International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.729

(01/2007)

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

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

Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)

ITU-T Recommendation G.729

ITU-T G-SERIES RECOMMENDATIONS

TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

INTERNATIONAL TELEPHONE CONNECTIONS AND CIRCUITS	G.100–G.199
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
TRANSMISSION MEDIA AND OPTICAL SYSTEMS CHARACTERISTICS	G.600–G.699
DIGITAL 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
QUALITY OF SERVICE AND PERFORMANCE – GENERIC AND USER-RELATED ASPECTS	G.1000–G.1999
TRANSMISSION MEDIA CHARACTERISTICS	G.6000–G.6999
DATA OVER TRANSPORT – GENERIC ASPECTS	G.7000–G.7999
PACKET OVER TRANSPORT ASPECTS	G.8000–G.8999
ACCESS NETWORKS	G.9000–G.9999

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

ITU-T Recommendation G.729

Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)

Summary

This Recommendation contains the description of an algorithm for the coding of speech signals using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP).

In its basic mode, the G.729 coder consists of a mono-rate speech coder at 8 kbits/s using fixed-point arithmetic operations. Annexes A, B and D to J extend its functionalities. Annex A provides a reduced-complexity version at the basic coding rate of 8 kbit/s. Annex B defines source-controlled rate operation for use with G.729 or Annex A. Annexes D, E and H provide multi-rate operation and specify rate-switching mechanisms: Annex D provides lower bit-rate extension at 6.4 kbit/s and Annex E provides higher bit-rate extension at 11.8 kbit/s, whereas Annex H provides bit-rate extensions at both 6.4 kbit/s and 11.8 kbit/s. Therefore, Annexes D, E and H do not implement the discontinuous transmission mode of Annex B. For this functionality, further annexes were developed. Annexes F and G use the basic algorithms in Annex B to provide discontinuous transmission (DTX) functionality for, respectively, Annexes D and E. Annex I provides DTX functionality for Annex H and describes the integration of G.729 main body with Annexes B, D and E. Annex J makes reference to the G.729 extension for the 8-32 kbit/s scalable wideband speech and audio coding algorithm in ITU-T Recommendation G.729.1, which is interoperable with G.729 and its Annexes A and B. As G.729 main body, its Annexes A, B and D to J use fixed-point arithmetic. Alternative implementations based on floating-point arithmetic operations are provided in Annex C for G.729 and Annex A, and in Annex C+ for Annex I.

This information is summarized in the Table below.

	Annexes											
Functionality	**-**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**C+**	**J**
Low complexity		X	X									
Fixed-point	X	X	X		X	X	X	X	X	X		X
Floating-point				X							X	
8 kbit/s	X	X	X	X	X	X	X	X	X	X	X	X
6.4 kbit/s					X		X		X	X	X	
11.8 kbit/s						X		X	X	X	X	
DTX			X				X	X		X	X	
Embedded variable bit rate, wideband												X

Appendix I deals with external synchronous reset capability in systems using external silence compression in conjunction with the speech coding algorithm in the main body of G.729 (fixed-point) or in its Annexes A (low complexity, fixed-point) and C (floating-point). Since the voice activity detection (VAD) algorithm in Annex B was optimized for transmission over connection-oriented circuits, Appendices II and III deal with optimization of the VAD in Annex B when it is used for packet circuits such as VoIP applications.

Reference ANSI C source code and test vectors are provided as an integral part of this Recommendation and its annexes. Appendices II and III are also associated with C source code and test vectors. No source code is associated with Appendix I. The C source code and test vectors are available as electronic attachments to this Recommendation.

Source

ITU-T Recommendation G.729 was approved on 13 January 2007 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure.

CONTENTS

Page

ITU-T Recommendation G.729

Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)

1 Scope

This Recommendation contains the description of an algorithm for the coding of speech signals at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP). This Recommendation includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 8 kbit/s.

This coder is designed to operate with a digital signal obtained by first performing telephone bandwidth filtering [ITU-T G.712] of the analogue input signal, then sampling it at 8000 Hz, followed by conversion to 16-bit linear PCM for the input to the encoder. The output of the decoder should be converted back to an analogue signal by similar means. Other input/output characteristics, such as those specified by [ITU-T G.711] for 64 kbit/s PCM data, should be converted to 16-bit linear PCM before encoding, or from 16-bit linear PCM to the appropriate format after decoding. The bit stream from the encoder to the decoder is defined within this Recommendation.

This Recommendation is organized as follows: Clause 2 gives a general outline of the CS-ACELP algorithm. In clauses 3 and 4, the CS-ACELP encoder and decoder principles are discussed, respectively. Clause 5 describes the software that defines this coder in 16 bit fixed-point arithmetic.

2 General description of the coder

The CS-ACELP coder is based on the code-excited linear prediction (CELP) coding model. The coder operates on speech frames of 10 ms corresponding to 80 samples at a sampling rate of 8000 samples per second. For every 10 ms frame, the speech signal is analysed to extract the parameters of the CELP model (linear prediction filter coefficients, adaptive and fixed-codebook indices and gains). These parameters are encoded and transmitted. The bit allocation of the coder parameters is shown in Table 1. At the decoder, these parameters are used to retrieve the excitation and synthesis filter parameters. The speech is reconstructed by filtering this excitation through the short-term synthesis filter, as is shown in Figure 1. The short-term synthesis filter is based on a 10th order linear prediction (LP) filter. The long-term, or pitch synthesis filter is implemented using the so-called adaptive-codebook approach. After computing the reconstructed speech, it is further enhanced by a postfilter.

Table 1 – Bit allocation of the 8 kbit/s CS-ACELP algorithm (10 ms frame)

Parameter	Codeword	Subframe 1	Subframe 2	Total per frame
Line spectrum pairs	$L0, L1, L2, L3$			18
Adaptive-codebook delay	$P1, P2$	8	5	13
Pitch-delay parity	$P0$	1		1
Fixed-codebook index	$C1, C2$	13	13	26
Fixed-codebook sign	$S1, S2$	4	4	8
Codebook gains (stage 1)	$GA1, GA2$	3	3	6
Codebook gains (stage 2)	$GB1, GB2$	4	4	8
Total				80

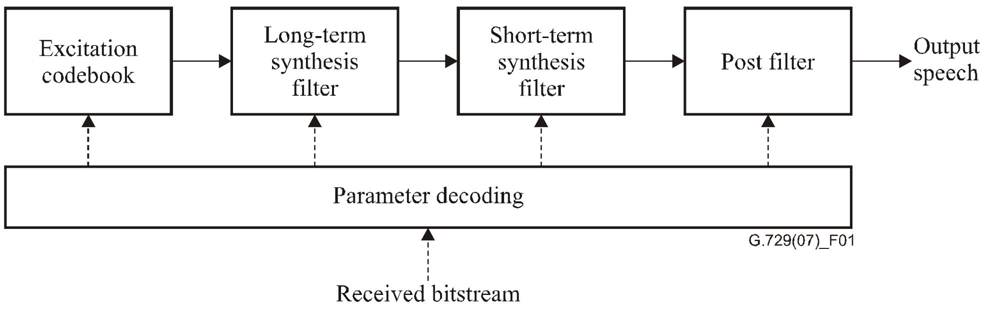


Figure 1 – Block diagram of conceptual CELP synthesis model

2.1 Encoder

The encoding principle is shown in Figure 2. The input signal is high-pass filtered and scaled in the preprocessing block. The preprocessed signal serves as the input signal for all subsequent analysis. LP analysis is done once per 10 ms frame to compute the LP filter coefficients. These coefficients are converted to line spectrum pairs (LSPs) and quantized using predictive two-stage vector quantization (VQ) with 18 bits. The excitation signal is chosen by using an analysis-by-synthesis search procedure in which the error between the original and reconstructed speech is minimized according to a perceptually weighted distortion measure. This is done by filtering the error signal with a perceptual weighting filter, whose coefficients are derived from the unquantized LP filter. The amount of perceptual weighting is made adaptive to improve the performance for input signals with a flat frequency-response.

The excitation parameters (fixed and adaptive-codebook parameters) are determined per subframe of 5 ms (40 samples) each. The quantized and unquantized LP filter coefficients are used for the second subframe, while in the first subframe interpolated LP filter coefficients are used (both quantized and unquantized). An open-loop pitch delay is estimated once per 10 ms 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 $W(z)/\hat{A}(z)$. The initial states of these filters are 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 done (to find the adaptive-codebook delay and gain), using the target $x(n)$ and impulse response $h(n)$, by searching around the value of the open-loop pitch delay. A fractional pitch delay with 1/3 resolution is used. The pitch delay is encoded with 8 bits in the first subframe and differentially encoded with 5 bits in the second subframe. The target signal $x(n)$ is updated by subtracting the (filtered) adaptive-codebook contribution, and this new target, $x'(n)$, is used in the fixed-codebook search to find the optimum excitation. An algebraic codebook with 17 bits is used for the fixed-codebook excitation. The gains of the adaptive and fixed-codebook contributions are vector quantized with 7 bits (with MA prediction applied to the fixed-codebook gain). Finally, the filter memories are updated using the determined excitation signal.

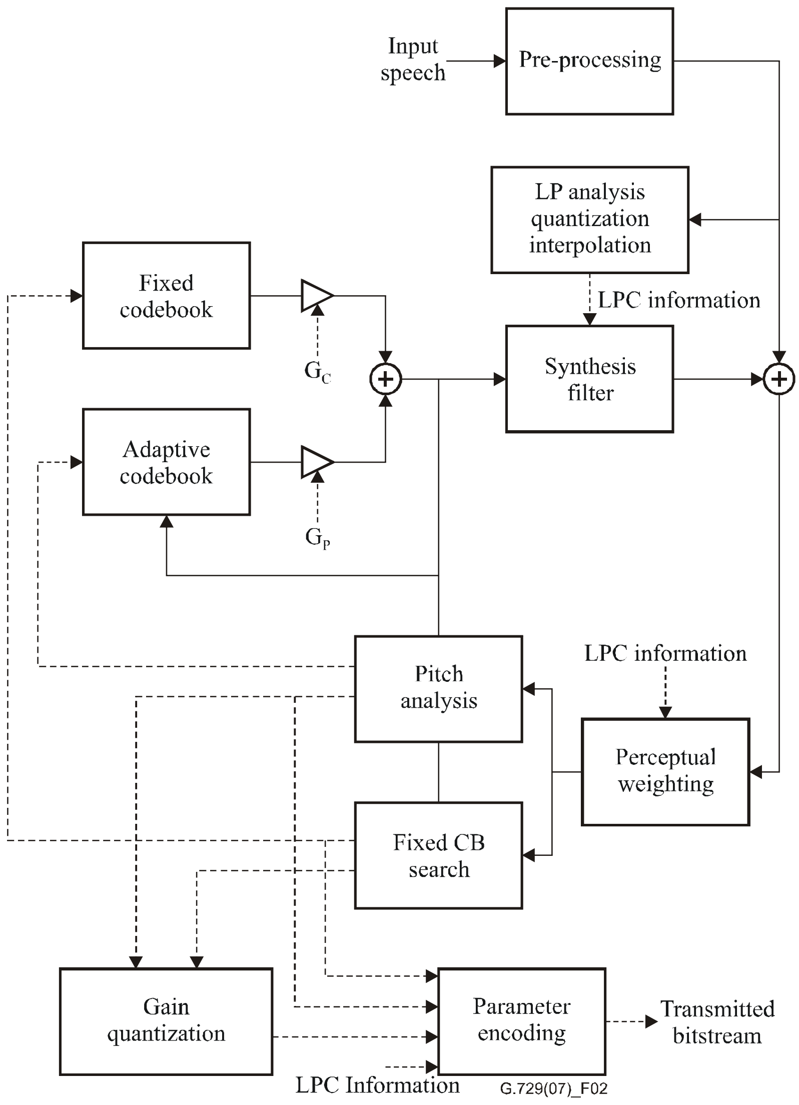


Figure 2 – Principle of the CS-ACELP encoder

2.2 Decoder

The decoder principle is shown in Figure 3. First, the parameter's indices are extracted from the received bit stream. These indices are decoded to obtain the coder parameters corresponding to a 10 ms speech frame. These parameters are the LSP coefficients, the two fractional pitch delays, the two fixed-codebook vectors, and the two sets of adaptive and fixed-codebook gains. The LSP coefficients are interpolated and converted to LP filter coefficients for each subframe. Then, for each 5 ms subframe the following steps are done:

– the excitation is constructed by adding the adaptive and fixed-codebook vectors scaled by their respective gains;

– the speech is reconstructed by filtering the excitation through the LP synthesis filter; and

– the reconstructed speech signal is passed through a post-processing stage, which includes an adaptive postfilter based on the long-term and short-term synthesis filters, followed by a high-pass filter and scaling operation.

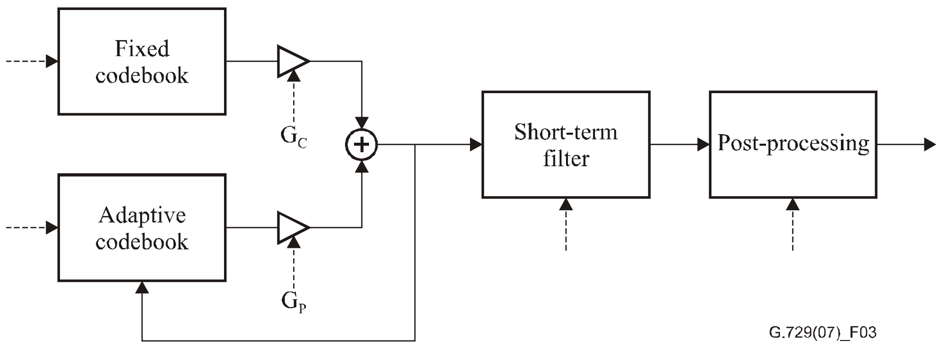


Figure 3 – Principle of the CS-ACELP decoder

2.3 Delay

This coder encodes speech and other audio signals with 10 ms frames. In addition, there is a look-ahead of 5 ms, resulting in a total algorithmic delay of 15 ms. All additional delays in a practical implementation of this coder are due to:

– processing time needed for encoding and decoding operations;

– transmission time on the communication link; and

– multiplexing delay when combining audio data with other data.

2.4 Speech coder description

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

2.5 Notational conventions

Throughout this Recommendation, it is tried to maintain the following notational conventions:

– Codebooks are denoted by calligraphic characters (e.g., $\mathcal{C}$).

– Time signals are denoted by their symbol and a sample index between parenthesis [e.g., $s(n)$]. The symbol n is used as sample index.

– Superscript indices between parenthesis (e.g., $g^{(m)}$) are used to indicate time-dependency of variables. The variable m refers, depending on the context, to either a frame or subframe index, and the variable n to a sample index.

– Recursion indices are identified by a superscript between square brackets (e.g., $E^{[k]}$).

– Subscript indices identify a particular element in a coefficient array.

– The symbol ^ identifies a quantized version of a parameter (e.g., $\hat{g}_c$).

– Parameter ranges are given between square brackets, and include the boundaries (e.g., [0.6, 0.9]).

– The function *log* denotes a logarithm with base 10.

– The function *int* denotes truncation to its integer value.

– The decimal floating-point numbers used are rounded versions of the values used in the 16 bit fixed-point ANSI C implementation.

Table 2 lists the most relevant symbols used throughout this Recommendation. A glossary of the most relevant signals is given in Table 3. Table 4 summarizes relevant variables and their dimensions. Constant parameters are listed in Table 5. The acronyms used in this Recommendation are summarized in Table 6.

Table 2 – Glossary of most relevant symbols

Name	Reference	Description
$1/\hat{A}(z)$	Equation (2)	LP synthesis filter
$H_{h1}(z)$	Equation (1)	Input high-pass filter
$H_p(z)$	Equation (78)	Long-term postfilter
$H_f(z)$	Equation (84)	Short-term postfilter
$H_t(z)$	Equation (86)	Tilt-compensation filter
$H_{h2}(z)$	Equation (91)	Output high-pass filter
$P(z)$	Equation (46)	Pre-filter for fixed codebook
$W(z)$	Equation (27)	Weighting filter

Table 3 – Glossary of most relevant signals

Name	Reference	Description
$c(n)$	3.8	Fixed-codebook contribution
$d(n)$	3.8.1	Correlation between target signal and $h(n)$
$ew(n)$	3.10	Error signal
$h(n)$	3.5	Impulse response of weighting and synthesis filters
$r(n)$	3.6	Residual signal
$s(n)$	3.1	Preprocessed speech signal
$\hat{s}(n)$	4.1.6	Reconstructed speech signal
$s'(n)$	3.2.1	Windowed speech signal
$sf(n)$	4.2	Postfiltered output
$sf'(n)$	4.2	Gain-scaled postfiltered output
$sw(n)$	3.6	Weighted speech signal
$x(n)$	3.6	Target signal
$x'(n)$	3.8.1	Second target signal
$u(n)$	3.10	Excitation to LP synthesis filter
$v(n)$	3.7.1	Adaptive-codebook contribution
$y(n)$	3.7.3	Convolution $v(n) * h(n)$
$z(n)$	3.9	Convolution $c(n) * h(n)$

Table 4 – Glossary of most relevant variables

Name	Size	Description
g_p	1	Adaptive-codebook gain
g_c	1	Fixed-codebook gain
g_l	1	Gain term for long-term postfilter
g_f	1	Gain term for short-term postfilter
g_t	1	Gain term for tilt postfilter
G	1	Gain for gain normalization
T_{op}	1	Open-loop pitch delay
a_i	11	LP coefficients ($a_0 = 1.0$)
k_i	10	Reflection coefficients
k'_1	1	Reflection coefficient for tilt postfilter
O_i	2	LAR coefficients
ω_i	10	LSF normalized frequencies
$\hat{p}_{i,j}$	40	MA predictor for LSF quantization
q_i	10	LSP coefficients
$r(k)$	11	Auto-correlation coefficients
$r'(k)$	11	Modified auto-correlation coefficients
w_i	10	LSP weighting coefficients
$\hat{l}_i$	10	LSP quantizer output

Table 5 – Glossary of most relevant constants

Name	Value	Description
f_s	8000	Sampling frequency
f_0	60	Bandwidth expansion
γ_1	0.94/0.98	Weight factor perceptual weighting filter
γ_2	0.60/[0.4 – 0.7]	Weight factor perceptual weighting filter
γ_n	0.55	Weight factor postfilter
γ_d	0.70	Weight factor postfilter
γ_p	0.50	Weight factor pitch postfilter
γ_t	0.90/0.2	Weight factor tilt postfilter
$\mathcal{C}$	Table 7	Fixed (algebraic) codebook
$\mathcal{L}0$	3.2.4	Moving-average predictor codebook
$\mathcal{L}1$	3.2.4	First stage LSP codebook
$\mathcal{L}2$	3.2.4	Second stage LSP codebook (low part)
$\mathcal{L}3$	3.2.4	Second stage LSP codebook (high part)
$\mathcal{GA}$	3.9	Gain codebook (first stage)
$\mathcal{GB}$	3.9	Gain codebook (second stage)
w_{lag}	Equation (6)	Correlation lag window
w_{lp}	Equation (3)	LP analysis window

Table 6 – Glossary of acronyms

Acronym	Description
CELP	Code-Excited Linear Prediction
CS-ACELP	Conjugate-Structure Algebraic CELP
MA	Moving Average
MSB	Most Significant Bit
MSE	Mean-Squared Error
LAR	Log Area Ratio
LP	Linear Prediction
LSP	Line Spectral Pair
LSF	Line Spectral Frequency
VQ	Vector quantization

3 Functional description of the encoder

This clause describes the different functions of the encoder represented in the blocks of Figure 2. A detailed signal flow is shown in Figure 4.

3.1 Preprocessing

As stated in clause 2, the input to the speech encoder is assumed to be a 16-bit PCM signal. Two preprocessing functions are applied before the encoding process:

1) signal scaling; and
2) high-pass filtering.

The scaling consists of dividing the input by a factor 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 second order pole/zero filter with a cut-off frequency of 140 Hz is used. Both the scaling and high-pass filtering are combined by dividing the coefficients at the numerator of this filter by 2. The resulting filter is given by:

$$H_{h1}(z) = \frac{0.46363718 - 0.92724705z^{-1} + 0.46363718z^{-2}}{1 - 1.9059465z^{-1} + 0.9114024z^{-2}} \quad (1)$$

The input signal filtered through $H_{h1}(z)$ is referred to as $s(n)$, and will be used in all subsequent coder operations.

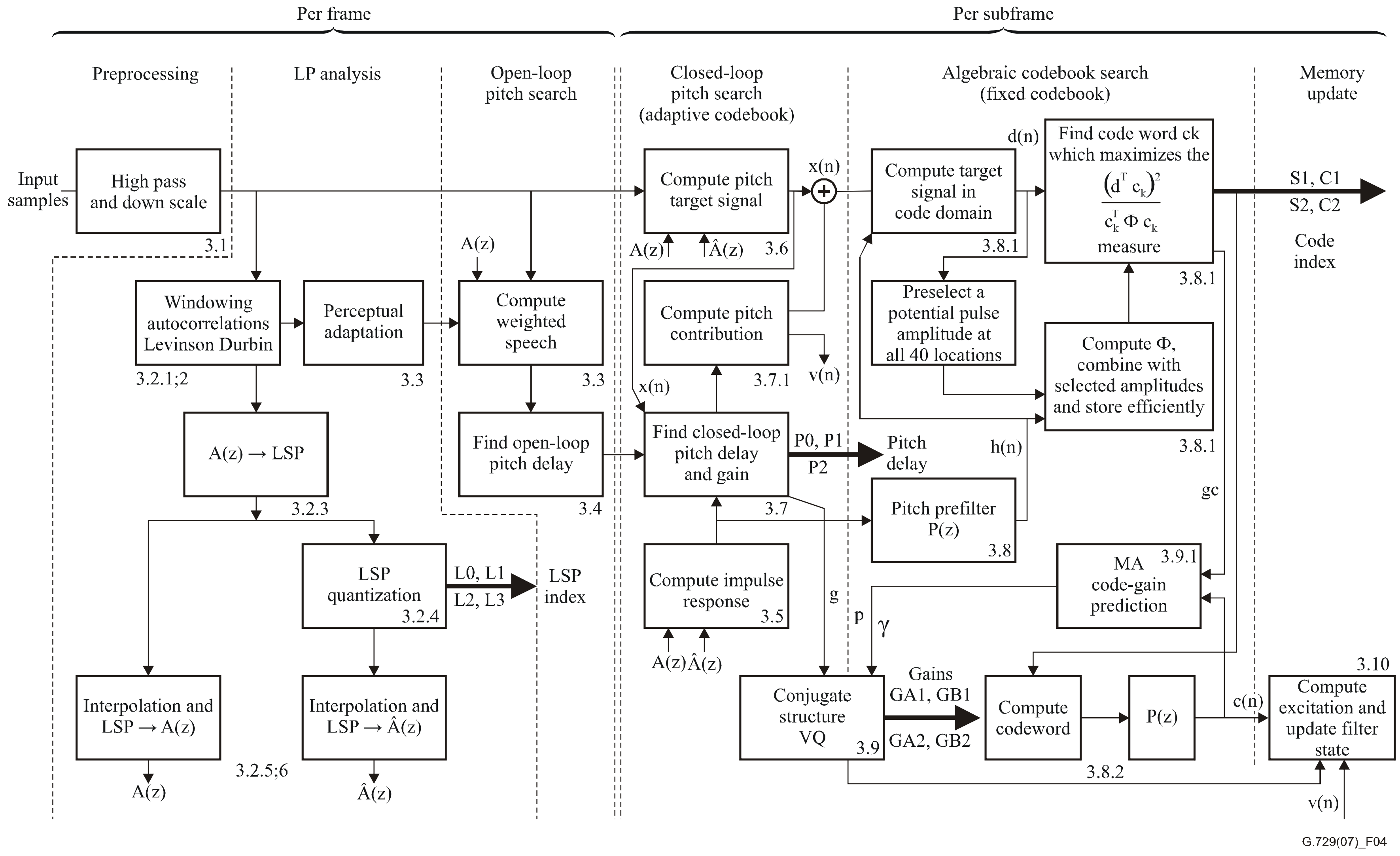
3.2 Linear prediction analysis and quantization

The short-term analysis and synthesis filters are based on 10th order linear prediction (LP) filters.

The LP synthesis filter is defined as:

$$\frac{1}{\hat{A}(z)} = \frac{1}{1 + \sum_{i=1}^{10} \hat{a}_i z^{-i}} \quad (2)$$

where $\hat{a}_i$, $i = 1,...,10$, are the (quantized) linear prediction (LP) coefficients. Short-term prediction, or linear prediction analysis is performed once per speech frame using the autocorrelation method with a 30 ms asymmetric window. Every 80 samples (10 ms), the autocorrelation coefficients of windowed speech are computed and converted to the LP coefficients using the Levinson-Durbin algorithm. Then the LP coefficients are transformed to the LSP 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 for each subframe).



Per frame
Per subframe
Preprocessing
LP analysis
Open-loop pitch search
Closed-loop pitch search (adaptive codebook)
Algebraic codebook search (fixed codebook)
Memory update
Input samples
High pass and down scale
3.1
Windowing autocorrelations Levinson Durbin
3.2.1;2
Perceptual adaptation
3.3
A(z) → LSP
3.2.3
LSP quantization
3.2.4
L0, L1
L2, L3
LSP index
Interpolation and LSP → A(z)
A(z)
3.2.5;6
Interpolation and LSP → Â(z)
Â(z)
A(z)
Compute weighted speech
3.3
Find open-loop pitch delay
3.4
Compute pitch target signal
A(z)
Â(z)
3.6
x(n)
Compute pitch contribution
v(n)
3.7.1
x(n)
Find closed-loop pitch delay and gain
3.7
P0, P1
P2
Pitch delay
Compute impulse response
3.5
A(z) Â(z)
g
Conjugate structure VQ
3.9
Gains
GA1, GB1
GA2, GB2
d(n)
Compute target signal in code domain
3.8.1
Preselect a potential pulse amplitude at all 40 locations
Find code word ck which maximizes the
$\frac{(d^T c_k)^2}{c_k^T \Phi c_k}$
measure
3.8.1
Compute Φ, combine with selected amplitudes and store efficiently
3.8.1
h(n)
Pitch prefilter P(z)
3.8
gc
MA code-gain prediction
3.9.1
p
γ
Compute codeword
P(z)
3.8.2
c(n)
S1, C1
S2, C2
Code index
3.10
Compute excitation and update filter state
v(n)
G.729(07)_F04

Figure 4 – Signal flow at the CS-ACELP encoder

3.2.1 Windowing and autocorrelation computation

The LP analysis window consists of two parts: the first part is half a Hamming window and the second part is a quarter of a cosine function cycle. The window is given by:

$$w_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi n}{399}\right) & n = 0,\ldots,199 \\ \cos\left(\dfrac{2\pi(n-200)}{159}\right) & n = 200,\ \ldots\ 239 \end{cases} \tag{3}$$

There is a 5 ms look-ahead in the LP analysis which means that 40 samples are needed from the future speech frame. This translates into an extra algorithmic delay of 5 ms at the encoder stage. The LP analysis window applies to 120 samples from past speech frames, 80 samples from the present speech frame, and 40 samples from the future frame. The windowing procedure is illustrated in Figure 5.

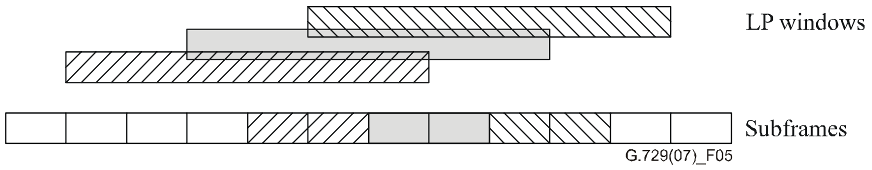


Figure 5 – Windowing procedure in LP analysis

The different shading patterns identify corresponding excitation and LP analysis windows.

The windowed speech:

$$s'(n) = w_{lp}(n)\, s(n) \quad n = 0,\ldots,239 \tag{4}$$

is used to compute the autocorrelation coefficients:

$$r(k) = \sum_{n=k}^{239} s'(n)\, s'(n-k) \quad k = 0,\ldots,10 \tag{5}$$

To avoid arithmetic problems for low-level input signals the value of $r(0)$ has a lower boundary of $r(0) = 1.0$. A 60 Hz bandwidth expansion is applied by multiplying the autocorrelation coefficients with:

$$w_{lag}(k) = exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 k}{f_s}\right)^2\right] \quad k = 1,\ldots,10 \tag{6}$$

where $f_0 = 60$ Hz is the bandwidth expansion and $f_s = 8000$ Hz is the sampling frequency. Furthermore, $r(0)$ is multiplied by a white-noise correction factor 1.0001, which is equivalent to adding a noise floor at –40 dB. The modified autocorrelation coefficients are given by:

$$\begin{aligned} r'(0) &= 1.0001\, r(0) \\ r'(k) &= w_{lag}(k)\, r(k) \quad k = 1,\ldots,10 \end{aligned} \tag{7}$$

3.2.2 Levinson-Durbin algorithm

The modified autocorrelation coefficients $r'(k)$ are used to obtain the LP filter coefficients, a_i, $i = 1,...,10$, by solving the set of equations:

$$\sum_{i=1}^{10} a_i r'(|i-k|) = -r'(k) \quad k = 1,...,10 \tag{8}$$

The set of equations in (8) is solved using the Levinson-Durbin algorithm. This algorithm uses the following recursion:

```
$E^{[0]} = r'(0)$
for $i = 1$ to $10$
    $a_0^{[i-1]} = 1$
    $k_i = -\left[\sum_{j=0}^{i-1} a_j^{[i-1]} r'(i-j)\right] / E^{[i-1]}$
    $a_i^{[i]} = k_i$
    for $j = 1$ to $i-1$
        $a_j^{[i]} = a_j^{[i-1]} + k_i a_{i-j}^{[i-1]}$
    end
    $E^{[i]} = (1-k_i^2)E^{[i-1]}$
end
```

The final solution is given as $a_j = a_j^{[10]}$, $j = 0,...,10$, with $a_0 = 1.0$.

3.2.3 LP to LSP conversion

The LP filter coefficients a_i, $i = 0,...,10$ are converted to line spectral pair (LSP) coefficients for quantization and interpolation purposes. For a 10th order LP filter, the LSP coefficients are defined as the roots of the sum and difference polynomials:

$$F_1'(z) = A(z) + z^{-11}A(z^{-1}) \tag{9}$$

and:

$$F_2'(z) = A(z) - z^{-11}A(z^{-1}) \tag{10}$$

respectively. The polynomial $F_1'(z)$ is symmetric, and $F_2'(z)$ is antisymmetric. It can be proven that all roots of these polynomials are on the unit circle and they alternate each other. $F_1'(z)$ has a root $z = -1$ ($\omega = \pi$) and $F_2'(z)$ has a root $z = 1$ ($w = 0$). These two roots are eliminated by defining the new polynomials:

$$F_1(z) = F_1'(z)/(1+z^{-1}) \tag{11}$$

and:

$$F_2(z) = F_2'(z)/(1-z^{-1}) \tag{12}$$

Each polynomial has five conjugate roots on the unit circle ($e^{\pm j\omega_i}$), and they can be written as:

$$F_1(z) = \prod_{i=1,3,...,9}(1-2q_i z^{-1} + z^{-2}) \tag{13}$$

and:

$$F_2(z) = \prod_{i=2,4,...,10} \left(1 - 2q_i z^{-1} + z^{-2}\right) \tag{14}$$

where $q_i = \cos(\omega_i)$. The coefficients ω_i are the line spectral frequencies (LSF) and they satisfy the ordering property $0 < \omega_i < \omega_2 < ... < \omega_{10} < \pi$. The coefficients q_i are referred to as the LSP coefficients in the cosine domain.

Since both polynomials $F_1(z)$ and $F_2(z)$ are symmetric only the first five coefficients of each polynomial need to be computed. The coefficients of these polynomials are found by the recursive relations:

$$\begin{aligned} f_1(i+1) &= a_{i+1} + a_{10-i} - f_1(i) \quad i = 0,...,4 \\ f_2(i+1) &= a_{i+1} - a_{10-i} + f_2(i) \quad i = 0,...,4 \end{aligned} \tag{15}$$

where $f_1(0) = f_2(0) = 1.0$. The LSP coefficients are found by evaluating the polynomials $F_1(z)$ and $F_2(z)$ at 60 points equally spaced between 0 and π and checking for sign changes. A sign change signifies the existence of a root and the sign change interval is then divided four times to allow better tracking of the root. The Chebyshev polynomials are used to evaluate $F_1(z)$ and $F_2(z)$. In this method the roots are found directly in the cosine domain. The polynomials $F_1(z)$ or $F_2(z)$, evaluated at $z = e^{j\omega}$, can be written as:

$$F(\omega) = 2e^{-j5\omega} C(x) \tag{16}$$

with:

$$C(x) = T_5(x) + f(1)T_4(x) + f(2)T_3(x) + f(3)T_2(x) + f(4)T_1(x) + f(5)/2 \tag{17}$$

where $T_m(x) = \cos(m\omega)$ is the mth order Chebyshev polynomial, and $f(i)$, $i = 1,...,5$, are the coefficients of either $F_1(z)$ or $F_2(z)$, computed using equation (15). The polynomial $C(x)$ is evaluated at a certain value of $x = \cos(\omega)$ using the recursive relation:

```
for $k = 4$ down to 1
    $b_k = 2xb_{k+1} - b_{k+2} + f(5-k)$
end
$C(x) = xb_1 - b_2 + f(5)/2$
```

with initial values $b_5 = 1$ and $b_6 = 0$.

3.2.4 Quantization of the LSP coefficients

The LSP coefficients q_i are quantized using the LSF representation ω_i in the normalized frequency domain $[0, \pi]$; that is:

$$\omega_i = \arccos(q_i) \quad i = 1,...,10 \tag{18}$$

A switched 4th order MA prediction is used to predict the LSF coefficients of the current frame. The difference between the computed and predicted coefficients is quantized using a two-stage vector quantizer. The first stage is a 10-dimensional VQ using codebook $\mathcal{L}1$ with 128 entries (7 bits). The second stage is a 10-bit VQ which has been implemented as a split VQ using two 5-dimensional codebooks, $\mathcal{L}2$ and $\mathcal{L}3$ containing 32 entries (5 bits) each.

To explain the quantization process, it is convenient to first describe the decoding process. Each coefficient is obtained from the sum of two codebooks:

$$\hat{l}_i = \begin{cases} \mathcal{L}1_i(L1) + \mathcal{L}2_i(L2) & i = 1,...,5 \\ \mathcal{L}1_i(L1) + \mathcal{L}3_{i-5}(L3) & i = 6,...,10 \end{cases} \tag{19}$$

where $L1$, $L2$ and $L3$ are the codebook indices. To avoid sharp resonances in the quantized LP synthesis filter, the coefficients $\hat{l}_i$ are arranged such that adjacent coefficients have a minimum distance of J. The rearrangement routine is shown below:

```
for $i = 2,...,10$
    if $\left(\hat{l}_{i-1} > \hat{l}_i - J\right)$
        $\hat{l}_{i-1} = \left(\hat{l}_i + \hat{l}_{i-1} - J\right)/2$
        $\hat{l}_i = \left(\hat{l}_i + \hat{l}_{i-1} + J\right)/2$
    end
end
```

This rearrangement process is done twice. First with a value of $J = 0.0012$, then with a value of $J = 0.0006$. After this rearrangement process, the quantized LSF coefficients $\hat{\omega}_i^{(m)}$ for the current frame m, are obtained from the weighted sum of previous quantizer outputs $\hat{l}_i^{(m-k)}$, and the current quantizer output $\hat{l}_i^{(m)}$.

$$\hat{\omega}_i^{(m)} = \left(1 - \sum_{k=1}^{4} \hat{P}_{i,k}\right)\hat{l}_i^m + \sum_{k=1}^{4} \hat{P}_{i,k}\hat{l}_i^{(m-k)} \quad i = 1,...,10 \tag{20}$$

where $\hat{p}_{i,k}$ are the coefficients of the switched MA predictor. Which MA predictor to use is defined by a separate bit $L0$. At start up the initial values of $\hat{l}_i^{(k)}$ are given by $\hat{l}_i = i\pi/11$ for all $k < 0$.

After computing $\hat{\omega}_i$, the corresponding filter is checked for stability. This is done as follows:

1) order the coefficient $\hat{\omega}_i$ in increasing value;

2) if $\hat{\omega}_1 < 0.005$ then $\hat{\omega}_1 = 0.005$;

3) if $\hat{\omega}_{i+1} - \hat{\omega}_i < 0.0391$ then $\hat{\omega}_{i+1} = \hat{\omega}_i + 0.0391$, i = 1,…,9;

4) if $\hat{\omega}_{10} > 3.135$ then $\hat{\omega}_{10} = 3.135$.

The procedure for encoding the LSF parameters can be outlined as follows. For each of the two MA predictors the best approximation to the current LSF coefficients has to be found. The best approximation is defined as the one that minimizes the weighted mean-squared error:

$$E_{lsf} = \sum_{i=1}^{10} w_i(\omega_i - \hat{\omega}_i)^2 \tag{21}$$

The weights w_i are made adaptive as a function of the unquantized LSF coefficients,

$$w_1 = \begin{cases} 1.0 & \textit{if}\ \omega_2 - 0.04\pi - 1 > 0 \\ 10(\omega_2 - 0.04\pi - 1)^2 + 1 & \text{otherwise} \end{cases}$$

$$w_i, \text{ for } 2 \le i \le 9 = \begin{cases} 1.0 & \textit{if}\ \omega_{i+1} - \omega_{i-1} - 1 > 0 \\ 10(\omega_{i+1} - \omega_{i-1} - 1)^2 + 1 & \text{otherwise} \end{cases} \quad (22)$$

$$w_{10} = \begin{cases} 1.0 & \textit{if}\ -\omega_9 + 0.92\pi - 1 > 0 \\ 10(-\omega_9 + 0.92\pi - 1)^2 + 1 & \text{otherwise} \end{cases}$$

In addition, the weights w_5 and w_6 are each multiplied by 1.2.

The vector to be quantized for the current frame m is obtained from

$$l_i = \left[\omega_i^{(m)} - \sum_{k=1}^{4} \hat{P}_{i,k}\ \hat{l}_i^{(m-k)} \right] \Big/ \left(1 - \sum_{k=1}^{4} \hat{P}_{i,k} \right) \quad i = 1,...,10 \quad (23)$$

The first codebook $\mathcal{L}1$ is searched and the entry $L1$ that minimizes the (unweighted) mean-squared error is selected. This is followed by a search of the second codebook $\mathcal{L}2$, which defines the lower part of the second stage. For each possible candidate, the partial vector $\hat{\omega}_i$, $i = 1,...,5$ is reconstructed using equation (20), and rearranged to guarantee a minimum distance of 0.0012. The weighted MSE of equation (21) is computed, and the vector $L2$ which results in the lowest error is selected. Using the selected first stage vector $L1$ and the lower part of the second stage $L2$, the higher part of the second stage is searched from codebook $\mathcal{L}3$. Again the rearrangement procedure is used to guarantee a minimum distance of 0.0012. The vector $L3$ that minimizes the weighted MSE is selected. The resulting vector $\hat{l}_i$, $i = 1,...,10$ is rearranged to guarantee a minimum distance of 0.0006. This process is done for each of the two MA predictors defined by $\mathcal{L}0$, and the MA predictor $L0$ that produces the lowest weighted MSE is selected. As was explained at the beginning of this clause, the resulting vector $\hat{l}_i$ is rearranged twice and a stability check is applied to produce the quantized LSF coefficients $\hat{\omega}_i$.

3.2.5 Interpolation of the LSP coefficients

The quantized (and unquantized) LP coefficients are used for the second subframe. For the first subframe, the quantized (and unquantized) LP coefficients are obtained by linear interpolation of the corresponding parameters in the adjacent subframes. The interpolation is done on the LSP coefficients in the cosine domain. Let $q_i^{(current)}$ be the LSP coefficients computed for the current 10 ms frame, and $q_i^{(previous)}$ the LSP coefficients computed in the previous 10 ms frame. The (unquantized) interpolated LSP coefficients in each of the two subframes are given by:

$$\begin{aligned} &\textit{Subframe}\,1: q_i^{(1)} = 0.5q_i^{(previous)} + 0.5q_i^{(current)} && i = 1,...,10 \\ &\textit{Subframe}\,2: q_i^{(2)} = q_i^{(current)} && i = 1,...,10 \end{aligned} \quad (24)$$

The same interpolation procedure is used for the interpolation of the quantized LSP coefficients by substituting q_i by $\hat{q}_i$ in equation (24).

