' - R_G(0) - R_G(2), & if\ R_G(0) != 288\ and\ R_G(1) != 256 \\ R_G(2) = R_{tot}' - R_G(0) - R_G(1), & if\ R_G(0) = 288\ or\ (R_G(0) != 288\ and\ R_G(1) = 256) \end{cases} \tag{1185}$$

and reset the factors $f_{norm\_modi}(0) = 2$ $\ and\ $ $f_{norm\_modi}(1) = 1.5$.

If the number of bits allocated to the third group $R_G(2)$ is less than 3, then the number of allocated bits in the third group are moved to the second group $R_G(1) = R_G(1) + R_G(2)$, and the number of bits allocated to the third group is set to zero, $R_G(2) = 0$.

5.3.4.2.1.3.2.1.2 Bit allocation in each group

For the 3 sub-band groups, the smallest thresholds $Thr_b^G(g)$ for the number of allocated bits are 5, 6 and 7, respectively, and the largest number of the sub-bands $\hat{N}_b^G(g)$ which are bit-allocated is calculated according to:

$$\hat{N}_b^G(g) = \frac{R_G(g)}{Thr_b^G(g)}, \qquad g = 0,1,2 \qquad (1186)$$

The norms in each group are re-ordered and the re-ordered norms are $I_{re\_order}^G(p), p = 0,\ldots,N_b^G(g)-1$, where $N_b^G(g)$ is the number of the sub-bands in each group.

The step length $\gamma = 1/N_b^G(g)$ is initialized, and the re-ordered norms are adjusted according to the step length, $f_{norm\_modi}(0)$ and $f_{norm\_modi}(1)$:

$$\begin{cases} \hat{I}_{re\_order}^G(b) = \left\lfloor I_{re\_order}^G(b) * (f_{norm\_modi}(0) - b*\gamma) \right\rfloor, & \text{the first group} \\ \hat{I}_{re\_order}^G(b) = \left\lfloor I_{re\_order}^G(b) * (f_{norm\_modi}(1) - b*\gamma) \right\rfloor, & \text{the second or third group} \end{cases} \qquad (1187)$$

The bits allocated to each sub-band are set according to the adjusted norms in each group as follows:

If $\hat{N}_b^G(g) \geq 0.75 * N_b^G(g)$, $g = 0,1,2$, initialize the allocated bit of each sub-band to 1. If the adjusted norm is less than 0, reset it to 0. Then the following 4 steps are performed :

1. Initialize the counter $c_b^G = N_b^G(g) - 1$.

2. Calculate the sum of the first c_b^G norms in the group, I_{sum}^G , and allocate the bits of each sub-band in the index range $b = [0, c_b^G]$:

$$R_b^G(p) = R_G(g) * \hat{I}_{re\_order}^G(p) / I_{sum}^G, \qquad p = 0,\cdots,c_b^G \qquad (1188)$$

3. If the number of bits allocated to the last sub-band is fewer than thresholds $Thr_b^G(g)$, $I_{sum}^G = I_{sum}^G - \hat{I}_{re\_order}^G(c_b^G)$, then the number of allocated bits in the $c_b^G th$ sub-band is set to zero and the counter c_b^G is decremented by one. Processing now returns to step 2.

4. If the number of bits allocated to the $c_b^G th$ sub-band is not fewer than thresholds $Thr_b^G(g)$, then the bit-allocation of the sub-bands in each group has been completed.

Otherwise, if $\hat{N}_b^G(g) < 0.75 * N_b^G(g)$, initialize the number of allocated bits of the first $\hat{N}_b^G(g)$ sub-bands to 1, and initialize the bit of the last $N_b^G(g) - \hat{N}_b^G(g)$ sub-bands to 0. If the adjusted norms of the first $\hat{N}_b^G(g)$ sub-bands are less than 0, set them to 0. Then the following 4 steps are performed:

1. Initialize the counter $c_b^G = \hat{N}_b^G(g)$.

2. Calculate the sum of the first c_b^G norms in the group, I_{sum}^G , and allocate the bits of each sub-band in the index range $b = [0, c_b^G - 1]$:

$$R_b^G(p) = R_G(g) * \hat{I}_{re\_order}^G(p) / I_{sum}^G, \qquad p = 0,\cdots,c_b^G - 1 \qquad (1189)$$

3. If the number of the bits allocated to sub-band q is fewer than thresholds $Thr_b^G(g)$,

$$I_{sum}^G = I_{sum}^G - \sum_{p=q}^{c_b^G-1} I_{re\_order}^G(p) \text{, then the number of allocated bits in the last } c_b^G - q \text{ sub-bands is set to zero}$$

and the counter c_b^G is set to to q. Processing now returns to step 2.

4. If the number of bits allocated to all c_b^G sub-bands is not fewer than thresholds $Thr_b^G(g)$, then the bit-allocation of the sub-bands in each group has been completed.

5.3.4.2.1.3.2.2 General envelope based bit allocation

The adaptive bit-allocation scheme uses the adjusted quantized norms $\hat{I}_M(b), b=0,\ldots,N_{bands}-1$, to allocate the available bits in a frame among the bands.

The maximum number of bits assigned to each normalized transform coefficient is by default set to $R_{max}=9$ bits/coefficient.

First, the bit-allocation vector entries, i.e., bit allocation of each band in bits per sample, are set to zero. This is done according to:

$$R_c(b)=0 \text{ for } b=0,\ldots,N_{bands}-1 \tag{1190}$$

The number of remainder bits, denoted R_{rem}, is set to the total available bit budget R'_{tot}. The latter is calculated after subtraction of the number of signalling bits, the bits used by the envelope coding and possibly the noise level bits (see subclause 5.3.4.2.1.4) from the total available bits for the frame.

The vector of bit-allocations, $R(b)$, $b=0,\ldots,N_{bands}-1$, remainder R_{rem} and R'_{tot} fulfil:

$$\sum_{b=0}^{N_{bands}-1} R_c(b)L_M(b) + R_{rem} = R'_{tot} \tag{1191}$$

At each iteration, the index b_{max} of the band which has the largest norm among number of bands used is found:

$$b_{\max} = \underset{b=0,\ldots,N_{bands}-1}{\arg\max}\left(\hat{I}_M(b)\right) \tag{1192}$$

For this band, the algorithm allocates 1 bit for each spectral coefficient, i.e., $R(b_{\max})$ is incremented by 1. The norm will, on the other hand, be decremented by 6 dB, i.e., the norm index is decreased by two. These two operations are performed according to:

$$\begin{aligned} R_c(b_{\max}) &= R_c(b_{\max})+1 \\ \hat{I}_M(b_{\max}) &= \hat{I}_M(b_{\max})-2 \end{aligned} \tag{1193}$$

The remainder Ω is updated as well to take into account the updated bit-allocation vector:

$$R_{rem} = R_{rem} - L_M(b_{\max}) \tag{1194}$$

Whenever the bit-allocation $R_c(b_{\max})$ reaches the maximum allowable bit-rate R_{max}, its norm is set to minus infinity (in reality, to -2^{15}) such that this vector is not taken into account in the next iterations. This procedure is repeated until R_{rem} is less than $L_M(b_{\max})$.

When the iterative procedure stops and, depending on the value of the remainder bits, the remaining bits are allocated to bands of a lower dimension than $L_M(b_{\max})$ which caused the stop of the bit-allocation loop.

The last step is to convert the bit allocation from bits per coefficients to total bits per band:

$$R(b) = R_c(b) \cdot L_M(b) \tag{1195}$$

5.3.4.2.1.3a Fine structure quantization

The normalized MDCT spectrum $\tilde{X}_M(k)$ is encoded in bands using the bit allocation $R(b)$.

5.3.4.2.1.3.a1 Fine gain bit allocation

The spectral envelope coefficients $\hat{E}_M(k)$ have been quantized in 3 dB steps. For higher rates, this resolution becomes too coarse and additional fine gain adjustments are made. The bits assigned for the band $R(b)$ is split into a number of bits for the PVQ shape quantizer $R_{PVQ}(b)$ and bits for the fine gain quantizer $R_{FG}(b)$. The assignment is done using a pre-trained look-up table based on the assigned band bits $R(b)$ and the bandwidth $L_M(b)$ according t

$$R_{FG}(b) = fine\_gain\_bits(R_{samp}) \tag{1196}$$

where $fine\_gain\_bits(R_{samp})$ is a lookup-table for fine gain bits (see table 135) and $R_{samp} = \max\left(\lfloor R(b)/L_M(b)\rfloor, 7\right)$ is the number of bits per sample rounded down. The bits for the PVQ shape quantization are obtained by subtracting the fine gain bits, $R_{PVQ}(b) = R(b) - R_{FG}(b)$.

Table 135: Fine gain bits table

R_{samp}	R_{FG}
0	0
1	0
2	0
3	1
4	2
5	2
6	4
7	5

5.3.4.2.1.3a.2 Fine structure quantization using PVQ

The quantization of the PVQ shape vector is performed as described in subclause 5.3.4.2.7. A decoded version of the PVQ shape vector is also obtained.

5.3.4.2.1.3a.3 Fine gain prediction, quantization and application

The fine gain adjustment is based on a predicted value, $g_{pred}(b)$. It is formed using an accuracy measure $a(b)$ which combines the band bitrate $R(b)$, band size $L_M(b)$ and the largest absolute integer pulse value $P_{max}(b) = \max_{k \in} |y(k)|$ of the synthesized band $y(k)$.

$$a(b) = N_{pulses}(b)/L_M(b) \cdot P_{max}(b) \tag{1197}$$

where $N_{pulses}(b)$ is the actual number of pulses used for encoding $y(k)$ and depends on the bitrate $R(b)$ and the PVQ encoding process. The accuracy measure $a(b)$ is then translated into a gain prediction $g_{pred}(b)$ following this relation:

$$g_{pred}(b) = 1 - 0.05/a(b) \tag{1198}$$

where $g_{pred}(b)$ approximates the optimal MMSE gain. In case there are bits allocated, further refinement of the fine gain may be done by encoding the gain prediction error g_{err}, defined as

$$g_{err}(b) = \frac{g_{opt}(b)}{g_{rms1}(b)g_{pred}(b)} \tag{1199}$$

where the optimal MMSE gain $g_{opt}(b)$ is defined as

$$g_{opt}(b) = \frac{\sum_{k=k_{start}(b)}^{k_{end}(b)} \widetilde{X}(k) x_q(k - k_{start}(b))}{\sum_{k=k_{start}(b)}^{k_{end}(b)} x_q(k - k_{start}(b))^2}$$

(1200)

and the normalization factor $g_{rms1}(b)$ to RMS=1.0 is

$$g_{rms1}(b) = \sqrt{\frac{L_M(b)}{\sum_{k=k_{start}(b)}^{k_{end}(b)} x_q(k - k_{start}(b))^2}}$$

(1201)

The gain prediction error $g_{err}(b)$ is encoded with a non-uniform scalar quantizer in log domain using $R_{FG}(b)$ bits.

5.3.4.2.1.4 Noise level adjustment

The spectral coefficients which belong to bands which are assigned zero bits from the bit-allocation procedure are not encoded. This means that not all transform coefficients are transmitted to the decoder.

The level of these non-coded spectral coefficients is estimated and quantized in the encoder. This is not done in WB at rates 24.4 kb/s and 32 kb/s.

The estimation of the non-quantized signal level is done directly in the normalized spectrum domain. Prior to estimating the noise level, a transition frequency between the noise fill region and high frequency noise fill region is estimated. This transition frequency is identically estimated in the encoder and decoder and marks the start of high frequency noise fill and the end of noise fill.

The transition frequency is estimated according to the criterion of the last quantized band. The general method consists in looping through all the bands, starting at $N_{bands} - 1$ down to 0. If there are no quantized coefficients in the current band, it will be flagged as filled by high frequency noise fill. If there are quantized coefficients in the band, the holes of this band as well as the following bands are filled using noise fill. Figure 80 illustrates such a procedure, where the transient frequency separates between noise-fill and high frequency noise-fill.
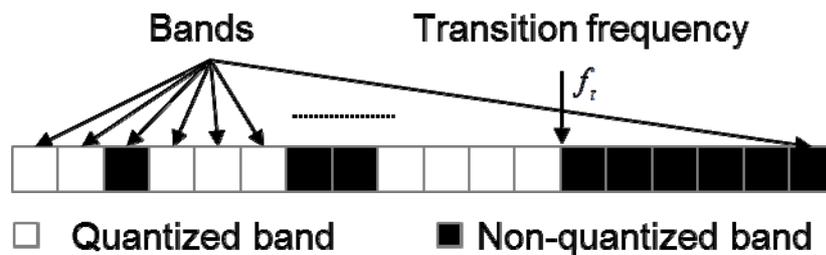
Figure 80: Estimation of transition frequency
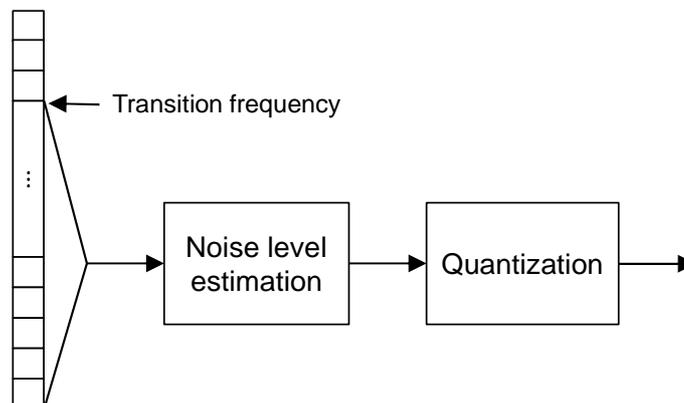
Figure 81: Noise level estimation done below the transition frequency

The noise level is in turn estimated by measuring the average level of the normalized non-coded signal below the transition frequency f_t, see figure 81. Formally this is obtained by:

$$NoiseLevel = \frac{1}{|\Omega|}\sum_{k \in \Omega} \log_2(X_M(k)^2) - \log_2\left\{\frac{1}{|\Omega|}\sum_{k \in \Omega} X_M(k)^2\right\} \tag{1202}$$

where Ω is the set of indices of coefficients allocated zero bits below the transition frequency f_t.

The above equation always returns a value which is below zero because of the convexity of the logarithm. The noise level is quantized using a two-bit scalar quantizer according to table 136.

Table 136: Codebook entries for the uniform noise level quantizer

Index	Output quantized NoiseLevel (dB)
0	0
1	−6
2	−12
3	−18

5.3.4.2.2 Transient Mode

5.3.4.2.2.1 Envelope calculation and quantization

The bands are sorted according to equation (1203), where $b_{sort}(b)$ is defined in tables 137, 138, and 139. Then the envelope calculation is done as in subclause 5.3.4.2.1.1 but using $\hat{I}_M^{sort}(b)$ instead of $\hat{I}_M(b)$. After the quantization then the sorting is inverted by doing the inverse step of equation (1203).

$$\hat{I}_M^{sort}(b) = \hat{I}_M\left(b_{sort}(b)\right) \tag{1203}$$

Table 137: Envelope sorting table in transient mode for WB

b	0	1	2	3	4	5	6	7	8	9	10
b_{sort}	0	1	8	9	16	20	24	21	17	11	10
b	11	12	13	14	15	16	17	18	19	20	21
b_{sort}	3	2	4	5	12	13	18	22	25	23	19
b	22	23	24	25							
b_{sort}	15	14	7	6							

Table 138: Envelope sorting table in transient mode for SWB

b	0	1	2	3	4	5	6	7	8	9	10
b_{sort}	0	1	8	9	16	20	24	28	32	36	37
b	11	12	13	14	15	16	17	18	19	20	21
b_{sort}	33	29	25	21	17	11	10	3	2	4	5
b	22	23	24	25	26	27	28	29	30	31	32
b_{sort}	12	13	18	22	26	30	34	38	35	31	27
b	33	34	35	36	37	38					
b_{sort}	23	19	15	14	7	6					

Table 139: Envelope sorting table in transient mode for FB

b	0	1	2	3	4	5	6	7	8	9	10
b_{sort}	0	1	8	9	16	20	24	28	32	36	40
b	11	12	13	14	15	16	17	18	19	20	21
b_{sort}	41	37	33	29	25	21	17	11	10	3	2
b	22	23	24	25	26	27	28	29	30	31	32
b_{sort}	4	5	12	13	18	22	26	30	34	38	42
b	33	34	35	36	37	38	39	40	41	42	43
b_{sort}	43	39	35	31	27	23	19	15	14	7	6

5.3.4.2.2.2 Bit allocation

For most bit rates bit allocation is done as is described in subclause 5.3.4.2.1.2.1 and 5.3.4.2.1.2.2. For SWB at 24.4 and 32 kbps the Generic mode bit allocation is used, as described in subclause 5.3.4.2.5.6.

5.3.4.2.2.3 Fine structure quantization using PVQ

The spectral coefficient quantization is done as is described in subclause 5.3.4.2.1.4.

5.3.4.2.3 Generic, Harmonic and HVQ mode detector

Non-transient signals are further classified as Generic mode, Harmonic mode or HVQ at 24.4 kb/s and 32 kb/s for SWB inputs. The detailed classification is described in subclause 5.3.4.1.1, with slight differences in the final decision logic as follows.

At 24.4 kb/s the current frame mode is classified as harmonic if $mode\_countl < 5$, $c_{sharp\_l} + c_{sharp\_h} \geq 5$ and $c_{sharp\_h} > 2$.

At 32 kb/s the current frame mode is classified as harmonic if $mode\_countl < 5$, $c_{sharp\_l} + c_{sharp\_h} \geq 10$ and $c_{sharp\_h} > 5$; or if $mode\_count \geq 5$.

5.3.4.2.3.1 HVQ classifier

Instantaneous noise-level $E_{ne}(k)$ and peak-level $E_{pe}(k)$ are estimated from the absolute values of transform coefficients $|X_M(k)|$, where $L = 224$ at 24.4 kb/s and $L = 320$ at 32 kb/s. The noise-level is calculated as:

$$E_{ne}(k) = \alpha E_{ne}(k) + (1 - \alpha)|X_M(k)| \qquad k = 0, \cdots, L-1 \qquad (1204)$$

where

$$\alpha = \begin{cases} 0.9578 & if \quad |X_M(k)| > E_{ne}(k-1) \\ 0.6472 & otherwise \end{cases} \qquad (1205)$$

The peak-level is calculated as

$$E_{pe}(k) = \beta E_{pe}(k) + (1 - \beta)|X_M(k)| \qquad k = 0, \cdots, L-1 \qquad (1206)$$

where

$$\beta = \begin{cases} 0.4223 & if \quad |X_M(k)| > E_{pe}(k-1) \\ 0.8029 & otherwise \end{cases} \qquad (1207)$$

and both $E_{ne}(-1)$ and $E_{pe}(-1)$ are initialized to 800.

The per-band averages of noise-level $\overline{E}_{ne}(b)$ and peak-level $\overline{E}_{pe}(b)$ are calculated by averaging instantaneous level in a band (every 32 bins). The number of bands is $N = 7$ at 24.4 kb/s and $N = 10$ at 32 kb/s.

The average of the elements of the first half of $\overline{E}_{ne}$ gives the noise-floor gain $G_{ne}(0)$, while the second half of coefficients forms $G_{ne}(1)$. In a similar way the per-band peak-levels $\overline{E}_{pe}$ produce two peak energy gains $G_{pe}(0)$ and $G_{pe}(1)$.

The decision to switch to HVQ mode is based on the threshold in table 140, and three variables, N_{peaks}, N_{sharp}, and D_{sharp}, defined below.

The threshold for selecting peak candidates is calculated as:

$$\Theta(b) = \left(\frac{\overline{E}_{pe}(b)}{\overline{E}_{ne}(b)} \right)^{0.88} \overline{E}_{pe}(b) \tag{1208}$$

Absolute values of transform coefficients $|X_M|$ are compared to the threshold $\Theta(b)$, and the ones with amplitude above it, form a vector of peak candidates. Elements from the peaks candidate vector are extracted in decreasing order, and when a peak is extracted the threshold over the neighboring peaks is adjusted with {0.7071068, 0.5000000, 0.2500000, 0.5000000, 0.7071068}. This procedure produces a set of spectral peaks, with number N_{peaks}.

A measure of frequency sharpness per-band $Sharp_{ne}(b)$, similar to the one in 5.3.4.1.1, is defined as peak to noise-floor ratio in each band:

$$Sharp_{ne}(b) = \frac{X_{\max}(b)}{\overline{E}_{ne}(b)} \tag{1209}$$

Variable N_{sharp} is calculated as the number of bands for each $Sharp_{ne}(b) > 9$

Variable D_{sharp} is calculated as

$$D_{sharp} = \sum_{b \in N} \left(Sharp_{ne}(b) - 9 \right) \tag{1210}$$

Table 140: Thresholds for HVQ mode decision

rate	T_s	T_p	T_d
24.4 kb/s	4	20	22
32 kb/s	7	23	22

The current frame is encoded in HVQ mode if: $N_{sharp} > T_s$, $N_{peaks} \leq T_p$, and $D_{sharp} > T_d$

5.3.4.2.4 Harmonic Mode

The harmonic mode is used to code the harmonic-like signal. For harmonic signals, usually less noise filling is performed. It is also important to mitigate any discontinuity in the higher sub-band due to changes in core coding and for the most part it is preferable at 24.4 kb/s and 32kb/s, when there are insufficient bits to encode the full signal, to stop short of coding right up to the Nyquist frequency.

5.3.4.2.4.1 Envelope calculation and quantization

Envelope calculation and quantization is performed as described in subclause 5.3.4.2.1.1.

In order to reconstruct harmonic characteristics of the higher frequency band signal, the widths of the bands are larger than the ones for non-harmonic mode.

5.3.4.2.4.2 Bit allocation

The band-energy limitation factor $f_{harmonic}$ is introduced before bit allocation to mitigate discontinuous core coding in the higher sub-band.

Initialize the band-energy limitation factor $f_{harmonic}$

$$f_{harmonic} = \begin{cases} 0.885, & 24.4\text{kbps} \\ 0.942, & 32\text{kbps} \end{cases} \tag{1211}$$

Calculate the energy of the first 10 sub-bands E_{I_l} and the energy of the first 28 sub-bands E_{I_t}, and add the norms $I_N^q(b), b = [10,28]$ to E_{I_l} when $E_{I\_l} < f_{harmonic} * E_{I\_t}$. When $E_{I\_t} \geq f_{harmonic} * E_{I\_l}$, b is the index of the highest encoded sub-band.

Reorder the quantized norms with the index range $b = [0, b_{index}]$ to obtain the reordered norms, and adjust the quantized norms as follows:

$$I_{N\_reorder}^q(b) = I_{N\_reorder}^q(b_{index}/2 - 1), \qquad b = b_{index}/2,\cdots,b_{index} \tag{1212}$$

Then, the bit allocation is performed based on the adjusted norms to the sub-bands with the index range $b = [0, b_{index}]$ as described in subclause 5.3.4.2.1.2.2.

5.3.4.2.4.3 PVQ

The spectral coefficient quantization and coding is done according to the number of bits allocated to each sub-band as is described in subclause 5.3.4.2.1.3.

5.3.4.2.5 HVQ

The HVQ mode is used only for SWB signals at 24.4 kb/s and 32 kb/s. At 24.4 kb/s it initially codes the first 224 MDCT coefficients (this corresponds to frequency range up to 5.6 kHz), while at 32 kb/s it initially codes the first 320 coefficients (this corresponds to frequency range up to 8 kHz). The major algorithmic steps at the encoder are: detect and code spectral peaks regions, code low-frequency spectral coefficients (the size of coded region depends on the remaining bits after peaks coding), code noise-floor gains for spectral coefficients outside the peaks regions, code high-frequency spectrum envelope to be used with the high-frequency noise-fill.

The input to the HVQ mode is the set of MDCT coefficients $X_M(k)$, the noise-floor gains G_{ne}, and the spectral peaks (both noise-floor gains and spectral peaks are calculated in the classification module, described in subclause 5.3.4.2.3.1). Each peak is normalized to unit energy and the surrounding 4 neighbours are normalized to with the peak gain. The peaks position, gain and sign are quantized. A VQ is applied to the four MDCT bins surrounding each peak. The coded number of peaks, peaks position, gain and sign, as well as the surrounding shape vectors are quantized and quantization indices transmitted to the decoder.

First sorted by energy peaks are arranged by position. Then peaks amplitudes $G_p(k)$ are differentially coded by 5 bits SQ on a log domain, and the indices Huffman coded to form a quantized peak gains $\hat{G}_p(k)$. Prior to quantization the peak gains are multiplied by 0.25 and after the quantization multiplied by 4.

The spectral peak positions P_k are coded by choosing between two alternative lossless coding schemes, where the coding scheme that requires the least number of bits is selected, and explicitly indicated to the decoder.

The first lossless spectral peak position coding scheme is delta Huffman coding, suitable for periodic or semi-periodic spectral peak position distributions. The second lossless spectral peak position coding scheme is or-ing, suitable for sparse spectral peak position distributions.

The first scheme consists of the following steps: deltas (differences) Δ_k between consecutive elements (positions) P_k are created. Then these deltas are Huffman coded. Since the peaks are selected in a way that they cannot be positioned closer than 3 positions apart, $D\_OFFSET = 3$ is subtracted from the peak differences.

$$\Delta_k = P_k - P_{k-1} - D\_OFFSET \tag{1213}$$

This eliminates the need of keeping codewords in the Huffman table, corresponding to unused deltas.

The second scheme consists of the following steps: the vector representing the spectral peak positions (absence of peak is indicated with 0) is divided into consecutive equal size (5 elements) bit groups, and the bits in each group are OR-ed forming a group bit vector (second layer), which is 5 time shorter. Each bit in this second layer indicates presence or absence or peak in the 5-dim group from layer below. In this way the only 5-dim groups from the first layer, which are not indicated as all-zero by the second layer have to be transmitted. The second (control) layer is always transmitted. The non-zero bit groups from the first layer also are mapped to exploit the constraints the fact that peaks cannot be closer than 3 positions (not all possible 5-dim vectors are allowed).

The selected coding scheme is explicitly signaled to the decoder with one bit: $CS = 1$ indicates sparse coding scheme, while $CS = 0$ indicates usage of delta coding scheme. Here $\Delta_{\max}$ is the largest distance between two consecutive peaks possible to code with the pre-stored Huffman tables, R_d is the total number of bits consumed by the delta coding scheme, and R_s is the total number of bits consumed by the sparse coding scheme

$$
\begin{aligned}
&\text{if} \quad R_d > R_s \quad or \quad \Delta_{\max} > 51 \\
&\qquad CS = 1 \\
&\text{else} \\
&\qquad CS = 0
\end{aligned}
\tag{1214}
$$

Sign of the spectral peaks is coded separately with 1 bit per-peak, with 0 indicating negative sign and 1 indicating positive sign.

The peak regions to be coded are 5-dim MDCT vectors; a bin corresponding to the spectral peak and 2 MDCT bins on each side of the peak. The peak amplitude $\hat{G}_p(k)$ of the central bin is used to normalize the entire peak region. In this way the central bin is scaled to unit energy, while surrounding 4 bins are normalized relative to the central one. The shape vector $\tilde{S}(k)$ of the peak region, centered at bin k is defined as:

$$\tilde{S}(k) = \left(X_M(k-2) \quad X_M(k-1) \quad X_M(k+1) \quad X_M(k+2) \right) \big/ \hat{G}_p(k) \tag{1215}$$

Each shape vector is quantized with 9 bits; 8 bits for the VQ index and 1 bit for classification. The numbers of peak regions vary over frames; this means different number bits will be required for coding the shape vectors, which will result in variable number bits used in the PVQ coding of low-frequency MDCT bands (except for the first low-frequency band, which has reserved bits).

Variations in the number of peaks per-frame results in different number of VQs, which leads to large variation in complexity. To keep the complexity nearly constant, while achieving low quantization error, the following approach is used: the search for each $\tilde{S}(k)$ is performed in a structured CB, with dynamically selected offset and size of the search region. The starting point for the search is determined by initial classification of the input shape vector, while the length of the search region dependents on the number of received shape vectors.

The codewords in the CB used for quantization of the shape vectors are order based on their distance to two pre-demined classes, with centroids C_0 and C_1. The CB is structured in a way that the codewords closest to C_0 and most distanced to C_1 are in one side of the CB, while codewords closest to C_1 and most distanced to C_0 are clustered in the other side of the CB. The distance between the input vectors $\tilde{S}(k)$ to each of the classes determines the starting point for the search. Since the codevectors in the codebook are sorted according to a distortion measure reflecting the distance between each codevectors and the centroids, the search procedure goes first over set of vectors that is likely to contain the best match.

The search space is dynamically adjusted to the number of input vectors. The maximum search space is used with 8 peaks or less at 24.4 kb/s, and 12 peaks or less at 32 kb/s. When larger numbers of peaks are to be quantized in the current frame, the search space is reduced to limit the peak complexity.

Table 141: Adaptive search space in HVQ at 24.4 kb/s

number _ peaks	17	16	15	14	13	12	11	10	9	8	…	1
cb _ size	128	136	145	155	167	181	197	217	241	256	…	256

Table 142: Adaptive search space in HVQ at 32 kb/s

number _ peaks	23	22	21	20	19	18	17	16	15	14	13	12	…	1
cb _ size	128	134	141	149	158	168	179	192	206	224	244	256	…	256

The target shape vector exhibits certain symmetries (the MDCT coefficients on the both sides of the spectral peak have similar statistics) that can be used to optimize centroids for the class selection and the CB. A "flipped" version of the centroids used in the initial classification, "flipped" version of the CB (not pre-stored, but through modified search) can capture this symmetry in the shape vectors.

First the input shape vector is compared to 4 centroids (each centroid representing a respective class of codevectors in a codebook), which determines a starting point to the search. If C_0^f or C_1^f are selected, the same logic is used, but the search is performed in a "flipped" CB.

Table 143: Centroids for the class selection in HVQ search

Class				
C_0	-0.2324457	-0.4390556	0.0651793	0.2109977
C_1	0.1471332	-0.1351437	0.4312476	-0.1384814
C_0^f	0.2109977	0.0651793	-0.4390556	-0.2324457
C_1^f	-0.1384814	0.4312476	-0.1351437	0.1471332

In case of close peaks with overlapping shape vectors, a weighted minimum-mean-squared error is used in the VQ search. Zero weights are assigned to the overlapping shape elements that belong to the peak with lowest amplitude. In this way the CB entries are matched against the meaningful coefficients only.

After the peak regions are extracted and quantized, all remaining bits that are not reserved for signalling are used to quantize the low frequency MDCT coefficients. This is done by grouping the remaining un-quantized MDCT coefficients into 24-dimensional bands (not including already coded peak regions). A bit budget for coding the first band always exists, but the total number of coded bands depends on the bits left after peak coding. The number of bands to be coded $nbands$ with remaining bits is determined by dividing the number of available bits R_{avail} by the maximum number allowed per-band, which is set to $R_{max} = 80$ at 24.4 kb/s and $R_{max} = 95$ at 32 kb/s.

$$nbands = \left\lfloor \frac{R_{avail}}{R_{max}} \right\rfloor + \max\left(\text{sign}\left(R_{avail} \bmod R_{max} - 30\right), 0\right) \tag{1216}$$

Then the selected bands are gain-shape quantized. Gains are SQ on log domain with 5 bits, and shape are PVQ quantized as described in subclause 5.3.4.2.7.

A coded band is introduced above 5.6 kHz for 24.4 kb/s and 8 kHz for 32 kb/s if energy in the high band is relatively high compared to the peak coded region in the lower band, the band has high energy compared to the neighbouring high-frequency bands, and there is sufficient number of bits for encoding band of that size.

To encode the high-frequency band b with band log energy $I_M(b)$ and bandwidth W_b following conditions have to be simultaneously met:

$$
\begin{aligned}
I_M(b) - \bar{I}_M(b) &> 3 \\
E_M(b) &> E_{peak} \cdot 10^{-5} \\
(R_{avail} - 5) \cdot 8 &\geq pulse2bits(W_b, 1)
\end{aligned}
\tag{1217}
$$

where an estimate of the low band energy E_{peak} is obtained through summation over all peak amplitudes, coded at low-frequencies:

$$
E_{peak} = 2\log_2 \left(\sum_{k=0}^{number\_peaks-1} \hat{G}_p(k) \right)
\tag{1218}
$$

and $pulse2bits(W_b, 1) / 8$ denotes the number of bits required to encode one pulse in a band of width W_b. The average band log energy at high-frequencies is denoted by $\bar{I}_M(b)$ and the amount available bits by R_{avail}.

$$
void
\tag{1219}
$$

The two noise floor gains use in the low-frequency noise-fill $G_{ne}(0)$ and $G_{ne}(1)$ are scalar quantized with 5 bits on a log domain. The high-frequency gains $\hat{G}_h(0)$ and $\hat{G}_h(1)$ are transmitted to the decoder for the high-frequency noise-fill. First two variables: the noise-floor and peak-energy gains $G_{ne}(b)$ and $G_{pe}(b)$ are calculated, in similar way as described in subclause 5.3.4.2.3.1, but over high-frequency MDCT coefficients (coefficients above 224 at 24.4 kb/s and above 320 at 32 kb/s). That is the summation is done over $k = 224, \ldots, 639$ for 24.4 kb/s and $k = 320, \ldots, 639$ for 32 kb/s. Then the two gains are calculated as:

$$
G_h(b) = \left(6.4 \frac{G_{ne}(b)}{G_{pe}(b)} \right)^3 \qquad b = 0,1
\tag{1220}
$$

These gains are quantized with 2 bit uniform SQ to form quantized high-frequency gains $\hat{G}_{ne}(0)$ and $\hat{G}_{ne}(1)$ transmitted and used in the high-frequency noise-fill.

5.3.4.2.6   Generic Mode

The first stage in this mode is the selection of one of three types of high frequency excitation class which is followed by the separate encoding of the low and high frequency envelopes. The low frequency envelope is quantized and coded in the same manner as the normal high rate mode of the HQ coder. The high frequency envelope is first quantised in the generic mode domain before being mapped onto the HQ normal mode domain, re-quantized and then combined with the low frequency envelope. An initial bit allocation is determined, and then delta's are calculated and coded. The coded values are used to update the combined envelope, before the final bit allocation is determined.
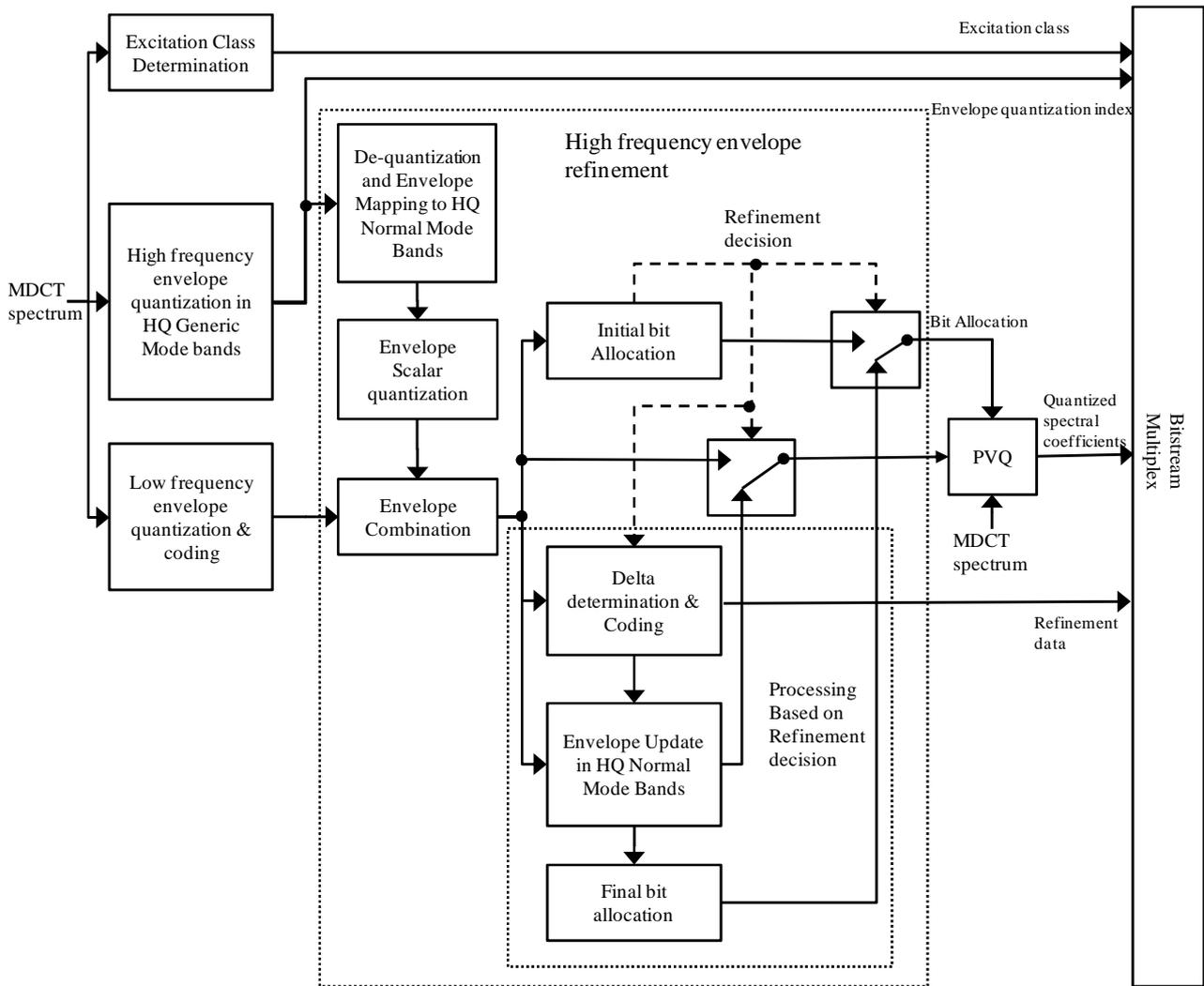
Figure 82: Generic mode Encoder Block Diagram

5.3.4.2.6.1   Band allocation for the Generic Mode

The band allocation for the low frequency envelope at 24.4 and 32kbps is the same as default mode band allocation defined in table 127. The band allocation for the high frequency envelope at 24.4 and 32kbps is shown in table 144.

Table 144: Band Allocation for the high frequency envelope

Band index	SWB, FB @ 24.4kbps			SWB, FB @ 32kbps		
	$k_{start\_G}(b)$	$k_{end\_G}(b)$	$L_{M\_G}(b)$	$k_{start\_G}(b)$	$k_{end\_G}(b)$	$L_{M\_G}(b)$
0	320	335	16	384	399	16
1	336	359	24	400	423	24
2	360	375	16	424	439	16
3	376	399	24	440	463	24
4	400	415	16	464	479	16
5	416	439	24	480	503	24
6	440	455	16	504	519	16
7	456	479	24	520	543	24
8	480	503	24	544	567	24
9	504	527	24	568	591	24
10	528	551	24	592	615	24
11	552	575	24	616	639	24
12	576	607	32	640 (FB)	679 (FB)	40 (FB)
13	608	639	32	680 (FB)	719 (FB)	40 (FB)
14	640 (FB)	679 (FB)	40 (FB)	720 (FB)	799 (FB)	80 (FB)
15	680 (FB)	719 (FB)	40 (FB)	-	-	-
16	720 (FB)	799 (FB)	80 (FB)	-	-	-

5.3.4.2.6.2   High frequency Excitation Class

There are three different high frequency excitation classes, one each for speech, tonal music and non-tonal music. The *HF_Speech_excitation_class* is determined by the instantaneous result of the speech/music classifier, i.e. by applying the output of the first speech/music classifier without adding any hang-over in subclause 5.1.13.6.3.

$$HF\_Excitation\_Class = \begin{cases} HF\_Speech\_Excitation\_Class, & if\ L_s - L_m > 0.5 \\ HF\_excitation\_class0\ OR\ HF\_excitation\_class1, & Otherwise \end{cases} \quad (1221)$$

If the output of the classifier indicates music then a tonality measurement is calculated.

$$Tonality = \frac{1}{N_{bands\_t}} \sum_{b=0}^{N_{bands\_t}-1} 10\log_{10}\left[\max_{k \in b}\left(X_M(k) * X_M(k)\right) \middle/ \left(\frac{1}{L_{M\_G}(b)} * \sum_{k=k_{start\_G}(b)}^{k_{end\_G}(b)} X_M(k) * X_M(k)\right) \right] \quad (1222)$$

where N_{bands_t} is the number of bands for calculating tonality, 10 at 24.4kbps and 8 at 32kbps.

This tonality value is then thresholded to further subdivide the excitation into *HF_excitation_class0* for noisy signals or *HF_excitation_class1* for tonal signals. The bit allocations are shown in table 145.

Table 145: Bit Allocation for the HF Excitation Classes

High Frequency Excitation Classes	Code	Num of bits
HF_excitation_class0	00	2
HF_Speech_excitation_class	1	1
HF_excitation_class1	01	2

5.3.4.2.6.3   Low Frequency Envelope Quantization and Coding

The low frequency envelope is quantized and coded in the same manner as described for the normal high rate mode of the HQ coder, subclause 5.3.4.2.1.1

For both SWB and FB the transmitted low frequency envelope is approximately 8 kHz at 24.4 kbps and increases to approximately 9.6 kHz at 32kbps. The number of bands transmitted by the low frequency envelope quantization and coding N_{bands_LF} is defined as 27 at 24.4kbps and 30 at 32kbps.

5.3.4.2.6.4   High Frequency Envelope Quantization

The high frequency envelope is quantized in a similar manner as described in subclause 5.2.6.2.1.5, but with different starting frequencies. For SWB at 24.4kbps the frequency range is 8 to16 kHz with 14 dimensions, while at 32kbps the range is 9.6 to 16 kHz with 12 dimensions. For FB at 24.4kbps the frequency range is 8 to 20 kHz, while at 32kbps the range is 9.6 to 20 kHz with 3 additional bands.

First the high frequency envelope is calculated, and then the energy control tool is applied as described in subclause 5.2.6.2.1.5 at SWB bands (14 bands at 24.4kbps, 12 bands at 32kbps) for both SWB and FB. When performing energy control, the variable ec_{gam_FD} from equation (715) in subclause 5.2.6.2.1.5, is set to 0.55. However, the copied spectrum used to generate the simulated spectrum is generated by using the frequency mapping as defined in the following table, rather than the mapping used in subclause 5.2.6.2.1.5 and defined in table 60.

Table 146: Frequency mapping to generate base excitation spectrum

BW, Bit-rate	l	$St_{src\_G}(l)$	$End_{src\_G}(l)$	$St_{dst\_G}(l)$	$End_{dst\_G}(l)$
SWB, FB @ 24.4kbps	0	2	239	320	447
	1	2	239	448	575
	2	80	143	576	639
SWB, FB@ 32kbps	0	2	239	384	511
	1	2	239	512	639

VQ is then applied as described in subclause 5.2.6.2.1.5. At 24.4kbps the VQ is identical to that used at Non-TRANSIENT mode, but at the 32kbps, the VQ is modified as shown in figure 83:
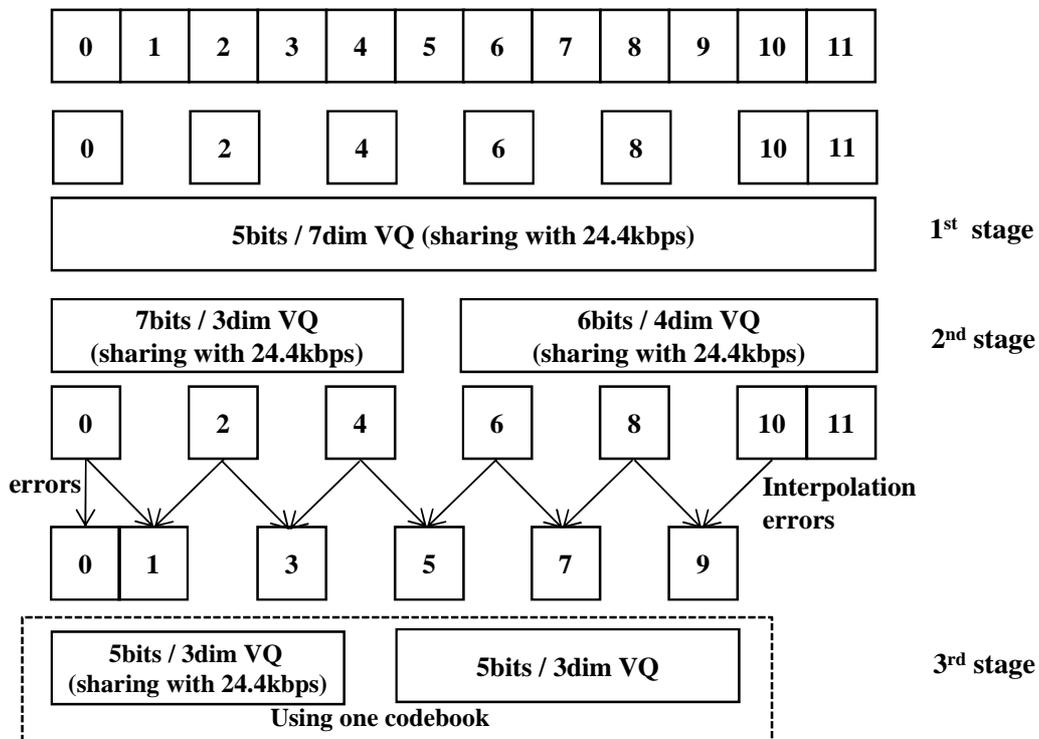
Figure 83: VQ for HQ generic mode at 32kbps

In the first stage, three candidate indices are chosen using the weighted mean squared error minimization criterion. The 6 values in even positions, as well as the last (11th) position are selected and quantized using a 7 dimensional VQ with 5 bits.

$$env_1(j) = \begin{cases} \bar{f}_{rms\_G}(2*j) & \text{for } j = 0,...,5 \\ \bar{f}_{rms\_G}(11) & \text{for } j = 6 \end{cases} \quad (1223)$$

The quantization error is calculated:

$$err_1(j) = \begin{cases} \bar{f}_{rms\_G}(2*j) - \hat{e}nv_1(j) & \text{for } j = 0,...,5 \\ \bar{f}_{rms\_G}(11) - \hat{e}nv_1(j) & \text{for } j = 6 \end{cases} \quad (1224)$$

where $\hat{e}nv_1(j)$ is the de-quantized value.

Then the errors are split into $err_{21}(j)$ and $err_{22}(j)$ and quantized:

$$\begin{aligned} err_{21}(j) &= \bar{f}_{rms\_G}(2*j) - \hat{e}nv_1(j) & \text{for } j = 0,1,2 \\ err_{22}(j) &= \bar{f}_{rms\_G}(2*(j+3)) - \hat{e}nv_1(j+3) & \text{for } j = 0,1,2,3 \end{aligned} \quad (1225)$$

The two quantized and de-quantized values $\hat{e}rr_{21}(j)$ and $\hat{e}rr_{22}(j)$ are then combined:

$$\hat{e}rr_2(j) = \begin{cases} \hat{e}rr_{21}(j) & \text{for } j = 0,1,2 \\ \hat{e}rr_{22}(j-3) & \text{for } j = 3,4,5,6 \end{cases} \quad (1226)$$

At odd positions (excluding position 11), an interpolation using boundary values is applied for intra-frame prediction and the predicted error is calculated and quantized:

$$Ierr_3(j) = \begin{cases} \bar{f}_{rms\_G}(j) - (\hat{e}nv_1(j) + \hat{e}rr_2(j)) & \text{for } j = 0 \\ \bar{f}_{rms\_G}(2*j-1) - \dfrac{(\hat{e}nv_1(j-1) + \hat{e}rr_2(j-1) + \hat{e}nv_1(j) + \hat{e}rr_2(j))}{2} & \text{for } j = 1,...,5 \end{cases} \quad (1227)$$

$Ierr_3(j)$ is then split into $Ierr_{31}(j)$ and $Ierr_{32}(j)$ and each is then quantised to 5 bits with a 3 dimensional VQ.

For FB, the energies for three additional bands are calculated. These are then quantised using 5 bits, after subtracting the mean vector (shown in table 147)

Table 147: Mean vector in FB

j	FB
0	13.75
1	6.29
2	3.70

The final selected set of indices $\{idx_{env1}, idx_{err21}, idx_{err_{22}}, idx_{Ierr_{31}}, idx_{Ierr_{32}}\}$ for SWB, or $\{idx_{env_1}, idx_{err21}, idx_{err_{22}}, idx_{Ierr_{31}}, idx_{Ierr_{32}}, idx_{FB}\}$ for FB are then transmitted.

5.3.4.2.6.5   High Frequency Envelope Refinement and Fine structure quantization

After de-quantisation the value in each band of the de-quantized high frequency envelope $\hat{E}_{HF}(b)$ is mapped to one of the HQ high rate normal mode bands $E_M(b)$ in order to match the frequencies. The following energy displacement factors are used:

$edf_0 = 0.25, \quad edf_1 = 0.3359375, \quad edf_2 = 0.5, \quad edf_3 = 0.6640525, \quad edf_4 = 0.75$

The mapping for SWB at 24.4kbps is:

$$E_M(27) = \hat{E}_{HF}(0)*edf_1 + \hat{E}_{HF}(1)*edf_3$$

$$E_M(28) = \hat{E}_{HF}(1)*edf_1 + \hat{E}_{HF}(2)*edf_3$$

$$E_M(29) = \hat{E}_{HF}(3)$$

$$E_M(30) = \hat{E}_{HF}(4)*edf_3 + \hat{E}_{HF}(5)*edf_1$$

$$E_M(31) = \hat{E}_{HF}(5)*edf_3 + \hat{E}_{HF}(6)*edf_1$$

$$E_M(32) = \hat{E}_{HF}(6)*edf_1 + \hat{E}_{HF}(7)*edf_3$$

$$E_M(33) = \hat{E}_{HF}(7)*edf_1 + \hat{E}_{HF}(8)*edf_3 \qquad (1228)$$

$$E_M(34) = \hat{E}_{HF}(8)*edf_1 + \hat{E}_{HF}(9)*edf_3$$

$$E_M(35) = \hat{E}_{HF}(9)*edf_1 + \hat{E}_{HF}(10)*edf_3$$

$$E_M(36) = \hat{E}_{HF}(10)*edf_0 + \hat{E}_{HF}(11)*edf_4$$

$$E_M(37) = \hat{E}_{HF}(12)$$

$$E_M(38) = \hat{E}_{HF}(13)$$

The mapping for SWB at 32kbps is:

$$E_M(30) = \hat{E}_{HF}(1)$$

$$E_M(31) = \hat{E}_{HF}(2)*edf_3 + \hat{E}_{HF}(3)*edf_1$$

$$E_M(32) = E_{HF}(3)*edf_3 + \hat{E}_{HF}(4)*edf_1$$

$$E_M(33) = \hat{E}_{HF}(4)*edf_1 + \hat{E}_{HF}(5)*edf_3$$

$$E_M(34) = \hat{E}_{HF}(5)*edf_1 + \hat{E}_{HF}(6)*edf_3 \qquad (1229)$$

$$E_M(35) = \hat{E}_{HF}(7)$$

$$E_M(36) = \hat{E}_{HF}(8)*edf_4 + \hat{E}_{HF}(9)*edf_0$$

$$E_M(37) = \hat{E}_{HF}(9)*edf_4 + \hat{E}_{HF}(10)*edf_0$$

$$E_M(38) = \hat{E}_{HF}(10)*edf_0 + \hat{E}_{HF}(11)*edf_4$$

The mapping for FB is:

$$E_M(39) = \hat{E}_{HF}(12)$$

$$E_M(40) = \hat{E}_{HF}(12)*edf_0 + \hat{E}_{HF}(13)*edf_4$$

$$E_M(41) = \hat{E}_{HF}(13)*edf_2 + \hat{E}_{HF}(14)*edf_2 \qquad (1230)$$

$$E_M(42) = \hat{E}_{HF}(14)$$

$$E_M(42) = \hat{E}_{HF}(14)$$

The resulting mapped envelope is then re-quantized using the same method as is used for the HQ high rate normal mode, as described in subclause 5.3.4.2.1.1.

The quantized high and low frequency envelopes are then combined, and an initial fractional bit allocation is carried out. The initial bit allocation for the SWB generic mode is described in 5.3.4.2.6.6. However, in the first step, equation (1235) is replaced with the following equation:

$$R_0(p,0) = \max\left\{ 0, \frac{L_M(p)}{3} * \left(\hat{I}_M(i) - \frac{\sum_{i=0}^{N_{bands\_LF}-1} L_M(i)*\hat{I}_M(i) - 3*TB}{\sum_{i=0}^{N_{bands}-1} L_M(i)} \right) \right\} \quad for\ p = 0,1,...,N_{bands}-1 \quad (1231)$$

The initial bit allocation used for FB generic mode is the same as the bit allocation used for the Normal mode as is described in 5.3.4.2.1.3.

The information obtained from the bit allocation indicates whether or not the envelope refinement is required for the current frame. If there are any high bands which have allocated bits, delta coding needs to be done to refine the high frequency envelope. In other words, if there are any important spectral components in the higher bands, the refinement is performed to provide a finer spectral envelope. If there are no bits allocated to the higher bands during the initial bit allocation, the envelope refinement is not required and the initial bit allocation is used. In this case the envelope requires no further modification so the following steps are not required and the fine structure quantization is applied to spectrum to quantize the coefficients as is described in subclause 5.3.4.2.1.3.

The quantized and de-quantized norms are calculated with the scalar quantizer as shown in subclause 5.3.4.2.1.1 by using the original spectrum, which it is defined to $\hat{I}_M(b)$. The scalar quantized and de-quantized norms are also calculated using the mapped norms with the vector quantization, which is defined to $\hat{I}_{M,map}(b)$.

Deltas are calculated at every band with allocated bits:

$$\Delta I_{M\_G}(b) = \begin{cases} \hat{I}_M(b) - \hat{I}_{M,map}(b) & \text{if } R(b) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for } b = N_{bands\_LF}, \ldots, N_{bands} - 1$$

(1232)

A possible bit BD_G for representing all $\Delta I_{M\_G}(b)$ spans 2,3,4 and 5 bits, and they are encoded as $BD_G - 2$ using 2 bits. So, the possible bit is calculated with $\Delta I_{M\_G}(b)$ and it then is adjusted to fit within the maximum number of bits (5). Any $\Delta I_{M,HQG}(b)$ which exceeds the maximum value is corrected to 31 or -32 depending on a sign of $\Delta I_{M,HQG}(b)$.

$$\hat{\Delta I}_{M\_G}(b) = \begin{cases} 31 & \text{if } \Delta I_{M\_G}(b) \geq 32 \quad \text{and } R(b) > 0 \\ -32 & \text{if } \Delta I_{M\_G}(b) < -32 \text{ and } R(b) > 0 \\ \Delta I_{M\_G}(b) & \text{otherwise} \end{cases}$$

$$\text{for } b = N_{bands\_LF}, \ldots, N_{bands} - 1$$

(1233)

The corrected $\hat{\Delta I}_{M\_G}(b)$ is transmitted with BD_G.

The value of $\hat{\Delta I}_{M\_G}(b)$ is then used to compute a final update to the envelope.

$$\hat{I}_{M\_G}(b) = \hat{I}_{M\_map}(b) + \hat{\Delta I}_{M\_Q}(b) \quad \text{for } b = N_{bands\_LF}, \ldots, N_{bands} - 1$$

(1234)

where $\hat{I}_{M\_G}(b)$ is the updated norm.

The fine structure quantization is then applied to the resulting spectrum to quantize the coefficients as is described in subclause 5.3.4.2.1.3.

Finally the initial bit allocation information is updated, based on the number of bits used for representing the deltas. This is done by reducing the bits allocated to some sub bands, to provide enough bits to code the deltas. If during the initial bit allocation a sub band was allocated more than 3 bits, its allocation is reduced by one bit until all the bits required for the deltas have been accounted for. This is shown in more detail in the pseudo code below.

```
while (Bits_needed_for_deltas > 0)
{
Current_band = number_of_bands-1
while(bits_needed_for_deltas >0 and Current_band >= 0)
{
if (Bits_allocated_to_current_band > 3)
{
Bits_allocated_to_current_band --
Bits_needed_for_deltas --
}
Current_band --
}
}
```

This procedure provides the final bit allocation.

5.3.4.2.6.6 Bit Allocation

In the Generic mode, a fractional bit allocation is used for the spectral quantizer bit allocation. This permits the allocation of bits with three bits fractional parts. Initially bits for each band are estimated by:

$$R_0(p,0) = \max\left\{ 0, \frac{L_M(p)}{3} * \left(\hat{I}_M(i) - \frac{\sum\limits_{i=0}^{N_{bands}-1} L_M(i) * \hat{I}_M(i) - 3*TB}{\sum\limits_{i=0}^{N_{bands}-1} L_M(i)} \right) \right\} \quad for \ p = 0,1,...,N_{bands}-1 \ (1235)$$

where *TB* is a total bit budget.

The fully allocated bits are calculated as a starting point and the first-stage iterations are done to re-distribute the allocated bits to the bands with non-zero bits until the number of fully allocated bits is equal to the total bit budget.

$$R_0(p,k) = \max\left\{ 0, R_0(p,k-1) - L_M(p) * \frac{\sum\limits_{i=0}^{N_{bands}-1} R_0(p,k-1) - TB}{NSL_0(k-1)} \right\} \quad (1236)$$

where $NSL_0(k-1)$ is the number of spectral lines in all bands with allocated bits after *k* iterations.

If too few bits are allocated, this can cause a quality degradation due to the reduced SNR. To avoid this problem a minimum bit limitation is applied to the allocated bits. The first minimums for the bits consist of constant values depending on the band index and bit-rate.

$$LNB(p) = \begin{cases} 3 & for \ p = 0,...,15 \\ 4 & for \ p = 16,...,23 \\ 5 & for \ p = 24,...,N_{bands}-1 \end{cases} \quad (1237)$$

In the second-stage iterations, the re-distribution of bits is done again to allocate bits to the bands with more than $L_M(p)$ bits. The value $L_M(p)$ bits indicates the number of bits that corresponds to 1 bit/sample in band *p* and is the second minimum required bits for each band. Initially, the allocated bits are calculated based on the result of the first-stage iteration and the first and second mininum required bits for each band.

$$R_1(p,0) = \begin{cases} 0 & if \ R(p) < bs + LNB(p) \\ L_M(p) & if \ bs + LNB(p) \leq R(p) <= L_M(p) \end{cases} \quad (1238)$$
$$for \ p = 0,...,N_{bands}-1$$

where $R(p)$ is the allocated bits after the first-stage iterations and *bs* is 2 at 24.4kbps and 3 at 32kbps.

The *TB* is updated by subtracting the number of bits in bands with $L_M(p)$ bits , and the band index *p* is updated to p' which indicates the band indices with higher bits than $L_M(p)$ bits. N_{bands} is updated to N'_{bands} which is the number of bands for p'. The second-stage iterations are then done until the updated *TB* (TB') is equal to the number of bits in bands with more than $L_M(p')$ bits.

$$R_1(p',k) = \max\left\{ L_M(p'), R_1(p',k-1) - L_M(p') * \frac{\sum\limits_{i=0}^{N_{bands}-1} R_1(p',k-1) - TB'}{NSL_1(k-1)} \right\} \quad (1239)$$

where $NSL_1(k-1)$ is the number of spectral lines in all bands with more than $L_M(p')$ bits after k iterations.

During the second-stage iterations, if there are no bands with more than $L_M(p')$ bits, the bits in bands with non-zero allocated bits from the highest bands are set to zero until TB' is equal to zero. Then the iterations are terminated.

Then, a final re-distribution of over-allocated and under-allocated bits is performed and finally, the fractional parts of bit allocation are adjusted to have three bits.

$$R(p) = \lfloor R(p)*8 \rfloor / 8 \qquad for\ p = 0,...,N_{bands} - 1 \qquad (1240)$$

5.3.4.2.7 Pyramid Vector Quantization (PVQ) and indexing

The PVQ is a lattice quantizer, with complexity growing linearly with the vector dimension. The PVQ quantizes a N dimensional vector by allocating K signed pulses to match the shape of that vector. In this way the PVQ codebook is defined as a combination of K signed pulses in a N dimensional space (with pulses at the same position having the same sign). The maximum size of PVQ codebooks is set to 32 bits, and if allocated bits are more than 32, the vector is split into sub-vectors, bits re-distributed, and a gain parameter is quantized to represent relative energy between these sub-vectors. Largest allowed target vector is of dimension 64.

5.3.4.2.7.2 PVQ split methodology

When the bits $R(b)$ assigned for the band b are above a pre-determined threshold, as described in subclause 5.3.4.2.7.2.1, an algorithm for band splitting is activated. First the input vector is split into uniform (or close to uniform) segments in a non-recursive way. Then angles α, which represent the ratio between energies E_L and E_R of a left and a right level segment, are calculated recursively. At each iteration, the level segments consist of one or several of the pre-determined segments from the initial split of the input vector. The angles are calculated from the top level (full size of the band to be quantized), and continuing towards the levels of shorter sub-vectors, i.e. shorter level segments.

The angle is determined from the energy ratio between one left and one right level segment. In case of an even number of splits, the left and right level segments will consist of an equal number of segments. In case of odd number of splits, the angle calculated during the first iteration of the recursion will be with the right segment having a larger number of segments than the left level segment, and the recursion will require more steps for the right level segment.

For example if the bit budget $R(b)$ for the band size $L_M(b)$ require split in 3 segments (here assuming that a split into equal sized segments is possible), the angle calculation and the bit distribution is done in the following way:

Step 0: the angle and the shape bits are $\left\{ \alpha^0, \quad R^0 \right\}$, the left level segment is of length $\frac{1}{3}L_M(b)$, while the right level segment is of length $\frac{2}{3}L_M(b)$, and bits for the two level segments are R_L^0 and R_R^0.

Step 1: the right level segment from the previous step is split in a left and a right level segment, both with the length $\frac{1}{3}L_M(b)$, the angle and the shape bits to be distributed are $\left\{ \alpha^1, \quad R_R^0 \right\}$, the two level segments (here equal to the second and third segment of the initial split of the input vector) are allocated the bits R_{RL}^1 and R_{RR}^1.

The angles are used to distribute bits recursively to the already determined segments. At each iteration, the bits R_L and R_R for a left and a right level segment are derived from the available bits for shape coding of these segments R, lengths of the level segments L_L and L_R, and the angle α:

$$R_L = \frac{R + \left(f_R - f_L - \log_2(\tan(\alpha)) \right) L_R}{1 + L_R / L_L} = \frac{R + \left(f_R - f_L - \frac{1}{2} \left(\log_2(E_R) - \log_2(E_L) \right) \right) L_R}{1 + L_R / L_L} \qquad (1241)$$

$$R_R = R - R_L \qquad (1242)$$

where f_L and f_R are compensation factors for differences in segment lengths within the level segments, defined as

$$f_L = \sum_j - \frac{1}{L_L}\left(\frac{R}{N_p} - G_2\left(\min\left(L_L^j\right), G_1\left(L_L^j, \frac{R}{N_p}\right)\right)\right), \quad L_L = \sum_j L_L^j \tag{1242a}$$

$$f_R = \sum_j - \frac{1}{L_R}\left(\frac{R}{N_p} - G_2\left(\min\left(L_R^j\right), G_1\left(L_R^j, \frac{R}{N_p}\right)\right)\right), \quad L_R = \sum_j L_R^j \tag{1242b}$$

where the function $G_1(\cdot)$ gives the number of unit pulses that giving the segment lengths L_L^j and L_R^j of the segments j within the left and the right level segment respectively, can be represented using R/N_p bits. N_p is the total number of pre-determined segments/splits within the left and right level segments. The function $G_2(\cdot)$ gives the number of bits used to represent the by the function $G_1(\cdot)$ determined number of unit pulses for the minimum dimension among the pre-determined segments.

The angle, which captures the relation between the energies of the left and right segment at certain split level, is defined as:

$$\alpha = \operatorname{atan}\left(\sqrt{\frac{E_R}{E_L}}\right) \tag{1243}$$

These angles are calculated recursively, starting with the top level, and continuing towards the levels of shorter sub-vectors. The angles are quantized by a range coder with asymmetric triangular PDF.

5.3.4.2.7.2.1 Band splitting analysis

The number of segments (parts) in the split PVQ vector N_p is determined in two steps. First, an initial number of segments N_{p_init} is computed as

$$N_{p\_init} = \left\lceil R_{PVQ}(b)/\left(R_{PVQpart\_max} + R_{splitcost}\right)\right\rceil \tag{1244}$$

where $R_{PVQpart\_max} = 32$ is the maximum bitrate for each PVQ vector segment. If $N_{p\_init} < 10$ and the band bit rate is high, an additional split is considered. Using N_{p_init} segments, the $\log_2$ energies of each PVQ target vector segment is computed.

$$E_{PVQsplit}(p) = \log_2\left(\sum_{k \in part(p)} x(k)^2\right), \quad p = 0,..,N_{p\_init} - 1 \tag{1245}$$

If the maximum absolute deviation from the mean $\log_2$ energy $\Delta E_{PVQsplit\_max}$,

$$\Delta E_{PVQsplit\_max} = \max_p \left|E_{PVQsplit}(p) - \frac{1}{N_{p\_init}}\sum_p E_{PVQsplit}(p)\right| \tag{1246}$$

is larger than the difference between the maximum number of bits for each segment $R_{PVQpart\_max}$ and the average bit rate for each segment, an additional split will be added. That is,

$$N_p = \begin{cases} N_{p\_init} + 1, & \Delta E_{PVQsplit\_max} > \left(R_{PVQpart\_max} - \left\lfloor R_{PVQ}(b)/N_{p\_init}\right\rfloor\right) \\ N_{p\_init}, & otherwise \end{cases} \tag{1247}$$

Depending on if the conditions for the additional split are met, the split increment flag $pvq\_split\_inc = 1$ or $pvq\_split\_inc = 0$ is signalled in the bitstream using 1 bit.

5.3.4.2.7.3 PVQ sub-vector shape search and shape normalization

5.3.4.2.7.3.1 PVQ-search introduction

The goal of the $PVQ(N,K)$ search procedure is to find the vector $\mathbf{x_q}$, which is defined as:

$$\mathbf{x_q} = g_{sub} \frac{\mathbf{y}}{\sqrt{\mathbf{y^T y}}} \qquad (1248)$$

where $\mathbf{y} = \mathbf{y}_{N,K}$ is a point on the surface of an -dimensional hyper-pyramid and the L1 norm of $\mathbf{y}_{N,K}$ is K. I.e. $\mathbf{y}_{N,K}$ is the selected integer shape code vector of size *N* according to:

$$\mathbf{y}_{N,K} = \left\{ e : \sum_{i=0}^{N-1} |e_i| = K \right\} \qquad (1249)$$

i.e. $\mathbf{x_q}$ is the unit energy normalized integer sub vector $\mathbf{y}_{N,K}$ scaled with a sub vector splitting gain g_{sub}, (if the band is not split in sub vectors the gain g_{sub} is 1).

The best $\mathbf{y}$ vector is the one minimizing the mean squared shape error between the target vector $\mathbf{x}$ and the scaled normalised quantized output vector $\mathbf{x_q}$. The target vector $\mathbf{x}$ can be the time domain or in the frequency domain. Finding the best $\mathbf{y}$ is achieved by minimizing the following search distortion:

$$d_{PVQ} = -\mathbf{x}^T \mathbf{x_q} = -g_{sub} \frac{\left(\mathbf{x}^T \mathbf{y}\right)}{\sqrt{\mathbf{y^T y}}} \qquad (1250)$$

By squaring the numerator and denominator and eliminating the predetermined constant gain scale factor g_{sub}, we may maximize the quotient Q_{PVQ}:

$$Q_{PVQ} = \frac{\left(\mathbf{x}^T \mathbf{y}\right)^2}{\mathbf{y^T y}} = \frac{\left(corr_{xy}\right)^2}{energy_y} \qquad (1251)$$

The L1-norm structured PVQ-quantizer, $PVQ(N,K)$ allows for several search optimisations, where the primary optimization is to move the target to the all positive "quadrant" in N-dimensional space and the second optimization is to use an L1-norm projection as a starting approximation for $\mathbf{y}$. A third optimization is to iteratively update the Q_{PVQ} quotient, instead of re-computing equation (1251) over the whole vector space N for every candidate change to the vector $\mathbf{y}$ in pursuit of reaching the L1-norm K, which is required for the subsequent PVQ-indexing step.

In the search of the optimal PVQ vector shape $\mathbf{y}$ with L1-norm K, iterative updates of the Q_{PVQ} variables are made in the all positive "quadrant" in N-dimensional space according to:

$$corr_{xy}(k,n) = corr_{xy}(k-1) + 1 \cdot x(n) \qquad (1252)$$

$$energy_y(k,n) = energy_y(k-1) + 2 \cdot 1^2 \cdot y(k-1,n) + 1^2 \qquad (1253)$$

where $corr_{xy}(k-1)$ signifies the correlation achieved so far by placing the previous $k-1$ unit pulses, and $energy_y(k)$ signifies the accumulated energy achieved so far by placing the previous $k-1$ unit pulses, and $\mathbf{y}(k-1,n)$ signifies the amplitude of $\mathbf{y}$ at position n from the previous placement of $k-1$ unit pulses. To further speed up the in-loop iterative processing the energy term $energy_y(k)$ is scaled down by 2, thus saving one multiplication in the inner-loop.

$$enloop_y(k,n) = energy_y(k,n)/2, \Rightarrow$$
$$enloop_y(k,n) = enloop_y(k-1) + y(k-1,n) + 0.5 \qquad (1254)$$

$$Q_{PVQ}(k,n) = \frac{corr_{xy}(k,n)^2}{enloop_y(k,n)}$$ (1255)

The best position n_{best} for the k'th unit pulse, is iteratively updated by iterating n over $0...N$.

$$n_{best} = n \quad ,if \quad Q_{PVQ}(k,n) > Q_{PVQ}(k,n_{best})$$ (1256)

To avoid divisions the Q_{PVQ} maximization update decision is performed using a cross-multiplication of the saved best squared correlation numerator *bestCorrSq* and the saved best energy denominator *bestEn* so far.

$$\left. \begin{array}{l} n_{best} = n \\ bestCorrSq = corr_{xy}(k,n)^2 \\ bestEn = enloop_y(k,n) \end{array} \right\} ,if \quad corr_{xy}(k,n)^2 \cdot bestEn > bestCorrSq \cdot enloop_y(k,n)$$ (1257)

The iterative maximization of $Q_{PVQ}(k,n)$ may start from a zero number of placed unit pulses or from an adaptive lower cost pre-placement number of unit pulses, based on an integer projection to a point below the K'th-pyramid's surface, with a guaranteed undershoot of unit pulses in the target L1 norm K.

Due to the structured nature of the $\mathbf{y}_{N,K}$ PVQ integer sub vector, where all possible sign combinations are allowed and it is possible to encode all sign combinations, as long as the resulting vector adheres to the L1 norm of K unit pulses, the search is performed in the all positive first *"quadrant"*. Further to achieve as a high accuracy as possible for a limited precision implementation the maximum absolute value $xabs_{max}$ of the input signal $\mathbf{x}$ is pre-analysed for future use in the setup of the inner search loop precision.

$$xabs(n) = |\mathbf{x}(n)|, for \quad n = 0,...,N-1$$ (1258)

$$xabs_{max} = \max(xabs_0,...xabs_{N-1})$$ (1259)

In case the input vector is an all zero vector or the sub-vector gain is very low, the PVQ-search is bypassed, and a valid PVQ-vector $\mathbf{y}_{N,K}$ is deterministically created by assigning half of the K unit pulses to the first position ($y[0] = \left\lfloor \dfrac{K}{2} \right\rfloor$), and the remaining unit pulses to the last position ($y[N-1] = y[N-1] + (K - y[0])$). (With this approach PVQ-search complexity is reduced and the indexing complexity is spread between encoder indexing and decoder de-indexing.)

5.3.4.2.7.3.4 PVQ pre-search projection

If the pulse density ratio K/N is larger than 0.5 unit pulses per coefficient and K is larger than *1*, a projection to the $K-1$ sub pyramid is made and used as a starting point for $\mathbf{y}$, on the other hand if the pulse density is less than 0.5 or K is *1*, the iterative PVQ-search will start off from *zero* pre-placed unit pulses. The projection is performed as:

$$proj_{fac} = \frac{K-1}{\displaystyle\sum_{n=0}^{n=N-1} \mathbf{xabs}(n)}$$ (1260)

$$\mathbf{y}(n) = \mathbf{y}_{start}(n) = \left\lfloor \mathbf{xabs}(n) \cdot proj_{fac} \right\rfloor for \quad n = 0,...,N-1$$ (1261)

If no projection is made the starting point is an all zeroed $\mathbf{y}$ vector. In preparation for the fine search to reach the K'th-pyramid's surface the accumulated number of unit pulses $pulse_{tot}$, the accumulated correlation $corr_{xy}(pulse_{tot})$ and the accumulated energy $energy_y(pulse_{tot})$ for the starting point $\mathbf{y}_{start} = \mathbf{y}$ is computed as:

$$pulse_{tot} = \sum_{n=0}^{n=N-1} \mathbf{y}(n)$$ (1262)

$$corr_{xy}(pulse_{tot}) = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \cdot \mathbf{xabs}(n) \qquad (1263)$$

$$energy_y(pulse_{tot}) = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \cdot \mathbf{y}(n) = \|\mathbf{y}\|_{L2} \qquad (1264)$$

$$enloop_y(pulse_{tot}) = energy_y(pulse_{tot})/2 \qquad (1265)$$

5.3.4.2.7.3.5 PVQ fine search

The final integer shape vector $\mathbf{y}_{N,K}$ must adhere to the L1 norm of K pulses. The fine search starts from a lower point in the pyramid and iteratively finds its way to the surface of the N-dimensional K'th-hyperpyramid. The K-value in the fine search ranges from 1 pulse to 512 unit pulses, to keep the complexity of the search at a reasonable level, the search is split into two main branches, one branch is used when the in-loop energy representation of $\mathbf{y}$ will stay within a signed 16 bit word, and another branch is used if the in-loop energy may or may not exceed the dynamic range of a 16 bit word.

When the final K is lower than or equal to 127 unit pulses, the dynamics of the 1 bit upshifted $enloop_y(K)$ will always stay within 15 bits, allowing efficient use of a signed 16 bit word for representing every $enloop_y(k)$ within all the fine pulse search inner loop iterations up to $k = K$.

In preparation for the next unit pulse addition, the near optimal maximum possible upshift of the next loop's accumulated in-loop correlation value in a signed 32 bit word is pre-analysed using the previously calculated maximum absolute input value as:

$$corr_{upshift} = 30 - \left\lfloor \log 2 \big(corr_{xy}(pulse_{tot}) + 2(1 \cdot xabs_{max}) \big) \right\rfloor \qquad (1266)$$

To make the critical equation (1257) update as efficient as possible, the $corr_{xy}(k,n)^2$ numerator is represented by a 16 bit signed word, by the following approach:

$$corr_{xy16}(k,n)^2 = \left(\mathrm{Round}_{16}\Big(\big(corr_{xy}(pulse_{tot}) + 2 \cdot (1 \cdot \mathbf{xabs}(n)) \big) \cdot 2^{corr_{upshift}} \Big) \right)^2 \qquad (1267)$$

$$\left. \begin{array}{l} n_{best} = n \\ bestCorrSq_{16} = corr_{xy16}(k,n)^2 \\ bestEn_{16} = enloop_y(k,n) \end{array} \right\}, if \quad corr_{xy16}(k,n)^2 \cdot bestEn_{16} > bestCorrSq_{16} \cdot enloop_y(k,n) \qquad (1268)$$

where the function "Round_{16}" extracts the top 16 bits of a signed 32 bit variable with rounding, this near optimal upshift and the use of 16 bit representation of the squared correlation ***bestCorrSq₁₆*** enables a very fast inner-loop search for performing the equation (1268) test and variable updates.

The location of the next unit pulse is now determined by iterating over the $n = 0, \ldots, N-1$ possible positions, while employing equations (1267), (1253) and (1268).

When the best position n_{best} of the unit pulse has been determined, the accumulated correlation $corr_{xy}(k)$ the accumulated inloop energy $enloop_y(k)$ and the number of accumulated unit pulses $pulse_{tot}$ are updated. If there are further unit pulses to add $(pulse_{tot} < K)$, a new inner-loop is started with a new near optimal $corr_{upshift}$ analysis (equation (1266)) for the next unit pulse.

When the final K is higher than 127 unit pulses, the dynamics of the 1 bit upshifted $enloop_y(K)$ may exceed 15 bits, the fine search will adaptively choose between 16 bit representation and 32 bit representation of the pair $\left\{ corr_{xy}(k,n)^2, enloop_y(k,n) \right\}$. The fine search will keep track of the maximum pulse amplitude $maxamp_y$ in $\mathbf{y}$ achieved so far, this information is used in a pre-analysis step before entering the optimized inner dimension loop,

to determine the precision to use for the inner unit pulse addition loop. If the pre-analysis indicates that more than a signed 16 bit word is needed to represent the in-loop energy without losing any energy information, a high precision unit pulse addition loop is employed, where both the saved best squared correlation term and the saved best accumulated energy term are represented by 32 bit words.

$$en_{margin} = 30 - \left\lfloor \log 2 \left(1 + energy_y(pulse_{tot}) + 2 \cdot maxamp_y\right)\right\rfloor \quad (1269)$$

$$highprecision_{active} = FALSE, \quad if\ (en_{margin} \geq 16)$$
$$highprecision_{active} = TRUE, \quad if\ (en_{margin} < 16) \quad (1270)$$

If $highprecision_{active}$ is *FALSE* the lower precision inner search loop in equations (1267), (1253) and (1268) is employed, on the other hand if $highprecision_{active}$ is *TRUE*, the location of the next unit pulse is determined by iterating over the $n = 0, \ldots, N-1$ possible positions, using equations (1271), (1253) and (1272).

$$corr_{xy32}(k,n)^2 = \left(\cdot \left(corr_{xy}(pulse_{tot}) + 2 \cdot \mathbf{xabs}(n)\right) \cdot 2^{corr_{upshift}}\right)^2 \quad (1271)$$

$$\left.\begin{array}{l} n_{best} = n \\ bestCorrSq_{32} = corr_{xy32}(k,n)^2 \\ bestEn_{32} = enloop_y(k,n) \end{array}\right\}, if \quad corr_{xy32}(k,n)^2 \cdot bestEn_{32} > bestCorrSq_{32} \cdot enloop_y(k,n) \quad (1272)$$

When the best position n_{best} of the unit pulse has been determined, the accumulated correlation $corr_{xy}(k,n)$, the accumulated inloop energy $enloop_y(k)$ and the number of accumulated unit pulses $pulse_{tot}$ are updated. Further the maximum amplitude $maxamp_y$ in the best integer vector so far, is kept up to date for the next unit pulse addition loop.

$$maxamp_y = \max\left(maxamp_y, y[n_{best}]\right) \quad (1273)$$

If there are further unit pulses to add $(pulse_{tot} < K)$, a new inner-loop is started with a new near optimal $corr_{upshift}$ analysis equation (1266) and a new energy precision analysis equations (1269) and (1270) and then commencing the next unit pulse loop equations (1271), (1253) and (1272)).

The effect of the in-loop accumulated energy based inner loop precision selection is that target sub vectors that have a high peakiness, or have very fine shape granularity (final K is high) will be using the higher precision loop, while non-peaky or lower shape granularity sub vectors will more often use the lower precision loop.

5.3.4.2.7.3.6 PVQ sub-vector finalization and normalization

After shape search each non-zero PVQ-sub-vector element is assigned its proper sign and the vector is L2-normalized to unit energy. Additionally if the band was split, it is further scaled with the previously determined sub-vector gain g_{sub}.

$$if\left(\mathbf{y}(n) > 0\right) \bigcap \left(\mathbf{x}(n) < 0\right) \Rightarrow \mathbf{y}(n) = -\mathbf{y}(n), for \quad n = 0, \ldots, N-1 \quad (1274)$$

$$norm_{gain} = \frac{g_{sub}}{\sqrt{\mathbf{y^T y}}} \quad (1275)$$

$$\mathbf{x_q}(n) = norm_{gain} \cdot \mathbf{y}(n), for \quad n = 0, \ldots, N-1 \quad (1276)$$

5.3.4.2.7.4 PVQ short codeword indexing

The PVQ short codeword indexing assigns a unique index to any possible vector in $PVQ(N,K)$. The incoming $PVQ(N,K)$ vector $\mathbf{y}$, which represents a point on the K'th hyper-pyramid in N-dimensional space is indexed into two short codeword(s) using a leading sign sections Modular PVQ recursive indexing method denoted $MPVQ(N,K)$. After the MPVQ indexing procedure, the integer codewords and the size of each codeword are sent to a Range encoder, an arithmetic encoder, which in turn provides a stream of arithmetically encoded bits to the to the bit stream.

5.3.4.2.7.4.1 MPVQ modular leading sign recursion definition

The first codeword represents a leading sign, which is the sign of the first non-zero position in **y** , the second codeword is a recursive representation of the amplitudes (including zeroes) and leading signs of the remaining vector **z** after the first occurring non-zero position sign in **y** has been extracted. The modular leading sign extraction is performed recursively until all amplitudes (including zeroes) and signs have been consumed.

The number of possible combinations of **y** is denoted $N_{PVQ}(N,K)$. The number of possible combinations of **z** , an $MPVQ(N,K)$ vector with K unit pulses in dimension N with an L1-norm of K and an initial positive sign is denoted $N_{MPVQ}(N,K)$. The elimination of the initial leading sign from **y** yields the number of possible entries in **z** as:

$$N_{MPVQ}(N,K) = \frac{N_{PVQ}(N,K)}{2} \tag{1277}$$

The MPVQ-index for vector **z** is based on the recursive $MPVQ(N,K)$ scheme where we:

Define $U(n,k)$ as the number of integer vectors of dimension n and L1-norm of k , that does not have a leading zero, and does not have the leading value *k***,** has a leading positive value, and has a positive leading sign, (where the leading sign is the first sign encountered after the current value in the direction of the recursion.

Define $A(n,k)$ as the number of integer vectors of dimension n and L1-norm of k , that has a positive leading value and that does not have a leading zero.

See table 148 for a structured view of $U(n,k)$ and its relation to the total number of vectors in the $MPVQ(n,k)$ structure with $N_{MPVQ}(n,k)$ vectors.

Table 148: MPVQ recursion overview: the three subsections for the $N_{MPVQ}(n,k)$ leading sign based recursion

Subsection (initial $z[0]$ value for the subsection)	Number of entries for each subsection
k	1
	All unit pulses consumed. Remaining $z[1,\dots,n-1]$ values all zeroes, recursion may stop
$k-1,\dots,1$ (non-zero with a positive or negative next leading sign)	$2 \times U(n,k)$
	First encountered non-zero position requires 1 bit of next leading sign information, stored as the LSB in this even sized $2 \times U(n,k)$ sub-section. The recursion consumes the $z[0]$ amplitude value and moves on to $z[1]$
0	$N_{MPVQ}(n-1,k)$
	Recursion moves on to $z[1]$ without any additional next leading sign information

The recursive definition of the total number of entries in **z** becomes:

$$N_{MPVQ}(n,k) = 1 + 2 \cdot U(n,k) + N_{MPVQ}(n-1,k) \tag{1278}$$

By applying Fischer's $PVQ(n,k)$ original recursion for $N_{MPVQ}(n,k)$ together with equation (1277) and (1278) , we establish that:

$$N_{MPVQ}(n,k) = 1 + U(n,k) + U(n,k+1) \tag{1279}$$

, from the $A(n,k)$ definition (and table 148) one can find that the relation between $A(n,k)$ and $U(n,k)$ is:

$$A(n,k) = 1 + 2 \cdot U(n,k) \tag{1280}$$

where (the "1" comes from the single initial-"k" valued vector, and the factor "2" is due to positive and negative sign possibilities of the coming next leading sign.

On the encoder side recursive relations are used for the forward update of the MPVQ indexing offsets $A(n,k)$ and $U(n,k)$ as the dimension increases up to the size of the vector to be indexed:

$$U(n,k+1) = 1 + U(n-1,k) + U(n,k) + U(n-1,k+1) \tag{1281}$$

$$U(n,k+1) = 1 + \left\lfloor \frac{A(n-1,k)}{2} \right\rfloor + \left\lfloor \frac{A(n,k)}{2} \right\rfloor + U(n-1,k+1) \tag{1282}$$

where the low dynamic range calculation equation (1282) is used for the last column $K+1$ forward update of the MPVQ offset matrix, a column which is needed for the calculation of the exact $MPVQ(N,K)$ size. An $A(n,k)$ based recursion is used for the offset matrix columns $0...K$, as these columns are known in advance to have a low dynamic range. To keep the dynamic range within a signed 32 bit word, the final $MPVQ(N,K)$ size calculation is performed as:

$$N_{MPVQ}(N,K) = 1 + \left\lfloor \frac{A(N,K)}{2} \right\rfloor + U(N,K+1) \tag{1283}$$

5.3.4.2.7.4.2 Detailed MPVQ indexing approach

The maximum index range of the found PVQ-vector $\mathbf{y}$'s is $[0.. 2^{32}-1]$ and the total resulting index would require an unsigned 32 bit word, however with the introduction of the modular leading sign approach the maximum index range is reduced to $[0.. 2^{31}-1]$, which enables fast implementation using signed 32 bit word arithmetic.

The MPVQ-indexing scheme on the encoder side is run in the order of position $N-1$ to position 0, (on the decoder side the de-indexing will start from position 0 and end at position $N-1$). The MPVQ indexing loop is carried out according to figure 84, where n is the row number of the MPVQ offset matrix, pos is the pointer into the samples/coefficients of the PVQ-vector , ">>1" denotes a 1 bit right shift and further the function "UpdateOffsetsFwd" iteratively updates the required MPVQ-offsets for the next larger dimension using combinations of equations (1281), (1280) and (1282).
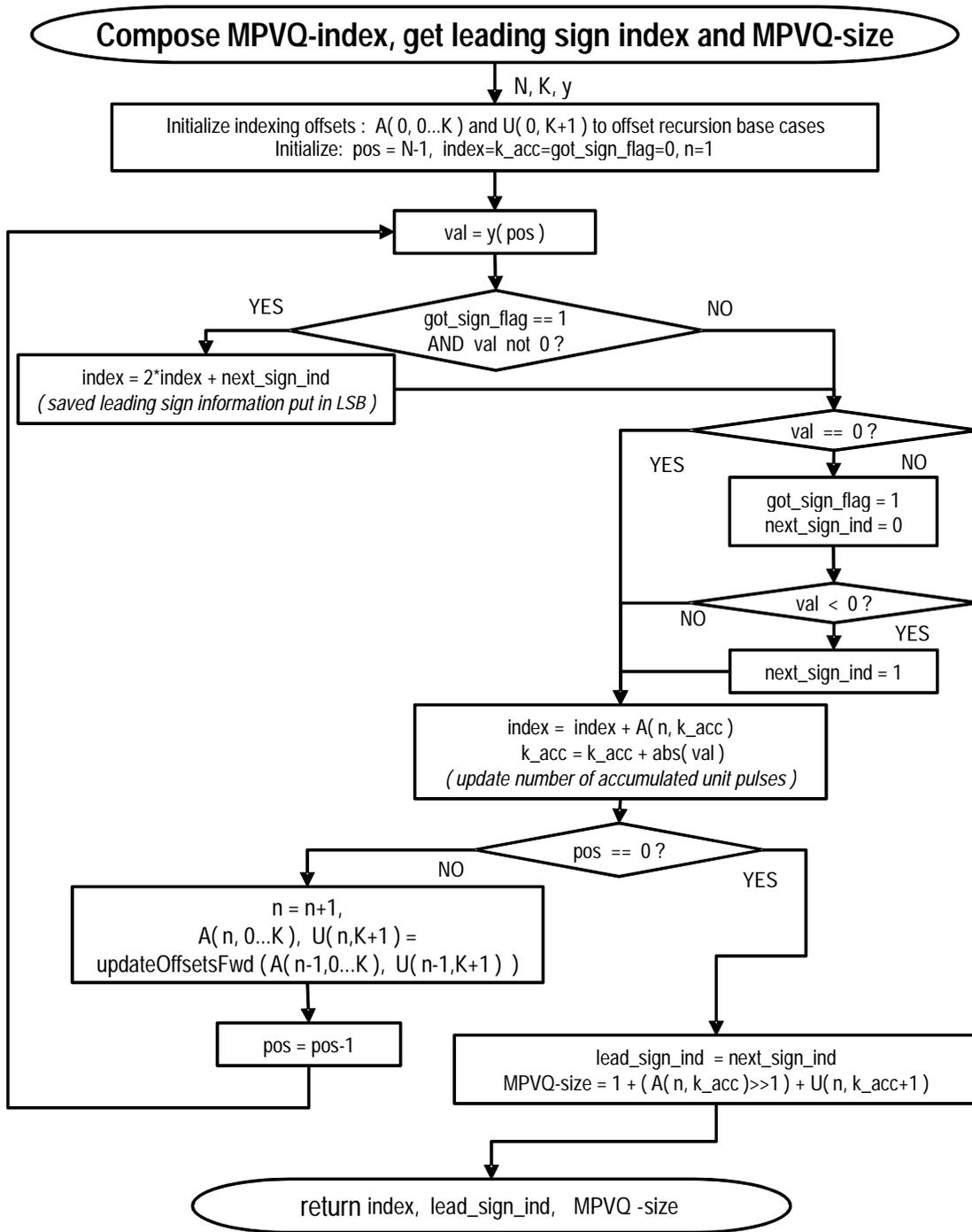
Figure 84: Detailed MPVQ-indexing, leading sign index extraction and size calculation

To speed up the indexing when the number of unit pulses K is high (and the dimension N is low), direct functions are used to calculate both the MPVQ-offsets and the MPVQ-size.

5.3.4.2.7.5   High Dynamic Range Arithmetic Encoding

The Split PVQ scheme relies on a Range encoder to encode symbols with higher granularity than bits. A high dynamic range implementation, where of up to 24 bits resolution may be used for the CDF is used to encode short codewords in three distinct ways:

• Direct bit encoding

- 1/8 bit resolution uniform PDF encoding

- Asymmetric triangular PDF encoding of band split energy angles, using tailor made CDFs.

Further special range encoder functions are used to determine the remaining bit budget in the frame and band quantization loops, and used to correct the bit budget for coming frames or coming bands.

5.4 Switching of Coding Modes

5.4.1 General description

As described in subclause 5.2 and 5.3, the EVS codec supports a CELP coding mode as well as a MDCT coding mode. The transitions between both within the same bit rate and audio bandwidth are described in 5.4.2 and 5.4.3.

The CELP or LP-based coding mode can operate on different sample rates depending on the frame configuration. The procedure how to handle sample rate changes during the encoding process is described in 5.4.4.

The switching between primary and AMR-WB IO modes is described in 5.4.5.

The handling of transitions in the context bit rate switches is described in 5.4.6.

5.4.2 MDCT coding mode to CELP coding mode

When a CELP encoded frame is preceded by a MDCT based encoded frame, the memories of the CELP encoded frame have to be updated before starting the encoding of the CELP frame. These memories include:

- Adaptive codebook memory

- LPC synthesis filter memory

- Weighting filter denominator memory (used to compute the target signal)

- The factor β_1 of the tilt part of the innovative codebook pre-filter

- De-emphasis filter memory

- MA/AR prediction memories used in end-frame LSF quantization

- Previous quantized end-frame LSP (for quantized LPC interpolation)

- Previous quantized end-frame LSF (for mid-frame LSF quantization)

The CELP memories update is performed depending on the bitrate and the previous encoding mode (either MDCT based TCX or HQ MDCT). In general, three different MDCT to CELP (MC1-3) transition methods are supported. The following table 149 lists the different cases depending on MDCT mode and bit rate.

Table 149: MDCT to CELP transition modes

Switching from	Switching to	Bitrate (kbps)	Transition mode
HQ MDCT	CELP	7.2	MC1
HQ MDCT	CELP	8	MC1
TCX	CELP	9.6	MC2
TCX or HQ MDCT	CELP	13.2	MC1
TCX	CELP	16.4	MC2
HQ MDCT	CELP	16.4	MC3
TCX	CELP	24.4	MC2
HQ MDCT	CELP	24.4	MC3
TCX or HQ MDCT	CELP	32	MC1
HQ MDCT	CELP	64	MC1

In following subclauses, the MDCT to CELP transitions are described in detail. Note that this description only considers switching cases within the same bit rate.

5.4.2.1   MDCT to CELP transition 1 (MC1)

MC1 is used when the previous frame was coded with HQ MDCT and the current frame is coded with CELP. In this case, the CELP state variables are reset in the current frame to predetermined (fixed) values. In particular the following memories are reset to 0 in the CELP encoder:

- Resampling memories of the CELP synthesis signal

- Pre-emphasis and de-emphasis memories

- LPC synthesis memories

- Past excitation (adaptive codebook memory)

The old LPC coefficients and associated representations (LSP, LSF) and CELP gain quantization memories are reset to predetermined (fixed) values. Since the past excitation is not available, the CELP coder in the current frame is forced to operate in Transition coding (TC) , i.e. without any adaptive codebook. The LPC coefficients from the previous frame are not available, therefore only one set of LPC coefficients corresponding to the end of frame are coded and used for all subframes of the current frame.

5.4.2.2   MDCT to CELP transition 2 (MC2)

MC2 is designed for CELP transitions coming from the MDCT based TCX mode. The TCX shares the same LPC analysis and quantization as in CELP, as described in subclause 5.3.3.2.1. The MA/AR/LSP/LSF memories are consequently updated during MDCT based TCX encoding, as it is done in CELP encoding. The only exception is at 9.6kbps, where the weighted LPC are quantized (instead of the unweighted LPC as in CELP) as described in subclause 5.3.3.2.1.1.1. In that case, the MA/AR/LSP/LSF memories are re-computed in the unweighted domain as described in subclause 5.3.3.2.1.1.2.

Moreover, MDCT based TCX includes an internal decoder which generates a decoded time-domain signal at the CELP sampling rate as described in subclause 5.3.3.2.12.1. This signal and the quantized LPC are then used to update CELP memories as described in 5.3.3.2.12.2, including the adaptive codebook memory, the LPC synthesis filter memory, the weighting filter denominator memory and the de-emphasis filter memory.

5.4.2.3   MDCT to CELP transition 3 (MC3)

Similarly to MC1, the CELP state variables are reset to predetermined values (in general 0), with the following exceptions::

- The factor β_1 is set to 0.3.

- The LSF quantization is run in safety-net mode, such that no prediction is used and the MA/AR memories are not used.

- The previous quantized end-frame LSP/LSF and the quantized mid-frame LSP/LSF are set to the current quantized end-frame LSP/LSF.

5.4.3   CELP coding mode to MDCT coding mode

When a MDCT encoded frame is preceded by a CELP encoded frame, a beginning portion of the MDCT encoded frame cannot be reconstructed properly due to the aliasing introduced by the missing previous MDCT encoded frame. To solve this problem, two approaches are used depending on the MDCT based coding mode (either MDCT based TCX or HQ MDCT). These approaches are described in detail in the following subclauses.

5.4.3.1   CELP coding mode to MDCT based TCX coding mode

When the previous frame is CELP and the current frame is MDCT based TCX, the MDCT length is increased, the left folding point is moved towards the past and the left overlap length is reduced such that the current MDCT based TCX can reconstruct the whole 20ms frame, without the need for the previous (and missing in this case) MDCT based frame. This is illustrated in the figure below.
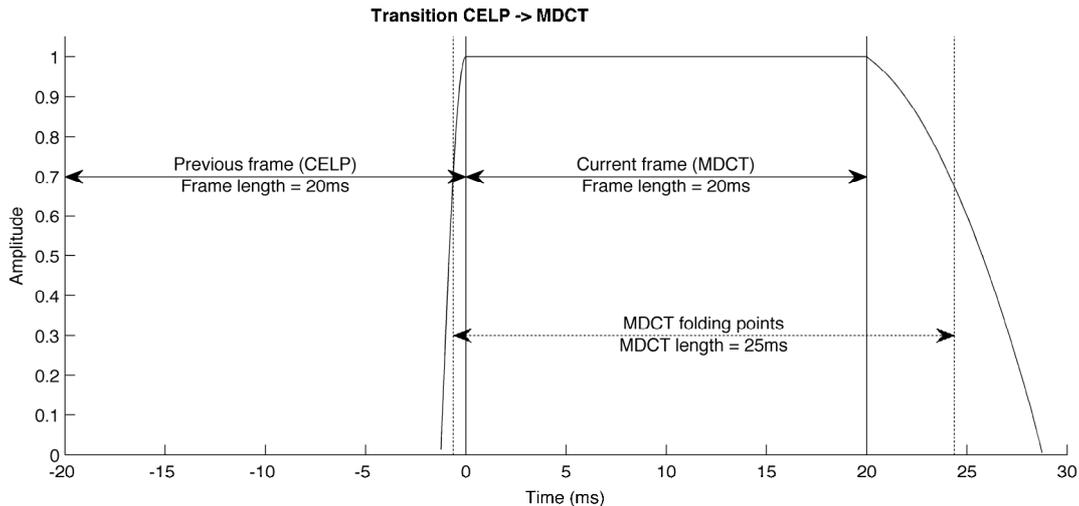
Figure 85: CELP to MDCT based TCX transition window (right part is here ALDO)

The right part of the transition window is not changed, such that it can be used in the next MDCT based frame as if it was a normal (non-transition) MDCT based frame. Similarly to the non-transition case, the right part of the transition window can have different shapes like ALDO, HALF or MINIMAL as described in subclause 5.3.2.3.

The left folding point is moved towards the past at 0.625ms before the transition border. This is equivalent to increasing the MDCT length from 20ms to 25ms. The corresponding numbers of MDCT bins are given in subclause 5.3.3.1.1.

The left part of the MDCT window is modified such that window segment with weight 1 covers the whole 20ms frame until the transition border, and the window segment before the transition border is a sine window $w_T(n)$ with length 1.25ms

$$w_T(n) = \sin\left(\frac{\pi}{2} \frac{n + \frac{1}{2}}{N_T}\right), \quad n = 0, ..., N_T - 1 \tag{1284}$$

with $N_T = 0.00125 sr_{mdct}$ is the window length in samples and sr_{mdct} is the sampling rate of the time-domain signal.

After windowing and MDCT, the MDCT-based TCX frame is encoded as described in subclause 5.3.3.

5.4.3.2 CELP coding mode to HQ MDCT coding mode

When the previous frame is CELP and the current frame is to be coded by HQ MDCT, the current frame is a transition frame in which two types of coding are used:

- Constrained CELP coding and (when required) simplified time-domain BWE coding

- HQ MDCT coding with a modified window

Constrained CELP coding means here that CELP is restricted to cover only the first subframe of the current frame, to code only a subset of CELP parameters, and to reuse parameters (LPC coefficients) from the previous CELP frame. These constraints are set to minimize the bit budget taken by continuing CELP coding in the current transition frame, this bit budget being taken out of HQ MDCT coding.

5.4.3.2.1 Constrained CELP coding and simplified BWE coding

The bit budget for CELP and BWE in the current (transition) frame is determined depending on the CELP coder used in the previous frame (12.8 kHz or 16 kHz) and coded audio bandwidth in the current frame. The following pseudo-code describes how this bit budget is subtracted from the total bit budget for the current frame (total_budget):

```
num_bits = total_budget
if CELP 12.8kHz was used in the previous frame
    cbrate = min(core_bitrate,  ACELP_24k40)
    if cbrate ≥ ACELP_11k60, num_bits = num_bits – 1, end
```

```
    num_bits = num_bits – table_ACB_bits[cbrate,GENERIC]
    num_bits =  num_bits – table_gain_bits [cbrate,TRANSITION]
    num_bits =  num_bits – table_FCB_bits [cbrate,GENERIC]
else (CELP 16 kHz was used in the previous frame)
    if core_bitrate ≤ ACELP_8k00, cbrate = ACELP_8k00
    else if core_bitrate ≤  ACELP_14k80, cbrate = ACELP_14k80
    else cbrate = min(core_bitrate, ACELP_22k60)
    end
    if cbrate ≥ ACELP_11k60, num_bits = num_bits – 1, end
    num_bits = num_bits – table_ACB_bits_16kHz[cbrate,GENERIC]
    num_bits =  num_bits – table_gain_bits_16kHz [cbrate,GENERIC]
    num_bits =  num_bits – table_FCB_bits_16kHz [cbrate,GENERIC]
end
if bandwidth is not NB and (bandwidth is not WB and CELP 16 kHz was not used in the
previous frame)
    num_bits =  num_bits –(6+6)
end
```

The bit rate for CELP coding is any case saturated by the minimum of HQ MDCT coding bit-rate and a predetermined bit-rate value (24 kbit/s or 22.6 kbit/s depending on whether the CELP core is at 12.8 or 16 kHz), then the numbers of bits allocated to CELP coding is subtracted and the remaining bit budget (denoted 'num_bits') is reserved for HQ MDCT coding normally operating at a bit rate 'core_bitrate' in the current frame. CELP coding in the extra subframe is configured to operate as if the current frame was CELP at a bit-rate denoted 'cbrate'; this CELP bit-rate depends on the CELP coder used in the previous frame (12.8 kHz or 16 kHz).

The coded CELP parameters in this extra subframe are: pitch filter flag (1 bit) if the CELP bit-rate is ≥ 11.6 kbit/s, pitch index for the adaptive codebook (ACB), codebook gains, fixed codebook (FCB) index. The bit allocation tables from CELP coding in Generic or Transition coding at 12.8 and 16 kHz are reused.

Besides, 12 bits are used for BWE in the extra subframe to code one gain (6 bits) and one pitch index (6 bits) for the high band above CELP synthesis.

Note that the current frame being a transition frame, one bit is used to indicate the type of CELP coding (12.8 kHz or 16 kHz) used in the extra CELP subframe; this bit is necessary to be able to decode correctly the transition frame in case of frame erasures.

LPC coefficients from the end of the previous frame are reused to code the extra subframe; constrained CELP coding reuses the subframe excitation coding with the same CELP core coder (12.8 kHz or 16 kHz) as in the previous frame, and this subframe coding is adapted from the procedure described in clause 5.2.3.1.

When the coded audio bandwidth is higher than the bandwidth of the core CELP coder, simplified BWE coding is applied. The previous and current input frames are high-pass FIR filtered to obtain the high-band; the cutoff frequency (6.4 or 8 kHz) depends on the core CELP coder. Then, pitch search based on correlation in the high band provides an estimated pitch lag and gain which are coded with 6 bits each.

5.4.3.2.2 HQ MDCT coding with a modified analysis window

HQ MDCT coding in the transition frame is identical to clause 5.3.4, except the MDCT analysis window is modified and the bit budget for HQ MDCT coding in the current frame is decreased as described in clause 5.4.3.2.1.
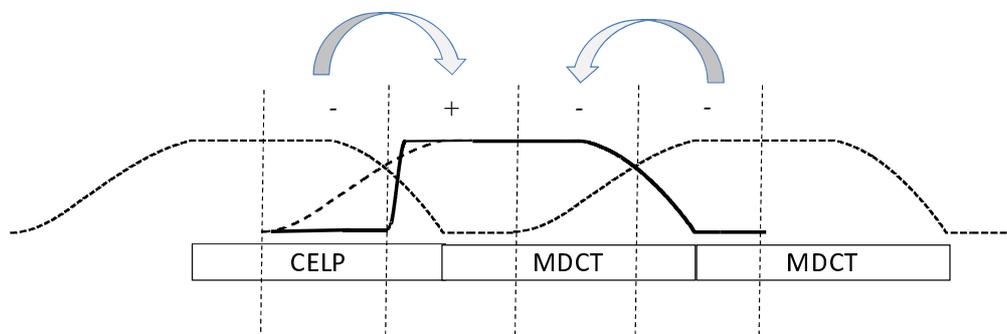
Figure 85a: Modified MDCT window in transition frame (CELP to MDCT transition)

The modified MDCT window is designed to avoid aliasing in the first part of the frame as shown in Figure 85a. Its shape also allows cross-fading between the synthesis from constrained CELP and simplified BWE and the synthesis from HQ MDCT as described in clause 6.3.3.2.3. Note that the frames labeled CELP and MDCT in Figure 85a represent the new frames of signal (20 ms) entering in the encoder; the actual coded frame is delayed by the encoder lookahead.

5.4.4 Internal sampling rate switching

The LP-based coding within EVS operates at two internal sampling rates, 12.8 kHz and 16 kHz. In active frames the 12.8 kHz internal sampling is employed at lower bit-rates ($\leq$ 13.2 kbps) while the 16 kHz internal sampling is employed at higher bit-rates ($\geq$ 16.4 kbps). Further in LP-based CNG, the 12.8 kHz internal sampling is employed at bit-rates $\leq$ 8.0 kbps while 16 kHz internal sampling is employed at bit-rates $\geq$ 9.6 kbps. Consequently a CELP internal sampling rate switching can happen either 1) in case of bit-rate switching or 2) in case of switching between active segments and LP-based CNG segments at 9.6 kbps and 13.2 kbps.

The MDCT-based TCX operates at 4 different internal sampling rates, 12.8, 16, 25.6 and 32 kHz. MDCT-based TCX internal sampling rate corresponds to the rate used for computing and transmitting its LP filter, filter employed for shaping the quantization noise in frequency domain. The same internal sampling rate is used for generating the low rate decoded signal computed at both encoder and decoder sides for updating memories of an eventual next CELP frame. The sampling-rate switching in MDCT-Based TCX can only happen either 1) in case of bit-rate switching or 2) in case of switching from CELP at 13.2kbps or switching from LP-based CNG segments at 9.6 kbps.

When changing the internal sampling rate, a number of memory and buffer updates needs to be done. These are described in subsequent subclauses.

5.4.4.1 Reset of LPC memory

In case of internal sampling rate switching, the LSF quantization is run in safety-net mode, such that no prediction is used and the MA/AR memories are not used.

5.4.4.2 Conversion of LP filter between 12.8 and 16 kHz internal sampling rates

When switching between internal sampling rates of 12.8 kHz and 16 kHz, the previous LP filter needs to be converted both at the encoder and the decoder between these two sampling rates in order to determine the interpolated LP parameters of the current frame. For this purpose, the LP filter of the previous frame could be recomputed at the current sampling rate based on the past synthesis signal that is already available. However, this would require complete LP analysis and resampling the past synthesis signal both at the encoder and the decoder. A less complex method is used here based on re-estimating the LP filter from its power spectrum modified corresponding to the current sampling frequency. The autocorrelation is computed from this modified power spectrum for solving the parameters of the converted LP filter with the Levinson Durbin algorithm. The converted LP filter is finally transformed to its line spectrum frequency representation for interpolation with the corresponding parameters of the current frame.

The computation and modification of the power spectrum as well as the computation of the autocorrelation are described in the following subclauses. The Levinson-Durbin algorithm is described in subclause 5.1.9.4 and the determination of the line spectrum frequencies in subclause 5.1.9.5.

Note that only the quantized LP filter is converted. Although the perceptual weighting filter uses the unquantized LP filter at the encoder, it is sufficient to use the converted quantized LP filter for interpolation when switching between internal sampling rates. This approximation avoids an additional conversion procedure for the unquantized LP filter.

5.5.4.1.1 Modification of the Power Spectrum

When switching the internal sampling rate down to 12.8 kHz from 16 kHz, the converted LP filter models the power spectrum of the LP filter originally estimated at a sampling rate of 16 kHz up to the new cut-off frequency. This is accomplished by computing the power spectrum of the LP filter at n_0 frequency points equispaced in $[0, 4\pi/5]$, corresponding to the Nyquist frequency at 12.8 kHz sampling rate. This frequency range of the power spectrum is then mapped onto $[0, \pi]$ when computing the autocorrelation for solving the parameters of the converted LP filter.

Conversely when switching the internal sampling rate up to 16 kHz from 12.8 kHz, the power spectrum of the LP filter originally estimated at a sampling rate of 12.8 kHz is computed at n_1 frequencies equispaced in $[0, \pi]$. This frequency range of the power spectrum is mapped onto $[0, 4\pi/5]$ for autocorrelation computation. The power spectrum unknown at frequencies $(4\pi/5, \pi]$ is approximated by extending the power spectrum value at $4\pi/5$ over this range by $(n_1-1)/4$ values for autocorrelation computation. This procedure thus re-estimates the LP filter at sampling frequency 16 kHz with an approximated, extended upper band.

By choosing $n_0 = n_1 = 41$, all power spectrum and autocorrelation computations can be accomplished on two equispaced frequency grids, one including the frequency points $0, \pi/40, K, \pi$ and one the points $0, \pi/50, K, \pi$. Converting down to 12.8 kHz is hence equivalent to dropping 10 last values away from the power spectrum of the original LP filter. Similarly, converting up to 16 kHz translates to adding 10 approximated values to the power spectrum of the original LP filter.

The same procedure is applied identically both at the encoder and the decoder.

5.5.4.1.2   Computation of the Power Spectrum

The LP filter $1/A(z)$ is converted to another sampling rate by truncating or extending its power spectrum

$$G(\omega) = 1/|A(\omega)|^2, \quad \omega \in [0, \pi], \tag{1285}$$

to the frequency range that corresponds to the new sampling rate and re-estimating linear prediction coefficients from the autocorrelation computed from this modified spectrum. For reduced complexity, the power spectrum is expressed on the real axis by utilizing the line spectrum frequency decomposition

$$A(z) = \frac{(1+z^{-1})F_1(z) + (1-z^{-1})F_2(z)}{2}, \tag{1286}$$

where the polynomials $F_1(z)$ and $F_2(z)$ are defined as in subclause 5.1.9.5. The zeros of these two polynomials give the line spectrum frequencies of $A(z)$.

Because of the symmetry properties of the polynomials $F_1(z)$ and $F_2(z)$, they can be expressed as the polynomials $\overline{F}_1(x)$ and $\overline{F}_2(x)$ of $x = \cos(\omega)$. It can be shown that for an LP filter of an even order,

$$|A(x)|^2 = 2(1+x)\overline{F}_1^2(x) + 2(1-x)\overline{F}_2^2(x), \quad x \in [-1, 1]. \tag{1287}$$

The use of this expression is motivated by the observation that the polynomials $\overline{F}_1(x)$ and $\overline{F}_2(x)$ can be evaluated efficiently for a given $x \in [-1, 1]$ with Horner's method [34]. The value of these polynomials is needed on frequency grids described in subclause 5.5.4.1.1. The expression (1287) obtains a particularly simple form at $x = -1, 0$, and 1 that can be utilized in computation.

The polynomials $\overline{F}_1(x)$ and $\overline{F}_2(x)$ can be derived by substituting the explicit forms of the Chebyshev polynomials

$$T_m(x) = \cos(m \arccos(x)), \quad m = 0, 1, L \tag{1288}$$

to the Chebyshev series representation of $F_1(z)$ and $F_2(z)$ [34]. This same representation is employed in subclause 5.1.9.5 for the determination of the line spectrum frequencies. The explicit forms of $T_m(x)$ can readily be written by using the recursion

$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x), \quad T_0(x) = 1, \ T_1(x) = x. \tag{1289}$$

By definition the zeros of $\overline{F}_1(x)$ and $\overline{F}_2(x)$ are respectively the cosines of the even and odd line spectrum frequencies. The coefficients of these polynomials are thus readily obtained when the line spectrum frequencies of $A(z)$ are known. Given that the order of $A(z)$ is 16, the polynomial $\overline{F}_1(x)$ is of order 8 and can be expressed as

$$\begin{aligned} \overline{F}_1(x) &= \overline{f}_1(0)x^8 + \overline{f}_1(1)x^7 + \cdots + \overline{f}_1(8) \\ &= \overline{f}_1(0)\,(x-q_0)(x-q_2)\,(x-q_4)\ldots(x-q_{14}). \end{aligned} \tag{1290}$$

The leading coefficient $\bar{f}_1(0) = 128$ is constant. This relation yields a simple recursion for solving the coefficients of $\bar{F}_1(x)$ from the even line spectrum pairs $q_k = \cos(\omega_k)$ for $k = 0, 2, \ldots, 14$. The coefficients of $\bar{F}_2(x)$ are obtained correspondingly from the odd line spectrum pairs.

If no line spectrum frequency representation is available for $A(z)$, one can alternatively first compute the coefficients of the polynomials $F_1(z)$ and $F_2(z)$ from those of $A(z)$ and then solve the coefficients of $\bar{F}_1(x)$ and $\bar{F}_2(x)$ from a set of equations that relate the two representations. These relating equations can be derived by substituting the explicit forms of the Chebyshev polynomials $T_m(x)$ to the Chebyshev series representation of $F_1(z)$ and $F_2(z)$. This approach is employed when switching from the AMR-WB IO mode, which uses the immittance spectrum frequency representation instead of line spectrum frequencies.

