SCTE® | STANDARDS

Data Standards Subcommittee

AMERICAN NATIONAL STANDARD

ANSI/SCTE 24-21 2022 (R2022)

BV16 Speech Codec Specification for Voice over IP Applications in Cable Telephony

1 DOCUMENT TYPES AND TAGS

Document Type: Specification
Document Tags:

☐ Test or Measurement	☐ Checklist	☐ Facility
☒ Architecture or Framework	☐ Metric	☒ Access Network
☐ Procedure, Process or Method	☐ Cloud	☐ Customer Premises

2 DOCUMENT RELEASE HISTORY

Release	Date
SCTE 24-21 2006	*6/30/2006*
SCTE 24-21 2012	*4/23/2012*
SCTE 24-21 2017	*7/24/2017*
SCTE 24-21 2017	*August 2022*

Note: Standards that are released multiple times in the same year use: a, b, c, etc. to indicate normative balloted updates and/or r1, r2, r3, etc. to indicate editorial changes to a released document after the year.

Note: This document is a reaffirmation of SCTE 24-21 2017. No substantive changes have been made to this document. Information components may have been updated such as the title page, NOTICE text, headers, and footers.

Contents

Figures

Tables

3 INTRODUCTION

This document is identical to SCTE 24-21 2012 except for informative components which may have been updated such as the title page, NOTICE text, headers and footers. No normative changes have been made to this document.

This document contains the description of the BV16 speech codec[1]. BV16 compresses 8 kHz sampled narrowband speech to a bit rate of 16 kb/s by employing a speech coding algorithm called Two-Stage Noise Feedback Coding (TSNFC), developed by Broadcom.

The rest of this document is organized as follows. Section 4 gives a high-level overview of the TSNFC algorithm. Sections 5 and 6 give detailed description of the BV16 encoder and decoder, respectively. The BV16 codec specification given in Sections 5 and 6 contain sufficient details to allow those skilled in the art to implement bit-stream compatible and functionally equivalent BV16 encoders and decoders.

4 OVERVIEW OF THE BV16 SPEECH CODEC

In this section, the general principles of Two-Stage Noise Feedback Coding (TSNFC) are first introduced. Next, an overview of the BV16 algorithm is given.

4.1 Brief Introduction of Two-Stage Noise Feedback Coding (TSNFC)

In conventional Noise Feedback Coding (NFC), the encoder modifies a prediction residual signal by adding a noise feedback signal to it. A scalar quantizer quantizes this modified prediction residual signal. The difference between the quantizer input and output, or the quantization error signal, is passed through a noise feedback filter. The output signal of this filter is the noise feedback signal added to the prediction residual. The noise feedback filter is used to control the spectrum of the coding noise in order to minimize the perceived coding noise. This is achieved by exploiting the masking properties of the human auditory system.

Conventional NFC codecs typically use only a short-term noise feedback filter to shape the spectral envelope of the coding noise, and a scalar quantizer is used universally. In contrast, Broadcom's Two-Stage Noise Feedback Coding (TSNFC) system uses a codec structure employing two stages of noise feedback coding in a nested loop: the first NFC stage performs short-term prediction and short-term noise spectral shaping (spectral envelope shaping), and the second nested NFC stage performs long-term prediction and long-term noise spectral shaping (harmonic shaping). Such a nested two-stage NFC structure is shown in Figure 1 below.

[1] The "BV16 speech codec" specification is based on Broadcom Corporation's BroadVoice®16 speech codec. The BroadVoice® open source software is provided under the GNU Lesser General Public License ("LGPL"), version 2.1, as published by the Free Software Foundation.

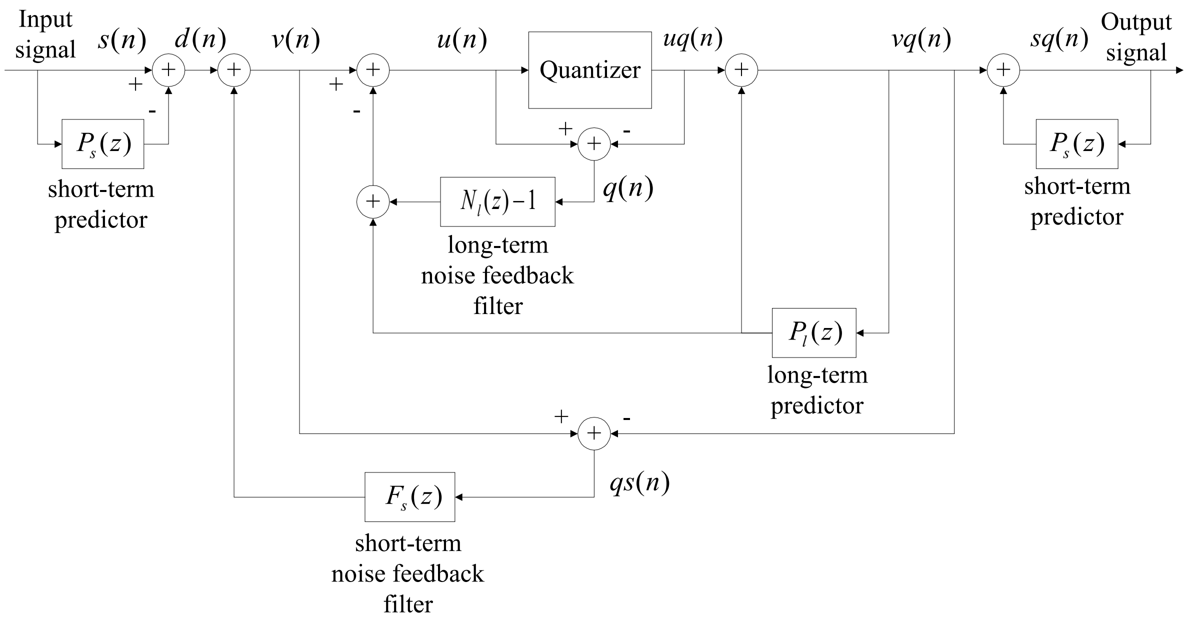

Figure 1 Basic codec structure of Two-Stage Noise Feedback Coding (TSNFC)

In Figure 1 above, the outer layer (including the two short-term predictors and the short-term noise feedback filter) follows the structure of the conventional NFC codec. The TSNFC structure in Figure 1 is obtained by replacing the simple scalar quantizer in the conventional (single-stage) NFC structure by a "predictive quantizer" that employs long-term prediction and long-term noise spectral shaping. This "predictive quantizer" is represented by the inner feedback loop in Figure 1, including the long-term predictor and long-term noise feedback filter. This inner feedback loop uses an alternative but equivalent conventional NFC structure, where $N_l(z)$ represents the filter whose frequency response is the desired noise shape for long-term noise spectral shaping. In the outer layer, the short-term noise feedback filter $F_s(z)$ is usually chosen as a bandwidth-expanded version of the short-term predictor $P_s(z)$. The choice of different NFC structures in the outer and inner layers is based on complexity consideration. By combining two stages of NFC in a nested loop, the TSNFC in Figure 1 can reap the benefits of both short-term and long-term prediction and also achieve short-term and long-term noise spectral shaping at the same time.

It is natural and straightforward to use a scalar quantizer in Figure 1. However, to achieve better coding efficiency, a vector quantizer is used in BV16. In the Vector Quantization (VQ) codebook search, the $u(n)$ vector cannot be generated before the VQ codebook search starts. Due to the feedback structure in Figure 1, the elements of $u(n)$ from the second element on will depend on the vector-quantized version of earlier elements. Therefore, the VQ codebook search is performed by trying out each of the candidate codevectors in the VQ codebook (i.e. fixing a candidate $uq(n)$ vector first), calculating the corresponding $u(n)$ vector and the corresponding VQ error $q(n) = u(n) - uq(n)$. The VQ codevector that minimizes the energy of $q(n)$ within the current vector time span is chosen as the winning codevector, and the corresponding codebook index becomes part of the encoder output bit stream for the current speech frame.

The TSNFC decoder structure is simply a quantizer decoder followed by the two feedback filter structures involving the long-term predictor and the short-term predictor, respectively, shown on the right half of Figure 1. Thus, the TSNFC decoder is similar to the decoders of other predictive coding techniques such as Adaptive Predictive Coding (APC), Multi-Pulse Linear Predictive Coding (MPLPC), and Code-Excited Linear Prediction (CELP).

If the alternative NFC structure in the inner feedback loop of Figure 1 is also used in the outer feedback loop, an alternative TSNFC codec structure is obtained, as shown in Figure 2 below. Here $N_s(z)$ represents a short-term filter whose frequency response is the desired noise shape for short-term noise spectral shaping. The codec structure in Figure 2 is mathematically equivalent to the structure in Figure 1, but it allows direct specification of the short-term noise spectral shape as defined by $N_s(z)$. This can be an advantage in some applications.

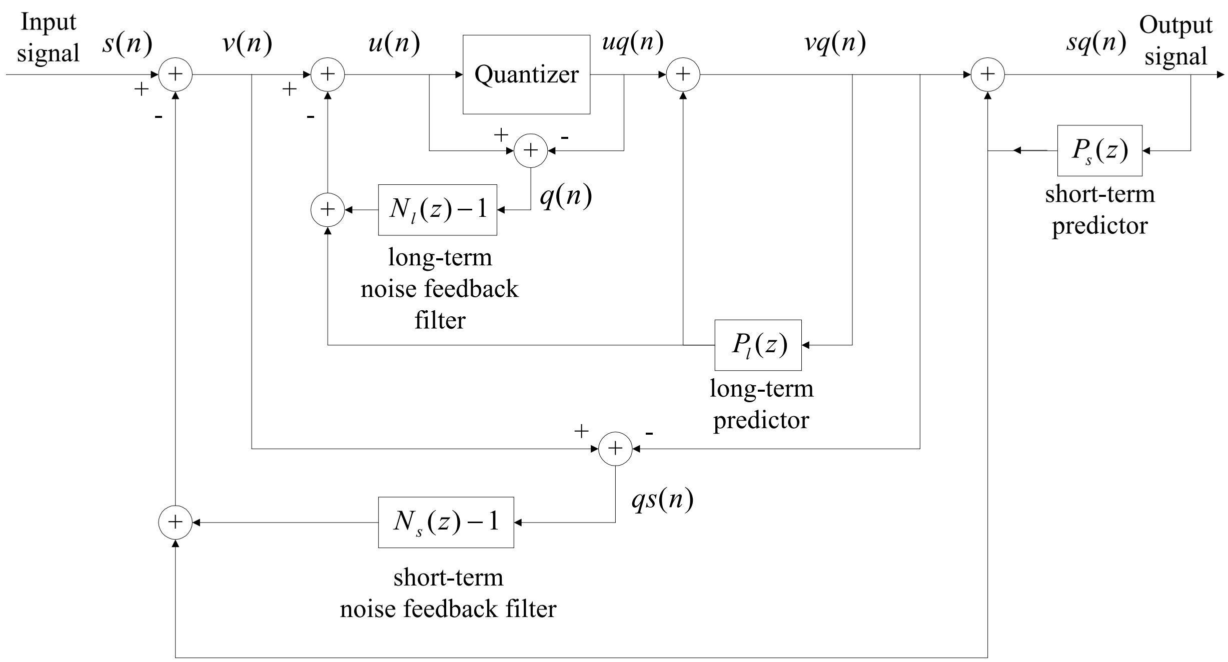

Figure 2 An alternative codec structure of Two-Stage Noise Feedback Coding (TSNFC)

4.2 Overview of the BV16 Codec

The BV16 codec is a purely forward-adaptive TSNFC codec. It operates at an input sampling rate of 8 kHz and an encoding bit rate of 16 kb/s, or 2 bits per sample. BV16 uses a frame size of 5 ms, or 40 samples. There is no look ahead. Therefore, the total algorithmic buffering delay is just the frame size itself, or 5 ms. The main design goal of BV16 is to make the coding delay and the codec complexity as low as possible, while providing toll speech quality exceeding or equivalent to that of G.728 and G.729E.

The block diagram of the BV16 encoder is shown in Figure 3. It is based on the alternative TSNFC codec structure shown in Figure 2. The BV16 decoder block diagram is shown in Figure 4.

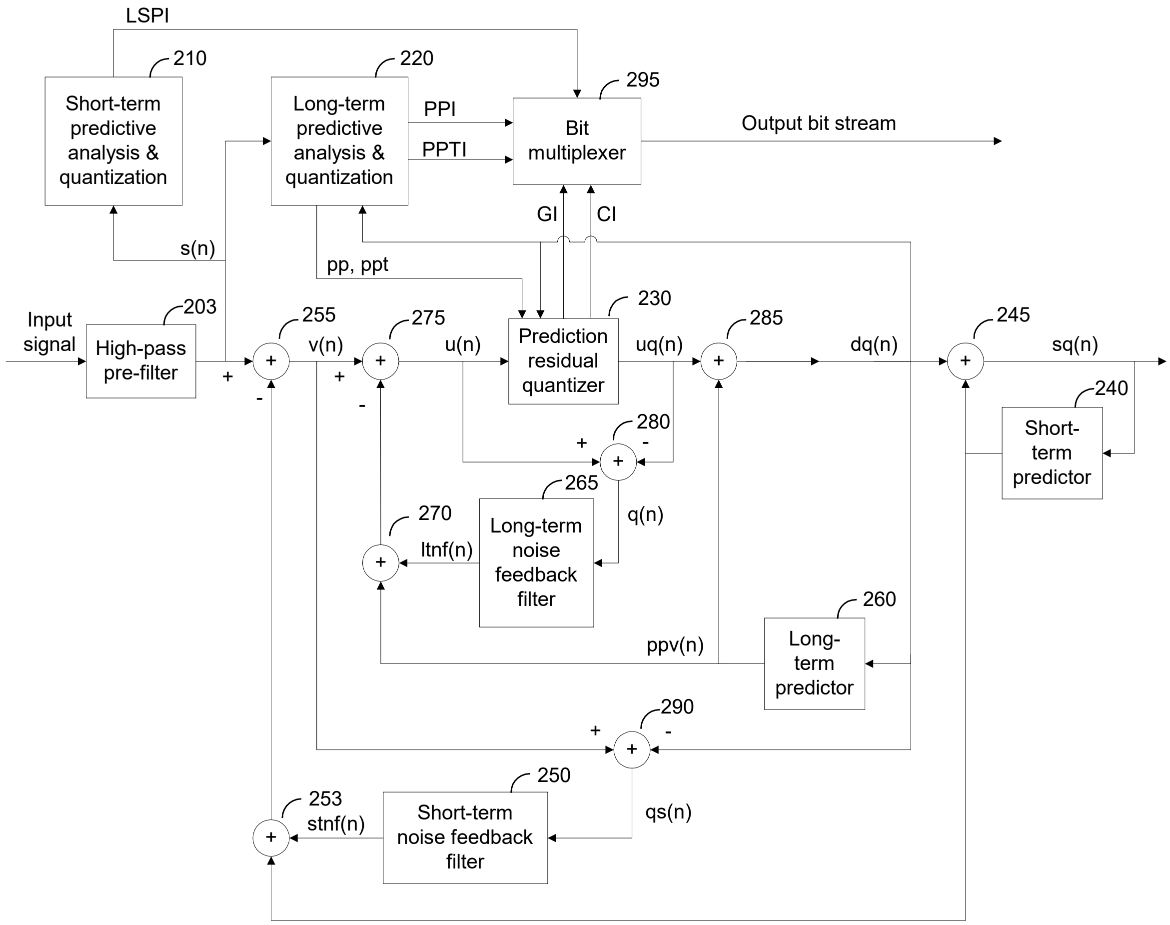

Figure 3 Block diagram of the BV16 encoder

Due to the small frame size, the parameters of the short-term predictor (also called the "LPC predictor") and the long-term predictor (also called the "pitch predictor") are both transmitted and updated once a frame. The gain of the excitation signal is transmitted once every frame. The excitation VQ uses a vector dimension of 4 samples. Hence, there are 10 excitation vectors in a frame.

The BV16 encoder first passes the input signal through a fixed pole-zero high-pass pre-filter to remove possible DC bias or low frequency rumble. The resulting signal is then used to derive the LPC predictor coefficients.

To keep the complexity low, BV16 uses a relatively low LPC predictor order of 8, and the LPC analysis window is 20 ms (160 samples) long. The LPC analysis window is asymmetric, with the peak of the window located at the center of the current frame, and the end of the window coinciding with the last sample of the current frame. Autocorrelation LPC analysis based on Levinson-Durbin recursion is used to derive the coefficients of the 8th-order LPC predictor. The derived LPC predictor coefficients are converted to Line-Spectrum Pair (LSP) parameters, which are then quantized by an inter-frame predictive coding scheme.

The inter-frame prediction of LSP parameters uses an 8^{th}-order moving-average (MA) predictor. The MA predictor coefficients are fixed. The time span that this MA predictor covers is 8×5 ms = 40 ms. The inter-frame LSP prediction residual is quantized by a two-stage vector quantizer. The first stage employs an 8-dimensional vector quantizer with a 7-bit codebook. The second stage uses an 8-dimensional sign-shape VQ with 1 bit for sign and 6 bits for shape.

For long-term prediction, a three-tap pitch predictor with an integer pitch period is used. To keep the complexity low, the pitch period and the pitch taps are both determined in an open-loop fashion.
The three pitch predictor taps are jointly quantized using a 5-bit vector quantizer. The distortion measure used in the codebook search is the energy of the open-loop pitch prediction residual. The 32 codevectors in the pitch tap codebook have been "stabilized" to make sure that they will not give rise to an unstable pitch synthesis filter.

The excitation gain is also determined in an open-loop fashion to keep the complexity low. The average power of the open-loop pitch prediction residual within the current frame is calculated and converted to the logarithmic domain. The resulting log-gain is then quantized using inter-subframe MA predictive coding. The MA predictor order for the log-gain is 8, corresponding to a time span of $8 \times 5 = 40$ ms. Again, the log-gain MA predictor coefficients are fixed. The log-gain prediction residual is quantized by a 4-bit scalar quantizer.

The 4-dimensional excitation VQ codebook has a simple sign-shape structure, with 1 bit for sign, and 4 bits for shape. In other words, only 16 four-dimensional codevectors are stored, but the mirror image of each codevector with respect to the origin is also a codevector.

In the BV16 decoder, the decoded excitation vectors are scaled by the excitation gain. The scaled excitation signal passes through a long-term synthesis filter and a short-term synthesis filter. Figure 4 shows the block diagram of the BV16 decoder.

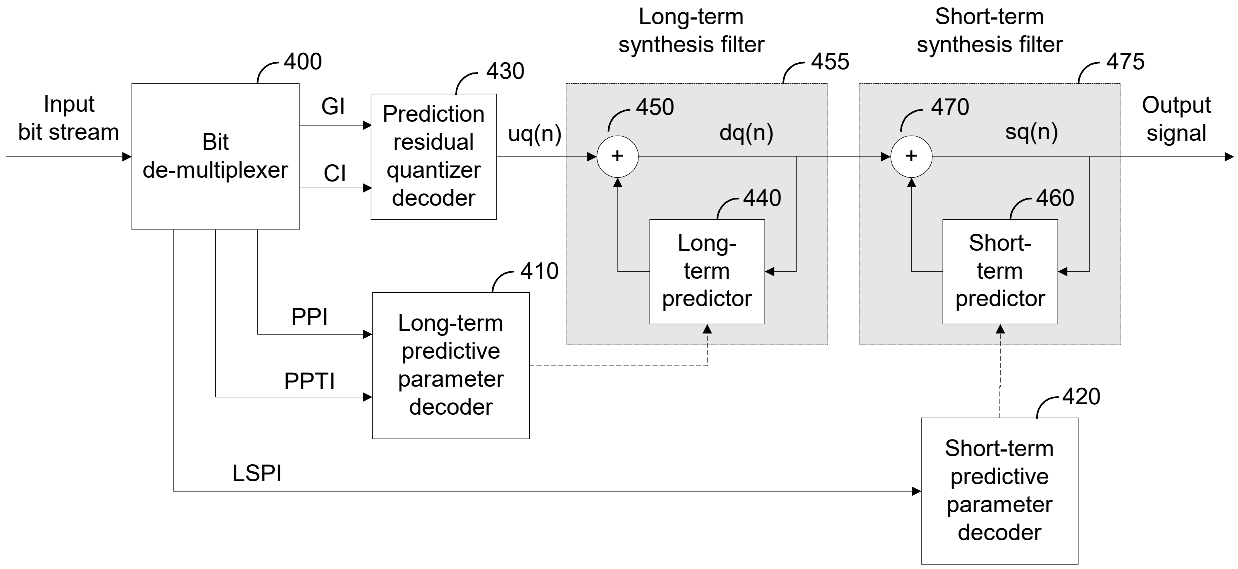

Figure 4 Block diagram of the BV16 decoder

Table 1 shows the bit allocation of BV16 in each 5 ms frame. The LSP parameters are encoded into 14 bits per frame, including 7 bits for the first-stage VQ, and 1 + 6 = 7 bits for the second-stage sign-shape VQ. The pitch period and pitch predictor taps are encoded into 7 and 5 bits, respectively. The excitation gain in each frame is encoded into 4 bits. The 10 excitation vectors are each encoded with 1 bit for sign and 4 bits for shape, resulting in 50 bits per frame for excitation VQ. Including the other 30 bits of side information, the grand total is 80 bits per 40-sample frame, which is 2 bits/sample, or 16 kb/s.