3.2.6 LSP to LP conversion

Once the LSP coefficients are quantized and interpolated, they are converted back to the LP coefficients a_i. This conversion is done as follows: The coefficients of $F_1(z)$ and $F_2(z)$ are found by expanding equations (13) and (14) knowing the quantized and interpolated LSP coefficients. The coefficients $f_1(i)$, $i = 1,...,5$ are computed from q_i using the recursive relation:

```
for $i = 1$ to 5
    $f_1(i) = -2q_{2i-1} f_1(i-1) + 2 f_1(i-2)$
    for $j = i-1$ down to 1
        $f_1^{[i]}(j) = f_1^{[i-1]}(j) - 2q_{2i-1} f_1^{[i-1]}(j-1) + f_1^{[i-1]}(j-2)$
    end
end
```

with initial values $f_1(0) = 1$ and $f_1(-1) = 0$. The coefficients $f_2(i)$ are computed similarly by replacing q_{2i-1} by q_{2i}.

Once the coefficients $f_1(i)$ and $f_2(i)$ are found, $F_1(z)$ and $F_2(z)$ are multiplied by $1 + z^{-1}$ and $1 - z^{-1}$, respectively, to obtain $F_1'(z)$ and $F_2'(z)$; that is:

$$\begin{aligned} f_1'(i) &= f_1(i) + f_1(i-1) \quad i = 1,...,5 \\ f_2'(i) &= f_2(i) - f_2(i-1) \quad i = 1,...,5 \end{aligned} \tag{25}$$

Finally the LP coefficients are computed from $f_1'(i)$ and $f_2'(i)$ by:

$$a_i = \begin{cases} 0.5 f_1'(i) + 0.5 f_2'(i) & i = 1,...,5 \\ 0.5 f_1'(11-i) - 0.5 f_2'(11-i) & i = 6,...,10 \end{cases} \tag{26}$$

This is directly derived from the relation $A(z) = (F_1'(z) + F_2'(z))/2$, and because $F_1'(z)$ and $F_2'(z)$ are symmetric and antisymmetric polynomials, respectively.

3.3 Perceptual weighting

The perceptual weighting filter is based on the unquantized LP filter coefficients a_i, and is given by:

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} = \frac{1 + \sum_{i=1}^{10} \gamma_1^i a_i z^{-i}}{1 + \sum_{i=1}^{10} \gamma_2^i a_i z^{-i}} \tag{27}$$

The values of γ_1 and γ_2 determine the frequency response of the filter $W(z)$. By proper adjustment of these variables it is possible to make the weighting more effective. This is done by making γ_1 and γ_2 a function of the spectral shape of the input signal. This adaptation is done once per 10 ms frame, but an interpolation procedure for each first subframe is used to smooth this adaptation process. The spectral shape is obtained from a 2nd order linear prediction filter, obtained as a by-product from the Levinson-Durbin recursion (see clause 3.2.2). The reflection coefficients k_i are converted to log area ratio (LAR) coefficients o_i by:

$$o_i = log \frac{(1.0 + k_i)}{(1.0 - k_i)} \quad i = 1, 2 \tag{28}$$

The LAR coefficients corresponding to the current 10 ms frame are used for the second subframe. The LAR coefficients for the first subframe are obtained through linear interpolation with the LAR parameters from the previous frame. The interpolated LAR coefficients in each of the two subframes are given by:

$$\begin{aligned} &Subframe\,1: o_i^{(1)} = 0.5 o_i^{(previous)} + 0.5 o_i^{(current)} \quad && i = 1, 2 \\ &Subframe\,2: o_i^{(2)} = o_i^{(current)} && i = 1, 2 \end{aligned} \tag{29}$$

The spectral envelope is characterized as being either flat (*flat* = 1) or tilted (*flat* = 0). For each subframe this characterization is obtained by applying a threshold function to the LAR coefficients. To avoid rapid changes, a hysteresis is used by taking into account the value of *flat* in the previous subframe $m - 1$,

$$flat^{(m)} = \begin{cases} 0 & \text{if } o_1^{(m)} < -1.74 \text{ and } o_2^{(m)} > 0.65 \text{ and } flat^{(m-1)} = 1 \\ 1 & \text{if } \left(o_1^{(m)} > -1.52 \text{ or } o_2^{(m)} < 0.43\right) \text{and } flat^{(m-1)} = 0 \\ flat^{(m-1)} & \text{otherwise} \end{cases} \tag{30}$$

If the interpolated spectrum for a subframe is classified as flat ($flat^{(m)} = 1$), the weight factors are set to $\gamma_1 = 0.94$ and $\gamma_2 = 0.6$. If the spectrum is classified as tilted ($flat^{(m)} = 0$), the value of γ_1 is set to 0.98, and the value of γ_2 is adapted to the strength of the resonances in the LP synthesis filter, but is bounded between 0.4 and 0.7. If a strong resonance is present, the value of γ_2 is set closer to the upper bound. This adaptation is achieved by a criterion based on the minimum distance between two successive LSP coefficients for the current subframe. The minimum distance is given by:

$$d_{min} = min[\omega_{i+1} - \omega_i] \quad i = 1,...,9 \tag{31}$$

The value of γ_2 is computed using the linear relationship:

$$\gamma_2 = -6.0 d_{min} + 1.0 \quad \text{bounded by } 0.4 \le \gamma_2 \le 0.7 \tag{32}$$

The weighted speech signal in a subframe is given by:

$$sw(n) = s(n) + \sum_{i=1}^{10} a_i \gamma_1^i s(n-i) - \sum_{i=1}^{10} a_i \gamma_2^i sw(n-i) \qquad n = 0,...,39 \tag{33}$$

The weighted speech signal $sw(n)$ is used to find an estimation of the pitch delay in the speech frame.

3.4 Open-loop pitch analysis

To reduce the complexity of the search for the best adaptive-codebook delay, the search range is limited around a candidate delay T_{op}, obtained from an open-loop pitch analysis. This open-loop pitch analysis is done once per frame (10 ms). The open-loop pitch estimation uses the weighted speech signal $sw(n)$ of equation (33), and is done as follows: In the first step, three maxima of the correlation:

$$R(k) = \sum_{n=0}^{79} sw(n) sw(n-k) \tag{34}$$

are found in the following three ranges:

$$\begin{aligned} i &= 1: \ 80,...,143 \\ i &= 2: \ 40,...,79 \\ i &= 3: \ 20,...,39 \end{aligned}$$

The retained maxima $R(t_i)$, $i = 1,...,3$, are normalized through:

$$R'(t_i) = \frac{R(t_i)}{\sqrt{\sum_n sw^2(n - t_i)}} \qquad i = 1,...,3 \tag{35}$$

The winner among the three normalized correlations is selected by favouring the delays with the values in the lower range. This is done by weighting the normalized correlations corresponding to the longer delays. The best open-loop delay T_{op} is determined as follows:

```
$T_{op} = t_1$
$R'(T_{op}) = R'(t_1)$
if $R'(t_2) \geq 0.85\, R'(T_{op})$
        $R'(T_{op}) = R'(t_2)$
        $T_{op} = t_2$
end
if $R'(t_3) \geq 0.85\, R'(T_{op})$
        $R'(T_{op}) = R'(t_3)$
        $T_{op} = t_3$
end
```

This procedure of dividing the delay range into three sections and favouring the smaller values is used to avoid choosing pitch multiples.

3.5 Computation of the impulse response

The impulse response $h(n)$ of the weighted synthesis filter $W(z)/\hat{A}(z)$ is needed for the search of adaptive and fixed codebooks. The impulse response $h(n)$ is computed for each subframe by filtering a signal consisting of the coefficients of the filter $A(z/\gamma_1)$ extended by zeros through the two filters $1/\hat{A}(z)$ and $1/A(z/\gamma_2)$.

3.6 Computation of the target signal

The target signal $x(n)$ for the adaptive-codebook search is usually computed by subtracting the zero-input response of the weighted synthesis filter $W(z)/\hat{A}(z) = A(z/\gamma_1)/[\hat{A}(z)A(z/\gamma_2)]$ from the weighted speech signal $sw(n)$ of equation (33). This is done on a subframe basis.

An equivalent procedure for computing the target signal, which is used in this Recommendation, is the filtering of the LP residual signal $r(n)$ through the combination of synthesis filter $1/\hat{A}(z)$ and the weighting filter $A(z/\gamma_1)/A(z/\gamma_2)$. After determining the excitation for the subframe, the initial states of these filters are updated by filtering the difference between the residual and excitation signals. The memory update of these filters is explained in clause 3.10.

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 40 as will be explained in the next clause. The LP residual is given by:

$$r(n) = s(n) + \sum_{i=1}^{10} \hat{a}_i s(n-i) \qquad n = 0,...,39 \tag{36}$$

3.7 Adaptive-codebook search

The adaptive-codebook parameters (or pitch parameters) are the delay and gain. In the adaptive-codebook approach for implementing the pitch filter, the excitation is repeated for delays less than the subframe length. In the search stage, the excitation is extended by the LP residual to simplify the closed-loop search. The adaptive-codebook search is done every (5 ms) subframe. In the first subframe, a fractional pitch delay T_1 is used with a resolution of 1/3 in the range of $\left[19\frac{1}{3}, 84\frac{2}{3}\right]$ and integers only in the range [85, 143]. For the second subframe, a delay T_2 with a resolution of 1/3 is always used in the range $int(T_1)-5\frac{2}{3}$, $\text{int}(T_1)+4\frac{2}{3}$, where $int(T_1)$ is the integer part of the fractional pitch delay T_1 of the first subframe. This range is adapted for the cases where T_1 straddles the boundaries of the delay range.

For each subframe, the optimal delay is determined using closed-loop analysis that minimizes the weighted mean-squared error. In the first subframe the delay T_1 is found by searching a small range (six samples) of delay values around the open-loop delay T_{op} (see clause 3.4). The search boundaries t_{min} and t_{max} are defined by:

```
$t_{min} = T_{op} - 3$
$if\ t_{min} < 20$ then $t_{min} = 20$
$t_{max} = t_{min} + 6$
if $t_{max} > 143$ then
    $t_{max} = 143$
    $t_{min} = \text{t}_{max} - 6$
end
```

For the second subframe, closed-loop pitch analysis is done around the pitch selected in the first subframe to find the optimal delay T_2. The search boundaries are between $t_{min} - \frac{2}{3}$ and $t_{max} + \frac{2}{3}$ where t_{min} and t_{max} are derived from T_1 as follows:

```
$t_{min} = int(T_1) - 5$
if $t_{min} < 20$ then $t_{min} = 20$
$t_{max} = t_{min} + 9$
if $t_{max} > 143$ then
    $t_{max} = 143$
    $t_{min} = \text{t}_{max} - 9$
end
```

The closed-loop pitch search minimizes the mean-squared weighted error between the original and reconstructed speech. This is achieved by maximizing the term:

$$R(k) = \frac{\sum_{n=0}^{39} x(n) y_k(n)}{\sqrt{\sum_{n=0}^{39} y_k(n) y_k(n)}} \tag{37}$$

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 a preselected value, which is the open-loop pitch T_{op} for the first subframe, and T_1 for the second subframe.

The convolution $y_k(n)$ is computed for the delay t_{min}. For the other integer delays in the search range $k = t_{min} + 1,...,t_{max}$, it is updated using the recursive relation:

$$y_k(n) = y_{k-1}(n-1) + u(-k)h(n) \quad n = 39,...,0 \tag{38}$$

where $u(n)$, $n = -143,...,39$, is the excitation buffer, and $y_{k-1}(-1) = 0$. Note that in the search stage, the samples $u(n)$, $n = 0,...,39$ are not known, and they are needed for pitch delays less than 40. To simplify the search, the LP residual is copied to $u(n)$ to make the relation in equation (38) valid for all delays.

For the determination of T_2, and T_1, if the optimum integer closed-loop delay is less than 85, the fractions around the optimum integer delay have to be tested. The fractional pitch search is done by interpolating the normalized correlation in equation (37) and searching for its maximum. The interpolation is done using a FIR filter b_{12} based on a Hamming windowed sinc function with the sinc truncated at ±11 and padded with zeros at ±12 ($b_{12}(12) = 0$). The filter has its cut-off frequency (–3 dB) at 3600 Hz in the oversampled domain. The interpolated values of $R(k)$ for the fractions $-\frac{2}{3}$, $-\frac{1}{3}$, 0, $\frac{1}{3}$ and $\frac{2}{3}$ are obtained using the interpolation formula:

$$R(k)_t = \sum_{i=0}^{3} R(k-i)b_{12}(t+3i) + \sum_{i=0}^{3} R(k+1+i)b_{12}(3-t+3i) \quad t = 0,1,2 \tag{39}$$

where $t = 0, 1, 2$ corresponds to the fractions 0, $\frac{1}{3}$ and $\frac{2}{3}$, respectively. Note that it is necessary to compute the correlation terms in equation (37) using a range $t_{min} - 4$, $t_{max} + 4$, to allow for the proper interpolation.

3.7.1 Generation of the adaptive-codebook vector

Once the pitch delay has been determined, the adaptive-codebook vector $v(n)$ is computed by interpolating the past excitation signal $u(n)$ at the given integer delay k and fraction t:

$$v(n) = \sum_{i=0}^{9} u(n-k+i)b_{30}(t+3i) + \sum_{i=0}^{9} u(n-k+1+i)b_{30}(3-t+3i) \quad n = 0,...,39 \quad t = 0,1,2 \tag{40}$$

The interpolation filter b_{30} is based on a Hamming windowed sinc functions truncated at ±29 and padded with zeros at ±30 [$b_{30}(30) = 0$]. The filter has a cut-off frequency (–3 dB) at 3600 Hz in the oversampled domain.

3.7.2 Codeword computation for adaptive-codebook delays

The pitch delay T_1 is encoded with 8 bits in the first subframe and the relative delay in the second subframe is encoded with 5 bits. A fractional delay T is represented by its integer part $int(T)$, and a fractional part $frac/3$, $frac = -1, 0, 1$. The pitch index $P1$ is now encoded as:

$$P1 = \begin{cases} 3(int(T_1) - 19) + frac - 1 & \text{if } T_1 = [19,...,85], frac = [-1, 0, 1] \\ (int(T_1) - 85) + 197 & \text{if } T_1 = [86,...,143], frac = 0 \end{cases} \tag{41}$$

The value of the pitch delay T_2 is encoded relative to the value of T_1. Using the same interpretation as before, the fractional delay T_2 represented by its integer part $int(T_2)$, and a fractional part $frac/3$, $frac = -1, 0, 1$, is encoded as:

$$P2 = 3(int(T_2) - t_{min}) + frac + 2 \tag{42}$$

where t_{min} is derived from T_1 as in clause 3.7.

To make the coder more robust against random bit errors, a parity bit $P0$ is computed on the delay index $P1$ of the first subframe. The parity bit is generated through an XOR operation on the six most significant bits of $P1$. At the decoder this parity bit is recomputed and if the recomputed value does not agree with the transmitted value, an error concealment procedure is applied.

3.7.3 Computation of the adaptive-codebook gain

Once the adaptive-codebook delay is determined, the adaptive-codebook gain g_p is computed as:

$$g_p = \frac{\sum_{n=0}^{39} x(n)y(n)}{\sum_{n=0}^{39} y(n)y(n)} \quad \text{bounded by } 0 \le g_p \le 1.2 \qquad (43)$$

where $x(n)$ is the target signal and $y(n)$ is the filtered adaptive-codebook vector [zero-state response of $W(z)/\hat{A}(z)$ to $v(n)$]. This vector is obtained by convolving $v(n)$ with $h(n)$:

$$y(n) = \sum_{i=0}^{n} v(i)h(n-i) \quad n = 0,...,39 \qquad (44)$$

3.8 Fixed codebook – Structure and search

The fixed codebook is based on an algebraic codebook structure using an interleaved single-pulse permutation (ISPP) design. In this codebook, each codebook vector contains four non-zero pulses. Each pulse can have either the amplitudes +1 or –1, and can assume the positions given in Table 7.

Table 7 – Structure of fixed codebook C

Pulse	Sign	Positions
i_0	s_0: ±1	m_0: 0, 5, 10, 15, 20, 25, 30, 35
i_1	s_1: ±1	m_1: 1, 6, 11, 16, 21, 26, 31, 36
i_2	s_2: ±1	m_2: 2, 7, 12, 17, 22, 27, 32, 37
i_3	s_3: ±1	m_3: 3, 8, 13, 18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39

The codebook vector $c(n)$ is constructed by taking a zero vector of dimension 40, and putting the four unit pulses at the found locations, multiplied with their corresponding sign:

$$c(n) = s_0\delta(n-m_0) + s_1\delta(n-m_1) + s_2\delta(n-m_2) + s_3\delta(n-m_3) \quad n = 0,...,39 \qquad (45)$$

where $\delta(0)$ is a unit pulse. A special feature incorporated in the codebook is that the selected codebook vector is filtered through an adaptive pre-filter $P(z)$ that enhances harmonic components to improve the quality of the reconstructed speech. Here the filter:

$$P(z) = 1/\left(1 - \beta z^{-T}\right) \qquad (46)$$

is used, where T is the integer component of the pitch delay of the current subframe, and β is a pitch gain. The value of β is made adaptive by using the quantized adaptive-codebook gain from the previous subframe, that is:

$$\beta = \hat{g}_p^{(m-1)} \quad \text{bounded by } 0.2 \le \beta \le 0.8 \qquad (47)$$

For delays less than 40, the codebook $c(n)$ of equation (45) is modified according to:

$$c(n) = \begin{cases} c(n) & n = 0,...,T-1 \\ c(n) + \beta c(n-T) & n = T,...,39 \end{cases} \tag{48}$$

This modification is incorporated in the fixed-codebook search by modifying the impulse response $h(n)$ according to:

$$h(n) = \begin{cases} h(n) & n = 0,...,T-1 \\ h(n) + \beta h(n-T) & n = T,...,39 \end{cases} \tag{49}$$

3.8.1 Fixed-codebook search procedure

The fixed codebook is searched by minimizing the mean-squared error between the weighted input speech $sw(n)$ of equation (33) and the weighted reconstructed speech. The target signal used in the closed-loop pitch search is updated by subtracting the adaptive-codebook contribution. That is:

$$x'(n) = x(n) - g_p y(n) \quad n = 0,...,39 \tag{50}$$

where $y(n)$ is the filtered adaptive-codebook vector of equation (44) and g_p the adaptive-codebook gain of equation (43).

The matrix **H** is defined as the lower triangular Toepliz convolution matrix with diagonal $h(0)$ and lower diagonal $h(1),...,h(39)$. The matrix Φ = $\mathbf{H}^t\mathbf{H}$ contains the correlations of $h(n)$, and the elements of this symmetric matrix are given by:

$$\phi(i,j) = \sum_{n=j}^{39} h(n-i)h(n-j) \quad i = 0,...,39 \quad j = i,...,39 \tag{51}$$

The correlation signal $d(n)$ is obtained from the target signal $x'(n)$ and the impulse response $h(n)$ by:

$$d(n) = \sum_{i=n}^{39} x'(i)h(i-n) \quad n = 0,...,39 \tag{52}$$

If c_k is the kth fixed-codebook vector, then the codebook is searched by maximizing the term:

$$\frac{C_k^2}{E_k} = \frac{\left(\sum_{n=0}^{39} d(n)c_k(n)\right)^2}{c_k^t \Phi c_k} \tag{53}$$

where t denotes transpose.

The signal $d(n)$ and the matrix Φ are computed before the codebook search. Note that only the elements actually needed are computed and an efficient storage procedure has been designed to speed up the search procedure.

The algebraic structure of the codebook C allows for a fast search procedure since the codebook vector c_k contains only four non-zero pulses. The correlation in the numerator of equation (53) for a given vector c_k is given by:

$$C = \sum_{i=0}^{3} s_i d(m_i) \tag{54}$$

where m_i is the position of the ith pulse and s_i is its amplitude. The energy in the denominator of equation (53) is given by:

$$E\sum_{i=0}^{3}\phi(m_i,m_i)+2\sum_{i=0}^{2}\sum_{j=i+1}^{3}s_is_j\phi(m_i,m_j) \tag{55}$$

To simplify the search procedure, the pulse amplitudes are predetermined by quantizing the signal $d(n)$. This is done by setting the amplitude of a pulse at a certain position equal to the sign of $d(n)$ at the position. Before the codebook search, the following steps are done. First, the signal $d(n)$ is decomposed into two parts: its absolute value $|d(n)|$ and its sign *sign* $[d(n)]$. Second, the matrix Φ is modified by including the sign information; that is:

$$\phi'(i,j)=sign[d(i)]sign[d(j)]\phi(i,j) \quad i=0,...,39 \quad j=i+1,...,39 \tag{56}$$

The main-diagonal elements of Φ are scaled to remove the factor 2 in equation (55)

$$\phi'(i,i)=0.5\phi'(i,i) \quad i=0,...,39 \tag{57}$$

The correlation in equation (54) is now given by:

$$C=|d(m_0)|+|d(m_1)|+|d(m_2)|+|d(m_3)| \tag{58}$$

and the energy in equation (55) is given by:

$$\begin{aligned}E/2=&\phi'(m_0,m_0)\\&+\phi'(m_1,m_1)+\phi'(m_0,m_1)\\&+\phi'(m_2,m_2)+\phi'(m_0,m_2)+\phi'(m_1,m_2)\\&+\phi'(m_3,m_3)+\phi'(m_0,m_3)+\phi'(m_1,m_3)+\phi'(m_2,m_3)\end{aligned} \tag{59}$$

A focused search approach is used to further simplify the search procedure. In this approach, a precomputed threshold is tested before entering the last loop, and the loop is entered only if this threshold is exceeded. The maximum number of times the loop can be entered is fixed so that a low percentage of the codebook is searched. The threshold is computed based on the correlation C. The maximum absolute correlation and the average correlation due to the contribution of the first three pulses, max_3 and av_3, are found before the codebook search. The threshold is given by:

$$thr_3=av_3+K_3(max_3-av_3) \tag{60}$$

The fourth loop is entered only if the absolute correlation (due to three pulses) exceeds thr_3, where $0\le K_3<1$. The value of K_3 controls the percentage of codebook search, and it is set here to 0.4. Note that this results in a variable search time. To further control the search, the number of times the last loop is entered (for the two subframes) cannot exceed a certain maximum, which is set here to 180 (the average worst case per subframe is 90 times).

3.8.2 Codeword computation of the fixed codebook

The pulse positions of the pulses i_0, i_1 and i_2, are encoded with 3 bits each, while the position of i_3 is encoded with 4 bits. Each pulse amplitude is encoded with 1 bit. This gives a total of 17 bits for the 4 pulses. By defining $s = 1$ if the sign is positive and $s = 0$ if the sign is negative, the sign codeword is obtained from:

$$S=s_0+2s_1+4s_2+8s_3 \tag{61}$$

and the fixed-codebook codeword is obtained from:

$$C=(m_0/5)+8(m_1/5)+64(m_2/5)+512(2(m_3/5)+jx) \tag{62}$$

where $jx = 0$ if $m_3 = 3, 8,...,38$, and $jx = 1$ if $m_3 = 4, 9,...,39$.

3.9 Quantization of the gains

The adaptive-codebook gain (pitch gain) and the fixed-codebook gain are vector quantized using 7 bits. The gain codebook search is done by minimizing the mean-squared weighted error between original and reconstructed speech which is given by:

$$E = \mathbf{x}^t\mathbf{x} + g_p^2\mathbf{y}^t\mathbf{y} + g_c^2\mathbf{z}^t\mathbf{z} - 2g_p\mathbf{x}^t\mathbf{y} - 2g_c\mathbf{x}^t\mathbf{z} + 2g_pg_c\mathbf{y}^t\mathbf{z} \tag{63}$$

where $\mathbf{x}$ is the target vector (see clause 3.6), $\mathbf{y}$ is the filtered adaptive-codebook vector of equation (44), and $\mathbf{z}$ is the fixed-codebook vector convolved with $h(n)$,

$$z(n) = \sum_{i=0}^{n} c(i)h(n-i) \quad n = 0,...,39 \tag{64}$$

3.9.1 Gain prediction

The fixed-codebook gain g_c can be expressed as:

$$g_c = \gamma g'_c \tag{65}$$

where g'_c is a predicted gain based on previous fixed-codebook energies, and γ is a correction factor.

The mean energy of the fixed-codebook contribution is given by:

$$E = 10\log\left(\frac{1}{40}\sum_{n=0}^{39} c(n)^2\right) \tag{66}$$

After scaling the vector $c(n)$ with the fixed-codebook gain g_c, the energy of the scaled fixed codebook is given by 20 log g_c + E. Let $E^{(m)}$ be the mean-removed energy (in dB) of the (scaled) fixed-codebook contribution at the subframe m, given by:

$$E^{(m)} = 20\log g_c + E - \overline{E} \tag{67}$$

where $\overline{E} = 30\,\text{dB}$ is the mean energy of the fixed-codebook excitation. The gain g_c can be expressed as a function of $E^{(m)}$, E and $\overline{E}$ by:

$$g_c = 10^{\left(E^{(m)} + \overline{E} - E\right)/20} \tag{68}$$

The predicted gain g'_c is found by predicting the log-energy of the current fixed-codebook contribution from the log-energy of previous fixed-codebook contributions. The 4th order MA prediction is done as follows. The predicted energy is given by:

$$\widetilde{E}^{(m)} = \sum_{i=1}^{4} b_i\hat{U}^{(m-i)} \tag{69}$$

where $[b_1\ b_2\ b_3\ b_4]$ = [0.68 0.58 0.34 0.19] are the MA prediction coefficients, and $\hat{U}^{(m)}$ is the quantized version of the prediction error $U^{(m)}$ at subframe m, defined by:

$$U^{(m)} = E^{(m)} - \widetilde{E}^{(m)} \tag{70}$$

The predicted gain g'_c is found by replacing $E^{(m)}$ by its predicted value in equation (68).

$$g'_c = 10^{\left(\widetilde{E}^{(m)} + \overline{E} - E\right)/20} \tag{71}$$

The correction factor γ is related to the gain-prediction error by:

$$U^{(m)} = E^{(m)} - \widetilde{E}^{(m)} = 20\log(\gamma) \tag{72}$$

3.9.2 Codebook search for gain quantization

The adaptive-codebook gain, g_p, and the factor γ are vector quantized using a two-stage conjugate structured codebook. The first stage consists of a 3 bit two-dimensional codebook $\mathcal{GA}$, and the second stage consists of a 4 bit two-dimensional codebook $\mathcal{GB}$. The first element in each codebook represents the quantized adaptive-codebook gain $\hat{g}_p$, and the second element represents the quantized fixed-codebook gain correction factor $\hat{\gamma}$. Given codebook indices GA and GB for $\mathcal{GA}$ and $\mathcal{GB}$, respectively, the quantized adaptive-codebook gain is given by:

$$\hat{g}_p = \mathcal{GA}_1(GA) + \mathcal{GB}_1(GB) \tag{73}$$

and the quantized fixed-codebook gain by:

$$\hat{g}_\text{c} = g'_c\hat{\gamma} = g'_c\left(\mathcal{GA}_2(GA) + \mathcal{GB}_2(GB)\right) \tag{74}$$

This conjugate structure simplifies the codebook search by applying a preselection process. The optimum pitch gain g_p, and fixed-codebook gain g_c, are derived from equation (63), and are used for the preselection. The codebook $\mathcal{GA}$ contains eight entries in which the second element (corresponding to g_c) has, in general, larger values than the first element (corresponding to g_p). This bias allows a preselection using the value of g_c. In this preselection process, a cluster of four vectors whose second elements are close to g_c are selected. Similarly, the codebook $\mathcal{GB}$ contains 16 entries in which each has a bias towards the first element (corresponding to g_p). A cluster of eight vectors whose first elements are close to g_p are selected. Hence for each codebook the best 50% candidate vectors are selected. This is followed by an exhaustive search over the remaining $4 \times 8 = 32$ possibilities, such that the combination of the two indices minimizes the weighted mean-squared error of equation (63).

3.9.3 Codeword computation for gain quantizer

The codewords *GA* and *GB* for the gain quantizer are obtained from the indices corresponding to the best choice. To reduce the impact of single bit errors the codebook indices are mapped.

3.10 Memory update

An update of the states of the synthesis and weighting filters is needed to compute the target signal in the next subframe. After the two gains are quantized, the excitation signal, $u(n)$, in the present subframe is obtained using:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n) \qquad n = 0,...,39 \tag{75}$$

where $\hat{g}_p$ and $\hat{g}_c$ are the quantized adaptive and fixed-codebook gains, respectively, $v(n)$ is the adaptive-codebook vector (interpolated past excitation), and $c(n)$ is the fixed-codebook vector including harmonic enhancement. 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 $A(\text{z}/\gamma_1)/A(z/\gamma_2)$ for the 40 sample subframe and saving the states of the filters. This would require three filter operations. A simpler approach, which requires only one filter operation, is as follows. The locally reconstructed 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 $e(n)$, $n = 30,...,39$. Updating the states of the filter $A(z/\gamma_1)/A(z/\gamma_2)$ can be done by filtering the error signal $e(n)$ through this filter to find the perceptually weighted error $ew(n)$. However, the signal $ew(n)$ can be equivalently found by:

$$ew(n) = x(n) - \hat{g}_p y(n) - \hat{g}_c z(n) \tag{76}$$

Since the signals $x(n)$, $y(n)$ and $z(n)$ are available, the states of the weighting filter are updated by computing $ew(n)$ as in equation (76) for $n = 30,...,39$. This saves two filter operations.

4 Functional description of the decoder

The principle of the decoder was shown in clause 2 (Figure 3). First, the parameters are decoded (LP coefficients, adaptive-codebook vector, fixed-codebook vector and gains). The transmitted parameters are listed in Table 8. These decoded parameters are used to compute the reconstructed speech signal as will be described in clause 4.1. This reconstructed signal is enhanced by a post-processing operation consisting of a postfilter, a high-pass filter and an upscaling (see clause 4.2). Clause 4.4 describes the error concealment procedure used when either a parity error has occurred, or when the frame erasure flag has been set. A detailed signal flow diagram of the decoder is shown in Figure 6.

Table 8 – Description of transmitted parameters indices

Symbol	Description	Bits
$L0$	Switched MA predictor of LSP quantizer	1
$L1$	First stage vector of quantizer	7
$L2$	Second stage lower vector of LSP quantizer	5
$L3$	Second stage higher vector of LSP quantizer	5
$P1$	Pitch delay first subframe	8
$P0$	Parity bit for pitch delay	1
$C1$	Fixed codebook first subframe	13
$S1$	Signs of fixed-codebook pulses 1st subframe	4
$GA1$	Gain codebook (stage 1) 1st subframe	3
$GB1$	Gain codebook (stage 2) 1st subframe	4
$P2$	Pitch delay second subframe	5
$C2$	Fixed codebook 2nd subframe	13
$S2$	Signs of fixed-codebook pulses 2nd subframe	4
$GA2$	Gain codebook (stage 1) 2nd subframe	3
$GB2$	Gain codebook (stage 2) 2nd subframe	4
NOTE – The bit stream ordering is reflected by the order in the table. For each parameter, the most significant bit (MSB) is transmitted first.		

4.1 Parameter decoding procedure

The decoding process is done in the following order.

4.1.1 Decoding of LP filter parameters

The received indices $L0$, $L1$, $L2$ and $L3$ of the LSP quantizer are used to reconstruct the quantized LSP coefficients using the procedure described in clause 3.2.4. The interpolation procedure described in clause 3.2.5 is used to obtain two sets of interpolated LSP coefficients (corresponding to two subframes). For each subframe, the interpolated LSP coefficients are converted to LP filter coefficients a_i, which are used for synthesizing the reconstructed speech in the subframe.

The following steps are repeated for each subframe:

1) decoding of the adaptive-codebook vector;
2) decoding of the fixed-codebook vector;
3) decoding of the adaptive and fixed-codebook gains; and
4) computation of the reconstructed speech.

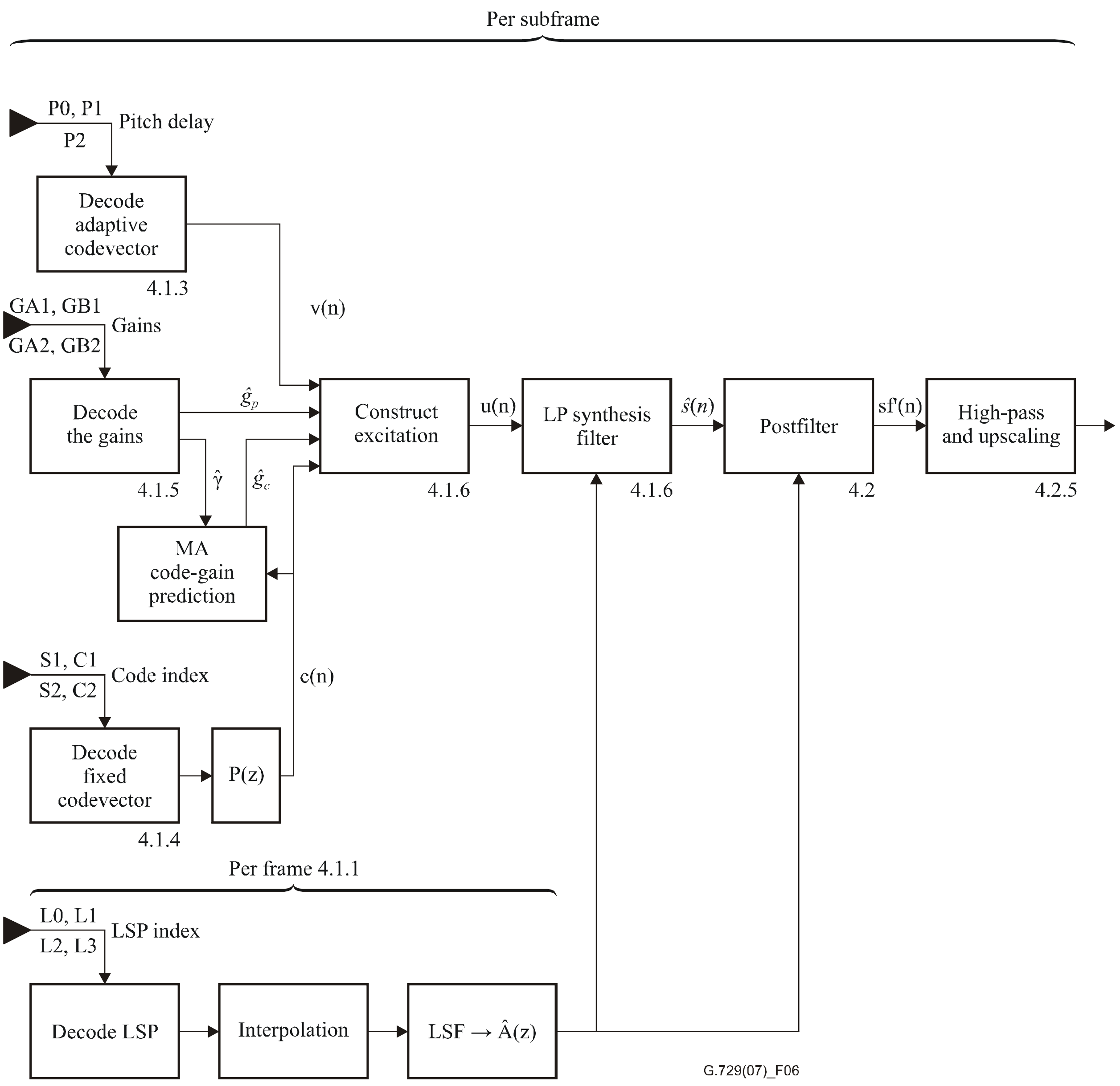


Figure 6 – Signal flow at the CS-ACELP decoder

4.1.2 Computation of the parity bit

Before the excitation is reconstructed, the parity bit is recomputed from the adaptive-codebook delay index *P*1 (see clause 3.7.2). If this bit is not identical to the transmitted parity bit *P*0, it is likely that bit errors occurred during transmission.

If a parity error occurs on *P*1, the delay value T_1 is set to the integer part of the delay value T_2 of the previous frame. The value T_2 is derived with the procedure outlined in clause 4.1.3, using this new value of T_1.

4.1.3 Decoding of the adaptive-codebook vector

If no parity error has occurred, the received adaptive-codebook index $P1$ is used to find the integer and fractional parts of the pitch delay T_1. The integer part $int(T_1)$ and fractional part *frac* of T_1 are obtained from $P1$ as follows:

```
if $P1 < 197$
        $int(T_1) = (P1 + 2)/3 + 19$
        $frac = P1 - 3\,int(T_1) + 58$
else
        $int(T_1) = P1 - 112$
        $frac = 0$
end
```

The integer and fractional part of T_2 are obtained from $P2$ and t_{min}, where t_{min} is derived from T_1 as follows:

```
$t_{min} = int(T_1) - 5$
if $t_{min} < 20$ then $t_{min} = 20$
$t_{max} = t_{min} + 9$
if $t_{max} > 143$ then
        $t_{max} = 143$
        $t_{min} = \mathrm{t}_{max} - 9$
end
```

Now T_2 is decoded using:

$$int(T_2) = (P2 + 2)/3 - 1 + t_{min}$$
$$frac = P2 - 2 - 3((P2 + 2)/3 - 1)$$

The adaptive-codebook vector $v(n)$ is found by interpolating the past excitation $u(n)$ (at the pitch delay) using equation (40).

4.1.4 Decoding of the fixed-codebook vector

The received fixed-codebook index C is used to extract the positions of the excitation pulses. The pulse signs are obtained from S. This is done by reversing the process described in clause 3.8.2. Once the pulse positions and signs are decoded, the fixed-codebook vector $c(n)$ is constructed using equation (45). If the integer part of the pitch delay T is less than the subframe size 40, $c(n)$ is modified according to equation (48).

4.1.5 Decoding of the adaptive and fixed-codebook gains

The received gain-codebook index gives the adaptive-codebook gain $\hat{g}_p$ and the fixed-codebook gain correction factor $\hat{\gamma}$. This procedure is described in detail in clause 3.9. The estimated fixed-codebook gain g'_c is found using equation (71). The fixed-codebook vector is obtained from the product of the quantized gain correction factor with this predicted gain equation (74). The adaptive-codebook gain is reconstructed using equation (73).

4.1.6 Computing the reconstructed speech

The excitation $u(n)$ [see equation (75)] is input to the LP synthesis filter. The reconstructed speech for the subframe is given by:

$$\hat{s}(n) = u(n) - \sum_{i=1}^{10} \hat{a}_i \hat{s}(n-i) \quad n = 0,...,39 \tag{77}$$

where $\hat{a}_i$ are the interpolated LP filter coefficients for the current subframe. The reconstructed speech $\hat{s}(n)$ is then processed by the post-processor described in the next clause.

4.2 Post-processing

Post-processing consists of three functions: adaptive postfiltering, high-pass filtering and signal upscaling. The adaptive postfilter is the cascade of three filters: a long-term postfilter $H_p(z)$, a short-term postfilter $H_f(z)$ and a tilt compensation filter $H_t(z)$, followed by an adaptive gain control procedure. The postfilter coefficients are updated every 5 ms subframe. The postfiltering process is organized as follows. First, the reconstructed speech $\hat{s}(n)$ is inverse filtered through $\hat{A}(z/\gamma_n)$ to produce the residual signal $\hat{r}(n)$. This signal is used to compute the delay T and gain g_t of the long-term postfilter $H_p(z)$. The signal $\hat{r}(n)$ is then filtered through the long-term postfilter $H_p(z)$ and the synthesis filter $1/[g_f\hat{A}(z/\gamma_d)]$. Finally, the output signal of the synthesis filter $1/[g_f\hat{A}(z/\gamma_d)]$ is passed through the tilt compensation filter $H_t(z)$ to generate the postfiltered reconstructed speech signal $sf(n)$. Adaptive gain control is then applied to $sf(n)$ to match the energy of $\hat{s}(n)$. The resulting signal $sf'(n)$ is high-pass filtered and scaled to produce the output signal of the decoder.

