ARIB STD-T64-C.S0030-0 v3.0

Selectable Mode Vocoder (SMV) Service Option for Wideband Spread Spectrum Communication Systems

Original Specification

This standard, ARIB-T64-C.S0030-0 v3.0, was prepared by 3GPP2-WG of Association of Radio Industries and Businesses (ARIB) based upon the 3GPP2 specification, C.S0030-0 v3.0.

Modification to the original specification

None.

Notes

None.

3GPP2 C.S0030-0

Version 3.0

Date: January 2004

3RD GENERATION PARTNERSHIP PROJECT 2 "3GPP2"

Selectable Mode Vocoder (SMV) Service Option for Wideband Spread Spectrum Communication Systems

Intentionally left blank.

FOREWORD

These technical requirements form a standard for Service Option 56, a variable-rate two-way speech service option. The maximum speech-coding rate of the service option is 8.55 kbps.

This standard does not address the quality or reliability of Service Option 56, nor does it cover equipment performance or measurement procedures.

NOTES

1. Accompanying "Recommended Minimum Performance Standard for the Selectable Mode Vocoder, Service Option 56," provides specifications and measurement methods.

2. "Base station" refers to the functions performed on the land-line side, which are typically distributed among a cell, a sector of a cell, a mobile switching center, and a personal communications switching center.

3. This document uses the following verbal forms: "Shall" and "shall not" identify requirements to be followed strictly to conform to the standard and from which no deviation is permitted. "Should" and "should not" indicate that one of several possibilities is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited. "May" and "need not" indicate a course of action permissible within the limits of the standard. "Can" and "cannot" are used for statements of possibility and capability, whether material, physical, or causal.

4. Footnotes appear at various points in this specification to elaborate and further clarify items discussed in the body of the specification.

5. Unless indicated otherwise, this document presents numbers in decimal form.

   Binary numbers are distinguished in the text by the use of single quotation marks. In some tables, binary values may appear without single quotation marks if table notation clearly specifies that values are binary. The character 'x' is used to represent a binary bit of unspecified value. For example 'xxx00010' represents any 8-bit binary value such that the least significant five bits equal '00010'.

   Hexadecimal numbers (base 16) are distinguished in the text by use of the form 0xh…h where h…h represents a string of hexadecimal digits. For example, 0x2fa1 represents a number whose binary value is '10111110100001' and whose decimal value is 913.

NOTES

6. The following conventions apply to mathematical expressions in this standard:

- $\lfloor x \rfloor$ indicates the largest integer less than or equal to x: $\lfloor 1.1 \rfloor = 1, \lfloor 1.0 \rfloor = 1$.
- $\lceil x \rceil$ indicates the smallest integer greater than or equal to x: $\lceil 1.1 \rceil = 2, \lceil 2.0 \rceil = 2$.
- $|x|$ indicates the absolute value of x: $|-17|=17$, $|17|=17$.
- $\oplus$ indicates exclusive OR.
- min(x, y) indicates the minimum of x and y.
- max(x, y) indicates the maximum of x and y.
- In figures, $\otimes$ indicates multiplication. In formulas within the text, multiplication is implicit. For example, if h(n) and $p_L(n)$ are functions, then $h(n)\, p_L(n) = h(n) \otimes p_L(n)$.
- x mod y indicates the remainder after dividing x by y: $x \bmod y = x - (y \lfloor x/y \rfloor)$.
- round(x) is traditional rounding: $\text{round}(x) = \lfloor x + 0.5 \rfloor$.

$$\text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} .$$

- $\sum$ indicates summation. If the summation symbol specifies initial and terminal values, and the initial value is greater than the terminal value, then the value of the summation is 0. For example, if N=0, and if f(n) represents an arbitrary function, then

$$\sum_{n=1}^{N} f(n) = 0 .$$

- The bracket operator, [ ], isolates individual bits of a binary value. VAR[n] refers to bit n of the binary representation of the value of the variable VAR, such that VAR[0] is the least significant bit of VAR. The value of VAR[n] is either 0 or 1.

REFERENCES

The following standards contain provisions, through reference in this text constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. ANSI and TIA maintain registers of currently valid national standards published by them.

—American National Standards:

1. ANSI/EIA/TIA-579-A-98, *Telecommunications Telephone Terminal Equipment Transmission Requirements for Digital Wireline Telephones*

—Other Standards:

2. ITU-T Recommendation G.711, *Pulse Code Modulation (PCM) of Voice Frequencies*, Vol. III, Geneva 1972.

3. ITU-T Recommendation G.712, *Transmission Performance Characteristics of Pulse Code Modulation,* November 1996.

4. IEEE Standard 269-2002, *IEEE Standard Methods for Measuring Transmission Performance of Analog and Digital Telephone Sets,* 2002.

5. IEEE Standard 661-1979, *Method for Determining Objective Loudness Ratings of Telephone Connections,* 1979.

6. 3GPP2 C.S0003-A, *Medium Access Control (MAC) Standard for cdma2000 Spread Spectrum Systems,* March, 2000.

7. 3GPP2 C.S0005-A, *Upper Layer (Layer 3) Signalling Standard for cdma2000 Spread Spectrum Systems,* March, 2000.

8. 3GPP2 C.S0034-0 Ver 1.0, *Recommended Minimum Performance Standard for the Selectable Mode Vocoder, Service Option 56.*

9. 3GPP2 C.S0014-0 Ver 1.0, *Enhanced Variable Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Digital Systems*, September, 1996.

Table of Contents

1 SERVICE OPTION 56: SELECTABLE MODE VOCODER TWO-WAY VOICE COMMUNICATION

1.1 General Description

Service Option 56, the Selectable Mode Vocoder (SMV), provides two-way voice communication between the base station and the mobile station using the dynamically variable data rate speech codec algorithm described in this standard. The transmitting speech codec receives voice samples and generates an encoded speech packet for every Fundamental Channel Traffic Channel frame.† The receiving station generates a speech packet from every Fundamental Channel Traffic Channel frame and supplies it to the speech codec for decoding into voice samples.

SMV communicates at one of four rates: 9600 bps, 4800 bps, 2400 bps and 1200 bps.

1.2 Overview of SMV Documentation

The SMV specification family consists of two standards. This standard provides the algorithmic description of the SMV as well as the floating-point C-simulation of the codec. The companion minimum performance standard, *"Recommended Minimum Performance Standard for the Selectable Mode Vocoder, Service Option 56"*, consists of the master fixed-point C-simulation of the SMV as well as a minimum performance specification. The minimum performance specification consists of a set of objective and subjective tests used to verify the quality of any non bit-exact SMV implementation.

Section 2 of this standard provides information on the interfaces between SMV and IS-2000 systems. Section 3 provides information on the audio interfaces to SMV. The specifications given by Sections 4, 5, and 6 of this standard provide the algorithmic description of SMV at a high level. Section 7 provides a description of the in-band data capabilities of SMV. The floating point C-code accompanying this document provides a more detailed complementary description of SMV. In the case of a discrepancy between the floating-point C-simulation and the algorithmic description, the floating-point C-simulation will prevail.

1.3 SMV Algorithmic Options

The specifications defined in SMV Encoder Pre-Processing (Section 5.2), Voice Activity Detection (Sections 5.3.2 and 5.3.3), Music Detection (Section 5.3.4), Frame Erasure Concealment (Section 6.8), and Post-Processing (Section 6.9), are optional for non bit-exact implementations intended for varying operational environments, such as base-station, mobile station handsets and in-vehicle hands-free.

There are two algorithmic options for both the Noise Suppression and the Voice Activity Detection modules as described below.

---

† IS-2000-A uses the term "frame" to represent a 20 ms grouping of data on the Fundamental Channel. Common speech codec terminology also uses the term "frame" to represent a quantum of processing. For Service Option 56, the speech codec frame corresponds to speech sampled over 20 ms. The speech samples are processed into a packet. This packet is transmitted in a Traffic Channel frame.

1.3.1 Standard Options for Noise Suppression

This standard defines two options (A and B) for Noise Suppression as described in Section 5.2.3. These two options are selected in the SMV simulation code by using the flags NS_A, and NS_B. If the option NS_A is chosen, the total fixed algorithmic delay of the codec is 33 ms. If the option NS_B is chosen, the total fixed algorithmic delay at the codec is 30 ms.

1.3.2 Standard Options for Voice Activity Detection

This document defines two options (A and B) for Voice Activity Detection as described in Sections 5.3.2 and 5.3.3. These two options are selected in the SMV simulation code by using the flags VAD_A, and VAD_B respectively. VAD_A was used in the original SMV candidate in the selection phase. VAD_B is based on the Rate Determination algorithm in IS-127 (EVRC), requires less program memory, and reuses modules from earlier implementations of IS-127.

1.4 SMV Implementation Options

There are two options available for implementation of the SMV:

If a bit-exact implementation approach is chosen, the implementation that is developed shall be end-to-end bit-exact to the reference fixed-point C-simulation set to one of the four possible combinations of noise suppression and voice activity detection. Bit-exactness is verified via application of the associated SMV minimum performance standard [8], which includes the reference fixed-point codec simulation.

Alternatively, an implementation that deviates from the end-to-end bit-exact SMV specification described above may be developed. Such an implementation shall pass objective and subjective tests defined by the SMV minimum performance standard [8]. The master floating-point SMV in such a subjective test shall use Noise Suppression option B and Voice Activity Detection option B since the SMV standard codec Characterization Test was performed using Noise Suppression option B and Voice Activity Detection option B.

1.5 Service Option Number

The variable data rate two-way voice service option, using the speech codec algorithm described by this standard, shall use service option number 56 and shall be called Service Option 56.

1.6 Allowable Delays

1.6.1 Allowable Transmitting Speech Codec Encoding Delay

The transmitting speech codec shall supply a packet to the multiplex sublayer no later than 20 ms after it has obtained the last input sample for the current speech frame.

1.6.2 Allowable Receiving Speech Codec Decoding Delay

The receiving decoder shall generate the first sample of speech using parameters from a packet supplied to it by the multiplex sublayer no later than 3 ms after being supplied the packet.

1.7 Special Cases

1.7.1 Blanked Packets

A blanked frame occurs when the transmitting station uses the entire frame for either signaling traffic or secondary traffic. The SMV does no special encode processing during the generation of a blank packet; i.e., the generated voice packet is simply not used. The decoder, in turn, treats a blank packet in the same manner as a frame erasure.

1.7.2 Null Traffic Channel Data

A Rate 1/8 packet with all bits set to '1' is considered as null Traffic Channel data. This packet is declared an erased packet and handled as described in Section 6.8.

1.8 Terms and Numeric Information

Autocorrelation Function. A function showing the relationship of a signal with a time-shifted version of itself.

Base Station. A station in the Public Radio Telecommunications Service, other than a personal/mobile station, used for radio communications with personal/mobile stations.

Codec. The combination of an encoder and decoder in series (encoder/decoder).

Code Excited Linear Predictive Coding (CELP). A speech coding algorithm. CELP coders use codebook excitation, a long-term pitch prediction filter, and a short-term formant prediction filter.

Codebook. A set of vectors used by the speech codec. For each speech codec codebook subframe, one particular vector is chosen and is used to excite the speech codec's filters. The codebook vector is chosen to minimize the weighted error between the original and synthesized speech after the pitch and formant synthesis filter coefficients have been determined.

Coder. Same as "encoder."

Decoder. Generally, a device for the translation of a signal from a digital representation into an analog format. For this standard, a device that converts speech encoded in the format specified in this standard to analog or an equivalent PCM representation.

Encoder. Generally, a device for the translation of a signal into a digital representation. For this standard, a device that converts speech from an analog, or from its equivalent PCM representation, to the digital representation described in this standard.

Formant. A resonant frequency of the human vocal tract causing a peak in the short term spectrum of speech.

IIR Filter. (Infinite-duration impulse response filter) A filter for which the output, in response to an impulse input, never totally converges to zero. This term is usually used in reference to digital filters.

Linear Predictive Coding (LPC). A method of predicting future samples of a sequence by a linear combination of the previous samples of the same sequence. Linear Predictive Coding is frequently used in reference to a class of speech codecs.

Line Spectral Frequencies (LSFs). A representation of digital filter coefficients in a pseudo-frequency domain. This representation has good quantization and interpolation properties.

LSB. Least significant bit.

MSB. Most significant bit.

Normalized Autocorrelation Function. A measure used to determine the pitch period and the degree of periodicity of the input speech. This measure is useful in distinguishing voiced from unvoiced speech.

Packet. The unit of information exchanged between service option applications in the base station and the personal/mobile station.

Personal/Mobile Station. A station in the Public Radio Telecommunications Service intended to be used while in motion or during halts at unspecified points.

Pitch. The fundamental frequency in speech caused by the periodic vibration of the human vocal cords.

RDA. Rate Determination Algorithm.

Receive Objective Loudness Rating (ROLR). A measure of receive audio sensitivity. ROLR is a frequency-weighted ratio of the line voltage input signal to a reference encoder to the acoustic output of the receiver. IEEE 269 defines the measurement of sensitivity and IEEE 661 defines the calculation of objective loudness rating.

SPL. Sound Pressure Level.

Transmit Objective Loudness Rating (TOLR). A measure of transmit audio sensitivity. TOLR is a frequency-weighted ratio of the acoustic input signal at the transmitter to the line voltage output of the reference decoder. IEEE 269 defines the measurement of sensitivity and IEEE 661 defines the calculation of objective loudness rating.

Voiced Speech. Speech generated when the vocal cords are vibrating at a fundamental frequency. Characterized by high energy, periodicity, and a large ratio of energy below 2 kHz to energy above 2 kHz.

Unvoiced Speech. Speech generated by forcing air through constrictions in the vocal tract without vibration of the vocal cords. Characterized by a lack of periodicity, and a near-unity ratio of energy below 2 kHz to energy above 2 kHz.

WAEPL. Weighted Acoustic Echo Path Loss. A measure of the echo performance under normal conversation. ANSI/EIA/TIA-579 defines the measurement of WAEPL.

Zero Input Response (ZIR). The filter output caused by the non-zero initial state of the filter when no input is present.

Zero State Response (ZSR). The filter output caused by an input when the initial state of the filter is zero.

ZIR. See Zero Input Response.

ZSR. See Zero State Response.

2 REQUIRED MULTIPLEX OPTION SUPPORT

Service Option 56 shall support an interface with Multiplex Option 1. Speech packets for Service Option 56 shall only be transported as primary traffic.

2.1 Interface to Multiplex Option 1

2.1.1 Transmitted Packets

The speech codec shall generate and shall supply exactly one packet to the multiplex sublayer every 20 milliseconds. The packet contains the service option information bits that are transmitted as primary traffic. The packet shall be one of four types as shown in Table 2.1-1. The number of bits supplied to the multiplex sublayer for each type of packet shall also be as shown in Table 2.1-1. Unless otherwise commanded, the speech codec may supply a Rate 1, Rate 1/2, Rate 1/4, or Rate 1/8 packet. Upon command, the speech codec shall generate a Blank packet. Also upon command, the speech codec shall generate a non-blank packet with a maximum rate of Rate 1/2.

A Blank packet contains no bits and is used for blank-and-burst transmission of signaling traffic or secondary traffic (see IS-2000.3-A and IS-2000.5-A).

Table 2.1-1. Packet Types Supplied by Service Option 56 to the Multiplex Sublayer

Packet Type	Bits per Packet
Rate 1	171
Rate 1/2	80
Rate 1/4	40
Rate 1/8	16
Blank	0

2.1.2 Received Packets

The multiplex sublayer in the receiving station categorizes every received Traffic Channel frame, and supplies the packet type and accompanying bits, if any, to the speech codec as shown in Table 2.1-1~~Table 2.1-1~~. The speech codec processes the bits of the packet as described in Section 6. The received packet types shown in Table 2.1-2~~Table 2.1-2~~ correspond to the transmitted packet types shown in Table 2.1-1~~Table 2.1-1~~. The Blank packet type occurs when the receiving station determines that a blank-and-burst frame for signaling traffic or secondary traffic was transmitted. The Rate 1 with bit errors packet type occurs when the receiving station determines that the frame was transmitted at 9600 bps and the frame has one or more bit errors. The insufficient frame quality packet type occurs when the mobile station is unable to decide upon the data rate of the received frame or when the mobile station detects a frame in error that does not belong to the Rate 1 with bit errors packet type.

Table 2.1-2. Packet Types Supplied by the Multiplex Sublayer to Service Option 56

Packet Type	Bits per Packet
Rate 1	171
Rate 1/2	80
Rate 1/4	40
Rate 1/8	16
Blank	0
Rate 1 with bit errors	171
Insufficient frame quality (erasure)	0

2.2 Negotiation for Service Option 56

The mobile station and base station can negotiate for Service Option 56 service negotiation, as described in IS-2000.5-A.

2.2.1 Procedures Using Service Negotiation

The mobile station and base station shall perform service negotiation for Service Option 56 as described in IS-2000.5-A, and the negotiated service configuration shall include only valid attributes for the service option as specified in Table 2.2-1~~Table 2.2-1~~.

Table 2.2-1. Valid Service Configuration Attributes for Service Option 56

Service Configuration Attribute	Valid Selections
Forward Multiplex Option	Multiplex Option 1
Reverse Multiplex Option	Multiplex Option 1
Forward Transmission Rates	Rate Set 1 with all rates enabled
Reverse Transmission Rates	Rate Set 1 with all rates enabled
Forward Traffic Type	Primary Traffic
Reverse Traffic Type	Primary Traffic

2.2.1.1 Initialization and Connection

2.2.1.1.1 Mobile Station Requirements

If the mobile station accepts a service configuration, as specified in a *Service Connect Message*, *General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 56, the mobile station shall perform the following:

- If the service option connection is new (that is, not part of the previous service configuration), the mobile station shall perform speech codec initialization at the time specified by the maximum of the action time associated with the message carrying

the Service Configuration Record, and the time that the corresponding Call Control Instance is instantiated. The mobile station shall initialize its SMV encoder mode of operation to a default value of 0. The mobile station shall complete the initialization within 40 ms.

- Beginning at the time specified by the maximum of the action time associated with the message carrying the Service Configuration Record, and the time that the corresponding Call Control Instance is instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 56 shall process received packets and generate and supply packets for transmission as follows:
  - If the Call Control Instance is in the *Conversation Substate*, Service Option 56 shall process the received packets and generate and supply packets for transmission in accordance with this standard.
  - If the Call Control Instance is not in the *Conversation Substate*, Service Option 56 shall process the received packets in accordance with this standard, and shall generate and supply Rate 1/8 Packets with all bits set to '1' for transmission, except when commanded to generate a Blank packet.

2.2.1.1.2 Base Station Requirements

If the base station establishes a service configuration, as specified in a *Service Connect Message*, *General Handoff Direction Message*, or *Universal Handoff Direction Message* that includes a service option connection using Service Option 56, the base station shall perform the following:

- If the service option connection is new (that is, not part of the previous service configuration), the base station shall perform speech codec initialization no later than the time specified by the maximum of the action time associated with the message carrying the Service Configuration Record, and the time that the corresponding Call Control Instance is Instantiated. The base station shall initialize its SMV encoder mode of operation to a default value of 0.
- Commencing at the time specified by the maximum of the action time associated with the message carrying the Service Configuration Record, and the time that the corresponding Call Control Instance is Instantiated, and continuing for as long as the service configuration includes the service option connection, Service Option 56 shall process received packets, and shall generate and supply packets for transmission in accordance with this standard. The base station may defer enabling the audio input and output.

2.2.1.2 Service Option Control Messages

2.2.1.2.1 Mobile Station Requirements

The mobile station shall support one pending *Service Option Control Message* for Service Option 56.

If the mobile station receives a *Service Option Control Message* for Service Option 56, then, at the action time associated with the message, the mobile station shall process the message as follows:

1. If the MOBILE_TO_MOBILE field is equal to '1', the mobile station should disable the audio output of the speech codec for 1 second after initialization.

If the MOBILE_TO_MOBILE field is equal to '0', the mobile station shall process each received packet as described in Section 6.

2. If the INIT_CODEC field is equal to '1', the mobile station shall perform speech codec initialization. The mobile station shall complete the initialization within 40 ms.

3. SMV accepts as input, a mode of operation through the RATE_REDUC field as defined in Table 2.2-2. Using this mode of operation, SMV generates Rate 1, Rate 1/2, and Rate 1/4 packets in a proportion that results in the average data rate given by Table 2.2-2.

Service Option 56 shall continue to use these fractions until either of the following events occur:

- The mobile station receives a *Service Option Control Message* specifying a different RATE_REDUC, or
- Service Option 56 is initialized.

Service Option 56 was developed using reduced rate operation (encoding mode of operation) as a network control criteria. Unlike the rate reduction mechanism in EVRC, Service Option 56 does not use a deterministic pattern of Rate 1 and Rate 1/2 to reduce the average encoding rate. Instead, the SMV codec selects the encoding rate based upon the characteristics of the input speech: Voiced, unvoiced, transient, stationary, as well as the encoding mode selected.

Table 2.2-2. SMV Encoding Rate Mode Control Parameters

RATE_REDUC (binary)	SMV Encoding Mode of Operation	Estimated Average Encoding Rate for Active Speech (Source Encoding Rates)
'000'	0	7.95 kbps
'001'	1	5.82 kbps
'010'	2	4.50 kbps
'011'	3	3.95 kbps
'100'	4 (1/2 rate Maximum applied to Mode 0)	4.0 kbps
'101'	5 (1/2 rate Maximum applied to Mode 1)	3.67 kbps
All other RATE_REDUC values are reserved.		

If the RATE_REDUC field is not equal to a value defined in Table 2.2-2~~Table 2.2-2~~, the mobile station shall reject the message by sending a *Mobile Station Reject Order* with the ORDQ field set equal to '00000100'.

2.2.1.2.2 Base Station Requirements

The base station may send a *Service Option Control Message* to the mobile station. If the base station sends a *Service Option Control Message*, the base station shall include the following type-specific fields for Service Option 56:

Table 2.2-3. Service Option Control Message Type-Specific Fields

Field	Length (bits)
RATE_REDUC	3
RESERVED	3
MOBILE_TO_MOBILE	1
INIT_CODEC	1

RATE_REDUC - SMV mode of operation.

The base station shall set this field to the RATE_REDUC value from Table 2.2-4~~Table 2.2-4~~ corresponding to the mode of operation that the mobile station is to operate in.

RESERVED - Reserved bits.

The base station shall set this field to '000'.

MOBILE_TO_MOBILE - Mobile-to-mobile processing.

