Ericsson

PDC Enhanced Full Rate Speech Coder

Specification Version 27H

Preface

This document gives a description of the Ericsson PDC Enhanced Full Rate coder, a 6.7 kbit/s ACELP speech codec.

Section 1 gives a general description of the speech coder and sections 2 and 3 give more detailed descriptions of the speech encoder and the speech decoder, respectively. Section 4 gives a detailed description of how the speech codec bits are mapped into the PDC Full Rate channel coding. The bad frame masking described in section 5 is provided as an example solution; other solutions are allowed.

This specification consists of this document and a soft copy distribution which contain a bit-exact description in the form of a fixed point ANSI C code and a number of test vectors. The ANSI C code defines the speech codec with a set of fixed point basic operators. All implementations shall be bit-exact. The ANSI C code is descibed in more detail in section 6.

In the case of discrepancy between the high level and the bit-exact descriptions, the description in the bit-exact ANSI C program will prevail.

Test vectors to test bitexactness are included in the soft copy distribution of the specification. These testvectors are described in Section 7. The contents of the soft copy distribution is described in section 8.

The user, hereby, acknowledge the following:

- Ericsson does not provide the user with any warranties, whether express or implied, including but not limited to warranties of merchantability and fitness for a particular pupose relating to this specification.

- Ericsson shall have no responsibility whatsoever to indemnify the user against actions or claims for infringment of any intellectual property rights, e.g. patents, copyrights, registered designs or any other rights, by reason of the users's use of this specification.

Table of Contents

List of Figures 5

List of Tables 6

1. General description of the speech codec 7

1.1 Principles of the ACELP encoder 7
1.2 Principles of the ACELP decoder 9
1.3 Audio Interface 9

2. Speech encoding 11

2.1 Pre-processing 11
2.2 Linear prediction analysis and quantization 11
  2.2.1 Windowing and autocorrelation computation 11
  2.2.2 Levinson-Durbin algorithm 12
  2.2.3 LP to LSP conversion 13
  2.2.4 LSP to LP conversion 14
  2.2.5 Quantization of the LSPs 15
  2.2.6 Interpolation of the LSPs 16
2.3 Open-loop pitch analysis 17
2.4 Impulse response computation 17
2.5 Target signal computation 17
2.6 Adaptive codebook search 18
2.7 Algebraic codebook structure and search 19
2.8 Quantization of the gains 21
2.9 Memory update 23
2.10 Vox processing 23
  2.10.1 VOX handler 24
  2.10.2 LSP 24
  2.10.3 Frame energy 25
2.11 Instability protection 27
  2.11.1 Monitoring a Resonance in LPC Spectrum 27
  2.11.2 Pitch Gain Control 28

3. Speech decoding 29

3.1 Decoding and speech synthesis 29
3.2 Instability protection 32
3.3 Post-processing 32
  3.3.1 Adaptive postfiltering 32
  3.3.2 High-pass filtering and up-scaling 33

4. Ordering of the speech encoder bitstream 34

5. Example solution for substitution and muting of lost speech frames (Bad Frame Masking) ... 37

5.1 State Machine 37
5.2 Bad Frame Masking actions 38
  5.2.1 BFI = 0, prevBFI = 0, State = 0 38
  5.2.2 BFI = 0, prevBFI = 1, State = 0 or 5 38
  5.2.3 BFI = 1, prevBFI = 0 or 1, State = 1...6 38
  5.2.4 LTP-lag update 39
  5.2.5 Innovation sequence 39

6. Bit-exact description of the PDC enhanced full rate codec 40

6.1 Description of the constants and variables used in the C code 40

7. Description of the test vectors 45
7.1 Speech encoder input test sequences: 45
7.2 Speech encoder reference output sequences: 47
7.3 Speech decoder input sequences: 47
7.4 Speech decoder reference output sequences: 47
8. Contents of the soft copy distribution 48
9. References 50

List of Figures

Figure 1 State machine for controlling the bad frame substitution 37
Figure 2 Simplified block diagram of the CELP synthesis model 51
Figure 3 Simplified block diagram of the ACELP encoder 52
Figure 4 Simplified block diagram of the ACELP decoder 53
Figure 5 VOX handler timing... 54

List of Tables

Table 1.1 Bit allocation of the 6.7 kbit/s coding algorithm.. 8
Table 2.1 Potential positions of individual pulses in the algebraic codebook. 19
Table 4.1 Ordering of speech parameters for the channel encoder 34
Table 6.1 Codec global constants. 41
Table 6.2 Description of the coder vectors and variables... 42
Table 6.3 Codec fixed tables 43
Table 6.4 Encoder – Decoder file interface 44
Table 6.5 VOX frame types.. 44
Table 7.1 VAD_flag settings for test sequence 6 46

1. General description of the speech codec

1.1 Principles of the ACELP encoder

The codec is based on the code-excited linear predictive (CELP) coding model [1]. A 10th order linear prediction (LP), or short-term, synthesis filter is used which is given by

$$H(z) = \frac{1}{\hat{A}(z)} = \frac{1}{1 + \sum_{i=1}^{m} \hat{a}_i z^{-i}}, \qquad (1.1)$$

where $\hat{a}_i, i = 1, \ldots, m,$ are the (quantized) LP parameters, and $m = 10$ is the predictor order. The long-term, or pitch, synthesis filter is given by

$$\frac{1}{B(z)} = \frac{1}{1 - g_p z^{-T}}, \qquad (1.2)$$

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

The CELP speech synthesis model is shown in Figure 2. In this model, the excitation signal at the input of the short-term LP synthesis filter is constructed by adding two excitation vectors from adaptive and fixed codebooks. The speech is synthesized by feeding the two properly chosen vectors from these codebooks through the short-term synthesis filter. The optimum excitation sequence in a codebook is chosen using an analysis-by-synthesis search procedure in which the error between the original and synthesis speech is minimized according to a perceptually weighted distortion measure.

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

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)}, \qquad (1.3)$$

where $A(z)$ is the unquantized LP filter and $0 < \gamma_2 < \gamma_1 \leq 1$ are the perceptual weighting factors. The values $\gamma_1 = 0.94$ and $\gamma_2 = 0.6$ are used. The weighting filter uses the unquantized LP parameters while the formant synthesis filter uses the quantized ones.

The coder operates on speech frames of 20 ms corresponding to 160 samples at the sampling frequency of 8000 samples/sec. At each 160 speech samples, the speech signal is analyzed to extract the parameters of the CELP model (LP filter, adaptive and innovative codebooks' indices and gains). These parameters are encoded and transmitted. At the decoder, these parameters are decoded and speech is synthesized by filtering the reconstructed excitation signal through the LP synthesis filter.

The signal flow at the encoder is shown in Figure 3. The LP analysis is performed once per frame. The set of LP parameters is converted to line spectrum pairs (LSP) and vector quantized using split vector quantization (SVQ) with 26 bits. The speech frame is divided into 4 subframes of 5 ms each (40 samples). The adaptive and fixed codebook parameters are transmitted every subframe. The quantized and unquantized LP parameters are used for the fourth subframe while in the first, second, and third subframes interpolated LP parameters are used (both quantized and unquantized). An open-loop pitch lag is estimated twice per frame (every 10 ms) based on the perceptually weighted speech signal. Then the following operations are repeated for each subframe:

- The target signal $x(n)$ is computed by filtering the LP residual through the weighted synthesis filter $W(z)H(z)$ with the initial states of the filters having been updated by filtering the error between LP residual and excitation (this is equivalent to the common approach of subtracting the zero-input response of the weighted synthesis filter from the weighted speech signal).

- The impulse response, $h(n)$ of the weighted synthesis filter is computed.

- Closed-loop pitch analysis is then performed (to find the pitch lag and gain), using the target $x(n)$ and impulse response $h(n)$, by searching around the open-loop pitch lag. Fractional pitch with 1/3 resolution is used. The pitch lag is encoded with 8 bits in the first and third subframes and relatively encoded with 4 bits in the second and fourth subframes.

- The target signal $x(n)$ is updated by removing the adaptive codebook contribution (filtered adaptive codevector), and this new target, $x_2(n)$, is used in the fixed algebraic codebook search (to find the optimum innovation). An algebraic codebook with 14 bits is used for the innovative excitation.

- The gains of the adaptive and fixed codebook are vector quantized with 7 bits (with MA prediction applied to the fixed codebook gain).

- Finally, the filters memories are updated (using the determined excitation signal) for finding the target signal in the next subframe.

The bit allocation of the codec is shown in Table 1.1. In each 20 ms speech frame, 134 bits are produced, corresponding to a bit rate of 6.7 kbit/s.

Table 1.1 Bit allocation of the 6.7 kbit/s coding algorithm.

Parameter	**1st & 3rd subframes**	**2nd & 4th subframes**	**total per frame**
LSP set			26
Pitch delay	8	4	24
Algebraic code	14	14	56
Gain VQ	7	7	28
Total			134

1.2 Principles of the ACELP decoder

The signal flow at the decoder is shown in Figure 4. At the decoder, the transmitted indices are extracted from the received bitstream. The indices are decoded to obtain the coder parameters at each transmission frame. These parameters are the LSP vector, the 4 fractional pitch lags, the 4 innovative codevectors, and the 4 sets of vector quantized pitch and innovative gains. The LSP vector is converted to the LP filter coefficients and interpolated to obtain LP filters at each subframe. Then, at each 40-sample subframe:

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

Finally, the reconstructed speech signal is passed through an adaptive postfilter.

1.3 Audio Interface

Whether the input is analog or digital, the signal presented to the input of the speech codec shall be sampled at a rate of 8000 samples per second and shall be quantized to a uniform Pulse Code Modulation, PCM, format with at least 13 bits of dynamic range.

The speech encoder input and the speech decoder output assume a 16-bit integer input normalization with a range from -32,768 through +32,767. If an input audio interface uses a different normalization scheme, then appropriate scaling shall be used.

At the network side, the normalization shall be done such that a sinusoidal signal with a level of -18 dBm0 on the digital trunk network (0 dBr) has an amplitude of 21.17 dB below the maximum undistorted amplitude in the speech codec.

The following examples show how the speech samples may be processed to achieve minimum distortion of the signal.

Example 1: Linear Normalization Scheme

A linear A/D or D/A converter must provide a resolution of at least 13 bits.

The output from the A/D converter must be left-justified in 16-bit words before being input to the speech encoder.

Pseudo-Code (assuming 13-bit A/D converter):

```
AccA = <input sample>   : 13-bit right-justified
AccA = AccA << 3        : 13-bit left-justified
```

At the speech decoder output, the data is rounded and right-justified to fit the resolution of the D/A converter.

Pseudo-Code (assuming 13-bit D/A converter):

```
AccA = <output sample>  : 16-bit linear sample
AccA = AccA + 0x0004    : Add rounding bit
AccA = AccA & 0xFFF8    : 13-bit left-justified
AccA = AccA >> 3        : 13-bit right-justified
```

Example 2: Compressed PCM Normalization Scheme

The conversion from 8-bit μ-Law-compressed PCM into uniform PCM at the speech encoder input is performed in two steps. First, the received data shall be expanded to 14-bit linear data. Second, this data shall be placed left-justified into a 16-bit word.

Pseudo-Code:

```
AccA = <input sample>   : 8-bit u-Law sample
AccA = ulawExpand(AccA) : 14-bit linear sample right-
                          justified
AccA = AccA << 2        : 14-bit left-justified
```

At the speech decoder output, the data is rounded to 14-bit resolution and left-justified prior to feeding the data into the μ-Law compressor.

Pseudo-Code:

```
AccA = <output sample>  : 16-bit linear sample
AccA = AccA + 0x0002    : Add rounding bit
AccA = AccA & 0xFFFC    : 14-bit left-justified
AccA = AccA >> 2        : 14-bit right-justified
AccA = ulawCompr(AccA)  : 8-bit u-Law sample
```

2. Speech encoding

In this section we describe the different functions of the speech encoder represented in the blocks of Figure 3.

2.1 Pre-processing

Two pre-processing functions are applied prior to the encoding process: high-pass filtering and signal down-scaling.

