INTERNATIONAL TELECOMMUNICATION UNION

ITU-T G.722.2

(01/2002)

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Digital terminal equipments – Coding of analogue signals by methods other than PCM

Wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB)

CAUTION !

PREPUBLISHED RECOMMENDATION

This prepublication is an unedited version of a recently approved Recommendation. It will be replaced by the published version after editing. Therefore, there will be differences between this prepublication and the published version.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU [had/had not] received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

WIDEBAND CODING OF SPEECH AT AROUND 16 KBIT/S USING ADAPTIVE MULTI-RATE WIDEBAND (AMR-WB)

Summary

This Recommendation describes the high quality Adaptive Multi-Rate Wideband (AMR-WB) encoder and decoder that is primarily intended for 7 kHz bandwidth speech signals. AMR-WB operates at a multitude of bit rates ranging from 6.6 kbit/s to 23.85 kbit/s. The bit rate may be changed at any 20 ms frame boundary.

Annex C of this Recommendation includes an integrated C source code software package which contains the implementation of the G.722.2 encoder and decoder and its Annexes A and B and Appendix I. A set of digital test vectors for developers is provided in Annex D/G.722.2. These test vectors are a verification tool providing an indication of success in implementing this codec.

G.722.2 AMR-WB is the same codec as the 3GPP AMR-WB. The corresponding 3GPP specifications are TS 26.190 for the speech codec and TS 26.194 for the Voice Activity Detector.

Source

ITU-T Recommendation G.722.2 was prepared by ITU-T Study Group 16 (2001-2004) and was approved under the WTSC Resolution No. 1 procedure on 13 January 2002.

CONTENTS

Recommendation G.722.2

WIDEBAND CODING OF SPEECH AT AROUND 16 KBIT/S USING ADAPTIVE MULTI-RATE WIDEBAND (AMR-WB)

(Geneva, 2002)

1 Scope

This Recommendation describes the detailed mapping from input blocks of 320 speech samples in 16-bit uniform PCM format to encoded blocks of 132, 177, 253, 285, 317, 365, 397, 461 and 477 bits and from encoded blocks of 132, 177, 253, 285, 317, 365, 397, 461 and 477 bits to output blocks of 320 reconstructed speech samples. The sampling rate is 16 000 samples/s leading to a bit rate for the encoded bit stream of 6.60, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s. The coding scheme for the multi-rate coding modes is the so-called Algebraic Code Excited Linear Prediction Coder, hereafter referred to as ACELP. The multi-rate wideband ACELP coder is referred to as AMR-WB. The codec described in this Recommendation also utilizes an integrated Voice Activity Detector (VAD).

The foreseen applications for this Recommendation are the following: Voice over IP (VoIP) and Internet applications, Mobile Communications, PSTN applications, ISDN wideband telephony, ISDN videotelephony and video-conferencing.

In addition to the algorithm specified in the main body of Recommendation G.722.2, Annexes A and B and Appendix I provide supplemental functionalities allowing interoperability with GSM and 3GPP wireless systems. These functionalities have originally been developed for these systems, but their use is not limited to mobile applications. Two other Annexes D and E describe test vectors and frame structure respectively. These Annexes may be implemented independently of this main body specification according to the different requirements of systems deploying the AMR-WB algorithm:

- Annex A describes comfort noise aspects for use of the AMR-WB algorithm in source controlled rate operation. The implementation of this Annex is essential for interoperability with GSM and 3GPP wireless systems.
- Annex B describes the source controlled rate operation for the AMR-WB algorithm. The implementation of this Annex is essential for interoperability with GSM and 3GPP wireless systems.
- Annex D describes the digital test sequences, which are a verification tool providing an indication of success in implementing the AMR-WB codec.
- Annex E describes the recommended frame structure for use with the different modes of operation for the AMR-WB algorithm.
- Appendix I describes an example solution for error concealment of erroneous or lost AMR-WB frames.

For better usability, the ANSI-C code with the low-level description of all these functionalities have been grouped in a single Annex, Annex C. Should there be any discrepancy between the descriptions in any of the different parts of G.722.2 and the implementation of such descriptions in Annex C, the descriptions in Annex C shall prevail.

In Section 8 a specific reset procedure, called codec homing, is described. This is a useful feature for bringing the codec into a known initial state (e.g. for testing purposes). Section 9 specifies the Voice Activity Detector (VAD) used in this codec as well as in the source controlled rate operation (DTX) specified in Annex B.

2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[1] ITU-T Recommendation G.711 (1988): "Coding of analogue signals by pulse code modulation Pulse code modulation (PCM) of voice frequencies".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of this Recommendation, the following definitions apply:

adaptive codebook: The adaptive codebook contains excitation vectors that are adapted for every subframe. The adaptive codebook is derived from the long-term filter state. The lag value can be viewed as an index into the adaptive codebook.

algebraic codebook: A fixed codebook where algebraic code is used to populate the excitation vectors (innovation vectors). The excitation contains a small number of nonzero pulses with predefined interlaced sets of potential positions. The amplitudes and positions of the pulses of the k^{th} excitation codevector can be derived from its index k through a rule requiring no or minimal physical storage, in contrast with stochastic codebooks whereby the path from the index to the associated codevector involves look-up tables.

anti-sparseness processing: An adaptive post-processing procedure applied to the fixed codebook vector in order to reduce perceptual artifacts from a sparse fixed codebook vector.

closed-loop pitch analysis: This is the adaptive codebook search, i.e., a process of estimating the pitch (lag) value from the weighted input speech and the long term filter state. In the closed-loop search, the lag is searched using error minimization loop (analysis-by-synthesis). In the adaptive multi-rate wideband codec, closed-loop pitch search is performed for every subframe.

direct form coefficients: One of the formats for storing the short term filter parameters. In the adaptive multi-rate wideband codec, all filters which are used to modify speech samples use direct form coefficients.

fixed codebook: The fixed codebook contains excitation vectors for speech synthesis filters. The contents of the codebook are non-adaptive (i.e., fixed). In the adaptive multi-rate wideband codec, the fixed codebook is implemented using an algebraic codebook.

fractional lags: A set of lag values having sub-sample resolution. In the adaptive multi-rate wideband codec a sub-sample resolution of 1/4th or 1/2nd of a sample is used.

frame: A time interval equal to 20 ms (320 samples at an 16 kHz sampling rate).

Immittance Spectral Frequencies: (see Immittance Spectral Pair)

Immittance Spectral Pair: Transformation of LPC parameters. Immittance Spectral Pairs are obtained by decomposing the inverse filter transfer function A(z) to a set of two transfer functions, one having even symmetry and the other having odd symmetry. The Immittance Spectral Pairs (also called as Immittance Spectral Frequencies) are the roots of these polynomials on the z-unit circle.

integer lags: A set of lag values having whole sample resolution.

interpolating filter: An FIR filter used to produce an estimate of sub-sample resolution samples, given an input sampled with integer sample resolution. In this implementation, the interpolating filter has low pass filter characteristics. Thus the adaptive codebook consists of the low-pass filtered interpolated past excitation.

inverse filter: This filter removes the short term correlation from the speech signal. The filter models an inverse frequency response of the vocal tract.

lag: The long term filter delay. This is typically the true pitch period, or its multiple or sub-multiple.

LP analysis window: For each frame, the short term filter coefficients are computed using the high pass filtered speech samples within the analysis window. In the adaptive multi-rate wideband codec, the length of the analysis window is always 384 samples. For all the modes, a single asymmetric window is used to generate a single set of LP coefficients. The 5 ms look-ahead is used in the analysis.

LP coefficients: Linear Prediction (LP) coefficients (also referred as Linear Predictive Coding (LPC) coefficients) is a generic descriptive term for the short term filter coefficients.

mode: When used alone, refers to the source codec mode, i.e., to one of the source codecs employed in the AMR-WB codec.

open-loop pitch search: A process of estimating the near optimal lag directly from the weighted speech input. This is done to simplify the pitch analysis and confine the closed-loop pitch search to a small number of lags around the open-loop estimated lags. In the adaptive multi-rate wideband codec, an open-loop pitch search is performed in every other subframe.

residual: The output signal resulting from an inverse filtering operation.

short term synthesis filter: This filter introduces, into the excitation signal, short term correlation which models the impulse response of the vocal tract.

perceptual weighting filter: This filter is employed in the analysis-by-synthesis search of the codebooks. The filter exploits the noise masking properties of the formants (vocal tract resonances) by weighting the error less in regions near the formant frequencies and more in regions away from them.

subframe: A time interval equal to 5 ms (80 samples at 16 kHz sampling rate).

vector quantization: A method of grouping several parameters into a vector and quantizing them simultaneously.

zero input response: The output of a filter due to past inputs, i.e. due to the present state of the filter, given that an input of zeros is applied.

zero state response: The output of a filter due to the present input, given that no past inputs have been applied, i.e., given that the state information in the filter is all zeroes.

3.2 Symbols

For the purposes of this TS, the following symbols apply:

$A(z)$	The inverse filter with unquantized coefficients
$\hat{A}(z)$	The inverse filter with quantized coefficients
$H(z) = \frac{1}{\hat{A}(z)}$	The speech synthesis filter with quantized coefficients
a_i	The unquantized linear prediction parameters (direct form coefficients)
$\hat{a}_i$	The quantified linear prediction parameters
m	The order of the LP model
$W(z)$	The perceptual weighting filter (unquantized coefficients)
g_1	The perceptual weighting factor
T	The integer pitch lag nearest to the closed-loop fractional pitch lag of the subframe
β	The adaptive pre-filter coefficient (the quantified pitch gain)
$H_{h1}(z)$	Pre-processing high-pass filter
$w(n)$	LP analysis window
L_1	Length of the first part of the LP analysis window $w(n)$

L_2	Length of the second part of the LP analysis window $w(n)$
$r(k)$	The auto-correlations of the windowed speech $s'(n)$
$w_{lag}(i)$	Lag window for the auto-correlations (60 Hz bandwidth expansion)
f_0	The bandwidth expansion in Hz
f_s	The sampling frequency in Hz
$r'(k)$	The modified (bandwidth expanded) auto-correlations
$E(i)$	The prediction error in the *i*th iteration of the Levinson algorithm
k_i	The *i*th reflection coefficient
$a_j^{(i)}$	The *j*th direct form coefficient in the *i*th iteration of the Levinson algorithm
$F_1'(z)$	Symmetric ISF polynomial
$F_2'(z)$	Antisymmetric ISF polynomial
$F_1(z)$	Polynomial $F_1'(z)$
$F_2(z)$	Polynomial $F_2'(z)$ with roots $z = 1$ and $z = -1$ eliminated
q_i	The immittance spectral pairs (ISPs) in the cosine domain
$\mathbf{q}$	An ISP vector in the cosine domain
$\hat{\mathbf{q}}_i^{(n)}$	The quantified ISP vector at the *i*th subframe of the frame *n*
w_i	The immittance spectral frequencies (ISFs)
$T_m(x)$	A *m*th order Chebyshev polynomial
$f_1(i), f_2(i)$	The coefficients of the polynomials $F_1(z)$ and $F_2(z)$
$f_1'(i), f_2'(i)$	The coefficients of the polynomials $F_1'(z)$ and $F_2'(z)$
$f(i)$	The coefficients of either $F_1(z)$ or $F_2(z)$
$C(x)$	Sum polynomial of the Chebyshev polynomials
x	Cosine of angular frequency w
l_k	Recursion coefficients for the Chebyshev polynomial evaluation
f_i	The immittance spectral frequencies (ISFs) in Hz
$\mathbf{f}^t = [f_1 f_2 \ldots f_{16}]$	The vector representation of the ISFs in Hz
$\mathbf{z}(n)$	The mean-removed ISF vector at frame n

$\mathbf{r}(n)$	The ISF prediction residual vector at frame n
$\mathbf{p}(n)$	The predicted ISF vector at frame n
$\hat{\mathbf{r}}(n-1)$	The quantified residual vector at the past frame
$\hat{\mathbf{r}}_i^k$	The quantified ISF subvector i at quantization index k
d_i	The distance between the immittance spectral frequencies f_{i+1} and f_{i-1}
$h(n)$	The impulse response of the weighted synthesis filter
$H(z)W(z)$	The weighted synthesis filter
T_1	The integer nearest to the fractional pitch lag of the previous (1st or 3rd) subframe
$s'(n)$	The windowed speech signal
$s_w(n)$	The weighted speech signal
$\hat{s}(n)$	Reconstructed speech signal
$x(n)$	The target signal for adaptive codebook search
$x_2(n)$, $\mathbf{x}_2^t$	The target signal for algebraic codebook search
$res_{LP}(n)$	The LP residual signal
$c(n)$	The fixed codebook vector
$v(n)$	The adaptive codebook vector
$y(n) = v(n) * h(n)$	The filtered adaptive codebook vector
$y_k(n)$	The past filtered excitation
$u(n)$	The excitation signal
$\hat{u}'(n)$	The gain-scaled emphasized excitation signal
T_{op}	The best open-loop lag
t_{min}	Minimum lag search value
t_{max}	Maximum lag search value
$R(k)$	Correlation term to be maximized in the adaptive codebook search
$R(k)_t$	The interpolated value of $R(k)$ for the integer delay k and fraction t
A_k	Correlation term to be maximized in the algebraic codebook search at index k
C_k	The correlation in the numerator of A_k at index k
E_{Dk}	The energy in the denominator of A_k at index k

$\mathbf{d} = \mathbf{H}^t\mathbf{x}_2$	The correlation between the target signal $x_2(n)$ and the impulse response $h(n)$, i.e., backward filtered target
$\mathbf{H}$	The lower triangular Toepliz convolution matrix with diagonal $h(0)$ and lower diagonals $h(1),\ldots,h(63)$
$\mathbf{F}=\mathbf{H}^t\mathbf{H}$	The matrix of correlations of $h(n)$
$d(n)$	The elements of the vector $\mathbf{d}$
$f(i,j)$	The elements of the symmetric matrix Φ
$\mathbf{c}_k$	The innovation vector
C	The correlation in the numerator of A_k
m_i	The position of the ith pulse
J_i	The amplitude of the ith pulse
N_p	The number of pulses in the fixed codebook excitation
E_D	The energy in the denominator of A_k
$res_{LTP}(n)$	The normalized long-term prediction residual
$b(n)$	The signal used for presetting the signs in algebraic codebook search
$s_b(n)$	The sign signal for the algebraic codebook search
$d'(n)$	Sign extended backward filtered target
$f'(i,j)$	The modified elements of the matrix Φ, including sign information
$\mathbf{z}^t$, $z(n)$	The fixed codebook vector convolved with $h(n)$
$E(n)$	The mean-removed innovation energy (in dB)
$\overline{E}$	The mean of the innovation energy
$\tilde{E}(n)$	The predicted energy
$[b_1\ b_2\ b_3\ b_4]$	The MA prediction coefficients
$\hat{R}(k)$	The quantified prediction error at subframe k
E_I	The mean innovation energy
$R(n)$	The prediction error of the fixed-codebook gain quantization
E_Q	The quantization error of the fixed-codebook gain quantization
$e(n)$	The states of the synthesis filter $1/\hat{A}(z)$

$e_w(n)$	The perceptually weighted error of the analysis-by-synthesis search
h	The gain scaling factor for the emphasized excitation
g_c	The fixed-codebook gain
g_c'	The predicted fixed-codebook gain
$\hat{g}_c$	The quantified fixed codebook gain
g_p	The adaptive codebook gain
$\hat{g}_p$	The quantified adaptive codebook gain
$g_{gc} = g_c / g_c'$	A correction factor between the gain g_c and the estimated one g_c'
$\hat{g}_{gc}$	The optimum value for g_{gc}
g_{sc}	Gain scaling factor

3.3 Abbreviations

For the purposes of this Recommendation, the following abbreviations apply.

ACELP	Algebraic Code Excited Linear Prediction
AGC	Adaptive Gain Control
AMR	Adaptive Multi-Rate
AMR-WB	Adaptive Multi-Rate Wideband
CELP	Code Excited Linear Prediction
FIR	Finite Impulse Response
ISF	Immittance Spectral Frequency
ISP	Immittance Spectral Pair
ISPP	Interleaved Single-Pulse Permutation
LP	Linear Prediction
LPC	Linear Predictive Coding
LTP	Long Term Predictor (or Long Term Prediction)
MA	Moving Average
S-MSVQ	Split-MultiStage Vector Quantization
WB	Wideband

4 Outline description

This Recommendation is structured as follows:

Section 4.1 contains a functional description of the audio parts including the A/D and D/A functions. Section 4.2 describes input format for the AMR-WB encoder and the output format for the AMR-WB decoder. Sections 4.3 and 4.4 present a simplified description of the principles of the AMR-WB codec encoding and decoding process respectively. In subclause 4.5, the sequence and subjective importance of encoded parameters are given.

Section 5 presents the functional description of the AMR-WB codec encoding, whereas clause 6 describes the decoding procedures. In section 7, the detailed bit allocation of the AMR-WB codec is tabulated. Section 8 describes the homing operation.

4.1 Functional description of audio parts

The analogue-to-digital and digital-to-analogue conversion will in principle comprise the following elements:

1) Analogue to uniform digital PCM [1]

   – microphone;

   – input level adjustment device;

   – input anti-aliasing filter;

   – sample-hold device sampling at 16 kHz;

   – analogue–to–uniform digital conversion to 14–bit representation.

   The uniform format shall be represented in two's complement.

2) Uniform digital PCM to analogue [1]

   – conversion from 14–bit/16 kHz uniform PCM to analogue;

   – a hold device;

   – reconstruction filter including x/sin(x) correction;

   – output level adjustment device;

   – earphone or loudspeaker.

In the terminal equipment, the A/D function may be achieved

– by direct conversion to 14-bit uniform PCM format;

For the D/A operation, the inverse operations take place.

4.2 Preparation of speech samples

The encoder is fed with data comprising of samples with a resolution of 14 bits left justified in a 16-bit word. The decoder outputs data in the same format. Outside the speech codec further processing must be applied if the traffic data occurs in a different representation.

4.3 Principles of the adaptive multi-rate wideband speech encoder

The AMR-WB codec consists of nine source codecs with bit-rates of 23.85 23.05, 19.85, 18.25, 15.85, 14.25, 12.65, 8.85 and 6.60 kbit/s.

The codec is based on the code-excited linear predictive (CELP) coding model. The input signal is pre-emphasized using the filter $H_{pre\text{-}emph}(z)=1-\mu z^{-1}$. The CELP model is then applied to the pre-emphasized signal. A 16th order linear prediction (LP), or short-term, synthesis filter is used which is given by:

$$H(z) = \frac{1}{\hat{A}(z)} = \frac{1}{1 + \sum_{i=1}^{m} \hat{a}_i z^{-i}}, \qquad (1)$$

where $\hat{a}_i, i=1,\ldots,m$ are the (quantized) linear prediction (LP) parameters, and $m = 16$ is the predictor order. The long-term, or pitch, synthesis filter is usually given by:

$$\frac{1}{B(z)} = \frac{1}{1 - g_p z^{-T}}, \qquad (2)$$

where T is the pitch delay and g_p is the pitch gain. The pitch synthesis filter is implemented using the so-called adaptive codebook approach.

The CELP speech synthesis model is shown in Figure 1. In this model, the excitation signal at the input of the short-term LP synthesis filter is constructed by adding two excitation vectors from adaptive and fixed (innovative) codebooks. The speech is synthesized by feeding the two properly chosen vectors from these codebooks through the short-term synthesis filter. The optimum excitation sequence in a codebook is chosen using an analysis-by-synthesis search procedure in which the error between the original and synthesized speech is minimized according to a perceptually weighted distortion measure.

The perceptual weighting filter used in the analysis-by-synthesis search technique is given by:

$$W(z) = A(z/g_1)H_{de-emph}(z), \qquad (3)$$

where $A(z)$ is the unquantized LP filter, $H_{de-emph} = \frac{1}{1 - 0.68z^{-1}}$, and $g_1=0.92$ is the perceptual weighting factor. The weighting filter uses the unquantized LP parameters.

The encoder performs the analysis of the LPC, LTP and fixed codebook parameters at 12.8 kHz sampling rate. The coder operates on speech frames of 20 ms. At each frame, the speech signal is analysed to extract the parameters of the CELP model (LP filter coefficients, adaptive and fixed codebooks' indices and gains). In addition to these parameters, high-band gain indices are computed in 23.85 kbit/s mode. These parameters are encoded and transmitted. At the decoder, these parameters are decoded and speech is synthesized by filtering the reconstructed excitation signal through the LP synthesis filter.

The signal flow at the encoder is shown in Figure 2. After decimation, high-pass and pre-emphasis filtering is performed. LP analysis is performed once per frame. The set of LP parameters is converted to immittance spectrum pairs (ISP) and vector quantized using split-multistage vector quantization (S-MSVQ). The speech frame is divided into 4 subframes of 5 ms each (64 samples at 12.8 kHz sampling rate). The adaptive and fixed codebook parameters are transmitted every subframe. The quantized and unquantized LP parameters or their interpolated versions are used depending on the subframe. An open-loop pitch lag is estimated in every other subframe or once per frame based on the perceptually weighted speech signal.

Then the following operations are repeated for each subframe:

- The target signal $x(n)$ is computed by filtering the LP residual through the weighted synthesis filter $W(z)H(z)$ with the initial states of the filters having been updated by filtering the error between LP residual and excitation (this is equivalent to the common approach of subtracting the zero input response of the weighted synthesis filter from the weighted speech signal).
- The impulse response, $h(n)$ of the weighted synthesis filter is computed.
- Closed-loop pitch analysis is then performed (to find the pitch lag and gain), using the target $x(n)$ and impulse response $h(n)$, by searching around the open-loop pitch lag. Fractional pitch with 1/4th or 1/2nd of a sample resolution (depending on the mode and the pitch lag value) is used. The interpolating filter in fractional pitch search has low pass frequency response. Further, there are two potential low-pass characteristics in the the adaptive codebook and this information is encoded with 1 bit.
- The target signal $x(n)$ is updated by removing the adaptive codebook contribution (filtered adaptive codevector), and this new target, $x_2(n)$, is used in the fixed algebraic codebook search (to find the optimum innovation).
- The gains of the adaptive and fixed codebook are vector quantified with 6 or 7 bits (with moving average (MA) prediction applied to the fixed codebook gain).
- Finally, the filter memories are updated (using the determined excitation signal) for finding the target signal in the next subframe.

The bit allocation of the AMR-WB codec modes is shown in Table 1. In each 20 ms speech frame, 132, 177, 253, 285, 317, 365, 397, 461 and 477 bits are produced, corresponding to a bit-rate of 6.60, 8.85 ,12.65, 14.25, 15.85, 18.25,

19.85, 23.05 or 23.85 kbit/s. More detailed bit allocation among the codec parameters is given in tables 12a-12i. Note that the most significant bits (MSB) are always sent first.

Table 1: Bit allocation of the AMR-WB coding algorithm for 20 ms frame

Mode	Parameter	1st subframe	2nd subframe	3rd subframe	4th subframe	total per frame
23.85 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	88	88	88	88	352
	Codebook gain	7	7	7	7	28
	HB-energy	4	4	4	4	16
	Total					477
23.05 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	88	88	88	88	352
	Gains	7	7	7	7	28
	Total					461
19.85 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	72	72	72	72	288
	Codebook gain	7	7	7	7	28
	Total					397
18.25 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	64	64	64	64	256
	Gains	7	7	7	7	28
	Total					365
15.85 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	52	52	52	52	208
	Gains	7	7	7	7	28
	Total					317
14.25 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	44	44	44	44	176
	Gains	7	7	7	7	28
	Total					285
12.65 kbit/s	VAD-flag					1
	ISP					46
	LTP-filtering	1	1	1	1	4
	Pitch delay	9	6	9	6	30
	Algebraic code	36	36	36	36	144
	Gains	7	7	7	7	28
	Total					253
8.85 kbit/s	VAD-flag					1
	ISP					46
	Pitch delay	8	5	8	5	26
	Algebraic code	20	20	20	20	80
	Gains	6	6	6	6	24
	Total					177
6.60 kbit/s	VAD-flag					1
	ISP					36
	Pitch delay	8	5	5	5	23
	Algebraic code	12	12	12	12	48
	Gains	6	6	6	6	24
	Total					132

4.4 Principles of the adaptive multi-rate speech decoder

The signal flow at the decoder is shown in Figure 3. At the decoder, the transmitted indices are extracted from the received bitstream. The indices are decoded to obtain the coder parameters at each transmission frame. These parameters are the ISP vector, the 4 fractional pitch lags, the 4 LTP filtering parameters, the 4 innovative codevectors, and the 4 sets of vector quantized pitch and innovative gains. In 23.85 kbit/s mode, also high-band gain index is decoded. The ISP vector is converted to the LP filter coefficients and interpolated to obtain LP filters at each subframe. Then, at each 64-sample subframe:

- The excitation is constructed by adding the adaptive and innovative codevectors scaled by their respective gains.
- The 12.8 kHz speech is reconstructed by filtering the excitation through the LP synthesis filter.
- The reconstructed speech is de-emphasized.