If the mobile station is to perform mobile-to-mobile processing (Section 2.2.1.2.1), the base station shall set this field to '1'. In addition, if the mobile station is to disable the audio output of the speech codec for 1 second after initialization, the base station shall set the INIT_CODEC field and the MOBILE_TO_-MOBILE field to '1'. If the mobile station is not to perform mobile-to-mobile processing, the base station shall set the MOBILE_TO_MOBILE field to '0'.

INIT_CODEC - Initialize speech codec.

If the mobile station is to initialize the speech codec, the base station shall set this field to '1'; otherwise, the base station shall set this field to '0'.

Table 2.2-4. SMV Encoding Rate Mode Control Parameters

RATE_REDUC (binary)	SMV Encoding Mode of Operation	Estimated Average Encoding Rate for Active Speech (Source encoding rates)
'000'	0	7.95 kbps
'001'	1	5.82 kbps
'010'	2	4.50 kbps
'011'	3	3.95 kbps
'100'	4 (1/2 rate Maximum applied to Mode 0)	4.0 kbps
'101'	5 (1/2 rate Maximum applied to Mode 1)	3.67 kbps
All other RATE_REDUC values are reserved.		

3 AUDIO INTERFACES

3.1 Input Audio Interface

3.1.1 Input Audio Interface in the Mobile Station

The input audio may be either an analog or digital signal.

3.1.1.1 Conversion and Scaling

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

The quantities in this standard assume a 16-bit integer input normalization with a range from -32,768 through +32,767. The following speech codec discussion assumes this 16-bit integer normalization. If an input audio interface uses a different normalization scheme, then appropriate scaling should be used.

3.1.1.2 Digital Audio Input

If the input audio is an 8-bit μ-Law PCM signal, it shall be converted to a uniform PCM format according to Table 2 in ITU-T Recommendation G.711 “Pulse Code Modulation (PCM) of Voice Frequencies.

3.1.1.3 Analog Audio Input

If the input is in analog form, the mobile station shall sample the analog speech and shall convert the samples to a digital format for speech codec processing. This shall be accomplished by either the following or an equivalent method. First, the input gain audio level is adjusted. Then, the signal is bandpass filtered to prevent aliasing. Finally, the filtered signal is sampled and quantized (Section 3.1.1.1).

3.1.1.3.1 Transmit Level Adjustment

The mobile station shall have a transmit objective loudness rating (TOLR) equal to -46 dB, when transmitting to a reference base station. The loudness ratings are described in IEEE Standard 661-1979 “IEEE Standard Method for Determining Objective Loudness Ratings of Telephone Connections.”

When a 97dBspl ± 8dB C-message weighted white noise acoustic signal from an artificial mouth is applied to the microphone of the subscriber unit, as specified by "IEEE Standard Method for Measuring Transmission Performance of Telephone Sets," IEEE standard 269-1983, the input samples shall be approximately three bits less than the maximum PCM value. With PCM samples generated from μ-Law /uniform code, resulting in samples with a maximum of 14 bits, only 11 bits of precision should be generated.

3.1.1.3.2 Band Pass Filtering

The analog-to-digital conversion shall conform to the mask shown in ITU-T G.712 (11/96) Figure 4/G.712, “Attenuation/frequency distortion for 4-wire analogue-to-digital channels. (E4in to Tout)”. Additional anti-aliasing filtering may be provided by the manufacturer.

3.1.1.3.3 Echo Return Loss

Provision shall be made to ensure adequate isolation between receive and transmit audio paths in all modes of operation, such that when no external transmit audio is present, the speech codec does not generate packets at rates higher than Rate 1/8 due to acoustic coupling of the receive audio into the transmit audio path (specifically with the receive audio at full volume). Target levels of 45 dB WAEPL should be met. See ANSI/EIA/TIA Standard 579 "Acoustic-to-Digital and Digital-to-Acoustic Transmission Requirements for ISDN Terminals."

3.1.2 Input Audio Interface in the Base Station

3.1.2.1 Sampling and Format Conversion

The base station converts the input speech (analog, μ-Law companded Pulse Code Modulation, or other format) into a uniform quantized PCM format with at least 13 bits of dynamic range. The sampling rate is 8000 samples per second. The sampling and conversion process shall be as in Section 3.1.1.1.

3.1.2.2 Transmit Level Adjust

The base station shall set the transmit level so that a 1004 Hz tone at a level of 0 dBm0 at the network interface produces a level 3.17 dB below maximum amplitude at the output of the quantizer. Measurement techniques and tolerances are described in *"Recommended Minimum Performance Standard for the Selectable Mode Vocoder, Service Option 56"*.

3.1.2.3 Line Echo Canceling

The base station shall provide a method to cancel echoes returned by the PSTN interface.† The echo canceling function shall meet ITU-T G.168 requirement. The echo canceling function should work over a range of PSTN echo return delays, from 0 to 48 ms; however, the latter requirement is subject to local PSTN configuration demands, e,g., a 64 ms (or greater) echo canceling capability may be required in cases where the PSTN does not provide echo cancellation for long distance service.

3.2 Output Audio Interface

3.2.1 Output Audio Interface in the Mobile Station

The PCM samples shall be converted to μ-Law samples according to ITU-T Recommendation G.711. This μ-Law signal shall be converted to analog as specified by ITU-T Red Book G.711.

3.2.2 Band Pass Filtering

The digital-to-analog conversion shall conform to the mask shown in ITU-T G.712 (11/96) Figure 4/G.712, "Attenuation/frequency distortion for 4-wire analogue-to-digital channels. (E4in to Tout)". Additional reconstruction filtering may be provided by the manufacturer.

3.2.2.1 Receive Level Adjustment

The mobile station shall have a nominal receive objective loudness rating (ROLR) equal to 51 dB when receiving from a reference base station. The loudness ratings are described in

† Because of the relatively long delays inherent in the speech coding and transmitting processes, echoes that are not sufficiently suppressed are noticeable to the mobile station user.

IEEE Standard 661-1979 “IEEE Standard Method for Determining Objective Loudness Ratings of Telephone Connections.”

When a 97dBspl ± 8dB C-message weighted white noise acoustic signal from an artificial mouth is applied to the microphone of the subscriber unit as specified by "IEEE Standard Method for Measuring Transmission Performance of Telephone Sets," IEEE standard 269-1983, the input samples shall be of approximately three bits less than the maximum PCM value. With PCM samples generated from μ-Law /uniform code resulting in samples with a maximum of 14 bits, only 11 bits of precision should be generated.

3.2.3 Output Audio Interface in the Base Station

Details of the digital and analog interfaces to the network are outside the scope of this document.

3.2.3.1 Receive Level Adjustment

The base station shall set the audio level so that a received 1004 Hz tone 3.17 dB below maximum amplitude produces a level of 0 dBm0 at the network interface. Measurement techniques and tolerances are described in *“Recommended Minimum Performance Standard for the Selectable Mode Vocoder, Service Option 56”*.

No Text

4 THE SELECTABLE MODE VOCODER (SMV) SPEECH CODING – INTRODUCTION AND TABLES

4.1 Introduction to the SMV Speech Coding Algorithm

The SMV speech-coding algorithm is based on four codecs (encoder/decoder) operating at the rates of 8.55 kbps, 4.0 kbps, 2.0 kbps and 0.8 kbps. These codecs are called Rate 1 (full-rate), Rate 1/2 (half-rate), Rate 1/4 (quarter-rate) and Rate 1/8 (eighth-rate), respectively.

The SMV algorithm has 4 network-controlled operating modes: Mode 0 (premium mode), Mode 1 (standard mode), Mode 2 (economy mode), and Mode 3 (capacity-saving mode). The different modes allow a tradeoff between average data rate (ADR) and speech quality. In addition, Mode 0 and Mode 1 can be configured to operate in a half-rate max (HRM) mode, where the maximum rate allowed is the Rate 1/2.

	Mode 0	Mode 1	Mode 2	Mode 3
Rate 1	68.90%	38.14%	15.43%	7.49%
Rate 1/2	6.03%	15.82%	38.34%	46.28%
Rate 1/4	0.00%	17.37%	16.38%	16.38%
Rate 1/8	25.07%	28.67%	29.85%	29.85%
ADR	7205 bps	5182 bps	4073 bps	3692 bps

Figure 4.1-1: Rate Percentages and Average Data Rates for Clean Speech at Nominal Level (-22 dBov)

	Mode 0	Mode 1	Mode 2	Mode 3
Rate 1	55.13%	29.13%	16.60%	6.86%
Rate 1/2	10.65%	19.49%	32.39%	42.13%
Rate 1/4	0.00%	13.17%	14.93%	14.93%
Rate 1/8	34.21%	38.20%	36.08%	36.08%
ADR	6214 bps	4507 bps	3940 bps	3472 bps

Figure 4.1-2: Rate Percentages and Average Data Rates for Speech in the Presence of –15 dB Street Noise

Figure 4.1-1 gives the percentages of the selected rates and the ADR for a clean speech database at nominal level (-22 dBov) that was used for speech quality tests during the SMV selection. The voice activity for this database is about 70%; hence, that database does not represent a typical conversational speech traffic.

Figure 4.1-2 gives the percentages of the selected rates and the ADR for speech that was generated from the same database, with the addition of street noise at the level of –15 signal-to-noise level.

A 20 ms frame of signal sampled at the rate of 8000 Hz (160 samples) is processed by one of the four codecs, according to a rate-determination algorithm (RDA). The rate selection is based upon the frame characteristics (voiced speech, unvoiced speech, background noise, stationarity, etc.), and is controlled by the SMV mode. For each frame, the best rate for the frame is selected under the ADR constraint dictated by the SMV operating mode.

The four codecs are based upon the excitation-filter approach, where a Linear-Prediction (LPC) filter is excited by an excitation source. The LPC filter parameters and the excitation parameters are transmitted from the encoder to the decoder.

The Rate 1 codec and the Rate 1/2 codec are based on the code-excited linear prediction (CELP) approach, where the excitation of the LPC filter is represented by entries in a fixed codebook and in an adaptive codebook, multiplied by a fixed-codebook gain and an adaptive-codebook gain, respectively. The SMV system also selects a frame type for each Rate 1 or Rate 1/2 frame. The type can be 0 or 1. Type-1 frames are frames of stationary voiced speech, while frames of type 0 are all other types of speech. The bit allocation for the excitation parameters and the LPC filter parameters differs between type-1 frames and type-0 frames. For Rate 1 and Rate 1/2, the number of subframes in each frame depends on the rate and on the frame type. Rate 1 frames are divided into 4 subframes of 40 samples, or 5 ms each. Rate 1/2 type-0 frames are divided into 2 subframes of 80 samples or 10 ms each. Rate 1/2 type-1 frames are divided into 3 subframes of 53, 53, and 54 samples, or 6.625, 6.625, and 6.750 ms, respectively.

The excitation of the LPC filter for Rate 1/4 is generated by a random number generator. The frame is divided 10 subframes of 2 ms, and each subframe is multiplied by a gain factor. The LPC excitation is then filtered by a selected frequency-shaping filter.

The excitation of the LPC filter for Rate 1/8 is generated by a random number generator, and multiplied by a single gain for the 20 ms frame.

4.2 Bit Allocation Tables

The bit allocation for Rate 1 is given in Table 4.2-1, the bit allocation for Rate 1/2 is given in Table 4.2-2, the bit allocation for Rate 1/4 is given in Table 4.2-3, and the bit allocation for Rate 1/8 is given in Table 4.2-4.

Parameter	Bits per 20 ms frame			
	Type 0 (4 subframes)		Type 1 (4 subframes)	
LSFs	Interpolation	2 bit		
	Predictor switch	1 bit	Predictor switch	1 bit
	1st stage	7 bits	1st stage	7 bits
	2nd stage	7 bits	2nd stage	7 bits
	3rd stage	6 bits	3rd stage	6 bits
	4th stage	4 bits	4th stage	4 bits
		27 bits		25 bits
Type	1 bit			
Adaptive codebook	8,5,8,5 bits/subframe	26 bits	8 bits/frame	8 bits
Fixed codebook	5-pulse codebook	21 bits/subframe	8-pulse codebook	30 bits/subframe
	5-pulse codebook	20 bits/subframe		
	5-pulse codebook	20 bits/subframe		
		22 bits/subframe		
	22 bits/subframe	88 bits	30 bits/subframe	120 bits
Adaptive-codebook gain	2D VQ/subframe	7 bits/subframe	4D preVQ/frame	6 bits
Fixed-codebook gain		28 bits	4D delayed VQ/frame	10 bits
Rate sanity flag	1 bit		1 bit	
TOTAL	171 bits		171 bits	

Table 4.2-1: Bit Allocation Table for the SMV Rate 1 Codec

Parameter	Bits per 20 ms frame			
	Type 0 (2 subframes)		Type 1 (3 subframes)	
LSFs		Predictor switch	1 bit	
		1st stage	7 bits	
		2nd stage	7 bits	
		3rd stage	6 bits	
			21 bits	
Type	1 bit			
Adaptive codebook	7 bits/subframe	14 bits	7 bits/frame	7 bits
Fixed codebook	2-pulse codebook	14 bits/subframe	2-pulse codebook	12 bits/subframe
	3-pulse codebook	13 bits/subframe	3-pulse codebook	11 bits/subframe
	Gaussian codebook	13 bits/subframe	5-pulse codebook	11 bits/subframe
		15 bits/subframe		13 bits/subframe
	15 bits/subframe	30 bits	13 bits/subframe	39 bits
Adaptive-codebook gain	2D VQ/subframe	7 bits/subframe	3D preVQ/frame	4 bits
Fixed codebook gain		14 bits	3D delayed VQ/frame	8 bits
TOTAL	80 bits		80 bits	

Table 4.2-2: Bit Allocation Table for the SMV Rate 1/2 Codec

Note: since the entries of 2-pulse codebook for Rate 1/2 type-0 frames occupy only 12800 entries in the 2^{14}=16384 possible entry locations, 3472 additional entries of the Gaussian codebook occupy the higher entry locations of the 2-pulse codebook.

Parameter	Bits per 20 ms frame	
LSFs	1st stage	7 bits
	2nd stage	7 bits
	3rd stage	6 bits
		20 bits
Gains		17 bits
Shape		2 bits
Rate sanity flag		1 bit
TOTAL	40 bits	

Table 4.2-3: Bit Allocation Table for the SMV Rate 1/4 codec

Parameter	Bits per 20 ms frame	
LSFs	1st stage	4 bits
	2nd stage	4 bits
	3rd stage	3 bits
		11 bits
Energy	5 bits/subframe	5 bits
TOTAL	16 bits	

Table 4.2-4: Bit Allocation Table for the SMV Rate 1/8 Codec

4.3 SMV Bit Streams

After encoding, the encoded speech packets shall be formatted as described in Table 4.3-1 through Table 4.3-7. For each parameter, bit index 0 corresponds to the most significant bit. The packet buffer bit index 1 indicates the bit position corresponding to the beginning of the packet buffer.

Bit Index	Rate 1 Frame Type 0 Packet Bits	Rate 1 Frame Type 1 Packet Bits
1	SVS_DECI=0	SVS_DECI=1
2	LSF_STAGE1[0]	LSF_STAGE1[0]
3	LSF_STAGE1[1]	LSF_STAGE1[1]
4	LSF_STAGE1[2]	LSF_STAGE1[2]
5	LSF_STAGE1[3]	LSF_STAGE1[3]
6	LSF_STAGE1[4]	LSF_STAGE1[4]
7	LSF_STAGE1[5]	LSF_STAGE1[5]
8	LSF_STAGE1[6]	LSF_STAGE1[6]
9	LSF_STAGE2[0]	LSF_STAGE2[0]
10	LSF_STAGE2[1]	LSF_STAGE2[1]
11	LSF_STAGE2[2]	LSF_STAGE2[2]
12	LSF_STAGE2[3]	LSF_STAGE2[3]
13	LSF_STAGE2[4]	LSF_STAGE2[4]
14	LSF_STAGE2[5]	LSF_STAGE2[5]
15	LSF_STAGE2[6]	LSF_STAGE2[6]
16	LSF_STAGE3[0]	LSF_STAGE3[0]
17	LSF_STAGE3[1]	LSF_STAGE3[1]
18	LSF_STAGE3[2]	LSF_STAGE3[2]
19	LSF_STAGE3[3]	LSF_STAGE3[3]
20	LSF_STAGE3[4]	LSF_STAGE3[4]
21	LSF_STAGE3[5]	LSF_STAGE3[5]
22	LSF_STAGE4[0]	LSF_STAGE4[0]
23	LSF_STAGE4[1]	LSF_STAGE4[1]
24	LSF_STAGE4[2]	LSF_STAGE4[2]
25	LSF_STAGE4[3]	LSF_STAGE4[3]
26	LSF_SWTCHPRD	LSF_SWTCHPRD
27	LSF_INTRPATH[0]	PITCH_FRM[0]
28	LSF_INTRPATH[1]	PITCH_FRM[1]
29	PITC_SBFRM1[0]	PITCH_FRM[2]

30	PITC_SBFRM1[1]	PITCH_FRM[3]
31	PITC_SBFRM1[2]	PITCH_FRM[4]
32	PITC_SBFRM1[3]	PITCH_FRM[5]
33	PITC_SBFRM1[4]	PITCH_FRM[6]
34	PITC_SBFRM1[5]	PITCH_FRM[7]
35	PITC_SBFRM1[6]	PITCH_VQ_GAIN[0]
36	PITC_SBFRM1[7]	PITCH_VQ_GAIN[1]
37	PITC_SBFRM2[0]	PITCH_VQ_GAIN[2]
38	PITC_SBFRM2[1]	PITCH_VQ_GAIN[3]
39	PITC_SBFRM2[2]	PITCH_VQ_GAIN[4]
40	PITC_SBFRM2[3]	PITCH_VQ_GAIN[5]
41	PITC_SBFRM2[4]	PLCB_VQ_GAIN[0]
42	PITC_SBFRM3[0]	PLCB_VQ_GAIN[1]
43	PITC_SBFRM3[1]	PLCB_VQ_GAIN[2]
44	PITC_SBFRM3[2]	PLCB_VQ_GAIN[3]
45	PITC_SBFRM3[3]	PLCB_VQ_GAIN[4]
46	PITC_SBFRM3[4]	PLCB_VQ_GAIN[5]
47	PITC_SBFRM3[5]	PLCB_VQ_GAIN[6]
48	PITC_SBFRM3[6]	PLCB_VQ_GAIN[7]
49	PITC_SBFRM3[7]	PLCB_VQ_GAIN[8]
50	PITC_SBFRM4[0]	PLCB_VQ_GAIN[9]
51	PITC_SBFRM4[1]	PLCB_PSIGN1_5_SBF1
52	PITC_SBFRM4[2]	PLCB_PSIGN2_6_SBF1
53	PITC_SBFRM4[3]	PLCB_PSIGN3_7_SBF1
54	PITC_SBFRM4[4]	PLCB_PSIGN4_8_SBF1
55	GAIN_SBFRM1[0]	PLCB_PLOC1_SBF1[0]
56	GAIN_SBFRM1[1]	PLCB_PLOC1_SBF1[1]
57	GAIN_SBFRM1[2]	PLCB_PLOC1_SBF1[2]
58	GAIN_SBFRM1[3]	PLCB_PLOC1_SBF1[3]
59	GAIN_SBFRM1[4]	PLCB_PLOC2_SBF1[0]
60	GAIN_SBFRM1[5]	PLCB_PLOC2_SBF1[1]
61	GAIN_SBFRM1[6]	PLCB_PLOC2_SBF1[2]
62	GAIN_SBFRM2[0]	PLCB_PLOC3_SBF1[0]
63	GAIN_SBFRM2[1]	PLCB_PLOC3_SBF1[1]
64	GAIN_SBFRM2[2]	PLCB_PLOC3_SBF1[2]
65	GAIN_SBFRM2[3]	PLCB_PLOC4_SBF1[0]
66	GAIN_SBFRM2[4]	PLCB_PLOC4_SBF1[1]

67	GAIN_SBFRM2[5]		PLCB_PLOC4_SBF1[2]
68	GAIN_SBFRM2[6]		PLCB_PLOC5_SBF1[0]
69	GAIN_SBFRM3[0]		PLCB_PLOC5_SBF1[1]
70	GAIN_SBFRM3[1]		PLCB_PLOC5_SBF1[2]
71	GAIN_SBFRM3[2]		PLCB_PLOC5_SBF1[3]
72	GAIN_SBFRM3[3]		PLCB_PLOC6_SBF1[0]
73	GAIN_SBFRM3[4]		PLCB_PLOC6_SBF1[1]
74	GAIN_SBFRM3[5]		PLCB_PLOC6_SBF1[2]
75	GAIN_SBFRM3[6]		PLCB_PLOC7_SBF1[0]
76	GAIN_SBFRM4[0]		PLCB_PLOC7_SBF1[1]
77	GAIN_SBFRM4[1]		PLCB_PLOC7_SBF1[2]
78	GAIN_SBFRM4[2]		PLCB_PLOC8_SBF1[0]
79	GAIN_SBFRM4[3]		PLCB_PLOC8_SBF1[1]
80	GAIN_SBFRM4[4]		PLCB_PLOC8_SBF1[2]
81	GAIN_SBFRM4[5]		PLCB_PSIGN1_5_SBF2
82	GAIN_SBFRM4[6]		PLCB_PSIGN2_6_SBF2
83	PLCB_SLCT1_SBF1=1	PLCB_SLCT1_SBF1=0	PLCB_PSIGN3_7_SBF2
84	PLCB_PSIGN1_3_SBF1	PLCB_SLCT2_SBF1	PLCB_PSIGN4_8_SBF2
85	PLCB_PSIGN2_4_SBF1	PLCB_PSIGN1_SBF1	PLCB_PLOC1_SBF2[0]
86	PLCB_PSIGN5_SBF1	PLCB_PSIGN2_SBF1	PLCB_PLOC1_SBF2[1]
87	PLCB_PLOC1_SBF1[0]	PLCB_PSIGN3_SBF1	PLCB_PLOC1_SBF2[2]
88	PLCB_PLOC1_SBF1[1]	PLCB_PSIGN4_SBF1	PLCB_PLOC1_SBF2[3]
89	PLCB_PLOC1_SBF1[2]	PLCB_PSIGN5_SBF1	PLCB_PLOC2_SBF2[0]
90	PLCB_PLOC1_SBF1[3]	PLCB_PLOC1_SBF1[0]	PLCB_PLOC2_SBF2[1]
91	PLCB_PLOC2_SBF1[0]	PLCB_PLOC1_SBF1[1]	PLCB_PLOC2_SBF2[2]
92	PLCB_PLOC2_SBF1[1]	PLCB_PLOC1_SBF1[2]	PLCB_PLOC3_SBF2[0]
93	PLCB_PLOC2_SBF1[2]	PLCB_PLOC2_SBF1[0]	PLCB_PLOC3_SBF2[1]
94	PLCB_PLOC3_SBF1[0]	PLCB_PLOC2_SBF1[1]	PLCB_PLOC3_SBF2[2]
95	PLCB_PLOC3_SBF1[1]	PLCB_PLOC2_SBF1[2]	PLCB_PLOC4_SBF2[0]
96	PLCB_PLOC3_SBF1[2]	PLCB_PLOC3_SBF1[0]	PLCB_PLOC4_SBF2[1]
97	PLCB_PLOC3_SBF1[3]	PLCB_PLOC3_SBF1[1]	PLCB_PLOC4_SBF2[2]
98	PLCB_PLOC4_SBF1[0]	PLCB_PLOC3_SBF1[2]	PLCB_PLOC5_SBF2[0]
99	PLCB_PLOC4_SBF1[1]	PLCB_PLOC4_SBF1[0]	PLCB_PLOC5_SBF2[1]
100	PLCB_PLOC4_SBF1[2]	PLCB_PLOC4_SBF1[1]	PLCB_PLOC5_SBF2[2]
101	PLCB_PLOC5_SBF1[0]	PLCB_PLOC4_SBF1[2]	PLCB_PLOC5_SBF2[3]
102	PLCB_PLOC5_SBF1[1]	PLCB_PLOC5_SBF1[0]	PLCB_PLOC6_SBF2[0]
103	PLCB_PLOC5_SBF1[2]	PLCB_PLOC5_SBF1[1]	PLCB_PLOC6_SBF2[1]