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

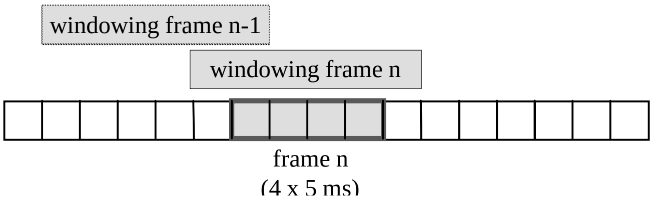
The high-pass filter serves as a precaution against undesired low frequency components. A filter at a cut off frequency of 80 Hz is used, and it is given by

$$H_{h1}(z) = \frac{0.927246093 - 1.8544941 z^{-1} + 0.927246093 z^{-2}}{1 - 1.906005859 z^{-1} + 0.911376953 z^{-2}}. \tag{2.1}$$

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

2.2 Linear prediction analysis and quantization

Short-term prediction, or LP, analysis is performed once per speech frame using the autocorrelation approach with 30 ms asymmetric windows. An overhead of 40 samples (5 ms) is used in the autocorrelation computation. The frame structure is depicted below.



The autocorrelations of windowed speech are converted to the LP coefficients using the Levinson 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 at each subframe).

2.2.1 Windowing and autocorrelation computation

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

$$\begin{aligned} w(n) &= 0.54 - 0.46\cos\left(\frac{2\pi n}{2L_1 - 1}\right), & n &= 0,\ldots,L_1 - 1, \\ &= \cos\left(\frac{2\pi(n - L_1)}{4L_2 - 1}\right), & n &= L_1,\ldots,L_1 + L_2 - 1 \end{aligned} \tag{2.2}$$

where the values $L_1 = 200$ and $L_2 = 40$ are used.

The autocorrelations of the windowed speech $s'(n), n = 0,\ldots,239$, are computed by

$$r(k) = \sum_{n=k}^{239} s'(n)s'(n-k)\ , \qquad k = 0,\ldots,10, \tag{2.3}$$

and a 60 Hz bandwidth expansion is used by lag windowing the autocorrelations using the window [2]

$$w_{lag}(i) = \exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right], \qquad i = 1,\ldots 10, \tag{2.4}$$

where $f_0 = 60$ Hz is the bandwidth expansion and $f_s = 8000$ Hz is the sampling frequency. Further, $r(0)$ is multiplied by the white noise correction factor 1.0001 which is equivalent to adding a noise floor at -40 dB.

2.2.2 Levinson-Durbin algorithm

The modified autocorrelations $r'(0) = 1.0001\, r(0)$ and $r'(k) = r(k)w_{lag}(k),\ k = 1,\ldots 10,$ are used to obtain the LP filter coefficients $a_k, k = 1,\ldots,10,$ by solving the set of equations.

$$\sum_{k=1}^{10} a_k r'(|i-k|) = -r'(i)\ , \qquad i = 1,\ldots,10. \tag{2.5}$$

The set of equations in (2.5) is solved using the Levinson-Durbin algorithm [3]. This algorithm uses the following recursion:

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

The final solution is given as $a_j = a_j^{(10)}, j = 1,\ldots,10$.

The LP filter coefficients are converted to the LSP representation [4] for quantization and interpolation purposes. The conversions to the LSP domain and back to the LP filter domain are described in the next section.

2.2.3 LP to LSP conversion

The LP filter coefficients $a_k, k = 1,\ldots,10$, are converted to the LSP representation for quantization and interpolation purposes. For a 10th order LP filter, the LSPs are defined as the roots of the sum and difference polynomials

$$f_1^{'}(z) = A(z) + z^{-11}A(z^{-1}) \tag{2.6}$$

and

$$f_2^{'}(z) = A(z) - z^{-11}A(z^{-1}) \tag{2.7}$$

respectively. The polynomials $f_1^{'}(z)$ and $f_2^{'}(z)$ are symmetric and antisymmetric, respectively. It can be proven that all roots of these polynomials are on the unit circle and they alternate each other [5]. $f_1^{'}(z)$ has a root $z = -1\,(\omega = \pi)$ and $f_2^{'}(z)$ has a root $z = 1(\omega = 0)$. To eliminate these two roots, we define the new polynomials

$$f_1(z) = f_1^{'}(z) / (1 + z^{-1}) \tag{2.8}$$

and

$$f_2(z) = f_2^{'}(z) / (1 - z^{-1}). \tag{2.9}$$

Each polynomial has 5 conjugate roots on the unit circle $\left(e^{\pm j\omega_i}\right)$, therefore, the polynomials can be written as

$$F_1(z) = \prod_{i=1,3,\ldots,9}\left(1 - 2q_i z^{-1} + z^{-2}\right) \tag{2.10}$$

and

$$F_2(z) = \prod_{i=2,4,\ldots,10}\left(1 - 2q_i z^{-1} + z^{-2}\right) \tag{2.11}$$

where $q_i = \cos(\omega_i)$ with ω_i being the line spectral frequencies (LSF) and they satisfy the ordering property $0 < \omega_1 < \omega_2 < \ldots < \omega_{10} < \pi$. We refer to q_i as the LSPs in the cosine domain.

Since both polynomials $f_1(z)$ and $f_2(z)$ are symmetric only the first 5 coefficients of each polynomial need to be computed. The coefficients of these polynomials are found by the recursive relations (for $i = 0$ to 4)

$$\begin{aligned} f_1(i+1) &= a_{i+1} + a_{m-i} - f_1(i), \\ f_2(i+1) &= a_{i+1} - a_{m-i} + f_2(i). \end{aligned} \tag{2.12}$$

where $m = 10$ is the predictor order.

The LSPs 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 4 times to better track the root. The Chebyshev polynomials are used to evaluate $F_1(z)$ and $F_2(z)$ [6]. In this method the roots are found directly in the cosine domain $\{q_i\}$. The polynomials $F_1(z)$ or $F_2(z)$ evaluated at $z = e^{j\omega}$ can be written as

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

with

$$C(x) = T_5(x) + f(1)T_4(x) + f(2)T_3(x) + f(3)T_2(x) + f(4)T_1(x) + f(5)/2, \tag{2.13}$$

where $T_m(x) = \cos(m\omega)$ is the mth order Chebyshev polynomial, and $f(i),\ i = 1,\ldots,5,$ are the coefficients of either $F_1(z)$ or $F_2(z)$, computed using the equations in (2.12). 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 $b_6 = 0$. The details of the Chebyshev polynomial evaluation method are found in [6].