Table 1 Bit allocation of the BV16 codec

Parameter	Bits per frame (40 samples)
LSP	7 + 7 = 14
Pitch Period	7
3 Pitch Predictor Taps	5
Excitation Gain	4
10 Excitation Vectors	(1 + 4) × 10 = 50
Total	80

5 DETAILED DESCRIPTION OF THE BV16 ENCODER

In this section, detailed description of each functional block of the BV16 encoder in Figure 3 is given. When necessary, certain functional blocks will be expanded into more detailed block diagrams. The description given in this section will be in sufficient detail to allow those skilled in the art to implement a mathematically equivalent BV16 encoder.

5.1 High-Pass Pre-Filtering

Refer to Figure 3. The input signal is assumed to be represented by 16-bit linear PCM. Block 203 is a high-pass pre-filter with fixed coefficients. It is a second-order pole-zero filter with the following transfer function.

$$H_{hpf}(z) = \frac{0.924133 - 1.848267\,z^{-1} + 0.924133\,z^{-2}}{1 - 1.899109\,z^{-1} + 0.905396\,z^{-2}}$$

This high-pass pre-filter removes undesirable low-frequency components from the input signal.

5.2 Short-Term Linear Predictive Analysis

The high-pass filtered signal $s(n)$ is buffered at block 210, which performs short-term linear predictive analysis and quantization to obtain the coefficients for the short-term predictor 240 and the short-term noise feedback filter 250. This block 210 is further expanded in Figure 5.

Refer to Figure 5. The input signal $s(n)$ is buffered in block 211, where a 20 ms asymmetric analysis window is applied to the buffered $s(n)$ signal array. The "left window" is 17.5 ms long, and the "right window" is 2.5 ms long. Let *LWINSZ* be the number of samples in the left window (*LWINSZ* = 140 for 8 kHz sampling), then the left window is given by

$$wl(n) = \frac{1}{2}\left[1 - \cos\left(\frac{n\pi}{LWINSZ + 1}\right)\right],\ n = 1, 2, \ldots, LWINSZ.$$

Let *RWINSZ* be the number of samples in the right window. Then, *RWINSZ* = 20 for 8 kHz sampling. The right window is given by

$$wr(n) = \cos\left(\frac{(n-1)\pi}{2RWINSZ}\right),\ n = 1, 2, \ldots, RWINSZ.$$

The concatenation of $wl(n)$ and $wr(n)$ gives the 20 ms asymmetrical analysis window, with the peak of the window located at the center of the current frame. When applying this analysis window, the last sample of the window is lined up with the last sample of the current frame. Therefore, the codec does not use any look ahead.

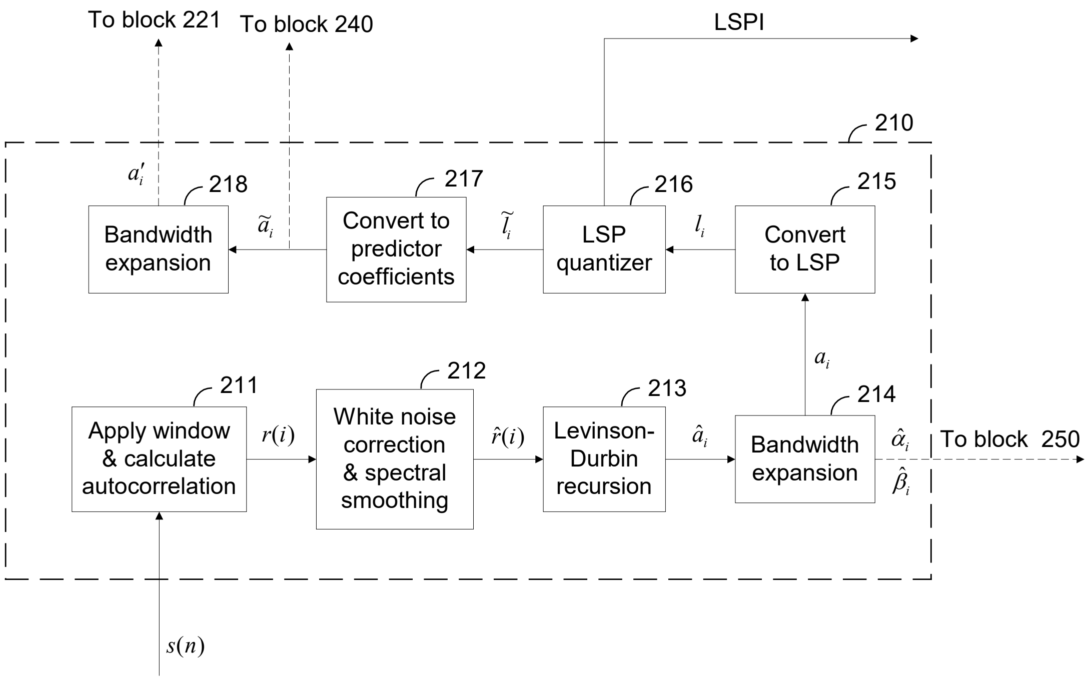

Figure 5 Short-term linear predictive analysis and quantization (block 210)

More specifically, without loss of generality, let the sampling time index range of n = 1, 2, …, *FRSZ* corresponds to the current frame, where the frame size *FRSZ* = 40. Then, the $s(n)$ signal buffer stored in block 211 is for n = -119, -118, …, -1, 0, 1, 2, …, 40. The asymmetrical LPC analysis window function can be expressed as

$$w(n) = \begin{cases} wl(n+120), & n = -119, -118, ..., 20 \\ wr(n-20), & n = 21, 22, ..., 40 \end{cases} .$$

The windowing operation is performed as follows.

$$s_w(n) = s(n)w(n), \quad n = \text{-119, -118, …, -1, 0, 1, 2, …, 40.}$$

Next, block 211 calculates the autocorrelation coefficients as follows.

$$r(i) = \sum_{n=-119+i}^{40} s_w(n)s_w(n-i), \quad i = 0, 1, 2, \ldots, 8.$$

The calculated autocorrelation coefficients are passed to block 212, which applies a Gaussian window to the autocorrelation coefficients to perform spectral smoothing. The Gaussian window function is given by

$$gw(i) = e^{\frac{-(2\pi\ i\sigma / f_s)^2}{2}} \quad , i = 1, 2, \ldots, 8,$$

where f_s is the sampling rate of the input signal, expressed in Hz, and σ is 40 Hz.

After multiplying the $r(i)$ array by such a Gaussian window, block 212 then multiplies $r(0)$ by a white noise correction factor of $WNCF = 1 + \varepsilon$, where $\varepsilon = 0.0001$. In summary, the output of block 212 is given by

$$\hat{r}(i) = \begin{cases} 1.0001 \times r(0), & i = 0 \\ gw(i)r(i), & i = 1,2,\ldots,8 \end{cases}$$

Block 213 performs the Levinson-Durbin recursion to convert the autocorrelation coefficients $\hat{r}(i)$ to the short-term predictor coefficients $\hat{a}_i$, $i = 0, 1, \ldots, 8$. If the Levinson-Durbin recursion exits pre-maturely before the recursion is completed (for example, because the prediction residual energy $E(i)$ is less than zero), then the short-term predictor coefficients of the last frame is also used in the current frame. To do the exception handling this way, there needs to be an initial value of the $\hat{a}_i$ array. The initial value of the $\hat{a}_i$ array is set to $\hat{a}_0 = 1$ and $\hat{a}_i = 0$ for $i = 1, 2, \ldots, 8$. The Levinson-Durbin recursion is performed in the following algorithm.

1. If $\hat{r}(0) \le 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.
2. $E(0) = \hat{r}(0)$
3. $k_1 = -\hat{r}(1) / \hat{r}(0)$
4. $\hat{a}_1^{(1)} = k_1$
5. $E(1) = (1 - k_1^2)E(0)$
6. If $E(1) \le 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.
7. For $i = 2, 3, 4, \ldots, 8$, do the following

$$k_i = \frac{-\hat{r}(i) - \sum_{j=1}^{i-1} \hat{a}_j^{(i-1)} \hat{r}(i-j)}{E(i-1)}$$

$$\hat{a}_i^{(i)} = k_i$$

$$\hat{a}_j^{(i)} = \hat{a}_j^{(i-1)} + k_i\ \hat{a}_{i-j}^{(i-1)}, \quad \text{for } j = 1, 2, \ldots, i-1$$

$$E(i) = (1 - k_i^2)E(i-1)$$

If $E(i) \le 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson - Durbin recursion.

If the recursion is exited pre-maturely, the $\hat{a}_i$ array of the last frame is used as the output of block 213. If the recursion is completed successfully (which is normally the case), then the final output of block 213 is taken as

$$\hat{a}_0 = 1$$

$$\hat{a}_i = \hat{a}_i^{(8)},\ \text{for } i = 1, 2, \ldots, 8$$

Block 214 performs bandwidth expansion as follows

$$a_i = (0.96852)^i \hat{a}_i \ , \text{for } i = 0, 1, \ldots, 8.$$

In addition, it also performs bandwidth expansion operations to derive the coefficients of the short-term noise feedback filter (block 250). Block 250 in Figure 3 has a transfer function of

$$F_s(z) = N_s(z) - 1 = \frac{\sum_{i=1}^{8} \hat{\beta}_i \ z^{-i}}{\sum_{i=0}^{8} \hat{\alpha}_i \ z^{-i}} \quad .$$

Block 214 calculates the coefficients of $F_s(z)$ as

$$\hat{\alpha}_i = (0.85)^i \hat{a}_i \ , \text{for } i = 0, 1, \ldots, 8,$$

$$\hat{\beta}_i = \left[(0.5)^i - (0.85)^i\right] \hat{a}_i \ , \text{for } i = 1, 2, \ldots, 8.$$

5.3 Conversion to LSP

In Figure 5, block 215 converts the LPC coefficients $a_i, i = 1, 2, \ldots, 8$ of the prediction error filter given by

$$A(z) = 1 + \sum_{i=1}^{8} a_i \ z^{-i}$$

to a set of 8 Line-Spectrum Pair (LSP) coefficients $l_i, i = 1, 2, \ldots, 8$. The LSP coefficients, also known as the Line Spectrum Frequencies (LSF), are the angular positions normalized to 1, i.e. 1.0 corresponds to the Nyquist frequency, of the roots of

$$A_p(z) = A(z) + z^{-9} A(z^{-1})$$

and

$$A_m(z) = A(z) - z^{-9} A(z^{-1})$$

on the upper half of the unit circle, $z = e^{j\omega}, 0 \le \omega \le \pi$, less the trivial roots in $z = -1$ and $z = 1$ of $A_p(z)$ and $A_m(z)$, respectively. Due to the symmetry and anti-symmetric of $A_p(z)$ and $A_m(z)$, respectively, the roots of interest can be determined as the roots of

$$G_p(\omega) = \sum_{i=0}^{4} g_{p,i} \cos(i\omega)$$

and

$$G_m(\omega) = \sum_{i=0}^{4} g_{m,i} \cos(i\omega)$$

where

$$g_{p|m,i} = \begin{cases} f_{p|m,4} & i = 0 \\ 2 f_{p|m,4-i} & i = 1,\ldots,4 \end{cases}$$

in which

$$f_{p,i} = \begin{cases} 1.0 & i = 0 \\ a_i + a_{9-i} - f_{p,i-1} & i = 1,\ldots,4 \end{cases}$$

and

$$f_{m,i} = \begin{cases} 1.0 & i = 0 \\ a_i - a_{9-i} + f_{p,i-1} & i = 1,\ldots,4 \end{cases} .$$

The subscript "p|m" means dual versions of the equation exist, with either subscript "p" or subscript "m". The roots of $A_p(z)$ and $A_m(z)$, and therefore the roots of $G_p(\omega)$ and $G_m(\omega)$, are interlaced, with the first root belonging to $G_p(\omega)$. The evaluation of the functions $G_p(\omega)$ and $G_m(\omega)$ are carried out efficiently using Chebyshev polynomial series. With the mapping $x = \cos(\omega)$,

$$\cos(m\omega) = T_m(x)$$

where $T_m(x)$ is the m^{th}-order Chebyshev polynomial, the two functions $G_p(\omega)$ and $G_m(\omega)$ can be expressed as

$$G_{p|m}(x) = \sum_{i=0}^{4} g_{p|m,i}\, T_i(x).$$

Due to the recursive nature of Chebyshev polynomials the functions can be evaluated as

$$G_{p|m}(x) = \frac{b_{p|m,0}(x) - b_{p|m,2}(x) + g_{p|m,0}}{2}$$

where $b_{p|m,0}(x)$ and $b_{p|m,2}(x)$ are calculated using the following recurrence

$$b_{p|m,i}(x) = 2x\, b_{p|m,i+1}(x) - b_{p|m,i+2}(x) + g_{p|m,i}$$

with initial conditions $b_{p|m,5}(x) = b_{p|m,6}(x) = 0$.

The roots of $G_p(x)$ and $G_m(x)$ are determined in an alternating fashion starting with a root in $G_p(x)$. Each root of $G_p(x)$ and $G_m(x)$ is located by identifying a sign change of the relevant function along a grid of 60 points, given in Appendix 1. The estimation of the root is then refined using 4 bisections followed by a final linear interpolation between the two points surrounding the root. It should be noted that the roots and grid points are in the cosine domain. Once the 8 roots

$$x_i = \cos(\omega_i), \quad i = 1, 2, \ldots, 8$$

are determined in the cosine domain, they are converted to the normalized frequency domain according to

$$l_i = \cos^{-1}(x_i)/\pi, \quad i = 1, 2, \ldots, 8$$

in order to obtain the LSP coefficients. In the rare event that less than 8 roots are found, block 215 returns the LSP coefficients of the previous frame, $l_i(k-1), i = 1, 2, \ldots, 8$, where the additional parameter k represents the frame index of the current frame. The LSP coefficients of the previous frame at the very first frame are initialized to

$$l_i(0) = i/9, \quad i = 1, 2, \ldots, 8.$$

5.4 LSP Quantization

Block 216 of Figure 5 vector-quantizes and encodes the LSP coefficient vector, $\mathbf{l} = [l_1 \quad l_2 \quad \ldots \quad l_8]^T$, to a total of 14 bits. The output LSP quantizer index array, $LSPI = \{LSPI_1, LSPI_2\}$, is passed to the bit multiplexer (block 295), while the quantized LSP coefficient vector, $\tilde{\mathbf{l}} = [\tilde{l}_1 \quad \tilde{l}_2 \quad \ldots \quad \tilde{l}_8]^T$, is passed to block 217.

The LSP quantizer is based on mean-removed inter-frame moving-average (MA) prediction with two-stage vector quantization (VQ) of the prediction error. The quantizer enables bit-error detection at the decoder by constraining the codevector selection at the encoder. It should be noted that the encoder must perform the specified constrained VQ in order to maintain interoperability properly. The first-stage VQ is searched using the simple mean-squared error (MSE) distortion criterion, while the second-stage sign-shape VQ is searched using the weighted mean-square error (WMSE) distortion criterion.

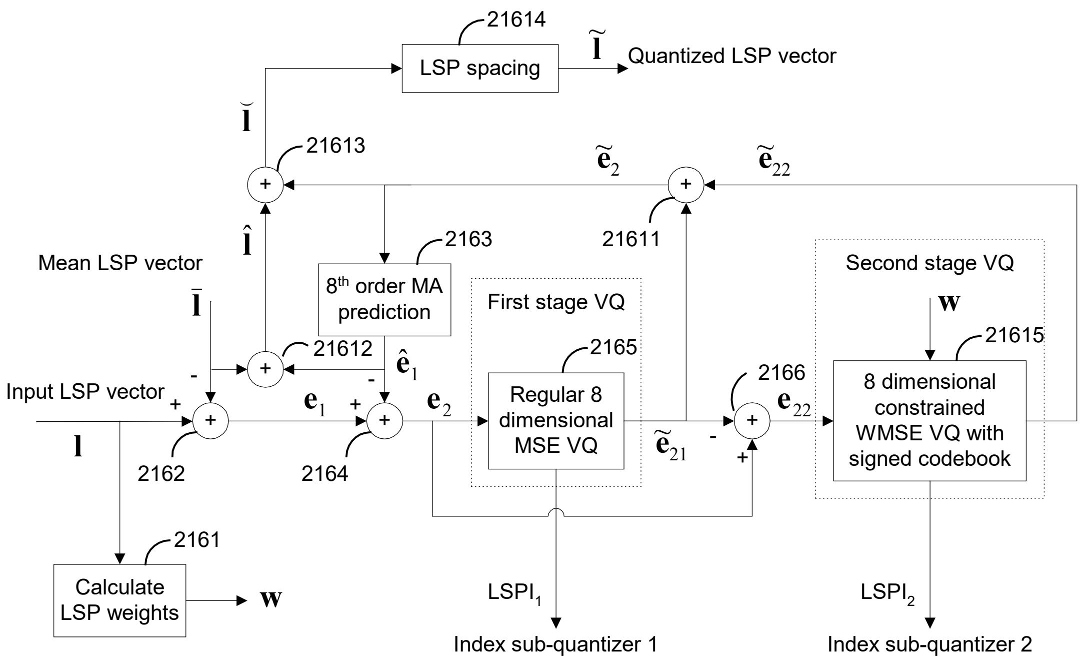

Figure 6 LSP quantizer (block 216)

Block 216 is further expanded in Figure 6. The first-stage VQ takes place in block 2165, and the second-stage constrained sign-shape VQ takes place in block 21615. Except for the LSP quantizer indices $LSPI_1, LSPI_2$ all signal paths in Figure 6 are for vectors of dimension 8. Block 2161 uses the unquantized LSP coefficient vector to calculate the weights to be used later in the second-stage WMSE VQ. The weights are determined as

$$w_i = \begin{cases} 1/(l_2 - l_1), & i = 1 \\ 1/\min(l_i - l_{i-1}, l_{i+1} - l_i), & 1 < i < 8 \\ 1/(l_M - l_{M-1}), & i = 8 \end{cases} .$$

Basically, the *i*-th weight is the inverse of the distance between the *i*-th LSP coefficient and its nearest neighbor LSP coefficient.