4.2.1 Long-term postfilter

The long-term postfilter is given by:

$$H_p(z) = \frac{1}{1+\gamma_p g_l}\left(1+\gamma_p g_l z^{-T}\right) \tag{78}$$

where T is the pitch delay, and g_l is the gain coefficient. Note that g_l is bounded by 1, and it is set to zero if the long-term prediction gain is less than 3 dB. The factor γ_p controls the amount of long-term postfiltering and has the value of $\gamma_p = 0.5$. The long-term delay and gain are computed from the residual signal $\hat{r}(n)$ obtained by filtering the speech $\hat{s}(n)$ through $\hat{A}(z/\gamma_n)$, which is the numerator of the short-term postfilter (see clause 4.2.2).

$$\hat{r}(n) = \hat{s}(n) + \sum_{i=1}^{10} \gamma_n^i \hat{a}_i \hat{s}(n-i) \tag{79}$$

The long-term delay is computed using a two-pass procedure. The first pass selects the best integer T_0 in the range $[int(T_1) - 1, int(T_1) + 1]$, where $int(T_1)$ is the integer part of the (transmitted) pitch delay T_1 in the first subframe. The best integer delay is the one that maximizes the correlation.

$$R(k) = \sum_{n=0}^{39} \hat{r}(n)\hat{r}(n-k) \tag{80}$$

The second pass chooses the best fractional delay T with resolution 1/8 around T_0. This is done by finding the delay with the highest pseudo-normalized correlation.

$$R'(k) = \frac{\sum_{n=0}^{39} \hat{r}(n)\hat{r}_k(n)}{\sqrt{\sum_{n=0}^{39} \hat{r}_k(n)\hat{r}_k(n)}} \tag{81}$$

where $\hat{r}_k(n)$ is the residual signal at delay *k*. Once the optimal delay *T* is found, the corresponding correlation *R'*(*T*) is normalized with the square-root of the energy of $\hat{r}(n)$. The squared value of this normalized correlation is used to determine if the long-term postfilter should be disabled. This is done by setting $g_l = 0$ if:

$$\frac{R'(T)^2}{\sum_{n=0}^{39} \hat{r}(n)\hat{r}(n)} < 0.5 \tag{82}$$

Otherwise the value of g_l is computed from:

$$g_l = \frac{\sum_{n=0}^{39} \hat{r}(n)\hat{r}_k(n)}{\sum_{n=0}^{39} \hat{r}_k(n)\hat{r}_k(n)} \quad \text{bounded by } 0 \le gl \le 1.0 \tag{83}$$

The non-integer delayed signal $\hat{r}_k(n)$ is first computed using an interpolation filter of length 33. After the selection of *T*, $\hat{r}_k(n)$ is recomputed with a longer interpolation filter of length 129. The new signal replaces the previous one only if the longer filter increases the value of *R'*(*T*).

4.2.2 Short-term postfilter

The short-term postfilter is given by:

$$H_f(z) = \frac{1}{g_f}\frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)} = \frac{1}{g_f}\frac{1+\sum_{i=1}^{10}\gamma_n^i \hat{a}_i z^{-i}}{1+\sum_{i=1}^{10}\gamma_d^i \hat{a}_i z^{-i}} \tag{84}$$

where $\hat{A}(z)$ is the received quantized LP inverse filter (LP analysis is not done at the decoder) and the factors γ_n and γ_d control the amount of short-term postfiltering, and are set to $\gamma_n = 0.55$, and $\gamma_d = 0.7$. The gain term g_f is calculated on the truncated impulse response $h_f(n)$ of the filter $\hat{A}(z/\gamma_n)/\hat{A}(z/\gamma_d)$ and is given by:

$$g_f = \sum_{n=0}^{19} |h_f(n)| \tag{85}$$

4.2.3 Tilt compensation

The filter $H_t(z)$ compensates for the tilt in the short-term postfilter $H_f(z)$ and is given by:

$$H_t(z) = \frac{1}{g_t}\left(1+\gamma_t k_1' z^{-1}\right) \tag{86}$$

where $\gamma_t k_1'$ is a tilt factor k_1' being the first reflection coefficient calculated from $h_f(n)$ with:

$$k_1' = -\frac{r_h(1)}{r_h(0)} \quad r_h(i) = \sum_{j=0}^{19-i} h_f(j) h_f(j+i) \tag{87}$$

The gain term $g_t = 1 - |\gamma_t k_1'|$ compensates for the decreasing effect of g_f in $H_f(z)$. Furthermore, it has been shown that the product filter $H_f(z)H_t(z)$ has generally no gain. Two values for γ_t are used depending on the sign of k_1'. If k_1' is negative, $\gamma_t = 0.9$, and if k_1' is positive, $\gamma_t = 0.2$.

4.2.4 Adaptive gain control

Adaptive gain control is used to compensate for gain differences between the reconstructed speech signal $\hat{s}(n)$ and the postfiltered signal $sf(n)$. The gain scaling factor G for the present subframe is computed by:

$$G = \frac{\sum_{n=0}^{39} |\hat{s}(n)|}{\sum_{n=0}^{39} |sf(n)|} \tag{88}$$

The gain-scaled postfiltered signal $sf'(n)$ is given by:

$$sf'(n) = g^{(n)} sf(n) \quad n = 0,...,39 \tag{89}$$

where $g^{(n)}$ is updated on a sample-by-sample basis and given by:

$$g^{(n)} = 0.85 g^{(n-1)} + 0.15\,G \quad \text{n} = 0,...,39 \tag{90}$$

The initial value of $g^{(-1)} = 1.0$ is used. Then for each new subframe, $g^{(-1)}$ is set equal to $g^{(39)}$ of the previous subframe.

4.2.5 High-pass filtering and upscaling

A high-pass filter with a cut-off frequency of 100 Hz is applied to the reconstructed postfiltered speech $sf'(n)$. The filter is given by:

$$H_{h2}(z) = \frac{0.93980581 - 1.8795834 z^{-1} + 0.93980581 z^{-2}}{1 - 1.9330735 z^{-1} + 0.93589199 z^{-2}} \tag{91}$$

The filtered signal is multiplied by a factor 2 to restore the input signal level.

4.3 Encoder and decoder initialization

All static encoder and decoder variables should be initialized to zero, except the variables listed in Table 9.

Table 9 – Description of parameters with non-zero initialization

Variable	Reference	Initial value
β	3.8	0.8
$g^{(-1)}$	4.2.4	1.0
$\hat{l}_i$	3.2.4	$i\pi/11$
q_i	3.2.4	$\arccos(i\pi/11)$
$\hat{U}^{(k)}$	3.9.1	–14

4.4 Concealment of frame erasures

An error concealment procedure has been incorporated in the decoder to reduce the degradation in the reconstructed speech because of frame erasures in the bits tream. This error concealment process is functional when the frame of coder parameters (corresponding to a 10 ms frame) has been identified as being erased. The mechanism for detecting frame erasures is not defined in this Recommendation, and will depend on the application.

The concealment strategy has to reconstruct the current frame, based on previously received information. The method replaces the missing excitation signal with one of similar characteristics, while gradually decaying its energy. This is done by using a voicing classifier based on the long-term prediction gain, which is computed as part of the long-term postfilter analysis. The long-term postfilter (see clause 4.2.1) finds the long-term predictor for which the prediction gain is more than 3 dB. This is done by setting a threshold of 0.5 on the squared normalized correlation of equation (82). For the error concealment process, a 10 ms frame is declared periodic if at least one 5 ms subframe has a long-term prediction gain of more than 3 dB. Otherwise the frame is declared non-periodic. An erased frame inherits its class from the preceding (reconstructed) speech frame. Note that the voicing classification is continuously updated based on this reconstructed speech signal.

The specific steps taken for an erased frame are:

1) repetition of the synthesis filter parameters;
2) attenuation of adaptive and fixed-codebook gains;
3) attenuation of the memory of the gain predictor; and
4) generation of the replacement excitation.

4.4.1 Repetition of synthesis filter parameters

The synthesis filter in an erased frame uses the LP parameters of the last good frame. The memory of the MA LSF predictor contains the values of the received codewords $\hat{l}_i$. Since the codeword is not available for the current frame m, it is computed from the repeated LSF parameters $\hat{\omega}_i$ and the predictor memory using:

$$\hat{l}_i = \left[\hat{\omega}_i^{(m)} - \sum_{k=1}^{4} \hat{P}_{i,k} \hat{l}_i^{(m-k)} \right] \Bigg/ \left(1 - \sum_{k=1}^{4} \hat{P}_{i,k} \right) \quad i = 1,...,10 \tag{92}$$

where the MA predictor coefficients $\hat{p}_{i,k}$ are those of the last received good frame.

4.4.2 Attenuation of adaptive and fixed-codebook gains

The fixed-codebook gain is based on an attenuated version of the previous fixed-codebook gain and is given by:

$$g_c^{(m)} = 0.98 g_c^{(m-1)} \tag{93}$$

where m is the subframe index. The adaptive-codebook gain is based on an attenuated version of the previous adaptive-codebook gain and is given by:

$$g_p^{(m)} = 0.9 g_p^{(m-1)} \quad \text{bounded by } g_p^{(m)} < 0.9 \tag{94}$$

4.4.3 Attenuation of the memory of the gain predictor

As was described in clause 3.9 the gain predictor uses the energy of previously selected fixed-codebook vectors c(n). To avoid transitional effects at the decoder, once good frames are received, the memory of the gain predictor is updated with an attenuated version of the codebook energy. The value of $\hat{U}^{(m)}$ for the current subframe m is set to the averaged quantized gain prediction-error, attenuated by 4 dB:

$$\hat{U}^{(m)} = \left(0.25\sum_{i=1}^{4}\hat{U}^{(m-i)}\right) - 4.0 \text{ bounded by } \hat{U}^{(m)} \geq -14 \tag{95}$$

4.4.4 Generation of the replacement excitation

The excitation used depends on the periodicity classification. If the last reconstructed frame was classified as periodic, the current frame is considered to be periodic as well. In that case, only the adaptive codebook is used, and the fixed-codebook contribution is set to zero. The pitch delay is based on the integer part of the pitch delay in the previous frame, and is repeated for each successive frame. To avoid excessive periodicity the delay is increased by one for each next subframe but bounded by 143. The adaptive-codebook gain is based on an attenuated value according to equation (94).

If the last reconstructed frame was classified as non-periodic, the current frame is considered to be non-periodic as well, and the adaptive-codebook contribution is set to zero. The fixed-codebook contribution is generated by randomly selecting a codebook index and sign index. The random generator is based on the function:

$$seed = 31821\,seed + 13849 \tag{96}$$

with the initial *seed* value of 21845. The fixed-codebook index is derived from the 13 least significant bits of the next random number. The fixed-codebook sign is derived from the 4 least significant bits of the next random number. The fixed-codebook gain is attenuated according to equation (93).

5 Bit-exact description of the CS-ACELP coder

ANSI C code simulating the CS-ACELP coder in 16 bit fixed-point is available from ITU-T. As of the approval of this version of this Recommendation, the current version of this ANSI C code is Version 3.3 of December 1995. More recent versions may become available through corrigenda or amendments to this Recommendation. Please ensure to use the latest available version from the ITU-T website. The following clauses summarize the use of this simulation code, and how the software is organized.

5.1 Use of the simulation software

The C code consists of two main programs, **coder.c**, which simulates the encoder, and **decoder.c**, which simulates the decoder. The encoder is run as follows:

coder inputfile bitstreamfile

The input file and output file are sampled data files containing 16-bit PCM signals. The decoder is run as follows:

decoder bitstreamfile outputfile

The mapping table of the encoded bit stream is contained in the simulation software.

5.2 Organization of the simulation software

In the fixed-point ANSI C simulation, only two types of fixed-point data are used as is shown in Table 10. To facilitate the implementation of the simulation code, loop indices, Boolean values and flags use the type **Flag**, which would be either 16 bits or 32 bits depending on the target platform.

Table 10 – Data types used in ANSI C simulation

Type	Maximal value	Minimal value	Description
Word16	0x7fff	0x8000	Signed 2's complement 16-bit word
Word32	0x7fffffffL	0x80000000L	Signed 2's complement 32-bit word

All the computations are done using a predefined set of basic operators. The description of these operators is given in Table 11. The tables used by the simulation coder are summarized in Table 12. These main programs use a library of routines that are summarized in Tables 13, 14 and 15.

Table 11 – Basic operations used in ANSI C simulation

Operation	Description
Word16 sature(Word32 L_var1)	Limit to 16 bits
Word16 add(Word16 var1, Word16 var2)	Short addition
Word16 sub(Word16 var1, Word16 var2)	Short subtraction
Word16 abs_s(Word16 var1)	Short absolute value
Word16 shl(Word16 var1, Word16 var2)	Short shift left
Word16 shr(Word16 var1, Word16 var2)	Short shift right
Word16 mult(Word16 var1, Word16 var2)	Short multiplication
Word32 L_mult(Word16 var1, Word16 var2)	Long multiplication
Word16 negate(Word16 var1)	Short negate
Word16 extract_h(Word32 L_var1)	Extract high
Word16 extract_l(Word32 L_var1)	Extract low
Word16 round(Word32 L_var1)	Round
Word32 L_mac(Word32 L_var3, Word16 var1, Word16 var2)	Multiply and accumulate
Word32 L_msu(Word32 L_var3, Word16 var1, Word16 var2)	Multiply and subtract
Word32 L_add(Word32 L_var1, Word32 L_var2)	Long addition
Word32 L_sub(Word32 L_var1, Word32 L_var2)	Long subtraction
Word32 L_negate(Word32 L_var1)	Long negate
Word16 mult_r(Word16 var1, Word16 var2)	Multiplication with rounding
Word32 L_shl(Word32 L_var1, Word16 var2)	Long shift left
Word32 L_shr(Word32 L_var1, Word16 var2)	Long shift right
Word16 shr_r(Word16 var1, Word16 var2)	Shift right with rounding
Word16 mac_r(Word32 L_var3, Word16 var1, Word16 var2)	Mac with rounding
Word16 msu_r(Word32 L_var3, Word16 var1, Word16 var2)	Msu with rounding
Word32 L_deposit_h(Word16 var1)	16-bit var1 into MSB part
Word32 L_deposit_l(Word16 var1)	16-bit var1 into LSB part
Word32 L_shr_r(Word32 L_var1, Word16 var2)	Long shift right with round
Word32 L_abs(Word32 L_var1)	Long absolute value

Table 11 – Basic operations used in ANSI C simulation

Operation	Description
Word16 norm_s(Word16 var1)	Short norm
Word16 div_s(Word16 var1, Word16 var2)	Short division
Word16 norm_1(Word32 L_var1)	Long norm

Table 12 – Summary of tables found in tab_ld8.c

Table name	Size	Description
tab_hup_s	28	Upsampling filter for postfilter
tab_hup_1	112	Upsampling filter for postfilter
inter_3	13	FIR filter for interpolating the correlation
inter_3	31	FIR filter for interpolating past excitation
lspcb1	128×10	LSP quantizer (first stage)
lspcb2	32×10	LSP quantizer (second stage)
fg	$2 \times 4 \times 10$	MA predictors in LSP VQ
fg_sum	2×10	Used in LSP VQ
fg_sum_inv	2×10	Used in LSP VQ
gbk1	8×2	Codebook GA in gain VQ
gbk2	16×2	Codebook GB in gain VQ
map1	8	Used in gain VQ
imap1	8	Used in gain VQ
map2	16	Used in gain VQ
ima21	16	Used in gain VQ
window	240	LP analysis window
lag_h	10	Lag window for bandwidth expansion (high part)
lag_1	10	Lag window for bandwidth expansion (low part)
grid	61	Grid points in LP to LSP conversion
tabsqr	49	Lookup table in inverse square root computation
tablog	33	Lookup table in base 2 logarithm computation
table	65	Lookup table in LSF to LSP conversion and vice versa
slope	64	Line slopes in LSP to LSF conversion
tabpow	33	Lookup table in 2^x computation

Table 13 – Summary of encoder-specific routines

Filename	Description
acelp_co.c	Search fixed codebook
cod_1d8k.c	Encoder routine
lpc.c	LP analysis
pitch.c	Pitch search
pre_proc.c	Preprocessing (HP filtering and scaling)
pwf.c	Computation of perceptual weighting coefficients
qua_gain.c	Gain quantizer
qua_1sp.c	LSP quantizer

Table 14 – Summary of decoder-specific routines

Filename	Description
de_acelp.c	Decode algebraic codebook
dec_gain.c	Decode gains
dec_lag3.c	Decode adaptive-codebook index
dec_ld8k.c	Decoder routine
lspdec.c	LSP decoding routing
post_pro.c	Post-processing (HP filtering and scaling)
pst.c	Postfilter routines

Table 15 – Summary of general routines

Filename	Description
basicop2.c	Basic operators
oper_32b.c	Extended basic operators
bits.c	Bit manipulation routines
dspfunc.c	Mathematical functions
filter.c	Filter functions
gainpred.c	Gain predictor
lpcfunc.c	Miscellaneous routines related to LP filter
lspgetq.c	LSP quantizer
p_parity.c	Compute pitch parity
pred_lt3.c	Generation of adaptive codebook
util.c	Utility functions

6 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; 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. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T G.711]	ITU-T Recommendation G.711 (1988), *Pulse code modulation (PCM) of voice frequencies*.
[ITU-T G.712]	ITU-T Recommendation G.712 (2001), *Transmission performance characteristics of pulse code modulation channels*.
[ITU-T G.723.1]	ITU-T Recommendation G.723.1 (2006), *Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s*.
[ITU-T G.726]	ITU-T Recommendation G.726 (1990), *40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*.
[ITU-T G.728]	ITU-T Recommendation G.728 (1992), *Coding of speech at 16 kbit/s using low-delay code excited linear prediction*.
[ITU-T G.729.1]	ITU-T Recommendation G.729.1 (2006), *G.729-based embedded variable bit-rate coder: An 8-32 kbit/s scalable wideband coder bitstream interoperable with G.729*.
[ITU-T V.70]	ITU-T Recommendation V.70 (1996), *Procedures for the simultaneous transmission of data and digitally encoded voice signals over the GSTN, or over 2-wire leased point-to-point telephone type circuits*.

Annex A

Reduced complexity 8 kbit/s CS-ACELP speech codec

(This annex forms an integral part of this Recommendation)

Summary

This annex describes the reduced complexity version of the G.729 speech codec. This version is bit-stream interoperable with the full version.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of reduced complexity CS-ACELP at 8 kbit/s.

A.1 Introduction

This annex provides the high level description of a reduced complexity version of the G.729 speech codec. This version is bit stream-interoperable with the full version, i.e., a reduced complexity encoder may be used with a full implementation of the decoder, and vice versa. However, implementers of the codec defined in this annex should be aware that the performance of this codec may not be as good as the full implementation of the main body of G.729 in certain circumstances.

The reduced complexity version of the codec has been developed for multimedia simultaneous voice and data applications, although the use of the codec is not limited to these applications.

The description of the codec is similar to that of the full implementation of the main body of G.729. This annex describes the changes to the full implementation which have been made in order to reduce the codec algorithmic complexity. For those parts of the algorithm which have not been changed, this annex refers to the appropriate clause of the main Recommendation.

A.2 General description of the codec

The general description of the coding/decoding algorithm is similar to that of the full version. The bit allocation is the same as that given in Table 1. It also has the same delay (speech frame of 10 ms and look-ahead of 5 ms). The major algorithmic changes to the full version of G.729 are summarized below:

– The perceptual weighting filter uses the quantized LP filter parameters and it is given by $W(z) = \hat{A}(z)/\hat{A}(z/\gamma)$ with a fixed value of $\gamma = 0.75$.

– Open-loop pitch analysis is simplified by using decimation while computing the correlations of the weighted speech.

– Computation of the impulse response of the weighted synthesis filter $W(z)/\hat{A}(z)$, computation of the target signal, and updating the filter states are simplified since $W(z)/\hat{A}(z)$ is reduced to $1/\hat{A}(z/\gamma)$.

– The search of the adaptive codebook is simplified. The search maximizes the correlation between the past excitation and the backward filtered target signal (the energy of filtered past excitation is not considered).

– The search of the fixed algebraic codebook is simplified. Instead of the nested-loop focused search, an iterative depth-first tree search approach is used.

– At the decoder, the harmonic postfilter is simplified by using only integer delays.

These changes are described in more detail in clauses A.3 and A.4.

Table A.1 – Summary of the principle routines which have been changed

G.729 routine name	G.729A routine name
Coder_ld8k ( )	Coder_ld8a ( )
Decod_ld8k ( )	Decod_ld8a ( )
Pitch_o1 ( )	Pitch_o1_fast ( )
Pitch_fr3 ( )	Pitch_fr3_fast ( )
ACELP_Codebook ( )	ACELP_Code_A ( )
Post ( )	Post-Filter ( )

A.2.1 Speech codec definition

The description of the reduced complexity speech codec is made in terms of bit-exact, fixed-point mathematical operations. The ANSI-C code indicated in clause A.5, which constitutes an integral part of this annex, reflects this bit-exact, fixed-point descriptive approach. The mathematical description of the encoder (see clause A.3) and the decoder (see clause A.4), can be implemented in several other fashions, possibly leading to a codec implementation not complying with this annex. Therefore, the algorithm description of the ANSI-C code of clause A.5 shall take precedence over the mathematical descriptions of clauses A.3 and A.4 whenever discrepancies are found. A non-exhaustive set of test signals, which can be used with the ANSI-C code, are available from ITU.

As of the approval of this Recommendation, the current version of this ANSI C code is Version 1.1 of September 1996. More recent versions may become available through corrigenda or amendments to this Recommendation. Please ensure to use the latest available version from the ITU-T website.

A.2.2 Notational conventions

Notational conventions are the same as those given in clause 2.5.

A.3 Functional description of the encoder

In this clause the different functions of the encoder represented in the blocks of Figure 4 are described. The main body of this Recommendation is referred to in most of this clause except the parts where algorithmic simplifications have been carried out.

A.3.1 Preprocessing

Same as clause 3.1.

A.3.2 Linear prediction analysis and quantization

A.3.2.1 Windowing and autocorrelation computation

Same as clause 3.2.1.

A.3.2.2 Levinson-Durbin algorithm

Same as clause 3.2.2.

A.3.2.3 LP to LSP conversion

Same as clause 3.2.3 with some simplifications. The number of points at which the polynomials $F_1(z)$ and $F_2(z)$ are evaluated is reduced to 50 (instead of 60), and the sign change interval is divided two times instead of four times for tracking the root of the polynomial.

A.3.2.4 Quantization of the LSP coefficients

Same as clause 3.2.4.

A.3.2.5 Interpolation of the LSP coefficients

Same as clause 3.2.5, but only the quantized LP coefficients are interpolated since the weighting filter uses the quantized parameters for simplicity.

A.3.2.6 LSP to LP conversion

Same as clause 3.2.6.

A.3.3 Perceptual weighting

Unlike clause 3.3, the perceptual weighting filter is based on the quantized LP filter coefficients $\hat{a}_i$, and is given by:

$$W(z)=\frac{\hat{A}(z)}{\hat{A}(z/\gamma)} \tag{A.1}$$

with $\gamma = 0.75$. This simplifies the combination of synthesis and weighting filters to $W(z)/\hat{A}(z) = 1/\hat{A}(z/\gamma)$, which reduces the number of filtering operations while computing the impulse response and the target signal and while updating the filter states. Note that the value of γ is fixed to 0.75 and the procedure for the adaptation of the factors of the perceptual weighting filter described in clause 3.3 is not used in this reduced complexity version.

The weighted speech signal is not used for computing the target signal since an alternative approach is used (see clause A.3.6). However, the weighted speech signal (low-pass filtered) is used to compute the open-loop pitch estimate. The low-pass filtered weighted speech is found by filtering the speech signal $s(n)$ through the filter $\hat{A}(z)/[\hat{A}(z/\gamma)(1 - 0.7z^{-1})]$. First, the coefficients of the filter $A'(z) = \hat{A}(z/\gamma)(1 - 0.7z^{-1})$ are computed, then the low-pass filtered weighted speech in a subframe is computed by:

$$S_w(n)=r(n)-\sum_{i=1}^{10} a_i' s_w(n-i), \quad n=0,...,39 \tag{A.2}$$

where $r(n)$ is the LP residual signal given by:

$$r(n)=s(n)+\sum_{i=1}^{10} \hat{a}_i s(n-i), \quad n=0,...,39 \tag{A.3}$$

The signal $s_W(n)$ is used to find an estimation of the pitch delay in the speech frame.

A.3.4 Open-loop pitch analysis

To reduce the complexity of the search for the best adaptive-codebook delay, the search range is limited around a candidate delay T_{op}, obtained from an open-loop pitch analysis. This open-loop pitch analysis is done once per frame (10 ms). The open-loop pitch estimation uses the low-pass filtered weighted speech signal $s_w(n)$ of equation (A.2), and is done as follows: in the first step, 3 maxima of the correlation:

$$R(k)=\sum_{n=0}^{39} s_w(2n)s_w(2n-k) \tag{A.4}$$

are found in the following three ranges:

$i = 1$: 20,...,39

$i = 2$: 40,...,79

$i = 3$: 80,...,143

The retained maxima $R(t_i)$, $i = 1,...,3$, are normalized through:

$$R'(t_i) = \frac{R(t_i)}{\sqrt{\sum_{n=0}^{39} s_w^2(2n - t_i)}}, \quad i = 1,...,3 \tag{A.5}$$

The winner among the three normalized correlations is selected by favouring the delays with the values in the lower range. This is done by augmenting the normalized correlations corresponding to the lower delay range if their delays are submultiples of the delays in the higher delay range.

Note that in computing the correlations in equation (A.4) only the even samples are used. Further, in the third delay region [80, 143] only the correlations at the even delays are computed in the first pass, then the delays at ±l of the selected even delay are tested.

A.3.5 Computation of the impulse response

The impulse response $h(n)$ of the weighted synthesis filter $W(z)/\hat{A}(z)$ *is* needed for the search of adaptive and fixed codebooks. The impulse response $h(n)$ is computed for each subframe by filtering a signal consisting of a unit sample extended by zeros through the filter $1/\hat{A}(z/\gamma)$.

A.3.6 Computation of the target signal

The target signal $x(n)$ for the adaptive-codebook search is computed by filtering of the LP residual signal $r(n)$ through the weighted synthesis filter $1/\hat{A}\ (z/\gamma)$. After determining the excitation for the subframe, the initial states of this filter are updated as explained in clause A.3.10.

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. The computation of the LP residual is given in equation (A.3).

A.3.7 Adaptive-codebook search

The adaptive-codebook structure is the same as in clause 3.7. In the first subframe, a fractional pitch delay T_1 is used with a resolution of 1/3 in the range $\left[19\frac{1}{3}, 84\frac{2}{3}\right]$ and integers only in the range [85, 143]. For the second subframe, a delay T_2 with a resolution of 1/3 is always used in the range $\left[int(T_1) - 5\frac{2}{3}, int(T_1) + 4\frac{2}{3}\right]$, where $int(T_1)$ *is* the integer part of the fractional pitch delay T_1 of the first subframe. This range is adapted for the cases where T_1 straddles the boundaries of the delay range.

The search boundaries t_{min} and $t_{\max}$ for both subframes are determined in the same way as in clause 3.7.

Closed-loop pitch search is usually performed by maximizing the term:

$$R(k) = \frac{\sum_{n=0}^{39} x(n) y_k(n)}{\sqrt{\sum_{n=0}^{39} y_k(n) y_k(n)}} \tag{A.6}$$

where $x(n)$ is the target signal and $y_k(n)$ is the past filtered excitation at delay k [past excitation convolved with $h(n)$]. In order to simplify the search in this reduced complexity version, only the numerator in equation (A.6) is maximized. That is, the term:

$$R_N(k) = \sum_{n=0}^{39} x(n) y_k(n) = \sum_{n=0}^{39} x_b(n) u_k(n) \tag{A.7}$$

is maximized, where $x_b(n)$ is the backward filtered target signal (correlation between $x(n)$ and the impulse response $h(n)$) and $u_k(n)$ is the past excitation at delay k ($u(n-k)$). Note that the search range is limited around a preselected value, which is the open-loop pitch T_{op} for the first subframe, and T_1 for the second subframe.

Note that in the search stage, the samples $u(n)$, $n = 0,...,39$ are not known, and they are needed for pitch delays less than 40. To simplify the search, the LP residual is copied to $u(n)$.

For the determination of T_2 and T_1 if the optimum integer delay is less than 85, the fractions around the optimum integer delay have to be tested. The fractional pitch search is done by interpolating the past excitation at fractions $-\frac{1}{3}$, 0 and $\frac{1}{3}$, and selecting the fraction which maximizes the correlation in equation (A.7). The interpolation of the past excitation is performed using the same FIR filter, b_{30}, which is defined in clause 3.7. The interpolated past excitation at a given integer delay k and fraction t is given by:

$$u_{kt}(n) = \sum_{i=0}^{9} u(n-k+i) b_{30}(t+3i) + \sum_{i=0}^{9} u(n-k+1+i) b_{30}(3-t+3i),\ n = 0,...,39,\ t = 0,1,2 \tag{A.8}$$

A.3.7.1 Generation of the adaptive-codebook vector

Same as clause 3.7.1.

A.3.7.2 Codeword computation for adaptive-codebook delays

Same as clause 3.7.2.

A.3.7.3 Computation of the adaptive-codebook gain

Same as clause 3.7.3.

A.3.8 Fixed codebook – Structure and search

The structure of the 17-bit algebraic codebook is the same as clause 3.8.

A.3.8.1 Fixed-codebook search procedure

The signs of the pulses are found using the same approach explained in clause 3.8.1. However, the pulse positions are found using a more efficient approach. Instead of the nested-loop search approach, an iterative depth-first, tree search approach is used. In this new approach a smaller number of pulse position combinations is tested and it has fixed complexity.

A.3.8.2 Codeword computation of the fixed codebook

Same as clause 3.8.2.

A.3.9 Quantization of the gains

Same as clause 3.9.

A.3.10 Memory update

An update of the states of the weighted synthesis filter is needed for computing the target signal in the next subframe. After the two gains are quantized, the excitation signal, $u(n)$, in the present subframe is obtained using:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n), \quad n = 0,...,39 \tag{A.9}$$

where $\hat{g}_p$ and $\hat{g}_c$, are the quantized adaptive and fixed-codebook gains, respectively, $v(n)$ is the adaptive-codebook vector (interpolated past excitation), and $c(n)$ *is* the fixed-codebook vector including harmonic enhancement. The states of the weighted synthesis filter can be updated by filtering the signal $r(n) - u(n)$ (difference between residual and excitation) through the filter $1/\hat{A}(z/\gamma)$ for the 40 sample subframe and saving the states of the filter. A simpler approach, which requires no filter operations, is as follows. The output of the filter due to the input $r(n) - u(n)$ is the weighted error signal $e_w(n)$ which can be found by:

$$e_w(n) = x(n) - \hat{g}_p y(n) - \hat{g}_c z(n) \tag{A.10}$$

where $x(n)$ is the target signal, $y(n)$ is the filtered adaptive-codebook vector and $z(n)$ is the filtered fixed-codebook vector. Since the signals $x(n)$, $y(n)$, and $z(n)$ are available, the states of the weighted synthesis filter are updated by computing $e_w(n)$ as in equation (A.10) for $n = 30,...,39$.

A.4 Functional description of the decoder

The principle of the decoder is shown in Figure 3. The transmitted parameters are the same as listed in Table 8. The decoded parameters are used to compute the reconstructed speech signal. This reconstructed signal is enhanced by a post-processing operation consisting of a postfilter, a high-pass filter and an upscaling (see clause A.4.2). The detailed signal flow diagram of the decoder is the same one shown in Figure 6.

The only change in the decoder is in the postfilter which is described in clause A.4.2.

A.4.1 Parameter decoding procedure

Same as clause 4.1.

A.4.2 Post-processing

The post-processing is the same as in clause 4.2 except for some simplification in the adaptive postfilter.

The adaptive postfilter is the cascade of three filters: a long-term postfilter $H_p(z)$, a short-term postfilter $H_f(z)$ and a tilt compensation filter $H_t(z)$, followed by an adaptive gain control procedure. The long-term postfilter is simplified by using only integer delays. In the short-term postfilter and the tilt compensation filter, the gain terms g_f and g_t are not used.

The postfiltering process is similar to that described in the main body of this Recommendation with the exception that the compensation filtering is performed before synthesis filtering through $1/\hat{A}(z/\gamma_d)$.

A.4.2.1 Long-term postfilter

The long-term postfilter is given by:

$$H_p(z) = \frac{1}{1 + \gamma_p gl}\left(1 + \gamma_p gl z^{-T}\right) \tag{A.11}$$

The only difference from clause 4.2.1 is that the long-term delay T is always an integer delay and it is computed by searching the range $[T_{cl} - 3, T_{cl} + 3]$, where T_{cl} is the integer part of the (transmitted) pitch delay in the current subframe bounded by $T_{cl} \leq 140$.

A.4.2.2 Short-term postfilter

The short-term postfilter is given by:

$$H_f(z) = \frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)} = \frac{1 + \sum_{i=1}^{10} \gamma_n^i \hat{a}_i z^{-i}}{1 + \sum_{i=1}^{10} \gamma_d^i \hat{a}_i z^{-i}} \qquad \text{(A.12)}$$

where $\hat{A}(z)$ is the received quantized LP inverse filter (LP analysis is not done at the decoder), and the factors γ_n and γ_d control the amount of short-term postfiltering, and are set to $\gamma_n = 0.55$ and $\gamma_d = 0.7$.

The only difference from clause 4.2.2 is that the gain factor g_f is eliminated.

A.4.2.3 Tilt compensation

The filter $H_t(z)$ compensates for the tilt in the short-term postfilter $H_f(z)$ and is given by:

$$H_t(z) = 1 + \gamma_t k_1' z^{-1} \qquad \text{(A.13)}$$

where $\gamma_t k_1'$ is a tilt factor, k_1' being the first reflection coefficient calculated by:

$$k_1' = -\frac{r_h(1)}{r_h(0)}; \quad r_h(i) = \sum_{j=0}^{21-i} h_f(j) h_f(j+i) \qquad \text{(A.14)}$$

where $h_f(n)$ is the truncated impulse response of the filter $\hat{A}(z/\gamma_n)/\hat{A}(z/\gamma_d)$. The value of $\gamma_t = 0.8$ is used if $k_1' < 0$ and γ_t is set to zero if $k_1' \geq 0$. The gain factor g_t which is used in clause 4.2.3 is eliminated.

A.4.2.4 Adaptive gain control

Same as clause 4.2.4. The only difference is that the gain scaling factor G for the present subframe is computed by:

$$G = \sqrt{\frac{\sum_{n=0}^{39} \hat{s}^2(n)}{\sum_{n=0}^{39} sf^2(n)}} \qquad \text{(A.15)}$$

and $g^{(n)}$ is given by:

$$g^{(n)} = 0.9 g^{(n-1)} + 0.1G, \quad n = 0,...,39$$

A.4.2.5 High-pass filtering and upscaling

Same as clause 4.2.5.

A.4.3 Encoder and decoder initialization

Same as clause 4.3.

A.4.4 Concealment of frame erasures

Same as clause 4.4 with the difference that no voicing detection is used. The excitation is always the addition of both adaptive and fixed codebook contributions.

A.5 Bit-exact description of the reduced complexity CS-ACELP codec

The reduced complexity CS-ACELP codec is simulated in 16-bit fixed-point ANSI-C code using the same set of fixed-point basic operators defined in Table 11.

A.5.1 Use of the simulation software

Same as clause 5.1.

A.5.2 Organization of the simulation software

Same as clause 5.2.

The tables used by the simulation codec are found in the file **tab_ld8a.c** which replaces the file **tab_ld8k.c** of the full Recommendation. The difference between these two files is that the tables **tab_hup_s, tab_hup_1**, and **inter_3** found in the file **tab_ld8k.c** are removed from the file **tab_ld8a.c**. Also, the table **grid** has been modified.

The main programs use a library of routines that are provided in the fixed-point ANSI-C simulation. Most of the routines are the same as those of the full Recommendation. The principal routines that have been changed are summarized in Table A.1. Refer to the **read.me** file provided with the software for more details.

Annex B

A silence compression scheme for G.729 optimized for terminals conforming to ITU-T Recommendation V.70

(This annex forms an integral part of this Recommendation)

Summary

This annex defines a voice activity detector and comfort noise generator for use with G.729 or Annex A optimized for V.70 DSVD applications.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 8 kbit/s with DTX functionality.

B.1 Introduction

This annex provides a high level description of the voice activity detection (VAD), discontinuous transmission (DTX) and comfort noise generator (CNG) algorithms. These algorithms are used to reduce the transmission rate during silence periods of speech. They are designed and optimized to work in conjunction with [ITU-T V.70]. [ITU-T V.70] mandates the use of Annex A speech coding methods. However, when it is desirable, the full version of G.729 can also be used to improve the quality of the speech. The algorithms are adapted to operate with both the full version of G.729 and Annex A. This description is for the full version of G.729, the only difference for Annex A is indicated in clause B.3.1.1. A block diagram of a silence compression speech communication system is depicted in Figure B.1.

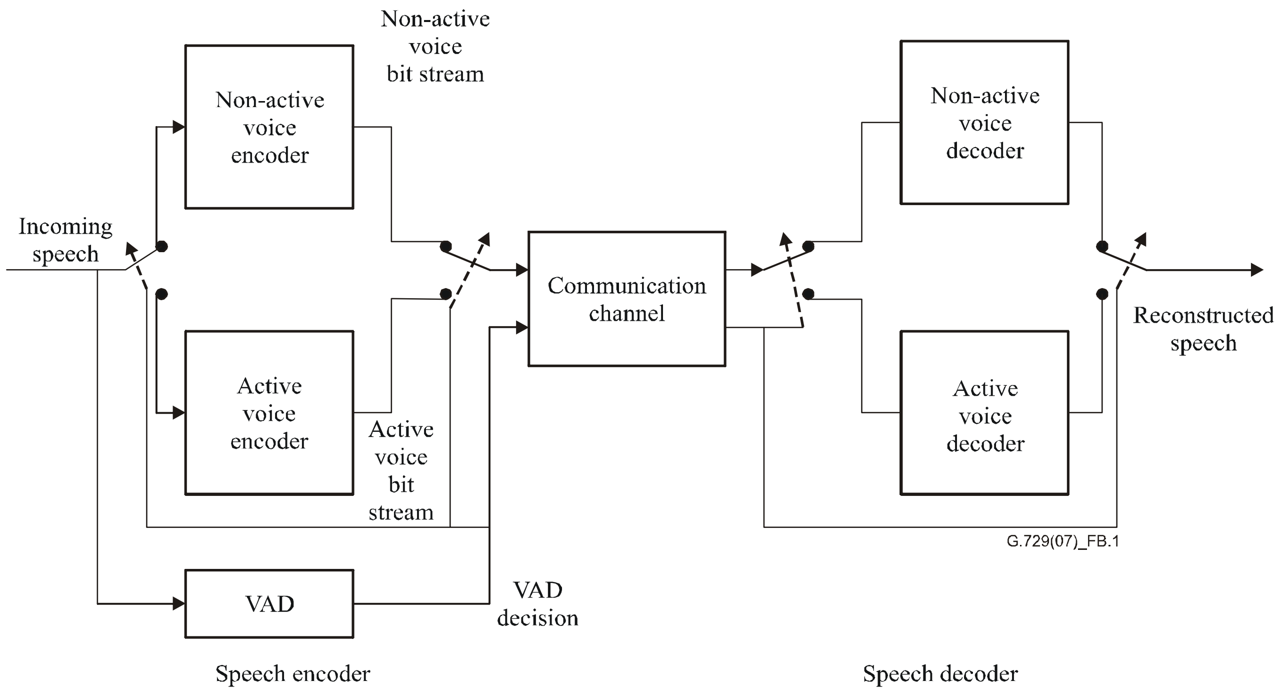


Figure B.1 – Speech communication system with VAD

B.2 General description of the VAD/DTX/CNG algorithms

The VAD algorithm makes a voice activity decision every 10 ms in accordance with the frame size of the G.729 speech coder. A set of difference parameters is extracted and used for an initial decision. The parameters are the full-band energy, the low-band energy, the zero-crossing rate and a spectral measure. The long-term averages of the parameters during non-active voice segments follow the changing nature of the background noise. A set of differential parameters is obtained at each frame. These are a difference measure between each parameter and its respective long-term average. The initial voice activity decision is obtained using a piecewise linear decision boundary between each pair of differential parameters. A final voice activity decision is obtained by smoothing the initial decision.

The output of the VAD module is either 1 or 0, indicating the presence or absence of voice activity respectively. If the VAD output is 1, the G.729 speech codec is invoked to code/decode the active voice frames. However, if the VAD output is 0, the DTX/CNG algorithms described herein are used to code/decode the non-active voice frames. Traditional speech coders and decoders use comfort noise to simulate the background noise in the non-active voice frames. If the background noise is not stationary, a mere comfort noise insertion does not provide the naturalness of the original background noise. Therefore it is desirable to intermittently send some information about the background noise in order to obtain a better quality when non-active voice frames are detected. The coding efficiency of the non-active voice frames can be achieved by coding the energy of the frame and its spectrum with as few as fifteen bits. These bits are not automatically transmitted whenever there is a non-active voice detection. Rather, the bits are transmitted only when an appreciable change has been detected with respect to the last transmitted non-active voice frame.

At the decoder side, the received bit stream is decoded. If the VAD output is 1, the G.729 decoder is invoked to synthesize the reconstructed active voice frames. If the VAD output is 0, the CNG module is called to reproduce the non-active voice frames.

B.3 Detailed description of the VAD algorithm

A flowchart of the VAD operation is given in Figure B.2. The VAD operates on frames of digitized speech. The frames are processed in time order and are consecutively numbered from the beginning of each conversation/recording.

At the first stage, four parametric features are extracted from the input signal. Extraction of the parameters is shared with the active voice encoder module and the non-active voice encoder for computational efficiency. The parameters are the full- and low-band frame energies, the set of line spectral frequencies (LSF) and the frame zero crossing rate.

If the frame number is less than N_i, an initialization stage of the long-term averages takes place, and the voice activity decision is forced to 1 if the frame energy from the LPC analysis is above 15 dB (see equation (B.1)). Otherwise, the voice activity decision is forced to 0. If the frame number is equal to N_i, an initialization stage for the characteristic energies of the background noise occurs.