2.2.4 LSP to LP conversion

Once the LSPs are quantized and interpolated, they are converted back to the LP coefficient domain $\{a_k\}$. The conversion to the LP domain is done as follows. The coefficients of $F_1(z)$ or $F_2(z)$ are found by expanding Equations (2.10) and (2.11) knowing the quantized and interpolated LSPs $q_i,\ i = 1,\ldots,10$. The following recursive relation is used to compute $f_1(i)$

```
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 $1$
        $f_1(j) = f_1(j) - 2q_{2i-1} f_1(j-1) + f_1(j-2)$
    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

$$\begin{aligned} f_1^{'}(i) &= f_1(i) + f_1(i-1), \quad i = 1,\ldots,5, \\ f_2^{'}(i) &= f_2(i) - f_2(i-1), \quad i = 1,\ldots,5, \end{aligned} \tag{2.14}$$

Finally the LP coefficients are found by

$$\begin{aligned} a_i &= 0.5 f_1^{'}(i) + 0.5 f_2^{'}(i), \quad i = 1,\ldots,5, \\ a_i &= 0.5 f_1^{'}(11-i) - 0.5 f_2^{'}(11-i), \quad i = 6,\ldots,10. \end{aligned} \tag{2.15}$$

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

2.2.5 Quantization of the LSPs

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

$$f_i = \frac{f_S}{2\pi} \arccos(q_i), \quad i = 1,\ldots,10,$$

where f_i are the LSFs in Hz [0,4000] and $f_s = 8000$ is the sampling frequency. The LSF vector is given by $\mathbf{f}^t = [f_1\ f_2\ \cdots\ f_{10}]$, with t denoting transpose.

A 1st order MA prediction is applied, and the residual LSF vector is quantized using SVQ. The prediction and quantization are performed as follows. Let $\mathbf{z}(n)$ denote the mean-removed LSF vector at frame n. The prediction residual vector $\mathbf{r}(n)$ is given by

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

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

$$p_j(n) = \alpha_j \hat{r}_j(n-1), \qquad j = 1,\dots,10 \tag{2.16}$$

where $\hat{r}_j(n-1)$ are the quantized residual vector at the past frame and α_j are the MA prediction coefficients corresponding to the *j*th element of the LSF vector.

The LSF residual vector $\mathbf{r}$ is quantized using SVQ. The vector is split into 3 subvectors of dimensions 3, 3, and 4. The 3 subvectors are quantized with 8, 9, and 9 bits, respectively.

A weighted LSP distortion measure is used in the quantization process. In general, for an input LSP vector $\mathbf{f}$ and a quantized vector at index k, $\hat{\mathbf{f}}^k$, the quantization is performed by finding the index k which minimizes

$$E_k = \sum_{i=1}^{10} \left[f_i w_i - \hat{f}_i^k w_i \right]^2 . \tag{2.17}$$

The weighting factors $w_i, i = 1,\dots,10,$, are given by

$$w_i = \begin{cases} 3.347 - \dfrac{1.547}{450} d_i , & \text{for } d_i < 450 \\ 1.8 - \dfrac{0.8}{1050}(d_i - 450), & \text{otherwise} \end{cases}$$

where $d_i = f_{i+1} - f_{i-1}$ with $f_0 = 0$ and $f_{11} = 4000$ Hz. In the quantization of each subvector, the weighting coefficients are used with their corresponding LSFs.

2.2.6 Interpolation of the LSPs

The set of quantized (and unquantized) LP parameters is used for the fourth subframe whereas the first, second, and third subframes use a linear interpolation of the parameters in the adjacent frames. The interpolation is performed on the LSPs in the $\mathbf{q}$ domain. Let $\hat{\mathbf{q}}_4^{(n)}$ be the LSP vector at the 4th subframe of the frame, and $\hat{\mathbf{q}}_4^{(n-1)}$ the LSP vector at the 4th subframe of the past frame $n-1$. The interpolated LSP vectors at the 1st, 2nd, and 3rd subframes are given by

$$\begin{aligned} \hat{\mathbf{q}}_1^{(n)} &= 0.75\hat{\mathbf{q}}_4^{(n-1)} + 0.25\hat{\mathbf{q}}_4^{(n)}, \\ \hat{\mathbf{q}}_2^{(n)} &= 0.5\hat{\mathbf{q}}_4^{(n-1)} + 0.5\hat{\mathbf{q}}_4^{(n)}, \\ \hat{\mathbf{q}}_3^{(n)} &= 0.25\hat{\mathbf{q}}_4^{(n-1)} + 0.75\hat{\mathbf{q}}_4^{(n)}. \end{aligned}$$

The same formula is used for interpolation of the unquantized LSPs. The interpolated LSP vectors are used to compute a different LP filter at each subframe (both quantized and unquantized) using the LSP to LP conversion method described in Section 2.2.4.

2.3 Open-loop pitch analysis

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

Open-loop pitch estimation is based on the weighted speech signal $s_w(n)$ which is obtained by filtering the input speech signal through the weighting filter $W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)}$. That is, in a subframe of size L, the weighted speech is given by

$$s_w(n) = s(n) + \sum_{i=1}^{10} a_i \gamma_1^i s(n-i) - \sum_{i=1}^{10} a_i \gamma_2^i s_w(n-i), \qquad n = 0,\ldots,L-1. \tag{2.18}$$

Open-loop pitch analysis is performed as follows. In the first step, 3 maxima of the correlation

$$C_k = \sum_{n=0}^{79} s_w(n) s_w(n-k) \tag{2.19}$$

are found in the three delay ranges, k = 20...39, 40...79, and 80...143, respectively. Negative indices of s and s_w refer to the previous frame. The retained maxima for each range i, C_{k_i}, $i = 1$ to 3, are normalized by dividing by $\sqrt{\sum_n s_w^2(n-k_i)}$, $i = 1$ to 3, respectively. The normalized maxima and corresponding delays are denoted by (R_i, k_i) $i = 1$ to 3. The winner, T_{op}, among the three normalized correlations is selected by favoring the delays in the lower ranges. That is, k_i is selected if $R_i > 0.85 R_{i+1}$. This procedure of dividing the delay range into 3 sections and favoring the lower sections is used to avoid choosing pitch multiples.

2.4 Impulse response computation

The impulse response, $h(n)$, of the weighted synthesis filter $H(z)W(z) = A(z/\gamma_1)/\left[\hat{A}(z)A(z/\gamma_2)\right]$ is computed each subframe. This impulse response is needed for the search of adaptive and fixed codebooks. The impulse response $h(n)$ is computed by filtering the vector of coefficients of the filter $A(z/\gamma_1)$ extended by zeros through the two filters $1/\hat{A}(z)$ and $1/A(z/\gamma_2)$.

2.5 Target signal computation

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

An equivalent procedure for computing the target signal, which is used in this codec, is the filtering of the LP residual signal $r(n)$ through the combination of synthesis filter $1/\hat{A}(z)$ and the weighting filter $A(z/\gamma_1)/A(z/\gamma_2)$. After determining the excitation for the subframe, the initial states of these filters are updated by filtering the difference between the LP residual and excitation. The memory update of these filters is explained in Section 2.9.

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 section. The LP residual is given by

$$r(n) = s(n) + \sum_{i=1}^{10} \hat{a}_i s(n-i), \qquad n = 0, \ldots, 39. \tag{2.20}$$

2.6 Adaptive codebook search

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

The adaptive codebook parameters (or pitch parameters) are the delay and gain of the pitch filter. In the 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.

In the first and third subframes, a fractional pitch delay is used with resolutions 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 and fourth subframes, integer pitch resolution is used in the range $\left[T_1-5, T_1+4\right]$, where T_1 is the nearest integer to the fractional pitch lag of the previous (1st or 3rd) subframe, bounded by 20…143. Additionally, a fractional resolution of $\frac{1}{3}$ is used in the range $\left[T_1-1\frac{2}{3}, T_1+\frac{2}{3}\right]$.

Closed-loop pitch analysis is performed around the open-loop pitch estimates on a subframe basis. In the first (and third) subframe the range $T_{op} \pm 3$, bounded by 20...143, is searched. For the other subframes, closed-loop pitch analysis is performed around the integer pitch selected in the previous subframe, as described above. The pitch delay is encoded with 8 bits in the first and third subframes and the relative delay of the other subframes is encoded with 4 bits.

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

$$T_k = \frac{\sum_{n=0}^{39} x(n) y_k(n)}{\sqrt{\sum_{n=0}^{39} y_k(n) y_k(n)}}, \tag{2.21}$$

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

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

$$y_k(n) = y_{k-1}(n-1) + u(-k)h(n) \tag{2.22}$$

where $u(n),\ n = -(143+11),\ldots,39,$ is the excitation buffer. Note that in search stage, the samples $u(n),\ n = 0,\ldots,39$, are not known, and they are needed for pitch delays less than 40. To simplify the search, the LP residual is copied to $u(n)$ in order to make the relation in Equation (2.22) valid for all delays.

Once the optimum integer pitch delay is determined, the fractions from $\frac{-2}{3}$ to $\frac{2}{3}$ with a step of $\frac{1}{3}$ around that integer are tested. The fractional pitch search is performed by interpolating the normalized correlation in Equation (2.21) and searching for its maximum. Once the fractional pitch lag is determined, the adaptive codebook vector $v(n)$ is computed by interpolating the past excitation signal $u(n)$ at the given phase (fraction). The interpolation is performed using two FIR filters (Hamming windowed sinc functions); one for interpolating the term in Equation (2.21) with the sinc truncated at ± 11 and the other for interpolating the past excitation with the sinc truncated at ± 29. The filters have their cut-off frequency (-3 dB) at 3600 Hz in the oversampled domain.

The adaptive codebook gain is then found by

$$g_p = \frac{\sum_{n=0}^{39} x(n)y(n)}{\sum_{n=0}^{39} y(n)y(n)}\ , \qquad \text{bounded by} \quad 0 \le g_p \le 1.2, \tag{2.23}$$

where $y(n) = v(n) * h(n)$ is the filtered adaptive codebook vector (zero-state response of $H(z)W(z)$ to $v(n)$).

2.7 Algebraic codebook structure and search

The codebook structure is based on interleaved single-pulse permutation (ISPP) design. In this codebook, the innovation vector contains 3 non-zero pulses. All pulses can have the amplitudes $+1$ or -1. The 40 positions in a subframe are divided into 3 tracks, where each track contains one pulse, as shown in Table 2.1.

Table 2.1 Potential positions of individual pulses in the algebraic codebook.

Pulse	Positions
i_0	0, 5, 10, 15, 20, 25, 30, 35
i_1	1, 6, 11, 16, 21, 26, 31, 36 3, 8, 13, 18, 23, 28, 33, 38
i_2	2, 7, 12, 17, 22, 27, 32, 37 4, 9, 14, 19, 24, 29, 34, 39

The first pulse position is coded with 3 bits and the second and third pulse positions with 4 bits, and the sign of the each pulse is encoded with 1 bit. This makes total of 14 bits per subframe.

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

$$x_2(n) = x(n) - g_p y(n)\,, \qquad n = 0,\ldots,39, \tag{2.24}$$

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

The matrix $\mathbf{H}$ is defined as the lower triangular Toeplitz convolution matrix with diagonal $h(0)$ and lower diagonals $h(1),\ldots,\mathrm{h}(39)$, and $\mathbf{d} = \mathbf{H}^t\mathbf{x}_2$ is the correlation between the target signal $x_2(n)$ and the impulse response $h(n)$, and $\Phi = \mathbf{H}^t\mathbf{H}$ is the matrix of correlations of $h(n)$.

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

$$d(n) = \sum_{i=n}^{39} x_2(i)h(i-n), \qquad n = 0,\ldots 39, \tag{2.25}$$

and the elements of the symmetric matrix Φ are computed by

$$\phi(i,j) = \sum_{n=j}^{39} h(n-i)h(n-j), \qquad i = 0, \ldots, 39, \quad j = i, \ldots, 39. \tag{2.26}$$

If c_k is the algebraic codevector at index k, then the algebraic codebook is searched by maximizing the term

$$\mathrm{T}_k = \frac{(\mathrm{C}_k)^2}{\mathrm{E}_k} = \frac{(\mathbf{d}^t\mathbf{c}_k)^2}{\mathbf{c}_k^t \Phi c_k}. \tag{2.27}$$

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

The algebraic structure of the codebooks allows for very fast search procedures since the innovation vector $\mathbf{c}_k$ contains only a few nonzero pulses. The correlation in the numerator of Equation (2.27) is given by

$$\mathrm{C} = \sum_{i=0}^{N_p-1} a_i d(m_i) \tag{2.28}$$

where m_i is the position of the i th pulse, a_i is its amplitude, and $N_p = 3$ is the number of pulses.

The energy in the denominator of Equation (2.27) is given by

$$\mathrm{E} = \sum_{i=0}^{N_p-1} \phi(m_i, m_i) + 2 \sum_{i=0}^{N_p-2} \sum_{j=i+1}^{N_p-1} a_i a_j \phi(m_i, m_j) \tag{2.29}$$

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 that 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,

$$\phi'(i,j) = sign[d(i)] sign[d(j)] \phi(i,j), \quad i = 0,\ldots,39, j = i+1,\ldots,39. \tag{2.30}$$

The correlation in Equation (2.28) is now given by

$$\mathrm{C} = \sum_{i=0}^{N_p-1} |d(m_i)| \tag{2.31}$$

and the energy in Equation (2.29) is given by

$$\mathrm{E} = \sum_{i=0}^{N_p-1} \phi'(m_i, m_i) + 2 \sum_{i=0}^{N_p-2} \sum_{j=i+1}^{N_p-1} \phi'(m_i, m_j) \tag{2.32}$$