104	PLCB_PLOC5_SBF1[3]	PLCB_PLOC5_SBF1[2]	PLCB_PLOC6_SBF2[2]
105	PLCB_SLCT1_SBF2=1	PLCB_SLCT1_SBF2=0	PLCB_PLOC7_SBF2[0]
106	PLCB_PSIGN13_SBF2	PLCB_SLCT2_SBF2	PLCB_PLOC7_SBF2[1]
107	PLCB_PSIGN24_SBF2	PLCB_PSIGN1_SBF2	PLCB_PLOC7_SBF2[2]
108	PLCB_PSIGN5_SBF2	PLCB_PSIGN2_SBF2	PLCB_PLOC8_SBF2[0]
109	PLCB_PLOC1_SBF2[0]	PLCB_PSIGN3_SBF2	PLCB_PLOC8_SBF2[1]
110	PLCB_PLOC1_SBF2[1]	PLCB_PSIGN4_SBF2	PLCB_PLOC8_SBF2[2]
111	PLCB_PLOC1_SBF2[2]	PLCB_PSIGN5_SBF2	PLCB_PSIGN1_5_SBF3
112	PLCB_PLOC1_SBF2[3]	PLCB_PLOC1_SBF2[0]	PLCB_PSIGN2_6_SBF3
113	PLCB_PLOC2_SBF2[0]	PLCB_PLOC1_SBF2[1]	PLCB_PSIGN3_7_SBF3
114	PLCB_PLOC2_SBF2[1]	PLCB_PLOC1_SBF2[2]	PLCB_PSIGN4_8_SBF3
115	PLCB_PLOC2_SBF2[2]	PLCB_PLOC2_SBF2[0]	PLCB_PLOC1_SBF3[0]
116	PLCB_PLOC3_SBF2[0]	PLCB_PLOC2_SBF2[1]	PLCB_PLOC1_SBF3[1]
117	PLCB_PLOC3_SBF2[1]	PLCB_PLOC2_SBF2[2]	PLCB_PLOC1_SBF3[2]
118	PLCB_PLOC3_SBF2[2]	PLCB_PLOC3_SBF2[0]	PLCB_PLOC1_SBF3[3]
119	PLCB_PLOC3_SBF2[3]	PLCB_PLOC3_SBF2[1]	PLCB_PLOC2_SBF3[0]
120	PLCB_PLOC4_SBF2[0]	PLCB_PLOC3_SBF2[2]	PLCB_PLOC2_SBF3[1]
121	PLCB_PLOC4_SBF2[1]	PLCB_PLOC4_SBF2[0]	PLCB_PLOC2_SBF3[2]
122	PLCB_PLOC4_SBF2[2]	PLCB_PLOC4_SBF2[1]	PLCB_PLOC3_SBF3[0]
123	PLCB_PLOC5_SBF2[0]	PLCB_PLOC4_SBF2[2]	PLCB_PLOC3_SBF3[1]
124	PLCB_PLOC5_SBF2[1]	PLCB_PLOC5_SBF2[0]	PLCB_PLOC3_SBF3[2]
125	PLCB_PLOC5_SBF2[2]	PLCB_PLOC5_SBF2[1]	PLCB_PLOC4_SBF3[0]
126	PLCB_PLOC5_SBF2[3]	PLCB_PLOC5_SBF2[2]	PLCB_PLOC4_SBF3[1]
127	PLCB_SLCT1_SBF3=1	PLCB_SLCT1_SBF3=0	PLCB_PLOC4_SBF3[2]
128	PLCB_PSIGN13_SBF3	PLCB_SLCT2_SBF3	PLCB_PLOC5_SBF3[0]
129	PLCB_PSIGN24_SBF3	PLCB_PSIGN1_SBF3	PLCB_PLOC5_SBF3[1]
130	PLCB_PSIGN5_SBF3	PLCB_PSIGN2_SBF3	PLCB_PLOC5_SBF3[2]
131	PLCB_PLOC1_SBF3[0]	PLCB_PSIGN3_SBF3	PLCB_PLOC5_SBF3[3]
132	PLCB_PLOC1_SBF3[1]	PLCB_PSIGN4_SBF3	PLCB_PLOC6_SBF3[0]
133	PLCB_PLOC1_SBF3[2]	PLCB_PSIGN5_SBF3	PLCB_PLOC6_SBF3[1]
134	PLCB_PLOC1_SBF3[3]	PLCB_PLOC1_SBF3[0]	PLCB_PLOC6_SBF3[2]
135	PLCB_PLOC2_SBF3[0]	PLCB_PLOC1_SBF3[1]	PLCB_PLOC7_SBF3[0]
136	PLCB_PLOC2_SBF3[1]	PLCB_PLOC1_SBF3[2]	PLCB_PLOC7_SBF3[1]
137	PLCB_PLOC2_SBF3[2]	PLCB_PLOC2_SBF3[0]	PLCB_PLOC7_SBF3[2]
138	PLCB_PLOC3_SBF3[0]	PLCB_PLOC2_SBF3[1]	PLCB_PLOC8_SBF3[0]
139	PLCB_PLOC3_SBF3[1]	PLCB_PLOC2_SBF3[2]	PLCB_PLOC8_SBF3[1]
140	PLCB_PLOC3_SBF3[2]	PLCB_PLOC3_SBF3[0]	PLCB_PLOC8_SBF3[2]

141	PLCB_PLOC3_SBF3[3]	PLCB_PLOC3_SBF3[1]	PLCB_PSIGN1_5_SBF4
142	PLCB_PLOC4_SBF3[0]	PLCB_PLOC3_SBF3[2]	PLCB_PSIGN2_6_SBF4
143	PLCB_PLOC4_SBF3[1]	PLCB_PLOC4_SBF3[0]	PLCB_PSIGN3_7_SBF4
144	PLCB_PLOC4_SBF3[2]	PLCB_PLOC4_SBF3[1]	PLCB_PSIGN4_8_SBF4
145	PLCB_PLOC5_SBF3[0]	PLCB_PLOC4_SBF3[2]	PLCB_PLOC1_SBF4[0]
146	PLCB_PLOC5_SBF3[1]	PLCB_PLOC5_SBF3[0]	PLCB_PLOC1_SBF4[1]
147	PLCB_PLOC5_SBF3[2]	PLCB_PLOC5_SBF3[1]	PLCB_PLOC1_SBF4[2]
148	PLCB_PLOC5_SBF3[3]	PLCB_PLOC5_SBF3[2]	PLCB_PLOC1_SBF4[3]
149	PLCB_SLCT1_SBF4=1	PLCB_SLCT1_SBF4=0	PLCB_PLOC2_SBF4[0]
150	PLCB_PSIGN13_SBF4	PLCB_SLCT2_SBF4	PLCB_PLOC2_SBF4[1]
151	PLCB_PSIGN24_SBF4	PLCB_PSIGN1_SBF4	PLCB_PLOC2_SBF4[2]
152	PLCB_PSIGN5_SBF4	PLCB_PSIGN2_SBF4	PLCB_PLOC3_SBF4[0]
153	PLCB_PLOC1_SBF4[0]	PLCB_PSIGN3_SBF4	PLCB_PLOC3_SBF4[1]
154	PLCB_PLOC1_SBF4[1]	PLCB_PSIGN4_SBF4	PLCB_PLOC3_SBF4[2]
155	PLCB_PLOC1_SBF4[2]	PLCB_PSIGN5_SBF4	PLCB_PLOC4_SBF4[0]
156	PLCB_PLOC1_SBF4[3]	PLCB_PLOC1_SBF4[0]	PLCB_PLOC4_SBF4[1]
157	PLCB_PLOC2_SBF4[0]	PLCB_PLOC1_SBF4[1]	PLCB_PLOC4_SBF4[2]
158	PLCB_PLOC2_SBF4[1]	PLCB_PLOC1_SBF4[2]	PLCB_PLOC5_SBF4[0]
159	PLCB_PLOC2_SBF4[2]	PLCB_PLOC2_SBF4[0]	PLCB_PLOC5_SBF4[1]
160	PLCB_PLOC3_SBF4[0]	PLCB_PLOC2_SBF4[1]	PLCB_PLOC5_SBF4[2]
161	PLCB_PLOC3_SBF4[1]	PLCB_PLOC2_SBF4[2]	PLCB_PLOC5_SBF4[3]
162	PLCB_PLOC3_SBF4[2]	PLCB_PLOC3_SBF4[0]	PLCB_PLOC6_SBF4[0]
163	PLCB_PLOC3_SBF4[3]	PLCB_PLOC3_SBF4[1]	PLCB_PLOC6_SBF4[1]
164	PLCB_PLOC4_SBF4[0]	PLCB_PLOC3_SBF4[2]	PLCB_PLOC6_SBF4[2]
165	PLCB_PLOC4_SBF4[1]	PLCB_PLOC4_SBF4[0]	PLCB_PLOC7_SBF4[0]
166	PLCB_PLOC4_SBF4[2]	PLCB_PLOC4_SBF4[1]	PLCB_PLOC7_SBF4[1]
167	PLCB_PLOC5_SBF4[0]	PLCB_PLOC4_SBF4[2]	PLCB_PLOC7_SBF4[2]
168	PLCB_PLOC5_SBF4[1]	PLCB_PLOC5_SBF4[0]	PLCB_PLOC8_SBF4[0]
169	PLCB_PLOC5_SBF4[2]	PLCB_PLOC5_SBF4[1]	PLCB_PLOC8_SBF[1]
170	PLCB_PLOC5_SBF4[3]	PLCB_PLOC5_SBF4[2]	PLCB_PLOC8_SBF4[2]
171	Rate Sanity Flag = 0		

Table 4.3-1: Bit Stream for Rate 1

1

Bit Index	Rate 1/2 Frame Type 0 Packet Bits
1	SVS_DECI=0
2	LSF_STAGE1[0]
3	LSF_STAGE1[1]
4	LSF_STAGE1[2]
5	LSF_STAGE1[3]
6	LSF_STAGE1[4]
7	LSF_STAGE1[5]
8	LSF_STAGE1[6]
9	LSF_STAGE2[0]
10	LSF_STAGE2[1]
11	LSF_STAGE2[2]
12	LSF_STAGE2[3]
13	LSF_STAGE2[4]
14	LSF_STAGE2[5]
15	LSF_STAGE2[6]
16	LSF_STAGE3[0]
17	LSF_STAGE3[1]
18	LSF_STAGE3[2]
19	LSF_STAGE3[3]
20	LSF_STAGE3[4]
21	LSF_STAGE3[5]
22	LSF_SWTCHPRD
23	PITC_SBFRM1[0]
24	PITC_SBFRM1[1]
25	PITC_SBFRM1[2]
26	PITC_SBFRM1[3]
27	PITC_SBFRM1[4]
28	PITC_SBFRM1[5]
29	PITC_SBFRM1[6]
30	PITC_SBFRM2[0]
31	PITC_SBFRM2[1]
32	PITC_SBFRM2[2]
33	PITC_SBFRM2[3]

34	PITC_SBFRM2[4]		
35	PITC_SBFRM2[5]		
36	PITC_SBFRM2[6]		
37	GAIN_SBFRM1[0]		
38	GAIN_SBFRM1[1]		
39	GAIN_SBFRM1[2]		
40	GAIN_SBFRM1[3]		
41	GAIN_SBFRM1[4]		
42	GAIN_SBFRM1[5]		
43	GAIN_SBFRM1[6]		
44	GAIN_SBFRM2[0]		
45	GAIN_SBFRM2[1]		
46	GAIN_SBFRM2[2]		
47	GAIN_SBFRM2[3]		
48	GAIN_SBFRM2[4]		
49	GAIN_SBFRM2[5]		
50	GAIN_SBFRM2[6]		
51	PLCB_SLCT1_SBF1=1	PLCB_SLCT1_SBF1=0	
52	PLCB_PSIGN1_2_SBF1	PLCB_SLCT2_SBF1=1	PLCB_SLCT2_SBF1=0
53	PLCB_PLOC1_2_SBF1[0]	PLCB_PSIGN1_SBF1	GCB_VSIGN1_SBF1
54	PLCB_PLOC1_2_SBF1[1]	PLCB_PSIGN2_SBF1	GCB_VSIGN2_SBF1
55	PLCB_PLOC1_2_SBF1[2]	PLCB_PSIGN3_SBF1	GCB_VEC1_2_SBF1[0]
56	PLCB_PLOC1_2_SBF1[3]	PLCB_PLOC1_SBF1[0]	GCB_VEC1_2_SBF1[1]
57	PLCB_PLOC1_2_SBF1[4]	PLCB_PLOC1_SBF1[1]	GCB_VEC1_2_SBF1[2]
58	PLCB_PLOC1_2_SBF1[5]	PLCB_PLOC1_SBF1[2]	GCB_VEC1_2_SBF1[3]
59	PLCB_PLOC1_2_SBF1[6]	PLCB_PLOC1_SBF1[3]	GCB_VEC1_2_SBF1[4]
60	PLCB_PLOC1_2_SBF1[7]	PLCB_PLOC2_SBF1[0]	GCB_VEC1_2_SBF1[5]
61	PLCB_PLOC1_2_SBF1[8]	PLCB_PLOC2_SBF1[1]	GCB_VEC1_2_SBF1[6]
62	PLCB_PLOC1_2_SBF1[9]	PLCB_PLOC2_SBF1[2]	GCB_VEC1_2_SBF1[7]
63	PLCB_PLOC1_2_SBF1[10]	PLCB_PLOC3_SBF1[0]	GCB_VEC1_2_SBF1[8]
64	PLCB_PLOC1_2_SBF1[11]	PLCB_PLOC3_SBF1[1]	GCB_VEC1_2_SBF1[9]
65	PLCB_PLOC1_2_SBF1[12]	PLCB_PLOC3_SBF1[2]	GCB_VEC1_2_SBF1[10]
66	PLCB_SLCT1_SBF2=1	PLCB_SLCT1_SBF2=0	
67	PLCB_PSIGN1_2_SBF2	PLCB_SLCT2_SBF2=1	PLCB_SLCT2_SBF2=0
68	PLCB_PLOC1_2_SBF2[0]	PLCB_PSIGN1_SBF2	GCB_VSIGN1_SBF2
69	PLCB_PLOC1_2_SBF2[1]	PLCB_PSIGN2_SBF2	GCB_VSIGN2_SBF2
70	PLCB_PLOC1_2_SBF2[2]	PLCB_PSIGN3_SBF2	GCB_VEC1_2_SBF2[0]

71	PLCB_PLOC1_2_SBF2[3]	PLCB_PLOC1_SBF2[0]	GCB_VEC1_2_SBF2[1]
72	PLCB_PLOC1_2_SBF2[4]	PLCB_PLOC1_SBF2[1]	GCB_VEC1_2_SBF2[2]
73	PLCB_PLOC1_2_SBF2[5]	PLCB_PLOC1_SBF2[2]	GCB_VEC1_2_SBF2[3]
74	PLCB_PLOC1_2_SBF2[6]	PLCB_PLOC1_SBF2[3]	GCB_VEC1_2_SBF2[4]
75	PLCB_PLOC1_2_SBF2[7]	PLCB_PLOC2_SBF2[0]	GCB_VEC1_2_SBF2[5]
76	PLCB_PLOC1_2_SBF2[8]	PLCB_PLOC2_SBF2[1]	GCB_VEC1_2_SBF2[6]
77	PLCB_PLOC1_2_SBF2[9]	PLCB_PLOC2_SBF2[2]	GCB_VEC1_2_SBF2[7]
78	PLCB_PLOC1_2_SBF2[10]	PLCB_PLOC3_SBF2[0]	GCB_VEC1_2_SBF2[8]
79	PLCB_PLOC1_2_SBF2[11]	PLCB_PLOC3_SBF2[1]	GCB_VEC1_2_SBF2[9]
80	PLCB_PLOC1_2_SBF2[12]	PLCB_PLOC3_SBF2[2]	GCB_VEC1_2_SBF2[10]

Table 4.3-2: Bit Stream for Rate 1/2 Type-0 Frames

Bit Index	Rate 1/2 Frame Type 1 Packet Bits
1	SVS_DECI=1
2	LSF_STAGE1[0]
3	LSF_STAGE1[1]
4	LSF_STAGE1[2]
5	LSF_STAGE1[3]
6	LSF_STAGE1[4]
7	LSF_STAGE1[5]
8	LSF_STAGE1[6]
9	LSF_STAGE2[0]
10	LSF_STAGE2[1]
11	LSF_STAGE2[2]
12	LSF_STAGE2[3]
13	LSF_STAGE2[4]
14	LSF_STAGE2[5]
15	LSF_STAGE2[6]
16	LSF_STAGE3[0]
17	LSF_STAGE3[1]
18	LSF_STAGE3[2]
19	LSF_STAGE3[3]
20	LSF_STAGE3[4]

21	LSF_STAGE3[5]		
22	LSF_SWTCHPRD		
23	PITCH_FRM[0]		
24	PITCH_FRM[1]		
25	PITCH_FRM[2]		
26	PITCH_FRM[3]		
27	PITCH_FRM[4]		
28	PITCH_FRM[5]		
29	PITCH_FRM[6]		
30	PITCH_VQ_GAIN[0]		
31	PITCH_VQ_GAIN[1]		
32	PITCH_VQ_GAIN[2]		
33	PITCH_VQ_GAIN[3]		
34	PLCB_VQ_GAIN[0]		
35	PLCB_VQ_GAIN[1]		
36	PLCB_VQ_GAIN[2]		
37	PLCB_VQ_GAIN[3]		
38	PLCB_VQ_GAIN[4]		
39	PLCB_VQ_GAIN[5]		
40	PLCB_VQ_GAIN[6]		
41	PLCB_VQ_GAIN[7]		
42	PLCB_SLCT1_SBF1=1	PLCB_SLCT1_SBF1=0	
43	PLCB_PSIGN1_SBF1	PLCB_SLCT2_SBF1=1	PLCB_SLCT2_SBF1=0
44	PLCB_PSIGN2_SBF1	PLCB_PSIGN1_SBF1	PLCB_PSIGN1_SBF1
45	PLCB_PLOC1_SBF1[0]	PLCB_PSIGN2_SBF1	PLCB_PSIGN2_SBF1
46	PLCB_PLOC1_SBF1[1]	PLCB_PSIGN3_SBF1	PLCB_PSIGN3_SBF1
47	PLCB_PLOC1_SBF1[2]	PLCB_PLOC1_SBF1[0]	PLCB_PSIGN4_SBF1
48	PLCB_PLOC1_SBF1[3]	PLCB_PLOC1_SBF1[1]	PLCB_PSIGN5_SBF1
49	PLCB_PLOC1_SBF1[4]	PLCB_PLOC1_SBF1[2]	PLCB_PLOC1_SBF1[0]
50	PLCB_PLOC2_SBF1[0]	PLCB_PLOC1_SBF1[3]	PLCB_PLOC1_SBF1[1]
51	PLCB_PLOC2_SBF1[1]	PLCB_PLOC2_SBF1[0]	PLCB_PLOC2_SBF1[0]
52	PLCB_PLOC2_SBF1[2]	PLCB_PLOC2_SBF1[1]	PLCB_PLOC3_SBF1[0]
53	PLCB_PLOC2_SBF1[3]	PLCB_PLOC3_SBF1[0]	PLCB_PLOC4_SBF1[0]
54	PLCB_PLOC2_SBF1[4]	PLCB_PLOC3_SBF1[1]	PLCB_PLOC5_SBF1[0]
55	PLCB_SLCT1_SBF2=1	PLCB_SLCT1_SBF2=0	
56	PLCB_PSIGN1_SBF2	PLCB_SLCT2_SBF2=1	PLCB_SLCT2_SBF2=0
57	PLCB_PSIGN2_SBF2	PLCB_PSIGN1_SBF2	PLCB_PSIGN1_SBF2