At the next stage, a set of difference parameters are calculated. This set is generated as a difference measure between the current frame parameters and running averages of the background noise characteristics. Four difference measures are calculated:

– a spectral distortion;
– an energy difference;
– a low-band energy difference; and
– a zero-crossing difference.

The initial voice activity decision is made at the next stage, using multi-boundary decision regions in the space of the four difference measures. The active voice decision is given as the union of the decision regions and the non-active voice decision is its complementary logical decision. Energy considerations, together with neighbouring past frames decisions, are used for decision smoothing.

The running averages have to be updated only in the presence of background noise, and not in the presence of speech. An adaptive threshold is tested, and the update takes place only if the threshold criterion is met.

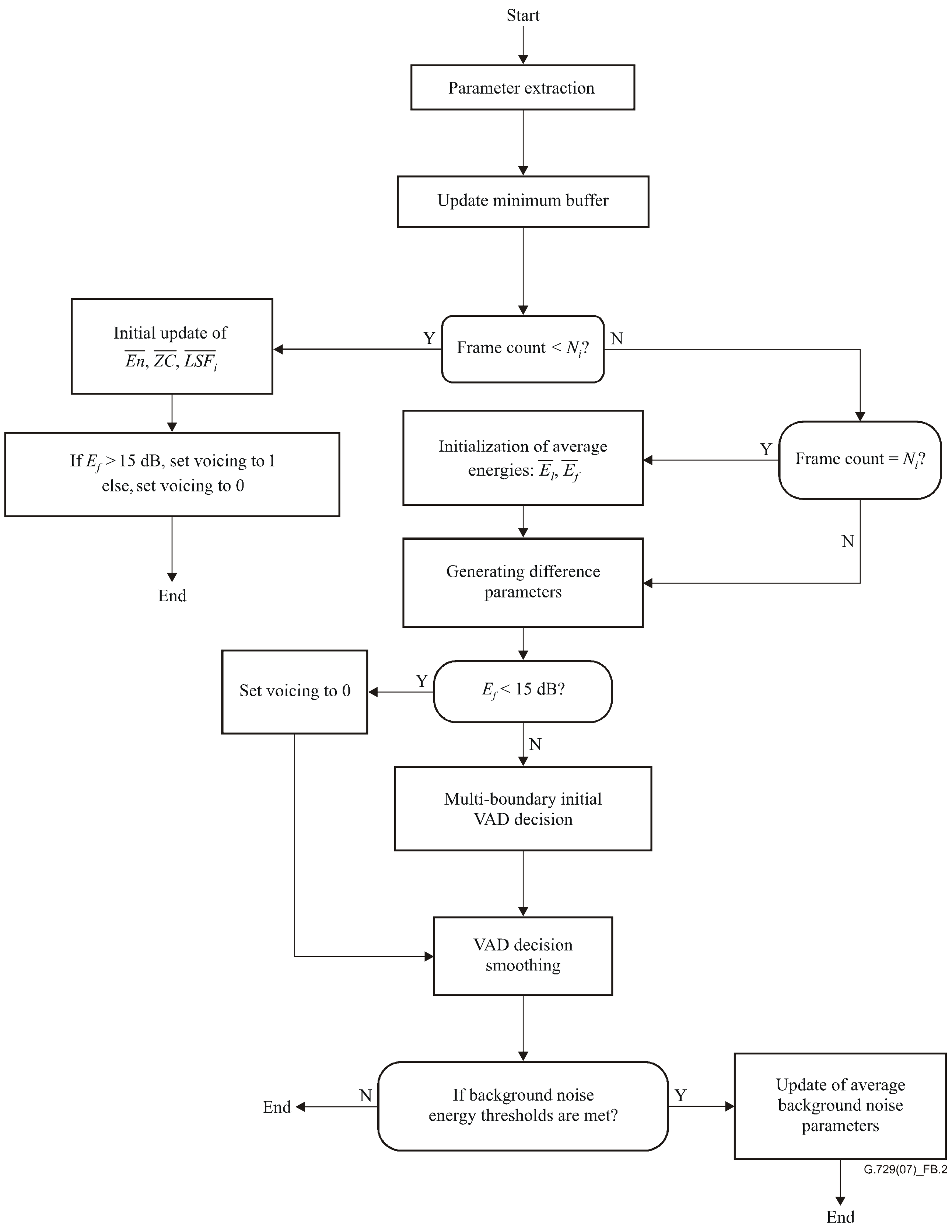

Start
Parameter extraction
Update minimum buffer
Frame count < N_i?
Y
N
Initial update of $\overline{En}$, $\overline{ZC}$, $\overline{LSF_i}$
If E_f > 15 dB, set voicing to 1 else, set voicing to 0
End
Initialization of average energies: $\overline{E_l}$, $\overline{E_f}$
Y
Frame count = N_i?
N
Generating difference parameters
Set voicing to 0
Y
E_f < 15 dB?
N
Multi-boundary initial VAD decision
VAD decision smoothing
End
N
If background noise energy thresholds are met?
Y
Update of average background noise parameters
G.729(07)_FB.2
End

Figure B.2 – VAD flowchart

B.3.1 Parameter extraction

For each frame, a set of parameters is extracted from the speech signal. The parameters extraction module can be shared between the VAD, the active voice encoder and the non-active voice encoder. The basic set of parameters is the set of autocorrelation coefficients, which is derived similarly to the full version of G.729 (see clause 3.2.1). The set of autocorrelation coefficients will be denoted by:

$$\{R(i)\}_{i=0}^{q}, \quad \text{where } q = 12$$

B.3.1.1 Line spectral frequencies (LSF)

A set of linear prediction coefficients is derived from the autocorrelation and a set of $\{LSF_i\}_{i=1}^{p}$, where p = 10, is derived from the set of linear prediction coefficients, as described in clauses 3.2.3 or A.3.2.3.

B.3.1.2 Full-band energy

The full-band energy E_f is the logarithm of the normalized first autocorrelation coefficient $R(0)$:

$$E_f = 10 \cdot log_{10}\left[\frac{1}{N}R(0)\right] \tag{B.1}$$

where $N = 240$ is the LPC analysis window size in speech samples.

B.3.1.3 Low-band energy

The low-band energy E_l measured on 0 to F_l Hz band, is computed as follows:

$$E_l = 10 \cdot log_{10}\left[\frac{1}{N}\mathbf{h}^{\mathbf{T}}\mathbf{Rh}\right] \tag{B.2}$$

where **h** is the impulse response of an FIR filter with cut-off frequency at F_l Hz, **R** is the Toeplitz autocorrelation matrix with the autocorrelation coefficients on each diagonal.

B.3.1.4 Zero-crossing rate

Normalized zero-crossing rate ZC for each frame is calculated by:

$$ZC = \frac{1}{2M}\sum_{i=0}^{M-1}\left[\,|\,sgn[x(i)] - sgn[x(i-1)]\,|\,\right] \tag{B.3}$$

where $\{x(i)\}$ is the preprocessed input signal (see clause 3.1) and $M = 80$.

B.3.2 Initialization of the running averages of the background noise characteristics

For the first N_i frames, the spectral parameters of the background noise, denoted by $\{\overline{LSF_i}\}_{i=1}^{p}$ are initialized as an average of the $\{LSF_i\}_{i=1}^{p}$ of the frames. The average of the background noise zero-crossings, denoted by $\overline{ZC}$ is initialized as an average of the zero-crossing rate ZC of the frames.

The running averages of the background noise energy, denoted by $\overline{E}_f$, and the background noise low-band energy, denoted by $\overline{E}_l$, are initialized as follows. First, the initialization procedure uses $\overline{En}$, defined as the average of the frame energy E_f over the first N_i frames. These three averaging ($\overline{En}, \overline{ZC}$, and $\{\overline{LSF}_i\}_{i=1}^{p}$) include only the frames that have an energy E greater than 15 dB. Second, the initialization procedure continues as follows:

```
if $\overline{En} \le T1$ then
    $\overline{E}_f = \overline{En} + K0$
    $\overline{E}_l = \overline{En} + K1$
else if $T1 < \overline{En} < T2$ then
    $\overline{E}_f = \overline{En} + K2$
    $\overline{E}_l = \overline{En} + K3$
else
    $\overline{E}_f = \overline{En} + K4$
    $\overline{E}_l = \overline{En} + k5$
```

See Table B.1 for constant values.

B.3.3 Generating the long-term minimum energy

A long-term minimum energy parameter, E_{min}, is calculated as the minimum of E_f over N_0 previous frames. Since N_0 is relatively large, E_{min} is calculated using stored values of the minimum of E_f over short segments of the past.

B.3.4 Generating the difference parameters

Four difference measures are generated from the current frame parameters and the running averages of the background noise.

B.3.4.1 The spectral distortion ΔS

The spectral distortion measure is generated as the sum of squares of the difference between the current frame $\{LSFi\}_{i=1}^{p}$ vector and the running averages of the background noise $\{\overline{LSF}_i\}_{i=1}^{p}$:

$$\Delta S = \sum_{i=1}^{p} \left(LSFi - \overline{LSF}_i\right)^2 \qquad \text{(B.4)}$$

B.3.4.2 The full-band energy difference ΔE_f

The full-band energy difference measure is generated as the difference between the current frame energy, E_f, and the running average of the background noise energy, $\overline{E}_f$:

$$\Delta E_f = \overline{E}_f - E_f \qquad \text{(B.5)}$$

B.3.4.3 The low-band energy difference ΔE_l

The low-band energy difference measure is generated as the difference between the current frame low-band energy, E_l, and the running average of the background noise low-band energy, $\overline{E}_l$:

$$\Delta E_l = \overline{E}_l - E_l \qquad \text{(B.6)}$$

B.3.4.4 The zero-crossing difference ΔZC

The zero-crossing difference measure is generated as the difference between the current frame zero-crossing rate, ZC, and the running average of the background noise zero-crossing rate, $\overline{ZC}$:

$$\Delta ZC = \overline{ZC} - ZC \tag{B.7}$$

B.3.5 Multi-boundary initial voice activity decision

The initial voice activity decision is denoted by I_{VD}, and is set to 0 ("FALSE") if the vector of difference parameters lies within the non-active voice region. Otherwise, the initial voice activity decision is set to 1 ("TRUE"). The fourteen boundary decisions in the four-dimensional space are defined as follows:

1) if $\Delta S > a_1 \cdot \Delta ZC + b_1$ then $I_{VD} = 1$
2) if $\Delta S > a_2 \cdot \Delta ZC + b_2$ then $I_{VD} = 1$
3) if $\Delta E_f < a_3 \cdot \Delta ZC + b_3$ then $I_{VD} = 1$
4) if $\Delta E_f < a_4 \cdot \Delta ZC + b_4$ then $I_{VD} = 1$
5) if $\Delta E_f < b_5$ then $I_{VD} = 1$
6) if $\Delta E_f < a_6 \cdot \Delta S + b_6$ then $I_{VD} = 1$
7) if $\Delta S > b_7$ then $I_{VD} = 1$
8) if $\Delta E_f < a_8 \cdot \Delta ZC + b_8$ then $I_{VD} = 1$
9) if $\Delta E_f < a_9 \cdot \Delta ZC + b_9$ then $I_{VD} = 1$
10) if $\Delta E_f < b_{10}$ then $I_{VD} = 1$
11) if $\Delta E_l < a_{11} \cdot \Delta S + b_{11}$ then $I_{VD} = 1$
12) if $\Delta E_l > a_{12} \cdot \Delta E_f + b_{12}$ then $I_{VD} = 1$
13) if $\Delta E_l < a_{13} \cdot \Delta E_f + b_{13}$ then $I_{VD} = 1$
14) if $\Delta E_l < a_{14} \cdot \Delta E_f + b_{14}$ then $I_{VD} = 1$

If none of the fourteen conditions is "TRUE" $I_{VD} = 0$. See Table B.1 for constant values.

Table B.1 – Table of constants

Name	Constant	Name	Constant
N_i	32	N_1	4
N_0	128	N_2	10
K_0	0	T_1	671088640
K_1	–53687091	T_2	738197504
K_2	–67108864	T_3	26843546
K_3	–93952410	T_4	40265318
K_4	–134217728	T_5	40265318
K_5	–161061274	T_6	40265318
a_1	23488	b_1	28521
a_2	–30504	b_2	19446
a_3	–32768	b_3	–32768
a_4	26214	b_4	–19661
a_5	0	b_5	–30802
a_6	28160	b_6	–19661
a_7	0	b_7	30199
a_8	16384	b_8	–22938
a_9	–19065	b_9	–31576
a_{10}	0	b_{10}	–17367
a_{11}	22400	b_{11}	–27034
a_{12}	30427	b_{12}	29959
a_{13}	–24576	b_{13}	–29491
a_{14}	23406	b_{14}	–28087

B.3.6 Voice activity decision smoothing

The initial voice activity decision is smoothed (hangover) to reflect the long-term stationarity nature of the speech signal. The smoothing is done in four stages.

A flag indicating that hangover has occurred is defined as v_flag. It is set to zero each time before the voice activity decision smoothing is performed. Denote the smoothed voice activity decision of the frame, the previous frame and frame before the previous frame by S_{VD}^{0}, S_{VD}^{-1} and S_{VD}^{-2}, respectively. S_{VD}^{-1} is initialized to 1, and S_{VD}^{-2} is initialized to 1. For start $S_{VD}^{0} = I_{VD}$. The first smoothing stage is:

$$\text{if}\left(I_{VD} = 0\right) \text{and} \left(S_{VD}^{-1} = 1\right) \text{and} \left(E > \overline{E}_f + T3\right) \text{then } S_{VD}^{0} = 1 \text{ and } v\_flag = 1$$

For the second smoothing stage define a Boolean parameter F_{VD}^{-1} and a smoothing counter C_e. F_{VD}^{-1} is initialized to 1 and C_e is initialized to 0. Denote the energy of the previous frame by E_{-1}. The second smoothing stage is:

```
if $(F_{VD}^{-1} = 1)$ and $(I_{VD} = 0)$ and $(S_{VD}^{-1} = 1)$ and $(S_{VD}^{-2} = 1)$ and $(|E_f - E_{-1}| \le T_4)$ {
        $S_{VD}^{0} = 1$
        $v\_flag = 1$
        $C_e = C_e + 1$
        $if(C_e \le N_1)\{$
                $F_{VD}^{-1} = 1$
        }
        else {
                $F_{VD}^{-1} = 0$
                $C_e = 0$
        }
}
else
$F_{VD}^{-1} = 1$
```

For the third smoothing stage define a noise continuity counter C_s, which is initialized to 0. If $S_{VD}^{0} = 0$ then C_s is incremented. The third smoothing stage is:

```
if $(S_{VD}^{0} = 1)$ and $(C_s > N_2)$ and $(E_f - E_{-1} \le T_5)$ {
        $S_{VD}^{0} = 0$
        $C_s = 0$
        }
if $(S_{VD}^{0} = 1)\, C_s = 0$
```

In the fourth stage, a voice activity decision is made if the following condition is satisfied:

if $((E_f < \overline{E}_f + T_6)$ and $(frm\_count > N_0)$ and $(v\_flag = 0))$ then $S_{VD}^{0} = 0$

B.3.7 Updating the running averages of the background noise characteristics

The running averages of the background noise characteristics are updated at the last stage of the VAD module. At this stage, the following condition is tested and the updating takes place if the following condition is met:

if $(E_f < \overline{E}_f + T_6)$ then update

The running averages of the background noise characteristics are updated using a first order auto-regressive (AR) scheme. Different AR coefficients are used for different parameters, and different sets of coefficients are used at the beginning of the recording/conversation or when a large change of the noise characteristics is detected.

Let β_{E_f} be the AR coefficient for the update of $\overline{E}_f$, β_{E_l} be the AR coefficient for the update of $\overline{E}_l$, β_{ZC} be the AR coefficient for the update of $\overline{ZC}$ and β_{LSF} be the AR coefficient for the update of $\left\{\overline{LSF}_i\right\}_{i=1}^{p}$. The total number of frames where the update condition was satisfied is counted by C_n. Different set of the coefficients β_{E_f}, β_{E_l}, β_{ZC} and β_{LSF} is used according to the value of C_n.

The AR update is done according to:

$$\begin{aligned}
\overline{E}_f &= \beta_{E_f} \cdot \overline{E}_f + \left(1-\beta_{E_f}\right) \cdot E_f \\
\overline{E}_l &= \beta_{E_l} \cdot \overline{E}_l + \left(1-\beta_{E_l}\right) \cdot E_l \\
\overline{ZC} &= \beta_{ZC} \cdot \overline{ZC} + \left(1-\beta_{ZC}\right) \cdot ZC \\
\overline{LSF}_i &= \beta_{LSF} \cdot \overline{LSF}_i + \left(1-\beta_{LSF}\right) \cdot LSF_i \qquad i = 1,\ldots,p
\end{aligned} \tag{B.8}$$

$\overline{E}_f$ and C_n are further updated according to:

```
if $(frame\ count > N_0)$ and $(\overline{E}_f < E_{min})${
    $\overline{E}_f = E_{min}$
    $C_n = 0$
}
```

B.4 Detailed description of the DTX/CNG algorithms

The DTX/CNG algorithms provide continuous and smooth information about the non-active voice periods, while keeping a low average bit rate.

B.4.1 Description of the DTX algorithm

For each non-active voice frame, the DTX module decides if a set of non-active voice update parameters ought to be sent to the speech decoder by measuring the changes in the non-active voice signal. Absolute and adaptive thresholds on the frame energy and the spectral distortion measure are used to obtain the update decision. If an update is needed, the non-active voice encoder sends the information needed to generate a signal which is perceptually similar to the original non-active voice signal. This information is comprised of an energy level and a description of the spectral envelope. If no update is needed, the non-active voice signal is generated by the non-active decoder according to the last received energy and spectral shape information of a non-active voice frame.

However, a minimum interval of $N_{min} = 2$ frames is required between two consecutive SID frames i.e., if a spectral or level change has occurred $n < N_{min}$ frames after a SID frame, the SID emission is delayed.

Situated at the transmitting end, the DTX module receives from the VAD module the active/non-active voice information, and from the encoder modules the autocorrelation function of the speech signal computed for each 80 sample frame and the past excitation sample. For each frame, the DTX decision $Ftyp_t$ (Frame type for frame numbered t) is output as one of the three values, 0, 1 or 2 corresponding to untransmitted frame, active speech frame or SID frame, respectively, according to the following procedure:

B.4.1.1 Store the frame autocorrelation function

For every frame t (active or inactive), the autocorrelation coefficients of the current frame t, including the bandwidth expansion and noise correction (see the G.729 description) are retained in memory. The set of frame t autocorrelations will be denoted $r_t'(j)$, for $j = 0,\ldots,10$.

B.4.1.2 Computation of the current frame type

If the current frame t is an active speech frame ($Vad_t = 1$), then the current frame type $Ftyp_t = 1$ and the normal speech encoder processing continues.

In the other case, a current LPC filter $A_t(z)$ calculated over $N_{cur} = 2$ previous frames including the current one t is first evaluated:

The N_{cur} autocorrelation functions are summed:

$$R^t(j) = \sum_{i=t-N_{cur}+1}^{t} r_i'(j), \qquad j = 0,...,10 \tag{B.9}$$

and $A_t(z)$ is calculated by the Levinson-Durbin procedure (see the G.729 description) using $R^t(j)$ as input. The coefficients of this filter will be noted $a_t(j)$, $j = 0,...,10$. The Levinson-Durbin procedure also provides the residual energy E_t, that will be rescaled and used as an estimate of the frame excitation energy.

Then the current frame type $Ftyp_t$ is determined in the following way:

– If the current frame is the first inactive frame of the inactive zone, the frame is selected as the SID frame. The variable $\overline{E}$ which reflects the energy sum is taken equal to E_t, and the number of frames involved in the summation, k_E, is initialized to 1:

$$(Vad_{t-1} = 1) \Rightarrow \begin{cases} Ftyp_t = 2 \\ \overline{E} = E_t \\ k_E = 1 \end{cases} \tag{B.10}$$

– For the other frames, the algorithm compares the preceding SID parameters to the current ones: if the current filter is significantly different to the preceding SID filter, or if the current excitation energy significantly differs from the preceding SID energy, the flag *flag_chang* is set to 1, else it does not change.

– The counter *count_fr* indicating how many frames are elapsed since the previous SID frame is incremented. If its value is greater than N_{min}, the emission of a SID frame is allowed. Then if *flag_chang* is equal to 1, a SID frame is sent. In all other cases, the current frame is untransmitted:

$$\left.\begin{array}{l} count\_fr \geq N_{min} \\ flag\_chang = 1 \end{array}\right\} \Rightarrow Ftyp_t = 2 \tag{B.11}$$

Otherwise: $Ftyp_t = 0$

In case of a SID frame, the counter *count_fr* and the flag *flag_chang* are re-initialized to 0.

LPC filters and energies are compared according to the following methods:

B.4.1.3 Comparison of the LPC filters

The previous SID-LPC filter will be noted $A_{sid}(z)$ and its coefficients $a_{sid}(j)$, $j = 0,...,10$ (the evaluation of this filter is described in clause B.4.2.2). The current and previous SID-LPC filters are considered as significantly different if the Itakura distance between the two filters exceeds a given threshold, which is expressed by:

$$\sum_{j=0}^{10} R_a(i) \times R^t(i) \geq E_t \times thr1 \tag{B.12}$$

where $R_a(j)$, $j = 0,...,10$ is a function derived from the autocorrelation of the coefficients of the SID filter, given by:

$$\begin{cases} R_a(j) = 2\sum_{k=0}^{10-j} a_{sid}(k) \times a_{sid}(k+j) & \text{if } j \neq 0 \\ R_a(0) = \sum_{k=0}^{10} a_{sid}(k)^2 \end{cases} \tag{B.13}$$

A value of 1.20226 is used for *thr*1.

B.4.1.4 Comparison of the energies

The sum the frame energies is calculated, k_E being first incremented up to the maximum value $Ng = 2$:

$$\overline{E} = \sum_{i=t-k_E+1}^{t} E_i \tag{B.14}$$

Then $\overline{E}$ is quantized, using the 5-bits logarithmic quantizer described in clause B.4.2.1. The decoded log-energy E_q is compared to the previous decoded SID log-energy E_q^{sid}. If the difference exceeds the threshold *thr*2=2 dB, the two energies will be considered as significantly different.

B.4.2 SID evaluation and quantization

The silence insertion descriptor (SID) is comprised of the quantized frame excitation energy (i.e., the current quantized excitation energy $Q(\overline{E})$ for the SID frames) and the quantized LSPs corresponding to the estimated SID-LPC filter. Four indices make up the SID frame. One index describes the energy and three indices describe the spectrum portion of the SID frame.

B.4.2.1 Energy quantization

The quantization of the energy $\overline{E}$ is performed as follows. First, a scaling factor $\alpha_w = 0.125$ is introduced that takes into account the effect of windowing and bandwidth expansions present in the subframes autocorrelation functions $r'(j)$.

The value used at the input of the gain quantizer is:

$$E' = \alpha_w \times \frac{1}{k_E \times N_{cur} \times 80} \overline{E} \tag{B.15}$$

The energy term E' is quantized with a 5-bit non-uniform quantizer in the logarithmic domain in the range of –12 dB to 66 dB. A uniform step size of 2 dB is used between 16 dB and 66 dB. A step size of 4 dB is used in the range of –4 dB to 16 dB. Below –4 dB, a single step size of 8 dB is used giving a quantization level of –12 dB. The quantization is straightforward and does not need the storage of a quantizer table.

Notice that since the energy comparison (see clause B.4.1.4) is performed with decoded energies, the quantization of the energy is done for all non-active voice frames.

B.4.2.2 SID-LPC filter estimation and quantization

The SID-LPC filter estimation takes into account the local stationarity or non-stationarity of the noise at the SID frame neighbourhood.

First, a past average filter $\overline{A}_p(z)$ built from N_p frames preceding the current SID one is calculated, using the following autocorrelation sum as input of the Levinson-Durbin procedure:

$$\overline{R}_p(j) = \sum_{k=t'-N_p}^{t'} r'_k(j), \qquad j = 0,...,10 \tag{B.16}$$

The number of frames involved in the summation has been fixed to $N_p = 6$.

The frame number t' varies in $[t-1, t-N_{cur}]$, depending on the rest of the Euclidian division of the current frame number t by N_{cur}.

The SID-LPC filter is then obtained with:

$$A_{sid}(z) = \begin{cases} A_t(z) \text{ if distance } \left(A_t(z), \overline{A}_p(z)\right) \geq thr3 \\ \overline{A}_p(z) \text{ otherwise} \end{cases} \tag{B.17}$$

The threshold value *thr*3 is fixed to 1.12202 and the distance between the current LPC filter and the past average one is calculated in the same manner as in clause B.4.1.3 (see equation (B.12)).

Then the SID-LPC filter is transformed to the LSF domain for quantization. The LSFs are quantized by a two-stage switched predictive vector quantization (VQ) with 5 and 4 bits each. The quantization of the LSF vector entails the determination of the best three indices. The first index is that of the predictor. The last two indices are each taken from a different vector table, as it is done in a two stage vector quantization. The overall quantization procedure follows the one given in clause 3.2.4 with the following modifications:

1) The second 4th-order MA predictor used in the full version of G.729 is modified as a linear combination of the first and second MA predictors as follows:

$$p_{i,k,2} = 0.6 p_{i,k,1} + 0.4 p_{i,k,2} \tag{B.18}$$

where

$$i = 1,...,10, k = 1,...,4$$

2) The first stage VQ quantization is similar to the one used in the full version of G.729. However, only a portion of the first table of the quantizer is used. The relevant subset entries of the table are stored in an auxiliary lookup table with 32 address indices. Moreover, a delayed decision quantization is used by keeping few candidates as inputs to the second stage.

3) The candidates from the first stage, in conjunction with those of the second stage, are used by the second stage VQ. The second stage VQ quantization is different from the one used in the full version of G.729. A full VQ is used as compared to the split VQ of the full version of G.729. Only a portion of the second stage tables is used as well. The relevant subset entries are stored in another lookup table with two 16-address entries. The combination of the predictor, a vector from the first stage and a vector from the second stage, leading to the minimum distortion in the weighted mean squared error sense, is chosen as the LSF descriptor.

B.4.3 SID bit stream description

The bit stream related to the transmission of a SID frame is described in Table B.2. The bit stream related to the transmission of an active frame is defined in Table 8. The bit stream ordering is reflected by the order in the table. For each parameter the most significant bit (MSB) is transmitted first.

Table B.2 – SID frame bit stream definition

Parameter description	Bits
Switched predictor index of LSF quantizer	1
First stage vector of LSF quantizer	5
Second stage vector of LSF quantizer	4
Gain (energy)	5

B.4.4 Non-active encoder/decoder (CNG) description

At the decoder part, the comfort noise is generated by introducing a pseudo-white excitation signal of controlled level into interpolated LPC filters in the same manner than the decoder produces active speech by filtering the decoded excitation. The excitation level and LPC filters are obtained from the previous SID information. The subframes interpolated LPC filters are obtained by using the SID-LSPs as current LSPs and performing the interpolation with the previous frame LSPs as done for active frames in the full version of G.729.

The pseudo-white excitation $ex(n)$ is a mixture between an excitation of the same type as the active speech one $ex_1(n)$ and a white Gaussian excitation $ex_2(n)$.

The G.729 excitation $ex_1(n)$ is composed of an adaptive excitation with a small gain and an ACELP fixed excitation, which improves the transition between active and non-active voice frames. The addition of a Gaussian excitation $ex_2(n)$ allows the generation of a whiter signal.

Since the encoder and decoder need to keep synchronized during non-active voice periods, the excitation generation is performed on both sides, for SID frames and for untransmitted frames.

First, let us define the target excitation gain $\tilde{G}_t$ as the square root of the average energy that must be obtained for the current frame t synthetic excitation. $\tilde{G}_t$ is calculated using the following smoothing procedure, where $\tilde{G}_{sid}$ is the SID gain derived for the decoded SID gain:

$$\tilde{G}_t = \begin{cases} \tilde{G}_{sid} & \text{if } Vad_{t-1} = 1 \\ \frac{7}{8}\tilde{G}_{t-1} + \frac{1}{8}\tilde{G}_{sid} & \text{otherwise} \end{cases} \tag{B.19}$$

The 80 samples of the frame are divided into 2 subframes of 40 samples. For each subframe, the CNG excitation samples are synthesized using the following algorithm.

A pitch lag is randomly chosen in the interval [40,103].

Next, the fixed codebook vector of the subframe is built by random selection of the grid and the signs and positions of the pulses according to the G.729 ACELP code structure.

An adaptive excitation signal of unity gain is then calculated, noted $e_a(n)$, $n = 0,...,39$. The selected subframe fixed excitation will be noted $e_f(n)$, $n = 0,...,39$.

The adaptive and fixed gains Ga and Gf are then computed in order to yield a subframe average energy equal to $\tilde{G}_t^2$, which is expressed by:

$$\frac{1}{40}\sum_{n=0}^{39}\left(Ga \times e_a(n) + Gf \times e_f(n)\right)^2 = \tilde{G}_t^2 \tag{B.20}$$

Notice that Gf can take a negative value.

Let us define $Ea = \left(\sum_{n=0}^{39} e_a(n)^2\right)$, $I = \left(\sum_{n=0}^{119} e_a(n)e_f(n)\right)$ and $K = 40 \times \tilde{G}_t^2$

Due to the ACELP excitation structure $\sum_{n=0}^{39} e_f(n)^2 = 4$

If we fix randomly the adaptive gain *Ga*, then equation (B.19) becomes a second order equation on the fixed gain *Gf*:

$$Gf^2 + \frac{Ga \times I}{2} Gf + \frac{Ea \times Ga^2 - K}{4} = 0 \tag{B.21}$$

A constraint may be imposed on *Ga* to be sure that this equation has a solution. Furthermore it is desirable to forbid the use of large adaptive gains. For this, the adaptive gain *Ga* will be randomly chosen in:

$$\left[0, Max\left\{0.5, \sqrt{\frac{K}{A}}\right\}\right], \quad \text{with } A = Ea - I^2/4 \tag{B.22}$$

The root of equation (B.20) that has the lowest absolute value is selected for *Gf.*

Finally the G.729 excitation is built, using:

$$ex_1(n) = Ga \times e_a(n) + Gf \times e_f[n], \quad n = 0,\ldots,39 \tag{B.23}$$

The method of deriving the composite excitation signal *ex*(*n*) is as follows:

Let E_1 be the energy of $ex_1(n)$, E_2 be the energy of $ex_2(n)$. $ex_2(n)$ has a unit variance and a zero mean. Let E_3 be the cross-energy between $ex_1(n)$ and $ex_2(n)$.

$$\begin{aligned} E_1 &= \sum ex_1^2(n) \\ E_2 &= \sum ex_2^2(n) \\ E_3 &= \sum ex_1(n) \cdot ex_2(n) \end{aligned} \tag{B.24}$$

where the summation is over the subframe size.

Let α and β be the scale proportion of $ex_1(n)$ and $ex_2(n)$ used in the mixture excitation respectively. α is set to be 0.6. β is found as the solution to the following quadratic equation:

$$\beta^2 E_2 + 2\alpha\beta E_3 + (\alpha^2 - 1)E_1 = 0, \quad \text{with } \beta > 0 \tag{B.25}$$

If no solution is found for β, it is set to 0 and α to 1.

The CNG excitation *ex*(*n*) becomes:

$$ex_1(n) = \alpha ex_1(n) + \beta ex_2(n) \tag{B.26}$$

B.4.5 Frame erasure concealment with regards to the CNG

When a frame erasure is detected by the decoder, the erased frame type depends on the preceding frame type:

– if the preceding frame was active, then the current frame is considered as active;

– else if the preceding frame was either a SID frame or an untransmitted frame, the current erased frame is considered as untransmitted:

$$\begin{cases} Ftyp_{t-1} = 1 & \Rightarrow \quad Ftyp_t = 1 \\ Ftyp_{t-1} = 0 \text{ or } 2 & \Rightarrow \quad Ftyp_t = 0 \end{cases} \tag{B.27}$$

If an untransmitted frame has been erased, no error is then introduced.

If a SID frame is erased, there are two possibilities:

– If it is not the first SID frame of the current inactive period, then the previous SID parameters are kept.

– If it is the first SID frame of an inactive period, a special protection has been taken.

Notice first that this case is detected by the fact that $Ftyp_{t-1} = 1$ and $Ftyp_t = 0$.

This combination of events does not imply that the preceding frame was a good active frame: several frames up to the preceding one may have been erased. What is certain is that the last good frame was an active frame, that the present frame was not erased, and that the SID frame supposed to provide information for the current untransmitted frame is lost.

To recover the SID information, the CNG module uses parameters provided by the G.729 decoder main part:

– the LSPs of the last valid active frame are used for the SID-LPC filter;

– an energy term is calculated on the excitation signal by the decoder during the processing of all valid active voice frames. To recover the missing SID gain $\tilde{G}_{sid}$, the energy term of the last valid active frame is quantized with the SID gain quantizer and decoded.

Finally to avoid desynchronization of the random generator used to compute the excitation, the pseudo-random sequence reset is performed at each active frame, both at the encoder and decoder parts.

B.5 Bit-exact description of the silence compression scheme

The silence compression scheme is simulated in 16-bit fixed-point ANSI-C code using the same set of fixed-point basic operators defined in Table 11. The ANSI-C code constitutes an integral part of this Recommendation reflecting the bit-exact fixed-point description of the silence compression scheme. In the event of any discrepancy between the printed text of this Recommendation and the C source, the C source code is presumed to be correct. As of the approval of this text, the current version of this ANSI C code is Version 1.5 of October 2006. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

B.5.1 Organization of the simulation software

Same as clause 5.2.

The Annex B ANSI-C software modules are listed in Table B.3. Refer to the **read.me** file provided with the software for more details.

Table B.3 – List of G.729 Annex B software files

G.729 Annex B ANSI-C module names	**Description**
Vad.c	VAD
Dtx.c	DTX decision
Qsidgain.c	SID gain quantization
QsidLSF.c	SID-LSF quantization
Calcexc.c	CNG excitation calculation
Dec_sid.c	Decode SID information
Miscel.c	Miscellaneous calculations
G.729 Annex B ANSI-C.h file names	Description
Vad.h	Prototype and constants
Dtx.h	Prototype and constants
Sid.h	Prototype and constants
Miscel.h	Prototype and constants

Annex C

Reference floating-point implementation for G.729 CS-ACELP 8 kbit/s speech coding

(This annex forms an integral part of this Recommendation)

Summary

This annex describes an alternative implementation of G.729 Annex A based on floating-point arithmetic. Subjective quality tests have been performed by NTT (Japan) and CNET (France) to assess the quality of these floating-point versions under various conditions (input level, error, background noise, tandeming). Different interoperability configurations with the fixed-point version of the algorithm have also been tested. These tests proved full interoperability of this floating-point implementation to both the full version of G.729 and Annex A.

This annex includes an electronic attachment containing reference C codes for floating-point implementation of CS-ACELP at 8 kbit/s full version and reduced complexity. The design of a set of test vectors remains for further study.

C.1 Scope

This annex provides a description of an alternative implementation in floating-point arithmetic for the full version of G.729 and Annex A. The development of an interoperable floating-point specification for voice activity detection (VAD), discontinuous transmission (DTX) and comfort noise generation (CNG) with similar properties as the fixed-point specification in Annex B is for further study.

C.2 Normative references

This annex refers to materials defined in the main body and Annex A of this Recommendation.

C.3 Overview

The full version of G.729 provides bit-exact fixed-point specification of an algorithm for the coding of speech signals at 8 kbit/s. Annex A is a reduced complexity version interoperable with the full version of G.729. Exact details of these specifications are given in bit-exact fixed-point C code available from ITU-T. This annex describes and defines an alternative implementation of the full version of G.729 and Annex A based on floating-point arithmetic.

C.4 Algorithmic description

This floating-point version of the full version of G.729 (respectively Annex A) has the same algorithm steps as the fixed-point version. Similarly, the bit stream is identical to that of G.729 (respectively to that of Annex A). For algorithmic details, see the full version of G.729 (respectively Annex A).

C.5 ANSI C code

ANSI C code simulating the floating-point version of the full version of G.729 (respectively Annex A) defined in this annex has been developed and is available as an attachment to this annex. The ANSI C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of the full version of G.729, Annex A or this annex. As of the approval of this text, the current version of this ANSI C code is Version 1.01 of 15 September 1998. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website. The structure of these floating-point source codes is related to the corresponding fixed-point source code. As for Annex B to [ITU-T G.723.1], the typedef.h file contains a statement enabling the definition of all floating-point variables and constants as type either double or single. A file called version.h is available to select whether the C code will operate according to the full version of G.729 or Annex A. Tables C.1 to C.3 give the list of the software files names with a brief description. Note that the fixed-point files basic_op.c, oper_32b.c, dspfunc.c and basic_op.h, oper_32b.h are not needed for floating-point arithmetic. A float to short conversion routine has been added to the file util.c.

Table C.1 – List of software files specific to G.729 floating-point source code

File name	Description
coder.c	Main program for G.729 encoder
cod_ld8k.c	G.729 encoder routine
acelp_co.c	G.729 fixed codebook search
lpc.c	G.729 LP analysis
lpcfunc.c	Miscellaneous routines related to LP filter
pitch.c	G.729 pitch search
pwf.c	G.729 computation of perceptual weighting coefficients
decoder.c	Main program for G.729 decoder
dec_ld8k.c	G.729 decoder routine
postfil.c	G.729 postfilter
tab_ld8k.c	G.729 constants tables
ld8k.h	G.729 prototypes and constant declarations
tab_ld8k.h	G.729 declaration of constants tables
version.h	Used to select the G.729 (main body) mode

Table C.2 – List of software files specific to G.729 Annex A floating-point source code

File name	Description
coder.c	Main program for G.729 Annex A encoder
acelp_ca.c	G.729 Annex A fixed codebook search
cod_ld8a.c	G.729 Annex A encoder routine
lpc.c	G.729 Annex A LP analysis
lpcfunc.c	Miscellaneous routines related to LP filter
pitch_a.c	G.729 Annex A pitch search
decoder.c	Main program for G.729 Annex A decoder
dec_ld8a.c	G.729 Annex A decoder routine
postfila.c	G.729 Annex A postfilter
tab_ld8a.c	G.729 Annex A tables of constants
ld8a.h	G.729 Annex A prototypes and constant declarations
tab_ld8a.h	Declaration of G.729 Annex A constants tables
version.h	Used to select the G.729 Annex A mode

Table C.3 – List of software files common to G.729 and G.729 Annex A floating-point source code

File name	Description
bits.c	Bit manipulation routines
qua_lsp.c	LSP quantizer
qua_gain.c	Gain quantizer
cor_func.c	Miscellaneous routines related to excitation computation
de_acelp.c	Algebraic codebook decoder
dec_gain.c	Gain decoder
dec_lag3.c	Adaptive-codebook index decoder
filter.c	Filter functions
gainpred.c	Gain predictor
lspdec.c	LSP decoding routine
lspgetq.c	LSP quantizer
p_parity.c	Pitch parity computation
post_pro.c	Post-processing (HP filtering)
pre_proc.c	Preprocessing (HP filtering)
pred_lt3.c	Generation of adaptive codebook
taming.c	Pitch taming functions
util.c	Utility function
typedef.h	Data type definition (machine-dependent)

Annex C+

Reference floating-point implementation for integrating G.729 CS-ACELP speech coding main body with Annexes B, D and E

(This annex forms an integral part of this Recommendation)

Summary

This annex, dubbed "Annex C+", extends the functionalities of Annex C.

Previous Annex C contains G.729 main body and G.729 Annex A and B floating-point implementation. Annex C+ defines the integration of G.729 main body with Annexes B, D and E in floating-point arithmetic.

This annex includes an electronic attachment containing reference C code for floating point implementation of CS-ACELP at 6.4 kbit/s, 8 kbit/s and 11.8 kbit/s with DTX functionality.

C+.1 Scope

This annex provides a description of integrating the G.729 main body with Annexes B, D and E in floating-point arithmetic. It presents a standard way of performing this integration and expansion of the functionality, hereby guiding the industry and ensuring a standard speech quality and compatibility worldwide. The integration has been performed with focus on several constraints in order to satisfy the need of the industry:

1) Bit-exactness with the main body in floating point (Annex C).
2) Minimum additional program code, memory and complexity usage.
3) Stringent quality requirements to new functionality, in line with quality and application areas of the according standard annexes.

C+.2 Normative references

This annex refers to materials defined in the G.729 main body and Annexes B, C, D and E.

C+.3 Overview

G.729 main body and Annexes B, D and E provide a bit-exact fixed-point specification of a CS-ACELP coder at 8 kbit/s, with DTX functionality, lower and higher bit extension capability at 6.4 and 11.8 kbit/s. Exact details of these specifications are given in bit-exact fixed-point C code in an electronic attachment to this annex. Annex C describes and defines an alternative implementation of G.729 main body. Annex C+ describes and defines the integration of the G.729 main body with Annexes B, D and E in floating-point arithmetic. It can be considered as an extension of Annex C.