5.5.4.1.3 Computation of the Autocorrelation

The autocorrelation of the LP filter is obtained by the inverse Fourier Transform of the modified power spectrum. Since the power spectrum is real and symmetric, the relation between the autocorrelation and the power spectrum can be expressed through the integral

$$r(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G(\omega) \cos k\omega \, d\omega, \qquad k = 0, 1, \ldots \tag{1291}$$

Because the power spectrum is real and symmetric, $G(\omega) = G(-\omega)$, it suffices to evaluate the integral over only the upper half of the unit circle. Due to this symmetry, the rectangle rule for approximating the integral can be expressed as

$$n_G r(k) \approx G(0) + (-1)^k G(\pi) + 2 \sum_{\omega \in \Omega} G(\omega) \cos(k\omega), \qquad k = 0, 1, \ldots \tag{1292}$$

where Ω is the set of $n_G - 2$ frequencies equispaced in $[0, \pi]$, but excluding 0 and π to avoid double counting. The number n_G of these frequencies is hereafter assumed odd.

Note that in sequel the factor n_G is omitted for simplicity from the approximation of the autocorrelation. Namely, the autocorrelation can be scaled for convenience, because the resulting linear prediction coefficients are invariant to this scaling.

The expression of autocorrelation can be rewritten equivalently for more efficient computation by utilizing the symmetries of the cosine term relative to $\omega = \pi/2$ through the following trigonometric identities:

$$\begin{aligned} \cos(\pi - k\omega) &= (-1)^k \cos(k\omega), \\ \cos(k\pi/2) &= (1/2)(1 + (-1)^{k+1})(-1)^{\lfloor k/2 \rfloor}, \end{aligned} \tag{1293}$$

where $k = 0, 1, \ldots$ and $\omega \in (0, \pi/2)$. The operator $\lfloor \cdot \rfloor$ rounds to the nearest integers towards minus infinity. The term $(1 + (-1)^{k+1})(-1)^{\lfloor k/2 \rfloor}$ simply generates the sequence 2, 0, –2, 0, 2, 0, –2, 0, 2, … and is hence readily implementable.

By employing the two trigonometric identities given above, the autocorrelation can be rewritten as

$$\begin{aligned} r(k) = G(0) &+ (-1)^k G(\pi) \\ &+ (1 + (-1)^{k+1})(-1)^{\lfloor k/2 \rfloor} G(\pi/2) + 2 \sum_{\omega \in \Lambda} (G(\omega) + (-1)^k G(\pi - \omega)) \cos(k\omega), \quad k = 0, 1, \ldots \end{aligned} \tag{1294}$$

where Λ is the set of $(n_G - 3)/2$ frequencies equispaced in $[0, \pi/2]$ but excluding 0 and $\pi/2$. The cosine term of the autocorrelation is evaluated using the recursion

$$\cos(k\omega) = 2\cos(\omega)\cos((k-1)\omega) - \cos((k-2)\omega), \qquad k = 2, 3, \ldots \tag{1295}$$

starting from $\cos(0)$ and $\cos(\omega)$. The value of $\cos(\omega)$ is stored in a table that holds all the entries needed for $\omega \in \Lambda$.

When switching the internal sampling rate from 16 kHz down to 12.8 kHz, a grid of $n_G = 41$ equispaced frequency points is used. Switching from 12.8 kHz up to 16 kHz uses a grid of $n_G = 51$ points, see subclause 5.5.4.1.1.

5.4.4.3   Extrapolation of LP filter

In case of sampling rate switching involving at least a sampling rate being neither 12.8 nor 16 kHz, the previous LP filter is not converted. Instead, the previous quantized end-frame LSP/LSF and the quantized mid-frame LSP/LSF are set to the current quantized end-frame LSP/LSF.

The LP filter is also extrapolated when the conversion of LP filter between 12.8 and 18 kHz, described in suclause 5.4.4.2, does not produce a stable filter. The stability of the filter is detected during the Levinson Durbin algorithm described in subclause 5.1.9.4. A filter is detected as unstable when at least one of the reflection coefficients k_i, for $i = 2,\ldots,16$ has an absolute value greater than 0.99945.

5.4.4.4   Buffer resampling with linear interpolation

Switching the internal sampling rate requires several memories to be resampled. . In order to reduce the complexity of the resampling processing, a simple linear interpolation is used most of the time instead of a conventional low-pass filtering method.

The basic operation for interpolating a point is done as follows:

$$y(n) = x(n') + (i - n') \cdot (x(n'+1) - x(n')) \tag{1296}$$

where $y(n)$, $n = 0,...,N_n - 1$ is the new resampled buffer of size N_n and $x(n')$, $n' = 0,...,N_o - 1$ is the old buffer of size N_o. The index n is initialized to 0 while index n' is equal to the integer part of the position i, $n = \lfloor i \rfloor$. The position is initialized to:

$$i = 0.5 \cdot \frac{N_o}{N_L} - 0.5 \tag{1297}$$

where $\frac{N_o}{N_L}$ is the increment of the position i for each unit increment of the index n.

5.4.4.5   Update of CELP input signal memories

The pre-emphasized input signal $s_{pre}(n)$ defined in subclause 5.1.4 is updated at both CELP internal sampling rate 12.8 and 16 kHz at any bit-rates. No specific processing is then needed in case of sampling rate switching.

The weighed synthesis filter $A(z/\gamma_1)H_{de-emph}(z)$ state is updated in three different ways in case of sampling rate switching:

- It is set to zero if a sampling rate different from 12.8 and 16 kHz is involved.

- At bit-rates <= 8, 13.2, 32 and 64 kbps, the memory states is updated in previous frame as usual and the difference in sampling rate between the previous and the current frame is simply ignored.

- Otherwise, the state is recomputed by filtering the resampled LPC synthesis filter state obtained in subclause 5.4.4.7 through the filter filter $A(z/\gamma_1)H_{de-emph}(z)$ and by taking computing the error between the obtained signal and the input weighted signal. The memory state is used for computed the target signal of the next frame as defined in subclause 5.2.3.1.2.

5.4.4.6   Update of MDCT-based TCX input signal memories

The past of the input signal and the past of weighted signal are needed for MDCT-based TCX and both past signal are resampled as in sub-clause 5.4.4.4.

5.4.4.7 Update of CELP synthesis memories

For being able to make a seamless transition from CELP to CELP or from MDCT-based TCX to CELP, the following memory states have to be maintained:

- The adaptive codebook state

- The LPC synthesis filter state

- The de-emphasis state

The three memory states are maintained at both encoder and decoder side in CELP mode and in MDCT-based TCX coding mode. The following specific processing is performed in case of internal sampling rate switching.

The adaptive codebook state covers at the encoder side a frame, i.e. 20 ms. In case of internal sampling rate switching between 12.8 and 16 kHz, the adaptive codebook is resampled with the method described in subclause 5.4.4.4. If it involves at least a sampling rate different from 12.8 and 16 kHz, the adaptive codebook is reset with zeros.

The LPC synthesis filter state doesn't cover a fixed time duration but a fixed number samples equal to the order of the LPC. This order is always 16. For being able to resample this state at any of the sampling rate between 12.8 and 48 kHz, the memory of the LPC synthesis filter state is extended from 16 to 60 samples, which represents 1.25ms at 48 kHz. The memory resampling from sampling rate fs_1 Hz to sampling rate fs_2 Hz can summarized as:

$$
\begin{aligned}
N_1 &= \left\lfloor \frac{1.25 \cdot fs_1}{1000} \right\rfloor \\
N_2 &= \left\lfloor \frac{1.25 \cdot fs_2}{1000} \right\rfloor \\
mem\_sym\_r[L\_SYN\_MEM - N_2] &= \\
resamp&(mem\_sym\_r[L\_SYN\_MEM - N_1], N_1, N_2)
\end{aligned}
\tag{1298}
$$

where $y = resamp(x, N_1, N_2)$ is the function resampling the input buffer x from N_1 to N_2 samples as described in subclause 5.4.4.4. L_SYN_MEM is the largest size in samples that the memory can cover and is equal to 60 samples. At any sampling rate and at any time, *mem_syn_r* is updated with the last L_SYN_MEM output samples and is then eventually resampled in case of internal sampling rate switching at the beginning of the next frame.

The de-emphasis has a fixed order of 1, which represents also a different time duration at different sampling rates. However the resampling stage is not performed and the memory update in done as usual even in case of intern sampling rate switching.

5.4.5 EVS primary and AMR-WB IO

The codec support a seamless switching between primary and AMR-WB IO modes. While most of memories and past buffers are shared between the two modes, there are some particularities that need to be properly handled. The following scenarios can happen:

5.4.5.1 Switching from primary modes to AMR-WB IO

When a CELP based AMR-WB IO encoded frame is preceded by a primary mode encoded frame, the memories of the CELP AMR-WB IO encoded frame have to be updated or converted before starting the encoding. These include:

- Convert previous quantized end-frame LSFs to ISFs

- Convert previous quantized end-frame LSPs to ISPs

- Set previous un-quantized end-frame ISPs to converted quantized end-frame ISPs

- Convert previous CNG quantized end-frame LSPs to ISPs

- Limit index of last encoded CNG energy to 63

- Reset the gain quantization memory to -14.0.

- In case the switching happens in the SNG segment, force SID frame

- Reset AR model LP quantizer memory

In case the AMR-WB IO frame is preceded by CELP primary frame at 16 kHz internal sampling rate, the processing described in subclause 5.4.4 is performed.

Finally in case the AMR-WB IO frame is preceded by MDCT primary frame, the processing described in subclause 5.4.2 is performed.

5.4.5.2 Switching from AMR-WB IO mode to primary modes

When a primary mode encoded frame is preceded by a CELP based AMR-WB IO encoded frame, the memories of the primary mode encoded frame have to be updated or converted before starting the encoding. These include:

- First three ACELP frames are processed using safety-net LP quantizer

- Convert previous quantized end-frame ISFs to LSFs

- Convert previous quantized end-frame ISPs to LSPs

- Convert previous CNG quantized end-frame LSPs to ISPs

- Reset BWE past buffers

- reset the unvoiced/audio signal improvement memories

In case of CELP at 16 kHz internal sampling rate primary mode frame is preceded by the AMR-WB IO frame, the processing described in subclause 5.4.4 is performed.

Finally in case the MDCT primary frame is preceded by AMR-WB IO frame, the processing described in subclause 5.4.3 is performed.

5.4.6 Rate switching

A seamless switching between all EVS primary rates is supported in the codec. Since most of the states and memories are shared and maintained at any bit-rates, a complete re-initialization is not needed. The coding tools are able to be reconfigured at the beginning of any frame. The different bit-rate dependent setups of each tool are described in each corresponding subclause. Rate switching doesn't require any specific handling, except in the following scenarios.

5.4.6.1 Rate switching along with internal sampling rate switching

In case the internal sampling rate changes when switching the bit-rate, the processing described in subclause 5.4.4 is performed at first.

5.4.6.2 Rate switching along with coding mode switching

In case the internal sampling rate changes when switching the bit-rate, the processing described in subclause 5.4.3 is performed. New possible transitions from MDCT to CELP mode are possible during rate switching compared to table 149. table 150 lists all different cases achievable during rate switching depending on MDCT mode and the new CELP bit rate.

Table 150: MDCT to CELP transition modes in case of rate switching

Switching from	Switching to	CELP Bitrate (kbps)	Transition mode
HQ MDCT or TCX	CELP	7.2	MC1
HQ MDCT or TCX	CELP	8	MC1
TCX	CELP	9.6	MC2
HQ MDCT	CELP	9.6	MC3
HQ MDCT or TCX	CELP	13.2	MC1
TCX	CELP	16.4	MC2
HQ MDCT	CELP	16.4	MC3
TCX	CELP	24.4	MC2
HQ MDCT	CELP	24.4	MC3
HQ MDCT or TCX	CELP	32	MC1
HQ MDCT or TCX	CELP	64	MC1

If the internal sampling rate is also changing, the processing of subclause 5.4.4 is performed beforehand.

5.5 Frame erasure concealment side information

The codec has been designed with emphasis on performance in frame erasure conditions and several techniques limiting the frame erasure propagation have been implemented, namely the TC mode, the safety-net approach for LSF quantization, and the memory-less gain quantization. To further enhance the performance in frame erasure conditions, side information, consisting of concealment/recovery parameters, is sent in the bitstream to the decoder. This supplementary information improves the frame erasure concealment (FEC) and the convergence and recovery of the decoder after erased frames. The detailed concealment and recovery processing is described in [6].

The concealment/recovery parameters that are transmitted to the decoder depend on the bitrate and coding mode. They will be described in the following subclauses together with the information about configurations when they are transmitted.

5.5.1 Signal classification parameter

The signal classification parameter is determined based on the classification for FEC, described in subclause 5.1.13.3. The classification uses the following five classes to classify speech signals: UNVOICED, UNVOICED TRANSITION, VOICED TRANSITION, ONSET and VOICED. For the purpose of the FEC classification, inactive signals fall into the UNVOICED category. Though there are five signal classes, they can be encoded with only two bits as the differentiation of the both TRANSITION classes can be done unambiguously determined based on the class of the preceding frame. The rules for the FEC signal classification are described in subclause 5.1.13.3.3.

The signal classification parameter is not transmitted at lowest bitrates. Further, it does not need to be transmitted in coding modes that allow to classify the frame implicitly, e.g. in the UC and VC modes. The signal classification parameter is transmitted in GC and TC modes at 13.2 kb/s, 32 kb/s and 64 kb/s.

5.5.2 Energy information

Precise control of the speech energy is very important in frame erasure concealment. The importance of the energy control becomes more evident when a normal operation is resumed after an erased block of frames. Since VC and GC modes are heavily dependent on prediction, the actual energy cannot be properly estimated at the decoder. In voiced speech segments, the incorrect energy can persist for several consecutive frames, which can be very annoying, especially when this incorrect-valued energy increases.

To better control the energy of the synthesized sound signal at the decoder in case of frame erasure, the energy information is estimated and sent using 5 bits. The goal of the energy control is to minimize energy discontinuities by scaling the synthesized signal to render the energy of the signal at the beginning of the recovery frame (a first non erased frame received following frame erasure) to be similar to the energy of the synthesized signal at the end of the last frame erased during the frame erasure. The energy of the synthesized signal in the received first non erased frame is

further made converging to the energy corresponding to the received energy parameter toward the end of that frame while limiting an increase in energy.

The energy information is the maximum of the signal energy for frames classified as VOICED or ONSET, or the average energy per sample for all other frames. For VOICED or ONSET frames, the maximum signal energy is computed pitch-synchronously at the end of the current frame as follows:

$$E = \max\left(\hat{s}_1^2(n)\right), \qquad n = L - T^{end}, \ldots, L - 1 \tag{1299}$$

where $L = 256$ is the frame length for the 12.8 kHz internal sampling rate, and $L = 320$ is the frame length for the 16 kHz sampling rate. Signal $\hat{s}_1(n)$ is the local synthesis signal sampled at 12.8 kHz or 16 kHz depending on the internal sampling rate. The integer pitch period length T^{end} is the rounded pitch period of the last subframe, i.e. $T^{end} = \left\lfloor d_{fr}^{[3]} + 0.5 \right\rfloor$ for the 12.8 kHz core, and $T^{end} = \left\lfloor d_{fr}^{[4]} + 0.5 \right\rfloor$ for the 16 kHz core.

For all other classes, E is the average energy per sample of the last two subframes of the current frame, i.e.,

$$E = \frac{1}{128} \sum_{n=L-128}^{L-1} \hat{s}_1^2(n) \tag{1300}$$

The energy information is quantized using a 5-bit linear quantizer in the range of 0 dB to 96 dB with a step of 3 dB. The quantization index is given by

$$I_E = \left\lfloor \frac{10\log(E + 0.001)}{3} \right\rfloor \tag{1301}$$

The index is limited to the range [0,…, 31].

The energy information is sent only in GC mode at 32 and 64 kb/s.

5.5.3   Phase control information

The phase control is particularly important when recovering after a lost voiced segment of a signal. After a block of erased frames, the decoder memories become unsynchronized with the encoder memories. Sending some phase information helps in the re-synchronization of the decoder. The rough position of the last glottal pulse in the previous frame is sent.

Let $T^{[-1]} = \left\lfloor d_{fr}^{[-1]} \right\rfloor$ be the integer closed-loop pitch lag for the last subframe of the previous frame. The position of the last glottal pulse, τ, is searched among the $T^{[-1]}$ last samples of the previous frame by looking for the sample with the maximum amplitude of low-pass filtered LP residual signal. A simple FIR low-pass filter with coefficients 0.25, 0.5 and 0.25 is used.

The position of the last glottal pulse, τ, is encoded using 8 bits in the following manner. The precision used to encode the position of the pulse depends on the integer part of the closed-loop pitch lag for the first subframe of the current frame, $T_0 = \left\lfloor d_{fr}^{[0]} \right\rfloor$. This is possible because this value is known both at the encoder and the decoder, and is not subject to erasure propagation after one or several frame losses. When T_0 is less than 128, the position of the last glottal pulse, relative to the end of the previous frame, is encoded directly with a precision of one sample. When $T_0 \geq 128$, the position of the last glottal pulse, relative to the end of the previous frame, is encoded with a precision of two samples by using a simple integer division, i.e., $\tau/2$. Finally, the information about the sign of the impulse is encoded by incrementing the transmitted index by 128 when the sign of the glottal pulse is negative. The MSB in the 8-bit index thus represents the sign of the last glottal pulse. The inverse procedure is done at the decoder.

The phase control information is sent only in GC mode at 32 and 64 kb/s.

5.5.4   Pitch lag information

To improve speech quality under erroneous channel, a pitch lag estimate of the next frame is calculated at the encoder and transmitted as a side information for better excitation at concealed frame. By exploiting the 8.75-ms look-ahead signal used for the frame-end autocorrelation calculation, the pitch lag can be obtained without any additional delay. This tool is activated only at ACELP frame under operational modes of 24.4kbps.

The side information includes activation flag. For the frames classified as ONSET or VOICED under GC or VC mode, the activation flag is set to 1. For the other frames, the activation flag is set to 0.

In case the activation flag equals to 1, the pitch lag is encoded with 4 bits and transmitted on top of the activation flag. In case the activation flag equals to 0, only the activation flag is transmitted as side information.

To estimate a pitch lag at the look-ahead signal, this tool uses an extrapolated LSF parameter $\dot{\omega}_i$ and corresponding LP coefficients. For the LSF parameter extrapolation, the mean LSF vector is updated every frame and the extrapolated LSF $\dot{\omega}_i$ is calculated as follows.

$$
\begin{aligned}
\dot{\omega}_i &= \alpha \omega_i^{[-1]} + (1-\alpha)\overline{\omega}_i \\
\overline{\omega}_i &= \beta \omega_i^C + (1-\beta)\frac{\omega_i^{[-3]} + \omega_i^{[-2]} + \omega_i^{[-1]}}{3}
\end{aligned}
\tag{1302}
$$

where $\omega_i^{[-j]}$ is LSF vector of the last 3 frame, and ω^C is the mean LSF vector. α, β depends on the previous coder type and the signal class. Decision rule for the constants used for LSF concealment applies to the decision of α, β. The extrapolated LSF $\dot{\omega}_i$ is converted into LSP parameter and extrapolated LP coefficients. The procedure is the same as the one under clean channel.

LP residual $x(n)$ is calculated for the 8.75-ms look-ahead signal with the extrapolated LP coefficients. LP residual $x(n)$ is used as target signal without perceptual weighting to estimate the pitch lag with low complexity. The pitch lag estimate is obtained by maximizing the following correlation.

$$
T_k = \frac{\sqrt{\sum_{n=0}^{L'-1} x(n) \cdot y_k(n)}}{\sqrt{\sum_{n=0}^{L'-1} y_k(n) \cdot y_k(n)}}
\tag{1303}
$$

where L' is the number of samples of the 8.75-ms look-ahead sub-frame, and $y_k(n)$ is the past excitation at the delay of k. The search range is limited to $[T_0 - 8, T_0 + 7]$, where T_0 is the pitch lag of the last sub-frame. Then the differential pitch lag from T_0 is included to the side information with 4 bits.

5.5.5   Spectral envelope diffuser

A frame loss around speech onset sometimes causes too sharp peaks at LP spectrum, and sudden power increase in concealed signal. Spectral envelope diffuser mitigates the sudden power change and provides better recovery of concealed signal. The activation flag for spectral envelope diffuser is encoded with 1 bit and transmitted as a side information. This tool is active only at 9.6, 16.4, 24.4 kbps.

The activation is based on the function of merits depending on LSF improvement counter lc, LSF parameter of the previous frame $\omega^{[-1]}$, the extrapolated LSF $\dot{\omega}^{[-1]}$ obtained in the guided PLC for pitch lag at the previous frame, and a modified LSF parameter $\tilde{\omega}^{[-1]}$. The modified LSF parameter $\tilde{\omega}^{[-1]}$ is calculated as follows.

$$
\tilde{\omega}_j^{[-1]} = \begin{cases} j \cdot \delta & (1 \le j < idx) \\ \dot{\omega}_j^{[-1]} & (idx \le j \le 16) \end{cases}
\tag{1304}
$$

where idx is the lowest number of j which satisfies the following equation.

$$\dot{\omega}_j^{[-1]} > Th_{freq} \tag{1305}$$

$$\delta = \frac{\sum_{j=1}^{idx-1} \dot{\omega}_j^{[-1]}}{idx-1} \tag{1306}$$

where δ is computed as follows. Th_{freq} is a threshold which equals to 1900 for 12.8 kHz internal sampling frequency, 2375 for 16 kHz internal sampling frequency.

After initialized with 0, the LSF improvement counter lc is computed as follows:

$$For\ j = 1\ldots16$$

$$lc = \begin{cases} lc+1 & \left(\left| \dot{\omega}_j^{[-1]} - \omega_j^{[-1]} \right|^2 > \left| \tilde{\omega}_j^{[-1]} - \omega_j^{[-1]} \right|^2 \right) \\ lc & (otherwise) \end{cases} \tag{1307}$$

In case the following 4 equations are satisfied, the activation flag is set to 1, otherwise set to 0. Th_{int} equals to 90 for 12.8 kHz internal sampling frequency, 112.5 for 16 kHz internal sampling frequency. Th_{mean} equals to 800 for 12.8 kHz internal sampling frequency, 1000 for 16 kHz internal sampling frequency.

$$\sum_{j=1}^{16} \left| \dot{\omega}_j^{[-1]} - \omega_j^{[-1]} \right|^2 > 1.15 \sum_{j=1}^{16} \left| \tilde{\omega}_j^{[-1]} - \omega_j^{[-1]} \right|^2 \tag{1308}$$

$$\frac{1}{4} \left(\sum_{j=1}^{4} \omega_j^{[-1]} - \sum_{j=1}^{4} \omega_j^{[-2]} \right) > Th_{int} \tag{1309}$$

$$\frac{1}{4} \left(\sum_{j=1}^{4} \omega_j^{[-2]} \right) > Th_{mean} \tag{1310}$$

$$lc > 2 \tag{1311}$$

The activation flag is updated based on algebraic codebook gain of the previous and the current frame. In case one of the following equations is satisfied, the activation flag is set to 0.

$$\frac{\overline{g}}{g^{[-1]}} < 1.098$$
$$g_{min} > 0.82 \tag{1312}$$

where g_{min} is the minimum value of algebraic codebook gains of current frame, $\overline{g}$ is the mean value of algebraic codebook gains of current frame, and $g^{[-1]}$ is the mean value of algebraic codebook gains of the previous frame.

The activation flag is further updated with the stability factor of LSF parameter. In case the stability factor is greater than 0.02, the activation flag is set to 0.

Finally the activation flag is encoded with 1 bit and transmitted as side information.

5.5.6 Tonality flag information

The flag *enablePlcWaveadjust* is set to one if the bit rate is one out of the set of {48 kbps, 96 kbps, 128 kbps}. For every frame for which *enablePlcWaveadjust* is one, two parameters of spectral flatness are computed as follows:

Let i be the sequence number of an arbitrary frame and SFM_i denote the spectral flatness of the i'th frame. SFM_i is defined as follows:

$$SFM_i = \frac{G_i}{A_i} \tag{1312a}$$

where $G_i = \left(\prod_{m=0}^{M-1} |c^i(m)| \right)^{1/M}$ is the geometric mean of the signal amplitudes, $A_i = 1/M \sum_{m=0}^{M-1} |c^i(m)|$ is the arithmetic mean of the signal amplitudes, $c^i(m)$ is the MDCT coefficient at frequency point m, and M is the number of the frequency points. The MDCT coefficients are either the original MDCT coefficients or the spectrum-shaped MDCT coefficients.

The original MDCT coefficients and the spectrum-shaped MDCT coefficients are used to compute two parameters of spectral flatness of this frame, denoted SFM and SFM' respectively. For the frame with the mode of TCX20, the spectral flatness of the frame is computed by using the MDCT coefficients of the whole frame. For the frame with the mode of TCX10, the spectral flatness of the frame is computed by using the MDCT coefficients of the second sub-frame. In both cases, M is the smaller value between the number of the frequency points of the MDCT coefficients and 200. If SFM is smaller than a threshold $K; (0 \le K \le 1)$, the flag of frame type is set to tonal type; otherwise, the flag of frame type is set to non-tonal type. If SFM' is smaller than another threshold $K'; (0 \le K' \le 1)$, the flag of frame type is reset to tonal type. Then the obtained flag of frame type, together with the coded bit stream, is transmitted to the decoder side.

5.6 DTX/CNG operation

5.6.1 Overview

This subclause describes the discontinuous transmission (DTX) scheme and the comfort noise generation (CNG) algorithm. The DTX/CNG operation, which is activated on a command line, is used to reduce the transmission rate by simulating background noise during inactive signal periods. The regular DTX/CNG modes are supported for bit rates up to 24.4 kbps. For higher bit rates, the EVS codec supports a less aggressive DTX/CNG scheme that only switches to CNG for low input signal power.

The reduction of the transmission rate during inactive periods is achieved by coding the parameters referred to as comfort noise (CN) parameters. These parameters are used at the decoder to regenerate the background noise as well as possible, by respecting the spectral and temporal content of the background noise at the encoder. In the EVS Codec, the CNG algorithm reproduces high quality comfort noise by choosing between a linear prediction-domain based coding mode (LP-CNG) and a frequency-domain based coding mode (FD-CNG), according to the input characteristics. Each of the two coding modes utilizes a different set of CN parameters. In the LP-CNG mode, four CN parameters are analyzed and encoded: the low-band excitation energy, the low-band signal spectrum, the low-band excitation envelope and the high-band energy, where the high-band energy is only encoded for SWB/FB input. In the FD-CNG mode, the CN parameters consisting of global gain and spectral energies grouped in critical bands. Those parameters are encoded by a vector quantizer for transmission.

When the codec is operated with the DTX/CNG operation, the signal activity detector (SAD) is used to analyse the input signal to determine whether the signal comprises an active or inactive signal (see SAD decision in subclause 6.2). Based on its analysis, the SAD generates a SAD flag, f_{SAD}, whose state indicates whether the signal is active ($f_{SAD} = 1$) or merely a background noise ($f_{SAD} = 0$). When $f_{SAD} = 1$, the regular encoding and decoding process is performed, as in the default option. When $f_{SAD} = 0$, DTX functions are run at the encoder that transmit either a silence insertion descriptor (SID) frame or a NO_DATA frame. The SID frame contains the CN parameters, which are used to update the statistics of the background noise at the decoder, whereas the NO_DATA frame is empty. The SID frame is always encoded using 48 bits regardless the actual CNG mode operating.

Further, hangover logic, as described in subclause 6.2, is used to enhance the quality of SID frames. The hangover logic in the SAD algorithm is such that the encoder waits for a certain number of frames before switching from the active signal to inactive signal. If the background noise contains transients that force the encoder to switch from inactive signal to active signal and then back to inactive signal in a very short time period, no hangover is used.

5.6.1.1 SID update

The CN parameters are transmitted at a fixed or adaptive rate during inactive signal periods using a command line parameter. The fixed rate is limited to between 3 and 100 frames. The adaptive rate, in general, is dependent on the background noise characteristics such as the current signal-to-noise ratio (SNR) and is limited to be between 8 and 50. Generally, at a high SNR, the SID frames are transmitted with a lower rate to achieve a significant reduction of average data rate at the cost of only minor quality degradation. On the other hand, at a low SNR, SID frames are transmitted with a higher frequency so that the comfort noise remains as natural as possible. Thus, increasing SNR implies decreasing SID frame frequency, whereas decreasing SNR implies increasing SID frame frequency.

To determine the adaptive SID transmission rate, the SNR is calculated based on the long-term energy of the active signal, $\overline{E}_{act}$, and background noise, $\overline{E}_{inact}$. The DTX algorithm updates both long-term values in each frame to take into account the possible evolutions of the level of the two respective signals. In the current frame, only one of these two energies is updated. If the current frame is classified as VOICED, the DTX module updates only the long-term energy of the active signal. Otherwise, it updates only the long-term energy of the background noise. The adaptive rate calculation is performed in every inactive signal frame after the preamble period. This period is characterized by at least 50 updates of both $\overline{E}_{act}$ and $\overline{E}_{inact}$.

The update of the long-term energy of an active signal is performed as follows:

$$\overline{E}_{act} = \alpha.\overline{E}_{act} + (1-\alpha) \cdot E_f \tag{1313}$$

and the update of the long-term energy of an inactive signal as

$$\overline{E}_{inact} = \alpha.\overline{E}_{inact} + (1-\alpha) \cdot E_f \tag{1314}$$

where $E_f = \|s_{nr}(n)\|$ is the energy of the denoised signal, $s_{nr}(n)$, in the current frame. α represents a forgetting factor. Its value is based on the energy evolution. The value of α is set either to 0.99 for slow adaptation, or 0.90 for fast adaptation of the long-term energy level. In case of $\overline{E}_{act}$, fast adaptation is chosen if $E_f > \overline{E}_{act}$ in the current frame, otherwise slow adaptation is applied. In case of $\overline{E}_{inact}$, the fast adaptation is chosen if $E_f < \overline{E}_{inact}$ in the current frame, otherwise slow adaptation is applied.

Having estimated the long-term energies, $\overline{E}_{act}$ and $\overline{E}_{inact}$, the SNR value in the logarithmic domain is calculated in every inactive signal frame as

$$SNR_{DTX} = 10 \cdot \log\left(\frac{\overline{E}_{act}}{\overline{E}_{inact}}\right) \tag{1315}$$

The SID transmission rate, r_{SID}, is finally adapted based on the current SNR value. The value of r_{SID} is linearly varied between a minimum value, r_{MIN}, that corresponds to a minimum SNR value, SNR_{MIN}, and a maximum value, r_{MAX}, that corresponds to a maximum SNR value, SNR_{MAX}. That is

$$r_{SID} = r_{MIN} + \frac{(r_{MAX} - r_{MIN}) \cdot (SNR_{DTX} - SNR_{MIN})}{SNR_{MAX} - SNR_{MIN}} \tag{1316}$$

where $r_{MIN} \leq r_{SID} \leq r_{MAX}$. The values r_{MIN}, SNR_{MIN}, r_{MAX} and SNR_{MAX} are selected as follows:

```
r_MIN = 8
SNR_MIN = 36 dB
r_MAX = 50
SNR_MAX = 51 dB.
```

Thus, the adaptive rate is limited to between 8 and 50 frames and is updated in every inactive signal frame. If the number of consecutive NO_DATA frames is equal to or greater than the current value of r_{SID}, the next inactive signal frame is denoted as a SID frame. There is one exception to this rule.

    a) SID frame sent due to detection of abrupt changes in the spectral characteristics of background noise, as described in Section 5.6.1.2.

After the rate r_{SID} is determined, a variation of the long-term energy of the inactive signal is calculated and subjected to a fixed threshold. This is performed in every NO_DATA frame after the preamble period. That is, if the following variation holds

$$10.0 \cdot \log\left(\frac{\overline{E}_{inact}}{\overline{E}_{lastSID}}\right) < -4.0 \qquad (1317)$$

the long-term energy of inactive speech is reset to $\overline{E}_{inact} = E_f$, where $\overline{E}_{lastSID}$ is the long-term energy of the inactive signal updated in every SID frame.

5.6.1.2   Spectral tilt based SID transmission

Adaptive SID update rate that relies only on fluctuations in SNR as described in Section 5.6.1.1 may sometimes fail to detect significant changes in the background noise characteristics. In some cases, inactive frames that are perceptually different will have similar energy characteristics (typically encoded as gain values in the SID frames). Although background noise in a street (street noise) may have an energy distribution over time that is similar to that of background noise in a crowded space (babble noise), for example, these two types of noise will usually be perceived very differently by a listener. Spectral tilt is a good measure to capture such changes in the background noise characteristics.

It would be beneficial for a DTX/CNG scheme to track such perceptual changes in the background noise, apart from tracking the SNR as described in Section 5.6.1.1. Hence a scheme to detect a sudden change in spectral tilt of the background noise and trigger a new SID frame indicating the parameters of the new background noise is employed in the encoder.

To ensure that there is no affect from an active talk spurt to the computation of spectral tilt of the background noise, this computation is performed in every inactive frame after 5 consecutive inactive frames that immediately follow an active talk spurt.

Linear prediction coefficients (LPCs) are derived using performing Linear prediction analysis (Section 5.1.5.1 – Section 5.1.5.3) on the current input frame with background noise and no active speech. LPCs are then converted to reflection coefficients (RCs) as follows using a backwards Levinson Durbin recursion. For a given Nth order LPC vector $LPC_N = [1, a_{N1}, ..., a_{NN}]$, the Nth reflection coefficient value is derived using the formula $refl(N) = -a_{NN}$, it is then possible to calculate the lower order LPC vectors $LPC_{N-1}, ..., LPC_1$ using the following recursion

$$refl(p) = a_{p,p}$$
$$F = 1 - refl(p)^2, \, p = N, N-1, ..., 2 \qquad (1318)$$
$$a_{p-1,m} = \frac{a_{p,m}}{F} - \frac{refl(p)a_{p,p-m}}{F}, 1 <= m < p$$

which yields the reflection coefficient vector $[refl(1), refl(2), ...refl(N)]$. The spectral tilt of background noise is indicated by the first reflection coefficient $refl(1)$. A smoothened running average of the spectral tilt of background noise in K^{th} inactive frame $T_{avg,K}$ is computed using a first order IIR filter as follows.

$$T_{avg,K} = 0.8 * T_{avg,K-1} + 0.2 * refl(1) \qquad (1319)$$

The running average differences from frame to frame are accumulated in $\Delta_{sum,K}$ during each during each successive inactive frame K as follows:

$$\Delta_{sum,K} = \Delta_{sum,K-1} + (T_{avg,K} - T_{avg,K-1}) \qquad (1320)$$

Absolute value of this delta-sum parameter $\Delta_{sum,K}$ is compared against a set threshold of 0.2. If this threshold is exceeded during an inactive frame indicating a change in spectral tilt characteristics of the background noise, and if the number of consecutive NO_DATA frames is equal to or greater than 8, a new SID frame is transmitted regardless of the

current value of the adaptive SID rate parameter r_{SID}. At this point, parameters $T_{avg,K}$ and $\Delta_{sum,K}$ are also reset to zero to start a fresh computation of spectral tilt of the background noise that follows this SID frame.

5.6.1.3   CNG selector

The CNG selector chooses one of the two CNG modes (FD_CNG or LP_CNG) for generating comfort noise. In case AMR-WB IO mode is used, LP-CNG is always selected. Otherwise, the decision is based on the energy ratio between a high and a low frequency range of the background noise signal and the bandwidth of the signal.

Noise energy estimates for the low frequency range up to 1270 Hz and for the two highest critical bands are estimated

by
$$N_{\text{lp}} = \frac{\sum_{i=0}^{9} N_{CB}(i)}{10} , \qquad (1321)$$

$$N_{\text{hp}} = \frac{N_{CB}(18) + N_{CB}(19)}{2} , \qquad (1322)$$

except for narrowband signals, where the lowest band is ignored and the highest bands are lower:

$$N_{\text{lp}} = \frac{\sum_{i=1}^{9} N_{CB}(i)}{9} , \qquad (1323)$$

$$N_{\text{hp}} = \frac{N_{CB}(15) + N_{CB}(16)}{2} . \qquad (1324)$$

N_{CB} is the background noise energy per critical band as described in subclause 5.1.11.1. Both values N_{lp} and N_{hp} are used to calculate the spectral tilt of the background noise energies and update the memory for N_{tilt} :

$$N_{\text{tilt}} = 0.9 \cdot N_{tilt} + 0.1 \cdot \frac{N_{lp}}{N_{hp}} \qquad (1325)$$

Depending on the previous CNG mode, input signal bandwidth (`input_bwidth`) as detected by the bandwidth detection module (subclause 5.1.6) and N_{tilt} the CNG mode is changed if the current frame is active and at least the past 20 frames were active and one of the following cases applies:

```
if (cng_mode == LP_CNG &&
((  input_bwidth == NB && Ntilt > 9.f) ||
    ( input_bwidth  > NB && Ntilt > 45.f)))
{
    cng_mode = FD_CNG;
}
else
    if ( cng_mode == FD_CNG &&
        (( input_bwidth == NB && Ntilt < 2.f) ||
( input_bwidth  > NB && Ntilt < 10.f)))
    {
cng_mode = LP_CNG;
    }
```

5.6.2   Encoding for LP-CNG

This section describes the operation in LP-CNG. Similar to the default operation, the LP-CNG also operates on the split-band basis. In WB/NB operation, only the LP parameters for low-band signal are analyzed and encoded. In SWB/FB operation, besides the LP analysis for the low-band signal, the high-band signal is analyzed and encoded separately as a kind of bandwidth extension. The LP parameters for low-band analysis include: the low-band excitation energy, the low-band signal spectrum and the low-band excitation envelope. The high-band analysis only involves one

parameter which is the high-band energy. The 1 CNG type bit (see subclause 7.2) is set to "0" for LP-CNG and transmitted in each SID frame.

5.6.2.1 LP-CNG CN parameters estimation

The CN parameters to be encoded into a LP-CNG SID frame are calculated over a certain period, which is called the CN averaging period. These parameters give information about the level and the spectrum of the background noise. The CN averaging period, N_{CN}, is equal to the number of consecutive frames including the current SID frame and its preceding NO_DATA frames, upper-limited by the value of 8 consecutive frames. It is a variable value depending on the current SID transmission rate. In particular, the first SID frame immediately after an active signal burst always uses the value $N_{CN} = 1$. The LP-CNG can generate two different types of SID frame – the WB SID frame, containing low-band (WB) only CN parameters, and the SWB SID frame, containing both low-band and high-band (SHB) CN parameters. One bit is encoded into the LP-CNG SID frame to indicate the bandwidth type of the SID frame, where "0" indicates WB SID and "1" indicates SWB SID. Only WB SID frames are transmitted when operating in NB/WB mode. In SWB/FB operation, the SWB SID frames are not always transmitted but WB SID frames can also be transmitted between two adjacent SWB SID updates. This means the high-band CN parameters are not updated at the decoder with the same rate the low-band CN parameters will be updated. Details in SWB/FB operation will be described in subclause 5.6.2.1.8. The bit allocation for the CN parameters in the respect WB and SWB SID frames are described in subclause 7.2.

5.6.2.1.1 LP-CNG Hangover analysis period determination

To enable high quality comfort noise synthesis on the receiving side, the encoder sends a three bit counter value $burstho_{tx}$ to the decoder. The transmitted value is derived from the $burstho_{cnt}$ counter, as:

$$burstho_{tx} = \min(burstho_{cnt}, 7) \tag{1326}$$

where $burstho_{cnt}$ is counter of the consecutive frames without any primary SAD active decisions where the DTX SAD flag f_{SAD_DTX}, as described in subclause 5.1.12.8, is set to 1. These hangover frames without primary SAD active decisions are deemed to be relevant for comfort noise analysis. For DTX operation, the $burstho_{cnt}$ counter is incremented in actively encoded speech segments whenever the primary SAD flag is set to 0 and the DTX SAD flag is set to 1 and it is set to zero whenever this is not the case.

5.6.2.1.2 LP-CNG filter parameters evaluation for low-band signal

In the DTX/ LP-CNG operation, the LP filter coefficients are quantized in the LSF domain, which is the same as in the default option. However, in the case of DTX/CNG operation, the LP filter coefficients are not interpolated within the frame. From the LP analysis, which is described in subclause 5.1.9, only the end-frame LSP vector, $q_{end,i}$, is used for quantization purposes in SID frames.

The end-frame LSP vector is not quantized directly. Instead, an averaged end-frame LSP vector is calculated over the CN averaging period which is then converted to an LSF vector and quantized. Not all end-frame LSP vectors in the CN averaging period are used for averaging, but two outlier LSP vectors are removed. The two outliers are found over the CN averaging period, as the two LSP vectors representing the lowest spectral entropy. This is to mitigate the possible corruption to the averaged LSP vector from interfering background frames, assuming that interfering background frames are usually more structural in their spectrum (leading to lower spectral entropy) than normal background noise frames. A parameter $C_{lsf}(i)$ which can reflect such a spectral entropy is thus calculated for each end-frame LSP vector over the CN averaging period. Each end-frame LSP vector is first converted to its LSF representation and $C_{lsf}(i)$ is calculated as

$$C_{lsf}(i) = \sum_{k=0}^{M-1} (\Delta_i(k) - \Delta)^2 \tag{1327}$$

where M equals 16 which is the order of the LP filter, Δ denotes the bandwidth of the partition if the signal bandwidth is divided by M equally spaced LSF coefficients, that is, $\Delta = fs/(M+1)$, where fs is the bandwidth of the signal, fs equals 6400 for 12.8kHz sampling rate core and 8000 for 16kHz sampling rate core, $\Delta_i(k)$ denotes the length of the kth partition divided by LSF coefficients of the ith LSP vector over the signal bandwidth, that is

$$\Delta_i(k) = \begin{cases} lsf_i(0) & for\ k = 0 \\ fs - lsf_i(M-1) & for\ k = M \\ lsf_i(k) - lsf_i(k-1) & for\ k = 1,2,...,M-1 \end{cases} \tag{1328}$$

where $lsf_i(k)$ is the kth LSF coefficient of the ith LSP vector, fs is either 6400 or 8000 depending on the sampling rate of the core. A more structured spectrum will result in a $C_{lsf}(i)$ having higher value representing lower spectral entropy. So the two LSP vectors resulting in the two maximum $C_{lsf}(i)$ over the CN averaging period is found as the two outliers. The averaged LSP vector is then calculated as

$$q_{avg}(k) = \frac{1}{N-2} \cdot \sum_{i=0,i\neq o1,i\neq o2}^{N-1} q_{end,i}(k) \tag{1329}$$

where N is the length of CN averaging period, $o1$, $o2$ denotes the index of the two LSP outliers. The outlier-removed LSP vector average q_{avg} is considered the best representation of the short-term spectral envelope of the background noise. It is converted to the LSF representation, quantized (see subclause 5.6.2.1.3) and transmitted in the SID frame.

5.6.2.1.3　　　LP-CNG CNG-LSF quantization for low-band signal

The quantization of the LSF vector follows the procedures used for the LSF vector within the ACELP block. They are described in subclause 5.2.2.1. The quantization is done with a two stage quantizer. The first stage consists of a non-predictive, non-structured, optimized VQ codebook. The second stage consists of a multiple scale lattice vector quantizer whose structure and search procedure are detailed in subclause 5.2.2.1.4. The number of lattice structures from the second stage corresponds to the number of codevectors from the first stage such that if a particular codevector is selected in the first stage, its corresponding lattice structure is used in the second stage. A lattice multiple scale lattice structure corresponds to a set of 6 numbers specifying the number of leader classes in each of the 6 lattice truncations and 6 numbers specifying the scale values for the same truncations. There are three lattice truncations that define the codebook for the first 8-dimensional LSF subvector and three lattice truncations defining the codebook for the second 8-dimensional LSF subvector. In addition, a 16-dimensional vector defines for each multiple scale lattice structure a normalization values vector, $\sigma_i, i = 1, M$.

The codebook from the first stage uses 4 bits, and each lattice structure from the second stage is defined for 25 bits. Consequently a total of 29 bits that are used for the quantization of the LSF vector. With the above described structure, the table ROM used for storing the CNG LSF codebook data covering 29 bits has 1.408kBytes.

The search in the first stage codebook is done taking into account the value of the last component of the 16 dimensional LSF vector. Based on this value only part of the first stage codebook is searched. If the last LSF vector component is larger than 6350 then the search is done only for the first 6 codevectors of the first stage and the LSF vector corresponds to internal sampling frequency of 16kHz, otherwise the search is performed within the last 10 codevectors of the first stage that correspond to the internal sampling rate of 12.8kHz. At the second stage, prior to quantization with the lattice structure, based on the selected first stage codevector some components of the LSF vector are permuted like specified by the following table:

Table 151: LSF vector component permutation

First stage codevector index	Permutations
0	(6,11), (7,15)
1	(6,15)
2	(5,8), (7,15)
3	(7,10)
7	(0,9), (7,10)
9	(7,15)
12	(6,10), (7,11)
13	(6,10), (7,12)
14	(6,10), (7,12)
15	(6,10), (7,12)

A permutation defined as (6,11) signifies that the 6th component, numbered starting from 0, is replaced by the 11th one and reciprocally. The permutations are performed between the two groups or subvectors, i.e the first index in the permutation is from the first half of the codevector and the second one from the second half or subvector. The permutations are performed only when one of the first stage codevectors whose index is mentioned in the previous table is obtained at the first stage. The resulting vector is component wise multiplied with the inverse of the corresponding σ vector and quantized with the corresponding multiple scale lattice structure. After quantization the components of the obtained second stage multiple scale lattice codevector are permuted back, and added to the first stage codevector in order to obtain the quantized LSF vector. 1 bit indicating the core sampling rate is transmitted in each SID frame. This bit signals the decoder the sampling domain on which the quantized LSF vector is. The bit is set to "0" for 12.8 kHz sampling rate and ''1' for 16 kHz sampling rate.

5.6.2.1.4   LP-CNG synthesis filter computation for local CNG synthesis

A smoothed LSP vector is used in every inactive frame to obtain the LP synthesis filter $\hat{A}(Z)$. The quantized LSF vector is converted back to the LSP domain. The smoothed LSP vector is updated by the last quantized LSP vector by means of an AR low-pass filter in each inactive frame except the first SID frame after an active burst. That is

$$\overline{\hat{q}}^{[0]} = \alpha \cdot \overline{\hat{q}}^{[-1]} + (1-\alpha) \cdot \hat{q}_{avg} \qquad (1330)$$

where $\overline{\hat{q}}^{[0]}, \overline{\hat{q}}^{[-1]}$ denote respectively the smoothed LSP vector at the current and the previous frame, $\hat{q}_{avg}$ denotes the last quantized LSP vector, $\alpha = 0.9$ is a smoothing factor. An additional constraint is applied to the inactive frames after the first SID frame of an inactive segment and before the second SID frame that the update to the smoothed LSP vector described above is suspended if the last SID excitation energy is an outlier and sufficient hangover frames are contained in the last active burst, that is, if $\hat{\overline{E}}^{[-1]} \geq 1.5 E_{CN}$ and $m \geq 3$, where $\hat{\overline{E}}^{[-1]}$ denotes the quantized excitation energy in the last SID frame, as calculated in equation (1339), m denotes the number of entries in enr_{hist} used for $enr_{hist-ave-weighted}$ calculation as described in subclause 6.7.2.1.2 and E_{CN} is the smoothed quantized excitation energy, further described in subclause 5.6.2.1.6. For the first SID frame after an active burst, the smoothed LSP vector is updated depending on whether the frame is an outlier in either energy or spectrum and whether there are past CN parameters to analyze in the CNG analysis buffer as described in subclause 6.7.2.1.2. If the step update flag f_{step} is set to 1 or there were no past CN-parameters to analyse in the CNG analysis buffer, the smoothed LSP vector is initialized to the quantized LSP vector of the current SID frame. Otherwise, if step update is not allowed and there are past CN parameters to analyze in the CNG analysis buffer, the overall and the maximum individual spectral distortion between the quantized LSP vector of the current SID frame and the average LSP vector, $ho_{lsp-ave-weighted}$, calculated over hangover frames in subclause 6.7.2.1.2 are calculated.

$$D_{q,sum} = \sum_{k=0}^{M-1} \left| ho_{lsp-ave-weighted}(k) - \hat{q}_{avg}(k) \right| \qquad (1331)$$

$$D_{q,\max} = MAX \left[ho_{lsp-ave-weighted}(k) - \hat{q}_{avg}(k) \right], \qquad k = 0,...,15 \qquad (1332)$$

where $D_{q,sum}$ is the overall spectral distance, $D_{q,\max}$ is the maximum spectral distance, $ho_{lsp-ave-weighted}(k)$ is the average LSP vector calculated over hangover frames, $\hat{q}_{avg}(k)$ is the quantized LSP vector of the current SID frame. If $ho_{lsp-ave-weighted}(k)$ and $\hat{q}_{avg}(k)$ are deviating to each other, that is, if $D_{q,sum} > 0.4$ or $D_{q,\max} > 0.1$, the quantized LSP vector of the current SID frame $\hat{q}_{avg}(k)$, is considered an outlier and the smoothed LSP vector is initialized to the average LSP vector calculated over hangover $ho_{lsp-ave-weighted}(k)$. Otherwise, the smoothed LSP vector is initialized by

$$\overline{\hat{q}}(k) = 0.8 \cdot ho_{lsp-ave-weighted}(k) + 0.2 \cdot \hat{q}_{avg}(k), \qquad k = 0,...,15 \qquad (1333)$$

The smoothed LSP vector $\overline{\hat{q}}$ is initially set to the quantized end-frame LSP vector from the previous frame, $\hat{q}_{end}^{[-1]}$, when the first SID frame is processed at the encoder. The step update flag f_{step} is set in each inactive frame by measuring the energy step between the instant energy and the long-term energy. For the first inactive frame after an

active signal period, the flag f_{step} is additionally set if there are past CN-parameters and the most recent energy value in enr_{hist} is more than four times larger than the smoothed quantized excitation energy E_{CN}. Finally, the smoothed LSP vector is converted to LP coefficients to obtain a smoothed LP synthesis filter, $\hat{A}(Z)$, which is used in the local CNG synthesis.

5.6.2.1.5   LP-CNG energy calculation and quantization

The excitation energy in the current frame is computed for each inactive frame according to the following equation:

$$E_{log} = \log_2\left(\frac{1}{L}\sum_{n=0}^{L-1} r^2(n)\right) \tag{1334}$$

where $r(n)$ is the LP residual signal, calculated by filtering the pre-emphasized inactive input signal, $s_{pre}(n)$, through the filter $\hat{A}(z)$, L =256 or 320 depending on the sampling rate of the core. Then a weighted average energy is computed over the whole CN averaging period by

$$\overline{E}_{log} = \frac{\displaystyle\sum_{n=-N_{CN}+1}^{0} w(n)E_{log}^{[n]}}{\displaystyle\sum_{n=-N_{CN}+1}^{0} w(n)} - E_{offset} \tag{1335}$$

where the weights are defined as $w(n)$ = [0.2, 0.16, 0.128, 0.1024, 0.08192, 0.065536, 0.0524288, 0.01048576], and E_{offset} is an energy offset value which is set to 0 for input bandwidth = NB, to 1.5 for input bandwidth greater than WB, and for signals of bandwidth = WB, the energy offset value is chosen from an energy attenuation table depending on the latest bitrate used for actively encoded frames R_{latest_active} as defined by Table 151a. The energy offset is only updated in the first SID frame after an active signal period if two criteria are both fulfilled. The first criterion is satisfied if AMR-WB IO mode is used or the bandwidth=WB. The second criterion is met if the number of consecutive active frames in the latest active signal segment was at least $MIN\_ACT\_CNG\_UPD$ = 20 number of frames or if the current SID is the very first encoded SID frame. The superscript [n] denotes a particular frame, e.g., [0] is the current frame.

Table 151a: Energy offset selection for LP-CNG

Latest active bitrate [kbps]	E_{offset}
$R_{latest\_active} \leq 7.2$	1.7938412
$7.2 < R_{latest\_active} \leq 8.0$	1.3952098
$8.0 < R_{latest\_active} \leq 9.6$	1.0962363
$9.6 < R_{latest\_active} \leq 13.2$	0.9965784
$R_{latest\_active} > 13.2$	0.9965784

The weighted average energy is then quantized using a 7-bit arithmetic quantizer. The integer quantization index in the current SID frame is found using the relation

$$I = \left\lfloor (\overline{E}_{log} + 2)\cdot\Delta \right\rfloor \tag{1336}$$

where $\Delta = 5.25$ is the quantization step. The quantization index is limited to [0, 127]. The quantization index is further limited not to increase by more than one from the value of the previous frame if the previous frame was also an inactive frame. An exception is that if the step update flag f_{step} is set to 1, then the quantization index is allowed to increase more than one from the value of the previous frame using the relation

$$I_q = I_q^{[-1]} + \left\lfloor 0.85 \cdot (I - I_q^{[-1]}) \right\rfloor \tag{1337}$$

where I_q denotes the final quantization index transmitted in each SID frame, $I_q^{[-1]}$ denotes the quantization index transmitted in the previous SID frame, I is the quantization index calculated in equation (1336). The quantized value of energy is used further in the local CNG synthesis and is found by

$$\hat{\hat{E}}_{\log} = \frac{I_q}{\Delta} - 2 \tag{1338}$$

which is converted to the linear domain by

$$\hat{\hat{E}} = 2^{\hat{\hat{E}}_{\log}} \tag{1339}$$

5.6.2.1.6 LP-CNG energy smoothing for local CNG synthesis

The quantized excitation energy, $\hat{\hat{E}}$, calculated in equation (1339) is not used directly in the local CNG synthesis. Instead, a smoothed quantized excitation energy, E_{CN}, is computed. The smoothed quantized excitation energy is updated in every inactive frame in a general form of

$$E_{CN} = \alpha E_{CN}^{[-1]} + (1 - \alpha)\hat{\hat{E}} \tag{1340}$$

where superscript [-1] demotes the value from the previous frame, $\hat{\hat{E}}$ is the quantized energy in the SID frame, calculated in equation (1339), α is the smoothing factor controlling the update rate. Variable update rates are utilized. For the first inactive frame after an active signal period, if there is no step update flag f_{step} set to 1 in the latest two SID frames, $\alpha = 0.8$ if the number of preceding hangover frames is less than 3 or $\hat{\hat{E}} < 1.5 E_{CN}^{[-1]}$, otherwise, $\alpha = 0.95$. Otherwise, if step update flag f_{step} is set to 1 in either of the latest two SID frames, $\alpha = 0$, i.e. E_{CN} is set directly to $\hat{\hat{E}}$. For consequent frames, if the step update flag f_{step} at the latest SID frame is not set to 1, $\alpha = 0.8$. Otherwise, $\alpha = 0$, i.e. E_{CN} is set directly to $\hat{\hat{E}}$. For the first inactive frame after an active signal period, before the smoothed quantized excitation energy E_{CN} is updated by $\hat{\hat{E}}$, the value of E_{CN} from the previous frame, that is $E_{CN}^{[-1]}$, is initialized to $enr_{hist-ave-weighted}$ which is the age weighted average excitation energy of the DTX hangover frames calculated in subclause 6.7.2.1.2, if step update flag f_{step} is not set to 1 and there are past CN parameters to analyse in the CNG analysis buffers. E_{CN} is initially set equal to $\hat{\hat{E}}$.

5.6.2.1.7 LP-CNG LF-BOOST determination and quantization

While the quantized LSF spectrum generally estimates well the spectrum of most background noises, it is found less sufficient for noises which have strong low frequency component for example the car noise. To compensate the missing low frequency component, the spectral envelope in the low frequency portion of the LP residual signal is quantized and transmitted in the SID frame. Note that the quantized residual spectral envelope is only transmitted in WB SID frame.

The LP residual signal $r(n)$ calculated in subclause 5.6.2.1.5 is first attenuated by multiplying an attenuation factor att for all input bandwidth except NB. The attenuation factor is calculated as

$$att = \frac{3}{3 + 4\,flr} \tag{1341}$$

where $flr = 0.6$ if the bandwidth is not WB or the latest bitrate used for actively encoded frames R_{latest_active} is larger than 16.4 kbps. Otherwise flr is determined from a hangover attenuation floor table as defined in table 35b. The attenuation factor att is finally lower limited to flr. Then a FFT is used to obtain the frequency representation of the LP residual signal. The spectral envelope to be quantized is the energies of the first 20 FFT bins (excluding the DC bin) and is calculated as

$$E_{env}(k) = \frac{2}{L_{FFT}}\left(X_R^2(k) + X_I^2(k)\right), \qquad k = 0,...,19 \tag{1342}$$

where $X_R(k)$ and $X_I(k)$ are, respectively, the real and the imaginary parts of the k-th frequency bin as outputted by the FFT, $L_{FFT} = 256$ is the size of FFT analysis. The low frequency spectral envelope of the LP residual is not quantized directly. Instead, an averaged spectral envelope is calculated over the CN averaging period. The averaging is similar to the process in subclause 5.6.2.1.2 that the spectral envelopes of the two outliers identified in subclause 5.6.2.1.2 are removed from the averaging. The averaged envelope is calculated as

$$Env_{avg}(k) = \frac{1}{N-2} \cdot \sum_{i=0, i \neq o1, i \neq o2}^{N-1} E_{env,i}(k), \qquad k = 0,...,19 \tag{1343}$$

where N is the length of CN averaging period, $E_{env,i}(k)$ denotes the low frequency envelope of the i-th frame in the CN averaging period, $o1$, $o2$ denotes the index of the two outliers. Prior to quantization, the dynamic range of the averaged envelope is reduced by subtracting 2 times of the quantized average excitation energy from the averaged envelope and bounded to non-negative value.

$$Env'_{avg}(k) = MAX\left[Env_{avg}(k) - 2\hat{\bar{E}}, 0\right], \qquad k = 0,...,19 \tag{1344}$$

The dynamic range reduced envelope $Env'_{avg}(k)$ is then converted to log domain

$$Env'_{\log}(k) = \log_2 Env'_{avg}(k), \qquad k = 0,...,19 \tag{1345}$$

where E_{offset} is the energy offset value calculated in subclause 5.6.2.1.5 and $Env'_{\log}(k)$ are bounded to non-negative value. A distance vector is calculated as

$$D(k) = \hat{E}_{exc} - Env'_{\log}(k), \qquad k = 0,...,19 \tag{1346}$$

where $\hat{E}_{exc}$ is the quantized total excitation energy calculated as

$$\hat{E}_{exc} = \log_2\left(L_{exc} \cdot \hat{\bar{E}}\right) \tag{1347}$$

where $L_{exc} = 256$ is the length of the excitation. The distance vector $D(k)$ is quantized by a vector quantization. The codeword having the minimum prediction error is found by a direct search in the codebook and the index of the codeword is transmitted in the WB SID frame. The quantized low frequency envelope is recovered and used in the local CNG synthesis.

5.6.2.1.8   LP-CNG high band analysis and quantization

To enable high perceptual quality in the inactive portions of speech on the decoder side, during SWB mode DTX/LP-CNG operation of the codec, the high band noise signal (6.4 - 14.4 kHz or 8 - 16 kHz depending on the core sampling rate) is analyzed and quantized. However, this is being done without transmitting any extra parameters from the encoder to decoder to model the high-band spectral characteristics of the inactive frames. Instead, the high band spectrum of the comfort noise is modelled purely in the decoder side. Only the energy of high band signal is quantized and transmitted in the SID frame.

The average energy of the high band signal is first calculated

$$\bar{E}_h = \frac{1}{L} \cdot \sum_{i=0}^{L-1} s_h(i)^2 \tag{1348}$$

where $s_h(i)$ is the high band signal, $L = 320$ is the length of high band signal. The log average energy of high band signal is calculated and to which an attenuation of 6.5 dB is applied

$$\overline{E}'_h = 10 \cdot \log_{10} \overline{E}_h - 6.5 \tag{1349}$$

The attenuated log energy $\overline{E}'_h$ is smoothed by an AR filtering as

$$\overline{\overline{E}}_h = 0.75\overline{\overline{E}}_h^{[-1]} + 0.25\overline{E}'_h \tag{1350}$$

where $\overline{\overline{E}}_h$ is the smoothed high band log average energy, $\overline{\overline{E}}_h^{[-1]}$ denotes the smoothed log average energy in the previous frame. The average energy of the low band signal is also calculated

$$\overline{E}_l = \frac{1}{L} \cdot \sum_{i=0}^{L-1} \hat{s}_l(i)^2 \tag{1351}$$

where $\hat{s}_l(i)$ is the synthesized low band signal as described in subclause 5.6.2.2, L is the length of the synthesized low band signal. The log average energy of the low band signal is calculated

$$\overline{E}'_l = 10 \cdot \log_{10} \overline{E}_l \tag{1352}$$

The log average energy of the low band signal $\overline{E}'_l$ is also smoothed by an AR filtering.

$$\overline{\overline{E}}_l = 0.1\overline{\overline{E}}_l^{[-1]} + 0.9\overline{E}'_l \tag{1353}$$

where $\overline{\overline{E}}_l$ is the smoothed low band log average energy, $\overline{\overline{E}}_l^{[-1]}$ denotes the smoothed log average energy in the previous frame. Step update to the smoothed low band and high band log average energy is allowed. If the low band log average energy $\overline{E}'_l$ of the current frame is deviating from the smoothed low band log average energy of the previous frame $\overline{\overline{E}}_l^{[-1]}$ by more than 12dB, a step update flag f_{step} is set to 1 indicating the permission of step update, otherwise is set to 0. If f_{step} is set to 1, the smoothed low band log average energy and the smoothed high band log average energy are respectively set to the low band log average energy $\overline{E}'_l$ and the high band log average energy $\overline{E}'_h$. The high band parameter, i.e. the energy of the high band signal is not quantized and transmitted in every SID frame. Instead, SWB SID frame which contains both the low band and high band parameters is only transmitted when the energy relationship (the energy ratio) between the low band and high band signals at the current frame is deviating from that relationship at previous SWB SID frame by more than 3dB. This can be described as, when a SID frame is about to be transmitted, if $\left| \left(\overline{\overline{E}}_l - \overline{\overline{E}}_h\right) - \left(\overline{\overline{E}}_l^{[-1]} - \overline{\overline{E}}_h^{[-1]}\right) \right| > 3$, then the high band parameter is transmitted in the SID frame. Besides, following conditions can also trigger the transmission of SWB SID frame, including: the first SID frame immediately after active frames, the SID frame which is within high band analysis initialization period, the SID frame before which there is no active and no SWB SID frame in the 100 preceding frames when operating in SWB mode or above, the SID frame where there is bandwidth switching between WB and SWB. If SWB SID transmission is not triggerd at an instance of SID frame update, the WB SID frame will be transmitted instead.

In each SWB SID frame, the smoothed high band log average energy $\overline{\overline{E}}_h$ is quantized and transmitted. The $\overline{\overline{E}}_h$ is first converted to $\log_2$ domain as

$$\overline{\overline{E}}_{h2} = \frac{0.1\overline{\overline{E}}_h}{\log_{10} 2} \tag{1354}$$

Then a 4-bit arithmetic quantizer is used for quantizing $\overline{\overline{E}}_{h2}$. The integer quantization index is found by

$$I = \left\lfloor (\overline{\overline{E}}_{h2} + 6) \cdot \Delta + 0.5 \right\rfloor \tag{1355}$$

where $\Delta = 0.9$ is the quantization step. The quantization index I is bounded to [0, 15].

5.6.2.2 LP-CNG local CNG synthesis

The local CNG synthesis is performed at the encoder for low-band signal in order to update the filters, the adaptive codebook memories, and to guide the high-band analysis and the DTX hangover control (see subclause 5.1.12.8). The local CNG is performed by filtering a scaled excitation signal through a smoothed LP synthesis filter. The scaled excitation, $e'(n)$, is a combination of a random excitation and an excitation representing the low frequency spectral details of the excitation signal. For the generation of $e'(n)$, see subclause 6.7.2.1.5. For the computation of the smoothed LP synthesis filter, see subclause 5.6.2.1.4.

5.6.2.3 LP-CNG CNG Memory update

When an inactive signal frame is encoded, the following updates are carried out:

– MA memory of the ISF quantizer is set to zero;

– AR memory of the ISF quantizer is set to its mean values (UC mode, WB case);

– synthesis excitation spectrum tilt is set to zero;

– weighting filter denominator memory is set to zero;

– gain of pitch clipping memory is set to initial values;

– open-loop pitch estimator parameters are set to zero;

– per-bin NR last critical band is set to zero (the whole spectrum subtraction);

– noise enhancer memory is set to zero;

– phase dispersion memory is set to zero;

– previous pitch gains are all set to zero;

– previous codebook gain is set to zero;

– voicing factors used by bandwidth extension are all set to 1;

– active frame counter is set to zero;

– bass post-filter is tuned off;

– floating point pitch for each subframe is set to the subframe length;

– class of last received good frame for FEC is set to UNVOICED_CLAS;

– synthesis filter memories are updated.

5.6.3 Encoding for FD-CNG

To be able to produce an artificial noise resembling the actual input background noise in terms of spectro-temporal characteristics, the FD-CNG makes use of a noise estimation algorithm to track the energy of the background noise present at the encoder input. The noise estimates are then transmitted as parameters in the form of SID frames to update the amplitude of the random sequences generated in each frequency band at the decoder side during inactive phases. Note, however, that the noise estimation is carried out continuously on every frame, i.e., regardless of the speech activity. Therefore, it can deliver some meaningful information about the noise spectrum at any time, in particular at the very beginning of a speech pause.

The FD-CNG noise estimator relies on a hybrid spectral analysis approach. Low frequencies corresponding to the core bandwidth are covered by a high-resolution FFT analysis, whereas the remaining higher frequencies are captured by the CLDFB which exhibits a significantly lower spectral resolution of 400Hz.

The size of an SID frame is however very limited in practice. To reduce the number of parameters describing the background noise, the input energies are averaged among groups of spectral bands called partitions in the sequel.

5.6.3.1 Spectral partition energies

The partition energies are computed separately for the FFT and CLDFB bands. The $L_{\text{part}}^{[\text{FFT}]}$ energies corresponding to the FFT partitions and the $L_{\text{part}}^{[\text{CLDFB}]}$ energies corresponding to the CLDFB partitions are then concatenated into a single array $E_{\text{FD-CNG}}$ of size $L_{\text{part}} = L_{\text{part}}^{[\text{FFT}]} + L_{\text{part}}^{[\text{CLDFB}]}$ which will serve as input to the noise estimator described in subclause 5.6.3.2.

5.6.3.1.1 Computation of the FFT partition energies

Partition energies for the frequencies covering the core bandwidth are obtained as

$$E_{\text{FD-CNG}}(i) = \frac{E_{\text{CB}}^{[0]}(i) + E_{\text{CB}}^{[1]}(i)}{2} H_{\text{de-emph}}(i) \qquad\qquad i = 0,...,L_{\text{part}}^{[\text{FFT}]}-1 , \qquad (1356)$$

where $E_{\text{CB}}^{[0]}(i)$ and $E_{\text{CB}}^{[1]}(i)$ are the average energies in critical band i for the first and second analysis windows, respectively, as explained in subclause 5.1.5.2. The number of FFT partitions $L_{\text{part}}^{[\text{FFT}]}$ depends on the sampling rate sr_{HP} of the input signal, as show in Table 133. The de-emphasis spectral weights $H_{\text{de-emph}}(i)$ are used to compensate for the high-pass filter described in subclause 5.1.4 and are defined as

$$\left\{H_{\text{de-emph}}(0),...,H_{\text{de-emph}}(L_{\text{part}}^{[\text{FFT}]}-1)\right\} = \{\, 9.7461, 9.5182, 9.0262, 8.3493, 7.5764, 6.7838, 5.8377,$$
$$4.8502, 4.0346, 3.2788, 2.6283, 2.0920, 1.6304, 1.2850, \qquad (1357)$$
$$1.0108, 0.7916, 0.6268, 0.5011, 0.4119, 0.3637 \,\}.$$

5.6.3.1.2 Computation of the CLDFB partition energies

The partition energies for frequencies above the core bandwidth are computed as

$$E_{\text{FD-CNG}}(i) = \frac{1}{16} \frac{1}{8(f_{\text{scl}})^2} \frac{\sum_{j=j_{\min}(i)}^{j_{\max}(i)} \overline{E}_{\text{C}}(j)}{j_{\max}(i) - j_{\min}(i) + 1} \qquad\qquad i = L_{\text{part}}^{[\text{FFT}]},...,L_{\text{part}}^{[\text{FFT}]} + L_{\text{part}}^{[\text{CLDFB}]}-1 , \qquad (1358)$$

where $j_{\min}(i)$ and $j_{\max}(i)$ are the indices of the first and last CLDFB bands in the i-th partition, respectively, $\overline{E}_{\text{C}}(j)$ is the total energy of the j-th CLDFB band (see subclause 5.1.2.2), and f_{scl} is a scaling factor computed in subclause 5.1.6.1. The constant 16 refers to the number of time slots in the CLDFB. The number of CLDFB partitions $L_{\text{part}}^{[\text{CLDFB}]}$ depends on the configuration used, as described in the next subclause.

5.6.3.1.3 FD-CNG configurations

The following table lists the number of partitions and their upper boundaries for the different FD-CNG configurations at the encoder, as a function of the input sampling rate sr_{HP} .

Table 152: Configurations of the FD-CNG noise estimation at the encoder

sr_{HP} (kHz)	$L_{\mathrm{part}}^{[\mathrm{FFT}]}$	$L_{\mathrm{part}}^{[\mathrm{CLDFB}]}$	$f_{\max}(i)$, $i = 0,...,L_{\mathrm{part}}^{[\mathrm{FFT}]}-1$ (Hz)	$f_{\max}(i)$, $i = L_{\mathrm{part}}^{[\mathrm{FFT}]},...,L_{\mathrm{part}}-1$ (Hz)
8	17	0	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700	×
16	20	1	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6350	8000
32/48	20	4	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6350	8000, 10000, 12000, 16000

For each partition $i = 0,...,L_{\mathrm{part}}-1$, $f_{\max}(i)$ corresponds to the frequency of the last band in the i-th partition. The indices $j_{\min}(i)$ and $j_{\max}(i)$ of the first and last bands in each spectral partition can be derived as a function of the common processing's sampling rate 12.8 kHz and FFT size 256 (see subclause 5.1):

$$j_{\max}(i) = \begin{cases} f_{\max}(i)\dfrac{256}{12800} & i = 0,...,L_{\mathrm{part}}^{[\mathrm{FFT}]}-1 \\[2mm] j_{\max}(L_{\mathrm{part}}^{[\mathrm{FFT}]}-1) + \dfrac{2f_{\max}(i)-12800}{800} & i = L_{\mathrm{part}}^{[\mathrm{FFT}]},...,L_{\mathrm{part}}-1 \end{cases}, \qquad (1359)$$

$$j_{\min}(i) = \begin{cases} f_{\min}(0)\dfrac{256}{12800} & i = 0 \\[2mm] j_{\max}(i-1)+1 & i > 0 \end{cases}, \qquad (1360)$$

where $f_{\min}(0) = 50\mathrm{Hz}$ is the frequency of the first band in the first spectral partition. Hence the FD-CNG generates some comfort noise above 50Hz only.

5.6.3.2 FD-CNG noise estimation

The FD-CNG relies on a noise estimator to track the energy of the background noise present in the input spectrum. This is mostly based on the minimum statistics algorithm [R. Martin, Noise Power Spectral Density Estimation Based on Optimal Smoothing and Minimum Statistics, 2001].

However, to reduce the dynamic range of the input energies $\{E_{\mathrm{FD-CNG}}(0),...,E_{\mathrm{FD-CNG}}(L_{\mathrm{part}}-1)\}$ and hence facilitate the fixed-point implementation of the noise estimation algorithm, a non-linear transform is applied before noise estimation (see subclause 5.6.3.2.1). The inverse transform is then used on the resulting noise estimates to recover the original dynamic range (see subclause 5.6.3.2.3). The resulting noise estimates are used in subclause 5.6.3.5 to encode the SID frames.

5.6.3.2.1 Dynamic range compression for the input energies

The input energies are processed by a non-linear function and quantized with 9-bit resolution as follows:

$$E_{\mathrm{MS}}(i) = \frac{\left\lfloor \log_2\left((1+E_{\mathrm{FD-CNG}}(i))2^9\right) \right\rfloor}{2^9} \qquad\qquad i = 0,...,L_{\mathrm{part}}-1. \qquad (1361)$$

Background of using log2 is that the (int)log2 can usually be calculated very quickly (in one cycle) on fixed-point processors using the "norm" function which determines the numbers of leading zeros in a fixed point number.

Background for adding a constant 1 inside the log2 function is to ensure that the converted energies $E_{\mathrm{MS}}(i)$ remain positive. This is especially important as the noise estimator rely on a statistical model of the noise energy. Performing noise estimation on negative values would strongly violate the model and can result in unexpected behaviour.

5.6.3.2.2 Noise tracking

The input energy $E_{\mathrm{MS}}(i)$ corresponds to an instantaneous power for the i-th partition, referred to as periodogram in the sequel. The minimum statistics algorithm relies on an optimally smoothed periodogram $P_{\mathrm{MS}}(i)$ which can be considered as an estimate of the input power spectral density. The algorithm derives therefore an estimate of the noise power spectral density which we denote in the following as $N_{\mathrm{MS}}(i)$. As described in the sequel, some additional smoothing of $N_{MS}(i)$ is applied, yielding the smoothed noise estimate $\overline{N}_{\mathrm{MS}}(i)$ introduced in subclause 5.6.3.2.2.5.

5.6.3.2.2.1 Initialization phase

To correctly initialize the noise estimation algorithm, an initialization phase is used as long as the input energy $E_{\mathrm{MS}}(0)$ of the first partition grows. Note that the initialization phase is also triggered when a reset of the noise estimation algorithm is judged necessary, as described in subclause 5.6.3.4.

The following applies for each partition $i = 0,...,L_{\mathrm{part}}-1$ during the initialization phase:

$$P_{\mathrm{MS}}(i) = N_{\mathrm{MS}}(i) = \overline{N}_{\mathrm{MS}}(i) = E_{\mathrm{MS}}(i). \tag{1362}$$

Moreover, the minimum statistics algorithm includes a bias compensation mechanism exploiting statistical moments $P_{\mathrm{MS,mean}}(i)$ and $P_{\mathrm{MS,var}}(i)$ of first and second orders, respectively. During the initialization phase, we have $P_{\mathrm{MS,mean}}(i) = E_{\mathrm{MS}}(i)$ and $P_{\mathrm{MS,var}}(i) = 0$.

It is also necessary to initialize several summed quantities. The total noise energy over the FFT and CLDFB partitions are computed during the initialization phase as

$$N_{\mathrm{MS,tot}}^{[\mathrm{FFT}]} = \sum_{i=0}^{L_{\mathrm{part}}^{[\mathrm{FFT}]}-1} E_{\mathrm{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right), \tag{1363}$$

and

$$N_{\mathrm{MS,tot}}^{[\mathrm{CLDFB}]} = \sum_{i=L_{\mathrm{part}}^{[\mathrm{FFT}]}}^{L_{\mathrm{part}}-1} E_{\mathrm{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right), \tag{1364}$$

respectively. Note that the quantity $j_{\max}(i) - j_{\min}(i) + 1$ corresponds to the size of the i-th partition. The total input energies $E_{\mathrm{MS,tot}}^{[\mathrm{FFT}]}$ and $E_{\mathrm{MS,tot}}^{[\mathrm{CLDFB}]}$ for the FFT and CLDFB partitions are initialized to $N_{\mathrm{MS,tot}}^{[\mathrm{FFT}]}$ and $N_{\mathrm{MS,tot}}^{[\mathrm{CLDFB}]}$, respectively, and the total smoothed input energies $P_{\mathrm{MS,tot}}^{[\mathrm{FFT}]}$ and $P_{\mathrm{MS,tot}}^{[\mathrm{CLDFB}]}$ are initialized with $E_{\mathrm{MS,tot}}^{[\mathrm{FFT}]}$ and $E_{\mathrm{MS,tot}}^{[\mathrm{CLDFB}]}$, respectively.

In subclause 5.6.3.2.2.4, some auxiliary arrays $P_{\min}$, $P_{\min,\mathrm{out}}$, $P_{\min,\mathrm{sub}}$ and $P_{\min,\mathrm{buf}}$ are also required to track minima in each spectral partition. They are all filled with the largest possible platform value during the initialization phase.

5.6.3.2.2.2 Optimal smoothing of the input energies

As mentioned earlier, the power spectral density estimate $P_{\mathrm{MS}}(i)$ is computed iteratively as a smoothed version of the input energy $E_{\mathrm{MS}}(i)$, i.e.,

$$P_{\mathrm{MS}}(i) = \alpha_{\mathrm{opt}}(i) P_{\mathrm{MS}}(i) + \left(1 - \alpha_{\mathrm{opt}}(i)\right) E_{\mathrm{MS}}(i) \qquad\qquad i = 0,...,L_{\mathrm{part}}-1, \tag{1365}$$

where $\alpha_{\mathrm{opt}}(i)$ is a time-varying optimal smoothing parameter. It is computed separately for the FFT and CLDFB:

$$\alpha_{\text{opt}}(i) = \begin{cases} \max\left(\dfrac{0.96\,\alpha_c^{[\text{FFT}]}\left(N_{\text{MS}}(i)\right)^2}{\left(N_{\text{MS}}(i)\right)^2 + \left(P_{\text{MS}}(i) - N_{\text{MS}}(i)\right)^2}, \alpha_{\min}^{[\text{FFT}]} \right) & i = 0,...,L_{\text{part}}^{[\text{FFT}]}-1 \\[4mm] \max\left(\dfrac{0.96\,\alpha_c^{[\text{CLDFB}]}\left(N_{\text{MS}}(i)\right)^2}{\left(N_{\text{MS}}(i)\right)^2 + \left(P_{\text{MS}}(i) - N_{\text{MS}}(i)\right)^2}, \alpha_{\min}^{[\text{CLDFB}]} \right) & i = L_{\text{part}}^{[\text{FFT}]},...,L_{\text{part}}-1 \end{cases}, \tag{1366}$$

where

$$\alpha_c^{[\text{FFT}]} = 0.7\,\alpha_c^{[\text{FFT}]} + 0.3\max\left(\dfrac{\left(E_{\text{MS,tot}}^{[\text{FFT}]}\right)^2}{\left(E_{\text{MS,tot}}^{[\text{FFT}]}\right)^2 + \left(P_{\text{MS,tot}}^{[\text{FFT}]} - E_{\text{MS,tot}}^{[\text{FFT}]}\right)^2}, 0.3 \right) \tag{1367}$$

$$\alpha_c^{[\text{CLDFB}]} = 0.7\,\alpha_c^{[\text{FFT}]} + 0.3\max\left(\dfrac{\left(E_{\text{MS,tot}}^{[\text{CLDFB}]}\right)^2}{\left(E_{\text{MS,tot}}^{[\text{CLDFB}]}\right)^2 + \left(P_{\text{MS,tot}}^{[\text{CLDFB}]} - E_{\text{MS,tot}}^{[\text{CLDFB}]}\right)^2}, 0.3 \right) \tag{1368}$$

are some correction factors,

$$\alpha_{\min}^{[\text{FFT}]} = \min\left(\dfrac{E_{\text{MS,tot}}^{[\text{FFT}]}}{\sum_{i=0}^{L_{\text{part}}^{[\text{FFT}]}-1} N_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right)}, 0.05 \right) \tag{1369}$$

$$\alpha_{\min}^{[\text{CLDFB}]} = \min\left(\dfrac{E_{\text{MS,tot}}^{[\text{CLDFB}]}}{\sum_{i=L_{\text{part}}^{[\text{FFT}]}}^{L_{\text{part}}-1} N_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right)}, 0.05 \right) \tag{1370}$$

impose a lower limit on the optimal smoothing parameter, and

$$E_{\text{MS,tot}}^{[\text{FFT}]} = \sum_{i=0}^{L_{\text{part}}^{[\text{FFT}]}-1} E_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right), \tag{1371}$$

$$P_{\text{MS,tot}}^{[\text{FFT}]} = \sum_{i=0}^{L_{\text{part}}^{[\text{FFT}]}-1} P_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right), \tag{1372}$$

$$E_{\text{MS,tot}}^{[\text{CLDFB}]} = \sum_{i=L_{\text{part}}^{[\text{FFT}]}}^{L_{\text{part}}-1} E_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right), \tag{1373}$$

$$P_{\text{MS,tot}}^{[\text{CLDFB}]} = \sum_{i=L_{\text{part}}^{[\text{FFT}]}}^{L_{\text{part}}-1} P_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right) \tag{1374}$$

denote some summed quantities.

5.6.3.2.2.3   Bias compensation

The minimum statistics algorithm essentially consists in tracking the minima of $P_{\text{MS}}(i)$ for each partition i over time. However, this method delivers some biased estimates and necessitates therefore the computation of a bias compensation factor which is dependent on the variance of $P_{\text{MS}}(i)$.

For each spectral partition $i = 0,...,L_{\text{part}}-1$, we first estimate the first-order moment of $P_{\text{MS}}(i)$ as

$$P_{\text{MS,mean}}(i) = \beta_{\text{MS}}(i)\,P_{\text{MS,mean}}(i) + \left(1 - \beta_{\text{MS}}(i)\right)P_{\text{MS}}(i), \tag{1375}$$

where $\beta_{\text{MS}}(i) = \min\left(\alpha_{\text{opt}}^2(i), 0.8\right)$ is a smoothing parameter. The variance of $P_{\text{MS}}(i)$ is then derived as follows:

$$P_{\text{MS,var}}(i) = \beta_{\text{MS}}(i)P_{\text{MS,var}}(i) + \left(1 - \beta_{\text{MS}}(i)\right)\left(P_{\text{MS}}(i) - P_{\text{MS,mean}}(i)\right)^2 . \tag{1376}$$

As shown in subclause 5.6.3.2.2.4, the minimum tracking uses in each partition i a window of $K_{\text{MS}} = 6$ sub-windows of length $L_{\text{MS}} = 12$ each. Minima are in fact computed over the entire buffer of size $K_{\text{MS}} \times L_{\text{MS}}$ past frames, but also over the last sub-window. The bias compensation factor $B_{\text{win}}(i)$ for the total window length, and $B_{\text{sub}}(i)$ for a sub-window are given for each partition $i = 0, ..., L_{\text{part}} - 1$ as

$$B_{\text{win}}(i) = 1 + \frac{(K_{\text{MS}}L_{\text{MS}} - 1)(1 - M_{\text{win}})Q_{\text{eq}}^{-1}(i)}{0.5 - M_{\text{win}} \, Q_{\text{eq}}^{-1}(i)} , \tag{1377}$$

$$B_{\text{sub}}(i) = 1 + \frac{(L_{\text{MS}} - 1)(1 - M_{\text{sub}})Q_{\text{eq}}^{-1}(i)}{0.5 - M_{\text{sub}} \, Q_{\text{eq}}^{-1}(i)} , \tag{1378}$$

with

$$Q_{\text{eq,inv}}(i) = \min\left(\frac{P_{\text{MS,var}}(i)}{2(N_{\text{MS}}(i))^2}, 0.2\right) , \tag{1379}$$

$$M_{\text{win}} = 0.865 + \left(\frac{\sqrt{80}\sqrt{60}}{\sqrt{72}} - \sqrt{60}\right)\frac{0.841 - 0.865}{\sqrt{80} - \sqrt{60}} , \tag{1380}$$

$$M_{\text{sub}} = 0.668 + \left(\frac{\sqrt{15}\sqrt{10}}{\sqrt{12}} - \sqrt{10}\right)\frac{0.610 - 0.668}{\sqrt{15} - \sqrt{10}} . \tag{1381}$$

For the sake of robustness, the bias compensation factors are furthermore increased proportionally to the mean of $Q_{\text{eq,inv}}^{[\text{FFT}]}(i)$ and $Q_{\text{eq,inv}}^{[\text{CLDFB}]}(i)$ among the spectral partitions. The correction factor is obtained for the FFT and CLDFB partitions as

$$B_{\text{c}}^{[\text{FFT}]} = 1 + 2.12\sqrt{Q_{\text{eq,inv}}^{[\text{FFT}]}(i)} , \tag{1382}$$

$$B_{\text{c}}^{[\text{CLDFB}]} = 1 + 2.12\sqrt{Q_{\text{eq,inv}}^{[\text{CLDFB}]}(i)} , \tag{1383}$$

with

$$\overline{Q}_{\text{eq,inv}}^{[\text{FFT}]}(i) = \frac{\sum_{i=0}^{L_{\text{part}}^{[\text{FFT}]} - 1} Q_{\text{eq,inv}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right)}{j_{\max}(L_{\text{part}}^{[\text{FFT}]} - 1) - j_{\min}(0) + 1} , \tag{1384}$$

$$\overline{Q}_{\text{eq,inv}}^{[\text{CLDFB}]}(i) = \frac{\sum_{i=L_{\text{part}}^{[\text{FFT}]}}^{L_{\text{part}} - 1} Q_{\text{eq}}^{-1}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right)}{j_{\max}(L_{\text{part}} - 1) - j_{\min}(L_{\text{part}}^{\text{FFT}}) + 1} . \tag{1385}$$

5.6.3.2.2.4 Minimum tracking

For the sake of simplicity, we provide a description of the minimum tracking algorithm for the FFT partitions only. The CLDFB partitions can be treated in the same way.

The bias compensation factor $B_{\text{win}}(i)$ and the correction factor $B_c^{[\text{FFT}]}$ computed in subclause 5.6.3.2.2.3 are used to obtain a more accurate estimate of the background noise energy in each partition. They are re-computed after each frame and tracking of the minimum $P_{\min}(i)$ is carried out in each FFT partition $i = 0,..., L_{\text{part}}^{[\text{FFT}]} - 1$ as $P_{\min}(i) = \min\left(B_c^{[\text{FFT}]} P_{\text{MS}}(i) B_{\text{win}}(i), P_{\min}(i)\right)$. When a new minimum $P_{\min,\text{win}}(i)$ is found, a flag $f_{\text{new\_min}}(i)$ is set to 1 and $P_{\min,\text{sub}}(i)$ is updated as $P_{\min,\text{sub}}(i) = \min\left(B_c^{[\text{FFT}]} P_{\text{MS}}(i) B_{\text{sub}}(i), P_{\min,\text{sub}}(i)\right)$. Otherwise $f_{\text{new\_min}}(i)$ is set to zero.

Note that $P_{\min}(i)$ is set to the maximum possible platform value after processing the last frame of each sub-window, i.e., every L_{MS} frames. $P_{\min}(i)$ and $P_{\min,\text{sub}}(i)$ refer therefore to a minimum within the current sub-window for the partition i. In the last frame of the current sub-window, the current minimum $P_{\min}(i)$ is stored into a buffer $P_{\min,\text{buf}}(i)$ collecting the minima found in the last K_{MS} sub-windows. The buffer is used at the end of each sub-window to determine the overall minimum among all sub-windows $P_{\min,\text{out}}(i)$.

For a frame in between the first and last frames of the current sub-window (i.e., frames 2 to $L_{\text{MS}} - 1$ of the sub-window), the overall minimum $P_{\min,\text{out}}(i)$ is updated as $P_{\min,\text{out}}(i) = \min\left(P_{\min,\text{sub}}(i), P_{\min,\text{out}}(i)\right)$, and a flag $f_{\text{local\_min}}(i)$ is set to 1 if a new minimum was found, i.e., if $f_{\text{new\_min}}(i) = 1$. The flag $f_{\text{local\_min}}(i)$ is set to 0 after processing the last frame of each sub-window.

To improve tracking of a time-varying noise, a local minimum $P_{\min,\text{sub}}(i)$ among the current sub-window can replace the overall minimum $P_{\min,\text{out}}(i)$ in the last frame of each sub-window provided that it yields only a moderate increase of $P_{\min,\text{out}}(i)$, and if the local minimum was not found in the first or last frame of the sub-window, i.e., if $f_{\text{local\_min}}(i) = 1$ and $f_{\text{new\_min}}(i) = 0$. In this case, $P_{\min,\text{sub}}(i)$ replaces also all values in the buffer $P_{\min,\text{buf}}(i)$. The search range $slope_{\max}^{[\text{FFT}]}$ for the local minima (and hence the tolerated amount of increase compared to the current overall minimum) lies between 1.1 and 2 and increases as $\overline{Q}_{\text{eq,inv}}^{[\text{FFT}]}(i)$ decreases:

$$slope_{\max}^{[\text{FFT}]} = \begin{cases} 2 & \text{if } \overline{Q}_{\text{eq,inv}}^{[\text{FFT}]}(i) < 0.01 \\ 1.6 & \text{if } \overline{Q}_{\text{eq,inv}}^{[\text{FFT}]}(i) < 0.03 \\ 1.3 & \text{if } \overline{Q}_{\text{eq,inv}}^{[\text{FFT}]}(i) < 0.05 \\ 1.1 & \text{otherwise} \end{cases}. \tag{1386}$$

The noise estimate $N_{\text{MS}}(i)$ is updated to $P_{\min,\text{out}}(i)$ after each frame, except for the first frame in each sub-window.

Furthermore, when the smoothed energy of the first spectral partition exceeds the instantaneous energy by a factor of more than 50 (i.e., $P_{\text{MS}}(0) > 50 E_{\text{MS}}(0)$) for at least two frames in a row, a sudden noise offset is assumed and the noise tracking is modified for all partitions $i = 0,..., L_{\text{part}} - 1$ as:

$$P_{\text{MS}}(i) = P_{\min,\text{out}}(i) = P_{\text{MS,mean}}(i) = E_{\text{MS}}(i), \tag{1387}$$

$$P_{\text{MS,var}}(i) = 0, \tag{1388}$$

$$\alpha_{\text{opt}}(i) = 0, \tag{1389}$$

and

$$\alpha_c^{[\text{FFT}]} = \alpha_c^{[\text{CLDFB}]} = 1, \tag{1390}$$

$$P_{\text{MS,tot}}^{[\text{FFT}]} = \sum_{i=0}^{L_{\text{part}}^{[\text{FFT}]} - 1} E_{\text{MS}}(i)\left(j_{\max}(i) - j_{\min}(i) + 1\right), \tag{1391}$$

$$P_{\text{MS,tot}}^{[\text{CLDFB}]} = \sum_{i=L_{\text{part}}^{[\text{FFT}]}}^{L_{\text{part}}-1} E_{\text{MS}}(i)\big(j_{\max}(i) - j_{\min}(i) + 1\big) . \tag{1392}$$

5.6.3.2.2.5 Smoothing of the noise estimates

The main outputs of the noise tracker are the noise estimates $N_{\text{MS}}(i), i = 0,..., L_{\text{part}}-1$. To obtain smoother transitions in the comfort noise, a first-order recursive filter is applied, i.e.

$$\overline{N}_{\text{MS}}(i) = 0.95\,\overline{N}_{\text{MS}}(i) + 0.05\,N_{\text{MS}}(i) . \tag{1393}$$

Furthermore, the input energy $E_{\text{MS}}(i)$ is averaged over the last 5 frames. This is used to apply an upper limit on $\overline{N}_{\text{MS}}(i)$ in each spectral partition.

5.6.3.2.3 Dynamic range expansion for the estimated noise energies

The estimated noise energies are processed by a non-linear function to compensate for the dynamic range compression applied in subclause 5.6.3.2.1:

$$N_{\text{FD-CNG}}(i) = 2^{\overline{N}_{\text{MS}}(i)-1} \qquad i = 0,..., L_{\text{part}}-1. \tag{1394}$$

5.6.3.3 Adjusting the first SID frame in FD-CNG

Before encoding the first SID frame of a CNG phase (i.e., an SID frame preceded by an active frame), an upper limit is applied to the noise estimates $N_{\text{FD-CNG}}(i)$ to minimize the risk of generating some noise bursts at the beginning of a CNG phase. To this end, the noise estimate $N_{\text{FD-CNG,old}}(i)$ obtained during the previous inactive frame (i.e., one frame before the last active phase) is used in combination with the input energy of the current frame as follows:

$$N_{\text{FD-CNG}}(i) = \min\big(N_{\text{FD-CNG}}(i), \lambda\,N_{\text{FD-CNG,old}}(i) + (1-\lambda)\,E_{\text{FD-CNG}}(i)\big), \tag{1395}$$

where $i = 0,..., L_{\text{part}}-1$ refers to the partition index, $\lambda = 0.96^{num\_active\_frames+1}$, and *num_active_frames* corresponds to the length of the last active phase.

5.6.3.4 FD-CNG resetting mechanism

To signal the need of resetting FD-CNG, a flag is computed based on a detection of fast increasing noise energy.

If the current total noise energy N_t as calculated in subclause 5.1.11.1 is bigger than the one of the last frame, up to four difference values of the total noise energy of the previous frames are summed up, in case the noise energy increased in these last frames consecutively.

If the encoder is out of initialisation phase, four or more frames with increasing N_t were detected and the sum of the last four of them was bigger than 5, the reset flag is set to 1. Besides that, in case the signal's input bandwidth of the current frame is larger than the previous one, the reset flag is also set to 1.

If none of this is the case but the flag was set before, it takes nine more frames before the flag is set to zero.

In case the flag is set to one, a reinitialization of the minimum statistics routine is triggered (subclause 5.6.3.2.2.1), and selection of SID or NO_DATA frame is forbidden.

5.6.3.5 Encoding SID frames in FD-CNG

The CN parameters to be encoded into a FD-CNG SID frame are the noise estimates $N_{\text{FD-CNG}}(i)$, with $i = 0,..., L_{\text{SID}}-1$, and L_{SID} depends on the bandwidth and the bitrate as given in the table below.

Table 153: Number of CN parameters encoded in a FD-CNG SID frame

Bandwidth	NB	WB		SWB
Bit-rates [kbps]	•	≤ 8	> 8	•
L_{SID}	17	20	21	24

The noise estimates $N_{\text{FD-CNG}}(i)$ are first converted to dB

$$N_{\text{FD-CNG}}^{\text{dB}}(i) = 10 \log 10 \left(N_{\text{FD-CNG}}(i) + 0.0001 \right).$$
(1396)

The noise estimates in dB are then normalized using

$$\overline{N}_{\text{FD-CNG}}^{\text{dB}}(i) = N_{\text{FD-CNG}}^{\text{dB}}(i) - \frac{\displaystyle\sum_{i=4}^{i<17} N_{\text{FD-CNG}}^{\text{dB}}(i)}{13}.$$
(1397)

The normalized noise estimates in dB $\overline{N}_{\text{FD-CNG}}^{\text{dB}}(i)$ are then quantized using a Multi-Stage Vector Quantizer (MSVQ). The MSVQ has 6 stages, with 7 bits in the first stage and 6 bits in the other stages (total of 37 bits). A M-best search algorithm is used, with M=24 is the number of survivors in a stage that will be searched in the next stage. Note that a single set of codebooks is used for all configurations. The vectors in the codebook have a length of 24, and they are simply truncated if L_{SID} is less than 24.

The MSVQ decoder output is given by

$$\overline{N}_{\text{FD-CNG}}^{[\text{SID}]}(i) = \sum_{k=0}^{5} V_k \left(I_{\text{MSVQ,FD-CNG}}(k), i \right),$$
(1398)

where $I_{\text{MSVQ,FD-CNG}}(k)$ are the indices encoded in the bitstream and $V_k(j, i)$ is the i-th coefficient of the j-th vector in the codebook of stage k.

A global gain is then computed

$$g_{\text{FD-CNG}} = \frac{\displaystyle\sum_{i=0}^{i<L_{\text{part}}} N_{\text{FD-CNG}}^{\text{dB}}(i) - \sum_{i=0}^{i<L_{\text{part}}} \overline{N}_{\text{FD-CNG}}^{[\text{SID}]}(i)}{L_{\text{SID}}} - \Delta_{\text{g,FD-CNG}},$$
(1399)

with $\Delta_{\text{g,FD-CNG}}$ is a scale which depends on the bandwidth and the bitrate as described in the table below.

Table 154: FD-CNG SID global gain scale

Bandwidth	NB				WB						SWB		
Bit-rates [kbps]	<=7.2	8	9.6	13.2	<=7.2	8	9.6	13.2	16.4	24.4	13.2	16.4	24.4
$\Delta_{\text{g,FD-CNG}}$ [dB]	-5.5	-5	-4	-3	-5.5	-5	-1.55	-3	-0.6	-0.2	-3	-0.8	-0.25

The global gain is then quantized on 7 bits using

$$I_{\text{g,FD-CNG}} = \min \left(\max \left(\lfloor g_{\text{FD-CNG}} * 1.5 + 60.5 \rfloor, 0 \right), 127 \right),$$
(1400)

producing the quantized global gain

$$g_{\text{FD–CNG}}^{[\text{SID}]} = \frac{I_{\text{g,FD–CNG}} - 60}{1.5}.$$
(1401)

The quantized noise estimates are then given by

$$N_{\text{FD–CNG}}^{[\text{SID}]}(i) = 10^{\left(\frac{\overline{N}_{\text{FD–CNG}}^{[\text{SID}]}(i) + g_{\text{FD–CNG}}^{[\text{SID}]}}{10}\right)}.$$
(1402)

Finally the last band parameter is adjusted in case the encoded last band size is different from the decoded last band size

$$if\ ((bandwidth = NB) \lor (bandwidth = SWB \land bitrate \le 13.2kbps))$$

$$then$$

$$N_{\text{FD–CNG}}^{[\text{SID}]}(L_{\text{SID}} - 1) = 0.8 N_{\text{FD–CNG}}^{[\text{SID}]}(L_{\text{SID}} - 1)$$

5.6.3.6 FD-CNG local CNG synthesis

Noise estimates encoded in the SID frame are then used to locally generate CNG, in order to update the encoder memories. The following table lists the number of partitions used for generating CNG in the FD-CNG encoder, and their upper boundaries, as a function of bandwiths and bit-rates.

Table 155: Configurations of the FD-CNG local CNG synthesis

	Bit-rates (kbps)	$L_{\text{SID}}^{[\text{FFT}]}$	$L_{\text{SID}}^{[\text{CLDFB}]}$	$f_{\max}^{[\text{SID}]}(i),$ $i = 0,...,L_{\text{SID}}^{[\text{FFT}]} - 1$ (Hz)	$f_{\max}^{[\text{SID}]}(i),$ $i = L_{\text{SID}}^{[\text{FFT}]},...,L_{\text{SID}} - 1$ (Hz)
NB	•	17	0	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3975	×
WB	≤ 8	20	0	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6375	×
	$8 < • \le 13.2$	20	1	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6375	8000
	> 13.2	21	0	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6375, 7975	×
SWB/ FB	≤ 13.2	20	4	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6375	8000, 10000, 12000, 14000
	> 13.2	21	3	100, 200, 300, 400, 500, 600, 750, 900, 1050, 1250, 1450, 1700, 2000, 2300, 2700, 3150, 3700, 4400, 5300, 6375, 7975	10000, 12000, 16000

For each partition $i = 0,...,L_{\text{SID}} - 1$, $f_{\max}^{[\text{SID}]}(i)$ corresponds to the frequency of the last band in the i-th partition. The indices $j_{\min}^{[\text{SID}]}(i)$ and $j_{\max}^{[\text{SID}]}(i)$ of the first and last bands in each spectral partition can be derived as a function of the core sampling rate sr_{CELP} and the FFT size $L_{\text{FFT}}^{[\text{FD–CNG}]} = sr_{\text{CELP}}/25$:

$$j_{\max}^{[SID]}(i) = \begin{cases} f_{\max}^{[SID]}(i) \dfrac{L_{FFT}^{[FD-CNG]}}{sr_{CELP}} & i = 0,...,L_{SID}^{[FFT]}-1 \\[2em] j_{\max}^{[SID]}(L_{SID}^{[FFT]}-1) + \dfrac{2 f_{\max}^{[SID]}(i) - sr_{CELP}}{800} & i = L_{SID}^{[FFT]},...,L_{SID}-1 \end{cases}, \qquad (1403)$$

$$j_{\min}^{[SID]}(i) = \begin{cases} f_{\min}^{[SID]}(0) \dfrac{L_{FFT}^{[FD-CNG]}}{sr_{CELP}} & i = 0 \\[2em] j_{\max}^{[SID]}(i-1)+1 & i > 0 \end{cases}, \qquad (1404)$$

where $j_{\min}^{[SID]}(0) = 50$Hz is the frequency of the first band in the first spectral partition. Hence the FD-CNG generates some comfort noise above 50Hz only.

5.6.3.6.1 SID parameters interpolation

The SID parameters are interpolated using linear interpolation in the log domain, as described in subclause 6.7.3.1.2. The interpolated SID parameters are noted $N_{FD-CNG}^{[SID,FR]}(j), j = 0,...,L_{FFT}/2$.

5.6.3.6.2 LPC estimation from the interpolated SID parameters

A set of LPC coefficients is estimated from the SID spectrum in order to update excitation and LPC related memories, as described in subclause 6.7.3.1.3. The LPC coefficients are noted $a_{FD-CNG}(n), n = 0,...,16$.

5.6.3.6.3 FD-CNG encoder comfort noise generation

A FD-CNG time-domain signal is generated similarly to the time-domain CNG signal generated at the decoder-side (see subclauses 6.7.3.3.2 and 6.7.3.3.3), except that the interpolated SID parameters $N_{FD-CNG}^{[SID,FR]}(j)$ are used to generate the noise in the frequency-domain (instead of the noise levels $N_{FD-CNG}^{[CNG]}(j)$ which are not available at the encoder side).

5.6.3.6.4 FD-CNG encoder memory update

The memories update is performed using the FD-CNG time-domain signal and the LPC coefficients $a_{FD-CNG}(n), n = 0,...,16$, similarly to the decoder memory update (see subclause 6.7.3.3.4).

Additionally, the weighted signal domain memories are updated by filtering the FD-CNG time-domain signal through a LP analysis filter with a weighted version of the LPC coefficients $a_{FD-CNG}(n), n = 0,...,16$.

5.7 AMR-WB-interoperable modes

The EVS codec supports modes to allow for interoperability with AMR-WB (and ITU-T G.722.2). The inclusion of the interoperable modes has been streamlined due to the fact that the core ACELP coder described in subclause 5.2 is similar to AMR-WB (when operating at 12.8 kHz internal sampling, using the same pre emphasis and perceptual weighting, etc.).

5.7.1 Pre-processing

The high-pass filtering, sampling conversion, pre-emphasis, spectral analysis, signal activity detection functions are the same as those described in subclause 5.1.

5.7.2 Linear prediction analysis and quantization

5.7.2.1 Windowing and auto-correlation computation

Short-term prediction analysis is performed once per speech frame using the autocorrelation approach per subclause 5.1.9. However, the 30 ms asymmetric window defined in subclause 5.2.1 of [9] and a look-ahead of 5 ms are used in the autocorrelation calculation. The frame structure is depicted in figure 86.
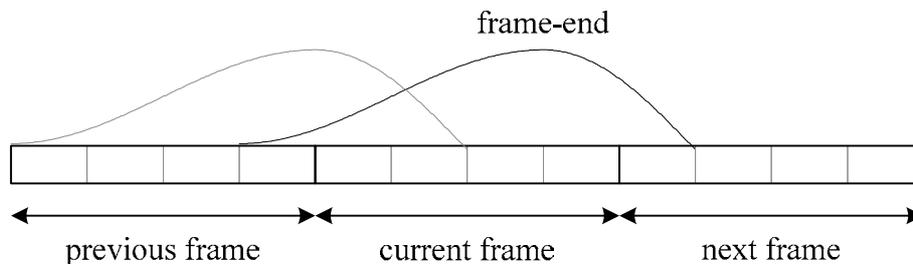
Figure 86: Relative positions and length of the LP analysis windows for the AMR-WB interoperable option

The autocorrelations of the windowed signal are computed in the same way as described in subclause 5.1.9.2, except that $L = 384$ in equation (45). Note that the autocorrelations are computed in the same way as in subclause 5.2.1 of [9] but with a different white noise correction factor value and lag windowing as described in subclause 5.1.9.3 of this Specification.

5.7.2.2 Levinson-Durbin algorithm

The Levinson-Durbin algorithm is the same as in subclause 5.5.1.3.

5.7.2.3 LP to ISP conversion

For a linear predictive model *A(z)* of order *m* we can define the line spectral polynomials as

$$P(z) = A(z) + z^{-m-l}A(z^{-1})$$
$$Q(z) = A(z) - z^{-m-l}A(z^{-1})$$

(1405)

where *l=1* for the line spectrum polynomials and *l=0* for the immittance spectrum polynomials. These polynomials *P(z)* and *Q(z)* are symmetric and anti-symmetric, respectively, such that the point of symmetry is $z^{-(m+l)/2}$. It follows that when evaluating at the unit circle $z = e^{i\alpha}$, then the obtained spectra of $z^{-(m+l)/2}P(z)$ and $z^{-(m+l)/2}Q(z)$ will be real and imaginary, respectively. Further, since polynomials *P(z)* and *Q(z)* have roots on the unit circle, they can be located by a zero-crossing search of the two spectra.

The evaluation of on the unit circle can be implemented with an FFT of length *N=256*. Since the two spectra are imaginary, we can evaluate both spectra simultaneously, that is, since $2A(z) = P(z) + Q(z)$, we can determine the

spectrum of $2\,A(z)$ by an FFT, multiply by $z^{-(m+l)/2}$ and then obtain the two spectra in the real and imaginary parts. Scaling by the factor 2 does not influence location of zeros, whereby it can be omitted.

To reduce numerical range of the spectrum, we can convolve $A(z)$ by a filter $B(z)$, where

$$B(z) = b_0 + b_1 z^{-1} + b_0 z^{-2} \tag{1406}$$

and the constants are $b_0 = A(1) - A(-1)$ and $b_1 = -2(A(1) + A(-1))$. That is, we calculate the spectrum of $A(z)B(z)$ and multiply with the phase-shift $z^{-(m+l+2)/2}$.

When calculating the FFT, we can reduce complexity by applying pruning methods. That is, since $A(z)B(z)$ is a sequence of length $m+3$, but the FFT is of length N, we can omit all those operations which involve computations with the $N - m - 3$ zeros.

5.7.2.4 ISP to LP conversion

The ISP to LP conversion is the same as in subclause 5.2.4 of [9].

5.7.2.5 Quantization of the ISP coefficients

For interoperability reasons, ISF quantization is the same as in subclause 5.2.5 of [9].

5.7.2.6 Interpolation of the ISPs

The set of LP parameters is used for the 4th subframe, whereas the 1st, 2nd and 3rd subframes use a linear interpolation of the parameters in the adjacent frames. The interpolation is performed on the ISPs in the $\mathbf{q}$ domain. Let $\mathbf{q}_{end}$ be the ISP vector at the 4th subframe of the current frame, and $\mathbf{q}_{end,p}$ is the ISP vector at the 4th subframe of the previous frame. The interpolated ISP vectors at the 1st, 2nd and 3rd subframes are given by

$$
\begin{aligned}
\mathbf{q}^{[0]} &= 0.55\,\mathbf{q}_{end,p} + 0.45\,\mathbf{q}_{end} \\
\mathbf{q}^{[1]} &= 0.2\,\mathbf{q}_{end,p} + 0.8\,\mathbf{q}_{end} \\
\mathbf{q}^{[2]} &= 0.04\,\mathbf{q}_{end,p} + 0.96\,\mathbf{q}_{end} \\
\mathbf{q}^{[3]} &= \mathbf{q}_{end}
\end{aligned}
\tag{1407}
$$

The same formula is used for interpolation of both quantized and unquantized ISPs. The interpolated ISP vectors are used to compute a different LP filter at each subframe (both quantized and unquantized) using the ISP to LP conversion method described in subclause 5.2.4 of [9].

5.7.3 Perceptual weighting

Perceptual weighting is performed as described in subclause 5.1.10.1 for a sub-frame size $L = 64$.

5.7.4 Open-loop pitch analysis

The open-loop pitch analysis is performed as described in subclause 5.1.10.

5.7.5 Impulse response computation

Same as subclause 5.2.3.1.3.

5.7.6 Target signal computation

Same as subclause 5.2.3.1.2.

5.7.7 Adaptive codebook search

Same as subclause 5.2.3.1.4.

5.7.8 Algebraic codebook search

For interoperability reasons, the algebraic codebook structure and pulse indexing is the same as clauses 5.8.1 and 5.8.2 of [9]. The algebraic codebook search procedure is the same as described in clause 5.8.3 of [9] except the pulse-sign pre-selection described in the last paragraph of Clause 5.2.3.1.5.9 (The search criterion at lower bitrates), which is also used at the lowest bit-rate of AMR-WB-interoperable modes.

5.7.9 Quantization of the adaptive and fixed codebook gains

For interoperability reasons, the quantization of gains is conducted in the same manner as described in subclause 5.9 of [9].

5.7.10 Memory update

The memory update for AMR-WB interoperable modes is similar to subclause 5.10 of [9], however some extra states defined in the EVS codec are also updated to maintain a consistent operation for the next coding frame when the last frame was coded with an AMR-WB interoperable mode.

5.7.11 High-band gain generation

For interoperability reasons, the quantization of high-band gain in each 5ms sub-frame is conducted in the same manner as described in subclause 5.11 of [9].

5.7.12 CNG coding

The CNG encoding in AMR-WB-interoperable mode is described by referring to subclause 5.6.2 with several differences described below. Instead of the LSF vector which is quantized and transmitted in the SID frame in primary mode, the ISF vector is used for quantization and transmitted in the SID frame in AMR-WB-interoperable mode. 28 bits are used for ISF quantization that is one bit less than the primary mode. The quantization of the ISF vector is described in [9]. The excitation energy used for quantization and transmission in the SID frame is calculated in the same way as described in subclause 5.6.2.1.5. However, the excitation energy is quantized in the AMR-WB-interoperable mode using different numbers of bits and a different quantization step-size from the primary mode. 6 bits are used for quantization in AMR-WB-interoperable mode, instead of the 7 bits used in the primary mode. The quantization step Δ described in subclause 5.6.2.1.5 is 2.625 in the AMR-WB-interoperable mode, instead of 5.25 in primary mode. The quantization index is also limited to [0, 63] in AMR-WB-interoperable mode, instead of to [0, 127] in primary mode.

The smoothed LP synthesis filter, $\hat{A}(Z)$, used for local CNG synthesis is obtained in the same way as described in subclause 5.6.2.1.4. And the smoothed quantized excitation energy, E_{CN}, used for local CNG synthesis is also obtained in the same way as described in subclause 5.6.2.1.6. However, the CNG type flag bit, the bandwidth indicator bit, the core sampling rate indicator bit, the hangover frame counter bits and the Low-band excitation spectral envelope bits used in primary SID mode are not encoded into the SID frame in AMR-WB-interoperable mode, but one dithering bit (always set to 0) is always encoded. The high-band analysis and quantization is also ignored for AMR-WB-interoperable mode, so the high-band energy bits are also not encoded into the AMR-WB-interoperable mode SID frame. Local CNG synthesis is performed by filtering the excitation signal (see subclause 6.8.4) through the smoothed LP synthesis filter $\hat{A}(Z)$.

5.8 Channel Aware Coding

5.8.1 Introduction

EVS offers partial redundancy based error robust channel aware mode at 13.2 kbps for both wideband and super-wideband audio bandwidths. Depending on the criticality of the frame, the partial redundancy is dynamically enabled or disabled for a particular frame, while keeping a fixed bit budget of 13.2 kbps.

5.8.2 Principles of Channel Aware Coding

In a VoIP system, packets arrive at the decoder with random jitters in their arrival time. Packets may also arrive out of order at the decoder. Since the decoder expects to be fed a speech packet every 20 msec to output speech samples in periodic blocks, a de-jitter buffer [7] is required to absorb the jitter in the packet arrival time. Larger the size of the de-jitter buffer, the better is its ability to absorb the jitter in the arrival time and consequently, fewer late arriving packets are discarded. Voice communications is also a delay critical system and therefore it becomes essential to keep the end to end delay as low as possible so that a two way conversation can be sustained.

The design of an adaptive de-jitter buffer reflects the above mentioned trade-offs. While attempting to minimize packet losses, the jitter buffer management algorithm in the decoder also keeps track of the delay in packet delivery as a result of the buffering. The jitter buffer management algorithm suitably adjusts the depth of the de-jitter buffer in order to achieve the trade-off between delay and late losses.

EVS channel aware mode uses partial redundant copies of current frames along with a future frame for error concealment. The partial redundancy technology transmits partial copies of the current frame along with a future frame with the hope that in the event of the loss of the current frame (either due to network loss or late arrival) the partial copy from the future frame can be retrieved from the jitter buffer to improve the recovery from the loss.