Finally, the reconstructed speech is upsampled to 16 kHz and high-band speech signal is added to the frequency band from 6 kHz to 7 kHz.

4.5 Sequence and subjective importance of encoded parameters

The encoder will produce the output information in a unique sequence and format, and the decoder must receive the same information in the same way. In table 12a-12i, the sequence of output bits and the bit allocation for each parameter is shown.

The different parameters of the encoded speech and their individual bits have unequal importance with respect to subjective quality.

5 Functional description of the encoder

In this clause, the different functions of the encoder represented in Figure 2 are described.

5.1 Pre-processing

The encoder performs the analysis of the LPC, LTP and fixed codebook parameters at 12.8 kHz sampling rate. Therefore, the input signal has to be decimated from 16 kHz to 12.8 kHz. The decimation is performed by first upsampling by 4, then filtering the output through lowpass FIR filter $H_{decim}(z)$ that has the cut off frequency at 6.4 kHz. Then, the signal is downsampled by 5. The filtering delay is compensated by adding zeroes into the end of the input vector.

After the decimation, two pre-processing functions are applied to the signal prior to the encoding process: high-pass filtering and pre-emphasizing (and signal down-scaling).

(Down-scaling consists of dividing the input by a factor of 2 to reduce the possibility of overflows in the fixed-point implementation.)

The high-pass filter serves as a precaution against undesired low frequency components. A filter at a cut off frequency of 50 Hz is used, and it is given by

$$H_{h1}(z) = \frac{0.989502 - 1.979004z^{-1} + 0.989502z^{-2}}{1 - 1.978882z^{-1} + 0.979126z^{-2}}. \qquad (4)$$

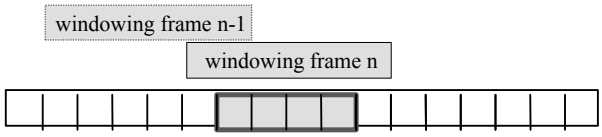
(Both down-scaling and high-pass filtering are combined by dividing the coefficients at the numerator of $H_{h1}(z)$ by 2.)

In the pre-emphasis, a first order high-pass filter is used to emphasize higher frequencies, and it is given by

$$H_{pre-emph}(z) = 1 - 0.68z^{-1} \qquad (5)$$

5.2 Linear prediction analysis and quantization

Short-term prediction, or LP, analysis is performed once per speech frame using the autocorrelation approach with 30 ms asymmetric windows. An overhead of 5 ms is used in the autocorrelation computation. The frame structure is depicted below.



frame n
(4 x 5 ms)

The autocorrelations of windowed speech are converted to the LP coefficients using the Levinson-Durbin algorithm. Then the LP coefficients are transformed to the ISP domain for quantization and interpolation purposes. The interpolated quantized and unquantized filters are converted back to the LP filter coefficients (to construct the synthesis and weighting filters at each subframe).

5.2.1 Windowing and auto-correlation computation

LP analysis is performed once per frame using an asymmetric window. The window has its weight concentrated at the fourth subframe and it consists of two parts: the first part is a half of a Hamming window and the second part is a quarter of a Hamming-cosine function cycle. The window is given by:

$$\begin{aligned} w(n) &= 0.54 - 0.46\cos\left(\frac{2\pi n}{2L_1 - 1}\right), \quad n = 0,\ldots,L_1 - 1, \\ &= \cos\left(\frac{2\pi (n - L_1)}{4L_2 - 1}\right), \quad n = L_1,\ldots,L_1 + L_2 - 1 \end{aligned} \tag{6}$$

where the values $L_1=256$ and $L_2=128$ are used.

The autocorrelations of the windowed speech $s'(n),n=0,\ldots,383$ are computed by

$$r(k) = \sum_{n=k}^{383} s'(n)s'(n-k), \quad k = 0,\ldots,16, \tag{7}$$

and a 60 Hz bandwidth expansion is used by lag windowing the autocorrelations using the window [3]

$$w_{lag}(i) = \exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right], \quad i = 1,\ldots 16, \tag{8}$$

where $f_0=60$ Hz is the bandwidth expansion and $f_s=12800$ Hz is the sampling frequency. Further, $r(0)$ is multiplied by the white noise correction factor 1.0001 which is equivalent to adding a noise floor at -40 dB.

5.2.2 Levinson-Durbin algorithm

The modified autocorrelations $r'(0) = 1.0001\, r(0)$ and $r'(k) = r(k)w_{lag}(k), k = 1,\ldots 16,$ are used to obtain the LP filter coefficients $a_k,k=1,\ldots,16$ by solving the set of equations.

$$\sum_{k=1}^{16} a_k r'(|i-k|) = -r'(i), \quad i = 1,\ldots,16. \tag{9}$$

The set of equations in (9) is solved using the Levinson-Durbin algorithm [3]. This algorithm uses the following recursion:

$E(0) = r'(0)$

For $i = 1$ to 16 do

$k_i = -\left[r'(i) + \sum_{j=1}^{i-1} a_j^{i-1} r'(i-j)\right] / E(i-1)$

$a_i^{(i)} = k_i$

For $j = 1$ to $i-1$ do

$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)}$

$E(i) = \left(1 - k_i^2\right) E(i-1)$

The final solution is given as $a_j = a_j^{(16)}$,j=1,...,16.

The LP filter coefficients are converted to the ISP representation [5] for quantization and interpolation purposes. The conversions to the ISP domain and back to the LP filter domain are described in the next two sections.

5.2.3 LP to ISP conversion

The LP filter coefficients a_k, $k=1,...,16$, are converted to the ISP representation for quantization and interpolation purposes. For a 16th order LP filter, the ISPs are defined as the roots of the sum and difference polynomials

$$f_1'(z) = A(z) + z^{-16} A(z^{-1}) \quad (10)$$

and

$$f_2'(z) = A(z) - z^{-16} A(z^{-1}) \quad (11)$$

respectively. (The polynomials $f'_1(z)$ and $f'_2(z)$ are symmetric and antisymmetric, respectively). It can be proven that all roots of these polynomials are on the unit circle and they alternate each other [6]. $f'_2(z)$ has two roots at $z = 1$ (w=0) and $z = -1$ ($w = p$). To eliminate these two roots, we define the new polynomials

$$f_1(z) = f_1'(z) \quad (12)$$

and

$$f_2(z) = f_2'(z)/(1 - z^{-2}). \quad (13)$$

Polynomials $f_1(z)$ and $f_2(z)$ have 8 and 7 conjugate roots on the unit circle $\left(e^{\pm j w_i}\right)$, respectively. Therefore, the polynomials can be written as

$$F_1(z) = (1 + a[16]) \prod_{i=0,2,\ldots,14} \left(1 - 2q_i z^{-1} + z^{-2}\right) \quad (14)$$

and

$$F_2(z) = (1 - a[16]) \prod_{i=1,3,\ldots,13} \left(1 - 2q_i z^{-1} + z^{-2}\right) \quad (15)$$

where $q_i = cos(w_i)$ with w_i being the immittance spectral frequencies (ISF) and a[16] is the last predictor coefficient. ISFs satisfy the ordering property $0 < w_1 < w_2 < \ldots < w_{16} < p$. We refer to q_i as the ISPs in the cosine domain.

Since both polynomials $f_1(z)$ and $f_2(z)$ are symmetric only the first 8 and 7 coefficients of each polynomial, respectively, and the last predictor coefficient need to be computed.

The coefficients of these polynomials are found by the recursive relations

for i=0 to 7

$$f_1(i) = a_i + a_{m-i},$$
$$f_2(i) = a_i - a_{m-i} + f_2(i-2). \qquad (16)$$
$$f_1(8) = 2a_8$$

where m=16 is the predictor order, and $f_2(-2) = f_2(-1) = 0$.

The ISPs are found by evaluating the polynomials $F_1(z)$ and $F_2(z)$ at 100 points equally spaced between 0 and p and checking for sign changes. A sign change signifies the existence of a root and the sign change interval is then divided 4 times to better track the root. The Chebyshev polynomials are used to evaluate $F_1(z)$ and $F_2(z)$ [7]. In this method the roots are found directly in the cosine domain $\{q_i\}$. The polynomials $F_1(z)$ and $F_2(z)$ evaluated at $z = e^{jw}$ can be written as

$$F_1(w) = 2e^{-j8w}C_1(x) \quad \text{and} \quad F_2(w) = 2e^{-j7w}C_2(x) \qquad (17)$$

with

$$C_1(x) = \sum_{i=0}^{7} f_1(i)T_{8-i}(x) + f_1(8)/2, \text{ and } C_2(x) = \sum_{i=0}^{6} f_2(i)T_{8-i}(x) + f_2(7)/2, \qquad (18)$$

where T_m=cos(mw) is the mth order Chebyshev polynomial, $f(i)$ are the coefficients of either $F_1(z)$ or $F_2(z)$, computed using the equations in (16). The polynomial $C(x)$ is evaluated at a certain value of x = cos(w) using the recursive relation:

$$\begin{aligned} &for\ k = n_f - 1\ \ down\ to\ 1 \\ &\quad b_k = 2xb_{k+1} - b_{k+2} + f(n_f - k) \\ &end \\ &C(x) = xb_1 - b_2 + f(n_f)/2, \end{aligned}$$

where n_f=8 in case of $C_1(x)$ and n_f=7 in case of $C_2(x)$, with initial values b_{nf}=f(0) and b_{nf+1}=0. The details of the Chebyshev polynomial evaluation method are found in [7].

5.2.4 ISP to LP conversion

Once the ISPs are quantized and interpolated, they are converted back to the LP coefficient domain $\{a_k\}$. The conversion to the LP domain is done as follows. The coefficients of $F_1(z)$ and $F_2(z)$ are found by expanding Equations (14) and (15) knowing the quantized and interpolated ISPs q_i=,i=0,…,m-1, where m=16. The following recursive relation is used to compute $f_1(z)$

```
for i = 2 to m/2
    f1(i) = -2q_{2i-2} f1(i-1) + 2f1(i-2)
    for j = i-1 down to 2
        f1(j) = f1(j) - 2q_{2i-2} f1(j-1) + f1(j-2)
    end
    f1(1) = f1(1) - 2q_{2i-2}
end
```

with initial values $f_1(0)$=1 and $f_1(1)$=-2q_0. The coefficients $f_2(i)$ are computed similarly by replacing q_{2i-2} by q_{2i-1} and m/2 by m/2-1, and with intial conditions $f_2(0)$=1 and $f_2(1)$=-2q_1.

Once the coefficients $f_1(z)$ and $f_2(z)$ are found, $F_2(z)$ is multiplied by 1-z^{-2}, to obtain $F'_2(z)$; that is

$$f_2'(i) = f_2(i) - f_2(i-2), \quad i = 2,\ldots,m/2-1,$$
$$f_1'(i) = f_1(i) \quad i = 0,\ldots,m/2 \qquad (19)$$

Then $F'_1(z)$ and $F'_2(z)$ are multiplied by $1+q_{m-1}$ and $1-q_{m-1}$, respectively. That is

$$f_2^{'}(i) = (1 - q_{m-1}) f'_2(i), \quad i = 0,\ldots,m/2-1,$$
$$f_1^{'}(i) = (1 + q_{m-1}) f'_1(i) \quad i = 0,\ldots,m/2$$

Finally the LP coefficients are found by

$$a_i = \begin{cases} 0.5 f_1^{'}(i) + 0.5 f_2^{'}(i), & i = 1,\ldots,m/2-1, \\ 0.5 f_1^{'}(i) - 0.5 f_2^{'}(i), & i = m/2+1,\ldots,m-1, \\ 0.5 f_1^{'}(m/2), & i = m/2, \\ q_{m-1}, & i = m. \end{cases} \qquad (20)$$

This is directly derived from the relation $A(z) = (F_1^{'}(z) + F_2^{'}(z))/2$, and considering the fact that $F'_1(z)$ and $F'_2(z)$ are symmetric and antisymmetric polynomials, respectively.

5.2.5 Quantization of the ISP coefficients

The LP filter coefficients are quantized using the ISP representation in the frequency domain; that is

$$f_i = \begin{cases} \frac{f_S}{2p}\arccos(q_i), & i = 0,\ldots 14, \\ \frac{f_S}{4p}\arccos(q_i), & i = 15, \end{cases} \qquad (21)$$

where f_i are the ISFs in Hz [0,6400] and f_s=12800 is the sampling frequency. The ISF vector is given by $\mathbf{f}^t = [f_0 f_1,\ldots,f_{15}]$, with t denoting transpose.

A 1st order MA prediction is applied, and the residual ISF vector is quantified using a combination of split vector quantization (SVQ) and multi-stage vector quantization (MSVQ). The prediction and quantization are performed as follows. Let $\mathbf{z}(n)$ denote the mean-removed ISF vector at frame n. The prediction residual vector $\mathbf{r}(n)$ is given by:

$$\mathbf{r}(n) = \mathbf{z}(n) - \mathbf{p}(n) \qquad (22)$$

where $\mathbf{p}(n)$ is the predicted LSF vector at frame n. First order moving-average (MA) prediction is used where:

$$\mathbf{p}(n) = \frac{1}{3}\hat{\mathbf{r}}(n-1), \qquad (23)$$

where $\hat{\mathbf{r}}(n-1)$ is the quantized residual vector at the past frame.

The ISF residual vector $\mathbf{r}$ is quantized using split-multistage vector quantization S-MSVQ. The vector is split into 2 subvectors $\mathbf{r}_1(n)$ and $\mathbf{r}_2(n)$ of dimensions 9 and 7, respectively. The 2 subvectors are quantized in two stages. In the first stage $\mathbf{r}_1(n)$ is quantized with 8 bits and $\mathbf{r}_2(n)$ with 8 bits.

For 8.85 ,12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes, the quantization error vectors $\mathbf{r}_i^{(2)} = \mathbf{r}_i - \hat{\mathbf{r}}_i, i = 1,2$ are split in the next stage into 3 and 2 subvectors, respectively. The subvectors are quantized using the bit-rates described in Table 2.

Table 2. Quantization of ISP vector for the 8.85 ,12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes

1. UNQUANTIZED 16-ELEMENT-LONG ISP VECTOR				
2. STAGE 1 ($\mathbf{r}_1$) 8 bits			**2. STAGE 1 ($\mathbf{r}_2$) 8 bits**	
3. STAGE 2 ($\mathbf{r}^{(2)}_{1,0-2}$) 6 bits	3. STAGE 2 ($\mathbf{r}^{(2)}_{1,3-5}$) 7 bits	3. STAGE 2 ($\mathbf{r}^{(2)}_{1,6-8}$) 7 bits	3. STAGE 2 ($\mathbf{r}^{(2)}_{2,0-2}$) 5 bits	3. STAGE 2 ($\mathbf{r}^{(2)}_{2,3-6}$) 5 bits

For 6.60 kbit/s mode, the quantization error vectors $\mathbf{r}_i^{(2)} = \mathbf{r}_i - \hat{\mathbf{r}}_i, i = 1,2$ are split in the next stage into 2 and 1 subvectors, respectively. The subvectors are quantized using the bit-rates described in Table 3.

Table 3. Quantization of ISP vector for the 6.60 kbit/s mode

1. UNQUANTIZED 16-ELEMENT-LONG ISP VECTOR		
2. STAGE 1 ($\mathbf{r}_1$) 8 bits		**2. STAGE 1 ($\mathbf{r}_2$) 8 bits**
3. STAGE 2 ($\mathbf{r}^{(2)}_{1,0-4}$) 7 bits	3. STAGE 2 ($\mathbf{r}^{(2)}_{1,5-8}$) 7 bits	3. STAGE 2 ($\mathbf{r}^{(2)}_{2,0-6}$) 6 bits

A squared error ISP distortion measure is used in the quantization process. In general, for an input ISP or error residual subvector $\mathbf{r}_i$,i=1,2 and a quantized vector at index k, $\hat{\mathbf{r}}_i^k$, the quantization is performed by finding the index k which minimizes

$$E = \sum_{i=m}^{n} \left[r_i - \hat{r}_i^k \right]^2 , \qquad (24)$$

where m and n are the first and last elements of the subvector.

5.2.6 Interpolation of the ISPs

The set of quantized (and unquantized) LP parameters is used for the fourth subframe whereas the first, second, and third 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 $\hat{\mathbf{q}}_4^{(n)}$ be the ISP vector at the 4th subframe of the frame, and $\hat{\mathbf{q}}_4^{(n-1)}$ the ISP vector at the 4th subframe of the past frame n-1. The interpolated ISP vectors at the 1st, 2nd, and 3rd subframes are given by

$$\hat{\mathbf{q}}_1^{(n)} = 0.55\hat{\mathbf{q}}_4^{(n-1)} + 0.45\hat{\mathbf{q}}_4^{(n)},$$
$$\hat{\mathbf{q}}_2^{(n)} = 0.2\hat{\mathbf{q}}_4^{(n-1)} + 0.8\hat{\mathbf{q}}_4^{(n)},$$
$$\hat{\mathbf{q}}_3^{(n)} = 0.04\hat{\mathbf{q}}_4^{(n-1)} + 0.96\hat{\mathbf{q}}_4^{(n)}.$$

The same formula is used for interpolation of the 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 Section 5.2.4.

5.3 Perceptual weighting

The traditional perceptual weighting filter $W(z) = A(z/g_1)/A(z/g_2)$ 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 the preemphasis filter at the input, compute the LP filter $A(z)$ based on the preemphasized speech $\boldsymbol{s}(n)$, and use a modified filter $W(z)$ by fixing its denominator. This structure substantially decouples the formant weighting from the tilt.

A weighting filter of the form $W(z) = A(z/g_1)H_{de-emph}(z)$ is used, where $H_{de-emph} = \frac{1}{1 - b_1 z^{-1}}$ and b_1=0.68.

Because $A(z)$ is computed based on the preemphasized speech signal $s(n)$, the tilt of the filter $1/A(z/g_1)$ is less pronounced compared to the case when $A(z)$ is computed based on the original speech. Since deemphasis is performed at the decoder end, it can be shown that the quantization error spectrum is shaped by a filter having a transfer function $W^{-1}(z)H_{de\text{-}emph}(z)=1/A(z/g_1)$. Thus, the spectrum of the quantization error is shaped by a filter whose transfer function is $1/A(z/g_1)$, with $A(z)$ computed based on the preemphasized speech signal.

5.4 Open-loop pitch analysis

Depending on the mode, open-loop pitch analysis is performed once per frame (each 10 ms) or twice per frame (each 10 ms) to find two estimates of the pitch lag in each frame. This is done in order to simplify the pitch analysis and confine the closed loop pitch search to a small number of lags around the open-loop estimated lags.

Open-loop pitch estimation is based on the weighted speech signal $s_w(n)$ which is obtained by filtering the input speech signal through the weighting filter $W(z) = A(z/g_1)H_{de-emph}(z)$, where $H_{de-emph} = \frac{1}{1 - b_1 z^{-1}}$ and b_1=0.68. That is, in a subframe of size L, the weighted speech is given by

$$s_w(n) = s(n) + \sum_{i=1}^{16} a_i g_1^i s(n-i) + b_1 s_w(n-1), n = 0,...,L-1. \quad (25)$$

The open-loop pitch analysis is performed to a signal decimated by two. The decimated signal is obtained by filtering $s_w(n)$ through a fourth order FIR filter $\mathrm{H}_{\mathrm{decim2}}(z)$ and then downsampling the output by two to obtain the signal $s_{wd}(n)$.

5.4.1 6.60 kbit/s mode

Open-loop pitch analysis is performed once per frame (every 20 ms) to find an estimate of the pitch lag in each frame.

The open-loop pitch analysis is performed as follows. First, the correlation of decimated weighted speech is determined for each pitch lag value d by:

$$C(d) = \sum_{n=0}^{128} s_{wd}(n) s_{wd}(n-d) w(d), d = 17,\ldots,115, \quad (26)$$

where $w(d)$ is a weighting function. The estimated pitch-lag is the delay that maximises the weighted correlation function $C(d)$. The weighting emphasises lower pitch lag values reducing the likelihood of selecting a multiple of the correct delay. The weighting function consists of two parts: a low pitch lag emphasis function, $w_l(d)$, and a previous frame lag neighbouring emphasis function, $w_n(d)$:

$$w(d) = w_l(d) w_n(d). \quad (27)$$

The low pitch lag emphasis function is a given by:

$$w_l(d) = cw(d) \quad (28)$$

where $cw(d)$ is defined by a table in the fixed point computational description. The previous frame lag neighbouring emphasis function depends on the pitch lag of previous speech frames:

$$w_n(d) = \begin{cases} cw(|T_{old} - d| + 98), & v > 0.8, \\ 1.0, & \text{otherwise}, \end{cases} \quad (29)$$

where T_{old} is the median filtered pitch lag of 5 previous voiced speech half-frames and v is an adaptive parameter. If the frame is classified as voiced by having the open-loop gain g>0.6, then the v-value is set to 1.0 for the next frame. Otherwise, the v-value is updated by v=0.9v. The open loop gain is given by:

$$g = \frac{\sum_{n=0}^{127} s_{wd}(n) s_{wd}(n - d_{\max})}{\sqrt{\sum_{n=0}^{127} s_{wd}^2(n) \sum_{n=0}^{127} s_{wd}^2(n - d_{\max})}} \quad (30)$$

where d_{max} is the pitch delay that maximizes $C(d)$. The median filter is updated only during voiced speech frames. The weighting depends on the reliability of the old pitch lags. If previous frames have contained unvoiced speech or silence, the weighting is attenuated through the parameter v.

5.4.2 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 and 23.85 kbit/s modes

Open-loop pitch analysis is performed twice per frame (every 10 ms) to find two estimates of the pitch lag in each frame.

The open-loop pitch analysis is performed as follows. First, the correlation of decimated weighted speech is determined for each pitch lag value d by:

$$C(d) = \sum_{n=0}^{63} s_{wd}(n) s_{wd}(n - d) w(d), d = 17, \ldots, 115, \quad (31)$$

where $w(d)$ is a weighting function. The estimated pitch-lag is the delay that maximises the weighted correlation function $C(d)$. The weighting emphasises lower pitch lag values reducing the likelihood of selecting a multiple of the correct delay. The weighting function consists of two parts: a low pitch lag emphasis function, $w_l(d)$, and a previous frame lag neighbouring emphasis function, $w_n(d)$:

$$w(d) = w_l(d) w_n(d). \quad (32)$$

The low pitch lag emphasis function is given by:

$$w_l(d) = cw(d) \quad (33)$$

where $cw(d)$ is defined by a table in the fixed point computational description. The previous frame lag neighbouring emphasis function depends on the pitch lag of previous speech frames:

$$w_n(d) = \begin{cases} cw(|T_{old} - d| + 98), & v > 0.8, \\ 1.0, & \text{otherwise}, \end{cases} \quad (34)$$

where T_{old} is the median filtered pitch lag of 5 previous voiced speech half-frames and v is an adaptive parameter. If the frame is classified as voiced by having the open-loop gain g>0.6, then the v-value is set to 1.0 for the next frame. Otherwise, the v-value is updated by v=0.9v. The open loop gain is given by:

$$g = \frac{\sum_{n=0}^{63} s_{wd}(n) s_{wd}(n - d_{\max})}{\sqrt{\sum_{n=0}^{63} s_{wd}^2(n) \sum_{n=0}^{63} s_{wd}^2(n - d_{\max})}} \quad (35)$$

where d_{max} is the pitch delay that maximizes $C(d)$. The median filter is updated only during voiced speech frames. The weighting depends on the reliability of the old pitch lags. If previous frames have contained unvoiced speech or silence, the weighting is attenuated through the parameter v.

5.5 Impulse response computation

The impulse response, *h(n)*, of the weighted synthesis filter $H(z)W(z) = A(z/g_1)H_{de-emph}(z)/\hat{A}(z)$ is computed each subframe. This impulse response is needed for the search of adaptive and fixed codebooks. The impulse response *h(n)* is computed by filtering the vector of coefficients of the filter $A(z/g_1)$ extended by zeros through the two filters $1/\hat{A}(z)$ and $H_{de-emph}(z)$.