C+.4 New functionality

This clause presents a brief overview of the modifications/additions to the algorithms in order to facilitate the integration of the main body and Annexes B, D and E. Also certain additions have been found necessary in order to accommodate the application area of the different modules.

C+.4.1 Annex B DTX operation with Annex D

Integrating Annexes B and D functionality in order to provide DTX operation with Annex D is straightforward. The voice activity detection (VAD), silence insertion description (SID) coding and comfort noise generation (CNG) of Annex B are reused without any modifications. Care is taken to update the parameters for the phase dispersion for the postfilter in Annex D during discontinued transmission (see clause C+.5.2).

C+.4.2 Annex B DTX operation with Annex E

Integrating Annexes B and E functionality in order to provide DTX operation with Annex E is slightly more involved. Since the DTX operation of Annex B is based on the 10th order LPC analysis, the VAD function of Annex B is performed after the 10th order forward adaptive LPC analysis and before the backward adaptive LPC analysis of Annex E. In case the VAD function detects "non-speech", the LPC mode of Annex E is forced to forward adaptive LPC and the backward adaptive LPC analysis is skipped. Furthermore, it has been found necessary to add a correctional module after the VAD in order to detect music and accommodate the somewhat expanded application area of Annex E – one of the purposes of Annex E is to provide transmission capability of music with a certain quality. Accordingly, during the development of Annex E there were strict requirements to the performance with music signals. On the other hand, for the main body and Annexes B and D there were no strict requirements to the performance with music signals. In order to guarantee the quality with music signals of Annex E during Annex B DTX operation, the music detection function forces the VAD to "speech" during music segments, hereby ensuring that the music segments are coded with the 11.8 kbit/s of Annex E. The SID coding and the CNG of Annex B are reused without any modifications. Furthermore, care is taken to appropriately update the parameters of the LPC mode selection algorithm of Annex E during discontinued transmission (see clause C+.5.3).

C+.5 Algorithm description

This clause presents the algorithm description of the necessary additions to the algorithms of the individual annexes in order to facilitate the integration. All remaining modules originate from the main body, Annex B, D or E.

C+.5.1 Music detection

The music detection is a new function. It is performed immediately following the VAD and forces the VAD to "speech" during music segments. It is active only during Annex E operation, though its parameters are updated continuously independently of bit-rate mode during DTX operation of the integrated G.729.

The music detection algorithm corrects the decision from the voice activity detection (VAD) in the presence of music signals. It is used in conjunction with Annex E during Annex B DTX operation, i.e., in discontinuous transmission mode. The music detection is based on the following parameters:

– *Vad_deci*: VAD decision of the current frame.

– *PVad_dec*: VAD decision of the previous frame.

– *Lpc_mod*: Flag indicator of either forward or backward adaptive LPC of the previous frame.

– *Rc*: Reflection coefficients from LPC analysis.

– *Lag_buf*: Buffer of corrected open-loop pitch lags of last 5 frames.

– *Pgain_buf*: Buffer of closed-loop pitch gain of last 5 subframes.

– *Energy*: First autocorrelation coefficient $R(0)$ from LPC analysis.

– *LLenergy*: Normalized log energy from VAD module.

– *Frm_count*: Counter of the number of processed signal frames.

– *Rate*: Selection of speech coder.

The algorithm has two main parts:

1) Computation of relevant parameters.

2) Classification based on parameters.

C+.5.1.1 Computation of relevant parameters

This clause describes the computation of the parameters used by the decision module.

Partial normalized residual energy

$$Lenergy = 10\log_{10}\left(\prod_{i=1}^{4}\left(1 - Rc(i)^2\right)\frac{Energy}{240}\right)$$

Spectral difference and running mean of partial normalized residual energy of background noise

A spectral difference measure between the current frame reflection coefficients *Rc* and the running mean reflection coefficients of the background noise *mRc* is given by:

$$SD = \sum_{i=1}^{10}(Rc(i) - mRc(i))^2$$

The running means $\overline{mrc}$ and *mLenergy* are updated as follows using the VAD decision *Vad_deci* that was generated by the VAD module.

```
if $Vad\_deci == NOISE\{$
    $\overline{mrc} = 0.9\overline{mrc} + 0.1\overline{rc}$
    $mLenergy = 0.9mLenergy + 0.1Lenergy$
$\}$
```

Open-loop pitch lag correction for pitch lag buffer update

The open-loop pitch lag T_{op} is corrected to prevent pitch doubling or tripling as follows:

```
$avg\_lag = \sum_{i=1}^{4}\frac{Lag\_buf(i)}{4}$
if $\left(abs\left(\frac{T_{op}}{2} - avg\_lag\right) <= 2\right)$
        $Lag\_buf(5) = \frac{T_{op}}{2}$
else if $\left(abs\left(\frac{T_{op}}{3} - avg\_lag\right) <= 2\right)$
        $Lag\_buf(5) = \frac{T_{op}}{3}$
else
        $Lag\_buf(5) = T_{op}$
```

It should be noted that the open loop pitch lag T_{op} is not modified and is the same as derived by the open-loop analysis.

Pitch lag standard deviation

$$std = \sqrt{\frac{Var}{4}}$$

where:

$$Var = \sum_{i=1}^{5}(Lag\_buf(i) - \mu)^2 \text{and } \mu = \sum_{i=1}^{5}\frac{Lag\_buf(i)}{5}$$

Running mean of pitch gain

$$mPgain = 0.8mPgain + 0.2\theta, \text{ where } \theta = \sum_{i=1}^{5}\frac{Pgain\_buf(i)}{5}$$

The pitch gain buffer *Pgain_buf* is updated after the subframe processing with a pitch gain value of 0.5 if $Vad\_deci = NOISE$ and otherwise with the quantized pitch gain.

Pitch lag smoothness and voicing strength indicator

A pitch lag smoothness and voicing strength indicator *Pflag* is generated using the following logical steps:

First, two intermediary logical flags *Pflag*1 and *Pflag*2 are obtained as:

```
if ($std$ < 1.3 and $mPgain$ > 0.45) set $Pflag1$ = 1 else 0
if ($mPgain$ > $Thres$) set $Pflag2$ = 1 else 0,
where $Thres$ = 0.73 if $Rate$ = $G729D$, otherwise $Thres$ = 0.63
```

Finally, *Pflag* is determined from the following:

```
if (($PVad\_dec == VOICE$ and ($Pflag1 == 1$ or $Pflag2 == 1$)) or ($Pflag2 == 1$))
set $Pflag = 1$ else 0
```

Stationarity counters

A set of counters are defined and updated as follows:

a) *count_consc_rflag* tracks the number of consecutive frames where the 2nd reflection coefficient and the running mean of the pitch gain satisfy the following condition:

```
if ($Rc(2) < 0.45$ and $Rc(2) >$ and $mPgain < 0.5$)
    $count\_consc\_rflag = count\_consc\_rflag + 1$
else
    $count\_consc\_rflag = 0$
```

b) *count_music* tracks the number of frames where the previous frame uses backward adaptive LPC and the current frame is "speech" (according to the VAD) within a window of 64 frames.

```
if ($Lpc\_mod == 1$ and $Vad\_deci == VOICE$)
    $count\_music = count\_music + 1$
```

Every 64 frames, a running mean of *count_music*, *mcount_music* is updated and reset to zero as described below:

```
if (($Frm\_count$ mod 64) == 0){
    if ($Frm\_count == 64$)
        $mcount\_music = count\_music$
else
    $mcount\_music = 0.9\ mcount\_music + 0.1 count\_music$
}
```

c) *count_consc* tracks the number of consecutive frames where the *count_music* remains zero:

```
if ($count\_music == 0$)
    $count\_consc = count\_consc + 1$
else
    $count\_consc = 0$
    if ($count\_consc > 500$ or $count\_consc\_rflag > 150$) set $mcount\_music = 0$
```

count_music in b) is reset to zero every 64 frames after the update of the relevant counters. The logic in c) is used to reset the running mean of $count\_music$.

d) *count_pflag* tracks the number of frames where *Pflag* = 1, within a window of 64 frames.

```
if ($Pflag == 1$)
    $count\_pflag = count\_pflag + 1$
```

Every 64 frames, a running mean of $count\_pflag$, $mcount\_pflag$, is updated and reset to zero as described below:

```
if (($Frm\_count$ mod 64) == 0){
  if ($Frm\_count == 64$)
    $mcount\_pflag = count\_pflag$
else{
  if ($count\_pflag > 25$)
    $mcount\_pflag = 0.98 mcount\_pflag + 0.02 count\_pflag$
  else ($count\_pflag > 20$)
    $mcount\_pflag = 0.95 mcount\_pflag + 0.05 count\_pflag$
  else
    $mcount\_pflag = 0.9 mcount\_pflag + 0.1 count\_pflag$
  }
}
```

e) *count_consc_pflag* tracks the number of consecutive frames satisfying the following condition.

```
if ($count\_pflag == 0$)
  $count\_consc\_pflag = count\_consc\_pflag + 1$
else
  $count\_consc\_pflag = 0$
if ($count\_consc\_pflag >$ )100 or $count\_consc\_rflag > 150$) set $mcount\_pflag = 0$
```

count_pflag is reset to zero every 64 frames. The logic in e) is used to reset the running mean of *count_pflag*.

C+.5.1.2 Classification

Based on the estimation of the above parameters, the VAD decision *Vad_deci* from the VAD module is reverted if the following conditions are satisfied:

```
if ($Rate = G729E$){
  if ($SD > 0.15$ and ($Lenergy - mLenergy$) > 4 and $LLenergy > 50$)
    $Vad\_deci = VOICE$
else if (($SD > 0.38$ or ($Lenergy - mLenergy$) > 4) and $LLenergy > 50$)
    $Vad\_deci = VOICE$
else if (($mcount\_pflag >= 10$ or $mcount\_music >= 5$ or $Frm\_count < 64$)
        and $LLenergy > 7$)
    $Vad\_deci = VOICE$
}
```

Note that the music detection function is called all the time regardless of the operational coding mode in order to keep the memories current. However, the VAD decision *Vad_deci* is altered only if the integrated G.729 is operating at 11.8 kbit/s (Annex E). It should be noted that the music detection only has the capability to change the decision from "non-speech" to "speech" and not vice versa.

C+.5.2 Update of state variables specific to Annex D during discontinued transmission

The only state variables specific to Annex D are the state variables of the phase dispersion module (see clause D.6.2) at the decoder. In case of inactive frames, the same update procedure as in case of nominal bit rate (8 kbit/s) is followed using as adaptive and ACELP gain estimations the gain values computed by the comfort noise excitation generator (see clause B.4.4). Note also that the update for the higher rate is identical to the update for the nominal bit rate.

C+.5.3 Update of state variables specific to Annex E during discontinued transmission

C+.5.3.1 Update of encoder state variables specific to Annex E

At the encoder in case of inactive frames, the update of state variables is identical to the update performed in Annex E in case of switch to the nominal bit rate 8 kbit/s. The update procedure is the following: the LP mode is set to 0, the global stationarity indicator is decreased and the high stationarity indicator is reset to 0 (see clause E.3.2.7.2), the interpolation factor used to smoothly switch from LP forward filter to backward LP filter is reset to its maximum value (see clause E.3.2.7.1). Note that this update is also performed in case of switch to the lower bit rate 6.4 kbit/s.

C+.5.3.2 Update of decoder state variables specific to Annex E during discontinued transmission

At the decoder in case of inactive frames, the update of state variables is almost identical to the update performed in Annex E in case of switch to the forward mode only rates (8 kbit/s and 6.4 kbit/s) except that the pitch delay stationary indicator is reset to 0 instead of being computed by the pitch tracking procedure (see clause E.4.4.5).

C+.6 Description of C source code

The Annex C+ integrating the G.729 main body, Annexes B, D and E functionality is simulated in floating point arithmetic ANSI-C code. As for Annex C, the typedef.h file contains a statement enabling the definition of all floating-point variables and constants as type either double or single. The ANSI-C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of G.729 and in Annexes B, C, D, E and C+. As of the approval of this text, the current version of this ANSI C code is Version 2.2 of October 2006. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

C+.6.1 Use of the simulation software

The C code consists of two main programs **codercp.c** and **decodercp.c**, which simulate encoder and decoder, respectively. The encoder is executed as follows:

codercp inputfile bitstreamfile dtx_option rate_option

The decoder is executed as follows:

decodercp bitstreamfile outputfile

The input file and output file are 8 kHz sampled data files containing 16-bit PCM signals. The bit stream file is a binary file containing the bit stream; the mapping table of the encoded bit stream is contained in the simulation software. The two options for the encoder are: dtx_option and rate_option where:

dtx_option =	1: DTX enabled 0: DTX disabled, the default is 0 (DTX disabled).
rate_option =	0 to select the lower rate (6.4 kbit/s); = 1 to select the main G.729 (8 kbit/s); = 2 to select the higher rate (11.8 kbit/s) or a file_rate_name: a binary file of 16-bit word containing either 0, 1, 2 to select the rate on a frame-by-frame basis; the default is 1 (8 kbit/s).

C+.6.2 Organization of the simulation software

Table C+.1 gives the list of the software files names with a brief description, and it is also indicated what annex the file has been derived from (identical or similar to Annex C file or fixed to floating-point transcription of the files). Note that the fixed-point files basic_op.c, oper_32b.c, dspfunc.c, basic_op.h and oper_32b.h are not needed for floating-point arithmetic. As for Annex C, a float to short conversion routine has been added to the file utilities file utilcp.c.

Table C+.1 – List of software files of integrated G.729 in floating point

File name	Description	Link
Gainpred.c	Gain predictor	C
Lpfunccp.c	Miscellaneous routines related to LP filter	C + E
Cor_func.c	Miscellaneous routines related to excitation computation	C
Pre_proc.c	Preprocessing (HP filtering and scaling)	C
P_parity.c	Compute pitch parity	C
Pwf.c	Computation of perceptual weighting coefficients (8 kbit/s)	C
Pred_lt3.c	Generation of adaptive codebook	C
Post_pro.c	Post-processing (HP filtering and scaling)	C
Tab_ld8k.c	ROM tables	C
Ld8k.h	Function prototypes	C
Tab_ld8k.h	Extern ROM table declarations	C
Typedef.h	Data type definition (machine-dependent)	C
Taming.c	Pitch instability control	C
Qsidgain.c	SID gain quantization	B
QsidLSF.c	SID-LSF quantization	B
Tab_dtx.c	ROM tables	B
Sid.h	Prototype and constants	B

Table C+.1 – List of software files of integrated G.729 in floating point

File name	Description	Link
Octet.h	Octet transmission mode definition	B
Tab_dtx.h	Extern ROM table declarations	B
Pwfe.c	Computation of perceptual weighting coefficients (11.8 kbit/s)	E
Vad.c	VAD	B
Dtx.c	DTX decision	B
Vad.h	Prototype and constants	B
Dtx.h	Prototype and constants	B
Calcexc.c	CNG excitation calculation	B
Dec_sid.c	Decode SID information	B
Utilcp.c	Utility functions	C + B
Phdisp.c	Phase dispersion	D
Bwfw.c	Backward/forward switch selection	E
Bwfwfunc.c	Miscellaneous routines related to backward/forward switch	E
Filtere.c	Filter functions	C + E
Lpccp.c	LP analysis	C + E
Lspcdece.c	LSP decoding routines	C + E
Lspgetqe.c	LSP quantizer	C + E
Qua_lspe.c	LSP quantizer	C + E
Track_pi.c	Pitch tracking	E
Codercp.c	Main encoder routine	C + B + D + E
Codld8cp.c	Encoder routine	C + B + D + E
Decodcp.c	Main decoder routine	C + B + D + E
Decld8cp.c	Decoder routine	C + B + D + E
Acelp_cp.c	search ACELP fixed codebook (6.4, 8, 11.8 kbit/s)	C + D + E
Dacelpcp.c	Decode algebraic codebook (6.4, 8, 11.8 kbit/s)	C + D + E
Pitchcp.c	Pitch search	C + D + E
Declagcp.c	Decode adaptive-codebook index	C + D + E
Q_gaincp.c	Gain quantizer	C + D + E
Degaincp.c	Decode gain	C + D + E
Pstpcp.c	Postfilter routines	C + B + E
Bitscp.c	Bit manipulation routines	C + B + D + E
Tabld8cp.c	ROM tables for G.729 at 6.4 and 11.8 kbit/s	D + E
Tabld8cp.h	Extern ROM declarations for G.729 at 6.4 and 11.8 kbit/s	D + E
Ld8cp.h	Constant and function prototypes for G.729 at 6.4 and 11.8 kbit/s	D + E
Mus_dtct.c	Music detection module	New

Annex D

+CS-ACELP speech coding algorithm at 6.4 kbit/s

(This annex forms an integral part of this Recommendation)

Summary

This annex provides the lower bit-rate extension designed to achieve a quality somewhat below the one achieved with the full version of G.729.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 6.4 kbit/s and 8 kbit/s.

D.1 Scope

This annex is intended as a lower rate extension to the algorithm in the full version of G.729, and is specified to increase the flexibility of the algorithm in the full version of G.729, e.g., to handle overload conditions. It does not provide the same level of quality as does the algorithm in the main body of G.729, but for most conditions it provides significantly higher quality than G.726 at 24 kbit/s. However, for high levels of car noise, the algorithm could have some performance limitations. The differences to the main body of G.729 are described in this annex.

D.2 Normative references

This annex refers to materials defined in the main body of this Recommendation.

D.3 General coder description for the 6.4 kbit/s extension

The coder is similar to that of the full version of G.729 with a few exceptions. The modifications are summarized below, and described in more detail in the following clauses.

1) The ACELP codebook of G.729 has been changed to a new ACELP codebook which is using two signed pulses in two overlapping tracks of different lengths (16 and 32 positions respectively).
2) The conjugate-structured codebook for the gains has been replaced with a new conjugate-structured codebook with 6 bits.
3) A modified coding of the pitch delay in the second subframe is used. The number of bits are reduced to 4 bits. The delta lag range is maintained, using an uneven distribution of fractional delta values.
4) An additional postfiltering technique is applied to reduce the effects of the sparser algebraic codebook.
5) The pitch-delay parity bit has been removed.

The new coder uses 6.4 kbit/s or 64 bits per frame instead of the 8.0 kbit/s or 80 bits per frame used in the full version of G.729.

D.4 Bit allocation

Table D.1 – Bit allocation for 6.4 kbit/s of G.729

Parameter	Number of bits per frame (10 ms)
LP parameters	18
Adaptive codebook	8 + **4**
Fixed codebook	2 * **11**
Gain quantizer	2 * **6**
Total	64
NOTE – Bold figures represent changes compared to those in the full version of G.729.	

D.5 Functional description of the encoder

D.5.1 Preprocessing

Same as that in the full version of G.729.

D.5.2 Linear prediction analysis and quantization

Same as that in the full version of G.729.

D.5.3 Perceptual weighting

Same as that in the full version of G.729.

D.5.4 Open-loop pitch analysis

Same as that in the full version of G.729.

D.5.5 Computation of the impulse response

Same as that in the full version of G.729.

D.5.6 Computation of the target signal

Same as that in the full version of G.729.

D.5.7 Adaptive codebook search

The LTP coding for the absolute coded subframes (first subframe) are the same as that in the main body of G.729. The number of LTP lags in the second subframe has been reduced from 32 to 16. Integer delta lag values are used for the ranges *int*(T1) – 5 to *int*(T1) – 2 and *int*(T1) + 1 to *int*(T1) + 4, where T1 is the LTP lag of the previous subframe. Fractional lags with a resolution of 1/3 are used in the range *int*(T1) – 1 2/3 to *int*(T1) + 2/3.

D.5.8 Fixed codebook structure and search

The original four-pulse codebook is exchanged for an ACELP codebook with 2 signed pulses in two overlapping tracks. The track table is given in Table D.2. The signs of the pulses are preset as in the main body of G.729. The search of pulse positions is an exhaustive yet computationally efficient search over all 512 vectors.

Table D.2 – ACELP track table

Pulse	Sign	Positions
i_0	+1/–1	1, 3, 6, 8, 11, 13, 16, 18, 21, 23, 26, 28, 31, 33, 36, 38
i_1	+1/–1	0, 1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 19, 20, 21, 22, 24, 25, 26, 27, 29, 30, 31, 32, 34, 35, 36, 37, 39

D.5.9 Quantization of the gains

The conjugate-structured gain codebook is redesigned. Six bits per subframe are allocated to the gain codebook. The codebook is trained with the condition of 0.1% bit error rate with a random distribution. This codebook requires 32 words of memory.

D.5.10 Memory update

Same as that in the full version of G.729.

D.6 Functional description of decoder

D.6.1 Parameter decoding procedure

Similar to that in the full version of G.729. The number of parameters is smaller. Less fixed excitation codebook parameters are used.

D.6.2 Fixed codebook post-processing

An additional post-processing filter is applied in the decoder in order to reduce the perceptually adverse effects of the sparse excitation. The filter alters the innovation signal such that a new innovation is created which has the energy more spread over the subframe. The filter alters mainly the phase of the innovation through a "semi-random" impulse response. The filtering is performed by circular convolution, using one of the three stored impulse responses. The filter selection is controlled by a voicing decision, based on the filtered received LTP gain. The three impulse responses correspond to different amounts of spreading. Maximum spreading is applied in noise-like segments, when the filtered LTP gain is low. Medium spreading is applied for intermediate LTP gains, and no spreading is applied in voiced speech, when the filtered LTP gain is high. Additionally, strong increase in codebook gain is detected, to avoid spreading of onsets.

D.6.3 Post filtering and post-processing

Same as that in the full version of G.729.

D.6.4 Concealment of frame-erasures

Same as that in the full version of G.729.

D.7 ANSI C code

ANSI C code specifying the G.729 lower bit rate extension is available as an attachment to this annex. As of the approval of this text, the current version of this ANSI C code is Version 1.3 of February 2000. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website. The ANSI C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of G.729 and in Annex D.

Tables D.3 to D.6 contain lists of the ANSI C code files grouped by function.

Table D.3 – List of software files specific to G.729 lower bit-rate extension encoder

File name	Description
acelpcod.c	Search fixed codebook
codld8kd.c	Encoder routine
coderd.c	Encoder
pitchd.c	Pitch search
qua_g6k.c	Gain quantizer

Table D.4 – List of software files specific to G.729 lower bit-rate extension decoder

File name	Description
declag3d.c	Decode adaptive-codebook index
decld8kd.c	Decoder routine
decoderd.c	Decoder
deacelpd.c	Decode algebraic codebook
dec_g6k.c	Decode gain

Table D.5 – List of software files specific to G.729 lower bit-rate extension routines common to encoder and decoder

File name	Description
bitsd.c	Bit manipulation routines
filterd.c	Filter functions
ld8kd.h	Switching variables
tabld8kd.c	Tables
tabld8kd.h	

Table D.6 – List of software files specific to G.729 lower bit-rate extension routines common to G.729 and Annex D

File name	Description
basic_op.h	*Common to G.729 main body*
ld8k.h	
oper_32b.h	
tab_ld8k.h	
typedef.h	
basic_op.c	
de_acelp.c	
dec_gain.c	
dspfunc.c	
gainpred.c	
lpc.c	
lpcfunc.c	
lspdec.c	
lspgetq.c	
oper_32b.c	
p_parity.c	
post_pro.c	
pre_proc.c	
pred_lt3.c	
pst.c	
pwf.c	
qua_gain.c	
qua_lsp.c	
tab_ld8k.c	
util.c	

Annex E

CS-ACELP speech coding algorithm at 11.8 kbit/s

(This annex forms an integral part of this Recommendation)

Summary

This annex provides the high level description of the higher bit-rate extensions of this Recommendation designed to accommodate a wide range of input signals, such as speech, with background noise and even music.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 8 kbit/s and 11.8 kbit/s.

E.1 Introduction

This annex provides the high-level description of the higher bit-rate extension of G.729 designed to accommodate a wide range of input signals, such as speech, with background noise and even music.

E.2 General description of the speech codec

The extension algorithm has been designed to limit as much as possible the modifications and additions brought to the original G.729 algorithm. The only actual additions to G.729 concern the LP part with the introduction of a backward LP analysis suited for music signals and stationary background noises and the design of two new algebraic excitation codebooks to extend the bit rate up to 11.8 kbit/s: one codebook is used in forward mode, the other one, larger, in backward mode. All the remaining procedures are strictly the same as in G.729 except some minor modifications to the postfiltering and perceptual weighting procedures. Error concealment has also been modified to be adapted to the backward/forward LP structure.

Two LP analyses are performed at the frame rate: one backward on the synthesis signal and one forward on the input signal. An adaptive decision procedure chooses the best filter and performs the switch if needed. The LP forward part of the algorithm is the same as the G.729 one with the same LSP quantization scheme. The backward LP analysis has an order of 30 and is performed both in the coder and in the decoder. Since the LP coefficients are not transmitted, the spare bit rate is used to increase the size of the algebraic excitation codebooks. One information bit is needed to indicate the LP mode and is protected by a parity bit. In the proposed extension, all the additional bit rate from 8 kbit/s to 11.8 kbit/s, except two bits (LP indication mode + parity bit), is used to increase the size of the algebraic codebooks. The bit allocation of the coder parameters is shown in Table E.1.

The backward/forward decision criterion enables to operate a real discrimination between speech (mainly coded in forward mode) and music (mainly coded in backward mode). The backward/forward procedure has also been designed to reduce the number of switches and to perform, when necessary, smooth switching between filters with no artefacts. The LP mode and the related information is used to better adapt postfiltering and perceptual weighting to either music or speech. This is also used for the error concealment.

In the following clauses, a high-level description of the 11.8 kbit/s extension of G.729 is provided. Only the modifications or additions to the G.729 algorithm will be described.

Table E.1 – Bit allocation of the 11.8 kbit/s CS-ACELP algorithm (10 ms frame)

	Extension at 11.8 kbit/s	
LP mode indication bit	**1 + 1 (parity)**	
	Forward	Backward
LP filter	18	0
LTP delay (1st/2nd sub-fr.)	8 + 1 (parity)/5	8 + 1 (parity)/5
EXC codes (1st/2nd sub-fr.) Gains (LTP + EXC) (1st/2nd sub-fr.)	**35/35** 7/7	**44/44** 7/7
Total	118	118
NOTE – The numbers of bits corresponding to modified parts of the structure (compared to G.729) are typed in bold.		

E.2.1 Encoder

In order to obtain this high quality with music while keeping a good robustness to transmission errors and avoiding degradation of less stationary signals and especially speech (compared with a pure forward structure used in G.729), a new technique called mixed backward/forward LP structure has been introduced. A criterion enables to choose the most suitable LP analysis given the stationarity of the input signal and the backward and forward filters' prediction gains.

For music signals, generally very stationary, the LP backward mode is mainly used: the LP analysis is performed on the synthesis signal with no transmission of the coefficients, with two benefits:

- The LP order is increased up to 30 coefficients which is far more suited for the complex spectrum of music signals (the 10 coefficients LP filter of LP forward codecs like G.729 is not sufficient for music).
- The bit rate is better allocated: no bit rate is wasted on successive very similar LP filters. All the spare bit rates are used to extend the size of the excitation codebook. An algebraic codebook with 44 bits is used for the fixed codebook excitation.

The weak points of pure backward LP analysis mainly concern the non-stationary signals with sharp spectrum transitions and the sensitivity to transmission errors. With the mixed LP backward/forward structure, if a spectrum transition occurs, the forward mode is selected and the 10 LP coefficients are coded and transmitted. Besides, even if backward mode is dominant, the transmission of forward LP filters clearly improves the robustness when compared with a pure backward structure.

In forward mode, the encoder is almost identical to G.729 with more bits allocated to the excitation codebooks. An algebraic codebook with 35 bits is used for the fixed codebook excitation.

E.2.2 Decoder

First, the parameter's indices are extracted from the received bit stream. These indices are decoded to obtain the coder parameters corresponding to a 10 ms speech frame. The first parameter decoded is the LP mode information and its parity bit. According to this information, the frame is classified either as forward, backward or erased. In forward mode, the parameters are the LSP coefficients, the two fractional pitch delays, the two forward fixed-codebook vectors, and the two sets of adaptive- and fixed-codebook gains. In backward mode, the parameters are the two fractional pitch delays, the two backward fixed-codebook vectors, and the two sets of adaptive- and fixed-codebook gains. First, the LP backward analysis is performed. Then, if the frame is in forward mode, the LSP coefficients are interpolated and converted to LP filter coefficients for each subframe. Except for the construction of fixed-codebook excitation, the decoding procedure is very similar to the G.729 decoding procedure.

Then, for each 5 ms subframe the following steps are done:

– the excitation is constructed by adding the adaptive- and fixed-codebook vectors scaled by their respective gains;
– the speech is reconstructed by filtering the excitation through the LP synthesis filter (either forward or backward); and
– the reconstructed speech signal is passed through a post-processing stage, which includes an adaptive postfilter based on the long-term and short-term synthesis filters, followed by a high-pass filter and scaling operation. Compared with G.729, the weighting factors of the postfilter have been made adaptive.

E.2.3 Delay

The same as clause 2.3.

E.2.4 Speech coder description

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

E.3 Functional description of the encoder

In this clause, the different functions of the encoder are described. The main body of this Recommendation is referred to in most of this clause, except the parts where algorithmic modifications or additions have been carried out.

E.3.1 Preprocessing

The same as clause 3.1.

E.3.2 Linear prediction analysis and quantization

Two LP analyses are performed simultaneously at the 10 ms frame rate: one forward analysis on the input signal which is strictly the same as G.729 with also the same quantization scheme, and one backward analysis performed on the past synthesized signal.

E.3.2.1 Windowing and autocorrelation computation

– *Forward LP analysis*

The same as clause 3.2.1.

– *Backward LP analysis*

A hybrid recursive windowing scheme the same as in G.728 is used.

Let sample 1 be the more recent sample of the more recent synthesized frame, and let the indices i represent past samples ordered so that oldest samples have highest indices. Samples i = 1 to 35 are windowed with the non-recursive part of the window:

$$w_{lpbwd}(i) = sin(i \times c_{lpbwd}), \quad i = 1,...,35 \text{ where } c_{lpbwd} = 0.047783$$

The recursive part of the window is given by the function (samples > 35):

$$w_{lpbwd}(i) = b_{lpbwd} \times a_{lpbwd}^{(i-36)}, \quad i > 35$$

with $a_{lpbwd} = 0.9928337491$ and $b_{lpbwd} = sin(36 \times c_{lpbwd})$

The recursive calculation of the autocorrelation coefficients is performed as described in [ITU-T G.728].

The same white noise correction factor as for forward LP analysis is applied to the first autocorrelation coefficient (1.0001), but the bandwidth expansion applied to the coefficients is reduced from 60 Hz (in G.729) to 5 Hz. A small additional spectral flattening is applied by a weighting function with $\gamma_{lpbwd} = 0.98$ on the LP coefficients (calculated in clause E.3.2.2).

E.3.2.2 Levinson-Durbin algorithm

The algorithm used is the same for forward and backward analysis. Compared to the G.729 algorithm, the size of some arrays has been extended to cope with the higher LP order.

E.3.2.3 LP to LSP conversion

For forward LP filter, the same as clause 3.2.3. For backward LP filter, no LSP calculation is needed.

E.3.2.4 Quantization of LSP coefficients

The same as clause 3.2.4 for forward LSP coefficients. For backward LP filter, no LSP quantization is needed.

E.3.2.5 Interpolation of LP coefficients

- *For the forward LP analysis*

  As in clause 3.2.5, the quantized (and unquantized) LP coefficients are used for the second subframe. For the first subframe, the forward quantized (and unquantized) LSP coefficients are interpolated as in clause 3.2.5 when the previous frame is in forward mode. When the previous frame is in backward mode, no interpolation is performed, the second subframe quantized (and unquantized) LP filter is also used for the first subframe.

- *For the backward LP analysis*

  For the second subframe, either the current backward LP filter A_{bwd} computed in clause E.3.2.2 or a transition filter, as will be described in clause E.3.2.7.1, is used.

  For the first subframe, the LP filter coefficients are directly interpolated with the same interpolation factors (0.5, 0.5) as G.729 between the second subframe backward LP filter and the previous frame filter.

E.3.2.6 LSP to LP conversion

For forward LP filter, the same as clause 3.2.6. For backward LP filter, no conversion is needed.

E.3.2.7 Backward/forward decision and switch procedure

E.3.2.7.1 Switching procedure

This clause describes how the switch is performed from a previous frame using a forward (respectively backward) filter to the current one where the backward (respectively forward) filter is chosen in order to avoid artefacts in the synthesized signal.

– *From forward LP filter to backward LP filter*

  This generally occurs when the signal is stationary. It is consequently important to avoid any filter transition which would bring an audible artificial spectrum transition in the synthesized signal. To achieve this, the following interpolation is performed both at the encoder and at the decoder:

If switch is decided at frame n:

Let $A_{fwd}(n-1)$ be the forward LP filter at frame $n-1$.

Let $A_{bwd}(n)$ be the backward LP filter at current frame n computed in clause E.3.2.2.

The LP filter A used at frame $n+i$ is given by:

$A(n+i) = 0.1 \times i \times A_{bwd}(n+i) + (1.-0.1 \times i) \times A(n+i-1), \qquad 0 \le i \le 9$

$A(n+i) = A_{bwd}(n+i) \qquad i \ge 10$

with $A(n-1) = A_{fwd}(n-1)$

After 10 transition frames, the filter used is exactly the backward filter.

– *From backward filter to forward filter*

This occurs when a spectrum transition exists in the input signal. No smoothing is then performed:

If switch is decided at frame n: $A(n) = A_{fwd}(n)$

E.3.2.7.2 The global stationarity indicator and high stationarity indicator

The global stationarity indicator at frame n (called Stat(n)) characterizes the global stationarity of the input signal. Calculated at frame n after the backward/forward decision has been taken, it will be used for the next frame ($n+1$) backward/forward decision calculated frame-by-frame to reduce the number of switches between filters. The principle is to progressively favour one mode according to the stationarity of the input signal and to reduce the number of switches to the other mode.

The computation of this indicator is based on the history of the backward/forward decisions and on the backward and forward filters prediction gains. It varies from a value representing a high stationarity of the input signal (value 32 000) to a value representing a low stationarity (value 0).

This indicator has slow frame-by-frame variations (with the given numerical values, it takes at least 80 frames to vary from min. to max.).

The adaptation depicted below is only performed for frames the energy of which is greater than 40 dB. For other frames that are considered as silence frames, Stat(n) is equal to Stat($n-1$) bounded by 13 000.

– The first step of the adaptation is based on the preceding switch decisions:

Let n be the index of the current frame.

Let $N_{bwd}(n)$ be the number of consecutive backward frames measured at frame n.

If frame n is a forward frame, then $N_{bwd}(n)$ is equal to 0.

Let the value $\text{Stat}^1(n)$ represent the output of the first step stationarity evaluation.

If frame n is a backward to forward transition frame (i.e., frame $n-1$ is backward and n is forward) and if less than 20 consecutive backward frames have occurred:

$\text{Stat}^1(n) = \text{Stat}(n-1) - (5000 - 250 \times N_{bwd}(n-1))$

else:

if ($N_{bwd}(n) > 20$) $\text{Stat}^1(n) = \text{Stat}(n-1) + 500$

else:

if ($N_{bwd}(n) = 20$) $\text{Stat}^1(n) = \text{Stat}(n-1) + 2500$

else: $\text{Stat}^1(n) = \text{Stat}(n-1)$

– The second step of the adaptation is based on the prediction gains:

Let x be the difference between the backward LP filter prediction gain and the forward LP filter prediction gain: $x = \text{Gpred}_b - \text{Gpred}_f$ (in dB).

Stat(n) = Stat^1(n) + Δ(x) with:

If Stat^1(n) < 13 000,

$$\Delta(x)=\begin{cases}3200 \text{ if } x>4\\ 2400 \text{ if } x\in\left]3,4\right]\\ 1600 \text{ if } x\in\left]2,3\right]\\ 800 \text{ if } x\in\left]1,2\right]\\ 400 \text{ if } x\in\left]0,1\right]\\ 0 \text{ if } x\in\left[-1,0\right]\\ -400 \text{ if } x\in\left[-2,-1\right[\\ -800 \text{ if } x\in\left[-3,-2\right[\\ -1600 \text{ if } x\in\left[-4,-3\right[\\ -3200 \text{ if } x\in\left[-4.7,-4\right[\\ -6400 \text{ if } x<-4.7\end{cases}$$

else:

$$\Delta(x)=\begin{cases}0 \text{ if } x\geq -1\\ -400 \text{ if } x\in\left[-2,-1\right[\\ -800 \text{ if } x\in\left[-3,-2\right[\\ -1600 \text{ if } x\in\left[-4,-3\right[\\ -3200 \text{ if } x\in\left[-4.7,-4\right[\\ -6400 \text{ if } x<-4.7\end{cases}$$

A high stationarity state is also determined with the parameter value High_Stat set to 1. This high stationarity state is detected when the percentage of backward frames becomes significantly higher than the percentage of forward frames:

Let N_{bwd} (respectively N_{fwd}) represent the number of backward (respectively forward) frames in the previous $N_{f/b}$ frames ($N_{f/b} = N_{bwd} + N_{fwd}$). For the first 100 frames, N_{bwd} (respectively N_{fwd}) is the actual number of backward (respectively forward) frames in the previous $N_{f/b}$ frames. Then whenever $N_{f/b}$ reaches the value 100, $N_{f/b}$, N_{bwd}, N_{fwd} are divided by 2.

If $N_{f/b} < 10$, High_Stat = 0

else:

If $N_{bwd} > 4 \times N_{\mathrm{fwd}}$ then High_Stat = 1

else: High_Stat = 0

This procedure is only performed for frames the energy of which is greater than 40 dB. For silence frames, $N_{f/b}$, N_{bwd}, N_{fwd} and High_Stat are not updated.

E.3.2.7.3 Backward/forward decision procedure

At current frame n, the backward/forward decision is taken according to 4 criteria which apply sequentially.

– *1st criterion on prediction gains*

The prediction gains (in dB) of the backward, backward interpolated (different from backward only during forward-to-backward transitions) and forward LP filters are computed (called respectively Gpred_b and Gpred_{int} and Gpred_f).

Let Gap be an adaptive decision threshold (in dB).

The first stage decision is:

The backward LP filter is selected if the following condition is verified:

$(Gpred_{int} > Gpred_f - Gap)$ and $(Gpred_b > Gpred_f - Gap)$ and $(Gpred_b > 0)$ and $(Gpred_{int} > 0)$

Otherwise, the forward LP filter is selected.

The Gap parameter is adapted according to the stationarity indicator:

$Gap(n) = 0.0366 \times (Stat(n-1)/320) + 1.0$ with $Stat(n-1) \in [0, 32\ 000]$ (see clause E.3.2.7.2).

– *2nd criterion using the global stationarity indicator*

While the value of the global stationarity indicator $Stat(n-1) \in [0, 32\ 000]$ remains below 13 000, the forward LP mode is selected (second stage decision). This avoids unnecessary switches to backward mode with speech or other signals of low or medium stationarity.

– *3rd criterion on LSP*

In order to avoid any artificial transition when the short term spectrum is stationary, the following Euclidean distance is computed between the LSP vectors of two successive forward LP filters:

$\mathbf{LSP_n}$ is the LSP vector of the forward LP filter calculated at current frame n.

$\mathbf{LSP_{n-1}}$ is the LSP vector of the forward LP filter calculated at frame n – 1.

$d_{LSP}(n) = \|\mathbf{LSP_n}, \mathbf{LSP_{n-1}}\|^2$ is the Euclidean distance between both vectors.

If $d_{LSP}(n) < Thresh_{LSP}(n)$, if the previous frame is in backward mode and if the prediction gains $Gpred_b$ and $Gpred_{int}$ are positive, switching from backward to forward is forbidden (selection of backward mode as a third stage decision in this case).

$Thresh_{LSP}$ is adapted at each frame according to the value of $Stat(n-1)$:

If $Stat(n-1) = 32\ 000$ (max. value), $Thresh_{LSP}(n) = 0.03$

Else $Thresh_{LSP}(n) = 0$

– *4th criterion on the energy*

In order to increase the robustness of the algorithm to transmission errors, the forward LP filter is imposed for frames with energy below 40 dB.

E.3.3 Perceptual weighting

The perceptual weighting filter is given by: $W(z) = \dfrac{A(z/\gamma_1)}{A(z/\gamma_2)}$

– *In forward mode*

The parameters γ_1 and γ_2 are computed as in clause 3.3. The LP filter $A(z)$ is the unquantized forward filter $A_{fwd}(z)$ when High_Stat is equal to 0, and the quantized forward filter otherwise.

– *In backward mode*

If no High_Stat state is detected, the LP filter $A(z)$ used is the unquantized forward filter, else it is the backward calculated filter.

The parameters γ_1 and γ_2 take fixed values depending on the high stationarity indicator (High_Stat).