58	PLCB_PLOC1_SBF2[0]	PLCB_PSIGN2_SBF2	PLCB_PSIGN2_SBF2
59	PLCB_PLOC1_SBF2[1]	PLCB_PSIGN3_SBF2	PLCB_PSIGN3_SBF2
60	PLCB_PLOC1_SBF2[2]	PLCB_PLOC1_SBF2[0]	PLCB_PSIGN4_SBF2
61	PLCB_PLOC1_SBF2[3]	PLCB_PLOC1_SBF2[1]	PLCB_PSIGN5_SBF2
62	PLCB_PLOC1_SBF2[4]	PLCB_PLOC1_SBF2[2]	PLCB_PLOC1_SBF2[0]
63	PLCB_PLOC2_SBF2[0]	PLCB_PLOC1_SBF2[3]	PLCB_PLOC1_SBF2[1]
64	PLCB_PLOC2_SBF2[1]	PLCB_PLOC2_SBF2[0]	PLCB_PLOC2_SBF2[0]
65	PLCB_PLOC2_SBF2[2]	PLCB_PLOC2_SBF2[1]	PLCB_PLOC3_SBF2[0]
66	PLCB_PLOC2_SBF2[3]	PLCB_PLOC3_SBF2[0]	PLCB_PLOC4_SBF2[0]
67	PLCB_PLOC2_SBF2[4]	PLCB_PLOC3_SBF2[1]	PLCB_PLOC5_SBF2[0]
68	PLCB_SLCT1_SBF3=1	PLCB_SLCT1_SBF3=0	
69	PLCB_PSIGN1_SBF3	PLCB_SLCT2_SBF3=1	PLCB_SLCT2_SBF3=0
70	PLCB_PSIGN2_SBF3	PLCB_PSIGN1_SBF3	PLCB_PSIGN1_SBF3
71	PLCB_PLOC1_SBF3[0]	PLCB_PSIGN2_SBF3	PLCB_PSIGN2_SBF3
72	PLCB_PLOC1_SBF3[1]	PLCB_PSIGN3_SBF3	PLCB_PSIGN3_SBF3
73	PLCB_PLOC1_SBF3[2]	PLCB_PLOC1_SBF3[0]	PLCB_PSIGN4_SBF3
74	PLCB_PLOC1_SBF3[3]	PLCB_PLOC1_SBF3[1]	PLCB_PSIGN5_SBF3
75	PLCB_PLOC1_SBF3[4]	PLCB_PLOC1_SBF3[2]	PLCB_PLOC1_SBF3[0]
76	PLCB_PLOC2_SBF3[0]	PLCB_PLOC1_SBF3[3]	PLCB_PLOC1_SBF3[1]
77	PLCB_PLOC2_SBF3[1]	PLCB_PLOC2_SBF3[0]	PLCB_PLOC2_SBF3[0]
78	PLCB_PLOC2_SBF3[2]	PLCB_PLOC2_SBF3[1]	PLCB_PLOC3_SBF3[0]
79	PLCB_PLOC2_SBF3[3]	PLCB_PLOC3_SBF3[0]	PLCB_PLOC4_SBF3[0]
80	PLCB_PLOC2_SBF3[4]	PLCB_PLOC3_SBF3[1]	PLCB_PLOC5_SBF3[0]

Table 4.3-3: Bit Stream for Rate 1/2 Type-1 Frames

1

Bit Index	Rate 1/4 Packet Bits
1	LSF_STAGE1[0]
2	LSF_STAGE1[1]
3	LSF_STAGE1[2]
4	LSF_STAGE1[3]
5	LSF_STAGE1[4]
6	LSF_STAGE1[5]
7	LSF_STAGE1[6]
8	LSF_STAGE2[0]
9	LSF_STAGE2[1]
10	LSF_STAGE2[2]
11	LSF_STAGE2[3]
12	LSF_STAGE2[4]
13	LSF_STAGE2[5]
14	LSF_STAGE2[6]
15	LSF_STAGE3[0]
16	LSF_STAGE3[1]
17	LSF_STAGE3[2]
18	LSF_STAGE3[3]
19	LSF_STAGE3[4]
20	LSF_STAGE3[5]
21	GAIN_VQ[0]
22	GAIN_VQ[1]
23	GAIN_VQ[2]
24	GAIN_VQ[3]
25	GAIN_VQ[4]
26	GAIN_VQ_1[0]
27	GAIN_VQ_1[1]
28	GAIN_VQ_1[2]
29	GAIN_VQ_1[3]
30	GAIN_VQ_1[4]
31	GAIN_VQ_1[5]
32	GAIN_VQ_2[0]
33	GAIN_VQ_2[1]
34	GAIN_VQ_2[2]

35	GAIN_VQ_2[3]
36	GAIN_VQ_2[4]
37	GAIN_VQ_2[5]
38	SF_IDX[0]
39	SF_IDX[1]
40	Rate Sanity Flag = 0

Table 4.3-4: Bit Stream for Rate 1/4

Bit Index	Rate 1/8 Packet Bits
1	LSF_STAGE1[0]
2	LSF_STAGE1[1]
3	LSF_STAGE1[2]
4	LSF_STAGE1[3]
5	LSF_STAGE2[0]
6	LSF_STAGE2[1]
7	LSF_STAGE2[2]
8	LSF_STAGE2[3]
9	LSF_STAGE3[0]
10	LSF_STAGE3[1]
11	LSF_STAGE3[2]
12	GAIN_08K[0]
13	GAIN_08K[1]
14	GAIN_08K[2]
15	GAIN_08K[3]
16	GAIN_08K[4]

Table 4.3-5: Bit Stream for Rate 1/8

1

Bit Index	Rate 1 Data + 1/2 Rate Speech Packet Bits	Rate 1 Data + Auxiliary Data Packet Bits
1	SVS_DECI = 0	SVS_DECI = 0
2	HALF_RATE_SPEECH[0]	AUX_DATA[0]
3	HALF_RATE_SPEECH [1]	AUX_DATA [1]
4	HALF_RATE_SPEECH [2]	AUX_DATA [2]
5	HALF_RATE_SPEECH [3]	AUX_DATA [3]
6	HALF_RATE_SPEECH [4]	AUX_DATA [4]
7	HALF_RATE_SPEECH [5]	AUX_DATA [5]
8	HALF_RATE_SPEECH [6]	AUX_DATA [6]
9	HALF_RATE_SPEECH [7]	AUX_DATA [7]
10	HALF_RATE_SPEECH [8]	AUX_DATA [8]
11	HALF_RATE_SPEECH [9]	AUX_DATA [9]
12	HALF_RATE_SPEECH [10]	AUX_DATA [10]
13	HALF_RATE_SPEECH [11]	AUX_DATA [11]
14	HALF_RATE_SPEECH [12]	AUX_DATA [12]
15	HALF_RATE_SPEECH [13]	AUX_DATA [13]
16	HALF_RATE_SPEECH [14]	AUX_DATA [14]
17	HALF_RATE_SPEECH [15]	AUX_DATA [15]
18	HALF_RATE_SPEECH [16]	AUX_DATA [16]
19	HALF_RATE_SPEECH [17]	AUX_DATA [17]
20	HALF_RATE_SPEECH [18]	AUX_DATA [18]
21	HALF_RATE_SPEECH [19]	AUX_DATA [19]
22	HALF_RATE_SPEECH [20]	AUX_DATA [20]
23	HALF_RATE_SPEECH [21]	AUX_DATA [21]
24	HALF_RATE_SPEECH [22]	AUX_DATA [22]
25	HALF_RATE_SPEECH [23]	AUX_DATA [23]
26	HALF_RATE_SPEECH [24]	AUX_DATA [24]
27	HALF_RATE_SPEECH [25]	AUX_DATA [25]
28	HALF_RATE_SPEECH [26]	AUX_DATA [26]
29	HALF_RATE_SPEECH [27]	AUX_DATA [27]
30	HALF_RATE_SPEECH [28]	AUX_DATA [28]
31	HALF_RATE_SPEECH [29]	AUX_DATA [29]
32	HALF_RATE_SPEECH [30]	AUX_DATA [30]
33	HALF_RATE_SPEECH [31]	AUX_DATA [31]
34	HALF_RATE_SPEECH [32]	AUX_DATA [32]

35	HALF_RATE_SPEECH [33]	AUX_DATA [33]
36	HALF_RATE_SPEECH [34]	AUX_DATA [34]
37	HALF_RATE_SPEECH [35]	AUX_DATA [35]
38	HALF_RATE_SPEECH [36]	AUX_DATA [36]
39	HALF_RATE_SPEECH [37]	AUX_DATA [37]
40	HALF_RATE_SPEECH [38]	AUX_DATA [38]
41	HALF_RATE_SPEECH [39]	AUX_DATA [39]
42	HALF_RATE_SPEECH [40]	AUX_DATA [40]
43	HALF_RATE_SPEECH [41]	AUX_DATA [41]
44	HALF_RATE_SPEECH [42]	AUX_DATA [42]
45	HALF_RATE_SPEECH [43]	AUX_DATA [43]
46	HALF_RATE_SPEECH [44]	AUX_DATA [44]
47	HALF_RATE_SPEECH [45]	AUX_DATA [45]
48	HALF_RATE_SPEECH [46]	AUX_DATA [46]
49	HALF_RATE_SPEECH [47]	AUX_DATA [47]
50	HALF_RATE_SPEECH [48]	AUX_DATA [48]
51	HALF_RATE_SPEECH [49]	AUX_DATA [49]
52	HALF_RATE_SPEECH [50]	AUX_DATA [50]
53	HALF_RATE_SPEECH [51]	AUX_DATA [51]
54	HALF_RATE_SPEECH [52]	AUX_DATA [52]
55	HALF_RATE_SPEECH [53]	AUX_DATA [53]
56	HALF_RATE_SPEECH [54]	AUX_DATA [54]
57	HALF_RATE_SPEECH [55]	AUX_DATA [55]
58	HALF_RATE_SPEECH [56]	AUX_DATA [56]
59	HALF_RATE_SPEECH [57]	AUX_DATA [57]
60	HALF_RATE_SPEECH [58]	AUX_DATA [58]
61	HALF_RATE_SPEECH [59]	AUX_DATA [59]
62	HALF_RATE_SPEECH [60]	AUX_DATA [60]
63	HALF_RATE_SPEECH [61]	AUX_DATA [61]
64	HALF_RATE_SPEECH [62]	AUX_DATA [62]
65	HALF_RATE_SPEECH [63]	AUX_DATA [63]
66	HALF_RATE_SPEECH [64]	AUX_DATA [64]
67	HALF_RATE_SPEECH [65]	AUX_DATA [65]
68	HALF_RATE_SPEECH [66]	AUX_DATA [66]
69	HALF_RATE_SPEECH [67]	AUX_DATA [67]
70	HALF_RATE_SPEECH [68]	AUX_DATA [68]
71	HALF_RATE_SPEECH [69]	AUX_DATA [69]

72	HALF_RATE_SPEECH [70]	AUX_DATA [70]
73	HALF_RATE_SPEECH [71]	AUX_DATA [71]
74	HALF_RATE_SPEECH [72]	AUX_DATA [72]
75	HALF_RATE_SPEECH [73]	AUX_DATA [73]
76	HALF_RATE_SPEECH [74]	AUX_DATA [74]
77	HALF_RATE_SPEECH [75]	AUX_DATA [75]
78	HALF_RATE_SPEECH [76]	AUX_DATA [76]
79	HALF_RATE_SPEECH [77]	AUX_DATA [77]
80	HALF_RATE_SPEECH [78]	AUX_DATA [78]
81	HALF_RATE_SPEECH [79]	AUX_DATA [79]
82	DATA_HEADER[0]	DATA_HEADER[0]
83	DATA_HEADER[1]	DATA_HEADER[1]
84	DATA_HEADER[2]	DATA_HEADER[2]
85	DATA_BLOCK[0]	DATA_BLOCK[0]
86	DATA_BLOCK [1]	DATA_BLOCK [1]
87	DATA_BLOCK [2]	DATA_BLOCK [2]
88	DATA_BLOCK [3]	DATA_BLOCK [3]
89	DATA_BLOCK [4]	DATA_BLOCK [4]
90	DATA_BLOCK [5]	DATA_BLOCK [5]
91	DATA_BLOCK [6]	DATA_BLOCK [6]
92	DATA_BLOCK [7]	DATA_BLOCK [7]
93	DATA_BLOCK [8]	DATA_BLOCK [8]
94	DATA_BLOCK [9]	DATA_BLOCK [9]
95	DATA_BLOCK[10]	DATA_BLOCK[10]
96	DATA_BLOCK [11]	DATA_BLOCK [11]
97	DATA_BLOCK [12]	DATA_BLOCK [12]
98	DATA_BLOCK [13]	DATA_BLOCK [13]
99	DATA_BLOCK [14]	DATA_BLOCK [14]
100	DATA_BLOCK [15]	DATA_BLOCK [15]
101	DATA_BLOCK [16]	DATA_BLOCK [16]
102	DATA_BLOCK [17]	DATA_BLOCK [17]
103	DATA_BLOCK [18]	DATA_BLOCK [18]
104	DATA_BLOCK [19]	DATA_BLOCK [19]
105	DATA_BLOCK[20]	DATA_BLOCK[20]
106	DATA_BLOCK [21]	DATA_BLOCK [21]
107	DATA_BLOCK [22]	DATA_BLOCK [22]
108	DATA_BLOCK [23]	DATA_BLOCK [23]

109	DATA_BLOCK [24]	DATA_BLOCK [24]
110	DATA_BLOCK [25]	DATA_BLOCK [25]
111	DATA_BLOCK [26]	DATA_BLOCK [26]
112	DATA_BLOCK [27]	DATA_BLOCK [27]
113	DATA_BLOCK [28]	DATA_BLOCK [28]
114	DATA_BLOCK [29]	DATA_BLOCK [29]
115	DATA_BLOCK[30]	DATA_BLOCK[30]
116	DATA_BLOCK [31]	DATA_BLOCK [31]
117	DATA_BLOCK [32]	DATA_BLOCK [32]
118	DATA_BLOCK [33]	DATA_BLOCK [33]
119	DATA_BLOCK [34]	DATA_BLOCK [34]
120	DATA_BLOCK [35]	DATA_BLOCK [35]
121	DATA_BLOCK [36]	DATA_BLOCK [36]
122	DATA_BLOCK [37]	DATA_BLOCK [37]
123	DATA_BLOCK [38]	DATA_BLOCK [38]
124	DATA_BLOCK [39]	DATA_BLOCK [39]
125	DATA_BLOCK[40]	DATA_BLOCK[40]
126	DATA_BLOCK [41]	DATA_BLOCK [41]
127	DATA_BLOCK [42]	DATA_BLOCK [42]
128	DATA_BLOCK [43]	DATA_BLOCK [43]
129	DATA_BLOCK [44]	DATA_BLOCK [44]
130	DATA_BLOCK [45]	DATA_BLOCK [45]
131	DATA_BLOCK [46]	DATA_BLOCK [46]
132	DATA_BLOCK [47]	DATA_BLOCK [47]
133	DATA_BLOCK [48]	DATA_BLOCK [48]
134	DATA_BLOCK [49]	DATA_BLOCK [49]
135	DATA_BLOCK[50]	DATA_BLOCK[50]
136	DATA_BLOCK [51]	DATA_BLOCK [51]
137	DATA_BLOCK [52]	DATA_BLOCK [52]
138	DATA_BLOCK [53]	DATA_BLOCK [53]
139	DATA_BLOCK [54]	DATA_BLOCK [54]
140	DATA_BLOCK [55]	DATA_BLOCK [55]
141	DATA_BLOCK [56]	DATA_BLOCK [56]
142	DATA_BLOCK [57]	DATA_BLOCK [57]
143	DATA_BLOCK [58]	DATA_BLOCK [58]
144	DATA_BLOCK [59]	DATA_BLOCK [59]
145	DATA_BLOCK[60]	DATA_BLOCK[60]

146	DATA_BLOCK [61]	DATA_BLOCK [61]
147	DATA_BLOCK [62]	DATA_BLOCK [62]
148	DATA_BLOCK [63]	DATA_BLOCK [63]
149	ILLEGAL_CODEWORD[0] = 1	ILLEGAL_CODEWORD[0] = 1
150	ILLEGAL_CODEWORD[1] = 0	ILLEGAL_CODEWORD[1] = 0
151	ILLEGAL_CODEWORD[2] = 0	ILLEGAL_CODEWORD[2] = 0
152	ILLEGAL_CODEWORD[3] = 0	ILLEGAL_CODEWORD[3] = 0
153	ILLEGAL_CODEWORD[4] = 0	ILLEGAL_CODEWORD[4] = 0
154	ILLEGAL_CODEWORD[5] = 0	ILLEGAL_CODEWORD[5] = 0
155	ILLEGAL_CODEWORD[6] = 0	ILLEGAL_CODEWORD[6] = 0
156	ILLEGAL_CODEWORD[7] = 0	ILLEGAL_CODEWORD[7] = 0
157	ILLEGAL_CODEWORD[8] = 0	ILLEGAL_CODEWORD[8] = 0
158	ILLEGAL_CODEWORD[9] = 0	ILLEGAL_CODEWORD[9] = 0
159	ILLEGAL_CODEWORD[10] = 0	ILLEGAL_CODEWORD[10] = 0
160	ILLEGAL_CODEWORD[11] = 0	ILLEGAL_CODEWORD[11] = 0
161	ILLEGAL_CODEWORD[12] = 1	ILLEGAL_CODEWORD[12] = 1
162	ILLEGAL_CODEWORD[13] = 0	ILLEGAL_CODEWORD[13] = 0
163	ILLEGAL_CODEWORD[14] = 1	ILLEGAL_CODEWORD[14] = 1
164	ILLEGAL_CODEWORD[15] = 1	ILLEGAL_CODEWORD[15] = 1
165	ILLEGAL_CODEWORD[16] = 1	ILLEGAL_CODEWORD[16] = 1
166	ILLEGAL_CODEWORD[17] = 0	ILLEGAL_CODEWORD[17] = 0
167	ILLEGAL_CODEWORD[18] = 1	ILLEGAL_CODEWORD[18] = 1
168	ILLEGAL_CODEWORD[19] = 0	ILLEGAL_CODEWORD[19] = 0
169	ILLEGAL_CODEWORD[20] = 1	ILLEGAL_CODEWORD[20] = 1
170	ILLEGAL_CODEWORD[21] = 0	ILLEGAL_CODEWORD[21] = 0
171	Rate Sanity Flag = 0	Rate Sanity Flag = 0

Table 4.3-6: Data Bit Streams for Rate 1

1

Bit Index	Rate 1/2 Half Rate Data Packet Bits	Rate 1/2 Half Rate TTY Data Packet Bits	Rate 1/2 Half Rate DTMF Data Packet Bits
1	SVS_DECI = 0	SVS_DECI = 0	SVS_DECI = 0
2	DATA_HEADER[0]	DATA_HEADER[0] = 0	DATA_HEADER[0] = 0
3	DATA_HEADER[1]	DATA_HEADER[1] = 1	DATA_HEADER[1] = 1
4	DATA_HEADER[2]	DATA_HEADER[2] = 0	DATA_HEADER[2] = 1
5	DATA_BLOCK[0]	TTY_TYPE[0]	DTMF_DIGIT[0]
6	DATA_BLOCK [1]	TTY_TYPE [1]	DTMF_DIGIT [1]
7	DATA_BLOCK [2]	TTY_TYPE [2]	DTMF_DIGIT [2]
8	DATA_BLOCK [3]	TTY_TYPE [3]	DTMF_DIGIT [3]
9	DATA_BLOCK [4]	TTY_TYPE [4]	DTMF_DIGIT [4]
10	DATA_BLOCK [5]	TTY_TYPE [5]	DTMF_DIGIT [5]
11	DATA_BLOCK [6]	TTY_TYPE [6]	DTMF_DIGIT [6]
12	DATA_BLOCK [7]	TTY_TYPE [7]	DTMF_DIGIT [7]
13	DATA_BLOCK [8]	TTY_HEADER[0]	DTMF_DIGIT [8]
14	DATA_BLOCK [9]	TTY_HEADER [1]	DTMF_DIGIT [9]
15	DATA_BLOCK[10]	TTY_HEADER [2]	DTMF_DIGIT[10]
16	DATA_BLOCK [11]	TTY_HEADER [3]	DTMF_DIGIT [11]
17	DATA_BLOCK [12]	TTY_HEADER [4]	DTMF_DIGIT [12]
18	DATA_BLOCK [13]	TTY_HEADER [5]	DTMF_DIGIT [13]
19	DATA_BLOCK [14]	TTY_HEADER [6]	DTMF_DIGIT [14]
20	DATA_BLOCK [15]	TTY_HEADER [7]	DTMF_DIGIT [15]
21	DATA_BLOCK [16]	TTY_CHAR[0]	RESERVED
22	DATA_BLOCK [17]	TTY_CHAR [1]	RESERVED
23	DATA_BLOCK [18]	TTY_CHAR [2]	RESERVED
24	DATA_BLOCK [19]	TTY_CHAR [3]	RESERVED
25	DATA_BLOCK[20]	TTY_CHAR [4]	RESERVED
26	DATA_BLOCK [21]	TTY_CHAR [5]	RESERVED
27	DATA_BLOCK [22]	TTY_CHAR [6]	RESERVED
28	DATA_BLOCK [23]	TTY_CHAR [7]	RESERVED
29	DATA_BLOCK [24]	TTY_CHAR [8]	RESERVED
30	DATA_BLOCK [25]	TTY_CHAR [9]	RESERVED
31	DATA_BLOCK [26]	TTY_CHAR[10]	RESERVED
32	DATA_BLOCK [27]	TTY_CHAR [11]	RESERVED
33	DATA_BLOCK [28]	TTY_CHAR [12]	RESERVED

34	DATA_BLOCK [29]	TTY_CHAR [13]	RESERVED
35	DATA_BLOCK[30]	TTY_CHAR [14]	RESERVED
36	DATA_BLOCK [31]	TTY_CHAR [15]	RESERVED
37	DATA_BLOCK [32]	TTY_RATE[0]	RESERVED
38	DATA_BLOCK [33]	TTY_RATE [1]	RESERVED
39	DATA_BLOCK [34]	TTY_RATE [2]	RESERVED
40	DATA_BLOCK [35]	TTY_RATE [3]	RESERVED
41	DATA_BLOCK [36]	TTY_RATE [4]	RESERVED
42	DATA_BLOCK [37]	TTY_RATE [5]	RESERVED
43	DATA_BLOCK [38]	TTY_RATE [6]	RESERVED
44	DATA_BLOCK [39]	TTY_RATE [7]	RESERVED
45	DATA_BLOCK[40]	TTY_RATE [8]	RESERVED
46	DATA_BLOCK [41]	TTY_RATE [9]	RESERVED
47	DATA_BLOCK [42]	TTY_RATE[10]	RESERVED
48	DATA_BLOCK [43]	TTY_RATE [11]	RESERVED
49	DATA_BLOCK [44]	TTY_RATE [12]	RESERVED
50	DATA_BLOCK [45]	TTY_RATE [13]	RESERVED
51	DATA_BLOCK [46]	TTY_RATE [14]	RESERVED
52	DATA_BLOCK [47]	TTY_RATE [15]	RESERVED
53	DATA_BLOCK [48]	RESERVED	RESERVED
54	DATA_BLOCK [49]	RESERVED	RESERVED
55	DATA_BLOCK[50]	RESERVED	RESERVED
56	DATA_BLOCK [51]	RESERVED	RESERVED
57	DATA_BLOCK [52]	RESERVED	RESERVED
58	DATA_BLOCK [53]	RESERVED	RESERVED
59	DATA_BLOCK [54]	RESERVED	RESERVED
60	DATA_BLOCK [55]	RESERVED	RESERVED
61	DATA_BLOCK [56]	RESERVED	RESERVED
62	DATA_BLOCK [57]	RESERVED	RESERVED
63	DATA_BLOCK [58]	RESERVED	RESERVED
64	DATA_BLOCK [59]	RESERVED	RESERVED
65	DATA_BLOCK[60]	RESERVED	RESERVED
66	ILLEGAL_CODEWORD[0] = 1	ILLEGAL_CODEWORD[0] = 1	ILLEGAL_CODEWORD[0] = 1
67	ILLEGAL_CODEWORD[1] = 1	ILLEGAL_CODEWORD[1] = 1	ILLEGAL_CODEWORD[1] = 1
68	ILLEGAL_CODEWORD[2] = 1	ILLEGAL_CODEWORD[2] = 1	ILLEGAL_CODEWORD[2] = 1
69	ILLEGAL_CODEWORD[3] = 1	ILLEGAL_CODEWORD[3] = 1	ILLEGAL_CODEWORD[3] = 1
70	ILLEGAL_CODEWORD[4] = 1	ILLEGAL_CODEWORD[4] = 1	ILLEGAL_CODEWORD[4] = 1