Having preset the pulse amplitudes, as explained above, the optimal pulse positions are determined using an efficient non-exhaustive analysis-by-synthesis search technique. In this technique, the term in Equation (2.27) is tested for a small percentage of position combinations.

A special feature incorporated in the codebook is that the selected codevector is filtered through an adaptive prefilter $F(z)$ which enhances special spectral components in order to improve the synthesis speech quality. Here the filter $F(z) = 1/(1-\beta z^{-T})$ is used, where T is the integer part of the pitch lag and β is a pitch gain. β is given by the quantized pitch gain, $\hat{g}_p$, from the previous subframe bounded by [0.0,0.8]. Note that prior to the codebook search, the impulse response $h(n)$ must include the prefilter $F(z)$. That is, $h(n) \leftarrow h(n) + \beta h(n-T)$.

2.8 Quantization of the gains

The adaptive codebook gain (pitch gain) and the fixed (algebraic) codebook gain are vector quantized using a 7-bit codebook.

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

$$E(n) = 10\log\left(\frac{1}{N}g_c^2\sum_{i=0}^{N-1}c^2(i)\right) - \overline{E} \tag{2.33}$$

where $N = 40$ is the subframe size, $c(i)$ is the fixed codebook excitation, and $\overline{E} = 28.75$ is the mean of the innovation energy. The predicted energy is given by

$$\tilde{E}(n) = \sum_{i=1}^{4} b_i \hat{R}(n-i) \tag{2.34}$$

where $[b_1\ b_2\ b_3\ b_4] = [0.68\ 0.58\ 0.34\ 0.19]$ are the MA prediction coefficients, and $\hat{R}(k)$ is the quantized energy prediction error at subframe k. The predicted energy is used to compute a predicted fixed-codebook gain $g_c^{'}$ as in Equation (2.33) (by substituting $E(n)$ by $\tilde{E}(n)$ and g_c by $g_c^{'}$). This is done as follows. First, the mean innovation energy is found by

$$E_i = 10\log\left(\frac{1}{N}\sum_{i=0}^{N-1}c^2(i)\right) \tag{2.35}$$

and then the predicted gain $g_c^{'}$ is found by

$$g_c^{'} = 10^{0.05(\tilde{E}(n)+\overline{E}-E_i)}. \tag{2.36}$$

A correction factor between the gain g_c and the estimated one $g_c^{'}$ is given by

$$\gamma = g_c / g_c^{'}. \tag{2.37}$$

Note that the prediction error is given by

$$R(n) = E(n) - \tilde{E}(n) = 20\ \log\ (\gamma). \tag{2.38}$$

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

$$E = x^t x + g_p^2 y^t y + g_c^2 z^t z - 2g_p x^t y - 2g_c x^t z + 2g_p g_c y^t z, \tag{2.39}$$

where the x is the target vector, y is the filtered adaptive codebook vector, and z is the filtered fixed codebook vector. Each gain vector in the codebook also has an element representing the quantized energy prediction error. The one associated with the chosen gains is used to update $\hat{R}(n)$. $\hat{R}(n)$ is related to the variable `past_qua_en` in the C-code.

2.9 Memory update

An update of the states of the synthesis and weighting filters is needed in order to compute the target signal in the next subframe.

After the two gains have been quantized, the excitation signal, $u(n)$, in the present subframe is found by

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n), \qquad n = 0,\ldots 39, \tag{2.40}$$

where $\hat{g}_p$ and $\hat{g}_c$ are the quantized adaptive and fixed codebook gains, respectively, $v(n)$ the adaptive codebook vector (interpolated past excitation), and $c(n)$ is the fixed codebook vector (algebraic code including pitch sharpening). The states of the filters can be updated by filtering the signal $r(n) - u(n)$ (difference between residual and excitation) through the filters $1/\hat{A}(z)$ and $A(z/\gamma_1)/A(z/\gamma_2)$ for the 40 sample subframe and saving the states of the filters. This would require 3 filterings. A simpler approach which requires only one filtering is as follows. The local synthesis speech, $\hat{s}(n)$, is computed by filtering the excitation signal through $1/\hat{A}(z)$. The output of the filter due to the input $r(n) - u(n)$ is equivalent to $e(n) = s(n) - \hat{s}(n)$. So the states of the synthesis filter $1/\hat{A}(z)$ are given by $e(n),\ n = 30,\ldots,39$. 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 $e_w(n)$. However, the signal $e_w(n)$ can be equivalently found by

$$e_w(n) = x(n) - \hat{g}_p y(n) - \hat{g}_c z(n). \tag{2.41}$$

Since the signals $x(n), y(n)$, and $z(n)$ are available, the states of the weighting filter are updated by computing $e_w(n)$ as in Equation (2.41) for $n = 30,\ldots,39$. This saves two filterings.

2.10 Vox processing

According to RCR STD 27F section 4.1.20 a mobile station shall have a VOX function. The VOX function includes a Voice Activity Detector (VAD) that detects voice presence/absence. The voice activity detector is not specified in RCR STD 27F and it is not specified in this specification.

In the case of voice absence, the speech encoder in the mobile station shall encode the information that is needed, denoted PST1, for generating comfort noise in the base station. This information consists of three parameters, the same 26-bit LSP set as is used when voice is present, a 3-bit index for the LSP prediction initialization and a 6-bit energy parameter. The PST1 frame contains the same number of bits as a normal speech frame and it is protected with the normal full rate channel coding. The 26-bit LSP set is transmitted in the same bit positions as in the case of voice present. The 6 bit frame energy is placed in the 6 MSB:s of the 1:st subframe adaptive codebook index parameter. The 3-bits representing the initial value for the LSP prediction are placed in the 3 MSB:s of the 3:rd subframe adaptive codebook index parameter. All other bits are set to zero.

2.10.1 VOX handler

The VOX handler controls the generation of the VOX parameters in the PST1 frame. It also controls the transmission of the POST and PRE frames defined in RCR STD 27F as well as controlling the TCH transmission on and off. Input to the VOX handler is the voice presence/absence decision from the voice activity detector.

The purpose of the VOX handler is to control the timing of the calculation of the PST1 parameters and the synchronization of this timing with the POST (PST0 and PST1) and PRE frames.

A signal sp_flag is generated from the voice presence/absence signal. The sp_flag signal controls the VOX handler state machine. The transmitted frame type (Normal Speech, PST0, PST1, PRE or No Transmission) is controlled by the state machine.

When the voice activity detection signals a voice absence segment the sp_flag stays in the voice mode (sp_flag = 1) for an additional 8 frames to ensure that the PST1 parameters are derived from a voice absence segment. After this a PST0 and PST1 frame is generated once every second as long as the voice activity detector signals voice absence. Figure 5 shows the timing diagram of the sp_flag signal and the associated frame types.

In order to improve performance in conditions where the voice activity detector toggles between voice presence and voice absence the following additions have been made:

- If a speech present segment is shorter than 20 frames the sp_flag will not stay in the voice mode for the additional 8 frames but it will directly switch to the no voice mode (sp_flag = 0). In this case no new parameter set for the PST1 frame will be calculated, instead the parameters from the previous PST1 frame will be transmitted.

- A second set of PST0 and PST1 frames are transmitted 8 frames after the first PST0 and PST1 frames when a new voice absence segment starts.

The update rate of the PST0 and PST1 frames in the fixed point C-code is once every 50:th frame, i.e. once every second. The update rate may be changed according to 4.1.20 in RCR STD 27F.

2.10.2 LSP

The comfort noise parameters to be encoded into a PST1 frame are calculated over 8 consecutive frames marked with voice absence, as follows:

The averaged LSF parameter vector of the frame *I*, $\mathbf{f}^{mean}(i)$, shall be computed according to the equation:

$$\mathbf{f}^{mean}(i)=\frac{1}{N}\sum_{n=0}^{N-1}\mathbf{f}(i-n) \tag{1}$$

where:

$\mathbf{f}(i)$ is the unquantized LSF parameter vector of the current frame i ($n = 0$);

$\mathbf{f}(i-n)$ is the unquantized LSF parameter vector of the previous frames ($n = 1,\ldots, 7$);

i is the frame index;

N is the number of averaging frames = 8;

The averaged LSF parameter vector $\mathbf{f}^{mean}(i)$ of the frame i is quantized using the same quantization tables as for the non-averaged LSF parameter vectors in ordinary speech encoding mode but the quantization algorithm is slightly modified. The parameter $\hat{r}_j(n-1)$ in eq 2.16 is initialized from a lookup table containing 8 vectors with indices corresponding to subvectors of the SVQ. The predictor coefficients, α_j in eq 2.16, are all set to one. The quantization process includes a search for the best of these 8 vectors. The vector which yields the lowest prediction error is selected.

After the above step the LSF parameter encoding procedure is performed as normal. A 3-bit index is used for the selected $\hat{r}_j(n-1)$ initialization vector.

2.10.3 Frame energy

The frame energy is computed for each frame marked with voice absence according to the equation :

$$en_{\log}(i)=\frac{1}{2}\log_2\left(\frac{1}{N_1}\sum_{n=0}^{N_1-1}s_n^2\right)$$

where $N_1 = 160$ is the number of samples in a frame. The parameter $en_{\log}(i)$ represent the logarithmic RMS level of the input signal of frame i.

The averaged logarithmic energy is computed from

$$en_{mean}=\frac{1}{N}\sum_{n=0}^{N}en_{\log}(i-n)$$

where $N = 8$ is the number of frames. The averaged logarithmic energy is quantized by a 6-bit algorithmic quantizer according to:

$$en_{index}=\max(0,\min(63,\lfloor(en_{mean}+2.5)*4+0.5\rfloor))$$

The index 0 is assumed to represent a level of zero.

The quantized value of en_{mean} is used for initialization of all four old values of the quantized energy prediction error in subframe $k, \hat{R}(k)$, see eqation 2.34. The initialization is done every PST1 frame.

2.11 Instability protection

The purpose of the instability protection is to prevent such divergence to occur in the contents of the adaptive codebook memories in the encoder and decoder that would make the decoder unstable. Typically, this divergence is due to channel errors. The speech codec allows pitch gains greater than unity which may result the pitch filter to become temporarily unstable. In the encoder, the stability is controlled by the analysis-by-synthesis loop. However, incorrect contents of adaptive codebook memory in the decoder may result the decoder to become unstable for highly correlated continuous (stationary) signals which keep the pitch gain continuously in a high value. A typical signal having these characteristics is a sinusoidal tone or a combination of tones.

The instability protection detects possible problem conditions by monitoring the occurrence of resonance in LPC-spectrum in the encoder. If a resonance is found in a several consecutive frames, periodicity detection is activated. The periodicity detection calculates the average pitch gain of the present and seven previous subframes, and limits the present pitch gain below unity if a threshold is exceeded. This guarantees the stability of the pitch synthesis filter.

In the speech decoder, an additional safety feature is used. Possible overflows in the LPC synthesis filter are monitored and, if necessary, the adaptive codebook memory is scaled down by a factor of 4.

2.11.1 Monitoring a Resonance in LPC Spectrum

The monitoring of a resonance signal is made in LSP domain using the unquantized LSPs $q_i, i = 1, \ldots, 10$. The LSPs are available after LP to LSP conversion presented in section 2.2.3. The values of LSPs are in range -1.0 to 1.0 corresponding to frequency range 4000 to 0 Hz. In the following, all the values referring to LSPs or distances between adjacent LSPs are presented in Q15 format as they are presented in the fixed point C-source code.

The algorithm utilizes the fact that LSPs are closely located around a peak in the spectrum. First, two minimum distances between adjacent LSPs are searched in two sections:

$$\begin{aligned} dist_1 &= \min(q_i - q_{i+1}), \quad i = 4, \ldots, 8 \\ dist_2 &= \min(q_i - q_{i+1}), \quad i = 2, 3 \end{aligned}$$

Either of the minimum distances conditions must be fulfilled to classify the frame to resonance frame. The counter calculating consecutive resonance frame is updated according to the decision.