Adder 2162 subtracts the constant LSP mean vector,

$$\bar{\mathbf{l}} = \begin{bmatrix} 0.0950317 & 0.1489563 & 0.2513123 & 0.3629456 & 0.4780884 & 0.5877075 & 0.7058105 & 0.8007202 \end{bmatrix}^T ,$$

from the unquantized LSP coefficient vector to get the mean-removed LSP vector,

$$\mathbf{e}_1 = \mathbf{l} - \bar{\mathbf{l}}\,.$$

In Figure 6, block 2163 performs 8th order inter-frame MA prediction of the mean-removed LSP vector $\mathbf{e}_1$ based on the $\tilde{\mathbf{e}}_2$ vectors in the previous 8 frames, where $\tilde{\mathbf{e}}_2$ is the quantized version of the inter-frame LSP prediction error vector[2]. Let $\tilde{e}_{2,i}(k)$ denote the i-th element of the vector $\tilde{\mathbf{e}}_2$ in the frame that is k frames before the current frame. Let $\hat{e}_{1,i}$ be the i-th element of the inter-frame-predicted mean-removed LSP vector $\hat{\mathbf{e}}_1$. Then, block 2163 calculates the predicted mean-removed LSP vector according to

$$\hat{e}_{1,i} = \mathbf{p}_{LSP,i}{}^T \cdot \left[\tilde{e}_{2,i}(1) \quad \tilde{e}_{2,i}(2) \quad \tilde{e}_{2,i}(3) \quad \tilde{e}_{2,i}(4) \quad \tilde{e}_{2,i}(5) \quad \tilde{e}_{2,i}(6) \quad \tilde{e}_{2,i}(7) \quad \tilde{e}_{2,i}(8)\right]^T,\ \ i = 1, 2, \ldots, 8\,,$$

where $\mathbf{p}_{LSP,i}$ holds the 8 prediction coefficients for the i-th LSP coefficient and is given by

$$\begin{aligned}
\mathbf{p}_{LSP,1}{}^T &= [1.040710 \quad 0.844971 \quad 0.682922 \quad 0.575989 \quad 0.464600 \quad 0.346008 \quad 0.226074 \quad 0.103577]\\
\mathbf{p}_{LSP,2}{}^T &= [1.034851 \quad 0.884094 \quad 0.723816 \quad 0.609863 \quad 0.489563 \quad 0.366516 \quad 0.240234 \quad 0.109253]\\
\mathbf{p}_{LSP,3}{}^T &= [1.055237 \quad 0.922180 \quad 0.762695 \quad 0.644531 \quad 0.512695 \quad 0.373474 \quad 0.238037 \quad 0.108337]\\
\mathbf{p}_{LSP,4}{}^T &= [1.076843 \quad 0.935608 \quad 0.790771 \quad 0.673523 \quad 0.540588 \quad 0.399841 \quad 0.264221 \quad 0.118774]\\
\mathbf{p}_{LSP,5}{}^T &= [1.065552 \quad 0.901978 \quad 0.746155 \quad 0.636047 \quad 0.514282 \quad 0.386169 \quad 0.256165 \quad 0.117493]\,.\\
\mathbf{p}_{LSP,6}{}^T &= [1.037476 \quad 0.848816 \quad 0.684326 \quad 0.577393 \quad 0.463684 \quad 0.347717 \quad 0.232666 \quad 0.107239]\\
\mathbf{p}_{LSP,7}{}^T &= [1.022278 \quad 0.809021 \quad 0.645081 \quad 0.535767 \quad 0.430481 \quad 0.325562 \quad 0.219055 \quad 0.099304]\\
\mathbf{p}_{LSP,8}{}^T &= [0.964844 \quad 0.743469 \quad 0.578125 \quad 0.484375 \quad 0.393250 \quad 0.297913 \quad 0.201416 \quad 0.091736]
\end{aligned}$$

Adder 2164 calculates the prediction error vector

$$\mathbf{e}_2 = \mathbf{e}_1 - \hat{\mathbf{e}}_1\,,$$

which is the input to the first-stage VQ. In block 2165 the 8-dimensional prediction error vector, $\mathbf{e}_2$, is vector quantized with the 128-entry, 8-dimensional codebook, $\mathbf{CB}_1 = \left\{\mathbf{cb}_1{}^{(0)}, \mathbf{cb}_1{}^{(1)}, \ldots, \mathbf{cb}_1{}^{(127)}\right\}$, listed in Appendix 2. The codevector minimizing the MSE is denoted $\tilde{\mathbf{e}}_{21}$ and the corresponding index is denoted $LSPI_1$:

$$LSPI_1 = \underset{k \in \{0,1,\ldots,127\}}{\arg\min} \left\{\left(\mathbf{e}_2 - \mathbf{cb}_1{}^{(k)}\right)^{\mathrm{T}} \left(\mathbf{e}_2 - \mathbf{cb}_1{}^{(k)}\right)\right\},$$

$$\tilde{\mathbf{e}}_{21} = \mathbf{cb}_1{}^{(LSPI_1)},$$

[2] At the first frame, the previous, non-existing, quantized interframe LSP prediction error vectors are set to zero-vectors.

where the notation $I = \arg\min_{i}\{D(i)\}$ means that I is the argument that minimizes the entity $D(i)$, i.e.

$$D(I) \le D(i) \text{ for all } i .$$

Adder 2166 subtracts the first-stage codevector from the prediction error vector to form the quantization error vector of the first stage,

$$\mathbf{e}_{22} = \mathbf{e}_2 - \tilde{\mathbf{e}}_{21} .$$

This is the input to the second-stage VQ, which is a sign-shape VQ with a 2-entry, 1-dimensional sign codebook, $\mathbf{S} = \{s_0, s_1\} = \{-1,+1\}$, and a 64-entry, 8-dimensional shape codebook, $\mathbf{CB}_2 = \{\mathbf{cb}_2^{(0)}, \mathbf{cb}_2^{(1)}, \ldots, \mathbf{cb}_2^{(63)}\}$, listed in Appendix 3. The product codevector that minimizes the WMSE, subject to the constraint that the 3 first elements of the intermediate quantized LSP vector,

$$\begin{aligned} \breve{\mathbf{l}} &= \hat{\mathbf{l}} + \tilde{\mathbf{e}}_2 \\ &= \bar{\mathbf{l}} + \hat{\mathbf{e}}_1 + \tilde{\mathbf{e}}_{21} + \tilde{\mathbf{e}}_{22} \end{aligned},$$

preserve the ordering property

$$\begin{aligned} \breve{l}_1 &\ge 0 \\ \breve{l}_2 &\ge \breve{l}_1 \\ \breve{l}_3 &\ge \breve{l}_2 \end{aligned},$$

is selected as,

$$\tilde{\mathbf{e}}_{22} = s_{I_{sg}} \mathbf{cb}_2^{(I_{sh})},$$

where the indices are given by

$$\{I_{sg}, I_{sh}\} = \underset{\{i,k\} \in \left\{\{h,j\} \,\middle|\, h \in \{0,1\}, \breve{l}_1^{(h,j)} \ge 0, \breve{l}_2^{(h,j)} \ge \breve{l}_1^{(h,j)}, \breve{l}_3^{(h,j)} \ge \breve{l}_2^{(h,j)}, j \in \{0,1,\ldots,63\}\right\}}{\arg\min} \left\{ \left(\mathbf{e}_{22} - s_i \mathbf{cb}_2^{(k)}\right)^T \mathbf{W} \left(\mathbf{e}_{22} - s_i \mathbf{cb}_2^{(k)}\right) \right\},$$

and the weighting matrix is

$$\mathbf{W} = \begin{bmatrix} w_1 & & & 0 \\ & w_2 & & \\ & & \ddots & \\ 0 & & & w_8 \end{bmatrix}.$$

The symbol $\breve{l}_i^{(h,j)}$ is the *i*-th element of the reconstructed LSP vector $\breve{\mathbf{l}}$ that is generated by using a sign index $I_{sg} = h$ and the *j*-th shape codevector in $\mathbf{CB}_2$. From the sign index, I_{sg}, and the shape index, I_{sh}, the index of the second stage VQ, $LSPI_2$, is calculated as

$$LSPI_2 = \begin{cases} 127 - I_{sh}, & I_{sg} = 0 \\ I_{sh}, & I_{sg} = 1 \end{cases},$$

In the unlikely event that no product codevector fulfills the constraint, the product codevector $\tilde{\mathbf{e}}_{22} = \mathbf{cb}_2^{(0)}$ is selected, and the index $LSPI_2 = 0$ is returned.

Once the quantization is complete, the remaining operations of block 216 construct the quantized LSP vector from the codevectors, LSP mean, and MA prediction. Adder 21611 calculates the quantized prediction error vector by adding the stage 1 and stage 2 quantized vectors,

$$\tilde{\mathbf{e}}_2 = \tilde{\mathbf{e}}_{21} + \tilde{\mathbf{e}}_{22} .$$

Adder 21612 adds the mean LSP vector and the predicted mean-removed LSP vector to obtain the predicted LSP vector,

$$\hat{\mathbf{l}} = \bar{\mathbf{l}} + \hat{\mathbf{e}}_1 \ .$$

Adder 21613 adds the predicted LSP vector and the quantized prediction error vector to get the intermediate reconstructed LSP vector,

$$\breve{\mathbf{l}} = \hat{\mathbf{l}} + \tilde{\mathbf{e}}_2 \ .$$

Block 21614 checks the elements of the reconstructed LSP vector to enforce certain minimum spacing rules. It enforces a minimum value of 6 Hz for the smallest LSP coefficient, a maximum value of 3991 Hz for the largest LSP coefficient, and a minimum distance between neighboring LSP coefficients of 50 Hz. In the normalized domain of the LSP coefficients, the spacing requirement is given by

$$\begin{array}{lll} \breve{l}_1 & \geq & 0.0015 \\ \breve{l}_{i+1} - \breve{l}_i & \geq & 0.0125 \qquad i = 1, 2, \ldots, 7 \, . \\ \breve{l}_8 & \leq & 0.99775 \end{array}$$

The spacing is carried out as follows:

(i) The elements of the intermediate reconstructed LSP vector are sorted such that

$$\breve{l}_1 \leq \breve{l}_2 \leq \ldots \leq \breve{l}_8 \, .$$

(ii) Set $l_{\max} = 0.91025$.

(iii) If $\breve{l}_1 < 0.0015$, set $\tilde{l}_1 = 0.0015$.
else if $\breve{l}_1 > l_{\max}$, set $\tilde{l}_1 = l_{\max}$.
else set $\tilde{l}_1 = \breve{l}_1$.

(iv) for i = 2, 3, … , 8 do the following:

1. Set $l_{\min} = \tilde{l}_{i-1} + 0.0125$.
2. Set $l_{\max} \leftarrow l_{\max} + 0.0125$.
3. If $\breve{l}_i < l_{\min}$, set $\tilde{l}_i = l_{\min}$.
   else if $\breve{l}_i > l_{\max}$, set $\tilde{l}_i = l_{\max}$.
   else set $\tilde{l}_i = \breve{l}_i$.

5.5 Conversion to Short-Term Predictor Coefficients

Refer back to Figure 5. In block 217, the quantized set of LSP coefficients $\{\tilde{l}_i\}$, which is determined once a frame, is converted to the corresponding set of linear prediction coefficients $\{\tilde{a}_i\}$, the quantized linear prediction coefficients for the current frame.

With the notation

$$\begin{aligned} x_{p,i} &= \cos(\pi\,\tilde{l}_{2i-1}), \quad i = 1,2,3,4 \\ x_{m,i} &= \cos(\pi\,\tilde{l}_{2i}), \quad i = 1,2,3,4 \end{aligned}$$

the 4 unique coefficients of each of the two polynomials $A_p^{\Delta}(z) = A_p(z)/(1+z^{-1})$ and $A_m^{\Delta}(z) = A_m(z)/(1-z^{-1})$ can be determined using the following recursion:

$$\begin{aligned} &\text{For } i = 1,2,3,4, \text{ do the following:} \\ &\quad a_{p|m,i}^{\Delta} = 2\left(a_{p|m,i-2}^{\Delta} - x_{p|m,i}\, a_{p|m,i-1}^{\Delta}\right) \\ &\quad a_{p|m,j}^{\Delta} = a_{p|m,j}^{\Delta} + a_{p|m,j-2}^{\Delta} - 2\,x_{p|m,i}\, a_{p|m,j-1}^{\Delta}, \quad j = i-1, i-2, \ldots, 1 \end{aligned}$$

with initial conditions $a_{p|m,0}^{\Delta} = 1$ and $a_{p|m,-1}^{\Delta} = 0$. In the recursion above, $\{a_{p,i}^{\Delta}\}$ and $\{a_{m,i}^{\Delta}\}$ are the sets of four unique coefficients of the polynomials $A_p^{\Delta}(z)$ and $A_m^{\Delta}(z)$, respectively. Similarly, let the two sets of coefficients $\{a_{p,i}\}$ and $\{a_{m,i}\}$, each of 4 unique coefficients except for a sign on $\{a_{m,i}\}$, represent the unique coefficients of the polynomials $A_p(z)$ and $A_m(z)$, respectively. Then, $\{a_{p,i}\}$ and $\{a_{m,i}\}$ can be obtained from $\{a_{p,i}^{\Delta}\}$ and $\{a_{m,i}^{\Delta}\}$ as

$$\begin{aligned} a_{p,i} &= a_{p,i}^{\Delta} + a_{p,i-1}^{\Delta}, \quad i = 1,2,3,4 \\ a_{m,i} &= a_{m,i}^{\Delta} - a_{m,i-1}^{\Delta}, \quad i = 1,2,3,4 \end{aligned}$$

From $A_p(z)$ and $A_m(z)$, the polynomial of the prediction error filter is obtained as

$$\widetilde{A}(z) = \frac{A_p(z) + A_m(z)}{2}.$$

In terms of the unique coefficients of $A_p(z)$ and $A_m(z)$, the coefficients $\{\widetilde{a}_i\}$ of $\widetilde{A}(z)$ can be expressed as

$$\widetilde{a}_i = \begin{cases} 1.0, & i = 0 \\ 0.5\,(a_{p,i} + a_{m,i}), & i = 1, 2, 3, 4 \\ 0.5\,(a_{p,9-i} - a_{m,9-i}), & i = 5, 6, 7, 8 \end{cases}$$

where the tilde signifies that the coefficients correspond to the quantized LSP coefficients. Note that

$$\widetilde{A}(z) = 1 - P_s(z) = 1 + \sum_{i=1}^{8} \widetilde{a}_i \; z^{-i} \;,$$

where

$$P_s(z) = -\sum_{i=1}^{8} \widetilde{a}_i \; z^{-i}$$

is the transfer function of the short-term predictor block 240 in Figure 3.

Block 218 performs further bandwidth expansion on the set of predictor coefficients $\{\widetilde{a}_i\}$ using a bandwidth expansion factor of $\gamma_1 = 0.75$. The resulting bandwidth-expanded set of filter coefficients is given by

$$a_i' = \gamma_1^{\,i}\widetilde{a}_i \;, \text{ for } i = 1, 2, \ldots, 8.$$

This bandwidth-expanded set of filter coefficients $\{a_i'\}$ is used to update the coefficients of the weighted short-term synthesis filter block 221 in Figure 7 (to be discussed later). This completes the description of short-term predictive analysis and quantization block 210 in Figure 3 and Figure 5.

5.6 Long-Term Linear Predictive Analysis (Pitch Extraction)

In Figure 3, the long-term predictive analysis and quantization block 220 uses the short-term prediction residual signal $d(n)$ of the current frame and its quantized version $dq(n)$ in the previous frames to determine the quantized values of the pitch period and the pitch predictor taps. This block 220 is further expanded in Figure 7 below.

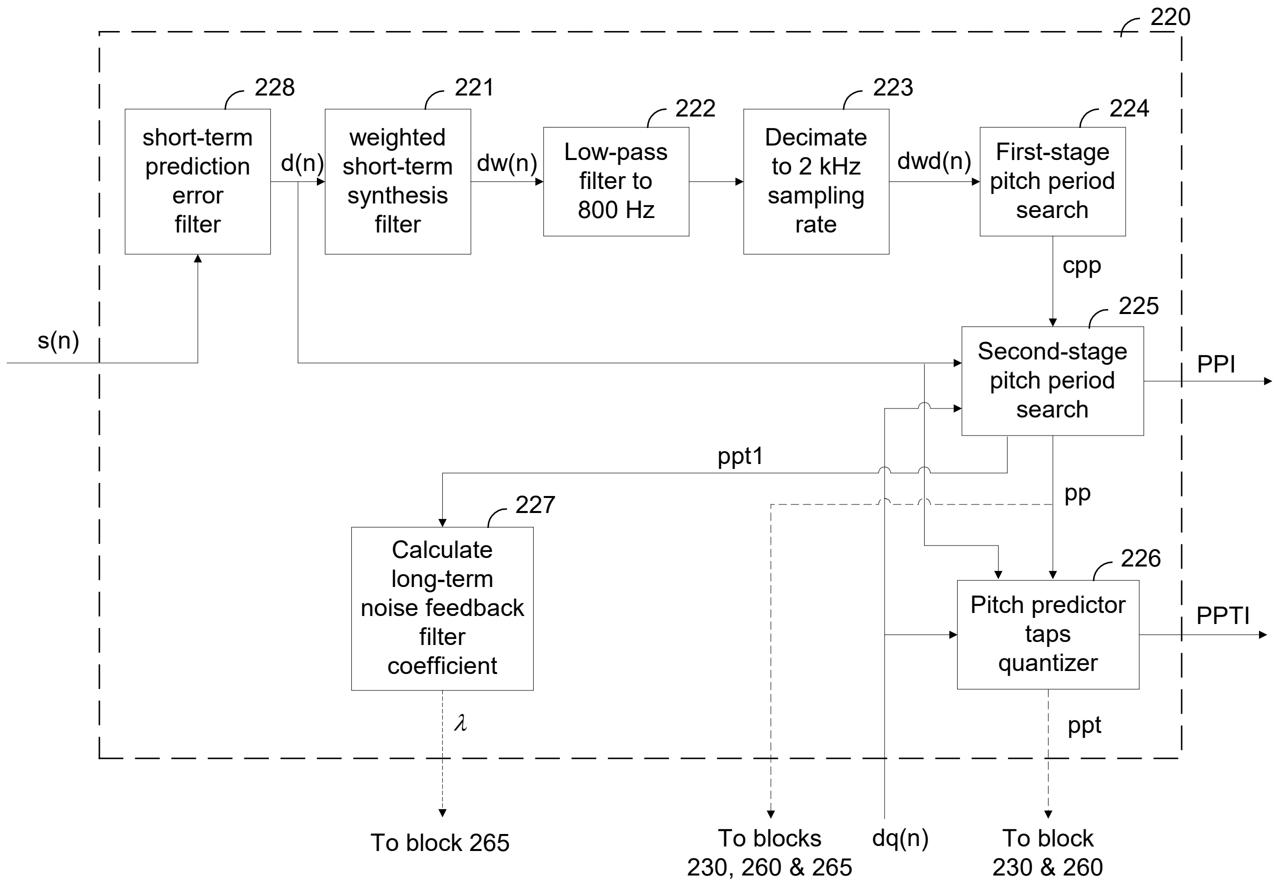

Figure 7 Long-term predictive analysis and quantization (block 220)

Now refer to Figure 7. Block 228 performs short-term prediction error filtering to get the short-term prediction residual $d(n)$ as follows.

$$d(n) = s(n) + \sum_{i=1}^{8} \tilde{a}_i s(n-i).$$

The short-term prediction residual signal $d(n)$ passes through the weighted short-term synthesis filter block 221, whose output is calculated as

$$dw(n) = d(n) - \sum_{i=1}^{8} a_i' dw(n-i)$$

The signal $dw(n)$ is passed through a fixed low-pass filter block 222, which has a –3 dB cut off frequency at about 800 Hz. A 4th-order elliptic filter is used for this purpose. The transfer function of this low-pass filter is

$$H_{lpf}(z) = \frac{0.0433083 - 0.0687180z^{-1} + 0.0991097z^{-2} - 0.0687180z^{-3} + 0.0433083z^{-4}}{1 - 2.9580236z^{-1} + 3.6337313z^{-2} - 2.1249529z^{-3} + 0.5003969z^{-4}}$$

Block 223 down-samples the low-pass filtered signal to a sampling rate of 2 kHz. This represents an 4:1 decimation.

The first-stage pitch search block 224 then uses the decimated 2 kHz sampled signal $dwd(n)$ to find a "coarse pitch period", denoted as *cpp* in Figure 7. The time lag represented by *cpp* is in terms of number of samples in the 2 kHz down-sampled signal $dwd(n)$. A pitch analysis window of 15 ms is used. The end of the pitch analysis window is aligned with the end of the current frame. At a sampling rate of 2 kHz, 15 ms correspond to 30 samples. Without loss of generality, let the index range of $n = 1$ to $n = 30$ correspond to the pitch analysis window for $dwd(n)$. Block 224 first calculates the following values

$$c(k) = \sum_{n=1}^{30} dwd(n)dwd(n-k),$$

$$E(k) = \sum_{n=1}^{30} \left[dwd(n-k)\right]^2,$$

$$c2(k) = \begin{cases} c^2(k), & \text{if } c(k) \geq 0 \\ -c^2(k), & \text{if } c(k) < 0 \end{cases}$$

for all integers from $k = MINPPD - 1$ to $k = MAXPPD + 1$, where *MINPPD* and *MAXPPD* are the minimum and maximum pitch period in the decimated domain, respectively, *MINPPD* = 2 sample and *MAXPPD* = 34 samples. Block 224 then searches through the range of k = *MINPPD, MINPPD* + 1, *MINPPD* + 2, …, *MAXPPD* to find all local peaks[3] of the array $\{c2(k)/E(k)\}$ for which $c(k) > 0$. Let N_p denote the number of such positive local peaks. Let $k_p(j)$, j =1, 2, …, N_p be the indices where $c2(k_p(j))/E(k_p(j))$ is a local peak and $c(k_p(j)) > 0$, and let $k_p(1) < k_p(2) < ... < k_p(N_p)$. For convenience, the term $c2(k)/E(k)$ will be referred to as the "normalized correlation square".

If $N_p = 0$, the output coarse pitch period is set to *cpp = MINPPD*, and the processing of block 224 is terminated. If $N_p = 1$, block 224 output is set to *cpp* = $k_p(1)$, and the processing of block 224 is terminated.