71	ILLEGAL_CODEWORD[5] = 1	ILLEGAL_CODEWORD[5] = 1	ILLEGAL_CODEWORD[5] = 1
72	ILLEGAL_CODEWORD[6] = 1	ILLEGAL_CODEWORD[6] = 1	ILLEGAL_CODEWORD[6] = 1
73	ILLEGAL_CODEWORD[7] = 1	ILLEGAL_CODEWORD[7] = 1	ILLEGAL_CODEWORD[7] = 1
74	ILLEGAL_CODEWORD[8] = 1	ILLEGAL_CODEWORD[8] = 1	ILLEGAL_CODEWORD[8] = 1
75	ILLEGAL_CODEWORD[9] = 0	ILLEGAL_CODEWORD[9] = 0	ILLEGAL_CODEWORD[9] = 0
76	ILLEGAL_CODEWORD[10] = 0	ILLEGAL_CODEWORD[10] = 0	ILLEGAL_CODEWORD[10] = 0
77	ILLEGAL_CODEWORD[11] = 1	ILLEGAL_CODEWORD[11] = 1	ILLEGAL_CODEWORD[11] = 1
78	ILLEGAL_CODEWORD[12] = 0	ILLEGAL_CODEWORD[12] = 0	ILLEGAL_CODEWORD[12] = 0
79	ILLEGAL_CODEWORD[13] = 0	ILLEGAL_CODEWORD[13] = 0	ILLEGAL_CODEWORD[13] = 0
80	ILLEGAL_CODEWORD[14] = 0	ILLEGAL_CODEWORD[14] = 0	ILLEGAL_CODEWORD[14] = 0

Table 4.3-7: Data Bit Streams for Rate 1/2

4.4 SMV Symbol Table

Description	Symbol	Comments
Half-rate max flag	F_{HRM}	
Selected SMV mode	M_{SMV}	
The frame counter	frm	
Size of encoding frame	L_{frm}	$L_{frm} = 160$
Size of subframe	L_{SF}	Subframe size depends on the coding rate and the frame type
Input speech	$s(n)$	
Output of silence enhancement	$s'(n)$	
Output of high-pass filter	$s''(n)$	

Output of noise suppression	$s'''(n)$	
Pre-processed speech	$\tilde{s}(n)$	
Size of LPC window	L_m^{LPC}	$m = 0,1,2$
LPC window	$W_m^{LPC}(n)$	$m = 0,1,2$
LPC Autocorrelation function	$R_m(i)$	$m = 0,1,2$
Order of autocorrelation function	O_R	10 for VAD-A, 16 for VAD-B
Prediction coefficients	$a_m(i)$	$m = 0,1,2$
Line spectral frequencies - LSF	$lsf_m(i)$	$m = 0,1,2$
Reflection coefficients	$k_m(i)$	$m = 0,1,2$
LPC prediction error	PE_m^{LPC}	$m = 0,1,2$
LPC prediction gain	G_m^{LPC}	$m = 0,1,2$
Interpolated LPC prediction gain	$\hat{G}_m^{LPC}$	$m = 0,1,2,3,4$
Interpolated reflection coefficients	$\hat{k}_m(i)$	$m = 0,1,2,3,4$
Interpolated prediction coefficients	$\hat{a}_m(i)$	$m = 0,1,2,3$ for Rate 1, 1/4, and 1/8 $m = 0,1$ for Rate 1/2 type 0 $m = 0,1,2$ for Rate 1/2 type 1
Quantized and interpolated prediction coefficients	$\hat{a}_m^q(i)$	$m = 0,1,2,3$ for Rate 1, 1/4, and 1/8 $m = 0,1$ for Rate 1/2 type 0 $m = 0,1,2$ for Rate 1/2 type 1
Buffer of previous open-loop pitch lags	$l_p^B(i)$	
Buffer of previous unquantized pitch gains	$g_p^B(i)$	
Buffer of previous normalized pitch correlation	$corr_p^B(i)$	
Voice activity decision	VAD	
Previous frame voice activity decision	VAD^{prv}	
Music detection flag	F_{MUS}	

Perceptual filter numerator weight	w_m^{num}	$m = 0,1,2,3,4$
Perceptual filter denominator weight	w_m^{den}	$m = 0,1,2,3,4$
Weighing filter numerator coefficients	$\hat{a}_m^{num}(i)$	$m = 0,1,2,3,4$
Weighing filter denominator coefficients	$\hat{a}_m^{den}(i)$	$m = 0,1,2,3,4$
Adaptive low-pass filter parameters	μ_m	$m = 0,1,2,3,4$
Weighted residual	$r_w(n)$	
Voiced/Unvoiced level	C_{VUV}	
Weighted residual half-frame energy	$E_{wr}(m)$	$m = 0,1$
Sharpness of weighted residual	S_{rp}^{wr}	
Weighted speech	$s_w(n)$	
Open-loop pitch lag	$l_p^{ol}(m)$	$m = 0,1,2$
Open-loop normalized correlation	$R_p^{ol}(m)$	$m = 0,1,2$
Frame classifier	C_{FR}	
Previous frame classifier	C_{FR}^{prv}	
Noise to signal ratio	NSR	
Auxiliary class decision	C_{SM}	
Onset flag	F_{onset}	
LPC onset flag	F_{LPC_onset}	
Noisy-voiced flag	F_{noisyV}	
SMV rate for the frame	$Rate$	
SMV rate for the previous frame	$Rate^{prv}$	
Low boundary of open-loop pitch refinement at the end of the frame	l_p^{low}	
High boundary of open-loop pitch refinement at the end of the frame	l_p^{high}	
Signal modification accumulated delay	Δ_{SM}	
Quantized open-loop pitch at the end of the encoding frame	l_{pp}^{q}	

Integer part of quantized open-loop pitch at the end of the encoding frame	$l_{pp,INT}^{q}$	
Frame type	$Type$	
Pitch contour	$Pit(n)$	
Modified weighted speech	$s_{mw}(n)$	
Subframe pitch gains (unquantized or quantized)	$g_a(m)$	
Subframe integer open-loop pitch	$l_p^{SF}(m)$	
Modified speech	$s_{\text{mod}}(n)$	
LSFs smoothing factor	β_{SMO}^{LSF}	
Smoothed LSFs for the encoding frame	$\underline{lsf}_1(i)$	
Smoothed LSFs for the previous encoding frame	$\underline{lsf}_{1,prv}(i)$	
Quantized LSFs for the frame	$lsf_1^q(i)$	
Interpolated LSFs for each subframe	$\hat{lsf}_m(i)$	$m = 0,1,2,3$ for Rate 1, 1/4, and 1/8 $m = 0,1$ for Rate 1/2 type 0 $m = 0,1,2$ for Rate 1/2 type 1
Quantized interpolated LSFs for each subframe	$\hat{lsf}_m^q(i)$	$m = 0,1,2,3$ for Rate 1, 1/4, and 1/8 $m = 0,1$ for Rate 1/2 type 0 $m = 0,1,2$ for Rate 1/2 type 1
Quantized adaptive-codebook gain for the last subframe of previous frame	g_a^{prv}	
Spectral flatness flag	F_{flat}	
LPC prediction gain from quantized parameters of the first subframe	$G_{0,q}^{LPC}$	
Random number seed	$seed$	
Random excitation (Rate 1/8)	$r(n)$	
Shaped random excitation (Rate 1/4)	$r_q(n)$	

Impulse response of combined synthesis and weighting filter	$h_{imp}(n)$	
Modified residual	$r_{\mathrm{mod}}(n)$	
Target for closed-loop pitch search / pitch interpolation	$T_{gs}(n)$	
Adaptive codebook	$C^{adp}(n)$	
Ideal excitation	$I_{ext}(n)$	
Adaptive-codebook contribution (vector)	$c_{unf}^{adp}(n)$	
Filtered adaptive-codebook contribution	$c_{f}^{adp}(n)$	
Fixed-codebook contribution	$c_{unf}^{fix}(n)$	
Filtered fixed-codebook contribution	$c_{f}^{fix}(n)$	
Normalized closed-loop pitch correlation	R_{p}^{wgt}	
Fractional pitch for subframe	l_{FRC}^{p}	
Fractional pitch for previous subframe	$l_{FRC}^{p,prv}$	
Integer pitch for subframe	l_{INT}^{p}	
Normalized closed-loop pitch correlation	R_{p}^{wgt}	
Subframe mode factor	C_{sub_mod}	
Gain normalization factor	β_E	
Modified pitch gain	$\hat{g}_a$	
(Possibly) modified impulse response of combined synthesis and weighting filter	$\hat{h}_{imp}(n)$	

1
2
3
4
5

5 SMV ENCODER

5.1 SMV Encoder Look Ahead Buffers and Delay

The SMV encoder buffers 160 samples (20 ms) in an encoding frame buffer, and 80 samples (10 ms) in a lookahead buffer. Two options of windowing and buffering schemes are implemented for the SMV noise-suppression module, and are selected in the SMV simulation code by option A and option B.

If the noise-suppression simulation code is compiled with option A, an additional 24 samples are used for the noise suppression overlap-and-add procedure. In this case, the total fixed buffering delay at the encoder is 264 samples, or 33 ms.

If the noise-suppression simulation code is compiled with option B, no additional samples are used for the noise suppression overlap-and-add procedure. In this case, the total fixed buffering delay at the encoder is 240 samples, or 30 ms.

The encoder buffers are schematically represented in Figure 5.1-1.

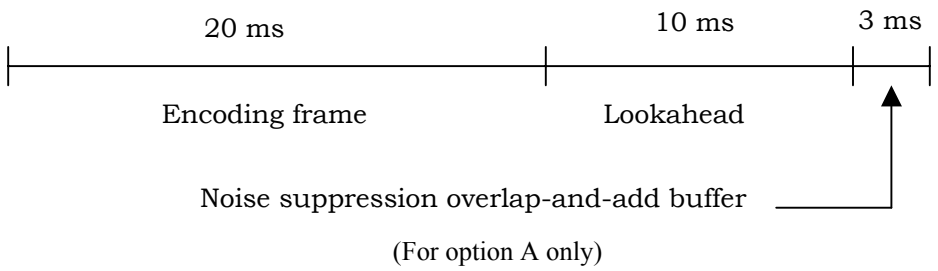

Figure 5.1-1: Selectable Mode Vocoder Encoder Buffering

A speech modification procedure introduces a variable buffering delay, which is limited to 20 samples (2.5 ms) (see Section 5.3.15).

5.2 SMV Encoder Pre-Processing

The speech samples input to the SMV encoder are processed by four pre-processing units. The pre-processing units are silence enhancement, high-pass filtering, noise suppression, and adaptive tilt filtering. The pre-processing flowchart is given in Figure 5.2-1.

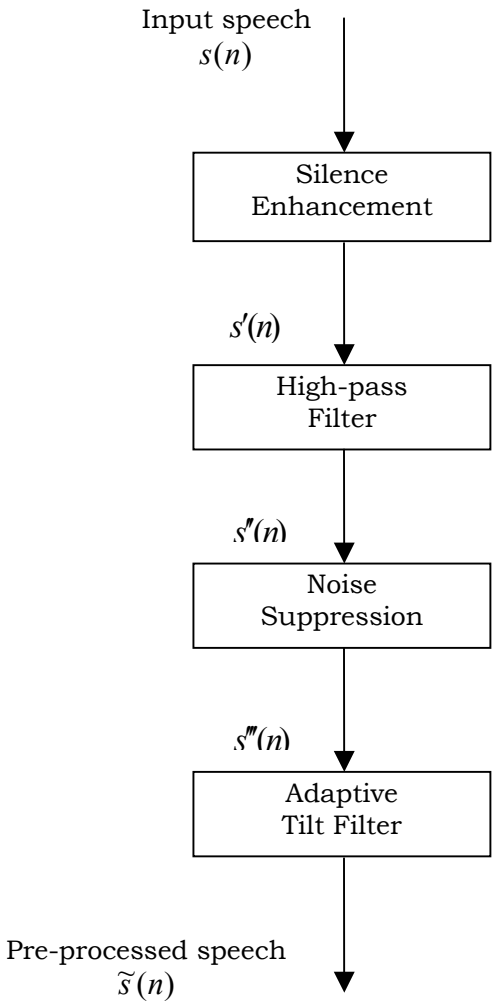

Figure 5.2-1: SMV Pre-Processing

5.2.1 Silence Enhancement

Routine name: `PPR_silence_enhan`

Input:

- Input speech: $s(n)$

Output:

- Silence enhanced speech: $s'(n)$

The input signal is analyzed and modified to minimize low-level noise, typical of A-law and μ-law PCM quantization process.

5.2.1.1 Identifying Minimum and Maximum Values

The two maximum values in the frame and two minimum values in the frame are found by:

$$\begin{aligned} Max_0 &= \{s(n_0) : s(n_0) \geq s(n),\ n = 0,\ldots,L_{frm}-1\} \\ Max_1 &= \{s(n_1) : s(n_1) \geq s(n),\ n = 0,\ldots,L_{frm}-1,\ n_1 \neq n_0\}, \end{aligned} \quad (5.2.1.1\text{-}1)$$

$$\begin{aligned} Min_0 &= \{s(n_0) : s(n_0) \leq s(n),\ n = 0,\ldots,L_{frm}-1\} \\ Min_1 &= \{s(n_1) : s(n_1) \leq s(n),\ n = 0,\ldots,L_{frm}-1,\ n_1 \neq n_0\}. \end{aligned} \quad (5.2.1.1\text{-}2)$$

Two smallest positive values in the frame and two largest negative values in the frame are found by:

$$MinPos_0 = \{s(n_0) : s(n_0) \leq s(n),\ n = 0,\ldots,L_{frm}-1, s(n_0) \geq 0\}, \quad (5.2.1.1\text{-}3)$$

$$MinPos_1 = \{s(n_1) : s(n_1) \leq s(n),\ n = 0,\ldots,L_{frm}-1, s(n_1) \geq 0, n_1 \neq n_0\}, \quad (5.2.1.1\text{-}4)$$

$$MaxNeg_0 = \{s(n_0) : s(n_0) \geq s(n),\ n = 0,\ldots,L_{frm}-1, s(n_0) < 0\}, \quad (5.2.1.1\text{-}5)$$

$$MaxNeg_1 = \{s(n_1) : s(n_1) \geq s(n),\ n = 0,\ldots,L_{frm}-1, s(n_1) < 0, n_1 \neq n_0\}. \quad (5.2.1.1\text{-}6)$$

A level that represents the zero value is initialized to $V_{ZERO} = 8$. If $MinPos_0 < V_{ZERO}$, the value of V_{ZERO} is set to $MinPos_0$.

5.2.1.2 Setting Histogram Thresholds

The four threshold values are initialized to $Th^2_{neg} = -24$, $Th^1_{neg} = -8$, $Th^1_{pos} = 8$, and $Th^2_{pos} = 24$. The difference between the maximum value in the frame and the minimum value in the frame is:

$$\Delta_{lev} = Max_0 - Min_0. \quad (5.2.1.2\text{-}1)$$

If $\Delta_{lev} < \Delta^{\min}_{lev}$ and $\Delta_{lev} \neq 0$

$$\Delta_{lev}^{\min} = \Delta_{lev} . \tag{5.2.1.2-2}$$

If $\Delta_{lev} < \Delta_{lev}^{\min}$, $\Delta_{lev} \neq 0$ and $\Delta_{lev}^{\min} \leq 48$, the threshold values for the histogram are updated according to:

$$\begin{aligned}
&\text{if } (Max_1 \geq 0 \text{ and } Max_0 > 0) \{ \\
&\quad Th_{pos}^1 = Max_1 \\
&\quad Th_{pos}^2 = Max_0 \\
&\} \\
&\text{else} \{ \\
&\quad \text{if } (MinPos_0 < 32767) \\
&\quad\quad Th_{pos}^1 = MinPos_0 \\
&\quad \text{if } (MinPos_1 < 32767) \\
&\quad\quad Th_{pos}^2 = MinPos_1 \\
&\}
\end{aligned} \tag{5.2.1.2-3}$$

$$\begin{aligned}
&\text{if } (Min_1 < 0 \text{ and } Min_0 < 0) \{ \\
&\quad Th_{neg}^1 = Min_1 \\
&\quad Th_{neg}^2 = Min_0 \\
&\} \\
&\text{else} \{ \\
&\quad \text{if } (MaxNeg_0 > -32766) \\
&\quad\quad Th_{neg}^1 = MaxNeg_0 \\
&\quad \text{if } (MaxNeg_1 > -32766) \\
&\quad\quad Th_{neg}^2 = MaxNeg_1 \\
&\}
\end{aligned} \tag{5.2.1.2-4}$$

5.2.1.3 Calculating the 3-Value Histogram

A 3-value histogram is calculated, where H_1^{SE} counts the number of samples in the frame on the interval $\left[Th_{neg}^1, 0\right)$, H_2^{SE} counts the number of samples in the frame on the interval $\left[0, Th_{pos}^1\right]$, and H_3^{SE} counts the number of samples in the frame on the combined interval $\left[-32768, Th_{neg}^2\right) \cup \left(Th_{pos}^1, 32767\right]$. Three population ratios for the frame are calculated according to:

$$PR_{ZERO}^0 = \frac{H_1^{SE}}{L_{frm}} , \tag{5.2.1.3-1}$$

$$PR_{LOW}^{0} = \frac{H_1^{SE} + H_2^{SE}}{L_{frm}}, \quad (5.2.1.3\text{-}2)$$

$$PR_{HIGH}^{0} = \frac{H_3^{SE}}{L_{frm}}. \quad (5.2.1.3\text{-}3)$$

The population ratios for the current frame and for the last 3 frames, as well as the previous frame 'silencing' decision, are used to generate the current frame 'silencing' decision. If the current frame is 'silenced', but the previous frame was not 'silenced', the samples in the current frame are given by:

$$s'(n) = \begin{cases} \frac{(39-n)\cdot s(n) + n \cdot V_{ZERO}}{40} & ;0 \le n < 40 \\ V_{ZERO} & ;40 < n < L_{frm} \end{cases}. \quad (5.2.1.3\text{-}4)$$

If the current frame is 'silenced' and the previous frame was also 'silenced', all of the values in the frame are set to V_{ZERO}. The samples of the current frame are not changed if the current frame is not "silenced" and the previous frame was not "silenced". If the current frame is not 'silenced' but previous frame was 'silenced', the samples in the current frame are given by:

$$s'(n) = \begin{cases} \frac{(39-n)\cdot V_{ZERO} + n \cdot s(n)}{40} & ;0 \le n < 40 \\ s(n) & ;40 < n < L_{frm} \end{cases}. \quad (5.2.1.3\text{-}5)$$

5.2.2 High-Pass Filtering

Routine name: `FLT_filterPZ` (generic pole-zero filter)

Input:

- Silence enhanced speech, $s'(n)$

Output:

- High-pass filtered speech, $s''(n)$

The speech is filtered by a high-pass filter with the transfer function:

$$H(z) = 0.5 \cdot \frac{0.92727435 - 1.8544941 \cdot z^{-1} + 0.92727435 \cdot z^{-2}}{1 - 1.9059465 \cdot z^{-1} + 0.9114024 \cdot z^{-2}}. \quad (5.2.2\text{-}1)$$

The input is scaled down 3 dB by the multiplication factor of 0.5.

5.2.3 Noise Suppression

Routine name: `SNS_modified_noise_suprs`

Input:

- High-pass filtered speech, $s''(n)$
- Previous frame class decision, $C_{FR}^{\Pr v}$

Output:

- Noise suppressed speech, $s'''(n)$

Noise Suppression is used to improve the signal quality that is presented to the Model Parameter Estimator. The procedures by which the noise suppression is implemented is as described in Section 5.2.3. The input to (and the output from) the core noise suppression algorithm can be processed through two different options that are described in the following section. One of the option flags must be set prior to compiling the SMV simulation C code. The difference between the two options generally corresponds to the use of different analysis/synthesis windowing and buffering methodology, resulting in zero algorithmic delay for the noise suppression module when option B is selected.

Processing: Although the frame size of the speech codec is 20 ms, the Noise Suppressor subframe size is 10 ms; therefore, the following procedures shall be executed two times per 20 ms speech frame and the current 10 ms subframe shall be denoted m.

5.2.3.1 Noise Suppression Initialization

The following variables shall be set to zero at initialization ($m = 0$):

- The overlapped portion of the input subframe buffer, $\{d(m)\}$
- The pre-emphasis and de-emphasis memories
- The overlap-and-add buffer, $\{h(n)\}$
- The output buffer history, $s'''(n);\ 0 \le n < L_{frm}$
- The onset counter C_{onset}.

The following shall be initialized to a startup value other than zero:

- The channel energy estimate, $E_{ch}(m)$, (see 5.2.3.3)
- The long-term power spectral estimate, $\overline{E}_{dB}(m)$, (see 5.2.3.6)
- The channel noise estimate, $E_n(m)$, (see 5.2.3.11)

Figure 5.2-2

~~Figure 5.2-2~~ depicts the Noise Suppression system that is described in the following sections.