```
$if\,(dist_1 < TH_1) \quad OR \quad if\,(dist_2 < TH_2)$
    $counter = counter + 1$
$else$
    $counter = 0$
```

A fixed threshold distance, $TH_1 = 1500$, is used for the first section. For the second section, the threshold distance depends on the location of q_2.

$$TH_2 = \begin{cases} 600, & q_2 > 32000 \\ 800, & 30500 < q_2 \leq 32000 \\ 1100, & otherwise \end{cases}$$

12 consecutive resonance frames are needed to set the *LSP_flag* which indicates possible problem condition. Otherwise, *LSP_flag* is cleared and the instability protection algorithm is completed

```
$if\ (counter \geq 12)$
    $counter = 12$
    $LSP\_flag = 1$
$else$
    $LSP\_flag = 0$
```

2.11.2 Pitch Gain Control

In case that 12 consecutive resonance frames are found (*LSP_flag* = 1) and the present pitch gain exceeds the gain threshold $GP_{th} = 0.95$ (15565 in Q14), the average pitch gain is calculated over the present unquantized pitch gain and quantized pitch gains of the seven previous subframes

$$GP_{ave} = mean\left\{ g_p(n), \hat{g}_p(n-1), \hat{g}_p(n-2), \ldots, \hat{g}_p(n-7) \right\},$$

where the *n* refers to the present subframe. The average pitch gain calculation can be done after calculating the pitch gain of the present subframe as presented in section 2.6. If the average pitch gain exceeds the GP_{th} the gain is limited to the threshold value and the *GpC_flag* is set to indicate the limitation.

```
$if\ (GP_{ave} > GP_{th})$
    $g_p = GP_{th}$
    $GpC\_flag = 1$
$else$
    $GpC\_flag = 0$
```

If the *GpC_flag* is set, the limitation of the pitch gain is taken into consideration in the gain Vector Quantization (section 2.8) where the codebook search range is limited to include only pitch gain values less than GP_{th}.

```
$if\ (GpC\_flag = 1)$
    $VQsearch\_range = 96$
$else$
    $VQsearch\_range = 128$
```

After the gain Vector Quantization, the buffer of past quantized pitch gains is updated always regardless the value of *GpC_flag*.

$$\hat{g}_p(n-i) = \hat{g}_p(n-i+1), \quad i = 7, \ldots, 1$$

3. Speech decoding

The function of the decoder consists of decoding the transmitted parameters (LP parameters, adaptive codebook vectors, adaptive codebook gains, fixed codebook vectors, fixed codebook gains) and performing synthesis to obtain the reconstructed speech. The reconstructed speech is then postfiltered and upscaled. The signal flow at the decoder is shown in Figure 4.

3.1 Decoding and speech synthesis

The decoding process is performed in the following order:

Decoding of LP filter parameters: The received indices of LSP quantization are used to reconstruct the quantized LSP vector. The interpolation described in Section 2.2.6 is performed to obtain 4 interpolated LSP vectors (corresponding to 4 subframes). For each subframe, the interpolated LSP vector is converted to the LP filter coefficient domain a_k, which is used for synthesizing the reconstructed speech in the subframe.

The following steps are repeated for each subframe:

1. **Decoding of the adaptive codebook vector:** The received pitch index (adaptive codebook index) is used to find the integer and fractional parts of the pitch lag. The adaptive codebook vector $v(n)$ is found by interpolating the past excitation $u(n)$ (at the pitch delay) using the FIR filter described in Section 2.6.
2. **Decoding of the innovative vector:** The received algebraic codebook index is used to extract the positions and amplitudes (signs) of the excitation pulses and to find the algebraic codevector $c(n)$. If the integer part of the pitch lag is less than the subframe size 40, the pitch sharpening procedure is applied which translates into modifying $c(n)$ by $c(n) \leftarrow c(n) + \beta c(n-T)$, where β is the decoded pitch gain from the previous subframe, $\hat{g}_p$, bounded by [0.0,0.8].
3. **Decoding of the adaptive and innovative codebook gains:** The received index gives the fixed codebook gain correction factor $\hat{\gamma}$. The estimated fixed codebook gain $g_c^{'}$ is found as described in Section 2.8. First, the predicted energy for every subframe n is found by

$$\tilde{E}(n) = \sum_{i=1}^{4} b_i \hat{R}(n-i) \tag{3.1}$$

and then the mean innovation energy is found by

$$E_i = 10 \log \left(\frac{1}{N} \sum_{i=0}^{N-1} c^2(i) \right) \tag{3.2}$$

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

$$g_c^{'} = 10^{0.05(\tilde{E}(n) + \bar{E} - E_i)} \tag{3.3}$$

The quantized fixed codebook gain is given by

$$\hat{g}_c = \hat{\gamma} g_c' . \tag{3.4}$$

The received index is used to readily find the quantified adaptive codebook gain, $\hat{g}_p$ from the quantization table.

4. **Smoothing of the fixed codebook gain:** An adaptive smoothing of the fixed codebook gain is performed to avoid unnatural fluctuations in the energy contour. The smoothing is based on a measure of the stationarity of the short-term spectrum in the **q** domain. The smoothing strength is computed from this measure. An averaged **q**-value is computed for each frame n by:

$$\bar{\mathbf{q}}(n) = 0.84 \cdot \bar{\mathbf{q}}(n-1) + 0.16 \cdot \hat{\mathbf{q}}_4(n) \tag{3.5}$$

For each subframe m, a difference measure between the averaged vector and the quantized and interpolated vector is computed by:

$$diff_m = \sum_j \sum_m \frac{\left| \bar{q}^{(j)}(n) - \hat{q}_m^{(j)}(n) \right|}{\bar{q}^{(j)}(n)} , \tag{3.6}$$

where j runs over the 10 LSPs. Furthermore, a smoothing factor, k_m, is computed by:

$$k_m = \min(K_2, \max(0, diff_m - K_1)) / K_2 , \tag{3.7}$$

where the constants are set to $K_1 = 0.4$ and $K_2 = 0.25$. A hangover period of 40 subframes is used where the k_m-value is set 1.0 if the $diff_m$ has been above 0.65 for 10 consecutive frames. A value of 1.0 corresponds to no smoothing. An averaged fixed codebook gain value is computed for each subframe by:

$$\bar{g}(m) = \frac{1}{5} \sum_{i=0}^{4} \hat{g}_c(m-i) . \tag{3.8}$$

The fixed codebook gain used for synthesis is now replaced by a smoothed value given by:

$$\hat{g}_c := \hat{g}_c \cdot k_m + \bar{g}_c \cdot (1 - k_m) . \tag{3.9}$$

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