If there are two or more local peaks ($N_p \geq 2$), then block 224 uses *Algorithms 3.8.1, 3.8.2, 3.8.3,* and *3.8.4* (to be described below), in that order, to determine the output coarse pitch period *cpp*. Variables calculated in the earlier algorithms will be carried over and used in the later algorithms.

[3] A value is characterized as a local peak if both of the adjacent values are smaller.

Block 224 first uses *Algorithm 3.8.1* below to identify the largest quadratically interpolated peak around local peaks of the normalized correlation square $c2(k_p)/E(k_p)$. Quadratic interpolation is performed for $c(k_p)$, while linear interpolation is performed for $E(k_p)$. Such interpolation is performed with the time resolution for the sampling rate of the input speech (8 kHz). In the algorithm below, D denotes the decimation factor used when decimating $dw(n)$ to $dwd(n)$. Thus, $D = 4$.

Algorithm 3.8.1 Find largest quadratically interpolated peak around $c2(k_p)/E(k_p)$:

(i) Set $c2max$ = -1, $Emax$ = 1, and $jmax$ = 0.

(ii) For j =1, 2, …, N_p, do the following 12 steps:

1. Set $a = 0.5\ \left[c(k_p(j)+1)+c(k_p(j)-1)\right]-c(k_p(j))$
2. Set $b = 0.5\ \left[c(k_p(j)+1)-c(k_p(j)-1)\right]$
3. Set $ji = 0$
4. Set $ei = E(k_p(j))$
5. Set $c2m = c2(k_p(j))$
6. Set $Em = E(k_p(j))$
7. If $c2(k_p(j)+1)E(k_p(j)-1) > c2(k_p(j)-1)E(k_p(j)+1)$, do the remaining part of step 7:

   $\Delta = [E(k_p(j)+1)-ei]/D$

   For k = 1, 2, … , D/2, do the following indented part of step 7:

   $ci = a\ (k/D)^2 + b\ (k/D) + c(k_p(j))$

   $ei \leftarrow ei + \Delta$

   If $(ci)^2 Em > (c2m)\, ei$, do the next three indented lines:

   $ji = k$

   $c2m = (ci)^2$

   $Em = ei$
8. If $c2(k_p(j)+1)E(k_p(j)-1) \le c2(k_p(j)-1)E(k_p(j)+1)$, do the remaining part of step 8:

   $\Delta = [E(k_p(j)-1)-ei]/D$

   For k = -1, -2, … , -D/2, do the following indented part of step 8:

   $ci = a\ (k/D)^2 + b\ (k/D) + c(k_p(j))$

   $ei \leftarrow ei + \Delta$

   If $(ci)^2 Em > (c2m)\, ei$, do the next three indented lines:

   $ji = k$

   $c2m = (ci)^2$

   $Em = ei$
9. Set $lag(j) = k_p(j) + ji/D$

10. Set $c2i(j) = c2m$
11. Set $Ei(j) = Em$
12. If $c2m \times Emax > c2max \times Em$, do the following three indented lines:
    $jmax = j$
    $c2max = c2m$
    $Emax = Em$

(iii) Set the first candidate for coarse pitch period as $cpp = k_p(jmax)$.

The symbol $\leftarrow$ indicates that the parameter on the left-hand side is being updated with the value on the right-hand side[4].

To avoid picking a coarse pitch period that is around an integer multiple of the true coarse pitch period, a search through the time lags corresponding to the local peaks of $c2(k_p)/E(k_p)$ is performed to see if any of such time lags is close enough to the output coarse pitch period of block 224 in the last frame, denoted as *cpplast*[5]. If a time lag is within 25% of *cpplast*, it is considered close enough. For all such time lags within 25% of *cpplast*, the corresponding quadratically interpolated peak values of the normalized correlation square $c2(k_p)/E(k_p)$ are compared, and the interpolated time lag corresponding to the maximum normalized correlation square is selected for further consideration. The following algorithm performs the task described above. The interpolated arrays $c2i(j)$ and $Ei(j)$ calculated in *Algorithm 3.8.1* above are used in this algorithm.

Algorithm 3.8.2 Find the time lag maximizing interpolated $c2(k_p)/E(k_p)$ *among all time lags close to the output coarse pitch period of the last frame:*

(i) Set index im = -1
(ii) Set $c2m$ = -1
(iii) Set Em = 1
(iv) For j =1, 2, …, N_p, do the following:
  If $|k_p(j) - cpplast| \le 0.25 \times cpplast$, do the following:
    If $c2i(j) \times Em > c2m \times Ei(j)$, do the following three lines:
      $im = j$
      $c2m = c2i(j)$
      $Em = Ei(j)$

Note that if there is no time lag $k_p(j)$ within 25% of *cpplast*, then the value of the index *im* will remain at –1 after *Algorithm 3.8.2* is performed. If there are one or more time lags within 25% of

---
[4] An equal sign is not applicable due to a potential mathematical conflict.
[5] For the first frame *cpplast* is initialized to 12.

cpplast, the index *im* corresponds to the largest normalized correlation square among such time lags.

Next, block 224 determines whether an alternative time lag in the first half of the pitch range should be chosen as the output coarse pitch period. Basically, block 224 searches through all interpolated time lags $lag(j)$ that are less than 16, and checks whether any of them has a large enough local peak of normalized correlation square near every integer multiple of it (including itself) up to 32. If there are one or more such time lags satisfying this condition, the smallest of such qualified time lags is chosen as the output coarse pitch period of block 224.

Again, variables calculated in *Algorithms 3.8.1* and *3.8.2* above carry their final values over to *Algorithm 3.8.3* below. In the following, the parameter *MPDTH* is 0.065, and the threshold array $MPTH(k)$ is given as $MPTH(2) = 0.63$, $MPTH(3) = 0.48$, $MPTH(4) = 0.42$, $MPTH(5) = 0.36$, and $MPTH(k) = 0.30$, for $k > 5$.

Algorithm 3.8.3 Check whether an alternative time lag in the first half of the range of the coarse pitch period should be chosen as the output coarse pitch period:

For $j = 1, 2, 3, \ldots, N_p$, in that order, do the following while $lag(j) < 16$:

(i) If $j \neq im$, set $threshold = 0.73$; otherwise, set $threshold = 0.4$.

(ii) If $c2i(j) \times Emax \leq threshold \times c2max \times Ei(j)$, disqualify this j, skip step (iii) for this j, increment j by 1 and go back to step (i).

(iii) If $c2i(j) \times Emax > threshold \times c2max \times Ei(j)$, do the following:

a) For $k = 2, 3, 4, \ldots$, do the following while $k \times lag(j) < 32$:

1. $s = k \times lag(j)$
2. $a = (1 - MPDTH)\, s$
3. $b = (1 + MPDTH)\, s$
4. Go through $m = j+1, j+2, j+3, \ldots, N_p$, in that order, and see if any of the time lags $lag(m)$ is between a and b. If none of them is between a and b, disqualify this j, stop step (iii), increment j by 1 and go back to step (i). If there is at least one such m that satisfies $a < lag(m) \leq b$ and $c2i(m) \times Emax > MPTH(k) \times c2max \times Ei(m)$, then it is considered that a large enough peak of the normalized correlation square is found in the neighborhood of the k-th integer multiple of $lag(j)$; in this case, stop step (iii) a) 4., increment k by 1, and go back to step (iii) a) 1.

b) If step (iii) a) is completed without stopping prematurely, that is, if there is a large enough interpolated peak of the normalized correlation square within

±100×*MPDTH*% of every integer multiple of $lag(j)$ that is less than 32, then stop this algorithm and stop the operation of block 224, and set $cpp = k_p(j)$ as the final output coarse pitch period of block 224.

If *Algorithm 3.8.3* above is completed without finding a qualified output coarse pitch period *cpp*, then block 224 examines the largest local peak of the normalized correlation square around the coarse pitch period of the last frame, found in *Algorithm 3.8.2* above, and makes a final decision on the output coarse pitch period *cpp* using the following algorithm. *Algorithm 3.8.4* performs this final decision. Again, variables calculated in *Algorithms 3.8.1* and *3.8.2* above carry their final values over to *Algorithm 3.8.4* below. In the following, the parameters are $SMDTH = 0.095$ and $LPTH1 = 0.79$.

Algorithm 3.8.4: Final decision of the output coarse pitch period

(i) If $im = -1$, that is, if there is no large enough local peak of the normalized correlation square around the coarse pitch period of the last frame, then use the *cpp* calculated at the end of *Algorithm 3.8.1* as the final output coarse pitch period of block 224, and exit this algorithm.

(ii) If $im = jmax$, that is, if the largest local peak of the normalized correlation square around the coarse pitch period of the last frame is also the global maximum of all interpolated peaks of the normalized correlation square within this frame, then use the *cpp* calculated at the end of *Algorithm 3.8.1* as the final output coarse pitch period of block 224, and exit this algorithm.

(iii) If $im < jmax$, do the following indented part:

If $c2m \times Emax > 0.43 \times c2max \times Em$, do the following indented part of step (iii):

a) If $lag(im) > MAXPPD/2$, set block 224 output $cpp = k_p(im)$ and exit this algorithm.

b) Otherwise, for $k = 2, 3, 4, 5$, do the following indented part:

1. $s = lag(jmax) / k$
2. $a = (1 - SMDTH)\, s$
3. $b = (1 + SMDTH)\, s$
4. If $lag(im) > a$ and $lag(im) < b$, set block 224 output $cpp = k_p(im)$ and exit this algorithm.

(iv) If $im > jmax$, do the following indented part:

If $c2m \times Emax > LPTH1 \times c2max \times Em$, set block 224 output $cpp = k_p(im)$ and exit this algorithm.

(v) If algorithm execution proceeds to here, none of the steps above have selected a final output coarse pitch period. In this case, just accept the *cpp* calculated at the end of *Algorithm 3.8.1* as the final output coarse pitch period of block 224.

Block 225 takes *cpp* as its input and performs a second-stage pitch period search in the undecimated signal domain to get a refined pitch period *pp*. Block 225 first converts the coarse pitch period *cpp* to the undecimated signal domain by multiplying it by the decimation factor *D*, where $D = 4$. Then, it determines a search range for the refined pitch period around the value $cpp \times D$. The lower bound of the search range is $lb = \max(MINPP, cpp \times D - D + 1)$, where $MINPP = 10$ samples is the minimum pitch period. The upper bound of the search range is $ub = \min(MAXPP, cpp \times D + D - 1)$, where *MAXPP* is the maximum pitch period, which is 136 samples.

Block 225 maintains a signal buffer with a total of $MAXPP + 1 + FRSZ$ samples, where *FRSZ* is the frame size, which is 40 samples. The last *FRSZ* samples of this buffer are populated with the open-loop short-term prediction residual signal $d(n)$ in the current frame. The first $MAXPP + 1$ samples are populated with the $MAXPP + 1$ samples of quantized version of $d(n)$, denoted as $dq(n)$, immediately preceding the current frame. For convenience of writing equations later, the symbol $dq(n)$ will be used to denote the entire buffer of $MAXPP + 1 + FRSZ$ samples, even though the last *FRSZ* samples are really $d(n)$ samples. Again, let the index range from $n = 1$ to $n = FRSZ$ denotes the samples in the current frame.

After the lower bound *lb* and upper bound *ub* of the pitch period search range are determined, block 225 calculates the following correlation and energy terms in the undecimated $dq(n)$ signal domain for time lags k within the search range $[lb, ub]$.

$$\tilde{c}(k) = \sum_{n=1}^{FRSZ} dq(n)dq(n-k)$$

$$\tilde{E}(k) = \sum_{n=1}^{FRSZ} dq(n-k)^2$$

The time lag $k \in [lb, ub]$ that maximizes the ratio $\tilde{c}^2(k)/\tilde{E}(k)$ is chosen as the final refined pitch period. That is,

$$pp = \arg\max_{k \in [lb, ub]} \left[\frac{\tilde{c}^2(k)}{\tilde{E}(k)} \right] .$$

Once the refined pitch period *pp* is determined, it is encoded into the corresponding output pitch period index *PPI*, calculated as

$$PPI = pp - 10 \ .$$

Possible values of *PPI* are all integers from 0 to 126. Therefore, the refined pitch period *pp* is encoded into 7 bits, without any distortion. The value of *PPI* = 127 is reserved for signaling purposes and therefore is not used by the codec.

Block 225 also calculates $ppt1$, the optimal tap weight for a single-tap pitch predictor, as follows

$$ppt1 = \frac{\tilde{c}(pp)}{\tilde{E}(pp)} \ .$$

In the degenerate case where $\tilde{E}(pp) = 0$, $ppt1$ is set to zero. Block 227 calculates the long-term noise feedback filter coefficient λ as follows.

$$\lambda = \begin{cases} 0.5, & ppt1 \geq 1 \\ 0.5 \times ppt1, & 0 < ppt1 < 1 \\ 0, & ppt1 \leq 0 \end{cases}$$

5.7 Long-Term Predictor Parameter Quantization

Pitch predictor taps quantizer block 226 quantizes the three pitch predictor taps to 5 bits using vector quantization. The pitch predictor has a transfer function of

$$P_l(z) = \sum_{i=1}^{3} b_i \ z^{-pp+2-i} \ ,$$

where *pp* is the pitch period calculated in Section 5.6.

Rather than minimizing the mean-square error of the three taps b_1, b_2, and b_3 as in conventional VQ codebook search, block 226 finds from the VQ codebook the set of candidate pitch predictor taps that minimizes the pitch prediction residual energy in the current frame. Using the same $dq(n)$ buffer and time index convention as in block 225, and denoting the set of three taps corresponding to the j-th codevector, $\mathbf{b}_j = [b_{j1} \quad b_{j2} \quad b_{j3}]^T$, as $\{ b_{j1}, b_{j2}, b_{j3} \}$, we can express such pitch prediction residual energy as

$$E_j = \sum_{n=1}^{FRSZ} \left[dq(n) - \sum_{i=1}^{3} b_{ji} dq(n - pp + 2 - i) \right]^2 .$$

The codevector is selected from a 3-dimensional codebook of 32 codevectors, $\{ \mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{31} \}$, listed in Appendix 4. The codevector that minimizes the pitch prediction residual energy is selected. The index of the selected codevector is given by

$$PPTI = j^* = \underset{j \in \{0,1,\ldots,31\}}{\arg\min} \{E_j\}$$

and the corresponding set of three quantized pitch predictor taps, denoted as $ppt = \{b_1, b_2, b_3\}$ in Figure 7, is given by

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{b}_{j^*}.$$

This completes the description of block 220, long-term predictive analysis and quantization.

5.8 Excitation Gain Quantization

There is one residual gain for each frame. The unquantized residual gain is based on the pitch prediction residual of the frame and is quantized in an open-loop fashion in the base-2 logarithmic domain. The quantization of the residual gain is part of the prediction residual quantizer block 230 in Figure 3. Block 230 is further expanded in Figure 8. All the operations in Figure 8 are performed on a frame-by-frame basis.

Block 300 in Figure 8 calculates the pitch prediction residual signal, given by

$$e(n) = dq(n) - \sum_{i=1}^{3} b_i dq(n - pp + 2 - i), \quad n = 1, 2, ..., FRSZ \,,$$

where the same $dq(n)$ buffer and time index convention of block 225 is used. That is, the current frame of $dq(n)$ for n = 1, 2, …, $FRSZ$ is the unquantized open-loop short-term prediction residual signal $d(n)$.

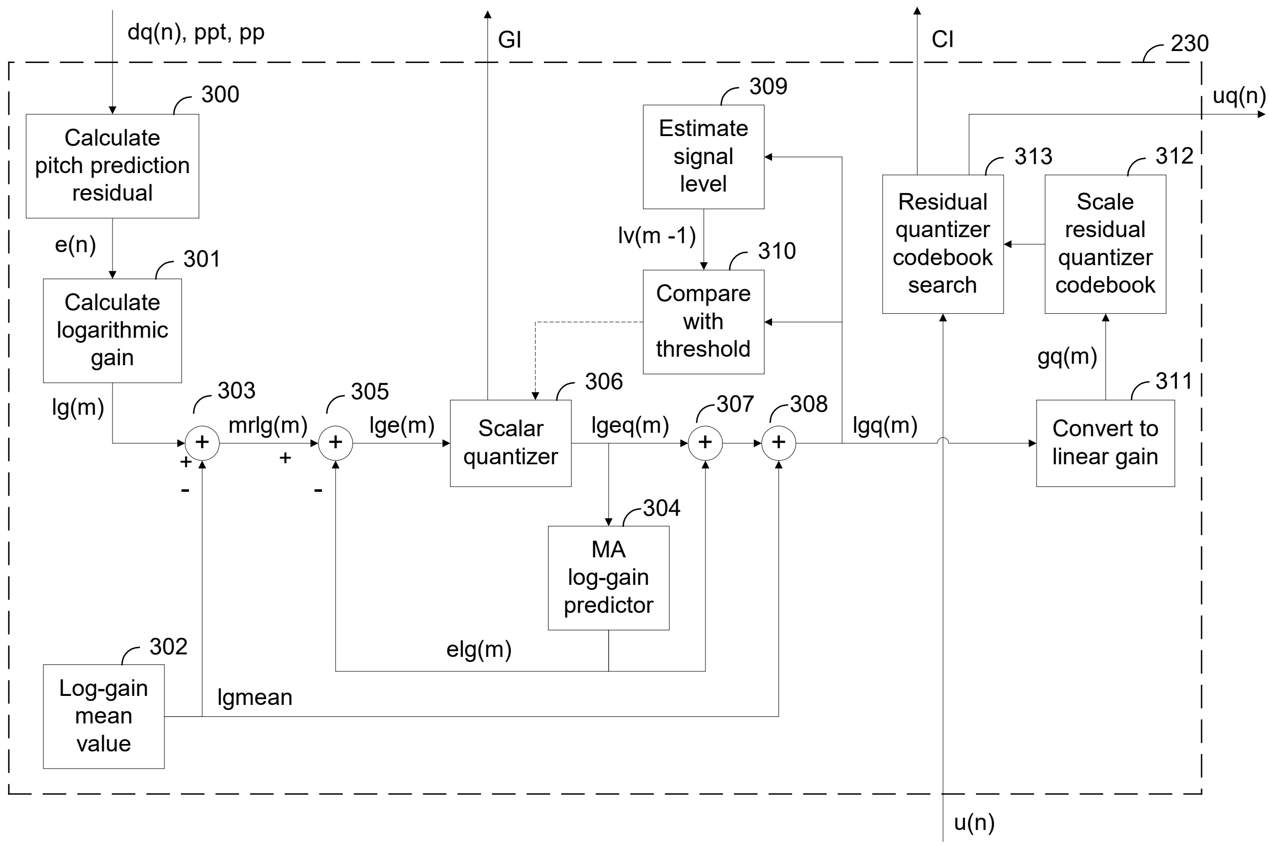

Figure 8 Prediction residual quantizer (block 230)

Block 301 calculates the residual gain in the base-2 logarithmic domain. First, the average power of the pitch prediction residual signal in the current frame, the *m*-th frame, is calculated as

$$P_e(m) = \frac{1}{FRSZ} \sum_{n=1}^{FRSZ} e^2(n)$$

The logarithmic gain (log-gain) of the current frame is calculated as

$$lg(m) = \begin{cases} \log_2 P_e(m), & \text{if } P_e(m) > 1 \\ 0, & \text{if } P_e(m) \le 1 \end{cases} .$$

The long-term mean value of the log-gain is calculated off-line and stored in block 302. This log-gain mean value is *lgmean* = 11.45752. The adder 303 calculates the mean-removed version of the log-gain as *mrlg*(*m*) = *lg*(*m*) - *lgmean*. The MA log-gain predictor block 304 is an 8^{th}-order FIR filter with its memory initialized to zero at the very first frame. The coefficients of this log-gain predictor *lgp*(*k*), *k* = 1, 2, 3, …, 8, are fixed, as given below:

$lgp(1) = 0.7801514$
$lgp(2) = 0.7377625$
$lgp(3) = 0.6150818$
$lgp(4) = 0.5926208$
$lgp(5) = 0.4674072$
$lgp(6) = 0.3635864$
$lgp(7) = 0.2378540$
$lgp(8) = 0.1286926$

Block 304 calculates its output, the estimated log-gain, as

$$elg(m) = \sum_{k=1}^{GPO} lgp(k) lgeq(m-k) \ ,$$

where $GPO = 8$ is the gain predictor order, and $lgeq(m - k)$ is the quantized version of the log-gain prediction error at frame $m - k$.