5.6 Target signal computation

The target signal for adaptive codebook search is usually computed by subtracting the zero-input response of the weighted synthesis filter $H(z)W(z) = A(z/g_1)H_{de-emph}(z)/\hat{A}(z)$ from the weighted speech signal $s_w(n)$. This is performed on a subframe basis.

An equivalent procedure for computing the target signal, which is used in this codec, is the filtering of the LP residual signal *r(n)* through the combination of synthesis filter $1/\hat{A}(z)$ and the weighting filter $A(z/g_1)H_{de-emph}(z)$. After determining the excitation for the subframe, the initial states of these filters are updated by filtering the difference between the LP residual and excitation. The memory update of these filters is explained in Section 5.10.

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 section. The LP residual is given by

$$r(n) = s(n) + \sum_{i=1}^{16} \hat{a}_i s(n-i), n = 0,...,63. \qquad (36)$$

5.7 Adaptive codebook

Adaptive codebook search is performed on a subframe basis. It consists of performing closed loop pitch search, and then computing the adaptive codevector by interpolating the past excitation at the selected fractional pitch lag.

The adaptive codebook parameters (or pitch parameters) are the delay and gain of the pitch filter. In the search stage, the excitation is extended by the LP residual to simplify the closed-loop search.

In 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes, in the first and third subframes, a fractional pitch delay is used with resolutions 1/4 in the range[34, $127\frac{3}{4}$], resolutions 1/2 in the range [128, $159\frac{1}{2}$], and integers only in the range [160, 231]. For the second and fourth subframes, a pitch resolution of 1/4 is always used in the range [T_1-8, T_1+$7\frac{3}{4}$], where T_1 is nearest integer to the fractional pitch lag of the previous (1st or 3rd) subframe.

In 8.85 kbit/s mode, in the first and third subframes, a fractional pitch delay is used with resolutions 1/2 in the range [34, $91\frac{1}{2}$], and integers only in the range [92, 231]. For the second and fourth subframes, a pitch resolution of 1/2 is always used in the range [T_1-8, T_1+$7\frac{1}{2}$], where T_1 is nearest integer to the fractional pitch lag of the previous (1st or 3rd) subframe.

In 6.60 kbit/s mode, in the first subframe, a fractional pitch delay is used with resolutions 1/2 in the range [34,$91\frac{1}{2}$], and integers only in the range [92, 231]. For the second, third and fourth subframes, a pitch resolution of 1/2 is always used in the range [T_1-8, T_1+$7\frac{1}{2}$], where T_1 is nearest integer to the fractional pitch lag of the first subframe.

Closed-loop pitch analysis is performed around the open-loop pitch estimates on a subframe basis. In 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes, in the first (and third) subframe the range T_{op}±7, bounded by 34...231, is searched. In 6.60 kbit/s mode, in the first subframe the range T_{op}±7, bounded by 34...231, is searched. For all the modes, for the other subframes, closed-loop pitch analysis is performed around the integer pitch selected in the

previous subframe, as described above. In 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes, the pitch delay is encoded with 9 bits in the first and third subframes and the relative delay of the other subframes is encoded with 6 bits. In 8.85 kbit/s mode, the pitch delay is encoded with 8 bits in the first and third subframes and the relative delay of the other subframes is encoded with 5 bits. In 6.60 kbit/s mode, the pitch delay is encoded with 8 bits in the first subframe and the relative delay of the other subframes is encoded with 5 bits.

The closed loop pitch search is performed by minimizing the mean-square weighted error between the original and synthesized speech. This is achieved by maximizing the term

$$T_k = \frac{\sum_{n=0}^{63} x(n) y_k(n)}{\sqrt{\sum_{n=0}^{63} y_k(n) y_k(n)}}, \qquad (37)$$

where $x(n)$ is the target signal and $y_k(n)$ is the past filtered excitation at delay k (past excitation convolved with $h(n)$). Note that the search range is limited around the open-loop pitch as explained earlier.

The convolution $y_k(n)$ is computed for the first delay in the searched range, and for the other delays, it is updated using the recursive relation

$$y_k(n) = y_{k-1}(n-1) + u(-k)h(n) \qquad (38)$$

where u(n),n=–(231+17),…,63, is the excitation buffer. Note that in search stage, the samples $u(n), n = 0,\ldots,63$, are not known, and they are needed for pitch delays less than 64. To simplify the search, the LP residual is copied to $u(n)$ in order to make the relation in Equation (38) valid for all delays.

Once the optimum integer pitch delay is determined, the fractions from $\frac{-3}{4}$ to $\frac{3}{4}$ with a step of $\frac{1}{4}$ around that integer are tested. The fractional pitch search is performed by interpolating the normalized correlation in Equation (37) and searching for its maximum. Once the fractional pitch lag is determined, $v'(n)$ is computed by interpolating the past excitation signal $u(n)$ at the given phase (fraction). (The interpolation is performed using two FIR filters (Hamming windowed sinc functions); one for interpolating the term in Equation (34) with the sinc truncated at ±17 and the other for interpolating the past excitation with the sinc truncated at ±63). The filters have their cut-off frequency (-3 dB) at 6000 Hz in the oversampled domain, which means that the interpolation filters exhibit low-pass frequency response Thus, even when the pitch delay is an integer value, the adaptive codebook excitation consists of a low-pass filtered version of the past excitation at the given delay and not a direct copy thereof. Further, for delays smaller than the subframe size, the adaptive codebook excitation is completed based on the low-pass filtered interpolated past excitation and not by repeating the past excitation.

In order to enhance the pitch prediction performance in wideband signals, a frequency-dependant pitch predictor is used. This is important in wideband signals since the periodicity doesn't necessarily extend over the whole spectrum. In this algorithm, there are two signal paths associated to respective sets of pitch codebook parameters, wherein each signal path comprises a pitch prediction error calculating device for calculating a pitch prediction error of a pitch codevector from a pitch codebook search device. One of these two paths comprises a low-pass filter for filtering the pitch codevector and the pitch prediction error is calculated for these two signal paths. The signal path having the lowest calculated pitch prediction error is selected, along with the associated pitch gain.

The low pass filter used in the second path is in the form $B_{LP}(z)$=0.18z+0.64+0.18z^{-1}. Note that 1 bit is used to encode the chosen path.

Thus, for 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes, there are two possibilities to generate the adaptive codebook $v(n)$, $v(n) = v'(n)$ in the first path, or $v(n) = \sum_{i=-1}^{1} b_{LP}(i+1)v'(n+i)$ in the second path, where $\mathbf{b}_{LP}$=[0.18,0.64,0.18]. The path which results in minimum energy of the target signal $x_2(n)$ defined in Equation (40) is selected for the filtered adaptive codebook vector. For 6.60 kbit/s mode, $v(n)$ is always $v(n) = \sum_{i=-1}^{1} b_{LP}(i+1)v'(n+i)$.

The adaptive codebook gain is then found by

$$g_p = \frac{\sum_{n=0}^{63} x(n)y(n)}{\sum_{n=0}^{63} y(n)y(n)}, \qquad \text{bounded by} \quad 0 \le g_p \le 1.2, \tag{39}$$

where $y(n) = v(n) * h(n)$ is the filtered adaptive codebook vector (zero-state response of $H(z)W(z)$ to $v_i(n)$). To insure stability, the adaptive codebook gain g_p is bounded by 0.95, if the adaptive codebook gains of the previous subframes have been small and the LP filters of the previous subframes have been close to being unstable.

5.8 Algebraic codebook

5.8.1 Codebook structure

The codebook structure is based on interleaved single-pulse permutation (ISPP) design. The 64 positions in the codevector are divided into 4 tracks of interleaved positions, with 16 positions in each track. The different codebooks at the different rates are constructed by placing a certain number of signed pulses in the tracks (from 1 to 6 pulses per track). The codebook index, or codeword, represents the pulse positions and signs in each track. Thus, no codebook storage is needed, since the excitation vector at the decoder can be constructed through the information contained in the index itself (no lookup tables).

An important feature of the used codebook is that it is a dynamic codebook consisting of an algebraic codebook followed by an adaptive prefilter $F(z)$ which enhances special spectral components in order to improve the synthesis speech quality. A prefilter relevant to wideband signals is used whereby $F(z)$ consists of two parts: a periodicity enhancement part $1/(1-0.85z^{-T})$ and a tilt part $(1 - b_1 z^{-1})$, where T is the integer part of the pitch lag and b_1 is related to the voicing of the previous subframe and is bounded by [0.0,0.5]. The codebook search is performed in the algebraic domain by combining the filter $F(z)$ with the weighed synthesis filter prior to the coddedbook search. Thus, the impulse response $h(n)$ must be modified to include the prefilter $F(z)$. That is, $h(n) \leftarrow h(n) * f(n)$.

The codebook structures of different bit rates are given below.

5.8.1.1 23.85 and 23.05 kbit/s mode

In this codebook, the innovation vector contains 24 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 six pulses, as shown in Table 4.

Table 4. Potential positions of individual pulses in the algebraic codebook, 23.85 and 23.05 kbit/s.

Track	Pulse	Positions
1	$i_0, i_4, i_8, i_{12}, i_{16}, i_{20}$	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	$i_1, i_5, i_9, i_{13}, i_{17}, i_{21}$	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	$i_2, i_6, i_{10}, i_{14}, i_{18}, i_{22}$	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	$i_3, i_7, i_{11}, i_{15}, i_{19}, i_{23}$	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

The six pulses in one track are encoded with 22 bits.

This gives a total of 88 bits (22+22+22+22) for the algebraic code.

5.8.1.2 19.85 kbit/s mode

In this codebook, the innovation vector contains 18 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 of the first two tracks contains five pulses and each of the other tracks contains four pulses, as shown in Table 5.

Table 5. Potential positions of individual pulses in the algebraic codebook, 19.85 kbit/s.

Track	Pulse	Positions
1	$i_0, i_4, i_8, i_{12}, i_{16}$	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	$i_1, i_5, i_9, i_{13}, i_{17}$	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2, i_6, i_{10}, i_{14}	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3, i_7, i_{11}, i_{15}	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

The five pulses in one track are encoded with 20 bits. The four pulses in one track is encoded with 16 bits.

This gives a total of 72 bits (20+20+16+16) for the algebraic code.

5.8.1.3 18.25 kbit/s mode

In this codebook, the innovation vector contains 16 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 four pulses, as shown in Table 6.

Table6. Potential positions of individual pulses in the algebraic codebook, 18.25 kbit/s.

Track	Pulse	Positions
1	i_0, i_4, i_8, i_{12}	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	i_1, i_5, i_9, i_{13}	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2, i_6, i_{10}, i_{14}	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3, i_7, i_{11}, i_{15}	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

The four pulses in one track are encoded with 16 bits.

This gives a total of 64 bits (16+16+16+16) for the algebraic code.

5.8.1.4 15.85 kbit/s mode

In this codebook, the innovation vector contains 12 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 three pulses, as shown in Table 7.

Table 7. Potential positions of individual pulses in the algebraic codebook, 15.85 kbit/s.

Track	Pulse	Positions
1	i_0, i_4, i_8	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	i_1, i_5, i_9	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2, i_6, i_{10}	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3, i_7, i_{11}	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

The three pulses in one track are encoded with 13 bits.

This gives a total of 52 bits (13+13+13+13) for the algebraic code.

5.8.1.5 14.25 kbit/s mode

In this codebook, the innovation vector contains 10 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 two or three pulses, as shown in Table 8.

Table 8. Potential positions of individual pulses in the algebraic codebook, 14.25 kbit/s.

Track	Pulse	Positions
1	i_0, i_4, i_8	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	i_1, i_5, i_9	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2, i_6	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3, i_7	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

Each two pulse positions in one track are encoded with 8 bits (4 bits for the position of every pulse), and the sign of the first pulse in the track is encoded with 1 bit.

The three pulse in one track are encoded with 13 bits.

This gives a total of 44 bits (13+13+9+9) for the algebraic code.

5.8.1.6 12.65 kbit/s mode

In this codebook, the innovation vector contains 8 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 two pulses, as shown in Table 9.

Table 9. Potential positions of individual pulses in the algebraic codebook, 12.65 kbit/s.

Track	Pulse	Positions
1	i_0, i_4	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	i_1, i_5	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2, i_6	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3, i_7	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

Each two pulse positions in one track are encoded with 8 bits (total of 32 bits, 4 bits for the position of every pulse), and the sign of the first pulse in the track is encoded with 1 bit (total of 4 bits). This gives a total of 36 bits for the algebraic code.

5.8.1.7 8.85 kbit/s mode

In this codebook, the innovation vector 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 10.

Table 10. Potential positions of individual pulses in the algebraic codebook, 8.85 kbit/s.

Track	Pulse	Positions
1	i_0	0, 4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60
2	i_1	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	i_2	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	i_3	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

Each pulse position in one track are 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.

5.8.1.8 6.60 kbit/s mode

In this codebook, the innovation vector contains 2 non-zero pulses. All pulses can have the amplitudes +1 or -1. The 64 positions in a subframe are divided into 2 tracks, where each track contains one pulse, as shown in Table 11.

Table 11. Potential positions of individual pulses in the algebraic codebook, 6.60 kbit/s.

Track	Pulse	Positions
1	i_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	i_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 are 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.

5.8.2 Pulse indexing

In the above section, the number of bits needed to encode a number of pulses in a track was given. In this section, the procedures used for encoding from 1 to 6 pulses per track will be described. The description will be given for the case

of 4 tracks per subframe, with 16 positions per track and pulse spacing of 4 (which is the case for all modes except the 6.6 kbit/s mode).

Encoding 1 signed pulse per track

The pulse position index is encoded with 4 bits and the sign index with 1 bit. The position index is given by the pulse position in the subframe divided by the pulse spacing (integer division). The division remainder gives the track index. For example, a pulse at position 31 has a position index of 31/4 = 7 and it belong to the track with index 3 (4^{th} track).

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

$$I_{1p} = p + s \times 2^{M}$$

where p is the position index, s is the sign index, and M=4 is the number of bits per track.

Encoding 2 signed pulses per track

In case of two pulses per track of $K=2^{M}$ potential positions (here M=4), each pulse needs 1 bit for the sign and M bits for the position, which gives a total of $2M+2$ bits. However, some redundancy exists due to the unimportance of the pulse ordering. For example, placing the first pulse at position p and the second pulse at position q is equivalent to placing the first pulse at position q and the second pulse at position p. One bit can be saved by encoding only one sign and deducing the second sign from the ordering of the positions in the index. Here the index is given by

$$I_{2p} = p_1 + p_0 \times 2^{M} + s \times 2^{2M}$$

where s is the sign index of the pulse at position index p_0. If the two signs are equal then the smaller position is set to p_0 and the larger position is set to p_1. On the other hand, of the two signs are not equal then the larger position is set to p_0 and the smaller position is set to p_1. At the decoder, the sign of the pulse at position p_0 is readily available. The second sign is deduced from the pulse ordering. If p_0 is larger than p_1 then the sign of the pulse at position p_1 is opposite to that at position p_0. If this is not the case then the two signs are set equal

Encoding 3 signed pulses per track

In case of three pulses per track, similar logic can be used as in the case of two pulses. For a track with 2^{M} positions, 3M+1 bits are needed instead of 3M+3 bits. A simple way of indexing the pulses is to divide the track positions in two sections (or halves) and identify a section that contains at least two pulses. The number of positions in the section is $K/2 = 2^{M}/2 = 2^{M-1}$, which can be represented with M-1 bits. The two pulses in the section containing at least two pulses are encoded with the procedure for encoding 2 signed pulses which requires 2(M-1)+1 bits and the remaining pulse which can be anywhere in the track (in either section) is encoded with the M+1 bits. Finally, the index of the section that contains the two pulses is encoded with 1 bit. Thus the total number of required bits is 2(M-1)+1 + M+1 + 1 = 3M+1.

A simple way of checking if two pulses are positioned in the same section is done by checking whether the most significant bits (MSB) of their position indices are equal or not. Note that a MSB of 0 means that the position belongs to the lower half of the track (0-7) and MSB of 1 means it belongs to the upper half (8-15). If the two pulses belong to the upper half, they need to be shifted to the range (0-7) before encoding them using 2×3+1 bits. This can be done by masking the M-1 least significant bits (LSB) with a mask consisting of M-1 ones (which corresponds to the number 7 in this case).

The index of the 3 signed pulses is given by

$$I_{3p} = I_{2p} + k \times 2^{2M-1} + I_{1p} \times 2^{2M}$$

where I_{2p} is the index of the two pulses in the same section, k is the section index (0 or 1), and I_{1p} is the index of the third pulse in the track.

Encoding 4 signed pulses per track

The 4 signed pulses in a track of length $K=2^{M}$ can be encoded using $4M$ bits. Similar to the case of 3 pulses, the K positions in the track are divided into 2 sections (two halves) where each section contains $K/2=8$ positions. Here we denote the sections as Section A with positions 0 to $K/2$-1 and Section B with positions $K/2$ to K-1. Each section can contain from 0 to 4 pulses. The table below shows the 5 cases representing the possible number of pulses in each section:

case	*Pulses in Section A*	*Pulses in Section B*	*Bits needed*
0	0	4	4*M*-3
1	1	3	4*M*-2
2	2	2	4*M*-2
3	3	1	4*M*-2
4	4	0	4*M*-3

In cases 0 or 4, the 4 pulses in a section of length $K/2=2^{M-1}$ can be encoded using $4(M-1)+1=4M-3$ bits (this will be explained later on).

In cases 1 or 3, the 1 pulse in a section of length $K/2=2^{M-1}$ can be encoded with $M-1+1 = M$ bits and the 3 pulses in the other section can be encoded with $3(M-1)+1 = 3M-2$ bits. This gives a total of $M+3M-2 = 4M-2$ bits.

In case 2, the pulses in a section of length $K/2=2^{M-1}$ can be encoded with $2(M-1)+1 = 2M-1$ bits. Thus for both sections, $2(2M-1) = 4M-2$ bits are required.

Now the case index can be encoded with 2 bits (4 possible cases) assuming cases 0 and 4 are combined. Then for cases 1, 2, or 3, the number of needed bits is $4M-2$. This gives a total of $4M-2 + 2 = 4M$ bits. For cases 0 or 4, one bit is needed for identifying either case, and $4M-3$ bits are needed for encoding the 4 pulses in the section. Adding the 2 bits needed for the general case, this gives a total of $1+4M-3+2= 4M$ bits.

The index of the 4 signed pulses is given by

$$I_{4p} = I_{AB} + k\times 2^{4M-2}$$

where k is the case index (2 bits), and I_{AB} is the index of the pulses in both sections for each individual case.

For cases 0 and 1, I_{AB} is given by

$$I_{AB\_0,4} = I_{4p\_section} + j\times 2^{4M-3}$$

where j is a 1-bit index identifying the section with 4 pulses and $I_{4p\_section}$ is the index of the 4 pulses in that section (which requires 4M-3 bits).

For case 1, I_{AB} is given by

$$I_{AB\_1} = I_{3p\_B} + I_{1p\_A} \times 2^{3(M-1)+1}$$

where I_{3p_B} is the index of the 3 pulses in Section B (3(M-1)+1 bits) and I_{1p_A} is the index of the pulse in Section A ((M-1)+1 bits).

For case 2, I_{AB} is given by

$$I_{AB\_2} = I_{2p\_B} + I_{2p\_A} \times 2^{2(M-1)+1}$$

where I_{2p_B} is the index of the 2 pulses in Section B (2(M-1)+1 bits) and I_{2p_A} is the index of the two pulses in Section A (2(M-1)+1 bits).

Finally, for case 3, I_{AB} is given by

$$I_{AB\_3} = I_{1p\_B} + I_{3p\_A} \times 2^{M}$$

where I_{1p_B} is the index of the pulse in Section B ((M-1)+1 bits) and I_{3p_A} is the index of the 3 pulses in Section A (3(M-1)+1 bits).

For cases 0 and 4, it was mentioned that the 4 pulses in one section are encoded using $4(M-1)+1$ bits. This is done by further dividing the section into 2 subsections of length $K/4=2^{M-2}$ (=4 in this case); identifying a subsection that contains at least 2 pulses; coding the 2 pulses in that subsection using $2(M-2)+1=2M-3$ bits; coding the index of the subsection that contains at least 2 pulses using 1 bit; and coding the remaining 2 pulses, assuming that they can be anywhere in the section, using $2(M-1)+1=2M-1$ bits. This gives a total of (2M-3)+(1)+(2M-1) = 4M-3 bits

Encoding 5 signed pulses per track

The 5 signed pulses in a track of length $K=2^M$ can be encoded using $5M$ bits. Similar to the case of 4 pulses, the K positions in the track are divided into 2 sections A and B. Each section can contain from 0 to 5 pulses. A simple

approach to encode the 5 pulses is to identify a section that contains at least 3 pulses and to encode the 3 pulses in that section using 3(M-1)+1= 3M-2 bits, and to encode the remaining 2 pulses in the whole track using 2M+1 bits. This gives 5M-1 bits. An extra bit is needed to identify the section that contains at least 3 pulses. Thus a total of 5M bits are needed to encode the 5 signed pulses.

The index of the 5 signed pulses is given by

$$I_{5p} = I_{2p} + I_{3p} \times 2^{2M} + k \times 2^{5M-1}$$

Where k is the index of the section that contains at least 3 pulses, I_{3p} is the index of the 3 pulses in that section (3(M-1)+1 bits), and I_{2p} is the index of the remaining 2 pulses in the track (2M+1 bits).

Encoding 6 signed pulses per track

The 6 signed pulses in a track of length $K=2^M$ are encoded using 6M-2 bits. Similar to the case of 5 pulses, the K positions in the track are divided into 2 sections A and B. Each section can contain from 0 to 6 pulses. The table below shows the 7 cases representing the possible number of pulses in each sections:

case	*Pulses in Section A*	*Pulses in Section B*	*Bits needed*
0	0	6	6M-5
1	1	5	6M-5
2	2	4	6M-5
3	3	3	6M-4
4	4	2	6M-5
5	5	1	6M-5
6	6	0	6M-5

Note that cases 0 and 6 are similar except that the 6 pulses are in different section. Similarly, cases 1 and 5 as well as cases 2 and 4 differ only in the section that contains more pulses. Therefore these cases can be coupled and an extra bit can be assigned to identify the section that contains more pulses. Since these cases initially need 6M-5 bits, the coupled cases need 6M-4 bits taking into account the Section bit. Thus, we have now 4 states of coupled cases, that is (0,6), (1,5), (2,4), and (3),with 2 extra bits needed for the state. This gives a total of 6M-4+2=6M-2 bits for the 6 signed pulses.

In cases 0 and 6, 1 bit is needed to identify the section which contains 6 pulses. 5 pulses in that section are encoded using 5(M-1) bits (since the pulses are confined to that section), and the remaining pulse is encoded using (M-1)+1 bits. Thus a total of 1+5(M-1)+M=6M-4 bits are needed for this coupled case. Extra 2 bits are needed to encode the state of the coupled case, giving a total of 6M-2 bits. For this coupled case, the index of the 6 pulses is given by

$$I_{6p} = I_{1p} + I_{5p} \times 2^{M} + j \times 2^{6M-5} + k \times 2^{6M-4}$$

where k is the index of the coupled case (2 bits), j is the index of the section containing 6 pulses (1 bit), I_{5p} is the index of 5 pulses in that section (5(M-1) bits), and I_{1p} is the index of the remaining pulse in that section ((M-1)+1 bits).

In cases 1 and 5, 1 bit is needed to identify the section which contains 5 pulses. The 5 pulses in that section are encoded using 5(M-1) bits and the pulse in the other section is encoded using (M-1)+1 bits. For this coupled case, the index of the 6 pulses is given by

$$I_{6p} = I_{1p} + I_{5p} \times 2^{M} + j \times 2^{6M-5} + k \times 2^{6M-4}$$

where k is the index of the coupled case (2 bits), j is the index of the section containing 5 pulses (1 bit), I_{5p} is the index of the 5 pulses in that section (5(M-1) bits), and I_{1p} is the index of the pulse in the other section ((M-1)+1 bits).

In cases 2 or 4, 1 bit is needed to identify the section which contains 4 pulses. The 4 pulses in that section are encoded using 4(M-1) bits and the 2 pulses in the other section are encoded using 2(M-1)+1 bits. For this coupled case, the index of the 6 pulses is given by

$$I_{6p} = I_{2p} + I_{4p} \times 2^{2(M-1)+1} + j \times 2^{6M-5} + k \times 2^{6M-4}$$

where k is the index of the coupled case (2 bits), j is the index of the section containing 4 pulses (1 bit), I_{4p} is the index of 4 pulses in that section (4(M-1) bits), and I_{2p} is the index of the 2 pulses in the other section (2(M-1)+1 bits).