In case of high stationarity of the input signal (High_Stat = 1), the noise masking effect is reinforced:

$\gamma_{1bwdh} = 0.98$

$\gamma_{2bwdh} = 0.4$

In case of normal stationarity (High_Stat = 0):

$\gamma_{1bwdl} = 0.9$

$\gamma_{2bwdl} = 0.4$

– The weighted speech is calculated as indicated in equation (33) of clause 3.3, the filtering order depending on the selected weighting filter chosen (10 or 30).

E.3.4 Open-loop pitch analysis

The same as clause 3.4.

E.3.5 Computation of the impulse response

Similar to clause 3.5. (The order of the LP filters could be 30 instead of 10.)

E.3.6 Computation of the target signals

Similar to clause 3.6. (The order of the LP filters could be 30 instead of 10.)

E.3.7 Adaptive-codebook search

The adaptive-codebook search, the generation of the adaptive-codebook vector, the codeword computation for the delay index $P1$ and $P2$ and the computation of the adaptive-codebook gain are identical to the procedure described in clause 3.7. The parity bit $P0$ is computed on the seven (instead of six in G.279) most significant bits of the delay index $P1$ of the first subframe.

E.3.8 Fixed-codebook structure and search

E.3.8.1 Fixed-codebook in forward LP mode

In the forward LP mode, an algebraic codebook with 35 bits is used as the fixed codebook. In this codebook, each excitation vector contains 10 non-zero pulses. The pulse amplitudes are either −1 or +1. The 40 positions in each subframe are divided into 5 tracks where each track contains two pulses. In the design, the two pulses for each track may overlap resulting in a single pulse with amplitude +2 or −2. The allowed positions for pulses are shown in Table E.2.

Table E.2 – Structure of fixed codebook in forward mode C_{fwd}

Track	Pulses	Signs	Positions
1	$p0, p1$	s_0, s_1: ± 1	0, 5, 10, 15, 20, 25, 30, 35
2	$p2, p3$	s_2, s_3: ± 1	1, 6, 11, 16, 21, 26, 31, 36
3	$p4, p5$	s_4, s_5: ± 1	2, 7, 12, 17, 22, 27, 32, 37
4	$p6, p7$	s_6, s_7: ± 1	3, 8, 13, 18, 23, 28, 33, 38
5	$p8, p9$	s_8, s_9: ± 1	4, 9, 14, 19, 24, 29, 34, 39

Similar to G.729, the selected codebook vector is filtered through the pre-filter:

$$P(z) = 1/\left(1 - \beta z^{-T}\right)$$

to enhance the harmonic components. The way β is adapted is the same as in the main body of G.729.

E.3.8.1.1 Search procedure of the 35-bit codebook

The fixed codebook is searched by minimizing the mean-squared error between the weighted input speech and the weighted reconstructed speech. If $c_k(n)$ is the algebraic codevector at index k, $h(n)$ is the impulse response of the weighted synthesis filter, and $d(n)$ is the correlation between the target vector and $h(n)$, then the algebraic codebook is searched by maximizing the criterion:

$$T_k = \frac{(C_k)^2}{E_k}$$

where C is the correlation between $c_k(n)$ and $d(n)$ and E is the energy of the filtered codevector $(c_k(n) \times h(n))$. Since the algebraic codevector contains few non-zero pulses, the correlation can be written as:

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

where m_i is the position of the ith pulse, s_i is its amplitude, and N_p is the number of pulses ($N_p = 10$), and the energy in the denominator is given by:

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

where $\phi(i, j)$ contains the correlations between $h(n-i)$ and $h(n-j)$. The signal $d(n)$ and the correlations $\phi(i, j)$ are computed before the codebook search.

Similar to G.729, in order to speed up the search procedure, the pulse amplitudes are preset outside the closed-loop search using the so-called *signal-selected pulse amplitude* approach. In this approach, the most likely amplitude of a pulse occurring at a certain position is estimated using a certain side information signal. In G.729, the signal $d(n)$ is used for preselecting the pulse amplitudes. In this bit rate extension, a signal $b(n)$, which is a weighted sum of the normalized $d(n)$ vector and the normalized long-term prediction residual, is used.

The signal $b(n)$ is given by:

$$b(n) = d(n)/\sigma_d + \mathrm{e(n)}/\ \sigma_e$$

where $e(n)$ is the long-term prediction residual and σ_d and σ_e are the r.m.s. values of $d(n)$ and $e(n)$, respectively. The sign of a pulse at a certain position is set a priori equal to the sign of $b(n)$ at that position. The sign information is incorporated into the signals $d(n)$ and $\phi(i, j)$ before starting the search for the best pulse positions, similar to G.729.

The optimal pulse positions are determined using a non-exhaustive analysis-by-synthesis search procedure. The procedure used is a special case of a general depth-first tree search method which is efficient for searching huge codebooks with a reasonable complexity. In this approach, the N_p excitation pulses are partitioned into M subsets of N_m pulses. The search begins with subset 1 and proceeds with subsequent subsets according to a tree structure whereby subset m is searched at the mth level of the tree. The search is repeated by changing the order in which the pulses are assigned to the position tracks. In this particular codebook structure, the pulses are partitioned into 5 subsets of 2 pulses (the tree has 5 levels).

The pulse positions are determined as follows:

For each of the five tracks, the pulse positions with maximum absolute values of $d(n)$ are found. From these, the two successive tracks, T_{k_0} and $T_{(k_0+1) mod\, 5}$ with the largest combined maxima are determined. This index k_0 is used for the initial assignment of pulses to tracks. Then the two successive tracks, T_{k_1} and $T_{(k_1+1) mod\, 5}$ with the second largest combined maxima and the two successive tracks, T_{k_2} and $T_{(k_2+1) mod\, 5}$ with the third largest combined maxima are also determined.

In the first iteration, the pulses are assigned to the tracks as follows: the pulses i_n, $n = 0,...,9$, are assigned to tracks $T_{(k_0+n) mod\, 5}$, $n = 0,...,9$, respectively.

The pulses are searched in subsets of two pulses. We start by setting pulse i_0 to the maximum of track T_{k_0} and pulse i_1 to the maximum of track $T_{(k_0+1) mod\, 5}$. We then proceed by searching the pulse pair (i_2, i_3) by testing all the 8×8 possible position combinations in tracks $T_{(k_0+2) mod\, 5}$ and $T_{(k_0+3) mod\, 5}$ (given pulses i_0 and i_1 are known). The same procedure is repeated for the rest of the pulse pairs(i_4, i_5), (i_6, i_7) and (i_8, i_9) by testing the 8×8 possible position combinations in their respective tracks. At each level of the tree, the test criterion is computed based only on the available pulses at that level. This results in a total of $4 \times 8 \times 8$ positions tested (since the first pulse pairs are set to their track maxima).

Another two iterations are carried out by changing pulse assignment to tracks (replacing k_0 by k_1 for the second iteration and k_0 by k_2 for the third iteration). All 10 initial pulse positions are assigned to tracks $T_{(k_1+n) mod\, 5}$ in the second iteration and to tracks $T_{(k_2+n) mod\, 5}$ in the third iteration. The same search procedure described above is repeated for these other two iterations. For the three iterations, the total number of tested position combinations is $3 \times 4 \times 8 \times 8 = 768$.

E.3.8.1.2 Codeword computation of the 35-bit fixed codebook

The two pulse positions in each track are encoded with 6 bits and the sign of the first pulse in each track is encoded with one bit. The second pulse sign is implicitly determined based on the order of pulse positions.

The two pulses in each track (2 positions and 2 signs) are encoded in 7 bits. Each pulse position needs 3 bits (8 possible positions) and each sign needs 1 bit. That is a total of 8 bits for each pair of pulses. However, 1 bit can be reduced considering the fact that about half the position combinations are redundant. For example, placing pulse 1 at position a and pulse 2 at position b is equivalent to placing pulse 1 at position b and pulse 2 at position a (when the signs are not considered). A simple approach of implementing the pulse encoding is to use only 1 bit for the sign information and 6 bits for the two positions, while ordering the positions in a way such that the other sign information can be easily deduced.

To better explain this, assume that the two pulses in a track are located at positions $p1$ and $p2$ with sign indices $s1$ and $s2$, respectively ($s = 0$ if the sign is positive and $s = 1$ if the sign is negative). The index of the two pulses is given by:

$$\mathrm{I} = (p1/5) + s1 \times 8 + (p2/5) \times 16$$

If $p1 \leq p2$ then $s2 = s1$; otherwise, $s2$ is different from $s1$. Thus, when constructing the codeword, if the two signs are equal, then the smaller position is assigned to $p1$ and the larger position to $p2$; otherwise, the larger position is assigned to $p1$ and the smaller position to $p2$.

This procedure is repeated for each track to obtain five 7-bit indices.

E.3.8.2 Fixed codebook in backward LP mode

In the backward LP mode, the 18 bits needed for the LP model are not transmitted. Thus, 9 bits are saved every subframe, which are used to increase the size of the fixed codebook form 35 to 44 bits. In this 44-bit codebook, each codebook vector contains 12 pulses. The positions in a subframe are divided into the same track structure described in Table E.2. However, two more pulses are placed, such that two consecutive tracks can contain three pulses instead of two. The two consecutive tracks containing three pulses will be called triple-pulse tracks and the other three tracks containing two pulses will be called double-pulse tracks.

The pulses in each double-pulse track are encoded with 7 bits (as in the 35-bit codebook) and those in each triple-pulse track are encoded with 10 bits. The index of the first triple-pulse track can have 5 different values (5 tracks). This index needs an extra 3 bits. This results in a total of 44 bits ($3 \times 7 + 2 \times 10 + 3$).

E.3.8.2.1 Search procedure of the 44-bit codebook

The codebook search is very similar to that of the 35-bit codebook, with the exception that the tree has now 6 levels of pulse pairs. The same search procedure described in clause E.3.8.1.1 is followed.

The same procedure is used for presetting the pulse signs.

The initial tracks T_k an d T_{k+1} are determined in the same manner.

The 12 pulses i_n, $n = 0,...,11$ are assigned to tracks $T_{(k+n)\, mod\, 5}$, $n = 0,...,11$ respectively.

The pulses are searched in subsets of two pulses, by initially setting pulse i_0 to the maximum of track T_k and pulse i_1 to the maximum of track $T_{(k+1)\, mod\, 5}$. Then it is proceeded by searching the pulse pair (i_2, i_3) by testing all the 8×8 possible position combinations in tracks $T_{(k+2)\, mod\, 5}$ and $T_{(k+2)\, mod\, 5}$ and repeating the procedure for the rest of the pulse pairs (i_4, i_5), (i_6, i_7), (i_8, i_9), and (i_{10}, i_{11}). This results now in a total of $5 \times 8 \times 8$ positions tested.

Two more iterations are carried out similar to the 35-bit codebook resulting in a total of $3 \times 5 \times 8 \times 8 = 960$ tested positions.

Similar to G.729 and to the 35-bit forward codebook, the selected codebook vector is filtered through the pre-filter $P(z) = 1/(1 - \beta z^{-T})$ to enhance the harmonic components.

E.3.8.2.2 Codeword computation of the 44-bit fixed codebook

The two pulses in each of the three double-pulse tracks are encoded using the same approach described in clause E.3.8.1.2.

The three pulses in a triple-pulse track are encoded using the same philosophy by adding three bits for the position of the third pulse. The three positions are encoded with 3 bits each and the sign of the first pulse is encoded with 1 bit. The signs of the other two pulses are deduced from the pulse orders, similar to the double-pulse tracks. Again, we will explain this with an example. Assume that the three pulses in a triple-pulse track are located at positions $p1$, $p2$ and $p3$ with sign indices $s1$, $s2$, and $s3$, respectively. The index of the three pulses is given by:

$$\mathrm{I} = (p1/5) + s1 \times 8 + (p2/5) \times 16 + (p3/5) \times 128$$

If $p1 \le p2$ then $s2 = s1$; otherwise, $s2$ is different from $s1$. Similarly, if $p2 \le p3$ then $s3 = s2$; otherwise, $s3$ is different from $s2$. When constructing the codeword, the pulse positions in a track are assigned to $p1$, $p2$, and $p3$ taking this sign relationship into consideration.

In total, 5 indices are returned, one for each track. The first index is that of the first triple-pulse track. This index is encoded with 13 bits; 10 for the positions and signs, as explained above, and 3 for the track index (0 to 4). The second index is that of the second triple-pulse track and is encoded with 10 bits. The last three indices are those of the three double-pulse tracks and are encoded with 7 bits each.

E.3.9 Quantization of the gains

The same as clause 3.9.

E.3.10 Memory update

The same as clause 3.10.

E.4 Functional description of the decoder

First, the parameters are decoded. The transmitted parameters are listed in Table E.3. The first parameter decoded is the LP mode information and its parity bit. According to this information, the frame is classified either as forward, backward or erased. In forward mode, the decoder parameters are the LSP coefficients, the two fractional pitch delays, the two forward fixed-codebook vectors, and the two sets of adaptive- and fixed-codebook gains. In backward mode, the decoded parameters are the two fractional pitch delays, the two backward fixed-codebook vectors, and the two sets of adaptive- and fixed-codebook gains. Then, the LP backward analysis is performed on the past synthesized signal and the decoded parameters are used to compute the reconstructed speech signal as will be described in clause E.4.1. This reconstructed signal is enhanced by a post-processing operation consisting of a postfilter, a high-pass filter and an upscaling (see clause E.4.2). Clause E.4.4 describes the error concealment procedure used when either a parity error has occurred, or when the frame erasure flag has been set.

Table E.3 – Description of transmitted parameters indices

a) Parameters indices in forward mode		
Symbol	**Description**	**Description**
*M*0	Switch LP mode	1
*M*1	Parity bit for LP mode	1
*L*0	Switched MA predictor of LSP quantizer	1
*L*1	First stage vector of quantizer	7
*L*2	Second stage lower vector of LSP quantizer	5
*L*3	Second stage higher vector of LSP quantizer	5
*P*1	Pitch delay first subframe	8
*P*0	Parity bit for pitch delay	1
*C*0_1	Track 0 fixed codebook first subframe	7
*C*1_1	Track 1 fixed codebook first subframe	7
*C*2_1	Track 2 fixed codebook first subframe	7
*C*3_1	Track 3 fixed codebook first subframe	7
*C*4_1	Track 4 fixed codebook first subframe	7
*GA*1	Gain codebook (stage 1) first subframe	3
*GB*1	Gain codebook (stage 2) first subframe	4
*P*2	Pitch delay second subframe	5
*C*0_2	Track 0 fixed codebook second subframe	7
*C*1_2	Track 1 fixed codebook second subframe	7
*C*2_2	Track 2 fixed codebook second subframe	7
*C*3_2	Track 3 fixed codebook second subframe	7
*C*4_2	Track 4 fixed codebook second subframe	7
*GA*2	Gain codebook (stage 1) second subframe	3
*GB*2	Gain codebook (stage 2) second subframe	4

Table E.3 – Description of transmitted parameters indices

b) Parameters indices in backward mode		
*M*0	Switch LP mode	1
*M*1	Parity bit for LP mode	1
*P*1	Pitch delay first subframe	8
*P*0	Parity bit for pitch delay	1
*C*0_1	Fixed codebook track index + pulses 0, 5 and 10 first subframe	13
*C*1_1	Fixed codebook pulses 1, 6 and 11 first subframe	10
*C*2_1	Fixed codebook pulses 2 and 7 first subframe	7
*C*3_1	Fixed codebook pulses 3 and 8 first subframe	7
*C*4_1	Fixed codebook pulses 4 and 9 first subframe	7
*GA*1	Gain codebook (stage 1) first subframe	3
*GB*1	Gain codebook (stage 2) first subframe	4
*P*2	Pitch delay second subframe	5
*C*0_2	Fixed codebook track index + pulses 0, 5 and 10 second subframe	13
*C*1_2	Fixed codebook pulses 1, 6 and 11 second subframe	10
*C*2_2	Fixed codebook pulses 2 and 7 second subframe	7
*C*3_2	Fixed codebook pulses 3 and 8 second subframe	7
*C*4_2	Fixed codebook pulses 4 and 9 second subframe	7
*GA*2	Gain codebook (stage 1) second subframe	3
*GB*2	Gain codebook (stage 2) second subframe	4
NOTE – The bit stream ordering is reflected by the order in the table. For each parameter, the most significant bit (MSB) is transmitted first.		

E.4.1 Parameter decoding procedure

Similar to G.729. The number of parameters is greater (more excitation codebook parameters and one LP mode indication parameter). The decoding process is done in the following order.

E.4.1.1 Backward/forward decoding procedure

One bit is used to indicate to the decoder the LP mode: backward or forward. Then, the parity bit mode is compared with this LP mode bit. If these bits are not identical, the frame is considered as erased and the procedure described in clause E.4.4 is applied. Otherwise, according to this LP mode indication, the same switching procedure as described in clause E.3.2.7 is performed at the decoder to obtain the LP filter that will be used for the synthesis.

The high stationarity indicator High_Stat(n) is computed once per frame as described in clause E.3.2.7.2.

Another high stationarity indicator High_Stat2 that will be used by the gain attenuation procedure in case of erased frame is computed each subframe (see clause E.4.4.3). If the current subframe is at least the 30th of consecutive backward subframes, High_Stat2 is set to 1, else it is set to 0.

E.4.1.2 Decoding of LP parameters

E.4.1.2.1 Computing the LP backward filter

In any LP mode (backward or forward) and even if the frame is erased (see clause E.4.4), one backward LP analysis per frame is performed, using the same procedures as those performed in the encoder in clause E.3.2 to obtain the encoder LP backward filter (windowing and autocorrelation computation, Levinson-Durbin algorithm).

E.4.1.2.2 Forward mode

In forward mode, the same decoding procedure of the LP parameters is applied as in G.729. The interpolation procedure of the LP coefficients is the same as described in clause E.3.2.5.

E.4.1.2.3 Backward mode

In case that one of the previous frames has been erased, the current backward filter computed in clause E.4.1.2.1 $A_{bwd}^{(current)}$ is not directly used but linearly interpolated with the last "correct" backward filter (see clause E.4.4) prior to the interpolation procedure of the LP coefficients described in clause E.3.2.5.

E.4.1.3 Computation of the parity bit of the adaptive-codebook delay

Before the excitation is reconstructed, the parity bit is recomputed from the adaptive-codebook delay index $P1$ (see clause E.3.7). If this bit is not identical to the transmitted parity bit $P0$, it is likely that bit errors occurred during transmission. If a parity error occurs on $P1$, the delay value T_1 is replaced by the delay value calculated in the previous subframe (see clause E.4.4.5).

E.4.1.4 Decoding of the adaptive-codebook vector

The same as clause 4.1.3.

E.4.1.5 Decoding of the fixed-codebook vector

The received codebook indices are used to extract the positions and signs of the pulses. This is done by reversing the process described in clauses E.3.8.1.2 and E.3.8.2.2 for the 35-bit and 44-bit codebooks, respectively. Once the pulse positions and signs are decoded, the fixed codebook vector $c(n)$ is constructed by:

$$c(n) = \sum_{i=0}^{N_p-1} s_i \delta(n - p_i)$$

where s_i are pulse signs, p_i are the pulse positions, and N_p is the number of pulses (10 or 12). If the integer part of the pitch delay is less than the subframe size 40, $c(n)$ is modified similar to equation (48) in G.729.

E.4.1.6 Decoding of the adaptive- and fixed-codebook gains

The same as clause 4.1.5.

E.4.1.7 Computing the reconstructed speech

Similar to clause 4.1.6. (The order of the LP filter could be 30 instead of 10.)

E.4.2 Post-processing

As in G.729. The post-processing consists of three functions: adaptive postfiltering, high-pass filtering and signal upscaling. The adaptive postfiltering is similar to G.729 postfiltering except for the parameters γ_p, γ_n, and γ_d that have been made adaptive according to the high stationarity indicator High_Stat and the current frame LP mode. After 20 consecutive high stationarity backward frames, there is no more postfiltering.

E.4.2.1 Long-term postfilter

The long-term postfiltering procedure is the same as clause 4.2.1:

Adaptive filter:

$$H_p(z) = \frac{1}{1+\gamma_p g_l}\left(1+\gamma_p g_l z^{-T}\right)$$

except for the value of the parameter γ_p that has been made adaptive according to the high stationarity indicator High_Stat and the current frame LP mode.

If the high stationarity state is detected on the input signal (High_Stat = 1), the long-term perceptual filter is progressively flattened.

At frame *n*, if (High_Stat = 1) and if the frame is in backward then:

$$\gamma_p(n) = \gamma_p(n-1) - (\gamma_{pmax}/20)$$

$$\text{if } (\gamma_p(n) < 0) \text{ then } \gamma_p(n) = 0$$

Else, the filter recovers progressively the initial value γ_{pmax}:

$$\gamma_p(\mathrm{n}) = \gamma_p(n-1) + (\gamma_{pmax}/20)$$

$$\text{if } (\gamma_p(n) > \gamma_{pmax}) \text{ then } \gamma_p(n) = \gamma_{pmax}$$

The value of γ_{pmax} is set to 0.25. When $\gamma_p(\mathrm{n})$ is equal to 0, there is no adaptive (neither harmonic, neither short-term) postfiltering.

E.4.2.2 Short-term postfilter

The only modifications brought to the G.729 algorithm concern:

– The LP filter used to calculate the short-term perceptual weighting filter $H_f(z)$ is the LP filter computed in clause E.4.1.2: either the 10 coefficients forward LP filter (computed in clause E.4.1.2.2) if the frame is in forward mode or the 30 coefficients backward LP filter (computed in clause E.4.1.2.3) if the frame is in backward mode.

$$H_f(z) = \frac{A(z/\gamma_n)}{A(z/\gamma_d)} = \frac{1+\sum_{i=1}^{i=m_{b/f}} \gamma_n^i a_i z^{-i}}{1+\sum_{i=1}^{i=m_{b/f}} \gamma_d^i a_i z^{-i}}$$

– The values of the parameters γ_n and γ_d that are adapted according to the high stationarity indicator High_Stat (see clause E.4.1.1) and the LP mode of the current frame (backward or forward).

If a high stationarity state is detected on the input signal (High_Stat = 1) and if the current frame is in backward, the short-term LP postfilter is progressively flattened down to no postfiltering at all ($\gamma_n(n) = \gamma_d(n) = 0$).

At frame *n*, if (High_Stat = 1 and LP_mode = 1) then:

$$\gamma_n(n) = \gamma_n(n-1) - (\gamma_{nmax}/20)$$

$$\gamma_d(n) = \gamma_d(n-1) - (\gamma_{dmax}/20)$$

$$\text{if } (\gamma_n(n) < 0) \text{ then } \gamma_n(n) = 0$$

$$\text{if } (\gamma_d(n) < 0) \text{ then } \gamma_d(n) = 0$$

Else, the filter recovers progressively the initial values γ_{nmax} and γ_{dmax}:

$$\gamma_n(n) = \gamma_n(n-1) + (\gamma_{nmax}/20)$$
$$\gamma_d(n) = \gamma_d(n-1) + (\gamma_{dmax}/20)$$
$$\text{if } (\gamma_n(n) > \gamma_{nmax}) \text{ then } \gamma_n(n) = \gamma_{nmax}$$
$$\text{if } (\gamma_d(n) > \gamma_{dmax}) \text{ then } \gamma_d(n) = \gamma_{dmax}$$

With $\gamma_{nmax} = 0.7$ and $\gamma_{dmax} = 0.65$

E.4.2.3 Tilt compensation

The tilt compensation filtering is the same as clause 4.2.3, except for the computation of the first parcor where the length of the impulse response is 32 instead of 20.

E.4.2.4 Adaptive gain control

The same as clause 4.2.4.

E.4.2.5 High-pass filtering and up-scaling

The same as clause 4.2.5.

E.4.3 Encoder and decoder initialization

All static encoder and decoder variables should be initialized to 0, except the variables listed in Tables 9 and E.4.

Table E.4 – Description of parameters with non-zero initialization

Variable	Reference	Initial value
$Stat(-1)$	E.3.2.7.2	10 000
$\gamma_p(-1)$	E.4.2.2	0.25
$\gamma_n(-1)$	E.4.2.2	0.7
$\gamma_d(-1)$	E.4.2.2	0.65
$\alpha_g^{(-1)}$	E.4.4.3	1.0
$T_{sav}^{(-1)}$	E.4.4.5	30

E.4.4 Concealment of frame erasures

Basically, the bad frame concealment procedure is similar to clause 4.4. The same voicing decision as in G.729 is used but some refinements in the gain attenuation procedure have been brought taking into account the high stationarity indicator High_Stat2 to adapt the muting factor. A special procedure has also been added to improve the backward filter robustness to frame erasures.

The specific steps taken for an erased frame are:

1) repetition of the LP mode;
2) in forward mode, repetition of the synthesis filter parameter; in backward mode, use of the second step backward LP filter as described in clause E.4.4;
3) attenuation of adaptive- and fixed-codebook gains;
4) attenuation of the memory of the gain predictor; and
5) generation of the replacement excitation.

E.4.4.1 Repetition of LP mode

When a frame is erased, the LP mode is set to the previous frame LP mode. The initial value is set to 0 (forward mode).

E.4.4.2 Computation of synthesis filter parameters

Note that the backward LP analysis described in clause E.4.1.2.1 is always performed, even if the frame is erased.

To improve the robustness of the backward filter, for each frame, a second step backward filter is computed. This filter is equal to the computed backward filter in error free conditions, but is different if an erasure has occurred at some frame before the current one: Let $A^*_{bwd}(n)$ denote the second step backward filter for frame n. The computed LP filter of the current frame n being denoted $A_{bwd}(n)$, $A^*_{bwd}(n)$ is obtained by linear interpolation of $A_{bwd}(n)$ and $A^*_{bwd}(n_e)$ where n_e represents the last erased frame (i.e., the last reliable second step backward filter).

$$A^*_{bwd}(n) = \alpha_{lpbwd} A^*_{bwd}(n_e) + (1.0 - \alpha_{lpbwd}) A_{bwd}(n)$$

The initial value of the interpolation factor α_{lpbwd} is 0.0. α_{lpbwd} is updated at the end of the current frame to be applied for the next frame. The adaptation procedure is the following:

Whenever an erased frame occurs, α_{lpbwd} is fixed to the maximum value 1.0.

For each valid frame n:

if frame n is in forward mode, $\alpha_{lpbwd} = 0.0$.

else (frame n is in backward mode) α_{lpbwd} is decreased by an amount depending on the value of the high stationarity indicator High_Stat: If High_Stat is equal to 1, then α_{lpbwd} is decreased by a step of 0.1, else it is decreased by a step of 0.5 (slow recovery for highly stationary signals, else fast recovery).

The second step backward filter $A^*_{bwd}(n)$ will then be used in the frame n processing.

If the erased frame is considered as forward, the same procedure as in clause 4.4.1 is applied.

E.4.4.3 Attenuation of adaptive- and fixed-codebook gains

The attenuation of the adaptive- and fixed-codebook gains depends on the number of consecutive erased subframes before the current subframe and the second stationary indicator High_Stat2 computed in clause E.4.1.1. Let N_{bf} be the number of consecutive erased subframes before the current subframe indexed m. The attenuation procedure is the following:

If less than 2 consecutive subframes have been erased ($N_{bf} < 2$),

$$g_c^{(m)} = g_c^{(m-1)}$$

If High_Stat2 is equal to 1 then $g_p^{(m)} = 1.0$, else $g_p^{(m)} = 0.95$

otherwise: ($N_{bf} \geq 2$)

$$g_p^{(m)} = g_p^{(m-1)} \times \alpha_g^{(m-1)}$$

$$g_c^{(m)} = g_c^{(m-1)} \times \alpha_g^{(m-1)}$$

The adaptation of the attenuation factor also depends on N_{bf} and on High_Stat2:

If less than 2 consecutive subframes have been erased ($N_{bf} < 2$),

$$\alpha_g{}^{(m)} = \alpha_g{}^{(m-1)}(=1.0)$$

otherwise: ($N_{bf} \geq 2$)

If High_Stat2 is equal to 1 then:

$$\text{if } (N_{bf} > 10) \text{ then } \alpha_g{}^{(m)} = \alpha_g{}^{(m-1)} \times \alpha_g^h$$

$$\text{else } \alpha_g{}^{(m)} = \alpha_g{}^{(m-1)} \times \alpha_g^l$$

with $\alpha_g^l = 0.98$ and $\alpha_g^h = 0.995$. When the subframe is not erased, $\alpha_g{}^{(m)}$ is reset to the initial value $\alpha_g{}^{(-1)}$ equal to 1.

E.4.4.4 Attenuation of the memory gain predictor

The same as clause 4.4.3.

E.4.4.5 Generation of the replacement excitation

As in clause 4.4.4, the excitation used depends on the periodicity classification. If the last reconstructed frame was classified as periodic, the current frame is considered to be periodic as well. In that case only the adaptive-codebook is used, and the fixed-codebook contribution is set to zero.

The adaptive-codevector index of an erroneous subframe m (either belonging to an erased frame or if the pitch delay parity bit has detected an error) uses a fractional pitch delay calculated and stored at the preceding subframe. Let $T^{(m)}$ be the fractional pitch delay of any valid or not subframe m and let $T_{sav}{}^{(m)}$ be the fractional pitch delay stored for the next subframe error concealment. If m is a valid subframe, $T^{(m)}$ takes the valid decoded value else $T^{(m)}$ is taken equal to $T_{sav}{}^{(m-1)}$. The computation of $T_{sav}{}^{(m)}$ is as follows:

Let us introduce the integer $stat_T{}^{(m)}$ with values in [0,7] that indicates the stationary nature of the pitch delay. $stat_T{}^{(m)}$ is initialized to 0. Let $\text{int}\left(T^{(m)}\right)$ denote the integer part of $T^{(m)}$.

If $\left|\text{int}\left(T^{(m)}\right) - \text{int}\left(T^{(m-1)}\right)\right| < 5$ then $stat_T{}^{(m)} = stat_T{}^{(m-1)} + 1$ and $T_{sav}{}^{(m)} = T^{(m)}$

else if there exists a multiple T_{mult} of $\min\left(\text{int}\left(T^{(m)}\right), \text{int}\left(T^{(m-1)}\right)\right)$
such that $\left|T_{mult} - \max\left(\text{int}\left(T^{(m)}\right), \text{int}\left(T^{(m-1)}\right)\right)\right| < 5$: if $stat_T{}^{(m-1)} > 0$ then $stat_T{}^{(m)} = stat_T{}^{(m-1)} - 1$ and $T_{sav}{}^{(m)} = T_{sav}{}^{(m-1)}$

otherwise $stat_T{}^{(m)} = 0$ and $T_{sav}{}^{(m)} = T^{(m)}$

Therefore multiples or submultiples of the pitch delay are replaced by the estimated pitch period during stationary voiced parts of the signal.

The adaptive-codebook gain is based on an attenuated value computed in clause E.4.4.3.

If the last reconstructed frame was classified as non-periodic, the current frame is considered to be non-periodic as well, and the adaptive-codebook contribution is set to zero. The fixed-codebook contribution is generated by randomly selecting the 5 codebook indices. The same random generator as that of G.729 is used. The fixed-codebook gain is attenuated with the procedure described in clause E.4.4.3.

E.5 Bit-exact description of the CS-ACELP coder

ANSI C code specifying the 11.8 kbit/s CS-ACELP coder in 16-bit fixed-point is available from ITU-T. As of the approval of this text, the current version of this ANSI C code is Version 1.3 of February 2000. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

The following clauses summarize the use of this simulation code, and how the software is organized.

E.5.1 Use of the simulation software

The C code consists of two main programs **codere.c**, which simulates the encoder, and **decodere.c**, which simulates the decoder. The encoder is run as follows:

codere inputfile bitstreamfile rate_option

The decoder is run as follows:

decodere bitstreamfile outputfile

The input file and output file are sampled data files containing 16-bit PCM signals. The mapping table of the encoded bit stream is contained in the simulation software. The rate_option is either 1 to select the high level extension (11.8 kbit/s) or 0 to select the main body of G.729 (8 kbit/s) or a file_rate_name: a binary file of 16-bit word containing either 0 or 1 to select the rate on a frame-by-frame basis; the default is 0 (8 kbit/s).

E.5.2 Organization of the simulation software

In the fixed-point ANSI C simulation, the types of fixed-point data and the set of basic operators used are the same as in the G.729 software. Some additional tables have been added that are found in tab_ld8e.h (see Table E.5).

Table E.5 – Summary of tables found in tab_ld8e.h

Table name	Size	Description
lag_h_bwd	30	Lag window for backward LP bandwidth expansion (high part)
lag_l_bwd	30	Lag window for backward LP bandwidth expansion (low part)
bitsno_E_fwd	18	Bit allocation in forward mode
bitsno_E_bwd	16	Bit allocation in backward mode
hw	145	Backward LP analysis window
bitrates	2	Table of available bit rates
tab_log	17	Lookup table in base 2 logarithm Q.11

The files can be classified into four groups:

1) Files identical to G.729 software files, part of the main body of G.729 listed in Table E.6.

2) Files similar to G.729 software files, some minor modifications have been introduced to cope with Annex E listed in Table E.7.

3) Files adapted from G.729 software files, some source code lines have been introduced to existing G.729 files to deal with Annex E listed in Table E.8.

4) Files specific to Annex E (new files) listed in Table E.9.

Table E.6 – List of software files identical to G.729 software

File name	Description
basic_op.c	Basic operators
oper_32b.c	Extended basic operators
dspfunc.c	Mathematical functions
gainpred.c	Gain predictor
lpcfunc.c	Miscellaneous routines related to LP filter
pred_lt3.c	Generation of adaptive codebook
pre_proc.c	Preprocessing (HP filtering and scaling)
p_parity.c	Compute pitch parity
qua_gain.c	Gain quantizer
pwf.c	Computation of perceptual weighting coefficients (8 kbit/s)
pitch.c	Pitch search
util.c	Utility functions
acelp_co.c	Search fixed codebook (8 kbit/s)
post_pro.c	Post processing (HP filtering and scaling)
de_acelp.c	Decode algebraic codebook (8 kbit/s)
dec_lag3.c	Decode adaptive-codebook index
basic_op.h	Basic operators prototypes
ld8k.h	Function prototypes
oper_32b.h	Extended basic operators prototypes
tab_ld8k.c	ROM tables
tab_ld8k.h	Extern ROM table declarations
typedef.h	Data type definition (machine-dependent)

Table E.7 – List of software files similar to G.729 software

File name	Description
qua_lspe.c	LSP quantizer
filtere.c	Filter functions

Table E.8 – List of software files adapted from G.729 software

File name	Description
codere.c	Main encoder routine
cod_ld8e.c	Encoder routine
decodere.c	Main decoder routine
dec_ld8e.c	Decoder routine
decgaine.c	Decode gains
pste.c	Postfilter routines
bitse.c	Bit manipulation routines
lspgetqe.c	LSP quantizer
lpce.c	LP analysis
lspdece.c	LSP decoding routing

Table E.9 – List of software files specific to Annex E software

File name	Description
bwfw.c	Backward/forward switch selection
bwfwfunc.c	Miscellaneous routines related to backward/forward switch selection
ld8e.h	Function prototypes for G.729, Annex E
pwfe.c	Computation of perceptual weighting coefficients (11.8 kbit/s)
acelp_e.c	Search fixed codebook (11.8 kbit/s)
deacelpe.c	Decode algebraic codebook (11.8 kbit/s)
tab_ld8e.c	ROM tables for G.729, Annex E
tab_ld8e.h	Extern ROM declarations for G.729, Annex E
track_pi.c	Pitch tracking

E.6 Bibliography

– ITU-T Recommendation G.728 (1992), *Coding of speech at 16 kbit/s using low-delay code excited linear prediction*.

Annex F

Reference implementation of G.729 Annex B DTX functionality for Annex D

(This annex forms an integral part of this Recommendation)

Summary

This annex provides the DTX functionality for the 6.4 kbit/s CS-ACELP algorithm of Annex D using the basic algorithm in Annex B.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 6.4 kbit/s and 8 kbit/s with DTX functionality.

F.1 Scope

This annex provides a description of integrating Annexes B and D, hereby defining DTX functionality for Annex D. It presents a standard way of performing this integration and expansion of the functionality thereby guiding the industry and ensuring a standard speech quality and compatibility worldwide. The integration has been performed with focus on several constraints in order to satisfy the needs of the industry:

1) Bit-exactness with the main body and individual annexes.
2) Minimum additional program code, memory, and complexity usage.
3) Stringent quality requirements to new functionality in line with quality and application areas of the according standard annexes.

F.2 Normative references

This annex refers to materials defined in the main body and Annexes B and D.

F.3 Overview

G.729 main body and Annexes B and D provide a bit-exact fixed-point specification of a CS-ACELP coder at 8 kbit/s, with DTX functionality and lower bit-rate extension capability at 6.4 kbit/s. Exact details of these specifications are given in bit-exact fixed-point C in an electronic file attached to this annex. This annex describes and defines the integration of Annexes B and D.

F.4 New functionality

This clause presents a brief overview of the modifications/additions to the algorithms in order to facilitate the integration of Annexes B and D.

F.4.1 Annex B DTX operation with Annex D

Integrating Annexes B and D functionality in order to provide DTX operation with Annex D is straightforward. The voice activity detection (VAD), silence insertion description (SID) coding and comfort noise generation (CNG) of Annex B are reused without any modifications. Care is taken to update the parameters for the phase dispersion for the postfilter in Annex D during discontinued transmission (see clause F.5.1).

F.5 Algorithm description

This clause presents the algorithm description of the necessary additions to the algorithms of the individual annexes in order to facilitate the integration. All remaining modules originate from the main body, Annex B or D.

F.5.1 Update of state variables specific to Annex D during discontinued transmission

The only state variables specific to Annex D are the state variables of the phase dispersion module (see clause D.6.2) at the decoder. In case of inactive frames, the same update procedure as in case of nominal bit rate (8 kbit/s) is followed using as adaptive and ACELP gain estimations the gain values computed by the comfort noise excitation generator (see clause B.4.4).

F.6 Description of C source code

Annex F, integrating Annexes B and D, is simulated in 16-bit fixed-point ANSI-C code using the same types of fixed-point data and the same set of fixed-point basic operators as in the G.729 software. The ANSI-C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of this Recommendation and in Annexes B, D and F. As of the approval of this text, the current version of this ANSI C code is Version 1.2 of October 2006. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

The following clauses summarize the use of this simulation code, and how the software is organized.

F.6.1 Use of the simulation software

The C code consists of two main programs **coderf.c** and **decoderf.c**, which simulate encoder and decoder, respectively. The encoder is run as follows:

```
coderf inputfile bitstreamfile dtx_option rate_option
```

The decoder is run as follows:

```
decoderf bitstreamfile outputfile
```

The **inputfile** and **outputfile** are 8 kHz sampled data files containing 16-bit PCM signals. The **bitstreamfile** is a binary file containing the bit stream; the mapping table of the encoded bit stream is contained in the simulation software. The two parameters are used for the encoder: dtx_option and rate_option where:

dtx_option =	1: DTX enabled 0: DTX disabled, the default is 0 (DTX disabled).
rate_option =	0 to select the lower rate (6.4 kbit/s); = 1 to select the main body of G.729 (8 kbit/s); or a file_rate_name: a binary file of 16-bit word containing either 0, 1 to select the rate on a frame-by-frame basis; the default is 1 (8 kbit/s).

F.6.2 Organization of the simulation software

The files can be classified into three groups:

1) Files identical to software files of G.729 main body, Annex B or D are listed in Table F.1.
2) Files adapted from software files of G.729 Annex B or D, listed in Table F.2, some minor modifications have been introduced to cope with the integration.
3) Files integrating software files from G.729 main body, Annexes B and D, listed in Table F.3.