The adder 305 calculates the log-gain prediction error as

$$lge(m) = mrlg(m) - elg(m).$$

The scalar quantizer block 306 performs 4-bit scalar quantization of the resulting log-gain prediction error $lge(m)$. The codebook entries of this gain quantizer, along with the corresponding codebook indices, are listed in Appendix 5. The operation of this quantizer is controlled by block 310, whose purpose is to achieve a good trade-off between clear-channel performance and noisy-channel performance of the excitation gain quantizer. The operation of block 310 will be described later.

For each temporarily quantized $lgeq(m)$, the adders 307 and 308 together calculate the corresponding temporarily quantized log-gain as

$$lgq(m) = lgeq(m) + elg(m) + lgmean$$

Block 309 estimates the signal level based on the final quantized log-gain, to be determined later subject to the constraint imposed by block 310. Let $lv(m)$ denote the output estimated signal level of block 309 at frame m. Since the final value of $lgq(m)$ has not been determined yet at this point, block 310 can only use the estimated signal level at the last frame, namely, $lv(m - 1)$. One way to think of this situation is that block 309 has a one-sample delay unit for its input $lgq(m)$.

At frame m, block 310 controls the quantization operation of block 306 based on $lv(m - 1)$, $lgq(m - 1)$, and $lgq(m - 2)$[6]. It uses an $NG \times NGC$ gain change threshold matrix $T(i, j)$, $i = 1, 2, \ldots, NG$, $j = 1, 2, \ldots, NGC$ to limit how high $lgq(m)$ can go. The parameter values are $NG = 18$ and $NGC = 12$. The threshold matrix $T(i, j)$ is given in Appendix 6.

[6] The initial values of $lgq(m - 1)$ and $lgq(m - 2)$ are 0, i.e. $lgq(0)= 0$ *and* $lgq(-1)= 0$.

Block 310 and block 306 work together to perform the quantization of *lge*(*m*) in the following way. First, the row index into the threshold matrix *T*(*i, j*) is calculated as

$$i = \left\lceil \frac{lgq(m-1) - lv(m-1) - GLB}{2} \right\rceil,$$

where *GLB* = –24, and the symbol $\lceil . \rceil$ means "take the next larger integer" or "rounding to the nearest integer toward infinity". If $i > NG$, *i* is clipped to *NG*. If $i < 1$, *i* is clipped to 1.

Second, the column index into the threshold matrix *T*(*i, j*) is calculated as

$$j = \left\lceil \frac{lgq(m-1) - lgq(m-2) - GCLB}{2} \right\rceil,$$

where *GCLB* = –8. If $j > NGC$, *j* is clipped to *NGC*. If $j < 1$, *j* is clipped to 1.

Third, with the row and column indices *i* and *j* calculated above, a gain quantization limit is calculated as

$$GL = lgq(m-1) + T(i, j) - elg(m) - lgmean$$

Fourth, block 306 performs normal scalar quantization of *lge*(*m*) into its nearest neighbor in the quantizer codebook. If the resulting quantized value is not greater than *GL*, this quantized value is accepted as the final quantized log-gain prediction error *lgeq*(*m*), and the corresponding codebook index is the output gain index GI_m. On the other hand, if the quantized value is greater than *GL*, the next smaller gain quantizer codebook entry is compared with *GL*. If it is not greater than *GL*, it is accepted as the final output *lgeq*(*m*) of block 306, and the corresponding codebook index is accepted as GI_m. However, if it is still greater the *GL*, then block 306 keeps looking for the next smaller quantizer codebook entry (in descending order of codebook entry value), until it finds one that is not greater than *GL*. In such a search, the first one (that is, the largest one) that it finds to be no greater than *GL* is chosen as the final output *lgeq*(*m*) of block 306, and the corresponding codebook index is accepted as GI_m. In the rare occasion when all the gain quantizer codebook entries are greater than *GL*, then the smallest gain quantizer codebook entry is chosen as the final output *lgeq*(*m*) of block 306, and the corresponding codebook index (0 in this case) is chosen as the output GI_m. The final gain quantizer codebook index GI_m is passed to the bit multiplexer block 295 of Figure 3.

Once the quantized log-gain prediction error *lgeq*(*m*) is determined in this way, adders 307 and 308 add *elg*(*m*) and *lgmean* to *lgeq*(*m*) to obtain the quantized log-gain *lgq*(*m*) as

$$lgq(m) = lgeq(m) + elg(m) + lgmean$$

After this final quantized log-gain *lgq*(*m*) subject to the constraint imposed by block 310 is calculated, it is used by block 309 to update the estimated signal level *lv*(*m*). This value *lv*(*m*) is used by block 310 in the next frame (the (*m* + 1)-th frame).

At frame m, after the final quantized log-gain $lgq(m)$ is calculated, block 309 estimates the signal level using the following algorithm. The parameter values used are $\alpha = 4095/4096$, $\beta = 511/512$, and $\gamma = 255/256$. At codec initialization, the related variables are initialized as: $lmax(m - 1) = -100$, $lmin(m - 1) = 100$, $lmean(m - 1) = 12.5$, $lv(m - 1) = 17$, and $x(m - 1) = 17$.

Algorithm for updating estimated long-term average signal level:

(i) If $lgq(m) > lmax(m - 1)$, set $lmax(m) = lgq(m)$;
otherwise; set $lmax(m) = lmean(m - 1) + \alpha\, [lmax(m - 1) - lmean(m - 1)]$.

(ii) If $lgq(m) < lmin(m - 1)$, set $lmin(m) = lgq(m)$;
otherwise; set $lmin(m) = lmean(m - 1) + \alpha\, [lmin(m - 1) - lmean(m - 1)]$.

(iii) Set $lmean(m) = \beta \times lmean(m - 1) + (1 - \beta)\, [lmax(m) + lmin(m)]/2$.

(iv) Set $lth = lmean(m) + 0.2\, [lmax(m) - lmean(m)]$.

(v) If $lgq(m) > lth$, set $x(m) = \gamma \times x(m - 1) + (1 - \gamma) lgq(m)$, and set $lv(m) = \gamma \times lv(m - 1) + (1 - \gamma)\, x(m)$;
Otherwise, set $x(m) = x(m - 1)$ and $lv(m) = lv(m - 1)$.

Block 311 converts the quantized log-gain $lgq(m)$ to the quantized gain $gq(m)$ in the linear domain as follows.

$$gq(m) = 2^{\frac{lgq(m)}{2}}$$

Block 312 scales the residual vector quantization (also called excitation VQ) codebook by simply multiplying every element of every codevector in the excitation VQ codebook by $gq(m)$. The resulting scaled codebook is then used by block 313 to perform Excitation VQ codebook search, as described in the next section.

5.9 Excitation Vector Quantization

The excitation VQ codebook has a sign-shape structure, with 1 bit for sign and 4 bits for shape. The vector dimension is 4. Thus, there are 16 independent shape codevectors stored in the codebook, but the negated version of each shape codevector (i.e., the mirror image with respect to the origin) is also a valid codevector for excitation VQ. The 16 shape codevectors, along with the corresponding codebook indices, are listed in Appendix 7.

Block 313 in Figure 8 performs the excitation VQ codebook search using the filter structure shown in Figure 9, which is essentially a subset of the encoder shown in Figure 3. The only difference is that the prediction residual quantizer (block 230) in Figure 3 is replaced by block 248 in Figure 9, which is labeled as "scaled VQ codebook". This scaled VQ codebook is calculated in Section 5.8.

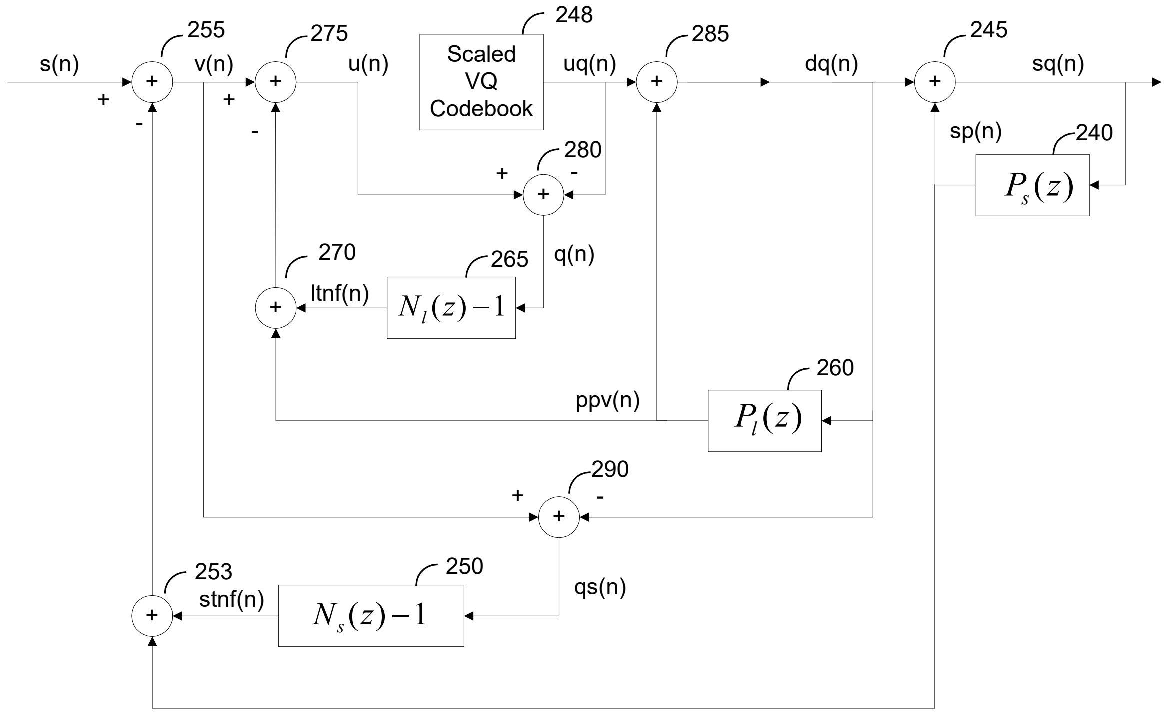

Figure 9 Filter structure used in excitation VQ codebook search

The four filters of blocks 240, 250, 260, and 265 have transfer functions given by

$$P_s(z) = -\sum_{i=1}^{8} \tilde{a}_i \; z^{-i} \quad \text{(see Section 5.5)},$$

where $\tilde{a}_i$ is the i-th coefficient of the quantized short-term prediction error filter;

$$N_s(z) - 1 = \frac{\sum_{i=1}^{8} \hat{\beta}_i \; z^{-i}}{\sum_{i=0}^{8} \hat{\alpha}_i \; z^{-i}} \quad \text{(see Section 5.2)};$$

$$P_l(z) = \sum_{i=1}^{3} b_i \; z^{-pp+2-i} \; ,$$

where pp is the pitch period, and b_i is the i-th long-term predictor coefficient;

$$N_l(z) - 1 = \lambda\, z^{-pp},$$

where λ is the long-term noise feedback filter coefficient calculated in Section 5.6.

Using the filter structure in Figure 9, block 313 in Figure 8 performs excitation VQ codebook search one excitation vector at a time. Each excitation vector contains four samples. The excitation gain $gq(m)$ is updated once a frame. Each frame contains 10 excitation vectors. Therefore, for each frame, the same scaled VQ codebook is used in 10 separate VQ codebook searches corresponding to the 10 excitation vectors in that frame.

Let n = 1, 2, 3, 4 denote the sample time indices corresponding to the current four–dimensional excitation vector. Before the excitation VQ codebook search for the current excitation vector starts, the high-pass filtered input $s(n)$, n = 1, 2, 3, 4 has been calculated in Section 5.1. In addition, before the VQ codebook search starts, the initial filter states (also called "filter memory") of the four filters in Figure 9 (blocks 240, 250, 260, and 265) are also known. All the other signals in Figure 9 are not determined yet for n = 1, 2, 3, 4.

The basic ideas of the excitation VQ codebook search are explained below. Refer to Figure 9. Block 248 stores the N scaled shape codevectors, where N = 16. Counting also the negated version of each scaled shape codevector, it is equivalent to having 2N scaled codevectors available for excitation VQ. From these 2N scaled codevectors, block 248 puts out one scaled codevector at a time as $uq(n)$, n = 1, 2, 3, 4. With the initial filter memories in blocks 240, 250, 260, and 265 set to what were left after vector-quantizing the last excitation vector, this $uq(n)$ vector then "drives" the rest of the filter structure until the corresponding quantization error vector $q(n)$, n = 1, 2, 3, 4 is obtained. The energy of this $q(n)$ vector is calculated and stored. This process is repeated for each of the 2N scaled codevectors, with the filter memories reset to their initial values before the process is repeated each time. After all 2N codevectors have been tried, the scaled codevector that minimizes the energy of the quantization error vector $q(n)$, n = 1, 2, 3, 4 is selected as the winning scaled codevector and is used as the VQ output vector. The corresponding output VQ codebook index is a 5-bit index consisting of a sign bit as the most significant bit (MSB), followed by 4 shape bits. If the winning scaled codevector is a negated version of a scaled shape codevector, then the sign bit is 1, otherwise, the sign bit is 0. The 4 shape bits are simply the binary representation of the codebook index of the winning shape codevector, as defined in Appendix 7. Note that there are 10 such excitation codebook indices in a frame, since each frame has 10 excitation vectors. These 10 indices are grouped in an excitation codebook index array, denoted as $CI = \{CI(1), CI(2), ..., CI(10)\}$, where $CI(k)$ is the excitation codebook index for the k-th excitation vector in the current frame. This excitation codebook index array CI is passed to the bit multiplexer block 295 in Figure 3.

Given a $uq(n)$ vector (taking the value of one of the 2N scaled codevectors), the way to derive the corresponding energy of the $q(n)$ vector is now described in more detail below. First, block 260 performs pitch prediction to produce the pitch-predicted vector $ppv(n)$ as

$$ppv(n) = \sum_{i=1}^{3} b_i\, dq(n - pp + 2 - i)\ ,\ n = 1, 2, 3, 4.$$

Adder 285 then updates the $dq(n)$ vector as

$$dq(n) = uq(n) + ppv(n) , n = 1, 2, 3, 4.$$

Next, block 240 and adder 245 together calculate short-term predicted speech vector $sp(n)$ and quantized speech vector $sq(n)$ as follows.

For $n = 1, 2, 3, 4$, calculate $sp(n)$ and $sq(n)$ as follows:

$$sp(n) = -\sum_{i=1}^{8} \tilde{a}_i \; sq(n-i)$$

$$sq(n) = dq(n) + sp(n)$$

Then, block 250 and adders 290, 253, and 255 work together to update the $v(n)$ vector as follows.

For $n = 1, 2, 3, 4$, calculate $stnf(n)$ and $v(n)$ as follows:

$$stnf(n) = \sum_{i=1}^{8} \hat{\beta}_i \; [v(n-i) - dq(n-i)] - \sum_{i=1}^{8} \hat{\alpha}_i \; stnf(n-i)$$

$$v(n) = s(n) - sp(n) - stnf(n)$$

Finally, the corresponding $q(n)$ vector is calculated as

$$q(n) = v(n) - ppv(n) - \lambda \, q(n-pp) - uq(n) , n = 1, 2, 3, 4.$$

The energy of the $q(n)$ vector is calculated as

$$E_q = \sum_{n=1}^{4} q^2(n) .$$

Such calculation from a given $uq(n)$ vector to the corresponding energy term E_q is repeated $2N$ times for the $2N$ scaled VQ codevectors. After the winning scaled codevector that minimizes the E_q term is selected, the filter memories of blocks 240, 250, 260, and 265 are updated by using the filter memories that were left after the calculation of the E_q term for that particular winning codevector was done. Such updated filter memories become the initial filter memories used for the excitation VQ codebook search for the next excitation vector.

5.10 Bit Multiplexing

The bit multiplexer block 295 in Figure 3 packs the five sets of indices *LSPI, PPI, PPTI, GI,* and *CI* into a single bit stream. This bit stream is the output of the BraodVoice16 encoder. It is passed to the communication channel.

Figure 10 shows the BV16 bit stream format in each frame. In Figure 10, the bit stream for the current frame is the shaded area in the middle. The bit stream for the last frame is on the left, while the bit stream for the next frame is on the right. Although the bit stream of different frames may not be sent next to each other in a packet voice system, this illustration is meant to show that time goes from left to right, and the 30 side information bits consisting of *LSPI, PPI, PPTI,* and *GI* goes before the excitation codebook indices *CI*(*k*), *k* =1, 2, …, 10 when the bit stream is transmitted in a serial manner. Note that for each index, the most significant bit (MSB) goes first (on the left), while the least significant bit (LSB) goes last.

This completes the detailed description of the BV16 encoder.

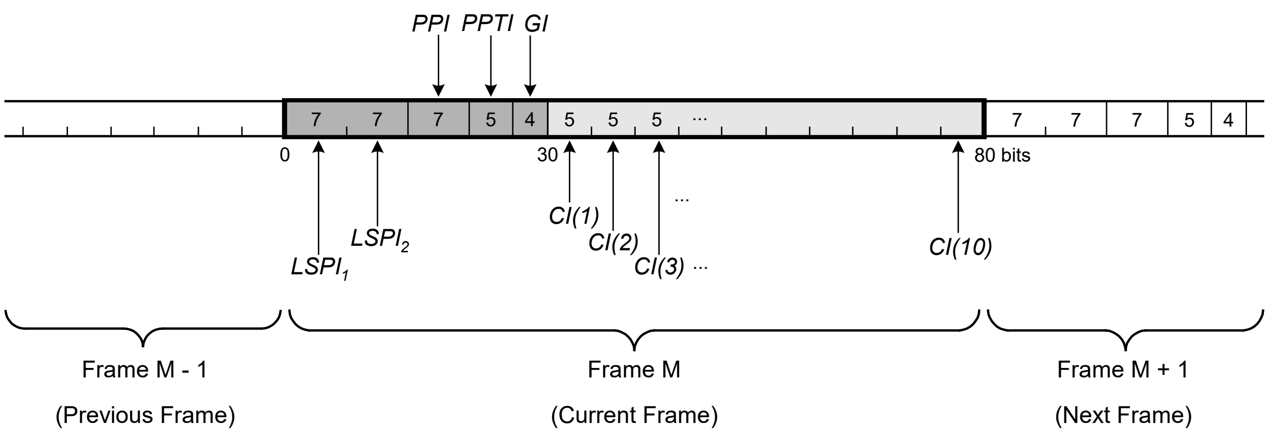

Figure 10 Bit stream format

6 DETAILED DESCRIPTION OF THE BV16 DECODER

This section gives a detailed description of each functional block in the BV16 decoder shown in Figure 4. Those blocks or signals that have the same labels as their counterparts in the encoder of Figure 3 have the same meaning as those counterparts.

6.1 Bit De-multiplexing

The bit de-multiplexer block 400 takes one frame of input bit stream at a time, and de-multiplexes, or separates, the five sets of indices *LSPI, PPI, PPTI, GI,* and *CI* from the current frame of input bit stream. As described in Section 5 above, *LSPI* contains two indices: a 7-bit first-stage VQ index and a 7-bit second-stage VQ index. *PPI* is a 7-bit pitch period index. *PPTI* is a 5-bit pitch predictor tap VQ index. *GI* is a 4-bit gain index, and *CI* contains ten 5-bit excitation VQ indices, each with 1 sign bit and 4 shape bits.

6.2 Long-Term Predictor Parameter Decoding

The long-term predictor parameter decoder (block 410) decodes the indices *PPI* and *PPTI*. The pitch period is decoded from *PPI* as

$$pp = PPI + 10$$

Let $\{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{31}\}$ be the 3-dimensional, 32-entry codebook used for pitch predictor tap VQ, as listed in Appendix 4. Let $\mathbf{b}_j$ be the j-th codevector in this codebook, where the subscript j is the codebook index listed in the first column of the table in Appendix 4. The three pitch predictor taps b_1, b_2, and b_3 are decoded from *PPTI* as

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{b}_{PPTI} \quad .$$

6.3 Short-Term Predictor Parameter Decoding

The short-term predictor parameter decoding takes place in block 420 of Figure 4. Block 420 receives the set of decoded LSP indices, $LSPI = \{LSPI_1, LSPI_2\}$, from the bit de-multiplexer, block 400 in Figure 4. First, block 420 reconstructs the LSP coefficients, $\{\tilde{l}_i\}$, from the LSP indices, and then it produces the coefficients of the short-term prediction error filter, $\{\tilde{a}_i\}$, from the LSP coefficients according to the conversion procedure specified in Section 5.5.