The difference in time units between the transmit time of the primary copy of a frame and the transmit time of the redundant copy of the frame (piggy backed onto a future frame) is called the FEC offset. If the depth of the jitter buffer at any given time is at least equal to the FEC offset, then it is quite likely that the future frame is available in the de-jitter buffer at the current time instance. The FEC offset is a configurable parameter at the encoder which can be dynamically adjusted depending on the network conditions. The concept of partial redundancy in EVS with FEC offset equal to 3 is shown in figure 87.
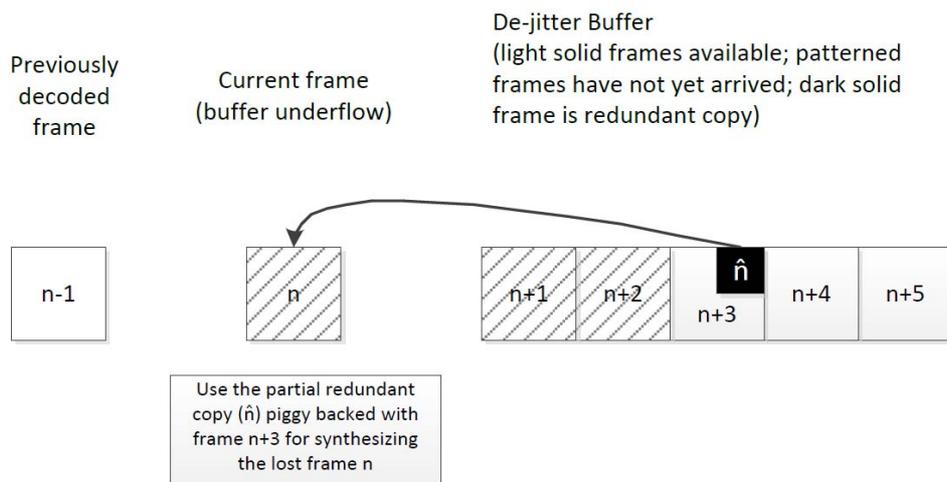
Figure 87 : Concept of partial redundancy in channel aware mode

The redundant copy is only a partial copy that includes just a subset of parameters that are most critical for decoding or arresting error propagation.

The EVS channel aware mode transmits redundancy in-band as part of the codec payload as opposed to transmitting redundancy at the transport layer (e.g., by including multiple packets in a single RTP payload). Including the redundancy in-band allows the transmission of redundancy to be either channel controlled (e.g., to combat network congestion) or source controlled. In the latter case, the encoder can use properties of the input source signal to determine which frames are most critical for high quality reconstruction at the decoder and selectively transmit redundancy for those frames only. Another advantage of in-band redundancy is that source control can be used to determine which frames of input can best be coded at a reduced frame rate in order to accommodate the attachment of redundancy without altering the total packet size. In this way, the channel aware mode includes redundancy in a constant-bit-rate channel (13.2 kbps).

5.8.3 Bit-Rate Allocation for Primary and Partial Redundant Frame Coding

5.8.3.1 Primary frame bit-rate reduction

A measure of compressibility of the primary frame is used to determine which frames can best be coded at a reduced frame rate. For TCX frame the 9.6kpbs setup is applied for WB as well as for SWB. For ACELP the following apply. The coding mode decision coming from the signal classification algorithm is first checked. Speech frames classified for Unvoiced Coding (UC) or Voiced Coding (VC) are suitable for compression. For Generic Coding (GC) mode, the correlation (at pitch lag) between adjacent sub-frames within the frame is used to determine compressibility. Primary frame coding of upper band signal (i.e., from 6.4 to 14.4 kHz in SWB and 6.4 to 8 kHz in WB) in channel aware mode uses time-domain bandwidth extension (TBE) as described in subclause 5.2.6.1. For SWB TBE in channel aware mode, a scaled down version of the non-channel aware mode framework is used to obtain a reduction of bits used for the primary frame. The LSF quantization is performed using an 8-bit vector quantization in channel aware mode while a 21-bit scalar quantization based approach is used in non-channel aware mode. The SWB TBE primary frame gain parameters in channel aware mode are encoded similar to that of non-channel aware mode at 13.2 kbps, i.e., 8 bits for gain parameters. The WB TBE in channel aware mode uses similar encoding as used in 9.6 kbps WB TBE of non-channel aware mode, i.e., 2 bits for LSF and 4 bits for gain parameters.

5.8.3.2 Partial Redundant Frame Coding

The size of the partial redundant frame is variable and depends on the characteristics of the input signal. Also criticality measure is an important metric. A frame is considered as critical to protect when loss of the frame would cause significant impact to the speech quality at the receiver. The criticality also depends on if the previous frames were lost or not. For example, a frame may go from being non-critical to critical if the previous frames were also lost. Parameters computed from the primary copy coding such as coder type classification information, subframe pitch lag, factor M etc are used to measure the criticality of a frame. The threshold, to determine whether a particular frame is critical or not, is a configurable parameter at the encoder which can be dynamically adjusted depending on the network conditions. For example, under high FER conditions it may be desirable to adjust the threshold to classify more frames as critical. Partial frame coding of upper band signal relies on coarse encoding of gain parameters and interpolation/extrapolation of LSF parameters from primary frame. The TBE gain parameters estimated during the primary frame encoding of the (n – FEC offset)-th frame is re-transmitted during the n-th frame as partial copy information. Depending on the partial frame coding mode, i.e., GENERIC or VOICED or UNVOICED, the re-transmission of the gain frame, uses different quantization resolution and gain smoothing.

The following sections describe the different partial redundant frame types and their composition.

5.8.3.2.1 Construction of partial redundant frame for Generic and Voiced Coding modes

In the coding of the redundant version of the frame, a factor M is determined based on the adaptive and fixed codebook energy.

$$M = \frac{\big(E(ACB) + E(FCB)\big/E(ACB) - E(FCB)\big) + 1}{4} \tag{1408}$$

In this equation, $E(ACB)$ denotes the adaptive codebook energy and $E(FCB)$ denotes the fixed codebook energy. A low value of M indicates that most of the information in the current frame is carried by the fixed codebook contribution. In such cases, the partial redundant copy (RF_NOPRED) is constructed using one or more fixed codebook parameters only (FCB pulses and gain). A high value of M indicates that most of the information in the current frame is carried by the adaptive codebook contribution. In such cases, the partial redundant copy (RF_ALLPRED) is constructed using one or more adaptive codebook parameters only (pitch lag and gain). If M takes mid values then a mixed coding mode is selected where one or more adaptive codebook parameters and one or more fixed codebook parameters are coded (RF_GENPRED). Under Generic and Voiced Coding modes, the TBE gain frame values are typically low and demonstrate less variance. Hence a coarse TBE gain frame quantization with gain smoothing is used.

5.8.3.2.2 Construction of partial redundant frame for Unvoiced Coding mode

The low bit-rate Noise Excited Linear Prediction coding scheme (subclause 5.2.5.4) is used to construct a partial redundant copy for an unvoiced frame type (RF_NELP). In Unvoiced coding mode, the TBE gain frame has a wider dynamic range. To preserve this dynamic range, the TBE gain frame quantization in Unvoiced coding mode uses a similar quantization range as that of the one used in the primary frame.

5.8.3.2.3   Construction of partial redundant frame for TCX frame

In case of TCX partial redundant frame type, a partial copy consisting of some helper parameters is used to enhance the frame loss concealment algorithm. There are three different partial copy modes available, which are RF_TCXFD, RF_TCXTD1 and RF_TCX_TD2. Similar to the PLC mode decision on the decoder side, the selection of the partial copy mode for TCX is based on various parameters such as the mode of the last two frames, the frame class, LTP pitch and gain.

5.8.3.2.3.1   Frequency domain concealment (RF_TCXFD) partial redundant frame type

29 bits are used for the RF_TCXFD partial copy mode.

- 13bits are used for the LSF quantizer which is the same as used for regular low rate TCX coding.

- The global TCX gain is quantized using 7 bits.

- The classifier info is coded on 2 bits.

5.8.3.2.3.2   Time domain concealment (RF_TCXTD1 and RF_TCXTD2) partial redundant frame type

The partial copy mode RF_TCXTD1 is selected if the frame contains a transient or if the global gain of the frame is much lower than the global gain of the previous frame. Otherwise RF_TCXTD2 is chosen.

Overall 18bits of side data are used for both modes.

- 9bits are used to signal the TCX LTP lag

- 2 bits for signalling the classifier info

5.8.3.2.4   RF_NO_DATA partial redundant frame type

This is used to signal a configuration where the partial redundant copy is not sent and all bits are used towards primary frame coding.

The primary frame bit-rate reduction and partial redundant frame coding mechanisms together determine the bit-rate allocation between the primary and redundant frames to be included within a 13.2 kbps payload.

5.8.3.3   Decoding

At the receiver, the de-jitter buffer provides a partial redundant copy of the current lost frame if it is available in any of the future frames. If present, the partial redundant information is used to synthesize the lost frame. In the decoding, the partial redundant frame type is identified and decoding performed based on whether only one or more adaptive codebook parameters, only one or more fixed codebook parameters, or one or more adaptive codebook parameters and one or more fixed codebook parameters, TCX frame loss concealment helper parameters, or Noise Excited Linear Prediction parameters are coded. If either the current frame or the previous frame adjacent to the current frame is a partial redundant frame, then the decoding parameter of the current frame, such as the LSP parameters, the gain of the adaptive codebook or the BWE gain, is firstly obtained and then post-processed according to decoding parameters or signal type from previous frames relative to the current frame position, or future frames relative to the current frame position. Additionally, the spectral tilt is also used for the post-procession. The post-processed parameters are used to reconstruct the output signal. Finally, the frame is reconstructed based on the coding scheme. The TCX partial info is decoded, but in contrast to ACELP partial copy mode, the decoder is run in concealment mode. The difference to regular concealment is just that the parameters available from the bitstream are directly used and not derived by concealment.

5.8.4   Channel aware mode encoder configurable parameters

The channel aware mode encoder may use the following configurable parameters to adapt its operation to track the channel characteristics seen at the receiver. These parameters maybe computed at the receiver (as described in subclause 6.3) and communicated to the encoder via a receiver triggered feedback mechanism.

Partial redundancy offset (o): The difference in time units between the transmit time of the primary copy of a frame (n) and the transmit time of the redundant copy of that frame which is piggy backed onto a future frame ($n + X$) is called the partial redundancy offset or FEC offset X . The optimal partial redundancy offset that maximizes the probability of partial redundant copy retrival may be computed in the decoder JBM solution and fed back to the encoder as described above.

Frame erasure rate indicator (p) having the following values: LO (low) for FER rates <5% or HI (high) for FER>5%. This parameter controls the threshold used to determine whether a particular frame is critical or not as described in subclause 5.8.3.2. Such an adjustment of the criticality threshold is used to control the frequency of partial copy transmission. The HI setting adjusts the criticality threshold to classify more frames as critical to transmit as compared to the LO setting.

It is noted that these encoder configurable parameters are optional with default set to $p = HI$ and $o = 3$.

6   Functional description of the Decoder

6.1   LP-based Decoding

6.1.1   General LP-based decoding

The LSF parameters are decoded from the received bitstream and converted to LSP coefficients and subsequently to LP coefficients. The interpolation principle, described in subclause 5.1.9.6, is used to obtain interpolated LSP vectors for all subframes, i.e. 4 subframes in case of 12.8 kHz internal sampling rate and 5 subframes in case of 16 kHz sampling rate. Then, the excitation signal is reconstructed and post-processed before performing LP synthesis (filtering with the LP synthesis filter) to obtain the reconstructed signal. The reconstructed signal is then de-emphasized (an inverse of the pre-emphasis applied at the encoder). Finally, a post-processing is applied for enhancing the format and harmonic structure of signal as well as the periodicity in the low frequency region of the signal. The signal is then up-sampled to the output sample rate. Finally, the high-band signal is generated and added to the up-sampled synthesized signal to obtain a full-band reconstructed signal (output signal).

6.1.1.1   LSF decoding

6.1.1.1.1   General LSF decoding

Depending on the predictor allocation per mode, like specified at encoder side in subclause 5.2.2.1.3 one first bit is read to select between safety net or predictive mode for the switched safety net/predictive cases. The bit value of one corresponds to safety net and value zero corresponds to predictive mode. The following bits are read in groups of a number equal to the stage sizes corresponding to each coding mode as specified in subclause 5.2.2.1.4 and the codevectors are retrieved from the corresponding codebooks. The last bits correspond to the lattice codevector, having the index I. The LSF residual after the first non-structured, optimized VQ was quantized by splitting the vector into two subvectors. The index I was obtained as a combined index of the two indexes corresponding to the first and the second subvector. The two indexes are retrieved as follows:

$$I_1 = \left[I / N_2 \right] \tag{1409}$$

$$I_2 = I - \left[I / N_2 \right] \tag{1410}$$

These indexes correspond each to a scale index, leader class index and leader vector permutation index. For each of the two indexes corresponding to the 8-dimensional subvectors the following operations are applied. The scale offset is determined by finding out the largest scale offset that is smallest than the index $I_i, i = 1,2$. The corresponding scale offset is removed from each of $I_i, i = 1,2$. Similarly the leader offset is calculated and removed for each of the two indexes. The index of the scale offset gives the index of the scale, j_i, and the index of the leader offset gives the index of the leader class, k_i. The remaining index values are $I'_i, i = 1,2$. The sign index, $I_{si}, i = 1,2$ and the leader index $I_{li}, i = 1,2$ are obtained

$$I_{si} = \left[I'_i / \pi_0(k_i) \right] \quad i = 1,2 \tag{1411}$$

$$I_{li} = I'_i - \left[I'_i / \pi_0(k_i) \right] \quad i = 1,2 \tag{1412}$$

where $\pi_0(k_i)$ is the cardinality of unsigned permutations for the leader class k_i, given in subclause 5.2.2.1.4. The indexes $I_{li}, i = 1,2$ and $I_{si}, i = 1,2$ are decoded using the position decoding based on counting the binomial coefficients and the sign decoding described in [26].

Decoding of the index corresponding to the unsigned permutation of the leader vector goes as follows. Knowing the leader class index, the number of distinct non zero values and the amount of each of these values which are tabulated (see subclause 5.2.2.1.4) can be determined. The used leader classes defined in subclause 5.2.2.1.4 have at most 4 distinct values. If there is a single value, v_0, in the leader class corresponding to the decoded leader class index, all decoded vector components have the same value

$$x(j) = v_0, \, j = 0,..., S-1 \tag{1413}$$

where $S = 8$ is the subvector dimension.

If there are two distinct values v_0, v_1, in the decoded leader vector, each appearing k_0 and k_1 times respectively, the decoded vector is initialized with

$$x(j) = v_1, \, j = 0,..., k_1 - 1. \tag{1414}$$

The leader vector permutation index is interpreted using binomial coefficients decoding. The positions of the k_0 values v_0 are determined within a vector of length S. The position of the first v_0, $p[0]$ is determined such that

$$\binom{S-1}{k_0-1} + \binom{S-2}{k_0-1} + ... + \binom{S-p[0]}{k_0-1} \le I_{li} < \binom{S-1}{k_0-1} + \binom{S-2}{k_0-1} + ... + \binom{S-1-p[0]}{k_0-1}. \tag{1415}$$

If $I_{li} < \binom{S-1}{k_0-1}$ then $p[0] = 0$. The position of the second v_0 value, $p[1]$, is determined similarly for an updated index

$$I_{li1} = I_{li} - \binom{S-1}{k_0-1} - \binom{S-2}{k_0-1} - ... - \binom{S-p[0]}{k_0-1} \tag{1416}$$

an updated number vector length, $S - p[0] - 1$ instead of S, and an updated number of values, $k_0 - 1$ instead of k_0.

The procedure follows until the positions of all v_0 values are determined. Once these positions are known the values v_0 are inserted in the vector x at the corresponding positions.

If there are 3 distinct values v_0, v_1, v_2 having k_0, k_1, k_2 number of occurrences respectively, the decoded vector is initialized with:

$$x(j) = v_2, \, j = 0,..., k_2 - 1. \tag{1417}$$

Out of $I_{li}, i = 1,2$, two subindexes are obtained:

$$L_{i1} = \left\lceil \frac{I_{li}}{\binom{k_2 + k_1}{k_1}} \right\rceil, i = 1,2. \tag{1418}$$

$$L_{i2} = I_{li} - \left\lceil \frac{I_{li}}{\binom{k_2 + k_1}{k_1}} \right\rceil \binom{k_2 + k_1}{k_1}, i = 1,2. \tag{1419}$$

The positions of the values v_1 are determined by binomial decoding of the index L_{i2} considering k_1 positions out of $k_1 + k_2$ and the values v_1 are inserted in the vector x. The decoding is performed according to equations (1208) and (1209). The positions for values v_0 are obtained by binomial decoding of the index L_{i1}, considering k_0 positions out of S.

If there are 4 distinct values the vector is initialized with

$$x(j) = v_3, \, j = 0,..., k_3 - 1. \tag{1420}$$

The index $I_{li}, i = 1,2$ is divided into:

$$L_{i1} = \left\lfloor \frac{I_{li}}{\binom{k_2 + k_3}{k_2}} \right\rfloor, i = 1,2 .$$ (1421)

$$L_{i2} = I_{li} - \left\lfloor \frac{I_{li}}{\binom{k_2 + k_3}{k_2}} \right\rfloor \binom{k_2 + k_3}{k_2}, i = 1,2 .$$ (1422)

$$L_{i3} = \left\lfloor \frac{L_{i1}}{\binom{k_3 + k_2 + k_1}{k_1}} \right\rfloor, i = 1,2 .$$ (1423)

$$L_{i4} = L_{i1} - \left\lfloor \frac{L_{i1}}{\binom{k_3 + k_2 + k_1}{k_1}} \right\rfloor \binom{k_3 + k_2 + k_1}{k_1}, i = 1,2 .$$ (1424)

The positions for values v_2 are obtained by binomial decoding the index L_{i2} for k_2 position out of $k_2 + k_3$. The positions for values v_1 are obtained by binomial decoding of the index L_{i3} for k_1 positions out of $k_1 + k_2 + k_3$. The positions for values v_0 are obtained by binomial decoding of the index L_{i4} for k_0 positions out of S.

The obtained subvectors are multiplied with the corresponding scales and component wise multiplied with the off-line computed standard deviations. The standard deviations are individually estimated for each coding mode and bandwidth. The result corresponds to the codevector from the last stage of the LSF quantizer. The codevectors from all stages are added together.

If the coding mode corresponds to a safety net only mode, or if it corresponds to a switched safety net/AR predictive mode and the safety net mode has been selected at the encoding stage, a vector representing the component wise mean for the current coding mode is added to the sum of codevectors and the result represents the decoded LSF vector. The decoded LSF vector is thus given by:

$$\hat{f}_t(i) = \sum_{k=0}^{N} l_k(i) + m(i) , \text{ for } i = 0,\ldots, M \text{ -1}$$ (1425)

where $\hat{f}_t(i), i = 0, M-1$ is the LSF vector for current frame t, $l_k(i), i = 0,...,M-1$ $l_k(i)$, i=0, M-1 is the codevector obtained at stage k out of the N quantization stages and $m(i), i = 0,...,M-1$ is the mean LSF vector for the current coding mode.

If AR predictive mode was selected at the encoding stage, the decoded LSF vector is given by:

$$\hat{f}_t(i) = \hat{f}_{t-1}(i) + \sum_{k=0}^{N} l_k(i) + m(i) , \text{ for } i = 0,\ldots, M \text{ -1}.$$ (1426)

If MA predictive mode was selected at the encoding stage, based on the coding mode, the decoded LSF vector is given by:

$$\hat{f}_t(i) = \hat{e}_{t-1}(i) + \sum_{k=0}^{N} l_k(i) + m(i) \text{, for } i = 0, \ldots, M-1. \tag{1427}$$

where $\hat{e}_{t-1}$ is the quantization error at the previous frame $t-1$.

6.1.1.1.2 LSF decoding for voiced coding mode at 16 kHz internal sampling frequency

The VC mode at the 16 kHz internal sampling frequency has two decoding rates: 31 bits per frame and 40 bits per frame. The VC mode is decoded by a 16-state and 8-stage BC-TCVQ. figure 88 shows the decoder of the predictive BC-TCVQ with safety-net using an encoding rate of 31 bits. The 31bit LSF decoding performed by the predictive BC-TCVQ with safety-net proceeds as follows. First, one bit is decoded at the *Scheme selection* block. This bit defines whether the predictive scheme or the safety-net scheme is used.

For the safety-net scheme, $\hat{z}_k(i)$ is decoded by equation (1428),

$$\hat{z}_k(i-1) = \hat{t}_k(i-1) + \mathbf{A}_{i-1}\hat{z}_k(i-2) \text{, for } i = 2, \ldots, M/2 \tag{1428}$$

where the prediction residual, $t_k(i)$, is decoded by the 1st BC-TCVQ.

If the predictive scheme is used, the prediction vector $p_k(i)$ is obtained using (1429):

$$p_k(i) = \rho(i)\hat{z}'_{k-1}(i) \text{, for } i = 0, \ldots, M-1 \tag{1429}$$

where $\rho(i)$ are the selected AR prediction coefficients for the VC mode at 16kHz isf, M is the LPC order, and $\hat{z}'_{k-1}(i) = [\hat{z}^t_{k-1}(0), \hat{z}^t_{k-1}(1), \ldots, \hat{z}^t_{k-1}(M/2-1)]$.

The decoding of $\hat{r}_k(i)$ is performed as given by equation (1430),

$$\hat{r}_k(i-1) = \hat{t}_k(i-1) + \mathbf{A}_{i-1}\hat{r}_k(i-2) \text{, for } i = 2, \ldots, M/2 \tag{1430}$$

where the prediction residual, $t_k(i)$, is decoded by the 2nd BC-TCVQ.

The quantized LSF vector $\hat{f}_k(i)$ for the predictive scheme is calculated by equation (1431),

$$\hat{f}_k(i) = p_k(i) + m(i) + \hat{r}'_k(i) \text{, for } i = 0, \ldots, M-1 \tag{1431}$$

where $m(i)$ is the mean vector for VC mode and $\hat{r}'_k(i) = [\hat{r}^t_k(0), \hat{r}^t_k(1), \ldots, \hat{r}^t_k(M/2-1)]$

The quantized LSF vector $\hat{f}_k(i)$ for the safety-net scheme is calculated by equation (1432).

$$\hat{f}_k(i) = m(i) + \hat{z}'_k(i) \text{, for } i = 0, \ldots, M-1 \tag{1432}$$
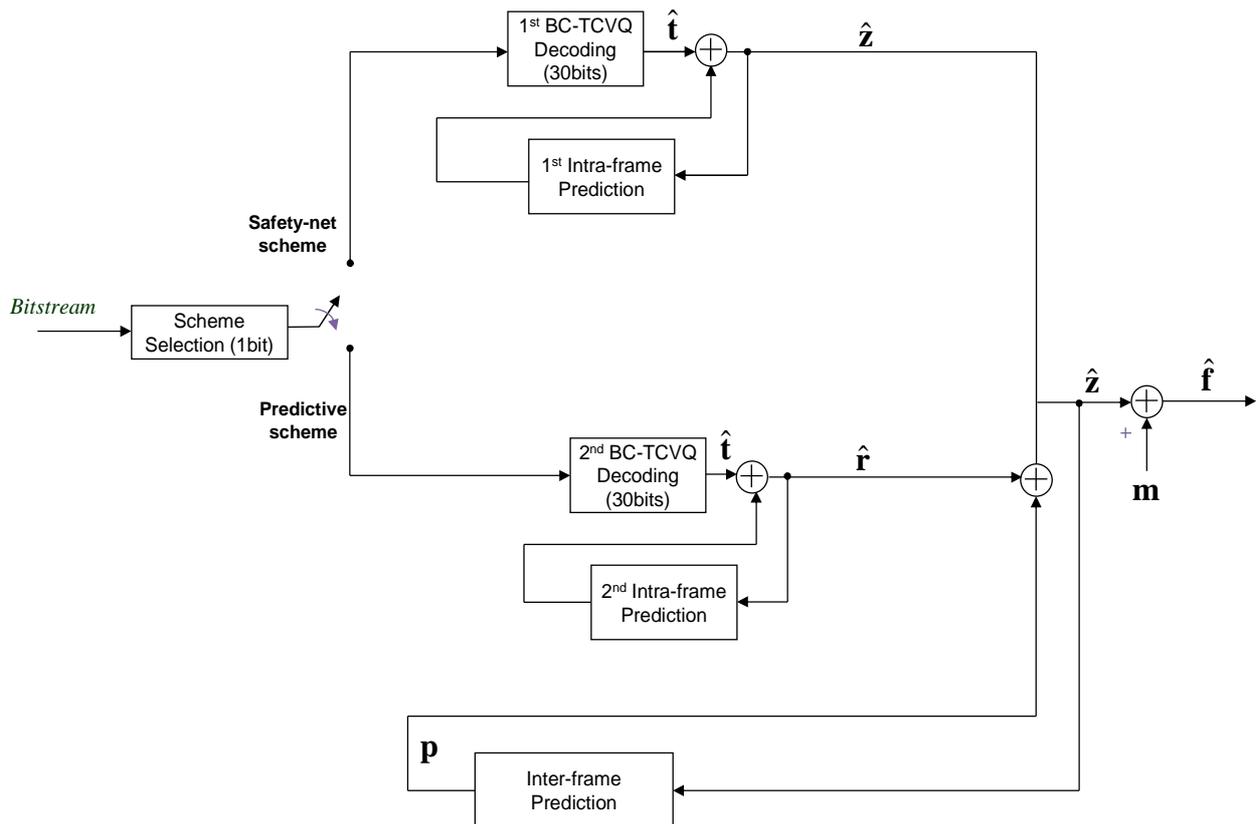
Figure 88: Block diagram of the decoder for the predictive BC-TCVQ with safety-net for an encoding rate of 31 bits per frame

Figure 89 shows the decoder of the predictive BC-TCVQ with safety-net for an encoding rate of 40 bits per frame. The 40-bit LSF decoding using the predictive BC-TCVQ with safety-net is performed as follows. The scheme selection and the decoding method of BC-TCVQ for both the predictive and safety-net schemes are the same as those of the 31-bit LSF decoding. $\hat{z}_2(i)$ and $\hat{r}_2(i)$ are decoded by the 3rd and 4th SVQ decoding respectively. The quantized LSF vector $\hat{f}_k(i)$ for the predictive scheme is calculated according to equation (1433),

$$\hat{f}_k(i) = p_k(i) + m(i) + \hat{r}_1(i) + \hat{r}_2(i), \text{ for } i = 0, \dots, M-1 \tag{1433}$$

where $\hat{r}_1(i)$ is the output of the 2nd BC-TCVQ and the 2nd intra-frame prediction.

The quantized LSF vector $\hat{f}_k(i)$ for the safety-net scheme is calculated by equation (1434),

$$\hat{f}_k(i) = m(i) + \hat{z}_1(i) + \hat{z}_2(i), \text{ for } i = 0, \dots, M-1 \tag{1434}$$

where $\hat{z}_1(i)$ is the output of the 1st BC-TCVQ and 1st intra-frame prediction.
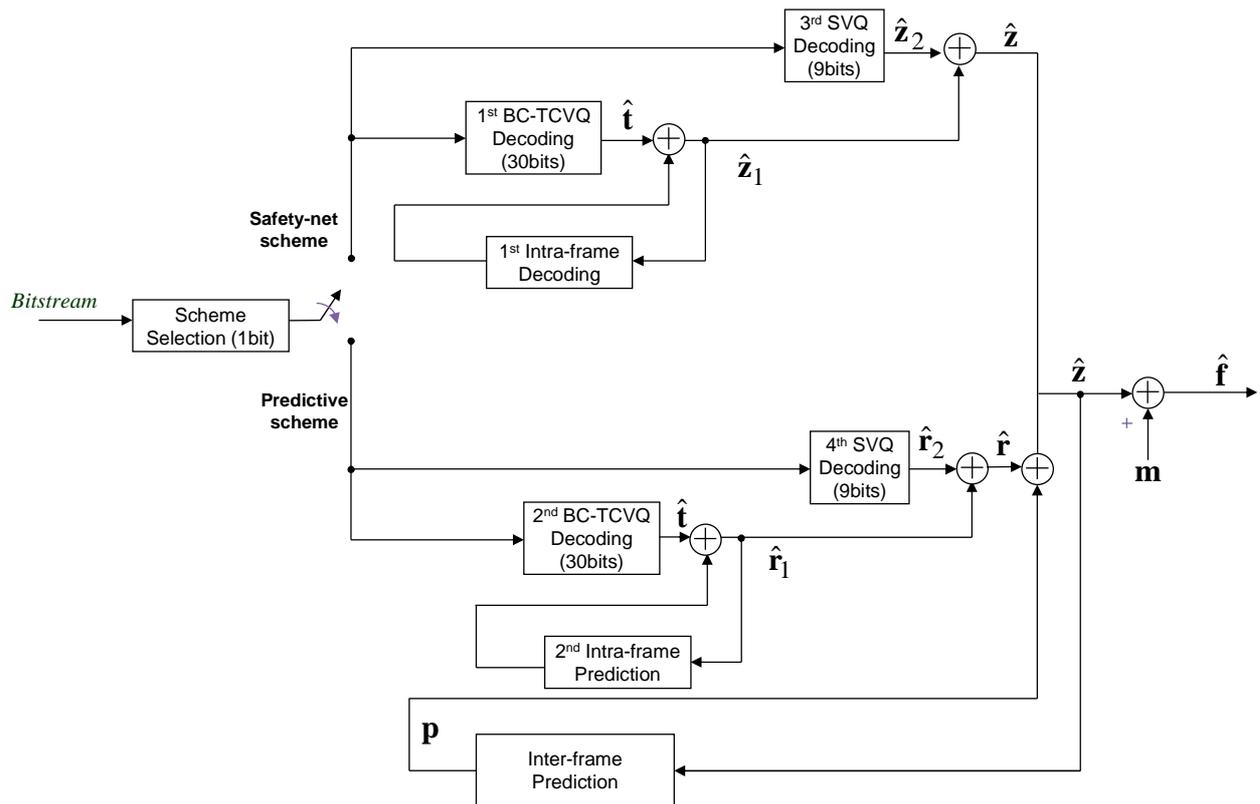
Figure 89: Block diagram of the decoder for the predictive BC-TCVQ/SVQ with safety-net for an encoding rate of 40 bits per frame

6.1.1.2 Reconstruction of the excitation

6.1.1.2.1 Reconstruction of the excitation in GC and VC modes and high rate IC/UC modes

6.1.1.2.1.1 Decoding the adaptive codebook vector

The received adaptive codebook parameters (or pitch parameters) are the closed-loop pitch, T_{CL}, and the pitch gain, $\hat{g}_p$ (adaptive codebook gain), transmitted for each subframe, serve to compute the adaptive codevector, $v(n)$.

6.1.1.2.1.2 Pulse index decoding of the 43-bit algebraic codebook

The joint indexing decoding procedure of three pulses on two tracks is described as follows:
    In the decoder side, the de-indexing procedure is as below for $Q(Q = 3)$ pulses, M positions on the track:

1   24 bits are extracted from the received bit-stream and then decoded as the temporary index $temp\_I$. If $temp\_I$ is smaller than *THR* which is the same as the encoder side, the joint index $Jo\text{int}\_index$ equals to $temp\_I$. If $temp\_I$ is bigger than or equal to *THR*, 1 more bit will be extracted from the bit-stream as *Bit*. Then the global index $temp\_I$ is adjusted as: $temp\_I = (temp\_I << 1) + Bit$. Then the joint index $Jo\text{int}\_index$ is computed by subtracting *THR* from $temp\_I$:

$$Jo\text{int}\_index = temp\_I - THR \qquad (1435)$$

2   Decompress the joint index $Jo\text{int}\_index$ into the two index for each track:

$$ind_1 = Jo\text{int}\_index / W_2 \qquad (1436)$$

$$ind_2 = Jo\text{int}\_index \% W_2 \qquad (1437)$$

3   Decoding the index for each track(ind_1 and ind_2) as below:
    1)   determining the quantity of pulse positions according to the first index $I_1(N)$

As the offset index $I_1(N)$ is saved in a table (available in encoder and decoder), and each offset index in the table indicates the unique number of pulse positions in the track. So $I_1(N)$ can be decoded from the index easily. Then the number of pulse position N, the sign index $I_4(N)$ and I_{23} are obtained.

2) As we know the number of pulse position N and index I_{23}, the index $I_2(N)$ and $I_3(N)$ can be decoded based on permutation method from the index I_{23}, and each pulse position is also decoded from $I_2(N)$ and $I_3(N)$. Separating and obtaining the second index $I_3(N)$ and the third index $I_3(N)$ in the following way:

$$I_2(N) = I_{23} \% C_M^N \qquad (1438)$$

$$I_3(N) = \text{Int}[I_{23} / C_M^N] \qquad (1439)$$

wherein $I_2(N)$ represents the second index, $I_3(N)$ represents the third index, N represents the quantity of the positions with pulse on it, % refers to taking the remainder, and "Int" refers to taking the integer

3) determining the distribution of the positions with a pulse on the track according to the second index; the $I_2(N)$ is obtained, the following calculation process is applied at the decoder:

(1) C_{M-1}^{N-1}, ..., and C_{M-y0}^{N-1} are subtracted from $I_2(N)$ one by one.

$$R(y0) = I_2(N) - C_{M-1}^{N-1} - ... - C_{M-y0}^{N-1} \qquad (1440)$$

until the $I_2(N)$ remainder $R(y0)$ changes from a positive number to a negative number, where M is the total quantity of positions on the track, N is the quantity of positions with pulses, $y0 \in [1, M-N-1]$, and C refers to calculating the combination function. The $p(0)$, namely, the serial number of the first position with a pulse(s) on the position, is recorded, where $p(0) = y0 - 1$.

(2) If $N > 1$, $C_{M-p(0)-1}^{N-2}$, ..., and $C_{M-p(0)-y1}^{N-2}$ are further subtracted from $R(y0)$ one by one until the $R(y0)$ remainder $R1(y1)$ changes from a positive number to a negative number. The $p(1)$ namely, the serial number of the second position with a pulse(s) on the position, is recorded, where $p(1) = y1 - 1$.

(3) And so on, $C_{M-p(0)-...-p(n-1)-1}^{N-n-2}$, ..., and $C_{M-p(0)-...-p(n-1)-yn}^{N-n-2}$ are further subtracted from $R(n-1)[p(n-1)]$ one by one until the $R(n-1)[p(n-1)]$ remainder $Rn(yn)$ changes from a positive number to a negative number, where $n \le N-1$. The $p(n)$ namely, the serial number of the n+1 position with a pulse(s) on the position, is recorded, where $p(n) = yn - 1$.

(4) The decoding of the $I_2(N)$ is completed, and $P(N) = \{p(0), p(1),..., p(N-1)\}$ is obtained.

4) determining the quantity of pulses in each position with pulses according to the third index; For each track, according to the third index $I_3(N)$, determine the number of pulses on each position that has a pulse. the $I_3(N)$ is obtained, the following calculation process is applied at the decoder:

(1) $T[q(0)] = I_3(N) - C_{PPT}^{\Delta N} - C_{PPT-q(0)}^{\Delta N}$ is calculated from a smaller $q(0)$ value to a greater $q(0)$ value, where: $q(0) \in [0, N-1]$, $\Delta N = Q - N$, $PPT = Q - 1$, and C refers to calculating the combination function. The last $q(0)$ value that lets $T[q(0)]$ be greater than zero is recorded as the position $v0$ of the first pulse on the track.

(2) If $\Delta N > 1$, $T1[q(1)] = T(v0) - (C_{PPT-1-v0}^{\Delta N} - C_{PPT-1-q(1)}^{\Delta N})$ is further calculated from a smaller $q(1)$ value to a greater $q(1)$ value, where $q(1) \in [v0, N-1]$; and the last $q(1)$ value that lets $T1[q(1)]$ be greater than zero is recorded as the position $v1$ of the second pulse on the track.

(3) By analogy, $Th[q(h)] = T(h-1)(q(h-1) - (C_{PPT-h-q(h-1)}^{\Delta N} - C_{PPT-h-q(h)}^{\Delta N})$ is calculated from a smaller $q(h)$ value to a greater $q(h)$ value, where: $q(h) \in [v(h-1), N-1]$, and $h \in [2, \Delta N-1]$; and the last $q(h)$ value that lets $Th[q(h)]$ be greater than zero is recorded as the position vh for the $(h+1)^{th}$ pulse(h+1 is an ordinal number) on the track.

(4) The decoding of the $I_3(N)$ is completed, and $SU'(N) = \{q(0), q(1),..., q(\Delta N-1)\}$ is obtained.

5) After obtain $SU'(N) = \{q(0), q(1),..., q(\Delta N-1)\}$, mean on each position $q(0) \le q(1) \le,..., \le q(\Delta N-1)$ have a pulse, if $q(i) = q(i+1)$, mean on the position $q(i)$ have more pulses. The $SU'(N)$ is the result after subtract value "1" from the number of pulses in each pulse position, so value "1" is need to be added back to N

position, and $SU(N)$ is rebuilt as following

$$su(i) = 1, 0 \leq i < N - 1$$

$$For(i = 0; i < \Delta N; i++)$$

$$su(q(i)) = su(q(i)) + 1$$

6) By now all the pulse positions, the quantity of pulses in each pulse position and associated signs are decoded, so the pulses on each track is reconstructed.

6.1.1.2.1.3 Mulit-track joint decoding `of` pulse indexing

All the muti-track joint decoding step is described as following:

1) extracting the $final\_index_0$, $final\_index_1$, $final\_index_2$, $final\_index_3$ and $track\_hi$ from the stream;

2) Get the parameter from the table 35 according to the pulse number of each track, include the index $bits_t$, $Hi\_Bit\_bits_t$, $Hi\_Bit\_range_t$, $re\-back\_bits_t$,

3) Extract h_0 and $track_0\_low$ from $final\_index_0$, extract h_1 and $track_1\_low$ from $final\_index_1$, extract h_2 and $track_2\_low$ from $final\_index_2$, extract h_3 and $track_3\_low$ from $final\_index_3$.

4) From $track\_hi$, h_0, h_1, h_2 and h_3, hi_0, hi_1, hi_2 and hi_3 are decoded out.

(1) The $track\_hi$ is combined with h_3 and obtain hi_{SLP2H}, hi_{SLP2H} is combined with h_2 and obtain hi_{SLP2}, then hi_3 can be get as following:

$$hi_3 = hi_{SLP2} \% Hi\_Bit\_range_3 \tag{1441}$$

$$hi_{SLP1H} = hi_{SLP2} / Hi\_Bit\_range_3 \tag{1442}$$

(2) The hi_{SLP1H} is combined with h_1 and obtain hi_{SLP1}, then hi_2 can be get as following:

$$hi_2 = hi_{SLP1} \% Hi\_Bit\_range_2 \tag{1443}$$

$$hi_{SLP0H} = hi_{SLP1} / Hi\_Bit\_range_2 \tag{1444}$$

(3) The hi_{SLP0H} is combined with h_0 and obtain hi_{SLP0}, then hi_0, hi_1 can be get as following:

$$hi_1 = hi_{SLP0} \% Hi\_Bit\_range_1 \tag{1445}$$

$$hi_0 = hi_{SLP0} / Hi\_Bit\_range_1 \tag{1446}$$

5) Combine hi_0, hi_1, hi_2, hi_3 with $track_0\_low$, $track_1\_low$, $track_2\_low$, $track_3\_low$, and get the index of each track.

6.1.1.2.1.4 Decoding the algebraic codebook vector

The received algebraic codebook index is used to extract the positions and amplitudes (signs) of the excitation pulses and to find the algebraic codevector $c(n)$. If the integer part of the pitch lag is less than the subframe size 64, the pitch sharpening procedure is applied, which translates into modifying $c(n)$ by filtering it through the adaptive pre-filter $F^{(0)}(z) = (1 - \beta_1 z^{-1})/(1 - 0.85z^{-T})$ which further consists of two parts: a periodicity enhancement part $1/(1 - 0.85z^{-T})$, where T is the integer part of the pitch lag representing the fine spectral structure of the speech signal, and a tilt part $(1 - \beta_1 z^{-1})$, where β_1 is related to the voicing of the previous subframe and is bounded by [0.28, 0.56] at 16.4 and 24.4 kbps, and by [0.0; 0.5] otherwise.

The periodicity enhancement part of the filter colours the spectrum by damping inter-harmonic frequencies, which are annoying to the human ear in case of voiced signals.

Depending on bitrates and coding mode, and the estimated level of background noise, the adaptive pre-filter also includes a filter based on the spectral envelope, which colours the spectrum by damping frequencies between the formant regions. The final form of the adaptive pre filter $F(z)$ is given by

$$F(z) = F^{(0)}(z) \frac{\hat{A}(z/\eta_1)}{\hat{A}(z/\eta_2)} \tag{1447}$$

where $\eta_1 = 0.75$ and $\eta_2 = 0.9$ if $sr_{celp} = 12800\,\text{Hz}$ and $\eta_1 = 0.8$ and $\eta_2 = 0.92$ if $sr_{celp} = 16000\,\text{Hz}$.

6.1.1.2.1.5 Decoding of the combined algebraic codebook

At 32 kbps and 64 kbps bit-rates, the pre-quantizer excitation contribution is obtained from the received pre-quantizer parameters as follows. The contribution from the pre-quantizer is obtained by first de-quantizing the decoded (quantized) spectral coefficients using an AVQ decoder and applying the iDCT to these de-quantized spectral coefficients. Further the pre-emphasis filter $1/F_p(z)$ is applied after the iDCT to form the pre-quantizer contribution $q(n)$. The pre-quantizer contribution $q(n)$ then scales using the quantized pre-quantizer gain $\hat{g}_q$ to form the pre-quantizer excitation contribution.

The same above procedure applies for decoding GC, TC and IC mode at 32 kbps and 64 kbps with the exception of non-harmonic signals at 32kbps GC mode where the iDCT stage is omitted. It is noted that at the decoder, the order of codebooks and corresponding codebook stages during the decoding process is not important as a particular codebook contribution does not depend on or affect other codebook contributions. Thus the codebook arrangement in the IC mode is identical to the GC mode codebook arrangement. The pre-quantizer gain $\hat{g}_q$ in GC and TC mode is obtained by

$$\hat{g}_q = \hat{E}_i \cdot \hat{g}_{q,norm} \tag{1448}$$

where $\hat{g}_{q,norm}$ is the decoded normalized pre-quantizer gain and $\hat{E}_i$ predicted algebraic codevector energy.

In IC mode, the de-quantizer gain is obtained by

$$\hat{g}_q = \hat{g}_c \cdot \hat{g}_{q,norm} \tag{1449}$$

where $\hat{g}_c$ is the quantized algebraic codebook gain.

6.1.1.2.1.6 AVQ decoding

The reading of the AVQ parameters from the bitstream is complementary to the insertion described in subclause 5.2.3.1.6.9.3. The codebook numbers n_j are used to estimate the actual bit-budget needed to encode AVQ parameters at the decoder and the number of unused AVQ bits is computed as a difference between the allocated and actual bit budgets.

6.1.1.2.1.6.1 Decoding of AVQ parameters

The parameters decoding involves decoding the AVQ parameters describing each 8-dimensional quantized sub-bands $\hat{\mathbf{S}}'(8j)$ of the quantized spectrum $S'(k)$. The $\hat{\mathbf{S}}'(8j)$ comprise several sub-bands (8 in case of combined algebraic codebook), each of 8 samples. The decoded AVQ parameters for each sub-band $\hat{\mathbf{S}}'(8j)$ comprise:

– the codebook number n_j,

– the vector index I_j,

– and, if the codevector (i.e. lattice point) is not in a base codebook, the Voronoi index $\mathbf{I}_j^v$.

The unary code for the codebook number n_j, is first read from the bitstream and n_j is determined. From the codebook number n_j, the base codebook and the Voronoi extension order r_j^v are then obtained. If $n_j < 5$, there is no Voronoi extension ($r_j^v = 0$) and the base codebook is Q_{n_j}. If $n_j > 5$ the base codebook is either Q_3 (n_j even) or Q_4 (n_j odd) and the Voronoi order (1 or 2) is also determined ($r_j^v = 1$ if $n_j < 7$; $r_j^v = 2$, otherwise).

Then, if $n_j > 0$, the vector index I_j, coded on $4n_j$ bits is read from the bitstream and the base codevector $\mathbf{z}_j$ is decoded.

After the decoding of the base codevector, if the Voronoi order r_j^v is greater than 0, the Voronoi extension index $\mathbf{I}_j^v$ is decoded to obtain the Voronoi extension vector $\mathbf{v}_j$. The number of bits in each component of index vector $\mathbf{I}_j^v$ is given by the Voronoi extension order r_j^v, and the scaling factor M_j^v of the Voronoi extension is given by $M_j^v = 2^{r_j^v}$.

Finally, from the scaling factor M_j^v, the Voronoi extension vector $\mathbf{v}_j$ and the base codebook vector $\mathbf{z}_j$, each 8-dimensional AVQ sub-band $\hat{\mathbf{S}}'(8j)$ is computed as:

$$\hat{\mathbf{S}}'(8j) = M_j^v \cdot \mathbf{z}_j + \mathbf{v}_j \tag{1450}$$

In case of decoding the pre-quantizer, resp. de-quantizer, contribution from subclause 6.1.1.2.1.3, the decoded sub-band blocks of $\hat{\mathbf{S}}'(k)$ corresponds to the decoded spectrum coefficients $\hat{Q}_d(k)$, resp. $\hat{U}_d(k)$.

6.1.1.2.1.6.2 De-indexing of codevector in base codebook

The index decoding of the codevector $\mathbf{z}_j$ is done in several steps. First, the absolute leader and its offset are identified by comparing the index with the offset in the look-up table. The offset is subtracted from the index to produce a new index. From this index, the sign index and the absolute vector index are extracted. The sign index is decoded and the sign vector is obtained. The absolute vector index is decoded by using a multi-level permutation-based index decoding method and the absolute vector is obtained. Finally, the decoded vector is reconstructed by combining the sign vector with the absolute vector.

6.1.1.2.1.6.2.1 Sign decoding

The sign vector is obtained by extracting from left to right all the sign bits for non-zero elements in the absolute vector. The bit number of the sign code is read from the (S_n). If the bit number of the sign index is not equal to the number of the non-zero elements in the decoded absolute vector, the sign of the last non-zero element is recovered.

6.1.1.2.1.6.2.2 Decoding of the absolute vector and of its position vector

The decoding method of the absolute vector index is described as follows:

1) The absolute vector index is decomposed into several mid-indices for each level from lowest level to highest level. The absolute vector index is the starting value for the lowest level. The mid-index of each lower level is obtained by dividing the absolute vector index by the possible index value count, $C_{m_{n-1}}^{m_n}$, the quotient is the absolute vector index for the next lower level. The remainder is the middle index, $I_{mid,n}$, for the current level.

2) The $I_{mid,n}$ of each lower level is decoded based on a permutation and combination function and the position vector of each lower level vector related to its upper level vector is obtained.

Finally, one-by-one from the lowest level to the highest level, each lower level absolute vector is used to partly replace the upper level absolute vector elements according to the position parameter. The highest level vector is the decoded output absolute vector. A example of the L_1 absolute vector partly replace the L_0 absolute vector elements is give as following:

The position vector	(	1,		4,		6	)	

$$\downarrow \qquad\qquad \downarrow \qquad \downarrow$$

| The element position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

The original absolute vector of L_0
(upper level vector)

$$(0 \quad 2, \quad 0, \quad 0, \quad 4, \quad 0, \quad 6, \quad 0)$$

Some elements are replaced by L_1 vector
which position is indicated in the position vector

$$(0 \quad 2, \quad 0, \quad 0, \quad 4, \quad 0, \quad 6, \quad 0)$$

$$\uparrow \qquad\qquad \uparrow \qquad \uparrow$$

The new absolute vector of L_1
(lower level vector)

$$(\quad 2, \quad\quad 4, \quad\quad 6 \quad)$$
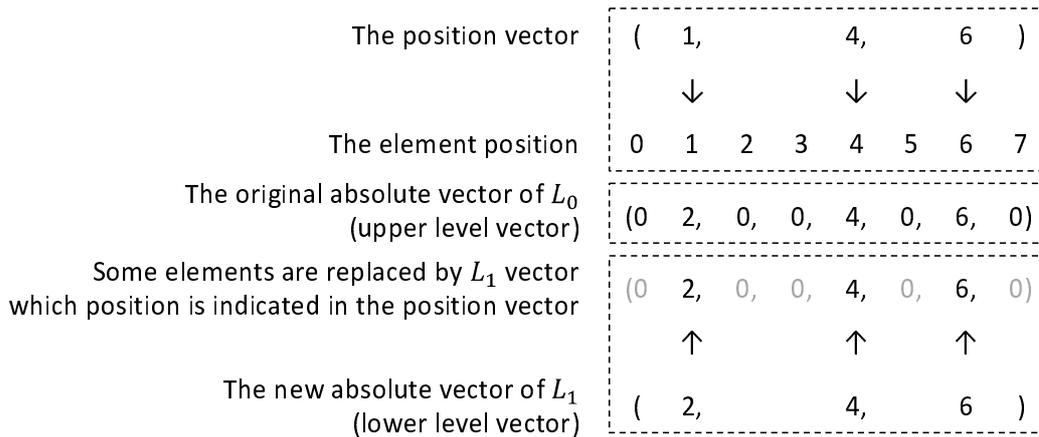
Figure 90: Replacing example between L_1 and L_0 for $K_a = 20$.

6.1.1.2.1.6.2.3   Position vector decoding

To obtain the position vector from the middle index in each lower level, the algorithm uses a permutation and combination procedure to estimate the position sequence. The procedure is as follows:

1)   Increment the pos value beginning from zero, until $I_{mid,n}$ is not more than $C_{m_{n-1}}^{m_n} - C_{m_{n-1}-pos}^{m_n}$.

2)   Let $q_0 = pos - 1$ be the first position, and subtract $C_{m_{n-1}}^{m_n} - C_{m_{n-1}-q_0}^{m_n}$ from the I_{mid}.

3)   Increase pos, beginning from $q_{i-1}+1$, until I_{mid} is not more than $C_{m_{n-1}-q_{i-1}-1}^{m_n-i} - C_{m_{n-1}-pos}^{m_n-i}$, where q_{i-1} is the position decoded at the previous step.

4)   Let $q_i = pos - 1$ be the position number i, and subtract $C_{m_{n-1}-q_{i-1}-1}^{m_n-i} - C_{m_{n-1}-q_i}^{m_n-i}$ from the I_{mid}.

5)   Repeat steps 3 and 4 until all positions are decoded for the current level position sequence.

6.1.1.2.1.6.2.4   Absolute vector decoding

For the lowest level, the absolute vector only includes one type of element whose value can be obtained from the decomposition order column in the table of subclause 5.2.3.1.6.9.3.2. The lowest level absolute vector is passed to the next level and at the next step another type of element is added. This new element is obtained from the decomposition order column in the table of subclause 5.2.3.1.6.9.3.2. This procedure is repeated until the highest level is reached.

6.1.1.2.1.6.2.5   Construction of the output codevector in base codebook

Constructing the 8-dimensional output codevector in the base codebook is the final step of the decoding procedure. The codevector $\mathbf{z}_j$ is obtained by combining the sign vector with the absolute vector. If the bit number of the sign index is not equal to the number of the non-zero elements in the decoded absolute vector, the sign of the last non-zero element is recovered. The recovery rule, based on the RE_8 lattice property, is as follows: if the sum of all output vector elements is not an integer multiple of 4, the sign of the last element is set to negative.

6.1.1.2.1.7   Decoding the gains

6.1.1.2.1.7.1   Decoding memory-less coded gains

Before calculating the adaptive and algebraic codebook gain in each subframe, the predicted algebraic codevector energy, $\hat{E}_i$, is decoded for the whole frame.

Now, let E_c denote the algebraic codebook excitation energy in dB in a given subframe, which is given by

$$E_c = 10\log\left(\frac{1}{64}\sum_{i=0}^{63}c^2(i)\right) \tag{1451}$$

In the equation above, $c(i)$ is the pre-filtered algebraic codevector.

A predicted algebraic codebook gain is then calculated as

$$g'_c = 10^{0.05(\hat{E}_i - E_c)} \tag{1452}$$

An index is then retrieved from the bitstream representing a jointly-quantized adaptive codebook gain along with a correction factor. The quantized adaptive codebook gain, $\hat{g}_p$, is retrieved directly from the codebook and the quantized algebraic codebook gain is given by

$$\hat{g}_c = \hat{\gamma}\,g'_c \tag{1453}$$

where $\hat{\gamma}$ is the decoded correction factor.

Note that no prediction based on parameters from past frames is used. This increases the robustness of the codec to frame erasures.

6.1.1.2.1.7.2   Decoding memory-less joint coded gains at lowest bit-rates

For the lowest bitrates of 7.2 and 8.0 kbps, slightly different memory-less joint gain coding scheme is used.

Similarly as in the encoder, the estimated (predicted) gain of the algebraic codebook in the first subframe is given by

$$g_{c0}^{[0]} = 10^{a_0 + a_1 CT - \log_{10}(\sqrt{E_c})} \tag{1454}$$

where CT is the coding mode, selected for the current frame in the pre-processing part, and E_c is the energy of the filtered algebraic codevector. The inner term inside the logarithm corresponds to the gain of innovation vector. The only parameter in the equation above is the coding mode CT which is constant for all subframes of the current frame. The superscript [0] denotes the first subframe of the current frame.

In all subframes following the first subframe the estimated value of the algebraic codebook gain is given by

$$g_{c0}^{[k]} = 10^{b_0 + b_1 CT + \sum_{i=1}^{k} b_{i+1}\log_{10}(g_c^{[i-1]}) + \sum_{i=1}^{k} b_{k+1+i} g_p^{[i-1]}} \tag{1455}$$

where k=1,2,3. Note, that the terms in the first and in the second sum of the exponent, there are quantized gains of algebraic and adaptive excitation of previous subframes, respectively. Note that the term including the gain of innovation vector $\log_{10}(\sqrt{E_c})$ is not subtracted. The reason is in the use of the quantized values of past algebraic codebook gains which are already close enough to the optimal gain and thus it is not necessary to subtract this gain again.

The gain de-quantization in the decoder is done by retrieving the codevector [g_p ; γ] according to the index receing in the bitstream. The quantized value of the fixed codebook gain is then calculated as

$$g_c = g_{c0}\cdot\gamma \tag{1456}$$

6.1.1.2.1.7.3   Decoding scalar coded gains at highest bit-rates

As described in subclause 6.1.1.2.1.3.1, before calculating the adaptive and algebraic codebook gain in each subframe, the predicted algebraic codevector energy, $\hat{E}_i$, is decoded for the whole frame. Then two indexes are retrieved from the bitstream and used to decode the adaptive codebook gain and a correction factor. The decoded algebraic codebook gain is further obtained using equation (1453).

6.1.1.2.1.8 Reconstructed excitation

The total excitation in each subframe is constructed by

$$u'(n) = \hat{g}_p \, v(n) + \hat{g}_c \, c(n), \qquad \text{for } n = 0,\ldots,63 \tag{1457}$$

where $c(n)$ is the pre-filtered algebraic codevector.

In case that combined algebraic codebook is used, the total excitation is each subframe is constructed by

$$u'(n) = \hat{g}_p \, v(n) + \hat{g}_c \, c(n) + \hat{g}_q \, q(n), \qquad \text{for } n = 0,\ldots,63 \tag{1458}$$

The excitation signal, $u'(n)$, is used to update the contents of the adaptive codebook for the next frame. The excitation signal, $u'(n)$, is then post processed as described in subclause 7.1.2.4 to obtain the post-processed excitation signal $u(n)$, which is finally used as an input to the synthesis filter $1/\hat{A}(z)$.

6.1.1.2.2 Reconstruction of the excitation in TC mode

In TC mode, the TC frame configuration (subclause 6.8.4.2.2) is decoded first. Then, the adaptive excitation signal is either a zero vector, a glottal-shape codevector or an adaptive codebook vector. In a subframe where the glottal-shape codebook is used, the reconstruction of the glottal-shape codevector is done using the received TC parameters as described in subclause 6.8.4.2.1. In a subframe where the adaptive codebook is used, the adaptive codevector is found as described in subclause 7.1.2.1.1. In all subframes after the one where the glottal-shape codebook is used, a low-pass filtering is applied and the filtered adaptive excitation is found as $v(n) = 0.18v'(n) + 0.64v'(n-1) + 0.18v'(n-2)$.

If a subframe contains a zero adaptive excitation vector, only the algebraic codebook gain is decoded using a 2-bit or 3-bit scalar quantizer (described in subclause 6.8.4.2.4). Otherwise, the adaptive and algebraic codebook gains are decoded as in GC and VC modes (described in subclause 7.1.2.1.3).

Finally, the reconstructed excitation is computed as described in subclause 7.1.2.1.4.

6.1.1.2.3 Reconstruction of the excitation in UC mode at low rates

6.1.1.2.3.1 Decoding the innovative vector

In UC mode, the signs and indices of the two random vectors are decoded and the excitation is reconstructed as in subclause 5.2.3.3.1. The correction of the random codebook tilt is used as described in subclause 5.2.3.3.2.

6.1.1.2.3.2 Decoding the random codebook gain

In UC mode, only the random codebook gain is transmitted. The received index k gives the gain in dB, $\hat{g}_c^{dB}$, using the relations and quantization step defined in subclause 5.2.3.3.4. The values $\Gamma_{\max}$ and δ given in subclause 5.2.3.3.4. The quantized gain, $\hat{g}_c$, is then given according to subclause 5.2.3.3.4.

6.1.1.2.3.3 Enhancement of background noise

The anti-swirling technique is applied in inactive periods, at 9.6 kb/s for NB signals, and 9.6 kb/s and below for WB and SWB signal. This technique is based on the decoded SAD and noisiness parameters. Basically, the anti-swirling effect is achieved by means of LP parameter smoothing in combination with reducing the power variations and spectral fluctuations of the excitation signal during detected periods of signal inactivity.

6.1.1.2.3.3.1 LP parameter smoothing

The LP parameter smoothing is done in two steps. First, a low-pass filtered set of LSP parameters is calculated by first-order autoregressive filtering according to

$$\bar{q}_{end}^{[0]} = \lambda \, \bar{q}_{end}^{[-1]} + (1 - \lambda) \, \hat{q}_{end}^{[0]} \tag{1459}$$

Here $\overline{q}_{end}^{[0]}$ represents the low-pass filtered frame-end LSP parameter vector obtained for the current frame, $\hat{q}_{end}^{[0]}$ is the decoded frame-end LSP parameter vector for the current frame, and $\lambda = 0.9$ is a weighting factor controlling the degree of smoothing.

In a second step, a weighted combination between the low-pass filtered LSP parameter vector, $\overline{q}_{end}^{[0]}$, and the decoded LSP parameter vectors, $\hat{q}_{end}^{[-1]}$, $\hat{q}_{end}^{[0]}$ and $\hat{q}_{mid}^{[0]}$, is calculated using a weighting factor β. That is

$$
\begin{aligned}
q'^{[-1]}_{end} &= (1-\beta)\,\overline{q}^{[-1]}_{end} + \beta\,\hat{q}^{[-1]}_{end} \\
q'^{[0]}_{mid} &= (1-\beta)\,0.5\left(\overline{q}^{[-1]}_{end} + \overline{q}^{[0]}_{end}\right) + \beta\,\hat{q}^{[0]}_{mid} \\
q'^{[0]}_{end} &= (1-\beta)\,\overline{q}^{[0]}_{end} + \beta\,\hat{q}^{[0]}_{end}
\end{aligned}
\tag{1460}
$$

As mentioned in subclause 7.1.1, LSP interpolation is performed to obtain four LSP vectors, each for an individual subframe. This interpolation is based on: the decoded frame-end LSP parameter vector   of the previous frame, the decoded mid-frame LSP parameter vector   in the current frame and the decoded frame-end LSP parameter vector   of the current frame. Subsequently, instead of using these parameters, their smoothed versions, given in equation (1460) are employed.

It is noteworthy that the degree of smoothing is controlled by means of the control factor β, which is described in subclause 6.1.1.2.3.3.3.

6.1.1.2.3.3.2 Modification of the excitation signal

One essential element of the anti-swirling technique is the reduction of power and spectrum fluctuations of the signal during periods of signal inactivity.

In the first step, tilt compensation of the excitation signal is performed with a first-order tilt compensation filter given as

$$
H(z) = 1 - \kappa\,z^{-1}
\tag{1461}
$$

The coefficient κ is calculated as

$$
\kappa = \frac{r_e(1)}{r_e(0)}
\tag{1462}
$$

where $r_e(0)$ and $r_e(1)$ are the zero-th and the first autocorrelation coefficients of the original excitation signal. The tilt compensation is carried out on a subframe basis.

In the second step, the spectral fluctuations of the excitation signal are further reduced by replacing a part of it with a white noise signal. To this end, first a properly scaled random sequence of unit variance is generated. This signal is then scaled by means of a gain factor, $\tilde{g}$, in such a way that its power equals the smoothed power of the excitation signal. The gain factor, $\tilde{g}$, is obtained by filtering the RMS value of the excitation signal, denoted as g_{RMS}, on a frame-by-frame basis. That is

$$
\tilde{g} = 0.9\,\tilde{g} + 0.1\,g_{RMS}
\tag{1463}
$$

The noise is scaled by multiplying all its samples by the gain factor $\tilde{g}$. Then, with some weighting factor, α, the excitation signal, $e(n)$, is combined with the scaled noise signal, denoted as $r(n)$. This is done according to the following equation leading to the smoothed excitation signal $\hat{e}(n)$:

$$
\hat{e}(n) = \frac{\alpha}{\sqrt{\alpha^2 + (1-\alpha)^2}}\,e(n) + (1-\alpha)r(n), \qquad \text{for } n = 0,\ldots,255
\tag{1464}
$$

It is noteworthy that the degree of excitation signal smoothing is controlled by means of the control factor α, which is described in subclause 6.1.1.2.3.3.3.

6.1.1.2.3.3.3 Controlling the background noise smoothing

The anti-swirling method described in the clauses above is controlled by means of the control parameters α and β in response to the received SAD and noisiness parameters.

First, the received and decoded noisiness parameter steers an intermediate smoothing control parameter, γ, such that it is ensured that the degree of smoothing is only increased gradually up to a maximum degree that is indicated by the received parameter. Given the received noisiness parameter, $\hat{v}$, an intermediate parameter, $\gamma^{[0]}$, is set according to the following relation:

$$\gamma^{[0]} = \max\left(\hat{v}, \gamma^{[-1]} - \delta\right) \tag{1465}$$

where $\gamma^{[-1]}$ is the stored intermediate control parameter from the previous frame and $\delta = 0.05$ is the step-size with which the smoothing control parameters are steered towards $\hat{v}$ as long as they are greater than $\hat{v}$. In case the current frame is erased ($f_{bfi} = 1$), $\gamma^{[0]}$ is set to the intermediate control parameter of the previous frame, $\gamma^{[-1]}$.

The SAD parameter activates the smoothing operation only when the received SAD flag, f_{SAD}, indicates inactivity. However, in order to decrease the risk that smoothing is enabled during active signal periods, erroneously declared as inactive, the background noise smoothing is only enabled after a hangover period of 5 frames. Further, whenever the SAD declares a frame as active, the smoothing operation is disabled. In order to avoid adding a new hangover period after spurious SAD activation, no hangover is added if the detected activity period is less or equal to 3 frames.

In addition to this SAD-driven activation, for quality reasons, it is important to avoid the anti-swirling operation being turned on too abruptly. To this end, after each hang-over period, a phase in period of $K = 5$ frames is applied, during which the smoothing operation is gradually steered from inactivate to fully enabled. Accordingly, for the n-th frame of the phase-in period, the smoothing control parameters α and β are calculated as follows:

$$\alpha = \beta = 1 + \frac{(\gamma - 1)n}{K} \tag{1466}$$

For all other frames (during which the smoothing is activated) $\alpha = \gamma$ and $\beta = \gamma$.

It is noteworthy that phase-in periods are only inserted after hangover periods, i.e., not after spurious SAD activations of less than 3 frames.

6.1.1.2.4 Reconstruction of the excitation in IC/UC mode at 9.6 kbps

6.1.1.2.4.1 Decoding of the innovative excitation

In IC and UC modes at 9.6 kbps the decoding the algebraic codebood ecitation is the same as described in n subclause 6.1.1.2.1.2.

At WB, an additional Gaussian noise excitation is generated as described in subclause 5.2.3.4.2.

6.1.1.2.4.2 Gains decoding

In NB, only the algebraic codeword gain $\hat{g}_c$ is calculated as

$$\hat{g}_c = 10^{(\hat{g}_c^{dB} - E_c)/20} \tag{1467}$$

The algebraic codebook excitation energy in dB, E_c, is computed as in equation (1451). The quantized gain in dB is given by

$$\hat{g}_c^{dB} = k \times 1.9 - 30 \tag{1468}$$

The quantization index k (6 bits) is retrieved directly from the bitstream (subclause 5.2.3.4.3.2)..

For WB the quantized algebraic codeword gain $\hat{g}_c$ and Gaussian noise excitation gain $\hat{g}_{c2}$ are decoded. They are calculated respectively as

$$\hat{g}_c = 10^{\hat{g}_c^{dB}/20} \tag{1469}$$

$$\hat{g}_{c2} = (0.25k2 + 0.25) \cdot \hat{g}_c \cdot \frac{\sum_{n=0}^{63}\sqrt{c^2(n)}}{\sum_{n=0}^{63}\sqrt{c2^2(n)}} \tag{1470}$$

The quantized gain in dB is given by

$$\hat{g}_c^{dB} = k \times 1.25 - 20 - E_c + \hat{E}_i \tag{1471}$$

The quantization index k (5 bits) and $k2$ (2 bits) are retrieved from the bitstream. The predicted algebraic codevector energy, $\hat{E}_i$, is decoded for the whole frame prior to calculating the algebraic codebook gain in each subframe (subclause 5.2.3.4.3.2).

6.1.1.2.4.4 Total excitation

The total excitation in each subframe is constructed by

$$u'(n) = \hat{g}_c\,c(n) + \hat{g}_{c2}\,c2(n), \qquad for\ n = 0,\ldots,63 \tag{1472}$$

where $c(n)$ and $c2(n)$ are the pre-filtered algebraic codevector and the pre-filtered Gaussian noise excitation respectively.

Only the algebraic codevector $\hat{g}_c\,c(n)$ is used to update the contents of the adaptive codebook for the next frame.

The excitation signal, $u'(n)$, is then post processed as described in subclause 6.1.1.3 to obtain the post-processed excitation signal $u(n)$, which is finally used as an input to the synthesis filter $1/\hat{A}(z)$.

6.1.1.2.5 Reconstruction of the excitation in GSC

In GSC mode, the attack flag is first decoded (subclause 5.1.13.5.3). Then, the number of subframe is decoded. To do so, if the bit rate is 13.2 kbit/s and the coding mode is INACTIVE, the first step is to decode 1 bit to verify if the coded frame is a SWB unvoiced frame which would implies 4 subframes. Otherwise when the number of subframe is less than 4, the noise level N_{lev} as defined in subclause 5.2.3.5.4 is decoded. If the bitrate is 13.2 kbit/s, then a supplementary bit is decoded to determine if the number of subframe is 1 or 2, for lower bitrate the number of subframe is 1.

Then the cut off frequency (as defined in subclause 5.2.3.5.6) is decoded and if it is different from 0, the time domain contribution is decoded (subclause 5.2.3.5.2). When a time domain contribution exists, it is converted in frequency domain and low pass filtered using the decoded cut-off frequency as described in subclause 5.2.3.5.6, otherwise the time domain contribution is set to 0.

Then the frequency domain component is decoded starting the gain of sub bands as defined in subclause 5.2.3.5.7. The gain information is then used to determine the bit allocation, the number of bands and the order of the bands to be decoded by the PVQ as described in subclause 5.2.3.5.8. Then the PVQ is decoding the spectral difference and spectral dynamic control and noise filling is applied on the decoded vector as described in subclause 5.2.3.5.10. When the spectral difference vector is complete, the gain is applied and it is combined, in the frequency domain, with the temporal contribution as described in subclause 5.2.3.5.11. If the decoded frequency excitation meets the given condition, predict the un-decoded frequency excitation by the decoded frequency excitation as described in subclause 5.2.3.5.12. The complete excitation in the frequency domain is revert back to time domain using the inverse DCT as described in subclause 5.2.3.5.12 and then a pre-echo removal is applied as in subclause 5.2.3.5.13 to get the total excitation $u'(n)$.

6.1.1.3　Excitation post-processing

Before the synthesis, a post-processing of the excitation signal, $u'(n)$, is performed to form the updated excitation signal, $u(n)$, as follows.

6.1.1.3.1　Anti-sparseness processing

An adaptive anti-sparseness post-processing procedure is applied to the pre-filtered algebraic codevector, $c(n)$. This is to reduce the perceptual artefacts arising from the sparseness of algebraic codebook vectors having only a few non-zero samples per subframe. The anti-sparseness processing consists of circular convolution of the algebraic codevector with an impulse response by means of an FFT. Three pre-stored impulse responses are used and a selection number $i_p = 0$, 1 or 2 and is set to select one of them. A value of 2 or greater corresponds to no modification; a value of 1 corresponds to medium modification and a value of 0 corresponds to strong modification. The selection of the impulse response is performed adaptively based on the decoded adaptive codebook gain, $\hat{g}_p$, coding mode and bit rate.

The following selection procedure is employed where $\hat{g}_c^{[-1]}$ is the algebraic codebook gain in the previous subframe, $\hat{g}_p^{[-k]}$ are current and 5 previous subframes' adaptive codebook gains and $i_p^{[-1]}$ is the previous selection number.

$initialize\ i = 2$
$if\ (!UC \wedge bitrate \leq 7200) \vee (UC \wedge (bitrate \geq 9600 \wedge bitrate \leq 16400))$
　$i = 0$
$else\ if\ !VC \wedge !UC \wedge bitrate \leq 9600\ then$
　$i = 1$
$if\ \hat{g}_p < 0.6\ then$
　$i_p = 0$
$else\ if\ \hat{g}_p < 0.9\ then$
　$i_p = 1$
$else$
　$i_p = 2$
$if\ \hat{g}_c > 3\hat{g}_c^{[-1]}\ then$
　$i_p = i_p + 1$
$else$
　$initialize\ j = 0$
　$for\ k = 0.5$
　　$if\ \hat{g}_p^{[-k]} > 0.6\ then$
　　　$j = j + 1$
$if\ j > 2\ then$
　$i_p = 0$
$if\ i_p - i_p^{[-1]} > 1\ then$ 　　　　　　　　　　　　　　　　　　　　　　　(1473)
　$i_p = i_p - 1$
$update\ i_p^{[-1]} = i_p$
$compute\ the\ final\ selection\ number\ as\ i_p = i_p + i$

6.1.1.3.2　Gain smoothing for noise enhancement

A nonlinear gain smoothing technique is applied to the algebraic codebook gain, $\hat{g}_c$, in order to enhance the excitation in noise. Based on the stability and voicing of the signal segment, the gain of the algebraic codebook vector is smoothed

in order to reduce fluctuation in the energy of the excitation in case of stationary signals. This improves the performance in case of stationary background noise. The voicing factor λ is given by

$$\lambda = 0.5(1 - r_v) \tag{1474}$$

with r_v giving a measure of signal periodicity

$$r_v = \frac{(E_v - E_C)}{(E_v + E_C)} \tag{1475}$$

where E_v and E_C are the energies of the scaled pitch codevector and scaled algebraic codevector, respectively. Note that since the value of r_v is between –1 and 1, the value of λ is between 0 and 1. Note that the factor λ is related to the amount of "unvoicing" with a value of 0 for purely voiced segments and a value of 1 for purely unvoiced segments.

A stability factor θ is computed based on a distance measure between the adjacent LP filters. Here, the factor θ is related to the LSF distance measure. The LSF distance is given by

$$LSF_{dist} = \sum_{i=0}^{14} \left[f^{[0]}(i) - f^{[-1]}(i) \right]^2 \tag{1476}$$

where $f^{[0]}(i)$ in the present frame, calculated in subclause 7.1.1, and $f^{[-1]}(i)$ are the LSFs in the previous frame. The stability factor θ is given by

$$\theta = 1.25 - LSF_{dist}/400000, \; constrained \; by \; 0 \le \theta \le 1 \tag{1477}$$

The LSF distance measure is smaller in case of stable signals. As the value of θ is inversely related to the LSF distance measure, then larger values of θ correspond to more stable signals. The gain smoothing factor, S_m, is given by

$$S_m = \lambda\theta \tag{1478}$$

The value of S_m approaches 1 for unvoiced and stable signals, which is the case of stationary background noise signals. For purely voiced signals, or for unstable signals, the value of S_m approaches 0. An initial modified gain, $g^{[0]}$, is computed by comparing the algebraic codebook gain, $\hat{g}_c$, to a threshold given by the initial modified gain from the previous subframe, $g^{[-1]}$. If $\hat{g}_c$ is larger than or equal to $g^{[-1]}$, then $g^{[0]}$ is computed by decrementing $\hat{g}_c$ by 1.5 dB, constrained by $g^{[0]} \ge g^{[-1]}$. If $\hat{g}_c$ is smaller than $g^{[-1]}$, then $g^{[0]}$ is computed by incrementing $\hat{g}_c$ by 1.5 dB, constrained by $g^{[0]} \le g^{[-1]}$. Finally, the algebraic codebook gain is modified using the value of the smoothed gain as follows

$$\hat{g}_c = S_m g^{[0]} + (1 - S_m)\hat{g}_c \tag{1479}$$

6.1.1.3.3 Pitch enhancer

A pitch enhancer scheme modifies the total excitation $u'(n)$ of voiced signals by filtering the algebraic codebook excitation through an innovation filter. The filter frequency response emphasizes the higher frequencies and reduces the energy of the low-frequency portion of the innovative codevector. The filter coefficients are related to the periodicity of the signal. Therefore, the pitch enhancer is not applied to excitation in UC at low bit rates, i.e. birates < 9600 kb/s.

A filter of the form

$$F_{inno}(z) = -c_{pe}z + 1 - c_{pe}z^{-1} \tag{1480}$$

is used where $c_{pe} = 0.125(1 - r_v)$ if $sr_{celp} = 12800 \, \text{Hz}$ and $c_{pe} = 0.15(1 - r_v)$ if $sr_{celp} = 16000 \, \text{Hz}$, with r_v being a periodicity factor given in equation (1475). The filtered algebraic codebook vector in the current subframe is given by

$$c'(n) = c(n) - c_{pe}[c(n+1) + c(n-1)] \quad for \; n = 0,\dots,63 \tag{1481}$$

where the out-of-subframe samples $c(-1)$ and $c(64)$ are set to zero. The updated post-processed excitation is given by

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c'(n) \tag{1482}$$

The above procedure can be done in one step by updating the excitation as follows

$$u(n) = u'(n) - \hat{g}_c c_{pe}(c(n+1) + c(n-1)) \tag{1483}$$

where $\hat{g}_c$ is the modified algebraic codebook gain from equation (1479).

6.1.1.3.4   Music post processing

In case of a sound signals coded with the GSC, a music enhancer scheme modifies the total excitation $u(n)$ corresponding to the sound signal in such a way that the quantization noise inserted between spectral tones during the encoding/decoding process can be reduced. The music enhancer consists of converting the decoded excitation into frequency domain, computing a weighting mask for retrieving spectral information lost in the quantization noise, and modifying the frequency domain excitation by applying the weighting mask to increase the spectral dynamics, and converting the modified frequency domain excitation back to time domain.

The current frequency domain post processing achieves higher frequency resolution, without adding delay to the synthesis. A weighting mask is created based on the past spectrum energy and used to improve the efficiency of the inter-tone noise removal. To achieve this post processing without adding delay to the codec, a symmetric trapezoidal window is used. It is centred on the current frame where the window is flat, and extrapolation is used to create the future signal. The advantage of working on the excitation signal rather than on the synthesis signal is that any potential discontinuities introduced by the post processing are smoothed out by the subsequent application of the LP synthesis filter. The following text describes the implementation of the music post processing.

6.1.1.3.4.1   Excitation buffering and extrapolation

To increase the frequency resolution, a frequency transform longer than the frame length is used. To do so, a concatenated excitation vector $u_c(n)$ is created by concatenating the last 192 samples of the previous frame excitation, the decoded excitation of the current frame $u(n)$, and an extrapolation of 192 excitation samples of the future frame $u_x(n)$. This is described below where L_w is the length of the past excitation as well as the length of the extrapolated excitation, and L is the frame length. These correspond to 192 and 256 samples respectively, giving the total length $L_c = 640$ samples:

$$u_c(n) = \begin{cases} u(n) & n = -L_w, \ldots -1 \\ u(n) & n = 0, \ldots, L-1 \\ u_x(n) & n = L, \ldots, L + L_w - 1 \end{cases} \tag{1484}$$

The extrapolation of the future excitation samples $u_x(n)$ is computed by periodically extending the current frame excitation signal $u(n)$ using the decoded factional pitch of the last subframe of the current frame. Given the fractional resolution of the pitch lag, an upsampling of the current frame excitation is performed using a 35 samples long Hamming windowed sinc function.

6.1.1.3.4.2   Windowing and frequency transform

Prior to the time-to-frequency transform a windowing is performed on the concatenated excitation. The selected window $u(wn)$ has a flat top corresponding to the current frame, and it decreases with the Hanning function to 0 at each end. The following equation represents the window used:

$$w(n) = \begin{cases} 0.5\left(1 - \cos\left(\dfrac{2\pi(n + L_w)}{2L_w - 1}\right)\right) & n = -L_w, \ldots -1 \\ 1.0 & n = 0, \ldots, L-1 \\ 0.5\left(1 - \cos\left(\dfrac{2\pi((n-L) + L_w)}{2L_w - 1}\right)\right) & n = L, \ldots, L + L_w - 1 \end{cases} \tag{1485}$$

When applied to the concatenated excitation, an input to the frequency transform having a total length $L_c = 640$ samples ($L_c = 2L_w + L$) is obtained in the prototype. The windowed concatenated excitation $u_{wc}(n)$ is centered on the current frame and is represented with the following equation:

$$u_{wc}(n) = \begin{cases} u(n)w(n) & n = -L_w, \ldots -1 \\ u(n)w(n) & n = 0, \ldots, L-1 \\ u_x(n)w(n) & n = L, \ldots, L + L_w - 1 \end{cases} \tag{1486}$$

During the frequency-domain post processing phase, the concatenated excitation is represented in a transform-domain using a type II DCT giving a resolution of 10Hz. The frequency representation of the concatenated and windowed time-domain CELP excitation f_u is given below:

$$f_u(k) = \begin{cases} \sqrt{\dfrac{1}{L_c}} \cdot \displaystyle\sum_{n=0}^{L_c-1} u_{wc}(n), & k = 0 \\ \sqrt{\dfrac{2}{L_c}} \cdot \displaystyle\sum_{n=0}^{L_c-1} u_{wc}(n) \cdot \cos\left(\dfrac{\pi}{L_c}\left(n + \dfrac{1}{2}\right)k\right), & 1 \le k \le L_c - 1 \end{cases} \tag{1487}$$

where $u_{wc}(n)$, is the concatenated and windowed time-domain excitation and L_c is the length of the frequency transform. The frame length L is 256 samples, but the length of the frequency transform L_c is 640 samples for a corresponding inner sampling frequency of 12.8 kHz.

6.1.1.3.4.3 Energy per band and per bin analysis

After the DCT, the resulting spectrum is divided into critical frequency bands. The critical frequency bands used in the prototype are as close as possible to what is specified in [17], and their upper limits are defined as follows:

$$\begin{aligned} C_B = \{ &100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, \ldots \\ &1720, 2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400 \}\, \text{Hz} \end{aligned} \tag{1488}$$

The 640-point DCT results in a frequency resolution of 10 Hz (6400Hz/640pts). The number of frequency bins per critical frequency band is

$$M_B = \{10, 10, 10, 10, 11, 12, 14, 15, 16, 19, 21, 24, 28, 32, 38, 45, 55, 70, 90, 110\} \tag{1489}$$

The average spectral energy per critical frequency band $E_B(i)$ is computed as follows:

$$E_B(i) = \frac{1}{L_c M_{CB}(i)} \sum_{h=0}^{M_B(i)-1} \left(f_u(h + j_i)^2\right), \qquad i = 0, \ldots, 20 \tag{1490}$$

where $f_e(h)$ represents the hth frequency bin of a critical band and j_i is the index of the first bin in the $\underline{i}^{\text{th}}$ critical band given by

$$j_i = \{0, 10, 20, 30, 40, 51, 63, 77, 92, 108, 127, 148, 172, 200, 232, 270, 315, 370, 440, 530\} \tag{1491}$$

The spectral analysis also computes the energy of the spectrum per frequency bin, $E_{BIN}(k)$ using the following relation:

$$E_{BIN}(k) = \frac{1}{L_c} f_u(k)^2, \qquad k = 0,...,639 \qquad (1492)$$

Finally, the spectral analysis computes a total spectral energy E_C of the concatenated excitation as the sum of the spectral energies of the first 17 critical frequency bands using the following relation:

$$E_C = 10\log_{10}\left(\sum_{i=0}^{16} E_B(i)\right) - 3.0103 \qquad (1493)$$

6.1.1.3.4.4 Excitation type classification

The method for enhancing decoded generic sound signal includes an additional analysis of the excitation signal designed to maximize the efficiency of the inter-harmonic noise reducer by identifying which frame is well suited for the inter-tone noise reduction.

This classifier not only separates the decoded concatenated excitation into sound signal categories, but it also gives instructions to the inter-harmonic noise reducer regarding the maximum level of attenuation and the minimum frequency where the reduction can starts.

The first operation consists in performing an energy stability analysis based on the total spectral energy of the concatenated excitation E_C:

$$\overline{E}_d = \frac{\left(\displaystyle\sum_{t=-40}^{t=-1} \Delta^t_{E_C}\right)}{40}, \qquad where \ \Delta^t_{E_C} = E_C^t - E_C^{(t-1)} \qquad (1494)$$

where $\overline{E}_d$ represents the average difference of the energies of the concatenated excitation vectors of two adjacent frames, E_C^t represents the energy of the concatenated excitation of the current frame t, and $E_C^{(t-1)}$ represents the energy of the concatenated excitation of the previous frame $t-1$. The average is computed over the last 40 frames.

Then, a statistical deviation σ_C of the energy variation over the last fifteen (15) frames is calculated using the following relation:

$$\sigma_C = p \cdot \sqrt{\sum_{t=-15}^{t=-1} \frac{\left(\Delta^t_{E_C} - \overline{E}_d\right)^2}{15}} \qquad (1495)$$

where, in the prototype, the scaling factor p is found experimentally and set to about 0.77. The resulting deviation σ_C is compared to four (4) floating thresholds to determine to what extend the noise between harmonics can be reduced. The output of this second stage classifier is split into five (5) sound signal categories e_{CAT}, named sound signal categories 0 to 4. Each sound signal category has its own inter-tone noise reduction tuning.

The five (5) sound signal categories 0-4 can be determined as indicated in the following table.

Table 156: Output characteristic of the excitation classifier

Category (e_{CAT})	Enhanced band (Hz)	Allowed reduction (dB)
0	NA	0
1	[510, 6400]	6
2	[510, 6400]	9
3	[400, 6400]	12
4	[300, 6400]	12

The sound signal category 0 is a non-tonal, non-stable sound signal category which is not modified by the inter-tone noise reduction technique. This category of the decoded sound signal has the largest statistical deviation of the spectral energy variation and in general comprises speech signal.

Sound signal category 1 (largest statistical deviation of the spectral energy variation after category 0) is detected when the statistical deviation of spectral energy variation history is lower than Threshold 1 and the last detected sound signal category is ≥ 0. Then the maximum reduction of quantization noise of the decoded tonal excitation within the frequency band 510 to $F_S/2$ Hz is limited to a maximum noise reduction R_{max} of 6 dB.

Sound signal category 2 is detected when the statistical deviation of spectral energy variation is lower than Threshold 2 and the last detected sound signal category is ≥ 1. Then the maximum reduction of quantization noise of the decoded tonal excitation within the frequency band 510 to $F_S/2$ Hz is limited to a maximum of 9 dB.

Sound signal category 3 is detected when the statistical deviation of spectral energy variation is lower than Threshold 3 and the last detected sound signal category is ≥ 2. Then the maximum reduction of quantization noise of the decoded tonal excitation within the frequency band 4000 to $F_S/2$ Hz is limited to a maximum of 12 dB.

Sound signal category 4 is detected when the statistical deviation of spectral energy variation is lower than Threshold 4 and when the last detected signal type category is ≥ 3. Then the maximum reduction of quantization noise of the decoded tonal excitation within the frequency band 300 to $F_S/2$ Hz is limited to a maximum of 12 dB.

The floating thresholds 1-4 help preventing wrong signal type classification. Typically, decoded tonal sound signal representing music gets much lower statistical deviation of its spectral energy variation than speech. However, even music signal can contain higher statistical deviation segment, and similarly speech signal can contain segments with lower statistical deviation. It is nevertheless unlikely that speech and music contents change regularly from one to another on a frame basis. The floating thresholds add decision hysteresis and act as reinforcement of previous state to substantially prevent any misclassification that could result in a suboptimal performance of the inter-harmonic noise reducer.

Counters of consecutive frames of sound signal category 0, and counters of consecutive frames of sound signal category 3 or 4, are used to respectively decrease or increase the thresholds.

For example, if a counter counts a series of more than 30 frames of sound signal category 3 or 4, all the floating thresholds (1 to 4) are increased by a predefined value for the purpose of allowing more frames to be considered as sound signal category 4.

The inverse is also true with sound signal category 0. For example, if a series of more than 30 frames of sound signal category 0 is counted, all the floating thresholds (1 to 4) are decreased for the purpose of allowing more frames to be considered as sound signal category 0. All the floating thresholds 1-4 are limited to absolute maximum and minimum values to ensure that the signal classifier is not locked to a fixed category.

In the case of frame erasure, all the thresholds 1-4 are reset to their minimum values and the output of the signal classifier is considered as non-tonal (sound signal category 0) for three (3) consecutive frames (including the lost frame).

If information from a Voice Activity Detector (VAD) is available and it is indicating no voice activity (presence of silence), or if the last frame didn't contain generic audio the decision of the signal type classifier is forced to sound signal category 0 ($e_{CAT} = 0$).

6.1.1.3.4.5 Inter-tone noise reduction in the excitation domain

Inter-tone noise reduction is performed on the frequency representation of the concatenated excitation as a first operation of the enhancement. The reduction of the inter-tone quantization noise is performed by scaling the spectrum in each critical band with a scaling gain g_s limited between a minimum and a maximum gain g_{min} and g_{max}. The scaling gain is derived from an estimated signal-to-noise ratio (SNR) in that critical band. The processing is performed on frequency bin basis and not on critical band basis. Thus, the scaling gain is applied on all frequency bins, and it is derived from the SNR computed using the bin energy divided by an estimation of the noise energy of the critical band including that bin. This feature allows for preserving the energy at frequencies near harmonics or tones, thus substantially preventing distortion, while strongly reducing the noise between the harmonics. The inter-tone noise reduction is performed in a per bin manner over all 640 bins.

The minimum scaling gain $g_{\min}$ is derived from the maximum allowed inter-tone noise reduction in dB, $R_{\max}$. As described above, the second stage of classification makes the maximum allowed reduction varying between 6 and 12 dB. Thus minimum scaling gain is given by

$$g_{\min} = 10^{-R_{\max}/20} \tag{1496}$$

The scaling gain is computed related to the SNR per bin. Then per bin noise reduction is performed on the entire spectrum to the maximum frequency of 6400 Hz. The noise reduction can start at the 2$^{\text{th}}$ critical band (i.e. no reduction is performed below 300Hz). The excitation type classifier module can push the starting critical band up to the 4$^{\text{th}}$ band (510 Hz), to reduce any potential degradation. This means that the first critical band on which the noise reduction is performed is between 300Hz and 920 Hz, and it can vary on a frame basis. In a more conservative implementation, the minimum band where the noise reduction starts can be set higher.

The scaling for a certain frequency bin k is computed as a function of SNR, given by

$$g_s(k) = \sqrt{k_s \, \text{SNR}(k) + c_s} \text{ , bounded by } g_{\min} \leq g_s \leq g_{\max} \tag{1497}$$

Usually $g_{\max}$ is equal to 1 (i.e. no amplification is allowed), then the values of k_s and c_s are determined such as $g_s = g_{\min}$ for $SNR = 1$ dB, and $g_s = 1$ for $SNR = 45$ dB. That is, for SNRs of 1 dB and lower, the scaling is limited to $g_{\min}$ and for SNRs of 45 dB and higher, no noise reduction is performed ($g_s = 1$). Thus, given these two end points, the values of k_s and c_s in equation are given by

$$k_s = (1 - g_{\min}^2)/44 \text{ and } c_s = (45 g_{\min}^2 - 1)/44 \tag{1498}$$

If $g_{\max}$ is set to a value higher than 1, then it allows the process to slightly amplify the tones having the highest energy. This can be used to compensate for the fact that the CELP codec, used in the prototype, doesn't match perfectly the energy in the frequency domain. This is generally the case for signals different from voiced speech.

The SNR per bin in a certain critical band i is computed as

$$\text{NRF}_{BIN}(h) = \frac{0.3 E_{BIN}^{(1)}(h) + 0.7 E_{BIN}^{(2)}(h)}{N_B(i)}, \qquad h = j_i, ..., j_i + M_B(i) - 1 \tag{1499}$$

where $E_{BIN}^{(1)}(h)$ and $E_{BIN}^{(2)}(h)$ denote the energy per frequency bin for the past and the current frame spectral analysis, respectively, as computed in subclause 5.1.5.2, $N_B(i)$ denotes the noise energy estimate of the critical band i, j_i is the index of the first bin in the ith critical band, and $M_B(i)$ is the number of bins in the critical band i as defined above.

The smoothing factor is adaptive and it is made inversely related to the gain itself. The smoothing factor is given by $\alpha_{gs} = 1 - g_s$. That is, the smoothing is stronger for smaller gains g_s. This approach substantially prevents distortion in high SNR segments preceded by low SNR frames, as it is the case for voiced onsets. The smoothing procedure is able to quickly adapt and to use lower scaling gains on onsets.

In case of per bin processing in a critical band with index i, after determining the scaling gain and using SNR the actual scaling is performed using a smoothed scaling gain $g_{BIN,LP}$, updated in every frequency analysis as follows

$$g_{BIN,LP}(k) = \alpha_{gs} g_{BIN,LP}(k) + (1 - \alpha_{gs}) g_s \tag{1500}$$

Temporal smoothing of the gains substantially prevents audible energy oscillations while controlling the smoothing using α_{gs} substantially prevents distortion in high SNR segments preceded by low SNR frames, as it is the case for voiced onsets or attacks.

The scaling in the critical band i is performed as

$$f_u'(h + j_i) = g_{BIN,LP}(h + j_i) f_u(h + j_i), \qquad h = 0, ..., M_B(i) - 1 \tag{1501}$$

where j_i is the index of the first bin in the critical band i and $M_B(i)$ is the number of bins in that critical band.

The smoothed scaling gains $g_{BIN,LP}(k)$ are initially set to 1. Each time a non-tonal sound frame is processed $e_{CAT}=0$, the smoothed gain values are reset to 1 to reduce any possible reduction in the next frame.

Note that in every spectral analysis, the smoothed scaling gains $g_{BIN,LP}(k)$ are updated for all frequency bins in the entire spectrum. Note that in case of low-energy signal, inter-tone noise reduction is limited to -1.25 dB. This happens when the maximum noise energy in all critical bands, $\max(N_B(i))$, $i=0,...,20,$ is less or equal to 10.

6.1.1.3.4.6 Inter-tone quantization noise estimation

The inter-tone quantization noise energy per critical frequency band is estimated as being the average energy of that critical frequency band excluding the maximum bin energy of the same band. The following formula summarizes the estimation of the quantization noise energy for a specific band i :

$$N_B(i)=\frac{1}{q(i)}\left(\frac{\left(E_B(i)M_B(i)-\max_h\left(E_{BIN}(h+j_i)\right)\right)}{(M_B(i)-1)}\right), \qquad h=0,...,M_B(i)-1 \qquad (1502)$$

where j_i is the index of the first bin in the critical band i, $M_B(i)$ is the number of bins in that critical band, $E_B(i)$ is the average energy of a band i, $E_{BIN}(h+j_i)$ is the energy of a particular bin and $N_B(i) \; N_B(i)$ is the resulting estimated noise energy of a particular band i. The variable $q(i)$ represents a noise scaling factor per band that is found experimentally and is set such that more noise can be removed in low frequencies and less noise in high frequencies as it is shown below:

$$q = \{10,10,10,10,10,10,11,11,11,11,11,11,11,11,11,15,15,15,15,15\} \qquad (1503)$$

6.1.1.3.4.7 Increasing spectral dynamic of the excitation

The second operation of the frequency post processing provides an ability to retrieve frequency information that is lost within the coding noise. The CELP codecs, especially when used at low bitrates, are not very efficient to properly code frequency content above 3.5-4 kHz. The following steps take advantage of the fact that music spectrum does not often changed substantially from frame to frame. Therefore a long term averaging can be done and some of the coding noise can be eliminated. The following operations are performed to define a frequency-dependent gain function. This function is then used to further enhance the excitation before converting it back to the time domain.

6.1.1.3.4.8 Per bin normalization of the spectrum energy

The first operation consists in creating a weighting mask based on the normalized energy of the spectrum of the concatenated excitation. The normalization is done such that the tones have a value above 1.0 and the valleys a value under 1.0. To do so, the energy spectrum $E_{BIN}(k)$ is normalized between 0.925 and 1.925 to get the normalized energy spectrum $E_n(k)$ using the following equation:

$$E_n(k)=\frac{E_{BIN}(k)}{\max(E_{BIN})}+0.925, \qquad k=0,...,639 \qquad (1504)$$

where $E_{BIN}(k)$ represents the bin energy as calculated in subclause 5.1.5.2. Since the normalization is performed in the energy domain, many bins have very low values. The offset 0.925 has been chosen such that only a small part of the normalized energy bins would have a value below 1.0. Once the normalization is done, the resulting normalized energy spectrum is passed through a power function of 8 to obtain a scaled energy spectrum as shown in the following formula:

$$E_p(k)=E_n(k)^8 \qquad k=0,...,639 \qquad (1505)$$

where $E_n(k)$ is the normalized energy spectrum and $E_p(k)$ is the scaled energy spectrum. A maximum limit of the scaled energy spectrum is fixed to 5, creating a ratio of approximately 10 between the maximum and minimum normalized energy values. The following equation shows how the function is applied:

$$E_{pl}(k) = \min(5, E_p(k)) \qquad k = 0,...,639 \qquad (1506)$$

Where $E_{pl}(k)$ represents limited scaled energy spectrum and $E_p(k)$ is the scaled energy spectrum.

6.1.1.3.4.9 Smoothing of the scaled energy spectrum along the frequency axis and the time axis

With the last two operations, the position of the most energetic pulses begins to take shape. Applying power of 8 on the bins of the normalized energy spectrum is a first operation to create the mask that increases the spectral dynamics. The next 2 operations enhance this spectrum mask. First the scaled energy spectrum is smoothed along the frequency axis from low frequency to the high frequency with an averaging filter. Then, the resulting mask is processed along the time domain axis to smooth the bin values from frame to frame.

The smoothing of the scaled energy spectrum along the frequency axis can be described with following function:

$$\overline{E}_{pl}(k) = \begin{cases} \dfrac{E_{pl}(k) + E_{pl}(k+1)}{2}, & k = 0 \\[2mm] \dfrac{E_{pl}(k-1) + E_{pl}(k) + E_{pl}(k+1)}{3}, & k = 1,...,638 \\[2mm] \dfrac{E_{pl}(k-1) + E_{pl}(k)}{2}, & k = 639 \end{cases} \qquad (1507)$$

Finally, the smoothing along time axis results in a time-averaged amplification/attenuation weighting mask G_m to be applied to the spectrum f_u'. The weighting mask, also called gain mask, is described with the following equation:

$$G_m^t(k) = \begin{cases} 0.95 \cdot G_m^{(t-1)}(k) + 0.05 \overline{E}_{pl}(k), & k = 0,...,319 \\[2mm] 0.85 \cdot G_m^{(t-1)}(k) + 0.15 \overline{E}_{pl}(k), & k = 320,...,639 \end{cases} \qquad (1508)$$

where $\overline{E}_{pl}$ is the scaled energy spectrum smoothed along the frequency axis, t is the frame index, and G_m is the time-averaged weighting mask.

A slower adaptation rate has been chosen for the lower frequencies to substantially prevent gain oscillation. A faster adaptation rate is allowed for higher frequencies since the positions of the tones are more likely to change rapidly in the higher part of the spectrum. With the averaging performed on the frequency axis and the long term smoothing performed along the time axis, the final vector $G_m^t(k)$ is used as a weighting mask to be applied directly on the enhanced spectrum f_u' of the concatenated excitation.

6.1.1.3.4.10 Application of the weighting mask to the enhanced concatenated excitation spectrum

The weighting mask defined above is applied differently depending on the output of the excitation type classifier (value of e_{CAT}). The weighting mask is not applied if the excitation is classified as category 0 ($e_{CAT} = 0$; i.e. high probability of speech content).

For the first 1 kHz, the mask is applied if the excitation is not classified as category 0 ($e_{CAT} \neq 0$). Attenuation is possible but no amplification is performed in this frequency range (maximum value of the mask is limited to 1).

If more than 25 consecutive frames are classified as category 4 ($e_{CAT} = 0$; i.e. high probability of music content), but not more than 40 frames, then the weighting mask is applied without amplification for all the remaining bins (the maximum gain $G_{\max 0}$ is limited to 1, and there is no limitation on the minimum gain).

When more than 40 frames are classified as category 4, for the frequencies between 1 and 2 kHz the maximum gain $G_{\max 1}$ is set to 1.5 for bitrates below 12650 bits per second (b/s). Otherwise the maximum gain $G_{\max 1}$ is set to 1. In this frequency band, the prototype fixes the minimum gain $G_{\min 1}$ to 0.75 only if the bitrate is higher than 15850 b/s, otherwise there is no limitation on the minimum gain.

For the band 2 to 4 kHz, the maximum gain $G_{\max 2}$ is limited to 2 for bitrates below 12650 b/s, and it is limited to 1.25 for the bitrates equal to or higher than 12650 b/s and lower than 15850 b/s. Otherwise, then maximum gain $G_{\max 2}$ is limited to 1. Still in this frequency band, the minimum gain $G_{\min 2}$ is 0.5 only if the bitrate is higher than 15850 b/s, otherwise there is no limitation on the minimum gain.

For the band 4 to 6.4 kHz, the maximum gain $G_{\max 3}$ is limited to 2 for bitrates below 15850 b/s and to 1.25 otherwise. In this frequency band, the prototype fixes the minimum gain $G_{\min 3}$ to 0.5 only if the bitrate is higher than 15850 b/s, otherwise there is no limitation on the minimum gain.

6.1.1.3.4.11   Inverse frequency transform and overwriting of the current excitation

After the frequency domain enhancement is completed, an inverse frequency-to-time transform is performed in order to get the enhanced temporal excitation back. The frequency-to-time conversion is achieved with the same type II DCT as used for the time-to-frequency conversion. The modified time-domain excitation $u'_{td}(n)$ is obtained as

$$
u'_{td}(n) = \begin{cases} \sqrt{\dfrac{1}{L_c}} \cdot \displaystyle\sum_{k=0}^{L_c-1} f''_u(k), & n = 0 \\ \sqrt{\dfrac{2}{L_c}} \cdot \displaystyle\sum_{k=0}^{L_c-1} f''_u(k) \cdot \cos\left(\dfrac{\pi}{L_c}\left(k+\dfrac{1}{2}\right)n\right), & 1 \le n \le L_c - 1 \end{cases} \tag{1509}
$$

where f''_u is the frequency representation of the modified excitation, $u'_{td}(n)$ is the enhanced concatenated excitation, and L_c is the length of the concatenated excitation vector.

To avoid adding delay to the synthesis, it has been decided to avoid overlap-and-add algorithm in the LP domain path. Thus, the exact length of the final excitation u_f is used to generate the synthesis directly from the enhanced concatenated excitation, without overlap as shown in the equation below:

$$
u_f(n) = u'_{td}(n + L_w), \qquad n = 0,\dots,255 \tag{1510}
$$

Here L_w represents the length of the section of the window that was applied on the past segment of the excitation, prior to the frequency transformation.

6.1.2   Source Controlled VBR decoding

6.1.3   Synthesis

The LP synthesis is performed by filtering the post-processed excitation signal $u(n)$ through the LP synthesis filter.. The decoded and interpolated LP coefficients, $\hat{a}_i$, are used to construct the synthesis filter, $1/\hat{A}(z)$.

The reconstructed speech for the subframe of size 64 is given by

$$
\hat{s}(n) = u(n) - \sum_{i=1}^{16} \hat{a}_i \hat{s}(n-i) \tag{1511}
$$

The synthesized signal is then de-emphasized by filtering through the filter $1/(1 - 0.68z^{-1})$ (inverse of the pre-emphasis filter applied at the encoder input).

The de-emphasis synthesis speech $\hat{s}(n)$ is then passed through an adaptive post-processing which is described in the following section.

6.1.4 Post-processing

The decoded signal is conveyed to several post-processing blocks. First an adaptive post-filtering is applied for enhancing the formant and harmonic structure of the signal. In a second step, a bass-post-filter treats the low frequencies.

6.1.4.1 Adaptive post-filtering

The post-filtering is similar to ITU-T G.729 post-processing with the main difference that it is performed at 12.8 or 16 kHz. The adaptive post-filter is a cascade of three filters: a long-term post-filter, $H_p(z)$, a short-term post-filter, $H_f(z)$, and a tilt compensation filter, $H_t(z)$, followed by an adaptive gain control procedure. The post-filter coefficients are updated in every subframe. The post-filtering process is organized as follows. First, the reconstructed signal, $\hat{s}_{pre}(n)$, is inverse-filtered through $\hat{A}(z/\gamma_n)$ to produce the residual signal, $\hat{r}(n)$. This signal is used to compute the delay, T, and gain, g_l, of the long-term post-filter $H_p(z)$. The signal, $\hat{r}(n)$, is then filtered through the long-term post-filter, $H_p(z)$, and the synthesis filter, $1/[g_f \hat{A}(z/\gamma_d)]$. Finally, the output signal of the synthesis filter, $1/[g_f \hat{A}(z/\gamma_d)]$ is passed through the tilt compensation filter, $H_p(z)$, to generate the post-filtered reconstructed signal, $\hat{s}_f(n)$. Adaptive gain control is then applied to $\hat{s}_f(n)$ to match its energy to the energy of $\hat{s}_{pre}(n)$. The post-filter parameters γ_n and γ_d are described in detail in subclauses 6.1.4.1.3. and 6.1.4.1.4.

The long-term post-filter is only applied for NB modes and is bypassed for WB and SWB. In WB and SWB cases, the post-filtering consists of a cascade of only two filters: a short-term post-filter, $H_f(z)$ (see subclause 6.1.4.3), and a tilt compensation filter, $H_t(z)$ (see subclause 6.1.4.4), followed by an adaptive gain control procedure (see subclause 6.1.4.5).

At 9.6 kbit/s NB decoding, the long-term post-filter, $H_p(z)$ is active only for clean speech when the level of background noise is less than 20 dB. It is also desactivacted for UC mode.

6.1.4.1.1 Long-term post-filter

The long-term post-filter is given by:

$$H_p(z) = \frac{1}{1+\gamma_p g_l}\left(1+\gamma_p g_l z^{-T}\right) \tag{1512}$$

where T is the pitch delay, and g_l is the gain coefficient. Note that g_l is bounded by 1, and is set to zero if the long-term prediction gain is less than 3 dB. The factor γ_p controls the amount of long-term post-filtering and has the value of $\gamma_p = 0.5$. The long-term delay and the gain are computed from the residual signal, $\hat{r}(n)$, obtained by filtering $\hat{s}(n)$ through $\hat{A}(z/\gamma_n)$, which is the numerator of the short-term post-filter (see subclause 6.1.4.2). That is

$$\hat{r}(n) = \hat{s}(n) + \sum_{i=1}^{16} \gamma_n^i \hat{a}_i \hat{s}(n-i) \tag{1513}$$

The long-term delay is computed using a two-pass procedure. The first pass selects the best integer pitch delay, T_0, in the range $\left\lfloor \left\lfloor d_{fr}^{[0]} \right\rfloor - 1; \left\lfloor d_{fr}^{[0]} \right\rfloor + 1; \right\rfloor$, where $\left\lfloor d_{fr}^{[0]} \right\rfloor$ is the integer part of the (transmitted) fractional pitch lag in the first subframe. The best integer, T_0, is the one that maximizes the correlation

$$R(k) = \sum_{n=0}^{63} \hat{r}(n)\hat{r}(n-k) \tag{1514}$$

The second pass chooses the best fractional pitch delay, T, with resolution 1/8 around T_0. This is done by finding the delay with the highest pseudo-normalized correlation

$$R'(k) = \frac{\sum_{n=0}^{63} \hat{r}(n)\hat{r}_k(n)}{\sqrt{\sum_{n=0}^{63} \hat{r}_k(n)\hat{r}_k(n)}} \qquad (1515)$$

where $\hat{r}_k(n)$ is the residual signal at a fractional delay, k. The fractional delayed signal, $\hat{r}_k(n)$, is first computed using an interpolation filter of length 33. Once the optimal fractional delay, T, is found, $\hat{r}_k(n)$ is recomputed with a longer interpolation filter of length 129. The new signal replaces the previous one only if the longer filter increases the value of $R'(T)$. Then, the corresponding correlation, $R'(T)$, is normalized with the square-root of the energy of $\hat{r}(n)$. The squared value of this normalized correlation is then used to determine if the long-term post-filter should be disabled. That is, if

$$\frac{R'(T)^2}{\sum_{n=0}^{63} \hat{r}(n)\hat{r}(n)} < 0.5 \qquad (1516)$$

the long-term post-filter is disabled by setting $g_l = 0$. Otherwise, the value of g_l is computed as

$$g_l = \frac{\sum_{n=0}^{63} \hat{r}(n)\hat{r}_k(n)}{\sum_{n=0}^{63} \hat{r}_k(n)\hat{r}_k(n)}, \quad \text{constrained by } 0 \le g_l \le 1.0 \qquad (1517)$$

6.1.4.1.2 Short-term post-filter

The short-term post-filter is given by

$$H_f(z) = \frac{1}{g_f} \frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)} = \frac{1}{g_f} \frac{1 + \sum_{i=1}^{16} \gamma_n^i \hat{a}_i z^{-i}}{1 + \sum_{i=1}^{16} \gamma_d^i \hat{a}_i z^{-i}} \qquad (1518)$$

where $\hat{A}(z)$ is the quantized LP analysis filter (LP analysis is not done at the decoder) and the factors γ_n and γ_d control the amount of short-term post-filtering. The gain, g_f, is calculated on the truncated impulse response, $h_f(n)$, of the filter $\hat{A}(z/\gamma_n)/\hat{A}(z/\gamma_d)$ and is given by

$$g_f = \sum_{n=0}^{19} |h_f(n)| \qquad (1519)$$

Note that the gain, g_f, will be modified according to the noise level as explained in the next clause.

6.1.4.1.3 Post-filter NB parameters

In the ITU-T G.729 codec, the post-filter parameters γ_n, γ_d and g_f have fixed values. If a variable, called the long-term normalized noise gain, $\overline{g}_{norm}$, is less than 25.0 and an active signal is detected, γ_n has a value limited in the range [0.55, 0.70] and γ_d has a value limited in the range [0.65, 0.75] as expressed by

$$\gamma_n = -0.05\overline{g}_{norm} + 1.45 \qquad (1520)$$

$$\gamma_d = -0.01\overline{g}_{norm} + 0.9 \tag{1521}$$

Otherwise (not an active signal or $\overline{g}_{norm} \geq 25.0$), $\gamma_n = 0.1$ and $\gamma_d = 0.15$.

In the case of the GSC mode the post-filter parameters γ_n, γ_d and g_f are set to 1.

The long-term normalized noise gain, $\overline{g}_{norm}$, is updated only when in UC mode and when no signal activity is detected ($f_{SAD} = 0$). The update is performed as

$$\overline{g}_{norm} = 0.95\overline{g}_{norm} + 0.05 g_{norm} \tag{1522}$$

where g_{norm} is the normalized gain of random excitation in the UC mode, calculated as

$$g_{norm} = 10^{0.05E_g}\sqrt{\frac{1}{64}\sum_{n=0}^{63} c^2(n)} \tag{1523}$$

In the equation above, E_g is the quantized gain of the random excitation, $c(n)$, used in TC mode, which has been quantized with 7 bits in the logarithmic energy domain. The modified value of g_f in equation (1523) is not filtered. The modified value of g_f is computed as

$$g'_f = g_f + \mu(1.0 - g_f) \tag{1524}$$

where the factor μ is derived from $\overline{g}_{norm}$ as follows

$$\mu = (\overline{g}_{norm} - 15.0)\frac{0.1}{4}, \quad \text{constrained by } 0 \leq \mu \leq 0.25 \tag{1525}$$

Thus, the short-term post-filter, described in subclause 6.1.4.1.2, is used with the modified value of gain, g'_f, and not g_f. These modifications help to diminish the effect of post-filtering in noisy conditions.

6.1.4.1.4 Post-filter WB and SWB parameters

The post-filter parameters γ_n, γ_d for WB and SWB have fixed values, which depend on decoding mode. The filter may operate at both internal sampling frequencies 12.8 kHz and 16 kHz. In case of 12.8 kHz internal frequency the parameters take the default value $\gamma_n = 0.7$, $\gamma_d = 0.75$

Table 157 Post filter WB and SWB parameters for 12.8 kHz

Mode	Inactive & AMRWB IO clean speech	< 13.2 kbit/s clean speech	< 24.4 kbit/s clean speech	≤ 32 kbit/s clean speech	< 15.85 kbit/s noisy speech	≤ 32 kbit/s noisy speech
γ_n	0.7	0.80	0.75	0.72	0.75	0.7

In case of 16 kHz internal frequency, noisy speech (the level of background noise is less than 20) and for any mode not depicted in the table below the parameters take the default value $\gamma_d = 0.76$, $\gamma_n = 0.76$.

Table 158 Post filter WB and SWB parameters for 16 kHz

Mode	13.2 kbit/s	16.4 kbit/s	24.4 kbit/s	32 kbit/s
γ_n	0.82	0.80	0.78	0.78

6.1.4.1.5 Tilt compensation

The filter $H_t(z)$ compensates for the tilt in the short-term post-filter $H_f(z)$ and is given by

$$H_t(z) = \frac{1}{g_t}\left(1 + \gamma_t k_1 z^{-1}\right) \qquad (1526)$$

where $\gamma_t k_1$ is a tilt factor with k_t being the first reflection coefficient, calculated from $h_f(n)$ as

$$k_1 = -\frac{r_h(1)}{r_h(0)} \qquad (1527)$$

where

$$r_h(i) = \sum_{j=0}^{19-i} h_f(j) h_f(j+i) \qquad (1528)$$

The gain term $g_t = 1 - |\gamma_t k_1|$ compensates for the decreasing effect of g_f in $H_f(z)$. Furthermore, it has been shown that the product $H_f(z) H_t(z)$ has generally no gain. Two values are used for γ_t depending on the sign of k_1. If k_1 is negative, $\gamma_t = 0.9$, and if k_1 is positive, $\gamma_t = 0.2$.

6.1.4.1.6 Adaptive gain control

Adaptive gain control is used to compensate for gain differences between the synthesized signal, $\hat{s}_{pre}(n)$, and the post-filtered signal, $\hat{s}_f(n)$. A gain factor, f_g, for the current subframe is computed by

$$f_g = \frac{\sum\limits_{n=0}^{63} |\hat{s}_{pre}(n)|}{\sum\limits_{n=0}^{63} |\hat{s}_f(n)|} \qquad (1529)$$

Then, the post-filtered signal, $\hat{s}_f(n)$, is scaled as

$$\hat{s}_f(n) = g_{cont}(n)\hat{s}_f(n), \quad \text{for } n = 0,\ldots,63 \qquad (1530)$$

where $g_{cont}(n)$ is a continuous gain, updated on a sample-by-sample basis for NB input as

for NB input

$$g_{cont}(n) = 0.9875 g_{cont}(n-1) + 0.0125 f_g, \quad \text{for } n = 0,\ldots,63 \qquad (1531)$$

for SWB TBE input

$$g_{cont}(n) = 0.85 g_{cont}(n-1) + 0.15 f_g, \quad \text{for } n = 0,\ldots,63 \qquad (1532)$$

The initial value of $g_{cont}(-1) = 1.0$ is used. Then, for each new subframe, $g_{cont}(-1)$ is set equal to $g_{cont}(63)$ of the previous subframe.

For NB signals, the post-filtered synthesized signal, $\hat{s}_f(n)$, is used instead of $\hat{s}_{pre}(n)$ for signal de-emphasis, as described in subclause 6.3.

6.1.4.2 Bass post-filter

This clause describes the functionality of the bass post-filter, a low-frequency pitch enhancement procedure, which is closely related to the corresponding procedures in [11].

The main difference compared to the previous standards is that the last step of post filtering is performed in the frequency domain. The reason for this is a different method of resampling from the internal sampling frequency to the

output sampling frequency. Instead of time domain resampling (see clause 7.6 in [25]) complex low delay filter bank synthesis is used (see subclause 6.9).

The filter is applied to all LP-based modes up to 32 kbit/s except for NB noisy speech (the level of background noise > 20).