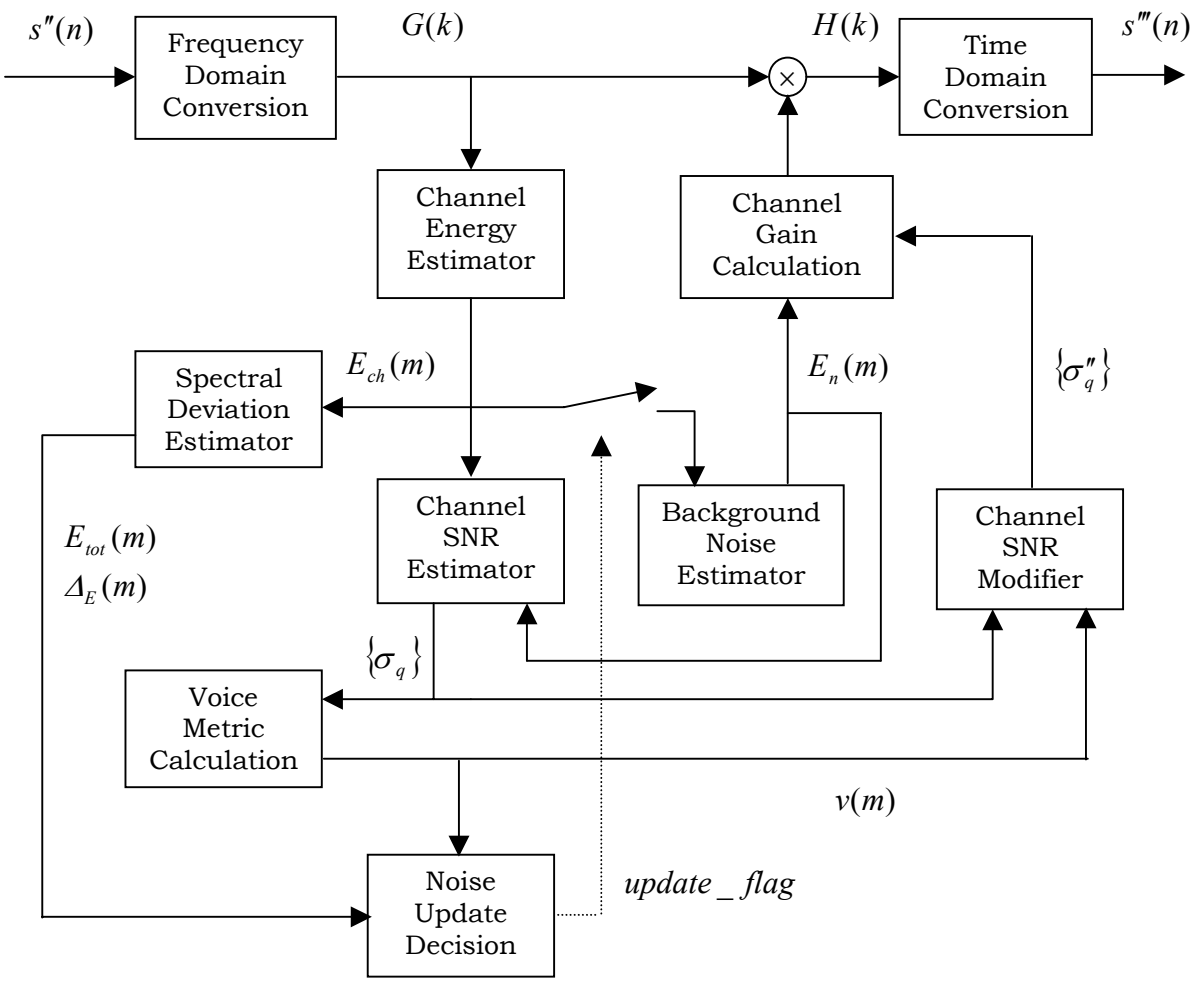

Figure 5.2-2: Noise Suppression Block Diagram

5.2.3.2 Frequency Domain Conversion

5.2.3.2.1 Windowing and Buffering - Option A

In this approach, the input signal is windowed using a smoothed trapezoid window wherein the first D samples of the input subframe buffer $\{d(m)\}$ overlap the last D samples of the previous subframe. This overlap is described as:

$$d(m,n) = d(m-1, L+n) \qquad 0 \le n < D \,, \tag{5.2.3.2-1}$$

where m is the current subframe, n is the sample index to the buffer $\{d(m)\}$, $L = 80$ is the subframe length, and $D = 24$ is the overlap (or delay) in samples.

The remaining samples of the input buffer are then pre-emphasized according to the following:

$$d(m, D+n) = s''(n) + \zeta_p \cdot s''(n-1); \qquad 0 \le n < L \,, \tag{5.2.3.2-2}$$

where $\zeta_p = -0.8$ is the pre-emphasis factor. This results in the input buffer containing $L + D = 104$ samples of which the first D samples are the pre-emphasized overlap from the previous subframe, and the following D samples are pre-emphasized input from the current subframe.

Next, a smoothed trapezoidal window is applied to the input buffer to form the DFT data buffer, $\{g(n)\}$ defined as:

$$g(n) = \begin{cases} d(m,n)\sin^2\left(\pi(n+0.5)/2D\right) & ;0 \le n < D \\ d(m,n) & ;D \le n < L \\ d(m,n)\sin^2\left(\pi(n-L+D+0.5)/2D\right) & ;L \le n < D+L \\ 0 & ;D+L \le n < M \end{cases} , \tag{5.2.3.2-3}$$

where $M = 128$ is the DFT sequence length and all other terms are previously defined.

5.2.3.2.2 Windowing and Buffering - Option B

A tapered trapezoidal window function as shown in Figure 5.2-3 is applied to the input sequence. The analysis and synthesis windows have slopes of D_{ana} = 12 and D_{syn} = 6 samples in length, respectively.

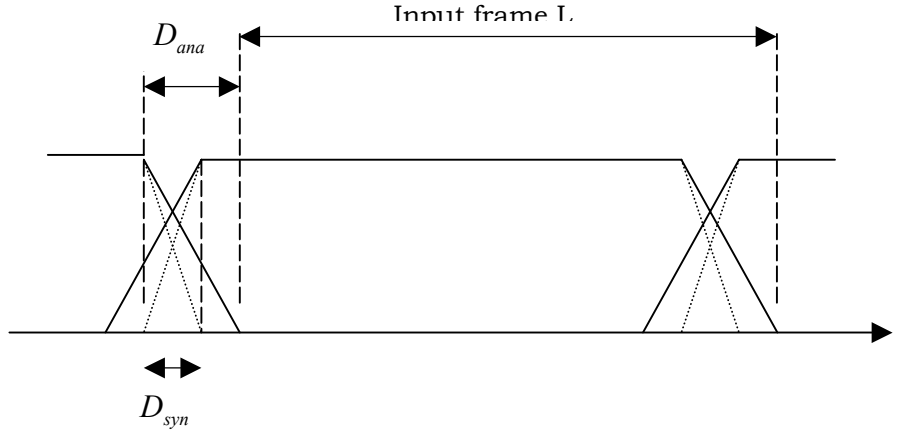

Figure 5.2-3: Two-Phase Trapezoidal Window with Overlap-and-Add. (The analysis window is illustrated using a solid line. The slopes of the synthesis window are shown with a dotted line.)

Similar to option A, the input is overlapped and pre-emphasized according to equations [5.2.3.2-1) and (5.2.3.2-2), respectively. Unlike option A, the constant D is now equal to $2D_{ana} - D_{syn}$. The slope of the analysis window is defined as follows:

$$f_{ana}(n) = (n+1)/(D_{ana}+1); \quad 0 \le n < D_{ana}. \tag{5.2.3.2-4}$$

The trapezoidal analysis window is applied to the pre-emphasized signal.

$$g(n) = \begin{cases} d(m,n) f_{ana}(n) & ;0 \le n < D_{ana} \\ d(m,n) & ; D_{ana} \le n < L + D_{ana} - D_{syn} \\ d(m,n) f_{ana}(L + 2D_{ana} - D_{syn} - n - 1) & ; L + D_{ana} - D_{syn} \le n < L + 2D_{ana} - D_{syn} \\ 0 & ; L + 2D_{ana} - D_{syn} \le n < M \end{cases}, \tag{5.2.3.2-5}$$

where $M = 128$ is the length of the FFT buffer.

The steps in Sections 5.2.3.3 to 5.2.3.11are common in both approaches and are described below.

The transformation of $g(n)$ to the frequency domain is performed using the Discrete Fourier Transform (DFT) defined† as:

† This atypical definition is used to exploit the efficiencies of the complex Fast Fourier Transform (FFT). The $2/M$ scale factor results from preconditioning the M point real sequence to form an $M/2$ point complex sequence that is transformed using an $M/2$ point complex FFT. Details on this technique can be found in Proakis, J. G. and Manolakis, D. G., *Introduction to Digital Signal Processing*, New York, Macmillan, 1988, pp. 721-722.

$$G(k) = \frac{2}{M}\sum_{n=0}^{M-1} g(n)\cdot e^{j2\pi nk/M}\ ; \qquad 0 \le k < M\ . \tag{5.2.3.2-6}$$

where e^{jw} is a unit amplitude complex phasor with instantaneous radial position ω.

5.2.3.3 Channel Energy Estimator

Calculate the channel energy estimate $E_{ch}(m)$ for the current subframe, m, as:

$$E_{ch}(m,i) = \max\left\{ E_{\min}\ , \alpha_{ch}(m)\cdot E_{ch}(m-1,i) + \left(1-\alpha_{ch}(m)\right)\cdot \frac{\sum_{k=f_L(i)}^{f_H(i)} \left|G(k)\right|^2}{f_H(i) - f_L(i) + 1} \right\};$$

$$0 \le i < N_c\ , \tag{5.2.3.3-1}$$

where $E_{\min} = 0.0625$ is the minimum allowable channel energy, $\alpha_{ch}(m)$ is the channel energy smoothing factor (defined below), $N_c = 16$ is the number of combined channels, and $f_L(i)$ and $f_H(i)$ are the i^{th} elements of the respective low and high channel combining tables, which are defined as:

$$\mathbf{f}^L = \{\ 2,\ 4,\ 6,\ 8,\ 10,\ 12,\ 14,\ 17,\ 20,\ 23,\ 27,\ 31,\ 36,\ 42,\ 49,\ 56\ \}, \tag{5.2.3.3-2}$$

$$\mathbf{f}^H = \{\ 3,\ 5,\ 7,\ 9,\ 11,\ 13,\ 16,\ 19,\ 22,\ 26,\ 30,\ 35,\ 41,\ 48,\ 55,\ 63\ \}. \tag{5.2.3.3-3}$$

The channel energy smoothing factor, $\alpha_{ch}(m)$ is defined as:

$$\alpha_{ch}(m) = \begin{cases} 0 & ;m = 0 \\ 0.25 & ;m > 0,\ C_{onset} = 1 \\ 0.45 & ;m > 0,\ C_{onset} \ne 1 \end{cases} \tag{5.2.3.3-4}$$

Therefore, $\alpha_{ch}(m)$ assumes a value of zero for the first subframe $(m = 0)$, the value of 0.25 for onset frames, and 0.45 for all other frames. This allows the channel energy estimate to be initialized to the unfiltered channel energy of the first subframe, and to be adaptively updated for the subsequent subframes.

5.2.3.4 Channel SNR Estimator

Estimate the quantized channel SNR indices as:

$$\sigma_q(i) = \max\left\{0, \min\left\{89, \mathrm{round}\left(10\log_{10}\frac{E_{ch}(m,i)}{E_n(m,i)} \Big/ 0.375\right)\right\}\right\};\ 0 \le i < N_c\ , \tag{5.2.3.4-1}$$

where $E_n(m,i)$ is the current channel noise energy estimate (Section 5.2.3.11), and the values of $\{\sigma_q\}$ are constrained to be between 0 and 89, inclusive.

5.2.3.5 Voice Metric Calculation

Next, calculate the sum of voice metrics as:

$$v(m) = \sum_{i=0}^{N_C-1} V\left(\sigma_q(i)\right), \tag{5.2.3.5-1}$$

where $V(k)$ is the k^{th} value of the 90 element voice metric table **V**, that is defined as:

V = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 7, 7, 7, 8, 8, 9, 9, 10, 10, 10, 11, 12, 12, 13, 13, 14, 15, 15, 16, 17, 17, 18, 19, 20, 20, 21, 22, 23, 24, 24, 25, 26, 27, 28, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50 }.

(5.2.3.5-2)

5.2.3.6 Spectral Deviation Estimator

Calculate the estimated log power spectrum as:

$$E_{dB}(m,i) = 10\log_{10}\left(E_{ch}(m,i)\right); \qquad 0 \le i < N_c \ . \tag{5.2.3.6-1}$$

Then, calculate the estimated spectral deviation between the current power spectrum and the average long-term power spectral estimate:

$$\Delta_E(m) = \sum_{i=0}^{N_c-1} \left| E_{dB}(m,i) - \overline{E}_{dB}(m,i) \right|, \tag{5.2.3.6-2}$$

where $\overline{E}_{dB}(m)$ is the average long-term power spectral estimate calculated during the previous subframe, as defined by Equation (5.2.3.6-7). The initial value of $\overline{E}_{dB}(m)$, however, is defined to be the estimated log power spectrum of subframe 0, or:

$$\overline{E}_{dB}(m) = E_{dB}(0). \tag{5.2.3.6-3}$$

Calculate the total channel energy estimate, $E_{tot}(m)$ for the current subframe, m, according to the following:

$$E_{tot}(m) = 10\log_{10}\left(\sum_{i=0}^{N_c-1} E_{ch}(m,i) \right). \tag{5.2.3.6-4}$$

Calculate the exponential windowing factor, $\alpha(m)$, as a function of total channel energy, $E_{tot}(m)$, as:

$$\alpha(m) = \alpha_H - \left(\frac{\alpha_H - \alpha_L}{E_H - E_L} \right)\left(E_H - E_{tot}(m)\right), \tag{5.2.3.6-5}$$

and then limit the result to be between α_H and α_H by

$$\alpha(m) = \max\{\alpha_L, \min\{\alpha_H, \alpha(m)\}\}, \quad \text{(5.2.3.6-6)}$$

where E_H and E_L are the energy endpoints (in dB) for the linear interpolation of $E_{tot}(m)$, that is transformed to $\alpha(m)$ that has the limits $\alpha_L \le \alpha(m) \le \alpha_H$. The values of these constants are defined as: $E_H = 50$, $E_L = 30$, $\alpha_H = 0.99$, $\alpha_L = 0.50$. As an example, a signal with relative energy of 40 dB would use an exponential windowing factor of $\alpha(m) = 0.745$ for the following calculation.

Update the average long-term power spectral estimate for the next subframe by:

$$\overline{E}_{dB}(m+1,i) = \alpha(m) \cdot \overline{E}_{dB}(m,i) + (1-\alpha(m)) \cdot E_{dB}(m,i);$$

$$0 \le i < N_c, \quad \text{(5.2.3.6-7)}$$

where all the variables are previously defined.

5.2.3.7 Background Noise Update Decision

The following logic, shown in pseudo-code, demonstrates how the noise estimate update decision is ultimately made:

```
/* Normal update logic */
update_flag = FALSE
if ( v(m) ≤ UPDATE_THLD ) {
        update_flag = TRUE
        update_cnt = 0
}
/* Forced update logic */
else if (( E_tot(m) > NOISE_FLOOR_DB ) and ( Δ_E(m) < DEV_THLD )) {
        update_cnt = update_cnt + 1
        if ( update_cnt ≥ UPDATE_CNT_THLD )
                update_flag = TRUE
}
/* "Hysteresis" logic to prevent long-term creeping of update_cnt */
if ( update_cnt == last_update_cnt )
        hyster_cnt = hyster_cnt + 1
else
        hyster_cnt = 0
last_update_cnt = update_cnt
if ( hyster_cnt > HYSTER_CNT_THLD )
        update_cnt = 0
```

The values of the previously used constants are UPDATE_THLD = 35, NOISE_FLOOR_DB = $10\log_{10}(E_{floor})$ (see Section 5.2.3.9), DEV_THLD = 28, UPDATE_CNT_THLD = 50, HYSTER_CNT_THLD = 6.

5.2.3.8 SNR Estimate Modification

Next, determine whether the channel SNR modification should take place, then proceed to modify the appropriate SNR indices:

```
/* Set or reset modify flag */
index_cnt = 0
for ( i = N_M  to  N_c - 1 step 1 ) {
        if ( σ_q(i) ≥ INDEX_THLD )
                index_cnt = index_cnt + 1
}
if ( index_cnt < INDEX_CNT_THLD )
        modify_flag = TRUE
else
        modify_flag = FALSE

/* Modify the SNR indices to get {σ'_q} */
if ( modify_flag == TRUE ) {
        for ( i = 0 to  N_c - 1 step 1 )
                if (( v(m) ≤ METRIC_THLD ) or (σ_q(i) ≤ SETBACK_THLD ))
                        σ'_q(i) = 1
                else
                        σ'_q(i) = σ_q(i)
}
else  {
        {σ'_q} = {σ_q}
}

/* Limit {σ''_q} to SNR threshold σ_th */
for ( i = 0 to  N_c - 1 step 1 )
        if ( σ'_q(i) < σ_th )
                σ''_q(i) = σ_th
        else
                σ''_q(i) = σ'_q(i)
```

The previous constants and thresholds are given to be: N_M = 5, INDEX_THLD = 12, INDEX_CNT_THLD = 5, METRIC_THLD = 45, SETBACK_THLD = 12, σ_{th} = 6.

5.2.3.9 Channel Gain Computation

Calculate the sum of the estimated noise spectrum $E_n(m)$ (see Section 5.2.3.7). Reduce the sum by a constant factor, and limit it to a positive $\varepsilon = 0.00001$. The result, T_{ne}, is given by:

$$T_{ne} = \max\left(\sum_{i=0}^{N_c-1} E_n(m,i) - 12.0412\,,\, 0.00001\right). \quad (5.2.3.9\text{-}1)$$

Compute the overall gain factor for the current subframe as:

$$\gamma_n = \max\left(\gamma_{\min}\,, -10\cdot\log\left(\frac{T_{ne}}{E_{floor}}\right)\right), \quad (5.2.3.9\text{-}2)$$

where $E_{floor} = 1$ is the noise floor energy, and the minimum overall gain $\gamma_{\min}$ = -13 dB if $F_{flat} = 0$, and $\gamma_{\min}$ = -14 dB if $F_{flat} = 1$. Next, calculate channel gains (in dB) as:

$$\gamma_{dB}^{i}(m) = \min\left(\left(\sigma_q''(i) - \sigma_{th}\right)\cdot\mu_g\cdot\left(1 - \frac{i}{4N_c}\right) + \gamma_n, 0\right) \quad ; 0 \le i < N_c\,, \quad (5.2.3.9\text{-}3)$$

where $\mu_g = 0.39$ is the gain slope, i is the channel index, and m is the subframe number. Calculate the average gain for the low spectrum and for the high spectrum by:

$$\gamma_L(m) = \frac{1}{N_M + 1}\sum_{i=0}^{N_M}\gamma_{dB}(i) \;\; ; \;\; \gamma_H(m) = \frac{1}{N_c - N_M}\sum_{i=N_M+1}^{N_c}\gamma_{dB}(i)\,, \quad (5.2.3.9\text{-}4)$$

where m is the subframe number. C_{onset} is calculated as a function of previous frame speech class $C_{FR}^{\Pr v}$, and is set to 1 if the previous frame was not an onset or voiced (stationary or non-stationary), and if one of the following conditions is met: i) $\gamma_L(m) - \gamma_L(m-1) > 4$, ii) $\gamma_L(m) - \gamma_L(m-1) > 2$ and $\gamma_L(m) - \gamma_H(m) > -3$.

If $C_{onset} \le 3$ it is incremented, and if $C_{onset} > 3$ it is set to 0. Based on the *modify_flag,* the previous frame class, $v(m)$, and C_{onset}, a voicing decision is made. If the voicing decision is TRUE, correct the channel gains for the channels indexed lower than 9 according to:

$$y_{dB}^{i}(m) = \max\left\{\left(0.25 + 0.75\cdot i/8\right)\cdot y_{dB}^{i}(m) + \left(0.75 - 0.75\cdot i/8\right)\cdot\max\left\{\gamma_L(m), \gamma_H(m)\right\}, y_{dB}^{i}(m)\right\}. \quad (5.2.3.9\text{-}5)$$

Further, if $C_{onset} \ne 1$, smooth the channel gains for the channels indexed lower than 9 are according to:

$$\gamma_{dB}^{i}(m) = \min\left(0.75\cdot\gamma_{dB}^{i}(m) + 0.25\cdot\gamma_{dB}^{i}(m-1)\,, \gamma_{dB}^{i}(m)\right), \quad (5.2.3.9\text{-}6)$$

and then convert the channel gains in dB to linear channel gains:

$$\gamma_{ch}(i) = 10^{\gamma_{dB}^{i}(m)/20} \quad ; \; 0 \le i < N_C\,. \quad (5.2.3.9\text{-}7)$$

5.2.3.10 Frequency Domain Filtering

Now, apply the channel gains to the transformed input signal $G(k)$:

$$H(k) = \begin{cases} \gamma_{ch}(i) \cdot G(k) & ; f_L(i) \le k \le f_H(i) \ , \ 0 \le i < N_c \\ G(k) & ; 0 \le k < f_L(0) \ , \ f_H(N_c - 1) < k \le M/2 \end{cases} . \qquad (5.2.3.10\text{-}1)$$

where the bottom part of the equation represents the frequencies that are not altered by the channel gains. Since it is also required that the magnitude of $H(k)$ be even, and the phase be odd, the following condition is also imposed:

$$H(M-k) = H^*(k) , \qquad 0 < k < M/2 , \qquad (5.2.3.10\text{-}2)$$

where * denotes complex conjugate. This guarantees that the imaginary part of the inverse DFT of $H(k)$ will be zero (in Equation (5.2.3.12-1)).

5.2.3.11 Background Noise Estimate Update

If (and only if) the update flag is set (*update_flag* = TRUE), then update the channel noise estimate for the next subframe by:

$$E_n(m+1,i) = \max\{E_{\min}, \alpha_n \cdot E_n(m,i)\} + (1-\alpha_n) \cdot E_{ch}(m,i) ; \qquad 0 \le i < N_c ,$$

(5.2.3.11-1)

where $E_{\min} = 0.0625$ is the minimum allowable channel energy, and $\alpha_n = 0.9$ is the channel noise smoothing factor. The channel noise estimate shall be initialized for each of the first four subframes to the estimated channel energy, i.e.:

$$E_n(m,i) = \max\{E_{init}, E_{ch}(m,i)\} \ ; \ 0 \le m \le 3 \ , \ 0 \le i < N_c , \qquad (5.2.3.11\text{-}2)$$

where $E_{init} = 16$ is the minimum allowable channel noise initialization energy.