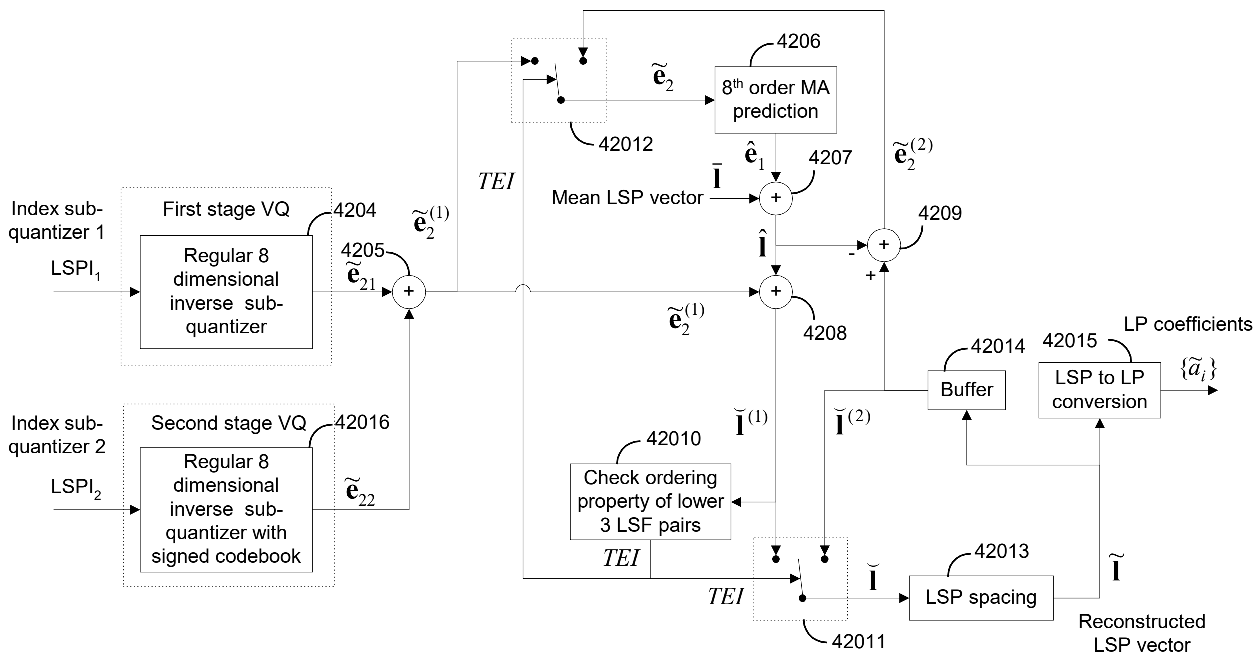

Figure 11 Short-term predictor parameter decoder (block 420)

Block 420 of Figure 4 is expanded in Figure 11. The reconstruction of the LSP coefficients from the LSP indices is the inverse of the LSP quantization, and many operations have equivalents in Section 5.4 and Figure 6. The first-stage VQ is decoded in block 4204, and the second-stage split VQ is decoded in block 42016.

In block 42016, the received index $LSPI_2$ is decoded into the sign index,

$$I_{sg} = \begin{cases} 0, & LSPI_2 > 63 \\ 1, & LSPI_2 \le 63 \end{cases},$$

and the shape vector index,

$$I_{sh} = \begin{cases} 127 - LSPI_2, & LSPI_2 > 63 \\ LSPI_2, & LSPI_2 \le 63 \end{cases}.$$

From the sign and shape indices the reconstructed output of the second stage VQ is calculated as

$$\tilde{\mathbf{e}}_{22} = s_{I_{sg}} \mathbf{cb}_2^{(I_{sh})}.$$

From the index for the first stage VQ, block 4204 looks up the quantized first stage vector from the codebook $\mathbf{CB}_1 = \{\mathbf{cb}_1^{(0)}, \mathbf{cb}_1^{(1)}, \ldots, \mathbf{cb}_1^{(127)}\}$,

$$\tilde{\mathbf{e}}_{21} = \mathbf{cb}_1^{(LSPI_1)}.$$

Adder 4205 performs the equivalent operation of Adder 21611 in Figure 6. It adds the first-stage and second-stage vectors to obtain a first reconstructed prediction error vector,

$$\tilde{\mathbf{e}}_2^{(1)} = \tilde{\mathbf{e}}_{21} + \tilde{\mathbf{e}}_{22}.$$

Equivalent to block 2163 in Figure 6, block 4206 performs the 8th-order MA prediction of the mean-removed LSP vector according to

$$\hat{e}_{1,i} = \mathbf{p}_{LSP,i}{}^{\mathrm{T}} \left[\tilde{e}_{2,i}(1) \quad \tilde{e}_{2,i}(2) \quad \tilde{e}_{2,i}(3) \quad \tilde{e}_{2,i}(4) \quad \tilde{e}_{2,i}(5) \quad \tilde{e}_{2,i}(6) \quad \tilde{e}_{2,i}(7) \quad \tilde{e}_{2,i}(8)\right]^T, \ i = 1, 2, \ldots, 8,$$

where $\tilde{e}_{2,i}(k)$ and $\mathbf{p}_{LSP,i}$ are defined in Section 5.4. Adder 4207, equivalent to Adder 21612 in Figure 6, generates the predicted LSP vector by adding the mean LSP vector and the predicted mean-removed LSP vector,

$$\hat{\mathbf{l}} = \bar{\mathbf{l}} + \hat{\mathbf{e}}_1.$$

Subsequently, adder 4208 adds the predicted LSP vector to the first reconstructed prediction error vector to obtain a first intermediate reconstructed LSP vector,

$$\breve{\mathbf{l}}^{(1)} = \hat{\mathbf{l}} + \tilde{\mathbf{e}}_2^{(1)}.$$

Adder 4209 subtracts the predicted LSP vector from a second intermediate reconstructed LSP $\breve{\mathbf{l}}^{(2)}$, to calculate a second reconstructed prediction error vector

$$\tilde{\mathbf{e}}_2^{(2)} = \breve{\mathbf{l}}^{(2)} - \hat{\mathbf{l}} \quad ,$$

to be used to update the MA predictor memory in the presence of bit-errors. Block 42010 determines the ordering property of the first 3 first intermediate reconstructed LSP coefficients,

$$\begin{aligned} \breve{l}_1^{(1)} &\geq 0 \\ \breve{l}_2^{(1)} &\geq \breve{l}_1^{(1)}, \\ \breve{l}_3^{(1)} &\geq \breve{l}_2^{(1)} \end{aligned}$$

This ordering property was enforced during the encoding operation of the constrained VQ of the second stage, block 21615 of Figure 6. If the ordering is found to be preserved, the *Transmission-Error-Indicator*, *TEI* , is set to 0 to indicate that no bit-errors in the LSP bits have been detected. Otherwise, if it is not preserved, the *Transmission-Error-Indicator* is set to 1 to indicate the likely presence of bit-errors in the LSP bits.

If the *Transmission-Error-Indicator* is 0, the switches 42011 and 42012 are in the left position, and they route the first reconstructed prediction error vector $\tilde{\mathbf{e}}_2^{(1)}$ and the first intermediate reconstructed LSP vector $\breve{\mathbf{l}}^{(1)}$ to the reconstructed prediction error vector $\tilde{\mathbf{e}}_2$ and the intermediate reconstructed LSP vector $\breve{\mathbf{l}}$, respectively. Otherwise, if the *Transmission-Error-Indicator* is 1, the switches 42011 and 42012 are in the right position, and they route the second reconstructed prediction error vector $\tilde{\mathbf{e}}_2^{(2)}$ and the second intermediate reconstructed LSP vector $\breve{\mathbf{l}}^{(2)}$ to the reconstructed prediction error vector $\tilde{\mathbf{e}}_2$ and the intermediate reconstructed LSP vector $\breve{\mathbf{l}}$, respectively. Hence, the reconstructed prediction error vector and the intermediate reconstructed LSP vector are obtained as

$$\tilde{\mathbf{e}}_2 = \begin{cases} \tilde{\mathbf{e}}_2^{(1)}, & \text{if } TEI = 0 \\ \tilde{\mathbf{e}}_2^{(2)}, & \text{if } TEI = 1 \end{cases}$$

and

$$\breve{\mathbf{l}} = \begin{cases} \breve{\mathbf{l}}^{(1)}, & \text{if } TEI = 0 \\ \breve{\mathbf{l}}^{(2)}, & \text{if } TEI = 1 \end{cases},$$

respectively. Block 42013 enforces LSP spacing; it is functionally identical to block 21614 in Figure 6, as specified in Section 5.4. Block 42014 buffers the reconstructed LSP vector for future use in the presence of bit-errors. The reconstructed LSP vector of the current frame becomes the second intermediate reconstructed LSP vector of the next frame,

$$\breve{\mathbf{l}}^{(2)}(k+1) = \tilde{\mathbf{l}}(k),$$

where the additional parameter k here represents the frame index of the current frame. For the very first frame the second intermediate reconstructed LSP vector is initialized to

$$\breve{\mathbf{l}}^{(2)} = \begin{bmatrix} 1/9 & 2/9 & \ldots & 8/9 \end{bmatrix}^T$$

The final step of the short-term predictor parameter decoding is to convert the reconstructed LSP coefficients to linear prediction coefficients. This operation takes place in block 42015, which is functionally identical to block 217 of Figure 5, described in Section 5.5.

6.4 Excitation Gain Decoding

The excitation gain decoder is shown in Figure 12. It is part of block 430 in Figure 4. It decodes the gain index in *GI* into the corresponding decoded frame excitation gain $gq(m)$ in the linear domain. All operations in Figure 12 are performed on a frame-by-frame basis.

Refer to Figure 12. Let m be the frame index of the current frame, and assume the same convention for the frame index m as in Section 5.8. Block 501 decodes the 4-bit gain index GI_m into the log-gain prediction error $lgeq(m)$ using the codebook in Appendix 5. Switch 502 is normally in the upper position, connecting the output of block 501 to the input of block 503. Then, the MA log-gain predictor (block 503) calculates the estimated log-gain for the current frame as

$$elg(m) = \sum_{k=1}^{GPO} lgp(k) lgeq(m-k) \ ,$$

where GPO = 8, and $lgp(k)$, k = 1, 2, …, GPO are the MA log-gain predictor coefficients given in Section 5.8.

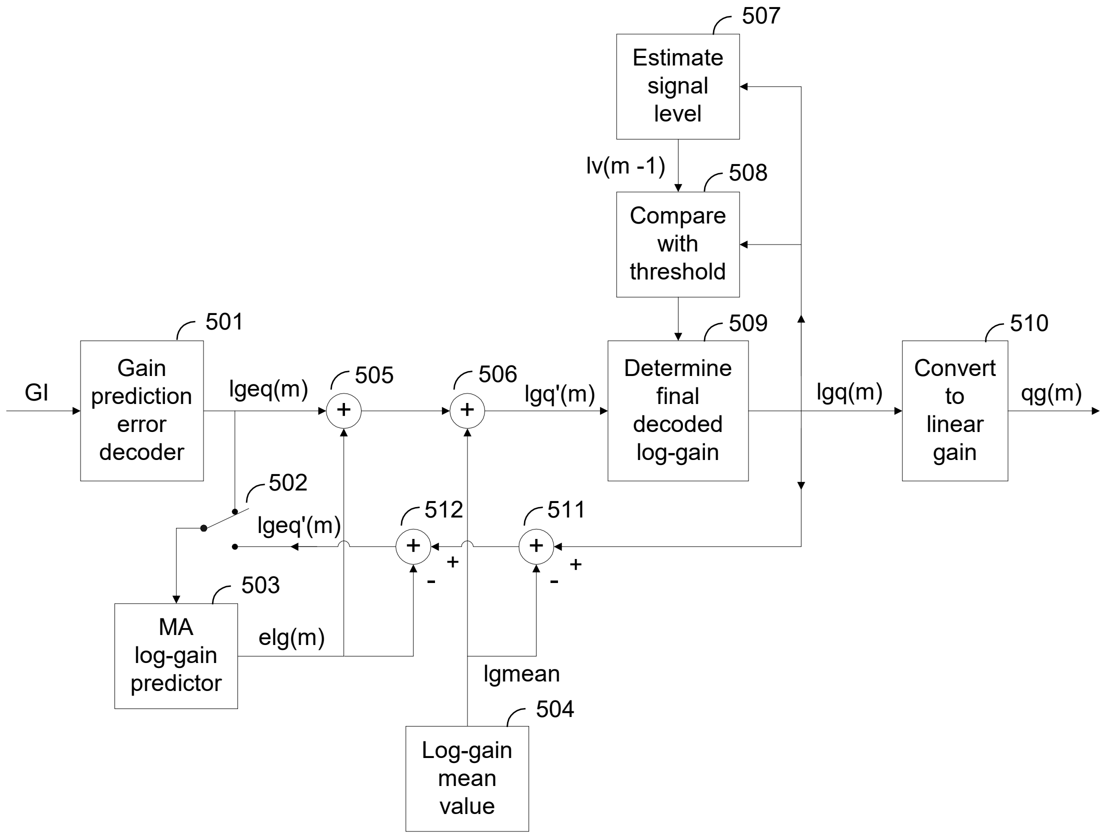

Figure 12 Excitation gain decoder

Block 504 holds the long-term average log-gain value $lgmean$ = 11.45752. Adders 505 and 506 adds $elg(m)$ and $lgmean$, respectively, to $lgeq(m)$, resulting in the temporarily decoded log-gain of

$$lgq'(m) = lgeq(m) + elg(m) + lgmean \ .$$

Block 507 is functionally identical to block 309 in Figure 8, described in Section 5.8. It is important to note that equivalently to the encoder, the log-gain value passed to block 507 for updating its estimate of the long-term average signal level is the final value of the decoded log-gain $lgq(m)$, i.e. after the threshold check of block 508 and potential log-gain extrapolation and substitution of block 509, respectively, as described below.

Block 508 calculates the row and column indices i and j into the threshold matrix $T(i, j)$ in the same way as block 310 in Figure 8. Namely, the row index is calculated as

$$i = \left\lceil \frac{lgq(m-1) - lv(m-1) - GLB}{2} \right\rceil,$$

where $GLB = -24$. If $i > NG$, i is clipped to NG. If $i < 1$, i is clipped to 1. The column index is calculated as

$$j = \left\lceil \frac{lgq(m-1) - lgq(m-2) - GCLB}{2} \right\rceil,$$

where $GCLB = -8$. If $j > NGC$, j is clipped to NGC. If $j < 1$, j is clipped to 1.

Block 508 controls the actions of block 509 and switch 502 in the following way. If $GI_m = 0$ or $lgq'(m) \le T(i, j) + lgq(m-1)$, then switch 502 is in the upper position, block 509 determines the final decoded log-gain as

$$lgq(m) = lgq'(m) \ ,$$

and the filter memory in the MA log-gain predictor (block 503) is updated by shifting the old memory values by one position, and then assigning $lgeq(m)$ to the newest position of the filter memory.

If, on the other hand, $GI_m > 0$ and $lgq(m) > T(i, j) + lgq(m-1)$, then the temporarily decoded log-gain $lgq'(m)$ is discarded, block 509 determines the final decoded log-gain as

$$lgq(m) = lgq(m-1)$$

(by extrapolating the decoded log-gain of the last sub-frame); furthermore, switch 502 is moved to the lower position, adders 511 and 512 subtract $lgmean$ and $elg(m)$, respectively, from $lgq(m)$ to get

$$lgeq'(m) = lgq(m) - lgmean - elg(m),$$

and this $lgeq'(m)$ is used to update the newest position of the filter memory of block 503, after the old memory values are shifted by one position.

Once the final decoded log-gain $lgq(m)$ subject to the constraint imposed by block 509 is determined as described above, it is used by block 508 to update the estimated signal level $lv(m)$. This value $lv(m)$ is then used by block 509 in the next frame (the $(m + 1)$-th frame).

Block 510 converts final decoded log-gain $lgq(m)$ to the linear domain as $gq(m) = 2^{\frac{lgq(m)}{2}}$.

6.5 Excitation VQ Decoding and Scaling

The excitation codebook index array *CI* of each frame contains 10 excitation codebook indices, *CI*(*k*), *k* = 1, …, 10, each containing 1 sign bit and 4 shape bits. The excitation vectors are decoded vector-by-vector.

Let *gq*(*m*) denote the decoded excitation gain in the linear domain for the current frame. In addition, let *CI*(*k*) denote the received excitation codebook index of the current excitation vector that needs to be decoded. This index assumes a value between 0 and 31. The most significant bit of this index is the sign bit. Therefore, if *CI*(*k*) < 16, the sign bit is 0; otherwise, the sign bit is 1. Let $c_j(n)$, *n* = 1, 2, 3, 4 represent the *j*-th shape codevector in Appendix 7, with a shape codebook index of *j*. Furthermore, without loss of generality, let *n* = 1, 2, 3, 4 correspond to the sample time indices of the current vector. Then, in Figure 4, the decoded and scaled excitation vector, or *uq*(*n*), *n* = 1, 2, 3, 4, is obtained as

$$uq(n) = \begin{cases} gq(m)\, c_{CI(k)}(n),\ n = 1,2,3,4, & \text{if } CI(k) < 16 \\ -gq(m)\, c_{CI(k)-16}(n),\ n = 1,2,3,4, & \text{if } CI(k) \geq 16 \end{cases}$$

6.6 Long-Term Synthesis Filtering

Let *n* = 1, 2, …, *FRSZ* correspond to the sample time indices of the current frame. In Figure 4, the long-term synthesis filter (block 455, consisting of block 440 and adder 450 in a feedback loop) performs sample-by-sample long-term synthesis filtering as follows.

$$dq(n) = uq(n) + \sum_{i=1}^{3} b_i\, dq(n - pp + 2 - i),\ n = 1, 2, \ldots\ FRSZ.$$

6.7 Short-Term Synthesis Filtering

The short-term synthesis filter (block 475, consisting of block 460 and adder 470 in a feedback loop) performs sample-by-sample short-term synthesis filtering to obtain the output signal as follows.

$$sq(n) = dq(n) - \sum_{i=1}^{8} \tilde{a}_i\, sq(n-i),\ n = 1, 2, \ldots\ FRSZ.$$

6.8 Example Postfilter

This document specifies codec components that need to be clearly specified in order to foster inter-operability. Decoder postfiltering is not a mandatory component of this BV16 Codec Specification, since such postfiltering does not affect bit-stream compatibility or encoder-decoder inter-operability. However, an example postfilter is described in this section for reference purposes only. An implementer of BV16 can utilize other postfilters without affecting inter-operability.

The example postfilter is an all-zero single tap pitch postfilter. The input to the pitch postfilter is the pitch period, *pp*, and the output signal, *sq*(*n*), from the short-term synthesis filter[7]. In principle, the postfiltering is given by

$$spf(n) = b_{pf}(1)\, sq(n) + b_{pf}(2)\, sq(n - pppf)\,,\ n = 1, 2, \ldots FRSZ,$$

where *spf*(*n*) denotes the postfiltered output signal and *pppf* is the pitch period used for the pitch postfilter.

First the pitch period of the decoder is refined by selecting the lag, *pppf*, corresponding to the highest squared normalized pitch correlation of the output signal in a ±4 sample range of the pitch period, *pp*, i.e. the lag, *pppf*, that maximizes,

$$Csq(pppf) = \frac{\left[\sum_{n=1}^{FRSZ} sq(n)\, sq(n-pppf)\right]^2}{\left[\sum_{n=1}^{FRSZ} sq(n)\, sq(n)\right]\left[\sum_{n=1}^{FRSZ} sq(n-pppf)\, sq(n-pppf)\right]}\,,\ pppf = pp_{min}, pp_{min}+1, \ldots, pp_{max},$$

where $pp_{min} = pp$-4 and $pp_{max} = pp$+4, with the following constraints:

if $pp_{min} < MINPP$: $pp_{min} = MINPP$, $pp_{max} = MINPP+8$, and similarly

if $pp_{max} > MAXPP$: $pp_{max} = MAXPP$, $pp_{min} = MAXPP-8$.