Table F.1 – List of software files identical to software files of G.729 main body, Annex B or D

File name	Description	Identical to
Basic_op.c	Basic operators	Main
Oper_32b.c	Extended basic operators	Main
Dspfunc.c	Mathematical functions	Main
Gainpred.c	Gain predictor	Main
Lpcfunc.c	Miscellaneous routines related to LP filter	Main
Pre_proc.c	Preprocessing (HP filtering and scaling)	Main
P_parity.c	Compute pitch parity	Main
Pwf.c	Computation of perceptual weighting coefficients (8 kbit/s)	Main
Pred_lt3.c	Generation of adaptive codebook	Main
Post_pro.c	Post-processing (HP filtering and scaling)	Main
Typedef.h	Data type definition (machine-dependent)	Main
Basic_op.h	Basic operators prototypes	Main
Oper_32b.h	Extended basic operators prototypes	Main
Filter.c	Filter functions	Main
Lspgetq.c	LSP quantizer	Main
De_acelp.c	ACELP decoding	Main
Lpc.c	LP analysis	B
Lspcdec.c	LSP decoding routines	B
Qua_lsp.c	LSP quantizer	B
Tab_ld8k.c	ROM tables	B
Taming.c	Pitch instability control	B
Dtx.c	DTX decision	B
Dtx.h	Prototype and constants	B
Qsidgain.c	SID gain quantization	B
QsidLSF.c	SID-LSF quantization	B
Tab_dtx.c	ROM tables	B
Pst.c	Postfilter routines	B
Vad.c	VAD	B
ld8k.h	Function prototypes	B
Vad.h	Prototype and constants	B
Tab_ld8k.h	Extern ROM tables declarations	B
Sid.h	Prototype and Constants	B
Octet.h	Octet transmission mode definition	B
Tab_dtx.h	Extern ROM table declarations	B
Util.c	Utility functions	B
Pitchd.c	Pitch search	D

Table F.1 – List of software files identical to software files of G.729 main body, Annex B or D

File name	Description	Identical to
Declag3d.c	Decode adaptive-codebook index	D
Acelpcod.c	ACELP codebook search	D
Deacelpd.c	Decode ACELP codebook	D
Qua_g8k.c	Gain quantizer	D
Dec_g8k.c	Decode gain	D
Qua_g6k.c	Gain quantizer	D
Dec_g6k.c	Decode gain	D
Tabld8kd.c	ROM tables for G.729 at 6.4 kbit/s	D
Tabld8kd.h	Extern ROM declarations for G.729 at 6.4 kbit/s	D
ld8kd.h	Function prototypes for G.729 Annex D	D

Table F.2 – List of software files adapted from software files of G.729 Annexes B and D

File name	Description	Adapted from
Calcexc.c	CNG excitation calculation	B
Dec_sidf.c	Decode SID information	B
Phdisp.c	Phase dispersion	D

Table F.3 – List of software files integrating software files from G.729 main body, Annexes B and D

File name	Description	Integrated from
Coderf.c	Main encoder routine	Main + B + D
Cod_ld8f.c	Encoder routine	Main + B + D
Decoderf.c	Main decoder routine	Main + B + D
Dec_ld8f.c	Decoder routine	Main + B + D
Bitsf.c	Bit manipulation routines	Main + B + D
Ld8f.h	Constant and function prototypes for G.729 Annex F	Main + B + D

Annex G

Reference implementation of Annex B DTX functionality for Annex E

(This annex forms an integral part of this Recommendation)

Summary

This annex provides the DTX functionality for the 11.8 kbit/s CS-ACELP algorithm of G.729 Annex E using the basic algorithm in Annex B.

This annex includes an electronic attachment containing reference C code and test vectors fixed-point implementation of CS-ACELP at 8 kbit/s and 11.8 kbit/s with DTX functionality.

G.1 Scope

This annex provides a description of integrating Annexes B and E, hereby defining DTX functionality for Annex E. It presents a standard way of performing this integration and expansion of the functionality thereby guiding the industry and ensuring a standard speech quality and compatibility worldwide. The integration has been performed with focus on several constraints in order to satisfy the needs of the industry:

1) Bit-exactness with the main body and individual annexes.
2) Minimum additional program code, memory, and complexity usage.
3) Stringent quality requirements to new functionality in line with quality and application areas of the according standard annexes.

G.2 Normative references

This annex refers to materials defined in the main body and Annexes B and E.

G.3 Overview

G.729 main body and Annexes B and E provide a bit-exact fixed-point specification of a CS-ACELP coder at 8 kbit/s, with DTX functionality and higher bit-rate extension capability at 11.8 kbit/s. Exact details of these specifications are given in bit-exact fixed-point C code in an electronic file attached to this annex. This annex describes and defines the integration of Annexes B and E.

G.4 New functionality

This clause presents a brief overview of the modifications/additions to the algorithms in order to facilitate the integration of Annexes B and E. Also certain additions have been found necessary in order to accommodate the application area of the different modules.

G.4.1 Annex B DTX operation with Annex E

Integrating Annexes B and E functionality in order to provide DTX operation with Annex E requires certain considerations. Since the DTX operation of Annex B is based on the 10th order LPC analysis, the VAD function of Annex B is performed after the 10th order forward adaptive LPC analysis and before the backward adaptive LPC analysis of Annex E. In case the VAD function detects "non-speech" the LPC mode of Annex E is forced to forward adaptive LPC and the backward adaptive LPC analysis is skipped. Furthermore, it has been found necessary to add a correctional module after the VAD in order to detect music and accommodate the somewhat expanded application area of Annex E – one of the purposes of Annex E is to provide transmission capability of music with a certain quality. Accordingly, during the development of Annex E there were strict requirements to the performance with music signals. On the other hand, for the main body and Annexes B and D, there were no strict requirements to the performance with music signals. In order to guarantee the quality with music signals of Annex E during Annex B DTX operation, the music detection function forces the VAD to "speech" during music segments, hereby ensuring that the music segments are coded with the 11.8 kbit/s of Annex E. The SID coding and the CNG of Annex B are reused without any modifications. Furthermore, care is taken to appropriately update the parameters of the LPC mode selection algorithm of Annex E during discontinued transmission (see clause G.5.2).

G.5 Algorithm description

This clause presents the algorithm description of the necessary additions to the algorithms of the individual annexes in order to facilitate the integration. All remaining modules originate from the main body, Annex B or E.

G.5.1 Music detection

The music detection is a new function. It is performed immediately following the VAD and forces the VAD to "speech" during music segments.

The music detection algorithm corrects the decision from the voice activity detection (VAD) in the presence of music signals. The music detection is based on the following parameters:

– *Vad_deci*: VAD decision of the current frame.
– *PVad_dec*: VAD decision of the previous frame.
– *Lpc_mod*: Flag indicator of either forward or backward adaptive LPC of the previous frame.
– *Rc*: Reflection coefficients from LPC analysis.
– *Lag_buf*: Buffer of corrected open-loop pitch lags of last 5 frames.
– *Pgain_buf*: Buffer of closed-loop pitch gain of last 5 subframes.
– *Energy*: First autocorrelation coefficient $R(0)$ from LPC analysis.
– *LLenergy*: Normalized log energy from VAD module.
– *Frm_count*: Counter of the number of processed signal frames.
– *Rate*: Selection of speech coder.

The algorithm has two main parts:

1) Computation of relevant parameters.
2) Classification based on parameters.

G.5.1.1 Computation of relevant parameters

This clause describes the computation of the parameters used by the decision module.

Partial normalized residual energy

$$Lenergy = 10\log_{10}\left[\prod_{i=1}^{4}\left(1 - Rc(i)^2\right)\frac{Energy}{240}\right]$$

Spectral difference and running mean of partial normalized residual energy of background noise

A spectral difference measure between the current frame reflection coefficients *Rc* and the running mean reflection coefficients of the background noise *mRc* is given by:

$$SD = \sum_{i=1}^{10} (Rc(i) - mRc(i))^2$$

The running means $\overline{mrc}$ and *mLenergy* are updated as follows using the VAD decision *Vad_deci* that was generated by the VAD module.

```
if Vad_deci == Noise{
  $\overline{mrc} = 0.9\overline{mrc} + 0.1\overline{rc}$
  $mLenergy = 0.9 mLenergy + 0.1 Lenergy$
}
```

Open-loop pitch lag correction for pitch lag buffer update

The open-loop pitch lag T_{op} is corrected to prevent pitch doubling or tripling as follows:

```
$avg\_lag = \sum_{i=1}^{4} \frac{Lag\_buf(i)}{4}$
if $\left[ abs\left[ \frac{T_{op}}{2} - avg\_lag \right] <= 2 \right]$
  $Lag\_buf(5) = \frac{T_{op}}{2}$
else if $\left[ abs\left[ \frac{T_{op}}{3} - avg\_lag \right] <= 2 \right]$
  $Lag\_buf(5) = \frac{T_{op}}{3}$
else
  $Lag\_buf(5) = T_{op}$
```

It should be noted that the open-loop pitch lag T_{op} is not modified and is the same as derived by the open-loop analysis.

Pitch lag standard deviation

$$std = \sqrt{\frac{Var}{4}}$$

where:

$$Var = \sum_{i=1}^{5} (Lag\_buf(i) - \mu)^2 \text{and } \mu = \sum_{i=1}^{5} \left[\frac{Lag\_buf(i)}{5} \right]$$

Running mean of pitch gain

$$mPgain = 0.8mPgain + 0.2\theta, \text{ where } \theta = \sum_{i=1}^{5}\left[\frac{Pgain\_buf(i)}{5}\right]$$

The pitch gain buffer *Pgain_buf* is updated after the subframe processing with a pitch gain value of 0.5 if *Vad_deci = NOISE*, and otherwise with the quantized pitch gain.

Pitch lag smoothness and voicing strength indicator

A pitch lag smoothness and voicing strength indicator *Pflag* is generated using the following logical steps:

First, two intermediary logical flags *Pflag*1 and *Pflag*2 are obtained as:

```
if ($std < 1.3$ and $mPgain > 0.45$) set $Pflag1 = 1$ else 0
if ($mPgain > Thres$) set $Pflag2 = 1$ else 0,
where $Thres = 0.63$
```

Finally, *Pflag* is determined from the following:

```
if (($PVad\_dec == VOICE$ and ($Pflag1 == 1$ or $Pflag2 == 1$)) or ($Pflag2 == 1$))
set $Pflag = 1$ else 0
```

Stationarity counters

A set of counters are defined and updated as follows:

a) *count_consc_rflag* tracks the number of consecutive frames where the 2nd reflection coefficient and the running mean of the pitch gain satisfy the following condition:

```
if ($Rc(2) < 0.45$ and $Rc(2) > 0$ and $mPgain < 0.5$)
    $count\_consc\_rflag = count\_consc\_rflag + 1$
else
    $count\_consc\_rflag = 0$
```

b) *count_music* tracks the number of frames where the previous frame uses backward adaptive LPC and the current frame is "speech" (according to the VAD) within a window of 64 frames.

```
if ($Lpc\_mod == 1$ and $Vad\_deci == VOICE$)
    $count\_music = count\_music + 1$
```

Every 64 frames, a running mean of *count_music*, *mcount_music* is updated and reset to zero as described below:

```
if (($Frm\_count$ mod 64) == 0){
    if ($Frm\_count == 64$)
        $mcount\_music = count\_music$
else
    $mcount\_music = 0.9\ mcount\_music + 0.1 count\_music$
}
```

c) *count_consc* tracks the number of consecutive frames where the *count_music* remains zero:

```
if ($count\_music == 0$)
    $count\_consc = count\_consc + 1$
```

```
else
    $count\_consc = 0$
    if ($count\_consc > 500$ or $count\_consc\_rflag > 150$) set $mcount\_music = 0$
```

count_music in b) is reset to zero every 64 frames after the update of the relevant counters. The logic in c) is used to reset the running mean of *count_music*.

d) *count_pflag* tracks the number of frames where *Pflag* = 1, within a window of 64 frames.

```
if ($Pflag == 1$)
    $count\_pflag = count\_pflag + 1$
```

Every 64 frames, a running mean of *count_pflag*, *mcount_pflag*, is updated and reset to zero as described below:

```
if (($Frm\_count$ mod 64) == 0){
  if ($Frm\_count == 64$)
    $mcount\_pflag = count\_pflag$
else{
  if ($count\_pflag > 25$)
    $mcount\_pflag = 0.98 mcount\_pflag + 0.02 count\_pflag$
  else ($count\_pflag > 20$)
    $mcount\_pflag = 0.95 mcount\_pflag + 0.05 count\_pflag$
  else
    $mcount\_pflag = 0.9 mcount\_pflag + 0.1 count\_pflag$
  }
}
```

e) *count_consc_pflag* tracks the number of consecutive frames satisfying the following condition:

```
if ($count\_pflag == 0$)
  $count\_consc\_pflag = count\_consc\_pflag + 1$
else
  $count\_consc\_pflag = 0$
if ($count\_consc\_pflag > 100$ or $count\_consc\_rflag > 150$) set $mcount\_pflag = 0$
```

count_pflag is reset to zero every 64 frames. The logic in e) is used to reset the running mean of *count_pflag*.

G.5.1.2 Classification

Based on the estimation of the above parameters, the VAD decision *Vad_deci* from the VAD module is reverted if the following conditions are satisfied:

```
if ($Rate = G729E$){
  if ($SD > 0.15$ and $(Lenergy - mLenergy) > 4$ and $LLenergy > 50$)
    $Vad\_deci = VOICE$
  else if (($SD > 0.38$ or $(Lenergy - mLenergy) > 4$) and $LLenergy > 50$))
    $Vad\_deci = VOICE$
  else if (($mcount\_pflag >= 10$ or $mcount\_music >= 1.0938$ or $Frm\_count < 64$)
        and $LLenergy > 7$)
    $Vad\_deci = VOICE$
}
```

Note that the music detection function is called all the time regardless of the operational coding mode in order to keep the memories current. However, the VAD decision *Vad_deci* is altered only if Annex G is operating at 11.8 kbit/s (Annex E). It should be noted that the music detection only has the capability to change the decision from "non-speech" to "speech" and not vice versa.

G.5.2 Update of state variables specific to Annex E during discontinued transmission

G.5.2.1 Update of encoder state variables specific to Annex E

At the encoder in case of inactive frames, the update of state variables is identical to the update performed in Annex E in case of switch to the nominal 8 kbit/s bit rate. The update procedure is the following: the LP mode is set to 0, the global stationarity indicator is decreased and the high stationarity indicator is reset to 0 (see clause E.3.2.7.2), the interpolation factor used to smoothly switch from LP forward filter to backward LP filter is reset to its maximum value (see clause E.3.2.7.1).

G.5.2.2 Update of decoder state variables specific to Annex E during discontinued transmission

At the decoder in case of inactive frames, the update of state variables is almost identical to the update performed in Annex E in case of switch to the forward mode only rates (8 kbit/s) except that the pitch delay stationary indicator is reset to 0 instead of being computed by the pitch tracking procedure (see clause E.4.4.5).

G.6 Description of C source code

This annex, integrating Annexes B and E, is simulated in 16-bit fixed-point ANSI-C code using the same types of fixed-point data and the same set of fixed-point basic operators as in the G.729 software. The ANSI-C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of G.729 and in Annexes B, E and G. As of the approval of this text, the current version of this ANSI C code is Version 1.2 of October 2006. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

The following clauses summarize the use of this simulation code, and how the software is organized.

G.6.1 Use of the simulation software

The C code consists of two main programs **coderg.c** and **decoderg.c**, which simulate encoder and decoder, respectively. The encoder is run as follows:

coderg inputfile bitstreamfile dtx_option rate_option

The decoder is run as follows:

decoderg bitstreamfile outputfile

The **inputfile** and **outputfile** are 8 kHz sampled data files containing 16-bit PCM signals. The **bitstreamfile** is a binary file containing the bit stream; the mapping table of the encoded bit stream is contained in the simulation software. The two parameters are used for the encoder: dtx_option and rate_option where:

dtx_option = 1: DTX enabled 0: DTX disabled, the default is 0 (DTX disabled).

rate_option = 1 to select the main G.729 (8 kbit/s); = 2 is to select the higher rate (11.8 kbit/s) or a file_rate_name: a binary file of 16-bit word containing either 1, 2 to select the rate on a frame-by-frame basis; the default is 1 (8 kbit/s).

G.6.2 Organization of the simulation software

The files can be classified into four groups:

1) Files identical to software files of G.729 main body, Annex B or E listed in Table G.1.
2) Files adapted from software files of G.729 Annex B or E, listed in Table G.2, some minor modifications have been introduced to cope with the integration.
3) Files integrating G.729 software files of G.729 main body, Annexes B and E, listed in Table G.3.
4) New files specific to integrated Annexes B and E, listed in Table G.4.

Table G.1 – List of software files identical to software files of G.729 main body and Annex B or E

File name	Description	Identical to
Basic_op.c	Basic operators	Main
Oper_32b.c	Extended basic operators	Main
Dspfunc.c	Mathematical functions	Main
Gainpred.c	Gain predictor	Main
Lpcfunc.c	Miscellaneous routines related to LP filter	Main
Pre_proc.c	Preprocessing (HP filtering and scaling)	Main
P_parity.c	Compute pitch parity	Main
Pwf.c	Computation of perceptual weighting coefficients (8 kbit/s)	Main
Pred_lt3.c	Generation of adaptive codebook	Main
Post_pro.c	Post-processing (HP filtering and scaling)	Main
Pitch.c	Pitch search	Main
Dec_lag3.c	Decode adaptive-codebook index	Main
Typedef.h	Data type definition (machine dependent)	Main
Basic_op.h	Basic operators prototypes	Main
Oper_32b.h	Extended basic operators prototypes	Main
Acelp_co.c	ACELP codebook search	Main
De_acelp.c	Decode ACELP codebook	Main
Qua_gain.c	Gain quantizer	Main
De_acelp.c	ACELP decoding	Main
Tab_ld8k.c	ROM tables	B
Taming.c	Pitch instability control	B
Qsidgain.c	SID gain quantization	B
QsidLSF.c	SID-LSF quantization	B
Tab_dtx.c	ROM tables	B
Calcexc.c	CNG excitation calculation	B
Util.c	Utility functions	B
Ld8k.h	Function prototypes	B

Table G.1 – List of software files identical to software files of G.729 main body and Annex B or E

File name	Description	Identical to
Tab_ld8k.h	Extern ROM tables declarations	B
Dtx.h	Prototype and constants	B
Sid.h	Prototype and constants	B
Octet.h	Octet transmission mode definition	B
Tab_dtx.h	Extern ROM table declarations	B
Vad.h	Prototype and constants	B
Pwfe.c	Computation of perceptual weighting coefficients	E
Filtere.c	Filter functions	E
Lspgetqe.c	LSP quantizer	E
Lspdece.c	LSP decoding routing	E
Qua_lspe.c	LSP quantizer	E
Bwfwfunc.c	Miscellaneous routines related to backward/forward switch selection	E
Ld8e.h	Function prototypes for G.729, Annex E	E
Acelp_e.c	Search fixed codebook (11.8 kbit/s)	E
Deacelpe.c	Decode algebraic codebook (11.8 kbit/s)	E
Decgaine.c	Decode gains	E
Tab_ld8e.c	ROM tables for G.729 at 11.8 kbit/s	E
Tab_ld8e.h	Extern ROM declarations for G.729 at 11.8 kbit/s	E
Track_pi.c	Pitch tracking	E

Table G.2 – List of software files adapted from software files of G.729 main body, Annexes B and E

File name	Description	Adapted from
Dtxg.c	DTX decision	B
Vadg.c	VAD	B
Dec_sidf.c	Decode SID information	B
Bwfwg.c	Backward/forward switch selection	E

Table G.3 – List of software files integrating software files from G.729 main body, Annexes B and E

File name	Description	Integrated from
Coderg.c	Main encoder routine	B + E
Cod_ld8g.c	Encoder routine	B + E
Decoderg.c	Main decoder routine	B + E
Dec_ld8g.c	Decoder routine	B + E
Bitsg.c	Bit manipulation routines	B + E
Lpcg.c	LP analysis	B + E
Pstg.c	Postfilter routines	B + E
Ld8g.h	Constant and function prototypes for G.729, Annex G	B + E

Table G.4 – List of software files specific to integrated G.729 Annexes B and E

File name	Description
Mus_dtct.c	Music detection module

Annex H

Reference implementation of switching procedure between Annexes D and E

(This annex forms an integral part of this Recommendation)

Summary

This annex defines the necessary mechanisms for switching operation between 6.4 kbit/s Annex D and 11.8 kbit/s Annex E. Previously, only one switching from 8 kbit/s G.729 was specified.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 6.4 kbit/s, 8 kbit/s and 11.8 kbit/s without DTX functionality.

H.1 Scope

This annex provides a description of the integration of Annexes D and E, hereby defining switching procedure between Annexes D and E. It presents a standard way of performing this integration and expansion of the functionality thereby guiding the industry and ensuring a standard speech quality and compatibility worldwide. The integration has been performed with focus on several constraints in order to satisfy the needs of the industry:

1) Bit-exactness with the main body and individual annexes.
2) Minimum additional program code, memory, and complexity usage.
3) Stringent quality requirements to new functionality in line with quality and application areas of the according standard annexes.

H.2 Normative references

This annex refers to materials defined in the main body and Annexes D and E.

H.3 Overview

G.729 main body and Annexes D and E provide a bit-exact fixed-point specification of a CS-ACELP coder at 8 kbit/s, lower and higher bit-rate extension capability at 6.4 and 11.8 kbit/s. Exact details of these specifications are given in bit-exact fixed-point C code in an electronic file attached to this annex. This annex describes and defines the integration of Annexes D and E.

H.4 Algorithm description

This clause presents the algorithm description of the necessary additions to the algorithms of the individual annexes in order to facilitate the integration. All remaining modules originate from the main body, Annex D or E.

H.4.1 Update of state variables specific to Annex D during Annex E frames

The only state variables specific to Annex D are the state variables of the phase dispersion module (see clause D.6.2) at the decoder. In case of 11.8 kbit/s frames, the same update procedure as in case of nominal bit rate (8 kbit/s) is followed.

H.4.2 Update of state variables specific to Annex E during Annex D frames

H.4.2.1 Update of encoder state variables specific to Annex E

At the encoder in case of 6.4 kbit/s frames, the update of state variables is identical to the update performed in Annex E in case of switch to the nominal bit rate 8 kbit/s. The update procedure is the following: the LP mode is set to 0, the global stationarity indicator is decreased and the high stationarity indicator is reset to 0 (see clause E.3.2.7.2), the interpolation factor used to smoothly switch from LP forward filter to backward LP filter is reset to its maximum value (see clause E.3.2.7.1).

H.4.2.2 Update of decoder state variables specific to Annex E during Annex D frames

At the decoder in case of 6.4 kbit/s frames, the update of state variables is identical to the update performed in Annex E in case of switch to the nominal bit-rate (8 kbit/s) mode.

H.5 Description of C source code

This annex, integrating Annexes D and E, is simulated in 16-bit fixed-point ANSI-C code using the same types of fixed-point data and the same set of fixed-point basic operators as in the G.729 software. The ANSI-C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of G.729 and in Annexes D, E and H. As of the approval of this text, the current version of this ANSI C code is Version 1.2 of October 2006. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

The following clauses summarize the use of this simulation code, and how the software is organized.

H.5.1 Use of the simulation software

The C code consists of two main programs **coderh.c** and **decoderh.c**, which simulate encoder and decoder, respectively. The encoder is run as follows:

coderh inputfile bitstreamfile rate_option

The decoder is run as follows:

decoderh bitstreamfile outputfile

The **inputfile** and **outputfile** are 8 kHz sampled data files containing 16-bit PCM signals. The **bitstreamfile** is a binary file containing the bit stream; the mapping table of the encoded bit stream is contained in the simulation software. The parameter used for the encoder is: rate_option where:

rate_option = 0 to select the lower rate (6.4 kbit/s); = 1 to select the main G.729 (8 kbit/s); = 2 is to select the higher rate (11.8 kbit/s) or a file_rate_name: a binary file of 16-bit word containing either 0, 1, 2 to select the rate on a frame-by-frame basis; the default is 1 (8 kbit/s).

H.5.2 Organization of the simulation software

The files can be classified into three groups:

1) Files identical to software files of G.729 main body, Annex D or E, listed in Table H.1.
2) Files adapted from software files of G.729 main body and Annexes D and E, listed in Table H.2, some minor modifications have been introduced to cope with the integration of Annexes D and E. Most modifications come from the integration of annexes routines prototypes declaration files in one file (ld8cp.h) or to the integration of extern ROM declaration annexes files into one file (tabld8cp.h). Some were introduced to deal with the update of the annexes state variables.
3) Files integrating G.729 software files of Annex D or E, listed in Table H.3.

Table H.1 – List of software files identical to software files of G.729 main body, Annex D or E

File name	Description	Identical to
Basic_op.c	Basic operators	Main
Oper_32b.c	Extended basic operators	Main
Dspfunc.c	Mathematical functions	Main
Gainpred.c	Gain predictor	Main
Lpcfunc.c	Miscellaneous routines related to LP filter	Main
Pre_proc.c	Preprocessing (HP filtering and scaling)	Main
P_parity.c	Compute pitch parity	Main
Pwf.c	Computation of perceptual weighting coefficients (8 kbit/s)	Main
Pred_lt3.c	Generation of adaptive codebook	Main
Post_pro.c	Post-processing (HP filtering and scaling)	Main
Tab_ld8k.c	ROM tables	Main
Basic_op.h	Basic operators prototypes	Main
Ld8k.h	Function prototypes	Main
Oper_32b.h	Extended basic operators prototypes	Main
Tab_ld8k.h	Extern ROM table declarations	Main
Typedef.h	Data type definition (machine-dependent)	Main
Taming.c	Pitch instability control	B
Qua_g8k.c	Gain quantizer	D
Qua_g6k.c	Gain quantizer	D
Tabld8kd.c	ROM tables for G.729 at 6.4 kbit/s	D
Tabld8kd.h	Extern ROM declarations for G.729 at 6.4 kbit/s	D
ld8kd.h	Function prototypes for G.729 Annex D	D
Bwfwfunc.c	Miscellaneous routines related to backward/forward switch selection	E
Filtere.c	Filter functions	E
Lpce.c	LP analysis	E
Lspcdece.c	LSP decoding routines	E
Lspgetqe.c	LSP quantizer	E
Qua_lspe.c	LSP quantizer	E
Pstpe.c	Postfilter routines	E
Track_pi.c	Pitch tracking	E
Tab_ld8e.c	ROM tables for G.729 at 11.8 kbit/s	E
Tab_ld8e.h	Extern ROM declarations for G.729 at 11.8 kbit/s	E
Util.c	Utility functions	E

Table H.2 – List of software files adapted from software files of G.729 main body and Annexes D and E

File name	Description	Adapted from
Phdisp.c	Phase dispersion	D
Bwfwh.c	Backward/forward switch selection	E

Table H.3 – List of software files integrating software files from G.729 main body, Annex D or E

File name	Description	Integrated from
Coderh.c	Main encoder routine	D + E
Cod_ld8h.c	Encoder routine	D + E
Decoderh.c	Main decoder routine	D + E
Dec_ld8h.c	Decoder routine	D + E
Acelp_h.c	Search ACELP fixed codebook (6.4, 8, 11.8 kbit/s)	D + E
Deacelph.c	Decode algebraic codebook (6.4, 8, 11.8 kbit/s)	D + E
Pitchh.c	Pitch search	D + E
Declagh.c	Decode adaptive-codebook index	D + E
Decgainh.c	Decode gain	D + E
Bitsh.c	Bit manipulation routines	D + E
Ld8h.h	Constant and function prototypes for G.729 Annex H	D + E

Annex I

Reference fixed-point implementation for integrating G.729 CS-ACELP speech coding main body with Annexes B, D and E

(This annex forms an integral part of this Recommendation)

Summary

This annex describes the integration of G.729 main body with Annexes B, D and E.

This annex includes an electronic attachment containing reference C code and test vectors for fixed-point implementation of CS-ACELP at 6.4 kbit/s, 8 kbit/s and 11.8 kbit/s with discontinuous transmission (DTX) functionality.

I.1 Scope

This annex provides a description of integrating the G.729 main body with Annexes B, D and E, hereby defining the integrated C code. It presents a standard way of performing this integration and expansion of the functionality thereby guiding the industry and ensuring a standard speech quality and compatibility worldwide. The integration has been performed with focus on several constraints in order to satisfy the needs of the industry:

1) Bit-exactness with the main body and individual annexes.
2) Minimum additional program code, memory, and complexity usage.
3) Stringent quality requirements to new functionality in line with quality and application areas of the according standard annexes.

I.2 Normative references

This annex refers to materials defined in the main body and Annexes B, D, and E.

I.3 Overview

G.729 main body and Annexes B, D and E provide a bit-exact fixed-point specification of a CS-ACELP coder at 8 kbit/s, with DTX functionality, lower and higher bit-rate extension capability at 6.4 kbit/s and 11.8 kbit/s. Exact details of these specifications are given in bit-exact fixed-point C code in an electronic attachment to this annex. This annex describes and defines the integration of the G.729 main body with Annexes B, D and E.

I.4 New functionality

This clause presents a brief overview of the modifications/additions to the algorithms in order to facilitate the integration of the main body and Annexes B, D and E. Also certain additions have been found necessary in order to accommodate the application area of the different modules.

I.4.1 Annex B DTX operation with Annex D

Integrating Annexes B and D functionality in order to provide DTX operation with Annex D is straightforward. The voice activity detection (VAD), silence insertion description (SID) coding, and comfort noise generation (CNG) of Annex B are reused without any modifications. Care is taken to update the parameters for the phase dispersion for the postfilter in Annex D during discontinued transmission (see clause I.5.2).

I.4.2 Annex B DTX operation with Annex E

Integrating Annexes B and E functionality in order to provide DTX operation with Annex E is slightly more involved. Since the DTX operation of Annex B is based on the 10th order LPC analysis, the VAD function of Annex B is performed after the 10th order forward adaptive LPC analysis and before the backward adaptive LPC analysis of Annex E. In case the VAD function detects "non-speech", the LPC mode of Annex E is forced to forward adaptive LPC and the backward adaptive LPC analysis is skipped. Furthermore, it has been found necessary to add a correctional module after the VAD in order to detect music and accommodate the somewhat expanded application area of Annex E – one of the purposes of Annex E is to provide transmission capability of music with a certain quality. Accordingly, during the development of Annex E there were strict requirements for performance with music signals. On the other hand, for the main body and Annexes B and D there were no strict requirements for performance with music signals. In order to guarantee the quality with music signals of Annex E during Annex B DTX operation, the music detection function forces the VAD to "speech" during music segments, hereby ensuring that the music segments are coded with the 11.8 kbit/s of Annex E. The SID coding and the CNG of Annex B are reused without any modifications. Furthermore, care is taken to appropriately update the parameters of the LPC mode selection algorithm of Annex E during discontinued transmission (see clause I.5.3).

I.5 Algorithm description

This clause presents the algorithm description of the necessary additions to the algorithms of the individual annexes in order to facilitate the integration. All remaining modules originate from the main body, Annex B, D or E.

I.5.1 Music detection

The music detection is a new function. It is performed immediately following the VAD and forces the VAD to "speech" during music segments. It is active only during Annex E operation, though its parameters are updated continuously independently of bit-rate mode during DTX operation of the integrated G.729.

The music detection algorithm corrects the decision from the voice activity detection (VAD) in the presence of music signals. It is used in conjunction with Annex E during Annex B DTX operation, i.e., in discontinuous transmission mode. The music detection is based on the following parameters:

- *Vad_deci*: VAD decision of the current frame.
- *PVad_dec*: VAD decision of the previous frame.
- *Lpc_mod*: Flag indicator of either forward or backward adaptive LPC of the previous frame.
- *Rc*: Reflection coefficients from LPC analysis.
- *Lag_buf*: Buffer of corrected open-loop pitch lags of last 5 frames.
- *Pgain_buf*: Buffer of closed-loop pitch gain of last 5 subframes.
- *Energy*: First autocorrelation coefficient $R(0)$ from LPC analysis.
- *LLenergy*: Normalized log energy from VAD module.
- *Frm_count*: Counter of the number of processed signal frames.
- *Rate*: Selection of speech coder.

The algorithm has two main parts:

1) Computation of relevant parameters.
2) Classification based on parameters.

I.5.1.1 Computation of relevant parameters

This clause describes the computation of the parameters used by the decision module.

Partial normalized residual energy

$$Lenergy = 10\log_{10}\left[\prod_{i=1}^{4}\left(1 - Rc(i)^2\right)\frac{Energy}{240}\right]$$

Spectral difference and running mean of partial normalized residual energy of background noise

A spectral difference measure between the current frame reflection coefficients *Rc* and the running mean reflection coefficients of the background noise *mRc* is given by:

$$SD = \sum_{i=1}^{10}(Rc(i) - mRc(i))^2$$

The running means $\overline{mrc}$ and *mLenergy* are updated as follows using the VAD decision *Vad_deci* that was generated by the VAD module.

```
$if\ Vad\_deci == NOISE\{$
    $\overline{mrc} = 0.9\overline{mrc} + 0.1\overline{rc}$
    $mLenergy = 0.9mLenergy + 0.1Lenergy$
$\}$
```

Open-loop pitch lag correction for pitch lag buffer update

The open-loop pitch lag T_{op} is corrected to prevent pitch doubling or tripling as follows:

```
$avg\_lag = \sum_{i=1}^{4}\frac{Lag\_buf(i)}{4}$
if $\left[abs\left[\frac{T_{op}}{2} - avg\_lag\right] <= 2\right]$
    $Lag\_buf(5) = \frac{T_{op}}{2}$
else if $\left[abs\left[\frac{T_{op}}{3} - avg\_lag\right] <= 2\right]$
    $Lag\_buf(5) = \frac{T_{op}}{3}$
else
    $Lag\_buf(5) = T_{op}$
```

It should be noted that the open loop pitch lag T_{op} is not modified and is the same as derived by the open-loop analysis.

Pitch lag standard deviation

$$std = \sqrt{\frac{Var}{4}}$$

where:

$$Var = \sum_{i=1}^{5} \left(Lag\_buf(i) - \mu\right)^2 \text{and } \mu = \sum_{i=1}^{5} \left[\frac{Lag\_buf(i)}{5}\right]$$

Running mean of pitch gain

$$mPgain = 0.8mPgain + 0.2\theta,\ \text{ where } \theta = \sum_{i=1}^{5} \left[\frac{Pgain\_buf(i)}{5}\right]$$

The pitch gain buffer *Pgain_buf* is updated after the subframe processing with a pitch gain value of 0.5 if *Vad_deci = NOISE*, and otherwise with the quantized pitch gain.

Pitch lag smoothness and voicing strength indicator

A pitch lag smoothness and voicing strength indicator $Pflag$ is generated using the following logical steps:

First, two intermediary logical flags $Pflag1$ and $Pflag2$ are obtained as:

```
if ($std$ < 1.3 and $mPgain$ > 0.45) set $Pflag1$ = 1 else 0
if ($mPgain$ > $Thres$) set $Pflag2$ = 1 else 0,
where $Thres$ = 0.73 if $Rate$ = $G729D$, otherwise $Thres$ = 0.63
```

Finally, $Pflag$ is determined from the following:

```
if (($PVad\_dec == VOICE$ and ($Pflag1 == 1$ or $Pflag2 == 1$)) or ($Pflag2 == 1$))
set $Pflag = 1$ else 0
```

Stationarity counters

A set of counters are defined and updated as follows:

a) $count\_consc\_rflag$ tracks the number of consecutive frames where the 2nd reflection coefficient and the running mean of the pitch gain satisfy the following condition:

```
if ($Rc(2)$ < 0.45 and $Rc(2)$ > and $mPgain$ < 0.5)
    $count\_consc\_rflag = count\_consc\_rflag + 1$
else
    $count\_consc\_rflag = 0$
```

b) *count_music* tracks the number of frames where the previous frame uses backward adaptive LPC and the current frame is "speech" (according to the VAD) within a window of 64 frames.

```
if ($Lpc\_mod == 1$ and $Vad\_deci == VOICE$)
    $count\_music = count\_music + 1$
```

Every 64 frames, a running mean of *count_music*, *mcount_music* is updated and reset to zero as described below:

```
    if (($Frm\_count$ mod 64) == 0){
    if ($Frm\_count$ mod 64)
        $mcount\_music = count\_music$
else
    $mcount\_music = 0.9\ mcount\_music + 0.1 count\_music$
}
```

c) *count_consc* tracks the number of consecutive frames where the *count_music* remains zero:

```
    if ($count\_music$ == 0)
    $count\_consc = count\_consc + 1$
else
    $count\_consc = 0$
    if ($count\_consc > 500$ or $count\_consc\_rflag > 150$) set $mcount\_music = 0$
```

count_music in b) is reset to zero every 64 frames after the update of the relevant counters.

The logic in c) is used to reset the running mean of *count_music*.

d) *count_pflag* tracks the number of frames where $Pflag = 1$, within a window of 64 frames.

```
if ($Pflag$ == 1)
    $count\_pflag = count\_pflag + 1$
```

Every 64 frames, a running mean of *count_pflag*, *mcount_pflag*, is updated and reset to zero as described below:

```
if (($Frm\_count$ mod 64) == 0){
  if ($Frm\_count$ == 64)
    $mcount\_pflag = count\_pflag$
else{
  if ($count\_pflag > 25$)
    $mcount\_pflag = 0.98 mcount\_pflag + 0.02 count\_pflag$
  else ($count\_pflag > 20$)
    $mcount\_pflag = 0.95 mcount\_pflag + 0.05 count\_pflag$
  else
    $mcount\_pflag = 0.9 mcount\_pflag + 0.1 count\_pflag$
  }
}
```

e) *count_consc_pflag* tracks the number of consecutive frames satisfying the following condition:

```
if ($count\_pflag$ == 0)
  $count\_consc\_pflag = count\_consc\_pflag + 1$
else
  $count\_consc\_pflag = 0$
if ($count\_consc\_pflag > 100$ or $count\_consc\_rflag > 150$) set $mcount\_pflag = 0$
```

count_pflag is reset to zero every 64 frames. The logic in e) is used to reset the running mean of *count_pflag*.

I.5.1.2 Classification

Based on the estimation of the above parameters, the VAD decision *Vad_deci* from the VAD module is reverted if the following conditions are satisfied:

```
if ($Rate = G729E$){
  if ($SD > 0.15$ and $(Lenergy - mLenergy) > 4$ and $LLenergy > 50$)
    $Vad\_deci = VOICE$
    else if (($SD > 0.38$ or ($Lenergy - mLenergy) > 4$) and $LLenergy > 50$)
        $Vad\_deci = VOICE$
    else if (($mcount\_pflag >= 10$ or $mcount\_music >= 1.0938$ or $Frm\_count < 64$)
            and $LLenergy > 7$)
        $Vad\_deci = VOICE$
}
```

Note that the music detection function is called all the time regardless of the operational coding mode in order to keep the memories current. However, the VAD decision *Vad_deci* is altered only if the integrated G.729 is operating at 11.8 kbit/s (Annex E). It should be noted that the music detection only has the capability to change the decision from "non-speech" to "speech" and not vice versa.

I.5.2 Update of state variables specific to Annex D during discontinued transmission

The only state variables specific to Annex D are the state variables of the phase dispersion module (see clause D.6.2) at the decoder. In case of inactive frames, the same update procedure as in case of nominal bit rate (8 kbit/s) is followed using as adaptive and ACELP gain estimations the gain values computed by the comfort noise excitation generator (see clause B.4.4). Note also that the update for the higher rate is identical to the update for the nominal bit rate.

I.5.3 Update of state variables specific to Annex E during discontinued transmission

I.5.3.1 Update of encoder state variables specific to Annex E

At the encoder in case of inactive frames, the update of state variables is identical to the update performed in Annex E in case of switch to the nominal bit rate 8 kbit/s. The update procedure is the following: the LP mode is set to 0, the global stationarity indicator is decreased and the high stationarity indicator is reset to 0 (see clause E.3.2.7.2), the interpolation factor used to smoothly switch from LP forward filter to backward LP filter is reset to its maximum value (see clause E.3.2.7.1). Note that this update is also performed in case of switch to the lower bit rate 6.4 kbit/s.

I.5.3.2 Update of decoder state variables specific to Annex E during discontinued transmission

At the decoder in case of inactive frames, the update of state variables is almost identical to the update performed in Annex E in case of switch to the forward mode only rates (8 kbit/s and 6.4 kbit/s) except that the pitch delay stationary indicator is reset to 0 instead of being computed by the pitch tracking procedure (see clause clause E.4.4.5).

I.6 Description of C source code

This annex, integrating the G.729 main body with Annexes B, D and E, is simulated in 16-bit fixed-point ANSI-C code using the same types of fixed-point data and the same set of fixed-point basic operators as in the G.729 software. The ANSI-C code represents the normative specification of this annex. The algorithmic description given by the C code shall take precedence over the texts contained in the main body of G.729 and in Annexes B, D, E and I. As of the approval of this text, the current version of this ANSI C code is Version 1.2 of October 2006. More recent versions may become available through corrigenda or amendments to G.729. Please ensure to use the latest available version from the ITU-T website.

The following clauses summarize the use of this simulation code, and how the software is organized.

I.6.1 Use of the simulation software

The C code consists of two main programs **coderi.c** and **decoderi.c**, which simulate encoder and decoder, respectively. The encoder is run as follows:

coderi inputfile bitstreamfile dtx_option rate_option

The decoder is run as follows:

decoderi bitstreamfile outputfile

The **inputfile** and **outputfile** are 8 kHz sampled data files containing 16-bit PCM signals. The **bitstreamfile** is a binary file containing the bit stream; the mapping table of the encoded bit stream is contained in the simulation software. The two parameters are used for the encoder: dtx_option and rate_option where:

dtx_option =	1: DTX enabled 0: DTX disabled, the default is 0 (DTX disabled).
rate_option =	0 to select the lower rate (6.4 kbit/s); = 1 to select the main G.729 (8 kbit/s); = 2 is to select the higher rate (11.8 kbit/s) or a file_rate_name: a binary file of 16-bit word containing either 0, 1, 2 to select the rate on a frame-by-frame basis; the default is 1 (8 kbit/s).