The bass post-filter uses two-band decomposition and adaptive filtering is applied only to the lower band. This results in a total post-processing that is mostly targeted at frequencies near the first harmonics of the synthesized signal.
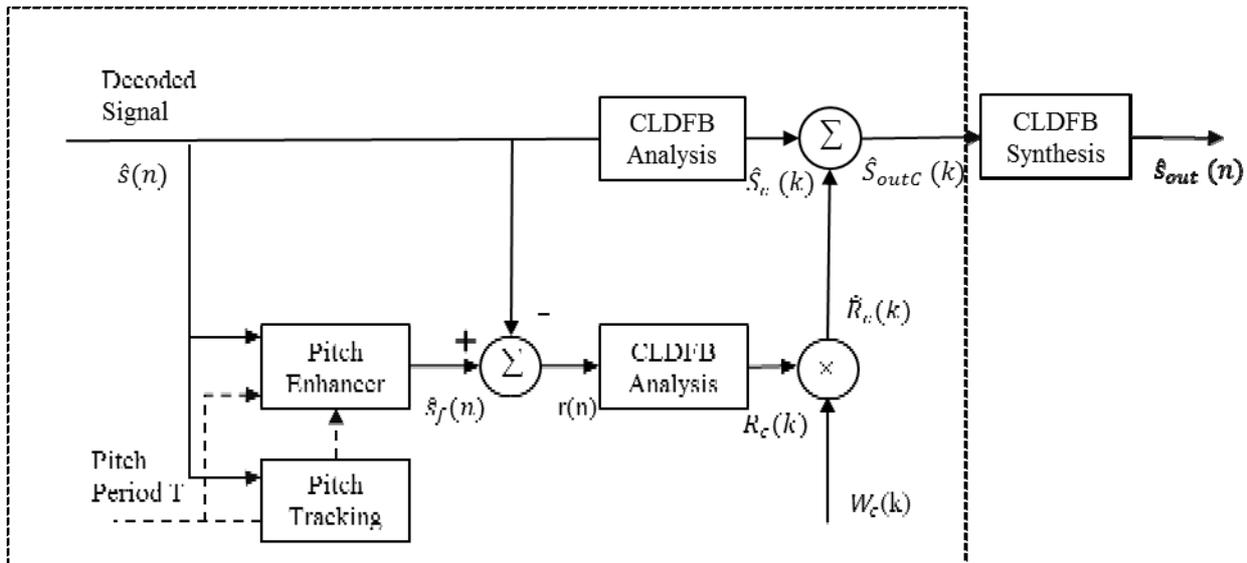
Figure 91: Block diagram of bass post-filter

Figure 91 shows the block diagram of the low-band pitch enhancer. Note that this is a simplified block diagram, which is equivalent to adding the low-pass filtered enhanced signal to the high-pass filtered signal (see subclause 6.1.3 in [11]). The decoded signal, $\hat{s}(n)$, is first processed through an adaptive pitch enhancer module leading to an enhanced (full-band) signal, $\hat{s}_f(n)$. By subtracting the decoded signal, an enhancement signal, $r(n)$, is obtained. Then CLDFB analysis (see subclause 5.1.2) is applied to transform signal into frequency domain $R_C(k)$. This signal is subsequently filtered in the frequency domain through a low-pass filter to obtain the signal $\hat{R}_C(k)$ which is the low-band part of this enhancement signal. Frequency domain filtering means multiplying $R_C(k)$ with a low-pass filter's frequency response $W_C(k)$. The enhanced signal after post-processing, $\hat{S}_{outC}(k)$, is then obtained by adding the low-band enhancement signal to the transformed into frequency domain decoded signal $\hat{S}_C(k)$. Resampling to the output sampling frequency and converting into time domain signal, $\hat{s}_{out}(n)$, which is performed by CLDFB synthesis, is not a part of the bass post-filter and is applied for all modes (see subclause 6.9).

The object of the pitch enhancer module is to reduce the inter-harmonic noise in the decoded signal, which is achieved here by a time-varying linear filter described by the following equation:

$$\hat{s}_f(n) = (1-\alpha)\hat{s}(n) + \alpha s_p(n) \tag{1533}$$

where $\hat{s}_f(n)$ is the output signal of the pitch enhancer, α is a coefficient that controls the inter-harmonic attenuation. The signal $s_p(n)$ is a coefficient that controls the inter-harmonic attenuation. The signal $s_p(n)$ is the two-sided long-term prediction signal that is computed in each subframe as

$$s_p(n) = 0.5\hat{s}(n-T) + 0.5\hat{s}(n+T) \tag{1534}$$

where T is the pitch period of the decoded signal $\hat{s}(n)$. Parameters T and α vary in time. With a value of $\alpha = 1$, the gain of the filter described by equation (1533) is exactly 0 at frequencies $1/(2T)$, $3/(2T)$, $5/(2T)$, etc.; i.e., at the mid-

point between the harmonic frequencies $1/T$, $3/T$, $5/T$, etc. When α approaches 0, the attenuation between the harmonics produced by the filter of equation (1533) decreases.

The pitch parameter   is the received closed-loop pitch lag of the respective subframe. However, this parameter is only accurate for the part of the two-sided long-term prediction of Equation (1534) predicting from the past pitch cycle. The prediction from the future pitch cycle may be less accurate, especially if the pitch lag value is not constant.

Thus, in order to improve the prediction accuracy, in case of voiced onset frames it is preferable to make use of the pitch lag value of the subframe containing the future pitch cycle, i.e., of that subframe whose closed-loop pitch lag points into the present subframe. This requires the availability of pitch lag parameters of a frame following the current frame.

The pitch lag parameter, , is further enhanced by means of a pitch tracker which makes the pitch contour smoother and avoids pitch doublings.

The factor α is computed as follows. First, the correlation between the signal and the predicted signal is given by

$$C_p = \sum_{n=0}^{N-1} \hat{s}(n) s_p(n) \tag{1535}$$

and the energy of the predicted signal is given by

$$E_p = \sum_{n=0}^{N-1} s_p(n) s_p(n) \tag{1536}$$

The factor α is given by

$$\alpha = \frac{C_p}{0.5(E_p + 10^{0.1\overline{E}_{pp}})}, \text{ constrained by } 0 \le \alpha \le 1, \tag{1537}$$

where $\overline{E}_{pp}$ is the mean prediction error energy in dB in the present subframe. The mean prediction error energy, $\overline{E}_{pp}$ is updated for the next subframe as follows. The long-term prediction error is first computed by

$$e_p(n) = \hat{s}(n) - \frac{C_p}{E_p} s_p(n) \tag{1538}$$

and then pre-emphasized using the relation

$$e_{pp}(n) = e_p(n) - 0.95 e_p(n-1) \tag{1539}$$

The energy of the pre-emphasized error signal is then computed in dB as

$$E_{pp} = 10 \log \left[\sum_{n=0}^{N-1} e_{pp}(n) e_{pp}(n) \right] \tag{1540}$$

and the mean error energy is then updated in every subframe by

$$\overline{E}_{pp} = 0.99 \overline{E}_{pp} + 0.01 E_{pp} \tag{1541}$$

with initial value $\overline{E}_{pp} = 40$.

The factor α is further adapted to a measure of signal stationarity, which limits the level of inter-harmonic attenuation when the signal is not in a steady-state mode. The adaptation is based on the stability factor of the current frame, $\theta^{[0]}$ and a recursively smoothed version of stability factor $\overline{\theta}^{[0]}$ defined as

$$\overline{\theta}^{[0]} = 0.8 \cdot \theta^{[0]} + 0.2 \cdot \overline{\theta}^{[-1]} \tag{1542}$$

The factor α, defined in equation (1537), is finally scaled as

$$\alpha = (1 + 0.15\sqrt{\theta^{[0]}} - 2.0\left|\overline{\theta}^{[0]} - \theta^{[0]}\right|) \cdot \alpha \qquad (1543)$$

Since larger portions of noise are aurally masked when the signal rapidly changes, the above adaptation gives a better balance between attenuation of quantization noise and signal degradation.

At 16.4 and 24.4 kbps, the factor α is adjusted by decoding the gain adjustment $\hat{\alpha}_\gamma$, which is quantized at the encoder (see subclause 5.2.4) and transmitted in the bitstream on 2 bits.

$$\alpha = \hat{\alpha}_\gamma \cdot \alpha \qquad (1544)$$

6.1.5 Decoding of upper band for LP-based Coding Modes

6.1.5.1 Decoding Time-domain Bandwidth Extension

The time domain bandwidth extension decoder synthesizes the high band excitation signal from the excitation signal generated by the low band ACELP decoder and a set of high band model parameters received from the time domain bandwidth extension encoder. The synthesized high band signal is then combined with the output from the lowband ACELP decoder to generate a superwideband output. The high level schematic of the time domain bandwidth extension decoder is shown in figure 92.
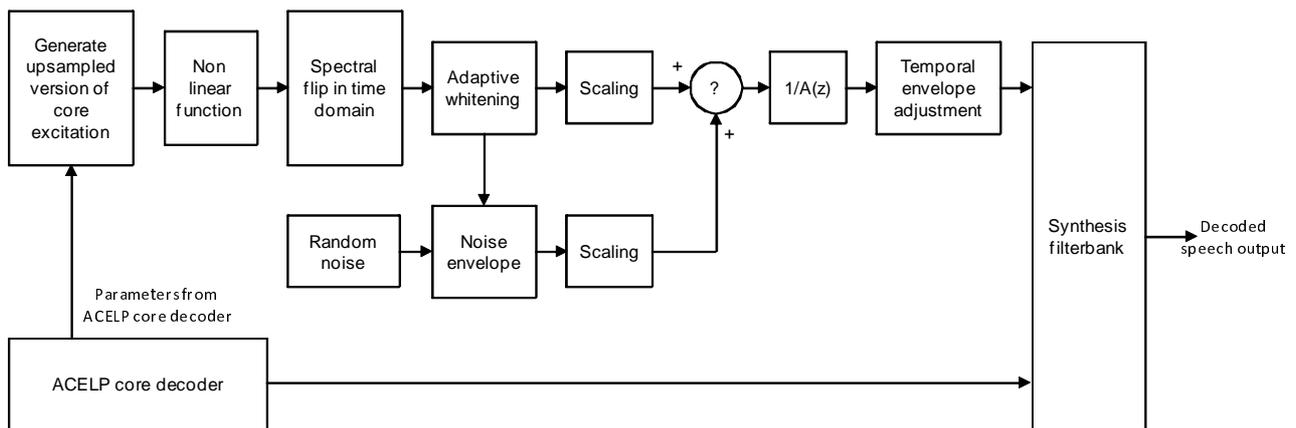
Figure 92: Time domain bandwidth extension decoder

6.1.5.1.1 Generation of the upsampled version of the lowband excitation

An upsampled version of the low band excitation signal is derived from the ACELP core as show in figure 92. For each ACELP core coding subframe, i, a random noise scaled by a factor voice factor, Vf_i is first added to the fixed codebook excitation that is generated by the ACELP core encoder. The voice factor is determined using the subframe maximum normalized correlation parameter, β_i, that is derived during the ACELP encoding. First the β_i factors are combined to generate Vf_i.

$$Vf_i = 0.34 + 0.5 \times \beta_i + (0.5 - 0.34) \times \beta_i^2 \qquad (1545)$$

Vf_i calculated above is limited to a maximum of 1 and a minimum of 0. When the ACELP coder type is voiced the V_f is modified based on the integer pitch value T_0 is modified as in the pseudo-code below:

```
if((coder_type == VOICED))
   if(T₀ <= 57.75f)
      Vfᵢ = -0.0126f*T₀ + 1.23f;
   else if(T₀ > 57.75f && T₀ < 115.5f)
      Vfᵢ = 0.0087f*T₀;
   end
```

```
end
```

Regardless of the ACELP coder type, if the open loop pitch lag T_0 exceeded 115, V_f is set to 1;

$\varepsilon_1(n) = code(n) + Vf_i \times random(n)\, for\, 0 \le n \le 256$ if the ACELP core encodes a maximum of 6.4 KHz or $0 \le n \le 320$ if the ACELP core encodes a maximum bandwidth of 8 KHz.

The ACELP fixed code book excitation signal mixed with noise is then resampled by a factor β. The resampling factor β is set to 5/2 when the ACELP core encodes a maximum bandwidth of 6.4 KHz and it is set to 2 when the ACELP core encodes a maximum bandwidth of 8 KHz.

The resampled output is scaled by the ACELP fixed codebook gain and added to a delayed version of itself.

$$\varepsilon(n) = g_c \times \varepsilon_2(n) + g_p \times \varepsilon_2(n - \beta P) \tag{1546}$$

where g_c is the subframe ACELP fixed codebook gain, g_p is the subframe ACELP adaptive codebook gain and P is the open loop pitch lag.

6.1.5.1.2 Non-Linear Excitation Generation

The excitation signal $\varepsilon(n)$ is processed through a non-linear function in order to extend the pitch harmonics in the low band signal into the high band. The non-linear processing is applied to a frame of $\varepsilon(n)$ in two stages; the first stage works on the first half subframe (160 samples) of $\varepsilon(n)$ and the second stage works on the second half subframe. The non-linear processing steps for the two stages are described below. In the first stage $n_1 = 0, n_2 = 160$, and in the second stage, $n_1 = 160, n_2 = 320$.

First, the maximum amplitude sample ε_{max} and its location relative to the first sample in the stage i_{max} are determined.

$$\varepsilon_{max} = \max(|\varepsilon(n)|)\, and\, i_{max} = argmax(|\varepsilon(n)|) - n_1\, for\, n_1 \le n < n_2 \tag{1547}$$

Based on the value of ε_{max}, the scale factor sf is determined.

$$sf = \begin{cases} \dfrac{0.67}{\varepsilon_{max}} & if\, \varepsilon_{max} > 1 \\ 0.67 & if\, \varepsilon_{max} \le 1 \end{cases} \tag{1548}$$

The scale factor sf and the previous scale factor parameter from the memory sf_{prev} are then used to determine the parameter scale step ss.

$$ss = \begin{cases} 1 & if\, sf_{prev} \le 0 \\ e^{\left(\frac{1}{i_{max}}\right)*\log\left(\frac{sf}{sf_{prev}}\right)} & otherwise \end{cases} \tag{1549}$$

If $sf_{prev} \le 0$, then $sf_{prev} = ss$

The output of the non-linear processing $\varepsilon_{NL}(n)$ is derived as per

$$\varepsilon_{NL}(n) = \begin{cases} \varepsilon^2(n) \times sf_{prev} & if\, \varepsilon(n) \le 0 \\ -\varepsilon^2(n) \times sf_{prev} & if\, \varepsilon(n) < 0 \end{cases} \quad for\, n_1 \le n < n_2 \tag{1550}$$

The previous scale factor parameter is updated recursively for all $j < i_{max}$ according to

```
for(j=n₁; j< n₂; j++)
    if(j<iₘₐₓ)
```

$$sf_{prev} = sf_{prev} \times ss$$
```
    end
end
```

6.1.5.1.3   De-quantization of high band parameters

The high band LSF, gain shape and gain frame parameters are de-quantized by looking up the quantization tables for these parameters based on the indices. The LSF de-quantization is done as follows:

6.1.5.1.3.1   LSF de-quantizing

The first five LSFs $\hat{\rho}_k^{SHB}$, $k = 1,\ldots,5$ are reconstructed directly from the received CB indices. The mirroring frequency and optimal grid are reconstructed from the received indics. The upper-half ot the coefficients $\hat{\rho}_k^{SHB}$, $k = 6,\ldots,10$ are reconstructed by flipping the lower-half of the coefficients over the reconstructed mirroring frequency, rescaling and then smoothing with the reconstructed optimal grid, as described in subclause 5.2.4.1.3.1.

$$\hat{\rho}_{k+5}^{SHB} = (1 - \lambda_k)\tilde{\rho}_k^{SHB} + \lambda_k \tilde{g}_{i^{opt},k} \quad k = 1,\ldots,5 \tag{1551}$$

Using the received VQ index parameter for the gain shape, the de-quantized gain shape vector that contains the gain shape parameter in the log domain is retrieved. The quantized gain shape parameters $gs_q(j)$ for $j = 1,\ldots 4$ are then obtained from the log domain values by inverse logarithm operation.

The de-quantized frame gain parameter GF_q is obtained by obtaining the log domain frame gain value from the table and by converting this back into linear domain by inverse logarithm operation.

For the bit rates of 24.4 kb/s and 32 kb/s, the de-quantized high band subframe residual energy $\hat{\vartheta}_{res\,q}(j)$, the de-quantized high band target energy, ϑ_q and the mixing factor, fac_q, are obtained by table lookup using the respective received indices.

6.1.5.1.4   LSP interpolation

Refer to subclauses 5.2.6.1.4 and 5.2.6.1.4.2 for 24.4 kbps and 32kbps LSP interpolation.

6.1.5.1.4.1   LSP interpolation at 13.2 kbps and 16.4 kbps

Refer to subclause 5.2.4.1.4.1

6.1.5.1.5   Spectral flip in time domain

The non-linear excitation $\varepsilon_{NL}(n)$ is spectrally flipped so that the high band portion of the excitation is modulated down to the low frequency region. This spectral flip is accomplished in time domain

$$\varepsilon_{flipped}(n) = (-1)^n \varepsilon_{NL}(n), for\, n = 0,\ldots 319 \tag{1552}$$

6.1.5.1.6   Down-sample using all-pass filters

$\varepsilon_{flipped}(n)$ is then decimated using a pair of all pass filters to obtain an 8 KHz bandwidth (16 KHz sampled) excitation singal $\varepsilon_{16k}(n)$. This is done by filtering the even samples of $\varepsilon_{flipped}(n)$ by an all pass filter whose transfer function is given by

$$H_{d,1} = \left(\frac{a_{0,1} + z^{-1}}{1 + a_{0,1}z^{-1}}\right)\left(\frac{a_{1,1} + z^{-1}}{1 + a_{1,1}z^{-1}}\right)\left(\frac{a_{2,1} + z^{-1}}{1 + a_{1,1}z^{-1}}\right) \tag{1553}$$

And the odd samples of $\varepsilon_{flipped}(n)$ by an all pass filter whose transfer function is given by

$$H_{d,2} = \left(\frac{a_{0,2} + z^{-1}}{1 + a_{0,2} z^{-1}} \right) \left(\frac{a_{1,2} + z^{-1}}{1 + a_{1,2} z^{-1}} \right) \left(\frac{a_{2,2} + z^{-1}}{1 + a_{1,2} z^{-1}} \right) \tag{1554}$$

The 16 KHz sampled excitation signal $\varepsilon_{16k}(n)\, for\, n = 0,\ldots 159$ are obtained by averaging the outputs of the above filter.

These filter coefficients are described in subclause 5.2.6.1.9.

6.1.5.1.7   Adaptive spectral whitening

Due to the nonlinear processing applied to obtain the excitation signal $\varepsilon_{16k}(n)$, the spectrum of this excitation is no longer flat. In order to flatten the spectrum of the excitation signal $\varepsilon_{16k}(n)$, 4th order linear prediction coefficients are estimated from $\varepsilon_{16k}(n)$. The spectrum of $\varepsilon_{16k}(n)$ is then flattened by inverse filtering $\varepsilon_{16k}(n)$ using the linear prediction filter.

The first step in the adaptive whitening process is to estimate the autocorrelation of the excitation signal

$$r_{exc}^{SHB}(k) = \sum_{n=k}^{L-1} \varepsilon_{16k}(n) \times \varepsilon_{16k}(n-k),\, k = 0,\ldots,4\, and\, n = 0,\ldots 159 \tag{1555}$$

A bandwidth expansion is applied to the autocorrelation coefficients by multiplying the coefficients by the expansion function:

$$\hat{r}_{exc}^{SHB}(k) = r_{exc}^{SHB}(k) \times wac(k),\, k = 0,\ldots,4 \tag{1556}$$

The bandwidth expanded autocorrelation coefficients are used to obtain LP filter coefficients, $a_k^{WHT}, k = 1,\ldots,5,$ by solving the following set of equations using the Levinson-Durbin algorithm as described in section.

$$\sum_{k=1}^{4} a_k^{WHT} \times \hat{r}_{exc}^{SHB}(i-k) = -\hat{r}_{exc}^{SHB}(i),\, i = 1,\ldots 4 \tag{1557}$$

It must be noted that $a_1^{WHT} = 1$.

The whitened excitation signal $\varepsilon_{WHT}(n)$ is obtained from $\varepsilon_{16k}(n)$ by inverse filtering

$$\varepsilon_{WHT}(n) = \varepsilon_{16k}(n) - \sum_{k=1}^{4} a_k^{WHT} \varepsilon_{16k}(n-k) \tag{1558}$$

4 samples of $\varepsilon_{16k}(n)$ from the previous frame are used as memory for the above filtering operation.

For bit rates 24.4 kb/s and 32 kb/s, the whitened excitation is further modulated by the normalized residual energy parameter $\hat{\vartheta}_{res}(j)$. In other words, for bitrates 24.4 kb/s and 32 kb/s,

$$\varepsilon_{WHT}((j-1) \times 80 + n) = \varepsilon_{WHT}((j-1) \times 80 + n) \times \hat{\vartheta}_{res}(j) \quad for\, (j-1) \times 80 \le n < j \times 80\, and\, j = 1,\ldots 4 \tag{1559}$$

6.1.5.1.8   Envelope modulated noise mixing

To the whitened excitation, a random noise vector whose amplitude has been modulated by the envelope of the whitened excitation is mixed using a mixing ratio that is dependent on the extent of voicing in the low band.

First, $eabs(n) = |\varepsilon_{WHT}(n)|$ is calculated and then the envelope of the envelope of the whitened excitation signal $\varepsilon_{WHT}(n)$ is calculated by smoothing $eabs(n)$

$$envNE(n) = \alpha_1 \times eabs(n) + \alpha_2 \times envNE(n-1) \tag{1560}$$

In SWB, the factors α_1 and α_2 are calculated using the voicing factors, Vf_i for subframes $i = 1, \ldots 4$, determined from the low band ACELP encoder. The average of the 4 voicing factors, $Vf = 0.25 \times \sum_{i=1,..4} Vf_i$, is calculated and modified as $Vf = 1.09875 - 0.49875 \times Vf$. This is then confined to values between 0.6 and 0.999. Then α_1 and α_2 are estimated as

$$\alpha_1 = 1 - Vf \tag{1561}$$

$$\alpha_2 = -Vf \tag{1562}$$

However, for bit rates 16.4 kb/s and 24.4 kb/s and if TBE was not used in the previous frame, α_1 and α_2 are set to

$$\alpha_1 = 0.2 \tag{1563}$$

$$\alpha_2 = -0.8 \tag{1564}$$

and for $n = 0$, $envNE(n-1)$ is substituted by an approximated value $envNE_{prev,approx}$ as

$$envNE_{prev,approx} = \frac{1}{20} \sum_{n=0}^{19} eabs(n) \tag{1565}$$

In WB mode, the factors α_1 and α_2 are initialized to $\alpha_1 = 0.05$ and $\alpha_2 = -0.96$. However, if the bitrate is 9.6kb/s, they might get reset to $\alpha_1 = 0.2$ and $\alpha_2 = -0.8$ if the if the low band coder type is voiced or $Vf > 0.35$, or to $\alpha_1 = 0.01$ and $\alpha_2 = -0.99$ if the low band coder type is unvoiced or $Vf < 0.2$.

A vector of random numbers, $rnd(n)$ of length 160 is then modulated by $envNE(n)$ to generate $rn_{wht}(n)$ as

$$rn_{wht}(n) = rnd(n) \times envNE(n) \tag{1566}$$

The whitened excitation is then de-emphasized with $\mu = 0.68$ which is the pre-emphasised effect since the used spectrum is flipped.

$$rn_{wht}(n) = rn_{wht}(n) + \mu \times rn_{wht}(n-1) \tag{1567}$$

If the lowband coder type is unvoiced, the excitation $rn_{wht}(n)$ is first rescaled to match the energy level of the whitened excitation $\varepsilon_{WHT}(n)$

$$rn_{wht}(n) = scale \times rn_{wht}(n) \tag{1568}$$

where $scale = \sqrt{\sum_{i=0}^{319} (\varepsilon_{wht}(n))^2 \Big/ \sum_{i=0}^{319} (rn_{wht}(n))^2}$ and then pre-emphasised with $\mu = 0.68$ to generate the final excitation which is the de-emphasised effect since the used spectrum is flipped.

$$\varepsilon_1(n) = rn_{wht}(n) - \mu \times rn_{wht}(n-1) \tag{1569}$$

If the lowband coder type was not un-voiced, the final excitation is calculated as

$$\varepsilon_1(n) = \alpha_1 \times rn_{wht}(n) + \alpha_2 \times \varepsilon_{wht}(n) \tag{1570}$$

For bit rates less than 24.4 kb/s, the mixing parameters α_1 and α_2 are estimated for other low band coder types as

$$\alpha_1 = (Vf_i)^{1/4} \tag{1571}$$

$$\alpha_2 = \sqrt{\frac{\left(\sum_{i=0}^{319} (\varepsilon_{wht}(n))^2\right) \times \left(1 - \sqrt{Vf_i}\right)}{\sum_{i=0}^{319} (rn_{wht}(n))^2}}$$

(1572)

For bit rates 24.4 kb/s and 32 kb/s, the mixing parameters α_1 is estimated for other low band coder types as follows:

$$\alpha_1 = \sqrt{(Vf_i) \times \left(1.0 - 0.15 \times fac_{formant}\right)}$$

(1573)

where the parameter $fac_{formant}$ is defined in in subclause 5.2.6.1.13.

For bit rates 24.4 kb/s and 32 kb/s, the mixing parameters α_2 is calculated as for bit rates at 13.2 kbps and 16.4kbps.

6.1.5.1.9 Spectral shaping of the noise added excitation

The excitation signal $\varepsilon_1(n)$ is then put through the high band LPC synthesis filter that is derived from the quantized LPC coefficients (see subclause 5.2.4.1.3).

For bitrates below 24.4 kb/s, a single LPC synthesis filter $\tilde{a}_k^{SHB}, k = 1,\dots11$ is used and the shaped excitation signal $s\varepsilon(n)$ is generated as

$$s\varepsilon(n) = \varepsilon_1(n) - \sum_{k=1}^{11} \tilde{a}_k^{SHB} \times s\varepsilon(n-k)$$

(1574)

For bitrates above 24.4 kb/s the LPC synthesis filter is applied to the excitation signal $\varepsilon_1(n)$ in four subframes based on

$$s\varepsilon(n + (j-1) \times 80) = \varepsilon_1(n + (j-1) \times 80) - \sum_{k=1}^{11} \tilde{a}_k^{SHB}(j) \times s\varepsilon(n + (j-1) \times 80 - k) \quad for \ j = 1,\dots,4$$

(1575)

6.1.5.1.10 Post processing of the shaped excitation

Refer to subclause 5.2.4.1.13

6.1.5.1.11 Gain shape update

The gain shapes are updated according to the coding type of both the current frame and the previous frame. The pitch gain of the current frame is also taken into account.

The pitch gain $pitch_{gs}$ of the current frame is calculated by $pitch\_buf(j), j = 1,\dots,N, N = 4\,or\,5$:

$$pitch_{gs} = \begin{cases} 0.25 * \sum_{j=1}^{4} pitch\_buf(j), & N = 4 \\ 0.2 * \sum_{j=1}^{5} pitch\_buf(j), & N = 5 \end{cases}$$

(1576)

If the coding type of the current frame and the previous frame are the same, or the coding type of the current frame is GENERIC and the coding type of the previous frame is VOICED, or the coding type of the current frame is VOICED

and the coding type of the previous frame is GENERIC, and the pitch gain of the current frame $pitch_{gs}$ is greater than 70, then the gain shape parameters are smoothed as follows:

$$gs_q(j) = 0.5 * Ener_{prev}(j) / Ener(j) * gs_{q\_prev}(j) + 0.5 * gs_q(j), \quad j = 1,2,3,4 \tag{1577}$$

$$Ener(j) = \sqrt{\sum_{n=0}^{79} (s\varepsilon(j*80+n)*0.0125)^2}, \quad j = 1,2,3,4 \tag{1578}$$

where $Ener(j), j = 1,2,3,4$ are the subframe energies in the shaped excitation signal of the current frame, $Ener_{prev}(j), j = 1,2,3,4$ are the subframe energies in the shaped excitation signal of the previous frameand $gs_{q\_prev}(j), j = 1,2,3,4$ are the quantized gain shape parameters of the previous frame.

6.1.5.1.12 SHB synthesis

In order to smooth the evolution of the post-processed spectrally shaped highband excitation signal at the frame boundary, the energy ratio between the current frame's overlap samples and the previous frame's overlap samples are calculated as below:

$$SclF = \sqrt{\frac{\sum_{n=L\_SHB\_LAHEAD+10}^{2*L\_SHB\_LAHEAD+10-1} \left(s\varepsilon b(n)^2\right)}{\sum_{n=0}^{L\_SHB\_LAHEAD+10-1} \left(s\varepsilon b(n)^2\right)}} \tag{1579}$$

where *L_SHB_LAHEAD* is 20 samples. The tenth-order LPC synthesis performed as described according to subclause 6.1.5.1.9 uses a memory of ten samples, thus there is atleast a ten sample energy propagation from the previous frame into the current frame. When calculating the energy scaling to be applied to the current frame, it should be noted that the first 10 samples of the present frame are considered as a part of previous frame energy. If the voicing factor Vf_0 is greater than 0.75, the numerator in the above equation is attenuated by 0.25. The spectrally shaped high band excitation signal is then modified by the above scaling factor as follows:

$$s\varepsilon b(n) = \begin{cases} SclF \times s\varepsilon b(n), & 0 \le n < L\_SHB\_LAHEAD \\ (SclF \times \frac{29-i}{10} + \frac{i-19}{10}) \times s\varepsilon b(n), & L\_SHB\_LAHEAD \le n < L\_SHB\_LAHEAD+10 \end{cases} \tag{1580}$$

For bit rates 24.4 kb/s or higher, gain shape values are then up sampled from 4 values to 16 values as described in below. First subframe energies from the spectrally shaped highband excitation signal are calculated.

$$Et(j) = \sqrt{0.0125 \times \sum_{i=j\times80}^{(j+1)\times80-1} s\varepsilon b(i)^2} \text{ for } j = 0,1,2,3 \tag{1581}$$

The interpolated gain shape parameters are obtained as follows

$$gs_{temp}(k) = gs_q(floor(k/4)) \text{ for } k = 0,1,...,15$$

$$gs_{int}(k) = \frac{0.5 \times \lfloor Et(k-1) \times gs_{temp}(k-1) / Et(k) + gs_{temp}(k) \rfloor \text{ if } Et(k) \times gs_{temp}(k) > Et(k-1) \times gs_{temp}(k-1)}{gs_{temp}(k) \text{ otherwise}}$$

Based on either $gs_{int}(k)$ or $gs_q(k)$ (depending on the bit rate), the shaped highband excitation signal is scaled. The scaling is performed using overlapping windows as described below:

$$s\varepsilon_{scaled}(n-20) = \sum_{j=1}^{4} gs_q(j) \times swin_1(n-(j-1)\times 80) \times s\varepsilon(n-20) \text{ for } n=0,\dots 359 \text{ where the definition of } swin_1 \text{ is}$$

described in section 5.2.6.1.15. The scaled excitation is then finally multiplied by the quantized frame gain to obtain the highband synthesized signal.

$$syn_{scaled}(n-20) = gf \times s\varepsilon_{scaled}(n-20) \text{ for } n=0,\dots 359$$

The overlap portion from the previous frame are then added to the first 20 samples of the current frame of syn_{scaled}.

The highband synthesized signal is then used to generate a 32 KHz sampled highband component of the final output decoded speech signal. First the highband synthesized signal is upsampled by 2 using an interpolator. The syn_{scaled} signal is filtered through all-pass filters as per below.

$$H_{I,1} = \left(\frac{b_{0,1} + z^{-1}}{1 + b_{0,1}z^{-1}}\right)\left(\frac{b_{1,1} + z^{-1}}{1 + b_{1,1}z^{-1}}\right)\left(\frac{b_{2,1} + z^{-1}}{1 + b_{1,1}z^{-1}}\right)$$

and

$$H_{I,2} = \left(\frac{b_{0,2} + z^{-1}}{1 + b_{0,2}z^{-1}}\right)\left(\frac{b_{1,2} + z^{-1}}{1 + b_{1,2}z^{-1}}\right)\left(\frac{b_{2,2} + z^{-1}}{1 + b_{1,2}z^{-1}}\right)$$

Table 159: All-pass filter coefficients for interpolation by a factor of 2

	All pass coefficients
$b_{0,1}$	0.06056541924291
$b_{1,1}$	0.42943401549235
$b_{2,1}$	0.80873048306552
$b_{0,2}$	0.22063024829630
$b_{1,2}$	0.63593943961708
$b_{2,2}$	0.94151583095682

And the output samples from both these filters are interlaced to generate the upsampled highband signal. At bitrate of 13.2 and 16.4 kb/s where the 12.8 KHz sampled core is used, the upsampled higband signal is downmixed using a Hilbert operator.

6.1.5.1.13   Core-switching and high-band memory updates

6.1.5.1.13.1   TBE/IGF switching

When switching from ACELP to TCX core and thus from TBE to IGF, or vice versa, the transition of the high-band signals is performed implicitly by the cross-fade transition mechanism of the core signals. Due to differing delay compensations, the high-band IGF and TBE signals overlap, when switching from IGF to TBE. On the other hand, when switching from TBE to IGF the differing delay compensations cause a gap in between the high-band signals. To fill this gap and additionally provide overlapping signals for the cross-fade, a transition signal $syn_{transition}$ is generated as follows.

To obtain a continuous high-band signal to the previous frame, the overlap portion of the high-band synthesized signal syn_{scaled} is used, as described in the SHB synthesis subclause 6.1.5.1.12. This overlap portion $syn_{overlap}$ is up-sampled using the same filters $H_{I,1}$ and $H_{I,2}$. The output samples of the filters are interlaced, if the underlying core is sampled at 16 kHz or processed using a Hilbert operator to generate the up-sampled high-band signal $syn_{overlap,32k}$.

The needed length of $syn_{transition}$ is 148 samples, so the 40 samples of $syn_{overlap,32k}$ are extrapolated using the temporally mirrored end $syn_{prev,mirror}$ of the high-band synthesized signal of the previous frame syn_{prev}, where

$$syn_{prev,mirror}(n) = syn_{prev}(641-n) \quad for\ n=1,\ldots,148 \tag{1582}$$

The signals $syn_{overlap,32k}$ and $syn_{prev,mirror}$ are merged to generate $syn_{transition}$ by overlap and add using the window $win_{transition}$ as described in table 160, as

$$syn_{transition}(n) =$$
$$\begin{cases} win_{transition}(41-n)*syn_{overlap,32k}(n) + win_{transition}(n)*syn_{prev,mirror(n)}, & n=1,\ldots,40 \\ syn_{prev,mirror}, & n=41,\ldots,148 \end{cases} \tag{1583}$$

Table 160 Window for generation of transition signal

n	$win_{transition}(n)$	n	$win_{transition}(n)$	n	$win_{transition}(n)$	n	$win_{transition}(n)$
1	0.04618346	11	0.4866045	21	0.8249975	31	0.9904104
2	0.09216993	12	0.5258704	22	0.8493099	32	0.9946717
3	0.1381564	13	0.5651364	23	0.8736224	33	0.9989330
4	0.1835534	14	0.6019916	24	0.8942082	34	0.9994665
5	0.2289505	15	0.6388468	25	0.9147939	35	1
6	0.2733710	16	0.6729768	26	0.9314773	36	1
7	0.3177914	17	0.7071068	27	0.9481606	37	1
8	0.3608562	18	0.7382205	28	0.9607993	38	1
9	0.4039210	19	0.7693340	29	0.9734381	39	1
10	0.4452628	20	0.7971658	30	0.9819242	40	1

6.1.5.1.14 TEC/TFA post processing

The TEC and the TFA are complementarily activated according to the transmitted information. The TEC is performed when an onset is detected in the high frequency band at the encoder (i.e. $tec\_flag > 0$). On the other hand, the TFA is performed when the temporal envelope of the high band signal is detected to be flat at the encoder (i.e. $tfa\_flag = 1$).

Decoding the transmitted codeword, the TEC and TFA parameters are set as:

Table 161: Decoding the transmitted codeword to the TEC and TFA parameters.

Codeword	tec_flag	tfa_flag
00	0	0
01	0	1
10	1	0
11	2	0

When $tec\_flag > 0$, the temporal envelope of the high frequency band is calculated from the temporal envelope of the low frequency band and the TEC parameter and then the high frequency band signal is shaped with the calculated temporal envelope of the high frequency band. Firstly, the temporal envelope of the low frequency band of the decoded signal $env_{l,dec}(i)$ is calculated as:

$$env_{l,dec}(i) = \frac{1}{3}\sum_{m=0}^{2} env_{ls,dec}{}^{m}(i) \quad for\ i=0,\ldots,15 \tag{1584}$$

where

$$env_{ls,dec}{}^m(i) = 10\log_{10}\left(\frac{1}{(k_{u,m} - k_{l,m} + 1)} \sum_{k=k_{l,m}}^{k_{u,m}} \left|\hat{S}_{outC}(k,i)\right|^2\right) \quad \text{for } i = 0,...,15 \tag{1585}$$

and where $\hat{S}_{outC}(k,i)$ is the low frequency band signal in the QMF domain described in subclause 6.1.4.2.

Then, the temporal envelope of the high frequency band is calculated from the temporal envelope of the low frequency band and the TEC parameter as:

$$env_{h,dec}(i) = \begin{cases} 10^{0.1env_{l,dec}(i)} & tec\_flag = 1 \\ 10^{0.1env_{l,sm,dec}(i)} & tec\_flag = 2 \end{cases} \quad \text{for } i = 0,...,15 \tag{1586}$$

where $env_{l,sm,dec}(i) = 1.19205 \times \sum_{j=0}^{5} sc(j) \cdot env_{l,dec}(i-j) \quad \text{for } i = 0,...,15$.

The gain values to be applied to the high frequency band signal is calculated as

$$g_{tec}(i) = \frac{env_{h,dec}(i)}{\frac{1}{16}\sum_{j=0}^{15} env_{h,dec}(j)} \quad \text{for } i = 0,...,15 \tag{1587}$$

The gain values are limited by the lower limit $g_{tec,lower}$:

$$g_{tec,lim}(i) = \begin{cases} g_{tec,lower} & g_{tec}(i) < g_{tec,lower} \\ g_{tec}(i) & otherwise \end{cases} \quad \text{for } i = 0,...,15 \tag{1588}$$

The lower limit $g_{tec,lower}$ is defined as:

$$g_{tec,lower} = 0.5 \cdot \min\left(0.1, \quad \frac{1}{inv\_enr\_max}\right) \tag{1589}$$

where

$$inv\_enr\_max = \max\left(10^{-6}, \quad \max_{i=0\cdots15}\left(\frac{1}{(16 \cdot enr_{shb}(i))}\sum_{i=0}^{15} enr_{shb}(i)\right)\right) \tag{1590}$$

and where

$$enr_{shb}(i) = \sum_{t=i\cdot l\_subfr_{tec}}^{(i+1)\cdot l\_subfr_{tec}-1} shb(t)^2 \quad \text{for } i = 0,...,15 \tag{1591}$$

Then, the gain values are modified for maintaining the energy of the high frequency band signal of the frame

$$g_{tec,norm}(i) = \frac{g_{tec,lim}(i)}{enr_{shb}(i)} \quad \text{for } i = 0,...,15 \tag{1592}$$

Finally, the gain values are applied to the high frequency band signal

$$syn_{scaled}'(t) = \min(3.0, \quad g_{tec,norm}(i)) \cdot syn_{scaled}(t)$$
$$\text{for } t = i \cdot l\_subfr_{tec}, \cdots, (i+1) \cdot l\_subfr_{tec} - 1 \text{ and } i = 0,...,15 \tag{1593}$$

where l_subfr_{tec} is the subframe length of TEC and TFA which is 1.25 ms at the output sampling rate ($1.25R$).

When the $tfa\_flag = 1$, the temporal envelope of the high frequency band signal is determined as "flat" and then it is flattened as follows. The gain values for the TFA $g_{tfa}(i)$ are calculated by:

$$g_{tfa}(i) = \sqrt{\frac{\sum_{i=0}^{15} enr_{shb}(i)}{16 \cdot enr_{shb}(i)}} \quad \text{for } i = 0,...,15 \quad (1594)$$

By applying the gain values, the temporal envelope of the high frequency band signal is flattened:

$$syn_{scaled}'(t) = g_{tfa}(i) \cdot syn_{scaled}(t) \quad \text{for } t = i \cdot l\_subfr_{tec}, \cdots, (i+1) \cdot l\_subfr_{tec} - 1 \text{ and } i = 0,...,15 \quad (1595)$$

6.1.5.1.15 Full-band synthesis

Four bits are decoded from the bitstream to obtain the energy ratio $\hat{\varphi}_{ratio\_q}$, and then calculate the φ_{ratio} described as follows:

$$\varphi_{ratio} = 2^{\hat{\varphi}_{ratio\_q}} \quad (1596)$$

Interpolate the signal $s\hat{\varepsilon}_{FB}(n), n = 0,...319$ (see subclause 5.2.6.1.17) from 16 kHz to 48 kHz with zeros

$$s\varepsilon_{FB\_48}(n) = \begin{cases} 3.0 * s\hat{\varepsilon}_{FB}(n), & if \mod(n,3) = 0 \\ 0, & oterwise \end{cases}, for n = 0,...,959 \quad (1597)$$

The interpolated signal $s\varepsilon_{FB\_48}(n), n = 0,...959$ passes through the bandpass filter and gets the signal $s\varepsilon'_{FB\_48}(n), n = 0,...959$. The calculation of the energy φ_{FB_48} is described as follows:

$$\varphi_{FB\_48} = \sum_{n=0}^{959} s\varepsilon'_{FB\_48}(n) * s\varepsilon'_{FB\_48}(n), n = 0,...,959 \quad (1598)$$

The energy φ_{syn} of $s\hat{\varepsilon}_{FB}(n), n = 0,...319$ is calculated as follows,

$$\varphi_{syn} = \sum_{n=0}^{319} s\hat{\varepsilon}_{FB}(n) * s\hat{\varepsilon}_{FB}(n) \quad (1599)$$

The synthesized full-band signal is calculated as follows,

$$s\varepsilon_{FB}(n) = s\varepsilon'_{FB\_48}(n) * \varphi_{ratio} * \sqrt{\frac{\varphi_{syn}}{\varphi_{FB\_48}}}, n = 0,...,959 \quad (1600)$$

6.1.5.2 Multi-mode FD Bandwidth Extension decoding

The super higher band (SHB) signal for SWB signal or the higher band (HB) signal for WB signal is adaptively decoded with multi-mode BWE algorithm according to the result of the classification decision process of the SHB or HB signal decoded from the received bitstream and a determined excitation signal. In case of FB mode, the energy ratio of the current frame is decoded, the full-band (FB) signal is synthesized based upon the energy ratio or the envelope ratio calculated from low band envelope and the generated SHB frequency excitations. Combining with the low band signal decoded based on the received bitstream, the output signal is obtained.

6.1.5.2.1 SWB multi-mode FD BWE decoding

First of all, the SHB signal class of current frame is decoded. Then the spectral envelopes or spectral/time envelopes are adaptively decoded depending upon the decoded SHB signal class. Four spectral envelopes and four time envelopes are decoded from the received bitstream for TRANSIENT frames. For all of the other cases, i.e. NON-TRANSIENT

frames, fourteen spectral envelopes are decoded from the received bitstream and no time envelope is decoded. Frequency excitations are then generated according to the SHB signal class and finally, the super higher band signal is synthesised based upon the signal class, the decoded envelopes and generated frequency excitations.

6.1.5.2.1.1 Decoding the multi-mode FD BWE signal class

Two bits are decoded from the bitstream to obtain the SHB signal class according to subclause 5.2.6.2.1.3.

6.1.5.2.1.2 Decoding the spectral envelope

In TRANSIENT frames, the envelope VQ indices $\left\{ idx_{env_1}, idx_{Ierr_2} \right\}$ are used to regenerate the synthesised signal envelope,

$$\hat{f}_{rms\_SHB}(j) = \begin{cases} \hat{e}nv_1(j) & for \ j = 0,2 \\[2mm] \hat{I}err_2(0) + \dfrac{\hat{e}nv_1(0) + \hat{e}nv_1(1)}{2} & for \ j = 1 \\[4mm] \hat{I}err_2(1) + \hat{e}nv_1(1) & for \ j = 3 \end{cases} \tag{1601}$$

In Non-TRANSIENT frames, the envelope VQ indices $\left\{ idx_{env_1}, idx_{err_{21}}, idx_{err_{22}}, idx_{Ierr_{31}}, idx_{Ierr_{32}} \right\}$ are used to generate the synthesised signal envelope,

$$\hat{f}_{rms\_SHB}(j) =$$
$$\begin{cases} \hat{e}nv_1(\lfloor j/2 \rfloor) + \hat{e}rr_2(\lfloor j/2 \rfloor) & for \ j = 0,2,4,6,8,10,12 \\[2mm] \hat{I}err_3\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \dfrac{\hat{e}nv_1\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \hat{e}nv_1\left(\left\lfloor \dfrac{j+1}{2} \right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor \dfrac{j+1}{2} \right\rfloor\right)}{2} & for \ j = 1,3,5,7,9,11 \\[4mm] \hat{I}err_3\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \dfrac{\hat{e}nv_1\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right)}{2} & for \ j = 13 \end{cases} \tag{1602}$$

The final de-quantized envelope is then calculated:

$$\hat{f}_{env}(j) = 10^{\left(\hat{f}_{rms\_SHB}(j) + f_{rms\_mean}(j)\right) * fac\_e} \tag{1603}$$
$$for \ j = 0,...,13 (Non\_TRANSIENT) \ \ OR \ j = 0,1,2,3 (TRANSIENT)$$

where ,

$$fac\_e = \begin{cases} 0.025 & for \ TRANSIENT \\ 0.05 & for \ Non-TRANSIENT \end{cases} \tag{1604}$$

6.1.5.2.1.3 Decoding the time envelope

If the current frame is a TRANSIENT frame, then four bits are decoded to obtain the index of each time envelope, $t'_{rms}(j)$. This envelope is converted into the linear domain as follows:

$$\hat{t}'_{rms}(j) = 2^{t'_{rms}(j)} \qquad j = 0,\ldots,3 \tag{1605}$$

The linear domain time envelope of the previous sub-frame is preserved as $\hat{t}'^{[-1]}_{rms}$. $\hat{t}'^{[-1]}_{rms}$ is set to zero for the first frame. Time envelope de-normalization is performed after the frequency domain processing in subclause 6.1.5.2.1.6.

6.1.5.2.1.4 Windowing and time-to-frequency transformation

640-point length MDCT is used for SWB FD BWE. Refer to subclause 5.3.2.

6.1.5.2.1.5 Frequency excitation generation

The base frequency excitation signal $\hat{X}_{M\_exc\_base}(k)$ is generated from the wideband MDCT coefficients $\hat{X}_{M\_exc\_WB}(k)$ of synthesized wideband signal or from random noise depending on the decoded SHB signal class.

To Non-TRANSIENT frames, four parameters,

$$E_{FENV} = \sum_{j=0}^{SWB\_FENV-1} \hat{f}_{env}(j) \Big/ SWB\_FENV \ , \ fenvL = \sqrt{\sum_{j=230}^{245} \hat{X}_{M\_exc\_WB}(j)^2 / 16} \ ,$$

$E_T = \sqrt{\sum_{j=16}^{255} \hat{X}_{M\_exc\_WB}(j)^2 / 240}$, and spectral tilt of WB signal are calculated. When $E_{FENV} > 75$ and

$fenvL > 25$, if one of the condition: $F_{FENV} > E_T$, $tilt\_nb > 7$ and $E_{FENV} > 0.5E_T$, $tilt\_nb > 12$ is satisfied, the fricative flag $F_{fractive}$ is set to 1. It is noted that parameter $F_{fractive}$ initialized to 0. It is calculated for every frame and preserved as $F_{fractive}^{[-1]}$.

– The base frequency excitation coefficients are obtained from the wideband MDCT coefficients:

$$\hat{X}_{M\_exc\_base}(k) = \hat{X}_{M\_exc\_WB}(k) \qquad k = 0,...,K_{cut\_off} - 1 \tag{1606}$$

where K_{cut_off} is the cut-off spectrum bin of WB signal, and $K_{cut\_off} = 246$ at 13.2kbps and $K_{cut\_off} = 320$ at 32kbps .

– If the current frame is a NOISE frame or $F_{fractive}$ is equal to 1, the base frequency excitation signal is generated from linear congruential uniform random noise generator as follows

$$\hat{X}_{M\_exc\_base}(K_{cut\_off} + k) = S_{noise}(k) \qquad k = 0,...,319 \tag{1607}$$

where

$$S_{noise}(k) = \frac{\lambda_{seed}}{32768} \quad \text{and} \quad \lambda_{seed} = 12345.\lambda_{seed} + 20101 \qquad k = 0,...,319 \tag{1608}$$

Parameter λ_{seed} is initialized as 21211 and updated for each MDCT coefficient. It should be noted that $S_{noise}(k)$ is calculated for every frame.

– Otherwise, the base frequency excitation signal is copied as defined in subclause 5.2.6.2.1.5 (Frequency mapping to generate base excitation spectrum in FD BWE).

6.1.5.2.1.6 Frequency excitation normalization and spectral envelope de-normalization

Firstly, the base frequency excitation signal or the spectral envelope is adjusted depending upon the SHB signal class. Then the base frequency excitation signal is adaptively normalized to remove the original low frequency envelope information. Finally, the spectral envelopes are applied to the normalized excitation signal.

If the current frame is NORMAL and the fricative flag $F_{fractive}$ (as defined in subclause 6.1.5.2.1.5) is equal to 0, the spectral envelope and the base frequency excitation signal are firstly adjusted. The spectral envelope is adjusted as follows:

$$\begin{aligned} \hat{f}_{env}(j+1) &= \hat{f}_{env}(j) * (0.8 + 0.015 \cdot j) \quad \text{if } \hat{f}_{env}(j+1) < 0.9\hat{f}_{env}(j) \\ \hat{f}_{env}(j) &= \hat{f}_{env}(j) * (0.8 + 0.015 \cdot j) \qquad \text{if } \hat{f}_{env}(j) < 0.9\hat{f}_{env}(j+1) \end{aligned} \tag{1609}$$

where $j = 0,...,12$.

And the base frequency excitation signal is adjusted.

– while weighting factor R_1 is smaller than 1, the base frequency excitation $\hat{X}_{M\_exc\_base}(k), k = St_{dst,FD}(1)$ is adjusted by multiplying by R_1, R_1 is then increased by 0.1 and the index k is then incremented by 1.

where, weighting factor R_1 is initialized as follows:

$$R_1 = \max\left(\frac{\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)-3\big)\right| + \left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)-2\big)\right| + \varepsilon}{\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)\big)\right| + \left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)+1\big)\right| + \varepsilon}, 0.3\right) \tag{1610}$$

where $St_{dst,FD}(1)$ is defined in subclause 5.2.6.2.1.5.

– while weighting factor R_2 is larger than 1, the base frequency excitation $\hat{X}_{M\_exc\_base}(k), k = St_{dst,FD}(1)-1$ is adjusted by multiplying by R_2, R_2 is then decreased by 0.5 and the index k is then decremented by 1.

where, weighting factor R_2 is initialized as follows:

$$R_2 = \min\left(\frac{\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)\big)\right| + \left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)+1\big)\right| + \varepsilon}{\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)-3\big)\right| + \left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)-2\big)\right| + \varepsilon}, 5\right) \tag{1611}$$

– while weighting factor R_3 is larger than 1, the base frequency excitation $\hat{X}_{M\_exc\_base}(k), k = St_{dst,FD}(2)$ is adjusted by multiplying by R_3, R_3 is then increased by 0.1 and the index k is then incremented by 1.

where, weighting factor R_3 is initialized as follows:

$$R_3 = \max\left(\frac{\sum_{j=1}^{4}\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(2)-j\big)\right| + \varepsilon}{\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(2)\big)\right| + \left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(2)+1\big)\right| + \varepsilon}, 0.3\right) \tag{1612}$$

and $St_{dst,FD}(2)$ is defined in subclause 5.2.6.2.1.5.

– while weighting factor R_4 is larger than 1, the base frequency excitation $\hat{X}_{M\_exc\_base}(k), k = St_{dst,FD}(2)-1$ is adjusted by multiplying by R_4, and then R_4 is multiplied by 0.95 and the index k is decremented by 1.

where, weighting factor R_4 is initialized as follows:

$$R_4 = 0.5 \cdot \frac{\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(2)\big)\right| + \left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(2)+1\big)\right| + \varepsilon}{\sum_{j=1}^{4}\left|\hat{X}_{M\_exc\_base}\big(St_{dst,FD}(1)-j\big)\right| + \varepsilon} \tag{1613}$$

Otherwise, there is no need to adjust the base frequency signal.

$$\hat{X}_{M\_exc\_ad1}(k) = \hat{X}_{M\_exc\_base}(k) \quad k = K_{cut\_off}, ..., K_{cut\_off}+319 \tag{1614}$$

In order to normalize the base frequency excitation, the parameter of adaptive normalization length L_{norm} is calculated depending on the decoded SHB signal class and the wideband MDCT coefficients:

– The 256 wideband MDCT coefficients in the 0-6400 Hz frequency range, $\hat{X}_{M\_exc\_WB}(k), k = 0,...,255$ are split into 16 sharpness bands (16 coefficients per band). In sharpness band j, if

$$23A_{sharp}(j) > 8\sum_{k=16j}^{16j+15}\left|\hat{X}_{M\_exc\_WB}(k)\right| \text{ and } A_{sharp}(j) > 10 \text{, the counter } C_{band} \text{ is incremented by one.}$$

where $j = 0,...,15$, and the maximum magnitude of the spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is:

$$A_{sharp}(j) = \max_{k=16j,...,16j+15}\left|\hat{X}_{M\_exc\_WB}(k)\right| \qquad j = 0,...,15 \tag{1615}$$

Parameter C_{band} is initialized to 0 and calculated for every frame.

– Then the normalization length L_{norm} is obtained:

$$L_{norm} = \lfloor 0.5L_{norm\_pre} + 0.5L_{norm\_cur}\rfloor \tag{1616}$$

where the current normalization length L_{norm_cur} is calculated depending on the SHB signal class:

$$L_{norm\_cur} = \begin{cases} \lfloor 4 + 0.25C_{band}\rfloor & if \ mode = TRANSIENT \\ \lfloor 8 + 0.5C_{band}\rfloor & if \ mode = NORMAL \\ \max(\lfloor 32 + 2C_{band}\rfloor, 24) & if \ mode = HARMONIC \end{cases} \tag{1617}$$

and the current normalization length is preserved as L_{norm_pre}.

When the current frame is not NOISE and the fricative flag $F_{fractive}$ is equal to 0, the noise content of the base frequency excitation signal should be generated, and the base frequency excitation signal should be normalized to remove the core envelope information. The above algorithm is according to the adaptive normalization length L_{norm}.

The normalization envelopes are firstly calculated:

$$f_{rms\_norm}(k) = \begin{cases} \sum_{j=k-\lfloor L_{norm}/2\rfloor}^{k+\lfloor(L_{norm}-1)/2\rfloor}\left|\hat{X}_{M\_exc\_ad1}(j)\right| & K_{cut\_off} \leq k < K_{cut\_off} + 319 - \lfloor L_{norm}/2\rfloor \\ \sum_{j=k-\lfloor L_{norm}/2\rfloor}^{K_{cut\_off}+319}\left|\hat{X}_{M\_exc\_ad1}(j)\right| & K_{cut\_off} + 319 - \lfloor L_{norm}/2\rfloor \leq k < K_{cut\_off} + 320 \end{cases} \tag{1618}$$

Then the signs and the amplitudes of the base frequency excitation signal are calculated by:

$$sign(k) = \text{sgn}\left(\hat{X}_{M\_exc\_ad1}(k)\right) \qquad K_{cut\_off} \leq k < K_{cut\_off} + 320 \tag{1619}$$

$$amplitude(k) = \left|\hat{X}_{M\_exc\_ad1}(k)\right| \qquad K_{cut\_off} \leq k < K_{cut\_off} + 320 \tag{1620}$$

The adjusted coefficients are obtained by the amplitudes, the normalization envelopes and adaptive normalization length:

$$\hat{X}_{M\_exc\_ad2}(k) = amplitude(k) - \frac{f_{rms\_norm}(k)}{L_{norm}} \qquad K_{cut\_off} \leq k < K_{cut\_off} + 320 \tag{1621}$$

If $\hat{X}_{M\_exc\_ad2}(k) > 0$, the adjusted coefficients are further modified by the modification factor W_1, and then the base frequency excitation signal with the noise content is obtained by the signs and the adjusted coefficients:

$$\hat{X}_{M\_exc\_norm}(k) = \begin{cases} sign(k) \cdot (1.2 - W_1) \cdot \hat{X}_{M\_exc\_ad2}(k) & if \ \hat{X}_{M\_exc\_ad2}(k) > 0 \\ sign(k) \cdot \hat{X}_{M\_exc\_ad2}(k) & otherwise \end{cases} \tag{1622}$$

W_1 is the modification factor and can be determined as

$$W_1 = 0.4W_{cur} + 0.6W_{pre} \tag{1623}$$

where, $W_{cur} = 0.25$ for HARMONIC frame, otherwise, $W_{cur} = \max(\min(3/L_{norm}, 0.5), 0.25)$, and W_1 is preserved for the next frame.

The normalized frequency excitation is obtained by removing the core envelope information:

$$\hat{X}_{M\_exc}(k) = \begin{cases} \dfrac{\hat{X}_{M\_exc\_ad1}(k)}{f_{rms\_norm}(k)} & for \ TRANSIENT \ frame \\[3mm] \dfrac{\hat{X}_{M\_exc\_norm}(k)}{f_{rms\_norm}(k)} & for \ F_{fractive} = 0 \ and \ !NOISE \ frame \end{cases} \tag{1624}$$

where $k = K_{cut\_off}, ..., K_{cut\_off} + 319$.

Finally, the spectral envelope is applied to the normalized excitation signal to obtain the SHB coefficients.

– For TRANSIENT frames, it is achieved as follows:

$$\hat{X}_{M\_SWB}^{SWB}(k) = \hat{X}_{M\_exc}(k) \cdot \hat{f}_{env}(j) \sqrt{\frac{N_{swbcf}(j)}{\left(\sum\limits_{k=b_{swb}(j)}^{b_{swb}(j+1)-1} \hat{X}_{M\_exc}(k)^2\right) + \varepsilon_{rms}}} \tag{1625}$$

where $K_{cut\_off} + b_{swb}(j) \le k < K_{cut\_off} + b_{swb}(j+1)$ and $j = 0, ..., 3$.

– For non-TRANSIENT frames, if $\hat{F}_{class} = HARMONIC \ AND \ \left|\hat{X}_{M\_exc}(k)\right| < \dfrac{1}{16} \sum\limits_{k=K_{cut\_off}+16j}^{K_{cut\_off}+16j+15} \left|\hat{X}_{M\_exc}(k)\right|$, $K_{cut\_off} + 16j \le k \le K_{cut\_off} + 16j + 15$ and $j = 0, ..., 18$, the frequency signal is multiplied by 0.2:

$$\hat{X}_{M\_exc\_temp}(k) = 0.2 \cdot \hat{X}_{M\_exc}(k) \tag{1626}$$

Otherwise, there is no adjustment.

$$\hat{X}_{M\_exc\_temp}(k) = \hat{X}_{M\_exc}(k) \tag{1627}$$

Then the MDCT coefficients of the reconstructed SHB signal are further adaptively adjusted with different modes:

$$\hat{X}_{M\_SWB\_temp}(k) = \hat{X}_{M\_exc\_temp}(k) \cdot \sqrt{\frac{b_{swb}(j+L) - b_{swb}(j)}{\left(\sum\limits_{k=K_{cut\_off}+b_{swb}(j)}^{K_{cut\_off}+b_{swb}(j+L)-1} \hat{X}_{M\_exc\_temp}(k)^2\right) + \varepsilon_{rms}}} \tag{1628}$$

where $L = 1$ for NOISE or NORMAL frame and $L = 2$ for HARMONIC frame, $K_{cut\_off} + b_{swb}(j) \le k < K_{cut\_off} + b_{swb}(j+1)$ and the index j is incremented by L if j is smaller than 14.

The weighted envelopes $W_{fenv}(j)$ are obtained from the spectral envelopes of the current frame and the previous frame:

$$W_{fenv}(j) = W_2 \cdot \hat{f}_{env}^{[-1]}(j) + (1 - W_2) \cdot \hat{f}_{env}(j) \qquad j = 0,...,13 \qquad (1629)$$

where,

$$W_2 = \begin{cases} 0.5 \cdot \dfrac{E_T}{E_T^{[-1]}} & if \ \dfrac{E_T^{[-1]}}{E_T} > 1.25 \\ 0.5 & otherwise \end{cases} \qquad (1630)$$

and $E_T = \sqrt{\sum_{j=16}^{255} \hat{X}_{M\_exc\_WB}(j)^2 / 240}$ is calculated for every frame and preserved as the previous

energy $E_T^{[-1]}$ for the next frame.

The index k is initialized to 0, and four weighting factors, $W_{fac1} = \sqrt{1/16 \sum_{j=230}^{245} \hat{X}_{M\_exc\_WB}(j)^2}$,

$W_{fac2} = 0.125 \cdot \left(W_{fenv}(0) - W_{fac1}\right)$, $W_{fac3}(j) = \hat{f}_{env}(j), \ 0 \le j < 13$,

$W_{fac4}(j) = smooth\_fac(j) \cdot \left(W_{fenv}(j+1) - \hat{f}_{env}(j)\right), \ 0 \le j < 13$, are calculated. The smoothing factor
$smooth\_fac(j)$ is defined in table 162.

Table 162: Smoothing factor $smooth\_fac(j)$

j	**Smoothing factor**
0	0.05
1	0.05
2	0.05
3	0.05
4	0.05
5	0.05
6	0.05
7	0.0417
8	0.0417
9	0.0417
10	0.0417
11	0.03125
12	0.03125

While the index k is smaller than 8, the frequency excitation $\hat{X}_{M\_SWB\_temp}(K_{cut\_off} + k)$ is adjusted by
multiplying W_{fac1} , and then the index k is incremented by 1, and W_{fac1} is increased by adding W_{fac2} .

In the jth sub-band $0 \le j < 13$, while the index k is smaller than $b_{swb}(j+1)$, the frequency excitation
$\hat{X}_{M\_SWB\_temp}(K_{cut\_off} + k)$ is adjusted by multiplying by $W_{fac3}(j)$, and then, the index k is incremented by
1, and $W_{fac3}(j)$ is increased by adding $W_{fac4}(j)$.

In the 13$^{\text{th}}$ sub-band, while the k is smaller than $b_{swb}(14)$, the frequency excitation $\hat{X}_{M\_SWB\_temp}(K_{cut\_off} + k)$ is adjusted by multiplying by $W_{fenv}(13)$, and then, the index k is incremented by 1.

The frequency excitation $\hat{X}_{M\_SWB\_temp}(k)$ is adjusted by the weighted spectral envelopes to obtain the final SHB frequency signal $\hat{X}_{M\_SWB}^{SWB}(k)$, $K_{cut\_off} \leq k < K_{cut\_off} + 320$.

It should be noted that, for Non-TRANSIENT frames, the spectral envelopes of current frame are preserved as $\hat{f}_{env}^{[-1]}(j)$. For TRANSIENT frame, the spectral envelope is calculated and preserved:

$$\hat{f}_{env}^{[-1]}(j) = \begin{cases} \hat{f}_{env}(j/4)^2 & for \quad j = 0,...,7 \\ \hat{f}_{env}(2+(j-8)/3)^2 & for \quad j = 8,...,13 \end{cases} \tag{1631}$$

Further adjustment is performed by:

$$\hat{X}_{M\_SWB}^{SHB}(K_{cut\_off} + k) = 0.5 \cdot \hat{X}_{M\_SWB}^{SHB}(K_{cut\_off} + k) \quad 0 \leq k < 4 \tag{1632}$$

6.1.5.2.1.7 Windowing and frequency-to-time transformation

640-point length inverse MDCT is used for SWB FD BWE. Refer to subclause 6.2.4.

6.1.5.2.1.8 Time domain post-processing

The SHB synthesis signal is adjusted depending upon the SHB class.

– If the current frame is TRANSIENT, the SHB synthesis signal in time domain is adjusted by the time envelopes to match the transient characteristics of the original signal.

The 640 SHB synthesized samples $\hat{s}'_{SHB}(n)$, $0 \leq n < 640$ are divided into 4 sub-frames, and the energy of each sub-frame is calculated:

$$E_{SHB}(j) = \sqrt{\frac{1}{160} \sum_{n=160j}^{160j+159} \left(\hat{s}'_{SHB}(n)\right)^2} \quad 0 \leq j < 4 \tag{1633}$$

next, the time envelope is adjusted:

$$\hat{t}_{env}(j) = \begin{cases} E_{SHB}(j) & if \quad E_{SHB}(j) < 0.8 * \hat{t}'_{rms}(j) < 2 \\ 0.8 * \hat{t}'_{rms}(j) & otherwise \end{cases} \quad 0 \leq j < 4 \tag{1634}$$

and finally, the SHB synthesis signal is adjusted as follows:

$$\hat{s}_{SHB}(n) = \hat{s}'_{SHB}(n) * \hat{t}_{env}(j) \big/ E_{SHB}(j) \tag{1635}$$

where, $160j \leq n \leq 160j + 159$ and $0 \leq j < 4$.

In this case, the value of $\hat{t}_{env}(3)$ is preserved for the next frame.

$$E_{td\_energy}^{[-1]} = \hat{t}_{env}(3) \tag{1636}$$

– Else if fricative flag $F_{fractive}$ defined in subclause 6.1.5.2.1.5 is equal to 1 and the previous fricative flag $F_{fractive}^{[-1]}$ is equal to 0, pre-echo reduction is performed and the preserved time-domain energy is updated.

Firstly, the 640 ACELP core synthesized samples, $\hat{s}_{ACELP}(n)$, $0 \leq n < 640$ are divided into 4 sub-frames, and the energy of each sub-frame is calculated:

$$E_{ACELP}^{[j]} = \sqrt{\frac{1}{160} \sum_{n=160j}^{160j+159} \left(\hat{s}_{ACELP}(n)\right)^2} \qquad 0 \leq j < 4 \tag{1637}$$

In j^{th} sub-frame, if $E_{ACELP}^{[j+1]} > 1.8 \cdot E_{ACELP}^{[j]}$ and $E_{ACELP}^{[j+1]} > 50$, the position is introduced to separate 4 sub-frames into two parts and it is initialized as 0: $pos = \begin{cases} j+1 & \text{for } j = 2 \\ j+2 & \text{for } j = 0 \text{ or } 1 \end{cases} \tag{1638}$

Next, if $pos > 2$, the SHB synthesis signal is adjusted.

$$\hat{s}_{SHB}(n) = \begin{cases} \hat{s}'_{SHB}(n) \cdot \dfrac{E_{td\_energy}^{[-1]}}{E_{td\_energy1}} & \text{for } 0 \leq n < 160 \cdot pos \\[3mm] \hat{s}'_{SHB}(n) \cdot \left(\left(1 - \dfrac{n - 160 \cdot pos}{160}\right) \dfrac{E_{td\_energy}^{[-1]}}{E_{td\_energy2}} + \dfrac{n - 160 \cdot pos}{160}\right) & \text{for } 160 \cdot pos \leq n < 160 \cdot (pos+1) \end{cases} \tag{1639}$$

where, $E_{td\_energy1}$ and $E_{td\_energy2}$ are the energies of the SHB synthesized samples $\hat{s}'_{SHB}(n)$, $0 \leq n < 160 \cdot pos$ and the energy of SHB synthesized samples $\hat{s}'_{SHB}(n)$, $160 \cdot pos \leq n < 160 \cdot (pos+1)$. $E_{td\_energy}^{[-1]}$ is the energy value from the previous frame, and if $E_{td\_energy}^{[-1]} < 0.2 \cdot E_{td\_energy1}$, the value of $E_{td\_energy}^{[-1]}$ is updated to $E_{td\_energy1}$.

– Otherwise the energy is calculated and preserved for next frame:

$$E_{td\_energy}^{[-1]} = \sqrt{\frac{1}{160} \sum_{n=480}^{639} \left(\hat{s}'_{SHB}(n)\right)^2} \tag{1640}$$

6.1.5.2.2 WB multi-mode FD BWE decoding

The HB signal class and the spectral envelopes are decoded (at 13.2kbps) or predicted (at 7.2/8kbps), and the frequency excitations are generated from the decoded low-band synthesized signal or from random noise, and then the frequency excitations are adjusted along with the signal class and decoded or predicted spectral envelopes to obtain the higher band signal.

6.1.5.2.2.1 Decoding the multi-mode FD BWE signal class

At 13.2kbps, one bit is decoded from bitstream to get the HB signal class according to subclause 5.2.6.2.2.2. And at 7.2kbps or 8kbps, the HB signal class is set to NORMAL.

6.1.5.2.2.2 Windowing and time-to-frequency transformation

320-point MDCT is used for WB FD BWE. Refer to subclause 5.3.2.

6.1.5.2.2.3 Decoding the spectral envelope

At 13.2kbps, five bits are decoded to obtain the index of spectral envelope, $\hat{k}_{ind}$. This envelope is converted into the linear domain as follows:

$$\hat{f}_{env\_WB}(j) = 2^{0.5 \cdot codebook\_HB(2 \cdot \hat{k}_{ind} + j)} \qquad j = 0,1 \tag{1641}$$

where $codebook\_HB(k)$ is defined in subclause 5.2.6.2.2.3.

The average BWE signal envelope of the current frame at 13.2kbps is preserved for the envelope estimation for 7.2kbps and 8kbps when bit-rate switching from 13.2kbps to 7.2 or 8kbps, as described as follows:

$$\hat{f}_{env\_WB\_av}^{[-1]} = 0.5\left(\hat{f}_{env\_WB}(0) + \hat{f}_{env\_WB}(1)\right) \tag{1642}$$

At 7.2kbps or 8kbps, the spectral envelope is predicted in the decoder. If the extended layer of the previous frame is different from the one of current frame, that is, $M_{last\_extl} != WB\_BWE$, the preserved spectral envelope $f_{env\_WB}^{[-1]}(j), j = 0,1$ are set to 0. In order to get the predicted spectral envelope, two bands $[128,191]$ and $[192,255]$ are firstly selected. Then three average energies are needed based on the frequency coefficients in the above two bands. Finally, the spectral envelopes used for the following frequency adjustment are obtained. In order to decrease the complexity, the energies are calculated with the MDCT coefficients $\hat{X}_{M\_core\_dec}(k), 0 \le k < 256$:

$$E_L = \sum_{k=128}^{191} \left(\hat{X}_{M\_core\_dec}(k)\right)^2 \tag{1643}$$

$$f_{env\_WB}(0) = \sum_{k=192}^{223} \left(\hat{X}_{M\_core\_dec}(k)\right)^2 \tag{1644}$$

$$f_{env\_WB}(1) = \sum_{k=224}^{255} \left(\hat{X}_{M\_core\_dec}(k)\right)^2 \tag{1645}$$

Two factors $voice\_fac = \sum_{j=0}^{3} voice\_factor(j)$, $T_{sum} = \sum_{j=0}^{3} pitch(j)$, are calculated. When $E_L < 16 \cdot max\left(f_{env\_WB}(0), f_{env\_WB}(1)\right)$ and $T_{sum} < 308$, the energy variation flag F_{ener_var} is set to 1. It should be noted that F_{ener_var} is initialized to 0 and calculated for every frame.

Then the weighting factor α_1 is initialized to 1, and updated according to the spectral envelope and code type CT:

$$\alpha_1 = \begin{cases} max\left(2 \cdot \dfrac{f_{env\_WB}(1)}{f_{env\_WB}(0)}, 0.1\right) & if \ f_{env\_WB}(0) > 2 \cdot f_{env\_WB}(1) \\ max\left(2 \cdot \dfrac{f_{env\_WB}(0)}{f_{env\_WB}(1)}, 0.1\right) & esle \ if \ f_{env\_WB}(1) > 2 \cdot f_{env\_WB}(0) \ AND \ CT != UNVOICED \end{cases} \tag{1646}$$

and the spectral envelope is accordingly adjusted by:

$$f_{env\_WB\_ad1}(0) = \begin{cases} \alpha_1 \cdot f_{env\_WB}(0) & if \ f_{env\_WB}(0) > 2 \cdot f_{env\_WB}(1) \\ f_{env\_WB}(0) & otherwise \end{cases} \tag{1647}$$

$$f_{env\_WB\_ad1}(1) = \begin{cases} \alpha_1 \cdot f_{env\_WB}(1) & if \ f_{env\_WB}(1) > 2 \cdot f_{env\_WB}(0) \ AND \ CT != UNVOICED \\ f_{env\_WB}(1) & otherwise \end{cases} \tag{1648}$$

Next, the first envelope is further adjusted.

– The first envelope is firstly adjusted with $f_{env\_WB\_ad1}(j), j = 0,1$ by:

$$f_{env\_WB\_ad2}(0) = \sqrt{(f_{env\_WB\_ad1}(0) + f_{env\_WB\_ad1}(1))/64} \tag{1649}$$

– If the conditions: $CT != AUDIO$, $CT != UNVOICED$, and $F_{ener\_var} = 0$, are all satisfied, the first envelope is adjusted by:

$$f_{env\_WB\_ad3}(0) = 1.5 \cdot f_{env\_WB\_ad2}(0) \tag{1650}$$

Otherwise, there is no adjustment.

$$f_{env\_WB\_ad3}(0) = f_{env\_WB\_ad2}(0) \tag{1651}$$

- If the conditions: $CT != TRANSITION$, $CT != AUDIO$, $CT != UNVOICED$, $\sqrt{E_L} > 40 \cdot f_{env\_WB\_ad3}(0)$, $\alpha_1 > 0.9$, $F_{adj} = 0$, are all satisfied, the first envelope is adjusted as follows:

$$f_{env\_WB\_ad4}(0) = \min\left(0.025 \cdot \sqrt{E_T} \Big/ f_{env\_WB\_ad3}(0),\ 4\right) \cdot f_{env\_WB\_ad3}(0) \tag{1652}$$

and

$$f_{env\_WB\_ad5}(0) = \begin{cases} 0.3 f_{env\_WB\_ad4}(0) + 0.7 f_{env\_WB}^{[-1]}(0) & \text{if } f_{env\_WB\_ad4}(0) > f_{env\_WB}^{[-1]}(0) \\ f_{env\_WB\_ad4}(0) & \text{otherwise} \end{cases} \tag{1653}$$

where, the adjustment flag F_{adj} is initialized to 0. If the coder type of current frame is equal to that of the previous frame and the first spectral envelope is larger than the envelope of the previous frame, that is, $CT = CT^{[-1]}$ AND $f_{env\_WB\_ad3}(0) > f_{env\_WB}^{[-1]}(0)$, the adjustment flag F_{adj} is set to 1. The values $f_{env\_WB}^{[-1]}(j), j = 0,...,1$ are the spectral envelopes of the previous frame.

Otherwise, there is no adjustment.

$$f_{env\_WB\_ad5}(0) = f_{env\_WB\_ad3}(0) \tag{1654}$$

- If the conditions: $\sqrt{E_L} > 64 \cdot \min(1.5, \max(0.5, 77 \cdot voice\_fac/T_{sum}) \cdot f_{env\_WB\_ad5}(0)$, $3 \cdot f_{env\_WB\_ad5}(0) \cdot f_{env\_WB\_ad5}(0) < \sqrt{E_L}$, $CT^{[-1]} != UNVOICED$, are all satisfied, the envelope variation flag F_{env_var} initialized to 0 is set to 1, and the first envelope is adjusted as follows:

$$f_{env\_WB\_ad6}(0) = \min\left(0.015625 \cdot \sqrt{E_T} \Big/ f_{env\_WB\_ad5}(0),\ 4\right) \cdot f_{env\_WB\_ad5}(0) \tag{1655}$$

and

$$f_{env\_WB\_ad7}(0) = \begin{cases} 0.3 f_{env\_WB\_ad6}(0) + 0.7 f_{env\_WB}^{[-1]}(0) & \text{if } f_{env\_WB\_ad6}(0) > f_{env\_WB}^{[-1]}(0) \\ f_{env\_WB\_ad6}(0) & \text{otherwise} \end{cases} \tag{1656}$$

Otherwise, there is no adjustment.

$$f_{env\_WB\_ad7}(0) = f_{env\_WB\_ad5}(0) \tag{1657}$$

- If the coder types of current frame or previous frame is UNVOICED, the first envelope is adjusted as follows:

$$f_{env\_WB\_ad8}(0) = \begin{cases} 0.5 \cdot f_{env\_WB\_ad7}(0) & \text{if } CT = UNVOICED \text{ or } CT^{[-1]} = UNVOICED \\ f_{env\_WB\_ad7}(0) & \text{otherwise} \end{cases} \tag{1658}$$

- If the coder types of current frame is not AUDIO, that is, $CT != AUDIO$, the first envelope is adjusted as follows:

$$f_{env\_WB\_ad9}(0) = \begin{cases} f_{env\_WB\_ad8}(0) \cdot \min(2, \max(0.125, T_{sum}/400)) / \max(1.2 \cdot voice\_fac, 1.0) & \text{if } CT != AUDIO \\ f_{env\_WB\_ad8}(0) & \text{otherwise} \end{cases} \tag{1659}$$

- If the last core bit-rate is larger than 8000, and the first spectral envelope is larger than the average BWE signal envelope of the previous frame, that is, $last\_core\_bitrate > 8000\ AND\ f_{env\_WB\_ad9}(0) > f^{[-1]}_{env\_WB\_av}$, the adjustment is by:

$$f_{env\_WB\_ad10}(0) = 0.1 \cdot f_{env\_WB\_ad9}(0) + 0.9 \cdot \hat{f}^{[-1]}_{env\_WB\_av} \tag{1660}$$

Otherwise, there is no adjustment:

$$f_{env\_WB\_ad10}(0) = f_{env\_WB\_ad9}(0) \tag{1661}$$

- If the extended layer of the previous frame is different from the one of current frame, the adjustment is as follows:

$$f_{env\_WB\_ad11}(0) = 0.5 \cdot f_{env\_WB\_ad10}(0) \quad if\ M_{last\_extl} \mathrel{!=} WB\_BWE \tag{1662}$$

Otherwise, there is no adjustment:

$$f_{env\_WB\_ad11}(0) = f_{env\_WB\_ad10}(0) \tag{1663}$$

Finally, the spectral envelopes are adjusted as follows:

$$f_{env\_WB\_ad2}(1) = \begin{cases} 1.5 \cdot f_{env\_WB\_ad11}(0) & if\ F_{env\_var} = 1 \\ f_{env\_WB\_ad11}(0) & otherwise \end{cases} \tag{1664}$$

and

$$\hat{f}_{env\_WB}(0) = \begin{cases} 0.75 \cdot f_{env\_WB\_ad11}(0) & if\ F_{env\_var} = 1 \\ f_{env\_WB\_ad11}(0) & otherwise \end{cases} \tag{1665}$$

$$\hat{f}_{env\_WB}(1) = \begin{cases} 0.5 \cdot f_{env\_WB\_ad2}(1) & if\ CT = UNVOICED\ OR\ CT^{[-1]} = UNVOICED \\ f_{env\_WB\_ad2}(1) & otherwise \end{cases} \tag{1666}$$

The spectral envelopes $\hat{f}_{env\_WB}(j), j = 0,1$ are used for the following frequency adjustment.

6.1.5.2.2.4 Frequency excitation generation

The base frequency excitation signal $\hat{X}_{M\_exc\_base\_WB}(k)$ is generated from the MDCT coefficients $\hat{X}_{M\_core\_dec}(k)$ of the core decoded signal or from random noise depending upon the bit-rate and coder type CT. The core type flag F_{core_type} is introduced. It is initialized to 1, and is set to 0 if the coder type is not AUDIO and the total bit-rate is not larger than 8000.

- The LF MDCT coefficients are obtained from the MDCT coefficients of core decoded signal:

$$\hat{X}_{M\_exc\_base\_WB}(k) = \hat{X}_{M\_core\_dec}(k) \qquad k = 0,...,239 \tag{1667}$$

- If the coder type of current frame is UNVOICED, the base frequency excitation signal is generated from linear congruential uniform random noise generator as follows:

$$\hat{X}_{M\_exc\_base\_WB}(240 + k) = S_{noise}(k) \qquad k = 0,...,79 \tag{1668}$$

where

$$S_{noise}(k) = \frac{\lambda_{seed}}{32768} \quad and \quad \lambda_{seed} = 12345.\lambda_{seed} + 20101 \qquad k = 0,...,79 \tag{1669}$$

Parameter λ_{seed} is initialized as 21211 and updated for each MDCT coefficient. It is noted that $S_{noise}(k)$ is calculated for every frame.

– Otherwise, the base frequency excitation signal is copied as defined in subclause 5.2.6.2.1.5.

6.1.5.2.2.5 Frequency excitation normalization and spectral envelope de-normalization

In order to normalize the base frequency excitation to remove the original low frequency envelope information, the parameter of adaptive normalization length L_{norm} is calculated depending on the HB signal class and the MDCT coefficients of core decoded signal:

– The 256 MDCT coefficients in the 0-6400 Hz frequency range, $\hat{X}_{M\_core\_dec}(k), k = 0,...,255$ are split into 16 sharpness bands (16 coefficients per band). In sharpness band j, if $19 \cdot A_{sharp}(j) > 4 \sum_{k=16j}^{16j+15} \left| \hat{X}_{M\_core\_dec}(k) \right|$ and $A_{sharp}(j) > 10$, the counter C_{band} is incremented by one.

where $j = 0,...,15$, and the maximum magnitude of the spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is:

$$A_{sharp}(j) = \max_{k=16j,...,16j+15} \left| \hat{X}_{M\_core\_dec}(k) \right| \qquad j = 0,...,15 \qquad (1670)$$

Parameter C_{band} is initialized to 0 and calculated for every frame.

– Then the normalization length L_{norm} is obtained:

$$L_{norm} = \left\lfloor 0.5 L_{norm\_pre} + 0.5 L_{norm\_cur} \right\rfloor \qquad (1671)$$

where the current normalization length L_{norm_cur} is calculated depending on the HB signal class:

$$L_{norm\_cur} = \begin{pmatrix} \left\lfloor 8 + 0.5 C_{band} \right\rfloor & if \ mode = NORMAL \\ \max\left(\left\lfloor 32 + 2 C_{band} \right\rfloor, 24 \right) & if \ mode = HARMONIC \end{pmatrix} \qquad (1672)$$

and the current normalization length is preserved as L_{norm_pre}.

Then, according to the adaptive normalization length, the noise content of the base frequency excitation signal is generated, and the base frequency excitation signal is normalized to remove the core envelope information.

– The normalized envelope is calculated:

$$f_{rms\_norm}(k) = \begin{cases} \sum_{j=k-\left\lfloor L_{norm}/2 \right\rfloor}^{k+\left\lfloor (L_{norm}-1)/2 \right\rfloor} \left| \hat{X}_{M\_exc\_base\_WB}(j) \right| & 240 \le k < 319 - \left\lfloor L_{norm}/2 \right\rfloor \\ \sum_{j=k-\left\lfloor L_{norm}/2 \right\rfloor}^{319} \left| \hat{X}_{M\_exc\_base\_WB}(j) \right| & 319 - \left\lfloor L_{norm}/2 \right\rfloor \le k < 320 \end{cases} \qquad (1673)$$

– If the bitrate is 7200 or 8000 and the coder type of current frame is not UNVOICED, the signs and amplitudes of HB coefficients are calculated by:

$$sign(k) = \mathrm{sgn}\left(\hat{X}_{M\_exc\_base\_WB}(k) \right) \qquad 240 \le k < 320 \qquad (1674)$$

$$amplitude(k) = \left| \hat{X}_{M\_exc\_base\_WB}(k) \right| \qquad 240 \le k < 320 \qquad (1675)$$

The adjusted coefficients are obtained by the amplitudes, the normalization envelopes and adaptive normalization length:

$$\hat{X}_{M\_exc\_ad}(k) = amplitude(k) - 0.45 \cdot \frac{f_{rms\_norm}(k)}{L_{norm}} \qquad 240 \leq k < 320 \qquad (1676)$$

If $\hat{X}_{M\_exc\_ad}(k) > 0$, the adjusted coefficients are modified further by the modification factor W_1, and then the base frequency excitation signal with the noise content is obtained by the signs and the adjusted coefficients:

$$\hat{X}_{M\_exc\_norm}(k) = \begin{cases} sign(k) \cdot (0.55 - W_1) \cdot \hat{X}_{M\_exc\_ad}(k) & if \ \hat{X}_{M\_exc\_ad}(k) > 0 \\ sign(k) \cdot \hat{X}_{M\_exc\_ad}(k) & otherwise \end{cases} \qquad (1677)$$

where, the modification factor $W_1 = 0.25$ for HARMONIC frame, otherwise, $W_1 = \max(\min(3/L_{norm}, 0.5), 0.25)$.

– Otherwise, there is no adjustment, that is,

$$\hat{X}_{M\_exc\_norm}(k) = \hat{X}_{M\_exc\_base\_WB}(k) \qquad 240 \leq k < 320 \qquad (1678)$$

– Next, the adjusted frequency excitation signal is normalized to remove the core envelope information:

$$\hat{X}_{M\_exc\_WB\_norm}(k) = \frac{\hat{X}_{M\_exc\_norm}(k)}{f_{rms\_norm}(k)} \qquad 240 \leq k < 320 \qquad (1679)$$

– If the coder type of current frame is not UNVOICED, the frequency signal is adaptively adjusted further as follows:

$$\hat{X}_{M\_exc\_WB}(k) = \begin{cases} \hat{X}_{M\_exc\_WB\_norm}(k) \cdot \sqrt{\dfrac{80}{\left(\displaystyle\sum_{l=240}^{319} \hat{X}_{M\_exc\_WB\_nrom}(l)^2\right) + \varepsilon_{rms}}} & for \ HARMONIC \ frame \\[4em] \hat{X}_{M\_exc\_WB\_norm}(k) \cdot \sqrt{\dfrac{N_{swbc}f(j)}{\left(\displaystyle\sum_{l=b_{swb}(j)}^{b_{swb}(j+1)-1} \hat{X}_{M\_exc\_norm}(l)^2\right) + \varepsilon_{rms}}} & otherwise \\[2em] 0 \leq j < 4, b_{swb}(j) \leq k < b_{swb}(j+1) \end{cases} \qquad (1680)$$

where, $b_{swb}(j)$ and $N_{swbc}f(j)$, $0 \leq j < 4$ are defined in table 163.

Otherwise, there is no adjustment.

$$\hat{X}_{M\_exc\_WB}(k) = \hat{X}_{M\_exc\_WB\_norm}(k) \quad for \ 240 \leq k < 320 \qquad (1681)$$

Table 163: Sub-band boundaries and number of coefficients per sub-band in NORMAL frames

j	$b_{swb}(j)$	$N_{swbc}f(j)$
0	240	16
1	256	24
2	280	16
3	304	24
4	320	-

Finally, the spectral envelope is applied to the normalized excitation signal to obtain the HB coefficients.

Three parameters: the energy E_L, two control factors α, β, are calculated according to the core type flag F_{core_type}, coder type and bit-rate as follows:

$$E_L = \begin{cases} \left. \begin{cases} \sum_{k=160}^{239}\left|\hat{X}_{M\_core\_dec}(k)\right| & if \; CT^{(-1)} != AUDIO \; AND \; bitrate \leq 8000 \\ \sum_{k=80}^{239}\left|\hat{X}_{M\_core\_dec}(k)\right| & otherwise \end{cases} \right\} & if \; F_{core\_type} = 1 \\ \left. \begin{cases} \sum_{k=80}^{239}\left|\hat{X}_{M\_core\_dec}(k)\right| & if \; CT^{(-1)} = AUDIO \\ \sum_{k=160}^{239}\left|\hat{X}_{M\_core\_dec}(k)\right| & otherwise \end{cases} \right\} & otherwise \end{cases} \tag{1682}$$

$$\alpha = \begin{cases} \left. \begin{cases} 0.8 & if \; bitrate \leq 8000 \\ 0.5 & otherwise \end{cases} \right\} & if \; F_{core\_type} = 1 \\ \left. \begin{cases} 0.4 & if \; CT = CT^{[-1]} \; AND \; \hat{f}_{env\_WB}(0) > f_{env\_WB}^{[-1]}(0) \\ 0.6 & otherwise \end{cases} \right\} & otherwise \end{cases} \tag{1683}$$

$$\beta = \begin{cases} \left. \begin{cases} 1.25 & if \; bitrate \leq 8000 \\ 2.0 & otherwise \end{cases} \right\} & if \; F_{core\_type} = 1 \\ \left. \begin{cases} 2.5 & if \; CT = CT^{[-1]} \; AND \; \hat{f}_{env\_WB}(0) > f_{env\_WB}^{[-1]}(0) \\ 1.67 & otherwise \end{cases} \right\} & otherwise \end{cases} \tag{1684}$$

When $F_{core\_type} = 0$, if $0.5 \cdot E_L^{[-1]} < E_L < 2 \cdot E_L^{[-1]}$ and $F_{env\_ad}^{[-1]} = 1$, or the coder type is GENERIC, the envelope adjustment flag F_{env_ad} is set to 1 and the spectral envelope is adjusted:

$$\hat{f}_{env\_WB\_ad1}(j) = 0.5 \cdot \hat{f}_{env\_WB}(j) \quad for \; j = 0,1 \tag{1685}$$

where, $E_L^{[-1]}$ is the energy of previous frame, and $F_{env\_ad}^{[-1]}$ is the previous envelope adjustment flag. It is noted that the envelope adjustment flag F_{env_ad} is initialized to 0 and preserved for the next frame.

Otherwise, there is no adjustment:

$$\hat{f}_{env\_WB\_ad1}(j) = \hat{f}_{env\_WB}(j) \quad for \; j = 0,1 \tag{1686}$$

The spectral envelopes are smoothed by the one between the current frame and the previous frame according to the following conditions, and when the current and previous frames apply different extended layer or the current frame is lost frame, the previous spectral envelopes are set to the current spectral envelopes $f_{env\_WB}^{[-1]}(j) = \hat{f}_{env\_WB}(j), j = 0,1$.

When the current frame is NORMAL, or the current frame is HARMONIC and $\hat{f}_{env\_WB\_ad1}(1) < 0.25 \cdot \hat{f}_{env\_WB\_ad1}(0)$, the adjustment is as follows.

– If $bitrate \leq 8000$ and $M_{last\_extl} = WB\_BWE$, and at least one of the coder types of current and previous frames is AUDIO, that is, $CT^{[-1]} = AUDIO \; AND \; CT != AUDIO$ or $CT^{[-1]} != AUDIO \; AND \; CT = AUDIO$, the smoothing process is performed by:

$$\begin{cases} \hat{f}_{env\_WB\_ad2}(0) = 0.3 \cdot \hat{f}_{env\_WB\_ad1}(0) + 0.7 \cdot f^{[-1]}_{env\_WB}(0) \\ \hat{f}_{env\_WB\_ad2}(1) = 0.3 \cdot \hat{f}_{env\_WB\_ad1}(1) + 0.7 \cdot f^{[-1]}_{env\_WB}(1) \end{cases} \quad if \quad \hat{f}_{env\_WB\_ad1}(0) > f^{[-1]}_{env\_WB}(0) \\ \begin{cases} \hat{f}_{env\_WB\_ad2}(0) = 0.5 \cdot \hat{f}_{env\_WB\_ad1}(0) + 0.5 \cdot f^{[-1]}_{env\_WB}(0) \\ \hat{f}_{env\_WB\_ad2}(1) = 0.4 \cdot \hat{f}_{env\_WB\_ad1}(j) + 0.4 \cdot f^{[-1]}_{env\_WB}(1) \end{cases} \quad otherwise \tag{1687}$$

Else if the conditions: $M_{last\_extl} = WB\_BWE$, $f^{[-1]}_{env\_WB}(0) \cdot E_L < \hat{f}_{env\_WB\_ad1}(0) \cdot E_L^{[-1]}$, $\hat{f}_{env\_WB\_ad1}(0) > f^{[-1]}_{env\_WB}(0)$, $CT != AUDIO$, $CT != UNVOICED$, and $bitrate \leq 8000$, are all satisfied, the smoothing process is performed by:

$$\hat{f}_{env\_WB\_ad2}(j) = 0.3 \cdot \hat{f}_{env\_WB\_ad1}(j) + 0.7 \cdot f^{[-1]}_{env\_WB}(j) \quad for \ j = 0,1 \tag{1688}$$

where, $E_L^{[-1]}$ is the energy of previous frame, and the E_L is preserved for the next frame at the end of spectral envelope adjustment.

Else if the conditions: $M_{last\_extl} = WB\_BWE$, $\alpha \cdot E_L^{[-1]} < E_L < \beta \cdot E_L^{[-1]}$, and $CT^{[-1]} != UNVOICED$, are all satisfied, the smoothing process is performed by:

$$\hat{f}_{env\_WB\_ad2}(j) = 0.5 \cdot \hat{f}_{env\_WB\_ad1}(j) + 0.5 \cdot f^{[-1]}_{env\_WB}(j) \quad for \ j = 0,1 \tag{1689}$$

Otherwise, there is no adjustment.

$$\hat{f}_{env\_WB\_ad2}(j) = \hat{f}_{env\_WB\_ad1}(j) \quad for \ j = 0,1 \tag{1690}$$

– The spectral envelope is further adjusted by:

$$\begin{cases} \hat{f}_{env\_WB\_ad3}(j) = att \cdot \hat{f}_{env\_WB\_ad2}(j) & if \ CT != AUDIO \ AND \ first\_CNG = 1 \ AND \ CNG\_mode \geq 2 \\ \hat{f}_{env\_WB\_ad3}(j) = \hat{f}_{env\_WB\_ad2}(j) & otherwise \end{cases} \tag{1691}$$

where, att is the attenuation factor. It is initialized to 1 and set to $att = BWE\_ATT[CNG\_mode - 2]$ if $first\_CNG = 1$ and $CNG\_mode \geq 2$ are satisfied. $BWE\_ATT(j)$, $j = 0,1,2$ are defined in table 164.

Table 164: The envelope attenuation factor

j	$BWE\_ATT(j)$
0	0.8000
1	0.7746
2	0.7483

– Then the adjusted spectral envelopes are applied to the excitation signal by:

$$\begin{cases} \hat{X}^{HB}_{M\_WB}(k) = \hat{X}_{M\_exc\_WB}(k) \cdot \hat{f}_{env\_WB\_ad3}(0) & for \ 240 \leq k < 280 \\ \hat{X}^{HB}_{M\_WB}(k) = \hat{X}_{M\_exc\_WB}(k) \cdot \hat{f}_{env\_WB\_ad3}(1) & for \ 280 \leq k < 320 \end{cases} \tag{1692}$$

When the conditions: the current frame is NORMAL frame, or the current frame is HARMONIC frame and $\hat{f}_{env\_WB\_ad1}(1) < 0.25 \cdot \hat{f}_{env\_WB\_ad1}(0)$, are not satisfied, the adjustment is as follows.

– The first spectral envelope is firstly processed as follows.

$$\hat{f}_{env\_WB\_ad2}(0) = 0.5 \cdot \left(\hat{f}_{env\_WB\_ad1}(0) + \hat{f}_{env\_WB\_ad1}(1) \right) \tag{1693}$$

– If the conditions: $M_{last\_extl} = WB\_BWE$, $0.5 \cdot E_L^{[-1]} < E_L < 2 \cdot E_L^{[-1]}$, are satisfied, the adjustment is performed by:

$$\hat{f}_{env\_WB\_ad3}(0) = 0.25 \cdot \hat{f}_{env\_WB\_ad2}(0) + 0.375 \cdot \left(f_{env\_WB}^{[-1]}(0) + f_{env\_WB}^{[-1]}(1) \right) \tag{1694}$$

Otherwise, there is no adjustment:

$$\hat{f}_{env\_WB\_ad3}(0) = \hat{f}_{env\_WB\_ad2}(j) \tag{1695}$$

– Then the adjusted spectral envelopes are applied to the excitation signal by:

$$\hat{X}_{M\_WB}^{HB}(k) = \hat{X}_{M\_exc\_WB}(k) \cdot att \cdot \hat{f}_{env\_WB\_ad3}(0) \quad for \ 240 \le k < 320 \tag{1696}$$

where, att is the attenuation factor. It is initialized to 1 and set to $att = BWE\_ATT[CNG\_mode - 2]$ if $first\_CNG = 1$ and $CNG\_mode \ge 2$. $BWE\_ATT(j), j = 0,1,2$ are defined in table 164.

The MDCT coefficients of HB signal are refined by:

$$\hat{X}_{M\_WB}^{HB}(k) = \begin{cases} \hat{X}_{M\_WB}^{HB}(k) \cdot (0.2 + 0.05 \cdot (k - 240)) & for \ 240 \le k < 256 \\ \hat{X}_{M\_WB}^{HB}(k) \cdot (1.0 - 0.02 \cdot (k - 280)) & for \ 280 \le k < 320 \quad if \ F_{core\_type} = 1 \\ \hat{X}_{M\_WB}^{HB}(k) \cdot (1.0 - 0.04 \cdot (k - 300)) & for \ 300 \le k < 320 \quad if \ F_{core\_type} = 0 \end{cases} \tag{1697}$$

And the spectral envelopes of current frame are preserved as $f_{env\_WB}^{[-1]}(j), j = 0,1$ for the next frame.

6.1.5.2.2.6 Windowing and frequency-to-time transformation

A 320-point inverse MDCT is used for WB FD BWE. Refer to subclause 6.2.4.

6.1.5.3 Decoding of upper band at 64 kb/s

The upper band at 64 kbps bit-rate decoding starts with dequantizing the received spectrum coefficients by means of the AVQ as described in subclause 6.1.1.2.1.6.

The spectrum between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB, is reconstructed using decoded part of the upper band spectrum.

Further the spectral envelope is decoded and the quantized spectral envelope (four bands in normal mode or two bands in transient mode) is used to denormalize per envelope the decoded spectrum. Each spectral coefficient in an overlap region (7.6 kHz – 8 kHz) is multiplied by a factor lower than 1.0. The overlap region corresponds to the part of the spectrum where the ACELP lower band synthesis is suppressed due to the attenuation of the resampling filters. Consequently spectral coefficients in an overlap region equalize the spectral gap that would be present if the upper band coding would start at 8 kHz only.

Finally the spectrum is de-normalized by the decoded global gain and transformed to the time domain using iDCT and OLA function.

6.1.5.3.1 Decoding in normal mode

The global gain $\hat{g}_{global}$ and the spectral envelope $\hat{f}_{env\_band}(j)$, $0 \le j < 4$ corresponding to 4 sub-bands are decoded. The global gain $\hat{g}_{global}$ is de-quantized using a 5-bit log gain de-quantizer at the range of [3.0; 500.0]. The spectral envelopes $\hat{f}_{env\_band}(j)$, $0 \le j < 4$ are de-quantized using two-dimensional VQs by means of 6bits and 5bits respectively as defined in subclause 5.2.6.3.1.

Then the band index with the minimum envelope pos_{en_min} is calculated by $pos_{en\_min} = \arg\min_{j=0,\ldots,3}(\hat{f}_{env\_band}(j))$, and the envelope of the spectrum between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB is predicted as follows:

1) If *if* $M_{extl} = SWB\_BW\_HIGHRATE$, $\hat{f}_{env\_non\_dec} = 0.5 \cdot \hat{f}_{env\_band}(pos_{en\_min})$.

2) If $M_{extl} = FB\_BW\_HIGHRATE$, the index of attenuation factor $\hat{I}_{att}$ is decoded. Then, the $\hat{f}_{env\_non\_dec}$ is adjusted depending upon the index:

$$\hat{f}_{env\_non\_dec} = \begin{cases} \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 0 \\ 0.5 \cdot \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 1 \\ 2.4 \cdot \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 2 \\ 1.6 \cdot \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 3 \end{cases} \tag{1698}$$

The number of the sub-bands N_{sv1} is obtained according to the number of the total bits R'_{tot} and the saturated threshold $Thr1 = 400$ as follows:

$$N_{sv1} = \begin{cases} 34 & if \ R'_{tot} > 400 \\ \left(208 - 8\left(pos_{en\_min}\%2\right)\right)/8 & otherwise \end{cases} \tag{1699}$$

The first stage sub-vectors $\hat{X}_{M\_norm1}(k), \ 0 \le k < 8 \cdot N_{sv1}$ are decoded by means of AVQ and the spectrum of the upper band is obtained by the first stage sub-vectors $\hat{X}_{M\_norm1}(k), \ 0 \le k < 8 \cdot N_{sv1}$,

$$\hat{X}_{M\_tmp}(k) = \hat{X}_{M\_norm1}(k), \ 0 \le k < 8 \cdot N_{sv1} \tag{1700}$$

and the $nq(b)$ is saved to $nq\_tmp(b)$, i.e. $nq\_tmp(b) = nq(b), \ 0 \le b < N_{sv1}$.

If the number of the remaining bits R_{rem} after the first stage decoding is larger than 14 and the first stage quantized spectrum is non-zero, then the second stage decoding is performed. The second stage global gain $\hat{g}_{global2}$, is decoded and updated by:

$$\hat{g}_{global2} = \hat{g}_{global2} * 0.0625 \tag{1701}$$

The number of the sub-bands N_{sv2} is obtained according to the remaining bits R_{rem} and the saturated threshold $Thr2 = 12$ as follows:

$$N_{sv2} = \begin{cases} 33 & if \ R_{rem} > 396 \\ R_{rem}/12 & otherwise \end{cases} \tag{1702}$$

The second stage sub-vectors $\hat{X}_{M\_norm2}(k), \ 0 \le k < 8 \cdot N_{sv2}$ are also decoded by means of AVQ.

Then, the spectrum of the upper band is reconstructed by the contribution of the second stage decoding as follows:

– The counter i is initialized to 0.

– In the sub-band b , $0 \le b < N_{sv1}$, if the first stage AVQ codebook index $nq(b) = 0 \ AND \ i < N_{sv2}$, the spectrum of the upper band is adjusted by $\hat{X}_{M\_tmp}(8 \cdot b + k) = \hat{g}_{global2} \cdot \hat{X}_{M\_norm2}(8 * i + k), \ 0 \le k < 8$, and the second stage AVQ codebook index $nq_2(i)$ is added to the first stage AVQ codebook index $nq(b) = nq(b) + nq_2(i)$. Then, the counter i are incremented by 1.

– The sub-band index b_1 is initialized to 0, while $i < N_{sv2}$, if $nq\_tmp(b_1) != 0$, the spectrum is adjusted by

$$\hat{X}_{M\_tmp}(8 \cdot b_1 + k) = \hat{X}_{M\_tmp}(8 \cdot b_1 + k) + \hat{g}_{global2} \cdot \hat{X}_{M\_norm2}(8 \cdot i + k), \ 0 \le k < 8, \text{ and } nq(b_1) = nq(b_1) + nq_2(i).$$

Then, the counter i is incremented by 1.

The spectrum is then reordered according to the number of the total bits R'_{tot}.

– If $R'_{tot} > 400$, the reordered spectrum is obtained by:

$$\hat{X}_{M\_spec}(k_{start} + k) = \hat{X}_{M\_temp}(k) \qquad 0 \le k < 272 \tag{1703}$$

where k_{start} is the start frequency bin of spectrum reconstruction, and $k_{start} = 304$ for normal frames.

– Otherwise, the reordered spectrum is obtained by:

$$\begin{cases} \hat{X}_{M\_spec}(k_{start} + k) = \hat{X}_{M\_temp}(k) & 0 \le k < k_{ind1} \\ \hat{X}_{M\_spec}(k_{start} + k_{ind2} + k) = \hat{X}_{M\_temp}(k + k_{ind1}) & 0 \le k < 272 - k_{ind2} \end{cases} \tag{1704}$$

where the k_{ind1} and k_{ind2} are set as follows:

$$k_{ind1} = 64 \cdot pos_{en\_min} + 8 \cdot \lfloor pos_{en\_min} / 2 \rfloor \tag{1705}$$

$$k_{ind2} = 64 \cdot (pos_{en\_min} + 1) + 8 \cdot \lfloor (pos_{en\_min} + 1) / 2 \rfloor \tag{1706}$$

And then the spectrum of the band with the minimum envelope is reconstructed as follows:

$$\begin{cases} \hat{X}_{M\_spec}(k_{start} + 200 + k) = \hat{X}_{M\_spec}(k_{start} + 128 + k) & 0 \le k < 72 & if \ pos_{en\_min} = 3 \\ \hat{X}_{M\_spec}(k_{start} + k_{ind1} + k) = \hat{X}_{M\_spec}(k_{start} + k_{ind2} + k) & otherwise \\ \qquad\qquad 0 \le k < 64 + 8 \cdot (pos_{en\_min} \% 2) \end{cases} \tag{1707}$$

Meanwhile, the AVQ codebook index $nq(b)$ is adjusted as follows:
If $pos_{en\_min} = 3$,

$$nq(b + 25) = nq(b + 16) \quad 0 \le b < 9 \tag{1708}$$

Otherwise,

$$nq(33 - (b - k_{ind1})) = nq(b), \qquad k_{ind1} \le b < N_{sv1} \tag{1709}$$

And then,

$$nq(k_{ind1} + b) = nq(k_{ind2} + b) \quad 0 \le b < 8 + pos_{en\_min} \% 2 \tag{1710}$$

Then the noise filling is applied to the spectrum. If the remaining bits after the above decoding $R_{rem1} < 200$, the 272 MDCT coefficients of the upper band $\hat{X}_{M\_spec}(k)$, $k = 304,...,575$ are divided into 34 sub-bands (8 coefficients per band). Two indices pos_{start}, pos_{end} are introduced to select a base frequency band which is used to reconstruct the un-decoded coefficients in the sub-bands $nq[b] = 0$.

– The sub-band index i is initialized to 0.

– If $nq[0] = 0$, in the index range from 0 to 34, the index pos_{start} of the first sub-band which AVQ codebook index $nq(pos_{start}) != 0$ is searched, and then in the index range from the index pos_{start} to 34, the index pos_{end} of the first sub-band which AVQ codebook index $nq(pos_{end}) = 0$ is searched, and the sub-band index is set by $i = pos_{end}$, and $pos_{end} = pos_{end} - 1$. If $pos_{end} - pos_{start} > pos_{start}$, the pos_{end} is adjusted by $pos_{end} = 2 \cdot pos_{start} - 1$, and then the MDCT coefficients in the sub-bands b, $0 \le b < pos_{start}$ are reconstructed by:

$$\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)=\alpha\cdot\hat{X}_{M\_spec}(k_{start}+8\cdot pos+k)+\beta\cdot s_{noise}(8\cdot b+k)/32768 \qquad 0\le k<8 \text{ (1711)}$$

where, α and β are the weighted factors, and $\alpha=0.25$, β is obtained by:

$$\beta=\begin{cases}0.25 & if\ \ tilt\_wb>5.0\\0.25 & else\ \ if\ \ tilt\_wb\le5.0\ AND\ 100>0.25\cdot T_{pitch}\\100/T_{pitch} & otherwise\end{cases} \tag{1712}$$

$$T_{pitch}=\sum_{i=0}^{4}pitch(i)+\varepsilon \tag{1713}$$

the index of sub-band b is from $pos_{start}-1$ to 0, and pos is initialized to pos_{end} and updated by:

$$pos=\begin{cases}pos_{end} & if\ \ pos<pos_{start}\\pos-1 & otherwise\end{cases} \tag{1714}$$

1. If $nq[i]!=0$, $i\le34$, the index i is incremented by 1; Otherwise, a base frequency band is selected to reconstruct the un-decoded coefficients as follows:

2. Initialize the indices $pos_{start}=i$ and $pos_{end}=i$, and then in the sub-band b, $pos_{start}\le b<34$, search the index pos_{end} of the first sub-band which AVQ codebook index $nq(pos_{end})!=0$, and set $i=pos_{end}$. If $pos_{start}=pos_{end}$, the index and the position are further adjusted: $i=34$, $pos_{end}=34$.

3. Then the MDCT coefficients of the upper band are adjusted by:

$$\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)=\alpha\cdot\hat{X}_{M\_spec}(k_{start}+8\cdot pos_2+k)+\beta\cdot s_{noise}(k)/32768 \quad 0\le k<8 \tag{1715}$$

where b is from $pos_{end}-1$ to pos_{start}, and the pos_2 is initialized to $pos_{start}-1$ and it is adjusted by:

$$pos_2=\begin{cases}pos_{start}-1 & if\ \ pos_2=0\\pos_2-1 & otherwise\end{cases} \tag{1716}$$

and,

$$s_{noise}(k)=\lambda_{seed} \quad and \quad \lambda_{seed}=31821\cdot\lambda_{seed}+13849 \qquad 0\le k<8 \tag{1717}$$

Parameter λ_{seed} is initialized as 12345 and updated for each MDCT coefficient. It should be noted that $s_{noise}(k)$ is calculated for every frame.

4. Finally, judge whether $i<34$ or not. If $i<34$, return to step 1.

If the spectral tilt $tilt\_wb$ of the decoded core signal $\hat{s}_{16}(n)$ is larger than 5, the MDCT coefficients of the upper band are further adjusted. In sub-band $b,0\le b<34$, if $\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)=0$, the adjustment is performed as follows:

$$\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)=\begin{cases}0.5\cdot E_{min} & if\ \ s_{noise}(k)>0\\-0.5\cdot E_{min} & otherwise\end{cases} \quad 0\le k<8 \tag{1718}$$

where $tilt\_wb$ is obtained by the algorithm described in equation (629) and (630), and E_{min} is obtained as follows:

$$E_{min}=\min_{k=0,...,7}\left(\left|\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)\right|\right) \quad if\ \left|\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)\right|>0 \tag{1719}$$

and if $E_{min}=\max_{k=0,...,7}\left(\left|\hat{X}_{M\_spec}(k_{start}+8\cdot b+k)\right|\right)AND\ E_{min}>1.0$, E_{min} is refined by:

$$E_{\min} = 0.5 \cdot E_{\min} \tag{1720}$$

When $pos_{en\_\min} = 3$ *AND* $R'_{tot} \leq 400$, the coefficients of the upper band in the index range [504, 511] are smoothed by:

$$\hat{X}_{M\_spec}(k_{start} + 200 + k) = \hat{X}_{M\_spec}(k_{start} + 200 + k) \cdot \left(\left(1 - \frac{k}{8}\right) \cdot R_1 + \frac{k}{8} \right) \quad k = 0,...,7 \tag{1721}$$

where R_1 is defined as follows:

$$R_1 = \sqrt{\frac{\sum_{k=-8}^{-1} \left(\hat{X}_{M\_spec}(k_{start} + 200 + k)\right)^2 + \varepsilon}{\sqrt{\sum_{k=0}^{7} \left(\hat{X}_{M\_spec}(k_{start} + 200 + k)\right)^2 + \varepsilon}}} \tag{1722}$$

The spectrum between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB is reconstructed by:

$$\hat{X}_{M\_spec}(576 + k) = \hat{X}_{M\_spec}(576 - L_{non\_dec} + k) \quad k = 0,...,L_{non\_dec} - 1 \tag{1723}$$

where L_{non_dec} is the number of the coefficients between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB:

$$L_{non\_dec} = \begin{cases} 64 & \text{for SWB signal} \\ 224 & \text{for FB signal} \end{cases} \tag{1724}$$

Then *if* $M_{extl} = SWB\_BW\_HIGHRATE$ or $\max\left(\hat{f}_{env\_band}(j)\right) / \min\left(\hat{f}_{env\_band}(j)\right) > 2.2$, $0 \leq j < 4$, the coefficients in the index range [576, 583] are smoothed by,

$$\hat{X}_{M\_spec}(576 + k) = \hat{X}_{M\_spec}(576 + 200 + k) \cdot R_2 \quad k = 0,...,7 \tag{1725}$$

where R_2 is defined as follows:

$$R_2 = \sqrt{\frac{\sum_{k=-8}^{-1} \left(\hat{X}_{M\_spec}(576 + k)\right)^2 + \varepsilon}{\sqrt{\sum_{k=0}^{7} \left(\hat{X}_{M\_spec}(576 + k)\right)^2 + \varepsilon}}} \tag{1726}$$

and the spectrum is de-normalized using the spectral envelope by:

$$\begin{cases} \hat{X}_{M\_denorm}(k_{start} + k) = \hat{X}_{M\_spec}(k_{start} + k) \cdot \hat{f}_{env\_band}(j) & 0 \leq j < 4,\ 68 \cdot j \leq k < 68 \cdot (j+1) \\ \hat{X}_{M\_denorm}(576 + k) = \hat{X}_{M\_spec}(576 + k) \cdot \left(\left(1 - \frac{k}{8}\right) \cdot \hat{f}_{env\_band}(3) + \frac{k}{8} \cdot \hat{f}_{env\_non\_dec} \right) & 0 \leq k < 8 \\ \hat{X}_{M\_denorm}(576 + k) = \hat{X}_{M\_spec}(576 + k) \cdot \hat{f}_{env\_non\_dec} & 8 \leq k < L_{non\_dec} \end{cases} \tag{1727}$$

Otherwise the spectrum is obtained as:

$$\begin{cases} \hat{X}_{M\_denorm}(576 + k) = \\ \qquad \hat{X}_{M\_spec}(576 + k) \cdot 2.2 \cdot \hat{f}_{env\_non\_dec} \cdot (1 - k/160), & 0 \leq k < L_{non\_dec1} \\ \hat{X}_{M\_denorm}(576 + k) = \\ \qquad \hat{X}_{M\_spec}(576 + k) \cdot 0.65 \cdot \hat{f}_{env\_non\_dec} \cdot \left(1 - (k - L_{non\_dec1})/320\right), & L_{non\_dec1} \leq k < L_{non\_dec} \end{cases} \tag{1727a}$$

where $L_{non\_dec1} = 64$ for 32 kHz sampled output or $L_{non\_dec1} = 88$ for 48 kHz smapled output. Next the overlap coefficients $f_{overlap\_coefs}(k), 0 \le k < 16$, which depend on the output sampling rate (32 kHz or 48 kHz), defined in table165 are used for adjustment as follows:

$$\hat{X}_{M\_denorm}(k_{start} + k) = f_{overlap\_coefs}(k) \cdot \hat{X}_{M\_denorm}(k_{start} + k) \qquad 0 \le k < 16 \qquad (1728)$$

Table 165: Overlap coefficients $f_{overlap\_coefs}(k)$

k	0	1	2	3	4	5	6	7
$f_{overlap\_coefs}(k)$, 32 kHz	0.27	0.306	0.324	0.351	0.378	0.396	0.414	0.4275
$f_{overlap\_coefs}(k)$, 48 kHz	0.30	0.34	0.36	0.39	0.42	0.44	0.46	0.475
k	8	9	10	11	12	13	14	15
$f_{overlap\_coefs}(k)$, 32 kHz	0.441	0.459	0.486	0.513	0.558	0.648	0.747	0.855
$f_{overlap\_coefs}(k)$, 48 kHz	0.49	0.51	0.54	0.57	0.62	0.72	0.83	0.95

Finally the decoded global gain $\hat{g}_{global}$ is applied to adjust the spectrum of the upper band.

$$\hat{X}_M(k_{start} + k) = \begin{cases} 0.85 \cdot \hat{g}_{global} \cdot \hat{X}_{M\_denorm}(k_{start} + k) & if \ R'_{tot} \le 400 \\ \hat{g}_{global} \cdot \hat{X}_{M\_denorm}(k_{start} + k) & otherwise \end{cases} \qquad 0 \le k < 272 + L_{non\_dec} \quad (1729)$$

The MDCT coefficients are preserved for the next frame as follows:

$$\begin{cases} \hat{X}_M^{[-1]}(k) = \hat{X}_M(k_{start} + k) & 0 \le k < 336 \qquad for \ SWB \ signal \\ \hat{X}_M^{[-1]}(k) = \hat{X}_M(k_{start} + k) & 0 \le k < 496 \qquad for \ FB \ signal \end{cases} \qquad (1730)$$

Two parameters of $E_{ener\_shb}^{[-1]}$ and $\hat{f}_{env\_SWB}^{[-1]}(j), 0 \le j < 14$ are updated by:

$$E_{ener\_shb}^{[-1]} = \hat{g}_{global} \cdot 0.25 \cdot \sum_{j=0}^{4} \hat{f}_{env\_band}(j) \qquad (1731)$$

$$\hat{f}_{env\_SWB}^{[-1]}(j) = \hat{g}_{global} \cdot \hat{f}_{env} \quad 0 \le j < 14 \qquad (1732)$$

6.1.5.3.2 Decoding in transient mode

There are 4 sub-fames for transient mode. In sub-frame j, $0 \le j < 4$, the global gain $\hat{g}_{global}$ and spectral envelopes $\hat{f}_{env\_band}(i)$, $i = 0,1$, are decoded. The global gain of each sub-frame $\hat{g}_{global}$ is de-quantized using a 5-bit log gain de-quantizer at the range of [3.0; 500.0]. The first sub-frame spectral envelopes $\hat{f}_{env\_band}(i)$, $i = 0,1$ are de-quantized firstly using two-dimensional VQs by means of 4 bits codebook defined in subclause 5.2.6.3.2. The indices of the first sub-frame spectral envelopes is noted as $\hat{I}_{env}(0)$.