With the refined lag the normalized pitch correlation is calculated as

$$Cpf = \frac{\left[\sum_{n=1}^{FRSZ} sq(n)\, sq(n-pppf)\right]}{\sqrt{\left[\sum_{n=1}^{FRSZ} sq(n)\, sq(n)\right]\left[\sum_{n=1}^{FRSZ} sq(n-pppf)\, sq(n-pppf)\right]}}$$

If the numerator is less than zero or the denominator is zero, the normalized pitch correlation is set to zero, $Cpf = 0$. Next, a running mean of the normalized pitch correlation is calculated as

$$Crm(m) = 0.75\, Crm(m-1) + 0.25\, Cpf\,,$$

where *Crm*(*m*) is the running mean of the current frame, and *Crm*(*m-1*) is the running mean of the previous frame[8]. Based on the normalized pitch correlation and the running mean of the normalized pitch correlation, the initial pitch postfilter tap is calculated as

[7] At the first frame, the history of *sq*(*n*) is set to zero.

$$a_{pf} = \begin{cases} 0 & Crm(m) < 0.55 \text{ and } Cpf < 0.8 \\ 0.3\ Cpf & \text{otherwise} \end{cases}.$$

Subsequently, a scaling factor is calculated as

$$g_{pf} = \sqrt{\frac{\sum_{n=1}^{FRSZ}[sq(n)]^2}{\sum_{n=1}^{FRSZ}[sq(n) + a_{pf}\ sq(n - pppf)]^2}}.$$

It is set to one if either the numerator or the denominator is zero. The two pitch postfilter coefficients of the current (m-th) frame is calculated as

$$b_{pf,m}(1) = g_{pf} \text{ and } b_{pf,m}(2) = g_{pf}\ a_{pf}.$$

In practice, for the first $Lint$=20 samples of each frame, the impulse responses of adjacent pitch postfilters are interpolated while the pitch postfilter of the current frame is used for the remaining samples of the frame:

$$spf(n) = b_{pf}(1,n)\ sq(n) + b_{pf}(2,n)\ sq(n - pppf_m) + b_{pf}(3,n)\ sq(n - pppf_{m-1}),\ n = 1, 2, \ldots\ FRSZ,$$

where $pppf_m$ and $pppf_{m-1}$ are the refined pitch period of the current and previous frames, respectively, and

$$\begin{aligned} b_{pf}(1,n) &= \begin{cases} \alpha(n)\ b_{pf,m}(1) + [1-\alpha(n)]\ b_{pf,m-1}(1) & n \le Lint \\ b_{pf,m}(1) & n > Lint \end{cases} \\ b_{pf}(2,n) &= \begin{cases} \alpha(n)\ b_{pf,m}(2) & n \le Lint \\ b_{pf,m}(2) & n > Lint \end{cases} \\ b_{pf}(3,n) &= \begin{cases} [1-\alpha(n)]\ b_{pf,m-1}(2) & n \le Lint \\ 0 & n > Lint \end{cases} \end{aligned}.$$

A linear interpolation between adjacent pitch postfilters[9] is used:

$$\alpha(n) = \frac{n}{Lint + 1}.$$

6.9 Example Packet Loss Concealment

[8] For the first frame, running mean of the previous frame is set to zero, i.e. $Crm(0)=0$.
[9] For the first frame, the parameters of the previous pitch postfilter are set to $pppf_0$=100, $b_0(1)$=1, $b_0(2)$=0.

Similar to decoder postfiltering, packet loss concealment is not a mandatory component of this BV16 Codec Specification, since packet lost concealment does not affect bit-stream compatibility or encoder-decoder inter-operability. However, an example packet loss concealment technique is described in this section for reference purposes only. An implementer of BV16 can utilize other packet loss concealment techniques without affecting inter-operability.

The example packet loss concealment technique utilizes the synthesis model of the decoder. In principle, all side information of the previous frame is repeated while the excitation of the cascaded long-term and short-term synthesis filters is from a random source, scaled to a proper level. Hence, with the additional index *m* denoting the m-th frame, during packet-loss:

- The pitch period, *pp*, is set to the pitch period of the last frame[10]:
  $pp = pp_{m-1}$.
- The pitch taps, b_1 b_2 and b_3, are set to the pitch taps of the last frame[11].
  $b_i = b_{m-1,i}$, i=1,2,3.
- The short-term synthesis filter coefficients, $\tilde{a}_i, i = 1,...,8$, are set to those of the last frame[12]:
  $\tilde{a}_i = \tilde{a}_{m-1,i}$, i=1,…,8.
- A properly scaled random sequence is used as long-term synthesis filter excitation, $uq(n)$, $n = 1, 2, \ldots FRSZ$.

The speech synthesis of the bad frame (part of lost packet) now takes place exactly as specified in Sections 6.6, 6.7, and 6.8 if the example postfilter is included.

The random sequence is scaled according to

$$uq(n) = g_{plc} \cdot \sqrt{\frac{E_{m-1}}{\sum_{n=1}^{FRSZ} [r(n)]^2}} \cdot r(n) \text{, } n = 1, 2, \ldots FRSZ,$$

where $r(n)$, $n = 1, 2, \ldots FRSZ$, is a random sequence, E_{m-1} is in principle the energy of the long-term synthesis filter excitation of the previous frame[13], and the scaling factor, g_{plc}, is calculated as detailed below.

During good frames an estimate of periodicity is updated as

$$per_m = 0.5\, per_{m-1} + 0.5\, bs \text{ ,}$$

[10] If the first frame is lost a value of 100 is used for the pitch period.
[11] If the first frame is lost the pitch taps are set to zero.
[12] If the first frame is lost the short-term filter coefficients are set to zero.
[13] The energy is initialized to zero, i.e. E_0=0.

where bs is the sum of the three pitch taps clipped at a lower threshold of zero and an upper threshold of one[14], while it is maintained during bad frames: $per_m = per_{m-1}$. Based on the periodicity the scaling factor is calculated as

$$g_{plc} = -2\ per_{m-1} + 1.9$$

with g_{plc} clipped at a lower threshold of 0.1 and an upper threshold of 0.9.

After synthesis of the signal output of a lost frame, memories of predictive quantizers are updated.

The memory of the inverse LSP quantizer is updated with

$$\tilde{e}_{2,i} = \tilde{I}_{m-1,i} - \hat{e}_{1,i} - \bar{I}_i,\ \text{i=1,2,\ldots,8},$$

where $\hat{e}_{1,i}$ is given in Section 6.3, $\bar{I}_i$ in Section 5.4, and $\tilde{I}_{m-1,i}$ denotes the i-th LSP coefficients of the (m-1)-th frame (as decoded according to Section 6.3 for a good frame, or repeated for a bad frame).

The memory of the inverse gain quantizer is updated with

$$lgeq(m) = lgq(m) - lgmean - elg(m),$$

where $elg(m)$ is given in Section 6.4, $lgmean$ in Section 5.8, and $lgq(m)$ is calculated as

$$lgq(m) = \begin{cases} \log_2 \dfrac{E_{m-1}}{FRSZ}, & \text{if } \dfrac{E_{m-1}}{FRSZ} > 1 \\ 0, & \text{if } \dfrac{E_{m-1}}{FRSZ} \le 1 \end{cases}.$$

The level estimation for a bad frame is updated exactly as for a good frame, see Section 6.4.

At the end of a good frame (after synthesis of the output) the estimate of periodicity is estimated as explained above, and the energy of the long-term synthesis filter excitation is updated as

$$E_m = \sum_{n=1}^{FRSZ} \left[uq(n)\right]^2.$$

At the end of the processing of a bad frame (after synthesis of the output and update of predictive quantizers), the energy of the long-term synthesis filter excitation and the long-term synthesis filter coefficients are scaled down when 8 or more consecutive frames are lost:

[14] The estimate of periodicity is initialized to zero, i.e. per_0=0.

$$E_m = \begin{cases} E_{m-1} & Nclf < 8 \\ \left(\beta_{Nclf}\right)^2 E_{m-1} & Nclf \geq 8 \end{cases},$$

$$b_{m,i} = \begin{cases} b_{m-1,i} & Nclf < 8 \\ \beta_{Nclf}\, b_{m-1,i} & Nclf \geq 8 \end{cases}, \quad \text{i=1,2,3,}$$

where *Nclf* is the number of consecutive lost frames, and the scaling, β_{Nclf} , is given by

$$\beta_{Nclf} = \begin{cases} 1 - 0.02\,(Nclf - 7) & 8 \leq Nclf \leq 57 \\ 0 & Nclf > 57 \end{cases}.$$

This will gradually mute the output signal when consecutive packets are lost for an extended period of time.

APPENDIX 1: GRID FOR LPC TO LSP CONVERSION

Grid point	Grid value
0	0.9999390
1	0.9935608
2	0.9848633
3	0.9725342
4	0.9577942
5	0.9409180
6	0.9215393
7	0.8995972
8	0.8753662
9	0.8487854
10	0.8198242
11	0.7887573
12	0.7558899
13	0.7213440
14	0.6853943
15	0.6481323
16	0.6101379
17	0.5709839
18	0.5300903
19	0.4882507
20	0.4447632
21	0.3993530
22	0.3531189
23	0.3058167
24	0.2585754
25	0.2109680
26	0.1630859
27	0.1148682
28	0.0657349
29	0.0161438
30	-0.0335693
31	-0.0830994
32	-0.1319580
33	-0.1804199
34	-0.2279663
35	-0.2751465
36	-0.3224487
37	-0.3693237
38	-0.4155884
39	-0.4604187
40	-0.5034180
41	-0.5446472
42	-0.5848999
43	-0.6235962
44	-0.6612244
45	-0.6979980
46	-0.7336731
47	-0.7675781
48	-0.7998962
49	-0.8302002
50	-0.8584290
51	-0.8842468
52	-0.9077148
53	-0.9288635
54	-0.9472046
55	-0.9635010
56	-0.9772034
57	-0.9883118
58	-0.9955139
59	-0.9999390

APPENDIX 2: FIRST-STAGE LSP CODEBOOK

Index	Element 1	Element 2	Element 3	Element 4	Element 5	Element 6	Element 7	Element 8
0	-0.0059814	-0.0075378	-0.0113449	-0.0002670	-0.0103607	-0.0055771	0.0091400	-0.0032730
1	-0.0053177	-0.0019302	0.0037079	-0.0106049	-0.0021820	-0.0003815	0.0100098	0.0037460
2	-0.0009308	-0.0001831	-0.0040741	-0.0110474	-0.0238800	-0.0042191	0.0014114	-0.0061035
3	0.0031128	0.0013046	0.0076218	-0.0042191	-0.0073776	-0.0045013	-0.0051651	0.0158539
4	-0.0023270	-0.0014496	-0.0036392	0.0071030	0.0026093	-0.0172119	-0.0009613	-0.0059662
5	-0.0081329	-0.0077362	0.0091782	0.0048294	0.0101395	0.0007172	0.0030212	0.0013885
6	0.0006104	0.0040817	-0.0010300	-0.0081787	-0.0126343	-0.0218582	-0.0007629	-0.0092163
7	0.0090561	0.0081329	0.0096436	0.0009613	0.0011063	-0.0042572	0.0038910	-0.0034485
8	-0.0044785	-0.0070572	-0.0158615	0.0019913	0.0087204	0.0005951	0.0022583	-0.0074539
9	0.0042114	0.0052719	-0.0061417	0.0057449	0.0057068	-0.0022202	0.0133896	0.0077362
10	-0.0039902	-0.0037308	-0.0103226	-0.0064774	-0.0049667	-0.0043411	-0.0066986	-0.0186844
11	0.0035553	0.0042877	0.0199356	0.0078812	0.0031281	-0.0082245	-0.0142746	-0.0015106
12	0.0032806	0.0013351	-0.0004501	0.0149384	0.0076141	0.0033264	-0.0038376	-0.0110245
13	0.0010910	0.0050964	0.0128632	0.0091553	0.0088348	0.0151443	0.0096664	0.0043411
14	-0.0047226	-0.0046234	0.0096664	0.0042496	-0.0064697	-0.0039902	-0.0056915	-0.0162430
15	-0.0000229	0.0000000	0.0265427	0.0128021	0.0049896	0.0054398	0.0008698	-0.0047150
16	-0.0074081	-0.0089569	-0.0175552	-0.0174561	-0.0057831	-0.0148010	0.0076141	0.0079803
17	-0.0019760	-0.0027161	-0.0077667	-0.0104675	-0.0090866	-0.0027542	0.0306244	0.0160751
18	-0.0044403	-0.0059509	-0.0128784	-0.0197525	-0.0304413	-0.0161133	0.0037613	0.0098114
19	-0.0001907	0.0020599	0.0160294	0.0045853	-0.0091476	-0.0058670	0.0226593	0.0125122
20	-0.0057526	-0.0060425	-0.0029755	-0.0092010	0.0054550	-0.0046692	-0.0137711	-0.0035477
21	-0.0022125	-0.0046158	-0.0083923	0.0117264	0.0248260	0.0126343	0.0082626	0.0001907
22	-0.0016632	0.0000076	-0.0051346	-0.0084305	-0.0128784	-0.0196915	-0.0223007	-0.0168076
23	0.0046158	0.0114517	0.0148926	0.0092087	0.0188599	-0.0058212	0.0079727	0.0046082
24	-0.0006714	-0.0006714	-0.0119095	-0.0186539	0.0112305	-0.0053024	0.0070267	-0.0016022
25	0.0114136	0.0131760	0.0045929	-0.0096207	0.0138092	0.0076675	0.0137863	0.0142441
26	0.0089951	0.0114975	-0.0246811	-0.0092545	-0.0067444	-0.0065155	-0.0055161	-0.0072098
27	0.0116730	0.0303574	0.0396042	0.0238495	0.0113144	0.0006714	-0.0080719	0.0067749
28	0.0061035	0.0072174	0.0028000	-0.0075989	0.0156174	0.0043716	-0.0073624	-0.0141525
29	0.0069580	0.0107727	0.0140839	0.0036621	0.0325394	0.0216980	0.0056152	0.0061188
30	0.0065002	0.0056458	0.0067139	0.0007935	0.0008087	-0.0099030	-0.0182724	-0.0288086
31	0.0147324	0.0161285	0.0276260	0.0238800	0.0214386	0.0131302	0.0047607	-0.0047836
32	-0.0083008	-0.0135345	-0.0167313	-0.0003433	-0.0090408	-0.0008469	-0.0017624	0.0161667
33	-0.0031662	-0.0056992	-0.0011444	0.0063324	-0.0090790	0.0121918	0.0022354	0.0048523
34	-0.0050354	-0.0077744	-0.0103531	-0.0145035	-0.0191193	-0.0035934	-0.0159454	0.0042343
35	0.0078888	0.0054169	0.0038223	-0.0016632	-0.0109177	-0.0039520	-0.0170212	-0.0018616
36	-0.0038910	-0.0082321	-0.0112686	0.0100861	-0.0043945	-0.0049820	-0.0151062	0.0018616
37	-0.0030060	-0.0051117	0.0013962	0.0250015	-0.0003738	-0.0045395	0.0120697	0.0071411
38	-0.0017471	-0.0031509	-0.0094299	-0.0154495	-0.0188980	-0.0264816	-0.0149384	0.0071030
39	0.0070190	0.0111084	0.0142746	0.0070648	-0.0085373	-0.0219345	0.0042267	0.0029221
40	-0.0084000	-0.0120621	-0.0198364	-0.0063629	0.0110550	0.0045700	0.0082169	0.0152664
41	-0.0012970	-0.0023575	0.0041809	0.0055084	0.0066299	0.0041122	0.0141602	0.0310822
42	-0.0008011	0.0027390	-0.0027847	-0.0278168	-0.0051651	-0.0065536	-0.0094833	-0.0070724
43	0.0257950	0.0224075	0.0190277	0.0123291	0.0018692	-0.0124512	-0.0261765	-0.0093994
44	0.0024414	0.0011520	-0.0020218	0.0018616	0.0149918	0.0050735	-0.0103073	0.0105972
45	0.0045166	0.0086136	0.0284348	0.0160980	0.0127563	0.0124054	0.0261307	0.0190277
46	-0.0064392	-0.0072556	0.0081406	0.0079956	-0.0225372	-0.0159760	-0.0059891	-0.0012741
47	-0.0008316	0.0018845	0.0423431	0.0217514	0.0008698	-0.0041199	0.0085602	0.0102158
48	-0.0227127	-0.0309753	-0.0029831	-0.0045471	-0.0044708	-0.0003662	0.0006409	0.0024567
49	0.0003204	-0.0007782	0.0007553	-0.0061646	-0.0099792	0.0272598	0.0179977	0.0155029
50	-0.0022583	-0.0034180	-0.0074692	-0.0160370	-0.0401917	-0.0083847	-0.0189896	-0.0101929
51	0.0022202	0.0045013	0.0243607	0.0083466	-0.0246048	0.0046997	-0.0021439	0.0023041
52	-0.0029678	-0.0052338	0.0025406	0.0110321	0.0029221	-0.0056763	-0.0311356	-0.0081024
53	0.0019226	0.0010529	0.0046844	0.0322113	0.0202255	0.0150070	0.0069733	0.0021973
54	0.0002441	0.0029984	0.0021286	0.0054932	-0.0150223	-0.0383453	-0.0137787	-0.0153046
55	0.0024185	0.0418625	0.0316925	0.0256805	0.0141296	0.0077591	0.0154495	0.0091095
56	-0.0076904	-0.0126266	-0.0251846	-0.0261307	0.0040588	0.0132675	0.0196609	0.0226059
57	0.0066910	0.0088730	0.0157623	0.0102997	0.0193558	0.0230255	0.0201874	0.0446930
58	-0.0050049	-0.0121231	-0.0460205	-0.0182266	-0.0260468	-0.0259018	-0.0209122	-0.0175323
59	0.0057602	0.0139847	0.0579147	0.0351944	-0.0040665	-0.0186386	-0.0284729	-0.0171432
60	0.0025711	0.0053101	0.0119553	0.0070419	0.0170135	0.0213165	-0.0242462	-0.0078735
61	0.0176849	0.0341110	0.0360947	0.0325394	0.0362167	0.0317612	0.0233765	0.0178757
62	0.0018082	0.0054245	0.0223770	0.0096283	-0.0214233	-0.0161209	-0.0263824	-0.0237961