5.2.3.12 Time Domain Signal Reconstruction

Convert the filtered signal to the time domain using the inverse DFT:

$$h(m,n) = \frac{1}{2} \sum_{k=0}^{M-1} H(k) \cdot e^{j 2\pi nk/M} \ ; \qquad 0 \le n < M . \qquad (5.2.3.12\text{-}1)$$

5.2.3.12.1 Overlap-and-Add – Option A

Complete the frequency domain filtering process by applying overlap-and-add:

$$h'(n) = \begin{cases} h(m,n) + h(m-1,n+L) & ; 0 \le n < M-L \\ h(m,n) & ; M-L \le n < L \end{cases} . \qquad (5.2.3.12\text{-}2)$$

Finally, apply signal de-emphasis by:

$$s'''(n) = h'(n) + \zeta_d \cdot s'''(n-1) ; \qquad 0 \le n < L , \qquad (5.2.3.12\text{-}3)$$

where $\zeta_d = 0.8$ is the de-emphasis factor and $\{s'''(n)\}$ is the output buffer.

5.2.3.12.2Overlap-and-Add – Option B

The slope of the synthesis window is defined as

$$f_{syn}(n) = (n+1)/(D_{syn}+1) \quad ;0 \le n < D_{syn} \,. \tag{5.2.3.12-4}$$

The output overlap-add is calculated as follows:

$$h'(m,n) = \begin{cases} h'(m-1, n+L+D_{ana}-D_{syn}) f_{syn}(D_{syn}-n-1) + \\ h(m, n+D_{ana}-D_{syn}) f_{syn}(n) / f_{ana}(n+D_{ana}-D_{syn}) & ; 0 \le n < D_{syn} \\ h(m, n+D_{ana}-D_{syn}) & ; D_{syn} \le n < L \\ h(m, n+D_{ana}-D_{syn}) / f_{ana}(L+D_{ana}-n-1) & ; L \le n < L+D_{ana} \end{cases} . \tag{5.2.3.12-5}$$

Finally, the output signal is de-emphasized as follows:

$$s'''(n) = h'(m, n+D_{ana}) + \zeta_d s'''(n-1); \qquad -D_{ana} \le n < L \,, \tag{5.2.3.12-6}$$

where $\zeta_d = 0.8$ is the de-emphasis factor and $\{s'''(n)\}$ is the output buffer.

Since a 10 ms subframe Noise Suppression is performed twice per 20 ms speech frame, the output presented to the adaptive tilt filter is a 20 ms frame $s'''(n)$.

5.2.4 Adaptive Tilt Filtering

Routine name: `PPR_lowpass`

Input:

- Noise suppressed speech, $s'''(n)$
- Spectral flatness flag, F_{flat}
- SMV mode of operation, M_{SMV}
- Half-rate-max flag, F_{HRM}

Output:

- Pre-processed speech, $\tilde{s}(n)$

The speech is filtered by an adaptive tilt filter with the transfer function:

$$T(z) = 1 + \lambda \cdot z^{-1}. \qquad (5.2.4\text{-}1)$$

The parameter λ depends on the SMV operating mode, the spectral flatness flag, and on the half-rate-max flag, as given by Table 5.2-1.

	SMV Mode 0	SMV Mode 1	SMV Mode 2	SMV Mode 3
$F_{flat} = 1$, $F_{HRM} = 1$	0.0	0.0	0.0	0.0
$F_{flat} = 1$, $F_{HRM} = 0$	-0.1	0.0	0.0	0.0
$F_{flat} = 0$, $F_{HRM} = 1$	0.1	0.1	0.1	0.1
$F_{flat} = 0$, $F_{HRM} = 0$	0.0	0.05	0.075	0.1

Table 5.2-1: Input Spectral Tilt Coefficient Table

5.3 SMV Encoder Frame Level Processing

The frame processing in the SMV encoder includes linear prediction (LP) analysis, LSFs smoothing and quantization. The speech parameters are used for voice activity detection (VAD), frame classification, type selection, and rate determination. Using the perceptually weighted speech, open-loop pitch detection and signal modification is performed. Figure 5.3-1 is a block diagram of the SMV frame level processing.

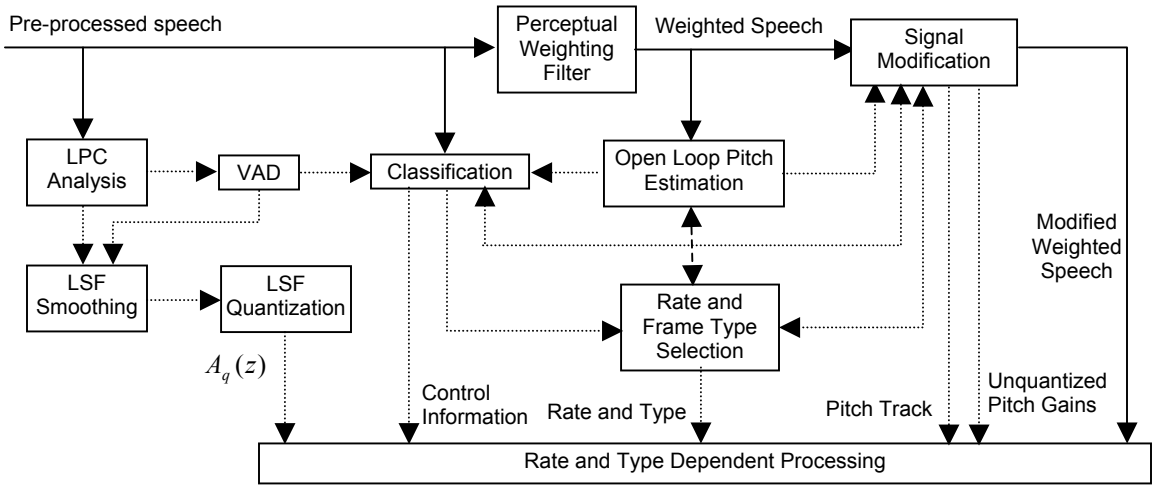

Figure 5.3-1: Signal Flow for Frame Processing in SMV Encoder

Depending on the encoding mode of operation, SMV classifies the input speech frame, and chooses an optimal Rate and Type for frame processing. Figure 5.3-2 illustrates the various stages of classification in the SMV encoder, as described in the following sections. Voiced activity detection is described in Sections 5.3.2 and 5.3.3. Music detection is described in Section 5.3.4. Voiced/unvoiced level detection is described in Section 5.3.7. Active speech classification is described in Section 5.3.10. Class correction is described in Section 5.3.12. The encoding mode dependent rate selection is described in Section 5.3.13. Voiced speech classification is described in Section 5.3.15.3. The final rate and class corrections are described in Section 5.3.16.

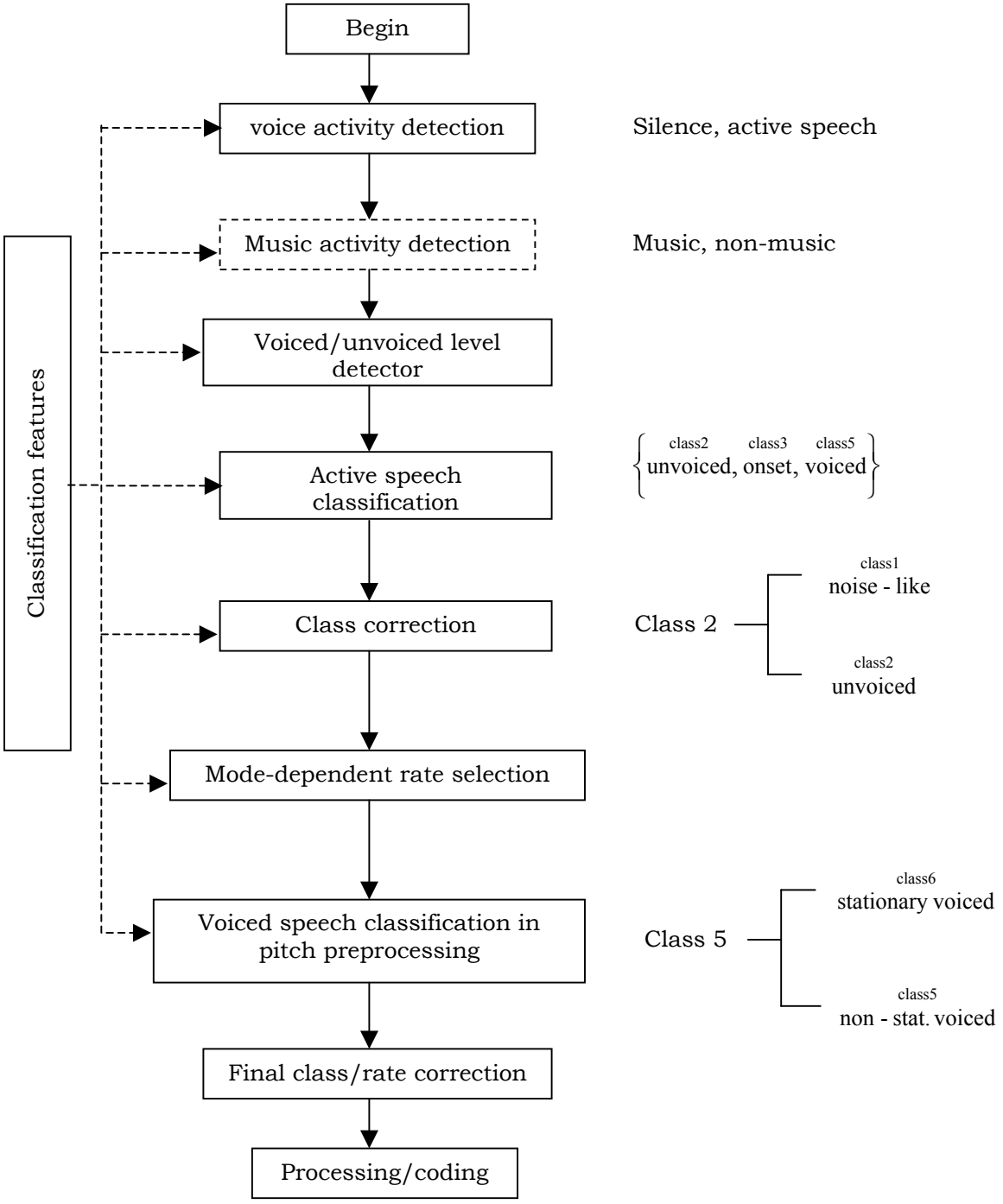

Figure 5.3-2: Stages of Classification in SMV Encoder

5.3.1 Linear Prediction (LPC) Analysis

The SMV encoder performs the LPC analysis three times for each frame. Each LPC analysis is performed with a different weighting window, which is centered on a different section of the frame or the lookahead.

The first LPC analysis is performed with the window centered on the second quarter of the encoding frame. The LPC parameters from this LPC analysis are used for generating the weighted speech, for the frame classification, and for determining the LSFs interpolation path for Rate 1 Type-0 frames. The second LPC analysis is performed with the window centered on the last quarter of the encoding frame. The LPC parameters from the second LPC analysis are smoothed (Section 5.3.21) and quantized (Section 5.3.22.3). The third LPC analysis is performed with the window centered on the lookahead buffer, and the LPC parameters are used for generating the weighted speech for the lookahead buffer.

5.3.1.1 The LPC Analysis Weighting Windows

Each of the three LPC analyses uses a different weighting window, but all weighting windows have the same size: $L_{LPC} = 240$. The first LPC weighting window, $W_0^{LPC}(n)$, is a symmetric Hamming window of size 240. The second LPC weighting window, $W_1^{LPC}(n)$, is formed by the concatenation of the rising half of a Hamming window of size 150 with the falling half of a Hamming window of size 90. The third LPC weighting window, $W_2^{LPC}(n)$, is formed by the concatenation of the rising half of a Hamming window of size 200 with the falling half of a raised-cosine window of size 40.

5.3.1.2 The LP Analysis Procedure

Routine name: `LPC_analysis`

Input (for the m^{th} LPC analysis, $m = 0,1,2$)**:**

- Pre-processed speech, $\tilde{s}(n)$
- Spectral flatness flag, F_{flat}
- LPC weighting window, $W_m^{LPC}(n)$
- The autocorrelation order, O_R, (10 for VAD-A, 16 for VAD-B)

Output (for the m^{th} LPC analysis, $m = 0,1,2$)**:**

- Autocorrelation function, $R_m(i)$
- Reflection coefficients, $k_m(i)$
- Prediction coefficient, $a_m(i)$
- LSFs, $lsf_m(i)$

- LPC prediction error, PE_m^{LPC}
- LPC prediction gain, G_m^{LPC}

5.3.1.2.1 Energy Calculation

The segment energy is calculated as

$$E_{LPC} = \sum_{n=0}^{L_{LPC}-1} \tilde{s}^2[n]. \qquad (5.3.1.2\text{-}1)$$

If $E_{LPC} = 0$ the LPC analysis procedure returns zero values for the autocorrelation, the reflection coefficients, the prediction coefficients, and the LP prediction error. The LP prediction gain (the prediction error in dB) is set to –DBL_MAX. The LSFs are set to equally spaced values. If $E_{LPC} \neq 0$, the LPC analysis proceeds to the next steps, described in Sections 5.3.1.2.2 to 5.3.1.2.5.

5.3.1.2.2 Calculation of the Autocorrelation Function

The signal is multiplied by the LPC window (m indicates the index of the LPC analysis):

$$s_w^{LPC}(n) = \tilde{s}(n) \cdot W_m^{LPC}(n) \quad ; \quad 0 \le \mathrm{n} \le L_{LPC} - 1. \qquad (5.3.1.2\text{-}2)$$

The autocorrelation function is calculated according to:

$$R_m(i) = \sum_{n=0}^{L_{LPC}-i-1} s_w^{LPC}(n) \cdot s_w^{LPC}(n+i) \quad ; \; 0 \le i \le O_R. \qquad (5.3.1.2\text{-}3)$$

The autocorrelation function is multiplied by a bandwidth expansion factor of 60 Hz:

$$R_m(i) \Leftarrow R_m(i) \cdot \exp\left(-0.5 \cdot \left(\frac{2\pi \cdot i \cdot 60}{F_s} \right)^2 \right) \quad ; \; 1 \le i \le 10, \qquad (5.3.1.2\text{-}4)$$

where $F_s = 8000$ is the sampling frequency in Hz. High frequency compensation is preformed by multiplying the first autocorrelation coefficient by 1.0001, which is equivalent to adding a noise floor of -40 dB.

5.3.1.2.3 Levinson-Durbin Algorithm

The reflection coefficients, the prediction coefficients, and the LP prediction error are calculated by the standard Levinson-Durbin algorithm.[1] The algorithm is given by (the index m was omitted for the local variables of the algorithm):

1 See: Rabiner, L. R. and Schafer, R. W., *Digital Processing of Speech Signals*, (New Jersey: Prentice-Hall Inc, 1978), pp. 411-412.

1
2
3
4

$$
\begin{aligned}
&E^{(0)} = R_m(0) \\
&i = 1 \\
&\text{while } (i < 10) \\
&\{ \\
&\quad k_m(i) = -\frac{R_m(i) + \sum_{j=1}^{i-1} \alpha_j^{(i-1)} \cdot R_m(i-j)}{E^{(i-1)}} \\
&\quad \alpha_i^{(i)} = k_m(i) \\
&\quad j = 1 \\
&\quad \text{while } (j < i-1) \\
&\quad \{ \\
&\quad\quad \alpha_j^{(i)} = \alpha_j^{(i-1)} + k_m(i) \cdot \alpha_{i-j}^{i-1} \\
&\quad\quad j = j + 1 \\
&\quad \} \\
&\quad E^{(i)} = \left(1 - k_m^2(i)\right) E^{(i-1)} \\
&\quad i = i + 1 \\
&\}
\end{aligned}
\qquad . \qquad \text{(5.3.1.2-5)}
$$

5

6 The LPC prediction coefficients for the m^{th} LPC analysis are given by:

7
$$a_m(i) = \alpha_i^{(10)} \qquad ;\ 1 \le i \le 10 \ . \qquad \text{(5.3.1.2-6)}$$
8

9 5.3.1.2.4 Conversion to LSFs

10 The prediction error filter transfer function, $A(z)$, is given by (the index m omitted for
11 clarity):

12
$$A(z) = 1 + \sum_{i=1}^{10} a(i) \cdot z^{-i} \ . \qquad \text{(5.3.1.2-7)}$$

13 Define two new transfer functions $P_A(z)$ and $Q_A(z)$ as

14
$$P_A(z) = A(z) - z^{-11} A(z^{-1}) = 1 + p_1 z^{-1} + \ldots, + p_5 z^{-5} - p_5 z^{-6} - \ldots - p_1 z^{-10} - z^{-11} \ , \qquad \text{(5.3.1.2-8)}$$
15

16 and

$$Q_A(z) = A(z) + z^{-11}A(z^{-1}) = 1 + q_1 z^{-1} + \ldots, + q_5 z^{-5} + q_5 z^{-6} + \ldots + q_1 z^{-10} z^{-11}, \quad (5.3.1.2\text{-}9)$$

where

$$p_i = a_i - a_{11-i} \qquad ; 1 \le i \le 5, \quad (5.3.1.2\text{-}10)$$

and

$$q_i = a_i + a_{11-i} \qquad ; 1 \le i \le 5. \quad (5.3.1.2\text{-}11)$$

The LSF frequencies are the ten roots that exist between $\omega = 0.0$ and $\omega = 1.0$ in the following two equations:

$$P'(\omega) = \cos(5\pi \cdot \omega) + p'_1 \cdot \cos(4\pi \cdot \omega) + \ldots + p'_4 \cdot \cos(\pi \cdot \omega) + \frac{p'_5}{2}, \quad (5.3.1.2\text{-}12)$$

$$Q'(\omega) = \cos(5\pi \cdot \omega) + q'_1 \cdot \cos(4\pi \cdot \omega) + \ldots + q'_4 \cdot \cos(\pi \cdot \omega) + \frac{q'_5}{2}, \quad (5.3.1.2\text{-}13)$$

where the parameters p'_i and q'_i are computed recursively from the parameters p_i and q_i as

$$p'_0 = q'_0 = 1, \quad (5.3.1.2\text{-}14)$$

$$p'_i = p_i + p'_{i-1} \qquad ; 1 \le i \le 5, \quad (5.3.1.2\text{-}15)$$

$$q'_i = p_i - q'_{i-1} \qquad ; 1 \le i \le 5. \quad (5.3.1.2\text{-}16)$$

Since the formant synthesis (LPC) filter is stable, the roots of the two functions alternate in the range from 0.0 to 1.0. If these ten roots are denoted as $[lsf(1), lsf(2), \ldots, lsf(10)]$ in the increasing order of magnitude, then $lsf(i)$ for $i = 1,3,5,7,9$ are roots of $P'(\omega)$ and $lsf(i)$ for $i = 2,4,6,8,10$ are those of $Q'(\omega)$.

The LSFs calculated with the window centered at the second quarter of the encoding frame are denoted $lsf_0(i)$, the LSFs calculated with the window centered at the fourth quarter of the encoding frame are denoted $lsf_1(i)$, and the LSFs calculated with the window centered at the lookahead are denoted $lsf_2(i)$.

5.3.1.2.5 LPC Prediction Error and LPC Prediction Gain

The LPC prediction error is calculated by:

$$PE_m^{LPC} = R_m(0) \cdot \prod_{i=1}^{10} \left(1 - k_m^2(i)\right). \quad (5.3.1.2\text{-}17)$$

Where $m = 0,1,2$ indicates the LPC analysis performed on the second quarter of the encoding frame, the last quarter of the encoding frame, and the lookahead, respectively. The LPC prediction gain is obtained from LPC prediction error by:

$$G_m^{LPC} = -10 \cdot \log_{10}\left(\frac{PE_m^{LPC}}{R_m(0)}\right) \quad ; m = 0,1,2 \,. \tag{5.3.1.2-18}$$

5.3.1.3 Interpolation of LPC Prediction Gains, Reflection Coefficients, and Prediction Coefficients

The encoding frame is divided into 4 subframes. The interpolated LPC prediction gain is obtained for each of the four subframes and the lookahead, indexed $m = 0,1,2,3,4$, by:

$$\begin{aligned}
\hat{G}_0^{LPC} &= 0.5 \cdot \left(G_{1,prv}^{LPC} + G_0^{LPC}\right) \\
\hat{G}_1^{LPC} &= G_0^{LPC} \\
\hat{G}_2^{LPC} &= 0.5 \cdot \left(G_0^{LPC} + G_1^{LPC}\right) \\
\hat{G}_3^{LPC} &= G_1^{LPC} \\
\hat{G}_4^{LPC} &= G_2^{LPC}
\end{aligned} \,. \tag{5.3.1.3-1}$$

Where $G_{1,prv}^{LPC}$ is G_1^{LPC} of the previous frame.

Five vectors of interpolated reflection coefficients are generated from the 3 vectors of reflection coefficients obtained during LPC analysis, by:

$$\begin{aligned}
\hat{k}_0(i) &= 0.5 \cdot \left(k_{1,prv}(i) + k_0(i)\right) \\
\hat{k}_1(i) &= k_0(i) \\
\hat{k}_2(i) &= 0.5 \cdot \left(k_0(i) + k_1(i)\right) \qquad ; i = 1,\ldots,10 \\
\hat{k}_3(i) &= k_1(i) \\
\hat{k}_4(i) &= k_2(i)
\end{aligned} \,. \tag{5.3.1.3-2}$$

where $k_{1,prv}(i)$ is $k_1(i)$ of the previous frame.

Five vectors of prediction coefficients, denoted by $\hat{a}_m(i)$, are generated from the 3 vectors of prediction coefficients obtained during LPC analysis and the 2 vectors of LSFs generated for the second and the fourth quarter of the encoding frame. Similar to the interpolation of the reflection coefficients, $\hat{a}_1(i)$ is the vector of prediction coefficients obtained from the LPC analysis on the second quarter of the encoding frame, $\hat{a}_3(i)$ is the vector of prediction coefficients obtained from the LPC analysis on the fourth quarter of the encoding frame, and $\hat{a}_4(i)$ is the vector of prediction coefficients obtained from the LPC analysis on the lookahead. The interpolated vector $\hat{a}_0(i)$ is obtained by interpolating $lsf_{1,prv}(i)$ and $lsf_0(i)$, where $lsf_{1,prv}(i)$ is $lsf_1(i)$ of the previous frame, and converting the interpolated LSF vector to the vector of prediction coefficients. The interpolated vectors $\hat{a}_2(i)$ obtained by interpolating $lsf_0(i)$ and $lsf_1(i)$, and converting the interpolated LSF vector to the vector of prediction coefficients.