In case 3, the 3 pulses in each section are encoded using 3(M-1)+1 bits in each Section. For this case, the index of the 6 pulses is given by

$I_{6p} = I_{3pB} + I_{3pA} \times 2^{3(M-1)+1} + k \times 2^{6M-4}$

where k is the index of the coupled case (2 bits), I_{3pB} is the index of 3 pulses Section B (3(M-1)+1 bits), and I_{3pA} is the index of the 3 pulses in Section A (3(M-1)+1 bits).

5.8.3 Codebook search

The algebraic codebook is searched by minimizing the mean square error between the weighted input speech and the weighted synthesis speech. The target signal used in the closed-loop pitch search is updated by subtracting the adaptive codebook contribution. That is

$$x_2(n) = x(n) - g_p y(n), \quad n = 0,\ldots,63, \tag{40}$$

where $y(n) = v(n) * h(n)$ is the filtered adaptive codebook vector and g_p is the unquantized adaptive codebook gain.

The matrix $\mathbf{H}$ is defined as the lower triangular Toeplitz convolution matrix with diagonal $h(0)$ and lower diagonals $h(1),\ldots,h(63)$, and $\mathbf{d} = \mathbf{H}^t \mathbf{x}_2$ is the correlation between the target signal $x_2(n)$ and the impulse response $h(n)$ (also known as the backward filtered target vector), and $\Phi = \mathbf{H}^t \mathbf{H}$ is the matrix of correlations of $h(n)$.

The elements of the vector $\mathbf{d}$ are computed by

$$d(n) = \sum_{i=n}^{63} x_2(i) h(i-n), \quad n = 0,\ldots 63, \tag{41}$$

and the elements of the symmetric matrix Φ are computed by

$$f(i,j) = \sum_{n=j}^{63} h(n-i) h(n-j), \quad i = 0,...,63, \quad j = i,...,63. \tag{42}$$

If $\mathbf{c}_k$ is the algebraic codevector at index k, then the algebraic codebook is searched by maximizing the search criterion

$$Q_k = \frac{(\mathbf{x}_2^t \mathbf{H} \mathbf{c}_k)^2}{\mathbf{c}_k^t \mathbf{H}^t \mathbf{H} c_k} = \frac{(\mathbf{d}^t \mathbf{c}_k)^2}{\mathbf{c}_k^t \Phi c_k} = \frac{(R_k)^2}{E_k}. \tag{43}$$

The vector $\mathbf{d}$ and the matrix F are usually computed prior to the codebook search.

The algebraic structure of the codebooks allows for very fast search procedures since the innovation vector $\mathbf{c}_k$ contains only a few nonzero pulses. The correlation in the numerator of Equation (43) is given by

$$C = \sum_{i=0}^{N_p - 1} a_i d(m_i) \tag{44}$$

where m_i is the position of the ith pulse, a_i is its amplitude, and N_p is the number of pulses. The energy in the denominator of Equation (43) is given by

$$E = \sum_{i=0}^{N_p - 1} f(m_i, m_i) + 2 \sum_{i=0}^{N_p - 2} \sum_{j=i+1}^{N_p - 1} a_i a_j f(m_i, m_j) \tag{45}$$

To simplify the search procedure, the pulse amplitudes are predetermined based on a certain reference signal $b(n)$. In this so-called signal-selected pulse amplitude approach, the sign of a pulse at position i is set equal to the sign of the reference signal at that position. Here, the reference signal $b(n)$ is given by

$$b(n) = \sqrt{\frac{E_d}{E_r}} r_{LTP}(n) + \alpha d(n) \tag{46}$$

where $E_d = \mathbf{d}^t\mathbf{d}$ is the energy of the signal $d(n)$ and $E_r = \mathbf{r}_{LTP}^t\mathbf{r}_{LTP}$ is the energy of the signal $r_{LTP}(n)$ which is the residual signal after long term prediction. The scaling factor a controls the amount of dependence of the reference signal on $d(n)$, and it is lowered as the bit rate is increased. Here a =2 for 6.6 and 8.85 modes; a =1 for 12.65, 14.25, and 15.85 modes; a =0.8 for 18.25 mode; a =0.75 for 19.85 mode; and a =0.5 for 23.05 and 23.85 modes.

To simplify the search the signal $d(n)$ and matrix $\mathbf{F}$ are modified to incorporate the pre-selected signs. Let $s_b(n)$ denote the vector containing the signs of $b(n)$. The modified signal $d'(n)$ is given by

$$d'(n) = s_b(n)d(n) \qquad n=0,\ldots,N\text{-}1$$

and the modified autocorrelation matrix $\mathbf{F}'$ is given by

$$f'(i,j) = s_b(i)s_b(j)f(i,j), \qquad i=0,\ldots,N\text{-}1; \quad j=i,\ldots,N\text{-}1.$$

The correlation at the numerator of the search criterion Q_k is now given by

$$R = \sum_{i=0}^{N_p-1} d'(i)$$

and the energy at the denominator of the search criterion Q_k is given by

$$E = \sum_{i=0}^{N_p-1} f'(m_i,m_i) + 2\sum_{i=0}^{N_p-2}\sum_{j=i+1}^{N_p-1} f'(m_i,m_j)$$

The goal of the search now is to determine the codevector with the best set of N_p pulse positions assuming amplitudes of the pulses have been selected as described above. The basic selection criterion is the maximization of the above mentioned ratio Q_k.

In order to reduce the search complexity, a fast search procedure known as depth-first tree search procedure is used, whereby the pulse positions are determined N_m pulses at a time. More precisely, the N_p available pulses are partitioned into M non-empty subsets of N_m pulses respectively such that $N_1+N_2...+N_m...+N_M = N_p$. A particular choice of positions for the first $J = N_1+N_2...+N_{m-1}$ pulses considered is called a level-m path or a path of length J. The basic criterion for a path of J pulse positions is the ratio $Q_k(J)$ when only the J relevant pulses are considered.

The search begins with subset #1 and proceeds with subsequent subsets according to a tree structure whereby subset m is searched at the m^{th} level of the tree. The purpose of the search at level 1 is to consider the N_1 pulses of subset #1 and their valid positions in order to determine one, or a number of, candidate path(s) of length N_1 which are the tree nodes at level l. The path at each terminating node of level m-1 is extended to length $N_1+N_2...+N_m$ at level m by considering N_m new pulses and their valid positions. One, or a number of, candidate extended path(s) are determined to constitute level-m nodes. The best codevector corresponds to that path of length N_p which maximizes the criterion $Q_k(N_p)$ with respect to all level-M nodes.

A special form of the depth-first tree search procedure is used here, in which two pulses are searched at a time, that is, N_m=2, and these 2 pulses belong to two consecutive tracks. Further, instead of assuming that the matrix $\mathbf{F}$ is precomputed and stored, which requires a memory of $N\times N$ words (64×64= 4k words), a memory-efficient approach is used which reduces the memory requirement. In this approach, the search procedure is performed in such a way that only a part of the needed elements of the correlation matrix are precomputed and stored. This part corresponds to the correlations of the impulse response corresponding to potential pulse positions in consecutive tracks, as well as the correlations corresponding to $f(j,j), j=0,\ldots,N\text{-}1$ (that is the elements of the main diagonal of matrix $\mathbf{F}$).

In order to reduce complexity, while testing possible combinations of two pulses, a limited number of potential positions of the first pulse are tested. Further, in case of large number of pulses, some pulses in the higher levels of the search tree are fixed. In order to guess intelligently which potential pulse positions are considered for the first pulse or in order to fix some pulse positions, a "pulse-position likelihood-estimate vector" **b** is used, which is based on speech-related signals. The p^{th} component $b(p)$ of this estimate vector **b** characterizes the probability of a pulse occupying position p (p = 0, 1, ... N-1) in the best codevector we are searching for. Here the estimate vector **b** is the same vector used for preselecting the amplitudes and given in Equation (46).

The search procedures for all bit rate modes are similar. Two pulses are searched at a time, and these two pulses always correspond to consecutive tracks. That is the two searched pulses are in tracks T_0-T_1, T_1-T_2, T_2-T_3, or T_3-T_0.

Before searching the positions, the sign of at pulse a potential position n is set the sign of $b(n)$ at that position. Then the modified signal $d'(n)$ is computed as described above by including the predetermined signs.

For the first 2 pulses (1st tree level), the correlation at the numerator of the search criterion is given by

$$R = d'(m_0) + d'(m_1)$$

and the energy at the denominator of the search criterion Q_k is given by

$$E = f'(m_0, m_0) + f'(m_1, m_1) + 2f'(m_0, m_1)$$

where the correlations $f'(m_i, m_j)$ has been modified to include the preselected signs at positions m_i and m_j.

For subsequent levels, the numerator and denominator are updated by adding the contribution of two new pulses. Assuming that two new pulses at a certain tree level with positions m_k and m_{k+1} from two consecutive tracks are searched, then the updated value of R is given by

$$R = R + d'(m_k) + d'(m_{k+1}) \tag{47}$$

and the updated energy is given by

$$E = E + f'(m_k, m_k) + f'(m_{k+1}, m_{k+1}) + 2f'(m_k, m_{k+1}) + 2R_{hv}(m_k) + 2R_{hv}(m_{k+1}) \tag{48}$$

where $R_{hv}(m)$ is the correlation between the impulse response $h(n)$ and a vector $v_h(n)$ containing the addition of delayed versions of impulse response at the previously determined positions. That is,

$$v_h(n) = \sum_{i=0}^{k-1} h(n - m_i)$$

and

$$R_{hv}(m) = \sum_{n=m}^{N-1} h(n) v_h(n - m)$$

At each tree level, the values of $R_{hv}(m)$ are computed online for all possible positions in each of the two tracks being tested. It can be seen from Equation (48) that only the correlations $f'(m_k, m_{k+1})$ corresponding to pulse positions in two consecutive tracks need to be stored (4×16×16 words), along with the correlations $f'(m_k, m_k)$ corresponding to the diagonal of the matrix $\mathbf{F}$ (64 words). Thus the memory requirement in the present algebraic structure is 1088 words instead of 64×64=4096 words.

The search procedures at the different bit rates modes are similar. The difference is in the number of pulses, and accordingly, the number of levels in the tree search. In order to keep a comparable search complexity across the different codebooks, the number of tested positions is kept similar.

The search in the 12.65 kbit/s mode will be described as an example. In this mode, 2 pulses are placed in each track giving a total of 8 pulses per subframe of length 64. Two pulses are searched at a time, and these two pulses always correspond to consecutive tracks. That is the two searched pulses are in tracks T_0-T_1, T_1-T_2, T_2-T_3, or T_3-T_0. The tree has 4 levels in this case. At the first level, pulse P_0 is assigned to track T_0 and pulse P_1 to track T_1. In this level, no search is performed and the two pulse positions are set to the maximum of $b(n)$ in each track. In the second level, pulse P_2 is assigned to track T_2 and pulse P_3 to track T_3. 4 positions for pulse P_2 are tested against all 16 positions of pulse P_3. The 4 tested positions of P_2 are determined based on the maxima of $b(n)$ in the track. In the third level, pulse P_4 is assigned to track T_1 and pulse P_5 to track T_2. 8 positions for pulse P_4 are tested against all 16 positions of pulse P_5. Similar to the previous search level, the 8 tested positions of P_4 are determined based on the maxima of $b(n)$ in the track. In the fourth level, pulse P_6 is assigned to track T_3 and pulse P_7 to track T_0. 8 positions for pulse P_6 are tested against all 16 positions of pulse P_7. Thus the total number of tested combination is 4×16+8×16+8×16=320. The whole process is repeated 4 times (4 iterations) by assigning the pulses to different tracks. For example, in the 2nd iteration, pulses P_0 to P_7 are assigned to tracks T_1, T_2, T_3, T_0, T_2, T_3, T_0, and T_1, respectively. Thus the total number of tested position combinations is 4×320=1280.

As another search example, in the 15.85 kbit/s mode, 3 pulses are placed in each track giving a total of 12 pulses. There are 6 levels in the tree search whereby two pulses are searched in each level. In the first two levels, 4 pulses are set to the maxima of $b(n)$. In the subsequent 4 levels, the number of tested combinations are 4×16, 6×16, 8×16, and 8×16, respectively. 4 iterations are used giving a total of 4×26×16=1664 combinations.

5.9 Quantization of the adaptive and fixed codebook gains

The adaptive codebook gain (pitch gain) and the fixed (algebraic) codebook gain are vector quantized using a 6-bit codebook for modes 8.85 and 6.60 kbit/s and using a 7-bit codebook for all the other modes.

The fixed codebook gain quantization is performed using MA prediction with fixed coefficients. The 4th order MA prediction is performed on the innovation energy as follows. Let $E(n)$ be the mean-removed innovation energy (in dB) at subframe n, and given by

$$E(n) = 10\log\left(\frac{1}{N}g_c^2\sum_{i=0}^{N-1} c^2(i)\right) - \overline{E} \qquad (49)$$

where N=64 is the subframe size, $c(i)$ is the fixed codebook excitation, and $\overline{E} = 30$ dB is the mean of the innovation energy. The predicted energy is given by

$$\widetilde{E}(n) = \sum_{i=1}^{4} b_i \hat{R}(n-i) \qquad (50)$$

where $[b_1\, b_2\, b_3\, b_4]$=[05.,0.4,0.3,0.2] are the MA prediction coefficients, and $\hat{R}(k)$ is the quantized energy prediction error at subframe k. The predicted energy is used to compute a predicted fixed-codebook gain g'_c as in Equation (49) (by substituting $E(n)$ by $\widetilde{E}(n)$ and g_c by g'_c). This is done as follows. First, the mean innovation energy is found by

$$E_i = 10\,\log\left(\frac{1}{N}\sum_{i=0}^{N-1} c^2(i)\right) \qquad (51)$$

and then the predicted gain g'_c is found by

$$g_c^{'} = 10^{0.05(\widetilde{E}(n)+\overline{E}-E_i)}. \qquad (52)$$

A correction factor between the gain g_c and the estimated one g'_c is given by

$$\gamma = g_c / g_c^{'}. \qquad (53)$$

Note that the prediction error is given by

$$R(n) = E(n) - \widetilde{E}(n) = 20\ \log\ (\gamma). \qquad (54)$$

The pitch gain, g_p, and correction factor γ are jointly vector quantized using a 6-bit codebook for modes 8.85 and 6.60 kbit/s, and 7-bit codebook for other modes. The gain codebook search is performed by minimizing the mean-square of the weighted error between original and reconstructed speech which is given

$$E = x^t x + g_p^2 y^t y + g_c^2 z^t z - 2g_p x^t y - 2g_c x^t z + 2g_p g_c y^t z, \qquad (55)$$

where the x is the target vector, y is the filtered adaptive codebook vector, and z is the filtered fixed codebook vector. (Each gain vector in the codebook also has an element representing the quantized energy prediction error.) The quantized energy prediction error associated with the chosen gains is used to update $\hat{R}(n)$. In the search, only the 64 codevectors that are closest to the unquantized pitch gain, g_p, are taken into account.

5.10 Memory update

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

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n), \quad n = 0,\ldots,63, \tag{56}$$

where $\hat{g}_p$ and $\hat{g}_c$ are the quantized adaptive and fixed codebook gains, respectively, $v_i(n)$ the adaptive codebook vector (interpolated past excitation), and $c(n)$ is the fixed codebook vector (algebraic code including pitch sharpening). The states of the filters can be updated by filtering the signal $r(n)-u(n)$ (difference between residual and excitation) through the filters $1/\hat{A}(z)$ and $A(z/g_1)H_{de-emph}(z)$ for the 64 sample subframe and saving the states of the filters. This would require 3 filterings. A simpler approach which requires only one filtering is as follows. The local synthesis speech, $\hat{s}(n)$, 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 $e(n) = s(n) - \hat{s}(n)$. So the states of the synthesis filter $1/\hat{A}(z)$ are given by $e(n)$,n=48,…,63. Updating the states of the filter $A(z/g_1)H_{de-emph}(z)$ can be done by filtering the error signal $e(n)$ through this filter to find the perceptually weighted error $e_w(n)$. However, the signal $e_w(n)$ can be equivalently found by

$$e_w(n) = x(n) - \hat{g}_p y(n) - \hat{g}_c z(n). \tag{57}$$

Since the signals $x(n)$, $y(n)$, and $z(n)$ are available, the states of the weighting filter are updated by computing $e_w(n)$ as in Equation (54) for $n = 48,\ldots,63$. This saves two filterings.

5.11 High-band gain generation

In order to compute the high band gain for 23.85 kbit/s mode, 16 kHz input speech is filtered through a band-pass FIR filter $H_{HB}(z)$ which has the passband from 6.4 to 7 kHz. The high band gain g_{HB} is obtained by

$$g_{HB} = \frac{\sum_{i=0}^{63} (s_{HB}(i))^2}{\sum_{i=0}^{63} (s_{HB2}(i))^2}, \tag{58}$$

where $s_{HB}(i)$ is band-pass filtered input speech and $s_{HB2}(i)$ is high-band speech synthesis obtained from high-band excitation $u_{HB2}(i)$ filtered through high-band synthesis filter $A_{HB}(z)$ described in Section 6.3.2.2.

6 Functional description of the decoder

The function of the decoder consists of decoding the transmitted parameters (LP parameters, adaptive codebook vector, adaptive codebook gain, fixed codebook vector, fixed codebook gain and high-band gain) and performing synthesis to obtain the reconstructed speech. The reconstructed speech is then postprocessed and upsampled (and upscaled). Finally high-band signal is generated to the frequency band from 6 to 7 kHz. The signal flow at the decoder is shown in Figure 3.

6.1 Decoding and speech synthesis

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 Section 5.2.6 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 Section 5.6. The received adaptive filter index is used to find out whether the filtered adaptive codebook is $v_1(n)= v(n)$ or $v_2(n) = 0.18v(n) + 0.64v(n-1) + 0.18v(n-2)$.

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 $1/(1\text{-}0.85z^{-T})$ and a tilt part $(1 - b_1 z^{-1})$, where T is the integer part of the pitch lag and $b_1(n)$ 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 $\hat{g}$. The estimated fixed codebook gain g'_c is found as described in Section 5.8. First, the predicted energy for every subframe n is found by

$$\tilde{E}(n) = \sum_{i=1}^{4} b_i \hat{R}(n-i) \qquad (59)$$

and then the mean innovation energy is found by

$$E_i = 10 \log\left(\frac{1}{N}\sum_{i=0}^{N-1} c^2(i)\right) \qquad (60)$$

The predicted gain $g_c^{'}$ is found by

$$g_c^{'} = 10^{0.05(\tilde{E}(n)+\bar{E}-E_i)} \qquad (61)$$

The quantized fixed codebook gain is given by

$$\hat{g}_c = \hat{g} g_c^{'}. \qquad (62)$$

4. **Computing the reconstructed speech:** The following steps are for n = 0, ..., 63. The total excitation is constructed by:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n), \qquad . \qquad (63)$$

Before the speech synthesis, a post-processing of excitation elements is performed

5. **Anti-sparseness processing (6.60 and 8.85 kbit/s modes):** An adaptive anti-sparseness post-processing procedure is applied to the fixed codebook vector $c(n)$ in order to reduce perceptual artifacts arising from the sparseness of the algebraic fixed codebook vectors with only a few non-zero samples per subframe. The anti-sparseness processing consists of circular convolution of the fixed codebook vector with an impulse response. Three pre-stored impulse responses are used and a number *impNr*=0,1,2 is set to select one of them. A value of 2 corresponds to no modification, a value of 1 corresponds to medium modification, while a value of 0 corresponds to strong modification. The selection of the impulse response is performed adaptively from the adaptive and fixed codebook gains. The following procedure is employed:

```
if ĝ_p < 0.6 then
    impNr = 0;
else if ĝ_p < 0.9 then
    impNr = 1;
else
    impNr = 2;
```

Detect onset by comparing the fixed codebook gain to the previous fixed codebook gain. If the current value is more than three times the previous value an onset is detected.

If not onset and *impNr*=0, the median filtered value of the current and the previous 4 adaptive codebook gains are computed. If this value is less than 0.6, *impNr*=0.

If not onset, the *impNr*-value is restricted to increase by one step from the previous subframe.

If an onset is declared, the *impNr* -value is increased by one if it is less than 2.

In case of 8.85 kbit/s mode, the *impNr* -value is increased by one.

6. **Noise enhancer:** A nonlinear gain smoothing technique is applied to the fixed codebook gain $\hat{g}_c$ in order to enhance excitation in noise. Based on the stability and voicing of the speech segment, the gain of the fixed codebook 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 is given by l=0.5(1-r_v) with r_v=(E_v-E_c)/(E_v+E_c), where E_v and E_c are the energies of the scaled pitch codevector and scaled innovation codevector, respectively. Note that since the value of r_v is between –1 and 1, the value of l is between 0 and 1. Note that the factor l 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 q is computed based on a distance measure between the adjacent LP filters. Here, the factor q is related to the ISP distance measure and it is bounded by 0≤q≤1, with larger values of q corresponding to more stable signals.

   Finally, a gain smoothing factor S_m is given by

$$S_m = lq. \tag{64}$$

   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 fixed 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 or equal to g_{-1}, then g_0 is computed by decrementing $\hat{g}_c$ by 1.5 dB bounded by $g_0 \geq g_{-1}$. If $\hat{g}_c$ is smaller than g_{-1}, then g_0 is computed by incrementing $\hat{g}_c$ by 1.5 dB bounded by $g_0 \leq g_{-1}$.

   Finally, the gain is update with the value of the smoothed gain as follows

$$\hat{g}_c = qg_0 + (1-q)\hat{g}_c, \tag{65}$$

7. **Pitch enhancer:** A pitch enhancer procedure modifies the total excitation $u(n)$ by filtering the fixed codebook excitation through an innovation filter whose frequency response emphasizes the higher frequencies more than lower frequencies, and whose coefficients are related to the periodicity in the signal. A filter of the form

$$F_{inno}(z) = -c_{pe}z + 1 - c_{pe}z^{-1}, \tag{66}$$

   where c_{pe}=0.125(1– r_v), with r_v=(E_v-E_c)/(E_v+E_c) as described above. The filtered fixed codevector is given by

$$c'(n) = c(n) - c_{pe}\big(c(n+1) + c(n-1)\big). \tag{67}$$

   and the updated excitation is given by

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c'(n). \tag{68}$$

   The above procedure can be done in one step by updating the excitation as follows

$$u(n) = u(n) - \hat{g}_c c_{pe}\big(c(n+1) + c(n-1)\big). \qquad (69)$$

8. **Post-processing of excitation elements (6.60 kbit/s mode):** A post-processing of excitation elements procedure is applied to the total excitation $u(n)$ by emphasizing the contribution of the adaptive codebook vector:

$$\hat{u}(n) = \begin{cases} u(n) + 0.25b\hat{g}_p v(n), & \hat{g}_p > 0.5 \\ u(n), & \hat{g}_p \le 0.5 \end{cases} \qquad (70)$$

Adaptive gain control (AGC) is used to compensate for the gain difference between the non-emphasized excitation $u(n)$ and emphasized excitation $\hat{u}(n)$ The gain scaling factor h for the emphasized excitation is computed by:

$$h = \begin{cases} \sqrt{\dfrac{\sum_{n=0}^{63} u^2(n)}{\sum_{n=0}^{63} \hat{u}^2(n)}}, & \hat{g}_p > 0.5, \\ 1.0, & \hat{g}_p \le 0.5. \end{cases} \qquad (71)$$

The gain-scaled emphasized excitation signal $\hat{u}'(n)$ is given by:

$$\hat{u}'(n) = \hat{u}(n)h\ . \qquad (72)$$

The reconstructed speech for the subframe of size 64 is given by

$$\hat{s}(n) = \hat{u}(n) - \sum_{i=1}^{16} \hat{a}_i \hat{s}(n-i), \qquad n = 0,\ldots,63. \qquad (73)$$

where $\hat{a}_i$ are the interpolated LP filter coefficients.

The synthesis speech $\hat{s}(n)$ is then passed through an adaptive postprocessing which is described in the following section.

6.2 High-pass filtering, up-scaling and interpolation

The high-pass filter serves as a precaution against undesired low frequency components. The signal is filtered through the high-pass filter $H_{h1}(z)$ and de-emphasis filter $H_{de\_emph}(z)$.

Finally, the signal is upsampled to 16 kHz to obtain the lower band synthesis signal $\hat{s}_{16k}(n)$. $\hat{s}_{16k}(n)$ is produced by first upsampling the lower band synthesis $\hat{s}_{12\,8k}(n)$ at 12.8 kHz by 5, then filtering the output through $H_{decim}(z)$, and finally downsampling it by 4.