```
IF      $\hat{g}_p < 0.6$      THEN     $impNr = 0$ ;
ELSE  IF   $\hat{g}_p < 0.9$   THEN   $impNr = 1$ ;
ELSE  $impNr = 2$ ;
```

Detection of onset is done by comparing the fixed codebook gain to the previous codebook gain. If the current value is more than twice the previous value an onset is detected.

If an onset is not detected and $impNr = 0$, the median filtered value of the current and the previous four adaptive codebook gains are computed. If this value is less than 0.6, $impNr$ is set to 0.

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

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

6. **Computing the reconstructed speech:** The following steps are for n = 0, ..., 39. The total excitation is constructed by:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n), \qquad . \tag{3.10}$$

Before the speech synthesis, a post-processing of excitation elements is performed. This means that the total excitation is modified by emphasizing the contribution of the adaptive codebook vector:

$$\hat{u}(n) = \begin{cases} u(n) + 0.5\beta_c \hat{g}_p v(n), & \hat{g}_p > 0.5 \\ u(n), & \hat{g}_p \le 0.5 \end{cases} \tag{3.11}$$

where β_c is the decoded pitch gain of the current subframe, $\hat{g}_p$, bounded by [0.0,0.8]. Adaptive gain control (AGC) is used to compensate for the gain difference between the nonemphasized excitation *u(n)* and emphasized excitation *û(n)* The gain scaling factor γ for the emphasized excitation is computed by:

$$\gamma = \begin{cases} \sqrt{\dfrac{\sum_{n=0}^{39} u^2(n)}{\sum_{n=0}^{39} \hat{u}^2(n)}}, & \hat{g}_p > 0.5 \\ 1.0 & \hat{g}_p \le 0.5 \end{cases} \tag{3.12}$$

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

$$\hat{u}'(n) = \hat{u}(n)\gamma \tag{3.13}$$

The reconstructed speech for the subframe of size 40 is given by

$$\hat{s}(n) = \hat{u}'(n) - \sum_{i=1}^{10} \hat{a}_i \hat{s}(n-i), \tag{3.14}$$

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

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

3.2 Instability protection

In the speech decoder, an additional protection is used by monitoring overflows in the synthesis filter (Equation 3.14). In case of overflow(s), the whole adaptive codebook memory, $v(n), n = -(143+11),\ldots,39$ in Equation 3.10, is scaled down by a factor of 4, and the the synthesis filtering is repeated using this down-scaled memory. If the down-scaling has occurred, the post-processing of the excitation signal in Equations 3.11 - 3.13 is by-passed, and the synthesis filtering in Equation 3.14 is performed for the down-scaled excitation $u(n)$ instead of the emphasized excitation $\hat{u}'(n)$.

3.3 Post-processing

Post-processing consists of three functions: adaptive postfiltering, high-pass filtering and signal up-scaling.

3.3.1 Adaptive postfiltering

The adaptive postfilter is the cascade of two filters: a formant postfilter, and a tilt compensation filter. The postfilter is updated every subframe of 5 ms.

The formant postfilter is given by

$$H_f(z) = \frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)}, \tag{3.15}$$

where $\hat{A}(z)$ is the received quantized (and interpolated) LP inverse filter (LP analysis is not performed at the decoder), and the factors γ_n and γ_d control the amount of the formant postfiltering.

Finally, the filter $H_t(z)$ compensates for the tilt in the formant postfilter $H_f(z)$ and is given by

$$H_t(z) = (1 - \mu z^{-1}), \tag{3.16}$$

where $\mu = \gamma_t k_1$ is a tilt factor, with k_1 being the first reflection coefficient calculated on the truncated impulse response, $h_f(n)$, of the filter $\hat{A}(z/\gamma_n)/\hat{A}(z/\gamma_d)$. k_1 is given by

$$k_1 = \frac{r_h(1)}{r_h(0)}; \qquad r_h(i) = \sum_{j=0}^{L_h-i-1} h_f(j)h_f(j+i), \qquad (L_h = 22, \quad i = 0, 1). \tag{3.17}$$

The postfiltering process is performed as follows. First, the synthesis speech $\hat{s}(n)$ is inverse filtered through $\hat{A}(z/\gamma_n)$ to produce the residual signal $\hat{r}(n)$. The signal $\hat{r}(n)$ is filtered by the synthesis filter $1/\hat{A}(z/\gamma_d)$. Finally, the signal at the output of the synthesis filter $1/\hat{A}(z/\gamma_d)$ is passed to the tilt compensation filter $h_t(z)$ resulting in the postfiltered synthesis speech signal $s_f(n)$.

AGC is used to compensate for the gain difference between the synthesis speech signal $\hat{s}(n)$ and the postfiltered signal $s_f(n)$. The gain scaling factor γ for the present subframe is computed by

$$\gamma = \sqrt{\frac{\sum_{n=0}^{39} \hat{s}^2(n)}{\sum_{n=0}^{39} s_f^2(n)}}. \tag{3.18}$$

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

$$s'(n) = \beta(n) s_f(n) , \tag{3.19}$$

where $\beta(n)$ is updated in sample-by-sample basis and given by

$$\beta(n) = \alpha\beta(n-1) + (1-\alpha)\gamma . \tag{3.20}$$

where α is a AGC factor given by 0.9. The adaptive postfiltering factors are given by: $\gamma_n = 0.55$, $\gamma_d = 0.7$ and $\gamma_t = 0.8$.

3.3.2 High-pass filtering and up-scaling

Two post-processing functions are applied after the decoding process: high-pass filtering and signal up-scaling.

The high-pass filter serves as a precaution against undesired low frequency components. A filter at a cut off frequency of 60 Hz is used, and it is given by

$$H_{h2}(z) = \frac{0.939819335 - 1.879638672 z^{-1} + 0.939819335 z^{-2}}{1 - 1.933105469 z^{-1} + 0.935913085 z^{-2}}. \tag{3.21}$$

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

4. Ordering of the speech encoder bitstream

The speech codec is using the same channel coding scheme as the PDC Full Rate codec. Table 4.1 describes how the speech codec bits are mapped into the PDC Full Rate codec bits (b0 is LSB).

Table 4.1 Ordering of speech parameters for the channel encoder

PDC-FR Parameter	**PDC Bit**	**PDC-EFR Parameter**	**EFR Bit**	**CHC Class**
R0	b0	Position of pulse 3 of 1st subframe	b0	2
R0	b1	Index of 3rd LSF subvector	b6	1
R0	b2	Index of 1st LSF subvector	b3	1 CRC
R0	b3	Index of 1st LSF subvector	b1	1 CRC
R0	b4	Index of 1st LSF subvector	b0	1 CRC
LPC1	b0	VQ-gain index 2nd subframe	b0	2
LPC1	b1	Adaptive codebook index 3rd subframe	b0	1
LPC1	b2	Index of 2nd LSF subvector	b1	1 CRC
LPC1	b3	Index of 1st LSF subvector	b6	1 CRC
LPC1	b4	Index of 1st LSF subvector	b2	1 CRC
LPC2	b0	VQ-gain index 4th subframe	b0	2
LPC2	b1	VQ-gain index 3rd subframe	b0	2
LPC2	b2	Adaptive codebook index 4th subframe	b0	1
LPC2	b3	Index of 2nd LSF subvector	b0	1 CRC
LPC2	b4	Index of 1st LSF subvector	b4	1 CRC
LPC3	b0	Sign of pulse 1 of 1st subframe	b0	2
LPC3	b1	VQ-gain index 1st subframe	b5	1
LPC3	b2	Index of 2nd LSF subvector	b2	1 CRC
LPC3	b3	Index of 1st LSF subvector	b5	1 CRC
LPC4	b0	Sign of pulse 3 of 1st subframe	b0	2
LPC4	b1	Sign of pulse 2 of 1st subframe	b0	2
LPC4	b2	Index of 3rd LSF subvector	b7	1
LPC4	b3	Index of 1st LSF subvector	b7	1 CRC
LPC5	b0	Sign of pulse 3 of 2nd subframe	b0	2
LPC5	b1	Sign of pulse 2 of 2nd subframe	b0	2
LPC5	b2	Sign of pulse 1 of 2nd subframe	b0	2
LPC5	b3	Index of 2nd LSF subvector	b3	1 CRC
LPC6	b0	Sign of pulse 2 of 3rd subframe	b0	2
LPC6	b1	Sign of pulse 1 of 3rd subframe	b0	2
LPC6	b2	VQ-gain index 3rd subframe	b1	1
LPC7	b0	Sign of pulse 2 of 4th subframe	b0	2
LPC7	b1	Sign of pulse 1 of 4th subframe	b0	2
LPC7	b2	Sign of pulse 3 of 3rd subframe	b0	2
LPC8	b0	Position of pulse 1 of 1st subframe	b1	2
LPC8	b1	Position of pulse 1 of 1st subframe	b0	2
LPC8	b2	Sign of pulse 3 of 4th subframe	b0	2
LPC9	b0	Position of pulse 2 of 1st subframe	b1	2
LPC9	b1	Position of pulse 2 of 1st subframe	b0	2
LPC9	b2	Position of pulse 1 of 1st subframe	b2	2
LPC10	b0	Position of pulse 2 of 1st subframe	b3	2
LPC10	b1	Position of pulse 2 of 1st subframe	b2	2
SOFINT	b0	Index of 3rd LSF subvector	b2	1
LAG_1	b0	Adaptive codebook index 2nd subframe	b2	1 CRC
LAG_1	b1	VQ-gain index 3rd subframe	b3	1 CRC
LAG_1	b2	VQ-gain index 1st subframe	b3	1 CRC
LAG_1	b3	Adaptive codebook index 3rd subframe	b7	1 CRC
LAG_1	b4	Adaptive codebook index 1st subframe	b4	1 CRC

LAG_1	b5	Index of 2nd LSF subvector	b8	1 CRC
LAG_1	b6	Index of 2nd LSF subvector	b4	1 CRC
CODE_1	b0	Position of pulse 2 of 2nd subframe	b1	2
CODE_1	b1	Position of pulse 2 of 2nd subframe	b0	2
CODE_1	b2	Position of pulse 1 of 2nd subframe	b2	2
CODE_1	b3	Position of pulse 1 of 2nd subframe	b1	2
CODE_1	b4	Position of pulse 1 of 2nd subframe	b0	2
CODE_1	b5	Position of pulse 3 of 1st subframe	b3	2
CODE_1	b6	Position of pulse 3 of 1st subframe	b2	2
CODE_1	b7	Position of pulse 3 of 1st subframe	b1	2
CODE_1	b8	Position of pulse 2 of 2nd subframe	b2	2
GSP0_1	b0	VQ-gain index 3rd subframe	b4	1
GSP0_1	b1	VQ-gain index 4th subframe	b1	1
GSP0_1	b2	VQ-gain index 4th subframe	b4	1
GSP0_1	b3	Adaptive codebook index 2nd subframe	b0	1
GSP0_1	b4	Index of 3rd LSF subvector	b3	1
GSP0_1	b5	Index of 3rd LSF subvector	b4	1
GSP0_1	b6	Adaptive codebook index 3rd subframe	b3	1 CRC
LAG_2	b0	Adaptive codebook index 2nd subframe	b1	1 CRC
LAG_2	b1	VQ-gain index 4th subframe	b6	1 CRC
LAG_2	b2	VQ-gain index 2nd subframe	b6	1 CRC
LAG_2	b3	Adaptive codebook index 3rd subframe	b6	1 CRC
LAG_2	b4	Adaptive codebook index 1st subframe	b3	1 CRC
LAG_2	b5	Adaptive codebook index 1st subframe	b7	1 CRC
LAG_2	b6	Index of 2nd LSF subvector	b5	1 CRC
CODE_2	b0	Position of pulse 1 of 3rd subframe	b2	2
CODE_2	b1	Position of pulse 1 of 3rd subframe	b1	2
CODE_2	b2	Position of pulse 1 of 3rd subframe	b0	2
CODE_2	b3	Position of pulse 3 of 2nd subframe	b3	2
CODE_2	b4	Position of pulse 3 of 2nd subframe	b2	2
CODE_2	b5	Position of pulse 3 of 2nd subframe	b1	2
CODE_2	b6	Position of pulse 3 of 2nd subframe	b0	2
CODE_2	b7	Position of pulse 2 of 2nd subframe	b3	2
CODE_2	b8	Position of pulse 2 of 3rd subframe	b0	2
GSP0_2	b0	VQ-gain index 2nd subframe	b2	1
GSP0_2	b1	VQ-gain index 2nd subframe	b5	1
GSP0_2	b2	VQ-gain index 1st subframe	b1	1
GSP0_2	b3	VQ-gain index 1st subframe	b4	1
GSP0_2	b4	Index of 3rd LSF subvector	b8	1
GSP0_2	b5	Index of 3rd LSF subvector	b0	1
GSP0_2	b6	Adaptive codebook index 3rd subframe	b2	1 CRC
LAG_3	b0	Adaptive codebook index 4th subframe	b3	1 CRC
LAG_3	b1	VQ-gain index 4th subframe	b3	1 CRC
LAG_3	b2	VQ-gain index 2nd subframe	b3	1 CRC
LAG_3	b3	Adaptive codebook index 3rd subframe	b5	1 CRC
LAG_3	b4	Adaptive codebook index 1st subframe	b2	1 CRC
LAG_3	b5	Adaptive codebook index 1st subframe	b6	1 CRC
LAG_3	b6	Index of 2nd LSF subvector	b6	1 CRC
CODE_3	b0	Position of pulse 1 of 4th subframe	b0	2
CODE_3	b1	Position of pulse 3 of 3rd subframe	b3	2
CODE_3	b2	Position of pulse 3 of 3rd subframe	b2	2
CODE_3	b3	Position of pulse 3 of 3rd subframe	b1	2
CODE_3	b4	Position of pulse 3 of 3rd subframe	b0	2
CODE_3	b5	Position of pulse 2 of 3rd subframe	b3	2
CODE_3	b6	Position of pulse 2 of 3rd subframe	b2	2
CODE_3	b7	Position of pulse 2 of 3rd subframe	b1	2
CODE_3	b8	Position of pulse 1 of 4th subframe	b1	2
GSP0_3	b0	VQ-gain index 3rd subframe	b2	1

GSP0_3	b1	VQ-gain index 3rd subframe	b5	1
GSP0_3	b2	VQ-gain index 4th subframe	b2	1
GSP0_3	b3	VQ-gain index 4th subframe	b5	1
GSP0_3	b4	Adaptive codebook index 1st subframe	b0	1
GSP0_3	b5	Index of 3rd LSF subvector	b5	1
GSP0_3	b6	Adaptive codebook index 3rd subframe	b1	1 CRC
LAG_4	b0	Adaptive codebook index 4th subframe	b2	1 CRC
LAG_4	b1	Adaptive codebook index 2nd subframe	b3	1 CRC
LAG_4	b2	VQ-gain index 3rd subframe	b6	1 CRC
LAG_4	b3	Adaptive codebook index 3rd subframe	b4	1 CRC
LAG_4	b4	Adaptive codebook index 1st subframe	b1	1 CRC
LAG_4	b5	Adaptive codebook index 1st subframe	b5	1 CRC
LAG_4	b6	Index of 2nd LSF subvector	b7	1 CRC
CODE_4	b0	Position of pulse 3 of 4th subframe	b2	2
CODE_4	b1	Position of pulse 3 of 4th subframe	b1	2
CODE_4	b2	Position of pulse 3 of 4th subframe	b0	2
CODE_4	b3	Position of pulse 2 of 4th subframe	b3	2
CODE_4	b4	Position of pulse 2 of 4th subframe	b2	2
CODE_4	b5	Position of pulse 2 of 4th subframe	b1	2
CODE_4	b6	Position of pulse 2 of 4th subframe	b0	2
CODE_4	b7	Position of pulse 1 of 4th subframe	b2	2
CODE_4	b8	Position of pulse 3 of 4th subframe	b3	2
GSP0_4	b0	VQ-gain index 2nd subframe	b1	1
GSP0_4	b1	VQ-gain index 2nd subframe	b4	1
GSP0_4	b2	VQ-gain index 1st subframe	b0	1
GSP0_4	b3	VQ-gain index 1st subframe	b2	1
GSP0_4	b4	Adaptive codebook index 4th subframe	b1	1
GSP0_4	b5	Index of 3rd LSF subvector	b1	1
GSP0_4	b6	VQ-gain index 1st subframe	b6	1 CRC

5. Example solution for substitution and muting of lost speech frames (Bad Frame Masking)

In the following sections **BFI = 1** (**BFI** = Bad Frame Indicator) is equivalent to a CRC error as it is specified in section 5.1.3.3 of the RCR STD 27F.

5.1 State Machine

This example solution for substitution and muting is based on a state machine with seven states. The system starts in state 0. Each time a bad frame is detected, the state counter is incremented by one and is saturated when it reaches 6. Each time a good speech frame is detected, the state counter is reset to zero, except when in state 6, where the state counter is set to 5. The state indicates the quality of the channel: the higher the value of the state counter, the worse the channel quality is. The control flow of the state machine can be described with the following C-code (**BFI** = bad frame indicator, **State** = state variable):