If $\hat{I}_{env}(0) < 8$, which means the first sub-frame spectral envelopes are de-quantized in the first part of the 4 bits codebook, the spectral envelope $\hat{f}_{env\_band}(i)$, $i = 0,1$ of the following three sub-frame are de-quantized using two-dimensional VQs by means of 3 bits codebook, i.e. $0 \le \hat{I}_{env}(j) < 8$, $1 \le j < 4$ as described in subclause 5.2.6.3.2.

If $\hat{I}_{env}(0) \geq 8$, the spectral envelope $\hat{f}_{env\_band}(i)$, $i = 0,1$ of the following three sub-frame are de-quantized using two-dimensional VQs by means of 3 bits codebook, i.e. $8 \leq \hat{I}_{env}(j) < 15$, $1 \leq j < 4$ as described in subclause 5.2.6.3.2.

And then the spectrum of the upper band is reconstructed.

The number of the coefficients between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB, L_{non_dec} , is set by:

$$L_{non\_dec} = \begin{cases} 16 & for \ SWB \ signal \\ 56 & for \ FB \ signal \end{cases} \qquad (1733)$$

And the envelope of the MDCT coefficients between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB $\hat{f}_{env\_non\_dec}$ is calculated by:

1) If $M_{extl} = SWB\_BWE\_HIGHRATE$

$$\hat{f}_{env\_non\_dec} = \hat{f}_{env\_band}(1) \qquad (1734)$$

2) If $M_{extl} = FB\_BWE\_HIGHRATE$, the index of attention factor $\hat{I}_{att}$ is decoded, and then the $\hat{f}_{env\_non\_dec}$ is adjusted depending upon the index $\hat{I}_{att}$.

$$\hat{f}_{env\_non\_dec} = \begin{cases} \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 0 \\ 0.1 \cdot \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 1 \\ 0.3 \cdot \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 2 \\ 0.5 \cdot \hat{f}_{env\_non\_dec} & if \ \hat{I}_{att} = 3 \end{cases} \qquad (1735)$$

The normalized spectrum of the upper band is decoded by the sub-vectors $\hat{X}_{M\_norm}(k)$ which are obtained by means of the AVQ:

$$\hat{X}_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) = \hat{X}_{M\_norm}(k) \quad 0 \leq k < 64 \qquad (1736)$$

where k_{start} is the start frequency bin of spectrum reconstruction, and $k_{start} = 64$ for transient frames. L_{fr} is the frame length. $L_{fr} = 640$ for SWB signal and $L_{fr} = 960$ for FB signal.

Then the noise filling is applied to the spectrum. The first 64 MDCT coefficients of the upper band $\hat{X}_{M\_spec}(k)$, $k = 76,...,139$ are divided into 8 sub-bands (8 coefficients per band). In the sub-band b, $0 \leq b < 8$, if the AVQ codebook index is equal to 0, that is, $nq(b) = 0$, the signal is generated from linear congruential uniform random noise generator as follows:

$$\hat{X}_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) = s_{noise}(k)/65536 \qquad 8 \cdot b \leq k < 8 \cdot (b+1) \qquad (1737)$$

where

$$s_{noise}(k) = \lambda_{seed} \quad and \quad \lambda_{seed} = 31821 \cdot \lambda_{seed} + 13849 \qquad 8 \cdot b \leq k < 8 \cdot (b+1) \qquad (1738)$$

Parameter λ_{seed} is initialized as 12345 and updated for each MDCT coefficient. It should be noted that $s_{noise}(k)$ is calculated for every frame.

If the spectral tilt $tilt\_wb$ of the decoded core signal $\hat{s}_{16}(n)$ is larger than 5, the MDCT coefficients of the upper band are further adjusted. If $\hat{X}_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) = 0$, the adjustment is by:

$$\hat{X}_{M\_spec}\left(k_{start} + j \cdot L_{fr}/4 + k\right) = \begin{cases} 0.5 \cdot E_{\min} & if \ \ s_{noise}(k) > 0 \\ -0.5 \cdot E_{\min} & otherwise \end{cases} \quad 8 \cdot b \leq k < 8 \cdot (b+1) \quad (1739)$$

where *tilt_wb* is obtained by the algorithm described in equations (629) and (630), and $E_{\min}$ is obtained as follows:

$$E_{\min} = \min_{k=8b,\dots,8b+7}\left(\left|\hat{X}_{M\_spec}\left(k_{start} + j \cdot L_{fr}/4 + k\right)\right|\right) \quad if \ \left|\hat{X}_M\left(k_{start} + j \cdot L_{fr}/4 + k\right)\right| > 0 \quad (1740)$$

and if $E_{\min} = \max\limits_{k=8b,\dots,8b+7}\left(\left|\hat{X}_{M\_spec}\left(k_{start} + j \cdot L_{fr}/4 + k\right)\right|\right) AND \ E_{\min} > 1.0$, $E_{\min}$ is refined by:

$$E_{\min} = 0.5 \cdot E_{\min} \quad (1741)$$

The spectrum between 14.4 kHz and 16 kHz in SWB, resp. 20 kHz in FB is reconstructed by replicating the nearby coefficients:

$$\hat{X}_{M\_spec}(140 + j \cdot L_{fr}/4 + k) = 0.5 \cdot \hat{X}_M(140 + j \cdot L_{fr}/4 - (L_{non\_dec} + 4) + k) \quad 0 \leq k < L_{non\_dec} + 4 \quad (1742)$$

Then the spectrum of the upper band is de-normalized using the spectral envelope by:

$$\begin{cases} \hat{X}_{M\_denorm}(k_{start} + j \cdot L_{fr}/4 + k) = \hat{X}_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) \cdot \hat{f}_{env\_band}(0) & 0 \leq k < 34 \\ \hat{X}_{M\_denorm}(k_{start} + j \cdot L_{fr}/4 + k) = \hat{X}_{M\_spec}(k_{start} + j \cdot L_{fr}/4 + k) \cdot \hat{f}_{env\_band}(1) & 34 \leq k < 68 \\ \hat{X}_{M\_denorm}(140 + j \cdot L_{fr}/4 + k) = \hat{X}_{M\_spec}(140 + j \cdot L_{fr}/4 + k) \cdot \hat{f}_{env\_non\_dec} & 4 \leq k < L_{non\_dec} + 4 \end{cases} \quad (1743)$$

and the modification factors $f_{overlap\_coefs}(k), 0 \leq k < 16$ defined in table 165 are used to adjust the overlapped coefficients as follows:

$$\hat{X}_{M\_denorm}(k_{start} + j \cdot L_{fr}/4 + k) = 0.9 \cdot \hat{X}_{M\_denorm}(k_{start} + j \cdot L_{fr}/4 + k) \cdot f_{overlap\_coefs}(4 \cdot k) \quad (1744)$$

where, $0 \leq k < 4$.

Finally, the decoded global gain $\hat{g}_{global}$ is applied to adjust the spectrum of the upper band.

$$\hat{X}_M(k_{start} + j \cdot L_{fr}/4 + k) = \hat{g}_{global} \cdot \hat{X}_{M\_denorm}(k_{start} + j \cdot L_{fr}/4 + k) \quad 0 \leq k < 68 + L_{non\_dec} \quad (1745)$$

6.1.5.3.3 Windowing and frequency-to-time transformation

A 640-point (SWB) or 960-point (FB) inverse MDCT is used for the upper band frequency signal. Refer to subclause 6.2.4.

6.1.5.3.4 Post-processing in temporal domain

If the current frame is a good one, further adjustment of the upper band synthesized temporal signal $\hat{s}_{hb\_syn}(n)$ is needed. The synthesized temporal signal of the upper band $\hat{s}_{hb\_syn}(n)$ is divided into 4 sub-frames, and the energy of each sub-frame $E^{[m]}$, is computed by:

$$E^{[m]} = \sum_{n=m \cdot L_{fr}/4}^{(m+1) \cdot L_{fr}/4 - 1}\left(\hat{s}_{hb\_syn}(n)\right)^2 \qquad for \ m = 0,\dots,3 \quad (1746)$$

For each sub-frame, the long term energy $E_{LT}^{[m]}$ is initialized to 0, and updated according to the following equation:

$$E_{LT}^{[m]} = (1-\alpha)E_{LT}^{[m-1]} + \alpha E^{[m]} \qquad for \ m = 0,\dots,3 \quad (1747)$$

In the above equation, the weighted factor α is set to 0.25, and the convention is that for the first sub-frame,

$E_{LT}^{[-1]} = E_{LT}^{[3]}$ from the previous frame. For each sub-frame m, a comparison between the short term energy $E^{[m]}$ and the long term energy $E_{LT}^{[m]}$ is performed. A transient is detected whenever the energy ratio is above a certain threshold and the sub-frame index m is recorded as pos. Formally, a transient is detected whenever:

$$E^{[m]} > \rho E_{LT}^{[m]} \tag{1748}$$

where ρ is the energy ratio threshold and is set to $\rho = 12.5$.

When the current frame is transient and the conditions: $pos > 0$, $tilt\_wb < 3$, $\sum_{j=0}^{4} pitch(j) + \varepsilon > 500$, are all satisfied, the adjustment is processed.

$E_{LT}^{[-1]}$ is obtained depending upon whether the current and previous frames apply the same extend layer:

$$E_{LT}^{[-1]} = \begin{cases} E & \text{if } M_{last\_extl} != M_{extl} \\ E_{LT}^{[-1]} & \text{otherwise} \end{cases} \tag{1749}$$

where the energy E is calculated by:

$$N = pos \cdot L_{fr} / 4 \tag{1750}$$

$$E = \sum_{n=0}^{N} \left(\hat{s}_{hb\_syn}(n) \right)^2 \tag{1751}$$

then,

$$\hat{s}'_{hb\_syn}(n) = \begin{cases} \hat{s}_{hb\_syn}(n) \cdot 0.2 \cdot \sqrt{pos \cdot \dfrac{E_{LT}^{[-1]}}{E}} & 0 \le n < N \\ \hat{s}_{hb\_syn}(n) \cdot \left(\left(1 - \dfrac{8 \cdot (n-N)}{L_{fr}} \right) \cdot 0.2 \cdot \sqrt{pos \cdot \dfrac{E_{LT}^{[-1]}}{E}} + \dfrac{8 \cdot (n-N)}{L_{fr}} \right) & N \le n < N + L_{fr}/8 \end{cases} \tag{1752}$$

The signal class F_{class} of the current frame is preserved as $F_{class}^{[-1]}$ for the next frame.

When $last\_core = HQ\_CORE$ or $M_{last\_extl} != M_{extl}$, if $tilt\_wb < 3$, the post-processing is performed in case of switching of different extend layers or different cores. The gain factor g_{post_pro} is first calculated by:

$$g_{post\_pro} = \begin{cases} \sqrt{\dfrac{1}{80} \sum_{n=L_{fr}-80}^{L_{fr}-1} \left(\hat{s}'_{hb\_syn}(n) \right)^2} & \text{if } pos_{max} < 160 \\ \sqrt{\dfrac{1}{80} \sum_{n=0}^{79} \left(\hat{s}'_{hb\_syn}(n) \right)^2} & \text{otherwise} \end{cases} \tag{1753}$$

where pos_{max} is the index of the time sample with the maximum magnitude, and

$$pos_{max} = \arg \max_{n=0,...,Lfr-1} \left(\left| \hat{s}'_{hb\_syn}(n) \right| \right) \tag{1754}$$

The gain factor g_{post_pro} is refined by:

$$g_{post\_pro} = \min(1, \frac{g_{post\_pro}}{\sqrt{\frac{1}{k_{ind2} - k_{ind1}} \sum_{n=k_{ind1}}^{k_{ind2}-1} \left(\hat{s}'_{hb\_syn}(n)\right)^2 + \varepsilon}}) \tag{1755}$$

where k_{ind1} and k_{ind2} are obtained by:

$$k_{ind1} = \max(0, pos_{max} - 40) \tag{1756}$$

$$k_{ind2} = \min(L_{fr}, pos_{max} + 40) \tag{1757}$$

Then the synthesized signal of the upper band is adjusted by:

$$\hat{s}''_{hb\_syn}(n) =$$

$$\begin{cases} \hat{s}'_{hb\_syn}(n) \cdot g_{post\_pro} & k_{ind1} \le n < L_{fr} & if\ M_{last\_extl} = SWB\_BWE\ OR\ FB\_BWE \\ \left. \begin{array}{l} \hat{s}'_{hb\_syn}(n) \cdot g_{post\_pro} \quad k_{ind2} \le n < k_{ind2} \\ \hat{s}'_{hb\_syn}(n) \cdot \left(g_{post\_pro} + \beta(n)\right) \ k_{ind2} \le n < L_{fr} \end{array} \right\} & otherwise \end{cases} \tag{1758}$$

where $\beta(n)$ is calculated as follows:

$$\alpha = \begin{cases} 1 & if\ g_{post\_pro} > 0.5 \\ 0.5 & otherwise \end{cases} \tag{1759}$$

$$\beta(n) = (n - k_{ind2}) \cdot \left(\alpha - g_{post\_pro}\right) / (L_{fr} - k_{ind2}) \qquad k_{ind2} \le n < L_{fr} \tag{1760}$$

And the preserved synthesized signal of the upper band for OLA function, $\hat{s}^{[-1]}_{hb\_syn}(n)$, is adjusted by:

$$\hat{s}^{[-1]}_{hb\_syn}(n) = \hat{s}^{[-1]}_{hb\_syn}(n) \cdot g_{post\_pro} \qquad 0 \le n < L_{fr} \tag{1761}$$

6.2 MDCT Coding mode decoding

6.2.1 General MDCT decoding

MDCT based codec frames can be either encoded by the MDCT based TCX or the High Quality Coder. Which mode is actually used, is determined by table 76 in subclause 5.3.1. For those configurations where both modes are switched, a signaling bit determines the MDCT mode.

6.2.2 MDCT based TCX

A general description of the MDCT based TCX coder module can be found in subclause 5.3.3.1. In the following, the configurations are described, the initialization process from the bit stream data and the general decoding process.

6.2.2.1 Rate dependent configurations

The rate dependent configuration of the TCX coder is described for the encoder in subclause 5.3.3.1.2. The configurations are valid for the decoder as well. Depending on the configuration, the initialization order of the modules changes. The following subclause describes the module initialization therefore just on a principle level.

6.2.2.2 Init module parameters

6.2.2.2.1 TCX block configuration

The coding modes TCX20 or TCX10 are signalled within the bit stream. Binary code for the overlap width as defined in subclause 5.3.2.3 is read from the bitstream. Overlap code for the current frame is formed from the short/long transform decision bit and from the binary code for the overlap width as defined in subclause 5.3.2.3. The TCX block is then configured as described in subclause 6.2.4.2.

6.2.2.2.2 LPC parameter

This subclause describes the inverse quantization of LPC.

6.2.2.2.2.1 Low-rate LPC

MDCT based TCX relies on smoothed LPC spectral envelope. Decoding process of LSF is common to that for ACELP except the case for 9.6 kbps (NB/WB/SWB). Decoding process of weighted LSF and conversion of LSF used for 9.6 kbps (NB/WB/SWB) are described in this subclause.

Weighted domain quantized LSF vector $\hat{\mathbf{q}}_\gamma$ is reconstructed with two stage or three stage VQ as described in 5.3.3.2.1. These VQ codebooks are used in two ways. In case of primary decoding, weighted LSF vector $\hat{\mathbf{q}}_{VQ}$ is retrieved from two stage VQ and after adding mean vector $\mathbf{m}$ and the MA predicted contribution vector $\mathbf{q}_{MA}$, reconstructed LSF vector, $\hat{\mathbf{q}}_\gamma (= \hat{\mathbf{q}}_{VQ} + \mathbf{m} + \mathbf{q}_{MA})$ is obtained. $\hat{\mathbf{q}}_\gamma$ is corresponding to the weighted envelope and can be directly used for inverse shaping of MDCT coefficients. The VQ uses 5 bits for 16 samples at the first stage, 4 bits for the lower 6 LSF parameters, and 4 bits for the higher 10 LSF parameters.

In the secondary decoding, weighted LSF vector is reconstructed without adding MA predicted contribution vector. This can inform the shape of envelope for the envelope base arithmetic coding even when the decoder cannot get the information from the previous frame. The conditional third stage VQ with addition of mean vector is applied to the retrieved vector from two stage VQ to get $\hat{\mathbf{q}}_\gamma''$. After reconstruction of LSF vector $\hat{\mathbf{q}}_{VQ}$ with two stage VQ and adding mean vector $\mathbf{m}$, the first and the second lower position $\hat{q}_\gamma'(0)$ and $\hat{q}_\gamma'(1)$ of the reconstructed LSF vector $\hat{\mathbf{q}}_\gamma'(= \mathbf{m} + \hat{\mathbf{q}}_{VQ})$ are checked. If the $\hat{\mathbf{q}}_\gamma'$ is expected to have large spectral distortion from $\hat{\mathbf{q}}_\gamma$, the third stage VQ is applied. The retrieved vector $\hat{\mathbf{q}}_{3VQ}$ at the third stage VQ with 2 bits is applied to modify $\hat{q}_\gamma'(0)$ and $\hat{q}_\gamma'(1)$ to get

$\hat{\mathbf{q}}''_\gamma$ $(= \hat{\mathbf{q}}_{VQ} + \mathbf{m} + \hat{\mathbf{q}}_{3VQ})$, only when, $\hat{q}'_\gamma(0)$ or $(\hat{q}'_\gamma(1) - \hat{q}'_\gamma(0))$ has smaller values than the threshold. Otherwise, the reconstructed LSF up to the second stage VQ, $\hat{\mathbf{q}}'_\gamma$ is used for the final reconstruction LSF, $\hat{\mathbf{q}}''_\gamma$.

For the interpolation of LSF between the LSF in the possible ACELP at the next frame, the reconstructed LSF vector $\hat{\mathbf{q}}_\gamma$ with MA prediction needs to be converted to unweighted domain LSF vector $\hat{\mathbf{q}}_1$. In envelope based arithmetic coding, unweighted LSF vector without depending on MA prediction $\hat{\mathbf{q}}'_1$ is also necessary to estimate the MDCT envelope. These can be achieved by low-complex direct matrix conversion described in 5.3.3.2.1.1

6.2.2.2.2.2 Mid-rate LPC

The inverse quantization for mid-rate bitrates (13.2 till 32 kbps) is the same as for ACELP (with the quantizer being in AUDIO mode). However there is a different interpolation, the same as described in subclause 5.3.3.2.1.2.

6.2.2.2.2.3 High-rate LPC

Inverse quantization of an LPC filter is performed as described in figure 93. The LPC filters are quantized using the line spectral frequency (LSF) representation. A first-stage approximation is computed by inverse Vector Quantization by a simple table look-up with an 8 bit index. An algebraic vector quantized (AVQ) refinement is then calculated as described in 6.1.1.2.1.4. The quantized LSF vector is reconstructed by adding the first-stage approximation and the inverse-weighted AVQ contribution.

Depending on the frame being coded as a single TCX20 frame or subdivided into TCX10/TCX5 sub-frames one or two sets of LPC have to be de-quantized. In case two sets are transmitted the first set is decoded in the same way a single set would be decoded. For the second set an initial bit signals if it has to be decoded depending on the first set or not. If zero, the first stage approximation of the second set has to be decoded with another 8 bit inverse Vector Quantizer. If one, the inverse quantized first set will be used as first stage approximation. The inverse weights in figure 93 are the reciprocal of the weights used in the encoder (see subclause 5.3.3.2.1.3.3).

The AVQ decoder is described in subclause 6.1.1.2.1.4.
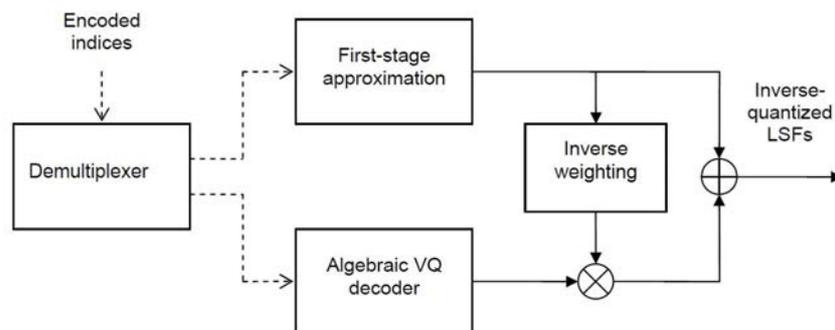
Figure 93: Overview high-rate LPC decoding

The inverse-quantized LSF vector is subsequently converted into a vector of LSF coefficients, then interpolated and converted again into LPC parameters. The interpolation is the same described in 5.3.3.2.1.2.

6.2.2.2.3 PLC Wavefrom adjustment

The waveform adjustment tool reads one bit for initialization where 1 stands for harmonic and 0 for non-harmonic.

6.2.2.2.4 Global Gain

The TCX global gain index $I_{TCX,gain}$ is transmitted in the bitstream as a 7 bit unsigned integer.

6.2.2.2.5   Noise fill parameter

The TCX noise factor index I_{NF} is transmitted in the bitstream as a 3 bit unsigned integer.

6.2.2.2.6   LTP

The LTP on/off flag LTP_{on} is transmitted in the bitstream as a single bit. If the LTP flag is one, the quantized pitch lag and gain are transmitted after the LTP flag. The pitch lag index $I_{LTP,lag}$ is transmitted as unsigned 9 bit integer. The gain index $I_{LTP,gain}$ is transmitted as unsigned 2 bit integer.

6.2.2.2.7   TNS parameter

The TNS on/off flat is transmitted in the bitstream as a single bit. If the TNS flag is one and if the configuration (see subclause 5.3.3.2.2) indicates that 2 filters are possible then an additional single bit transmitted in the bitstream indicates if the parameters for 1 or 2 filters are transmitted in the bitstream (nMaxFilters = 1 or nMaxFilters = 2). For each filter, order and coefficients are the parameters transmitted in the bitstream. The order and the coefficients are coded using Huffman coding. Which Huffman code will be used for the filter order depends on the frame configuration (TCX20/TCX10/TCX5) and on the bandwidth (SWB,WB). Which Huffman code will be used for a parcor coefficient depends on the frame configuration (TCX20/TCX10/TCX5), on the bandwidth (SWB,WB) and on the parcor coefficent's index. If the parameters for 1 filter are transmited in the bitstream, then the second filter is inactive. If the parameters for 2 filters are transmitted in the bitstream, then order 1 and the filter coefficient set to 0 for the first filter indicates that the first filter is disabled and that only the second filter is active.

6.2.2.2.8   Harmonic model

For both context and envelope based arithmetic coding, a harmonic model is used for efficient coding of frames with harmonic content. The harmonic model is disabled if any of the following conditions apply:
- The bit-rate is not one of 9.6, 13.2, 16.4, 24.4, 32, 48 kbps.
- The previous frame was coded by ACELP.
- Envelope based arithmetic coding is used and the coder type is neither Voiced nor Generic.
In the above cases, no further signalling is used.

Otherwise, a single-bit harmonic model flag is read from the bit-stream. When the flag is non-zero, the decoding proceeds by reading the harmonic model interval parameter as follows.

6.2.2.2.8.1   Decoding of Interval of harmonics

When pitch lag and gain are used for the LTP post processing, the lag parameter is utilized for representing the interval of harmonics in the frequency domain. Otherwise, normal representation of interval is applied.

6.2.2.2.8.1.1   Decoding interval depending on time domain pitch lag

According to the procedure in subclause 5.3.3.2.8.1.8.1.1, T_{UNIT}, $Index_T$, $Index_{Bandwidth}$ are set up, $Index_{MUL}$ is read from the bit-stream, and finally T_{MDCT} is calculated.

6.2.2.2.8.1.2   Decoding interval without depending on time domain pitch lag

When pitch lag and gain in the time domain is not used or the pitch gain is less than or equals to 0.46, normal decoding of the interval with un-equal resolution is used. The 8-bit $index$ is read from the bit-stream and T_{UNIT} and T_{MDCT} are calculated as in subclause 5.3.3.2.8.1.8.1.1.

6.2.2.2.8.2   Decoding of gain

In case of envelope based arithmetic coding and Voiced coder type, a 2-bit gain index is read from the bit-stream.

6.2.2.2.9   IGF bit stream reader

On the decoder side the IGF scale factors, the IGF whitening levels and the IGF temporal flatness indicator flag are extracted from the bit stream and subsequently decoded. The decoding of the IGF scale factors is described in subclause 6.2.2.2.9.2.

6.2.2.2.9.1 IGF whitening level decoding

The IGF whitening levels are decoded according to the following pseudo code:

```
nT    = nT ;
for (k = 0; k < nT; k++) {
   currWLevel(k) = 0;
}

tmp = -1;
if (isIndep) {
  tmp = 0;
} else {
  tmp = read_bit();
}
if (tmp == 1) {
  for (k = 0; k < nT; k++) {
     currWLevel(k) = prevWLevel(k);
  }
} else {
  k = 0;
   currWLevel(k) = decode_whitening_level(k);
  tmp = read_bit();
  if (tmp == 1) {
    for (k = 1; k < nT; k++) {
       currWLevel(k) = decode_whitening_level(k);
    }
  } else {
    for (k = 1; k < nT; k++) {
       currWLevel(k) = currWLevel(0);
    }
  }
}
for (k = 0; k < nT; k++) {
   prevWLevel(k) = currWLevel(k);
}
```

wherein the vector *prevWLevel* contains the whitening levels from the previous frame and the function
`decode_whitening_level` takes care of the decoding the actual whitening level from the bit stream. The function
is implemented according to the pseudo code below:

```
tmp = read_bit();
if (tmp == 1) {
  tmp = read_bit();
  if (tmp == 1) {
    return 2;
  } else {
    return 0;
  }
} else {
  return 1;
}
```

Finally the IGF temporal flatness indicator flag is extracted from the bit stream.

In case of a TCX10 frame (*isTCX10 = true*), the IGF sideinfo for both sub-frames is extracted from the bit stream prior
to the IGF processing. Therefore, the decoded sideinfo for the individual sub-frames is stored in a temporary buffer, so
that the sideinfo for the current sub-frame under IGF processing can be accessed via the temporary buffer.

6.2.2.2.9.2 IGF noiseless decoding of scale factors

The noiseless decoding of the IGF scale factor vector g is very similar to the noiseless encoding part. The entire
encoding and decoding procedures are highly symmetric, and therefore the decoding procedure can be uniquely and
unambiguously derived from the encoding procedure.

The module uses the common raw arithmetic decoder functions from the infrastructure, which are available from the
core coder. The functions used are $ari\_decode\_14bits\_bit\_ext(returnValue)$, which decodes one bit into

$returnValue$, $ari\_decode\_14bits\_s27\_ext(returnValue, cumulativeFrequencyTable)$, which decodes one value into $returnValue$ from an alphabet of 27 symbols ($SYMBOLS\_IN\_TABLE$) using the cumulative frequency table $cumulativeFrequencyTable$, $ari\_start\_decoding\_14bits()$, which initializes the arithmetic decoder. Note that there is no $ari\_finish\_decoding\_14bits()$ to finalize the arithmetic decoder. Instead, the equivalent function $arith\_flush\_decoding()$ was locally defined, which returns the last 14 bits read back to the bit stream reader.

6.2.2.2.9.2.1 IGF independency flag

The behaviour and processing related to the $isIndepFlag$ flag is identical to the encoder side.

6.2.2.2.9.2.2 IGF all-Zero flag

The $allZero$ flag is read from the bit stream first. In case the flag is 1, the decoder state is reset and no further data is read from the bit stream, because the decoded scale factors are all set to zero:

$$g(k) := 0, \text{for all } 0 \le k < nB \tag{1762}$$

Otherwise, if the flag is 0, the arithmetic coded scale factor vector g is decoded from the bit stream.

6.2.2.2.9.2.3 IGF arithmetic decoding helper functions

6.2.2.2.9.2.3.1 The reset function

The behaviour and processing related to the reset function is identical to the encoder side.

6.2.2.2.9.2.3.2 The arith_decode_bits function

The $arith\_decode\_bits$ function decodes an unsigned integer of length $nBits$ bits, by reading one bit at a time.

```
arith_decode_bits(nBits)
{
  x = 0;
  for (i = 0; i < nBits; ++i) {
    ari_decode_14bits_bit_ext(&bit);
    x = (x << 1) | bit;
  }

  return x;
}
```

6.2.2.2.9.2.4 IGF arithmetic decoding

The $arith\_decode\_residual$ function decodes an integer valued prediction residual, using the cumulative frequency table $cumulativeFrequencyTable$, and the table offset $tableOffset$.

The behaviour and processing related to the function is very similar to the corresponding $arith\_encode\_residual$ function in the encoder.

```
arith_decode_residual(cumulativeFrequencyTable, tableOffset)
{
  ari_decode_14bits_s27_ext(&val, cumulativeFrequencyTable);

  if ((val != 0) && (val != SYMBOLS_IN_TABLE - 1)) {
    x = (val - 1) + MIN_ENC_SEPARATE;

    x -= tableOffset;
    return x;
  }

  extra = arith_decode_bits(4);
  if (extra == 15) {
    extra_tmp = arith_decode_bits(6);
    if (extra_tmp == 63) {
      extra_tmp = 63 + arith_decode_bits(7);
```

```
    }
    extra = 15 + extra_tmp;
  }

  if (val == 0) {
    x = (MIN_ENC_SEPARATE - 1) - extra;
  } else { /* val == SYMBOLS_IN_TABLE - 1 */
    x = (MAX_ENC_SEPARATE + 1) + extra;
  }

  x -= tableOffset;
  return x;
}
```

The function $decode\_sfe\_vector$ decodes the scale factor vector g, which consists of nB integer values. The value t and the $prev$ vector, which constitute the decoder state, are used as additional parameters for the function. Note that the top level function $iisIGFSCFDecoderDecode$ must call the common arithmetic decoder initialization function $ari\_start\_decoding\_14bits$ before calling the function $decode\_sfe\_vector$, and also call the locally defined arithmetic decoder finalization function $arih\_decode\_flush$ afterwards.

```
decode_sfe_vector(t, prev, g, nB)
{
  for (f = 0; f < nB; f++) {
    if (t == 0) {
      if (f == 0) {
        ari_decode_14bits_s27_ext(&pred, cf_se00);
        g[f] = pred << 2;
        g[f] += arith_decode_bits(2); /* LSBs as 2 bit raw */
      }
      else if (f == 1) {
        pred = g[f - 1]; /* pred = b */
        g[f] = pred + arith_decode_residual(cf_se01, cf_off_se01);
      } else { /* f >= 2 */
        pred = g[f - 1]; /* pred = b */
        ctx = quant_ctx(g[f - 1] - g[f - 2]); /* Q(b - e) */
        g[f] = pred + arith_decode_residual(cf_se02[CTX_OFFSET + ctx],
            cf_off_se02[CTX_OFFSET + ctx]);
      }
    }
    else { /* t == 1 */
      if (f == 0) {
        pred = prev[f]; /* pred = a */
        g[f] = pred + arith_decode_residual(cf_se10, cf_off_se10);
      } else { /* (t == 1) && (f >= 1) */
        pred = prev[f] + g[f - 1] - prev[f - 1]; /* pred = a + b - c */
        ctx_f = quant_ctx(prev[f] - prev[f - 1]); /* Q(a - c) */
        ctx_t = quant_ctx(x[f - 1] - prev[f - 1]); /* Q(b - c) */
        g[f] = pred + arith_decode_residual(
            cf_se11[CTX_OFFSET + ctx_t][CTX_OFFSET + ctx_f],
            cf_off_se11[CTX_OFFSET + ctx_t][CTX_OFFSET + ctx_f]);
      }
    }
  }
}
```

The cumulative frequency tables and the corresponding table offsets are initialized identically to the encoder side.

6.2.2.2.10   Spectral data

The quantized spectral coefficients are read from the bit-stream by the means of arithmetic decoding.

6.2.2.2.11   Residual bits

The left-over bits (to the target bit budget) after arithmetic decoding are read by the residual decoding module.

6.2.2.3 Decoding process

6.2.2.3.1 Arithmetic decoder

The arithmetic decoder is described by the following pseudo-code. It takes as input arguments the cumulative frequency table *cum_freq[]* and the size of the alphabet *cfl*.

```
symbol = ari_decode(cum_freq[], cfl) {
    if (arith_first_symbol()) {
    value = 0;
    for (i=1; i<=16; i++) {
    value = (val<<1) | arith_get_next_bit();
    }
    low  = 0;
    high  = 65535;
    }

    range = high-low+1;
    cum =((((int) (value-low+1))<<14)-((int) 1))/range;
    p = cum_freq-1;

    do {
    q = p + (cfl>>1);
    if ( *q > cum ) { p=q; cfl++; }
    cfl>>=1;
    }
    while ( cfl>1 );

    symbol = p-cum_freq+1;
    if (symbol)
    high = low + ((range*cum_freq[symbol-1])>>14) - 1;

    low += (range * (cum_freq[symbol])>>14);

    for (;;) {
    if (high<32768) { }
    else if (low>=32768) {
    value -= 32768;
    low   -= 32768;
    high  -= 32768;
    }
    else if (low>=16384 && high<49152) {
    value -= 16384;
    low   -= 16384;
    high  -= 16384;
    }
    else break;

    low += low;
    high += high+1;
    value = (value<<1) | arith_get_next_bit();
    }
    return symbol;
}
```

6.2.2.3.1.1 Context-based arithmetic decoder

The context-based arithmetic decoder reads the following data in the following order:

1- $\left\lceil \log_2(\frac{L}{2}) \right\rceil$ bits for decoding *lastnz/2-1*.

2- The entropy-coded MSBs bits

3- The sign bits

4- The residual quantization bits

5- The LSBs bits are read backwardly from the end of the bitstream buffer.

The following pseudo-code shows how the spectral coefficients *X[]* or $\hat{X}_M$ are decoded. It takes as input argument the allocated bit budget *target_bits* and the number of coded samples *lastnz*. The helper functions are given in encoder subsection from 5.3.3.2.8.1.2 to 5.3.3.2.8.1.2.

```
X=ari_context_decode(target_bits,pi,hi,last_nz) {
    c[0]=c[1]=p1=p2=0;
    for (k=0; k<L; k++) {
    X[k]=0;

    for (k=0; k<lastnz; k+=2) {
      a=b=0;

    (a1_i,p1,idx1) = get_next_coeff(pi,hi,lastnz);
    (b1_i,p2,idx2) = get_next_coeff(pi,hi,lastnz);

    t=get_context(idx1,idx2,c,p1,p2);

    /* MSBs decoding */
    for (lev=esc_nb=0;;){
      pki = ari_context_lookup [t + 1024*esc_nb ];
      ari_decode(ari_cf_m [pki],17);

      if(r<16) break;

      /*LSBs decoding*/
      a=(a)+read_bit_end() <<(lev));
      b=(b)+ read_bit_end() <<(lev));
      lev++;
    esc_nb=min(lev,3);
    }

    /*MSBs contributions*/
    b1= r>>2;
    a1= r&0x3;
    a += (a1)<<lev;
    b += (b1)<<lev;

    /*Dectect overflow*/
    if(nbbits>target_bits){
      break;
    }

    c=update_context(a,b,a1,b1,c,p1,p2);

    /* Store decoded data */
     X[a1_i] = a;
     X[b1_i] = b;
  }
    /*decode signs*/
for (i=0; i<L; i++){
    if(X[i]>0){
      if ( read_bit()==1 ){
          X[i] = -X[i];
      }
    }
  }
 }
}
```

6.2.2.3.1.2 Envelope-based arithmetic decoder

The probability model is computed as described in the encoder subclause 5.3.3.2.8.1.2.3.

6.2.2.3.2 Adaptive low frequency de-emphasis

A general description of ALFE can be found in subclause 5.3.3.2.4.1.

6.2.2.3.2.1 Adaptive de-emphasis algorithm 1

ALFE algorithm 1 reverses the encoder-side LF emphasis 1 (see subclause 5.3.3.2.4.2). First, as was done in the encoder, the minimum and maximum of the first nine gains are found using comparison operations executed within a loop over the gain indices 0 to 8.

Then, if the ratio between the minimum and maximum exceeds a threshold of 1/32, a gradual lowering of the lowest lines in x is performed such that the first line is attenuated by $(\mathtt{max}/(32\ \mathtt{min}))^{0.25}$ and the 33rd line is not attenuated:

```
tmp = 32 * min;
if ((max < tmp) && (tmp > 0)) {
  fac = tmp = pow(max / tmp, 1/128);
  for (i = 31; i >= 0; i--) {      /* gradual lowering of lowest 32 lines */
    X[i] *= fac;
    fac  *= tmp;
  }
}
```

Adaptive de-emphasis algorithm 2

ALFE algorithm 2 reverses the encoder-side LF emphasis 2 (see subclause 5.3.3.2.4.3) by checking for modifications to the quantized LF MDCT lines and undoing them. As was done in the encoder, the procedure is split into five steps:

- Step 1: first find first magnitude maximum at index i_max in lower spectral quarter ($k = 0 \ldots L_{TCX}^{(bw)}$ / 4) for which $|\mathtt{Xq}[k]| \geq 4$ and modify the maximum as follows: Xq[i_max] += (Xq[i_max] < 0) ? 2 : -2

- Step 2: then expand value range of all x[k] up to i_max by multiplying all lines at $k = 0\ldots$i_max–1 with 0.5

- Step 3: again find first magnitude maximum in lower quarter of spectrum if the i_max found in step 1 is > -1

- Step 4: again expand value range of all x[i] up to i_max as in step 2, but using the i_max found in step 3

- Step 5: finish and always expand two lines at the latest i_max found, i.e. at $k = $ i_max+1, i_max+2. If the line magnitude at k is greater than or equal to 4, move it toward zero by two, otherwise multiply it by 0.5. As in the encoder all i_max are initialized to –1. For details please see AdaptLowFreqDeemph() in tcx_utils.c.

6.2.2.3.3 Global gain decoding

The global gain $\hat{g}_{TCX}$ is decoded from the index $I_{TCX,gain}$ transmitted in the bit stream as follows:

$$\hat{g}_{TCX} = \sqrt{\frac{160}{L_{TCX}^{(bw)}}} 10^{\frac{I_{TCX,gain}}{28}} \qquad (1763)$$

6.2.2.3.4 Residual bits decoding

At 13.2 kbps and above the 3 first bits are used for refining the global gain. The variable *n* is initialized to *0*:

if $(read\_bit() == 0)$ *then*

$\hat{g}_{TCX} = \hat{g}_{TCX} \cdot 10^{-2^{-n-2}/28}$

else then

$\hat{g}_{TCX} = \hat{g}_{TCX} \cdot 10^{2^{-n-2}/28}$

$n = n+1$

The following bits refine the non-zeroed decoded lines. 1 bit per non-zeroed spectral value is read. The rounding offset used in the first quantization stage with dead-zone is taking into account for computing the reconstructed points:

$$fac\_p = (1 - 0.375)/2$$

$$fac\_m = 0.375/2$$

$$if\ (read\_bit() == 0)\ then$$

$$\hat{X}[k] = \begin{cases} \hat{X}[k] - fac\_m & \text{if } \hat{X}[k] > 0 \\ \hat{X}[k] - fac\_p & \text{otherwise} \end{cases}$$

$$else\ then$$

$$\hat{X}[k] = \begin{cases} \hat{X}[k] + fac\_p & \text{if } \hat{X}[k] > 0 \\ \hat{X}[k] + fac\_m & \text{otherwise} \end{cases}$$

If at least 2 bits are left to read, a zeroed value is refined as:

$$fac\_z = (1 - 0.375) \cdot 0.33$$

$$if\ (read\_bit() == 0)\ then$$

$$\hat{X}[k] = 0$$

$$else\ then$$

$$if\ (read\_bit() == 0)\ then$$

$$\hat{X}[k] = -2 \cdot fac\_z$$

$$else\ then$$

$$\hat{X}[k] = 2 \cdot fac\_z$$

6.2.2.3.5 TCX formant enhancement

The TCX formant enhancement intends to replicate a behavior similar to that of the ACELP formant enhancement. It operates based on the LPC frequency-band gains, `lpcGains[]`. First, the square-root of each gain is computed. Then,

```
fac = 1 / min(sqrtGains[0], sqrtGains[1]);
k = 0;
for (i = 1; i < numGains - 1; i++) {
  if ((sqrtGains[i-1] <= sqrtGains[i]) && (sqrtGains[i+1] <= sqrtGains[i])) {
    step = max(sqrtGains[i-1], sqrtGains[i+1]);
    step = (1 / step - fac) / (i - k);
    sqrtGains[k] = 1;
    fac += step;
    for (j = k + 1; j < i; j++) {
      sqrtGains[j] = min(1, sqrtGains[j] * fac);
      fac += step;
    }
    k = i;
  }
}
```

where `sqrtGains[]` contains the square-roots of the `lpcGains[]`, and `numGains` denotes the number of LPC gains. In order to complete the above algorithm for the last gain at $i = $ `numGains` $- 1$, the following operation is executed,

```
    step = min(sqrtGains[i-1], sqrtGains[i]),
```

and the above steps inside the `if`-condition, starting with "`step = (1 / step - fac) / (i - k)`", are repeated. Finally, we set `sqrtGains[numGains - 1] = 1` and multiply the modified set of gains onto the decoded spectrum:

```
for (i = j = 0; i < L_TCX^(bw)  j++) {
  for (k = 0; k < L_TCX^(bw) / numGains; i++, k++) {
    X̂_M[i] *= sqrtGains[j];
  }
}
```

with $\hat{X}_M$ being the decoded spectrum. Like its ACELP counterpart, TCX formant enhancement is only used at 9.6 kbps.

6.2.2.3.6 Noise Filling

Noise filling is applied to fill gaps in the MDCT spectrum where coefficients have been quantized to zero. Pseudo-random noise is inserted into the gaps, starting at bin $k_{NFstart}$ up to bin $k_{NFstop} - 1$. The amount of noise inserted is controlled by a noise factor transmitted in the bit stream. To compensate for LPC tilt, a tilt compensation factor is computed. At each side of a noise filling segment a fadeout over t_{NF} bins is applied to the inserted noise to smooth the transition.

The start and stop bins $k_{NFstart}$ and k_{NFstop} are determined as described in subclause 5.3.3.2.10.2 . $k_{NFstop,LP}$ is set to the same value as k_{NFstop}:

$$k_{NFstop,LP} = k_{NFstop} \tag{1764}$$

Computation of the tilt compensation factor t_{NF} is described in subclause 5.3.3.2.10.1. The transition width w_{NF} is computed as described in 5.3.3.2.10.3.

6.2.2.3.6.1 Decoding of Noise Factor

The dequantized noise factor $\hat{f}_{NF}$ is obtained from the transmitted index I_{NF} as follows:

$$\hat{f}_{NF} = 0.09375 \cdot I_{NF} \tag{1765}$$

6.2.2.3.6.2 Noise Filling Seed

The inserted noise is generated as a sequence of pseudo-random numbers, which is computed in a recursive way starting with a seed s_{NF} computed from the quantized MDCT coefficients $\hat{X}_M$:

$$s_{NF} = \left(\left(\left(32768 + \sum_{i=0}^{k_{NFstop}-1} 2 \cdot i \cdot \left| \hat{X}_M(i) \right| \right) \bmod 65536 \right) - 32768 \right) \tag{1766}$$

6.2.2.3.6.3 Filling Noise Segments

Determining the number and start/stop bins of noise filling segments is described in 5.3.3.2.10.2 and 5.3.3.2.10.4.

For each segment pseudo-random noise is generated and normalized, so that it has an RMS of one. Then tilt compensation, noise factor and transition fadeouts are applied. The resulting coefficients are inserted to the quantized MDCT spectrum $\hat{X}_M$ and replace the zeroes in the noise filling segments. The following pseudo-code defines the exact procedure:

$$s = s_{NF}$$

$for \ j = 0...n_{NF} - 1\{$

$\quad E = 0$

$\quad for \ k = k_{NF0}(j)...k_{NF1}(j) - 1\{$

$\qquad s = \left((31821 \cdot s + 13849 + 32768) \bmod 65536\right) - 32768$

$\qquad r(k) = s$

$\qquad E = E + s^2$

$\quad\}$

$$R = \sqrt{\frac{E}{k_{NF0}(j) - k_{NF1}(j)}}$$

$\quad for \ k = k_{NF0}(j)...k_{NF1}(j) - 1\{$

$$\qquad \hat{X}_M(k) = \hat{f}_{NF} \frac{r(k)}{R} \frac{\min(k - k_{NF0}(j) + 1, \, w_{NF})}{w_{NF}} \frac{\min(k_{NF1}(j) - k, \, w_{NF})}{w_{NF}} (t_{NF})^i$$

$\quad\}$

$\}$

6.2.2.3.7 Apply global gain and LPC shaping in MDCT domain

The decoded global gain factor $\hat{g}_{TCX}$ is applied to all MDCT coefficients. The LPC shaping of the MDCT spectrum applied on encoder side is inverted by multiplying the spectral coefficients by the LPC shaping gains g_{LPC} .

The computation of the shaping gains is performed in the same way as on encoder side, see subclause 5.3.3.2.3.2.

The following pseudo-code defines how global gain and LPC shaping are applied to the MDCT bins corresponding to the CELP frequency range:

$$w = \left\lfloor \frac{L_{TCX}^{(celp)}}{64} \right\rfloor , \quad r = L_{TCX}^{(celp)} - 64w$$

if $r = 0$ *then*
  $s = 1, \quad w_1 = w_2 = w$
else if $r \leq 32$ *then*
  $s = \left\lfloor \frac{64}{r} \right\rfloor \quad w_1 = w, \quad w_2 = w + 1$
else
  $s = \left\lfloor \frac{64}{64 - r} \right\rfloor \quad w_1 = w + 1, \quad w_2 = w$

$i = 0$
for $j = 0...63$ {
  if $j \bmod s \neq 0$ *then*
    $w = w_1$
  else
    $w = w_2$

  for $l = 0...\min\left(w, L_{TCX}^{(celp)} - i\right) - 1$ {
    $X'_M(i) = \hat{g}_{TCX} \cdot g_{LPC}(j) \cdot \hat{X}_M(i)$
    $i = i + 1$
  }
}

For the remaining MDCT coefficients above the CELP frequency range (if any) the last LPC shaping gain is used:

$$X'_M(i) = \hat{g}_{TCX} \cdot g_{LPC}(63) \cdot \hat{X}_M(i) \quad , \quad i = L_{TCX}^{(celp)}...L_{TCX}^{(bw)} - 1 \tag{1767}$$

6.2.2.3.8   IGF apply

6.2.2.3.8.1   IGF independent noise filling

IGF uses independent noise filling in the IGF range. Through independent noise filling, core coder noise filling is replaced by random noise which is de-correlated from the core coder noise filling. Therefore a vector φ is filled with either 0 or 1 by evaluating the TCX noise-filling routine from subclause 6.2.2.3.6.3 such that every subband, which is noise-filled by TCX noise-filling, represents a 1 in φ, all other entries are set to 0 in φ.

First, the total noise energy E in the IGF source range in the decoded MDCT spectrum X is calculated:

$$E(k) := \sum_{i=minSB}^{t(0)-1} \left(\varphi(i) \cdot X(i)^2\right) + \varepsilon , \tag{1768}$$

where $\varepsilon = 1.175494351 \cdot 10^{-38}$. The noise indicated by φ is replaced according to the following formula:

$$X_n(i,j) := \begin{cases} r(i), & \varphi(i) = 1 \\ X(i), & \varphi(i) = 0 \end{cases} , \quad \forall i \in [minSB, minSB+1, minSB+2,...,t(0)[, \; \forall j \in [0,1,...,nT[\tag{1769}$$

where X_n contains nT copies of the spectrum with independent noise per copy, i.e. per IGF tile. For creating pseudo random numbers r(i), the random generator described in subclause 6.2.2.3.6.3 is used.

The energy E_{IGF} of the inserted pseudo random numbers is measured with

$$E_{IGF}(j) := \sum_{i=minSB}^{t(0)-1} \left(\varphi(i) \cdot X_n(i,j)^2 \right), \forall j \in [0,1,\ldots,nT[. \tag{1770}$$

Now the inserted noise is adjusted to the same energy level as the original noise. Therefore the correction factor g is calculated:

$$g(j) := \begin{cases} \sqrt{\dfrac{E}{E_{IGF}(j)}} \,, & E_{IGF} > 0 \\ 0, \ otherwise \end{cases}, \forall j \in [0,1,\ldots,nT[\tag{1771}$$

Using g, the replaced noise is rescaled to match the original noise energy level in X:

$$X_n(i,j) := \begin{cases} g(j) \cdot X_n(i,j), \ \varphi(i)=1 \\ X(i), \ \varphi(i)=0 \end{cases}, \forall i \in [minSB, minSB+1, minSB+2, \ldots, t(0)[, \ \forall j \in [0,1,\ldots,nT[\tag{1772}$$

6.2.2.3.8.2 IGF whitening generation

In order to remove possible formant structure of the tiled signal X and to suppress strong tonal components the routine *IGF_getWhiteSpectralData()* will be applied to the TCX spectrum X if the bitstream element $currWLevel(k)$ is 1 for any tile t. The algorithm is a low complex simplification of the following formula:

$$Y(b) = \frac{X(b)}{\sqrt{\sum_{i=b-7}^{b+7} X(i)^2}} \tag{1773}$$

which is a division of the spectrum by the square root of a moving average calculated on the spectrum. $X(b)$ denotes the TCX coefficient of the decoded core signal with index b prior to application of the LPC filter.

Since the above formula would need a division and a square root operation per line b – two complex operations (18 and 10 OPS) – the operations are done in logarithmic domain, while the logarithm is replaced with a low complex rounded integer logarithm to the basis 2.

$$Y(b) = X(b) \cdot 2^{-0.5 \cdot INT\_LOG\_2 \left(\sum_{i=b-7}^{b+7} X(i)^2 \right)} \tag{1774}$$

where

$$INT\_LOG\_2(x) = \left\lfloor \frac{\log(x)}{\log(2)} \right\rfloor \tag{1775}$$

The length of the moving average (MA) filter $Xma_b = \sum_{i=b-7}^{b+7} X(i)^2$ is 15 bins in total.

The range of bins b on which this whitening operation has to be carried out is going from $minSb$ to $startLine$, where $startLine$ is the index of the first scale factor band - 1. Because $minSb$ is always greater 7, the MA filter will be calculated from $minSb-7$ on. However, the calculation of the MA filter has to be different for the last 6 bins below $startLine$:

$$Xma_{stopLine-n+7+1} = \sum_{i=stopLine-n}^{stopLine} \frac{15}{n} X(i)^2, \ n = [14..7] \tag{1776}$$

If the bitstream element $currWLevel(k)$ is 2 for any tile k no core signal will be copied but a sequence of pseudo-random numbers will be used instead as described in subclause 6.2.2.3.6.2.

The seed for the pseudo random number generator (described in subclause 6.2.2.3.6.3) is derived from the TCX noise filling seed by:

$$IGF\_SEED = S_{NF} \cdot 31821 + 13849 \tag{1777}$$

6.2.2.3.8.3 IGF envelope reconstruction

The IGF envelope reconstruction tool shapes the noise components filled into the gaps in the IGF range in order to adjust the spectral envelope as a function of the transmitted IGF scale factors.

6.2.2.3.8.3.1 Dequantizing IGF scale factors

For de-quantizing the IGF scale factors N_{bs}, transmitted in the bitstream, to N_d the following mapping is applied:

$$N_d(k) = 2^{\frac{1}{4}N_{bs}([k])-4}, \ \forall k \in [0,1,2,\ldots,nB[\tag{1778}$$

6.2.2.3.8.3.2 Refining IGF scale factor borders

In order to optionally smooth the transmitted scale factors along the frequency axis, a refinement of the IGF scale-factor borders is introduced:

$$t'(2k+0) := t(k), \ t'(2k+1) := \left\lfloor t(k) + \frac{9}{20}(t(k+1) - t(k)) + \frac{1}{2} \right\rfloor, \ \forall k \in [0,1,2,\ldots,nB[\tag{1779}$$

The de-quantized IGF scale factors shall be mapped:

$$N_d^{'[2k+0]} := N_d^{'[2k+1]} := N_d[k], \ \forall k \in [0,1,2,\ldots,nB[\tag{1780}$$

For simplicity, $t[k]$ is also mapped:

$$\begin{aligned} t(k) &:= t'(k), \ \forall k \in [0,1,2,\ldots,2nB[\\ N_d(k) &:= N_d^{'}(k), \ \forall k \in [0,1,2,\ldots,2nB[\end{aligned} \tag{1781}$$

The IGF envelope refinement is active for bitrates ≤ 64 kbps for all operating modes WB, SWB, FB.

6.2.2.3.8.3.3 Collecting energies below $t(0)$

To stabilize energy distribution in the range of $t(0)$, energy below $t(0)$ is collected:

$$N_b = \sqrt{\frac{1}{24} \sum_{i=t(0)-24}^{t(0)-1} X(i)^2} \tag{1782}$$

The energy N_b is later used to adapt the first IGF scale factor band energy as described in subclause 6.2.2.3.8.3.7.

6.2.2.3.8.3.4 Collecting residual energies in IGF range

The residual energy N_s determines the energy of the non-zero subbands in the IGF range:

$$N_s(k) := \sum_{i=t(k)}^{t(k+1)-1} X(i)^2, \ \forall k \in [0,1,2,\ldots,2nB[\tag{1783}$$

N_s is therefore the energy of the de-quantized subband values above $t(0)$ which are not quantized to zero by the tonal mask detection of the encoder described in subclause 5.3.3.2.11.5.

6.2.2.3.8.3.5 Collecting tile energies in IGF range

The tile energy N_{IGF} determines the energy of the signal which is filled into the gaps in the IGF range:

$$N_{IGF}(k) := \sum_{i=t(k)}^{t(k+1)-1} X_{IGF}(i)^2 b(i), \ \forall k \in [0,1,2,\ldots,nB[\tag{1784}$$

$$b(i) = \begin{cases} 0, & X(i) \neq 0 \\ 1, & \textit{otherwise} \end{cases} \tag{1785}$$

where X_{IGF} is the signal after filling the gaps in X using the mapping function m as described in subclause 5.3.3.2.11.1.8:

$$X_{IGF}(i) := \begin{cases} X(m(i)), & X(i) = 0 \\ X(i), & \textit{otherwise} \end{cases}, \ \forall i \in [t(0), t(1), \ldots, t(2nB)[\tag{1786}$$

N_{IGF} is therefore the energy of the synthesized subband values above $t(0)$ which are quantized to zero by the tonal mask detection of the encoder described in subclause 5.3.3.2.11.5.

6.2.2.3.8.3.6 Rescaling IGF scale factor band energies

The rescaling of the IGF scale factors has to be done in order to bring them on the correct energy level for the subsequent calculation of the IGF gains. In dependency of the refinement rescaling is applied on a scale factor band basis or on a group of scale factor bands. The rescaled IGF scale factor band energy is therefore called IGF destination energy N_d.

In case refinement is not active, the rescaling is applied as follows:

$$N_d(k) := \sqrt{max\left(0.001 \cdot \frac{N_s(k)}{t(k+1)-t(k)}, N_d(k)^2 - \frac{N_s(k)}{t(k+1)-t(k)}\right)}, \ \forall k \in [0,1,2,\ldots,nB[\tag{1787}$$

In case refinement is active, two subsequent scale factor band energies are mapped:

$$N_d(k) := \begin{cases} \sqrt{max\left(0.001 \cdot \frac{N_s(k)+N_s(k+1)}{t(k+2)-t(k)}, N_d\left(\frac{k}{2}\right)^2 - \frac{N_s(k)+N_s(k+1)}{t(k+2)-t(k)}\right)}, & \forall k \in [0,2,4,\ldots,2nB[\\ N_d(k-1), & \forall k \in [1,3,5,\ldots,2nB[\end{cases} \tag{1788}$$

6.2.2.3.8.3.7 Adaption of IGF scale factor band energies

The IGF scale factor band energies have to be adapted to fulfil the signal requirements. The first scale factor band energy is adapted to the energy below $t(0)$ using N_b as introduced in subclause 6.2.2.3.8.3.3:

$$N_d{}'(0) := \begin{cases} N_d(0) + fFac \cdot (N_b - N_d(0)), & N_b < N_d(0) \\ N_d(0), & \textit{otherwise} \end{cases} \tag{1789}$$

where $N_d{}'$ is the adapted IGF scale factor band energy N_d and $fFac$ is the adaption factor for the first scale factor band energy according to table 166:

Table 166: IGF scale factor band energy adaption factors

Bitrate	Mode	*fFac*	*gFac*	*lFac*
9.6 kbps	WB	0.7	0.8	0.6
9.6 kbps	SWB	0	1.0	1.0
13.2 kbps	SWB	0.2	0.93	0.85
16.4 kbps	SWB	0.2	0.93	0.85
24.4 kbps	SWB	0.2	0.965	0.85
32.0 kbps	SWB	0.2	0.965	0.85
48.0 kbps	SWB	0.2	1.0	1.0
64.0 kbps	SWB	0.2	1.0	1.0
16.4 kbps	FB	0.2	0.93	0.85
24.4 kbps	FB	0.2	0.965	0.85
32.0 kbps	FB	0.2	0.965	0.85
48.0 kbps	FB	0.2	1.0	1.0
64.0 kbps	FB	0.2	1.0	1.0
96.0 kbps	FB	0	1.0	1.0
128.0 kbps	FB	0	1.0	1.0

The last scale factor band energy is adapted as follows:

$$N_d^{'}(2nB-1) = lFac \cdot N_d(2nB-1) \tag{1790}$$

where $lFac$ is the adaption factor for the last scale factor band energy according to table 166.

If refinement is active and $currWLevel(0) = 2$, the remaining scale factor band energies are low-pass filtered in order to smooth the frequency envelope:

$$N_d^{'}(k) := 0.201 \cdot N_d(k-1) + 0.389 \cdot N_d(k) + 0.41 \cdot N_d(k+1), \ \forall k \in [1,2,3,\ldots,nB-1[\tag{1791}$$

Otherwise the scale factor band energies are not affected by further modifications:

$$N_d^{'}(k) := N_d(k), \ \forall k \in [1,2,3,\ldots,nB-1[\tag{1792}$$

6.2.2.3.8.3.8 Calculation of IGF gain factors

The IGF gain factors are used to finally shape the tiled subband values in order to adjust the spectral envelope of the synthesized signal above $t(0)$. First, the target energy level N_t has to be calculated:

$$N_t(k) := \sum_{i=0}^{hop-1} \left(N_d^{'}(min(k+i,2nB-1))^2 \cdot t(min(k+i+1,2nB)) - t(min(k+i,2nB)) \right), \tag{1793}$$

$$\forall k \in [0,hop,2hop,3hop,\ldots,2nB[\tag{1794}$$

where hop is the hop-size of the refinement in dependency of $currWLevel[0]$ and the maximal possible hop-size $mHop$ according to table 167:

$$hop = min\left(mHop, \begin{cases} 4, & currWLevel(0) = 0 \\ 1, & currWLevel(0) = 2 \\ 2, & otherwise \end{cases} \right) \tag{1795}$$

Table 167: Maximal IGF hop-size

Bitrate	mode	*mHop*
9.6 kbps	WB	4
9.6 kbps	SWB	2
13.2 kbps	SWB	4
16.4 kbps	SWB	4
24.4 kbps	SWB	4
32.0 kbps	SWB	4
48.0 kbps	SWB	4
64.0 kbps	SWB	4
16.4 kbps	FB	4
24.4 kbps	FB	2
32.0 kbps	FB	2
48.0 kbps	FB	2
64.0 kbps	FB	2
96.0 kbps	FB	1
128.0 kbps	FB	1

Second, a normalization term *norm* for normalizing the target energy N_t is calculated as follows:

$$norm(k) := \sum_{i=0}^{hop-1} \left(t(min(k+i+1,2nB)) - t(min(k+i,2nB)) \right), \quad (1796)$$

$$\forall k \in [0, hop, 2hop, 3hop, \ldots, 2nB[\quad (1797)$$

Finally, the IGF gain factors have to be calculated according to the following formula:

$$g(k) = \sqrt{\frac{1}{N_{IGF}(k)}(t(k+1) - t(k))\left(gFac \cdot \sqrt{\frac{hop \cdot N_t(k)}{norm(k)}} \right)^2}, \quad \forall k \in [0, hop, 2hop, 3hop, \ldots, 2nB[\quad (1798)$$

where *gFac* is the general adaption factor for all scale factor band energy according to table 166.

If hop-size $hop > 1$, the IGF gain factors in between a particular hop are hold beginning at the hop-start:

$$g(k) = g(i), \ \forall i \in [0, hop, 2hop, 3hop, \ldots, 2nB[, \ \forall k \in [i+1, i+2, \ldots, i+hop[\quad (1799)$$

In order to prevent the envelope adjustment from unwanted overshoots, the IGF gain factors are compressed using hard-knee compression:

$$g(k) = \begin{cases} \dfrac{g(k)+255}{256}, & g(k) > 1 \\ g(k), & otherwise \end{cases}, \ \forall k \in [0,1,2,..,2nB[\quad (1800)$$

6.2.2.3.8.3.9   IGF envelope adjustment

With the calculated IGF gain factors g as described in subclause 6.2.2.3.8.3.8, the envelope of the spectrum above $t(0)$ is adjusted as follows:

$$X(tb) := \begin{cases} g([k]) \cdot X_{IGF}(tb), & X(i) = 0 \\ X_{IGF}([tb]), & otherwise \end{cases}, \forall tb \in [t(k), t(k)+1, t(k)+2, \ldots, t(k+1)[, \ \forall k \in [0,1,2,\ldots,nB[\quad (1801)$$

where $t(0), t(1), \ldots, t(nB)$ shall be already mapped with the function tF, see subclause 5.3.3.2.11.1.1, and nB being the number of bands. X_{IGF} is the gap-filled signal in accordance with subclause 6.2.2.3.8.3.5.

6.2.2.3.9 Inverse window grouping (TCX5 separation)

If the configuration, determined as described in subclause 6.2.4.2, indicates that some sub-frames are coded using TCX5 then a sub-frame containing MDCT bins of 2 TCX5 sub-frames is de-interleaved to form 2 consecutive TCX5 sub-frames before the optional Temporal Noise Shaping, the optional IGF temporal flattening and before the transformation with the inverse MDCT:

$$X_M^1(k) = X_M(2k),$$
$$X_M^2(k) = X_M(2k+1), \quad k = 0,...,\frac{L}{4}-1 \tag{1802}$$

6.2.2.3.10 Temporal Noise Shaping

The decoding process for Temporal Noise Shaping is carried out separately on each window of the current frame by applying the so called lattice filter to selected regions of the spectral coefficients (see in subclause 5.3.3.2.2). The number of noise shaping filters applied to each window is specified by "nMaxFilters ".

For TCX5 the same rearrangement is done as described in subclause 5.3.3.2.2 prior to the TNS filtering and the rearrangement is reverted after the filtering.

First the transmitted filter coefficients have to be decoded, i.e. conversion to signed numbers, inverse quantization. Then the so called lattice filters are applied to the target frequency regions of the spectral coefficients (see subclause 5.3.3.2.2). The maximum possible filter order is defined by the constant TNS_MAX_FILTER_ORDER.

The application of TNS shaping filter is done before the optional IGF temporal flattening and before the transformation with the inverse MDCT

The decoding process for one window can be described as follows pseudo code:

```
/* TNS decoding for one window */
tns_decode()
{
    set_zero( state, TNS_MAX_FILTER_ORDER );
    for (iFilter = nMaxFilters-1; iFilter >= 0; iFilter--) {
    tns_decode_coef( order, index[iFilter], parCoeff[iFilter] );
    tns_filter( spectrum, startLine, endLine, parCoeff[iFilter], order, state );
    }
}
/* Decoder transmitted coefficients index[] for one TNS filter */
tns_decode_coef( order, index[], parCoeff[] )
{
    /* Conversion to signed integer */
    for (i = 0; i < order; i++)
    tmp[i] = index[i] + (1 << (TNS_COEF_RES-1));

    /* Inverse quantization */
    iqfac   = ((1 << (TNS_COEF_RES-1)) - 0.5) / (π/2.0);
    iqfac_m = ((1 << (TNS_COEF_RES-1)) + 0.5) / (π/2.0);
    for (i = 0; i < order; i++)  {
    parCoeff[i] = sin( tmp[i] / ((tmp[i] >= 0) ? iqfac : iqfac_m) );
    }
}
/* Lattice filter */
tns_filter( spectrum[], startLine, endLine, parCoeff[], order, state )
{
    for (j = startLine; j <= endLine; j++) {
    spectrum[j] -= parCoeff[order-1] * state[order-1];
    for (i = order-2; i >= 0; i--) {
    spectrum[j] -= parCoeff[i] * state[i];
    state[i+1] = parCoeff[i] * spectrum[j] + state[i];
    }
    state[0] = spectrum[j];
    }
}
```

Filter order 1 with the first coefficient equal to 0 identifies disabled filter.

Please note that this pseudo code uses a „C"-style interpretation of arrays and vectors, i.e. if parCoeff describes the coefficients for all filters, parCoeff[iFilter] is a pointer to the coefficients of one particular filter.

6.2.2.3.11   IGF temporal flattening

The reconstructed signal by IGF is temporally flattened in the frequency domain when $isIgfTemFlat = 1$. The temporal flattening is performed in a frequency-selective manner as follows.

The selection of the spectral contents to be temporally flattened is done by comparing the quantized spectral coefficients with 0 and the contents whose coefficients are quantized to 0 are selected.

In order to maintain the significant spectral contents, they are temporarily replaced by the spectra which are similarly generated to the filled spectra by IGF:

$$X_{tempFlat}[tb] = \begin{cases} X[tb] & X_{dec}[tb] = 0 \\ g[k]X_{dec}[m(k)] & abs(X_{dec}[tb]) > 0 \end{cases}, \quad t(k) \le tb < t(k+1), \quad k = 0,1,\ldots,nB-1 \qquad (1803)$$

where $X_{dec}[tb]$ is the quantized MDCT coefficient after arithmetic decoding and $X[tb]$ is the reconstructed MDCT coefficient by IGF.

The linear prediction of the spectra $X_{tempFlat}[tb]$ is done and the linear prediction coefficients $a_{igfTemp}(m), \; m = 1,2,\ldots,8$ are calculated. Then the temporally flattened spectrum is given by the following filtering:

$$X_{tempFlat}'[tb] = X_{tempFlat}[tb] + \sum_{m=1}^{M} a_{igfTemp}(m) \cdot X_{tempFlat}[tb-m], \quad t(k) \le tb < t(k+a), \quad k = 0,1,\ldots,nB-1 . (1804)$$

Finally, the significant spectral contents are restored by:

$$X_{tempFlat}'[tb] = \begin{cases} X_{tempFlat}'[tb] & X_{dec}[tb] = 0 \\ X[tb] & abs(X_{dec}[tb]) > 0 \end{cases}, \quad t(k) \le tb < t(k+a), \quad k = 0,1,\ldots,nB-1, \qquad (1805)$$

and then the frequency-selectively temporally flattened spectrum $X_{tempFlat}'[tb]$ is output to IMDCT for getting the time domain signal.

6.2.3   High Quality MDCT decoder (HQ)

6.2.3.1   Low-rate HQ decoder

6.2.3.1.1   Mode decoding

Based on the encoded bandwidth and operated bit-rate, mode information is decoded from 1 or 2 bits. Based on the decoded mode information, decoding configurations like band structures are set. The band structure definition for NB, WB, SWB, and FB is the same as encoder presented in table 103 to 108.

6.2.3.1.2   Energy Envelope decoding

From the received low-rate HQ envelope coding method bit and coding mode bit, the low-rate HQ energy decoding mode is determined and the coded quantization differential indices are decoded by the Large symbol decoding method or the Small symbol decoding method. From the received coding mode bits for energy envelope decoding, the envelope coding mode flag $DENG\_CMODE$ is determined, based on the coding mode the transmitted differential indices are decoded. For example, if flag $DENG\_CMODE$ has value 1 the Small symbol decoding method is used otherwise the Large symbol decoding method is used for decoding the differential indices.

The final resulting reconstructed quantized energies $\hat{I}_M(b), \; b = 0,\ldots,N_{bands}-1$ are obtained equally as in the encoder, described in subclause 5.3.4.1.3.

6.2.3.1.2.1 Small symbol decoding method

If the flag $DENG\_CMODE$ has value 1, the flag LC_{mode} information is extracted from the bit stream. If the LC_{mode} has value 1 resized Huffman decoding mode is used otherwise context based Huffman decoding mode is used for decoding the differential indices.

If IsTransient is True,

The decoded differential indices $\Delta I_M(b), \quad b = 1,....,N_{bands-1}$ is extracted either from context based or resized Huffman coding mode according to flag LC_{mode} and the differential indices for band $b=0$ $\Delta I_M(0)$, are up packed directly with 5 bits. The decoded differential indices are adjusted to extract the original values according to

$$\Delta I_M(b) \leftarrow \Delta I_M(b) - 15, \quad b = 0,....,N_{bands-1} \tag{1806}$$

If IsTransient is False,

The decoded differential indices $\Delta I_M(b), \quad b = 1,....,N_{bands-1}$ is extracted either from context based or resized Huffman coding mode according to flag LC_{mode} and the differential indices for band $b=0$ $\Delta I_M(0)$, are up packed directly with 5 bits. Once the differential indices are extracted, least significant code R_{LSB} are up packed directly with 1 bit and the differential indices are reconstructed according to

$$\Delta I_M(b) = 2\Delta I_M(b) \, OR \, R_{LSB} \, (\text{Binary operation}) \quad b = 0,....,N_{bands-1} \tag{1807}$$

The decoded differential indices are adjusted to extract the original values according to

$$\Delta I_M(b) \leftarrow \Delta I_M(b) - 32, \quad b = 1,....,N_{bands-1}$$
$$\Delta I_M(0) \leftarrow \Delta I_M(0) - 46 \tag{1808}$$

6.2.3.1.2.1.1 Context based Huffman decoding mode

If the context based Huffman decoding mode has been determined, the decoding is performed by referring table 168 and table 169 based on the context described in subclause 5.3.4.1.3.3.1. Four LSBs of entries in table 168 and table 169 indicates how many bits shall be read from bit-stream buffer to decode the next symbol and the signs indicate if the Huffman decoding is terminated or not. The procedure of how to perform Huffman decoding is shown below:

```
i=0
while( hufftab[i] > 0)
{
    read_bits += hufftab & 0xF
    i = (hufftab[i]>>4)+read_bits(hufftab[i] & 0xF)
}
return hufftab[i]
```

Table 168: Huffman decoding table for the context based Huffman decoding (*group0*,*group2*)

Index	Code	Index	Code	Index	Code	Index	Code	Index	Code
0	0X13	11	-0X0D	22	-0X18	33	0X41	44	-0X05
1	-0X10	12	0X51	23	-0X16	34	-0X1C	45	-0X1E
2	-0X0F	13	0X62	24	0X71	35	-0X08	46	-0X04
3	-0X11	14	-0X14	25	-0X0B	36	0X31	47	-0X1F
4	0X51	15	0X81	26	0X71	37	0X41	48	0X11
5	0X61	16	-0X0C	27	-0X1A	38	-0X1D	49	-0X03
6	-0X0E	17	0X81	28	0X71	39	-0X06	50	0X11
7	-0X12	18	-0X15	29	-0X09	40	0X31	51	-0X02
8	0X51	19	-0X17	30	-0X1B	41	0X41	52	0X11
9	0X61	20	0X71	31	-0X0A	42	-0X07	53	-0X01
10	-0X13	21	0X81	32	-0X19	43	0X41	54	0X00

Table 169: Huffman decoding table for the context based Huffman decoding (*group1*)

Index	Code	Index	Code	Index	Code	Index	Code	Index	Code
0	0X12	12	-0X12	24	0X51	36	-0X18	48	-0X1B
1	0X41	13	0X42	25	0X61	37	-0X05	49	-0X1A
2	-0X0F	14	-0x0C	26	-0X16	38	-0X04	50	0X11
3	0X41	15	0X61	27	-0X09	39	-0X03	51	0X00
4	-0X10	16	-0X13	28	0X51	40	0X51	52	0X11
5	-0X0E	17	0X61	29	0X61	41	-0X06	53	-0X1D
6	0X31	18	0X71	30	-0X17	42	0X51	54	0X11
7	-0X11	19	-0X0A	31	0X62	43	-0X19	55	-0X1E
8	0X31	20	0X71	32	-0X08	44	0X51	56	-0X1F
9	0X41	21	-0X0B	33	0X81	45	-0X01	57	-0X1B
10	-0X0D	22	-0X14	34	-0X07	46	-0X1C	-	-
11	0X41	23	-0X15	35	0X81	47	-0X02	-	-

6.2.3.1.2.1.2   Resized Huffman decoding mode

If *IsTransient* is *True*

If the frame is Transient, the Huffman decoding is then performed on the transmitted differential indices. The Huffman codes for the differential indices are given in table 111 in subclause 5.3.4.1.3.3.

If *IsTransient* is *False*

For Non-Transient frames, the Huffman decoding is then performed on the transmitted differential indices. The Huffman codes for decoding the indices are given in table 115 in subclause 5.3.4.1.3.3.3. The differential indices decoded using table 115 takes the form, $\Delta I'_M(b)$ the decoded differential indices $\Delta I'_M(b)$ are reconstructed which is exactly reverse to the encoder described in subclause 5.3.4.1.3.3.3 equation (1040). The way to reconstruct the differential index, which corresponds to the modification in encoder, can be done as shown in the following equation.

$$
\begin{aligned}
&if \ \Delta I_M(b-1) > 17, \\
&\quad \Delta I_M(b) = \Delta I'_M(b) - \min(\Delta I_M(b-1) - 17, 3) \\
&if \ \Delta I_M(b-1) > 13, \\
&\quad \Delta I_M(b) = \Delta I'_M(b) - \max(\Delta I_M(b-1) - 13, -3)
\end{aligned}
\qquad b = 2, \ldots, N_{bands} - 1 \qquad (1809)
$$

6.2.3.1.2.2   Large symbol decoding method

If the Large symbol coding method is determined, the encoded envelope data should be decoded using the reverse process of encoding either the pulse mode or the scale mode as described in subclause 5.3.4.1.3.4

The Huffman data in the encoded envelope data is decoded by a Huffman decoding method described in subclause 6.2.3.1.2.1.1 using table 170.

Table 170: Huffman decoding table for the Large symbol decoding method

Index	Code	Index	Code	Index	Code
0	0X11	5	0X21	10	-0X01
1	0X21	6	-0X02	11	-0X06
2	-0X04	7	-0X05	12	0X11
3	0X21	8	0X11	13	-0X07
4	-0X03	9	0X21	14	-0X00

6.2.3.1.3   Spectral coefficients decoding

6.2.3.1.3.1   Normal Mode

Figure 94 shows the overview of the normal mode decoder.
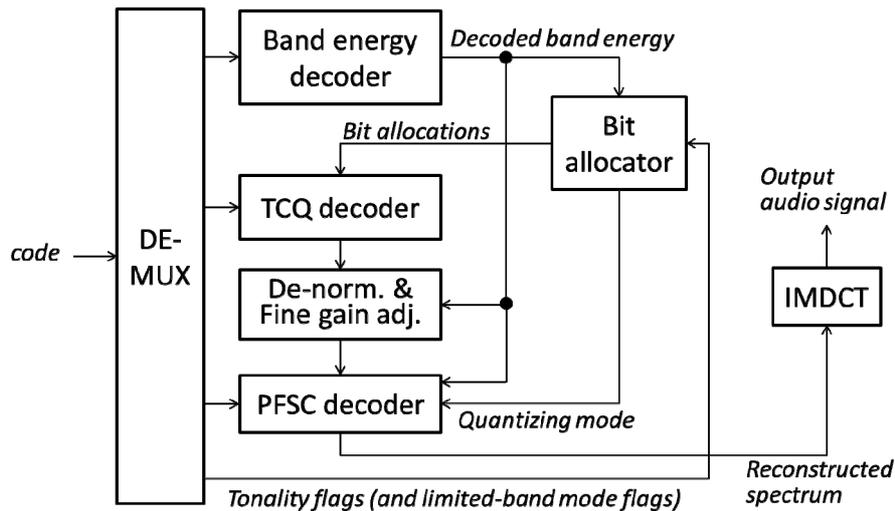
Figure 94: Block diagram of the Normal mode decoder overview

6.2.3.1.3.1.1   Energy envelope decoding

Details are described in subclause 6.2.3.1.2.

6.2.3.1.3.1.2   Tonality flag decoding

Tonality flags described in subclause 5.3.4.1.4.1.3 are decoded and used for calculation of the bit allocations.

6.2.3.1.3.1.3   Bit allocation

The processing is in the same manner with the one at the encoder side.

Firstly, the fine gain adjustment bits are derived.

Secondly, the bands of the limited-band mode are identified based on the decoded limited-band mode flags, and their corresponding bandwidths are set if the limited-band mode is used in encoding.

Thirdly, bands encoded using PFSC are identified based on the decoded tonality flags among the four highest bands and necessary bits (1 or 2 bits) are allocated to each of the identified bands.

Finally, remaining bits are allocated to other bands based on perceptual importance using the decoded quantized band energies. When there is any band whose assigned bit results in zero in the four bands, such band is re-identified as a PFSC encoding band and the bit allocations are re-calculated.

6.2.3.1.3.1.4   Fine structure decoding

6.2.3.1.3.1.4.1   TCQ decoding

6.2.3.1.3.1.4.1.1   Joint USQ and TCQ

In order to de-quantize the fine structure of the normalized spectrum, the ISC and the information for the selected ISCs in each band are decoded by the position, number, sign and magnitude of the ISCs.

The magnitude information is decoded by the joint USQ and TCQ with an arithmetic decoding, while the position, number and sign are decoded by an arithmetic decoding.

The decoding method is selected at the Selecting Decoding Method block by the bit allocation and the information for each band. If a bit allocated for a band is zero, all the samples in the band are decoded to zero by the zero decoding block. Otherwise, each band is decoded by the selected de-quantizer.

The quantizer selection information selects between the TCQ and USQ quantizes to get the same results as that of encoder.

The Estimating Number of Pulses block determines the number of pulses per a band using the band length and the bit allocation data R[]. Its principle of operation is same as that of the method which is used in the Scaling Bands module in encoder, see subclause 5.3.4.1.4.1.5.1.1.

The Lossless Decoding and the Decoding Position Info block reconstruct the position information of the ISCs, i.e. the number of ISCs and their positions. This process is similar to encoder side and same probabilities should be used for the proper decoding, see subclause 5.3.4.1.4.1.5.1.

In the Joint USQ and TCQ Decoding block, the magnitudes of the gathered ISCs are decoded by the arithmetic decoding and de-quantized by the joint USQ and TCQ decoding. In this block the non-zero position and the number of the ISC is utilized for the arithmetic decoding. The joint USQ and TCQ have two types of decoding methods. One is TCQ and USQ with 2nd bit allocation for the NB and WB, and the other one is the LSB TCQ for USQ for the SWB and FB. These methods are described in subclause 6.2.3.1.3.1.4.1.2 TCQ and USQ with second bit allocation and 6.2.3.1.3.1.4.1.3 LSB TCQ for USQ.

In the Decoding Signs block, the sign information of the selected ISC is decoded by the arithmetic decoding with equal probabilities for the positive and negative signs.

In order to recover the quantized components for each band, the position, sign and magnitude information is added to the quantized components to recover the real components at the Recovering Quantized Components block.

At this point the determined bands with no transmitted data are filled by zeroes. Then the number of pulses in the non-zero bands is estimated and the position information, including the number and position of ISCs, is decoded using this estimated number. After the magnitude information is decoded using the lossless decoder, the joint USQ and TCQ decoding is performed. For non-zero magnitude values the signs and quantized components are finally reconstructed.

In the Inverse Scaling Bands block, the inverse scaling of the quantized components is performed by using the transmitted norm information. The inverse scaled signal is the output of the TCQ decoding.
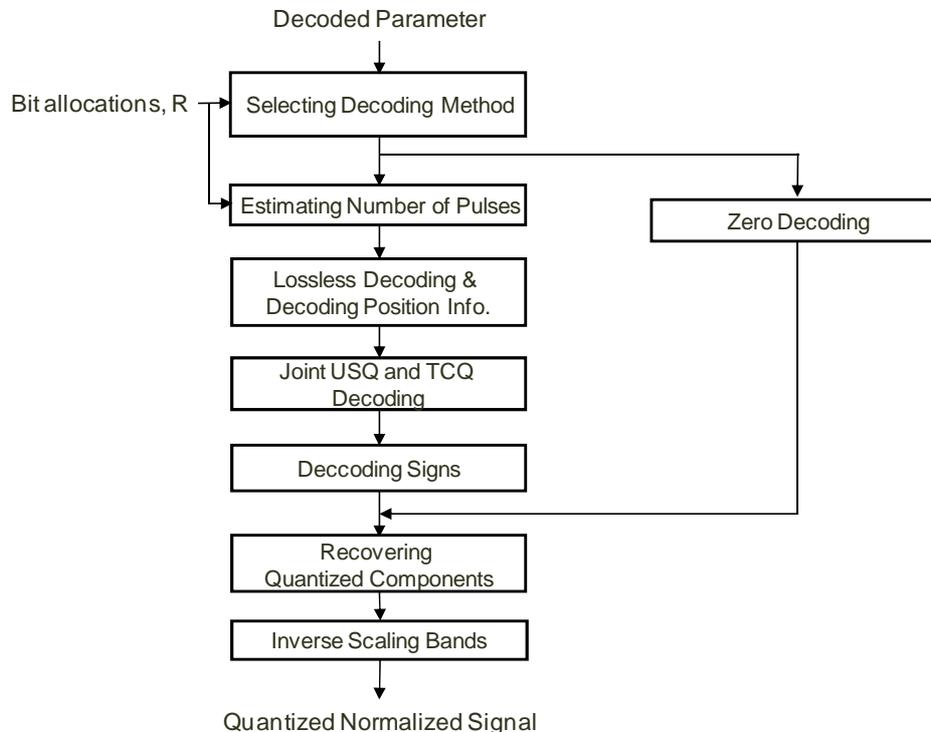
Figure 95: Block diagram of fine structure decoding using TCQ

6.2.3.1.3.1.4.1.2   TCQ and USQ with second bit allocation

The general de-quantization and decoding scheme of the TCQ and USQ with second bit allocation consists of several main blocks: quantizer decision, TCQ decoder, USQ decoder, lossless decoder, and Second bit allocation. In the

quantizer decision module the quantization mode of the current band is selected by using the results of the Selecting Decoding Method block. Then the selected decoder restores the current band in association with the lossless decoder, based on the arithmetic decoding with the transmitted bit stream.
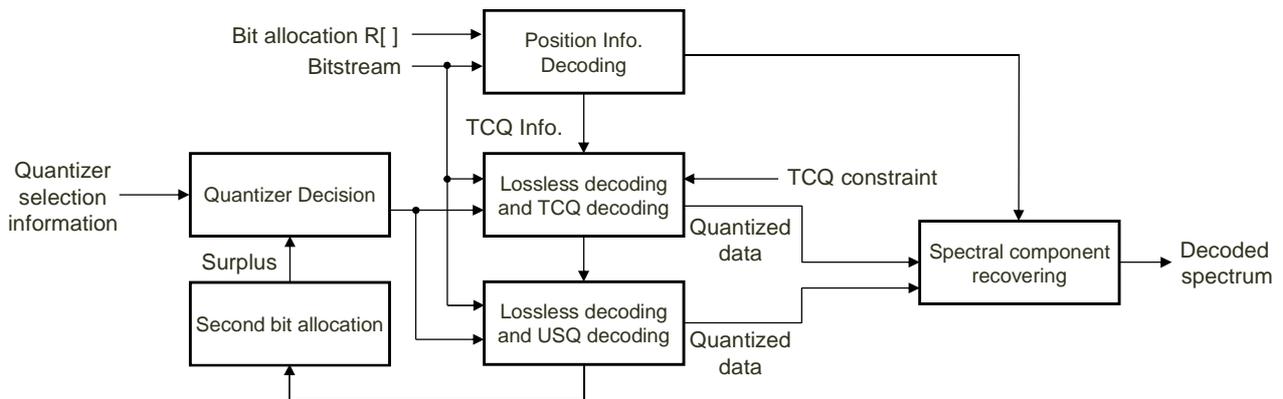
Figure 96: Block diagram of TCQ and USQ decoding with second bit allocation

The decoding process is started by the recovering the non-zero bands and positions using the transmitted bit-stream and the bit allocation R[] for the selected quantizer. By using this information, the appropriate magnitude for the decoded band is selected. The difference between bit allocation R[] and actual decoded bits per band is accumulated and called the surplus. This surplus will be used while decoding two band determined by second bit allocation procedure described in encoder in subclause 5.3.4.1.4.1.5.1.2.

The magnitude decoding based on binary arithmetic decoding is as follows. First the probability of symbol is calculated by the equations in encoder subclause 5.3.4.1.4.1.5.1.2. Then the number of pulses for each magnitude is decoded by using the probabilities $\hat{p}_1$ and $\hat{p}_0$, where $\hat{p}_1$ corresponds to last pulse in magnitude and $\hat{p}_0$ to all other pulses. The magnitude of the pulse probabilities are then modified after this calculation with respect to the trellis code limitation, i.e. magnitudes that are impossible are assigned zero probability.

This algorithm was modified to save complexity for bands with a large number of pulses. The procedure is same as that of encoder subclause 5.3.4.1.4.1.5.1.2.

Location decoding is done based on the same algorithm as that of magnitudes decoding and uses the same complexity reduction technique.

Signs are decoded with the arithmetic decoder, using equal probabilities of positive and negative signs.

6.2.3.1.3.1.4.1.3 LSB TCQ for USQ

The idea of the LSB TCQ for USQ is to use advantages of both quantizers (USQ and TCQ) in one scheme and exclude the path limitation from the TCQ.
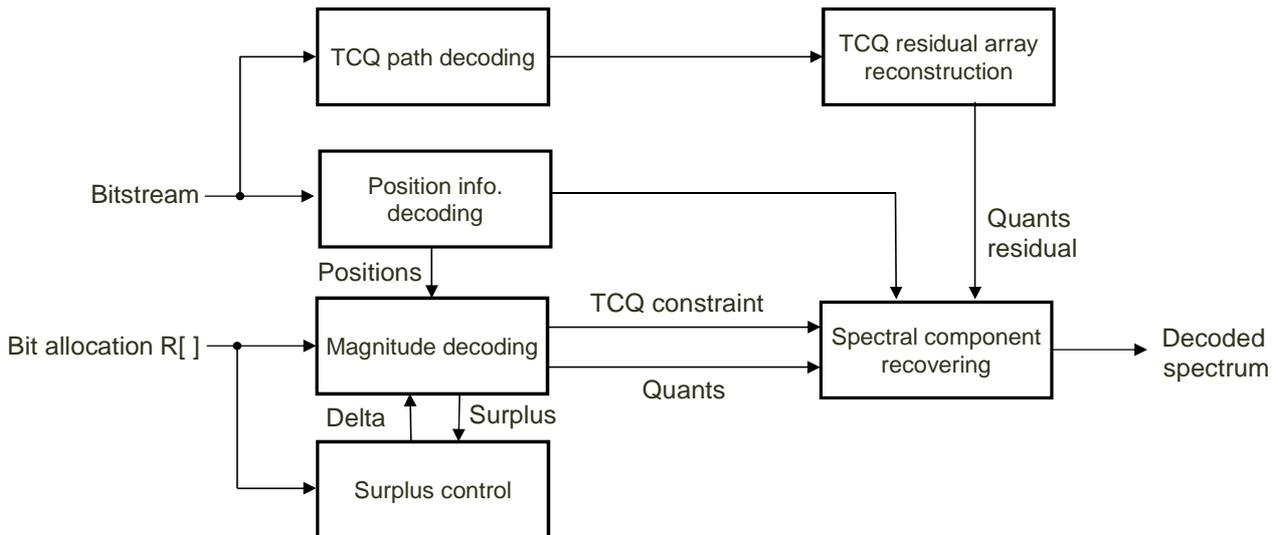
Figure 97: Block diagram for LSB TCQ decoding

The decoding process starts from receiving the bit allocation R[] and the decoding of the band information including:

- Number of nonzero positions for ISCs

- Nonzero positions

- USQ magnitude

- Signs for nonzero magnitudes

First the number of nonzero pulses and their positions are decoded using the arithmetic decoder. Then the USQ magnitudes are decoded band by band using bit allocation with surplus control. This generates Delta values in the same manner as the encoder, see subclause 5.3.4.1.4.1.5.1.3. The difference between the bit allocation R[] and actual decoded bits per band is accumulated and called the surplus, which is then used in the next bands.

The algorithms used for decoding positions and magnitudes are the same as those described in subclause 6.2.3.1.3.1.4.1.2 in TCQ and USQ decoder.

After receiving the USQ magnitudes, the TCQ path is decoded from the bit-stream using the arithmetic decoder.

The decoded path is used to reconstruct the residual array according to the decoded trellis state. From each path bit, two LSB bits are generated in the residual array. This process shown in pseudo code:

```
for( state = 0, i = 0; i < bcount; i++)
{
    residualbuffer[2*i]     = dec_LSB[state][dpath[i]] & 0x1;
    residualbuffer [2*i + 1] = dec_LSB[state][dpath[i]] & 0x2;
    state  = trellis_nextstate[state][dpath[i]];
}
```

Starting from *state* 0, the decoder moves through the trellis using decoded *dpath* bits, and extracts two bits corresponding to the current trellis edge.

In the Spectrum recovering block the decoded residual array is added to the non-zero spectral components. The output of this block is the reconstructed spectrum.

The decoded MDCT coefficients are de-normalized using the decoded band energies.

Finally, as described in subclause 5.3.4.1.4.1.4.4.1, fine gain adjustment is performed on the dominant bands. Decoded fine gain adjustment factor is applied to the de-normalized decoded MDCT coefficients.

6.2.3.1.3.1.4.2   Noise-filling

The processing is the same with the one at the encoder side.

6.2.3.1.3.1.4.3   PFSC decoding

6.2.3.1.3.1.4.3.1   Envelope normalization

This process is the same with the one described in subclause 5.3.4.1.4.1.5.3.2.

6.2.3.1.3.1.4.3.2   Lag information decoding

Lag indices for the last four sub-bands (i.e. b=18 to 21 in 13.2 kbps and b=20 to 23 in 16.4 kbps) are decoded if the corresponding decoded tonality flag is set to "0". The starting position is decoded as $k^i + lag^i[k']$ as described in subclause 5.3.4.1.4.1.5.3.3. Based on the starting position and width of search band, the predicted high-frequency spectrum is generated from the envelope normalized TCQ-decoded low-frequency spectrum.

6.2.3.1.3.1.4.3.3   Scaling and noise smoothing

Scaling factors are calculated for the predicted bands using the decoded band energies. Each scaling factor is calculated as the square root of the quotient of the quantized band energy divided by its corresponding band energy from the predicted high-frequency spectrum. The calculated scaling factors are attenuated by the scaling factor of 0.9 and applied to the predicted high-frequency spectrum.

Inter-frame smoothing process for the noise components are applied as described in subclause 5.3.4.1.4.1.5.3.3.3.

The Normal mode PFSC decoding overview is shown in figure 98.
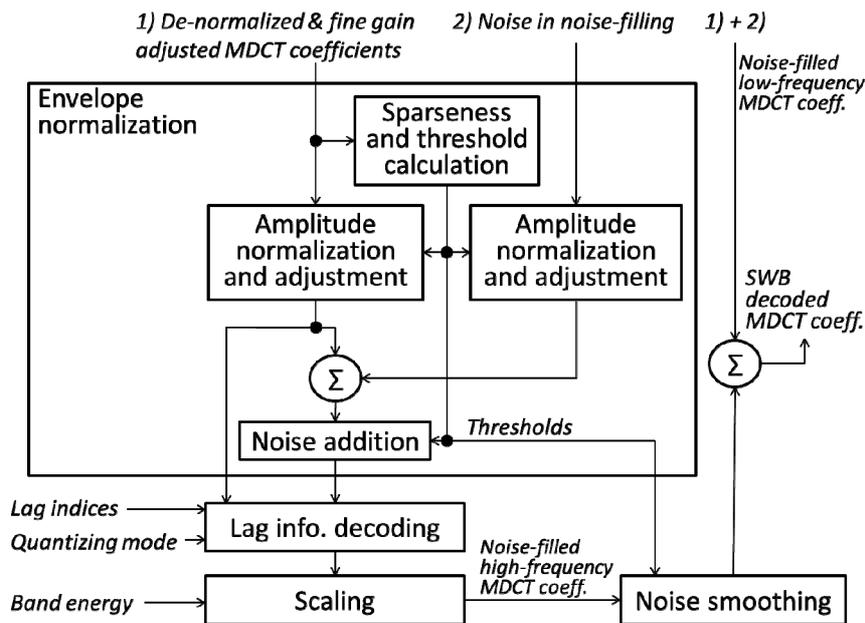
Figure 98: Block diagram of the Normal mode PFSC decoder

6.2.3.1.3.2   Transient Mode

6.2.3.1.3.2.1   Energy envelope decoding

Details are described in subclause 6.2.3.1.2.

6.2.3.1.3.2.2   Bit allocation

The processing is the same as subclause 5.3.4.1.4.2.2

6.2.3.1.3.2.3   Fine structure decoding

TCQ decoding with Transient mode configurations is performed.

6.2.3.1.3.3   Harmonic Mode

6.2.3.1.3.3.1   Overview

The high-level decoder structure of the Harmonic mode is basically the same with the Normal mode. The main difference can be found in its detailed structure of the PFSC block, and it is shown in the following figure.
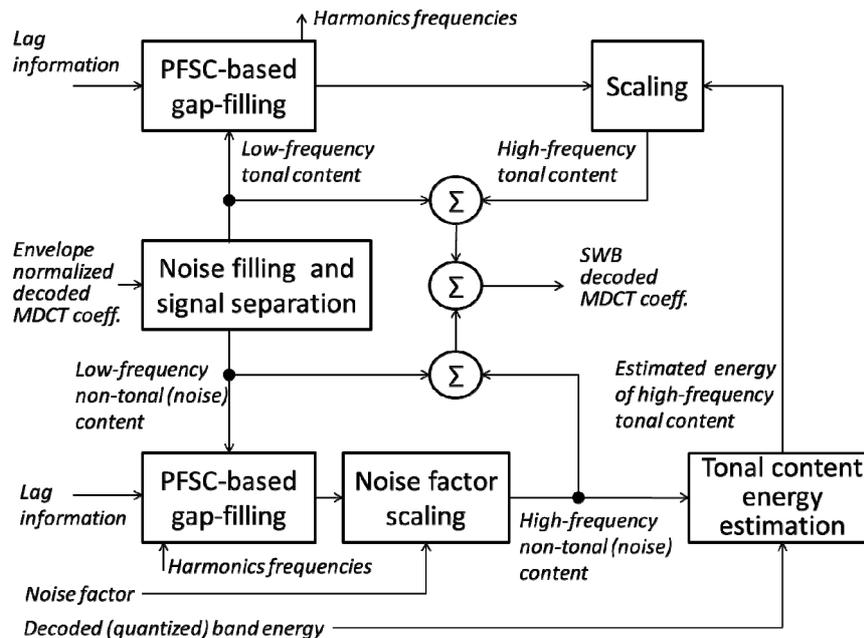
Figure 99: Block diagram of the Harmonic mode decoder overview

6.2.3.1.3.3.2   Energy envelope decoding

Details are described in subclause 6.2.3.1.2.

6.2.3.1.3.3.3   Bit allocation

The processing is the same as subclause 6.2.3.1.3.1.3.

6.2.3.1.3.3.4   Fine structure decoding

6.2.3.1.3.3.4.1   TCQ decoding

This part is the same with the Normal mode as described in subclause 6.2.3.1.3.1.4.1.

6.2.3.1.3.3.4.2   Noise filling for quantized spectrum

In this subclause noise is filled in the quantized spectrum where coefficients have been quantized to zero when the bit allocation subclause allocates non zero bits to the bands and also fills the un quantized bands up to the transition frequency f_t in the same manner as in the encoder, see subclause 5.3.4.1.4.3.3.2

6.2.3.1.3.3.4.3　　　PFSC-based gap filling

6.2.3.1.3.3.4.3.1　　　Overview

This subclause is only applied to SWB and FB input signals. The spectral coefficients which belong to bands which are assigned zero bits from the bit-allocation subclause are not quantized. This means that not all transform coefficients are transmitted to the decoder. From the noise filled quantized spectrum, the gaps in the high frequency region $b = N_{bands} - 4,..,N_{bands} - 1$ which has zero bit allocation are identified and are filled with the new generated spectrum. The predicted spectrum is generated using normalized noise filled quantized spectrum described in subclause 6.2.3.1.3.3.4.3.2.

Based on the bit allocation described in subclause 6.2.3.1.3.3.3, if any of $R(b)$, $b = N_{bands} - 4,..,N_{bands} - 1$ is allocated with zero bits, the corresponding band with start and end positions k_{start}, k_{end} according to table 108 in the $\hat{X}_M(k)$ has a gap and it is filled with the predicted spectrum described in subclause 6.2.3.1.3.3.4.3.5 corresponding to k_{start}, k_{end} .in $\tilde{H}_M(k)$

6.2.3.1.3.3.4.3.2　　　Envelope Normalization

The envelope normalization is performed equally as in the encoder, described in subclause 5.3.4.1.4.3.3.2. As a result the envelope normalized signal $\tilde{X}'_M(k) = \tilde{X}_M(k) + \tilde{N}_M(k)$ is obtained, where $\tilde{X}_M(k)$ is the envelope normalized low frequency quantized spectrum and $\tilde{N}_M(k)$ is the envelope normalized low frequency noise spectrum.

6.2.3.1.3.3.4.3.3　　　Decoding of lag index

Lag index $LagIndex^i$ for sub-bands i=0,1 is decoded from the bit stream. For sub-bands 0 and 1, the starting position k^i using lag index $LagIndex^i$ is defined in equation (1147) and end of the desired portion k^i_{end} is defined equation (1148).

Based on the starting position and width of search band the predicted spectrum is generated from the envelope normalized noise filled quantized spectrum. The detailed description of the predicted spectrum generation is described in following subclause 6.2.3.1.3.3.4.3.5.

6.2.3.1.3.3.4.3.4　　　Structure analysis for Harmonics

The structure analysis for Harmonic mode is performed equally as in the encoder, described in subclause 5.3.4.1.4.3.3.3.4. As a result estimated harmonic Est_{freq_2} is obtained; the estimated harmonic is used for generating the predicted spectrum for the HF region

6.2.3.1.3.3.4.3.5　　　Predicted spectrum generation

Predicted spectrum $\tilde{H}_M(k)$ is generated for the high frequency region by using the envelope normalized noise-filled quantized spectrum $\tilde{X}_M(k), \tilde{N}_M(k)$, which is obtained from subclause 6.2.3.1.3.3.4.3.2. Predicted spectrum is generated, first by extracting the desired noise components from the $\tilde{N}_M(k)$ described in subclause 6.2.3.1.3.3.4.3.6 followed by tonal generation using $\tilde{X}_M(k)$ described in subclause 6.2.3.1.3.3.4.3.7.

Noise filled spectrum $\tilde{H}_M(k)$ is used for estimating the tonal energy $E'_M(b), b = N_{bands} - 4,..,N_{bands} - 1$ and the tonal components $pul^i[l]$ extracted from the normalized quantized spectrum are normalized using the estimated tonal energy $E'_M(b)$, where $E'_M(b)$ is calculated as follows:

$$E'_M(b) = 2^{E_M(b)} - E_{Noise}(b) \quad b = N_{bands} - 4,...,N_{bands} - 1 \tag{1810}$$

$E_{Noise}(b)$: is the noise energy obtained using the noise filled spectrum $\tilde{H}_M(k)$ according to

$$E_{Noise}(b) = \sum_{k=k_{start}(b)}^{k_{end}(b)} \tilde{H}_M(k)^2 \qquad (1811)$$

The noise energy obtained from equation (1811) is adjusted, when the noise filled spectrum has low level noise and / or when the noise filled spectrum $\tilde{H}_M(k)$ has high level noise. Low noise level is detected using the energy ratio $E_{ratio}(b)$ between noise and the total band energy and high noise level is detected when the estimated tonal energy $E'_M(b)$ is negative. The adjustment factor $N_{gain}(b)$ is estimated according to

$$
\begin{aligned}
&for\, b = N_{bands} - 4\, to\, N_{bands} - 1 \\
&\quad E_{ratio}(b) = \frac{E_{Noise}}{2^{E_M(b)}} \\
&\quad if\, E_{ratio}(b) < 0.06 \\
&\quad\quad fac = 0.6 \\
&\quad\quad if\, pul^b{}_{res} \mathrel{!}= 0 \\
&\quad\quad\quad fac = \sqrt{\frac{E_{Noise}}{k_{width}(b)\, pul^b[0]}} \\
&\quad\quad end \\
&\quad\quad n_g = fac \sqrt{\frac{2^{E_M(b)}}{k_{width}(b)}} \\
&\quad\quad if\, E_{ratio} n_g{}^2 >= 0.12 \\
&\quad\quad\quad n_g \leftarrow 0.05 n_g \\
&\quad\quad end \\
&\quad\quad N_{gain}(b) = \max(n_g, 1.4) \\
&\quad\quad \tilde{H}_M(k) \leftarrow N_{gain}(b)\tilde{H}_M(k), \quad k = k_{start}(b),..,k_{end}(b) \\
&\quad end \\
&\quad E'_M(b) = 2^{E_M(b)} - E_{Noise}(b) \\
&\quad\quad if\, E'_M(b) < 0 \\
&\quad\quad\quad N_{gain}(b) = 0.25 \\
&\quad\quad\quad \tilde{H}_M(k) \leftarrow N_{gain}(b)\tilde{H}_M(k), \quad k = k_{start}(b),..,k_{end}(b) \\
&\quad\quad end \\
&end
\end{aligned}
\qquad (1812)
$$

For each band, based on the $N_{gain}(b)$ obtained from equation (1812) is used to re-calculate the tonal energy $E'_M(b)$ and estimated noise $E_{Noise}(b)$ using equations (1810) and (1811). The tonal components $pul^b(l)$ extracted from envelope-normalized quantized spectrum is normalized using the scale factor $ton_{fac}(b)$ calculated as follows

$$ton_{fac}(b) = \sqrt{\frac{E'_M(b)}{\sum_{l=0}^{l=pul^b_{res}-1} pul^b(l)^2}}, \quad b = N_{bands} - 4,..., N_{bands} - 1 \qquad (1813)$$

The calculated scale factor $ton_{fac}(b)$ and extracted tonal components are used for injecting the tonal components into the noise filled spectrum $\tilde{H}_M(k)$ according to

$$b = N_{bands} - 4\, to\, N_{bands} - 1$$
$$j = 0$$
$$l = 0$$
$$while\, k_{start}(b) < pk_{pos}(j) < k_{end}(b)\, AND\, j < res_{tot}$$

$$\tilde{H}_M(pk_{pos}(j)) = pul^b(l)ton_{fac}(b)\sqrt{\frac{E'_M(b)}{2^{E_M(b)}}} \tag{1814}$$

$$j = j + 1$$
$$l = l + 1$$
$$end$$

where, $pk_{pos}(j)$ $j = 0,..,res_{tot}$ is the tonal positions obtained from subclause 5.3.4.1.4.3.3.3.5

6.2.3.1.3.3.4.3.6   Noise filling for the predicted spectrum

Noise filling for the predicted spectrum is performed equally as in the encoder, described in subclause 5.3.4.1.4.3.3.3.5. As a result noise filled spectrum $\tilde{H}_M(k)$ and tonal positions $pk_{pos}[j]$ is obtained, where j is the pulse resolution. The obtained predicted spectrum which contains noise is adjusted using the noise factor $\hat{N}_{fac}$ according to

$$\tilde{H}_M(k) \leftarrow \tilde{H}_M(k)\hat{N}_{fac}, \quad k = hf_{start},...,hf_{width} - 1 \tag{1815}$$

where $\hat{N}_{fac}$ is the noise factor which is decoded from the bit stream and the decoded noise factor is converted to linear domain as follows

$$\hat{N}_{fac} \leftarrow 10^{\hat{N}_{fac}} \tag{1816}$$

6.2.3.1.3.3.4.3.7   Signal generation for predicted spectrum

First, the tonal components $pul^i(l)$ is extracted from the desired portion of envelope normalized quantized spectrum $\tilde{X}_M(k)$ which is described below. The start position k^i and the end k^i_{end} of each desired portion in the normalized quantized spectrum $\tilde{X}_M(k)$, $k = k^i,....k^i_{end}$ is the same as equations (1147) and (1148): As the normalized quantized spectrum characteristics are flat all the values during the normalization process will have similar values, all the non-zero coefficients in the desired region of $\tilde{X}_M(k)$ is identified as follows

$$i = 0\, to\, 1$$
$$l = 0$$
$$pos = k^i$$
$$while\, j < SB^i_{width}$$
$$\quad j = j + 1$$
$$\quad if\, \tilde{X}_M(A) != 0$$
$$\quad\quad pul^i(l) = \tilde{X}_M(A)$$
$$\quad\quad l = l + 1$$
$$\quad end$$
$$end$$
$$pul^i_{res} = l$$
$$A = \begin{cases} pos + j & i = 0 \\ pos - j & i = 1 \end{cases}$$

where, $pul^i_{res}, pul^i(l)$ are defined as follows

pul_{res}^i is the tonal resolution obtained from the normalized quantized spectrum for sub band $i=0,1$

$pul^i(l)$ is the tonal components extracted from the normalized quantized spectrum for sub band $i=0,1$

The tonal information $pul_{res}^i, pul^i(l)$, for $i=0, 1$ obtained from normalized quantized spectrum is used for sub band $i=2, 3$.

Based on the band definition described in table 108, the high frequency band ranges are defined $b=N_{bands}-4,..,N_{bands}-1$. Using the band definitions for high frequency region, the extracted tonal components and its corresponding pulse resolutions are restructured, now the restructured $pul_{res}^b, pul^b(l)$ is used for generating predicted spectrum. For example, the restructured information for sub band $i=0$ is equivalent to $b=N_{bands}-4$

6.2.3.2 High-rate HQ decoder

A high level structural block diagram of the high-rate HQ decoder is in figure 100.
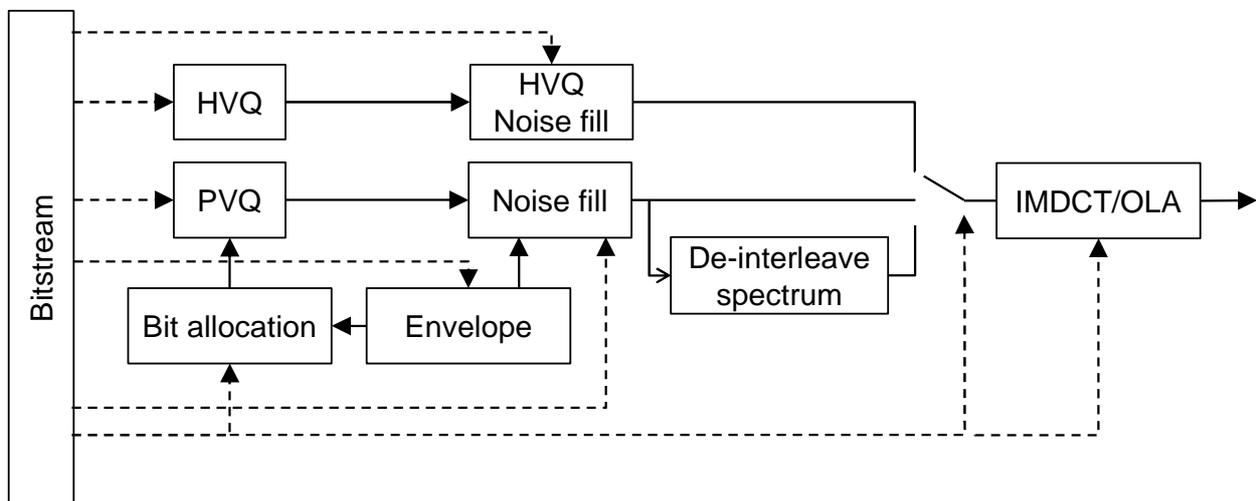
Figure 100: High level structure of the high-rate HQ decoder

Firstly, the High-rate HQ coding mode information is decoded.

6.2.3.2.1 Normal Mode

6.2.3.2.1.1 Envelope decoding

From the received high-rate HQ norm coding mode bits, the high-rate HQ norm coding mode is determined and the transmitted differential indices are decoded using the selected method. The quantization index of the lowest-frequency band, i.e., $I_M(0)$, is directly decoded in all modes.

6.2.3.2.1.1.1 Context based Huffman decoding mode

If this coding mode is determined for the current frame, the context based Huffman decoding is then performed on the transmitted quantization differential indices using the method described in subclause 6.2.3.1.2.1.1 and the tables shown in 168 and 169.

6.2.3.2.1.1.2 Re-sized Huffman decoding mode

If this coding mode is determined for the current frame, the resized Huffman decoding is then performed on the transmitted quantization differential indices using the method described in subclause 6.2.3.1.2.1.2. The Huffman codes for the differential indices are given in table 105.