(Up-scaling consists of multiplying the output from the high-pass filtering by a factor of 2 in order to compensate the down-scaling at the pre-processing stage.)

6.3 High frequency band

For the higher frequency band (6.4 – 7.0 kHz), excitation is generated to model the highest frequencies. The high frequency content is generated by filling the upper part of the spectrum with a white noise properly scaled in the excitation domain, then converted to the speech domain by shaping it with a filter derived from the same LP synthesis filter used for synthesizing the down-sampled signal.

6.3.1 Generation of high-band excitation

The high-band excitation is obtained by first generating white noise $u_{HB1}(n)$. The power of the high-band excitation is set equal to the power of the lower band excitation $u_2(n)$ which means that

$$u_{HB2}(n) = u_{HB1}(n)\sqrt{\sum_{k=0}^{63} {u_2}^2(k) \Big/ \sum_{k=0}^{63} {u_{HB1}}^2(k)} \,. \tag{74}$$

Finally the high-band excitation is found by

$$u_{HB}(n) = \hat{g}_{HB} u_{HB2}(n)\,, \tag{75}$$

where $\hat{g}_{HB}$ is a gain factor.

In the 23.85 kbit/s mode, $\hat{g}_{HB}$ is decoded from the received gain index.

In 6.60, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85 and 23.05 kbit/s modes, g_{HB} is estimated using voicing information bounded by [0.1,1.0]. First, tilt of synthesis e_{tilt} is found

$$e_{tilt} = \sum_{n=1}^{63} \hat{s}_{hp}(n)\hat{s}_{hp}(n-1) \Big/ \sum_{n=0}^{63} {\hat{s}_{hp}}^2(n) \tag{76}$$

where $\hat{s}_{hp}(n)$ is high-pass filtered lower band speech synthesis $\hat{s}_{12\,8k}(n)$ with cut-off frequency of 400 Hz. The $\hat{g}_{HB}$ is then found by

$$g_{HB} = w_{SP} g_{SP} + (1 - w_{SP}) g_{BG}\,, \tag{77}$$

where g_{SP} = 1 - e_{tilt} is gain for speech signal, g_{BG} = 1.25g_{SP} is gain for background noise signal, and w_{SP} is a weighting function set to 1, when VAD is ON, and 0 when VAD is OFF. g_{HB} is bounded between [0.1, 1.0]. In case of voiced segments where less energy is present at high frequencies, e_{tilt} approaches 1 resulting in a lower gain g_{HB}. This reduces the energy of the generated noise in case of voiced segments.

6.3.2 LP filter for the high frequency band

6.3.2.1 6.60 kbit/s mode

The high-band LP synthesis filter $A_{HB}(z)$ is found by extrapolating the quantized ISF vector **f** into 20th order ISF vector $\mathbf{f}_e$. First, maximum of the autocorrelation $C_{max}(i)$ of ISF vector difference vector $f_\Delta(i) = f(i+1) - f(i), i = 1,...,14$ is obtained. Then new 16kHz ISF vector $f_e'(i)$ is computed by

$$f_e'(i) = \begin{cases} f(i-1), & i = 1,..,15 \\ f_e'(i-1) + f_e'(i - C_{\max}(i) - 1) - f_e'(i - C_{\max}(i) - 2), & i = 16,..,19 \end{cases}. \tag{78}$$

An approximation of the last element of new ISF vector f_{e19} is updated based on lower frequency coefficients. New extrapolated ISF vector difference vector $f_{e\Delta}'(i)$ is

$$f_{e\Delta}'(i) = c_{scale}\left(f_e'(i) - f_e'(i-1)\right), i = 16,...,19\,, \tag{79}$$

where c_{scale} scales $f_{e\Delta}'(i)$ so that $f_e(19)$ will be equal to f_{e19}. In order to insure stability, $f_{e\Delta}'(i)$ is bounded by

$$f_{e\Delta}'(i) + f_{e\Delta}'(i-1) > 500, i = 17,...,19\,. \tag{80}$$

Finally, the extrapolated ISF vector $\mathbf{f}_e$ is obtained by

$$f_e(i) = \begin{cases} f(i), & i = 1,...,15 \\ f_{e\Delta}'(i) + f_e(i-1), & i = 16,...,19 \\ f(16), & i = 20 \end{cases}. \tag{81}$$

$\mathbf{f}_e$ is converted to cosine domain to obtain $\mathbf{q}_e$ with 16000 Hz sampling rate. The high-band LP synthesis filter $A_{HB}(z)$ is obtained by converting $\mathbf{q}_e$ to LP filter as described in 5.2.4 with m=20.

6.3.2.2 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05 or 23.85 kbit/s modes

The high-band LP synthesis filter $A_{HB}(z)$ is weighted low-band LP synthesis filter

$$A_{HB}(z) = \hat{A}(z/0.8), \tag{82}$$

where $\hat{A}(z)$ is the interpolated LP synthesis filter. $\hat{A}(z)$ has been computed analysing signal with the sampling rate of 12.8 kHz but it is now used for a 16 kHz signal. Effectively, this means that the frequency response $FR_{16}(f)$ of $A_{HB}(z)$ is obtained by

$$FR_{16}(f) = FR_{12\,8}(\frac{12.8}{16} f), \tag{83}$$

where $FR_{12\,8}(f)$ is the frequency response of $A(z)$. This means that the band 5.1 – 5.6 kHz in 12.8 kHz domain will be mapped to 6.4 – 7.0 kHz in 16 kHz domain.

6.3.3 High band synthesis

$u_{HB}(n)$ is filtered through $A_{HB}(z)$. The output of this high-band synthesis $s_{HB}(n)$ is filtered through a band-pass FIR filter $H_{HB}(z)$ which has the passband from 6 to 7 kHz. Finally, s_{HB} is added to synthesized speech $\hat{s}_{16k}(n)$ to produce the synthesized output speech signal $\hat{s}_{output}(n)$.

7 Detailed bit allocation of the adaptive multi-rate wideband codec

The detailed allocation of the bits in the adaptive multi-rate wideband speech encoder is shown for each mode in table 12a-12i. These tables show the order of the bits produced by the speech encoder. Note that the most significant bit (MSB) of each codec parameter is always sent first.

Table 12a: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 477 bits/20 ms, 23.85 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s68	Codebook Index1 for track 1
s69 – s79	Codebook Index1 for track 2
ss80 –s90	Codebook Index1 for track 3
s91–s101	Codebook Index1 for track 4
s102–s112	Codebook Index2 for track 1
s113–s123	Codebook Index2 for track 2
s124 – s134	Codebook Index2 for track 3
s135 – s145	Codebook Index2 for track 4
s146 – s152	codebook gains
s153 – s156	High-band energy
subframe 2	
s157 – s162	adaptive codebook index (relative)
s163 – s262	same description as s57 – s156
subframe 3	
s263 – s371	same description as s48 – s156
subframe 4	
s372 – s477	same description as s157 – s262

Table 12b: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 461 bits/20 ms, 23.05 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s68	Codebook Index1 for track 1
s69 – s79	Codebook Index1 for track 2
ss80 –s90	Codebook Index1 for track 3
s91–s101	Codebook Index1 for track 4
s102–s112	Codebook Index2 for track 1
s113–s123	Codebook Index2 for track 2
s124 – s134	Codebook Index2 for track 3
s135 – s145	Codebook Index2 for track 4
s146 – s152	codebook gains
subframe 2	
s153 – s158	adaptive codebook index (relative)
s159 – s254	same description as s57 – s152
subframe 3	
s255 – s359	same description as s48 – s152
subframe 4	
s360 – s461	same description as s153 – s254

Table 12c: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 397 bits/20 ms, 19.85 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s67	Codebook Index1 for track 1
s68 – s77	Codebook Index1 for track 2
s78 – s79	Pulse Selector for track 3
s80 – s81	Pulse Selector for track 4
s82 – s91	Codebook index2 for track 1
s92 – s101	Codebook index2 for track 2
s102 – s115	Codebook index for track 3
s116 – s129	Codebook index for track 4
s130 – s136	VQ gain
subframe 2	
s137 – s142	adaptive codebook index (relative)
s143 – s222	same description as s57 – s136
subframe 3	
s223 – s311	same description as s48 – s136
subframe 4	
s312 – s397	same description as s137 – s222

Table 12d: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 365 bits/20 ms, 18.25 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s59	Pulse Selector for track 1
s60 – s61	Pulse Selector for track 2
s62 – s63	Pulse Selector for track 3
s64 – s65	Pulse Selector for track 4
s66 – s79	Codebook index for track 1
s80 – s93	Codebook index for track 2
s94 – s107	Codebook index for track 3
s108 – s121	Codebook index for track 4
s122 – s128	VQ gain
subframe 2	
s129 – s134	adaptive codebook index (relative)
s135 – s206	same description as s57 – s128
subframe 3	
s207 – s287	same description as s48 – s128
subframe 4	
s288 – s365	same description as s129 – s206

Table 12e: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 317 bits/20 ms, 15.85 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s70	Codebook index for track 1
s71 – s83	Codebook index for track 2
s84 – s96	Codebook index for track 3
s97 – s109	Codebook index for track 4
s110 – s116	VQ gain
subframe 2	
s117 – s122	adaptive codebook index (relative)
s123 – s182	same description as s57 – s116
subframe 3	
s183 – s251	same description as s48 – s116
subframe 4	
s252 – s317	same description as s117 – s182

Table 12f: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 285 bits/20 ms, 14.25 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s70	Codebook index for track 1
s71 – s83	Codebook index for track 2
s84 – s92	Codebook index for track 3
s93 – s101	Codebook index for track 4
s102 – s108	VQ gain
subframe 2	
s109 – s114	adaptive codebook index (relative)
s115 – s166	same description as s57 – s108
subframe 3	
s167 – s227	same description as s48 – s108
subframe 4	
s228 – s285	same description as s109 – s166

Table 12g: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 253 bits/20 ms, 12.65 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s56	adaptive codebook index
s57	LTP-filtering-flag
s58 – s66	Codebook index for track 1
s67 – s75	Codebook index for track 2
s76 – s84	Codebook index for track 3
s85 – s93	Codebook index for track 4
s94 – s100	VQ gain
subframe 2	
s101 – s106	adaptive codebook index (relative)
s107 – s150	same description as s57 – s100
subframe 3	
s151 – s203	same description as s48 – s100
subframe 4	
s204 – s253	same description as s101 – s150

Table 12h: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 177 bits/20 ms, 8.85 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s23	index of 3rd ISP subvector
s24 – s30	index of 4th ISP subvector
s31 – s37	index of 5th ISP subvector
s38 – s42	index of 6th ISP subvector
s43 – s47	index of 7th ISP subvector
subframe 1	
s48 – s55	adaptive codebook index
s56 – s60	Codebook index for track 1
s61 – s65	Codebook index for track 2
s66 – s70	Codebook index for track 3
s71 – s75	Codebook index for track 4
s76 – s81	VQ gain
subframe 2	
s82 – s86	adaptive codebook index (relative)
s87 – s112	same description as s56 – s81
subframe 3	
s113 – s146	same description as s48 – s81
subframe 4	
s147 – s177	same description as s82 – s112

Table 12i: Source encoder output parameters in order of occurrence and bit allocation within the speech frame of 132 bits/20 ms, 6.60 kbit/s mode.

Bits (MSB-LSB)	Description
s1	VAD-flag
s2 – s9	index of 1st ISP subvector
s10 – s17	index of 2nd ISP subvector
s18 – s24	index of 3rd ISP subvector
s25 – s31	index of 4th ISP subvector
s32 – s37	index of 5th ISP subvector
subframe 1	
s38 – s45	adaptive codebook index
s46 –57	Codebook Index
s58 – s63	VQ gain
subframe 2	
s64 – s68	adaptive codebook index (relative)
s69 – s86	same description as s46 – s63
subframe 3	
s87 – s109	same description as s64 – s86
subframe 4	
s110 – s132	same description as s64 – s86

8 Homing sequences

8.1 Functional description

The adaptive multi-rate wideband speech codec is described in a bit-exact arithmetic to allow easy type approval as well as general testing of correct operation of the adaptive multi-rate wideband speech codec.

The response of the codec to a predefined input sequence can only be foreseen if the internal state variables of the codec are in a predefined state at the beginning of the experiment. Therefore, the codec has to be put in a so called home state

before a bit-exact test can be performed. This is usually done by a reset (a procedure in which the internal state variables of the codec are set to their defined initial values). The codec mode of the speech encoder and speech decoder shall be set to the tested codec mode by external means at reset.

To allow a reset of the codec in remote locations, special homing frames have been defined for the encoder and the decoder, thus enabling a codec homing by inband signalling.

The codec homing procedure is defined in such a way, that in either direction (encoder or decoder) the homing functions are called after the processing of the homing frame. The output corresponding to the first homing frame is therefore dependent on the used codec mode and the codec state when receiving that frame and hence usually not known. The response of the encoder to any further homing frame is by definition the corresponding decoder homing frame for the used codec mode. The response of the decoder to any further homing frame is by definition the encoder homing frame. This procedure allows homing of both the encoder and decoder from either side, if a loop back configuration is implemented, taking proper framing into account.

8.2 Definitions

Encoder homing frame: The encoder homing frame consists of 320 identical samples, each 13 bits long, with the least significant bit set to "one" and all other bits set to "zero". When written to 16-bit words with left justification, the samples have a value of 0008 hex. The speech decoder has to produce this frame as a response to the second and any further decoder homing frame if at least two decoder homing frames were input to the decoder consecutively. The encoder homing frame is identical for all codec modes.

Decoder homing frame: There exist nine different decoder homing frames, which correspond to the nine AMR-WB codec modes. Using one of these codec modes, the corresponding decoder homing frame is the natural response of the speech encoder to the second and any further encoder homing frame if at least two encoder homing frames were input to the encoder consecutively. In Annex C/G.722.2, for each decoder homing frame the parameter values are given.

8.3 Encoder homing

Whenever the adaptive multi-rate wideband speech encoder receives at its input an encoder homing frame exactly aligned with its internal speech frame segmentation, the following events take place:

Step 1: The speech encoder performs its normal operation including VAD and SCR and produces in accordance with the used codec mode a speech parameter frame at its output which is in general unknown. But if the speech encoder was in its home state at the beginning of that frame, then the resulting speech parameter frame is identical to that decoder homing frame, which corresponds to the used codec mode (this is the way how the decoder homing frames were constructed).

Step 2: After successful termination of that operation the speech encoder provokes the homing functions for all sub-modules including VAD and SCR and sets all state variables into their home state. On the reception of the next input frame, the speech encoder will start from its home state.

NOTE: Applying a sequence of N encoder homing frames will cause at least N-1 decoder homing frames at the output of the speech encoder.

8.4 Decoder homing

Whenever the speech decoder receives at its input a decoder homing frame, which corresponds to the used codec mode, then the following events take place:

Step 1: The speech decoder performs its normal operation and produces a speech frame at its output which is in general unknown. But if the speech decoder was in its home state at the beginning of that frame, then the resulting speech frame is replaced by the encoder homing frame. This would not naturally be the case but is forced by this definition here.

Step 2: After successful termination of that operation the speech decoder provokes the homing functions for all sub-modules including the comfort noise generator and sets all state variables into their home state. On the reception of the next input frame, the speech decoder will start from its home state.

NOTE 1: Applying a sequence of N decoder homing frames will cause at least N-1 encoder homing frames at the output of the speech decoder.

NOTE 2: By definition the first frame of each decoder test sequence must differ from the decoder homing frame at least in one bit position within the parameters for LPC and first subframe. Therefore, if the decoder is in its home state, it is sufficient to check only these parameters to detect a subsequent decoder homing frame. This definition is made to support a delay-optimized implementation.

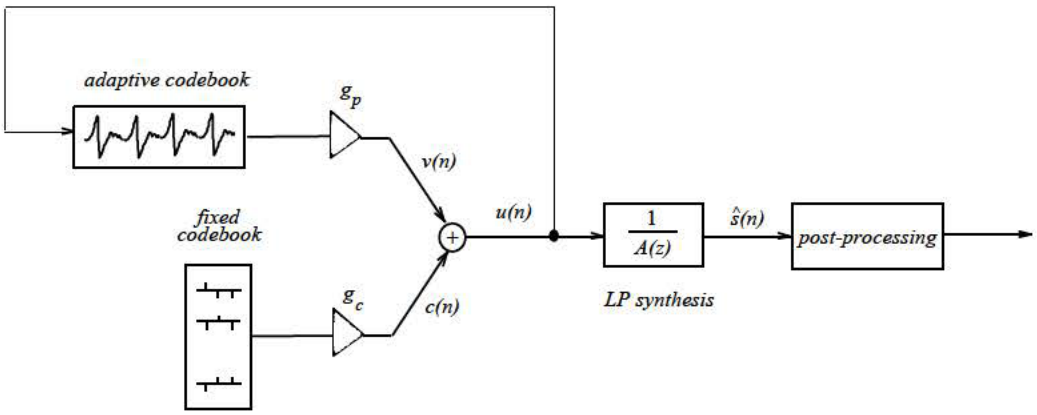


Figure 1 Simplified block diagram of the CELP synthesis model

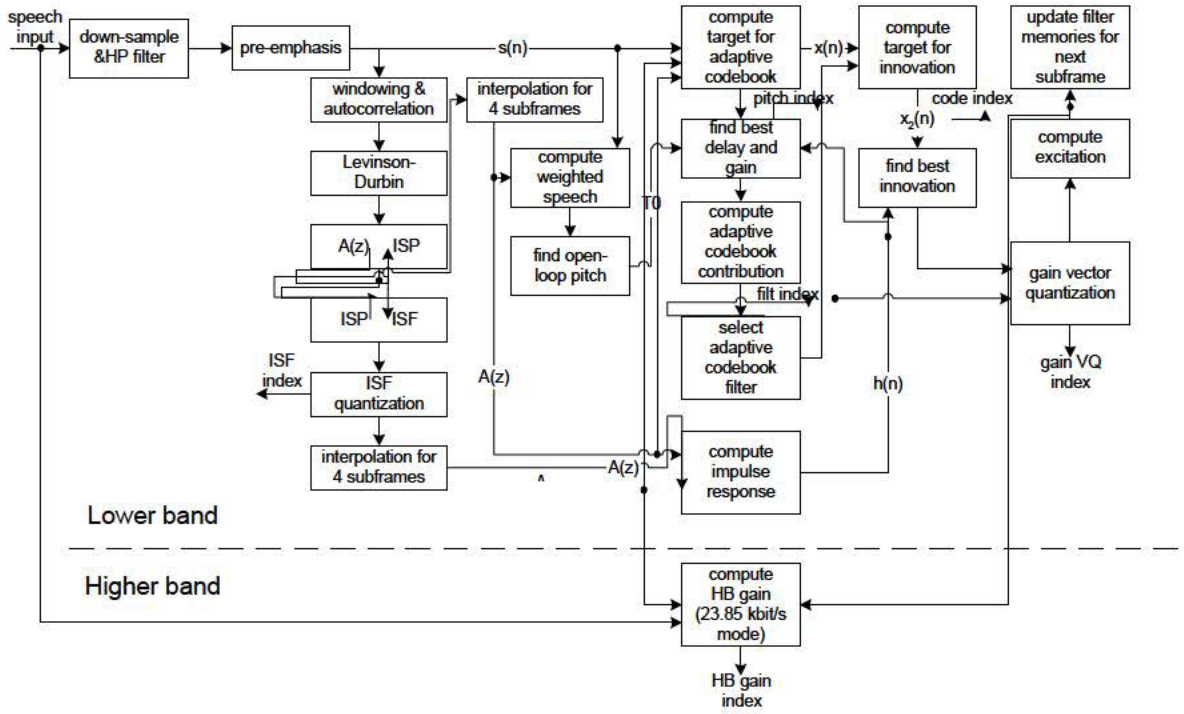


Figure 2 Detailed block diagram of the ACELP encoder

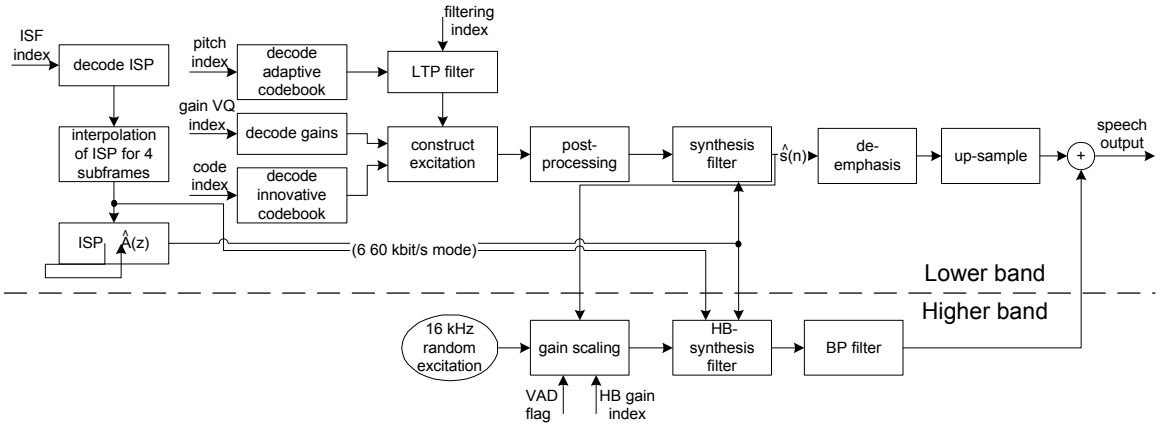


Figure 3 Detailed block diagram of the ACELP decoder

9 Voice Activity Detector (VAD)

The function of the VAD algorithm is to indicate whether each 20 ms frame contains signals that should be transmitted, e.g. speech, music or information tones. The output of the VAD algorithm is a Boolean flag (VAD_flag) indicating presence of such signals. This flag is used in the AMR-WB speech coder and Annex B/G.722.2.

9.1 VAD Symbols

For the purposes of this Section, the following symbols apply.

9.1.1 VAD Variables

bckr_est[n]	background noise estimate at the frequency band "n"
burst_count	counts length of a speech burst, used by VAD hangover addition
hang_count	hangover counter, used by VAD hangover addition
level[n]	signal level at the frequency band "n"
new_speech	pointer of the speech encoder, points a buffer containing last received samples of a speech frame
noise_level	estimated noise level
pow_sum	input power
s(i)	samples of the input frame
snr_sum	measure between input frame and noise estimate
speech_level	estimated speech level
stat_count	stationary counter
stat_rat	measure indicating stationary of the input frame
tone_flag	flag indicating the presence of a tone
vad_thr	VAD threshold
VAD_flag	Boolean VAD flag
vadreg	intermediate VAD decision

9.1.2 VAD Constants

ALPHA_UP1	constant for updating noise estimate
ALPHA_DOWN1	constant for updating noise estimate
ALPHA_UP2	constant for updating noise estimate
ALPHA_DOWN2	constant for updating noise estimate
ALPHA3	constant for updating noise estimate
ALPHA4	constant for updating average signal level
ALPHA5	constant for updating average signal level
BURST_HIGH	constant for controlling VAD hangover addition
BURST_P1	constant for controlling VAD hangover addition
BURST_SLOPE	constant for controlling VAD hangover addition

COEFF3	coefficient for the filter bank
COEFF5_1	coefficient for the filter bank
COEFF5_2	coefficient for the filter bank
HANG_HIGH	constant for controlling VAD hangover addition
HANG_LOW	constant for controlling VAD hangover addition
HANG_P1	constant for controlling VAD hangover addition
HANG_SLOPE	constant for controlling VAD hangover addition
FRAME_LEN	size of a speech frame, 256 samples (20 ms)
MIN_SPEECH_LEVEL1	constant for speech estimation
MIN_SPEECH_LEVEL2	constant for speech estimation
MIN_SPEECH_SNR	constant for VAD threshold adaptation
NO_P1	constant for VAD threshold adaptation
NO_SLOPE	constant for VAD threshold adaptation
NOISE_MAX	maximum value for noise estimate
NOISE_MIN	minimum value for noise estimate
POW_TONE_THR	threshold for tone detection
SP_ACTIVITY_COUNT	constant for speech estimation
SP_ALPHA_DOWN	constant for speech estimation
SP_ALPHA_UP	constant for speech estimation
SP_CH_MAX	constant for VAD threshold adaptation
SP_CH_MIN	constant for VAD threshold adaptation
SP_EST_COUNT	constant for speech estimation
SP_P1	constant for VAD threshold adaptation
SP_SLOPE	constant for VAD threshold adaptation
STAT_COUNT	threshold for stationary detection
STAT_THR	threshold for stationary detection
STAT_THR_LEVEL	threshold for stationary detection
THR_HIGH	constant for VAD threshold adaptation
TONE_THR	threshold for tone detection
VAD_POW_LOW	constant for controlling VAD hangover addition

9.1.3 Functions

+	Addition
-	Subtraction
*****	Multiplication
/	Division

| x |	absolute value of x
AND	Boolean AND
OR	Boolean OR

$$\sum_{n=a}^{b} x(n) = x(a) + x(a+1) + \ldots + x(b-1) + x(b)$$

MIN(x,y) $$= \begin{cases} x, x \le y \\ y, y < x \end{cases}$$

MAX(x,y) $$= \begin{cases} x, x \ge y \\ y, y > x \end{cases}$$

9.2 Functional description

The block diagram of the VAD algorithm is depicted in Figure 4. The VAD algorithm uses parameters of the speech encoder to compute the Boolean VAD flag (VAD_flag). This input frame for VAD is sampled at the 12.8 kHz frequency and thus it contains 256 samples. Samples of the input frame (s(i)) are divided into sub-bands and level of the signal (level[n]) in each band is calculated. The normalized open-loop pitch gains are the input for the tone detection function, gains which are calculated by open-loop pitch analysis of the speech encoder. The tone detection function computes a flag (tone_flag) which indicates presence of a signalling tone, voiced speech, or other strongly periodic signal. Background noise level (bckr_est[n]) is estimated in each band based on the VAD decision, signal stationarity and the tone-flag. Intermediate VAD decision is calculated by comparing input SNR (level[n]/bckr_est[n]) to an adaptive threshold. The threshold is adapted based on noise and long term speech estimates. Finally, the VAD flag is calculated by adding hangover to the intermediate VAD decision.