```
if(BFI != 0 )
State = State + 1;
else if(State == 6)
State = 5;
else
State = 0;
if(State > 6 )
State = 6;
```

In addition to this state machine, the **BFI** from the previous frame is checked (**prevBFI**). The processing depends on the value of the **State**-variable. In states 0 and 5, the processing depends also on the two flags **BFI** and **prevBFI**.

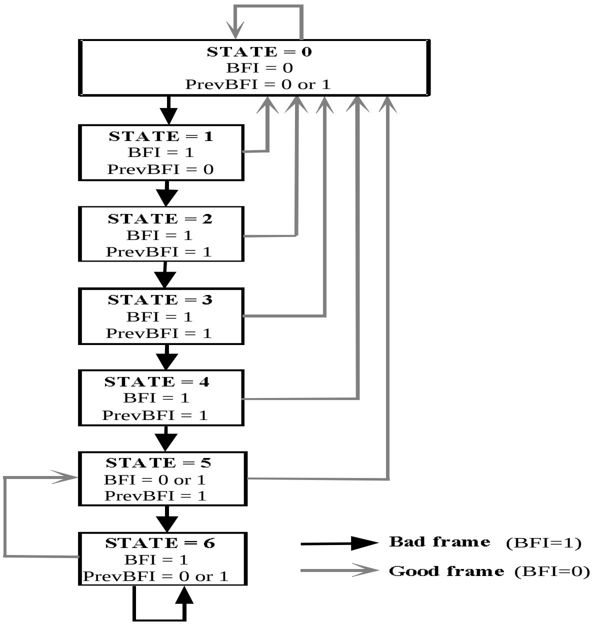


Figure 1 State machine for controlling the bad frame substitution

5.2 Bad Frame Masking actions

5.2.1 BFI = 0, prevBFI = 0, State = 0

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

5.2.2 BFI = 0, prevBFI = 1, State = 0 or 5

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

$$g^p = \begin{cases} g^p, g^p \le g^p(-1) \\ g^p(-1), g^p > g^p(-1) \end{cases}$$

where g^p = current decoded LTP-gain, $g^p(-1)$ = LTP-gain used for the last good subframe (BFI = 0), and

$$g^c = \begin{cases} g^c, & g^c \le g^c(-1) \\ g^c(-1), & g^c > g^c(-1) \end{cases}$$

where g^c = current decoded fixed codebook-gain and $g^c(-1)$ = fixed codebook gain used for the last good subframe (BFI = 0).

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

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

An error is detected in the received speech frame and the substitution and muting procedure is started. The LTP-gain and fixed codebook gain are replaced by attenuated values from the previous subframes:

$$g^p = \begin{cases} P(state)\ \ g^p(-1), & g^p(-1) \le median5(g^p(-1),...,g^p(-5)) \\ P(state)\ \ median5(g^p(-1),...,g^p(-5)), & g^p(-1) > median5(g^p(-1),...,g^p(-5)) \end{cases}$$

where g^p = current decoded LTP-gain, $g^p(-1), \dots, g^p(-n)$ = LTP-gains used for the last n subframes, *median5()* = 5-point median operation, *P(state)* = attenuation factor (*P(1)* = 0.98, *P(2)* = 0.98, *P(3)* = 0.8, *P(4)* = 0.3, *P(5)* = 0.2, *P(6)* = 0.2), *state* = state number, and

$$g^c = \begin{cases} C(state)\ \ g^c(-1), & g^c(-1) \le median5(g^c(-1),...,g^c(-5)) \\ C(state)\ \ median5(g^c(-1),...,g^c(-5)), & g^c(-1) > median5(g^c(-1),...,g^c(-5)) \end{cases}$$

where g^c = current decoded fixed codebook gain, $g^c(-1), \ldots, g^c(-n)$ = fixed codebook gains used for the last n subframes, *median5()* = 5-point median operation, *C(state)* = attenuation factor (*C(1)* = 0.98, *C(2)* = 0.98, *C(3)* = 0.98, *C(4)* = 0.98, *C(5)* = 0.98, *C(6)* = 0.7), and *state* = state number.

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

$$ener(0) = \frac{1}{4}\sum_{i=1}^{4} ener(-i)$$

The past LSFs are used by shifting their values towards their mean:

$$lsf\_q(i) = \alpha \cdot past\_lsf\_q(i) + (1-\alpha) \cdot mean\_lsf(i), i = 0...9$$

where α = 0.95, *lsf_q* is the set of LSF-values for the current frame, *past_lsf_q* is *lsf_q* from the previous frame, and *mean_lsf* is the average LSF-vector.

5.2.4 LTP-lag update

The LTP-lag values are replaced by the lag value from the previous subframe incremented by one. The value in the first subframe of the first frame in a sequence of bad frames is the last correctly received value incremented by one.

5.2.5 Innovation sequence

The received fixed codebook innovation pulses from the erroneous frame are always used in the state they were received.

6. Bit-exact description of the PDC enhanced full rate codec

The various components of the 6.7 kbit/s PDC enhanced full rate codec are described in the form of a fixed-point bit-exact ANSI C-code. This software consists of the speech codec and bad frame handler functions. It also includes the VOX processing in the speech encoder.

The fixed point C-code uses a binary file interface between encoder and decoder. Each encoded frame consists of 134 speech encoder parameter bits and 3 bits of additional information. Each bit is stored in the file as a 16-bit integer. An integer value of 1 indicates a logical one and the value 0 indicates a logical zero. The bits are ordered according to table 4.1. Table 6.4 and 6.5 describes the file interface in more detail.

Note that the decoder part of the C code does not use the two bits associated with VOX and that the BFI output from the encoder is set to one if the VOX handler is in the no transmission state. Thus, the decoder can not be used to test the function of the VOX mode. Instead test vector 6 could be used to check the correctness of an implementation of the VOX functionality in the encoder.

In the fixed-point ANSI C-code, all the computations are performed using a predefined set of basic operators.

Two types of variables are used in the fixed-point implementation. These two types are signed integers in 2's complement representation, defined by:

`Word16`	16-bitvariables
`Word32`	32-bit variables

6.1 Description of the constants and variables used in the C code

The ANSI C code simulation of the codec is, to a large extent, self-documented. However, a description of the variables and constants used in the code is given to facilitate the understanding of the code. The fixed-point precision (in terms of Q format, double precision (DP), or normalized precision) of the vectors and variables is given, along with the vectors dimensions and constant values.

Table 6.1 gives the coder global constants and table 6.2 describes the variables and vectors used in the encoder routine with their precision. Table 6.3 describes the fixed tables in the codec.

Table 6.1 Codec global constants.

Parameter	Value	Description
L_TOTAL	240	Size of speech buffer
L_WINDOW	240	Size of LP analysis window
L_FRAME	160	Size of speech frame
L_FRAME_BY2	80	Half the speech frame size
L_SUBFR	40	Size of subframe
M	10	Order of LP analysis
MP1	11	**M+1**
AZ_SIZE	44	**4*M+4**
GAMMA 1	30802	Weighting factor in numerator of W(z) (0.94 in Q15)
GAMMA 2	19661	Weighting factor in denominator of W(z) (0.6 in Q15)
PIT_MAX	143	Maximum pitch lag
PIT_MIN	20	Minimum pitch lag
L_INTER10	10	Order of sinc filter for interpolating The excitations is **2*L_INTER10*3+1**
PRM_SIZE	21	Size of vector of analysis parameters
SERIAL_SIZE	137	Number of speech bits + bfi + vox
GAMMA 3	18022	Formant postfilter factor (numerator) (0.55 in Q15)
GAMMA 4	22938	Formant postfilter factor (denominator) (0.70 in Q15)
MU	26214	Tilt compensation filter factor (0.8 in Q15)
GP_CLIP	15565	Pitch gain clipping (0.95 in Q14)
N_FRAME	7	Old pitch gains in average calculation
AGC_FAC	29491	Automatic gain control factor (0.9 in Q15)

1 **Table 6.2 Description of the coder vectors and variables**

Parameter	Precision	Initial value	Description
speech [-40..199]	Q0	0	Speech buffer
wsp [-143..-1]	Q0	0	Weighted speech buffer
wsp [0..79]	Q0	---	Weighted speech buffer
exc [-(143+11)..-1]	Q0	0	LP excitation
exc [0..39]	Q0	---	LP excitation
lsp_old [0..9]	Q15	30000, 26000, 21000, 15000, 8000, 0, -8000, -15000, -21000, -26000	LSP vector in past frame
lsp_old_q [0..9]	Q15	Same as for lsp_old	Quantized LSP vector in past frame
mem_syn [0..9]	Q0	0	Memory of synthesis filter
mem_w [0..9]	Q0	0	Memory of weighting filter (applied to input)
mem_w0 [0..9]	Q0	0	Memory of weighting filter (applied to error)
error [-10..-1]	Q0	0	Error signal (input minus synthesis)
error [0..39]	Q0	---	Error signal (input minus synthesis)
r_l [0..10] & r_h [0..10]	Normalized DP	--- ---	Correlations of windowed speech (low and hi)
A_t [11x4]	Q12	---	LP filter coefficients in 4 subframes
Aq_t [11x4]	Q12	---	Quantized LP filter coefficients in 4 subframes
Ap1 [0..10]	Q12	---	LP coefficients with spectral expansion
Ap2 [0..10]	Q12	---	LP coefficients with spectral expansion
lsp_new [0..9]	Q15	---	LSP vector in 4th subframe
lsp_new_q [0..9]	Q15	---	Quantized LSP vector in 4th subframe
code [0..39]	Q13	---	Fixed codebook excitation vector
h1 [-40..-1]	Q12	0	Impulse response of weighted synthesis filter
h1 [0..39]	Q12	---	Impulse response of weighted synthesis filter
xn [0..39]	Q0	---	Target vector in pitch search
xn2 [0..39]	Q0	---	Target vector in algebraic codebook search
dn [0..39]	scaled max < 8192	---	Backward filtered target vector
y1 [0..39]	Q0	---	Filtered adaptive codebook vector
y2 [0..39]	Q12	---	Filtered fixed codebook vector
zero [-11..39]		0	Zero vector