6.2.3.2.1.1.3 Normal Huffman decoding and bit-packing mode

If this coding mode is determined for the current frame, the Normal Huffman decoding is then performed on the transmitted differential indices. The Huffman codes for the differential indices are given in subclause 5.3.4.2.1.2.3.

When the bit-packing mode is determined; the adjusted differential indices are un-packed directly with 5 bits.

The actual quantized norms $\hat{E}_M(b)$ are obtained by lookup table, defined in subclause 5.3.4.2.1.1.

6.2.3.2.1.2 Normal mode fine structure inverse quantization

6.2.3.2.1.2.1 Fine structure inverse PVQ-quantization

The spectral coefficient inverse quantization is done as is described in subclause 6.2.3.2.6

6.2.3.2.1.2.2 Fine gain prediction, inverse quantization and application

The bit allocation for the PVQ shape vector $R_{PVQ}(b)$ and fine gain adjustment $R_{FG}(b)$, as well as $g_{pred}(b)$ and $g_{rms1}(b)$ are obtained as in subclause 5.3.4.2.1.3a.1. The quantized gain prediction error $\hat{g}_{err}(b)$ is obtained by using the assigned bitrate $R_{FG}(b)$ and the fine gain adjustment $g_{fg}(b)$ is obtained by

$$g_{fg}(b) = \hat{g}_{err}(b)g_{rms1}(b)g_{pred}(b) \tag{1817}$$

with $\hat{g}_{err}(b) = 1$ for $R_{FG}(b) = 0$. The gain of the synthesis is adjusted by scaling the decoded fine structure with the fine gain $g_{fg}(b)$.

6.2.3.2.1.3 Spectral filling

This subclause gives a technical overview of the spectrum filling processing which is applied at the decoder in HQ high rate mode.

6.2.3.2.1.3.1 Wideband adaptive noise filling at 24.4/32kbps

Wideband adaptive noise filling at 24.4 and 32 kbps proceeds by calculating the total available bits and the bits variance for the sub-bands in non-transient frames over the index range $b = [8, b_{last\_sfm}]$,

$$bit\_tot = \sum_{i=8}^{b_{last\_sfm}} R(i) \tag{1818}$$

$$bit\_var = \frac{\sum_{i=8}^{b_{last\_sfm}} |R(i) - R(i-1)|}{bit\_tot} \tag{1819}$$

The average bit allocation threshold $THR_{bit}[b]$ is initialized for each coefficient in each sub-band according to the values in table 171.

Table 171: Threshold for average bit allocation

Band	0	1	2	3	4	5	6	7	8	9	10	11	12
$THR_{bit}[b]$	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Band	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
$THR_{bit}[b]$	1.5	1.5	1.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.8

For the sub-bands in the index range $b = [8, b_{last\_sfm}]$, N_{sat} denotes the number of the sub-bands where the average number of allocated bits for each coefficient is not less than the threshold $THR_{bit}[b]$. The harmonic parameter S_M for those sub-bands is calculated as follows:

$$S_M = \sum_{b=8, R(b)/L_M(b) >= THR_{bit}(b)}^{b_{last\_sfm}} \frac{L_M(b) * \tilde{X}_{max}(b)}{\sum_{k=k_{start}[b]}^{k_{end}[b]} \tilde{X}_M(k)} \tag{1820}$$

$$S_M = \frac{2 * N_{sat}}{S_M} \tag{1821}$$

The step length, *step*, is calculated according to:

$$step = \frac{5 * S_M}{b_{last\_sfm}} \tag{1822}$$

For any sub-band in non-transient frames the following procedure is then followed. If the average number allocated bits for each coefficient in the sub-band is greater than or equal to the threshold 1.5, then the bit allocation for the sub-band is saturated and the un-decoded coefficients of the sub-band are not processed further by the noise filling. Otherwise, the bit allocation to the sub-band is un-saturated, and the un-decoded coefficients of the sub-band are reconstructed by noise filling. For any un-saturated sub-band with zero bits allocated to its coding, the envelope of the un-decoded coefficients in the sub-band are set to the decoded norm for that sub-band. Otherwise, if the un-saturated sub-band has bits allocated to it, the envelope of the un-decoded coefficients is calculated as follows:

The average energy of the sub-band $E_{av}(b)$ is then calculated using the de-quantized norm $\tilde{I}_N^q(b)$ as follows:

$$E_{av}(b) = \tilde{I}_N^q(b) * \tilde{I}_N^q(b) * L_M(b) \tag{1823}$$

The energy sum of the all decoded non-zero coefficients in this subband $E_{sub}(b)$ is then calculated

$$E_{sub}(b) = \sum_{k=k_{start}(b)}^{k_{end}(b)} \tilde{X}_M(k) * \tilde{X}_M(k), \qquad \tilde{X}_M(k) \neq 0 \tag{1824}$$

A search of the maximum magnitude $\tilde{X}_{max}$ and the minimum magnitude $\tilde{X}_{min}$ of the decoded coefficients in each subband is also calculated for further processing.

The energy difference $E_{diff}(b) = E_{sub}(b) - E_{av}(b)$ is next calculated. If $E_{diff}(b) \leq 0$, the envelope of the un-decoded coefficients is set to zero $\tilde{I}_{nf}(b) = 0$. Otherwise, the envelope of the un-decoded coefficients $\tilde{I}_{nf}(b)$ is calculated as follows:

The initial envelope $I_{nf}(b)$ of the un-decoded coefficients in the un-saturated sub-band is calculated by the energy difference,

$$I_{nf}(b) = \sqrt{E_{diff}(b)/L_M(b)} \tag{1825}$$

The average norm of the un-saturated sub-band $\tilde{I}_{av}^q$ is calculated

$$\tilde{I}_{av}^q = \begin{cases} (\tilde{I}_N^q(0) + \tilde{I}_N^q(1) + \tilde{I}_N^q(2))/3, & b = 0 \\ (\tilde{I}_N^q(b-1) + \tilde{I}_N^q(b) + \tilde{I}_N^q(b+1))/3, & 0 < b < N_{bands} - 1 \\ (\tilde{I}_N^q(N_{bands}-3) + \tilde{I}_N^q(N_{bands}-2) + \tilde{I}_N^q(N_{bands}-1))/3, & b = N_{bands} - 1 \end{cases} \tag{1826}$$

If $bit\_var > 0.3$ or $4 * \tilde{I}_N^q(b) < \tilde{X}_{\max}$, then the spectrum of the sub-band is *sharp*. The envelope of the un-decoded coefficients is obtained by modifying the initial envelope as follows:

$$\tilde{I}_{nf}(b) = \begin{cases} I_{nf}(b) * 3 * S_M * \tilde{I}_{av}^q * / \tilde{X}_{\max}, & \text{if } bit\_var > 0.3 \text{ or } 4 * \tilde{I}_N^q(b) < \tilde{X}_{\max} \\ I_{nf}(b) * S_M, & \text{otherwise} \end{cases} \tag{1827}$$

If the spectrum of the sub-band is not *sharp* and $\tilde{X}_{\max} > 2.5 * \tilde{I}_N^q(b)$, $\tilde{I}_{nf}(b) = \tilde{I}_{nf}(b) * \tilde{I}_{nf}(b) / \tilde{X}_{\max}$, then the harmonic parameter S_M is added by the step length *step* .

If the envelope is more than the half of the minimum magnitude $\tilde{X}_{\min}$, then the envelope is set to be equal to the half of the minimum magnitude $\tilde{X}_{\min}$.

$$\tilde{I}_{nf}(b) = 0.5 * \tilde{X}_{\min}, \qquad \text{if } \tilde{I}_{nf}(b) > 0.5 * \tilde{X}_{\min} \tag{1828}$$

If the ratio of the average norms $\tilde{I}_{av}^q$ in the current frame to that of the previous frame lies in the range (0.5,…, 2), and the previous frame is a non-transient frame, the the envelope of the un-decoded coefficients for the current frame and the previous frame are weighted as follows.

$$\tilde{I}_{nf}(b) = 0.5 * \tilde{I}_{nf}(b) + 0.5 * \tilde{I}_{nf}^{(-1)}(b) \tag{1829}$$

For the un-decoded coefficients in the sub-band, the coefficients are generated using a random noise generator and multiplied by the estimated envelope as described above.

For the last sub-band, a check is made whether the mode of the previous frame was not a transient, and whether the ratio of the decoded norms of the current frame to those of the previous frame are in the range (0.5, …,2.0), and the bit variance *bit_var* is not more than 0.3, and the sub-band of the current frame is bit allocated and the sub-band of the previous frame is not bit allocated or vice versa. If all of the above conditions are fulfilled, then the coefficients in the current frame and the previous frame for the last 20 coefficients in the last sub-band are weighted as follows:

$$\tilde{X}_M(k) = sign(\tilde{X}_M(k)) * (0.5 * |\tilde{X}_M(k)| + 0.5 * |\tilde{X}_M^{(-1)}(k)|),$$
$$\text{if } |\tilde{X}_M(k)| > 4.0 * |\tilde{X}_M^{(-1)}(k)| \text{ or } |\tilde{X}_M(k)| < 0.25 * |\tilde{X}_M^{(-1)}(k)|, 300 \le k < 320 \tag{1830}$$

In the transient mode, un-decoded coefficients in a sub-band are generated from random noise, and de-normalized by the decoded norm for that sub-band.

6.2.3.2.1.3.2   General spectral filling

Based on the received bit-allocation, the transition frequency f_t is estimated in the same manner as in the encoder, see subclause 5.3.4.2.1.4. Spectral filling consists of two algorithms. The first algorithm fills the low-frequency spectrum up to the transition frequency f_t , the second algorithm regenerates the possibly non-coded high-frequency components by using the low-frequency noise-filled spectrum.

The interaction between these two algorithms is shown in figure 101. The resulting spectrum from both the noise-filling algorithm and the high frequency noise fill is a normalized spectrum which is shaped by the received quantized norms.
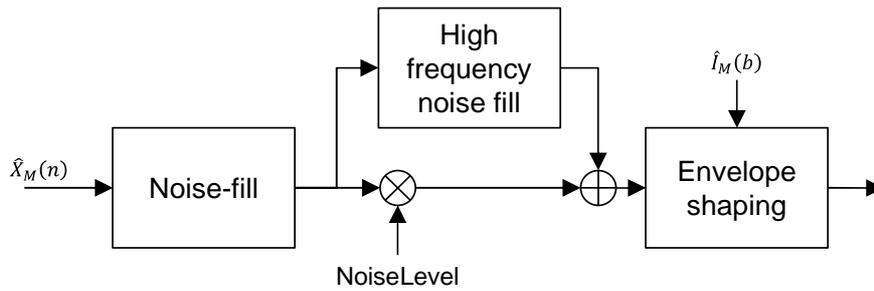
Figure 101: Spectrum filling block diagram

6.2.3.2.1.3.2.1 Noise filling

The first step of the noise fill procedure relies on the building of the so-called spectral codebook from the received (decoded) normalized transform coefficients. This step is achieved by concatenating the perceptually relevant coefficients of the decoded spectrum. Figure 102 illustrates this procedure. The decoded spectrum has several series of zero coefficients that are called spectral holes of a certain length. This length is the sum of the consecutive lengths of bands which were allocated zero bits.
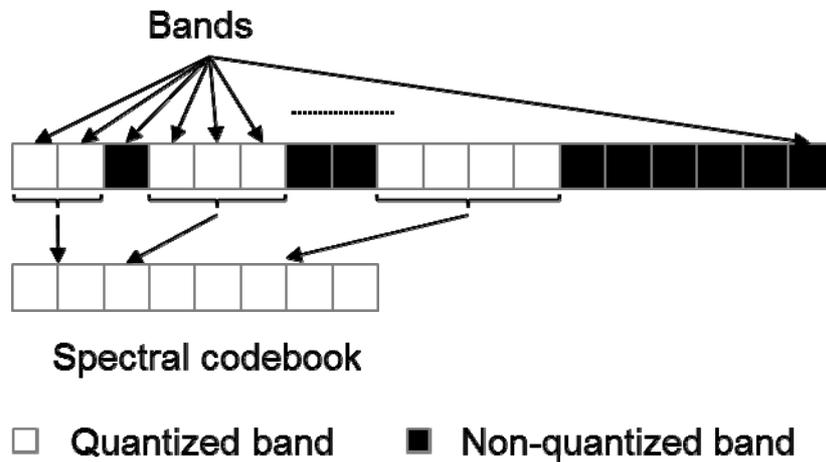
Figure 102: Building the spectral codebook from the decoded transform signal

Since the length of all spectral holes can be higher than the length of the spectral codebook, the codebook elements might be re-used for filling several spectral holes.
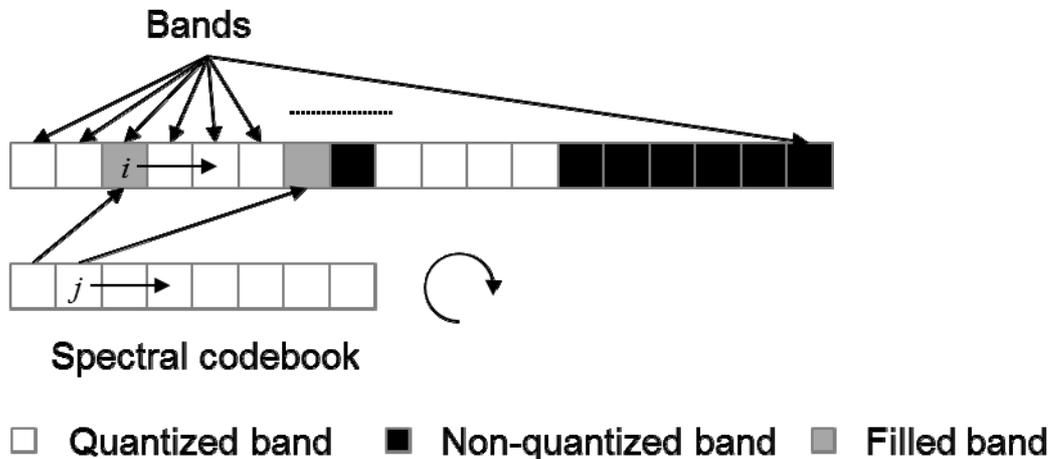
Figure 103: Noise filling from the spectral codebook up to the transition frequency

Figure 103 shows how, based on the spectral codebook C, the non-quantized spectral coefficients are filled. Spectral holes are filled by increasing the codebook index j as much as the index i, used to cover all the spectral holes up to the transition frequency. Reading from the spectral codebook is done sequentially and as a circular buffer according to the following:

```
i=0; j=0
(1:) if  R(b_i)=0 then  X̂_M(i)=C(j) ,
     increment i,j (if out of bound, rewind j to start of codebook)
     if i=0 then
     STOP
     else
     goto (1:)
     endif
```

For low bit rates, many of the quantized bands will contain few pulses and have a sparse structure. For signals which require a more dense and noise-like fill, a set of two anti-sparseness processed codebooks are created instead of the regular spectral codebook as illustrated in figure 104.
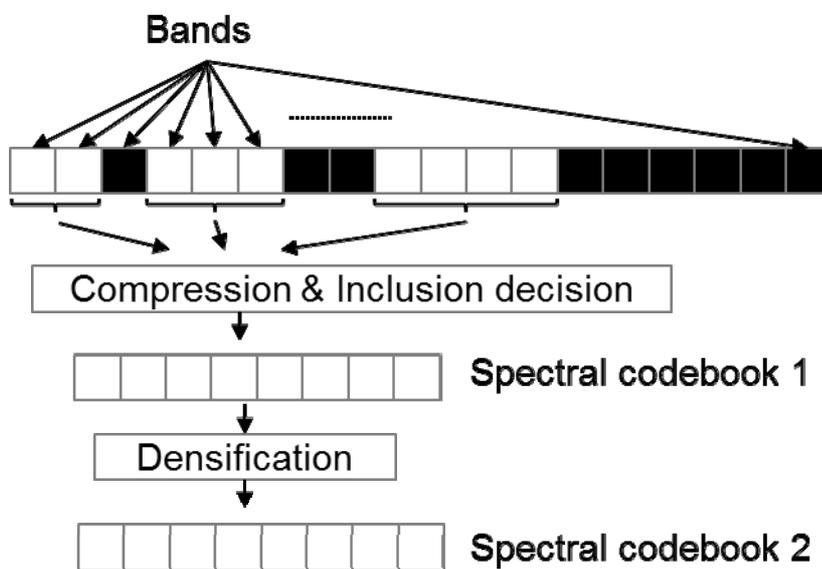
Figure 104: Creation of two parallel codebooks to handle sparse coded vectors.

The compression of the coded residual vectors is done according to the following definition:

$$\hat{X}_{M,comp}(k) = \begin{cases} 1, & \hat{X}_M > 0 \\ 0, & \hat{X}_M = 0 \\ -1, & \hat{X}_M < 0 \end{cases} \tag{1831}$$

The virtual codebook which constitutes the spectral codebook is built only from "populated" sub-vectors, where each sub-vector has a length of 8. If a coded sub-vector does not fulfill the criterion:

$$\sum_{k=k_{start}(b)+8i}^{k_{start}(b)+8i+7} \left| \hat{X}_{M,comp}(k) \right| \geq 2, \; i=0,...,\frac{L_M(b)}{8} \tag{1832}$$

it is considered sparse, and is rejected. Since the sub-vector length is 8, this corresponds to a rejection criterion if less than 25% of the vector positions are populated. The remaining compressed sub-vectors are concatenated into Spectral codebook 1, $Y_M(k)$ with the length L_Y. The final step of the anti-sparseness processing is to combine the codebook samples pair-wise sample-by-sample with a frequency reversed version of the codebook. The combination can be described with the following relation:

$$Y'_M(k) = \begin{cases} sign(Y_M(k)) \times (|Y_M(k)| + |Y_M(L_Y - k)|), & Y_M(k) \neq 0 \\ |Y_M(L_Y - k)|, & Y_M(k) = 0 \end{cases}, \quad k = 0,1,...,L_Y \tag{1833}$$

For SWB processing at 24.4 or 32 kbps in case of low spectral stability, spectral codebook 1 is used below band $f_{cb} = 20$ and the spectral codebook 2 is used above and including band $f_{cb} = 20$. The spectral filling using these two codebooks is depicted in figure 105.
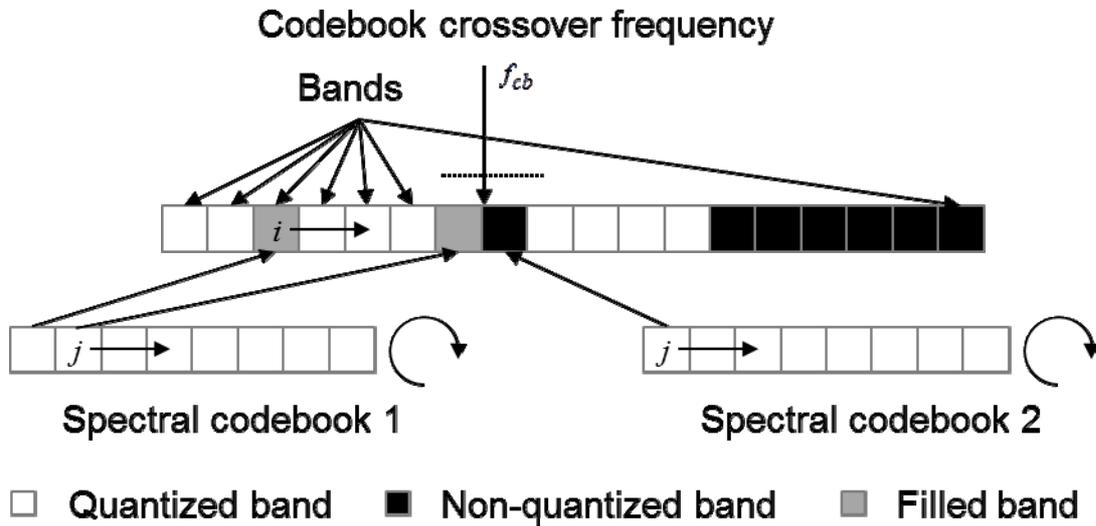
Figure 105: Creation of two parallel codebooks to handle sparse coded vectors.

6.2.3.2.1.3.2.2   High frequency noise fill

Based on the low-frequency filled spectrum, and prior to noise level attenuation, as described in the previous clause, the last step of the spectral filling consists of the generation of the target bandwidth audio signal. In other words, the process synthesizes a high-frequency spectrum from the filled spectrum by spectral folding based on the value of the transition frequency.

The target bandwidth generation is based on the spectral folding of the spectrum below the transition frequency to the high-frequency spectrum (zeroes above the transition frequency), see figure 106. A first spectral folding is achieved with respect to the point of symmetry defined by the transition frequency f_t. No spectrum coefficients from frequencies below $f_t/2$ are folded into the high frequencies. In other words, only the upper half of the low frequencies are folded. If there are not enough coefficients in the upper half of the low frequencies to fill the whole spectrum above the transition frequency, the spectrum is folded again around the last filled coefficient. This process is repeated until the last band is filled.
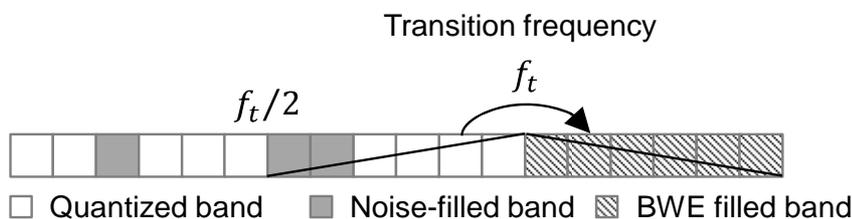
Figure 106: The spectrum above the transition frequency is regenerated using spectral folding from the transition frequency

6.2.3.2.1.3.2.3   Noise level adjustment

After the fine structure of the spectral holes has been determined, the noise-filled part of the spectrum is attenuated according to the received *NoiseLevel* index. In the case of transient mode, the *NoiseLevel* is not estimated in the encoder and is automatically set to the value corresponding to zero index, i.e., 0 dB.

This operation is summarized by the following equation:

$$\hat{X}_M^{fill}(k) = 2^{-NoiseLevel}\hat{X}_M(k), \quad \text{for } k \text{ such that } R(b)=0 \text{ and } k \leq f_t \qquad (1834)$$

For SWB processing at 24.4 or 32 kbps in case of low spectral stability, an additional adaptive noise-fill level adjustment is employed. First, an envelope adjustment vector $g_{adj\_1st}(b)$ is derived according to the following pseudo-code:

> *For* $b = 0,1,...,N_{bands}$,
>   *if* $K(b)=0$,
>   *if* $L_M(b) \leq 16$,
>   *if* $b > 0$
>   *if* $Qadj(b)$,
> $g_{adj\_1st}(b) = 0.36$
>   *else*
> $g_{adj\_1st}(b) = 0.54$
>   *else*
> $g_{adj\_1st}(b) = 0.72$
>   *else*
>   *if* $b < b_{last}$ *and* $Qadj(b)$,
> $g_{adj\_1st}(b) = 0.54$
>   *else*
> $g_{adj\_1st}(b) = 0.72$
>   *else*
> $g_{adj\_1st}(b) = 1.0$

where $Qadj(b) = TRUE$ *if* $K(b-1) > 0$ *and* $K(b+1) > 0$. Further, $K(b)$ denotes the number of pulses for band b as described in subclause 5.3.4.2.7, where $K(b)=0$ corresponds to the case when zero bits are assigned to band b. In short it permits strong attenuation for short bands where the neighboring bands are quantized, and gradually less when these requirements are not fulfilled. Once $g_{adj\_1st}(b)$ has been obtained, attenuation regions of consecutive bands b where $g_{adj\_1st}(b) < 1.0$ are identified. The attenuation for each of these regions are adjusted according to

$$g_{adj\_lim}(b) = \alpha_{adj}(w) + (1 - \alpha_{adj}(w))g_{adj\_1st}(b) \qquad (1835)$$

where w is the number of consecutive bands in the attenuation region. The width-dependent attenuation function $\alpha_{adj}(w)$ is a piece-wise linear function defined as

$$\alpha_{adj}(w) = \begin{cases} 0, & w < 6 \\ 1, & (w-6)/5 > 1 \\ (w-6)/5, & otherwise \end{cases} \qquad (1836)$$

The resulting vector $g_{adj\_lim}(b)$ is further combined with a limiting function which prevents attenuation during audio with high spectral stability. The spectral stability is calculated based on a low-pass filtered Euclidian distance $\tilde{D}_M^{[n]}$ between the spectral envelope values $E_M^{[n]}(b)$ of adjacent frames:

$$D_M^{[n]} = \sqrt{\frac{1}{27} \sum_{b=0}^{26} \left(E_M^{[n]}(b) - E_M^{[n-1]}(b) \right)^2} \tag{1837}$$

$$\tilde{D}_M^{[n]} = 0.1 D_M^{[n]} + 0.9 D_M^{[n-1]} \tag{1838}$$

Here $.^{[n]}$ denotes the value of the variable for frame n. The spectral envelope stability parameter $S^{[n]}$ is derived by mapping $\tilde{D}_M^{[n]}$ to the range $[0,1]$ using a discretely sampled sigmoid function implemented as a lookup table $stab\_trans(I)$. Due to the symmetry of the function, the table is mirrored around the mid-point such that the final stability parameter can be obtained by

$$S^{[n]} = \begin{cases} 1 - stab\_trans(I), & \tilde{D}_M^{[n]} < 2.571757 \\ stab\_trans(I), & \tilde{D}_M^{[n]} \geq 2.571757 \end{cases} \tag{1839}$$

where the quantization index I is found by $I' = \left[\left| \tilde{D}_M^{[n]} - 2.571757 \right| \middle/ 0.103138 \right]$ and clamping the index I' to the range $[0,9]$. Finally, the gain adjustment vector $g_{adj}(b)$ is derived as

$$g_{adj}(b) = \max\left(S^{[n]}, g_{adj\_lim}(b) \right) \tag{1840}$$

where the envelope stability $S^{[n]}$ acts as a limiting function for the gain adjustment vector.

For WB processing, a slightly different gain adjustment vector is derived. Here, the $g_{adj}(b)$ is computed as

$$g_{adj}(b) = \begin{cases} gain\_att(I), & 0 < K(b) < 40 \\ 1.0, & otherwise \end{cases} \tag{1841}$$

where $gain\_att(I)$ is a gain attenuation table for index I, which in turn is derived by

$$I = \left[K(b) \times att\_step\left(\lfloor L_M(b)/8 \rfloor \right) \right] - 1 \tag{1842}$$

For SWB and 24.4 and 32 kbps, the gain adjustment is applied using a hangover logic which only permits attenuation in case a sequence of 150 frames without transients has been observed. In case this requirement is met for SWB encoded bandwidth or if the encoded bandwidth is WB, the gain adjustment vector is combined with the quantized envelope vector $\hat{E}_M(b)$ to form the gain adjusted envelope vector $\tilde{E}_M(b) = \hat{E}_M(b) \cdot g_{adj}(b)$.

6.2.3.2.1.3.2.4 Spectral fill envelope shaping

When the full-bandwidth fine spectral structure is generated, the resulting spectrum is shaped by applying the gain adjusted envelope vectors for each band $b = 0,1,...,N_{bands} - 1$ according to:

$$X'_M(k) = \tilde{E}_M(b)\hat{X}_M(k), \quad k = k_{start}(b),...,k_{end}(b) \tag{1843}$$

6.2.3.2.2 Transient Mode

6.2.3.2.2.1 Envelope decoding

The envelope is decoded as is described in subclause 6.2.3.2.1.1. In addition to those step the norms are also sorted as is done in the encoder, see subclause 5.3.4.2.2.1.

6.2.3.2.2.2 Fine structure inverse quantization (spectral coefficients decoding)

The spectral coefficients are decoded as for the Normal HQ mode as described in subclause 6.2.3.2.1.2.

6.2.3.2.2.3   Spectral filling

The spectral filling is done as described in 6.2.3.2.1.3, but the bandwidth extension in subclause 6.2.3.2.1.3.2.2 is not done.

6.2.3.2.3   Harmonic Mode

6.2.3.2.3.1   Core decoding

Envelope decoding and the PVQ decoder are described in subclause 6.2.3.2.1.1 and subclause 6.2.3.2.1.2, respectively.

If a sub-band has bits allocated to it, then the decoded coefficients of the sub-band are de-normalized by multiplying the de-quantized norm of the sub-band, and in this way the de-normalized coefficients $\tilde{X}_M(k), k = 0,...,k_{end}(b_{last\_sfm})$ are obtained. Otherwise, if a sub-band has no bits allocated to it, the de-normalized coefficients $\tilde{X}_M(k)$ of that sub-band are set to 0. And the higher frequency band coefficients with the index of sub-band above b_{last_sfm} are 0 and are reconstructed by bandwidth extension, where b_{last_sfm} is the index of the highest frequency sub-band of the decoded low frequency band signal.

6.2.3.2.3.2   Bandwidth extension decoding for harmonic mode

The start index for the bandwidth extension is adaptively obtained according to the value of b_{last_sfm}.

Firstly preset the start index for bandwidth extension b_{high_sfm}:

$$b_{high\_sfm} = \begin{cases} 21, & 24.4\text{kbps} \\ 24, & 32\text{kbps} \end{cases} \tag{1844}$$

Then, in order to predict the excitation signal of bandwidth extension, judge whether the index of the highest frequency sub-band of the decoded low frequency band signal b_{last_sfm} is less than the start index for bandwidth extension b_{high_sfm}, i.e. judge whether the highest frequency bin of bit allocation is less than the preset start frequency bin for bandwidth extension,

- if $b_{last\_sfm} < b_{high\_sfm}$, b_{last_sfm} is then set to b_{high_sfm}. The excitation signal of bandwidth extension is predicted by the preset start index b_{high_sfm} and the chosen excitation signal from the decoded low frequency band signal with the given bandwidth length.

- Otherwise, the excitation signal of bandwidth extension is predicted by the preset start index b_{high_sfm}, the index of the decoded highest frequency sub-band b_{last_sfm} and the chosen excitation signal from the decoded low frequency signal with the given bandwidth length.

Finally, the higher frequency band signal is reconstructed by the predicted excitation signal and the envelopes as described in subclause 6.2.3.2.2.1.

6.2.3.2.3.2.1   Calculate excitation adaptive normalization lengths

The de-normalized coefficients calculated in subclause 6.2.3.2.3.1 need to be recovered to remove the original core envelope effects to give the excitation for bandwidth extension. The normalization length L_{mdct_norm} is adaptively obtained according to the signal characteristics. The normalization length of the previous frame $L_{mdct\_norm}^{[-1]}$ is initialized to 8.

208 MDCT coefficients in the 0-5200 Hz frequency range are split into 13 normalization sub-bands with 16 coefficients per sub-band. The peak magnitude and average magnitude in each normalization sub-band are then calculated. The counter n_band is initialized to zero and increased by one if $r_{p2a}(j) > 8$ and $peak(j) > 10$, where

$$r_{p2a}(j) = \begin{cases} \dfrac{15 \cdot peak(j)}{avg(j) - peak(j)}, & if \ avg(j) \neq peak(j) \\ 0, & otherwise \end{cases} \quad j = 0,\cdots,12 \tag{1845}$$

The normalization length L_{mdct_norm} is set to $32 + 2.5 \cdot n\_band$, and it is adjusted with reference to the value from the previous frame, $L_{mdct\_norm}^{[-1]}$

$$L_{mdct\_norm} = 0.1 \cdot L_{mdct\_norm} + 0.9 \cdot L_{mdct\_norm}^{[-1]} \tag{1846}$$

6.2.3.2.3.2.2 Calculate envelopes for excitation normalization

The normalization envelopes, $E_{base\_exc}(k), k = 0,...,201$ for each spectral bin are calculated as follow:

$$E_{base\_exc}(k) = \begin{cases} \displaystyle\sum_{k=0}^{L_{mdct\_norm}/2+i} \left| \tilde{X}_M(k) \right|, & i = 0,\cdots,L_{mdct\_norm}/2 \\ \displaystyle\sum_{k=i-L_{mdct\_norm}/2}^{L_{mdct\_norm}/2+i} \left| \tilde{X}_M(k) \right|, & i = L_{mdct\_norm}/2,\cdots,201 - L_{mdct\_norm}/2 \\ \displaystyle\sum_{k=i-L_{mdct\_norm}/2}^{201} \left| \tilde{X}_M(k) \right|, & i = 201 - L_{mdct\_norm}/2,\cdots,201 \end{cases} \tag{1847}$$

The value $\tilde{X}_M(k), k = 0,...,201$ are then normalized using the normalization envelopes $E_{base\_exc}[k], k = 0,...,201$ to obtain the normalized coefficients $\tilde{X}_{base\_exc}(k), k = 0,...,201$,

$$\tilde{X}_{base\_exc}(k) = \tilde{X}_M(k) / E_{base\_exc}(k), \quad k = 0,...,201 \tag{1848}$$

6.2.3.2.3.2.3 Adaptive excitation generation

The normalized coefficients in the frequency range 1500-5025Hz, i.e. the $60^{th} - 201^{st}$ coefficients, are selected for the excitation calculation. The starting frequency bin of the excitation, src, is calculated as follows,

$$src = \begin{cases} \tilde{X}_{base\_exc} + 60 + k_{end}(b_{last\_sfm}) - k_{end}(b_{high\_sfm}), & if \ k_{end}(b_{last\_sfm}) - k_{end}(b_{high\_sfm}) \leq 202 - 60 \\ \tilde{X}_{base\_exc} + 202 - 1, & otherwise \end{cases} \tag{1849}$$

The selected low frequency normalized coefficients from which the re-constructed higher band coefficients $\tilde{X}_{M\_norm}$ are obtained are copied to the high band starting at frequency, dst, as follows

$$dst = \tilde{X}_{M\_norm} + k_{end}(b_{last\_sfm}) \tag{1850}$$

The low frequency normalized coefficients may in practice be copied N times as a circular buffer in order to fill in the re-constructed higher bands, where N can be a decimal fraction.

6.2.3.2.3.2.4 Weighting the re-constructed higher band coefficients and random noise

The envelopes $\tilde{I}_{norm}(b)$ of the re-constructed higher band coefficients are calculated according to the band structure given in table 129, and then the re-constructed higher band signal is weighted and random noise added.

$$\tilde{X}_M(k) = \alpha * \tilde{X}_{M\_norm}(k) * \tilde{I}_N^q(b) / \tilde{I}_{norm}(b) + \beta * random(k) / 32768.0 * \tilde{I}_N^q(b) \tag{1851}$$

Where $b = [b_{last\_sfm} + 1, N_{bands} - 1]$ and $k = [k_{start}(b), k_{end}(b)]$.

The weighting factor for the normalized re-constructed higher band signal, α, is

$$\alpha = \sqrt{1 - f_{noise\_level}} \tag{1852}$$

The weighting value of the normalized re-constructed higher band signal, β, is

$$\beta = 0.5\sqrt{f_{noise\_level}} \tag{1853}$$

where the noise level f_{noise_level} is estimated as follows:

$$f_{noise\_level} = \max(\min(0.25 - \tilde{I}_{norm\_diff\_sum} / \tilde{I}_{norm\_sum}, \ 0), \quad 0.25) \tag{1854}$$

and the sum of the differences between the consecutive norms $\tilde{I}_{norm\_diff\_sum}$ and the sum of the norms $\tilde{I}_{norm\_sum}$ in the index range $b = [0,28]$, are given by

$$\tilde{I}_{norm\_diff\_sum} = \sum_{b=0}^{28} \left| I_N^q(b+1) - I_N^q(b) \right| \tag{1855}$$

$$\tilde{I}_{norm\_sum} = \sum_{b=0}^{28} \left| I_N^q(b) \right| \tag{1856}$$

6.2.3.2.4   HVQ

First the HVQ decoder extracts from the bitstream number of coded peaks, and reconstructs spectral peaks positions $\hat{P}_k$ and peak gains $\hat{G}_p(k)$. The peaks positions are decoded with either Huffman decode or space coding decoder, based on the received mode decision. The peak shapes vectors $\hat{S}(k)$ are reconstructed from the received VQ indices and further scaled with reconstructed peak gains $\hat{G}_p(k)$ for the corresponding shape region. The low-frequency bands are PVQ decoded, with number of bands determined as described in 5.3.4.2.5

The unquantized coefficients below 5.6 kHz for 24.4 kb/s and 8 kHz for 32 kb/s are grouped into 2 sections and noise filled and scaled. Each of the sections covers half of coded band (of 112 bins at 24.4 kbps and 160 bins at 32 kbps). After the noise fill each of the sections is scaled with the corresponding reconstructed gains $\hat{G}_{ne}(0)$ and $\hat{G}_{ne}(1)$. The gains reconstructed in the current frame t are smoothed with the levels from the past frame $t-1$

$$\hat{G}_h^{(t)}(b) = 0.9\hat{G}_h^{(t-1)} + 0.1\hat{G}_h^{(t)} \qquad b = 0,1 \tag{1857}$$

The reconstructed envelope levels used above 5.6 kHz for 24.4 kb/s and 8 kHz for 32 kb/s are adjusted based on the presence or absence of peak in the low-frequency fine structure used in the noise-fill.

$$\begin{aligned} &if\_peak \\ &\qquad I_b = 0.1I_{b-1} + 0.8I_b + 0.1I_{b+1} \\ &else \\ &\qquad \min(I_{b-1}, I_b, I_{b+1}) \end{aligned} \tag{1858}$$

6.2.3.2.5 Generic Mode
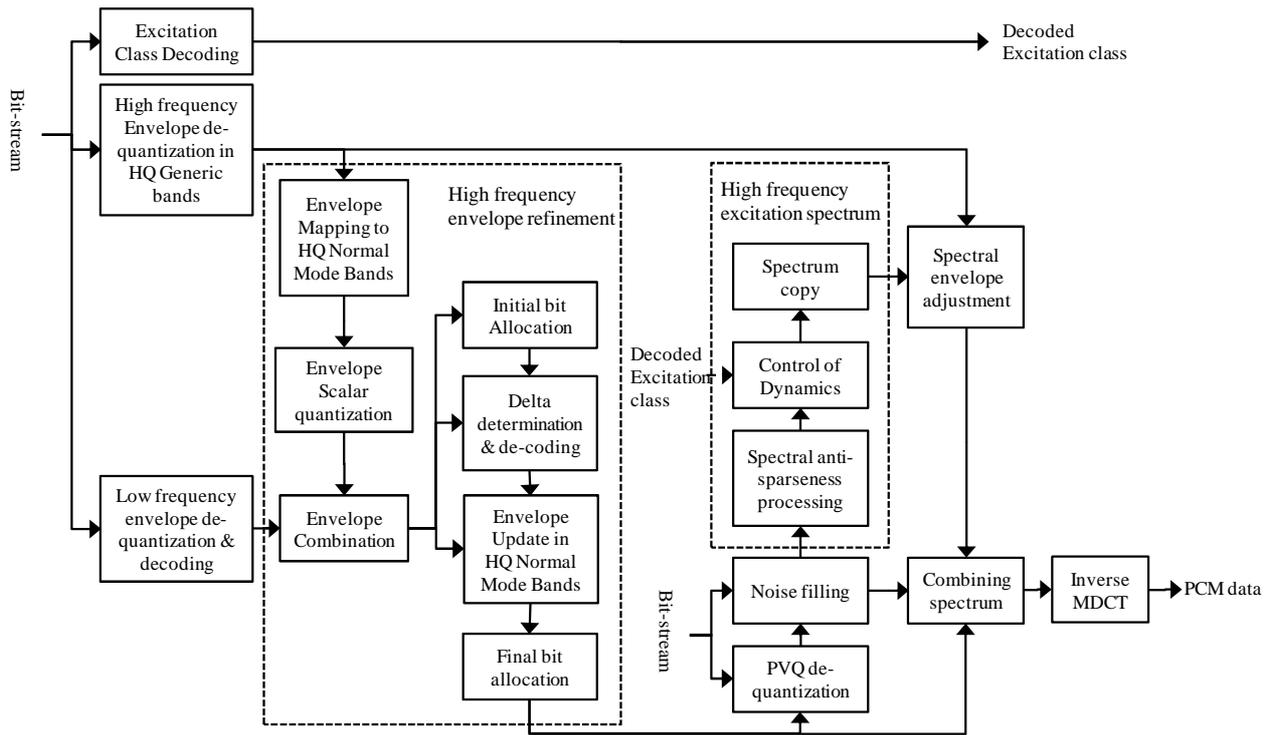
Figure 107: Generic mode Decoder Block Diagram

6.2.3.2.5.1 Low frequency envelope decoding

This is described in subclause 6.2.3.2.1.1.

6.2.3.2.5.2 High frequency envelope de-quantization

The envelope VQ indices $\left\{idx_{env1}, idx_{err21}, idx_{err_{22}}, idx_{Ierr_{31}}, idx_{Ierr_{32}}\right\}$ for SWB or
$\left\{idx_{env_1}, idx_{err21}, idx_{err_{22}}, idx_{Ierr_{31}}, idx_{Ierr_{32}}, idx_{FB}\right\}$ for FB are used to de-quantize the high frequency envelope.

At 24.4kbps, the de-quantized high frequency envelope can be determined by:

$$
\hat{f}_{rms\_G}(j) =
$$

$$
\begin{cases}
\hat{e}nv_1(\lfloor j/2\rfloor) + \hat{e}rr_2(\lfloor j/2\rfloor) & \text{for } j = 0,2,4,6,8,10,12 \\[2em]
\hat{I}err_3\left(\left\lfloor\frac{j-1}{2}\right\rfloor\right) + \dfrac{\hat{e}nv_1\left(\left\lfloor\frac{j-1}{2}\right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor\frac{j-1}{2}\right\rfloor\right) + \hat{e}nv_1\left(\left\lfloor\frac{j+1}{2}\right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor\frac{j+1}{2}\right\rfloor\right)}{4} & \\[1em]
& \text{for } j = 1,3,5,7,9,11 \\[2em]
\hat{I}err_3\left(\left\lfloor\frac{j-1}{2}\right\rfloor\right) + \dfrac{\hat{e}nv_1\left(\left\lfloor\frac{j-1}{2}\right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor\frac{j-1}{2}\right\rfloor\right)}{2} & \text{for } j = 13
\end{cases}
\tag{1859}
$$

While at 32kbps, the de-quantized high frequency envelope can be determined by:

$$\hat{f}_{rms\_G}(j) =$$

$$\begin{cases} \hat{e}nv_1(0) + \hat{e}rr_2(0) + \hat{I}err_3(0) & \text{for } j = 0 \\ \\ \hat{I}err_3\left(\left\lfloor \dfrac{j+1}{2} \right\rfloor\right) + \dfrac{\hat{e}nv_1\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor \dfrac{j-1}{2} \right\rfloor\right) + \hat{e}nv_1\left(\left\lfloor \dfrac{j+1}{2} \right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor \dfrac{j+1}{2} \right\rfloor\right)}{4} & \\ & \text{for } j = 1,3,5,7,9 \\ \\ \hat{e}nv_1\left(\left\lfloor j/2 \right\rfloor\right) + \hat{e}rr_2\left(\left\lfloor j/2 \right\rfloor\right) & \text{for } j = 2,4,6,8,10,11 \end{cases}$$

(1860)

The final de-quantized envelope is then calculated:

$$\hat{E}nv(j) = 10^{\left(\hat{f}_{rms\_G}(j) + f_{rms\_mean}(j)\right)*0.05} \quad \text{for } j = 0,...,N_{bands\_G} - 1$$

(1861)

In FB case, idx_{FB} is further used to generate the de-quantized high frequency envelope.

6.2.3.2.5.3 High frequency envelope refinement

The high frequency envelope refinement is described in subclause 5.3.4.2.6.5. After de-quantizing the high frequency envelope using the VQ described in subclause 6.2.3.2.5.2, the de-quantized high frequency envelope is mapped to one of the HQ high rate normal mode bands. To generate the norms the mapped high frequency envelope is quantized and de-quantized with the scalar quantizer, as shown in subclause 5.3.4.2.6. The de-quantized low frequency envelope and the de-quantized high frequency norms are then combined. Using the combined norms, the bit allocation information per each band is calculated using the fractional bit allocation method. If there are any high bands which have allocated bits, the refinement data is decoded and used to update the high frequency norms. The updated norms are used for de-normalizing the de-quantized spectrum by the PVQ algorithm in subclause 6.2.3.2.5.4 and the noise filling algorithm in subclause 6.2.3.2.5.5.Then the initial bit allocation information is updated, based on the number of bits used for representing the refinement data.

6.2.3.2.5.4 PVQ

This is described in subclause 6.2.3.2.1.2.2

6.2.3.2.5.5 Noise filling

Noise filling is performed described in subclause 6.2.3.2.1.3.2.1. The last band for this noise filling in Generic mode is defined as $\max(core\_sfm, N_{band,LF} - 1)$, where $core\_sfm$ is the last band index where the spectrum was quantized using PVQ. The filled spectrum is further de-normalized to generate $X'_{M\_Q}(k)$ as described in subclause 6.2.3.2.1.3.2.4. If $core\_sfm$ is higher than N_{band_LF}-1, then the $\tilde{E}_M(b)$ are the high frequency norms described in subclause 6.2.3.2.5.3.

6.2.3.2.5.6 High frequency excitation spectrum

The high frequency excitation spectrum is based on a copy of the decoded low frequency spectrum. First spectral anti-sparseness processing is applied, and then dynamic range control is applied to depending on the decoded excitation class. Finally, a simple spectral copy is done to create the high frequency excitation spectrum.

6.2.3.2.5.6.1 Spectral anti-sparseness processing

The spectral anti-sparseness processing is performed on the low frequency spectrum by inserting a 0.5 amplitude coefficient, with a random sign, where the normalized spectrum is zero. The end band for the spectral anti-sparseness processing is specified by B_{anti} (=$\max(core\_sfm, N_{band\_LF}-1)$) and the end frequency is specified by L_{anti}(=$k_{end}(B_{anti})$).

$$\hat{X}_{M\_anti}(k) = \begin{cases} X'_{M\_Q}(k) & \text{if } X'_{M\_Q}(k) \,! = 0 \\ 0.5 & \text{if } X'_{M\_Q}(k) = 0 \text{ and } \lambda_{seed\_G} > 0 \\ -0.5 & \text{if } X'_{M\_Q}(k) = 0 \text{ and } \lambda_{seed\_G} \leq 0 \end{cases} \qquad \text{for } k = 0,1,\ldots,L_{anti} \qquad (1862)$$

where λ_{seed_G} is a random seed and updated by $\lambda_{seed\_G} = \lambda_{seed\_G}*31821 + 13849$.

After applying this anti-sparseness processing, the energy is further modified by applying the low band dequantized envelope as described in subclause 6.2.3.2.5.1.

6.2.3.2.5.6.2 Control of dynamics based on the excitation class

Following the spectral anti-sparseness processing, the spectrum is further modified by additional processing to control the dynamics.

The spectrum is first normalised by calculating the envelope of the processed spectrum, then dividing the spectrum by this envelope. The window size, L_{norm}, for this normalisation depends on the signal characteristics..

The 256 low frequency MDCT coefficients in the 0-6400 Hz frequency range, $\hat{X}_{M\_anti}(k), k = 0,\ldots,255$ are split into 16 sharpness bands (16 coefficients per band). In sharpness band j, if $23A_{sharp}(j) > 8 \sum_{k=16j}^{16j+15} |\hat{X}_{M\_anti}(k)|$

and $A_{sharp}(j) > 10$, the counter C_{band} is incremented by one.

The maximum magnitude of the spectral coefficients in a sharpness band, denoted $A_{sharp}(j)$, is:

$$A_{sharp}(j) = \max_{k=16j,\ldots16j+15} |\hat{X}_{M\_anti}(k)| \qquad j = 0,\ldots,15 \qquad (1863)$$

Parameter C_{band} is initialized to 0 and calculated for every frame. Then the normalization length L_{norm} is obtained:

$$L_{norm} = \left\lfloor 0.1 L_{norm}^{[-1]} + 0.9 L_{norm} + 0.5 \right\rfloor \qquad (1864)$$

where the current normalization length L_{norm} is calculated as follows:

$$L_{norm} = \left\lfloor 8 + 0.5 C_{band} \right\rfloor \qquad (1865)$$

and the current normalization length is preserved as $L_{norm}^{[-1]}$.

The spectrum is then normalized

$$X_{Norm}(k) = \frac{\hat{X}_{M\_anti}(k)}{\sum_{l=k-\lfloor L_{norm}/2 \rfloor}^{k+\lfloor (L_{norm}-1)/2 \rfloor} |\hat{X}_{M\_anti}(l)|} \qquad \text{for } k = 2,\ldots,N_{bands\_d}*16+1 \qquad (1866)$$

where N_{bands_d} is the number of bands used in the control of dynamics.

The sign vectors for the spectrum are then removed, leaving just the magnitude, and the mean is then calculated for each band p. The bands are 16 frequency bins wide, and start at frequency bin 2. For the SWB case, at 24.4kbps there are 9 bands ending at frequency bin 145, while for 32kbps there are 8 bands, ending at frequency bin 129. For the FB case, at 24.4kbps there are 19 bands ending at frequency bin 305, while for 32kbps there are 18 bands, ending at frequency bin 289

The amplitude of each frequency bin is then reduced by a dynamics control factor of the difference between the bin amplitude and mean of the band.

$$\left|\tilde{X}_{Norm}(k)\right| = \left|X_{Norm}(k)\right| - \left\{\left|X_{Norm}(k)\right| - Mean(p)\right\} * drf \quad for\ k = 2,...,N_{bands\_d}*16+1 \tag{1867}$$

where *drf* is the dynamics control factor depending on the decoded excitation class,

$$drf = \begin{cases} 0.05 & for\ HF\_excitation\_class1 \\ 0.8 & for\ HF\_excitation\_class0 \\ 0.2 & for\ HF\_speech\_excitation\_class \end{cases} \tag{1868}$$

The original signs are then re-applied for the *HF_Speech_excitation_class* and the *HF_excitation_class1*; however random signs are used for the *HF excitation_class0*. If $\lambda_{seed,dc} = \lambda_{seed,dc}*31821+13849$ is higher than 0, the original sign is re-applied, otherwise, a reversed sign of the original is applied. The initial random seed is

$$\lambda_{seed\_dc} = R(0)*8 + R(1)*4 + R(2)*2 + R(3) \tag{1869}$$

where $R(b)$ is the number of allocated integer bits for each band.

The spectrum is then normalised:

$$\tilde{X}(k) = \frac{\tilde{X}_{Norm}(k)}{\sqrt{\frac{1}{16}\sum_{i=p*16}^{p*16+15}\left[\tilde{X}_{Norm}(i)\right]^2}} \quad for\ k = 2,...,N_{bands\_d}*16+1 \tag{1870}$$

The normalised spectrum is then copied, using the mapping in table 172, to create the high frequency excitation spectrum.

Table 172: Frequency mapping to generate high frequency excitation spectrum

	l	$St_{src\_G}(l)$	$End_{src\_G}(l)$	$St_{dst\_G}(l)$	$End_{dst\_G}(l)$
24.4kbps	0	2	129	320	447
	1	2	129	448	575
	2	80	143	576	639
24.4kbps FB	3	144	303	640	799
32kbps	0	2	129	384	511
	1	2	129	512	639
32kbps FB	2	130	289	640	799

Finally the high frequency excitation spectrum is adjusted at the junction boundaries,

$$\tilde{X}(j) = \tilde{X}(j)*R_3(j) \quad j = St_{dst\_G}(1), St_{dst\_G}(1)+1,... \quad j \in \left\{R_3(j)<1\right\} \tag{1871}$$

where $R_{10} = \sum_{i=St_{dst\_G}(1)}^{St_{dst\_G}(1)+4}\left|\tilde{X}_{Norm}(i)\right|, R_{20} = \sum_{i=St_{dst\_G}(1)-6}^{St_{dst\_G}(1)-2}\left|\tilde{X}_{Norm}(i)\right|, R_3(St_{dst\_G}(1)) = max(0.3, R_{20}/R_{10})$ and $R_3(j+1) = R_3(j)+0.1$.

If $R_{10}/R_{20} > 5$, and then

$$\tilde{X}(j) = \tilde{X}(j)*R_4(j)\ for\ j = St_{dst\_G}(1)-1, St_{dst\_G}(1)-2,... \quad j \in \{R_4(j)>1\} \tag{1872}$$

where $R_4(St_{dst\_G}(1)-1) = min(5, R_{10}/R_2 0)$ and $R_4(j-1) = R_4(j)-0.5\ for\ j = 0,1,...$

At 24.4kbps,

$$\tilde{X}(j) = \tilde{X}(j)*R_3(j) \quad j = St_{dst\_G}(2), St_{dst\_G}(2)+1,... \quad j \in \{R_3(j)<1\} \tag{1873}$$

where $R_{11} = \sum_{i=St_{dst\_G}(2)}^{St_{dst\_G}(2)+1}\left|\tilde{X}_{Norm}(i)\right|, R_{21} = \sum_{i=St_{dst\_G}(2)-4}^{St_{dst\_G}(2)-1}\left|\tilde{X}_{Norm}(i)\right|, R_3(St_{dst\_G}(2)) = max(0.3, R_{21}/R_{11})$ and $R_3(j+1) = R_3(j)+0.1$, and then

$$\widetilde{X}(j) = \widetilde{X}(j) * R_6(j) \quad for \ j = St_{dst\_G}(2)-1, St_{dst\_G}(2)-2,... \quad j \in \{R_6(j) > 1\} \tag{1874}$$

where $R_6(St_{dst\_G}(2)-1) = 0.5 * R_{11}/R_{21}$, $R_5 = 0.025 * R_{11}/R_{21}$ and $R_6(j+1) = R_6(j) - R_5$.

6.2.3.2.5.7   Spectral envelope adjustment

Spectral envelope adjustment is used to generate high frequency spectrum with combining the high frequency excitation spectrum and the interpolated envelope according to the frequency. The low frequency envelope is used in the first band. The interpolated envelope is calculated as follows:

$$\hat{E}nv_t(k) = \begin{cases} \hat{E}nv(-1) \cdot w_0(k) + \hat{E}nv(0) \cdot (1 - w_0(k)) & for \ k = k_{st}(0),...,k_{st}(0)+7 \\ \hat{E}nv(p-1) \cdot w_p(k) + \hat{E}nv(p) \cdot (1 - w_p(k)) & for \ k = \frac{k_{st}(p-1)+k_{st}(p)}{2},...,\frac{k_{st}(p)+k_{st}(p+1)}{2}-1 \\ \hat{E}nv(N_{band\_G}-1) & for \ k = \frac{k_{st}(N_{band\_G}-1)+k_{end}(N_{band\_G}-1)-1}{2},...,639 \end{cases} \tag{1875}$$

where p ($p=1,...,N_{band\_G}-1$) is a band index;

$\quad w_p(k)$ is a interpolation function where $w_0(k) = 0.125(k - k_{st}(0))$ and

$\quad w_p(k) = 2 \cdot (k - (k_{st}(p-1) + k_{st}(p))/2)/(k_{st}(p+1) - k_{st}(p-1))$; and

$\quad \hat{E}nv(-1)$ is the initial value of the spectral envelope $\left(\hat{E}nv(-1) = \sqrt{\frac{1}{16} \sum_{k=k_{start\_G}(0)-16}^{k_{start\_G}(0)-1} X'_{M\_Q}(k) * X'_{M\_Q}(k)} \right)$.

The envelope is then multiplied by the generated high frequency excitation spectrum in subclause 6.2.3.2.5.6.

$$\hat{X}_{M\_G}(k) = \widetilde{X}(k) * \hat{E}nv_t(k) \quad for \ k = St_{dst,HQG}(0), St_{dst,HQG}(0)+1,...,639 \tag{1876}$$

In the FB case, the maximum value of k in equations (1875) and (1876) is corrected to 799 and a decision is made on whether or not to interpolate the envelope by comparing the envelope of the first band in FB and the last band in SWB.

$$Inf_{int} = \begin{cases} 1 & if \ \hat{E}nv(p_b-1) - \hat{E}nv(p_b) > 15 \ or \ \hat{E}nv(p_b) < 5 \\ 0 & otherwise \end{cases} \tag{1877}$$

where p_b is the index of the first band in FB, 14 at 24.4kbps or 12 at 32kbps.

6.2.3.2.5.8   Spectral combining

The final step of the Generic mode is to combine the noise filled spectrum $X'_{M\_Q}(k)$, obtained from decoding the quantized spectrum in subclauses 6.2.3.2.5.4 and 6.2.3.2.5.5, with the high frequency spectrum $\widetilde{X}_{M\_G}(k)$, generated in subclause 6.2.3.2.5.7. The noise filled spectrum includes the low frequency spectrum and some bands in the high frequency spectrum where bits were allocated during the spectral quantization. The generated high frequency spectrum includes only the high frequency spectrum.

There are two kinds of overlap bands between these two spectra; one is a partial overlap at the junction between the low frequency and the high frequency (376~400 at 32kbps, 304~328 at 24.4kbps). The other is a full overlap due to the difference between the two band allocations, i.e. due to some bands in the high frequency spectrum being allocated bits during the spectral quantisation.

In the partial overlap band, the spectral combining is performed based on an overlap and add process. If there are any allocated bits from the spectral quantizer, the noise filled spectrum $X'_{M\_Q}(k)$ is used directly for the final decoded spectrum. If there were no bits allocated by the spectral quantizer, a overlap and add process between the two spectra is performed:

$$X'_M(k) = X'_{M\_Q}(k) * \left(\frac{l}{L_{ov}}\right) + \tilde{X}_{M\_G}(k) * (1 - \frac{l}{L_{ov}}) \quad for \ k = St_{dst\_G}(0), St_{dst\_G}(0)+1,..., St_{dst\_G}(0) - L_{ov} - 1 \ (1878)$$

where L_{ov} is the overlapped length at the junction band, 16 at 24.4kbps and 8 at 32kbps.

In the full overlap bands, the spectrum is combined in a selective way. If there are any allocated bits from the spectral quantizer, the noise filled spectrum $X'_{M\_Q}(k)$ is used directly for the final decoded spectrum. If there were no bits allocated by the spectral quantizer, the high frequency spectrum $\tilde{X}_{M\_G}(k)$ is used for generating the final decoded spectrum $X'_M(k)$.

6.2.3.2.6 PVQ decoding and de-indexing

6.2.3.2.6.1 High dynamic range arithmetic decoding

The PVQ-codewords are extracted from the bit stream using the Range decoder.

6.2.3.2.6.2 Split-PVQ decoding approach

The PVQ-split parameters are obtained as inverse of the functions in subclause 5.3.4.2.7.2

6.2.3.2.6.2.1 Split-PVQ Decoder band splitting calculation

The initial number of segments (parts) N_{p_init} is computed according to the first equation in subclause 5.3.4.2.7.2.1. In case $N_{p\_init} < 10$ and the band bit rate is high, the flag pvq_split_inc is read from the bit stream. Finally, N_p is computed as

$$N_p = \begin{cases} N_{p\_init} + 1, & pvq\_split\_inc = 1 \\ N_{p\_init}, & pvq\_split\_inc = 0 \end{cases} \quad (1879)$$

6.2.3.2.6.2.2 PVQ sub vector gain decoding

The decoded Split-PVQ angles are converted into sub-vector gains.

6.2.3.2.6.3 PVQ sub-vector MPVQ de-indexing

First the N, K values for the sub vector $\mathbf{y}$ to be decoded are used in a 'FindSizeAndOffsets(N,K)' function which pre-computes the row $n = N$ of the MPVQ offset matrix [$A(N,0...K)$, $U(N,K+1)$] and also calculates the integer size of the MPVQ-index *MPVQ-size* using this last row. The last four equation in subclause 5.3.4.2.7.4.1 are employed for the offset and size calculations.

Secondly the 1 bit leading sign index and the MPVQ-index *index* is obtained from the Range decoder using the calculated *MPVQ-size* information. The leading sign *lead_sign* is decoded from the 1 bit leading sign index, where a zero sign index yields a positive *lead_sign* value of "+1", and a non-zero sign index yields a negative *lead_sign* value of "-1".

The third step is the actual MPVQ-de-indexing scheme, converting the leading sign *lead_sign* and the *index* to a valid integer $PVQ(N,K)$ vector $\mathbf{y}$.

The MPVQ de-indexing loop is carried out according to figure 108, where *index* is the MPVQ-index, *n* is the current row number of the MPVQ offset matrix, *pos* is the pointer into the samples/coefficients of the received PVQ-vector $\mathbf{y}$. The function "FindAmplitudeAndOffset" obtains the amplitude k_delta and the MPVQ indexing offset amp_offset for the current number of accumulated pulses k_max_local, by searching in the current row $A(n,0...k\_max\_local)$ in the MPVQ offset matrix. Further the function "UpdateOffsetsBwd" iteratively updates the required MPVQ-offsets for the next larger dimension using combinations of the last four equations in subclause 5.3.4.2.7.4.1. The function "GetLeadSign" obtains the next leading sign value *lead_sign* from the LSB of *index*, and shifts the *index* one bit to the

right. On the decoder side the MPVQ recursion is run in the order of position 0 to position $N-1$, with a dimension n decreasing from N to 1.
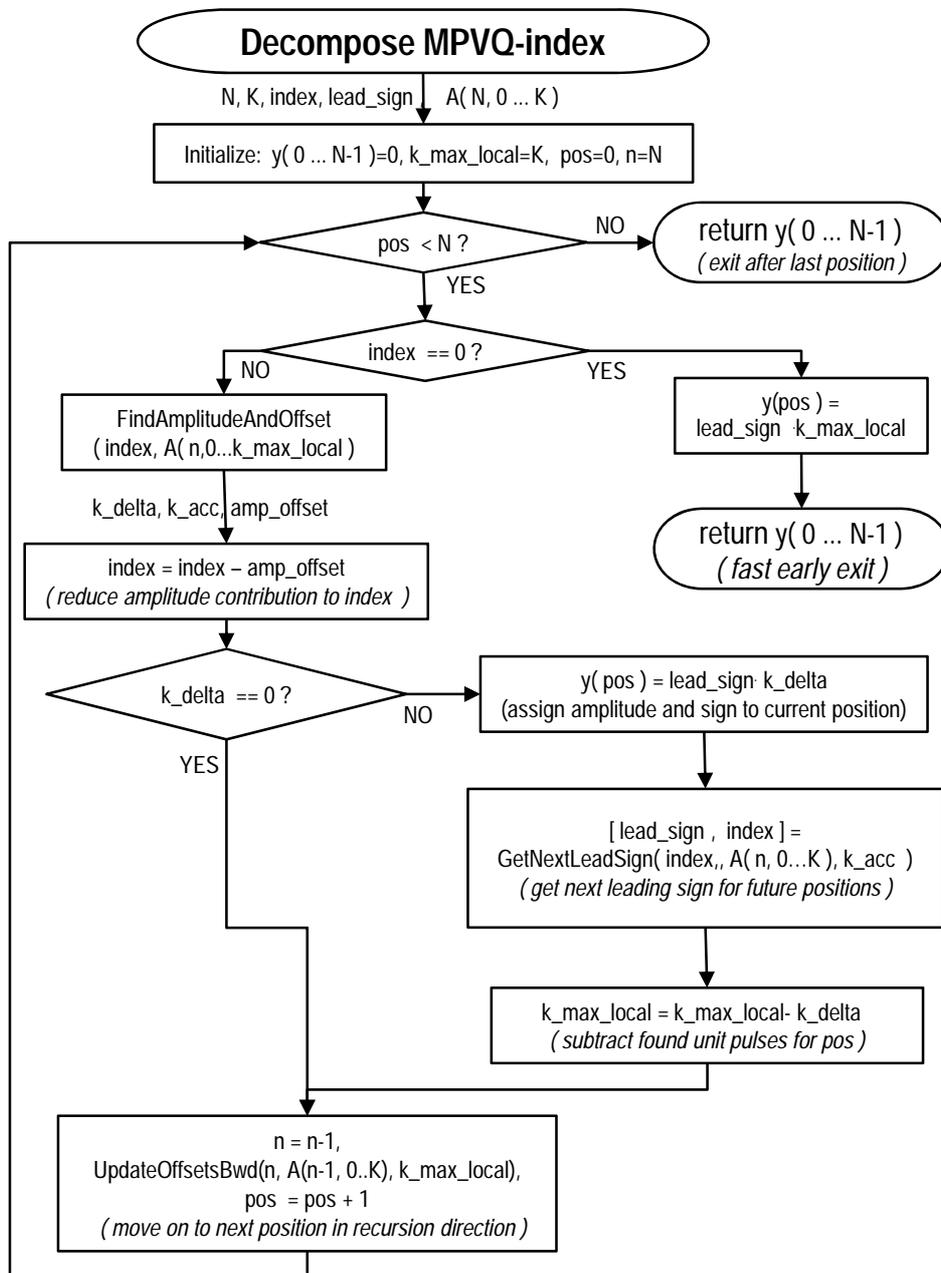
Figure 108: Detailed MPVQ-de-indexing

The calculation of the indexing offset matrix is optimized to use direct calculations up to row $n = 3$ for any combination of N and K, further if the number of unit pulses K are low enough and the dimension N is 5 or lower, a direct row n initialization of the offset matrix is used for the offset determination, where the last column in row n is calculated using the low dynamic "row-only" relation:

$$U(n,k) = \left\lfloor \frac{A(n,k-2)}{2} \right\rfloor + \frac{\left(n \cdot A(n,k-1) - \left(1 + \left\lfloor \frac{A(n,k-1)}{2} \right\rfloor + \left\lfloor \frac{A(n,k-2)}{2} \right\rfloor \right) \right)}{k-1} \tag{1880}$$

6.2.4   Frequency-to-time transformation

6.2.4.1   Long block transformation (ALDO window)

6.2.4.1.1   eDCT

The IDCT$_{IV}$ is identical to the DCT$_{IV}$ and is given by the following equation, with omitted normalization:

$$\tilde{x}^q(n) = \sum_{k=0}^{L-1} z^q(k) \cos\left[\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{L}\right], \quad n = 0,1,\ldots,L-1 \tag{1881}$$

6.2.4.1.2   Unfolding and windowing

The frame $\tilde{x}^q(n)$, $n = 0,\ldots,L-1$, coming from the inverse eDCT transform is unfolded in order to obtain two frames that can be used for overlap- add with the previous unfolded frame to remove the aliasing introduced by the folding process at the encoder.

Similar to the folding done at the encoder, unfolding and window decimation operations are combined in the same process to automatically resample the ALDO windows at 48 and 25.6 kHz while keeping perfect reconstruction conditions. The decimation factor and offset parameters are the same are the one used in the encoder.

The frame $\tilde{x}^q(n)$ issued from the eDCT inverse transform is unfolded into a block T_{2L}^q of length $2L$. The ALDO window is stored at a sampling rate corresponding to two frames of length N ($N \geq L$). The ratio between N and L is called the decimation factor (d_f). The unfolding and windowing process is illustrated in figure 109.
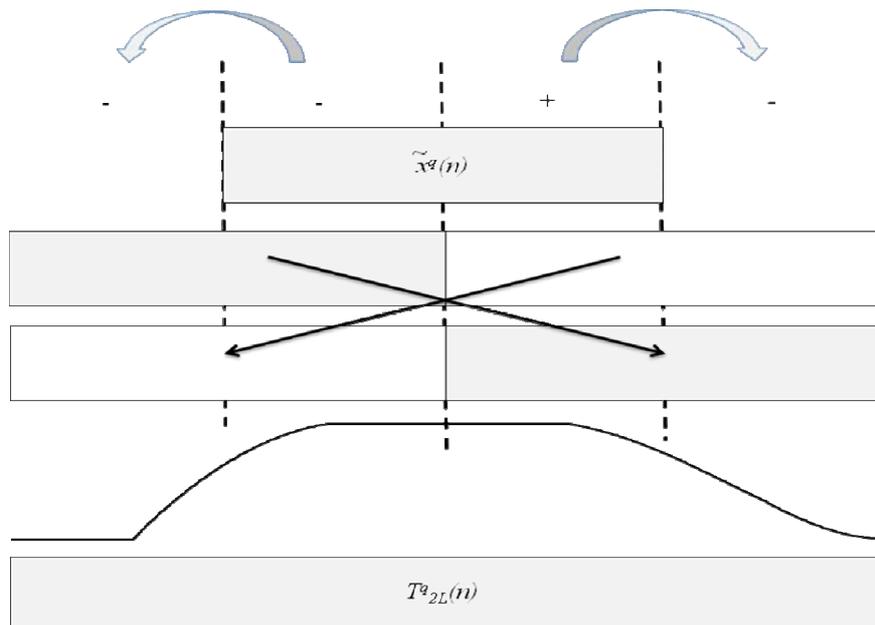
Figure 109: Unfolding and windowing with ALDO window.

The unfolded frame is obtained for $n = 0,\ldots,L/2-1$:

$$T_{2L}^q(n) = \tilde{x}^q\left(\frac{L}{2}+n\right)h_s\left(n.d_f + d\right), \tag{1882}$$

$$T_{2L}^q\left(\frac{L}{2}+n\right) = -\tilde{x}^q\left(L-n-1\right)h_s\left(\frac{N}{2}-1+(n+1)d_f - d\right), \tag{1883}$$

$$T_{2L}^q (L+n) = -\tilde{x}^q \left(\frac{L}{2} - n - 1 \right) h_s \left(N + n.d_f + d \right), \tag{1884}$$

$$T_{2L}^q \left(\frac{3L}{2} + n \right) = -\tilde{x}^q (n) h_s \left(\frac{3L}{2} - 1 + (n+1)d_f - d \right), \tag{1885}$$

where $h_s(n)$ is the time-reversed version of the ALDO window $h_a(n)$ used in the encoder

$$h_s(n) = h_a (2N - n - 1), \tag{1886}$$

d_f is the decimation factor and d is the offset.

For the 32 kHz case, to have perfect reconstruction, the ALDO window decimated from 48 to 16 kHz applied on one sample over 2, the other samples are weighted by a complementary window $h_{comp}(n)$. For this 32 kHz case, the unfolded frames are given by:

$$T_{2L}^q (2n) = \tilde{x}^q \left(\frac{N_{16}}{2} + 2n \right) h_a \left((2N_{16} - n) d_f - 1 - d \right), \tag{1887}$$

$$T_{2L}^q (2n+1) = \tilde{x}^q \left(\frac{L}{2} + 2n + 1 \right) h_{comp} \left(2N_{16} - n - 1 \right), \tag{1888}$$

$$T_{2L}^q \left(\frac{L}{2} + 2n + 1 \right) = -\tilde{x}^q \left(L - (2n+1) \right) h_a \left(\left(\frac{3N_{16}}{2} - 1 - n \right) d_f + d \right), \tag{1889}$$

$$T_{2L}^q \left(\frac{L}{2} + 2n \right) = -\tilde{x}^q \left(L - 2n - 1 \right) h_{comp} \left(\frac{3N_{16}}{2} - 1 - n \right), \tag{1890}$$

$$T_{2L}^q (L + 2n) = -\tilde{x}^q \left(\frac{L}{2} - 2n - 1 \right) h_a \left((N_{16} - n) d_f - d \right), \tag{1891}$$

$$T_{2L}^q (L + 2n + 1) = -\tilde{x}^q \left(\frac{L}{2} - (2n+1) - 1 \right) h_{comp} \left(N_{16} - n \right), \tag{1892}$$

$$T_{2L}^q \left(\frac{3L}{2} + 2n + 1 \right) = -\tilde{x}^q (2n+1) h_a \left(\left(\frac{N_{16}}{2} - n - 1 \right) d_f + d \right), \tag{1893}$$

$$T_{2L}^q \left(\frac{3L}{2} + 2n \right) = -\tilde{x}^q (2n) h_{comp} \left(\frac{N_{16}}{2} - n - 1 \right), \tag{1894}$$

where N_{16} is the length of the 16kHz frame, L is the length of MDCT core frame at 32kHz, $d_f = 3$ is the decimation factor and $d = 1$ is the offset.

6.2.4.1.2 Overlap-add

Finally, the output full-band signal is constructed by overlap-adding the signals $\tilde{x}^{(r)}(n)$ for two successive frames:

$$x^{(r)}(n) = T_{2L}^q{}^{(r-1)}(n+L) + T_{2L}^q{}^{(r)}(n), \quad n = 0, \ldots, 2L - 1 \tag{1895}$$

6.2.4.1.3   Pre-echo attenuation

A typical artefact in transform coding known as pre-echo is observed especially when the signal energy grows suddenly, like speech onsets or music percussions. The origin of pre-echoes is explained below. The quantization noise in the frequency domain is translated into the time domain by an inverse MDCT transform and an add/overlap operation. Thus the quantization noise is spread uniformly in the MDCT synthesis window. In case of an onset, the part of the input signal preceding the onset often has a very low energy compared to the energy of the onset part. Since the quantization noise level depends on the mean energy of the frame, it can be quite high in the whole synthesis window. In this case, the signal to noise ratio (SNR) is very low (often negative) in the low energy part. The quantization noise can be audible before the onset as an extra artificial signal called pre-echo. To prevent the pre-echo artefact, an attenuation scheme is necessary when there is a significant energy increase (attack or onset) in some part of the synthesis window, and the pre-echo reduction has to be performed in the low energy part of the synthesis window preceding the onset. In the following, this low energy part preceding an onset will be referenced as "echo zone". On the other hand the signal energy after pre-echo reduction should not lower than the mean energy in the preceding frames. However, if the preceding frame have low frequency spectrum, knowing that the pre-echo has often white noise like spectrum, even if the energy of the echo zone is reduced to the level of the previous frames the pre-echo is still audible in the higher frequencies.

To improve the pre-echo reduction, an adaptive spectral shaping filtering is applied in the pre-echo zone up to the detected attack or onset to eliminate undesirable higher frequency pre-echo noise. This adaptive spectral shaping filter is realized by a two-band filterbank: the decoded signal is decomposed into two sub-signals according to a frequency criterion to obtain two sub-bands and a pre-echo attenuation factor is calculated in the determined echo zone for each sample in both sub-bands. The attenuation factors of the sub-bands that determinate the spectral response of the filter are computed in function of several parameters of the full-band and sub-band signals as detailed below. The pre-echo attenuation is made in the sub-bands by applying these attenuation factors in the echo-zone. Finally the two attenuated sub-bands are combined to obtain the pre-echo attenuated decoded signal. The pre-echo attenuation is activated for received frames, when the previous frame was also received, and when the bitrate is not higher than 32 kbit/s.

A pre-echo in the current frame can be caused by a sharp onset in the current or the next frame, as the MDCT analysis window covers these two consecutive frames. An onset in the next frame can be detected by analysing the memory of inverse MDCT that will be used in the next frame in the overlap-add operation. A discrimination of echo/non-echo zones and the attenuation factor computation are based on two signals of the inverse MDCT transform: on the decoded output full-band signal $x^{(r)}(n)$, $n = 0,...,L-1$ and on the first $L_{premem} = \dfrac{7}{32}L$ un-windowed memory of inverse MDCT $x_{premem}(n) = \tilde{x}^{wq(r)}(n+L)$, $n = 0,...,L_{premem}-1$ that will be used in the next frame in the overlap-add operation to synthesize the output content for the next frame and the pre-echo reduction is done in echo zones preceding the onsets.

Decomposition in two sub-bands

The decoded signal $x^{(r)}(n)$ is decomposed in a lower and an upper frequency band sub-signals. The first, lower band sub-signal is obtained by a first filtering of the full-band signal by the low-pass filter

$$x_{LB}(n) = \begin{cases} 0.25x^{(r-1)}(L-1) + 0.5x^{(r)}(n) + 0.25x^{(r-1)}(n+1) & n = 0 \\ 0.25x^{(r-1)}(n-1) + 0.5x^{(r)}(n) + 0.25x^{(r-1)}(n+1) & n = 1,...,L-2 \\ 0.25x^{(r-1)}(n-1) + 0.5x^{(r)}(n) & n = L-1 \end{cases} \qquad (1896)$$

and the second, higher band sub-signal is obtained by subtracting the lower band sub-signal from the decoded signal:

$$x_{HB}(n) = x^{(r)}(n) - x_{LB}(n) \qquad (1897)$$

For the memory part $x_{premem}(n)$ only the higher-band component is computed as

$$x_{premem\_LB}(n) = \begin{cases} -0.25x^{(r-1)}(L-1) + 0.5x_{premem}(n) - 0.25x_{premem}(n+1) & n = 0 \\ -0.25x_{premem}(n-1) + 0.5x_{premem}(n) - 0.25x_{premem}(n+1) & n = 1,...,L_{premem}-2 \\ 0.25x_{premem}(n-1) + 0.5x_{premem}(n) & n = L_{premem}-1 \end{cases} \quad (1898)$$

Discrimination procedure of echo/non-echo zones

The discrimination procedure between echo zones and non-echo zones is based on the concatenated signal formed from $x^{(r)}(n)$, $n = 0,...,L\text{-}1$ and $x_{premem}(n) = \tilde{x}^{wq^{(r)}}(n+L)$, $n = 0,...,L_{premem}\text{-}1$. This signal is divided in sub-blocks and its temporal envelope is computed.

The current frame part of the concatenated signal, $x^{(r)}(n)$, $n = 0,...,L\text{-}1$ is divided into N_{sf} sub-blocks of $L_{sf} = L/N_{sf}$ samples where $N_{sf} = 8$ (2.5 ms sub-blocks). The temporal envelope $Es_{MDCT}(i)$ of this signal is computed as successive sub-block energies.

$$Es_{MDCT}(i) = \sum_{n=L_{sf}.i}^{L_{sf}.(i+1)-1} \left[x^{(r)}(n) \right]^2 , \ i = 0,...,N_{sf}-1 \tag{1899}$$

The memory part of the concatenated signal forms one sub-block, its energy is computed as

$$Es_{MDCT}(N_{sf}) = \sum_{n=0}^{L_{premem}-1} \left[x_{premem}(n) \right]^2 \tag{1900}$$

The energy of the first half and the first ¾ samples of each sub-blocks of the current frame are also memorized:

$$Es_{MDCT\_h}(i) = \sum_{n=L_{sf}.i}^{L_{sf}.(i+1)+L_{sf}/2-1} \left[x^{(r)}(n) \right]^2 , \ i = 0,...,N_{sf}-1 \tag{1901}$$

$$Es_{MDCT\_tq}(i) = \sum_{n=L_{sf}.i}^{L_{sf}.(i+1)+3L_{sf}/4-1} \left[x^{(r)}(n) \right]^2 , \ i = 0,...,N_{sf}-1 \tag{1902}$$

The temporal envelope of the higher band in the current frame is also computed:

$$Es_{MDCT\_hb}(i) = \sum_{n=L_{sf}.i}^{L_{sf}.(i+1)-1} \left[x_{hb}^{(r)}(n) \right]^2 , \ i = 0,...,N_{sf}-1 \tag{1903}$$

Then, $Es_{MDCT\_h}(i)$, $i = 0,...,N_{sf}-1$ is then modified as follows:

$$Es_{MDCT\_hb}(i) = \begin{cases} 2\left(Es_{MDCT}(i) - Es_{MDCT\_hb}(i) \right) & \text{if } Es_{MDCT\_h}(i) < Es_{MDCT}(i)/2 \\ Es_{MDCT}(i) & \text{otherwise} \end{cases} \tag{1904}$$

In this paragraph, index n is used for samples, and index i is used for sub-blocks.

In the concatenated signal the sub-block with maximal energy, including the memory sub-block, is also searched:

$$Max_{Es} = \max_{i=0,...,N_{sf}} Es_{MDCT}(i) \tag{1905}$$

The transition of the temporal envelope to a high-energy zone is detected in the sub-block with the index $Maxind_{Es}$ given by:

$$Maxind_{Es} = \arg \max_{i=0,...,N_{sf}} Es_{MDCT}(i) \tag{1906}$$

Note that when $Maxind_{Es} = 0$ either no pre-echo attenuation is made or the pre-echo attenuation of the previous frame is finished on the first samples of the current frame.

The zero-crossing rate $zcr(i)$, $i = 0,\ldots,N_{sf} - 1$ is also computed for each sub-block. A zero-crossing is detected when the product of two consecutive samples is smaller or equal to 0. The parameter $zcr(i)$, $i = 0,\ldots,N_{sf} - 1$ is defined as the number of times when the following condition is verified:

$$x^{(r)}(iN_{sf} + n)x^{(r)}(iN_{sf} + n) \leq 0 , \quad n = 1,\ldots,L_{sf} - 1 \tag{1907}$$

The zero crossings between two consecutive sub-blocks count for the next sub-block. The zero crossing rate of the memory part is also computed.

The maximum length without zero crossing $nzcr(i)$, $i = 0,\ldots,N_{sf} - 1$ is also stored for each sub-block. A period without zero crossing that covers a sub-block border is taken into account for the previous sub-block.

The maximal energy Max_{Es} is compared to that of the preceding sub-blocks:

$$r_{Es}(i) = \frac{Max_{Es}}{Es_{MDCT}(i)} , \quad i = 0,\ldots,Maxind_{Es} - 1 \tag{1908}$$

The low energy sub-blocks preceding the sub-block in which a transition has been detected with $r_{Es}(i) > 16$ are determined as echo zone. However in the following cases the sub-block is considered as non-echo zone:

if $Es_{MDCT\_hb}(i) > 100(Es_{prev\_nc} + 500000)$

or

if $Es_{MDCT\_hb}(i) > 100(Es_{prev\_nc} + 500000)$ and $zcr(i) < lim_{zcr}$

or

if $Es_{MDCT\_hb}(i) > 100(Es_{prev\_nc} + 500000)$ and $6Es_{MDCT\_tq}(i) < Es_{MDCT}(i)$

where, computed in the previous frame and memorised,

$$Es_{prev\_nc} = \max\left(\frac{\sum_{i=0}^{N_{sf}-1} Es_{MDCT}(i)}{N_{sf}} , 2\frac{\sum_{i=N_{sf}/2}^{N_{sf}-1} Es_{MDCT}(i)}{N_{sf}} \right) \tag{1909}$$

and $lim_{zcr} = 10$ for narrowband signals and 16 otherwise.

Even the previous sub-blocks are considered as non-echo zone if their energy is higher than $Es_{MDCT}(i)/2$.

The pre-echo attenuation of low energy sub-block determined as echo zone is made by multiplying the two sub-band signals, the lower band $x_{LB}(n)$ and the higher band $x_{HB}(n)$ by attenuation factors $g_{pre}(n)$ and $g_{pre\_hb}(n)$ respectively, where $g_{pre}(n)$ and $g_{pre\_hb}(n)$ are determined as a function of the temporal envelope of the concatenated signal $Es_{MDCT}(i)$, $i = 0,\ldots,N_{sf}$.

For each sample of the echo zone sub-blocks, these gains are set to 0.01 if $r_{Es}(i) > 32$ and to 0.1 otherwise. For the other sub-blocks, the initial gains are set to 1, they form the non-echo zone. Following this $Maxind2_{Es}$ is set as the index of the first non-echo sub-block (where the initial gain is equal to 1).
A false alarm detection is made at this point. If the last pre-echo attenuation gain in the previous frame is higher than 0.5 and in the current frame only one sub-block has attenuation gain of 0.1 and the other gains are 0, $Maxind2_{Es}$ is set to 0.

The initial pre-echo attenuation gains depend also on the energy of the previous frame: a minimal attenuation value for each echo zone sub-block and for both sub-bands are also fixed as a function of the temporal envelope of the

reconstructed signal of the previous frame. This value is fixed in a way that the attenuated sub-block energy in the sub-band cannot be lower than the pre-echo attenuation gain compensated mean energy of the previous frame in that sub-band, to preserve background noise energy. In the lower band:

$$
g_{pre'}(n) = \begin{cases} \max\left(0.01, \min_{i=0,\dots,Maxind2_{Es}-1}\left(\sqrt{\dfrac{Es_{prev}}{Es_{MDCT}(i)}},1\right)\right) & \text{if } r_{ES}(i) > 32 \\[2em] \max\left(0.1, \min_{i=0,\dots,Maxind2_{Es}-1}\left(\sqrt{\dfrac{Es_{prev}}{Es_{MDCT}(i)}},1\right)\right) & \text{if } 16 < r_{ES}(i) \le 32 \\[2em] 1 & \text{if } r_{ES}(i) \le 16 \end{cases}
\tag{1910}
$$

for $n = L_{sf}.i,\dots,L_{sf}.(i+1)-1$ and where Es_{prev} was computed in the previous frame as:

$$
Es_{prev} = \max\left(\frac{\displaystyle\sum_{i=0}^{N_{sf}-1} Es_{MDCT}(i) g_{pre'}(L_{sf}.i)^2}{N_{sf}}, 2\frac{\displaystyle\sum_{i=N_{sf}/2}^{N_{sf}-1} Es_{MDCT}(i) g_{pre'}(L_{sf}.i)^2}{N_{sf}}, 2\frac{\displaystyle\sum_{i=N_{sf}-2}^{N_{sf}-1} Es_{MDCT}(i)}{2}\right)
\tag{1911}
$$

However $g_{pre'}(n)$ is set to 1 if $zcr(i) < lim_{zcr}/2$ or $\max nzcr(i) > L/24$.

In a similar way the initial pre-echo attenuation gain for the higher band signal is computed as:

$$
g_{pre\_hb'}(n) = \begin{cases} \max\left(0.01, \min_{i=0,\dots,Maxind2_{Es}-1}\left(\sqrt{\dfrac{Es_{prev\_hb}}{Es_{MDCT\_hb}(i)}},1\right)\right) & \text{if } r_{ES}(i) > 32 \\[2em] \max\left(0.1, \min_{i=0,\dots,Maxind2_{Es}-1}\left(\sqrt{\dfrac{Es_{prev\_hb}}{Es_{MDCT\_hb}(i)}},1\right)\right) & \text{if } 16 < r_{ES}(i) \le 32 \\[2em] 1 & \text{if } r_{ES}(i) \le 16 \end{cases}
\tag{1912}
$$

for $n = L_{sf}.i,\dots,L_{sf}.(i+1)-1$ where

$$
Es_{prev\_hb} = \max\left(\frac{\displaystyle\sum_{i=0}^{N_{sf}-1} Es_{MDCT\_hb}(i) g_{pre\_hb'}(L_{sf}.i)^2}{N_{sf}}, 2\frac{\displaystyle\sum_{i=N_{sf}/2}^{N_{sf}-1} Es_{MDCT\_hb}(i) g_{pre\_hb'}(L_{sf}.i)^2}{N_{sf}}, 2\frac{\displaystyle\sum_{i=N_{sf}-2}^{N_{sf}-1} Es_{MDCT\_hb}(i)}{2}\right)
$$

$$
\tag{1913}
$$

Note that the initial attenuation gain in both the lower band and the higher band are identical for each samples of a sub-block.

Before applying the pre-echo attenuation gains the position of the onset is refined. If the onset was detected in the current frame, each sub frames from index $Maxind2_{Es}$ to $N_{sf}-1$ are divided into N_{ssf} sub-sub-blocks where

$N_{ssf} = 4$ if the sampling frequency is 8 kHz and $N_{ssf} = 8$ otherwise. If $Maxind2_{Es} = 0$ only the first sub-block is considered.

The energy of these sub-sub-blocks is computed:

$$Eshr_{MDCT}(j) = \sum_{n=L_{sf}.Maxind2_{Es}+L_{ssf}.j}^{L_{sf}.Maxind2_{Es}+L_{ssf}.(j+1)-1} \left[x^{(r)}(n)\right]^2 , \quad j = 0,\ldots,Num_{ssf}-1 \tag{1914}$$

where

$$L_{ssf} = \frac{L_{sf}}{N_{ssf}} \tag{1915}$$

and

$$Num_{ssf} = \begin{cases} N_{ssf} & \text{if } Maxind2_{Es} = 0 \\ \left(N_{sf} - Maxind2_{Es}\right)N_{ssf} & \text{otherwise} \end{cases} \tag{1916}$$

When the onset was detected in the future memory part $x_{premem}(n)$, only the first L_{sf} samples are examined and $Num_{ssf} = N_{ssf}$ and

$$Eshr_{MDCT}(j) = \sum_{n=L_{ssf}.j}^{L_{ssf}.(j+1)-1} \left[x_{premem}(n)\right]^2 , \quad j = 0,\ldots,Num_{ssf}-1 \tag{1917}$$

The maximum of these values is searched:

$$Eshr_{\max} = \max_{j=0,\ldots,Num_{ssf}} \left(Eshr_{MDCT}(j)\right), \quad j = 0,\ldots,Num_{ssf}-1 \tag{1918}$$

The values $Eshr_{MDCT}(j)$ are compared to adaptive thresholds. The first one is independent of the sub-sub-block index:

$$Thres_{MDCT1} = \frac{Eshr_{\max}}{8} \tag{1919}$$

The second one is computed as:

$$Thres_{MDCT2}(j) = \begin{cases} \max(Eshr_{MDCT}(0), Es_{prev\_nc}) & \text{if } Maxind2_{Es} = 0 \text{ and } j = 0 \\ \max\left(\dfrac{Es_{prev\_nc}+4\sum\limits_{k=0}^{j-1}Eshr_{MDCT}(k)}{j+1}, Es_{prev\_nc}\right) & \text{if } Maxind2_{Es} = 0 \text{ and } j > 0 \\ \max(Eshr_{MDCT}(0), mcrit) & \text{if } Maxind2_{Es} > 0 \text{ and } j = 0 \\ \max\left(\dfrac{Eshr_{MDCT}(Maxind2_{Es}-1)+4\sum\limits_{k=0}^{j-1}Eshr_{MDCT}(k)}{j+1}, Es_{prev\_nc}\right) & \text{if } Maxind2_{Es} > 0 \text{ and } j > 0 \end{cases} \tag{1920}$$

where

$$mcrit = Es_{MDCT}(Maxind2_{Es}-1).g_{pre'}\left((Maxind2_{Es}-1)L_{sf}\right) \tag{1921}$$

If $Max_{Es}/80 > mcrit$ and $zcr(Maxind2_{Es}) > lim_{zcr}$ this value is modified as:

$$mcrit = Max_{Es}/80 \tag{1922}$$

Initially the starting position of the onset for both the lower and the higher band is the beginning of the sub-block $Maxind2_{Es}$. This position is delayed by L_{ssf} samples by sub-sub-blocks as long as $Eshr_{MDCT}(j) < \min(Thres_{MDCT1}, Thres_{MDCT2}(j))$. The pre-echo attenuation gain of these samples moved from the

non-echo zone to the echo-zone is set equal to the gain of last sample of the original echo zone $g_{pre'}\big((Maxind2_{Es}-1)L_{sf}\big)$ and $g_{pre\_hb'}\big((Maxind2_{Es}-1)L_{sf}\big)$ respectively in the 2 sub-bands. In the following these new samples in the pre-echo zone are considered as the part of the last pre-echo zone sub-block (index $Maxind2_{Es}-1$), the length of this sub-block $Maxind2_{Es}-1$ can be longer than L_{sf}.

To avoid false pre-echo detection, the energies of the last 2 or 3 sub-blocks preceding the onset is verified for both the full-band and the high-band signals: the regression coefficient for these sub-blocks energies is computed by the least squares estimation technique and compared to thresholds. If at least one regression coefficient is lower to its threshold the pre-echo attenuation is inhibited. In fact it is checked whether the sub-blocks preceding the onset have stable or increasing energy, this is always true for pre-echos. For easy comparison to threshold the regression coefficients are normalised by the sub-band energies when the threshold is different to 0. If the threshold is 0, only the sign of the regression coefficient is checked, no normalisation is needed.

When the onset is detected in the first or second sub-block this verification is not possible.

When the onset is detected in the third sub-block only the high-band regression coefficient b_{pre_hb2} is computed and compared to the threshold $thpre_{hb2}=0$. As only the sign is checked here no normalization is needed for the regression coefficient:

$$b_{pre\_hb2} = Es_{MDCT\_hb}(1) - Es_{MDCT\_hb}(0)$$

If $b_{pre\_hb2} < thpre_{hb2}$ the pre-echo attenuation in the pre-echo zone is inhibited.

When the onset is detected in the fourth or later sub-block both the full-band regression coefficient b_{pre3} and the normalized high-band regression coefficient $b_{pre\_hb3\_norm}$ are computed on the last 3 sub-blocks preceding the onset and they are compared to the thresholds $thpre_3=0$ and $thpre_{hb3}=0.2$ respectively. Let's note the index of the sub-block where the onset is detected id, $id>2$.

In the full-band only the sign is checked, no normalization of the regression coefficient is needed:

$$b_{pre3} = Es_{MDCT}(id-1) - Es_{MDCT}(id-3)$$

In the higher band the normalized regression coefficient is estimated as:

$$b_{pre\_hb3\_norm} = \frac{3\big(Es_{MDCT\_hb}(id-1) - Es_{MDCT\_hb}(id-3)\big)}{2\big(Es_{MDCT\_hb}(id-1) + Es_{MDCT\_hb}(id-2) + Es_{MDCT\_hb}(id-3)\big)}$$

The comparison $b_{pre\_hb3\_norm} < 0.2$ is equivalent to :

$$Es_{MDCT\_hb}(id-1) - Es_{MDCT\_hb}(id-3) < \frac{2}{15}\big(Es_{MDCT\_hb}(id-1) + Es_{MDCT\_hb}(id-2) + Es_{MDCT\_hb}(id-3)\big)$$

If the $b_{pre3} < thpre_3$ or $b_{pre\_hb3\_norm} < thpre_{hb3}$ the pre-echo attenuation in the pre-echo zone is inhibited.

The pre-echo attenuation functions $g_{pre'}(n)$ and $g_{pre\_hb'}(n)$ are stair-like, the gain is constant within a sub-block. To avoid annoying noise due to this discontinuity, the final pre-echo attenuation gain for the lower band $g_{pre}(n)$ is obtained by linear smoothing of the initial pre-echo attenuation gain $g_{pre'}(n)$ introducing *smoothlen* intermediate levels between the gains of consecutive sub-blocks. For narrow band signals *smoothlen* = 20, for other bandwidths *smoothlen* = 4. This smoothing is done before the detected onset position and at the beginning of each sub-block. For the first sub-block the smoothing is done between the memorized last gain value of the previous frame and the gain of the first sub-block of the current frame. If the onset position is detected in the next frame no smoothing is done at the end of the frame, this will be done at the beginning of the next frame. For example at the beginning sub-block i, $i=1,\ldots,N_{sf}-1$ and if the gains determined for the sub-blocks $i-1$ and i are g_1 and g_2 respectively, for wide band signals (*smoothlen* = 4) the gains are smoothed in the following way:

Before smoothing:

index	...	$i.L_{sf} - 2$	$i.L_{sf} - 1$	$i.L_{sf}$	$i.L_{sf} + 1$	$i.L_{sf} + 2$	$i.L_{sf} + 3$	$i.L_{sf} + 4$	$i.L_{sf} + 5$	...
gain	...	g_1	g_1	g_2	g_2	g_2	g_2	g_2	g_2	...

After smoothing:

index	...	$i.L_{sf} - 2$	$i.L_{sf} - 1$	$i.L_{sf}$	$i.L_{sf} + 1$	$i.L_{sf} + 2$	$i.L_{sf} + 3$	$i.L_{sf} + 4$	$i.L_{sf} + 5$	...
gain	...	g_1	g_1	$\dfrac{4g_1 + g_2}{5}$	$\dfrac{3g_1 + 2g_2}{5}$	$\dfrac{2g_1 + 3g_2}{5}$	$\dfrac{g_1 + 4g_2}{5}$	g_2	g_2	...

In the higher band no smoothing is necessary, $g_{pre\_hb}(n) = g_{pre\_hb'}(n)$.

In both sub-band, the pre-echo is attenuated in the echo-zone by applying these gains to the sub-signals:

$$x_{LB\_pre}(n) = g_{pre}(n)x_{LB}(n) \tag{1923}$$

$$x_{HB\_pre}(n) = g_{pre\_hb}(n)x_{HB}(n) \tag{1924}$$

The final pre-echo attenuated synthesized signal $x_{pre}{}^{(r)}(n)$ is obtained by combining the two attenuated sub-signals:

$$x_{pre}{}^{(r)}(n) = x_{LB\_pre}(n) + x_{HB\_pre}(n) \tag{1925}$$

6.2.4.2   Transient location dependent overlap and transform length

The configuration for the overlap and transform lengths is depends on the overlap code for the current frame and on the overlap code for the previous frame as described in subclause 5.3.2.3, where the overlap code is obtained as described in subclause 6.2.2.2.1.

6.2.4.3   Short block transformation

6.2.4.3.1   Short window transform in TDA domain

This processing is done when the Transient mode is selected, the spectrum is first de-interleaved into four spectra $\hat{X}_M^{[m]}(k)$ with m = 0,...,3. This operation is the inverse of the interleaving performed in the encoder, see subclause 5.3.2.4.1.3.

The four spectra corresponding to short 5-ms transforms are first transformed to the time-aliased domain using the short inverse DCT$_{IV}$ transforms. The obtained signals are denoted $\hat{x}_{TDA}^{[m]}(n)$ $m = 0,\ldots,3$ and are each of a length $L/4$.

The obtained time-domain aliased signals are further expanded into the time domain by using the inverse time-domain aliasing operation. This operation can be seen as a pseudo-inverse of the matrix used in equation (8) (with L replaced by $L/4$).

Formally, this is performed according to:

$$\hat{\mathbf{x}}_{ITDA}^{[m]} = \begin{bmatrix} 0 & \mathbf{I}_{L/8} \\ 0 & -\mathbf{J}_{L/8} \\ -\mathbf{J}_{L/8} & 0 \\ -\mathbf{I}_{L/8} & 0 \end{bmatrix} \hat{x}_{TDA}^{[m]} \tag{1926}$$

The length of the resulting signal for each sub-frame index m is equal to double the length of the input spectrum, i.e., $L/2$.
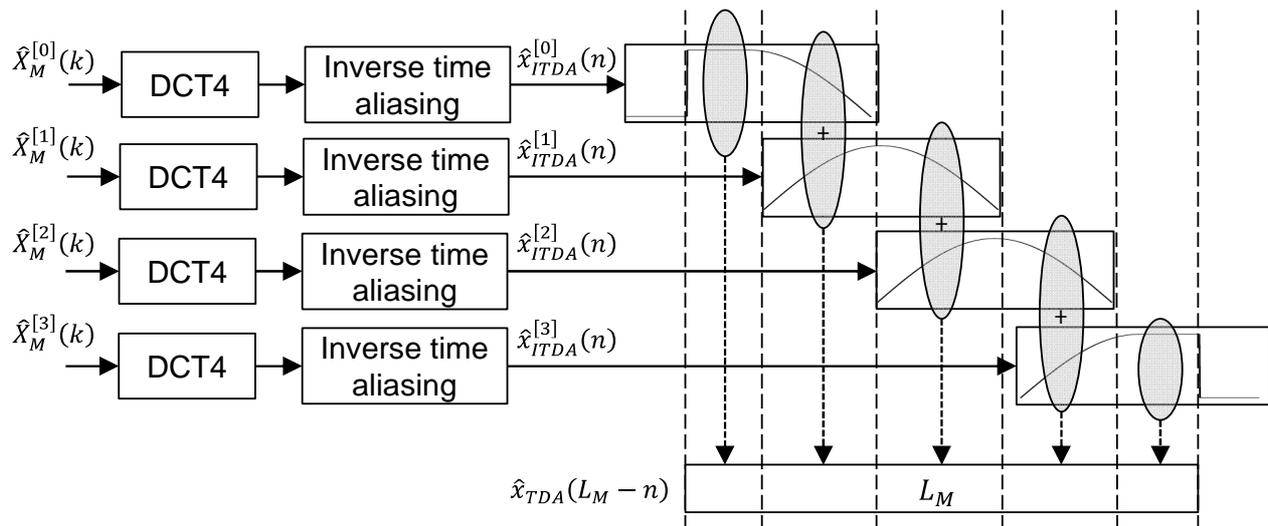
Figure 110: Algorithm for inverse transform in the case of transient mode.

The resulting time domain aliased signals for each sub-frame are windowed using the same configuration of windows as those in the encoder. The resulting windowed signals are overlap-added. Note that the window for the first m = 0 and last m = 3 sub-frame is zero. This is due to the zero padding that is used in the encoder. These two frame edges do need to be computed and are effectively dropped. The resulting signal of the overlap-add operations of all sub-frames is reordered using the inverse operation performed in the encoder, which leads to the signal $\hat{x}_{TDA}(n)$, $n = 0,\ldots,L-1$. An overview of these operations is shown in figure 110. Then windowing and overlap-add are performed on $\hat{x}_{TDA}(n)$ the same as for the long window transform in subclause 5.3.2.2.

6.2.4.3.2 Short window transform for MDCT based TCX

After choosing the transform and overlap length configuration for transient frame as described in subclause 6.2.4.2 each sub-frame (TCX5 or TCX10) is windowed and transformed using the inverse MDCT, which is implemeted using $IDCT_{IV}$ and inverse TDA.

6.2.4.4 Special window transitions

The overlap mode FULL is used for transitions between long ALDO windows and short symmetric windows (HALF, MINIMAL) for short block configurations (TCX10, TCX5) described in subclause 6.2.4.2. The transition window is derived from the ALDO and sine window overlaps.

6.2.4.4.1 ALDO to short transition

The left part of the transition window uses the left slope of the ALDO synthesis window (short slope), which has a length of 8.75ms (see figure 111).

For the right part of the transition window HALF or MINIMAL overlap is used.
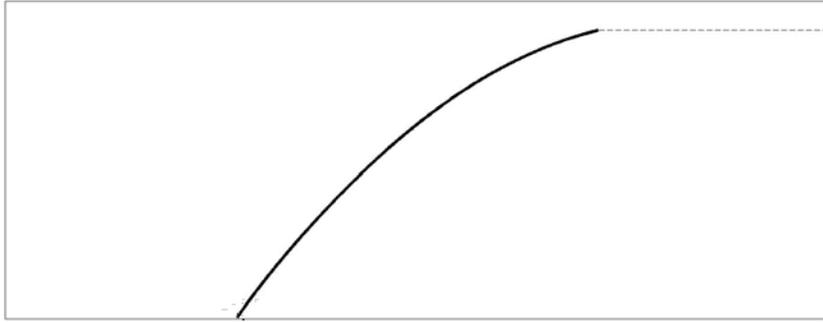
Figure 111: ALDO to short transition

6.2.4.4.2 Short to ALDO transition

HALF or MINIMAL overlap is used for the left part of the transition window.

The right overlap of the transition window has a length of 8.75ms (FULL overlap) and is derived from the ALDO window. The right slope of the ALDO synthesis window (long slope) is first shortened to 8.75ms by removing the last 5.625ms. Then the last 1.25ms of the remaining slope are multiplied with the 1.25ms MINIMAL overlap slope to smooth the edge. The resulting 8.75ms slope is depicted in figure112 .
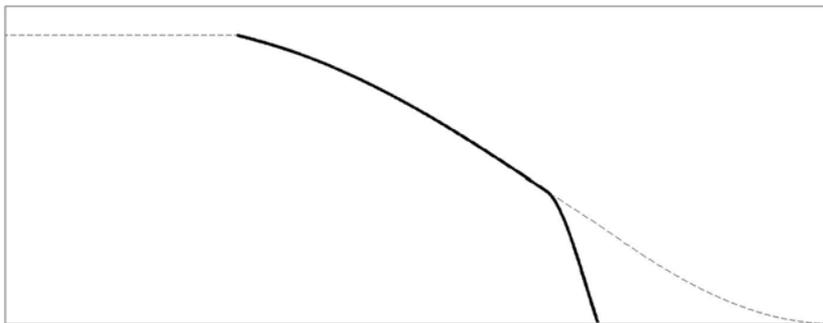
Figure 112: Short to ALDO transition

6.2.4.5 Low Rate MDCT Synthesis

In addition to the full MDCT synthesis of length $L_{TCX}^{(full)}$, which gives an output signal at the configured output sampling rate sr_{out} , a further MDCT synthesis of the lower part of the spectrum is performed to obtain an output signal at the CELP sampling rate sr_{celp} . The low rate output is required for switching from TCX to ACELP. An MDCT synthesis of length $L_{TCX}^{(celp)}$ is performed on the lowest $L_{TCX}^{(celp)}$ MDCT coefficients. In case the output sampling rate is set lower than the CELP sampling rate (so that $L_{TCX}^{(celp)} < L_{TCX}^{(full)}$), the spectrum is padded with zeroes to the required length $L_{TCX}^{(celp)}$. The low rate synthesis transform is performed as defined above with the only difference being the transform length.

6.3 Switching coding modes in decoding

6.3.1 General description

This clause describes all transitions between coding modes including changes for sample rates, bit rates and audio bandwidths for the decoding process. The transitions between CELP coding mode and MDCT coding mode within the same bit rate and audio bandwidth are described in 6.3.2 and 6.3.3.

The handling of sample rate changes within the CELP or LP-based coding mode and MDCT-based TCX mode is described in 6.3.4.

The switching between primary and AMR-WB IO modes is described in 6.3.5.

The handling of transitions in the context of bit rate switching is described in 6.3.6.

Finally, the transition between NB, WB, SWB and FB are described in 6.3.7.

6.3.2 MDCT coding mode to CELP coding mode

When a CELP encoded frame is preceded by a MDCT based encoded frame, the memories of the CELP encoded frame have to be updated before starting the decoding of the CELP frame, similarly to the encoder case (see clause 5.4.2).

Additionally, cross-fading is applied in the time-domain at the output sampling rate sr_{out} to avoid any discontinuities between the MDCT based output and the CELP based output including bandwidth extension.

The CELP memories update and the cross-fading are performed depending on the bitrate and the previous encoding mode. In general three different MDCT to CELP (MC1 to MC3) transitions are supported. The table in clause 5.4.2 describes which transition mode is used for a specific configuration. The different decoding cases are described in detail in the following sub-clauses.

6.3.2.1 MDCT to CELP transition 1 (MC1)

MC1 is used when the previous frame was decoded with HQ MDCT and the current frame is decoded with CELP. The CELP state variables are reset in the current frame to predetermined (fixed) values. In particular the following memories are reset to 0 in the CELP decoder:

- Resampling memories of the CELP synthesis

- Pre-emphasis and de-emphasis memories

- LPC synthesis memories

- Past excitation (adaptive codebook memory)

- Bass post-filter memories

The old LPC coefficients and associated representations (LSP, LSF) and CELP gain quantization memories are reset to predetermined (fixed) values. The CELP decoder in the current frame is forced to operate in Transition coding (TC), i.e. without using an adaptive codebook from the previous frame. Since the LPC coefficients from the previous frame are not available, only one set of LPC coefficients corresponding to the end of frame are decoded and using for all subframes in the current frame.

To avoid discontinuities at the output sampling rate between the decoded HQ MDCT signal in the previous frame and the decoded CELP signal (including time-domain BWE) in the current frame, cross-fading (i.e. overlap-add) is used.

The decoded HQ MDCT signal could be windowed to compensate for the MDCT synthesis window, however in practice this compensation is not done. Note that the synthesis of the CELP decoder is more delayed than the synthesis of the MDCT decoder as shown in Figure 112a; the time difference is denoted here D. In the current CELP frame, the first samples are replaced by the HQ MDCT synthesis from the previous frame (D samples). The unweighted HQ MDCT aliased memory is overlap-added with the CELP output.
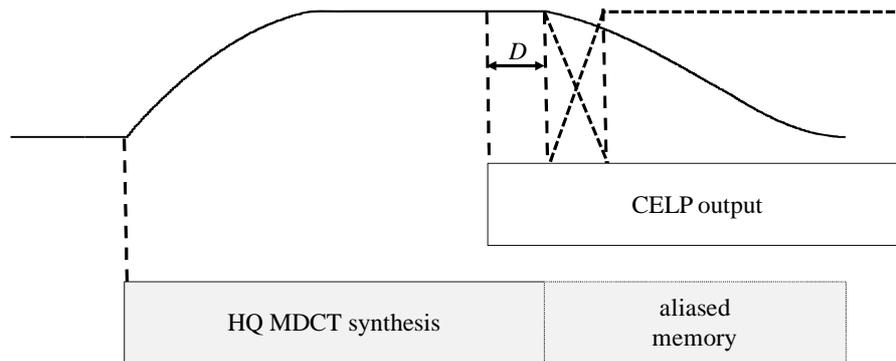
Figure 112a: Overlap-add in MDCT to CELP transition.

6.3.2.2   MDCT to CELP transition 2 (MC2)

As described in clause 6.2.4.5, the MDCT based TCX decoder generates two time-domain signals, one at the output sampling rate sr_{out} and one at the CELP sampling rate sr_{celp}. The signal at the CELP sampling rate is used to update the CELP memories, similarly to the encoder case (see clause 5.4.2.2). It is also used at the decoder side to update the memories of the CLDFB based resampler that is used to resample the decoded CELP signal. The signal at the output sampling rate is the signal that is actually sent to the decoder output. To avoid discontinuities between this MDCT based TCX signal and the decoded CELP signal at the output sampling rate, cross-fading is used.

The decoded CELP signal at the output sampling rate is obtained after CELP decoding at the CELP sampling rate, CLDFB based resampling and time-domain bandwidth extension decoding (see clause 6.1). Both the CLDFB based resampling and the time-domain bandwidth extension decoding introduce a delay. Consequently, the decoded CELP frame is delayed compared to the MDCT based TCX frame and an overlap between the two frames is then introduced. This overlap is used to perform a cross-fade and thus to avoid any discontinuities between the two frames. The output signal is given by

$$s_{out}(n) = \begin{cases} s_{mdct}(n), & n = 0,...,d_{LTP}-1 \\ \dfrac{\Delta-n}{\Delta-d_{LTP}} s_{mdct}(n) + \dfrac{n-d_{LTP}}{\Delta-d_{LTP}} s_{celp}(n), & n = d_{LTP},...,\Delta-1 \\ s_{celp}(n), & n = \Delta,....,N_{out}-1 \end{cases} \qquad (1927)$$

with $s_{out}(n)$ is the output signal, $s_{mdct}(n)$ is the MDCT based TCX signal, $s_{celp}(n)$ is the CELP signal, d_{LTP} is the delay of the TCX LTP postfilter (0.25ms), Δ is the delay introduced by the CLDFB resampler and the time-domain BWE (1.25ms if $sr_{out} = 8kHz$ and 2.3125ms otherwise), and N_{out} is the frame length at the output sampling rate (20ms).

6.3.2.3   MDCT to CELP transition 3 (MC3)

Similarly to MC1, the CELP memories are either reset or extrapolated similarly as in the encoder (see clause 5.4.2.3).

To avoid discontinuities between the decoded MDCT based TCX signal and the decoded CELP signal (including time-domain BWE) at the output sampling rate, cross-fading is used. The same approach as used in clause 6.3.2.1 is used.

6.3.3   CELP coding mode to MDCT coding mode

When a MDCT encoded frame is preceded by a CELP encoded frame, a beginning portion of the MDCT encoded frame cannot be reconstructed properly due to the aliasing introduced by the missing previous MDCT encoded frame. As already described in clause 5.4.3, two approaches are used to solve this problem, depending on the MDCT based coding mode (either MDCT based TCX or HQ MDCT). The decoder part of these two approaches is described in detail in the following sub-clauses.

6.3.3.1 CELP coding mode to MDCT based TCX coding mode

If MDCT based TCX is used in the current frame and if the previous frame was encoded with CELP, the MDCT based TCX frame length is increased and the left part of the MDCT window is modified, as already described in clause 5.4.3.1.

At the decoder side, the MDCT coefficients are decoded as described in clause 6.2.2. The decoded MDCT coefficients are then transformed back to the time-domain as described in clause 6.2.4. Two inverse transforms are performed and two time-domain signals are generated, as described in clause 6.2.4.5. One signal is generated at the CELP sampling rate and follows the previous decoded CELP signal at the CELP sampling rate. The other signal is generated at the output sampling rate and follows the previous decoded CELP signal at the output sampling rate (after CLDFB resampling and time-domain BWE). The decoding of the CELP signal (including the time-domain BWE) was described in clause 6.1.

To avoid any discontinuities that could be introduced in the decoded time-domain signal at the border between the two frames, a smoothing mechanism is applied. This algorithm is described in detail in the following.

Let's first assume the following notations. The frame length at the CELP sampling rate is noted N_{celp}. The decoded CELP signal at the CELP sampling rate is noted $s_{celp}(n)$ with $n = -N_{celp},...,-1$. The decoded MDCT signal (including the windowed portion overlapping with the previous CELP frame) is noted $s_{mdct}(n)$ with $n = -N_T,...,N_{celp}-1$, $N_T = 0.00125 sr_{celp}$ is the length of the segment where both the MDCT signal and the CELP signal overlap (it is also equal to the length of the sine window used in the left part of the transition window as described in clause 5.4.3.1), and sr_{celp} is the CELP sampling rate. The LPC synthesis filter used in last subframe of the previous CELP frame is noted $1/A_{celp}(z)$ with $A_{celp}(z) = \sum_{m=0}^{M} a_{celp}^m z^{-m}$ and $M = 16$ is the LPC filter.