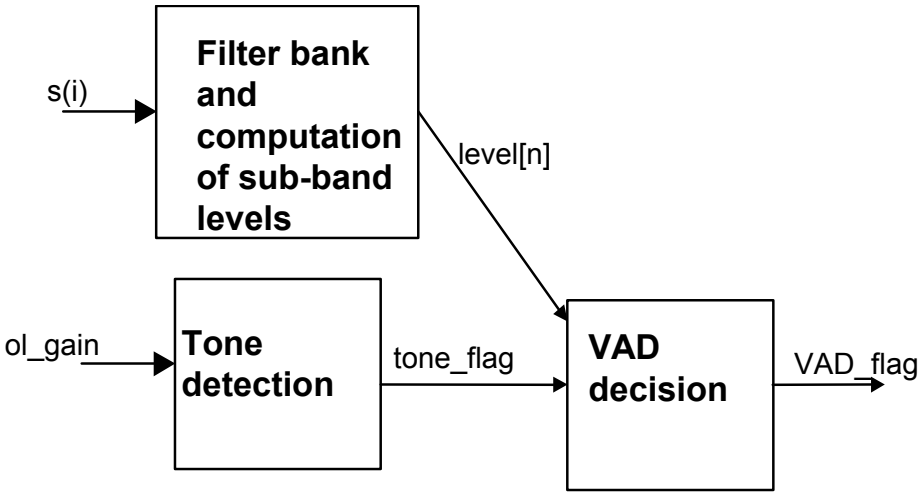


Figure 4. Simplified block diagram of the VAD algorithm

9.2.1 Filter bank and computation of sub-band levels

The input signal is divided into frequency bands using a 12-band filter bank (Figure 5). Cut-off frequencies for the filter bank are shown in Table 13.

Table 13. Cut-off frequencies for the filter bank

Band number	Frequencies
1	0 – 200 Hz
2	200 – 400 Hz
3	400 – 600 Hz
4	600 – 800 Hz
5	800 – 1200 Hz
6	1200 – 1600 Hz
7	1600 – 2000 Hz
8	2000 – 2400 Hz
9	2400 - 3200 Hz
10	3200 – 4000 Hz
11	4000 – 4800 Hz
12	4800 – 6400 Hz

Input for the filter bank is a speech frame pointed by the new_speech pointer of the G.722.2 speech encoder. Input values for the filter bank are scaled down by one bit. This ensures safe scaling, i.e. saturation can not occur during calculation of the filter bank.

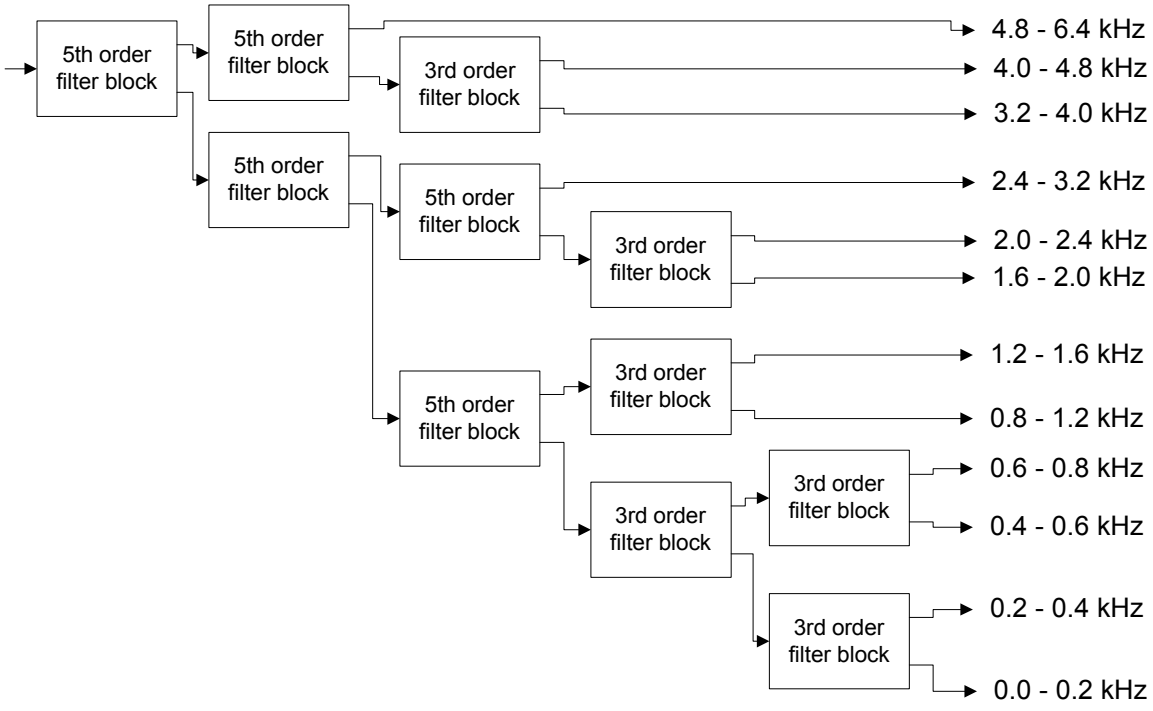


Figure 5. Filter bank

The filter bank consists of 5th and 3rd order filter blocks. Each filter block divides the input into high-pass and low-pass parts and decimates the sampling frequency by 2. The 5th order filter block is calculated as follows:

$$x_{lp}(i) = 0.5 * (A_1(x(2*i)) + A_2(x(2*i+1))) \quad (84a)$$

$$x_{hp}(i) = 0.5 * (A_1(x(2*i)) - A_2(x(2*i+1))) \quad (84b)$$

where

x(i) input signal for a filter block

$x_{lp}(i)$ low-pass component

$x_{hp}(i)$ high-pass component

The 3rd order filter block is calculated as follows:

$$x_{lp}(i) = 0.5 * (x(2*i+1) + A_3(x(2*i))) \quad (85a)$$

$$x_{hp}(i) = 0.5 * (x(2*i+1) - A_3(x(2*i))) \quad (85b)$$

The filters $A_1()$, $A_2()$, and $A_3()$ are first order direct form all-pass filters, whose transfer function is given by:

$$A(z) = \frac{C + z^{-1}}{1 + C * z^{-1}}, \quad (86)$$

where C is the filter coefficient.

Coefficients for the all-pass filters $A_1()$, $A_2()$, and $A_3()$ are COEFF5_1, COEFF5_2, and COEFF3, respectively.

Signal level is calculated at the output of the filter bank at each frequency band as follows:

$$level(n) = \sum_{i=START_n}^{END_n} |x_n(i)|, \quad (87)$$

where:

n index for the frequency band

$x_n(i)$ sample i at the output of the filter bank at frequency band n

$$START_n = \begin{cases} -6, & 1 \le n \le 4 \\ -12, & 5 \le n \le 8 \\ -24, & 9 \le n \le 11 \\ -48, & n = 12 \end{cases}$$

$$END_n = \begin{cases} 7, & 1 \le n \le 4 \\ 15, & 5 \le n \le 8 \\ 31, & 9 \le n \le 11 \\ 63, & n = 12 \end{cases}$$

Negative indices of $x_n(i)$ refer to the previous frame.

9.2.2 Tone detection

The purpose of the tone detection function is to detect information tones, vowel sounds and other periodic signals. The tone detection uses normalized open-loop pitch gains (ol_gain), which are received from the speech encoder. If the pitch gain is higher than the constant TONE_THR, tone is detected and the tone flag is set:

```
if (ol_gain > TONE_THR)
    tone_flag = 1
```

The open-loop pitch search and correspondingly the tone flag is computed twice in each frame, except for mode 6.60 kbit/s, where it is computed only once.

9.2.3 VAD decision

The block diagram of the VAD decision algorithm is shown in Figure 6.

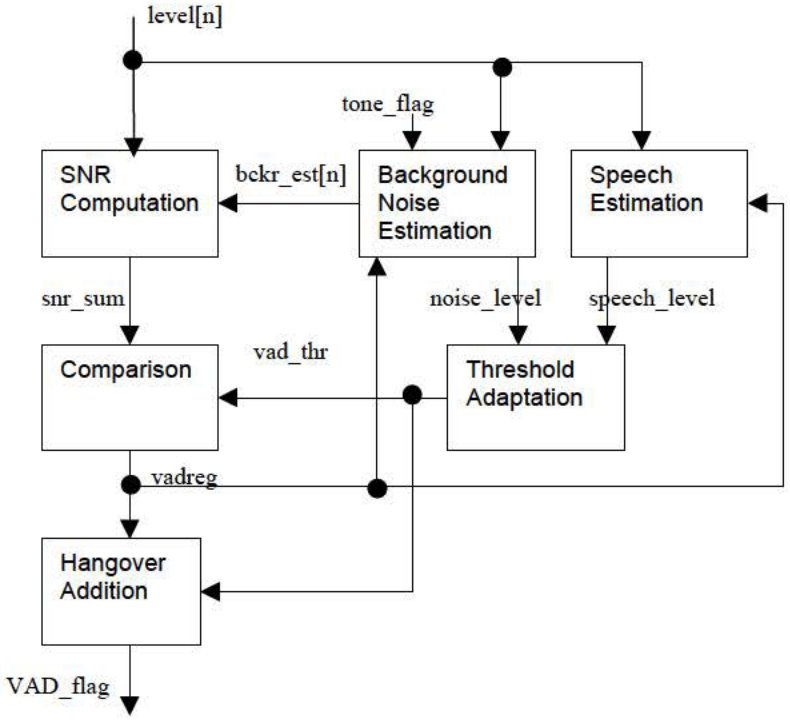


Figure 6. Simplified block diagram of the VAD decision algorithm

Power of the input frame is calculated as follows:

$$frame\_pow = \sum_{i=0}^{FRAME\_LEN} s(i) * s(i), \tag{88}$$

where samples s(i) of the input frame are pointed by the new_speech pointer of the speech encoder. Variable pow_sum is sum of the powers of the current and previous frames. If pow_sum is lower than the constant POW_TONE_THR, tone-flag is set to zero.

The difference between the signal levels of the input frame and the background noise estimate is calculated as follows:

$$snr\_sum = \sum_{n=1}^{12} MAX(1.0, \frac{level[n]}{bckr\_est[n]})^2, \tag{89}$$

where:

level[n] signal level at band n

bckr_est[n] level of background noise estimate at band n

VAD decision is made by comparing the variable snr_sum to a threshold. The threshold (vad_thr) is adapted to get desired sensitivity depending on estimated speech and background noise levels.

Average background noise level is calculated by adding noise estimates at each band except the lowest band:

$$noise\_level = \sum_{n=2}^{12} bckr\_est[n] \tag{90}$$

If SNR is lower that the threshold (MIN_SPEECH_SNR), speech level is increased as follows:

```
If (speech_level/noise_level < MIN_SPEECH_SNR)
    Speech_level = MIN_SPEECH_SNR * noise_level
```

Logarithmic value for noise estimate is calculated as follows:

$$i\log 2\_noise\_level = \log_2 (noise\_level) \tag{91}$$

Before logarithmic value from the speech estimate is calculated, MIN_SPEECH_SNR*noise_level is subtracted from the speech level to correct its value in low SNR situations.

$$i\log 2\_speech\_level = \log_2 (speech\_level - MIN\_SPEECH\_SNR * noise\_level) \tag{92}$$

Threshold for VAD decision is calculated as follows:

$$Vad\_thr = NO\_SLOPE * (ilog2\_noise\_level - NO\_P1) + THR\_HIGH + MIN(SP\_CH\_MAX, MAX(SP\_CH\_MIN, SP\_CH\_MIN + SP\_SLOPE * (ilog2\_speech\_level - SP\_P1))), \tag{93}$$

where NO_SLOPE, SP_SLOPE, NO_P1, SP_P1, THR_HIGH, SP_CH_MAX and SP_CH_MIN are constants.

The variable vadreg indicates intermediate VAD decision and it is calculated as follows:

```
if (snr_sum > vad_thr)
    vadreg = 1
else
    vadreg = 0
```

9.2.3.1 Hangover addition

Before the final VAD flag is given, a hangover is added. The hangover addition helps to detect low power endings of speech bursts, which are subjectively important but difficult to detect.

VAD flag is set to "1" if less that hang_len frames with "0" decision have been elapsed since burst_len consecutive "1" decisions have been detected. The variables hang_len and burst_len are computed using vad_thr as follows:

$$hang\_len = MAX(HANG\_LOW, (HANG\_SLOPE * (vad\_thr - HANG\_P1) + HANG\_HIGH)) \tag{94}$$

$$burst\_len = BURST\_SLOPE * (vad\_thr - BURST\_P1) + BURST\_HIGH) \tag{95}$$

The power of the input frame is compared to a threshold (VAD_POW_LOW). If the power is lower, the VAD flag is set to "0" and no hangover is added. The VAD_flag is calculated as follows:

```
Vad_flag = 0;
if (pow_sum < VAD_POW_LOW)
    burst_count = 0
    hang_count = 0
else
```

```
if (vadreg = 1)
    burst_count = burst_count + 1
    if (burst_count >= burst_len)
        hang_count = hang_len
    VAD_flag = 1
else
    burst_count = 0
    if (hang_count > 0)
        hang_count = hang_count - 1
        VAD_flag=1
```

9.2.3.2 Background noise estimation

Background noise estimate (bckr_est[n]) is updated using amplitude levels of the previous frame. Thus, the update is delayed by one frame to avoid undetected start of speech bursts to corrupt the noise estimate. The update speed for the current frame is selected using intermediate VAD decisions (vadreg) and stationarity counter (stat_count) as follows:

```
if (vadreg for the last 4 frames has been zero)
    alpha_up = ALPHA_UP1
    alpha_down = ALPHA_DOWN1
else if (stat_count = 0)
    alpha_up = ALPHA_UP2
    alpha_down = ALPHA_DOWN2
else
    alpha_up = 0
    alpha_down = ALPHA3
```

The variable stat_count indicates stationary and its purpose is explained later in this subclause. The variables alpha_up and alpha_down define the update speed for upwards and downwards, respectively. The update speed for each band "n" is selected as follows:

if ($bckr\_est_m[n] < level_{m-1}[n]$)
    alpha[n] = alpha_up
else
    alpha[n] = alpha_down

Finally, noise estimate is updated as follows:

$$bckr\_est_{m+1}[n] = (1.0 - alpha[n]) * bckr\_est_m[n] + alpha[n] * level_{m-1}[n], \quad (96)$$

where:

n index of the frequency band

m index of the frame

Level of the background estimate (bckr_est[n]) is limited between constants NOISE_MIN and NOISE_MAX.

If level of background noise increases suddenly, vadreg will be set to "1" and background noise is not normally updated upwards. To recover from this situation, update of the background noise estimate is enabled if the intermediate VAD decision (vadreg) is "1" for long enough time and spectrum is stationary. Stationary (stat_rat) is estimated using following equation:

$$stat\_rat = \sum_{n=1}^{12} \frac{MAX(\text{STAT\_THR\_LEVEL}, \text{MAX}(ave\_level_m[n], level_m[n]))}{MAX(\text{STAT\_THR\_LEVEL}, \text{MIN}(ave\_level_m[n], level_m[n]))}, \quad (97)$$

where:

STAT_THR_LEVEL a constant

n	index of the frequency band
m	index of the frame
ave_level	average level of the input signal

If the stationary estimate (stat_rat) is higher than a threshold, the stationary counter (stat_count) is set to the initial value defined by constant STAT_COUNT. If the signal is not stationary but speech has been detected (VAD decision is "1"), stat_count is decreased by one in each frame until it is zero.

```
if (5 last tone flags have been one)
    stat_count = STAT_COUNT
else
    if (8 last internal VAD decisions have been zero) OR (stat_rat > STAT_THR)
        stat_count = STAT_COUNT
    else
        if (vadreg) AND (stat_count ≠ 0)
            stat_count = stat_count – 1
```

The average signal levels (ave_level[n]) are calculated as follows:

$$ave\_level_{m+1}[n] = (1.0 - alpha) * ave\_level_{m}[n] + alpha * level_{m}[n] \quad (98)$$

The update speed (alpha) for the previous equation is selected as follows:

```
if (stat_count = STAT_COUNT)
    alpha = 1.0
else if (vadreg = 1)
    alpha=ALPHA5
else
    alpha = ALPHA4
```

9.2.3.3 Speech level estimation

First, full-band input level is calculated by summing input levels in each band except the lowest band as follows:

$$in\_level = \sum_{n=2}^{12} level[n] \quad (99)$$

A frame is assumed to contain speech if its level if high enough (MIN_SPEECH_LEVEL1), and the intermediate VAD flag (vadreg) is set or the input level is higher than the current speech level estimate. Maximum level (sp_max) from SP_EST_COUNT frames is searched. If the SP_ACTIVITY_COUNT number of speech frames is located in within SP_EST_COUNT number of frames, speech level estimate is updated by the maximum signal level (sp_max). The pseudocode for the speech level estimation is as follows:

```
If  (SP_ACTIVITY_COUNT > SP_EST_COUNT – sp_est_cnt + sp_max_cnt)
    sp_est_cnt  = 0
    sp_max_cnt = 0
    sp_max = 0
sp_est_cnt = sp_est_cnt + 1
if (in_level > MIN_SPEECH_LEVEL1) AND ((vadreg = 1) OR (in_level > speech_level))
    sp_max_cnt = sp_max_cnt + 1
    sp_max = MAX(sp_max, in_level)
    if (sp_max_cnt > SP_ACTIVITY_COUNT)
        if (sp_max > MIN_SPEECH_LEVEL2)
            if (sp_max > speech_level)
                speech_level = speech_level + SP_ALPHA_UP * (sp_max – speech_level)
            else
                speech_level = speech_level + SP_ALPHA_DOWN * (sp_max – speech_level)
        sp_max_cnt = 0
        sp_max = 0
        sp_est_cnt = 0
```

10 Bibliography (informative)

[2] M.R. Schroeder and B.S. Atal, "Code-Excited Linear Prediction (CELP): High quality speech at very low bit rates," in Proc. *ICASSP'85*, pp. 937-940, 1985.

[3] L.R. Rabiner and R.W. Schaefer. *Digital processing of speech signals*. Prentice-Hall Int., 1978.

[4] F. Itakura, "Line spectral representation of linear predictive coefficients of speech signals," *J. Acoust. Soc. Amer.*, vol. 57, Supplement no. 1, S35, 1975.

[5] Y. Bistritz and S. Pellerm, "Immittance Spectral Pairs (ISP) for speech encoding," in *Proc. ICASSP'93*, pp. II-9 - II-12.

[6] K.K Paliwal and B.S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame", *IEEE Trans. Speech and Audio Processing*, vol. 1, no 1, pp. 3-14, 1993.

[7] P. Kabal and R.P. Ramachandran, "The computation of line spectral frequencies using Chebyshev polynomials", *IEEE Trans. on ASSP*, vol. 34, no. 6, pp. 1419-1426, Dec. 1986.

[8] K. Järvinen, J. Vainio, P. Kapanen, T. Honkanen, P. Haavisto, R. Salami, C. Laflamme, and J.-P. Adoul, "GSM enhanced full rate speech codec", in *Proc. ICASSP'97*, pp. 771-774.

[9] T. Honkanen, J. Vainio, K. Järvinen, P. Haavisto, R. Salami, C. Laflamme, and J.-P. Adoul, "Enhanced full rate speech codec for IS-136 digital cellular system", in *Proc. ICASSP'97*, pp. 731-734.

[10] R. Hagen, E. Ekudden, B. Johansson, and W.B. Kleijn, "Removal of sparse-excitation artifacts in CELP", in *Proc. ICASSP'98*, pp. I-145-I-148.

[11] 3GPP TS 26.171 : "AMR Wideband Speech Codec; General description".

[12] 3GPP TS 26.190 : "AMR Wideband Speech Codec; Transcoding functions".

[13] 3GPP TS 26.173 : "AMR Wideband Speech Codec; ANSI-C code".

[14] 3GPP TS 26.174 : "AMR Wideband Speech Codec; Test sequences".

[15] 3GPP TS 26.193 : "AMR Wideband Speech Codec; Source Controlled Rate operation".

[16] 3GPP TS 26.194 : "AMR Wideband Speech Codec; Voice Activity Detection (VAD)".

[17] 3GPP TS 26.192 : "AMR Wideband Speech Codec; Comfort Noise Aspects".

[18] 3GPP TS 26.191 : "AMR Wideband Speech Codec; Error Concealment of Lost Frames.

[19] 3GPP TS 26.201 : "AMR Wideband Speech Codec; Frame Structure".

[20] 3GPP TR 26.901 : "AMR Wideband Speech Codec; Performance characterisation".

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.722.2

Appendix I
(01/2002)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Digital terminal equipments – Coding of analogue signals by methods other than PCM

Wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB)

Appendix I: Error concealment of erroneous or lost frames

CAUTION !

PREPUBLISHED RECOMMENDATION

This prepublication is an unedited version of a recently approved Recommendation. It will be replaced by the published version after editing. Therefore, there will be differences between this prepublication and the published version.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU [had/had not] received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

WIDEBAND CODING OF SPEECH AT AROUND 16 KBIT/S USING ADAPTIVE MULTI-RATE WIDEBAND (AMR-WB)

APPENDIX I

ERROR CONCEALMENT OF ERRONEOUS OR LOST FRAMES

Summary

This document specifies a non-normative example solution for concealment of erroneous or lost frames for the G.722.2 AMR-WB codec.

The concealment operations described here were also adopted by 3GPP in 3GPP specification TS 26.191.

Source

Appendix I to ITU-T Recommendation G.722.2 was prepared by ITU-T Study Group 16 (2001-2004) and was approved under the WTSC Resolution No. 1 procedure on 13 January 2002.

Contents

Recommendation G.722.2

WIDEBAND CODING OF SPEECH AT AROUND 16 KBIT/S USING ADAPTIVE MULTI-RATE WIDEBAND (AMR-WB)

APPENDIX I

ERROR CONCEALMENT OF ERRONEOUS OR LOST FRAMES

(Geneva, 2001)

I.1 Scope

This specification defines an example procedure for error concealment, also termed frame substitution and muting procedure, for use by the AMR-WB speech codec receiving end when one or more erroneous/lost speech or lost Silence Descriptor (SID) frames are received.

The algorithm specified in this Appendix is available as part of the ANSI-C code in Annex C/G.722.2. In case of discrepancy between the specification in this Appendix and the fixed point computational description of this algorithm contained in Annex C/G.722.2, the description in Annex C/G.722.2 will prevail.

I.2 Definitions and abbreviations

I.2.1 Definitions

For the purposes of this document, the following definition applies:

N-point median operation: Consists of sorting the N elements belonging to the set for which the median operation is to be performed in an ascending order according to their values, and selecting the (int (N/2) + 1) -th largest value of the sorted set as the median value.