63	0.0040436	0.0186539	0.0682678	0.0692520	0.0290146	0.0145493	0.0086975	0.0001144
64	-0.0073166	-0.0097580	-0.0165558	-0.0164719	-0.0054932	0.0104904	0.0003052	-0.0026093
65	0.0012283	0.0010452	-0.0012741	-0.0147095	0.0082169	0.0179520	0.0043182	0.0050583
66	-0.0025101	-0.0054626	-0.0107498	-0.0207672	-0.0277328	0.0124207	0.0075836	0.0025177
67	0.0082932	0.0077057	0.0032272	-0.0084229	-0.0114975	0.0151215	0.0005341	-0.0019226
68	-0.0067825	-0.0089111	-0.0184479	0.0017242	-0.0064545	-0.0217209	-0.0024490	-0.0019455
69	-0.0006104	-0.0049820	0.0121994	0.0176392	0.0069962	0.0066605	-0.0074310	0.0169830
70	-0.0000534	0.0012283	-0.0020981	-0.0062180	-0.0171661	-0.0240707	0.0151367	0.0081100
71	0.0176697	0.0149918	0.0140686	0.0097809	0.0034790	0.0105209	0.0014572	0.0027390
72	-0.0064240	-0.0083618	-0.0160828	-0.0105820	0.0212021	0.0123367	-0.0018921	-0.0081329
73	-0.0053406	-0.0044250	-0.0000076	0.0004807	0.0158310	0.0097198	0.0251846	0.0064545
74	-0.0037766	-0.0038528	-0.0154724	-0.0229874	0.0008011	0.0042114	-0.0171432	-0.0225220
75	0.0014191	0.0062637	0.0232925	0.0124817	0.0117035	0.0014648	-0.0106812	-0.0250015
76	-0.0061264	-0.0116348	-0.0150681	0.0248032	0.0146561	0.0051270	-0.0047836	-0.0073013
77	-0.0086670	-0.0107346	0.0193634	0.0210648	0.0206528	0.0170822	0.0147781	0.0120239
78	-0.0034256	-0.0035858	-0.0015869	-0.0019073	-0.0142975	0.0131302	-0.0151138	-0.0197067
79	-0.0020447	0.0006714	0.0414658	0.0249710	0.0197296	0.0175705	-0.0016098	0.0003967
80	-0.0080795	-0.0119095	-0.0216980	-0.0280533	-0.0105896	-0.0002365	-0.0009079	0.0161667
81	0.0123291	0.0056229	-0.0075455	-0.0211258	-0.0128326	0.0001068	0.0061417	0.0149689
82	-0.0029373	-0.0080795	-0.0181351	-0.0304947	-0.0477219	0.0010223	0.0070724	0.0156937
83	0.0141983	0.0176315	0.0056763	-0.0044098	-0.0164795	0.0085678	0.0159912	0.0168228
84	-0.0061722	-0.0097046	-0.0205307	-0.0133286	0.0058441	-0.0068512	-0.0228195	0.0043335
85	-0.0051880	-0.0111084	-0.0178680	0.0235138	0.0195084	0.0178680	0.0160370	0.0151443
86	0.0019531	-0.0010757	-0.0006256	-0.0082626	-0.0146942	-0.0227509	-0.0364304	0.0097427
87	0.0283661	0.0346222	0.0130768	0.0101700	0.0174866	0.0197144	0.0173874	0.0145874
88	-0.0066528	-0.0088272	-0.0246811	-0.0331345	0.0143738	0.0057602	-0.0002747	-0.0007629
89	0.0018539	0.0035934	-0.0006332	-0.0124893	0.0327225	0.0176163	0.0233994	0.0193710
90	-0.0135727	-0.0340042	-0.0894012	-0.0189590	-0.0093231	-0.0084381	-0.0090332	-0.0088577
91	0.0099640	0.0462646	0.0453796	0.0276489	0.0160370	0.0025406	-0.0106049	-0.0192184
92	-0.0034714	-0.0054245	-0.0114441	0.0039444	0.0178375	0.0054550	-0.0171051	-0.0267639
93	-0.0011978	0.0003204	0.0082169	0.0209274	0.0536499	0.0358963	0.0197830	0.0092850
94	0.0023346	0.0034943	-0.0014572	0.0014343	-0.0054932	-0.0127716	-0.0328522	-0.0487366
95	0.0088501	0.0262756	0.0493164	0.0377655	0.0499496	0.0279236	-0.0018158	-0.0138168
96	-0.0091782	-0.0182266	-0.0430679	0.0019302	0.0016556	0.0016785	-0.0005188	-0.0010605
97	-0.0042267	-0.0044327	-0.0011826	0.0068283	0.0034714	0.0279465	0.0020370	-0.0107651
98	-0.0088577	-0.0077744	-0.0206451	-0.0217667	-0.0166931	-0.0166321	-0.0041122	-0.0111160
99	0.0114822	0.0137863	0.0051422	0.0017090	-0.0126801	-0.0010223	-0.0035172	-0.0167770
100	-0.0047989	-0.0082703	-0.0190659	0.0137558	-0.0056305	-0.0116806	-0.0150452	-0.0211258
101	-0.0013351	-0.0021439	0.0114136	0.0434952	0.0037308	-0.0042496	-0.0063400	-0.0076370
102	-0.0027390	-0.0010986	-0.0103989	-0.0135803	-0.0289612	-0.0440826	0.0028915	0.0004730
103	0.0343170	0.0352859	0.0218124	0.0093842	-0.0054703	-0.0086594	0.0062637	0.0050507
104	-0.0057678	-0.0092545	-0.0224991	-0.0150681	0.0306778	0.0213089	0.0150299	0.0065155
105	-0.0029221	-0.0022583	0.0017624	0.0089951	0.0164185	0.0172653	0.0484390	0.0271988
106	-0.0026627	0.0002594	-0.0224304	-0.0537415	-0.0119095	-0.0128479	-0.0069809	-0.0067749
107	0.0076294	0.0422287	0.0109787	0.0002823	0.0060196	0.0020523	-0.0027847	-0.0075226
108	-0.0038986	-0.0030670	0.0042572	0.0127869	0.0384750	0.0027542	-0.0117798	-0.0054169
109	-0.0036087	0.0035782	0.0424576	0.0331345	0.0332794	0.0281830	0.0280609	0.0207291
110	0.0046082	0.0080490	0.0111084	-0.0065918	-0.0519485	-0.0157547	-0.0019455	-0.0029144
111	0.0034485	0.0175858	0.0821762	0.0272980	0.0156860	0.0159531	0.0184174	0.0143356
112	-0.0314102	-0.0573120	-0.0361557	-0.0185471	-0.0215912	-0.0093765	-0.0097656	-0.0018997
113	0.0007401	0.0014343	0.0100021	0.0084229	0.0087585	0.0477676	0.0178299	0.0107193
114	-0.0015106	-0.0081024	-0.0302200	-0.0461807	-0.0730972	-0.0303497	-0.0201721	-0.0060349
115	0.0099411	0.0259781	0.0389252	-0.0061798	-0.0111618	0.0061264	0.0084991	-0.0016098
116	-0.0020447	-0.0056534	-0.0065460	-0.0004425	-0.0104218	-0.0178070	-0.0585556	-0.0127106
117	-0.0028152	-0.0029221	0.0108337	0.0616455	0.0292282	0.0216827	0.0112686	0.0042725
118	0.0102158	0.0152206	-0.0010757	-0.0186920	-0.0433731	-0.0584030	-0.0330887	-0.0182495
119	0.0341263	0.0759430	0.0464096	0.0330963	0.0261612	0.0209045	0.0158920	0.0113602
120	-0.0082016	-0.0157471	-0.0547714	-0.0575638	0.0062866	0.0029297	0.0097733	0.0060959
121	0.0022659	0.0080261	0.0222321	0.0229797	0.0417252	0.0456924	0.0649796	0.0428009
122	-0.0109406	-0.0265427	-0.0845337	-0.0739746	-0.0471725	-0.0384445	-0.0217361	-0.0125504
123	0.0284500	0.0826721	0.1126251	0.0532761	0.0114975	-0.0184174	-0.0278168	-0.0385132
124	-0.0009766	-0.0013504	0.0082321	0.0411072	0.0223083	0.0020828	-0.0291138	-0.0363312
125	0.0068207	0.0242920	0.0580826	0.0683975	0.0789490	0.0621414	0.0445786	0.0190887
126	0.0293198	0.0630722	0.0497131	-0.0120468	-0.0440521	-0.0440979	-0.0299225	-0.0291214
127	0.0271988	0.0838776	0.1353760	0.1022873	0.0741501	0.0458984	0.0275192	-0.0002823

APPENDIX 3: SECOND-STAGE LSP SHAPE CODEBOOK

Index	Element 1	Element 2	Element 3	Element 4	Element 5	Element 6	Element 7	Element 8
0	-0.00045776	0.00002289	0.00099182	0.00270081	0.00746155	0.00529480	-0.00106049	-0.00178528
1	-0.00029755	-0.00101471	-0.00086212	0.00087738	-0.00106049	0.00087738	-0.01163483	0.00027466
2	-0.00118256	-0.00199127	-0.00380707	-0.00403595	-0.00030518	0.00240326	0.00474548	0.00889587
3	-0.00337219	-0.00188446	0.00494385	-0.00525665	0.00161743	-0.00501251	0.00176239	0.00527191
4	0.00021362	0.00082397	-0.00271606	-0.00733185	0.00150299	0.00543213	0.00144958	-0.00333405
5	-0.00189972	-0.00291443	-0.00379944	-0.00436401	-0.00100708	-0.00173187	-0.00433350	-0.00628662
6	-0.00256348	-0.00302124	-0.00202942	0.00238037	-0.00732422	0.00712585	0.00381470	0.00191498
7	-0.00487518	-0.00572205	0.00082397	0.00188446	-0.00714111	-0.00489044	-0.00206757	-0.00262451
8	-0.00010681	-0.00145721	-0.00476074	0.00331116	0.01039886	-0.00167847	0.00269318	0.00548553
9	-0.00064850	-0.00139618	-0.00781250	-0.00374603	0.00415039	-0.00397491	-0.00685883	0.00500488
10	-0.00432587	-0.00485229	-0.00765228	0.00385284	0.00276184	-0.00205231	0.00595856	-0.00077057
11	-0.01011658	0.00389099	-0.00176239	0.00331116	0.00072479	-0.00367737	-0.00161743	-0.00171661
12	-0.00463867	-0.00547791	-0.00512695	-0.00231171	0.00556183	0.00563049	-0.00107574	0.00090027
13	-0.00424957	-0.00586700	-0.00111389	0.00374603	0.00692749	-0.00564575	-0.00616455	-0.00193024
14	-0.00263214	-0.00723267	-0.00938416	0.00535583	-0.00386810	0.00198364	-0.00283813	0.00453949
15	0.00451660	-0.01326752	-0.00163269	-0.00040436	0.00058746	-0.00355530	-0.00116730	-0.00093842
16	0.00000763	0.00039673	0.00560760	-0.00251770	0.00186157	0.01089478	-0.00137329	0.00457001
17	-0.00207520	-0.00434113	0.00380707	0.00775909	-0.00274658	0.00917053	-0.00515747	-0.00215149
18	-0.00079346	-0.00209808	0.00601196	-0.00038147	-0.00785828	0.00248718	-0.00209808	0.01184082
19	-0.00298309	-0.00412750	0.01414490	0.00212097	-0.00019073	-0.00061798	-0.00274658	0.00065613
20	0.00035858	0.00170898	0.00259399	-0.01609802	-0.00000763	0.00162506	-0.00133514	0.00100708
21	-0.00503540	-0.00462341	0.00434875	-0.00257874	0.00315094	0.00467682	0.00090790	-0.01029968
22	-0.00057220	-0.00221252	-0.00485992	-0.00782013	-0.00888824	0.00464630	-0.00598145	0.00544739
23	-0.00719452	-0.01031494	0.00247192	-0.00617218	-0.00236511	0.00193024	-0.00309753	0.00254059
24	-0.00290680	-0.00582886	0.00511169	0.00539398	0.00604248	0.00421143	0.00669861	0.00392151
25	-0.00224304	-0.00450134	0.00026703	0.00598907	0.00528717	0.00506592	-0.00660706	0.01150513
26	-0.00428772	-0.00616455	0.00080872	0.00531769	-0.00407410	-0.00506592	0.00903320	0.00828552
27	-0.01528931	-0.00518799	0.00449371	0.00463104	0.00074005	0.00062561	0.00030518	0.00215912
28	-0.00177765	-0.00195313	-0.00070953	-0.01091003	0.01485443	0.00244141	0.00580597	-0.00041199
29	-0.00414276	-0.00515747	-0.00126648	0.00601959	0.01933289	0.00696564	-0.00712585	-0.00523376
30	-0.00677490	-0.01074219	-0.01129150	-0.00234985	-0.00765991	-0.00885773	0.00039673	0.00651550
31	-0.00891113	-0.01766968	0.00561523	0.00578308	0.00490570	0.00194550	-0.00356293	-0.00082397
32	-0.00096130	-0.00098419	-0.00413513	0.00173950	0.00495911	0.01183319	0.00845337	-0.00122833
33	-0.00326538	-0.00333405	0.00357056	-0.00922394	0.00468445	0.00469971	-0.01351166	-0.00124359
34	-0.00380707	-0.00453949	-0.00596619	-0.00867462	-0.00395966	-0.00065613	0.00699615	-0.00128174
35	-0.00752258	-0.00469208	0.00773621	-0.00899506	0.00354004	-0.01000214	0.00416565	-0.00489044
36	0.00000763	-0.00075531	-0.00958252	-0.00546265	-0.00563049	0.01355743	0.00136566	-0.01033783
37	-0.00630951	-0.00655365	-0.00659943	0.00173950	-0.00452423	0.00329590	-0.00886536	-0.01154327
38	-0.00095367	-0.00313568	-0.00857544	0.00354767	-0.01430511	0.00208282	0.00795746	-0.00600433
39	-0.00719452	-0.01290894	0.00675201	0.00386047	-0.01235962	-0.00559235	0.00556183	-0.00572968
40	-0.00801086	-0.00989532	-0.00411224	-0.00192261	0.01290894	-0.00526428	0.00534058	0.01023865
41	-0.00186920	-0.00344086	-0.00266266	-0.01030731	0.00197601	-0.00857544	-0.01191711	0.01605988
42	-0.00836945	-0.00853729	-0.00543976	0.00074005	0.00202942	-0.00057983	0.01557922	-0.00852203
43	-0.01739502	0.00542450	-0.00310516	-0.00865936	-0.00068665	0.00004578	0.00126648	0.00104523
44	-0.00631714	-0.01346588	-0.01500702	-0.00602722	0.00646210	0.01339722	-0.00647736	-0.00049591
45	-0.00720215	-0.01295471	-0.00642395	-0.00480652	0.01066589	-0.01364136	-0.01339722	-0.00752258
46	-0.00552368	-0.01621246	-0.01331329	0.01739502	-0.00738525	0.00836182	0.00555420	0.00202942
47	0.00701904	-0.02751923	0.00681305	0.00177765	-0.00151062	-0.00057220	0.00139618	0.00026703
48	0.00125885	0.00099945	0.00302887	-0.00566864	-0.00022125	0.02838135	-0.00177765	0.00128937
49	-0.00152588	-0.00276184	0.01119232	0.00526428	-0.00367737	0.01557159	-0.01921844	-0.00600433
50	-0.00518036	-0.00488281	0.00891876	-0.00863647	-0.00762939	0.00898743	0.00974274	0.00371552
51	-0.00656891	-0.00149536	0.02567291	-0.01044464	-0.00462341	-0.00055695	-0.00331879	-0.00172424
52	0.00264740	0.00337219	-0.00964355	-0.03454590	0.00086975	0.01454163	0.00080109	0.00027466
53	-0.00373077	-0.00283813	0.00484467	-0.00320435	0.00572968	0.00675201	-0.00530243	-0.03117371
54	-0.00197601	-0.01000214	-0.00423431	-0.00650787	-0.02683258	0.01259613	-0.00359344	0.00352478
55	-0.01738739	-0.02180481	0.00868988	-0.01496887	-0.00506592	-0.00038147	-0.00035858	0.00020599
56	-0.00734711	-0.00748444	0.01610565	0.00942993	0.01790619	0.01776123	0.01367950	0.01322937
57	-0.00354767	-0.00787354	-0.00128937	0.00877380	-0.00132751	0.00061798	-0.02446747	0.01464081
58	-0.00668335	-0.00952911	0.00102997	0.00148010	-0.00872040	0.00305176	0.03495026	0.01872253
59	-0.03459930	0.00503540	0.00631714	0.00146484	0.00110626	-0.00109100	-0.00064850	0.00020599
60	-0.00075531	-0.00411224	-0.02098846	-0.01145172	0.03794098	0.02877808	0.01161194	0.00054932
61	-0.00887299	-0.02392578	-0.01149750	0.03517151	0.02966309	0.00566864	-0.02129364	-0.02059937
62	-0.01180267	-0.03713226	-0.03850555	-0.00773621	-0.01717377	-0.01065826	-0.00489044	-0.00129700
63	-0.02636719	-0.04943848	0.01272583	0.01393127	0.00457001	0.00045776	0.00072479	0.00040436

APPENDIX 4: PITCH PREDICTOR TAB CODEBOOK

Index	Element 1	Element 2	Element 3
0	-0.0112610	-0.1767275	0.0010985
1	-0.2130735	0.4948120	0.0315245
2	0.1685180	-0.5930480	0.2047120
3	0.1311035	0.2366640	0.1258850
4	0.2671205	0.3766175	0.0750425
5	0.2727660	0.8498230	-0.1647035
6	0.1418760	0.2827150	-0.0781860
7	0.4192810	0.5174865	0.0445555
8	-0.0790405	0.2878420	-0.0986025
9	0.0102235	0.5396730	-0.2112120
10	0.1182555	-0.3234560	-0.1257935
11	0.0584410	0.4372560	0.0544435
12	0.2677000	0.6710510	-0.0020140
13	0.0510560	0.9555970	-0.0752870
14	0.2883300	0.4584655	-0.1362610
15	0.4607545	0.6708375	-0.1777040
16	0.0267945	0.1656190	0.0300600
17	-0.1670530	0.6494750	0.4698790
18	0.0991210	-0.2931520	0.1622010
19	0.0505675	0.3623655	0.2523805
20	0.0785520	0.4996340	0.4010620
21	-0.0176085	0.7084045	0.2586975
22	0.2232055	0.3234255	0.2569580
23	0.2735290	0.4712525	0.2355345
24	-0.0867615	0.2794190	0.1318055
25	-0.2120360	0.8054810	0.3201295
26	-0.1777955	-0.3514100	0.0110170
27	-0.1222230	0.4702455	0.2991335
28	0.1735230	0.5760195	0.1928100
29	-0.0948180	0.9347840	0.1232910
30	0.3175050	0.4074400	0.2554320
31	0.0957640	0.7896425	0.0609740

APPENDIX 5: GAIN CODEBOOK

Index	Element
0	-5.38477
1	-3.68066
2	-2.76855
3	-2.09717
4	-1.47217
5	-0.33984
6	0.67285
7	1.82031
8	-0.88525
9	0.16748
10	1.20313
11	2.62549
12	3.80518
13	5.64551
14	8.70605
15	11.85156

APPENDIX 6: GAIN CHANGE THRESHOLD MATRIX

			Log-gain change of previous frame, $[dB_2]$											
			-8 to -6 j=1	-6 to -4 j=2	-4 to -2 j=3	-2 to 0 j=4	0 to 2 j=5	2 to 4 j=6	4 to 6 j=7	6 to 8 j=8	8 to 10 j=9	10 to 12 j=10	12 to 14 j=11	14 to 16 j=12
Relative log-gain of previous frame, $[dB_2]$	-24 to –22	i=1	0.00000	0.79102	0.55664	14.26563	14.08398	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	-22 to –20	i=2	0.00000	13.85156	1.73047	13.76758	13.92773	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	-20 to –18	i=3	-1.96094	8.91211	7.83594	14.09961	13.77930	0.91016	-2.41406	0.00000	0.00000	0.00000	0.00000	0.00000
	-18 to –16	i=4	-1.96094	8.66992	13.53125	14.09570	13.95117	12.97461	2.14648	0.00000	0.00000	0.00000	0.00000	0.00000
	-16 to –14	i=5	-1.47266	9.29297	13.92578	13.89063	13.87891	13.93750	12.20703	-4.99023	0.00000	0.00000	0.00000	0.00000
	-14 to –12	i=6	4.60547	12.33398	14.09180	14.14258	14.16016	13.48633	12.39063	2.01172	0.00000	0.00000	0.00000	0.00000
	-12 to –10	i=7	10.66016	10.72656	13.83203	13.68359	13.93945	13.77930	13.09570	10.17578	-0.15430	-2.92578	0.00000	0.00000
	-10 to –8	i=8	6.59375	10.19531	13.34375	12.87305	13.36719	13.36328	13.12891	12.66797	0.72852	0.30078	4.87109	7.85742
	-8 to –6	i=9	2.64063	9.52539	9.85547	10.35938	10.63086	12.92383	12.70508	12.65234	8.96680	1.32422	4.86719	7.81445
	-6 to –4	i=10	6.24805	8.26758	8.78125	9.08594	9.03125	10.34180	11.21875	11.07227	8.32617	8.41992	7.70313	7.86133
	-4 to –2	i=11	6.18945	6.71875	7.98438	7.37109	7.50391	7.69922	9.09180	8.73633	6.91211	7.68750	7.22266	3.50977
	-2 to 0	i=12	4.40430	5.46484	6.17773	6.04492	6.14063	6.84766	5.89063	5.43750	4.67188	5.58008	7.70898	7.46094
	0 to 2	i=13	3.39648	5.41602	5.40039	4.77734	4.59375	4.63477	6.43359	3.54102	4.37891	3.70117	6.64844	4.74414
	2 to 4	i=14	0.00000	3.50000	4.60352	3.92188	3.68164	4.21680	4.18750	3.32617	3.38867	2.32813	5.15039	1.76563
	4 to 6	i=15	0.00000	1.10156	3.04492	3.18945	2.60156	2.43164	2.91016	1.48438	0.43555	0.44336	1.50391	1.75391
	6 to 8	i=16	0.00000	-0.11914	-1.13672	1.41602	1.49609	0.72852	0.60352	-0.35352	-0.98242	-1.15039	-1.99414	0.00000
	8 to 10	i=17	0.00000	0.00000	0.00000	1.36861	1.18557	-0.36990	-4.01682	-2.21214	0.00000	-1.33077	-3.04360	0.00000
	10 to 12	i=18	0.00000	0.00000	0.00000	0.52843	0.43190	0.00000	0.00000	-2.86324	0.00000	0.00000	0.00000	0.00000

APPENDIX 7: EXCITATION VQ SHAPE CODEBOOK

Index	Element 1	Element 2	Element 3	Element 4
0	-0.514526	0.847412	0.166748	0.120605
1	0.389648	1.125000	-1.070557	0.048584
2	-0.263916	-0.053101	0.189209	0.177734
3	2.927368	-0.262695	-0.092896	0.274292
4	-0.348755	-0.356812	-0.765747	-0.639038
5	1.912231	0.890869	-2.045654	-0.802124
6	-0.180298	-1.221802	-1.728760	-0.965210
7	1.743286	-1.338379	0.184204	-0.281128
8	-1.407593	1.109497	1.724487	-0.347900
9	2.324219	1.637939	0.742188	0.526001
10	-0.330933	-0.405396	0.890747	1.477661
11	1.545532	-0.195068	0.148560	0.073486
12	-0.583740	0.456055	0.253296	-1.269043
13	0.587769	-0.129028	0.616699	-0.256714
14	-1.211426	-0.743896	-0.608887	-0.219360
15	0.196289	-1.870728	-0.309326	1.111694