A modified CELP signal is first computed by performing an overlap-add operation with the decoded MDCT signal on the overlap region and by artificially compensating the aliasing introduced by the decoded MDCT signal using the decoded CELP signal. The modified CELP signal can be defined as follow

$$\hat{s}_{celp}(n) =$$
$$\begin{cases} s_{celp}(n), & n = -N_{celp},...,-N_T-1 \\ s_{mdct}(n) + s_{celp}(n)w_T(-n-1)w_T(-n-1) + s_{celp}(-n-N_T-1)w_T(n+N_T)w_T(-n-1), & n = -N_T,...,-1 \end{cases} \quad (1928)$$

with $w_T(n)$ is the sine window used in the left-part of the transition window as described in clause 5.4.3.1. Contrary to the non-modified CELP signal case, discontinuities are significantly reduced (or even completely supressed in most cases) in the modified CELP signal preceding the MDCT signal, due to the overlap-add operation. However, the modified CELP signal cannot be used directly to generate the decoder output signal of the current frame, because it would introduce an additional decoder delay equal to the overlap length. Instead, the modified CELP signal is used only to generate a zero-input-response (ZIR) of the LPC synthesis filter. This ZIR is then used to modify the decoded MDCT signal, reducing significantly (or even removing in most cases) the possible discontinuity, and without introducing any additional decoder delay. The ZIR $s_{zir}(n)$ of the LPC synthesis filter $1/A_{celp}(z)$ is generated by first computing the memory of the LPC synthesis filter as follow

$$s_{zir}(n) = S_{celp}(n) - \hat{S}_{celp}(n), \quad n = -M,...,-1 \quad (1929)$$

and then computing the zero-input-response as follow

$$s_{zir}(n) = -\sum_{m=1}^{M} a_m s_{zir}(n-m), \quad n = 0,..., N_{zir}-1 \quad (1930)$$

with $N_{zir} = 64$ is the number of generated ZIR samples. The ZIR is then windowed such that its amplitude always decreases to 0, producing the windowed ZIR

$$\hat{s}_{zir}(n) = s_{zir}(n)\frac{N_{zir}-n}{N_{zir}}, \quad n = 0,..., N_{zir}-1 \quad (1931)$$

Finally, the windowed ZIR is added to the beginning portion of the decoded MDCT signal, corresponding to the time samples $n = 0,...,N_{zir} - 1$.

The same smoothing mechanism is applied to the decoded signals at the output sampling rate, with the exception that the ZIR is not re-computed but obtained by resampling the ZIR computed at the CELP sampling rate. The resampling is performed using linear interpolation as described in clause 5.4.4.4. The resampled ZIR is then added to the decoded MDCT signal at the output sampling rate.

The smoothing mechanism described above ensures a smooth transition between the CELP and MDCT signals at the output sampling rate, but only in the CELP bandwidth part. Due to the delay introduced by the time-domain BWE, a gap is introduced in the high frequency region as explained in clause 6.1.5.1.13.1. To fill this gap, a transition signal is generated as described in clause 6.1.5.1.13.1. This transition signal is long enough to cover not only the gap but also an additional signal portion following the gap. This additional signal portion is then used to perform a cross-fading with the decoded MDCT signal, ensuring smooth transition at the output sampling rate.

6.3.3.2   CELP coding mode to HQ MDCT coding mode

When the previous frame is CELP and the current frame is to be coded by HQ MDCT, the current frame is a transition frame in which two types of decoding are used:

- Constrained CELP coding and (when required) simplified time-domain BWE coding

- HQ MDCT coding with a modified window

Constrained CELP decoding means here that CELP is restricted to decode only a subset of CELP parameters, to reuse parameters (LPC coefficients) from the previous CELP frame, and to cover only the first subframe of the current frame. These constraints are set to minimize the bit budget taken by continuing CELP decoding in the current frame, this bit budget being taken out of HQ MDCT decoding.

As shown in Figure 112b, the transition frame includes at the decoder side a gap between the previous output frame (decoded by CELP) and the decoded signal with only the contribution from HQ MDCT. The length of this gap at the decoder is 4.375 ms, which corresponds to 10-5.625 ms (10 ms for ¼ of MDCT window support – 5.625 ms which is the length of the zero segment at the beginning of the ALDO synthesis window). In addition, an overlap period of 1,825 ms is used to attenuate discontinuities between CELP and HQ MDCT decoded signal. The total transition region between CELP and HQ MDCT in the decoder (grey zone decoder in Figure 112b) is 4.375+1.825 = 6.25 ms..
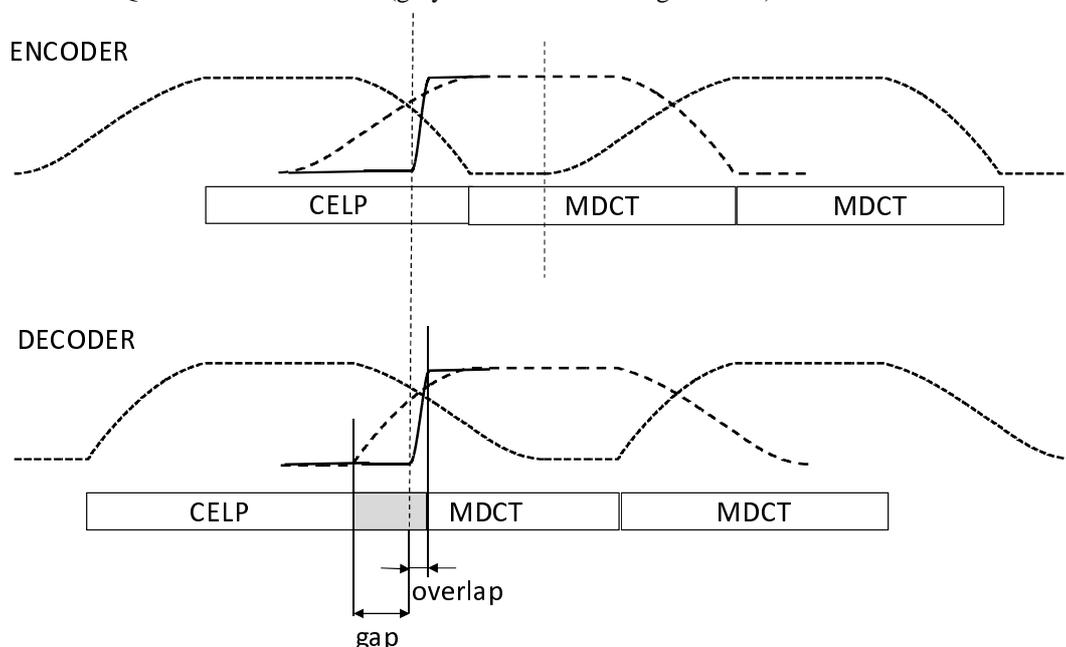
Figure 112b: Modified MDCT synthesis window in the transition frame (CELP to HQ MDCT).

6.3.3.2.1 Constrained CELP decoding and simplified BWE decoding

The bit budget for CELP and BWE in the current (transition) frame is determined depending on the CELP coder used in the previous frame (12.8 kHz or 16 kHz) and decoded audio bandwidth in the current frame, as described in the pseudo-code in clause 5.4.3.2.1. Note that the current frame being a transition frame, one bit is used to indicate the type of CELP coding (12.8 kHz or 16 kHz); this bit is allows decoding the transition frame even when the information about the previous CELP coding type was lost due to frame erasures..

LPC coefficients from the end of the previous frame are reused, constrained CELP decoding only relies on decoding an extra subframe with the same CELP core decoder (12.8 kHz or 16 kHz) as in the previous frame; subframe decoding similar to clause 6.1 is applied (without LSF decoding). The length of the decoded subframe is 5 ms if CELP is at 12.8 kHz and 4 ms if CELP is at 16 kHz. The decoded CELP synthesis is normally delayed by 1.25 ms at the decoder due to the FIR resampling/delay operation. However, to have enough samples to cover the transition region of 6.25 ms, the resampling memory is resampled with 0-delay using the optimized cubic interpolation described in clause 6.3.3.2.1.1. Note that the cubic interpolation requires to have two future samples, which are estimated by simply repeating the last value of the FIR memory.
Hence, if CELP is at 12.8 kHz,, the CELP synthesis (after FIR resampling and 0-delay resampling) is 6.25 ms long; if CELP is at 16 kHz, the subframe is 1 ms shorter and the CELP synthesis is extended by ringing to get a decoded signal of 6.25 ms; the ringing is used only in the overlap region and its influence is perceptually minimal.

When the coded audio bandwidth is higher than the bandwidth of the core CELP coder, BWE decoding is applied. The previous frame is high-pass FIR filtered to obtain the high-band, and the decoded pitch lag and gain are decoded to repeat 6.25 ms of high-band signal which is added to the 6.25 ms of decoded CELP signal.

6.3.3.2.1.1 Optimized cubic interpolation

The missing signal at the output sampling frequency is partly available in the memory buffer at the internal sampling frequency, 12.8 kHz or 16 kHz. By doing low delay resampling like interpolation method of this memory a good estimation of the missing signal can be obtained. Third order cubic interpolation is used here, where cubic curves are used to interpolate the output values within 3 input interval delimited by 4 input samples. Respectively, in each input interval the interpolation can be made by using 3 different cubic curves. To further improve the quality of this estimation the interpolated samples are obtained by computing a weighted mean value of the possible cubic interpolated values computed on the plurality intervals covering the time position of the sample to interpolate.

The length of the resampling buffer (input to cubic interpolation) is 1.25 ms (16 samples at 12.8 kHz sampling rate or 20 samples at 16 kHz sampling rate) plus 2 past samples used as memory for the first cubic interpolations of the first 2 intervals. In cubic interpolation, 4 consecutive input samples determinate a cubic curve, the general equation of this curve is $vc(x) = a_3 x^3 + b_3 x^2 + c_3 x + d_3$. To simplify the computations of the coefficients the temporal index of the 4 consecutive input samples are always considered as $x = -1$, $x = 0$, $x = 1$ and $x = 2$ and so they define 3 intervals, [-1, 0], [0, 1] et [1, 2]. Noting the values of these 4 input samples $v_{in}(-1)$, $v_{in}(0)$, $v_{in}(1)$ and $v_{in}(2)$, the coefficients a_3, b_3, c_3 and d_3 can be computed as:

$$b_3 = \frac{v_{in}(-1) + v_{in}(1)}{2} - v_{in}(0) \tag{1931a}$$

$$a_3 = \frac{v_{in}(-1) + v_{in}(2) - v_{in}(0) - v_{in}(1) - 4b_3}{6} \tag{1931b}$$

$$c_3 = v_{in}(1) - v_{in}(0) - a_3 - b_3 \tag{1931c}$$

$$d_3 = v_{in}(0) \tag{1931d}$$

To get the output resampled signal often the value of the output is needed to be determined between two input samples, in the interval limited by these input samples. As mentioned above, in cubic interpolation one cubic curve covers 3 intervals and respectively each interval can be covered by 3 different cubic curves: by the interval central [0, 1] of the central cubic curve or by the interval [1,2] of the previous cubic curve or the interval [-1, 0] of the next cubic curve. In the following the index $x = 0$ corresponds to the beginning of the input interval where the output interpolated sample is computed. Let's note the coefficients of the cubic curve of which the central interval is used a_n, b_n, c_n, d_n, the

coefficients of the previous cubic curve a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1} and the coefficients of the next cubic curve a_{n+1}, b_{n+1}, c_{n+1}, d_{n+1}. This gives 3 possible values for a given time instant x, $0 \leq x \leq 1$

$$vc_{n-1}(x) = a_{n-1}(x+1)^3 + b_{n-1}(x+1)^2 + c_{n-1}(x+1) + d_{n-1} \qquad (1931e)$$

$$vc_n(x) = a_n x^3 + b_n x^2 + c_n x + d_n \qquad (1931f)$$

$$vc_{n+1}(x) = a_{n+1}(x-1)^3 + b_{n+1}(x-1)^2 + c_{n+1}(x-1) + d_{n+1} \qquad (1931g)$$

The interpolated output value $v_{out}(x)$ for a given instant $0 \leq x \leq 1$ is computed as the weighted mean value of these 3 possible interpolated values:

$$v_{out}(x) = w_{n-1} vc_{n-1}(x) + w_n vc_n(x) + w_{n+1} vc_{n+1}(x) \qquad (1931h)$$

The weights used are same for each interpolated value, $w_{n-1} = w_n = w_{n+1} = 1/3$. To reduce the complexity the values of x/3, x²/3, x³/3, (x-1)/3, (x-1)²/3, (x-1)³/3, (x+1)/3, (x+1)²/3, and (x+1)³/3, are tabulated for all possible values of x needed for the interpolations. So the weighting by 1/3 is integrated in these tables, only the coefficients d_{n-1}, d_n and d_{n+1} are needed with a multiplication by 1/3 when the output value is computed. For example to upsample from 12.8 kHz to 32 kHz the required values of x are 0.2, 0.4, 0.6 and 0.8.

The last 2intervals cannot be covered by 3 cubic curves as future samples are not available to compute all curves. Here simplified interpolation is used. For the last but one input interval the central interval of the last possible cubic curve is used to compute the interpolated signal:

$$v_{out}(x) = a_n x^3 + b_n x^2 + c_n x + d_n \qquad (1931i)$$

and for the last input interval the interval [1,2] of the same last cubic curve is used to compute the interpolated signal

$$v_{out}(x) = a_{n-1}(x+1)^3 + b_{n-1}(x+1)^2 + c_{n-1}(x+1) + d_{n-1} \qquad (1931j)$$

In case of subsampling, the output samples after the last input sample cannot be interpolated, that causes a small delay of up to 3 output samples.

6.3.3.2.2 HQ MDCT decoding with a modified synthesis window

HQ MDCT decoding in the transition frame is identical to clause 6.2.3, except the MDCT synthesis window is modified and the bit budget in the current frame is decreased as described in clause 6.3.3.2.1.

The modified MDCT window is designed to avoid aliasing in the first part of the frame. Its shape also allows cross-fading between the synthesis from constrained CELP and simplified BWE and the synthesis from HQ MDCT.

6.3.3.2.3 Cross-fading

As shown in Figure 112b, the CELP and HQ MDCT decoded signals are overlapping; the length of this overlapping region is 1,825 ms. The HQ MDCT synthesis is already windowed by the modified MDCT window at the decoder. The CELP decoded signal is windowed by the complementary window and added to the HQ MDCT output in the overlap region.

6.3.4 Internal sampling rate switching

When changing the internal sampling rate in CELP or MDCT-based TCX, a number of memory and buffer updates needs to be done. These are described in subsequent sub-clauses.

6.3.4.1 Reset of LPC memory

Same as subclause 5.4.4.1.

6.3.4.2   Conversion of LP filter between 12.8 and 16 kHz internal sampling rates

Same as subclause 5.4.4.2.

6.3.4.3   Extrapolation of LP filter

Same as subclause 5.4.4.3.

6.3.4.4   Update of CELP synthesis memories

Same as subclause 5.4.4.7.

6.3.4.5   Update of CELP decoded past signal

When switching from CELP coding mode to MDCT-base TCX coding mode, the CELP decoded past signal $s_{celp}(n)$ with $n = -N_{celp}, ..., -1$ is needed as described in subclause 6.3.3.1. The past signal is resampled with the method described in subclause 5.4.4.4 in case of internal sampling rate switching before proceeding as described in subclause 6.3.3.1.

6.3.4.6   Post-processing

In case of sampling rate switching, the post-processing module described in clause 6.1.4 has a specific behavior during the transition.

6.3.4.6.1   Adaptive post-filtering

The memories of the adaptive post-filtering are resampled with the linear interpolation described in subclause 5.4.4.4 in case of sampling rate switching and in case post-filtering was applied in the previous frame.

If the post-filtering was not employed in the previous frame, the adaptive post-filter is not employed in the first frame after the transition. In such a case, only the memories of the post-filter are populated and updated for the next frame.

Moreover, in case the adaptive post-filter was activated in the previous frame and is switched off in the current frame, a smoothing mechanism is applied for avoiding any discontinuities. It is achieved by computing the zero impulse response of the post-filter states and adding it to the zero memory response of the current decoded frame. First, the first subframe of the current decoded frame is filtered by the corresponding set of LPC analysis filter $\hat{A}(z)$ and using as memory the previously decoded frame samples before being post-processed. The past non post-processed samples are eventually resampled as described in clause 5.4.4.4 in case of internal sampling rate switching. The residual is re-synthesized with $1/\hat{A}(z)$ using this time as memory the past decoded frame sampled computing after being post-processed. As stated above, the past non post-processed samples are resampled in case of internal sampling rate switching. The rest the current decoded frame is not processed further.

6.3.4.6.2   Bass post filter

At 9.6, 16.4 and 24.4 kbps, the past memory of signal $\hat{s}_f(n)$ defined in clause 6.1.4.2 is reset in case of sampling rate switching. That means that the Bass post-filter has no effect during the transition.

6.3.4.7   CLDFB

In case of internal sampling rate switching, the states of the analysis CLDFB needs to be resampled for both the decoded signal coming from either CELP or MDCT-based TCX decoded module, and also for the error signal $r(n)$ provided by the Bass post-filter as defined in clause 6.1.4.2.

The resampling of the two set of states is performed with the help of the linear interpolation described in subclause 5.4.4.4.

6.3.5 EVS primary modes and AMR-WB IO

6.3.5.1 Switching from primary modes to AMR-WB IO

In addition to processing described Subclause 5.4.5.1, the following is done:

- Reset the unvoiced/audio signal improvement memories

- Reset AMR-WB BWE memories

- bass post-filter is not employed in the first AMR-WB IO frame

- formant post-filter is not employed in the first AMR-WB IO frame

6.3.5.2 Switching from AMR-WB IO mode to primary modes

Same as Subclause 5.4.5.2.

6.3.6 Rate switching

When the bit-rate is changing, the different coding tools are reconfigured at the beginning of the frame. The different bit-rate dependent setups of each tool are described in each corresponding clause. Rate switching doesn't require any specific handling, except in the following scenarios.

6.3.6.1 Rate switching along with internal sampling rate switching

In case the internal sampling rate changes when switching the bit-rate, the processing described in clause 6.3.4 is performed at first.

6.3.6.2 Rate switching along with coding mode switching

In case the internal sampling rate changes when switching the bit-rate, the processing described in clause 6.3.3 is performed. If the internal sampling rate is also changing, the processing of clause 6.3.4 is performed beforehand.

6.3.6.3 Adaptive post-filter reset and smoothing

The adaptive post-filter can be reset and it effect smoothed when it is switched on or off from frame to frame, respectively. The procedure is described in suclause 6.3.4.6.1, where the buffer resampling is performed only in case of internal sampling rate switching.

6.3.7 Bandwidth switching

When rate switching happens and the bandwidth of the output signal is changed from WB to SWB and from SWB to WB, bandwidth switching post-processing is performed in order to improve the perceptual quality for the end users. The smoothing is applied for switching from WB to SWB and the blind bandwidth extension is employed for switching from SWB to WB.

6.3.7.1 Bandwidth switching detector

Firstly, bandwidth switching detector is employed to detect if there is bandwidth switching or not.

Initialize the counter of bandwidth switching from WB to SWB $bws\_cnt1 = 20$, and initialize the counter of bandwidth switching from SWB to WB $bws\_cnt = 40$.

The counter bws_cnt1 is calculated as follows:

1) If the counter $bws\_cnt1 \geq 20$, the counter is reset to 0;

2) else if the output bandwidth of the current frame is SWB, the total bit rate of the current frame is large than 9.6kbps, and the bandwidth of the previous frame is WB, the total bit rate of the previous frame is not large than 9.6kbps, the counter *bws_cnt1* is incremented 1;

3) else if the counter *bws_cnt1* > 0 , the counters *bws_cnt1* and *bws_cnt* will be reset as follows:

$$bws\_cnt1 = \begin{cases} 0, & if\ BW < BW^{[-1]} \\ bws\_cnt1 + 1, & else\ if\ BW = SWB \\ 0, & otherwise \end{cases} \tag{1932}$$

$$bws\_cnt = \begin{cases} 2*(20 - bws\_cnt1) - 1, & if\ BW < BW^{[-1]} \\ 0, & otherwise \end{cases} \tag{1933}$$

The counter *bws_cnt* is calculated as follows:

1) If the counter *bws_cnt* $\geq$ 40 , the counter is reset to 0;

2) else if the conditions $bitrate \leq 9600\ AND\ bitrate^{[-1]} > 9600\ AND\ BW < BW^{[-1]}\ AND\ BW = WB$ are all satisfied, the counter *bws_cnt* is incremented 1

3) else if the counter *bws_cnt* > 0 , the counters *bws_cnt1* and *bws_cnt* will be reset as follows:

$$bws\_cnt = \begin{cases} 0, & if\ BW > BW^{[-1]} \\ bws\_cnt + 1, & else\ if\ BW = WB \\ 0, & otherwise \end{cases} \tag{1934}$$

$$bws\_cnt1 = \begin{cases} (40 - bws\_cnt)/2, & if\ BW > BW^{[-1]} \\ 0, & otherwise \end{cases} \tag{1935}$$

Finally, the counter *bws_cnt* > 0 indicates the switching from super wideband to wideband; and the counter *bws_cnt1* > 0 indicates the switching from wideband to super wideband.

6.3.7.2 Super wideband switching to wideband

TBE mode, multi-mode FD BWE or MDCT based scheme will be employed to generate the SHB signal when switching to wideband.

If the following conditions

$$CT != AC\ AND\ CT != IC\ AND\ core\_brate > 2400\ AND\ core = ACELP\ AND\ Fs \geq 32000$$
$$AND\ BW > NB\ AND\ bws\_cnt > 0\ AND\ !ppp\_mode\_dec \tag{1936}$$

are satisfied, TBE mode is applied to reconstruct the SHB signal.

Otherwise, if the conditions
$core = ACELP\ AND\ Fs \geq 32000\ AND\ BW > NB\ AND\ bws\_cnt > 0\ AND\ !ppp\_mode\_dec$ are satisfied, multi-mode FD BWE algorithm is applied;

If the core is MDCT coding, the MDCT coefficients of the upper band are predicted.

6.3.7.2.1 TBE mode

The following steps are performed when TBE mode is used to generate the SHB signal for wideband output:

1) Estimate the high band LSF, gain shape according to the corresponding parameters of the previous frame or by looking for the pre-determined tables

2) Reconstruct an initial SHB signal according to the TBE algorithm described in subclause 5.2.6.1.

3) Predict a global gain of the initial SHB signal according to the spectral tilt parameter of the current frame and the correlation of the low frequency signal between the current frame and the previous frame

4) Modify the initial SHB signal by the predicted global gain to obtain a final SHB signal

5) Finally, the final SHB signal and low frequency signal are combined to obtain the output signal.

The spectral tilt parameter can be calculated as described by the algorithm described in equations (800) and (801), and the correlation of the low frequency signals of the current frame and the previous frame can be the energy ratio between the current frame and the previous frame..

In detail, the algorithm of predicting the global gain is described as follows:

1) Classify the signal of the current frame to fricative signal $F_{fractive} = 1$ or non- fricative signal $F_{fractive} = 0$

according to the spectral tilt parameter and the correlation of the low frequency signal between the current frame and the previous frame. When the spectral tilt parameter of the current frame is larger than 5 and the FEC class of the low frequency signal is UNVOICED_CLAS, or the spectral tilt parameter is larger than 10. If the signal of the previous frame is non-fricative signal, and the correlation parameter is larger than a threshold, or if the signal of the previous frame is fricative signal and the correlation parameter is less than a threshold, the current frame is classified as fricative signal $F_{fractive} = 1$.

2) For non-fricative signal, the spectral tilt parameter is limited to the range [0.5, 1.0]. For fricative signal, the spectral tilt parameter is limited to not larger than 8. The limited spectral tilt parameter is used as the global gain of the SHB signal.

3) If the energy of the SHB signal is larger than the energy of the signal with the frequency range in [3200, 6400] E_{LH}, the global gain of the SHB signal is calculated as follows:

$$G_{global}^{SHB} = 0.5 * E_{LH} / E'_{SHB}$$ (1937)

where E'_{SHB} is the energy of the initial SHB signal.

If the energy of the SHB signal is less than 0.05 times of the energy of the signal with the frequency range in [3200, 6400] E_{LH}, the global gain of the SHB signal is calculated as follows:

$$G_{global}^{SHB} = 0.25 * E_{LH} / E'_{SHB}$$ (1938)

1) For non-fricative signal, the global gain is multiplied by 8; and for fricative signal, the global gain is multiplied by 2. And then the global gain of the SHB signal G_{global}^{SHB} will be smoothed further as follows:

- If the signal of the current frame and the previous frame are both fricative signal and $G_{global}^{SHB} > R_{ener}$

$$G_{global}^{SHB} = 0.2 * G_{global}^{SHB} + 0.8 * R_{ener}$$ (1939)

- else if the energy ratio of the low frequency signal between the current frame and the previous frame is in the range of [0.5, 2], and the modes of the signal of the current frame and the previous frame are both fricative or are both non-fricative

$$G_{global}^{SHB} = 0.5 * G_{global}^{SHB} + 0.5 * R_{ener}$$ (1940)

- Otherwise

$$G_{global}^{SHB} = \begin{cases} (1 - 0.1 \cdot G_{global}^{SHB}) \cdot G_{global}^{SHB} + 0.1 \cdot G_{global}^{SHB} \cdot R_{ener}, & if \; F_{fractive} = 0 \; AND \; F_{fractive}^{[-1]} = 1 \\ 0.5 \cdot G_{global}^{SHB} + 0.5 \cdot \min(G_{global}^{SHB}, R_{ener}), & otherwise \end{cases} \tag{1941}$$

where R_{ener} is the energy ratio between the final SHB signal of the previous frame and the initial SHB signal of the current frame.

1) Then , fade out the global gain of the SHB signal frame by frame as follows:

$$G_{global}^{SHB} = G_{global}^{SHB} * (40 - bws\_cnt) / 40 \tag{1942}$$

6.3.7.2.2 Multi-mode FD BWE mode

Predict the SHB signal of the current frame, and weight the SHB signal of the current frame and the previous frame to obtain the final SHB signal of the current frame. Then the SHB signal and the low frequency signal are combined to obtain the output signal.

The SHB signal of the current frame is generated as follows:

1) Predict the fine structure of the SHB signal of the current frame as described in subclause 6.1.5.2.1.5.

2) Predict the envelope of the SHB signal of the current frame, and weight the envelopes of the current frame and the previous frame to obtain the final envelope of the current frame.

3) Reconstruct the SHB signal of the current frame by the predicted excitation and the weighted envelope.

In detail, the algorithm of predicting the envelope is described as follows:

1) The 224 MDCT coefficients in the 800-6400 Hz frequency range $X_{M\_C}(k), k = 0,...,223$ are split into 7 sharpness bands (32 coefficients per band). Calculate the peak of the magnitudes and the average magnitude in each sharpness band.

2) The low frequency signal is classified into NORMAL or HARMONIC according to the ratios of the peak of the magnitudes and the average magnitude.

3) Predict the initial envelope of the SHB signal according to the average magnitudes, the maximum value of the average magnitudes, the minimum value of the average magnitudes and the tilt parameter of the low frequency signal.

4) An energy ratio r_{bws} between the same parts of the low frequency signal of the current frame and the previous frame is introduced to control the weighted value of the predicted envelope. This ratio r_{bws} reflects the correlation of the current frame and the previous frame.

5) Factor w_{bws} is the weighting factor for the spectral envelope of the current frame, and $(1 - w_{bws})$ is the one for the previous frame. w_{bws} is initialized to 0.1. Note that w_{bws} is reset to 0.1 if bws_cnt equals to zero.

6) If $0.5 < r_{bws} < 2.0$, the predicted spectral envelope is weighted according to the energy ratio w_{bws}

$$\hat{f}_{env}(j) = \begin{cases} w_{bws} \cdot \hat{f}'_{env}(j) + (1 - w_{bws}) \cdot \hat{f}_{env}^{[-1]}(j), & if \; w_{bws} < 0.9 \\ 0.9 \cdot \hat{f}'_{env}(j) + 0.1 \cdot \hat{f}_{env}^{[-1]}(j), & otherwise \end{cases} \tag{1943}$$

and $\hat{f}_{env}(j) = 0.1 \cdot \hat{f}'_{env}(j) + 0.9 \cdot \hat{f}_{env}^{[-1]}(j)$, when $M_{extl} != M_{last\_extl} \; AND \; \hat{f}'_{env}(j) > 2.0 \cdot \hat{f}_{env}^{[-1]}(j) \; AND \; w_{bws} < 0.9$

where $\hat{f}_{env}^{[-1]}(j)$ is the envelope of the previous frame. Factor w_{bws} is incremented by 0.05 and saved for next frame.

Otherwise, the predicted envelope is weighted as follows:

$$\hat{f}_{env}(j) = 0.9 \cdot \hat{f}'_{env}(j) + 0.1 \cdot \hat{f}_{env}^{[-1]}(j) \quad j = 0, \cdots, 13 \tag{1944}$$

Then, fade out the predicted envelope of the SHB signal frame by frame as follows:

$$\hat{f}_{env}(j) = \hat{f}_{env}(j) * (40 - bws\_cnt) / 40 \tag{1945}$$

6.3.7.2.3 MDCT core

For low bit rate core: predict the envelopes of the upper band from the 2 decoded highest frequency envelopes and the average envelope of all the decoded envelopes, and reconstruct he normalized coefficients by random noise. And then the MDCT coefficients of the upper band are reconstructed by the envelopes and the normalized coefficients.

For high bit rate core: for transient mode, predict the envelopes by the 20 decoded highest frequency coefficients; and for non-transient mode, predict the envelopes by the 2 decoded highest frequency envelopes. The normalized coefficients are predicted by random noise or by weighting the random noise and the decoded low frequency coefficients. And then the MDCT coefficients of the upper band are reconstructed by the envelopes and the normalized coefficients.

6.3.7.3 Wideband switching to super wideband

In the case of switching from wideband to super wideband for multi-mode FD BWE or the MDCT core the coefficients in the frequency domain are faded using the factor $bws\_cnt1/20$. When operating with TBE, the global gain in the temporal domain is faded using the same factor $bws\_cnt1/20$.

6.4 De-emphasis

The reverse of the filtering process described in subclause 5.1.4 is performed at the output of the decoder.

$$H_{de-emph}(z) = \frac{1}{1 - \beta_{pre-emph}z^{-1}} \tag{1946}$$

where $\beta_{pre-emph}$ is the de-emphasis factor which is set according to the sample rate and core type as described in subclause 5.1.4.

6.5 Resampling to the output sampling frequency

The CELP and MDCT decoder output signals are sample rate converted, depending on the selected output sample rate, i.e. 8 kHz, 16 kHz, 32 kHz or 48 kHz. The CELP signal is resampled by the CLDFB, while the MDCT part is resampled by its frequency to time transformation.

6.6 Decoding of frame erasure concealment side information

The parameters described in subclause 5.5 are decoded to aid in the error concealment procedure.

These parameters are;

- Signal classification parameter

- Energy information

- Phase control information

- Pich lag information

- Spectral envelope diffuser

For more information how these parameters are employed, refer to subclause 5.3.3 of [6].

6.7 Decoding in DTX/CNG operation

6.7.1 Overview

When the decoder is in the CNG operation, the first bit in the SID frame is decoded to point to the CNG mode in which the decoder will be operating. In the LP-CNG mode, excitation(s) and synthesis filter(s) are calculated from the decoded CN parameters, and the comfort noise is synthesized by a LP synthesis approach.

In the FD-CNG mode, a spectral envelope is generated with the help of the energy level information decoded from the SID data. The spectral envelope is refined by a noise estimator running during active frames. The resulting envelope determines the actual comfort noise which is rendered inside different frequency domains, such as MDCT, FFT or CLDFB and finally transformed to time-domain.

Subsequent subclauses describe the respect LP-CNG decoding and FD-CNG decoding in details.

6.7.2 Decoding for LP-CNG

6.7.2.1 LP-CNG decoding Overview

When the decoder is in the LP-CNG operation, a procedure to synthesize a comfort noise signal is applied.

For each received SID frame, the one bit indicating the bandwidth type of the SID frame is first decoded. WB SID frame is received if the bandwidth bit equals "0", otherwise the SWB SID frame is received. The LP-CNG decoder only operates in WB mode if no SWB SID frame has been received, in which case the comfort noise is only generated for low-band. Otherwise, the LP-CNG decoder will switch to SWB mode upon the receiving of the SWB SID frame. Since the transmission of high-band CN parameter is not synchronized with the transmission of the low-band CN parameters, WB SID frames can be received even the LP-CNG decoder operates in SWB mode. In which case, the parameter for high-band CN synthesis is extrapolated from the low-band CN synthesis. The low-band excitation energy is decoded from each LP-CNG SID frame and a smoothed quantized excitation energy used for CNG synthesis is computed, as described in subclause 6.7.2.1.3. The low-band LSF vector is decoded from each LP-CNG SID frame then converted to LSP vector and a smoothed LSP vector is computed then converted to LP coefficients to obtain the CNG synthesis filter, as described in subclause 6.7.2.1.4. If WB LP-CNG SID frame is received, from which the quantized residual spectral envelope is decoded and a smoothed residual spectral envelope is computed, as described in subclause 6.7.2.1.5. A random excitation signal is generated from the smoothed quantized excitation energy which is combined with a second excitation signal generated from the smoothed residual spectral envelope to form the final excitation signal for the low-band CNG synthesis, as described in subclause 6.7.2.1.5. Low-band comfort noise is synthesized by filtering the low-band final excitation through the low-band synthesis filter, as described in subclause 6.7.2.1.6.

In subclause 6.7.2.1.7, high-band decoding and synthesis is described if the decoder is operating in SWB mode. When SWB LP-CNG SID frame is received, the high-band energy is decoded from the SID frame, The low-band energy is calculated from the low-band CNG synthesis in each CN frame and smoothed. A smoothed high-band energy is extrapolated from the smoothed low-band energy together with the decoded high-band energy and the smoothed low-band energy at the latest received SWB LP-CNG SID frame. The high-band LSF spectrum used to obtain the high-band CNG synthesis filter is interpolated from the LSF spectrum of the hangover frames. The high-band comfort noise is synthesized by filtering a random excitation through the high-band synthesis filter, then scaled to the level corresponding to the smoothed high-band energy. The scaled high-band synthesis is finally spectral flipped to the bandwidth from 12.8 kHz to 14.4kHz, as described in subclause 6.1.5.1.12. The resulting high-band synthesis signal is added to the low-band synthesis signal so to form the final SWB comfort noise synthesis signal.

6.7.2.1.1 CNG parameter updates in active periods

During actively encoded periods without comfort noise parameters, four buffers of the fixed predetermined size *HO_HIST_SIZE* are kept updated with the current actively encoded frame's LSPs, an LSP domain flag memory, the frame's excitation energy(in the LP-residual domain) and the current low frequency spectral envelope of the excitation as:

$$ho_{lsp}(frame,i) = lsp_{new,dec}(i), \quad for \quad i = 0...M-1 \tag{1947}$$

$$ho_{16kLSP}(frame) = FALSE, \quad if \quad L_{frame} = 256$$
$$ho_{16kLSP}(frame) = TRUE, \quad if \quad L_{frame} \neq 256 \tag{1948}$$

$$enr_{dec}(frame) = \frac{1}{L_{frame}} \sum_{i=0}^{L_{frame}-1} exc(i)^2 \tag{1949}$$

$$ho_{env}(frame, i) = att \frac{2}{L_{FFT}} \left(X_R^2(i) + X_I^2(i) \right), \quad for \quad i = 0,...,19 \tag{1950}$$

where $X_R(i)$ and $X_I(i)$ are, respectively, the real and the imaginary parts of the i-th frequency bin as outputted by the FFT of the LP excitation signal, $L_{FFT} = 256$ is the size of FFT analysis. The attenuation factor att is given by

$$att = \frac{1}{2^{Tab_{enr\_att}(R_{latest\_active})}} \tag{1950a}$$

where Tab_{enr_att} is determined by the latest bitrate used for actively encoded frames R_{latest_active}, not including the current frame, according to Table 172a.

Table 172a: Attenuation factor selection

Latest active bitrate [kbps]	Tab_{enr_att}
$R_{latest\_active} \leq 7.2$	1.7938412
$7.2 < R_{latest\_active} \leq 8.0$	1.3952098
$8.0 < R_{latest\_active} \leq 9.6$	1.0962363
$9.6 < R_{latest\_active} \leq 13.2$	0.9965784
$R_{latest\_active} > 13.2$	0.9965784

These buffers are implemented as circular FIFO (First in First Out) buffers of size *HO_HIST_SIZE* to save complexity.

6.7.2.1.2 DTX-hangover based parameter analysis in LP-CNG mode

To provide smoother sounding comfort noise synthesis in transitions from active to inactive (CNG) coding, the 3 bit parameter $burstho_{cnt,rec}$ is decoded from the bit stream and used as the indicator for determining the initial subset of the *HO_HIST_SIZE* sized buffers with ($ho_{lsp}, ho_{16kLSP}, enr_{dec}, ho_{env}$) parameters from the last active frames. The most recent $burstho_{cnt,rec}$ number of frames of the stored parameters ($ho_{lsp}, ho_{16kLSP}, enr_{dec}, ho_{env}$), are used for an additional comfort noise parameter analysis in the very first SID frame after an active speech segment.

Before copying the $burstho_{cnt,rec}$ most recent ho_{lsp} vectors to the CNG-analysis buffer $ho_{lsp-hist}$, the past LSP's which were analysed with a different sampling frequency than the current SID frame's sampling frequency is converted to the current frames sampling frequency according to the information available in the flag vector ho_{16kLSP}. The $burstho_{cnt,rec}$ most recent enr_{dec} values are copied into the analysis vector enr_{hist} and the $burstho_{cnt,rec}$ most recent ho_{env} values are copied into the analysis vector $ho_{env-hist}$.

An age weighted average energy of the enr_{hist} entries which are less than 103% of the most recent energy value and greater than 70 % of the most recent energy value, is computed as $enr_{hist-ave-weighted}$, further the number of entries in enr_{hist} used for this average calculation is stored as m. The age weights for the $enr_{hist-ave-weighted}$ computation are:

$$w_{ho-enr} = \left[0.2, 0.16, 0.128, 0.1024, 0.08192, 0.065536, 0.0524288, 0.01048576 \right] \tag{1951}$$

Further the m $ho_{lsp-hist}$ vectors corresponding in time to the past residual energies in enr_{hist} used for $enr_{hist-ave-weighted}$ are saved in a buffer $ho_{lsp-hist-sel}$. The buffer $ho_{lsp-hist-sel}$ is converted into the LSF domain in buffer $ho_{lsf-hist-sel}$.

Two outlier vector indices [max_{idx0}, max_{idx1}] in the $ho_{lsf-hist-sel}$ buffer among the m vector entries are found, by analysing the maximum average LSF deviation to a uniform LSF spectrum. An average LSP-vector $ho_{lsp-ave-weighted}$ is calculated, by computing the LSP-average with exclusion of zero, one or two of the found outlier vectors, depending on the value of m.

The sum of the LSP-deviations with respect to the received SID-frame's quantized LSPs $lsp_{new,dec}$, is computed as:

$$lsp_{dist} = \sum_{i=0}^{M-1} \left| ho_{lsp-ave-weighted}(i) - lsp_{new,dec}(i) \right| \qquad (1952)$$

Further the maximum individual distortion contribution in the summation above is saved as lsp_{maxdev}.

If there were no past CN-parameters to analyse or an residual energy step was detected, the received $lsp_{new,dec}$ vector is used as the final lsp_{CNG} vector right away, on the other hand if there were some past active SAD hangover frames to analyse and there was no energy step detected , the lsp_{dist} and lsp_{maxdev} are now used to control the vector update over $i = 0...M-1$, of the final CNG LSPs lsp_{CNG} using the average LSP-vector $ho_{lsp-ave-weighted}$ as follows:

$$
\begin{aligned}
lsp_{CNG}(i) &= ho_{lsp-ave-weighted}(i), & &if \quad (lsp_{dist} > 0.4) \vee (lsp_{maxdev} > 0.1) \\
lsp_{CNG}(i) &= 0.8 \cdot ho_{lsp-ave-weighted}(i) + 0.2 \cdot lsp_{new,dec}(i), & &otherwise
\end{aligned}
\qquad (1953)
$$

The energy step is detected if it is the first CN frame after an active frame and the energy quantization index I_q decoded from the current SID frame is greater than the previous energy quantization index $I_q^{[-1]}$ by more than 1, where $I_q^{[-1]} \geq 0$. Additionally, if there were past CN-parameters, an energy step is detected if the most recent energy value in enr_{hist} is more than four times larger than the smoothed quantized excitation energy E_{CN}. Further the m_1 $ho_{env-hist}$ vectors that originate from active or WB SID frames among the m $ho_{env-hist}$ vectors corresponding in time to the past residual energies in enr_{hist} used for $enr_{hist-ave-weighted}$ are saved in a buffer $ho_{env-hist-sel}$. The average envelope of $ho_{env-hist-sel}$ is computed and from which two times of the smoothed residual spectral envelope of the previous frame, $Env_{CN}^{[-1]}(k)$, calculated in equation (1958) is subtracted. The resulting average envelope is used to initialize the smoothed residual spectral envelope if there is no energy step detected.

When a SID frame is received and there was no energy step detected, first the received and decoded LSP vector $lsp_{new,dec}$ is added to the CNG-analysis buffer $ho_{lsp-hist}$ in a FIFO manner for a buffer size of up to *HO_HIST_SI ZE* , and secondly the decoded residual energy value in the SID frame E_{new} is added to the CNG analysis buffer enr_{hist} in a FIFO manner for a buffer size of up to *HO_HIST_SI ZE* , then thirdly, if applicable (depending on if the SID frame is of WB type or not), the decoded low frequency envelope of the excitation from the SID frame is added to the CNG analysis buffer $ho_{env-hist}$ for a buffer size of up to *HO_HIST_SI ZE* .

During actively encoded periods, i.e. not including SID frames, the currently least recent buffer element (firstly added) in the buffer $ho_{lsp-hist}$ and the corresponding element in the buffer enr_{hist} are excluded from the buffers with a period of number of consecutive actively encoded frames given by the decrement factor *BUF _ DEC _ RATE* . As circular FIFO buffers are implemented the elements do not actually have to be deleted but the variable $ho_{hist-size}$ representing the number of valid buffer elements, i.e. elements used for determination of $ho_{lsp-hist-sel}$ and $enr_{hist-ave-weighted}$, is given by:

$$ho_{hist-size} = ho_{hist-size}^{[0]} - \eta \quad for \quad \eta \cdot BUF \_ DEC \_ RATE \leq act_{cnt} < (\eta+1) \cdot BUF \_ DEC \_ RATE \quad (1953a)$$

where $ho_{hist-size}^{[0]}$ is the number of valid buffer elements in the very beginning of the actively encoded period, η is a non-negative integer and act_{cnt} is a counter of consecutive actively encoded frames. The variable $ho_{hist-size}$ does together with a pointer to the most recently added buffer element determine the valid buffer elements.

6.7.2.1.3 LP-CNG low-band energy decoding

The quantized logarithmic excitation energy is found using the procedure in subclause 5.6.2.1.5 and converted to linear domain. The quantized energy $\hat{\tilde{E}}$ is used to obtain the smoothed quantized excitation energy E_{CN} used for CNG synthesis in the same way as described in subclause 5.6.2.1.6.

6.7.2.1.4 LP-CNG low-band filter parameters decoding

The quantized LSF vector is found in the same way as described in subclause 6.1.1.1.1. For the two stage quantizer there are two indexes that define the LSF vector. The index of the first stage codevector is retrieved and the codevector components are obtained from the 16-dimensional codebook of 16 codevectors. The second stage index is interpreted like in subclause 6.1.1.1.1. and the corresponding multiple scale lattice codevector is obtained. If the codevector index from the first stage has one of the values 0, 1, 2, 3, 7, 9, 12, 13, 14, 15 the permutations specified in subclause 5.6.2.1.3 are applied to the decoded codevector. The resulting codevector is added to the codevector obtained in the first stage and the result corresponds to the decoded LSF vector. The sampling frequency of the LP-CNG frame can be determined by checking the value of the highest order LSF coefficient (last coefficient). If the last decoded LSF coefficient is larger than 6350 the decoded frame has sampling rate of 16 kHz, otherwise it is sampled at 12.8kHz and contains either NB or WB LSF data. The smoothed LP synthesis filter, $\hat{A}(Z)$, is then obtained in the same way as described in subclause 5.6.2.1.4.

6.7.2.1.5 LP-CNG low-band excitation generation

The low-band excitation signal used for CNG synthesis is generated by combining a random excitation and an excitation representing the low frequency spectral details of the excitation signal.

The random excitation is generated for each subframe using a random integer generator, the seed of which is updated by

$$s_r = \text{short}[s_r \times 31821 + 13849] \tag{1954}$$

where s_r is the seed value, initially set to 21845, and short[.] limits the value to the interval [–32767; 32768]. The generated random sequence, denoted as $r(n), n = 0,...,L_{sf} - 1$, is scaled for each subframe by

$$e_r(n) = r(n) \sqrt{\frac{\tilde{E}_{CN} \cdot L_{sf}}{\sum_{i=0}^{L_{sf}-1} (r(i))^2}} \quad n = 0,\ldots,L_{sf} - 1 \tag{1955}$$

where L_{sf} denotes the length of subframe, $\tilde{E}_{CN}$ is the smoothed quantized excitation energy, as described in subclause 6.7.2.1.3, with some random variation between subframes added. The random variation is added to the smoothed quantized excitation energy by

$$\tilde{E}_{CN} = E_{CN} (1 + 0.000011 \cdot s_E) \tag{1956}$$

where s_E is a random integer number generated for each subframe using the same equation (1954) with the initial value of 21845. The purpose of which is to better model the variance of background noise during inactive signal periods. The scaled random sequence in each subframe, $e_r(n), n = 0,...,L_{sf}$, is concatenated to form the random excitation signal for the whole frame, $e_r(n), n = 0,...,L$, where L is the frame length.

The excitation representing the low frequency spectral details of the excitation signal is generated from the quantized residual spectral envelope. The quantized residual spectral envelope is recovered from each WB SID frame in the inverse way as described in subclause 5.6.2.1.6 that

$$E\hat{n}v(k) = 2^{\hat{E}_{exc}-D(k)}, \qquad k = 0,...,19 \tag{1957}$$

where $E\hat{n}v(k)$ is the quantized residual spectral envelope, $D(k)$ is the entry of the residual spectral envelope codebook found with the index decoded from the SID frame, $\hat{E}_{exc}$ is the quantized total excitation energy calculated using the similar equation in subclause 5.6.2.1.6,. A smoothed residual spectral envelope is updated at each CN frame by $E\hat{n}v(k)$ through an AR filtering

$$Env_{CN}(k) = 0.9Env_{CN}^{[-1]}(k) + 0.1E\hat{n}v(k), \qquad k = 0,...,19 \tag{1958}$$

where $Env_{CN}^{[-1]}(k)$ denotes the smoothed residual spectral envelope from the previous frame. If the current frame is of NO_DATA type, the $E\hat{n}v(k)$ from the last received SID frame is used. The FFT spectrum of the random excitation, $e_r(n), n = 0,..., L$, generated in equation (1955) is computed and based on this the low frequency spectral envelope, $Env_{Rd}(k), k = 0,...,19$, corresponding to the one transmitted in the SID frame is calculated in a similar manner. The difference envelope between the smoothed spectral envelope $Env_{CN}(k)$ and the random excitation envelope $Env_{Rd}(k)$ is calculated

$$D_{Env}(k) = MAX\left[Env_{CN}(k) + 2E_{CN} - Env_{Rd}(k), 0\right], \qquad k = 0,...,19 \tag{1959}$$

where E_{CN} is the smoothed quantized excitation energy obtained in subclause 6.7.2.1.3, 2 times of E_{CN} is compensated to the $Env_{CN}(k)$ before the difference envelope is calculated. Slight random variation is added to the difference envelope.

$$\tilde{D}_{Env}(k) = D_{Env}(k) + 0.000011 \cdot s_{env}(k) \cdot D_{Env}(k), \qquad k = 0,...,19 \tag{1960}$$

where $s_{env}(k), k = 0,...,19$ is a series of random integer numbers generated for each envelope band using the same equation (1954) with the initial value of 21845. A series of 256-point random-phase FFT coefficients are generated where its low frequency spectral envelope is made equal to the difference envelope $\tilde{D}_{Env}(k)$ and the coefficients corresponding to other frequencies are set to 0. An IFFT is performed to the above FFT coefficients and a 256-point time domain sequence $d(n)$ is outputted. $d(n)$ is re-sampled to 320 points if operating in 16kHz core. $d(n)$ is scaled for each subframe in a similar way to equation (1955) that

$$e_d(n) = d(n)\sqrt{\frac{E_d(1+0.000011s_{env}) \cdot L_{sf}}{\sum_{i=0}^{L_{sf}-1}(d(i))^2}} \qquad n = 0,\ldots,L_{sf}-1 \tag{1961}$$

where $e_d(n)$ is the scaled $d(n)$ with random variation added, L_{sf} denotes the length of subframe, s_{env} is a random integer number generated for each subframe using the same random generator as used in equation (1960), E_d is the average energy of $d(n)$ calculated as

$$E_d = \frac{1}{L}\sum_{n=0}^{L-1}(d(n))^2 \tag{1962}$$

where L is the frame length. Energy increasing is not allowed if the current frame is the first SID frame after an active burst in which case, for subframe with $e_d(n) > d(n)$, $e_d(n)$ is limited to $d(n)$. The $e_d(n)$ is the excitation representing the low frequency spectral details of the excitation signal whichis attenuated and combined with the earlier calculated random excitation $e_r(n)$

$$e(n) = 0.75e_d(n) + e_r(n), \qquad n = 0,1,...,L-1 \tag{1963}$$

where L is the frame length, $e(n)$ is the combined excitation signal. The combined excitation $e(n)$ is scaled if its average energy is higher than the smoothed quantized excitation energy E_{CN} obtained in subclause 6.7.2.1.3.

$$e'(n) = e(n) \sqrt{\min\left(\frac{E_{CN}}{\frac{1}{L}\sum_{i=0}^{L-1}(e(i))^2}, 1\right)} \qquad n = 0, \ldots, L-1 \qquad (1964)$$

where L is the frame length, $e'(n)$ is the scaled combined excitation which is the final excitation signal for low-band CNG synthesis.

6.7.2.1.6 LP-CNG low-band synthesis

The low-band comfort noise is synthesized by filtering the scaled combined excitation signal, $e'(n)$, obtained in previous subclause 6.7.2.1.5 through the smoothed LP synthesis filter, $\hat{A}(Z)$, obtained in subclause 6.7.2.1.4.

6.7.2.1.7 LP-CNG high-band decoding and synthesis

To enable high perceptual quality in the inactive portions of speech on the decoder side, during SWB mode operation of the codec, a high band comfort noise synthesis (SHB-CNG) (12.8 - 14.4 kHz) is added to the low bandwidth (0-12.8 kHz) LP-CNG synthesis output. This also helps to ensure smooth transitions between active and inactive speech.

However, this is being done without transmitting any extra parameters from the encoder to decoder to model the high-band spectral characteristics of the inactive frames. Instead, to model the high band spectrum (12.8 - 14.4 kHz) of the comfort noise, the high band LSF parameters of the active speech frames preceding the current inactive frames are used after interpolation as described below. The hangover setting in the SAD algorithm ensures the active speech segments used for the spectral characteristics estimation of the inactive frames, sufficiently capture the background noise characteristics without significant impact from the talk spurt.

The quantized LSF vectors of order 10 corresponding to active speech high band (subclauses 5.2.4.1.3.1 and 6.1.5.1.3.1) received at the decoder are buffered up to two past active frames (N-1) and (N), denoted by $\rho_{N-1,k}^{SHB}$ $k = 1, \ldots, 10$ and $\rho_{N,k}^{SHB}$ $k = 1, \ldots, 10$ where N+M is the current inactive frame. Using these, the LSF vector corresponding to SHB-CNG of (N+M) $^{\text{th}}$ inactive frame $\rho_{N+M,k}^{SHB-CNG}$ $k = 1, \ldots, 10$ is interpolated as

$$\rho_{N+M,k}^{SHB-CNG} = T * \rho_{N,k}^{SHB} + (1-T) * \rho_{N-1,k}^{SHB} \qquad k = 1, \ldots, 10 \qquad (1965)$$

where interpolation factor T is computed as

$$T = \min\left(\frac{M}{32}, 1\right) \qquad (1966)$$

using the number of inactive frames M leading up to the current inactive frame (N+M) since the last active frame N.

This interpolated LSF vector $\rho_{N+M}^{SHB-CNG}$ is then converted to LPC coefficients and used as the coefficients of LP synthesis filter to generate a synthesized signal. The quantized high-band log average energy, $\hat{\bar{E}}_h'$, is recovered from each received SWB SID frame by

$$\hat{\bar{E}}_h' = 10\log_{10} 2 \cdot \left(\frac{I}{0.9} - 6\right) \qquad (1967)$$

where I is the high-band energy index decoded from SID frame. If I is 0, I is set to -15 for a lower noise floor. The log average energy of the low-band signal is calculated from the low-band synthesis signal

$$\bar{E}_l' = 10\log_{10}\left(\frac{1}{L} \cdot \sum_{i=0}^{L-1} \hat{s}_l(i)^2\right) \qquad (1968)$$

where $\hat{s}_l(i)$ is the low-band synthesis signal as obtained in subclause 6.7.2.1.6, $L = 640$ is the length of the low-band synthesis signal. If the low-band log average energy $\overline{E}_l'$ of the current frame is deviating from the smoothed low-band log average energy of the previous frame $\overline{\overline{E}}_l^{[-1]}$ by more than 12 dB, a step update flag f_{step} is set to 1 indicating the permission of step update, otherwise is set to 0. If the flag f_{step} is set to 1, the smoothed low-band log average energy at the current frame, $\overline{\overline{E}}_l$, is set to the current frame's low-band log average energy $\overline{E}_l'$. Otherwise, if the flag f_{step} is set to 0, the smoothed low-band log average energy is updated at the current frame as

$$\overline{\overline{E}}_l = 0.1\overline{\overline{E}}_l^{[-1]} + 0.9\overline{E}_l' \tag{1969}$$

where $\overline{\overline{E}}_l^{[-1]}$ denotes the smoothed low-band log average energy of the previous frame. The smoothed high-band log average energy is extrapolated from the smoothed low-band log average energy, that is

$$\overline{\overline{E}}_h = \alpha\overline{\overline{E}}_h^{[-1]} + (1-\alpha)(\overline{\overline{E}}_h^{[-i]} + \Delta_l) \tag{1970}$$

where $\overline{\overline{E}}_h^{[-1]}$ denotes the smoothed high-band log average energy of the previous frame, α is the forgetting factor which is set to 0 if f_{step} is set to 1 or the current frame is the first frame after an active burst, otherwise α is set to 0.75, $\overline{\overline{E}}_h^{[-i]}$ denotes the smoothed high-band log average energy at the last received SWB SID frame i frames ago, Δ_l is the energy deviation of the current frame's smoothed low-band log average energy from its value at the last received SWB SID frame. $\Delta_l = \overline{\overline{E}}_l - \overline{\overline{E}}_l^{[-i]}$, where $\overline{\overline{E}}_l^{[-i]}$ denotes the smoothed low-band log average energy at the last received SWB SID frame i frames ago. Δ_l is limited to not greater than 15 dB. $\overline{\overline{E}}_h^{[-i]} + \Delta_l$ is equivalent to extrapolating a non-smoothed high-band energy at the current frame from the low-band energy at the current frame and high-band to low-band energy ratio at the last received SWB SID frame. The high-band comfort noise is synthesized by filtering a 320-point white noise excitation signal through the LP synthesis filter derived earlier in this subclause. The synthesized comfort noise signal is then level adjusted to match the calculated smoothed high-band log average energy $\overline{\overline{E}}_h$. A smoothing period is setup for the first 5 frames after an active burst of more than 3 frames and if the core technology used in the last active frame is not HQ-core. Within the smoothing period, the synthesized comfort noise is not level adjusted to the calculated smoothed high-band log average energy $\overline{\overline{E}}_h$, but to an interpolated energy between $\overline{\overline{E}}_h$ and the high-band log average energy calculated at the last active frame. The interpolated energy is calculated as

$$\overline{\overline{E}}_{hs,i} = \overline{\overline{E}}_h + SIN\left[\frac{\pi}{2} \cdot \frac{6-i}{15}\right](E_{h,act} - \overline{\overline{E}}_h), \qquad i = 1,2,3,4,5 \tag{1971}$$

where $\overline{\overline{E}}_{hs,i}$ denotes the interpolated high-band log average energy of the i-th CN frame in the smoothing period, $E_{h,act}$ denotes the high-band log average energy of the last active frame, $SIN[\cdot]$ denotes the sine function. Finally, the level adjusted synthesized high-band comfort noise is spectral flipped to the bandwidth from 12.8kHz to 14.4kHz as described in subclause 6.1.5.1.12. The resulting high-band synthesis signal is later added to the low-band synthesis signal to form the final SWB comfort noise synthesis signal.

6.7.2.2   Memory update

When an inactive signal frame is processed, the following updates are performed:

– MA memory of the ISF quantizer is set to zero;

– AR memory of the ISF quantizer is set to mean values (UC mode, WB case);

– phase dispersion memory is set to zero;

– synthesis excitation spectrum tilt is set to zero;

– noise enhancer memory is set to zero;

– class of last received good frame for FEC is set to UNVOICED_CLAS;

– floating point pitch for each subframe is set to the subframe length;

– the low-pass filtered pitch gain for FEC is set to zero;

– the filtered algebraic codebook gain for FEC is set to the square root of the smoothed quantized CNG excitation energy, E_{CN}, from subclause 6.7.2.1.3;

– the excitation buffer memory is updated;

– previous pitch gains are all set to zero;

– previous codebook gain is set to zero;

– active frame counter is set to zero;

– voicing factors used by the bandwidth extension are all set to 1;

– bass post-filter is tuned off;

– synthesis filter memories are updated.

6.7.3 Decoding for FD-CNG

In FD-CNG, the comfort noise is generated in the frequency domain. Based on the information provided by the SID frames, the amplitude of the random sequences can be individually computed in each band such that the spectrum of the generated comfort noise resembles the spectrum of the actual background noise in the input signal.

Unfortunately, the limited number of parameters transmitted in the SID frames allows only to reproduce the smooth spectral envelop of the background noise. Hence, it cannot capture the fine spectral structure of the noise. At the output of a DTX system, the discrepancy between the smooth spectrum of the reconstructed comfort noise and the spectrum of the actual background noise can become very audible at the transitions between active frames (involving regular coding and decoding of a noisy speech portion of the signal) and the comfort noise frames. Since the fine spectral structure of the background noise cannot be transmitted efficiently from the encoder to the decoder, it is highly desirable to recover this information directly at the decoder side. This can be carried out using a noise estimator.

Note that noise-only frames are considered as inactive frames in a DTX system. Therefore, the noise estimation in the decoder must operate during active phases only, i.e., on noisy speech contents. In FD-CNG, the decoder uses in fact the same noise estimation algorithm as in the encoder, but applying a significantly higher spectral resolution at the decoder than at the encoder.

6.7.3.1 Decoding SID frames in FD-CNG

The decoded SID parameters $N_{\text{FD-CNG}}^{[\text{SID}]}(i), i = 0,...,L_{\text{SID}}-1$ describe the energy of the background noise in the spectral partitions defined in subclause 5.6.3.6. The first $L_{\text{SID}}^{[\text{FFT}]}$ parameters capture the spectral energy of the noise in FFT bins covering the core bandwidth. The remaining $L_{\text{SID}}^{[\text{CLDFB}]}$ parameters capture the spectral energy of the noise in CLDFB bands above the core bandwidth.

6.7.3.1.1 SID parameters decoding

The SID parameters $N_{\text{FD-CNG}}^{[\text{SID}]}(i), i = 0,...,L_{\text{SID}}-1$ are decoded using MSVQ decoding and global gain adjustment.

Seven indices are decoded from the bitstream. The first six indices $I_{\text{MSVQ,FD-CNG}}(k), k = 0,...,5$ are used for MSVQ decoding. They correspond to the six stages of the MSVQ. The first index is encoded on 7 bits and the five next indices are encoded on 6 bits. The last index $I_{\text{g,FD-CNG}}$, encoded on 7 bits, is used for decoding the global gain.

The MSVQ decoder output is given by

$$\overline{N}_{\text{FD-CNG}}^{[\text{SID}]}(i) = \sum_{k=0}^{5} V_k\big(I_{\text{MSVQ,FD-CNG}}(k), i\big),$$ (1972)

where $V_k(j,i)$ is the i-th coefficient of the j-th vector in the codebook of stage k.

The decoded global gain is given by

$$g_{\text{FD-CNG}}^{[\text{SID}]} = \frac{I_{\text{g,FD-CNG}} - 60}{1.5}.$$ (1973)

The SID parameters are then obtained

$$N_{\text{FD-CNG}}^{[\text{SID}]}(i) = 10^{\left(\frac{\overline{N}_{\text{FD-CNG}}^{[\text{SID}]}(i) + g_{\text{FD-CNG}}^{[\text{SID}]}}{10}\right)}.$$ (1974)

Finally the last band parameter is adjusted in case the encoded last band size is different from the decoded last band size

$$if \; \big((bandwidth = NB) \vee (bandwidth = SWB \wedge bitrate \leq 13.2 kbps)\big)$$
$$then$$
$$N_{\text{FD-CNG}}^{[\text{SID}]}(L_{\text{SID}}-1) = 0.8 N_{\text{FD-CNG}}^{[\text{SID}]}(L_{\text{SID}}-1)$$

6.7.3.1.2 SID parameters interpolation

The SID parameters are interpolated using linear interpolation in the log domain. The interpolation is carried out separately for the FFT partitions ($i < L_{\text{SID}}^{[\text{FFT}]}$) and CLDFB partitions ($i \geq L_{\text{SID}}^{[\text{FFT}]}$).

$$N_{\text{FD-CNG}}^{[\text{SID,FR}]}(j) = \begin{cases} 0 & j = 0,...,j_{\min}(0)-1 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i) & j = j_{\min}(i),...,j_{\text{mid}}(i) \; i = 0 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i-1)(\Delta(i))^{j-j_{\text{mid}}^{[\text{SID}]}(i-1)} & j = j_{\text{mid}}(i-1)+1,...,j_{\text{mid}}(i) \; i = 1,...,L_{\text{SID}}^{[\text{FFT}]}-1 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i)\min(\Delta(i),1)^{j-j_{\text{mid}}^{[\text{SID}]}(i)} & j = j_{\text{mid}}(i)+1,...,j_{\max}(i) \; i = L_{\text{SID}}^{[\text{FFT}]}-1 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i) & j = j_{\min}(i),...,j_{\text{mid}}(i) \; i = L_{\text{SID}}^{[\text{FFT}]} \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i-1)(\Delta(i))^{j-j_{\text{mid}}^{[\text{SID}]}(i-1)} & j = j_{\text{mid}}(i-1)+1,...,j_{\text{mid}}(i) \; i = L_{\text{SID}}^{[\text{FFT}]},...,L_{\text{SID}}-1 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i)\min(\Delta(i),1)^{j-j_{\text{mid}}^{[\text{SID}]}(i)} & j = j_{\text{mid}}(i)+1,...,j_{\max}(i) \; i = L_{\text{SID}}-1 \\ 0 & j = j_{\max}(i)+1,...,L_{FFT}/2 \; i = L_{\text{SID}}-1 \end{cases}$$ (1975)

where

$$j_{\text{mid}}(i) = \begin{cases} j_{\max}(0) & i = 0 \\ \left\lfloor \dfrac{j_{\max}(i-1) + j_{\max}(i)+1}{2} \right\rfloor & i = 1,...,L_{\text{SID}}-1 \end{cases}$$ (1976)

denotes the centre bin in each spectral partition, and

$$\Delta(i) = \exp\left(\frac{\log(N_{\text{FD-CNG}}^{[\text{SID}]}(i)) - \log(N_{\text{FD-CNG}}^{[\text{SID}]}(i-1))}{j_{\text{mid}}(i) - j_{\text{mid}}(i-1)}\right)$$ (1977)

is the multiplicative increment.

6.7.3.1.3 LPC estimation from the interpolated SID parameters

A set of LPC coefficients is estimated from the SID spectrum in order to update excitation and LPC related memories.

A noise floor is first added to the interpolated SID parameters and a pre-emphasis function is then applied in the frequency domain

$$\tilde{N}_{\text{FD}-\text{CNG}}^{[\text{SID,FR}]}(j) = \max\left(N_{\text{FD}-\text{CNG}}^{[\text{SID,FR}]}(j), 0.001\left(1 + pf^2 - 2pf\cos\left(\frac{-2\pi j}{L_{\text{FFT}}^{[\text{FD}-\text{CNG}]}}\right)\right)\right),$$ (1978)

with pf is the pre-emphasis factor (0.68 at 12.8 kHz and 0.72 at 16 kHz).

The noise estimates are then transformed using an inverse FFT, producing autocorrelation coefficients

$$r_{\text{FD}-\text{CNG}}(n) = \frac{L_{\text{FFT}}^{[\text{FD}-\text{CNG}]}}{2}\sum_{j=0}^{L_{\text{FFT}}^{[\text{FD}-\text{CNG}]}/2}\tilde{N}_{\text{FD}-\text{CNG}}^{[\text{SID,FR}]}(j)\cos\left(2\pi n\frac{j}{L_{\text{FFT}}^{[\text{FD}-\text{CNG}]}}\right), \quad n = 0,...,16.$$ (1979)

Then the first autocorrelation coefficient is adjusted

$$r_{\text{FD}-\text{CNG}}(0) = 1.0005\max(r_{\text{FD}-\text{CNG}}(0), 100).$$ (1980)

And finally LPC coefficients $a_{\text{FD}-\text{CNG}}(n), n = 0,...,16$ are estimated from the autocorrelation coefficients $r_{\text{FD}-\text{CNG}}$ using Levinson-Durbin (see subclause 5.1.9.4).

6.7.3.2 Noise tracking during active frames in FD-CNG

At the decoder side, the noise estimator is applied at the output of the core decoder during active frames. To achieve a trade-off between spectral resolution and computational complexity, spectral energies are averaged among groups of spectral bands called partitions, just like in the encoder (see subclause 5.6.3.1). However, the size of each partition is significantly smaller in the decoder compared to the encoder, yielding thereby a finer quantization of the frequency axis in the decoder. Moreover, the decoder-side noise estimation operates solely in the FFT domain and covers only the core bandwidth. Hence FFT partitions are formed, but no CLDFB partitions.

6.7.3.2.1 Spectral partition energies

The output of the core decoder is first transformed by an FFT of size $L_{\text{FFT}}^{[\text{FD}-\text{CNG}]} = sr_{\text{CELP}}/25$, where sr_{CELP} refers to the sampling rate of the core decoder output. Then L_{shaping} partitions are formed as follows:

$$E_{\text{FD}-\text{CNG}}(i) = \frac{4}{(L_{\text{FFT}}^{[\text{FD}-\text{CNG}]})^2}\frac{\sum_{j=j_{\min}^{[\text{shaping}]}(i)}^{j_{\max}^{[\text{shaping}]}(i)}|X(j)|^2}{j_{\max}^{[\text{shaping}]}(i) - j_{\min}^{[\text{shaping}]}(i) + 1} \qquad i = 0,...,L_{\text{shaping}} - 1,$$ (1981)

where $X(j)$ is the FFT transform of the core output signal. The following table lists the number of partitions and their upper boundaries for the different FD-CNG configurations at the decoder, as a function of bandwidths and bit-rates.

Table 173: FD-CNG decoder parameters

	Bit-rates (kbps)	L_{shaping}	$f_{\max}^{[\text{shaping}]}(i), i = 0,...,L_{\text{shaping}} - 1$ (Hz)
NB	•	56	50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1075, 1175, 1275, 1375, 1475, 1600, 1725, 1850, 2000, 2150, 2325, 2500, 2700, 2925, 3150, 3400, 3975
WB/ SWB/ FB	≤ 13.2	62	50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1075, 1175, 1275, 1375, 1475, 1600, 1725, 1850, 2000, 2150, 2325, 2500, 2700, 2925, 3150, 3375, 3700, 4050, 4400, 4800, 5300, 5800, 6375
	> 13.2	61	50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1075, 1175, 1275, 1375, 1475, 1600, 1725, 1850, 2000, 2150, 2325, 2500, 2700, 2925, 3150, 3400, 3700, 4400, 5300, 6400, 7700, 7975

For each partition $i = 0,...,L_{\text{shaping}} - 1$, $f_{\max}^{[\text{shaping}]}(i)$ corresponds to the frequency of the last band in the i-th partition. The indices $j_{\min}^{[\text{shaping}]}(i)$ and $j_{\max}^{[\text{shaping}]}(i)$ of the first and last bands in each spectral partition can be derived as a function of the FFT size $L_{\text{FFT}}^{[\text{FD-CNG}]}$ and the sampling rate of the core decoder sr_{CELP} as follows:

$$j_{\max}^{[\text{shaping}]}(i) = f_{\max}^{[\text{shaping}]}(i) \frac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{sr_{\text{CELP}}} \qquad i = 0,...,L_{\text{shaping}} - 1, \qquad (1982)$$

$$j_{\min}^{[\text{shaping}]}(i) = \begin{cases} f_{\min}^{[\text{shaping}]}(0) \dfrac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{sr_{\text{CELP}}} & i = 0 \\ j_{\max}^{[\text{shaping}]}(i-1)+1 & i > 0 \end{cases}, \qquad (1983)$$

where $f_{\min}^{[\text{shaping}]}(0) = 50\text{Hz}$ is the frequency of the first band in the first spectral partition. Hence the FD-CNG generates some comfort noise above 50Hz only.

6.7.3.2.2   FD-CNG noise estimation

In FD-CNG, encoder and decoder rely on the same noise estimator, except that the number of partitions differs. As in subclause 5.6.3.2, the input partition energies are first processed by a non-linear transform before applying the noise tracking algorithm on the inputs $E_{\text{FD-CNG}}(i), i = 0,...,L_{\text{shaping}} - 1$. The inverse transform is then used to recover the original dynamic range. In the sequel, the resulting decoder-side noise estimates are referred to as $N_{\text{FD-CNG}}^{[\text{shaping}]}(i), i = 0,...,L_{\text{shaping}} - 1$. They are used as shaping parameters in the next subclause.

6.7.3.2.3   Noise shaping in FD-CNG

Note that the shaping parameters $N_{\text{FD-CNG}}^{[\text{shaping}]}(i), i = 0,...,L_{\text{shaping}} - 1$ computed directly at the decoder differ from the SID parameters $N_{\text{FD-CNG}}^{[\text{SID}]}(i), i = 0,...,L_{\text{SID}}^{[\text{FFT}]} - 1$ which are transmitted via SID frames. Both sets are computed from FFT partitions covering the core bandwidth, but the decoder benefits from a significantly higher number of spectral partitions, i.e., $L_{\text{shaping}} > L_{\text{SID}}^{[\text{FFT}]}$. In fact, the high-resolution noise estimates obtained at the decoder capture information about the fine spectral structure of the background noise. However, the decoder-side noise estimates cannot adapt to changes in the actual background noise during inactive phases. In contrast, the SID frames deliver new information about the spectral envelop at regular intervals during inactive phases. The FD-CNG therefore combines these two

sources of information in an effort to reproduce the fine spectral structure captured from the background noise present during active phases, while updating only the spectral envelop of the comfort noise during inactive parts with the help of the SID information.

6.7.3.2.3.1 Conversion to a lower spectral resolution

Interpolation is first applied to the shaping parameters $N_{\mathrm{FD-CNG}}^{[\mathrm{shaping}]}(i)$ to obtain a full-resolution FFT power spectrum as follows:

$$
N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,FR}]}(j) = \begin{cases} 0 & j = 0,...,j_{\min}^{[\mathrm{shaping}]}(0)-1 \\ N_{\mathrm{FD-CNG}}^{[\mathrm{shaping}]}(i) & j = j_{\min}^{[\mathrm{shaping}]}(i),...,j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i) & i = 0 \\ N_{\mathrm{FD-CNG}}^{[\mathrm{shaping}]}(i-1)\big(\Delta(i)\big)^{j-j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i-1)} & j = j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i-1)+1,...,j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i) & i = 1,...,L_{\mathrm{shaping}}-1 \\ N_{\mathrm{FD-CNG}}^{[\mathrm{shaping}]}(i)\min\big(\Delta(i),1\big)^{j-j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i)} & j = j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i)+1,...,j_{\max}^{[\mathrm{shaping}]}(i) & i = L_{\mathrm{shaping}}-1 \end{cases},
$$

(1984)

where

$$
j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i) = \begin{cases} j_{\max}^{[\mathrm{shaping}]}(0) & i = 0 \\ \left\lfloor \dfrac{j_{\max}^{[\mathrm{shaping}]}(i-1)+j_{\max}^{[\mathrm{shaping}]}(i)+1}{2} \right\rfloor & i = 1,...,L_{\mathrm{shaping}}-1 \end{cases}
$$

(1985)

denote the center FFT bins in each spectral partition, and $\Delta(i) = \exp\left(\dfrac{\log(N_{\mathrm{FD-CNG}}^{[\mathrm{shaping}]}(i))-\log(N_{\mathrm{FD-CNG}}^{[\mathrm{shaping}]}(i-1))}{j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i)-j_{\mathrm{mid}}^{[\mathrm{shaping}]}(i-1)}\right)$ is the

multiplicative increment for the interpolation. The above corresponds in fact to a linear interpolation in the log domain of the FFT shaping partitions.

The full-resolution spectrum $N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,FR}]}(j)$ is subsequently converted again to a lower resolution based on the SID spectral partitions (see subclause 5.6.3.6). The resulting noise energy spectrum $N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,LR}]}(i), i = 0,...,L_{\mathrm{SID}}^{[\mathrm{FFT}]}-1$ exhibits therefore the same spectral resolution as the SID parameters $N_{\mathrm{FD-CNG}}^{[\mathrm{SID}]}(i), i = 0,...,L_{\mathrm{SID}}^{[\mathrm{FFT}]}-1$. Hence, both sets are comparable and can be combined in the next subclause.

6.7.3.2.3.2 Combining SID and shaping parameters

Comparing the low-resolution noise estimates $N_{\mathrm{FD-CNG}}^{[\mathrm{SID}]}(i)$ and $N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,LR}]}(i)$ obtained from the encoder (via SID frames) and decoder, respectively, the full-resolution noise spectrum $N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,FR}]}(i)$ can now be scaled to yield a full-resolution noise power spectrum as follows:

$$
N_{\mathrm{FD-CNG}}^{[\mathrm{CNG}]}(j) = \begin{cases} 0 & j = 0,...,j_{\min}^{[\mathrm{SID}]}(0)-1 \\ \dfrac{N_{\mathrm{FD-CNG}}^{[\mathrm{SID}]}(i)}{N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,LR}]}(i)} N_{\mathrm{FD-CNG}}^{[\mathrm{shaping,FR}]}(j) & j = j_{\min}^{[\mathrm{SID}]}(i),...,j_{\max}^{[\mathrm{SID}]}(i) & i = 0,...,L_{\mathrm{SID}}^{[\mathrm{FFT}]}-1 \end{cases},
$$

(1986)

where $j_{\min}^{[\mathrm{SID}]}(i)$ and $j_{\max}^{[\mathrm{SID}]}(i)$ refer to the first and last FFT bin of the *i*-th SID partition (see subclause 5.6.3.6). The full-resolution noise power spectrum $N_{\mathrm{FD-CNG}}^{[\mathrm{CNG}]}(j)$ is recomputed for each active frame. It can be used to accurately adjust the level of comfort noise in each FFT bin during SID frames or zero frames, as shown in the next subclause.

6.7.3.3 Noise generation for SID or zero frames in FD-CNG

6.7.3.3.1 Update of the noise levels for FD-CNG

During the first non-active frame following an active frame, the low-resolution shaping parameters $N_{\text{FD-CNG}}^{[\text{shaping,LR}]}(i), i = 0,...,L_{\text{SID}}^{[\text{FFT}]}-1$ are recomputed from $N_{\text{FD-CNG}}^{[\text{shaping,FR}]}(j)$ by averaging over the SID partitions as in subclause 5.6.3.1.1.

If an SID frame occurs while the noise estimator (subclause 6.7.3.2.2) is still in its initialization phase, the interpolated SID parameters $N_{\text{FD-CNG}}^{[\text{SID,FR}]}(i)$ (see subclause 6.7.3.1.2) are used as comfort noise levels.

If an SID frame occurs once the noise estimator left the initialization phase, the comfort noise levels are computed for FFT bins by combining the noise estimates from encoder and decoder as described in subclause 6.7.3.2.3.2, while the CLDFB levels are obtained directly by interpolating the SID parameters corresponding to the CLDFB partitions, i.e.,

$$
N_{\text{FD-CNG}}^{[\text{CNG}]}(j) = \begin{cases} 0 & j = 0,...,j_{\min}^{[\text{SID}]}(0)-1 \\ \dfrac{N_{\text{FD-CNG}}^{[\text{SID}]}(i)}{N_{\text{FD-CNG}}^{[\text{shaping,LR}]}(i)} N_{\text{FD-CNG}}^{[\text{shaping,FR}]}(j) & j = j_{\min}^{[\text{SID}]}(i),...,j_{\max}^{[\text{SID}]}(i) & i = 0,...,L_{\text{SID}}^{[\text{FFT}]}-1 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i) & j = j_{\min}^{[\text{SID}]}(i),...,j_{\text{mid}}^{[\text{SID}]}(i) & i = L_{\text{SID}}^{[\text{FFT}]} \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i-1)(\Delta(i))^{j-j_{\text{mid}}^{[\text{SID}]}(i-1)} & j = j_{\text{mid}}^{[\text{SID}]}(i-1)+1,...,j_{\text{mid}}^{[\text{SID}]}(i) & i = L_{\text{SID}}^{[\text{FFT}]},...,L_{\text{SID}}-1 \\ N_{\text{FD-CNG}}^{[\text{SID}]}(i)\min(\Delta(i),1)^{j-j_{\text{mid}}^{[\text{SID}]}(i)} & j = j_{\text{mid}}^{[\text{SID}]}(i)+1,...,j_{\max}^{[\text{SID}]}(i) & i = L_{\text{SID}}-1 \end{cases}
$$ (1987)

where $N_{\text{FD-CNG}}^{[\text{SID}]}(i)$ are the SID parameters decoded from the SID frames, $N_{\text{FD-CNG}}^{[\text{shaping,LR}]}(i)$ are computed during the first non-active frame following an active frame, as explained just above, and $N_{\text{FD-CNG}}^{[\text{shaping,FR}]}(j)$ is the full-resolution noise spectrum obtained by interpolating the decoder-side noise estimates during active frames (see subclause 6.7.3.2.3.1).

6.7.3.3.2 Comfort noise generation in the frequency domain

The FFT noise spectrum corresponding to the first $j_{\max}^{[\text{SID}]}(L_{\text{SID}}^{[\text{FFT}]}-1)$ parameters in the array $N_{\text{FD-CNG}}^{[\text{CNG}]}(j)$ can finally be used to generate some random Gaussian noise of zero mean and variance $N_{\text{FD-CNG}}^{[\text{CNG}]}(j)/2$ separately for the real and imaginary parts of the FFT coefficients.

The second part of the CNG spectrum, i.e., $N_{\text{FD-CNG}}^{[\text{CNG}]}(j), j = j_{\min}^{[\text{SID}]}(L_{\text{SID}}^{[\text{FFT}]}),...,j_{\max}^{[\text{SID}]}(L_{\text{SID}}-1)$ corresponds to the CLDFB noise levels for frequencies above the core bandwidth. For each CLDFB time slot, some random Gaussian noise of zero mean and variance $N_{\text{FD-CNG}}^{[\text{CNG}]}(j)/2$ is generated separately for the real and imaginary parts of the CLDFB coefficients corresponding to frequencies above the core bandwidth.

6.7.3.3.3 Comfort noise generation in the time domain

The FFT coefficients obtained after comfort noise generation in the frequency domain are transformed by an inverse FFT, producing a CNG time-domain signal of length $L_{\text{FFT}}^{[\text{FD-CNG}]}$. This signal is then windowed using a sine-based window that can be defined as follow

$$
w(n) = 0 \text{, for } n = 0,...,\frac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{8}-1 .
$$ (1988)

$$
w(n) = \sin\left[\left(n - \frac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{8} + \frac{1}{2}\right)\frac{2\pi}{L_{\text{FFT}}^{[\text{FD-CNG}]}}\right] \text{, for } n = \frac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{8},...,\frac{3L_{\text{FFT}}^{[\text{FD-CNG}]}}{8}-1 .
$$ (1989)

$$w(n) = 1 \text{, for } n = \frac{3L_{\text{FFT}}^{[\text{FD-CNG}]}}{8}, \ldots, \frac{5L_{\text{FFT}}^{[\text{FD-CNG}]}}{8} - 1 \text{.} \tag{1990}$$

$$w(n) = \sin\left[\left(n - \frac{3L_{\text{FFT}}^{[\text{FD-CNG}]}}{8} + \frac{1}{2}\right)\frac{2\pi}{L_{\text{FFT}}^{[\text{FD-CNG}]}}\right] \text{, for } n = \frac{5L_{\text{FFT}}^{[\text{FD-CNG}]}}{8}, \ldots, \frac{7L_{\text{FFT}}^{[\text{FD-CNG}]}}{8} - 1 \text{.} \tag{1991}$$

$$w(n) = 0 \text{, for } n = \frac{7L_{\text{FFT}}^{[\text{FD-CNG}]}}{8}, \ldots, L_{\text{FFT}}^{[\text{FD-CNG}]} - 1 \text{.} \tag{1992}$$

An overlap-add method is finally applied on the current windowed CNG signal and the previous windowed CNG signal. The final FD-CNG frame corresponds to $n = \frac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{8}, \ldots, \frac{3L_{\text{FFT}}^{[\text{FD-CNG}]}}{8} - 1$.

To avoid discontinuities at transitions from active frames to inactive frames, a cross-fading mechanism is employed. Several approaches are described in the following subclauses, depending on the bitrate and the previous encoding mode.

6.7.3.3.3.1   Transitions from MDCT to FD-CNG at 9.6kbps, 16.4kbps and 24.4kbps

At 9.6 kbps, 16.4 kbps and 24.4 kbps and when the previous frame was active and encoded with an MDCT-based coding mode, a cross-fading operation is performed using the MDCT window.

First, the left part of the FD-CNG frame is not windowed after the inverse FFT and there is no overlap-add operation with the previous (missing) FD-CNG frame.

Instead, the left part of the FD-CNG frame is windowed using the complementary version of the MDCT window used in the right part of the previous MDCT-based frame (see subclause 6.2.4). An overlap-add method is then applied on the MDCT-windowed FD-CNG frame and the previous MDCT-based frame.

6.7.3.3.3.2   Transitions from ACELP to FD-CNG at 9.6kbps, 16.4kbps and 24.4kbps

At 9.6 kbps, 16.4 kbps and 24.4 kbps and when the previous frame was active and encoded with ACELP, a cross-fading operation is performed using an extrapolation of the previous ACELP frame.

A random excitation with the same energy as the last half of the excitation of the previous ACELP frame is computed

$$exc(n) = randn(n)\sqrt{\frac{2\sum_{n=\frac{L_{celp}}{2}}^{L_{celp}-1} u'(n)}{\sum_{n=0}^{L_{celp}-1} rand(n)}} , \quad n = 0, \ldots, L_{celp} - 1 \tag{1993}$$

where $u'(n)$ is the total excitation of the previous ACELP frame (see subclause 6.1.1.2), $rand(n)$ is a random Gaussian noise with zero mean and $L_{celp} = \frac{L_{\text{FFT}}^{[\text{FD-CNG}]}}{2}$ is the length of the ACELP frame.

The random excitation $exc(n)$ is then filtered through the same LPC synthesis filter and de-emphasis filter as used in the last subframe of the previous ACELP frame (see subclause 6.1.3), producing the extrapolated ACELP signal.

Finally the extrapolated ACELP signal is windowed and the overlap-add method is applied as if the extrapolated ACELP signal was the previous FD-CNG frame.

6.7.3.3.3.3   Transitions from active to FD-CNG at bitrates<=8kbps and at 13.2kbps

At bitrates<=8 kbps and at 13.2 kbps, a FD-CNG frame is converted to a combination of an excitation signal and a set of LPC coefficients. It then becomes very similar to a LP-CNG frame, and can be further processed as such.

First, the left part of the FD-CNG frame is not windowed after the inverse FFT and there is no overlap-add operation with the previous (missing) FD-CNG frame.

The time-domain FD-CNG signal is then pre-emphasized and filter through the LP analysis filter $a_{\text{FD}-\text{CNG}}(n), n = 0,...,16$ (see subclause 6.7.3.1.3), producing an excitation signal.

The set of LPC coefficients associated with the excitation is $a_{\text{FD}-\text{CNG}}(n), n = 0,...,16$. These LPC coefficients are then used to synthesize the final CNG signal using the excitation and the filter memories from the previous frame, as it is done in LP-CNG.

6.7.3.3.4 FD-CNG decoder memory update

The LPC coefficients $a_{\text{FD}-\text{CNG}}(n), n = 0,...,16$ are converted to LSP and to LSF. The LPC, LSP and LSF are then used to update all LPC/LSP/LSF-related memories.

The FD-CNG time-domain output is used to update all signal domain memories.

The FD-CNG time-domain signal is pre-emphasized and the pre-emphasized signal is then used to update all pre-emphasized signal domain memories.

The pre-emphasized signal is filtered through the LP analysis filter $a_{\text{FD}-\text{CNG}}(n), n = 0,...,16$ and the obtained residual is then used to update all excitation domain memories.

All the other memories are updated similarly to LP-CNG (see subclause 6.7.2.2).

6.8   AMR-WB-interoperable modes

6.8.1   Decoding and speech synthesis

6.8.1.1   Excitation decoding

The decoding process is performed in the following order:

Decoding of LP filter parameters: The received indices of ISP quantization are used to reconstruct the quantized ISP vector. The interpolation described in subclause 5.7.2.6 is performed to obtain 4 interpolated ISP vectors (corresponding to 4 subframes). For each subframe, the interpolated ISP vector is converted to LP filter coefficient domain a_k, which is used for synthesizing the reconstructed speech in the subframe.

The following steps are repeated for each subframe:

1. **Decoding of the adaptive codebook vector:** The received pitch index (adaptive codebook index) is used to find the integer and fractional parts of the pitch lag. The adaptive codebook vector $v(n)$ is found by interpolating the past excitation $u(n)$ (at the pitch delay) using the FIR filter described in subclause 5.7. The received adaptive filter index is used to find out whether the filtered adaptive codebook is $v1(n) = v(n)$ or $v2(n) = 0.18v(n) + 0.64v(n-1) + 0.18v(n-2)$.

2. **Decoding of the innovative vector:** The received algebraic codebook index is used to extract the positions and amplitudes (signs) of the excitation pulses and to find the algebraic codevector $c(n)$. If the integer part of the pitch lag is less than the subframe size 64, the pitch sharpening procedure is applied which translates into modifying $c(n)$ by filtering it through the adaptive prefilter $F(z)$ which consists of two parts: a periodicity enhancement part $1/(1 - 0.85z^{-T})$ and a tilt part $(1 - \beta_1 z^{-1})$, where T is the integer part of the pitch lag and $\beta_1(n)$ is related to the voicing of the previous subframe and is bounded by [0.0,0.5].

3. **Decoding of the adaptive and innovative codebook gains:** The received index gives the fixed codebook gain correction factor $\hat{\gamma}$. The estimated fixed codebook gain g_c' is found as described in subclause 5.8. First, the predicted energy for every subframe n is found by

$$\tilde{E}(n) = \sum_{i=1}^{4} b_i \hat{R}(n-i) \tag{1994}$$

and then the mean innovation energy is found by

$$E_i = 10\log(\frac{1}{N}\sum_{i=0}^{N-1} c^2(i)) \tag{1995}$$

The predicted gain g_c' is found by

$$g_c' = 10^{0.05(\tilde{E}(n) + \overline{E} - E_i)} \tag{1996}$$

The quantized fixed codebook gain is given by

$$\hat{g}_c = \hat{\gamma} g_c'. \tag{1997}$$

4. **Computing the reconstructed speech:** The following steps are for $n = 0, ..., 63$. The total excitation is constructed by:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n) \tag{1998}$$

6.8.1.2 Excitation post-processing

Before the synthesis, a post-processing of the excitation signal is performed to form the updated excitation signal, $u(n)$, as follows.

6.8.1.2.1 Anti-sparseness processing

This is the same as described in subclause 6.1.1.3.1

6.8.1.2.2 Gain smoothing for noise enhancement

This is the same as described in subclause 6.1.1.3.2

6.8.1.2.3 Pitch enhancer

This is the same as described in subclause 6.1.1.3.3.

6.8.1.3 Synthesis filtering

Once the excitation post-processing is done, the modified excitation is passed through the synthesis filter, as described in subclause 6.1.3, to obtain the decoded synthesis for the current frame. Based on the content bandwidth in the decoded synthesis signal, an output mode is determined (e.g., NB or WB). If the output mode is determined to be NB, then the content above 4 kHz is attenuated using CLDFB synthesis (e.g., as described in clause 6.9.3) and, subsequently, high frequency synthesis (6.8.3) is not performed on the bandlimited content.

6.8.1.4 Music and Unvoiced/inactive Post-processing

6.8.1.4.1 Music post processing

Most of the music post processing is the same as in as clause 6.1.1.3.4. The main difference related to the fact that a first synthesis is computed and a first stage classification is derived from this synthesis as described in subclause 5.3.1 of [6]. If the synthesis is classified as unvoiced or the content is INACTIVE (VAD ==0) or the long term background noise (E_{lt_noise}) as defined below is greater or equal to 15 dB, the AMR-IO decoder will go through the unvoiced, inactive post processing path as described in subclause 6.8.1.1.5.

The long term background noise energy is updated in case of INACTIVE frame as:

$$E_{lt\_noise} = 0.9 \cdot E_{lt\_noise} + 0.1 \cdot \left(10 \log_{10} \left(\frac{1}{T'} \sum_{n=0}^{T'-1} \hat{s}^2 (L - T' + n) \right) \right) \tag{1999}$$

and E_{lt_noise} is the long-term background noise energy. E_{lt_noise} is updated only when a current frame is classified as INACTIVE. The pitch lag value, T', over which the background noise energy, E_{lt_noise}, is given by

$$\begin{aligned} p &= \text{round}(0.5 d_{fr}^{[2]} + 0.5 d_{fr}^{[3]}) \\ T' &= p \quad \text{if} \quad p \geq L_{subfr} \\ T' &= 2p \quad \text{if} \quad p < L_{subfr} \end{aligned} \tag{2000}$$

where $d_{fr}^{[i]}$ is the fractional pitch lag at subframe i, L is the frame length and L_{subfr} is the subframe length. Otherwise it enters the music post processing is entered as described below.

6.8.1.4.1.1 Excitation buffering and extrapolation

This is the same as described in subclause 6.1.1.3.4.1

### 6.8.1.4.1.2	Windowing and frequency transform

This is the same as described in subclause 6.1.1.3.4.2

### 6.8.1.4.1.3	Energy per band and per bin analysis

This is the same as described in subclause 6.1.1.3.4.3

### 6.8.1.4.1.4	Excitation type classification

This is the same as described in subclause 6.1.1.3.4.4

### 6.8.1.4.1.5	Inter-tone noise reduction in the excitation domain

This is the same as described in subclause 6.1.1.3.4.5

### 6.8.1.4.1.6	Inter-tone quantization noise estimation

This is the same as described in subclause 6.1.1.3.4.6

### 6.8.1.4.1.7	Increasing spectral dynamic of the excitation

This is the same as described in subclause 6.1.1.3.4.7

### 6.8.1.4.1.8	Per bin normalization of the spectrum energy

This is the same as described in subclause 6.1.1.3.4.8

### 6.8.1.4.1.9	Smoothing of the scaled energy spectrum along the frequency axis and the time axis

This is the same as described in subclause 6.1.1.3.4.9

### 6.8.1.4.1.10	Application of the weighting mask to the enhanced concatenated excitation spectrum

This is the same as described in subclause 6.1.1.3.4.10

### 6.8.1.4.1.11	Inverse frequency transform and overwriting of the current excitation

This is the same as described in subclause 6.1.1.3.4.11

## 6.8.1.4.2	Unvoiced and inactive post processing

When the classifier described in subclause 5.3.1 of [6] considers the synthesis as unvoiced or inactive and containing background noise, the unvoiced and inactive post processing module is used to determine a cut-off frequency where the time-domain contributions should stop. Then the content above this cut-off frequency is replaced with random noise giving a smoother rendering of the synthesis. This post processing module is used when the local attack flag (l_{af} as defined in subclause 5.3.1 [6] is set to 0 and the coding type is INACTIVE and the bitrate is below or equal to 12650 bps. It is also used at 6600 bps if the synthesis is classified as UNVOICED or VOICED_TRANSITION.

When the synthesis is considered as INACTIVE and the energy of the synthesis as defined in subclause 5.3.1 of [6] is greater than -3 dB, the LP filter coefficients that will be used to do the synthesis filtering, as described below in subclause 6.8.1.1.4.5, are smoothed as between past and current frame as follow:

$$A_q{}'(k) = 0.7 \cdot A_{q(t-1)}(k) + 0.3 \cdot A_q(k), \qquad\qquad for\ 0 < k < 4 \cdot (16+1) \qquad\qquad (2001)$$

where $A_{q(t-1)}$ represents the LP filter of the previous frame. At the end of the post processing $A_{q(t-1)}$ is updated using $A_q{}'$.

6.8.1.4.2.1 Frequency transform

During the frequency-domain modification phase, the excitation needs to be represented into the transform-domain. The time-to-frequency conversion is achieved with a type II DCT giving a resolution of 25Hz. The frequency representation of the time-domain CELP excitation $f_u(k)$ is given below:

$$f_u(k) = \begin{cases} \sqrt{\dfrac{1}{L}} \cdot \sum_{n=0}^{L-1} u(n), & k = 0 \\ \sqrt{\dfrac{2}{L}} \cdot \sum_{n=0}^{L-1} u(n) \cdot \cos\left(\dfrac{\pi}{L}\left(n + \dfrac{1}{2}\right)k\right), & 1 \le k \le L-1 \end{cases} \tag{2002}$$

where $u(n)$ $e_{td}(n)$, is the time-domain excitation, and L is the frame length and its value is 256 samples for a corresponding inner sampling frequency of 12.8 kHz.

6.8.1.4.2.2 Energy per band analysis

Before any modification to the excitation, the energy per band $E_b(i)$ is computed and kept in memory for energy adjustment after the excitation spectrum reshaping. The energy can be computed as follow :

$$E_b(i) = \sqrt{\sum_{j=C_{Bb}(i)}^{j=C_{Bb}(i)+B_b(i)} f_u(j)^2} \tag{2003}$$

where C_{Bb} is the cumulative frequency bins per band and B_b number of bins per band defined as :

$$B_b = \{4,4,4,4,4,5,6,6,6,8,8,10,11,13,15,18,22,16,16,20,20,20,16\}$$

and

$$C_{Bb} = \left\{ \begin{array}{l} 0,8,12,16,20,25,31,37,43,51,59,69,80,93, \\ 108,126,148,164,180,200,220,240 \end{array} \right\}$$

The low frequency bands correspond to the critical audio bands, but the frequency band above 3700 Hz are a little shorter to better match the possible spectral energy variation in those bands.

6.8.1.4.2.3 Excitation modification

6.8.1.4.2.3.1 Cut off frequency of the temporal contribution

To achieve a transparent switching between the non-modified excitation and the modified excitation for unvoiced and inactive signals, it is preferable to keep at least the lower frequencies of the temporal contribution. The frequency where the temporal contribution stop to be used, the cut-off frequency f_c, has a minimum value of 1.2 kHz. It means that the first 1.2 kHz of the decoded excitations is always kept and depending of the pitch value, this cut-off frequency can be higher. The 8th harmonic is computed from the lowest pitch of all subframes and the temporal contribution is kept up to this 8th harmonic. The estimate is performed as follow:

$$h_{8th} = \frac{(8 \cdot F_s)}{\min(T(k))_{k=0}^{k=3}} \tag{2004}$$

where $F_s = 12800$ and T the decoded subframe pitch.

For all bands a verification is made to find the band in which the 8th harmonic is located by searching for the highest frequency band L_f for which the following inequality is still verified:

$$\left(h_{8^{th}} \geq L_f(i)\right) \tag{2005}$$

where the frequency band L_f is defined as :

$$L_f = \left\{ \begin{array}{l} 175, 275, 375, 475, 600, 750, 900, 1050, 1250, 1450, 1700, 1975, \\ 2300, 2675, 3125, 3675, 4075, 4475, 4975, 5475, 5975, 6375 \end{array} \right\}$$

The index of that band will be called $i_{8^{th}}$ and it indicates the band where the 8th harmonic is likely located. The finale cut-off frequency f_{tc} is computed as the higher frequency between the 1.2 kHz and the last frequency of the frequency band in which the 8th harmonic is located $\left(L_f(i_{8^{th}})\right)$, using the following relation:

$$f_{tc} = \max\left(L_f(i_{8^{th}}), 1.2\,\text{kHz}\right) \tag{2006}$$

6.8.1.4.2.3.2   Normalization and noise fill

For unvoiced and inactive frames, the frequency bins below f_{tc} f_c are normalized between [0, 4] :

$$\overline{f_u}(j) = \begin{cases} \dfrac{4 \cdot f_u(j)}{\max\limits_{0 \leq i < f_c} \left(|f_u(i)|\right)} , & for \quad 0 \leq j < f_{tc} \\ 0 , & for \ f_{tc} \leq j < 256 \end{cases} \tag{2007}$$

And the frequency bins above f_{tc} f_c are zeroed. Then, a simple noise fill is performed to add noise over all the frequency bins at a constant level. The function describing the noise addition is defined below as:

$$\begin{array}{l} for \ j = 0 : L\text{-}1 \\ \overline{f_u}'(j) = \overline{f_u}(j) + 0.75 \cdot s_r(j) \end{array} \tag{2008}$$

Where s_r is a random number generator which is limited between -1 to 1 as :

$$s_r(j) = \frac{\text{float}(\text{short}[s_r(j-1) \times 31821 + 13849])}{32768} \tag{2009}$$

6.8.1.4.2.3.3   Energy per band analysis of the modified excitation spectrum

The energy per band after the spectrum reshaping E_b' is calculated again with exactly the same method as described in subclause 6.8.1.1.4.2.

6.8.1.4.2.3.4   Amplification of high frequencies

An amplification factor α compensates for the poor energy matching in high frequency of the LP filter at low bit rate. It is based on the voice factor V_f and computed as follow:

$$\alpha = 0.5 * \left(1 - V_f\right) \tag{2010}$$

where V_f is given by:

$$V_f = 0.34 + 0.5 \cdot \lambda + 0.16 \cdot \lambda^2 \tag{2011}$$

and λ is defined in sub-clause 6.1.1.3.2.

The amplification factor is applied linearly between 6kHz and 6.4kHz as follow:

$$\overline{f_u}"(j) = \begin{cases} \overline{f_u}'(j) & , \quad for \quad 0 \le j < 240 \\ \overline{f_u}'(j) \cdot \max\left(1, \alpha \cdot (0.067 \cdot j - 15.0)\right) & , \quad for \quad 240 \le j < 256 \end{cases} \tag{2012}$$

6.8.1.4.2.3.5 Energy matching

The energy matching consists in adjusting the energy per band after the excitation spectrum modification to its initial value. For each bands i, the gain G_b to apply to all bins in the band for matching the energy of the original excitation f_u is defined as:

$$G_b(i) = \frac{E_b(i)}{E_b'(i)} \tag{2013}$$

For a specific band i, the denormalized f_{edN}^t f_u' spectral excitation can be written as :

$$\text{for } C_{Bb}(i) \le j < C_{Bb}(i) + B_b(i)$$
$$f_u'(j) = G_b(i) \cdot \overline{f_u}"(j) \tag{2014}$$

where C_{Bb} and B_b are defined in subclause 6.8.1.1.4.2.

6.8.1.4.2.4 Inverse frequency transform

After the frequency domain is completed, an inverse frequency-to-time transform is performed in order to find the temporal excitation. The frequency-to-time conversion is achieved with the same type II DCT as used for the time-to-frequency conversion. The modified time-domain excitation u' is obtained as below:

$$u'(k) = \begin{cases} \sqrt{\dfrac{1}{L}} \cdot \displaystyle\sum_{n=0}^{L-1} f_u'(n), & k = 0 \\ \sqrt{\dfrac{2}{L}} \cdot \displaystyle\sum_{n=0}^{L-1} f_u'(n) \cdot \cos\left(\dfrac{\pi}{L}\left(n + \dfrac{1}{2}\right)k\right), & 1 \le k \le L-1 \end{cases} \tag{2015}$$

where $f_u'(n)$ $f_{edN}'(n)$, is the frequency representation of the modified excitation, and L is the frame length that is equal to 256 samples.

6.8.1.5 Synthesis filtering and overwriting the current CELP synthesis

Once the excitation modification is done, the modified excitation is passed through the synthesis filter, as described in in subclause 6.1.3, to obtain a modified synthesis for the current frame. This modified synthesis is then used to overwrite the decoded synthesis.

6.8.1.6 Formant post-filter

The decoded synthesis is post-filtered as described in subclause 6.1.4.1.

6.8.1.7 Comfort noise addition

For frames exhibiting a high background noise level (background noise level >= 15), comfort noise is added for bitrates of 8.85 kbps and below. The comfort noise addition is described in subclause 6.9.1.

6.8.1.8 Bass post-filter

This is the same as described in subclause 6.1.4.2

6.8.2   Resampling

The decoded synthesis (after post-filtering and comfort noise addition) is resampled as described in subclause 6.5. Note that the bass-postfilter described in subclause 6.8.1.8 is actually realized as part of the resampling.

6.8.3   High frequency band

The high-frequency band generation for modes from 6.6 to 23.05 kbit/s is illustrated in figure 113. The high band is generated by generating an over-sampled excitation signal in DCT domain that is extended in the 6400-8000 Hz band above the 0-6400 Hz band. Note that in reality the high band is extended to a slightly wider band (6000-8000 Hz) to facilitate the addition of low and high-band, especially in the cross-over region around 6400 Hz. Tonal and ambiance components in the extended band are extracted and combined adaptively to obtain the extended excitation signal, which is then filtered in DCT domain. After inverse DCT, gains are applied in time domain (by sub-frame) and the extended excitation signal is filtered by an LP filter whose coefficients are derived from the LP filter.
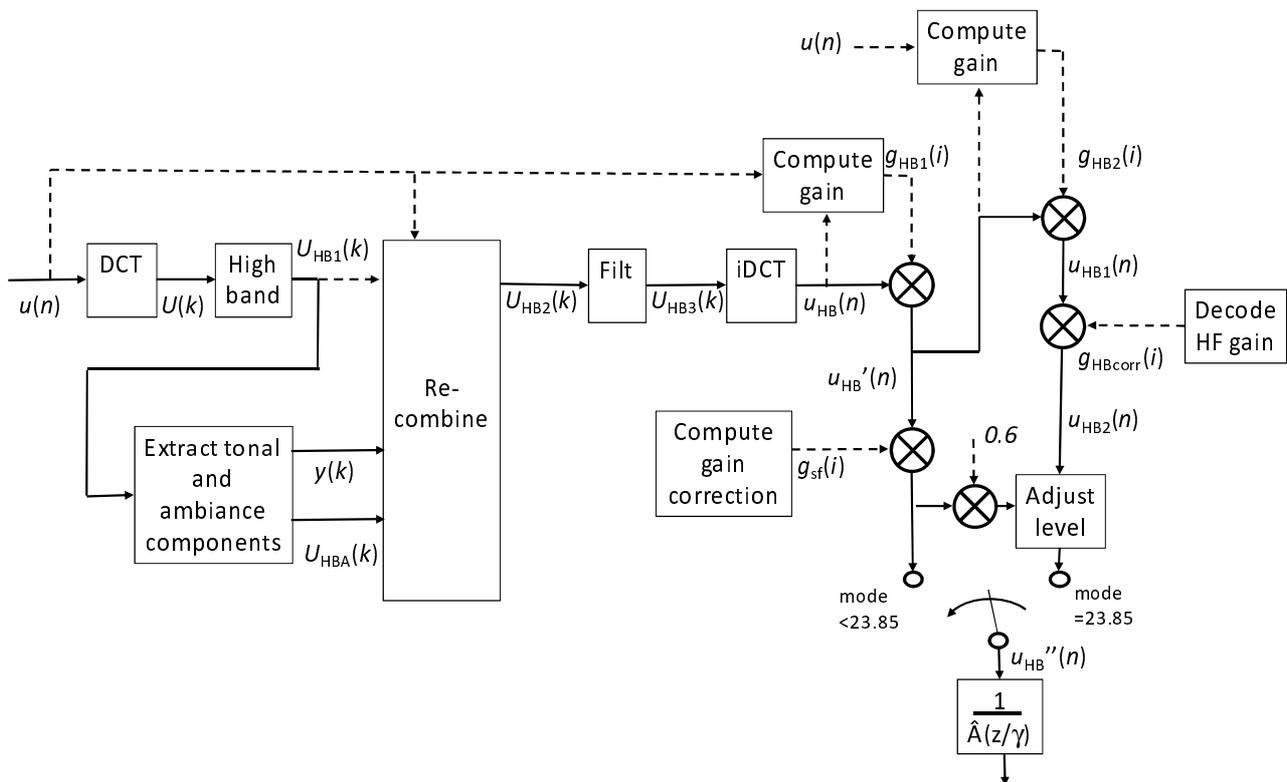
Figure 113: High-frequency band generation in AMR-WB IO modes

6.8.3.1   Preliminary estimation steps

The low-frequency band signal is extended to obtain the high frequency band signal by bandwidth extension algorithm, and the bandwidth extension algorithm includes the estimation of gains and the prediction of the excitation of the high frequency band signal.

The gains of the high frequency band signal are estimated by pitch, noise gate factor, voice factor, classification parameter and LPC.

The excitation of the high frequency band signal is adaptively predicted from the decoded low frequency band excitation signal (the sum of adaptive codebook contribution and algebraic codebook contribution) according to LSF and the bitrates.

The excitation of the high frequency band signal is modified by the gains of the high frequency band signal, and the LP synthesis is performed by filtering the modified excitation signal through the LP synthesis filter to obtain the high frequency band signal.

The parameters of estimating the gains and predicting excitation of the high frequency band signal are decoded from the bitstream of low band or calculated by the decoded low band signal.

6.8.3.1.1 Estimation of tilt, figure of merit and voice factors

1) Calculate the spectrum tilt factor of each subframe according to the decoded low frequency band as follows:

$$Til(i) = \frac{\sum\limits_{k=L_{subfr} \times i}^{L_{subfr} \times (i+1)-2} Syn(k) \times Syn(k+1)}{\sum\limits_{k=L_{subfr} \times i}^{L_{subfr} \times (i+1)-1} Syn(k) \times Syn(k)}, \qquad i = 0,1,2,3 \qquad (2016)$$

where L_{subfr} denotes the length of sub frame and $Syn(k)$ denotes the decoded low frequency band signal. $Til(i)$ is preserved as $Til0(i)$ for the following unvoiced flag calculation.

2) Calculate the sum of the differences between every two adjacent pitch values:

$$Sum_P^{diff} = \left| d^{[0]} - d^{[1]} \right| + \left| d^{[1]} - d^{[2]} \right| + \left| d^{[2]} - d^{[3]} \right| \qquad (2017)$$

where $d^{[i]}$ is the pitch value of each subframe.

3) If the frame counter f_{count} is greater than 100 and the FEC class of current frame is $UNVOICE\_CLAS$, set f_{count} to 0 and set the minimum noise gate Ng_{min} to -30. Otherwise, if f_{count} is greater than 200, set f_{count} to 200; if not, f_{count} is increased by 1; If the noise gate Ng is less than Ng_{min}, set Ng_{min} to Ng.

4) Calculate the average voice factor as follows:

$$V_{fac}^{aver} = \frac{\sum\limits_{i=0}^{3} V_{fac}(i)}{4} \qquad (2018)$$

where $V_{fac}(i)$ denotes the voice factor of each subframe.

5) Based on the classify parameter $fmerit$ from FEC classification, determine two parameters $fmerit_w$ and $fmerit_m$.

$$fmerit_w = \begin{cases} 0.35, & if\ fmerit > 0.35 \\ 0.15, & else\ if\ fmerit < 0.15 \\ fmerit, & otherwise \end{cases} \qquad (2019)$$

and if the FEC class of current frame is $AUDIO\_CLAS$, $fmerit_w = 0.5 \times fmerit_w$.

Then $fmerit_w$ is further modified by the average voice factor V_{fac}^{aver} and smoothed as follows:

$$fmerit_w = (1 + V_{fac}^{aver}) \times fmerit_w \qquad (2020)$$

$$fmerit_w^{sm} = 0.9 \times fmerit_w^{sm} + 0.1 \times fmerit_w \qquad (2021)$$

Set $fmerit_w$ to $fmerit_w^{sm}$.

If $fmerit$ is less than 0.5, set $fmerit_m$ to 1; Otherwise, set $fmerit_m$ to $(2 - fmerit)$. Then smooth $fmerit_m$ as follows:

$$fmerit_m^{sm} = 0.5 \times fmerit_m^{sm} + 0.5 \times fmerit_m \qquad (2022)$$

Set $fmerit_m$ to $fmerit_m^{sm}$.

6) If the sum of the differences between every two adjacent pitch values Sum_P^{diff} is less than 10 and the spectrum tilt factor of current subframe $Til(i)$ is less than zero, reset $Til(i)$ to 0.2. Then, if $Til(i)$ is greater than 0.2, reset $Til(i)$ to 0.8; Otherwise, reset $Til(i)$ to $(1 - Til(i))$. Finally, modify $Til(i)$ as follows:

$$Til(i) = (Til(i) + (30 + Ng_{\min}) \times 0.007) * fmerit_m \tag{2023}$$

6.8.3.1.2 Estimation of sub-frame gains based on LP spectral envelopes

The signal in low-band (0-64000 Hz) is generated based on a source-filter model, where the filter is given by the synthesis filter $1/\hat{A}(z)$. Similarly, as shown in subclause 6.8.3.3, the signal in high-band (above 6400 Hz) is generated based on a source-filter model; the filter in high-band is an linear predictive (LP) filter $A_{HB}(z) = \hat{A}(z/\gamma_{HB})$ derived from the LP filter in low-band.

Since the low and high-band are combined in the final synthesis, a preliminary equalization step is performed to match the levels of the two LP filters at a given frequency. At 6400 Hz the shape of $1/\hat{A}(z)$ is already too decreasing, therefore a frequency of 6000 Hz has been chosen for this equalization frequency point.

In each sub-frame, the frequency response of the LP filter $\hat{A}(z)$ in the low-band and the LP filter $A_{HB}(z) = \hat{A}(z/\gamma_{HB})$ in the high-band are computed at the frequency of 6000 Hz:

$$R = \frac{1}{\left|\hat{A}(e^{j\theta})\right|} = \frac{1}{\left|\sum_{i=0}^{M} \hat{a}_i e^{-ji\theta}\right|} , \theta = 2\pi \frac{6000}{12800} \tag{2024}$$

and

$$P = \frac{1}{\left|\hat{A}(e^{j\theta'}/\gamma_{HB})\right|} = \frac{1}{\left|\sum_{i=0}^{M} \hat{a}_i \gamma_{HB}^{i} e^{-ji\theta'}\right|} , \theta' = 2\pi \frac{6000}{16000} \tag{2025}$$

where γ_{HB} =0.9 at 6.6 kbit/s and 0.6 at other modes (from 8.85 to 23.85 kbit/s)

These values are computed efficiently using the following pseudo-code:

```
px = py = 0
rx = ry = 0
for i=0 to 16
    px = px + Ap[i]*exp_tab_p[i]
    py = py + Ap[i]*exp_tab_p[33-i]
    rx = rx + Aq[i]*exp_tab_q[i]
    ry = ry + Aq[i]*exp_tab_q[33-i]
end for
P = 1/sqrt(px*px+py*py)
R = 1/sqrt(rx*rx+ry*ry)
```

where Aq[i]= $\hat{a}_i$ are the coefficients of $\hat{A}(z)$, Ap[i]= $\gamma_{HB}^{i}\hat{a}_i$ are the coefficients of $\hat{A}(z/\gamma_{HB})$, sqrt() corresponds to the square root operation and the tables exp_tab_p and exp_tab_q of size 34 contain the real and imaginary parts of complex exponentials at 6000 Hz:

$$\text{exp\_tab\_p[i]} = \begin{cases} \cos\left(2\pi \frac{6000}{12800}i\right) & i = 0, \cdots, 16 \\ -\sin\left(2\pi \frac{6000}{12800}(33-i)\right) & i = 17, \cdots, 33 \end{cases} \tag{2026}$$

and

$$\text{exp\_tab\_q[i]} = \begin{cases} \cos\left(2\pi \dfrac{6000}{16000} i\right) & i = 0, \cdots, 16 \\ -\sin\left(2\pi \dfrac{6000}{16000}(33-i)\right) & i = 17, \cdots, 33 \end{cases} \tag{2027}$$

The ratio R/P provides an estimated gain to be used in each sub-frame to align at the given frequency point (6000 Hz) the level of LP spectral envelopes in two different bands. This value is further refined to optimize overall quality.