I.2.2 Abbreviations

For the purposes of this document, the following abbreviations apply:

AMR-WB	Adaptive Multi Rate - WideBand
AN	Access Network
BFI	Bad Frame Indication from AN
BSI_netw	Bad Sub-block Indication obtained from AN interface CRC checks
prevBFI	Bad Frame Indication of previous frame
RX	Receive
SCR	Source Controlled Rate (operation)
SID	Silence Descriptor frame (Background noise)
CRC	Cyclic Redundancy Check
ECU	Error Concealment Unit
BFH	Bad Frame Handling
medianN	N-point median operation

I.3 General

The purpose of the error concealment procedure is to conceal the effect of erroneous/lost AMR-WB speech frames. The purpose of muting the output in the case of several erroneous/lost frames is to indicate the breakdown of the channel to the user and to avoid generating possible annoying sounds as a result from the error concealment procedure.

The network shall indicate erroneous/lost speech or lost SID frames by setting the RX_TYPE values [Annex B/G.722.2] to SPEECH_BAD, SID_BAD or SPEECH_LOST. If these flags are set, the speech decoder shall perform parameter substitution to conceal errors.

The example solution provided in section I.5 apply only to bad frame handling on a complete speech frame basis. Sub-frame based error concealment may be derived using similar methods.

I.4 Requirements

I.4.1 Error detection

If the most sensitive bits of the AMR-WB speech data are received in error, the network shall indicate RX_TYPE = SPEECH_BAD in which case the BFI flag is set. When the frame is not received, the network shall indicate RX_TYPE = RX_SPEECH_LOST in which case the BFI flag is set as well. If a SID frame is received in error, the network shall indicate RX_TYPE = SID_BAD..

I.4.2 Erroneous or lost speech frames

Normal decoding of erroneous/lost speech frames would result in very unpleasant noise effects. In order to improve the subjective quality, erroneous/lost speech frames shall be substituted with either a repetition or an extrapolation of the previous good speech frame(s). This substitution is done so that it gradually will decrease the output level, resulting in silence at the output. Subclause I.5 provides example solution.

I.4.3 First lost SID frame

A lost SID frame shall be substituted by using the SID information from earlier received valid SID frames and the procedure for valid SID frames be applied as described in Annex B/G.722.2.

I.4.4 Subsequent lost SID frames

For many subsequent lost SID frames, a muting technique shall be applied to the comfort noise that will gradually decrease the output level. For subsequent lost SID frames, the muting of the output shall be maintained. Subclause I.5 provides example solutions.

I.5 Example ECU/BFH Solution

I.5.1 State Machine

This example solution for substitution and muting is based on a state machine with seven states (Figure I.1).

The system starts in state 0. Each time a bad frame is detected, the state counter is incremented by one and is saturated when it reaches 6. Each time a good speech frame is detected, the state counter is right-shifted by one. The state indicates the quality of the channel: the larger the value of the state counter, the worse the channel quality is. The control flow of the state machine can be described by the following C code (**BFI** = bad frame indicator, **State** = state variable):

```
if(BFI != 0 )
    State = State + 1;
    if(State > 6)
        State = 6;
else
    State = State >> 1;
```

In addition to this state machine, the **Bad Frame Flag** from the previous frame is checked (**prevBFI**). The processing depends on the value of the **State**-variable. In states 0 and 6, the processing depends on the **BFI** flag.

The state machine is summarized in Figure I.1.

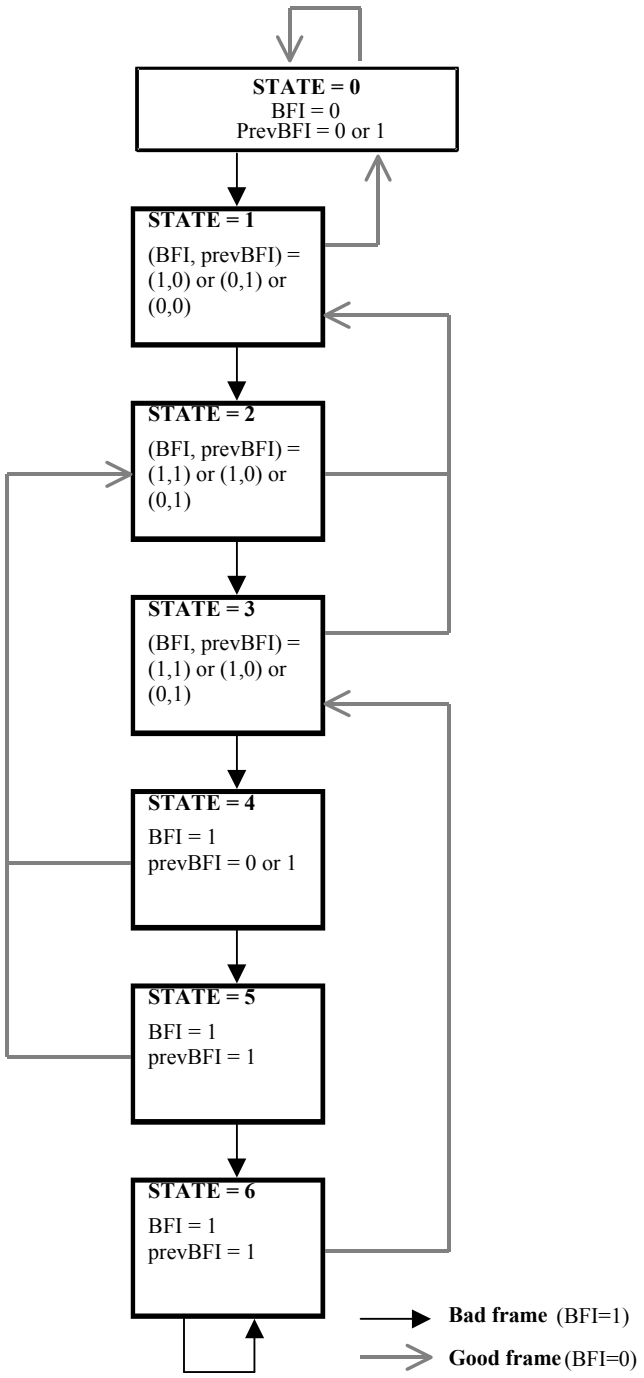


Figure I.1: State machine for controlling the bad frame substitution

I.5.2 Substitution and muting of erroneous/lost speech frames

I.5.2.1 BFI = 0, prevBFI = 0, State = 0 or 1

No error is detected in the received or in the previous received speech frame. The received speech parameters are used normally in the speech synthesis. The current frame of speech parameters is saved.

I.5.2.2 BFI = 0, prevBFI = 1, State = 0 to 3

No error is detected in the received speech frame but the previous received speech frame was bad. The LTP gain is used normally in the speech synthesis and fixed codebook gain are limited below the values used for the last received good subframe:

$$g^c(n) = \begin{cases} g^c_{received} & , g^c_{received} \le 100 \; or \; g^c_{received} \le g^c(n-1) \times 1.25 \\ 1.25 * g^c(n-1) & , otherwise \end{cases} \quad (1)$$

where

$g^c_{received}$ = current decoded fixed codebook-gain

$g^c(n-1)$ = fixed codebook gain used for the last good subframe (BFI = 0)

$g^c(n)$ = fixed codebook gain to be used for the current frame.

The rest of the received speech parameters are used normally in the speech synthesis. The current frame of speech parameters is saved.

I.5.2.3 BFI = 1, prevBFI = 0 or 1, State = 1...6

An error is detected in the received speech frame and the substitution and muting procedure is started.

I.5.2.3.1 LTP gain & fixed codebook gain concealment when RX_FRAMETYPE = SPEECH_BAD

The LTP gain g^p and fixed codebook gain g^c are replaced by attenuated values from the previous subframes:

$$g^p = P^p(state) * median5(g^p(n-1),...,g^p(n-5)) \quad (2)$$

$$g^c = \begin{cases} P^c(state) * median5(g^c(n-1),...,g^c(n-5)) & , \text{VAD\_HIST} \le 2 \\ median5(g^c(n-1),...,g^c(n-5)) & , \text{VAD\_HIST} > 2 \end{cases} \quad (3)$$

where:

g^p = current decoded LTP gain,

g^c = current decoded fixed codebook gain,

$g^p(n-1),...,g^p(n-5)$ = LTP gains used for the last 5 subframes,

$g^c(n-1),...,g^c(n-5)$ = fixed codebook gains used for the last 5 subframes,

median5() = 5-point median operation,

$P^p(state)$ = attenuation factor ($P^p(1)$ = 0.98, $P^p(2)$ = 0.96, $P^p(3)$ = 0.75, $P^p(4)$ = 0.23, $P^p(5)$ = 0.05, $P^p(6)$ = 0.01),

$P^c(state)$ = attenuation factor ($P^c(1)$ = 0.98, $P^c(2)$ = 0.98, $P^c(3)$ = 0.98, $P^c(4)$ = 0.98, $P^c(5)$ = 0.98, $P^c(6)$ = 0.70),

state = state number {0..6},

VAD_HIST is number of consecutive VAD=0 decisions.

The higher the state value is, the more the gains are attenuated. Also the memory of the predictive fixed codebook gain is updated by using the average value of the past four values in the memory:

$$ener(0) = \frac{1}{4}\left[\sum_{i=1}^{4} ener(n-i)\right] - 3 \qquad (4)$$

I.5.2.3.2 LTP gain & fixed codebook gain concealment when RX_FRAMETYPE = SPEECH_LOST

The LTP gain g^p and fixed codebook gain g^c are replaced by attenuated values from the previous subframes:

$$g^p = P^p(state) * median5(g^p(n-1),...,g^p(n-5)) \qquad (5)$$

$$g^c = \begin{cases} P^c(state) * median5(g^c(n-1),...,g^c(n-5)) & , \text{VAD\_HIST} \le 2 \\ median5(g^c(n-1),...,g^c(n-5)) & , \text{VAD\_HIST} > 2 \end{cases} \qquad (6)$$

where:

g^p = current decoded LTP gain,
g^c = current decoded fixed codebook gain,
$g^p(n-1),...,g^p(n-5)$ = LTP gains used for the last 5 subframes,
$g^c(n-1),...,g^c(n-5)$ = fixed codebook gains used for the last 5 subframes,
median5() = 5-point median operation,
$P^p(state)$ = attenuation factor ($P^p(1)$ = 0.95, $P^p(2)$ = 0.90, $P^p(3)$ = 0.75, $P^p(4)$ = 0.23, $P^p(5)$ = 0.05, $P^p(6)$ = 0.01),
$P^c(state)$ = attenuation factor ($P^c(1)$ = 0.50, $P^c(2)$ = 0.25, $P^c(3)$ = 0.25, $P^c(4)$ = 0.25, $P^c(5)$ = 0.15, $P^c(6)$ = 0.01),
state = state number {0..6},
VAD_HIST is number of consecutive VAD=0 decisions.

The higher the state value is, the more the gains are attenuated. Also the memory of the predictive fixed codebook gain is updated by using the average value of the past four values in the memory:

$$ener(0) = \frac{1}{4}\left[\sum_{i=1}^{4} ener(n-i)\right] - 3 \qquad (7)$$

I.5.2.3.3 ISF concealment

The past ISFs are shifted towards their partly adaptive mean:

$$ISF_q(i) = a * past\_ISF_q(i) + (1-a) * ISF_{mean}(i) \qquad i = 0..16 \qquad (8)$$

where

a = 0.9,
$ISF_q(i)$ is ISF-vector for a current frame,
$past\_ISF_q(i)$ is ISF-vector from the previous frame,
$ISF_{mean}(i)$ vector is combination of adaptive mean and constant mean ISF-vectors in the following manner:

$$ISF_{mean}(i) = b * ISF_{const\_mean}(i) + (1-b) * ISF_{adaptive\_mean}(i), \qquad i = 0..16 \qquad (9)$$

where

b = 0.75,

$ISF_{adaptive\_mean}(i) = \frac{1}{3}\sum_{i=0}^{2} past\_ISF_q(i)$ and is updated whenever BFI =0.

$ISF_{const\_mean}(i)$ is a vector containing long time average of ISF-vectors.

I.5.2.3.4 LTP-lag concealment

The histories of five last good LTP-lags and LTP-gains are used for finding the best method to update.

I.5.2.3.4.1 LTP-lag concealment when RX_FRAMETYPE = SPEECH_BAD

The usability of the received LTP lag (Q_{lag}) is defined as follows: (Predicts if the received lag is most probably very close to one that was sent and therefore its usage should not introduce any bad artifacts)

$$Q_{lag} = \begin{cases} 1 & ,T_{dif} < 10 \text{ and } \mathrm{T}_{\min} - 5 < T_{received} < \mathrm{T}_{\min} + 5 \\ 1 & ,\mathrm{g}^{\mathrm{p}}(n-1) > 0.5 \text{ and } \mathrm{g}^{\mathrm{p}}(n-2) > 0.5 \text{ and } \mathrm{T}\ (\mathrm{n}-1) - 10 < T_{received} < \mathrm{T}\ (\mathrm{n}-1) + 10 \\ 1 & ,g^{p}_{\min} < 0.4 \text{ and } g^{p}(n-1) = g^{p}_{\min} \text{ and } \mathrm{T}_{\min} < T_{received} < \mathrm{T}_{\max} \\ 1 & ,T_{dif} < 70 \text{ and } \mathrm{T}_{\min} < T_{received} < \mathrm{T}_{\max} \\ 1 & ,T_{mean} < T_{received} < \mathrm{T}_{\max} \\ 0 & ,otherwise \end{cases} \quad (10)$$

where:

$T(n-1)$ is LTP lag from the previous good frame,

$T_{dif} = \left|T_{received} - T(n-1)\right|$,

$T_{\min} = \min(T_{buffer})$,

$T_{\max} = \max(T_{buffer})$,

$T_{received}$ is received lag,

$g^{p}_{\min} = \min(g^{p}_{buffer})$,

g^{p} is LTP gain of the current frame,

g^{p} (-1) is LTP gain of the previous good frame,

g^{p} (-2) is LTP gain of the frame before previous good frame,

$T_{mean} = average(T_{buffer})$

LPT lag value for the current frame is defined as follows:

$$T = \begin{cases} T_{received} & ,\mathrm{Q}_{\mathrm{lag}} = 1 \\ \frac{1}{3}\sum(T_{\max} + T_{\max-1} + T_{\max-2}) + RND(T_{\max} - T_{\max-2}) & ,\mathrm{Q}_{\mathrm{lag}} = 0 \end{cases} \quad (11)$$

where:

$T_{\max} = \max(T_{buffer})$,

$T_{\max-1}$ is second largest value in T_{buffer} ,

$T_{\max-2}$ is second largest value in T_{buffer} ,

$RND(x)$ is random value generated to range $\left[-\frac{x}{2},+\frac{x}{2}\right]$

I.5.2.3.4.2 LTP-lag concealment when RX_FRAMETYPE = SPEECH_LOST

The usability of the LTP lag from last good frame (Q_{lag_t-1}) is defined as follows: (Predicts if the received lag is most probably very close to one that was sent and therefore its usage should not introduce any bad artifacts)

$$Q_{lag\_t-1} = \begin{cases} 1 & ,g^{p}_{\min} > 0.5 \text{ and } \mathrm{T_{dif}} < 10 \\ 1 & ,\mathrm{g^{p}}(n-1) > 0.5 \text{ and } \mathrm{g^{p}}(n-2) > 0.5 \\ 0 & ,otherwise \end{cases} \quad (12)$$

where:

$g^{p}_{\min} = \min(g^{p}_{buffer})$,

g^{p} (n-1) is LTP gain of the previous good frame,

g^{p} (n-2) is LTP gain of the frame before previous good frame

LPT lag value for the current frame is defined as follows:

$$T = \begin{cases} T(n-1) & ,\mathrm{Q_{lag\_t\text{-}1}} = 1 \\ \frac{1}{3}\sum(T_{\max} + T_{\max-1} + T_{\max-2}) + RND(T_{\max} - T_{\max-2}) & ,\mathrm{Q_{lag\_t\text{-}1}} = 0 \end{cases} \quad (13)$$

where:

$T(n-1)$ is LTP lag from the previous good frame,

$T_{\max} = \max(T_{buffer})$,

$T_{\max-1}$ is second largest value in T_{buffer} ,

$T_{\max-2}$ is second largest value in T_{buffer} ,

$RND(x)$ is random value generated to range $\left[-\frac{x}{2},+\frac{x}{2}\right]$

I.5.2.4 Innovation sequence

When RX_FRAMETYPE = SPEECH_BAD, the received fixed codebook innovation pulses from the erroneous frame are used as they are received.

When RX_FRAMETYPE = SPEECH_LOST, the received fixed codebook innovation pulses from the erroneous frame are not used and the fixed codebook innovation vector is filled with random signal (values limited to range [-1, +1]).

I.5.3 Substitution and muting of lost SID frames

In the speech decoder a single frame classified as SID_BAD shall be substituted by the last valid SID frame information and the procedure for valid SID frames be applied. If the time between SID information updates (updates are specified by SID_UPDATE arrivals and occasionally by SID_FIRST arrivals) is greater than one second this shall lead to attenuation.

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.722.2

Annex C

(01/02)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Digital transmission systems – Terminal equipments – Coding of analogue signals by methods other than PCM

Wideband coding of speech at around 16 kbit/s using Adaptive Multi-Rate Wideband (AMR-WB)

Annex C: Fixed-point C-code

ITU-T Recommendation G.722.2 – Annex C

(Previously CCITT Recommendation)

ITU-T G-SERIES RECOMMENDATIONS

TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

INTERNATIONAL TELEPHONE CONNECTIONS AND CIRCUITS	G.100–G.199
INTERNATIONAL ANALOGUE CARRIER SYSTEM	
GENERAL CHARACTERISTICS COMMON TO ALL ANALOGUE CARRIER-TRANSMISSION SYSTEMS	G.200–G.299
INDIVIDUAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON METALLIC LINES	G.300–G.399
GENERAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON RADIO-RELAY OR SATELLITE LINKS AND INTERCONNECTION WITH METALLIC LINES	G.400–G.449
COORDINATION OF RADIOTELEPHONY AND LINE TELEPHONY	G.450–G.499
TESTING EQUIPMENTS	
TRANSMISSION MEDIA CHARACTERISTICS	G.600–G.699
DIGITAL TRANSMISSION SYSTEMS	
TERMINAL EQUIPMENTS	G.700–G.799
General	G.700–G.709
Coding of analogue signals by pulse code modulation	G.710–G.719
Coding of analogue signals by methods other than PCM	**G.720–G.729**
Principal characteristics of primary multiplex equipment	G.730–G.739
Principal characteristics of second order multiplex equipment	G.740–G.749
Principal characteristics of higher order multiplex equipment	G.750–G.759
Principal characteristics of transcoder and digital multiplication equipment	G.760–G.769
Operations, administration and maintenance features of transmission equipment	G.770–G.779
Principal characteristics of multiplexing equipment for the synchronous digital hierarchy	G.780–G.789
Other terminal equipment	G.790–G.799
DIGITAL NETWORKS	G.800–G.899
DIGITAL SECTIONS AND DIGITAL LINE SYSTEM	G.900–G.999

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION G.722.2

WIDEBAND CODING OF SPEECH AT AROUND 16 KBIT/S USING ADAPTIVE MULTI-RATE WIDEBAND (AMR-WB)

ANNEX C

FIXED-POINT C-CODE

Summary

This document specifies the bit-exact ANSI C-code implementation of the AMR-WB algorithm specified in Recommendation G.722.2, its Annexes A and B, and its Appendix I (non-normative).

The C code specified in this Annex was also adopted by 3GPP in 3GPP specification TS 26.173.

Source

Annex C to ITU-T Recommendation G.722.2 was prepared by ITU-T Study Group 16 (2001-2004) and was approved under the WTSC Resolution No. 1 procedure on 13 January 2002.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation the term *recognized operating agency (ROA)* includes any individual, company, corporation or governmental organization that operates a public correspondence service. The terms *Administration, ROA* and *public correspondence* are defined in the *Constitution of the ITU (Geneva, 1992)*.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Recommendation G.722.2

WIDEBAND CODING OF SPEECH AT AROUND 16 KBIT/S USING ADAPTIVE MULTI-RATE WIDEBAND (AMR-WB)

ANNEX C

Fixed-point C-code

(Geneva, 2001)

C.1 C code structure

This Annex gives an overview of the structure of the bit-exact C code for the correct implementation of the G.722.2 main body, its Annex A (Comfort noise aspects), Annex B (Source Controlled Rate operation), and Appendix I (Error concealment of erroneous or lost frames). Itprovides an overview of the contents and organization of the C code attached to this document. In case of discrepancy between the description given in the several parts of Recommendation G.722.2 (including its Annexes A, B and Appendix I) and the ANSI C-source code, the algorithm description of the ANSI C code shall prevail.

The C code has been verified on a number of systems (please see the readme.txt file for the complete list).

ANSI-C was selected as the programming language because portability was desirable.

C.1.1 Contents of the C source code

The C code distribution has all files in the root level.

The distributed files with suffix "c" contain the source code and the files with suffix "h" are the header files. The ROM data is contained mostly in files with suffix "tab".

The C code distribution also contains one speech coder installation verification data file, "spch_dos.inp". The reference encoder output file is named "spch_dos.cod", the reference decoder input file is named "spch_dos.dec" and the reference decoder output file is named "spch_dos.out". These four files are formatted such that they are correct for an IBM PC/AT compatible computer. The same files with reversed byte order of the 16 bit words are named "spch_unx.inp", "spch_unx.cod", "spch_unx."dec" and "spch_unx.out", respectively.

Final verification of bit-exactness is to be performed using the Adaptive Multi-Rate Wideband test sequences described in Annex D/G.722.2.

Makefiles are provided for the platforms in which the C code has been verified (listed above). Once the software is installed, this directory will have a compiled version of *encoder* and *decoder* (the bit-exact C executables of the speech codec) and all the object files.

C.1.2 Program execution

The Adaptive Multi-Rate Wideband codec is implemented in two programs:

- (*encoder*) speech encoder;
- (*decoder*) speech decoder.

The programs should be called like:

- encoder [encoder options] <speech input file> <parameter file>;
- decoder <parameter file> <speech output file>.

The speech files contain 16-bit linear encoded PCM speech samples and the parameter files contain encoded speech data and some additional flags.

The encoder and decoder options will be explained by running the applications without input arguments. See the file readme.txt for more information on how to run the *encoder* and *decoder* programs.

C.1.3 Code hierarchy

Tables C.1 to C.3 are call graphs that show the functions used in the speech codec, including the functions of VAD, DTX, and comfort noise generation.

Each column represents a call level and each cell a function. The functions contain calls to the functions in rightwards neighbouring cells. The time order in the call graphs is from the top downwards as the processing of a frame advances. All standard C functions: printf(), fwrite(), etc. have been omitted. Also, no basic operations (add(), L_add(), mac(), etc.) or double precision extended operations (e.g. L_Extract()) appear in the graphs. The initialization of the static RAM (i.e. calling the _init functions) is also omitted.

The basic operations are not counted as extending the depth, therefore the deepest level in this software is level 6.

The encoder call graph is broken down into two separate call graphs, Table C.1 to C.2.