gain_pit	Q14	---	Adaptive codebook gain
gain_code	Q1	---	Algebraic codebook gain
past_qua_en [0..3]	Q10	-14336	Past quantized energy prediction errors

1

Table 6.3 Codec fixed tables

Parameter	Vector size	Precision	Description
grid [ ]	61	Q15	Grid points at which Chebyshew polynomials are evaluated
lag_h [ ] and lag_l [ ]	10	DP[1]	Higher and lower parts of the lag window table
F_gamma1	10	Q15	Spectral expansion factors
F_gamma2	10	Q15	Spectral expansion factors
F_gamma3	10	Q15	Spectral expansion factors (post filter)
F_gamma4	10	Q15	Spectral expansion factors (post filter)
window_200_40 [ ]	240	Q15	LP analysis window
table [ ] in Lsf_lsp ( )	65	Q15	Table to compute cos(x) in Lsf_lsp ( )
slope [ ] in Lsp_lsf ( )	64	Q12	Table to compute acos(x) in LSP_lsf ( )
table [ ] in Inv_sqrt ( )	49		Table used in inverse square root computation
table [ ] in Log2 ( )	33		Table used in base 2 logarithm computation
table [ ] in Pow2 ( )	33		Table used in 2 to the power computation
mean_lsf [ ]	10	Q15	LSF means in normalized frequency [0.0, 0.5]
pred_fac[ ]	10	Q15	MA coefficients for LSF prediction
dico1_lsf [ ]	256 x 3	Q15	1st LSF quantizer in normalized frequency [0.0, 0.5]
dico2_lsf [ ]	512 x 3	Q15	2nd LSF quantizer in normalized frequency [0.0, 0.5]
dico3_lsf [ ]	512 x 4	Q15	3rd LSF quantizer in normalized frequency [0.0, 0.5]
table_gain [ ]	128*3		Quantization table of
		Q14	Adaptive codebook gain
		Q12	Fixed codebook gain and
		Q10	Energy prediction error
inter_3[ ] in Interpol_3( )	13	Q15	Interpolation filter coefficients in Interpol_3 ( )
inter_3[ ] in Pred_lt_3( )	31	Q15	Interpolation filter coefficients in Pred_lt_3 ( )
ph_imp_low	40	Q15	Anti-sparseness filter coefficients
ph_imp_mid	40	Q15	Anti-sparseness filter coefficients
REORD_efr2fr	134	Q0	Reordering table
b [ ] in Pre_Process()	3	Q12	HP filter coefficients (numerator)
a [ ] in Pre_Process()	3	Q12	HP filter coefficients (denominator)
b [ ] in Post_Process()	3	Q13	HP filter coefficients (numerator)
past_rq_init	80	Q15	LSP prediction initialization
a [ ] in Post_Process()	3	Q13	HP filter coefficients (denominator)
bitno [ ]	21	Q0	Number of bits corresponding to transmitted parameters

---

[1] Special double precision format, see oper_32b.c listing.

Table 6.4 Encoder – Decoder file interface

Word number	Name	Description
0	BFI	Bad Frame Indicator, indicates CRC error
1-2	VOX	VOX frame type indicator
3-136	Parameters	Speech coder parameters ordered according to table 4.1

Table 6.5 VOX frame types

Bit 1	Bit 2	Description
0	0	Normal speech frame or PST1 frame
0	1	PRE frame
1	0	PST0 frame
1	1	No transmission

7. Description of the test vectors

The purpose of the test vectors is to check the bit-exactness of the speech codec implementation. All test vectors are in MS-DOS binary format. The low and high bytes must be swapped in order to use them in other platforms (for example HP-UNIX or SUN). The input files have 13-bit resolution: The 3 LSBs of the 16-bit samples are masked to zero.

7.1 Speech encoder input test sequences:

tstseq1.inp - test vector for LPC quantization codebooks

- A test vector which selects every index in each of the three SVQ codebooks (LSF residual) at least once.
- The file is composed of various kinds of input material including speech, noise, and synthetic signals
- File length: 1060 frames

tstseq2.inp - test vector for LTP (adaptive) codebook

- A test vector which selects every possible long term prediction filter delay (lag). The file contains pulse like harmonic signal and noise.
- File length: 966 frames

tstseq3.inp - test vector containing speech signal

- A test vector containing female speech samples with flat characteristics.
- File length: 360 frames

tstseq4.inp - test vector containing low level speech signal

- A test vector containing low level (-42 dB) male speech samples with flat characteristics.
- File length: 296 frames

tstseq5.inp - test vector for instability protection

- A test vector containing a sequence of DTMF tones as well as single-frequency and swept sinusoidal signals.
- File length: 350 frames.

tstseq6.inp - test vector for VOX parameters

- The file is composed of different types of input material, mainly noise.
- File length: 450 frames.

The implementation of the VOX parameter encoding is tested using this input sequence while forcing the speech encoder to VOX mode by setting **VAD_flag** according to table 7.1. **VAD_flag** = 1 indicates voice present, **VAD_flag** = 0 indicates voice absent.

Table 7.1 VAD_flag settings for test sequence 6

Frame number	Setting
0 - 98	1
99 - 138	0
139	1
140 – 168	0
169 – 170	1
171 – 198	0
199 – 201	1
202 – 248	0
249 – 298	1
299 – 348	0
349	1
350 – 378	0
379 – 380	1
381 – 418	0
419 – 421	1
422 - 449	0

7.2 Speech encoder reference output sequences:

tstseq1.cod
tstseq2.cod
tstseq3.cod
tstseq4.cod
tstseq5.cod
tstseq6.cod

The implementation of the speech encoder can be verified by encoding the speech encoder input sequences and comparing the output with the corresponding speech encoder reference output sequences.

7.3 Speech decoder input sequences:

The speech encoder reference output sequences (tstseq1.cod, tstseq2.cod, tstseq3.cod, tstseq4.cod, tstseq5.cod) are used as the speech decoder input test sequences. Note that test sequence 6 should not be used as input to the speech decoder. The only purpose of test sequence 6 is to verify the VOX part of the speech encoder.

7.4 Speech decoder reference output sequences:

tstseq1.out
tstseq2.out
tstseq3.out
tstseq4.out
tstseq5.out

The implementation of the speech decoder can be verified by decoding the speech decoder input sequences and comparing the output with the corresponding speech decoder reference output sequences.

8. Contents of the soft copy distribution

The soft copy distribution consists of a CD-ROM in MS-DOS format. The detailed contents of the CD-ROM is listed below.

Directory src:

agc.c
autocorr.c
az_lsp.c
basic_op.h
basicop2.c
bits2prm.c
c3_14pf.c
c_g_aver.c
cnst.h
cod_6k7.c
codec.h
coder.c
coder.rsp
convolve.c
copy.c
d3_14pf.c
d_plsf_3.c
dec_6k7.c
dec_gain.c
dec_lag3.c
decoder.c
decoder.rsp
ec_gains.c
gmed5.c
grid.tab
int_lpc.c
int_lsf.c
inv_sqrt.c
inv_sqrt.tab
lag_wind.c
lag_wind.tab
levinson.c
log2.c
log2.tab
lsp_avg.c
lsp_az.c
lsp_lsf.c
lsp_lsf.tab
makefile.pc
makefile.sun
n_stack.h
oper_32b.c
oper_32b.h
ph_disp.c
ph_disp.tab
pitch.c
post_pro.c
postfilt.c
pow2.c
pow2.tab
pre_proc.c
pred_lt3.c
preemph.c
prm2bits.c
proc_hd.c
q_plsf_3.c
q_plsf_3.tab
qua_gain.c
qua_gain.tab
readme.txt
refl_a.c
reord_ch.c
reord_ch.tab
reorder.c
residu.c
set_zero.c
sig_proc.h
sqrt_s.c
syn_filt.c
typedef.h
vox_hand.c
weight_a.c
weight_f.c
window.tab

Directory testinp:

tstseq1.inp
tstseq2.inp
tstseq3.inp
tstseq4.inp
tstseq5.inp
tstseq6.inp

Directory testcod:

tstseq1.cod
tstseq2.cod
tstseq3.cod
tstseq4.cod
tstseq5.cod
tstseq6.cod

Directory testout:

tstseq1.out
tstseq2.out
tstseq3.out
tstseq4.out
tstseq5.out

9. References

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

[2] Y. Tohkura and F. Itakura, “Spectral smoothing technique in PARCOR speech analysis-synthesis,” *IEEE Trans. on ASSP*, vol. 26, no. 6, pp. 587-596, Dec. 1978.

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

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

[5] F.K. Soong and B.H. Juang, “Line spectrum pair (LSP) and speech data compression”, *Proc. ICASSP’84*, pp. 1.10.1-1.10.4, 1984.

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

[7] C. Laflamme, J-P. Adoul, R. Salami, S. Morissette, and P. Mabilleau, “16 kpbs wideband speech coding technique based on algebraic CELP” *Proc. ICASSP’91*, pp. 13-16.

[8] Association of Radio Industries and Businesses, *“RCR STD-27F*”, 1997

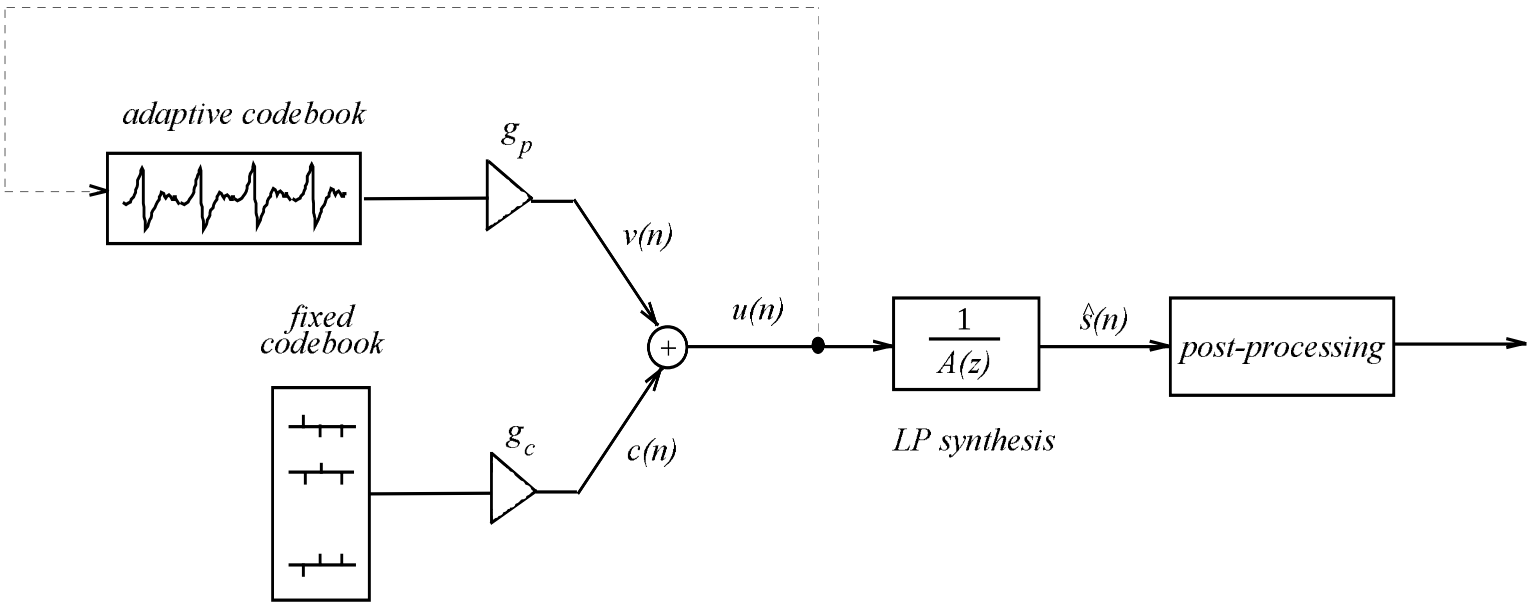


Figure 2 Simplified block diagram of the CELP synthesis model

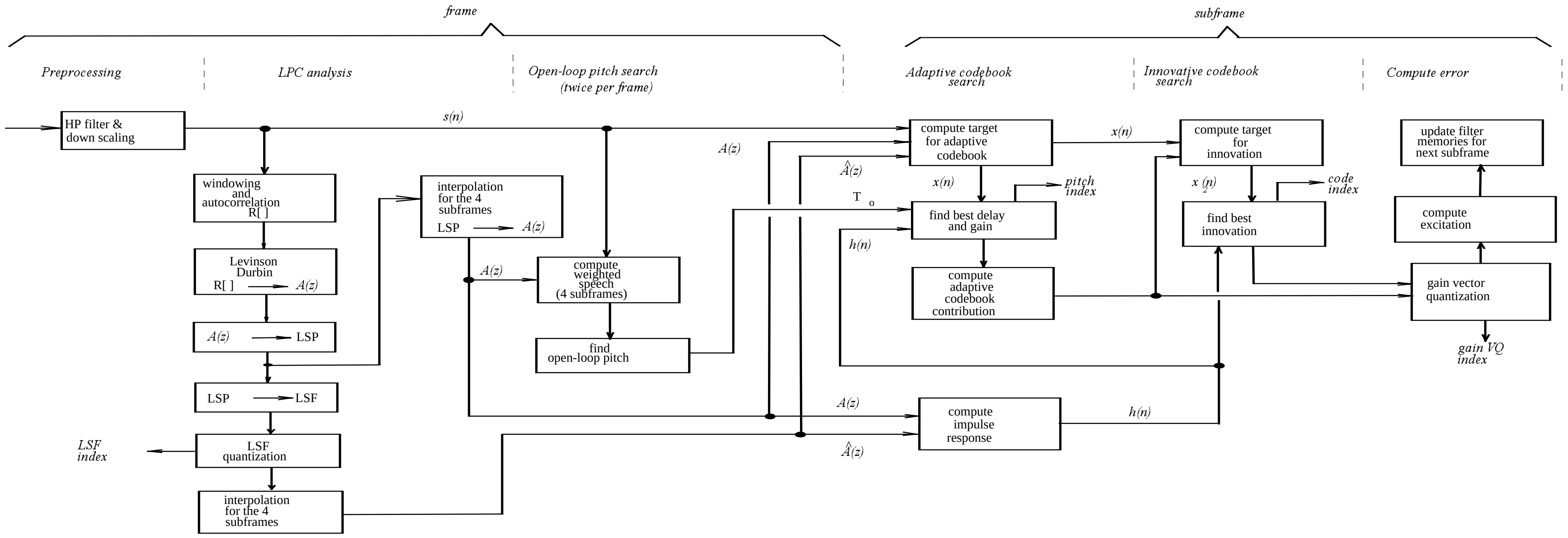


Figure 3 Simplified block diagram of the ACELP encoder

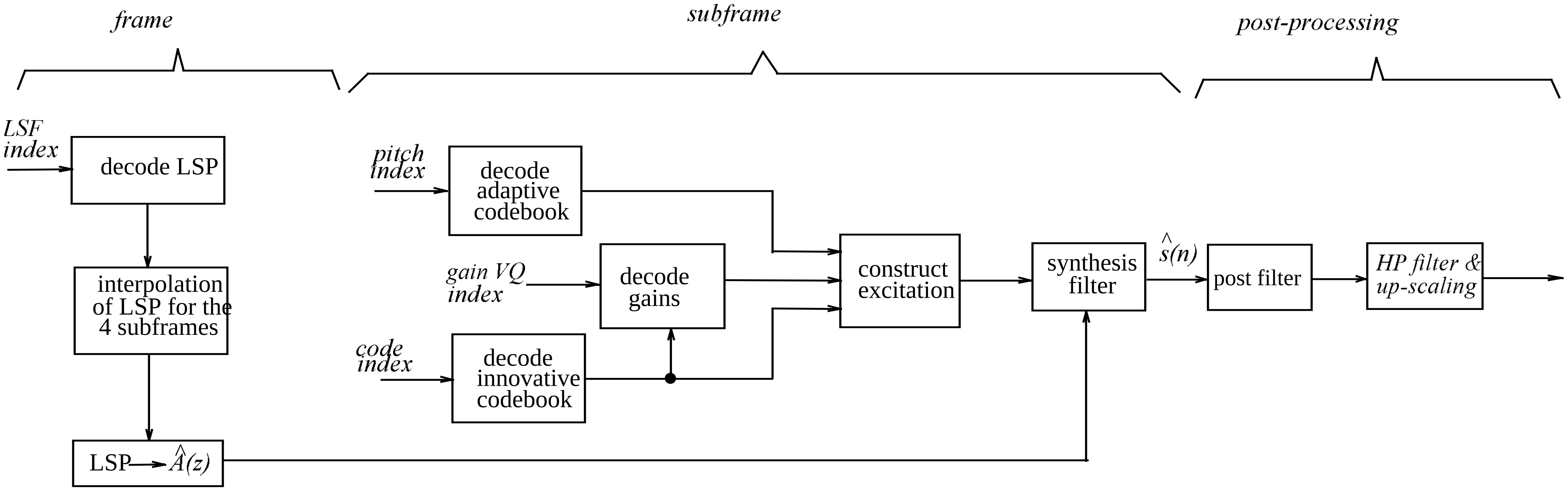
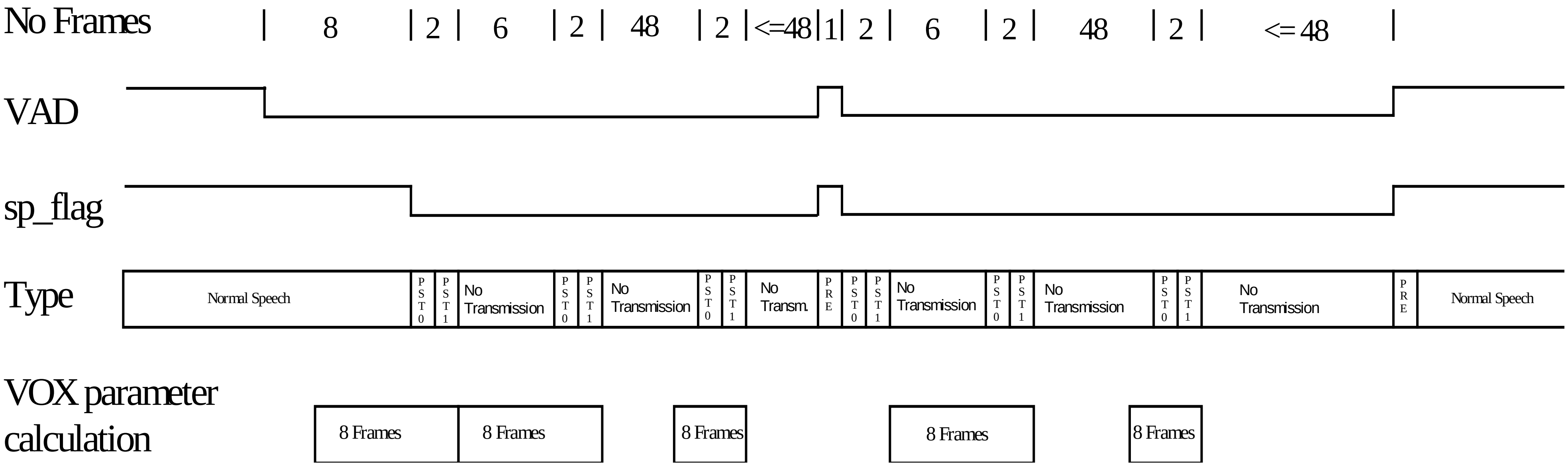


Figure 4 Simplified block diagram of the ACELP decoder



Note that the timescale is not linear

Figure 5 VOX handler timing