To avoid over-estimating the sub-frame gain in high-band which could result in too high enrgy in the high hand, an additional LP filter of lower order is also computed based on the lower-band LP filter. An LP filter of order 2 is derived by truncating the filter $\hat{A}(z)$ decoded in low band to an order of 2 (instead of an order of 16). The stability of this truncated filter is ensured by the following steps:

- The filter is initialized as: $\hat{a}_i' = \hat{a}_i$, i=1, 2

- Reflection coefficients are computed: $k_1 = \hat{a}_1'/(1 + \hat{a}_2')$, $k_2 = \hat{a}_2'$

- Filter stability and control of resonance is forced by applying the following conditions:

$$k_2 \leftarrow \begin{cases} \min(0.6, k_2) & k_2 > 0 \\ \max(-0.6, k_2) & k_2 < 0 \end{cases} \tag{2028}$$

$$k_1 \leftarrow \begin{cases} \min(0.99, k_2) & k_1 > 0 \\ \max(-0.99, k_2) & k_1 < 0 \end{cases} \tag{2029}$$

- The coefficients of the LP filter of order 2 are then given by: $\hat{a}_1' = (1 + k_2)k_1$, $\hat{a}_2' = k_2$

The frequency response of the resulting LP filter of order 2 is computed as follows:

$$Q = \frac{1}{\left|\displaystyle\sum_{k=0}^{2} \hat{a}_k' e^{-jk\theta}\right|}, \theta = 2\pi \frac{6000}{12800} \tag{2030}$$

which can be computed efficiently using a similar pseudo-code with tables exp_tab_p and exp_tab_q It was found that, for some signals, using the value Q instead of the value R takes better into account the influence of spectral tilt in the actual signal spectrum and therefore avoids the influence of spectral peaks or valleys near the reference frequency point (6000 Hz) which could bias the value R.

The optimized gain to shape the excitation in high-band is then estimated based on R, P, Q.

Before the gain is estimated, an unvoiced flag is determined first so that the gain estimation will be different for unoiced speech and voiced speech. An unvoicing parameter is defined as,

$$P_{c\_unvoicing\_tmp} = 0.5\,(1 - Til0(i)) \cdot (1 - P_{voicing}) \cdot MIN(Til(i)/1.5 - 1,\quad 1) \tag{2031}$$

wherein $P_{voicing}$ is a smoothed voicing parameter of V_{fac}^{aver}. The unvoicing parameter is first smoothed by

$$P_{c\_unvoicing} = 0.5 \, P_{c\_unvoicing} + 0.5 \, P_{c\_unvoicing\_tmp} \tag{2032}$$

Then, it is further smoothed by,

$$
\begin{aligned}
&if \, (P_{c\_unvoicing\_sm} > P_{c\_unvoicing}) \quad \{ \\
&\quad P_{c\_unvoicing\_sm} = 0.9 \, P_{c\_unvoicing\_sm} + 0.1 \, P_{c\_unvoicing} \\
&\} \\
&else \quad \{ \\
&\quad P_{c\_unvoicing\_sm} = 0.99 \, P_{c\_unvoicing\_sm} + 0.01 \, P_{c\_unvoicing} \\
&\}
\end{aligned}
\tag{2033}
$$

A relative difference parameter is now defined as

$$P_{c\_unvoicing\_diff} = P_{c\_unvoicing} - P_{c\_unvoicing\_sm} \tag{2034}$$

An initial unvoiced flag is decided by the following procedure,

$$
\begin{aligned}
&if \quad (P_{c\_unvoicing\_diff} \; > \; 0.1) \quad \{ \\
&\quad Unvoiced\_flag = TRUE; \\
&\} \\
&else \quad if \quad (P_{c\_unvoicing\_diff} \; < \; 0.05) \quad \{ \\
&\quad Unvoiced\_flag = FALSE; \\
&\} \\
&else \quad \{ \\
&\quad Unvoiced\_flag \; is \; not \; changed \; (previous \; Unvoiced\_flag \; is \; kept). \\
&\}
\end{aligned}
\tag{2035}
$$

A final unvoiced flag is limited to

$$Final\_Unvoiced\_flag \; = \; Unvoiced\_flag \quad AND \quad (R{>}P) \tag{2036}$$

The gain computed is performed according to the voicing of the signal:

If the sub-frame is classified as unvoiced

$$g_{sf}(i) = \min\big(5, \max\big(\min(R', Q'), P\big)/P\big) \tag{2037}$$

where the smoothed value $R^{(m)}$ in the current sub-frame of index m is computed as

$$R^{(i)} = \begin{cases} 0.5R + 0.5R^{(i-1)} & R > R^{(i-1)} \\ R & otherwise \end{cases} \tag{2038}$$

and

$$R' = R^{(i)} . \min\big(1, Til(i).\big(1.6 - V_{fac}(i)\big)\big) \tag{2039}$$

and

$$Q' = Q . \max\big(1, Til(i).\big(1.6 - V_{fac}(i)\big)\big) \tag{2040}$$

Otherwise, if the sub-frame is not classified as unvoiced:

$$g_{sf}(i) = \min\left(5, \min(lev_1, lev_2) / P\right) \tag{2041}$$

where the smoothed value $R^{(i)}$ in the current sub-frame of index i is computed as

$$R^{(i)} = (1 - \alpha)R + \alpha R^{(i-1)} \tag{2042}$$

with $\alpha = 1 - R^2$ if $R < 1$ and $R^{(m-1)} < 1$, $\alpha = 0$ otherwise, and

$$Q' = Q.\max\left(1, Til(i).\left(1.6 - V_{fac}(i)\right)\right) \tag{2043}$$

and where

$$lev_1 = Q.\min\left(1, Til(i).\left(1.6 - V_{fac}(i)\right)\right) \tag{2044}$$

and

$$lev_2 = \min\left(R^{(i)}, P, Q\right)\left(1 + \left|Til(i) - 1\right|.\left(1.6 - V_{fac}(i)\right)\right) \tag{2045}$$

6.8.3.2 Generation of high-band excitation

6.8.3.2.1 DCT

The current frame of decoded excitation from the low-band, $u(n)$, $n = 0, \cdots, 255$, sampled at 12.8 kHz, is transformed in DCT domain as described in sub-clause 5.2.3.5.3.1, to obtain the spectrum, $U(k)$, $k = 0, \cdots, 255$.

6.8.3.2.2 High band generation

6.8.3.2.2.1 Adaptive start frequency bin prediction

The start frequency bin of predicting the high band excitation from the low band excitation k_{start} is adaptively determined by the line spectrum frequency (LSF) parameters. The LSF parameters are decoded from the bitstream of low frequency band. Based on the decoded LSF parameters of the low band signal, the differences between every two adjacent LSF parameters are calculated and the minimum difference is searched since the minimum difference corresponds to an energy peak of the low band spectral envelope. The start frequency bin k_{start} is determined by the position of the minimum difference, where the low band excitation is decoded from the bitstream of the low band as described in subclause 6.8.1.1.

In order to mitigate switching the start frequency bin frequently in $VOICE\_CLAS$ or $AUDIO\_CLAS$, the voicing flag F_{voice} will be determined according to the average voice factor V_{fac}^{aver} and the FEC class of current frame $class_{FEC}$:

$$F_{voice} = \begin{cases} 1 & \text{if } V_{fac}^{aver} > 0.4 \text{ OR } \left(V_{fac}^{aver} > 0.3 \text{ AND class} \geq 3\right) \text{ OR } class_{FEC} = AUDIO\_CLAS \\ 0 & \text{otherwise} \end{cases} \tag{2046}$$

The voicing flag F_{voice} is further refined to 0 if $V_{fac}^{aver} < 0.2 \text{ AND } class_{FEC} < VOICE\_CLAS$.

Initially the start frequency bin k_{start} is 160. If the bitrate is not less than 23050, the start frequency bin $k_{start} = 160$; Otherwise, the start frequency bin k_{start} is adaptively searched as follows:

1) Calculate the LSF differences between every two adjacent LSF parameters:

$$d_{LSF}(k) = LSF(k) - LSF(k-1), \qquad k = 1, 2, \cdots, M - 1 \tag{2047}$$

where M is the order of the LP filter and $M = 16$.

2) Determine the range of search the minimum LSF difference in $d_{LSF}(k)$:

Initialize the range to $[2, M2]$, $M2 = M - 2$, if voicing flag $F_{voice} = 1$, reset $M2$:

$$M2 = \begin{cases} M - 8, & \text{if bitrate} \leq 8850 \\ M - 6, & \text{else if bitrate} \leq 12650 \\ M - 4, & \text{else if bitrate} \leq 15850 \end{cases} \tag{2048}$$

3) Search the minimum value V_{min} of the adjusted LSF difference $V_{cri}(k)$ in the range $[2, M2)$, $V_{cri}(k)$ is calculated as follows:

$$V_{cri}(k) = d_{LSF}(k) * \max(1 - LSF(k) * W, 0.001), \qquad k \in S \tag{2049}$$

and $V_{min} = \min_{k \in [2, M2)} (V_{cri}(k))$, the position k_{min} of the minimum value V_{min} is

$$k_{min} = \underset{k \in [2, M2)}{\operatorname{argmin}} (V_{cri}(k)) \tag{2050}$$

where W is adjust factor of LSF parameters based on the core bitrate and the FEC class of current frame:

$$W = \begin{cases} 0.75 \cdot \left(\dfrac{bitrate}{19850}\right)^2 \Big/ 6000 & \text{if } class_{FEC} = AUDIO\_CLAS \\ \left(\dfrac{bitrate}{19850}\right)^2 \Big/ 6000 & \text{otherwise} \end{cases} \tag{2051}$$

4) The start frequency bin of of predicting the high band excitation from the low band excitation is calculated:

$$k_{start} = \min\left(\max\left(\left\lfloor \frac{0.5 \cdot (LSF(k_{min}) + LSF(k_{min} - 1)) \cdot 40}{1000} \right\rfloor - 40, 40\right), 160\right) \tag{2052}$$

5) In order to decrease the distortion of the spectrum of the high band, the start frequency bin of the current frame k_{start} is reset with the start frequency bin of the previous frame $k_{start}^{[-1]}$ when the below conditions is satisfied:

– If one of the conditions $F_{voice}^{[-1]} \neq F_{voice}$, $F_{voice} = 0$ *AND* $V_{min} < V_{min}^{[-1]}$, or $V_{min} < 0.7 \cdot V_{min}^{[-1]}$ *AND* $V_{min}^{[-1]} > 64$ is satisfied, V_{min} of current frame is preserved for the next frame, and

$$k_{start} = k_{start}^{[-1]}, \qquad if \left| k_{start} - k_{start}^{[-1]} \right| < 20 \ AND \ F_{voice} = 1 \ AND \ F_{voice}^{[-1]} = 1 \tag{2053}$$

– Otherwise, the start frequency bin of bandwidth extension is set to $k_{start}^{[-1]}$, and the V_{min} of current frame is preserved for the next frame if $V_{min} < V_{min}^{[-1]}$ *AND* $F_{voice} = 1$.

The start frequency bin of predicting the high band excitation from the low band excitation is further refined if the FEC class of current frame is $AUDIO\_CLAS$:

$$k_{start} = \min(k_{start}, 120) \tag{2054}$$

If k_{start} is not an even number, k_{start} is decremented by one.

Then, obtain the high band excitation by choosing low band excitation with a given length of the bandwidth according to the start frequency bin k_{start} .

6.8.3.2.2.2 Extension of excitation spectrum

The DCT spectrum covering the 0-6400 Hz band is extended to the 0-8000 Hz band as follows:

$$U_{HB1}(k) = \begin{cases} 0 & k = 0, \cdots, 199 \\ U(k) & k = 200, \cdots, 239 \\ U(k + k_{start} - 240) & k = 240, \cdots, 319 \end{cases} \tag{2055}$$

where k_{start} is the adaptive start band as computed according to subclause 6.8.3.2.2.1. The 5000-6000 Hz band in $U_{HB1}(k)$ is copied from $U(k)$ in the same band, this allows keeping the original spectrum in this band to avoid introducing distortions when the high-band is added to the decoded low-band signal. The 6000-8000 Hz band in $U_{HB1}(k)$ is copied from $U(k)$ e.g. in the 4000-6000 Hz band when $k_{start} = 160$.

6.8.3.2.3 Extraction of tonal and ambiance components

Tonal and ambiance components are extracted in the 6000-8000 Hz. This extraction is implemented according to the following steps:

- Computation of total energy $ener_{HB}$ in the extended low-band signal:

$$ener_{HB} = \sum_{k=240}^{319} U_{HB1}(k)^2 + \varepsilon \tag{2056}$$

where $\varepsilon = 0.1$.

- Computation of the ambiance component (in absolute value) corresponding to the average (bin-by-bin) level of the spectrum $lev(i)$ and computation of the energy $ener_{tonal}$ of dominant tonal components in high frequency:

The average level is given by the following equation:

$$lev(i) = \frac{1}{fn(i) - fb(i) + 1} \sum_{j=fb(i)}^{fn(i)} \left| U_{HB1}(j + 240) \right|, \ i = 0...L-1 \tag{2057}$$

where $L = 80$. This level gives an average level in absolute value and represents a sort of spectral envelope. Note that the index $i = 0...L-1$ corresponds to indices $j + 240$ from 240 to 319, i.e. the 6000-8000 Hz band. In general, $fb(i) = i - 7$ and $fn(i) = i + 7$, however for the first and last 7 indices ($i = 0, \cdots, 6$ et $i = L-7, \cdots, L-1$) the following values are used:

$fb(i) = 0$ and $fn(i) = i + 7$ for $i = 0, \cdots, 6$

$fb(i) = i - 7$ and $fn(i) = L-1$ for $i = L-7, \cdots, L-1$

- Detection and computation of the residual signal which defines tonal components:

$$y(i) = \left| U_{HB1}(i + 240) \right| - lev(i), \ i = 0...L-1 \tag{2058}$$

Tonal components are detected using the criterion $y(i) > 0$.

- Computation of the energy $ener_{tonal}$ of dominant tonal components in high frequency:

The energy of tonal components is computed as follows:

$$ener_{tonal} = \sum_{i=0...7|y(i)|>0} y(i)^2 \; , \; i = 0...L-1 \tag{2059}$$

6.8.3.2.4 Recombination

The extracted tonal and ambiance components are re-mixed adaptively. The combined signal is obtained using absolute values as:

$$y'(i) = \begin{cases} \Gamma y(i) + \dfrac{1}{\Gamma} lev(i) & y(i) > 0 \\ y(i) + \dfrac{1}{\Gamma} lev(i) & y(i) \le 0 \end{cases} , \; i = 0...L-1 \tag{2060}$$

where the factor controlling the ambiance

$$\Gamma = \beta \frac{ener_{HB} - ener_{tonal}}{ener_{HB} - \beta ener_{tonal}} \tag{2061}$$

and β is a multiplicative factor given by:.

$$\beta = 1 - fmerit_m \tag{2062}$$

Tonal components, that were detected using the criterion $y(i) > 0$, are reduced by a factor Γ and the average level is amplified by $1/\Gamma$.

Signs from $U_{HB1}(k)$ are then applied as follows:

$$y''(i) = \mathrm{sgn}\big(U_{HB1}(i+240)\big).y'(i) \; , \; i = 0...L-1 \tag{2063}$$

where

$$\mathrm{sgn}(x) = \begin{cases} 1 & x \ge 0 \\ -1 & x < 0 \end{cases} \tag{2064}$$

The combined high-band signal $U_{HB2}(k)$ is then obtained by adjusting the energy as follows:

$$U_{HB2}(k) = fac.y''(k-240) \; , \; k = 240, \cdots, 319 \tag{2065}$$

where the adjustment factor is given by:

$$fac = \gamma \sqrt{\frac{ener_{HB}}{\displaystyle\sum_{i=0}^{L-1} y''(i)}} \tag{2066}$$

The factor γ is used to avoid over-estimation of energy and is given by:

$$\gamma = \max\big(0.3, \min\big(1, 1.05 - 0.95\alpha\big)\big) \tag{2067}$$

and

$$\alpha = \sqrt{fmerit_w.\big(1.1 - 0.00625.\text{start\_band}\big)} \tag{2068}$$

6.8.3.2.5 Filtering in DCT domain

The excitation is de-emphasized as follows:

$$U_{HB2}'(k) = \begin{cases} 0 & k = 0, \cdots, 199 \\ G_{deemph}(k-200)U_{HB2}(k) & k = 200, \cdots, 255 \\ G_{deemph}(55)U_{HB2}(k) & k = 256, \cdots, 319 \end{cases} \tag{2069}$$

where $G_{deemph}(k)$ is the frequency responses of the filter $1/\left(1 - 0.68z^{-1}\right)$ over a limited frequency range. Taking into account the (odd) frequencies of the DCT, $G_{deemph}(k)$ is given by:

$$G_{deemph}(k) = \frac{1}{\left| e^{j\theta_k} - 0.68 \right|}, \quad k = 0, \cdots, 255 \tag{2070}$$

where

$$\theta_k = \frac{256 - 80 + k + \frac{1}{2}}{256}, \quad k = 0, \cdots, 255 \tag{2071}$$

The de-emphasis is applied in two steps, for $k = 200, \cdots, 255$ where the response of $1/\left(1 - 0.68z^{-1}\right)$ is applied in the 5000-6400 Hz band, and for $k = 256, \cdots, 319$ corresponding to the 6400-8000 Hz band. This de-emphasis is used to bring the signal in a domain consistent with the low-band signal (in the 0-6.4 band), which is useful for the subsequent energy estimation and adjustment.

Then, the high-band is bandpass filtered in DCT domain, by splitting fixed high-pass filtering and adaptive low-pass filtering. The partial response of the low-pass filter in DCT domain is computed as follows:

$$G_{lp}(k) = 1 - 0.999\frac{k}{N_{lp} - 1}, \quad k = 0, \cdots, 255 \tag{2072}$$

where $N_{lp} = 60$ at 6.6 kbit/s, 40 at 8.85 kbit/s, and 20 for modes >8.85 bit/s. It defines a low-pass filter with variable cut-off frequency, depending on the mode in the current frame. Then, the band-pass filter is applied in the following form:

$$U_{HB3}(k) = \begin{cases} 0 & k = 0, \cdots, 199 \\ G_{hp}(k-200)U_{HB2}'(k) & k = 200, \cdots, 255 \\ U_{HB2}'(k) & k = 256, \cdots, 319 - N_{lp} \\ G_{lp}(k - 320 - N_{lp})U_{HB2}'(k) & k = 320 - N_{lp}, \cdots, 319 \end{cases} \tag{2073}$$

The definition of the factor $G_{hp}(k)$, $k = 0, \cdots, 55$ is given in table 174.

Table 174: High-pass filter in DCT domain

k	$G_{hp}(k)$	k	$G_{hp}(k)$	k	$G_{hp}(k)$	k	$G_{hp}(k)$
0	0.001622428	14	0.114057967	28	0.403990611	42	0.776551214
1	0.004717458	15	0.128865425	29	0.430149896	43	0.800503267
2	0.008410494	16	0.144662643	30	0.456722014	44	0.823611104
3	0.012747280	17	0.161445005	31	0.483628433	45	0.845788355
4	0.017772424	18	0.179202219	32	0.510787115	46	0.866951597
5	0.023528982	19	0.197918220	33	0.538112915	47	0.887020781
6	0.030058032	20	0.217571104	34	0.565518011	48	0.905919644
7	0.037398264	21	0.238133114	35	0.592912340	49	0.923576092
8	0.045585564	22	0.259570657	36	0.620204057	50	0.939922577
9	0.054652620	23	0.281844373	37	0.647300005	51	0.954896429
10	0.064628539	24	0.304909235	38	0.674106188	52	0.968440179
11	0.075538482	25	0.328714699	39	0.700528260	53	0.980501849
12	0.087403328	26	0.353204886	40	0.726472003	54	0.991035206
13	0.100239356	27	0.378318805	41	0.751843820	55	1.000000000

6.8.3.2.6 Inverse DCT

The current frame of extended excitation in high-band, $U_{HB3}(k)$, $k = 0, \cdots, 320$, sampled at 16 kHz, is transformed in time domain as described in subclause 5.2.3.5.13, to obtain the signal $u_{HB}(n)$, $n = 0, \cdots, 320$.

6.8.3.2.7 Gain computation and scaling of excitation

6.8.3.2.7.1 6.6, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85 or 23.05 kbit/s modes

The signal $u_{HB}(n)$, $n = 0, \cdots, 320$ is scaled by sub-frame of 5 ms as follows:

$$u_{HB}{}'(n) = g_{HB1}(i) u_{HB}(n) \text{ , } n = 80i, \cdots, 80(i+1)-1 \tag{2074}$$

where $m = 0,1,2,3$ is the sub-frame index and

$$g_{HB1}(i) = \sqrt{\frac{e_3(i)}{e_2(i)}} \tag{2075}$$

with

$$e_1(i) = \sum_{n=0}^{63} u(n+64i)^2 + \varepsilon$$

$$e_2(i) = \sum_{n=0}^{79} u_{HB}(n+80i)^2 + \varepsilon \tag{2076}$$

$$e_3(i) = e_1(i) \frac{\sum_{n=0}^{319} u_{HB}(n)^2 + \varepsilon}{\sum_{n=0}^{255} u(n)^2 + \varepsilon}$$

and $\varepsilon = 0.01$. The sub-frame gain $g_{HB1}(m)$ can be further written as:

$$g_{HB1}(i) = \sqrt{\dfrac{\dfrac{\sum\limits_{n=0}^{63} u(n+64i)^2 + \varepsilon}{\sum\limits_{n=0}^{255} u(n)^2 + \varepsilon}}{\dfrac{\sum\limits_{n=0}^{79} u_{HB}(n+80i)^2 + \varepsilon}{\sum\limits_{n=0}^{319} u_{HB}(n)^2 + \varepsilon}}} \qquad (2077)$$

which shows that this gain is used to have in $u_{HB}'(n)$ the same ratio of sub-frame vs frame energy than in the low-band

signal $u(n)$.

6.8.3.2.7.2   23.85 kbit/s mode

In the 23.85 kbit/s mode, a high-frequency (HF) gain is transmitted at a bit rate of 0.8 kbit/s (4 bits per 5 ms sub-frame). This information is transmitted only at 23.85 kbit/s and it used in EVS AMR-WB IO to improve quality by adjusting the excitation gain.

To be able to use the HF gain information, the excitation has to be converted to a signal domain similar to AMR-WB high-band coding. To do so the energy of the excitation is adjusted in each subframe as follows:

$$u_{HB1}(n) = g_{HB2}(i)u_{HB}'(n) \text{ , } n = 80m, \cdots, 80(i+1)-1 \qquad (2078)$$

where the sub-frame gain $g_{HB2}(i)$ is computed as:

$$g_{HB2}(i) = \sqrt{\dfrac{\sum\limits_{n=0}^{63} u(n+64i)^2}{5.\sum\limits_{n=0}^{79} u_{HB}'(n+80i)^2}} \qquad (2079)$$

The factor 5 in the the denominator is used to compensate the difference in bandwith between the signal $u(n)$ and the signal $u_{HB}'(n)$, noting that in AMR-WB the HF excitation is a white noise in the 0-8000 Hz band.

The 4-bit index in each sub-frame, $\text{index}_{HF\_gain}(m)$, transmitted at 23.85 kbit/s is demultiplexed from the bitstream and decoded as follows:

$$g_{HBcorr}(i) = 2.HP\_\text{gain}(\text{index}_{HF\_gain}(i)) \qquad (2080)$$

where $HP\_\text{gain}(.)$ is the codebook used for HG gain quantization in AMR-WB, as defined in table 175.

Table 175: AMR-WB gain codebook for high band

j	$HP\_\text{gain}(j)$	j	$HP\_\text{gain}(j)$
0	0.110595703125000	8	0.342102050781250
1	0.142608642578125	9	0.372497558593750
2	0.170806884765625	10	0.408660888671875
3	0.197723388671875	11	0.453002929687500
4	0.226593017578125	12	0.511779785156250
5	0.255676269531250	13	0.599822998046875f
6	0.284545898437500	14	0.741241455078125
7	0.313232421875000	15	0.998779296875000

Then, the signal $u_{HB1}(n)$ is scaled according to this decoded HF gain as follows:

$$u_{HB2}(n) = g_{HBcorr}(i)u_{HB1}(n), \quad n = 80i, \cdots, 80(i+1) - 1 \tag{2081}$$

The energy of the excitation is further adjusted by sub-frame under the following conditions. A factor $fac(m)$ is computed:

$$fac(i) = \sqrt{\frac{\sum_{n=0}^{79}\left(0.6 g_{sf}(i)u_{HB}'(n+80i)\right)^2}{\sum_{n=0}^{79}u_{HB2}(n+80i)^2}} \tag{2082}$$

Here the term 0.6 corresponds to the average magnitude ratio between the frequency response of the de-emphasis filter $1/\left(1 - 0.68z^{-1}\right)$ in the 5000-6400 Hz band. Therefore, the term $\sum_{n=0}^{79}\left(0.6 g_{sf}(i)u_{HB}'(n)\right)^2$ represents the energy of the high-band excitation that would be obtained at 23.05 kbit/s.

Based on the tilt information of the low-band signal, the excitation is then computed for $n = 80i, \cdots, 80(i+1) - 1$ as follows:

If $fac(i) > 1$ or $Til0(i) < 0$:

$$u_{HB}''(n) = u_{HB2}(n) \tag{2083}$$

Otherwise:

$$u_{HB}''(n) = \max\left(\min\left(1, (1 - Til0(i)).(1.6 - V_{fac}(m))\right), fac(i)\right)u_{HB2}(n) \tag{2084}$$

6.8.3.3   LP filter for the high frequency band

Predict the high-band LP synthesis filter $A_{HB}(z)$ using the weighted low-band LP synthesis filter

$$A_{HB}(z) = \hat{A}(z / \gamma_{HB}) \tag{2085}$$

where $\hat{A}(z)$ is the interpolated LP synthesis filter in each 5-ms sub-frame and $\gamma_{HB} = 0.9$ at 6.6 kbit/s and 0.6 at other modes (from 8.85 to 23.85 kbit/s). $\hat{A}(z)$ has been computed analysing signal with the sampling rate of 12.8 kHz but it is now used for a 16 kHz signal.

6.8.3.4   High band synthesis

The excitation in high-band is filtered by $1/A_{HB}(z)$ to obtain the decoded high-band signal, which is added to synthesized low band signal to produce the synthesized output signal.

6.8.4   CNG decoding

The CNG decoding in AMR-WB-interoperable mode is described by referring to subclause 6.7.2. The CNG parameter updates in active and inactive periods is the same as described in subclasue 6.7.2.1.1. The DTX-hangover based parameter analysis is the same as described in subclause 6.7.2.1.2. The quantized logarithmic excitation energy is found from the SID frame using $\Delta = 2.625$ and converted to linear domain using the procedure described in subclause 5.6.2.1.5. The quantized energy $\hat{\bar{E}}$ is used to obtain the smoothed quantized excitation energy E_{CN} used for CNG synthesis in the same way as described in subclause 5.6.2.1.6. The quantized ISF vector is found in the same way as described in subclause 5.7.12. The smoothed LP synthesis filter, $\hat{A}(Z)$, is then obtained in the same way as described in subclause 5.6.2.1.4 with the only difference that the ISP vector is used instead of the LSP vector. The CNG excitation

signal, $e(n)$, $n = 0,...,L$, where L is the frame length, is generated in the same way the random excitation signal $e_r(n)$, $n = 0,...,L$ is generated as described in subclause 6.7.2.1.5. The comfort noise is synthesized by filtering the excitation signal, $e(n)$, through the smoothed LP synthesis filter, $\hat{A}(Z)$.

6.9 Common post-processing

6.9.1 Comfort noise addition

In this clause, we describe a post-processing technique for enhancing the quality of noisy speech coded and transmitted at bit-rates up to 13.2 kbps. At such low bit-rates, the coding of noisy speech, i.e. speech recorded with background noise, is usually not as efficient as the coding of clean speech. The decoded synthesis is usually prone to artifacts as the two different kinds of sources - the noise and the speech - cannot be efficiently coded by a coding scheme relying on a single-source model.

The comfort noise addition (CNA) consists in modelling and synthesizing the background noise at the decoder side, requiring thereby no side-information. It is achieved by estimating the level and spectral shape of the background noise at the decoder side, and by generating artificially a comfort noise in the frequency domain. In principle, the noise estimation and generation in CNA is therefore similar to the FD-CNG presented in clause 6.7.3. However, a noticeable difference is that FD-CNG is applied in DTX operations only, whereas CNA can be used in any case when coding noisy speech at bit-rates up to 13.2 kbps. The generated noise is added to the decoded audio signal and allows masking coding artifacts.

6.9.1.1 Noisy speech detection

The CNA should be triggered in noisy speech scenarios only, i.e., not in clean speech or clean music situations. To this end, a noisy speech detector is used in the decoder. It consists in estimating the long-term SNR by separately adapting long-term estimates of either the noise or the speech/music energies, depending on a VAD decision f_{VAD}.

The VAD decision is deduced directly from the information decoded from the bitstream. It is 0 if the current frame is a SID frame, a zero frame, or an IC (Inactive Coding mode, see clause 5.1.13) frame. It is 1 otherwise.

The long-term noise estimate $\overline{N}_{\text{NSD}}$ and long-term speech/music estimate $\overline{S}_{\text{NSD}}$ are initialized with -20 dB and +25 dB, respectively. When $f_{\text{VAD}} = 0$, the long-term noise energy is updated on a frame-by-frame basis as follows:

$$\overline{N}_{\text{NSD}} = 0.995\,\overline{N}_{\text{NSD}} + 0.005 \cdot 10\log_{10}\left(\sum_{i=0}^{L_{\text{shaping}}-1} N_{\text{FD-CNG}}^{[\text{shaping}]}(i)\left(j_{\max}^{[\text{shaping}]}(i) - j_{\min}^{[\text{shaping}]}(i) + 1\right)\right), \quad (2086)$$

where $N_{\text{FD-CNG}}^{[\text{shaping}]}(i)$ refers to the noise energy spectrum estimated in the decoder to apply FD-CNG, L_{shaping} is the number of spectral partitions, and $j_{\max}^{[\text{shaping}]}(i) - j_{\min}^{[\text{shaping}]}(i) + 1$ corresponds to the size of each partition (see clause 6.7.3.2.2). Otherwise, i.e. if $f_{\text{VAD}} = 1$, the long-term speech/music energy is updated on a frame-by-frame basis as follows:

$$\overline{S}_{\text{NSD}} = 0.995\,\overline{S}_{\text{NSD}} + 0.005 \times 10\log_{10}\left(\frac{2}{L_{\text{celp}}}\sum_{n=0}^{L_{\text{celp}}-1}\left(s_{\text{celp}}(n)\right)^2\right), \quad (2087)$$

where L_{celp} denotes the frame size in samples and $s_{\text{celp}}(n)$ is the output frame of the core decoder at the CELP sampling rate. Furthermore, the long-term noise estimate $\overline{N}_{\text{NSD}}$ is lower limited by $\overline{S}_{\text{NSD}} - 45$ for each frame.

The flag for noisy speech detection is set to 1 if the SNR is smaller than 28dB, i.e.

$$f_{\text{NSD}} = \begin{cases} 1 & \text{if } \overline{S}_{\text{NSD}} - \overline{N}_{\text{NSD}} < 28 \\ 0 & \text{otherwise} \end{cases}. \quad (2088)$$

6.9.1.2 Noise estimation for CNA

To be able to produce an artificial noise resembling the actual input background noise in terms of spectro-temporal characteristics, the CNA needs an estimate of the noise spectrum in each FFT bin.

6.9.1.2.1 CNA noise estimation in DTX-on mode when FD-CNG is triggered

In DTX-on mode and provided that FD-CNG is triggered, the FD-CNG noise levels $N_{FD-CNG}^{[CNG]}(j), j = 0,..., j_{max}^{[SID]}(L_{SID}^{[FFT]}-1)-1$ can be directly used. As described in clause 6.7.3, they are obtained by capturing the fine spectral structure of the background noise present during active phases, while updating only the spectral envelop of the noise during inactive parts with the help of the SID information.

6.9.1.2.2 CNA noise estimation in DTX-on mode when LP-CNG is triggered

To enable tracking of the noise spectrum when LP-CNG is triggered in DTX-on mode, the FD-CNG noise estimation algorithm (see clause 6.7.3.2.2) is applied at the output of the LP-CNG during inactive frames, yielding noise estimates $N_{FD-CNG}^{[shaping]}(i)$ in each spectral partition $i = 0,..., L_{shaping}-1$. Following the technique described in clause 6.7.3.2.3.1., the parameters $N_{FD-CNG}^{[shaping]}(i)$ are then interpolated to yield the full-resolution FFT power spectrum $N_{FD-CNG}^{[shaping,FR]}(j)$, which overwrites the current FD-CNG levels, i.e. $N_{FD-CNG}^{[CNG]}(j) = N_{FD-CNG}^{[shaping,FR]}(j)$.

6.9.1.2.3 CNA noise estimation in DTX-off mode

In DTX-off mode, the noise estimates $N_{FD-CNG}^{[shaping]}(i)$ are obtained by applying the FD-CNG noise estimation algorithm at the output of the core decoder when $f_{VAD} = 0$ only, i.e. during speech pauses. As in the previous clause, the interpolation techniques described in clause 6.7.3.2.3.1 is then used to obtain a full-resolution FFT power spectrum $N_{FD-CNG}^{[shaping,FR]}(j)$, which overwrites the current FD-CNG levels, i.e. $N_{FD-CNG}^{[CNG]}(j) = N_{FD-CNG}^{[shaping,FR]}(j)$.

6.9.1.3 Noise generation in the FFT domain and addition in the time domain

In CNA and when the current frame is not a MDCT-based TCX frame, a random noise is generated in the FFT domain, separately for the real and imaginary parts. This is the same approach as in the FD-CNG (see clause 6.7.3.3.2). The noise is then added to the decoder output after performing an inverse FFT transform of the random noise using the overlap-add method.

The level of added comfort noise should be limited to preserve intelligibility and quality. The comfort noise is hence scaled to reach a pre-determined target noise level. Typically, the decoded audio signal exhibits a higher SNR than the original input signal, especially at low bit-rates where the coding artifacts are the most severe. This attenuation of the noise level in speech coding is coming from the source model paradigm which expects to have speech as input. Otherwise, the source model coding is not entirely appropriate and won't be able to reproduce the whole energy of no-speech components. Hence, the amount of additional comfort noise is adjusted to roughly compensate for the noise attenuation inherently introduced by the coding process. The assumed amount of noise attenuation g_{CNA} is chosen depending on the bandwidth and the bit-rate, as shown in the tables below.

Table 176: Assumed noise attenuation level for EVS primary modes

Bandwidth	NB				WB				SWB	
Bit-rates [kbps]	< 8	8	9.6	13.2	< 8	8	9.6	13.2	≤ 9.6	13.2
g_{CNA} [dB]	-3.5	-3	-2.5	-2	-3	-2.5	-1.5	-2.5	-2	-1

Table 177: Assumed noise attenuation level for EVS primary modes for AMR-WB IO modes

Bandwidth	AMR-WB IO	
Bit-rates [kbps]	6.60	8.85
g_{CNA} [dB]	-4	-3

The energy $N_{\text{CNA}}(j)$ of the random noise is adjusted for each FFT bin j as

$$N_{\text{CNA}}(j) = \ell_{\text{NSD}} \cdot \left(10^{-g_{\text{CNA}}/10} - 1 \right) \cdot N_{\text{FD-CNG}}^{[\text{CNG}]}(j),$$ (2089)

where

$$\ell_{\text{NSD}} = 0.99\,\ell_{\text{NSD}} + 0.01 f_{\text{NSD}}$$ (2090)

can be interpreted as the likelihood of being in a noisy speech situation. It is used as a soft decision to reject clean speech or music situations where the noisy speech detection flag f_{NSD} becomes zero (see clause 6.9.1.1).

6.9.1.4 Noise generation and addition in the MDCT domain

If the current frame is an MDCT based TCX frame, the comfort noise addition is performed directly in the MDCT domain. The random noise adjustment for each MDCT bin j is derived from the FFT-based comfort noise adjustment:

$$N_{CNA,MDCT}(j) = N_{CNA}(j) \cdot \sqrt{160}.$$ (2091)

The adjusted random noise is subsequently added to the MDCT bins as last steps before doing the inverse transformation to time-domain:

$$X(j) = X(j) + N_{CNA,MDCT}(j).$$ (2092)

6.9.2 Long term prediction processing

For the TCX coding mode and bitrates up to 48kbps, LTP post filtering is applied to the output signal, using the LTP parameters transmitted in the bitstream.

6.9.2.1 Decoding LTP parameters

If LTP is active, integer pitch lag d_{LTP}, fractional pitch lag f_{LTP} and gain g_{LTP} are decoded from the transmitted indices $I_{LTP,lag}$ and $I_{LTP,gain}$:

$$d_{LTP} = \begin{cases} p_{\min} + \left\lfloor \dfrac{I_{LTP,lag}}{p_{res}} \right\rfloor & , \; if \; I_{LTP,lag} < \left(p_{fr2} - p_{\min}\right)p_{res} \\[3mm] p_{fr2} + \left\lfloor \dfrac{I_{LTP,lag} - \left(p_{fr2} - p_{\min}\right)p_{res}}{p_{res}/2} \right\rfloor & , \; if \; 0 \le \left(I_{LTP,lag} - \left(p_{fr2} - p_{\min}\right)p_{res}\right) < \left(p_{fr1} - p_{fr2}\right)\dfrac{p_{res}}{2} \\[3mm] p_{fr1} + I_{LTP,lag} - \left(p_{fr2} - p_{\min}\right)p_{res} - \left(p_{fr1} - p_{fr2}\right)\dfrac{p_{res}}{2} & , \; if \; I_{LTP,lag} \ge \left(p_{fr1} - p_{fr2}\right)\dfrac{p_{res}}{2} + \left(p_{fr2} - p_{\min}\right)p_{res} \end{cases} \tag{2093}$$

$$f_{LTP} = \begin{cases} I_{LTP,lag} - \left(d_{LTP} - p_{\min}\right)p_{res} & , \; if \; I_{LTP,lag} < \left(p_{fr2} - p_{\min}\right)p_{res} \\[3mm] 2\left(I_{LTP,lag} - \left(p_{fr2} - p_{\min}\right)p_{res} - \left(d_{LTP} - p_{fr2}\right)\dfrac{p_{res}}{2}\right) & , \; if \; 0 \le \left(I_{LTP,lag} - \left(p_{fr2} - p_{\min}\right)p_{res}\right) < \left(p_{fr1} - p_{fr2}\right)\dfrac{p_{res}}{2} \\[3mm] 0 & , \; if \; I_{LTP,lag} \ge \left(p_{fr1} - p_{fr2}\right)\dfrac{p_{res}}{2} + \left(p_{fr2} - p_{\min}\right)p_{res} \end{cases}$$

$$g_{LTP} = 0.15625\left(I_{LTP,gain} + 1\right) \tag{2094}$$

If LTP is not active, LTP parameters are set as follows:

$$if \; \left(LTP_{on} = 0\right) then$$
$$d_{LTP} = p_{\max}$$
$$f_{LTP} = 0$$
$$g_{LTP} = 0$$

On encoder side the pitch lag is computed on the LTP sampling rate, therefore it has to be converted to the output sampling rate first:

$$t = \left\lfloor \frac{\left(d_{LTP}\, p_{res} + f_{LTP}\right)L_{out} + L_{out}/2}{N_{LTP}} \right\rfloor$$

$$d'_{LTP} = \left\lfloor \frac{t}{p_{res}} \right\rfloor \quad , \quad f'_{LTP} = t \bmod p_{res} \tag{2095}$$

For 48kbps bitrate the LTP gain is reduced as follows:

$$g_{LTP} = \begin{cases} 0.32\, g_{LTP} & , \; if \; \left(bitrate = 48000\right) \wedge \left(N_{LTP} = 320\right) \\ 0.4\, g_{LTP} & , \; if \; \left(bitrate = 48000\right) \wedge \left(N_{LTP} = 512\right) \\ 0.64\, g_{LTP} & , \; else \end{cases} \tag{2096}$$

6.9.2.2 LTP post filtering

For long-term prediction with fractional pitch lags polyphase FIR interpolation filters are used to interpolate between past synthesis samples. For each combination of LTP sampling rate and output sampling rate a different set of filter coefficients is used. The index i_{filt} of the interpolation filter to use is determined according to the following table:

Table 178: LTP index i_{filt} **of the interpolation filter**

	$sr_{LTP} = 12800$	$sr_{LTP} = 16000$	$sr_{LTP} = 25600$
$sr_{out} = 8000$	0	4	8
$sr_{out} = 16000$	1	5	9
$sr_{out} = 32000$	2	6	10
$sr_{out} = 48000$	3	7	11

The predicted signal s_{pred} is computed by filtering the past synthesis signal with the selected FIR filter. The filtered range of the past synthesis signal is determined by the integer part of the pitch lag d'_{LTP}. The polyphase index of the filter is determined by the fractional part of the pitch lag f'_{LTP}.

The filtered signal s_{filt} is computed by low-pass filtering the current synthesis signal with polyphase index 0 of the selected interpolation filter, so that its frequency response matches the one of the predicted signal.

Both s_{pred} and s_{filt} are multiplied with the LTP gain g_{LTP}. The filtered signal is then subtracted from the synthesis signal , the predicted signal is added to it.

If both LTP gain and pitch lag are the same as in the previous frame, the full frame can be processed the same way:

$$s_{pred}(n) = \sum_{i=0}^{l_{filt}-1} \left(h_{LTP}^{(i_{filt}, f'_{LTP})}(p_{res}i)s_{LTP}(n - d'_{LTP} + i) + h_{LTP}^{(i_{filt}, p_{res} - f'_{LTP})}(p_{res}i)s_{LTP}(n - d'_{LTP} - 1 - i) \right)$$

$$s_{filt}(n) = \sum_{i=0}^{l_{filt}-1} \left(h_{LTP}^{(i_{filt}, 0)}(p_{res}i)s(n + i) + h_{LTP}^{(i_{filt}, p_{res})}(p_{res}i)s(n - 1 - i) \right) \qquad (2097)$$

$$s_{LTP}(n) = s(n) + g_{LTP}(s_{pred}(n) - s_{filt}(n)) \qquad , \quad n = 0...L_{out} - 1$$

However, if gain and/or pitch lag have changed compared to the previous frame, a 5ms transition is used to smooth the parameter change. If no delay compensation is needed, the transition starts at the beginning of the frame. If a delay of D_{LTP} needs to be compensated, the transition starts at offset D_{LTP} from the beginning of the frame. In that case the signal part before the transition is processed using the LTP parameters of the previous frame:

$$s_{pred}(n) = \sum_{i=0}^{l_{filt}-1} \left(h_{LTP}^{(i_{filt}^{(prev)}, f'_{LTP}{}^{(prev)})}(p_{res}i)s_{LTP}(n - d'_{LTP}{}^{(prev)} + i) + h_{LTP}^{(i_{filt}^{(prev)}, p_{res} - f'_{LTP}{}^{(prev)})}(p_{res}i)s_{LTP}(n - d'_{LTP} - 1 - i) \right)$$

$$s_{filt}(n) = \sum_{i=0}^{l_{filt}-1} \left(h_{LTP}^{(i_{filt}^{(prev)}, 0)}(p_{res}i)s(n + i) + h_{LTP}^{(i_{filt}^{(prev)}, p_{res})}(p_{res}i)s(n - 1 - i) \right) \qquad (2098)$$

$$s_{LTP}(n) = s(n) + g_{LTP}^{(prev)}(s_{pred}(n) - s_{filt}(n)) \qquad , \quad n = 0...D_{LTP} - 1$$

If the LTP gain of the previous frame is zero (i.e. LTP was inactive in the previous frame), a linear fade-in is used for the gain in the transition region:

$$s_{LTP}(n) = s(n) + \frac{4(n - D_{LTP})}{L_{out}} g_{LTP}(s_{pred}(n) - s_{filt}(n)) \qquad , \quad n = D_{LTP}...D_{LTP} + \frac{L_{out}}{4} - 1 \qquad (2099)$$

If the LTP gain of the current frame is zero (LTP is inactive, but was active in the previous frame), a linear fade-out is used for the gain in the transition region, using the LTP parameters of the previous frame:

$$s_{pred}(n) = \sum_{i=0}^{l_{filt}-1}\left(h_{LTP}^{\left(i_{filt}^{(prev)},f_{LTP}'^{(prev)}\right)}(p_{res}i)s_{LTP}\left(n-d_{LTP}'^{(prev)}+i\right)+h_{LTP}^{\left(i_{filt}^{(prev)},p_{res}-f_{LTP}'^{(prev)}\right)}(p_{res}i)s_{LTP}\left(n-d_{LTP}'-1-i\right)\right)$$

$$s_{filt}(n) = \sum_{i=0}^{l_{filt}-1}\left(h_{LTP}^{\left(i_{filt}^{(prev)},0\right)}(p_{res}i)s(n+i)+h_{LTP}^{\left(i_{filt}^{(prev)},p_{res}\right)}(p_{res}i)s(n-1-i)\right) \qquad (2100)$$

$$s_{LTP}(n) = s(n)+\frac{L_{out}-4(n-D_{LTP})}{L_{out}}g_{LTP}^{(prev)}\left(s_{pred}(n)-s_{filt}(n)\right) \qquad, \quad n=D_{LTP}...D_{LTP}+\frac{L_{out}}{4}-1$$

If LTP is active in previous and current frame and LTP parameters have changed, a zero input response z is used to smooth the transition.

The LPC coefficients for zero input LP filtering are computed from the past 20ms LTP output before the beginning of the transition, using autocorrelation and Levinson-Durbin algorithm as described in 5.1.9.

$$z_{pred}(n) = \sum_{i=0}^{l_{filt}-1}\left(h_{LTP}^{\left(i_{filt},f_{LTP}'\right)}(p_{res}i)s_{LTP}\left(n-d_{LTP}'+i\right)+h_{LTP}^{\left(i_{filt},p_{res}-f_{LTP}'\right)}(p_{res}i)s_{LTP}\left(n-d_{LTP}'-1-i\right)\right)$$

$$z_{filt}(n) = \sum_{i=0}^{l_{filt}-1}\left(h_{LTP}^{\left(i_{filt},0\right)}(p_{res}i)s(n+i)+h_{LTP}^{\left(i_{filt},p_{res}\right)}(p_{res}i)s(n-1-i)\right) \qquad (2101)$$

$$z(n) = \left(s(n)-g_{LTP}s_{filt}(n)\right)-\left(s_{LTP}(n)-g_{LTP}s_{pred}(n)\right) \qquad, \quad n=D_{LTP}-m...D_{LTP}-1$$

The zero input response is then computed by LP synthesis filtering with zero input, and applying a linear fade-out to the second half of the transition region:

$$z(n) = -\min\left(\frac{2L_{out}-8(n-D_{LTP})}{L_{out}},1\right)\sum_{j=1}^{m}a(j)z(n-j) \quad, \quad n=D_{LTP}...D_{LTP}+\frac{L_{out}}{4}-1 \qquad (2102)$$

Finally the output signal in the transition region is computed by LTP filtering using the current frame parameters and subtracting the zero input response:

$$s_{LTP}(n) = s(n)+g_{LTP}\left(s_{pred}(n)-s_{filt}(n)\right)-z(n) \qquad, \quad n=D_{LTP}...D_{LTP}+\frac{L_{out}}{4}-1 \qquad (2103)$$

6.9.3 Complex low delay filter bank synthesis

The analysis stage of the CLDFB is described in sub-clause 5.1.2.1. The synthesis stage transforms the time-frequency matrix of the complex coefficients $X_{CR}(t,k)$ and $X_{CI}(t,k)$ to the time domain. The combination of analysis and synthesis is used for sample rate conversions. Also adaptive sample rate conversions are handled by the CLDFB, including sample rate changes in the signal flow.

The sample rate of the reconstructed output signal $s_{Crec}(n)$ depends on the number of bands L_{Cs} used for the synthesis stage, i.e. $sr_{Cs}=L_{Cs}\cdot 800Hz$. In case, $L_{Cs}>L_{Ca}$ (number of bands in analysis stage), the coefficients $>L_{Ca}$ are initialized to zero before synthesizing.

For the synthesis operation, a demodulated vector $z_t(n)$ is computed for each time step t of the sub-bands.

$$z_t(n) = \frac{1}{2}\cdot\frac{1}{L_{Cs}}\left[\sum_{k=0}^{k=L_{Cs}-1}X_{CR}(t,k)\cos\left[\frac{\pi}{L_{Cs}}(n+n_0)\left(k+\frac{1}{2}\right)\right]+\sum_{k=0}^{k=L_{Cs}-1}X_{CI}(t,k)\sin\left[\frac{\pi}{L_{Cs}}(n+n_0)\left(k+\frac{1}{2}\right)\right]\right] \qquad (2104)$$

$$for\ n=0..10L_{Cs}-1$$

where n_0 is identical to the one defined for the analysis operation (see 5.1.2.1). The vector is then windowed by the filter bank prototype to prepare the overlap-add operation

$$z_{wt}(n) = w_c(n) \cdot z_t(n) \quad for\ n = 0..10L_{Cs} - 1 \tag{2105}$$

Then the recent ten windowed vectors are combined in an overlap-add operation to reconstruct the signal from the CLDFB coefficients.

$$s_{Crec}(n) = \sum_{t=0}^{-9} z_{wt}(n + t \cdot L_{Cs}) \quad for\ n = 0..L_{Cs} - 1 \tag{2106}$$

6.9.4　High pass filtering

At the final stage, the signal is high pass filtered to generate the final output signal. The high pass operation is identical to the one used in the pre-processing of the EVS encoder as described in 5.1.1.

7　Description of the transmitted parameter indices

7.1　Bit allocation for the default option

The allocation of the bits for various operating modes in the EVS encoder is shown for each bitrate in the following tables. Note that the most significant bit (MSB) of each codec parameter is always sent first. In the tables below, the abbreviation CT is used to denote the coder type and the abbreviation BW is used to denote the bandwidth.

7.1.1　Bit allocation at VBR 5.9, 7.2 – 9.6 kbps

The EVS codec encodes NB and WB content at 7.2 and 8.0 kbps with CELP core or HQ-MDCT core. No extension layer is used at these bitrates. The EVS codec encodes NB and WB content at 9.6 kbps with CELP core or TCX core. To encode WB signals at 9.6 kbps, the CELP core uses TBE extension layer and the TCX core uses IGF extension layer. Similarly to encode SWB signals at 9.6 kbps, the CELP core uses TBE extension layer and the TCX core uses IGF extension layer.

VBR mode uses 4 different active frame types with different bit rates to achieve the average bit rate of 5.9 kbps. The 4 different frame rates are 2.8 kbps PPP frame, 2.8 kbps NELP frame, and 7.2 kbps and 8 kbps CELP frames. The CT bits are allocated as 1 bit to differentiate active 2.8 kbps (PPP or NELP) frames from any other 2.8 kbps frames (such as SID frame with payload header) and the remaining 2 bits are used to represent NB PPP, WB PPP, NB NELP and WB NELP frames.

Table 179: Bit allocation at 7.2 – 9.6 kbps and 2.8 kbps PPP/NELP

Description	2.8 PPP	2.8 NELP	7.2	8.0	9.6		
core	CELP	CELP	CELP HQ-MDCT	CELP HQ-MDCT	CELP		TCX
ext. layer	NO	NO	NO	NO	SWB TBE	WB TBE	IGF
Number of bits per frame	56	56	144	160	192		
BW			4		2		
CT	3	3			3		
core bits	53	53	140	156	171	181	187
WB/SWB ext. layer bits					16	6	

Note that the BW and CT parameters are combined together to form a single index at 7.2 and 8.0 kbps. This index conveys the information whether CELP core or HQ-MDCT core is used. At 9.6 kbps, the information about using the CELP core or the TCX core is encoded as a part of the CT parameter.

7.1.2 Bit allocation at 13.2 kbps

The EVS codec encodes NB, WB and SWB content at 13.2 kbps with CELP core, HQ-MDCT core, or TCX core. For WB signals, the CELP core uses TBE or FD extension layer. For SWB signals, the CELP core uses TBE or FD extension layer, and the TCX core uses IGF extension layer.

Table 180: Bit allocation at 13.2 kbps

Description	13.2						
core	CELP	HQ-MDCT	TCX	CELP			TCX
ext. layer	NO	NO	NO	WB TBE	WB FD	SWB TBE SWB FD	IGF
Number of bits per frame	264						
BW, CT, RF	5						
TCX/HQ-MDCT core flag		1	1				1
TCX CT			2				2
TD/FD ext. layer flag				1	1	1	
core bits	259	258	256	238	252	227	256
WB/SWB ext. layer bits				20	6	31	

Note that the BW, CT, and RF parameters are combined together to form a single index. This index also conveys the information whether LP-based core or MDCT-based core (TCX or HQ-MDCT) is used. The decision between the HQ-MDCT core and the TCX core is encoded with one extra bit called MDCT core flag. At this bitrate, the TCX coder type is encoded with 2 extra bits (TCX CT).

7.1.3 Bit allocation at 16.4 and 24.4 kbps

The EVS codec encodes NB, WB, SWB and FB content at 16.4 and 24.4 kbps with CELP core, HQ-MDCT core or TCX core. For SWB and FB signals, the CELP core uses TBE extension layer and the TCX core uses IGF extension layer.

Table 181: Bit allocation at 16.4 kbps

Description	16.4					
core	CELP	TCX	HQ-MDCT	CELP	TCX	CELP
ext. layer	NO	NO	NO	SWB TBE	IGF	FB TBE
Number of bits per frame	328					
BW	2					
Reserved flag	1					
CT	3	4	2	3	4	3
core bits	322	321	323	286	321	287
SWB ext. layer bits				33		31
FB ext. layer bits						4
Padding bits				3		

Table 182: Bit allocation at 24.4 kbps

Description	24.4					
core	CELP	TCX	HQ-MDCT	CELP	TCX	CELP
ext. layer	NO	NO	NO	SWB TBE	IGF	FB TBE
Number of bits per frame	488					
BW	2					
Reserved flag	1					
CELP/MDCT core flag	1					
TCX/HQ-MDCT core flag		1	1		1	
CELP->HQ core switching flag			1-2			
CT	2	2		2	2	2
core bits	482	481	481-2	422	481	423
SWB ext. layer bits				57		55
FB ext. layer bits						4
Padding bits				3		

The information about using the CELP core or the MDCT-based core (HQ-MDCT or TCX) is transmitted as a 1-bit CELP/MDCT core flag. In the case of MDCT-based core, the next bit decides whether HQ-MDCT core or TCX core is used. In the case of TCX, the remaining 2 bits are used to represent the TCX coder type (TCX CT). In the case of HQ-MDCT core, the next one or two bits signal whether the previous frame was encoded with the CELP core or not. The second bit is used to signal its internal sampling rate (12.8 or 16 kHz) only when the previous frame was encoded with the CELP core.

7.1.4   Bit allocation at 32 kbps

The EVS codec encodes WB, SWB and FB content at 32 kbps with CELP core, HQ-MDCT core, or TCX core. For SWB and FB signals, the CELP core uses TBE or FD extension layer and the TCX core uses IGF extension layer.

Table 183: Bit allocation at 32 kbps

Description	32					
core	CELP	HQ-MDCT	TCX	CELP	TCX	CELP
ext. layer	NO	NO	NO	SWB TBE/FD	IGF	FB TBE/FD
Number of bits per frame	640					
CELP/MDCT core flag	1					
CELP->HQ core switching flag		1-2				
TCX/HQ-MDCT core flag		1	1		1	
BW	4	2	2	4	2	4
CT			2		2	
TBE/FD ext. layer flag				1		
core bits	634	632-3	632	602	633	576
SWB ext. layer bits				55/31		55/31
FB ext. layer bits						4

The information about using the CELP core or the MDCT-based core (HQ-MDCT or TCX) is transmitted as a 1-bit CELP/MDCT core flag. If CELP core is selected, the BW and CT parameters are combined together to form a single index. In the case of MDCT-based core, the next bit decides whether HQ-MDCT core is used or the TCX core is used. In the case of TCX, the remaining 2 bits are used to represent the TCX coder type (TCX CT). In the case of HQ-MDCT core, the next one or two bits signal whether the previous frame was encoded with the CELP core or not. The second bit is used to signal its internal sampling rate (12.8 or 16 kHz) only when the previous frame was encoded with the CELP core. Finally, 1 bit is used to distinguish between TBE and FD extension layer in the case of CELP core.

7.1.5 Bit allocation at 48, 64, 96 and 128 kbps

The EVS codec encodes WB, SWB and FB content at 48 kbps with TCX core only. For SWB and FB signals, the TCX core uses IGF extension layer. At 64 kbps, the EVS codec encodes WB, SWB and FB content with CELP core or HQ-MDCT core. For SWB and FB signals, the CELP core uses FD extension layer.

Table 184: Bit allocation at 48, 64, 96 and 128 kbps

Description	48		64			96		128	
core	TCX	TCX	CELP	HQ-MDCT	CELP	TCX	TCX	TCX	TCX
ext. layer	NO	IGF	NO	NO	SWB FD FB FD	NO	IGF	NO	IGF
Number of bits per frame	960		1280			1920		2560	
CELP/MDCT core flag			1						
CELP->HQ core switching flag				1-2					
TCX/HQ-MDCT core flag				1					
BW	2		4	2	4	2		2	
CT									
Reserved flag	1					1		1	
TCX CT	3					3		3	
core bits	954	954	1275	1274-5	954	1914	1914	2554	2554
ext. layer bits					326				

At 64 kbps, the information about using the CELP core or the HQ-MDCT core is transmitted as a 1-bit CELP/MDCT core flag. If CELP core is selected, the BW and CT parameters are combined together to form a single index. In the case of HQ-MDCT core, the next one or two bits signal whether the previous frame was encoded with the CELP core or not. The second bit is used to signal its internal sampling rate (12.8 or 16 kHz) only when the previous frame was encoded with the CELP core.

7.2 Bit allocation for SID frames in the DTX operation

The SID payload consists of 48 bits independent of the bandwidth, bit rate and mode. The EVS codec supports three types of SID frames, one for the FD-CNG and two for the LP-CNG scheme.

Table 185: Bit allocation of FD-CNG SID frame

Description	FD-CNG
Number of bits per frame	48
CNG type flag	1
Bandwidth indicator	2
CELP sample rate	1
Global gain	7
Spectral band energy	37

The CNG type flag determines the usage of FD-CNG or LP-CNG. The bandwidth indicator indicates NB, WB, SWB or FB. The CELP sample rate can be 12.8 kHz or 16 kHz. The remaining bits are used for the spectral envelope information.

Table 186: Bit allocation of LP-CNG SID frame

Description	WB SID	SWB SID
Number of bits per frame	48	48
CNG type flag	1	1
Bandwidth indicator	1	1
Core sampling rate indicator	1	1
Hangover frame counter	3	3
LSF bits	29	29
Low-band energy bits	7	7
Low-band excitation spectral envelope bits	6	N/A
High-band energy bits	N/A	4
Unused bits	N/A	2

The CNG type flag determines if the SID belongs to FD-CNG or LP-CNG. The bandwidth indicator indicates whether the SID is a WB or a SWB SID. The core sampling rate indicator indicates whether the core is running at 12.8 kHz or 16 kHz sampling rate. The hangover frame counter indicates the number of hangover frames preceding the SID. The low-band excitation spectral envelope bits are only applicable to WB SID. The high-band energy bits are only applicable to SWB SID.

7.3 Bit allocation for the AMR-WB-interoperable option

The AMR-WB-interoperable option has the same bit allocation as AMR-WB. For more details see clause 7 of [9].

7.4 Bit Allocation for the Channel-Aware Mode

The EVS codec encodes WB and SWB content at 13.2 kbps channel aware mode with CELP core or TCX core for the primary frame as well as the partial redundant frame (RF). For both WB and SWB signals, the CELP core uses TBE extension layer and the TCX core uses IGF extension layer.

The [BW, CT, and RF] information is packed in 5 bits. When RF flag is set to zero, the channel aware mode at 13.2 kbps will be a bit exact implementation of the EVS 13.2 kbps mode described in subclause 7.1.2. An ACELP partial RF information can be transmitted along with an ACELP or a TCX primary copy. Similarly, a TCX partial RF information can be transmitted along with an ACELP or a TCX primary copy. The RF frame offset information (i.e., offset = 2 or 3, or 5, or 7) at which the partial copy is transmitted with the primary frame is included in the bit stream. Similarly, the RF frame type with 3 bits that signals (RF_NO_DATA, RF_TCXFD, RF_TCXTD1, RF_TCXTD2, RF_ALLPRED, RF_NOPRED, RF_GENPRED, and RF_NELP) is included in the bit stream. Depending on the RF frame type, the distribution of number of bits used for primary copy and partial RF information varies. The last three bits in the bit stream contains the RF frame type information. The two bits before the RF frame type information contains the RF offset data. The signalling [BW, CT, and RF] is carried in the first 5 bits in the bit stream for ease of parsing by the JBM.

Table 187: Bit allocation at 13.2 kbps channel aware mode

Description	13.2 channel aware		
core	**CELP**		**TCX**
ext. layer	**WB TBE**	**SWB TBE**	**IGF**
Number of bits per frame	264		
BW, CT, RF	5		
core bits (primary)	183-248	171-236	189-254
WB/SWB ext. layer bits (primary)	6	18	
Core bits (partial RF)	0-60	0-60	0-65
WB/SWB ext. layer bits (partial RF)	0-5	0-5	
RF offset	2		
RF frame type	3		

Annex A (normative):
RTP Payload Format and SDP Parameters

A.0   General

This Annex describes a generic RTP payload format and SDP parameters for the EVS codec. The payload format consists of the RTP header, payload header, and payload data.

The byte order used in this specification is the network byte order, i.e., the most significant byte is transmitted first. The bit order is most significant bit first. This practice is presented in all figures as having the most significant bit located left-most on each line and indicated with the lowest number.

A.1   RTP Header Usage

The format of the RTP header is specified in RFC 3550 [30]. This payload format uses the fields of the header in a manner consistent with the usages in RFC 3550 [30].

The timestamp clock frequency for EVS codec is 16 kHz regardless of audio bandwidth. The duration of one speech frame-block is 20 ms for both EVS Primary and EVS AMR-WB IO modes. Thus, the timestamp is increased by 320 for each consecutive frame-block.

The RTP header marker bit (M) shall be set to 1 if the first frame-block carried in the packet contains a speech frame which is the first in a talkspurt. For all other packets the marker bit shall be set to zero (M=0).

A.2   RTP Payload Format

The EVS RTP Payload Format includes a Compact format and a Header-Full format which are used depending on the required functionalities within a session and whether only a single frame is transmitted. These two formats can be switched during a session, but the RTP Payload Format can also be restricted to use only Header-Full format as described in Annex A.3 and TS 26.114 [13].

In addition to the EVS RTP Payload Format, RFC 4867 [15] format shall also be supported for the EVS AMR-WB IO modes to provide the backward interoperability with legacy AMR-WB terminals.

A.2.1   EVS codec Compact Format

In the Compact format, the RTP payload consists of exactly one coded frame for the EVS Primary mode, and one coded frame and one 3-bit CMR field for the EVS AMR-WB IO mode. The Compact format uses protected payload sizes that uniquely identify EVS codec modes (EVS Primary or EVS AMR-WB IO mode) and bit-rates. The protected payload sizes are used for determining the bit-rate of a received coded frame at the receiver.

Table A.1 shows the protected payload sizes and the corresponding bit-rates to be used for Compact RTP payload format.

Table A.1: Protected payload sizes

Mode	Payload Size (bits)	Bitrate (kbps)
EVS Primary	48	2.4 (EVS Primary SID)
Special case (see clause A.2.1.3)	56	2.8
EVS AMR-WB IO	136	6.6
EVS Primary	144	7.2
EVS Primary	160	8
EVS AMR-WB IO	184	8.85
EVS Primary	192	9.6
EVS AMR-WB IO	256	12.65
EVS Primary	264	13.2
EVS AMR-WB IO	288	14.25
EVS AMR-WB IO	320	15.85
EVS Primary	328	16.4
EVS AMR-WB IO	368	18.25
EVS AMR-WB IO	400	19.85
EVS AMR-WB IO	464	23.05
EVS AMR-WB IO	480	23.85
EVS Primary	488	24.4
EVS Primary	640	32
EVS Primary	960	48
EVS Primary	1280	64
EVS Primary	1920	96
EVS Primary	2560	128

A.2.1.1   Compact format for EVS Primary mode

In the Compact format for EVS Primary mode, the RTP payload consists of exactly one coded frame. Hence, the coded frame follows the RTP header without any additional EVS RTP payload header.

The payload represents a speech frame of 20 ms encoded with the EVS codec bit-rate identified by the payload size. The bits are in the same order as produced by the EVS encoder, where the first bit is placed left-most immediately following the RTP header.

A.2.1.2   Compact format for EVS AMR-WB IO mode (except SID)

In the Compact format for EVS AMR-WB IO mode, except SID, the RTP payload consists of one 3-bit CMR field, one coded frame, and zero-padding bits if necessary.

A.2.1.2.1   Representation of Codec Mode Request (CMR) in Compact format for EVS AMR-WB IO mode

The 3-bit CMR field carries the codec mode request information to signal to the sender the requested AMR-WB [37] or EVS AMR-WB IO codec mode to be applied for encoding. The signallings of AMR-WB and EVS AMR-WB IO with the 3-bit CMR field are defined as shown in Table A.2. The CMR field in Compact format for EVS AMR-WB IO mode comprises a 3-bit element $[c(0), c(1), c(2)]$ for signalling codec mode requests for the following EVS AMR-WB IO or AMR-WB codec modes.

**Table A.2: 3-bit signalling element and EVS AMR-WB IO/AMR-WB CMR**

C(0)	C(1)	C(2)	Requested Mode
0	0	0	6.6
0	0	1	8.85
0	1	0	12.65
0	1	1	15.85
1	0	0	18.25
1	0	1	23.05
1	1	0	23.85
1	1	1	none

Due to the 3-bit limitation, there is not enough signalling space for all EVS AMR-WB IO codec modes. Consequently, CMRs in Compact format for EVS AMR-WB IO are limited to include the most frequently used set of EVS AMR-WB IO /AMR-WB modes as shown in Table A.2. CMRs for EVS AMR-WB IO / AMR-WB modes 14.25 and 19.85 are not supported in Compact format for EVS AMR-WB IO. In case a request needs to be transmitted for either mode, it should be re-mapped to the next lower rate mode (12.65 and 18.25, respectively). Alternatively the Header-Full format may be used to transmit CMRs to 14.25 and 19.85 modes.

In case of restrictions in the allowed codec modes to an Active Codec Set (ACS) defined in TS 26.103 [39] or codec mode set (e.g. Config-WB-Code=0 used in TS 26.114 [13]), the mode shall be re-mapped to the next lower rate mode in this ACS. Codec mode requests for EVS primary modes shall be made using the Header-Full format.

A.2.1.2.2 Payload structure of Compact EVS AMR-WB IO mode frame

In order to minimize the need for bit re-shuffling in media gateways in case of payload format conversion to or from AMR-WB bandwidth-efficient format according to [15], the speech data bits are inserted after CMR, starting with bit $d(1)$. Speech data bit $d(0)$ is appended after the last speech data bit.
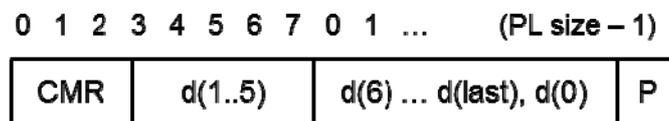
Figure A.1. Payload structure of Compact EVS AMR-WB IO.

The speech data payload represents a speech frame of 20 ms encoded with EVS AMR-WB IO bit-rate (mode) identified by the payload size. The order and numbering notation of the bits are as specified for Interface Format 1 (IF1) in Annex B of [36] for AMR-WB. The bits of the speech frames are arranged in the order of decreasing sensitivity, giving a re-ordered bit sequence $\{d(0),d(1),...,d(K-1)\}$.

If a total of three CMR bits and coded frame bits is not a multiple of 8, zero-padding bits are added so that the total becomes a multiple of 8. One zero-padding bit is required for EVS AMR-WB IO mode 6.6 and four zero-padding bits are required for EVS AMR-WB IO mode 8.85. In other mode no padding bits are inserted. With the exception of SID frames, the EVS AMR-WB IO Compact payload follows the RTP header without any additional EVS RTP payload header.

Note that no Compact frame format EVS AMR-WB IO SID frames is defined. For such frames the Header-Full format with CMR byte shall be used (see clause A.2.1.3).

NOTE: The Q bit defined in RFC 4867 [15] is not present in the Compact payload structure of EVS AMR-WB IO. Therefore it shall be ensured that the speech payload is not damaged. In case of a conversion of RFC 4867 formatted packets to Compact payload format, damaged frames (indicated by the Q bit) shall be discarded and not converted.

A.2.1.3 Special case for 56 bit payload size (EVS Primary or EVS AMR-WB IO SID)

The Compact format for EVS Primary 2.8 kbps frames (56 bits) has the same payload size (56 bits) as the Header-Full format for EVS AMR-WB IO SID frames with CMR byte.
Hence, two types of frames can be carried in the 56 bit payload case:

- EVS Primary 2.8 kbps frame in Compact format.

- EVS AMR-WB IO SID frame in Header-Full format (see clause A.2.2) with one CMR byte.

- The payload structure and bit ordering of EVS Primary 2.8 kbps frame in Compact format is defined in Figure A.2.
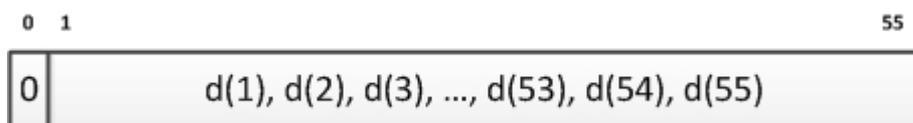
Figure A.2. Payload structure for EVS Primary 2.8 kbps (56-bit) payload

The resulting ambiguity between EVS Primary 2.8 kbps and EVS AMR-WB IO SID frames is resolved through the most significant bit (MSB) of the first byte of the payload. By definition, the first data bit d(0) of the EVS Primary 2.8 kbps is always set to '0'. Therefore, if the MSB of the first byte of the payload is set to '0' (see Figure A.2), then the payload is an EVS Primary 2.8 kbps frame in Compact format. Otherwise it is an EVS AMR-WB IO SID frame in Header-Full format with one CMR byte. The structure of EVS AMR-WB IO SID frame with Header-Full format is described in clause A.2.2.

A.2.2 EVS codec Header-Full format

In the Header-Full format, the payload consists of one or more coded frame(s) with EVS RTP payload header(s). There are two types of EVS RTP payload header: Table of Content (ToC) byte and Codec Mode Request (CMR) byte. The detailed header structure is described in clause A.2.2.1.

A.2.2.1 EVS RTP payload structure

The complete payload of Header-Full EVS frames comprises an optional CMR byte, followed by one or several ToC bytes, followed by speech data bits and possible zero-padding bits. Padding bits shall be discarded by the receiver. The purpose of padding is two-fold:

- In the case of EVS AMR-WB IO frames, payload data may need to be octet-aligned using zero-padding bits at the end of the payload. Note that EVS Primary frames are by definition octet-aligned (see clause A.2.2.1.4.1).

- When required, zero-padding bits are also used to increase the total payload size by byte increments such that conflicts with any of the protected sizes reserved for the Compact format are avoided (see clause A.2.2.1.4.2).

CMR and ToC bytes use MSB as Header Type identification bit (H) in order to identify the type of EVS RTP payload header. If the H bit is set to 0, the corresponding byte is a ToC byte, and if set to 1, the corresponding byte is a CMR byte. A CMR byte, if present, shall be located before ToC byte(s).

Figure A.3 shows the general structure of Header-Full payload format.
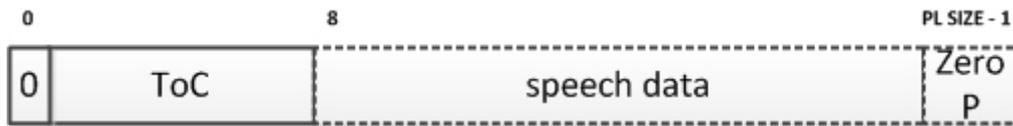
(a) Payload structure of Header-Full format with ToC single frame
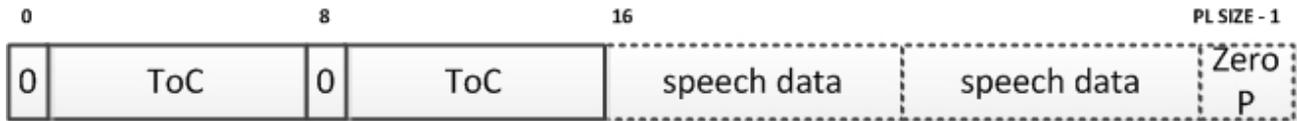
(b) Payload structure of Header-Full format with ToC multiple frames
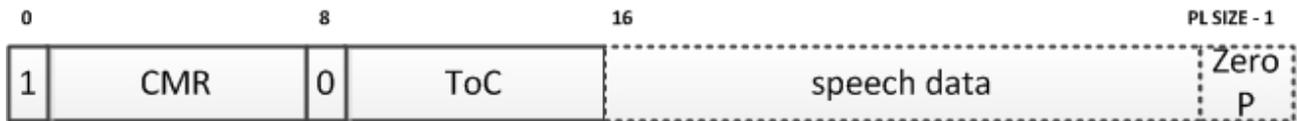
(c) Payload structure of Header-Full format with CMR + ToC single frame

(d) Payload structure of Header-Full format with CMR + ToC multiple frames

Figure A.3 Payload structure of Header-Full format

NOTE: The zero padding at the end of packet, indicated in Figure A.3 as "Zero P", does not represent the octet-alignment for AMR-WB IO data described in clause A.2.2.1.4.1, but it represents the zero-padding for size collision avoidance described in clause A.2.2.1.4.2.

A.2.2.1.1   CMR byte

Codec Mode Request (CMR) byte structure is shown in Figure A.4. This CMR byte shall be present for EVS AMR-WB IO speech and SID frames in Header-Full format. For EVS Primary mode, the CMR byte is only used when a CMR needs to be transmitted or if required by session negotiation. The request indicated in the CMR byte shall comply with the media type parameters (e.g. allowed bit-rates or audio bandwidths) that are negotiated in the session.

NOTE 1: There is no SDP MIME signalling parameter defined that can be used to disallow all CMRs with T-bits "001". However, the mode-set MIME parmeter can be used to restrain the allowed EVS AMR-WB IO codec modes. If this mode-set parameter is not included in the media type parameters, then all 9 modes of the EVS AMR-WB IO codes modes are allowed.

The media receiver in the MTSI terminal shall be prepared to receive any speech frames within the negotiated media type parameter set as well as SID frames, irrespective of the CMR it sends or receives.

NOTE 2: The media receiver can receive such frames for various reasons. For instance, after a handover to AMR-WB, a MGW can send speech frames with an EVS AMR-WB IO codec mode even if it receives CMR byte of EVS Primary mode (T-bits not "001").

The bit-rate in the CMR byte of EVS Primary mode (T-bits not "001") indicates the highest bit-rate that the media receiver wants to receive. The media sender should use the bit-rate requested in the received CMR and shall not use a higher bit-rate. The media sender may use a lower bit-rate than the requested bit-rate within the set of negotiated bit-rates.

If a single audio bandwidth is negotiated for EVS Primary mode, the CMR shall indicate this bandwidth in its T-bits, unless the mode of operation is switched by a received CMR between EVS Primary and EVS AMR-WB IO.

If a range of audio bandwidths is negotiated for EVS Primary mode, then the audio bandwidth in the CMR byte of EVS Primary mode indicates the highest audio bandwidth that the media receiver wants to receive. The media sender should use the audio bandwidth requested in the received CMR.

A CMR with T-bits "001" (i.e. a CMR for the EVS AMR-WB IO mode of operation) indicates the highest EVS AMR-WB IO codec mode that the media receiver wants to receive. When receiving a CMR with T-bits "001", the media

sender shall use the EVS AMR-WB IO mode of operation. The media sender should use the EVS AMR-WB IO codec mode (bit rate) requested in the received CMR and shall not use a higher codec mode (higher bit rate). The media sender may use a lower EVS AMR-WB IO codec mode within the negotiated mode-set.

When a CMR is received, requesting a bit-rate and/or audio bandwidth that does not comply with the negotiated media parameters, it shall be ignored.

The request in the received CMR is valid until a new request is received.

Figure A.4. CMR byte

H (1 bit): Header Type identification bit. For the CMR byte this bit is always set to 1.

T (3 bits): These bits indicate the Type of Request in order to distinguish EVS AMR-WB IO and EVS Primary bandwidths.

D (4 bits): These bits indicate the codec mode request.

The possible values and corresponding CMRs are defined in Table A.3.

Table A.3: Structure of CMR

Code		Definition		Code		Definition	
T	D			T	D		
000	0000	NB	5.9 (VBR)	010	0000	WB	5.9 (VBR)
	0001	NB	7.2		0001	WB	7.2
	0010	NB	8.0		0010	WB	8
	0011	NB	9.6		0011	WB	9.6
	0100	NB	13.2		0100	WB	13.2
	0101	NB	16.4		0101	WB	16.4
	0110	NB	24.4		0110	WB	24.4
	0111		Not used		0111	WB	32
	1000		Not used		1000	WB	48
	1001		Not used		1001	WB	64
	1010		Not used		1010	WB	96
	1011		Not used		1011	WB	128
	1100		Not used		1100		Not used
	1101		Not used		1101		Not used
	1110		Not used		1110		Not used
	1111		Not used		1111		Not used
001	0000	IO	6.6	011	0000		Not used
	0001	IO	8.85		0001		Not used
	0010	IO	12.65		0010		Not used
	0011	IO	14.25		0011	SWB	9.6
	0100	IO	15.85		0100	SWB	13.2
	0101	IO	18.25		0101	SWB	16.4
	0110	IO	19.85		0110	SWB	24.4
	0111	IO	23.05		0111	SWB	32
	1000	IO	23.85		1000	SWB	48
	1001		Not used		1001	SWB	64
	1010		Not used		1010	SWB	96
	1011		Not used		1011	SWB	128
	1100		Not used		1100		Not used
	1101		Not used		1101		Not used
	1110		Not used		1110		Not used
	1111		Not used		1111		Not used

Table A.3: Structure of CMR (continued)

Code		Definition		Code		Definition	
T	**D**			**T**	**D**		
100	0000		Not used	110	0000	SWB	13.2 CA-L-O2
	0001		Not used		0001	SWB	13.2 CA-L-O3
	0010		Not used		0010	SWB	13.2 CA-L-O5
	0011		Not used		0011	SWB	13.2 CA-L-O7
	0100		Not used		0100	SWB	13.2 CA-H-O2
	0101	FB	16.4		0101	SWB	13.2 CA-H-O3
	0110	FB	24.4		0110	SWB	13.2 CA-H-O5
	0111	FB	32		0111	SWB	13.2 CA-H-O7
	1000	FB	48		1000		Not used
	1001	FB	64		1001		Not used
	1010	FB	96		1010		Not used
	1011	FB	128		1011		Not used
	1100		Not used		1100		Not used
	1101		Not used		1101		Not used
	1110		Not used		1110		Not used
	1111		Not used		1111		Not used
101	0000	WB	13.2 CA-L-O2	111	0000		Reserved
	0001	WB	13.2 CA-L-O3		0001		Reserved
	0010	WB	13.2 CA-L-O5		0010		Reserved
	0011	WB	13.2 CA-L-O7		0011		Reserved
	0100	WB	13.2 CA-H-O2		0100		Reserved
	0101	WB	13.2 CA-H-O3		0101		Reserved
	0110	WB	13.2 CA-H-O5		0110		Reserved
	0111	WB	13.2 CA-H-O7		0111		Reserved
	1000		Not used		1000		Reserved
	1001		Not used		1001		Reserved
	1010		Not used		1010		Reserved
	1011		Not used		1011		Reserved
	1100		Not used		1100		Reserved
	1101		Not used		1101		Reserved
	1110		Not used		1110		Reserved
	1111		Not used		1111		NO_REQ

A.2.2.1.2   ToC byte

The Table of Content (ToC) byte structure is shown in Figure A.5.

0	1	2	3	4	5	6	7
H	F	FT					
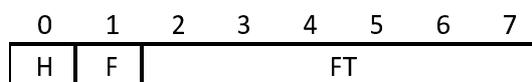
Figure A.5. ToC byte

H (1 bit): Header Type identification bit. For the ToC byte this bit is always set to 0.

F (1 bit): If set to 1, the bit indicates that the corresponding frame is followed by another speech frame in this payload, implying that another ToC byte follows this entry. If set to 0, the bit indicates that this frame is the last frame in this payload and no further header entry follows this entry.

FT (6 bits): Frame type index. These bits indicate whether the EVS Primary or EVS AMR-WB IO mode, or comfort noise (SID) mode of the corresponding frame is carried in this payload. FT is further divided into 3 fields: EVS mode (1 bit), Unused/Q bit (1 bit) depending on the value of EVS mode bit, and EVS bit-rate (4 bits). The value of FT is defined in Tables A.4 and A.5.

Table A.4: Frame Type index when EVS mode bit = 0

EVS mode bit (1 bit)	Unused (1 bit)	EVS bit rate	Indicated EVS mode and bit rate
0	0	0000	Primary 2.8 kbps
0	0	0001	Primary 7.2 kbps
0	0	0010	Primary 8.0 kbps
0	0	0011	Primary 9.6 kbps
0	0	0100	Primary 13.2 kbps
0	0	0101	Primary 16.4 kbps
0	0	0110	Primary 24.4 kbps
0	0	0111	Primary 32.0 kbps
0	0	1000	Primary 48.0 kbps
0	0	1001	Primary 64.0 kbps
0	0	1010	Primary 96.0 kbps
0	0	1011	Primary 128.0 kbps
0	0	1100	Primary 2.4kbps SID
0	0	1101	For future use
0	0	1110	SPEECH_LOST
0	0	1111	NO_DATA

Table A.5: Frame Type index when EVS mode bit = 1

EVS mode bit (1 bit)	AMR-WB Q bit (1 bit)	EVS bit rate (4 bits)	Indicated EVS mode and codec mode
1	Q	0000	AMR-WB IO 6.6 kbps
1	Q	0001	AMR-WB IO 8.85 kbps
1	Q	0010	AMR-WB IO 12.65 kbps
1	Q	0011	AMR-WB IO 14.25 kbps
1	Q	0100	AMR-WB IO 15.85 kbps
1	Q	0101	AMR-WB IO 18.25 kbps
1	Q	0110	AMR-WB IO 19.85 kbps
1	Q	0111	AMR-WB IO 23.05 kbps
1	Q	1000	AMR-WB IO 23.85 kbps
1	Q	1001	AMR-WB IO 2.0 kbps SID
1	Q	1010	For future use
1	Q	1011	For future use
1	Q	1100	For future use
1	Q	1101	For future use
1	Q	1110	SPEECH_LOST
1	Q	1111	NO_DATA

NOTE: The 4-bit EVS bit-rate index and the mapping to EVS AMR-WB IO codec mode in Table A.4 are the same as used for the Frame Type of AMR-WB. See Table 1a [36]. The Q bit for EVS AMR-WB IO has the same definition as in [15]. If Q bit is set to 0, this indicates that the corresponding frame is severely damaged. The receiver should handle such a severly damaged frame properly by applying bad frame processing according to [6].

Packets containing only NO_DATA frames should not be transmitted in any payload format configuration, except for situations, when CMR needs to be sent immediately. Frame-blocks containing only NO_DATA frames at the end of the packet should not be transmitted in any payload format configuration. In addition, frame blocks containing only NO_DATA frames in the beginning of the packet should not be included in the payload.

For sessions with multiple mono-channels, see clause A.2.5.

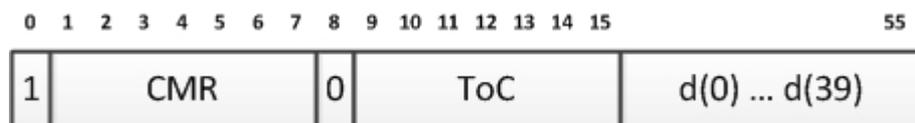## A.2.2.1.3   Speech Data

In Header-Full format, the RTP payload comprises, apart from headers and possible padding, one or several coded frames, the Speech Data.

In case the frame is coded EVS Primary mode data, the bits are in the same order as produced by the EVS encoder, where the first bit is placed left-most immediately following the EVS RTP payload header (CMR byte if present, and ToC bytes).

In case the frame is coded EVS AMR-WB IO mode data, the Speech Data field is constructed as described in RFC 4867 [15] for octet-aligned Mode, sub-clause 4.4.3. In accordance with this, in case multiple frames are included in the payload, the last octet of each frame shall be padded with zero bits at the end if some bits in the octet are not used. The padding bits shall be ignored on reception.

In case the frame is coded EVS AMR-WB IO SID data, the payload structure and bit-ordering are defined in Figure A.6. The bits $d(0)$ to $d(39)$ are as defined in TS 26.201 [36], sub-clause 4.2.3.

Figure A.6. Payload structure for EVS AMR-WB IO SID (56 bit) payload

The EVS AMR-WB IO SID frame payload is identified by MSB of the first byte of the payload set to '1'.

A.2.2.1.4 Zero padding

A.2.2.1.4.1 Zero padding for octet alignment of speech data (EVS AMR-WB IO)

In EVS AMR-WB IO mode, the payload length is always made an integral number of octets by padding with zero bits if necessary (see clause A.2.2.1.3).

Note that, by definition, EVS Primary speech data is octet-aligned.

A.2.2.1.4.2 Zero padding for size collision avoidance

When "hf-only=0" or "hf-only" is not present, the RTP payload formatting function of the sender shall control the size of Header-Full RTP payload so that the Header-Full format RTP payload size does not collide with any of the protected Compact format RTP payload sizes listed in Table A.1, except for the special case of the 56-bit payload. If a Header-Full format RTP payload size collides with one of the protected Compact format RTP payload sizes, the RTP payload formatting function of the sender shall append an appropriate number of zero-padding bytes to the end of the payload such that payload sizes do not collide.

The Header-Full format representing an EVS AMR-WB IO SID frame (with one CMR byte and one ToC byte) is allowed to have the same 56 bits as EVS Primary 2.8 kbps in Compact format. In this special case, no padding bits shall be appended to the EVS AMR-WB IO SID frame.

A.2.2.1.4.3 Additional zero padding

If additional padding is required to bring the payload length to a larger multiple of octets or for some other purposes, then the P bit in the RTP header may be set and padding bits are appended as specified in [30].

A.2.3 Header-Full/Compact format handling

There are two format handling modes: Default mode and Header-Full-only mode.

A.2.3.1 Default format handling

When "hf-only=0" is present or when the "hf-only" attribute is not present, the Compact format shall be used in the following cases:

- A single mono EVS Primary mode frame is carried in an RTP packet without sending CMR.

- A single mono EVS AMR-WB IO mode frame with 3-bit CMR is carried in an RTP packet.

Otherwise, the Header-Full format with size collision avoidance shall be used.

The only exception in this default format handling is as follows: the Header-Full format may be used to transmit a single EVS AMR-WB IO frame to request 14.25 or 19.85 kbps in EVS AMR-WB IO mode as these two bit-rates cannot be indicated with the 3-bit CMR defined for Compact format.

A.2.3.2 Header-Full-only format handling

When "hf-only=1" is present, only the Header-Full format shall be used during the session. In other words, the Compact format shall not be used. The size collision avoidance shall not be performed by the RTP payload formatting function of the sender. The RTP payload decoding function of the receiver shall use ToC byte(s) to obtain the mode (i.e., EVS Primary or EVS AMR-WB IO) and the bit-rate regardless of the RTP payload size.

A.2.4 AMR-WB backward compatible EVS AMR-WB IO mode format

In order to provide backward interoperability with AMR-WB, the payload format in [15] shall also be supported for EVS AMR-WB IO mode. This payload format shall be used to communicate with a terminal not supporting EVS but supporting AMR-WB.

A.2.5 Sessions with multiple mono channels

The Header-Full EVS payload format supports transmission of multiple mono channels in the same way as described in the AMR-WB payload format [15].

A.2.5.1 Encoding of multiple mono channels

The speech encoders for different channels are not synchronized, which means that they may use different codec modes and may result in different VAD decisions depending on the content in each channel.

A.2.5.2 RTP header usage

The RTP time stamp is derived from the media time of the first frame of the first channel in the packet, even if that frame is a NO_DATA frame.

If a frame in the packet is an onset frame, then the Marker bit in the RTP header is set to '1'. However, since the encoders are not synchronized, they may use different VAD decisions for different channels. Hence, it is not sufficient to only use the Marker bit to detect onset frames, and to for example reset the jitter buffers in the receiver. The receiver needs to monitor the content of the channels, e.g., the Frame Type identifier, to find the transition from DTX to active speech for each individual channel.

A.2.5.3 Construction of the RTP payload

The ToC bytes of all frames from a frame-block are placed in consecutive order as defined in Section 4.1 [38]. Therefore, with N channels and K speech frame-blocks in a packet, there shall be N*K ToC bytes in the EVS RTP payload header, and the first N ToC bytes will be from the first frame-block, the second N ToC bytes will be from the second frame-block, and so on.

The payload shall include frames from all channels for each media time that is included. If a frame is not available for a channel, e.g., when the encoder for that channel is currently in DTX mode, then a NO_DATA frame shall be included instead. Since the payload always contains two or more frames, the Header-Full payload format shall be used.

The payload may contain a CMR byte according to the same rules as defined for single-channel session. When a CMR is received, it is applied equally to all channels. It may still happen that different channels are encoded in different modes, especially if independent encoders are used.

A.2.6 Storage Format

The storage format is used for storing EVS Primary or EVS AMR-WB IO speech frames in a file or as an email attachment. Multiple channel content is supported.

For EVS AMR-WB IO, the storage format of [15] can be used.

For EVS, the storage format has the following structure:

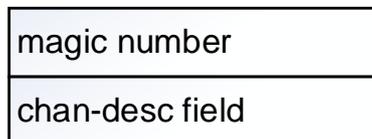| Header |
|--------|
| Speech frame 1 |
| …… |
| Speech frame n |
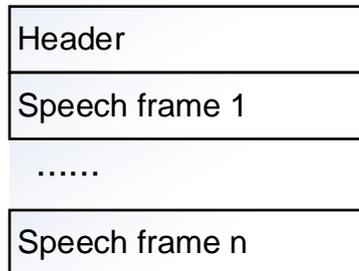
Figure A.7. Storage format for EVS

There is another storage format that is suitable for applications with more advanced demands on the storage format, like random access or synchronization with video. This format is the 3GPP-specified ISO-based multimedia file format specified in [40]. Its media type is specified in [41].

A.2.6.1 Header

The header consists of a magic number followed by a 32-bit channel description field, giving the header the following structure:

magic number
chan-desc field

Figure A.8. Header for EVS

The magic number shall consist of the ASCII character string:

"#!EVS_MC1.0\n" or (0x23214556535f4d43312e30)

The version number in the magic number string refers to the version of the file format.

The 32-bit channel description field is defined as a 32-bit number (unsigned integer, MSB first). This number indicates the number of audio channels contained in this storage file starting from 1 for mono to N for a multi-mono signal with N channels.

A.2.6.2 Speech Frames

After the header, speech frame-blocks consecutive in time are stored in the file. Each frame-block contains a number of octet-aligned speech frames equal to the number of channels stored in the increasing order, starting with channel 1. Each stored speech frame starts with a ToC byte (see clause A.2.2.1.2). Note that no CMR byte is needed.

Non-received speech frames or frame-blocks between SID frames during non-speech periods shall be stored as NO_DATA frames. Frames or frame-blocks lost during transmission shall be stored as SPEECH_LOST frames in complete frame-blocks to keep synchronization with the original media.

A.3 Payload Format Parameters

A.3.1 EVS Media Type Registration

The media type for the EVS codec is to be allocated from the standards tree. This clause defines parameters of the EVS payload format. This media type registration covers real-time transfer via RTP and non-real-time transfers via stored files. All media type parameters defined in this Annex shall be supported. The receiver must ignore any unspecified parameter.

Media type name: audio

Media subtype name: EVS

Required parameters: none

Optional parameters:

The parameters defined below apply to RTP transfer only.

The following parameters are applicable to both EVS Primary mode and EVS AMR-WB IO mode:

ptime: see RFC 4566 [27].

maxptime: see RFC 4566 [27].

evs-mode-switch: Permissible values are 0 and 1. If evs-mode-switch is 0 or not present, EVS primary mode is used at the start or update of the session for the send and the receive directions. If evs-mode-switch is 1, EVS AMR-WB IO mode is used at the start or update of the session for the send and the receive directions.

hf-only: Permissible values are 0 and 1. If hf-only is 0 or not present, both Compact and Header-Full formats can be used in the session for the send and the receive directions. If hf-only is 1, only Header-Full format without zero padding for size collision avoidance is used.

dtx: Permissible values are 0 and 1. If dtx is 0, DTX is disabled in the session for the send and the receive directions. If dtx is 1 or not present, DTX is enabled. If dtx is included, dtx-recv is redundant but if dtx-recv is included, it shall be identical to dtx.

dtx-recv: Permissible values are 0 and 1. If dtx-recv=0 is included for a payload type in the received SDP offer or the received SDP answer, and the payload type is accepted, the receiver shall disable DTX for the send direction. If dtx-recv=1 is included for a payload type in the received SDP offer or the received SDP answer, and this payload type is accepted, or if dtx-recv is not present for an accepted payload type, DTX is enabled.

max-red: See RFC 4867 [15].

channels: The number of audio channels. See RFC 3551 [38]. If channels is not present, its default value is 1. If both ch-send and ch-recv are included in the SDP with different numbers of channels for sending and receiving directions, channels is set to the larger of the two parameters.

cmr: Permissible values are -1, 0, and 1. If cmr is -1 and the session is in the EVS primary mode, CMR on the RTP payload header is disabled in the session. If cmr is -1 and the session is in the EVS AMR-WB IO mode, CMR in the CMR byte is restricted to the values of EVS AMR-WB IO bit-rates and NO_REQ as specified in Table A.3. If cmr is 0 or not present, the values of CMR specified in Table A.3 are enabled. If cmr is 1, CMR shall be present in each packet. CMR shall be compliant with the negotiated bit-rate and bandwidth media type attributes for EVS primary and EVS AMR-WB IO modes.

The following parameters are applicable only to EVS Primary mode:

br: Specifies the range of source codec bit-rate for EVS Primary mode (see Table 1 [2]) in the session, in kilobits per second, for the send and the receive directions. The parameter can either have: a single bit-rate (br1); or a hyphen-separated pair of two bit-rates (br1-br2). If a single value is included, this bit-rate, br1, is used. If a hyphen-separated pair of two bit-rates is included, br1 and br2 are used as the minimum bit-rate and the maximum bit-rate respectively. br1 shall be smaller than br2. br1 and br2 have a value from the set: 5.9, 7.2, 8, 9.6, 13.2, 16.4, 24.4, 32, 48, 64, 96, and 128. 5.9 represents the average bit-rate of source controlled variable bit rate (SC-VBR) coding, and 7.2, …, 128 represent the bit-rates of constant bit-rate source coding. Only bit-rates supporting at least one of the allowed audio bandwidth(s) shall be used in the session (see clause A.3.3.1). If br is not present, all bit-rates consistent with the negotiated bandwidth(s) are allowed in the session unless br-send or br-recv is present. If br is included, br-send or br-recv is redundant but if either br-send or br-recv, or both are included, they shall be identical to br. If br-send and br-recv are not identical, br shall not be used.

br-send: Specifies the range of source codec bit-rate for EVS Primary mode (see Table 1 [2]) in the session, in kilobits per second, for the send direction. The parameter can either have: a single bit-rate (br1); or a hyphen-separated pair of two bit-rates (br1-br2). If a single value is included, this bit-rate, br1, is used. If a hyphen-separated pair of two bit-rates is included, br1 and br2 are used as the minimum bit-rate and the maximum bit-rate respectively. br1 shall be smaller than br2. br1 and br2 have a value from the set: 5.9, 7.2, 8, 9.6, 13.2, 16.4, 24.4, 32, 48, 64, 96, and 128. 5.9 represents the average bit-rate of source controlled variable bit-rate (SC-VBR) coding, and 7.2, …, 128 represent the bit-rates of constant bit-rate source coding. Only bit-rates supporting at least one of the allowed audio bandwidth(s) shall be used in the session (see clause A.3.3.1). If br-send is not present, all bit-rates consistent with the negotiated bandwidth(s) are allowed in the session unless br is present.

br-recv: Specifies the range of source codec bit-rate for EVS Primary mode (see Table 1 [2]) in the session, in kilobits per second, for the receive direction. The parameter can either have: a single bit-rate (br1); or a hyphen-separated pair of two bit-rates (br1-br2). If a single value is included, this bit-rate, br1, is used. If a hyphen-separated pair of two bit-rates is included, br1 and br2 are used as the minimum bit-rate and the maximum bit-rate respectively. br1 shall be smaller than br2. br1 and br2 have a value from the set: 5.9, 7.2, 8, 9.6, 13.2, 16.4, 24.4, 32, 48, 64, 96, and 128. 5.9 represents the average bit-rate of source controlled variable bit-rate (SC-VBR) coding, and 7.2, …, 128 represent the bit-rates of constant bit-rate source coding. Only bit-rates supporting at least one of the allowed audio bandwidth(s) shall be used in the session (see clause A.3.3.1). If br-recv is not present, all bit-rates consistent with the negotiated bandwidth(s) are allowed in the session unless br is present.

bw: Specifies the audio bandwidth for EVS Primary mode (see Table 1 [2]) to be used in the session for the send and the receive directions. bw has a value from the set: nb, wb, swb, fb, nb-wb, nb-swb, and nb-fb. nb, wb, swb, and fb represent narrowband, wideband, super-wideband, and fullband respectively, and nb-wb, nb-swb, and nb-fb represent all bandwidths from narrowband to wideband, super-wideband, and fullband respectively. If bw is not present, all bandwidths consistent with the negotiated bit-rate(s) are allowed in the session unless bw-send or bw-recv is present. If bw is included, bw-send or bw-recv is redundant but if either bw-send or bw-recv, or both are included, they shall be identical to bw. If bw-send and bw-recv are not identical, bw shall not be used.

bw-send: Specifies the bandwidth (see Table 1 [2]) to be used in the session for the send direction. bw-send has a value from the set: nb, wb, swb, fb, nb-wb, nb-swb, and nb-fb. nb, wb, swb, and fb represent narrowband, wideband, super-wideband, and fullband respectively, and nb-wb, nb-swb, and nb-fb represent all bandwidths from narrowband to wideband, super-wideband, and fullband respectively. If bw-send is not present, all bandwidths consistent with the negotiated bit-rate(s) are allowed in the session unless bw is present.

bw-recv: Specifies the bandwidth (see Table 1 [2]) to be used in the session for the receive direction. bw-recv has a value from the set: nb, wb, swb, fb, nb-wb, nb-swb, and nb-fb. nb, wb, swb, and fb represent narrowband, wideband, super-wideband, and fullband respectively, and nb-wb, nb-swb, and nb-fb represent all bandwidths from narrowband to wideband, super-wideband, and fullband respectively. If bw-recv is not present, all bandwidths consistent with the negotiated bit-rate(s) are allowed in the session unless bw is present.

ch-send: Specifies the number of audio channels to be used in the session for the send direction. ch-send has an integer value from 1 to the maximum number of audio channels (see also clause A.3.2). If ch-send is not present, ch-send=1, mono, is supported.

ch-recv: Specifies the number of audio channels to be used in the session for the receive direction. ch-recv has an integer value from 1 to the maximum number of audio channels (see also clause A.3.2). If ch-recv is not present, ch-recv=1, mono, is supported.

ch-aw-recv: Specifies how channel-aware mode is configured or used for the receive direction. Permissible values are -1, 0, 2, 3, 5, and 7. If ch-aw-recv is -1, channel-aware mode is disabled in the session for the receive direction. If ch-aw-recv is 0 or not present, partial redundancy (channel-aware mode) is not used at the start of the session for the receive direction. If ch-aw-recv is positive (2, 3, 5, or 7), partial redundancy (channel-aware mode) is used at the start of the session for the receive direction using the value as the offset (See NOTE below). Partial redundancy is supported only when the bit-rate is 13.2 kbps and the bandwidth is wb or swb.

NOTE 1: If a positive (2, 3, 5, or 7) value of ch-aw-recv is declared for a payload type and the payload type is accepted, the receiver of the parameter shall send partial redundancy (channel-aware mode) at the start of the session using the value as the offset. If ch-aw-recv=0 is declared or not present for a payload type and the payload type is accepted, the receiver of the parameter shall not send partial redundancy (channel-aware mode) at the start of the session. If ch-aw-recv=-1 is declared for a payload type and the payload type is accepted, the receiver of the parameter shall not send partial redundancy (channel-aware mode) in the session. If ch-aw-recv is not present or a non-negative (0, 2, 3, 5, or 7) value of ch-aw-recv is declared for a payload type and the payload type is accepted, partial redundancy (channel-aware mode) can be activated or deactivated during the session based on the expected or estimated channel condition through adaptation signaling, such as CMR (see Annex A.2) or RTCP based signaling (see clause 10.2 of [13]). If ch-aw-recv is not present or a non-negative (0, 2, 3, 5, or 7) value of ch-aw-recv is declared for a payload type and the payload type is accepted, the partial redundancy offset value can also be adjusted during the session based on the expected or estimated channel condition through adaptation signaling.

NOTE 2: The frame erasure rate indicator for the channel-aware mode has two permissible values (LO, HI) and this indicator has to be initialized to HI, as specified in clause 5.8.4.

The following parameters are applicable only to EVS AMR-WB IO mode:

mode-set: Restricts the active codec mode set to a subset of all modes when the EVS codec operates in AMR-WB IO, for example, to be able to support transport channels such as GSM or UMTS networks. Possible value is a comma-separated list of modes from the set: 0, …, 8 (see Table 1a [36]). If mode-set is specified, it must be abided, and frames encoded with AMR-WB IO outside of the subset must not be sent in any RTP payload or used in codec mode request signal. If not present, all codec modes of AMR-WB IO are allowed for the payload type.

mode-change-period: See RFC 4867 [15].

mode-change-capability: See RFC 4867 [15], except that the default and the only allowed value of mode-change-capability is 2 for EVS AMR-WB IO. As the default and the only allowed value of mode-change-capibility is 2 in EVS AMR-WB IO, it is not required to include this parameter in the SDP.

mode-change-neighbor: See RFC 4867 [15].

Optional parameters of AMR-WB (see clause 8.2 of [15]) not defined above shall not be used in the EVS AMR-WB IO mode.

A.3.2 Mapping Media Type Parameters into SDP

The information carried in the media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [27], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the EVS codec, the mapping is as follows:

- The media type ("audio") goes in SDP "m=" as the media name.

- The media subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" shall be 16000, and the encoding parameters (number of channels) shall either be explicitly set to

N or omitted, implying a default value of 1. The values of N that are allowed are specified in Section 4.1 in [38]. If ch-send and/or ch-recv paramaters are supplied, the number of channels N shall be the larger value given in those parameters.

- The parameters "ptime" and "maxptime" go in the SDP "a=ptime" and "a=maxptime" attributes, respectively.

- Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the media type parameter string as a semicolon-separated list of parameter=value pairs.

Mapping to fields in SDP is specified in clause 6 of [13].

A.3.3 Detailed Description of Usage of SDP Parameters

A.3.3.1 Offer-Answer Model Considerations

The following considerations apply when using SDP Offer-Answer procedures to negotiate the use of EVS payload in RTP:

dtx: When dtx is not offered, i.e., not included, for a payload type, the answerer may include dtx for the payload type in the SDP answer. When dtx is offered for a payload type and the payload type is accepted, the answerer shall not modify or remove dtx for the payload type in the SDP answer.

hf-only: When hf-only is not offered for a payload type, the answerer may include hf-only for the payload type in the SDP answer. When hf-only is offered for a payload type and the payload type is accepted, the answerer shall not modify or remove hf-only for the payload type in the SDP answer.

evs-mode-switch: When evs-mode-switch is not offered for a payload type, the answerer may include evs-mode-switch for the payload type in the SDP answer. When evs-mode-switch is offered for a payload type and the payload type is accepted, the answerer shall not modify or remove evs-mode-switch for the payload type in the SDP answer.

br: When the same bit-rate or bit-rate range is defined for the send and the receive directions, br should be used but br-send and br-recv may also be used. br can be used even if the session is negotiated to be sendonly, recvonly, or inactive. For sendonly session, br and br-send can be interchangeably used. For recvonly session, br and br-recv can be interchangeably used. When br is not offered for a payload type, the answerer may include br for the payload type in the SDP answer. When br is offered for a payload type and the payload type is accepted, the answerer shall include br in the SDP answer which shall be identical to or a subset of br for the payload type in the SDP offer.

br-send: When br-send is not offered for a payload type, the answerer may include br-recv for the payload type in the SDP answer. When br-send is offered for a payload type and the payload type is accepted, the answerer shall include br-recv in the SDP answer, and the br-recv shall be identical to or a subset of br-send for the payload type in the SDP offer.

br-recv: When br-recv is not offered for a payload type, the answerer may include br-send for the payload type in the SDP answer. When br-recv is offered for a payload type and the payload type is accepted, the answerer shall include br-send in the SDP answer, and the br-send shall be identical to or a subset of br-recv for the payload type in the SDP offer.

bw: When the same bandwidth or bandwidth range is defined for the send and the receive directions, bw should be used but bw-send and bw-recv may also be used. bw can be used even if the session is negotiated to be sendonly, recvonly, or inactive. For sendonly session, bw and bw-send can be interchangeably used. For recvonly session, bw and bw-recv can be interchangeably used. When bw is not offered for a payload type, the answerer may include bw for the payload type in the SDP answer. When bw is offered for a payload type and the payload type is accepted, the answerer shall include bw in the SDP answer, which shall be identical to or a subset of bw for the payload type in the SDP offer.

bw-send: When bw-send is not offered for a payload type, the answerer may include bw-recv for the payload type in the SDP answer. When bw-send is offered for a payload type and the payload is accepted,

the answerer shall include bw-recv in the SDP answer, and the bw-recv shall be identical to or a subset of bw-send for the payload type in the SDP offer.

bw-recv   When bw-recv is not offered for a payload type, the answerer may include bw-send for the payload type in the SDP answer. When bw-recv is offered for a payload type and the payload is accepted, the answerer shall include bw-send in the SDP answer, and the bw-send shall be identical to or a subset of bw-recv for the payload type in the SDP offer.

cmr: When cmr is not offered for a payload type, the answerer may include cmr for the payload type in the SDP answer. When cmr is offered for a payload type and the payload type is accepted, the answerer shall not modify or remove cmr for the payload type in the SDP answer.

channels   See <encoding parameters> of a=rtpmap attribute specified in RFC 4566 [27]. If ch-send and ch-recv are offered for a payload type with different numbers of channels for sending and receiving directions, channels is set to the larger of the two parameters.

ch-send: When ch-send is offered for a payload type and the payload type is accepted, the answerer shall include ch-recv in the SDP answer, and the ch-recv shall be identical to the ch-send parameter for the payload type in the SDP offer.

ch-recv   When ch-recv is offered for a payload type and the payload type is accepted, the answerer shall include ch-send in the SDP answer, and the ch-send shall be identical to the ch-recv parameter for the payload type in the SDP offer.

When a single bit-rate is offered, the answerer may accept the offered bit-rate or reject the offered bit-rate. If the offered bit-rate is accepted, this bit-rate shall be used also in the SDP answer. If the offered bit-rate is accepted but the session is changed from sendrecv to sendrecv or recvonly, the offered bit-rate shall be used in the br, br-send or br-recv parameter included in the SDP answer. Otherwise, the RTP payload type shall be rejected.

When a bit-rate range is offered, the answerer: may accept the offered bit-rate range, modify the offered bit-rate range, select a single bit-rate, or may reject the offered bit-rate range. Otherwise, the RTP payload type shall be rejected.

When an offered bit-rate range is modified for the answer, the following rules apply:

-    The lower bit-rate limit 'br1' can be kept unchanged or can be increased up to 'br2', but cannot be decreased.

-    The upper bit-rate limit 'br2' can be kept unchanged or can be decreased down to 'br1', but cannot be increased.

When an offered bit-rate range is answered with a single bit-rate, this bit-rate shall be one of the offered bit-rates.

Rejecting all RTP payload types may lead to rejecting the media type and possibly even the whole SIP INVITE.

The bit-rates and bandwidths indicated in the negotiated media type attributes shall be consistent with Table A.6. Each 'x' represents a bit-rate and bandwidth combination supported by the EVS codec.

Table A.6: Allowed bit-rates and audio bandwidths

	5.9	7.2	8	9.6	13.2	16.4	24.4	32	48	64	96	128
nb	x	x	x	x	x	x	x					
wb	x	x	x	x	x	x	x	x	x	x	x	x
swb				x	x	x	x	x	x	x	x	x
fb						x	x	x	x	x	x	x

If no bit rate parameter and no bandwidth parameter are specified, all bit-rates and bandwidths combinations as specified in Table A.6 are allowed in the session.

A.3.3.2   Examples

SDP offer/answer procedure examples for MTSI are in A.14 of [13].

Setting up a symmetric dual-mono session in both sending and receiving direction, can be done with SDP offer and SDP answer negotiating the same number of channels on the 'a=rtpmap' line in the SDP offer and SDP answer. An

example SDP offer/answer negotiation for using the same number of channels for sending and receiving directions is included below:

Example SDP offer

`m=audio 49152 RTP/AVP 96 97 98 99 100 101 102 103`
`a=rtpmap:96 EVS/16000/`**`2`**
`a=fmtp:96 br=16.4; bw=nb-swb; max-red=220`
`a=rtpmap:97 EVS/16000/`**`1`**
`a=fmtp:97 br=13.2-24.4; bw=nb-swb; max-red=220`
`a=rtpmap:98 AMR-WB/16000/`**`2`**
`a=fmtp:98 mode-change-capability=2; max-red=220`
`a=rtpmap:99 AMR-WB/16000/`**`2`**
`a=fmtp:99 mode-change-capability=2; max-red=220; octet-align=1`
`a=rtpmap:100 AMR-WB/16000/`**`1`**
`a=fmtp:100 mode-change-capability=2; max-red=220`
`a=rtpmap:101 AMR-WB/16000/`**`1`**
`a=fmtp:101 mode-change-capability=2; max-red=220; octet-align=1`
`a=rtpmap:102 AMR/8000/`**`1`**
`a=fmtp:102 mode-change-capability=2; max-red=220`
`a=rtpmap:103 AMR/8000/`**`1`**
`a=fmtp:103 mode-change-capability=2; max-red=220; octet-align=1`
`a=ptime:20`
`a=maxptime:240`

Example SDP answer

`m=audio 49152 RTP/AVP 96`
`a=rtpmap:96 EVS/16000/`**`2`**
`a=fmtp:96 br=16.4; bw=nb-swb; max-red=220`
`a=ptime:20`
`a=maxptime:240`

It is possible to use one m= line when setting up a session with equal number of channels in both directions.

Setting up a session with asymmetric number of channels for different directions is possible by negotiating different number of channels using the 'ch-send=<#>' and the 'ch-recv=#' parameters.

Annex B (informative): Change history

Date	TSG #	TSG Doc.	CR	Rev	Cat	Subject/Comment	New version
2014-09	SP-65	SP-140460				Presented to TSG SA#65 for approval	1.0.0
2014-09	SP-65					Approved at TSG SA#65	12.0.0
2014-12	SP-66	SP-140726	0001	3		Corrections to Algorithmic Description Text	12.1.0
2014-12	SP-66	SP-140726	0002	3		Incorporating RTP Payload Format and Media Type Parameters	12.1.0
2015-03	SP-67	SP-150086	0003	1		Corrections to the Algorithmic and the RTP Payload Format Descriptions	12.2.0
2015-04						Editorial Corrections (date and version number in the headings of each multi-part files)	12.2.1
2015-06	SP-68	SP-150203	0004	1		Corrections to the Algorithmic Description	12.3.0
2015-09	SP-69	SP-150434	0005	1		Corrections to the Algorithmic Description	12.4.0
2015-09	SP-69	SP-150434	0006	4		Corrrections to Payload Format Parameters	12.4.0
2015-12	SP-70	SP-150639	0007	1		Corrections to the Algorithmic Description	12.5.0
2015-12	SP-70	SP-150639	0008			Handling Received CMR	12.5.0
2015-12	SP-70					Version for Release 13	13.0.0
2016-03	SP-71	SP-160064	0013	1		Correction of mode-change-capability and channel-aware configuration	13.1.0
2016-06	72	SP-160257	0015		A	Corrections to the Algorithmic Description	13.2.0
2016-06	72	SP-160257	0017	1	A	Corrections to CMR Handling for AMR-WB IO mode	13.2.0
2016-06	72	SP-160257	0019	2	A	EVS-CMR-Only packets	13.2.0

History

Document history		
V13.0.0	February 2016	Publication
V13.1.0	May 2016	Publication
V13.2.0	August 2016	Publication