I.6.2 Organization of the simulation software

The files can be classified into four groups:

1) Files identical to software files of G.729 main body and Annex B, D or E, listed in Table I.1.
2) Files adapted from software files of G.729 main body and Annex B, D or E, listed in Table I.2, some minor modifications have been introduced to cope with the integration. Most modifications come from the integration of annexes routine prototype declaration files in one file (ld8cp.h) or to the integration of extern ROM declaration annexes files into one file (tabld8cp.h). Some were introduced to deal with the update of the annexes state variables.
3) Files integrating G.729 software files of Annex B, D or E, listed in Table I.3.
4) Files specific (new files) to this integrated G.729 listed in Table I.4.

Table I.1 – List of software files identical to software files of G.729 main body and Annex B, D or E

File name	Description	Identical to
Basic_op.c	Basic operators	Main
Oper_32b.c	Extended basic operators	Main
Dspfunc.c	Mathematical functions	Main
Gainpred.c	Gain predictor	Main
lpcfunc.c	Miscellaneous routines related to LP filter	Main
Pre_proc.c	Preprocessing (HP filtering and scaling)	Main
P_parity.c	Compute pitch parity	Main
pwf.c	Computation of perceptual weighting coefficients (8 kbit/s)	Main
Pred_lt3.c	Generation of adaptive codebook	Main
Post_pro.c	Post-processing (HP filtering and scaling)	Main
Tab_ld8k.c	ROM tables	Main
Basic_op.h	Basic operators prototypes	Main

Table I.1 – List of software files identical to software files of G.729 main body and Annex B, D or E

File name	Description	Identical to
Ld8k.h	Function prototypes	Main
Oper_32b.h	Extended basic operators prototypes	Main
Tab_ld8k.h	Extern ROM table declarations	Main
Typedef.h	Data type definition (machine-dependent)	Main
Taming.c	Pitch instability control	B
Qsidgain.c	SID gain quantization	B
QsidLSF.c	SID-LSF quantization	B
Tab_dtx.c	ROM tables	B
Sid.h	Prototype and constants	B
Octet.h	Octet transmission mode definition	B
Tab_dtx.h	Extern ROM table declarations	B
Pwfe.c	Computation of perceptual weighting coefficients (11.8 kbit/s)	E

Table I.2 – List of software files adapted from software files of G.729 main body and Annex B, D or E

File name	Description	Adapted from
Vad.c	VAD	B
Dtx.c	DTX decision	B
Vad.h	Prototype and constants	B
Dtx.h	Prototype and constants	B
Calcexc.c	CNG Excitation calculation	B
Dec_sid.c	Decode SID information	B
Utilcp.c	Utility functions	B
Phdisp.c	Phase dispersion	D
Bwfw.c	Backward/forward switch selection	E
Bwfwfunc.c	Miscellaneous routines related to backward/forward switch selection	E
Filtere.c	Filter functions	E
Lpccp.c	LP analysis	E
Lspcdece.c	LSP decoding routines	E
Lspgetqe.c	LSP quantizer	E
Qua_lspe.c	LSP quantizer	E
Track_pi.c	Pitch tracking	E

Table I.3 – List of software files integrating software files from G.729 main body and Annex B, D or E

File name	Description	Integrated from
Coderi.c	Main encoder routine	B + D + E
Codld8i.c	Encoder routine	B + D + E
Decodi.c	Main decoder routine	B + D + E
Decld8i.c	Decoder routine	B + D + E
Acelpcp.c	Search ACELP fixed codebook (6.4, 8, 11.8 kbit/s)	D + E
Dacelpcp.c	Decode algebraic codebook (6.4, 8, 11.8 kbit/s)	D + E
Pitchcp.c	Pitch search	D + E
Declagcp.c	Decode adaptive-codebook index	D + E
Q_gaincp.c	Gain quantizer	D + E
Degaincp.c	Decode gain	D + E
Pstpcp.c	Postfilter routines	B + E
Bitscp.c	Bit manipulation routines	B + D + E
Tabld8cp.c	ROM tables for G.729 at 6.4 and 11.8 kbit/s	D + E
Tabld8cp.h	Extern ROM declarations for G.729 at 6.4 and 11.8 kbit/s	D + E
Ld8cp.h	Constant and Function prototypes for G.729 at 6.4 and 11.8 kbit/s	D + E

Table I.4 – List of software files specific to integrated G.729 Annexes B, D and E

File name	Description
Mus_dtct.c	Music detection module

Annex J

An embedded variable bit-rate extension to G.729: An interoperable 8-32 kbit/s scalable wideband extension to G.729

(This annex forms an integral part of this Recommendation)

This annex describes an extension of G.729 for 8-32 kbit/s scalable wideband speech and audio coding algorithm interoperable with the main body of G.729 and its Annexes A and B.

The details of this annex are specified and published in ITU-T Rec. G.729.1 in order to provide for easier maintenance and to give it better visibility.

Appendix I

External synchronous reset performance for G.729 codecs in systems using external VAD/DTX/CNG

(This appendix does not form an integral part of this Recommendation)

Summary

This appendix deals with external synchronous reset capability in systems using external VAD/DTX/CNG, e.g., circuit multiplication equipments (CME) in conjunction with G.729 main body and Annexes A and C.

The use of the external synchronous reset is intended for systems using external VAD/DTX/CNG in conjunction with G.729 main body and Annex A or C. In this situation, the use of external synchronous reset is generally preferable to obtain the best possible speech quality in noisy scenarios where VAD is used. This is especially true when an aggressive VAD is used. When the external VAD has a sufficiently long hangover period (i.e., a less-aggressive VAD), the quality increase of external synchronous reset case compared with "no reset" case is less perceivable.

Scope

Although Annex B defines a "native" (or internal) VAD/DTX/CNG mechanism, some applications require that a different algorithm be used, because of system or complexity constraints. In these cases, when an external VAD/DTX/CNG algorithm (i.e., one that operates independently and does not exploit the internal information of the encoder) is used, there is the possibility that the state of the encoder and decoder will differ significantly, which will degrade quality. Hence, synchronous reset of the encoder and decoder can be beneficial to the overall quality when such external VAD/DTX/CNG algorithms are used. This appendix deals with external synchronous reset capability in systems using external VAD/DTX/CNG, such as CME (circuit multiplication equipment) in conjunction with G.729 main body and Annexes A and C.

I.1 Introduction

The definition of the synchronous reset is that both the encoder state variables and the decoder state variables are set to their respective initial values at the same frame time.

The use of the external synchronous reset is intended for systems using an external VAD/DTX/CNG in conjunction with G.729 main body, Annex A or C. In this situation, the use of external synchronous reset is generally preferable to obtain the best possible speech quality in noisy scenarios where VAD is used. This is especially true when an aggressive VAD using a relatively short hangover period is used. When the external VAD has a sufficiently long hangover period (i.e., a less-aggressive VAD), the quality increase of external synchronous reset case compared to the "no-reset" case is less perceivable. In any case, no harm is expected on quality by applying synchronous reset to the G.729 encoder and decoder in systems using an external VAD/DTX/CNG. Conversely, in spite of the quick convergence of the G.729 algorithm after loss of synchronization, there is evidence that the use of synchronous reset will generally allow attainment of the best possible speech quality.

I.2 Experimental design

Some limited experiments have been performed to test the impact on quality of the introduction of synchronous reset in G.729 codecs into systems using external VAD/DTX/CNG, such as CME (circuit multiplication equipment) in conjunction with G.729 main body, Annexes A and C. The experience has been limited to simulation of CME operation in a pooled codec configuration using Annex C (and G.729 main body). In this CME operation, the "one-to-one relationship" between encoders and decoders cannot be expected throughout the call, which will lead to loss of synchronization between encoder and decoder.

To test the effect of the introduction of synchronous reset in G.729 codecs, some experiments have been run to evaluate the quality of both schemes (*with* synchronous reset and *without* synchronous reset). Various test conditions were used: clean speech at nominal-, high- and low-input levels, and speech with different types of background noise (babble noise, hall noise, vehicular noise) at different signal-to-noise ratio (SNR) values. For each condition, one male and one female talker were used. Two expert listening experiments were performed, one in North American english and the other in french, each experiment using its own external VAD indicator.

To simulate CME operation with pooled codecs configuration, the input bit stream for the G.729 decoder has been composed by interleaving two bit stream files coming from two different G.729 encoders. The interleaving was done according to the respective VAD of the two input files (first active segment of file 1, first active segment of file 2, second active segment of file 1, second active segment of file 2, etc.). Finally, the decoder output file was decomposed into two decoded files according the interleaving scheme. When synchronous reset was used, both encoder and decoder were reset at the beginning of each active spurt, otherwise no reset was used.

I.3 Performance observations

To evaluate the impact on quality of both schemes, an informal expert listening test has been performed using pair-comparison of the active speech segments in the decoded files. The results depended on the external VAD and on the background noise similarities of the two interleaved files. When the external VAD has a sufficiently long hangover period (i.e., a less-aggressive VAD), the two schemes have similar performances when the two interleaved files have similar or high SNR background noise; no artefacts were perceived. When low SNR background noise segments were interleaved with high SNR background noise segments, some artefacts were heard at the beginning of active periods, although their duration was short thanks to the quick convergence of G.729 after loss of synchronization. When a more aggressive VAD was used, the synchronous reset provides a clear improvement.

I.4 Conclusion

Some limited experiments have been performed to test the impact on quality of the introduction of synchronous reset in G.729 codecs. The existing evidence confirms the expectation that no degradation in quality occurs by applying synchronous reset of the G.729 encoder and decoder in CME scenarios. Furthermore, it has been found that the introduction of synchronous reset was generally preferable to obtain the best possible speech quality in noisy scenarios where VAD is used. It is expected that this result can be extended to other systems using external VAD/DTX/CNG in conjunction with G.729.

Appendix II

G.729 Annex B enhancements in voice-over-IP applications – Option 1

(This appendix does not form an integral part of this Recommendation)

II.1 Scope

Although Annex B, defines a VAD/DTX/CNG mechanism, some applications require that a different VAD algorithm be used, because of specific constraints. In particular, this is the case for VoIP applications, where the algorithm described in Annex B shows bad performance under the following conditions:

1) Undesired performance for input signals starting at levels below 15 dB.
2) Annoying breathing-like noise in CNG phase.
3) VAD bad performance under noisy conditions.
4) Additionally, wrong variable initialization has been depicted in the current Annex B.

This appendix deals with corresponding proposals to correct points 1) to 4) as described above, as an alternative to the current Annex B.

II.2 Abbreviations and acronyms

This appendix uses the following abbreviations and acronyms.

DSVD	Digital Simultaneous Voice and Data
DTX	Discontinuous Transmission Mode
G.729B	Silence compression scheme defined in Annex B
SID	Silence Insertion Descriptor
SNR	Signal-to-Noise Ratio
VAD	Voice Activity Detection

II.3 Introduction

Annex B to ITU-T Rec. G.729 ("G.729B") specifies a silence compression scheme for use with G.729, which was optimized for V.70 digital simultaneous voice and data (DSVD) applications. Despite its initial target application, G.729B has been heavily used in VoIP applications, and will continue to serve the industry in the future. G.729B allows G.729 (and its annexes) to operate in two transmission modes, voice and silence, which are classified using voice activity detection (VAD). The discontinuous transmission mode (DTX) is used to determine which of the silence frames are represented with the silence insertion descriptor (SID). During the last few years, problems concerning the use of G.729B in VoIP applications have been reported.

II.4 Identified problems of G.729B in VoIP applications

The reported problems addressed by this appendix are the following:

1) Undesired performance for input signals starting at levels below 15 dB.
2) Annoying breathing-like noise in silence frames.
3) VAD performance on noise condition.

Problem 1: Undesired performance for input signals starting at levels below 15 dB

If the energy of the initial 32 input frames (320 ms) is below 15 dB, G.729B detects all the consecutive input frames above 15 dB as voice. This defeats the purpose of using G.729B; it adds the extra MIPS/memory cost of running G.729B and yet results in no bandwidth savings.

The proposed solution is to restart the initialization frame counter every 320 ms until background noise characteristics are properly initialized.

Problem 2: Annoying breathing-like noise in silence frames

When digital silence or very low level noise follows more than 129 frames (1.29 s) of tones or other stationary signals, the noise gain in the first SID frame is estimated at a very high level; this introduces high level breathing-like noise and causes speech quality degradation. This problem has been reported from various customers of Alcatel, Texas Instruments. The proposed solution is to introduce a hangover at the end of voice frames.

Problem 3: VAD performance on noisy environments

With SNR below 15 dB, current VAD has two problems: on one hand, it oscillates between voice and noise decision, thus reducing the benefits in terms of spare bandwidth due to SID updates, and on the other hand, noise decision is often taken during voice signal.

This appendix results in very few modifications to the existing coder and no change to the existing decoder. The goal is to try to lose as little information as possible from the voice signal instead of having as much as possible "noise decision".

The proposed solution reduces misclassification from voice to noise, especially for low SNR (≤15 dB). It consists of modifying the condition of background noise updates and the condition of voice activity decision smoothing.

The three changes that have been introduced to the existing coder are the following:

– the test that fixes NOISE, when the difference between current signal and previous noise is too small, is withdrawn;
– hysteresis is introduced in the decision to switch from VOICE to NOISE when the mean energy of the background noise is important enough. In the proposal, at least 6 consecutive NOISEs have to be detected before switching from VOICE to NOISE; and
– the required condition to update the running averages of the background noise characteristics is modified.

The modified C code addressing these three problems is provided in the electronic attachment to this appendix.

II.5 Experimental design

Experiments have been performed to test the impact of the introduction of these enhancements.

Problem 1: Undesired performance for input signals starting at levels below 15 dB

Annex B VAD has the following running averages of the background noise characteristics: average full band energy $\overline{E}_f$; average low band energy $\overline{E}_l$; average zero crossing rate $\overline{ZC}$; and average spectral parameters $\{\overline{LSF}_i\}_{i=1}^{p}$. $\overline{E}_f$, $\overline{E}_l$, $\overline{ZC}$ and $\{\overline{LSF}_i\}_{i=1}^{p}$ are initialized to 0 at the very beginning in function vad_init. In Annex B source codes, Annex B VAD makes initial update of the running averages of background noise characteristics using only the frames that have energy E_f greater than 15 dB during the first 32 frames of the input signal. When the energy of all 32 beginning frames is below 15 dB, the running averages of the background noise characteristics are not updated. Practically, the initial 320 ms input occurs during the channel establishment period, so it may not reflect the real background noise. If the phone is in mute mode or on-hook at the beginning of channel connection, the input signal level is usually less than 15 dB. In this case, the running averages of the background noise characteristics are set to zeros without any initial updates. It follows that Annex B VAD detects all the frames as voice in that channel.

To make the initial update more robust than it is now, we proposed that frame counter should be restarted if energies of all the 32 frames are less than 15 dB at the beginning of call set-up. The proposed version of Annex B source codes is compliant with current Annex A and B provisions, and with Annex B test vectors.

Problem 2: Annoying breathing-like noise in silence frames

Different tones at different levels were tested (especially French and US ringing-back tones).

Problem 3: VAD performance on noisy environments

Various test conditions were used: clean speech at nominal-, high- and low-input levels, and speech with different types of background noise (white noise, babble noise, fan noise) at different signal-to-noise ratio (SNR) values.

II.5.1 The fourth smoothing stage

The smoothing of the voice activity decision is divided into four stages (see clause B.3.6). The fourth smoothing stage is:

```
if (($E_f < \overline{E}_f + 614$) and ($Frm\_count > 128$) and ($v\_flag = 0$) and
($rc < 19661$)) then (marker = noise)
```

First, this test is the only smoothing test that does not take the previous decisions into consideration, whereas the aim of this smoothing function is to "*reflect the long-term stationary nature of the speech signal*". Moreover, it is assumed in this test that whatever the background noise, there will always be at least 614 between E and $\overline{E}_f$, if the current frame is a voiced one. In fact, when the energy of the background noise becomes too high, this assertion is no more true, and this test introduces mistakes in the VAD.

The fourth smoothing stage has been suppressed. The Boolean *v_flag*, which is only used in this test, is removed.

II.5.2 Hysteresis

In a noisy environment, it is more difficult for the multi-boundary initial voice activity decision function to clearly make the distinction between voice and noise because the coefficients of the vector of difference parameters are smaller. That is why the initial VAD output is more erratic in this case. In order to avoid this comportment and to obtain a more coherent output signal, a hysteresis has been added at the beginning of the smoothing function:

```
if ($marker = VOICE$) then ($Count\_inert = 0$)
if (($marker = NOISE$) and ($Count\_inert < 6$) and ($MeanSE > 8000$)) then{
    $Count\_inert; ++$
    $marker = VOICE$; }
```

In the third smoothing stage, line 264, the equation $Count\_inert = 6$ is added, in order to avoid interactions between both tests.

II.5.3 The updating test

According to clause B.3.7, the running averages of the background noise characteristics are updated if $(E_f < \overline{E}_f + T_6)$. In the C source code, this condition is replaced by: (($E_f < \overline{E}_f + 614$) and ($rc < 24576$) and ($\Delta S < 83$)).

Let us consider the test vector called `tstseq3.bin`. in Annex B.

Figure II.1 shows the input signal and the decision taken by the multi-boundary initial VAD. The last section of the signal, where there is only noise, is not supposed to be transmitted. Nevertheless, the initial VAD decision is constant and equal to one, even if the noise is maintained for a longer time. This is due to the fact that the update condition described above is always wrong, because of the too high ΔS value. Figure II.2 shows ΔS with the current test (solid red curve) and without the ($SD < 83$) condition (dotted blue curve).

NOTE – ΔS, the spectral distortion, is called SD in the C source code.

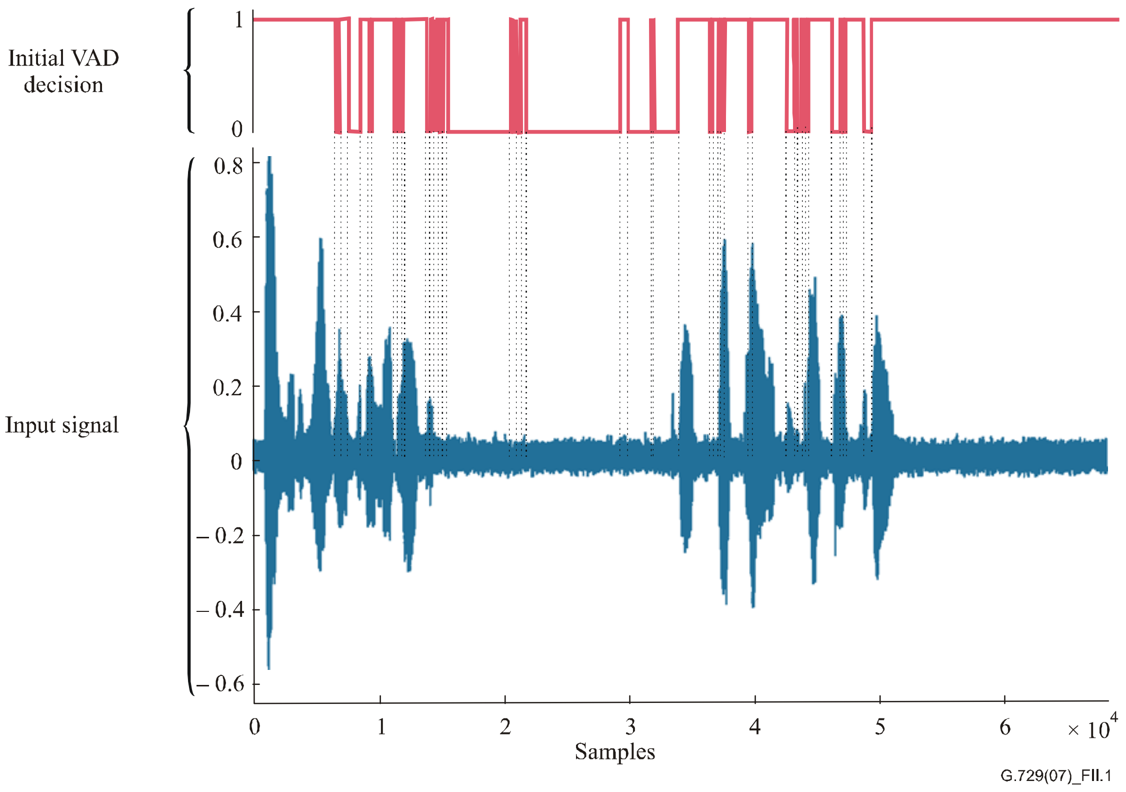


Figure II.1 – G.729 + VAD simulation (Code ITU version 1.3)

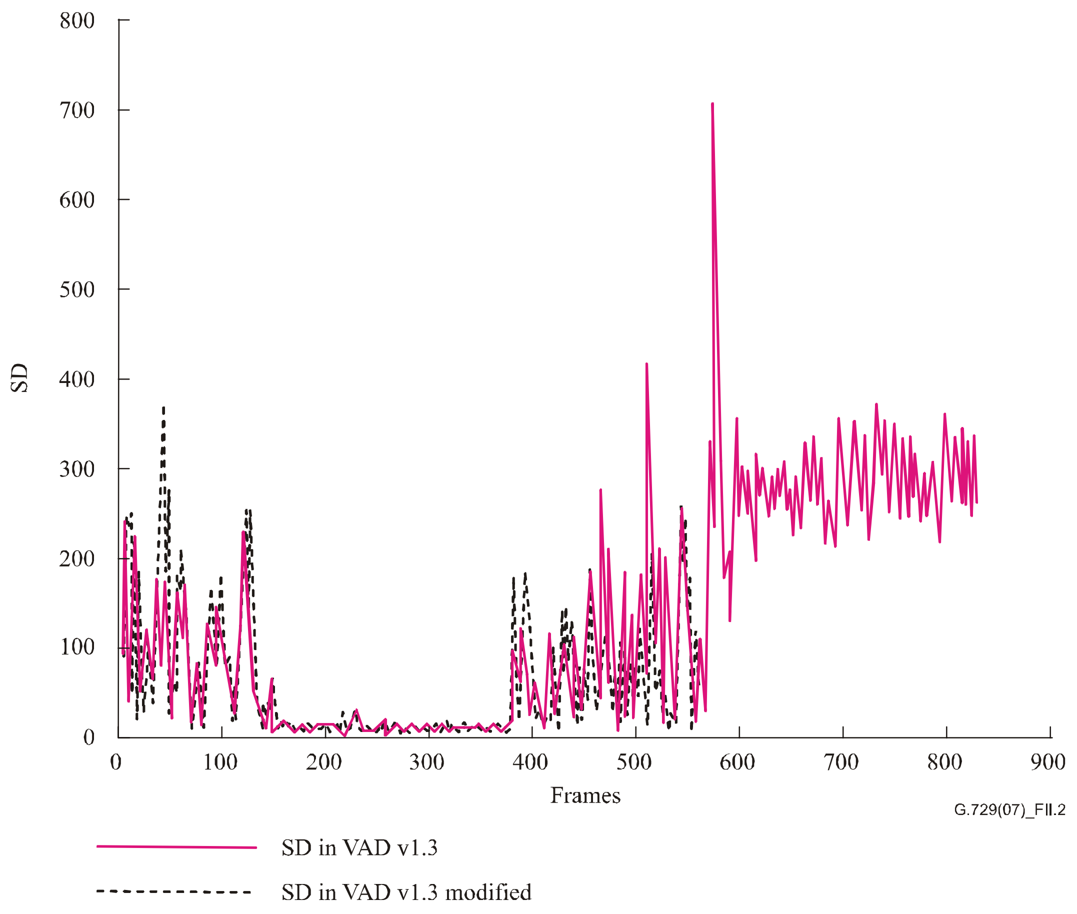


Figure II.2 – Comparison of SD values

Without this condition, the running averages are correctly updated, and the ΔS value is much smaller as it is supposed to be in a noisy environment. The decision taken by the VAD algorithm is correct again (see Figure II.3).

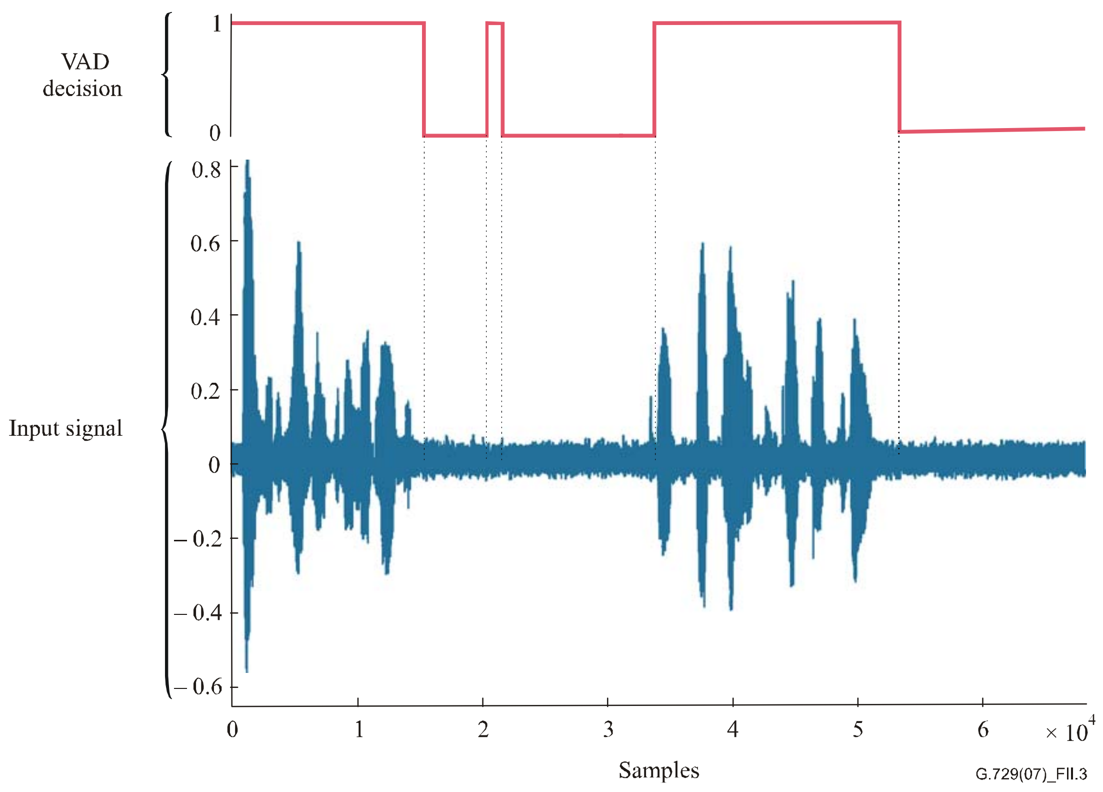


Figure II.3 – G.729 + VAD simulation (Code ITU version 1.3, modified)

II.6 Electronic attachments

Two sets of electronic attachments are provided with this appendix.

The first one is the modified file `vad.c`, which implements modifications in the VAD algorithm of Annex B according to the descriptions in this appendix.

The second electronic attachment is a set of test files that were used to test the algorithm in this appendix.

Appendix III

Annex B enhancements in voice-over-IP applications – Option 2

(This appendix does not form an integral part of this Recommendation)

III.1 Scope

Annex B provides a silence compression scheme for G.729 implementations that are optimized for terminals conforming to [ITU-T V.70]. The silence compression scheme includes voice activity detection (VAD), discontinuous transmission (DTX), silence insertion description (SID), and comfort noise generation (CNG). The application of this Recommendation has expanded beyond V.70 devices and is commonly utilized in voice packet networks (e.g., VoIP), which also require voice activity detection and discontinuous transmission algorithms for bandwidth-efficient communication. Several issues were reported with respect to the operation of Annex B for voice packet networks. This Appendix describes a solution to the problems reported in Annex B, providing improved voice quality while maintaining high bandwidth efficiency, suitable for voice packet-network applications.

This appendix addresses in particular the following issues noted for Annex B:

1) Initialization of background noise statistics at the beginning of the call and updates of background noise statistics when the background noise characteristics change.
2) Early estimation of SID parameters which generate breathing-like noise in sharp-edge energy offsets.
3) Classification of portions of very long and high-level tonal signals as "inactive speech".
4) Frequent changes between "inactive speech" and "active speech" for particular types of background noises.
5) Frequent SID update frames, which is undesirable for VoIP applications where the packet overhead information is considerably larger than the payload for such frames.

III.2 Solutions for the reported issues with Annex B

The issues reported with respect to Annex B were resolved by several modifications. The modifications are under the flag `VAD_VOIP_APP_III` in the attached C program files vad.c and dtx.c. This flag needs to be defined in the project or the makefile.

III.3 Examples for the solutions of reported issues with Annex B

For packet voice applications (e.g., VoIP), the following issues on the operation of VAD and the DTX of Annex B were noted. This appendix provides the solution to these issues:

1) The initial estimation of the background noise characteristics is done during the first 320 ms with a threshold of 15 dB. These two constraints can result in ineffective initial estimate of the background noise characteristics. For example, some voice packet network processors start the G.729 encoder before the actual speech channel is established, which can result in this ineffective initial estimate of the background noise characteristics. Similar behaviour may also happen if the type or the level of background noise changes abruptly. This ineffective update might result in classifying subsequent background noise frames "active-speech", which reduces the bandwidth efficiency expected from the silence compression scheme in Annex B. Figure III.1 contains an example of the result of ineffective estimation of the background noise characteristics in Annex B and how it is handled by this appendix.

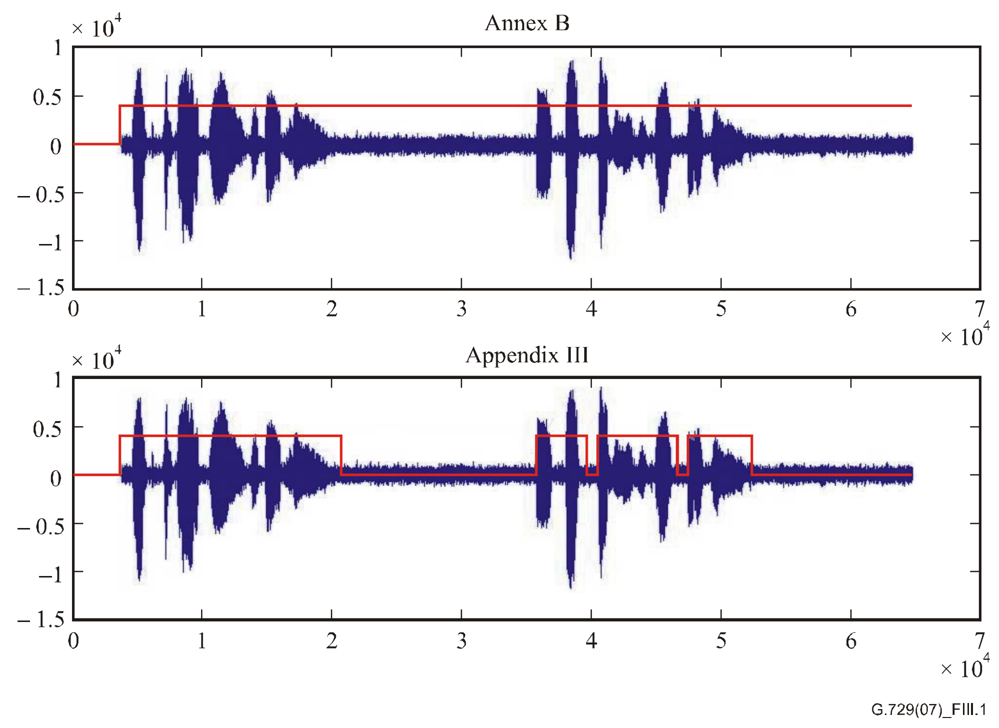


Figure III.1 – Example of estimation of the background noise characteristics

2) Early estimation of the energy and the spectral content of the first SID frame after a sharp edge from high level into a very low level (e.g., silence) can occur, resulting in a breathing-like noise. This issue is mostly noticeable at the sharp offset edge of voice-band tones, such as ring-back or busy tones. Figure III.2 contains an example of sharp-edge offset issue in Annex B and its resolution by this appendix.

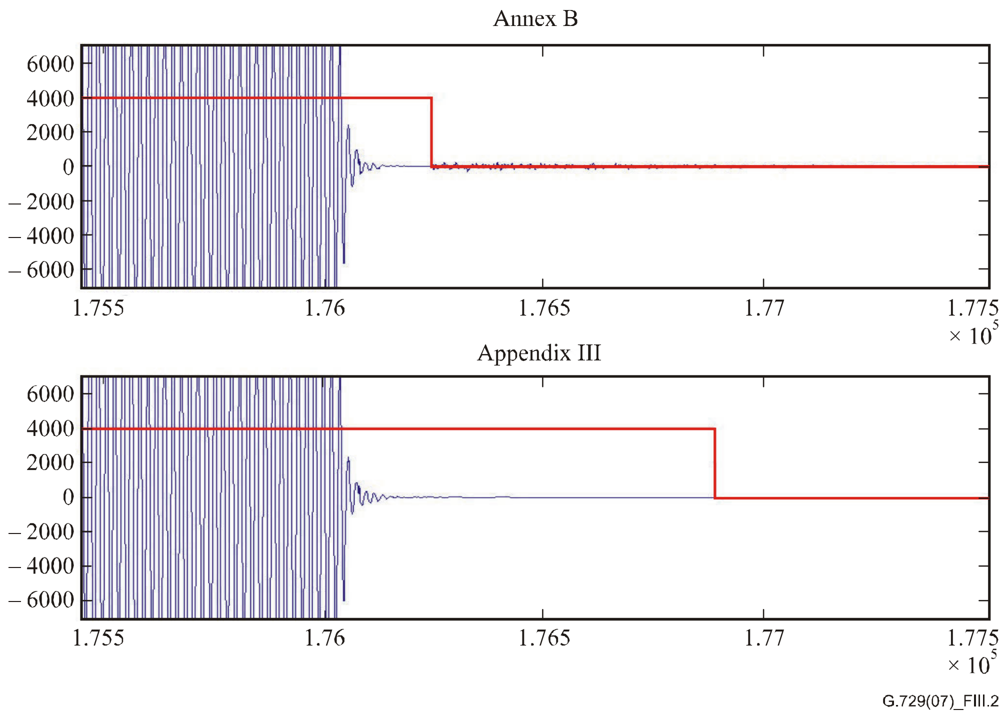


Figure III.2 – Example of sharp-edge offset issue

3) For very long and high-level tonal signals (typically longer than 1 second), the later portions could be classified as "inactive-speech". These later portions of such very long tonal signals might be erroneously reproduced at the decoder as a high-level noise signal. Figure III.3 contains an example of the issue of very long and high-level tonal signals in Annex B and its resolution by this appendix.

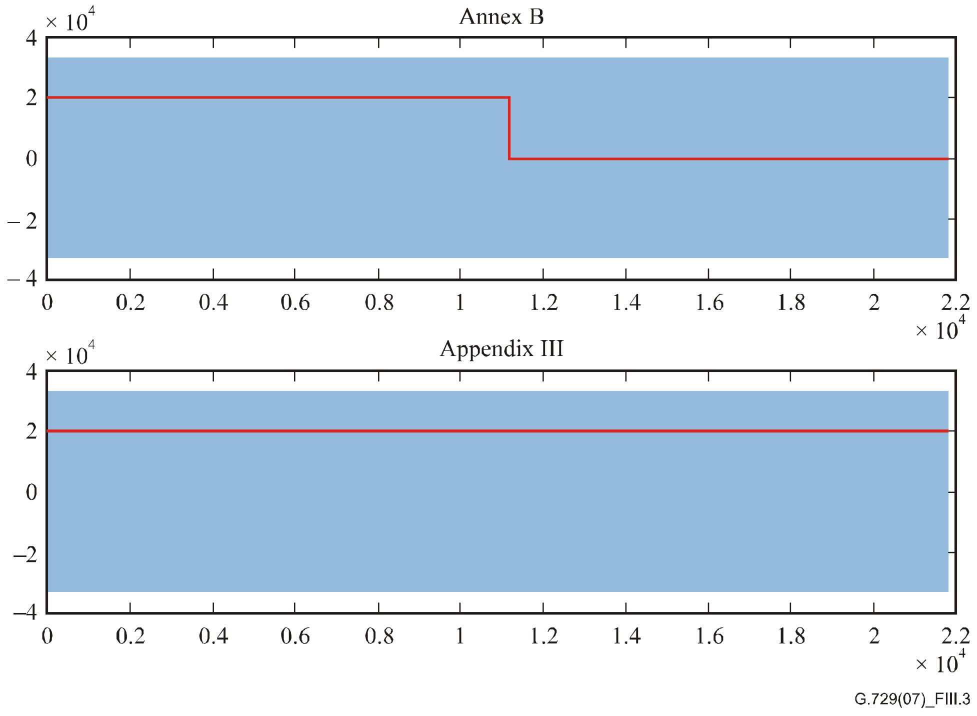


Figure III.3 – Example of very long and high-level tonal signals issue

4) For particular types of background noise, frequent changes between full-rate encoding at 8 kbit/s and low-rate silence encoding may occur. Figure III.4 contains an example of the issue of frequent changes between full-rate encoding at 8 kbit/s and low-rate silence encoding in Annex B and its resolution by this appendix.

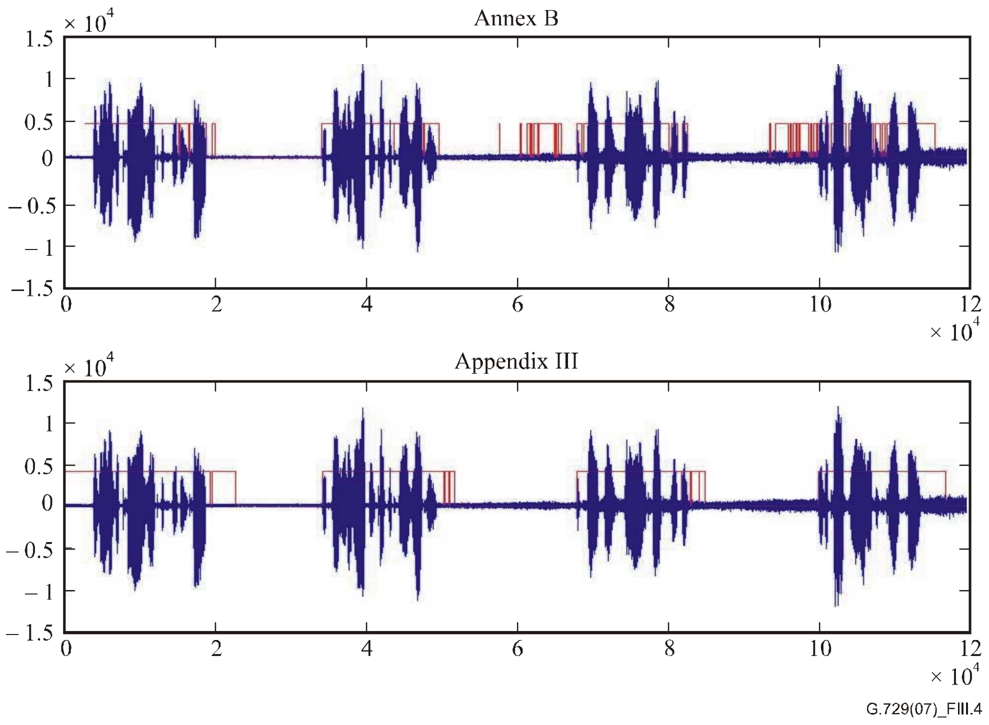


Figure III.4 – Example of frequent changes between full-rate encoding at 8 kbit/s and low-rate silence encoding

5) In voice packet network applications, the transmitted-signal bandwidth is affected not only by the payload but also by the address and header information. Although the size of the payload of a SID frame is small, frequent update of the SID might be a considerable factor on the bandwidth used by Annex B for these applications.

Table III.1 contains an example of the percentage of active frames, SID frames and NT frames in Annex B and in this appendix, demonstrating the reduction in SID frames by this appendix.

Table III.1 – Example of the reduction in active frames, SID frames and NT frames

Noise type	VAD	Speech frames	SID frames	NT frames	SID/Inactive
15 dB street noise	G.729B	4805 (46.2%)	799 (7.68%)	4796 (46.12%)	14.3%
	App III	6079 (58.45%)	275 (2.64%)	4046 (38.90%)	6.4%
15 dB car noise	G.729B	4986 (47.94%)	846 (8.13%)	4568 (43.92%)	15.6%
	App III	5806 (55.83%)	292 (2.81%)	4302 (41.37%)	6.4%
15 dB babble noise	G.729B	4991 (47.99%)	1287 (12.38%)	4122 (39.63%)	23.8%
	App III	5896 (56.69%)	561 (5.39%)	3943 (37.91%)	12.5%

III.4 Electronic attachments

There are three electronic attachments to this appendix. The first two are the modified C source code files vad.c and dtx.c. For correct compilation, the flag `VAD_VOIP_APP_III` needs to be defined in the project or makefile. The other is a set of test files that were used to test the algorithm in this appendix.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	**Transmission systems and media, digital systems and networks**
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2008