Table C.1: Speech encoder call structure

coder	Copy			
	Decim_12k8	Down_samp	Interpol (function)	
		Copy		
	Set_zero			
	HP50_12k8			
	Scale_sig			
	wb_vad	Filter_bank	Filter5	
			Filter3	
			Level_calculation	
		vad_decision	llog2	
			Noise_estimate_update	update_cntrl
			hangover_addition	
		Estimate_Speech		
	tx_dtx_handler			
	Parm_serial			
	Autocorr			
	Lag_window			
	Levinson			
	Az_isp	Chebps2		
	Int_isp	Isp_Az	Get_isp_pol	
	Isp_isf			
	Gp_clip_test_isf			
	Weight_a			
	Residu			
	Deemph2			
	LP_Decim2			
	Scale_mem_Hp_wsp			
	Pitch_med_ol	Hp_wsp		
		Isqrt_n		
	wb_vad_tone_detection			
	Med_olag	median5		
	dtx_buffer	Copy		
	dtx_enc	Find_frame_indices		
		Aver_isf_history		
		Qisf_ns	Sub_VQ	
			Disf_ns	Reorder_isf
		Parm_serial		
		Pow2		
		Random		
		Dot_product12		
		Isqrt_n		
	Isf_isp			
	Isp_Az	Get_isp_pol		
	Synthesis	Copy		
		Syn_filt_32		
		Deemph_32		
		HP50_12k8		
		Random		
		Scale_sig		
		Dot_product12		
		Isqrt_n		
		HP400_12k8		
		Weight_a		
		Syn_filt		
		Filt_6k_7k		
	Reset_encoder	Set_zero		
		Init_gp_clip		
		Init_Phase_dispersion	Set_zero	
	Qpisf_2s_36b	VQ_stage1		
		Sub_VQ		
		Dpisf_2s_36b	Reorder_isf	
	Qpisf_2s_46b	VQ_stage1		
		Sub_VQ		
		Dpisf_2s_46b	Reorder_isf	
	Syn_filt			
	Preemph2			
	Pitch_fr4	Norm_Corr	Convolve	
			Isqrt_n	
		Interpol_4		
	Gp_clip			
	Pred_lt4			
	Convolve			
	G_pitch	Dot_product12		
	Updt_tar			
	Preemph			
	Pit_shrp			
	Cor_h_x			
	ACELP_2t64_fx	Dot_product12		
		Isqrt_n		
	ACELP_4t64_fx	See Table 2		
	Q_gain2	Dot_product12		
		Pow2		
	Gp_clip_test_gain_pit			
	voice_factor	Dot_product12		

Table C.2: ACELP_4t64_fx call structure

ACELP_4t64_fx	Dot_product12			
	Isqrt_n			
	cor_h_vec			
	search_ixiy			
	quant_1p_N1			
	quant_2p_2N1			
	quant_3p_3N1	quant_2p_2N1		
		quant_1p_N1		
	quant_4p_4N	quant_4p_4N1	Quant_2p_2N1	
		quant_1p_N1		
		quant_3p_3N1	Quant_2p_2N1	
			Quant_1p_N1	
		quant_2p_2N1		
	quant_5p_5N	quant_3p_3N1	Quant_2p_2N1	
			Quant_1p_N1	
		quant_2p_2N1		
	quant_6p_6N_2	quant_5p_5N	Quant_3p_3N1	quant_2p_2N1
				Quant_1p_N1
			quant_2p_2N1	
		quant_1p_N1		
		quant_4p_4N	quant_4p_4N1	quant_2p_2N1
			quant_1p_N1	
			quant_3p_3N1	quant_2p_2N1
				quant_1p_N1
			quant_2p_2N1	
		quant_2p_2N1		
		quant_3p_3N1	quant_2p_2N1	
			Quant_1p_N1	

Table C.3: Speech decoder call structure

decoder	Rx_dtx_handler				
	Dtx_dec	Copy			
		Disf_ns	Reorder_isf		
		Serial_parm			
		Pow2			
		Random			
		Dot_product12			
		Isqrt_n			
	Serial_parm				
	Isf_isp				
	Isp_Az	Get_isp_pol			
	Copy				
	Synthesis	Copy			
		Syn_filt_32			
		Deemph_32			
		HP50_12k8			
		Oversamp_16k	Copy		
			Up_samp	Interpol	
		Random			
		Scale_sig			
		Dot_product12			
		Isqrt_n			
		HP400_12k8			
		Isf_Extrapolation	Isf_isp		
		Isp_Az	Get_isp_pol		
		Weight_a			
		Syn_filt			
		Filt_6k_7k	Copy		
		Filt_7k	Copy		
	Reset_decoder	Set_zero			
		Init_Phase_dispersion	Set_zero		
	Dpisf_2s_36b	Reorder_isf			
	Dpisf_2s_46b	Reorder_isf			
	Int_isp	Isp_Az	Get_isp_pol		
	Lagconc	insertion_sort	Insert		
		Random			
	Pred_lt4				
	Random				
	DEC_ACELP_2t64_fx				
	DEC_ACELP_4t64_fx	dec_1p_N1			
		add_pulses			
		dec_2p_2N1			
		dec_3p_3N1	Dec_2p_2N1		
			dec_1p_N1		
		dec_4p_4N	dec_4p_4N1	dec_2p_2N1	
			dec_1p_N1		
			Dec_3p_3N1	Dec_2p_2N1	
				Dec_1p_N1	
			Dec_2p_2N1		
		dec_5p_5N	dec_3p_3N1	Dec_2p_2N1	
				Dec_1p_N1	
			Dec_2p_2N1		
		dec_6p_6N_2	Dec_5p_5N	dec_3p_3N1	Dec_2p_2N1
					Dec_1p_N1
				dec_2p_2N1	
			dec_1p_N1		
			dec_4p_4N	dec_4p_4N1	dec_2p_2N1
				dec_1p_N1	
				Dec_3p_3N1	Dec_2p_2N1
					Dec_1p_N1
				Dec_2p_2N1	
			dec_2p_2N1		
			dec_3p_3N1	Dec_2p_2N1	
				Dec_1p_N1	
	Preemph				
	Pit_shrp				
	D_gain2	Dot_product12			
		Isqrt_n			
		Median5			
		Pow2			
	Scale_sig				
	voice_factor	Dot_product12			
	Phase_dispersion	Set_zero			
	Agc2	Isqrt	Isqrt_n		
	Set_zero				
	Dtx_dec_activity_update	Copy			

C.1.4 Variables, constants and tables

The data types of variables and tables used in the fixed point implementation are signed integers in 2's complement representation, defined by:

- **Word16** 16 bit variable;
- **Word32** 32 bit variable.

C.1.4.1 Description of constants used in the C-code

This subclause contains a listing of all global constants defined in cnst h.

Table C.4: Global constants

Constant	Value	Description
L_TOTAL	384	total size of speech buffer.
L_WINDOW	384	window size in LP analysis
L_NEXT	64	Look-ahead size
L_FRAME	256	frame size in 12.8 kHz
L_FRAME16k	320	frame size in 16 kHz
L_SUBFR	64	Subframe size in 12.8 kHz
L_SUBFR16k	80	Subframe size in 16 kHz
NB_SUBFR	4	Number of subframes
M16k	20	order of LP filter in high-band synthesis in 6.60 mode
M	16	order of LP filter
L_FILT16k	15	Delay of down-sampling filter in 16 kHz
L_FILT	12	Delay of down-sampling filter in 12.8 kHz
GP_CLIP	15565	Pitch gain clipping
PIT_SHARP	27853	pitch sharpening factor
PIT_MIN	34	minimum pitch lag (all modes)
PIT_FR2	128	Minimum pitch lag with resolution ½
PIT_FR1_9b	160	Minimum pitch lag with resolution for 9 bit quantization
PIT_FR1_8b	92	Minimum pitch lag with resolution for 8 bit quantization
PIT_MAX	231	maximum pitch lag
L_INTERPOL	(16+1)	length of filter for interpolation
OPL_DECIM	2	Decimation in open-loop pitch analysis
PREEMPH_FAC	22282	preemphasis factor
GAMMA1	30147	Weighting factor (numerator)
TILT_FAC	22282	tilt factor (denominator)
Q_MAX	8	scaling max for signal
RANDOM_INITSEED	21845	random init value
L_MEANBUF	3	Size of ISF buffer
ONE_PER_MEANBUF	10923	Inverse of L_MEANBUF

C.1.4.2 Description of fixed tables used in the C-code

This section contains a listing of all fixed tables sorted by source file name and table name. All table data is declared as **Word16**.

Table C.5: Fixed tables

File	Table name	Length	Description
C4t64fx.c	Tipos	36	starting points of iterations
Cod_main.c	HP_gain	16	High band gain table for 23.85 kbit/s mode
Cod_main.c	Interpol_frac	4	LPC interpolation coefficients
Cod_main.c	Isp_init	16	isp tables for initialization
Cod_main.c	Isf_init	16	isf tables for initialization
D_gain2.c	cdown_unusable	7	attenuation factors for codebook gain in lost frames
D_gain2.c	cdown_usable	7	attenuation factors for codebook gain in bad frames
D_gain2.c	pdown_unusable	7	attenuation factors for adaptive codebook gain in lost frames
D_gain2.c	pdown_usable	7	attenuation factors for adaptive codebook gain in bad frames
D_gain2.c	Pred	4	algebraic code book gain MA predictor coefficients
Dec_main.c	HP_gain	16	High band gain table for 23.85 kbit/s mode
Dec_main.c	Interpol_frac	4	LPC interpolation coefficients
Dec_main.c	Isp_init	16	isp tables for initialization
Dec_main.c	Isf_init	16	isf tables for initialization
Decim54.c	fir_down	120	Downsample FIR filter coefficients
Decim54.c	fir_up	120	Upsample FIR filter coefficients
Dtx.c	en_adjust	9	Energy scaling factor for each mode during comfort noise
Grid100.tab	grid	101	grid points at wich Chebyshev polynomials
Ham_wind.tab	Window	384	LP analysis window
Hp400.c	A	3	HP filter coefficients (denominator) in higher band energy estimation
Hp400.c	B	3	HP filter coefficients (numerator) in higher band energy estimation
Hp50.c	A	3	HP filter coefficients (denominator) in pre-filtering
Hp50.c	B	3	HP filter coefficients (numerator) in pre-filtering
Hp6k.c	Fir_6k_7k	31	Bandpass FIR filter coefficients for higher band generation
Hp7k.c	Fir_7k	31	Bandpass FIR filter coefficients for higher band in 23.85 kbit/s mode
Hp_wsp.c	A	3	HP filter coefficients (denominator) in open-loop lag gain computation
Hp_wsp.c	B	3	HP filter coefficients (numerator) in open-loop lag gain computation
Isp_isf.tab	slope	128	table to compute cos(x) in Lsf_lsp()
Isp_isf.tab	Table	129	table to compute acos(x) in Lsp_lsf()
Lag_wind.tab	lag_h	16	high part of the lag window table
Lag_wind.tab	lag_l	16	low part of the lag window table
Lp_dec2.c	h_fir	5	HP FIR filter coefficients in open-loop lag search
Math_op.c	table_isqrt	49	table used in inverse square root computation
Math_op.c	table_pow2	33	table used in power of two computation
P_med_ol.tab	Corrweight	199	weighting of the correlation function in open loop LTP search
Ph_disp.c	ph_imp_low	64	phase dispersion impulse response
Ph_disp.c	ph_imp_mid	64	phase dispersion impulse response
Pitch_fr4.c	inter4_1	32	interpolation filter coefficients
Pred_lt4.c	inter4_2	128	interpolation filter coefficients
Q_gain2.c	pred	4	algebraic code book gain MA predictor coefficients
Q_gain2.tab	t_qua_gain6b	2*64	gain quantization table for 6-bit gain quantization
Q_gain2.tab	t_qua_gain7b	2*128	gain quantization table for 7-bit gain quantization
Qisf_ns.tab	dico1_isf_noise	2*64	1st ISF quantizer for comfort noise
Qisf_ns.tab	dico2_isf_noise	3*64	2nd ISF quantizer for comfort noise
Qisf_ns.tab	Dico3_isf_noise	3*64	3rd LSF quantizer for comfort noise
Qisf_ns.tab	Dico4_isf_noise	4*32	4th LSF quantizer for comfort noise
Qisf_ns.tab	Dico5_isf_noise	4*32	5th LSF quantizer for comfort noise
Qisf_ns.tab	mean_isf_noise	16	ISF mean for comfort noise
Qpisf_2s.tab	dico1_isf	9*256	1st ISF quantizer of the 1st stage
Qpisf_2s.tab	Dico2_isf	7*256	2nd ISF quantizer of the 1st stage
Qpisf_2s.tab	Dico21_isf	3*64	1st ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Qpisf_2s.tab	Dico21_isf_36b	5*128	1st ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Qpisf_2s.tab	Dico22_isf	3*128	2nd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Qpisf_2s.tab	Dico22_isf_36b	4*128	2nd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)

(continued)

Table C.5 (concluded): Fixed tables

File	Table name	Length	Description
Qpisf_2s.tab	Dico23_isf	3*128	3rd ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Qpisf_2s.tab	Dico23_isf_36b	7*64	3rd ISF quantizer of the 2nd stage (the 6.60 kbit/s mode)
Qpisf_2s.tab	Dico24_isf	3*32	4th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Qpisf_2s.tab	Dico25_isf	4*32	5th ISF quantizer of the 2nd stage (not the 6.60 kbit/s mode)
Qpisf_2s.tab	Mean_isf	16	ISF mean

C.1.4.3 Static variables used in the C-code

In this section two tables that specify the static variables for the speech encoder and decoder respectively are shown. All static variables are declared within a C **struct.**

Table C,6: Speech encoder static variables

Struct name	Variable	Type[Length]	Description
Coder_State	mem_decim	Word16[30]	Decimation filter memory
	mem_sig_in	Word16[6]	Prefilter memory
	mem_preemph	Word16	Preemphasis filter memory
	old_speech	Word16[128]	speech buffer
	old_wsp	Word16[115]	buffer holding spectral weighted speech
	old_exc	Word16[248]	excitation vector
	mem_levinson	Word16[18]	Levinson memories
	lspold	Word16[16]	Old ISP vector
	ispold_q	Word16[16]	Old quantized ISP vector
	past_isfq	Word16[16]	past quantized ISF prediction error
	mem_wsp	Word16	Open-loop LTP deemphasis filter memory
	mem_decim2	Word16[3]	Open-loop LTP decimation filter memory
	mem_w0	Word16	weighting filter memory (applied to error signal)
	mem_syn	Word16[16]	synthesis filter memory
	tilt_code	Word16	Preemhasis filter memory
	old_wsp_max	Word16	Open loop scaling factor
	old_wsp_shift	Word16	Maximum open loop scaling factor
	Q_old	Word16	Old scaling factor
	Q_max	Word16[2]	Maximum scaling factor
	gp_clip	Word16[2]	memory of pitch clipping
	qua_gain	Word16[4]	Gain quantization memory
	old_T0_med	Word16	weighted open loop pitch lag
	ol_gain	Word16	Open-loop gain
	ada_w	Word16	weigthing level depeding on open loop pitch gain
	ol_wght_flg	Word16	switches lag weighting on and off
	old_ol_lag	Word16[5]	Open loop lag history
	hp_wsp_mem	Word16[9]	Open-loop lag gain filter memory
	old_hp_wsp	Word16[243]	Open-loop lag
	vadSt	VadVars*	see below in this table
	dtx_encSt	dtx_encState*	see below in this table
	first_frame	Word16	First frame indicator
	lsfold	Word16[16]	Old ISF vector
	L_gc_thres	Word16	Noise enhancer threshold
	mem_syn_hi	Word16[16]	synthesis filter memory (most significant word)
	mem_syn_lo	Word16[16]	synthesis filter memory (least significant word)
	mem_deemph	Word16	Deemphasis filter memory
	mem_sig_out	Word16[6]	HP filter memory in the synthesis
	mem_hp400	Word16[6]	HP filter memory
	mem_oversamp	Word16[2*12]	Oversampling filter memory
	mem_syn_hf	Word16[16]	Higher band synthesis filter memory
	mem_hf	Word16[30]	Estimated BP filter memory (23.85 kbit/s mode)
	mem_hf2	Word16[30]	Input BP filter memory (23.85 kbit/s mode)
	mem_hf3	Word16[30]	Input LP filter memory (23.85 kbit/s mode)
	seed2	Word16	Random generation seed
	disp_mem	Word16[8]	Phase dispersion memory
	vad_hist	Word16	VAD history
	Gain_alpha	Word16	Higher band gain weighting factor (23.85 kbit/s mode)
dtx_encState	lsf_hist	Word16[128]	LSP history (8 frames)
	Log_en_hist	Word16[8]	logarithmic frame energy history (8 frames)
	Hist_ptr	Word16	pointer to the cyclic history vectors
	Log_en_index	Word16	Index for logarithmic energy
	Cng_seed	Word16	Comfort noise excitation seed
	D	Word16[28]	ISF history distance matrix
	sumD	Word16[8]	Sum of ISF history distances
	dtxHangoverCount	Word16	is decreased in DTX hangover period
	decAnaElapsedCount	Word16	counter for elapsed speech frames in DTX
vadState1	bckr_est	Word16[12]	background noise estimate
	ave_level	Word16[12]	averaged input components for stationary estimation
	old_level	Word16[12]	input levels of the previous frame
	sub_level	Word16[12]	input levels calculated at the end of a frame (lookahead)
	a_data5	Word16[5][2]	memory for the filter bank
	a_data3	Word16[6]	memory for the filter bank
	burst_count	Word16	counts length of a speech burst

Struct name	Variable	Type[Length]	Description
	Hang_count	Word16	hangover counter
	Stat_count	Word16	stationary counter
	Vadreg	Word16	15 flags for intermediate VAD decisions
	Tone_flag	Word16	15 flags for tone detection
	sp_est_cnt	Word16	Speech level estimation counter
	Sp_max	Word16	Maximum signal level
	sp_max_cnt	Word16	Maximum level estimation counter
	Speech_level	Word16	Speech level
	prev_pow_sum	Word16	Power of previous frame

Table C.7: Speech decoder static variables

Struct name	Variable	Type[Length]	Description
Decoder_State	old_exc	Word16[248]	excitation vector
	ispold	Word16[16]	Old ISP vector
	isfold	Word16[16]	Old ISF vector
	isf_buf	Word16[48]	ISF vector history
	past_isfq	Word16[16]	past quantized ISF prediction error
	tilt_code	Word16	Preemhasis filter memory
	Q_old	Word16	Old scaling factor
	Qsubfr	Word16	Scaling factor history
	L_gc_thres	Word16	Noise enhancer threshold
	mem_syn_hi	Word16[16]	synthesis filter memory (most significant word)
	mem_syn_lo	Word16[16]	synthesis filter memory (least significant word)
	mem_deemph	Word16	Deemphasis filter memory
	mem_sig_out	Word16[6]	HP filter memory in the synthesis
	mem_oversamp	Word16[24]	Oversampling filter memory
	mem_syn_hf	Word16[20]	Higher band synthesis filter memory
	mem_hf	Word16[30]	Estimated BP filter memory (23.85 kbit/s mode)
	mem_hf2	Word16[30]	Input BP filter memory (23.85 kbit/s mode)
	mem_hf3	Word16[30]	Input LP filter memory (23.85 kbit/s mode)
	seed	Word16	Random code generation seed for bad frames
	seed2	Word16	Random generation seed for higher band
	old_T0	Word16	Old LTP lag (integer part)
	old_T0_frac	Word16	Old LTP lag (fraction part)
	lag_hist	Word16[5]	LTP lag history
	dec_gain	Word16[23]	Gain decoding memory
	seed3	Word16	Random LTP lag generation seed for bad frames
	disp_mem	Word16[8]	Phase dispersion memory
	mem_hp400	Word16[6]	HP filter memory
	prev_bfi	Word16	Previous BFI
	state	Word16	BGH state machine memory
	first_frame	Word16	First frame indicator
	dtx_decSt	dtx_decState*	see below in this table
	Vad_hist	Word16	VAD history
dtx_decState	Since_last_sid	Word16	number of frames since last SID frame
	true_sid_period_inv	Word16	inverse of true SID update rate
	log_en	Word16	logarithmic frame energy
	old_log_en	Word16	previous value of log_en
	isf	Word16[16]	ISF vector
	lsf_old	Word16[16]	Previous ISF vector
	Cng_seed	Word16	Comfort noise excitation seed
	lsf_hist	Word16[128]	ISF vector history (8 frames)
	Log_en_hist	Word16[8]	logarithmic frame energy history
	Hist_ptr	Word16	index to beginning of LSF history
	dtxHangoverCount	Word16	counts down in hangover period
	DecAnaElapsedCount	Word16	counts elapsed speech frames after DTX
	sid_frame	Word16	flags SID frames
	valid_data	Word16	flags SID frames containing valid data
	log_en_adjust	Word16	mode-dependent frame energy adjustment
	dtxHangoverAdded	Word16	flags hangover period at end of speech
	dtxGlobalState	Word16	DTX state flags
	data_updated	Word16	flags CNI updates

C.2 Homing procedure

The principles of the homing procedures are described in the main body of this Recommendation. This section only includes a detailed description of the 9 decoder homing frames. For each AMR-WB codec mode, the corresponding decoder homing frame has a fixed set of parameters. The parameters in serial format are packed into parameters in 15-bit-long format where the first serial bit is inserted into most significant bit in the 15-bit-long format. These 15-bit-long parameters do not represent real speech parameters, but they decrease memory consumption compared to the speech parameters. Table C.8 shows the homing frame in 15-bit-long format for different modes. In the decoder, the received speech parameters in serial format are first converted into 15-bit-long format. Then the obtained parameters are compared against the homing frame table values (Table C.8).

Table C.8: Table values for the decoder homing frame in 15-bit-long format for different modes

Mode	Value (MSB=b0)
0	25351, 4331, 515, 15620, 20992, 0, 0, 0, 0
1	25351, 14010, 26489, 30912, 5254, 3459, 0, 0, 0, 0, 0, 0
2	25351, 14010, 29177, 18070, 19971, 3968, 32492, 8430, 13280, 0, 0, 0, 0, 0, 0, 0, 0
3	25351, 14010, 1912, 16326, 25140, 16384, 502, 15167, 1772, 11512, 0, 0, 0, 0, 0, 0, 0, 0, 0
4	25351, 14010, 1912, 30593, 14594, 19990, 864, 4635, 20446, 27456, 21310, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
5	25351, 14010, 19995, 14446, 6159, 7329, 20752, 4228, 19488, 24383, 364, 20124, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
6	25351, 14010, 3567, 560, 32536, 20534, 5139, 16384, 26161, 18755, 20444, 22173,12623, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
7	25351, 14010, 2912, 28827, 15347, 28610, 9853, 1316, 30720, 786, 32259, 13279,14336, 29152, 23302, 20352, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0, 0, 0
8	025351, 14010, 1601, 16734, 7923, 15017, 5450, 5477, 5760, 2187, 1534, 12142, 30894, 13419, 13141, 2376, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

C.3 File formats

This section describes the file formats used by the encoder and decoder programs. The test sequences also use the file formats described here.

C.3.1 Speech file (encoder input / decoder output)

Speech files read by the encoder and written by the decoder consist of 16-bit words where each word contains a 14-bit, left aligned speech sample. The byte order depends on the host architecture (e.g. MSByte first on SUN workstations, LSByte first on PCs etc.). Both the encoder and the decoder program process complete frames (of 320 samples) only.

This means that the encoder will only process *n* frames if the length of the input file is *n*320 + k* words, while the files produced by the decoder will always have a length of *n*320* words.

C.3.2 Mode control file (encoder input)

The encoder program can optionally read in a mode control file which specifies the encoding mode for each frame of speech processed. The file is a text file containing one number per speech frame. Each line contains one of the mode numbers 0-8.

C.3.3 Parameter bitstream file (encoder output / decoder input)

The files produced by the speech encoder/expected by the speech decoder contain an arbitrary number of frames in the following format.

TYPE_OF_FRAME_TYPE	FRAME_TYPE	MODE	B1	B2	…	Bnn

Each box corresponds to one `Word16` value in the bitstream file, for a total of 3+nn words or 6+2nn bytes per frame, where nn is the number of encoded bits in the frame. The fields have the following meaning:

TYPE_OF_FRAME_TYPE transmit frame type, which is one of
TX_TYPE (0x6b21)
RX_TYPE (0x6b20)

If TYPE_OF_FRAME_TYPE is TX_TYPE,

FRAME_TYPE transmit frame type, which is one of
TX_SPEECH (0x0000)
TX_SID_FIRST (0x0001)
TX_SID_UPDATE (0x0002)
TX_NO_DATA (0x0003)

If TYPE_OF_FRAME_TYPE is RX_TYPE,

FRAME_TYPE transmit frame type, which is one of
RX_SPEECH_GOOD (0x0000)
RX_SPEECH_PROBABLY_DEGRADED (0x0001)
RX_SPEECH_LOST (0x0002)
RX_SPEECH_BAD (0x0003)
RX_SID_FIRST (0x0004)
RX_SID_UPDATE (0x0005)
RX_SID_BAD (0x0006)
RX_NO_DATA (0x0007)

B0…B2nn speech encoder parameter bits (i.e. the bitstream itself). Each *Bx* either has the value 0x0081 (for bit 0) or 0x007F (for bit 1).

MODE_INFO encoding mode information, which is one of
6.60 kbit/s mode (0x0000)
8.85 kbit/s mode (0x0001)
12.65 kbit/s mode (0x0002)
14.25 kbit/s mode (0x0003)
15.85 kbit/s mode (0x0004)
18.25 kbit/s mode (0x0005)
19.85 kbit/s mode (0x0006)
23.05 kbit/s mode (0x0007)
23.85 kbit/s mode (0x0008)

As indicated in section C.3.1 above, the byte order depends on the host architecture.