5.3.2 Voice Activity Detection – Option A

Routine name: `VAD_voice_detection`

A voice activity detector (VAD) is used to classify the input signal as either active voice (speech) or inactive voice (silence or background noise). The VAD extracts several parameters from the input signal and makes a voicing decision using a set of thresholds.

Input:

- Pre-processed speech, $\tilde{s}(n)$
- Reflection coefficients from the LPC analysis on the last quarter of the coding frame, $k_1(i)$
- PE_1^{LPC}, the LP prediction error from the LPC analysis on the last quarter of the coding frame
- Buffer of 5 previous open-loop pitch lags, $l_p^B(i) \quad i = 1,\ldots,5$
- Buffer of 5 previous pitch gains (unquantized adaptive-codebook gains), $g_p^B(i) \quad i = 1,\ldots,5$
- Vector of LSFs from the LPC analysis on the last quarter of the coding frame, $lsf_1(i)$
- Autocorrelation function from the LPC analysis on the last quarter of the coding frame, $R_1(i)$.

Output

- Voice activity decision (initial value), VAD

5.3.2.1 Computation of VAD Parameters

This section describes the computation of the VAD parameters. The frame energy, E, is calculated:

$$E = \max\left(10, 10 \cdot \log_{10}\left(\frac{R_1(0)}{L_{LPC}}\right)\right). \qquad \text{(5.3.2.1-1)}$$

A partial residual energy, E^{res}, is calculated:

$$E^{res} = \max\left(0, 10 \cdot \log_{10}\left(\frac{R_1(0) \cdot \prod_{i=1}^{4}\left(1 - k_1^2(i)\right)}{L_{LPC}}\right)\right). \qquad \text{(5.3.2.1-2)}$$

The frame LPC gain, G_{LPC}, is given by

$$G_{LPC} = -10 \cdot \log_{10}\left(\prod_{i=1}^{10}\left(1 - k_1^2(i)\right)\right). \quad \text{(5.3.2.1-3)}$$

Pitch lags standard deviation is calculated by:

$$\sigma_p = \sqrt{\frac{1}{4}\sum_{i=1}^{5}\left(l_p^B(i) - \frac{1}{5}\sum_{k=1}^{5} l_p^B(k)\right)^2} \quad . \quad \text{(5.3.2.1-4)}$$

A running mean of the pitch gain is given by:

$$\overline{g}_p = 0.8 \cdot \overline{g}_p + 0.2 \cdot \left(\frac{1}{5} \cdot \sum_{i=1}^{i=5} g_p^B(i)\right). \quad \text{(5.3.2.1-5)}$$

A periodicity flag, F_p, is set, using σ_p and $\overline{g}_p$. Two average vectors of the LSFs are given by (the negative index *j* denotes the saved previous frame LSFs, and the index 0 corresponds to the LSFs of the current frame):

$$\overline{lsf}_1(i) = \frac{1}{4}\sum_{j=-3}^{j=0} lsf_j^B(i) \quad \text{and} \quad \overline{lsf}_2(i) = \frac{1}{3}\sum_{j=-3}^{j=-1} lsf_j^B(i) \ \text{ for i} = 1,\ldots,10 \quad . \quad \text{(5.3.2.1-6)}$$

A normalized maximum measure is calculated

$$M_N = \frac{\max_{i=1}^{160}\left|\tilde{s}(i)\right|}{\overline{M}_N} \quad , \quad \text{(5.3.2.1-7)}$$

where $\overline{M}_N$ is a running mean of M_N, updated during silence segments. Three spectral distortion measures are obtained. The first measure is given by:

$$SD_1 = \frac{\overline{a}^T \cdot \mathrm{R} \cdot \overline{a}}{PE_1^{LPC}} \quad , \quad \text{(5.3.2.1-8)}$$

where $\overline{a}$ is the LP prediction coefficients vector obtained from $\overline{lsf}_2$, and $\mathbf{R}$ is a symmetric Toëplitz matrix obtained for R. The second measure is given by

$$SD_2 = \sum_{i=1}^{10}\left(lsf_1(i) - \overline{lsf}_N(i)\right)^2 \quad , \quad \text{(5.3.2.1-9)}$$

where $\overline{lsf}_N(i)$ is a running mean of the LSFs during silence segments (Section 5.3.2.4). The third spectral distortion measure is given by:

$$SD_3 = \sum_{i=1}^{10}\left(lsf_1(i) - \overline{lsf}(i)\right)^2 \quad , \quad \text{(5.3.2.1-10)}$$

where $\overline{lsf}$ is a continuous running mean of the LSFs. A minimum residual energy, $E_{\min}^{res}$, is tracked over a sliding window of the last 64 frames. The minimum is tracked on 8 contiguous sub-segment for fast implementation. An active-speech mean energy is updated if $E^{res} > E_{\min}^{res} + 12$ according to:

$$\overline{E}_{speech}^{res} = 0.9 \cdot \overline{E}_{speech}^{res} + 0.1 \cdot E^{res} \quad . \tag{5.3.2.1-11}$$

A running mean of E^{res} that tracks the noise floor, $\overline{E}_{noise}^{res}$, is updated during inactive frames (Section 5.3.2.4). A signal-to-noise ratio measure is estimated by:

$$SNR = \overline{E}_{speech}^{res} - \overline{E}_{noise}^{res} \quad . \tag{5.3.2.1-12}$$

An onset flag F_{onset} is set according to energy increases and spectral changes. The running mean and history parameters are either initialized to default values or averages of their updating values over the first 32 frames.

5.3.2.2 VAD Decision

An initial voice activity decision is taken by comparing the VAD parameters to each other and to a set of fixed thresholds, controlled by the various flags. For the first 32 frames the thresholds are fixed, since the *SNR* convergence is not yet obtained, while for all subsequent frames a combination of fixed and adaptive thresholds, controlled by the *SNR*, dictates the initial VAD decision.

5.3.2.3 Hangover

An active speech VAD segment is further extended to avoid speech clipping. The energy E, F_{onset} and SNR are used to control the amount of hangover used to extend the active speech decision to the current frame if the initial decision were inactive.

5.3.2.4 Parameter Update During Silence/Noise

The running-mean parameters that track the silence are updated if either the extended VAD decision is inactive speech or ($E^{res} > \max\left(\overline{E}_{noise}^{res}, E_{\min}^{res}\right) + 3.5$) and F_p indicates no periodicity. The running-mean parameters are updated using an auto-regressive scheme.

5.3.2.5 Reset of Noise Floor Energy

The noise floor energy, $\overline{E}_{noise}^{res}$ is set for any frame subsequent to the 64th frame, according to:

$$\overline{E}_{noise}^{res} = \begin{cases} \max\left(0.75 \cdot \max\left(0.5 \cdot \left(E_{\min}^{res} + \overline{E}_{noise}^{res}\right), 0\right) + 0.25 \cdot E_{\min}^{res}, 0\right) & ; \overline{E}_{noise}^{res} < E_{\min}^{res} - 10 \text{ and } F_p = 0 \\ \max\left(E_{\min}^{res}, 0\right) & ; \overline{E}_{noise}^{res} > E_{\min}^{res} + 4 \\ \overline{E}_{noise}^{res} & ; \text{otherwise} \end{cases}$$

(5.3.2.5-1)

5.3.3 Voice Activity Detection – Option B

This scheme is used to select the voice activity decision (VAD) of the encoded frame prior to signal classification and rate-selection. Active speech is encoded at Rate 1, Rate 1/2, or Rate 1/4 and background noise is encoded at Rate 1/8. The actual data rate used for encoding the input signal is determined later by the signal classifier module.

Inputs:

- Bandwidth expanded correlation coefficients, R_1
- Long-term prediction gain, $\beta = R_p^{ol}(1)$
- SMV Operating Mode, M_{SMV}

Outputs:

- Voice activity decision (initial value), VAD

Initialization:

- Local VAD Rate histories, *VAD_Rate*(*frm*-1) and *VAD_Rate*(*frm*-2), are set to Rate 1/8 frames.
- .
- Band energy, $E_{f(i)}^{sm}(frm)$, initialization is given in 5.3.3.2.1.
- Background noise estimate, $B_{f(i)}(frm)$, initialization is given in 5.3.3.2.2.
- Signal energy estimate, $S_{f(i)}(frm)$, initialization is given in 5.3.3.2.3.

5.3.3.1 Estimating the Data Rate and VAD Based on Current Signal Parameters

A data rate local to Voice Activity Detection Option B is computed to aid the determination of the VAD. The local VAD data rate is determined by comparing the current frame energy in each of two frequency bands, $f(1)$ and $f(2)$, to background noise estimates in these respective bands. Thresholds above the background noise in each band are determined by an estimated signal-to-noise ratio in each band. These thresholds are set for Rate 1, Rate 1/2, and Rate 1/8 encoding. The highest rate calculated from the two frequency bands is then selected as the rate for the current frame. If Rate 1/8 encoding was selected VAD is set to 0, and if otherwise VAD is set to 1.

5.3.3.1.1 Computing Band Energy

The rate determination algorithm uses energy thresholds to determine the rate for the current frame. The input speech is divided into two distinct bands: band $f(1)$ spans 0.3-2.0 kHz, band $f(2)$ spans 2.0-4.0 kHz.† The band energy for band $f(i)$, $B_{f(i)}$, is calculated as

† Whenever a variable (or symbol or value) with a subscript *f*(*i*) appears in any equation or pseudocode in 5.3.3.1.4, 5.3.3.2.2, and 5.3.3.2.3, it refers to a variable (or symbol or value) associated with either band *f*(1) or band *f*(2).

$$BE_{f(i)} = R_1(0)R_{f(i)}(0) + 2.0\sum_{k=1}^{L_h-1} R_1(k)R_{f(i)}(k), \quad (5.3.3.1\text{-}1)$$

where:

$$R_{f(i)} = \sum_{n=0}^{L_h-1-k} h_i(n)h_i(n+k); \qquad 0 \le k < L_h, \quad (5.3.3.1\text{-}2)$$

and $h_i(n)$ is the impulse response of the band-pass filter i, where i = 1, 2. $R_1(k)$ is the autocorrelation sequence defined in Section 5.3.1.2.2, and $L_h = 17$ is the length of the impulse response of the band-pass filters.

The band-pass filters used for both frequency bands are defined in Table 5.3-1.

n	$h_1(n)$ (lower band)	n	$h_2(n)$ (upper band)
0	-0.0555725097656250	0	-0.0122985839843750
1	-0.0721588134765625	1	0.0437622070312500
2	-0.0103759765625000	2	0.0123901367187500
3	0.0234527587890625	3	-0.0624389648437500
4	-0.0607147216796875	4	-0.0124511718750000
5	-0.1398925781250000	5	0.1053619384765625
6	-0.0122528076171875	6	0.0124816894531250
7	0.2799224853515625	7	-0.3180694580078125
8	0.4375000000000000	8	0.4875030517578125
9	0.2799224853515625	9	-0.3180694580078125
10	-0.0122528076171875	10	0.0124816894531250
11	-0.1398925781250000	11	0.1053619384765625
12	-0.0607147216796875	12	-0.0124511718750000
13	0.0234527587890625	13	-0.0624389648437500
14	-0.0103759765625000	14	0.0123901367187500
15	-0.0721588134765625	15	0.0437622070312500
16	-0.0555725097656250	16	-0.0122985839843750

Table 5.3-1 FIR Filter Coefficients Used for Band Energy Calculations

5.3.3.1.2 Calculating Rate Determination Thresholds

The rate determination thresholds for each frequency band $f(i)$ are a function of both the background noise estimate, $B_{f(i)}(frm-1)$, and the signal energy estimate, $B_{f(i)}(frm-1)$, of the previous or $(frm-1)^{th}$ frame. Two thresholds for each band are computed as

$$T_1\left(B_{f(i)}(frm-1), SNR_{f(i)}(frm-1)\right) = k1\left(SNR_{f(i)}(frm-1)\right) B_{f(i)}(frm-1), \quad \text{(5.3.3.1-3)}$$

$$T_2\left(B_{f(i)}(frm-1), SNR_{f(i)}(frm-1)\right) = k2\left(SNR_{f(i)}(frm-1)\right) B_{f(i)}(frm-1), \quad \text{(5.3.3.1-4)}$$

where the integer $SNR_{f(i)}(frm-1)$ is:

$$SNR_{f(i)}(frm-1) = \begin{cases} 0 & ;QSNRU_{f(i)}(frm-1) < 0 \\ QSNRU_{f(i)}(frm-1) & ;0 \le QSNRU_{f(i)}(frm-1) \le 7 \\ 7 & ;QSNRU_{f(i)}(frm-1) > 7 \end{cases}, \quad \text{(5.3.3.1-5)}$$

and where:

$$QSNRU_{f(i)}(frm-1) = round\left\{\frac{\left(10\log_{10}\left(S_{f(i)}(frm-1)B_{f(i)}(frm-1)\right) - 20\right)}{5}\right\}, \quad \text{(5.3.3.1-6)}$$

The functions $k1(\bullet)$ and $k2(\bullet)$ for the different SMV modes are defined in Table 5.3-2, Table 5.3-3 and Table 5.3-4, and $B_{f(i)}(frm-1)$ and $S_{f(i)}(frm-1)$ are defined in Sections 5.3.3.2.2 and 5.3.3.2.3, respectively.

$SNR_{f(i)}(m-1)$	$k1(SNR_{f(i)}(m-1))$	$k2(SNR_{f(i)}(m-1))$
0	7.0	9.0
1	7.0	12.5937
2	8.0	17.0
3	8.59375	18.5
4	8.90625	19.40625
5	9.40625	20.90625
6	22.0	51
7	63.1875	159.1875

Table 5.3-2 Threshold Scale Factors as a Function of SNR, M_{SMV} = 0

$SNR_{f(i)}(m-1)$	$k1(SNR_{f(i)}(m-1))$	$k2(SNR_{f(i)}(m-1))$
0	7.0	9.0
1	7.0	12.59375
2	8.0	17.0
3	8.59375	18.5
4	8.90625	19.40625
5	11.28125	25.09375
6	33.0	76.5
7	126.40625	318.40625

Table 5.3-3 Threshold Scale Factors as a Function of SNR, M_{SMV} = 1

$SNR_{f(i)}(m-1)$	$k1(SNR_{f(i)}(m-1))$	$k2(SNR_{f(i)}(m-1))$
0	7.0	9.0
1	8.6875	15.8125
2	9.0	18.5
3	10.0	20.0
4	10.59375	22.8125
5	13.8125	31.59375
6	50.1875	158.5
7	200.0	631.0

Table 5.3-4 Threshold Scale Factors as a Function of SNR, M_{SMV} = 2, 3

The threshold scale factors are identical for the low- and high-frequency bands.

5.3.3.1.3 Comparing Thresholds

Band energy, $BE_{f(i)}$, is compared with two thresholds: $T_1\left(B_{f(i)}(frm-1), SNR_{f(i)}(frm-1)\right)$ and $T_2\left(B_{f(i)}(frm-1), SNR_{f(i)}(frm-1)\right)$. If $BE_{f(i)}$ is greater than both thresholds, Rate 1 is selected. If $BE_{f(i)}$ is greater than only one threshold, Rate 1/2 is selected. If $BE_{f(i)}$ is at or below both thresholds, Rate 1/8 is selected. This procedure is performed for both frequency bands and the higher of the two VAD rates selected from the individual bands is chosen as the preliminary VAD rate, VAD_Rate, for the current frame.

5.3.3.1.4 Performing Hangover

If the preliminary VAD data rate decision (from Section 5.3.3.1.3) transitions from at least two consecutive Rate 1 frames to a lower VAD rate frame, then the next *M* frames are selected as Rate 1 before allowing the VAD rate to drop to Rate 1/2 and finally to Rate 1/8. The number of hangover frames, M, is a function of the $SNR_{f(1)}(frm-1)$ (the SNR in the

lower frequency band) and is denoted as $Hangover\left(SNR_{f(1)}\left(frm-1\right)\right)$ for the different SMV modes in Table 5.3-5, Table 5.3-6 and Table 5.3-7. $SNR_{f(1)}\left(frm-1\right)$ is calculated as defined in Equation (5.3.3.1-5). The hangover algorithm is defined by the following pseudocode:

```
{
    if ( VAD_Rate(frm) == Rate 1 )
            count = 0
    if (VAD_Rate(frm-1) == Rate 1 and VAD_Rate(frm-2) == Rate 1 and VAD_Rate(frm) !=
Rate 1) {
            if ( count == 0 )
                    M = Hangover(SNRf(1)(frm-1))
            if ( count < M ) {
                    VAD_Rate(frm) = Rate 1
                    count = count + 1
            }
    }
}
```

where *VAD_Rate*(*frm*) is the VAD rate of the current frame and *VAD_Rate*(*frm*-1) and *VAD_Rate*(*frm*-2) are the VAD rates of the previous two frames, respectively. Also, the VAD rates for the previous frames, as used here, are those generated by the RDA prior to override logic, and are, therefore, not subject to modification by external rate control as described in Section 5.3.3.1.5.

$SNR_{f(1)}$(m-1)	Hangover ($SNR_{f(1)}$(m-1))
0	6
1	6
2	5
3	3
4	0
5	0
6	0
7	0

Table 5.3-5 Hangover Frames as a Function of SNR, M_{SMV} = 0

$SNR_{f(1)}(m-1)$	Hangover ($SNR_{f(1)}(m-1)$)
0	5
1	4
2	3
3	2
4	0
5	0
6	0
7	0

Table 5.3-6 Hangover Frames as a Function of SNR, M_{SMV} = 1

$SNR_{f(1)}(m-1)$	Hangover ($SNR_{f(1)}(m-1)$)
0	5
1	4
2	3
3	2
4	0
5	0
6	0
7	0

Table 5.3-7 Hangover Frames as a Function of SNR, M_{SMV} = 2, 3

5.3.3.1.5 Constraining VAD Rate Selection

The VAD rate selected by the procedures described in Sections 5.3.3.1.3 and 5.3.3.1.4 are used for the current frame except where it is modified by the following constraints.

If the previous frame was selected as Rate 1 and the current frame is selected as Rate 1/8, then the VAD rate of the current frame should be modified to Rate 1/2. There are no other restrictions on VAD rate transitions.

5.3.3.2 Updating RDA Parameters

After the RDA is complete, RDA parameters shall be updated as described in Sections 5.3.3.2.1 through 5.3.3.2.3.

5.3.3.2.1 Updating the Smoothed Band Energy

The band energy, $BE_{f(i)}$, calculated in Equation (5.3.3.1-1) is smoothed and used to estimate both the background noise energy (see Section 5.3.3.2.2) and signal energy (see Section 5.3.3.2.3) in each band. The smoothed band energy, $E_{f(i)}^{sm}(frm)$, is computed as:

$$E_{f(i)}^{sm}(frm) = \alpha_E(frm)E_{f(i)}^{sm}(frm-1) + (1-\alpha_E(frm))BE_{f(i)} \qquad ;1 \le i \le 2, \tag{5.3.3.2-1}$$

where *frm* refers to the current frame, and

$$\alpha_E(frm) = \begin{cases} 0 & ;frm \le 1 \\ 0.6 & ;frm > 1 \end{cases}. \tag{5.3.3.2-2}$$

This allows the smoothed band energy to be initialized to the band energy of the first frame $(frm = 0)$.

5.3.3.2.2 Updating the Background Noise Estimate

An estimate of the background noise level, $B_{f(i)}(frm)$, is computed for the current, or *frm*th, frame using $B_{f(i)}(frm-1)$, $E_{f(i)}^{sm}(frm)$ (see Section 5.3.3.2.1) and $SNR_{f(i)}(frm-1)$ (see Section 5.3.3.1.2). Pseudocode describing the background noise update for band $f(i)$ is given as

```
{
    if ( β < 0.30 for 4 or more consecutive frames )
        Bf(i)(m)= min{ Esmf(i)(m), 80954304, max{ 1.06Bf(i)(m-1), Bf(i)(m-1)+ 1 } }
    else {
        if ( SNRf(i)(m-1) > 1)
            Bf(i)(m)= min{ Esmf(i)(m), 80954304, max{ 1.00547Bf(i)(m-1), Bf(i)(m-
1)+ 1 } }
        else
            Bf(i)(m)= min{ Esmf(i)(m), 80954304, Bf(i)(m-1) }
    }
    if ( Bf(i)(m) < lownoise(i) )
        Bf(i)(m) = lownoise(i)
}
```

where $\beta = R_p^{ol}(1)$, is defined in Section 5.3.9.2, and $E_{f(i)}^{sm}(frm)$, as well as $SNR_{f(i)}(frm-1)$ are defined in Section 5.3.3.2.1 and Section 5.3.3.1.2, respectively. *lownoise*(1) equals 160.0 and *lownoise*(2) equals 80.0.

At initialization, the background noise estimate for the first frame, $B_{f(i)}(0)$, is set to 80,954,304 for both frequency bands, and the consecutive frame counter for β is set to zero. Initialization also occurs if the audio input to the encoder is disabled and then enabled.†

† This prevents the silence before the audio is connected from being mistaken as unusually low background noise.

5.3.3.2.3 Updating the Signal Energy Estimate

The signal energy, $S_{f(i)}(frm)$, is computed as

{
If (β > 0.5 for 5 or more consecutive frames)
$S_{f(i)}(frm) = \max\{ E^{sm}_{f(i)}(frm), \min\{ 0.97 S_{f(i)}(frm\text{-}1), S_{f(i)}(frm\text{-}1) - 1 \} \}$
else
$S_{f(i)}(frm) = \max\{ E^{sm}_{f(i)}(frm), S_{f(i)}(frm\text{-}1) \}$
}

where $\beta = R_p^{ol}(1)$, is defined in Section 5.3.9.2, and $E^{sm}_{f(i)}(frm)$ is defined in Section 5.3.3.2.1.

At initialization, the signal energy estimates for the first frame, $S_{f(1)}(frm)$ and $S_{f(2)}(frm)$, are set to 32000 and 3200, respectively, and the consecutive frame counter for β is set to zero. Initialization also occurs if the audio input to the encoder is disabled and then enabled.

5.3.4 Music Detection

Routine name: `MUS_musdetect`

A music detector is used to classify the input signal as either music or speech. The detection takes place after the VAD decision is made. The music detector reuses some of the parameters of the VAD and extracts additional parameters from the speech. It makes a music detection decision and also corrects the initial voice activity decision from the VAD module using a set of thresholds.

5.3.4.1 Input to Music Detector

In addition to several parameters that are obtained in the VAD option A, or that are calculated independently if VAD option B is used, the music detector uses the following input:

- a buffer of 5 previous normalized pitch correlations, $corr_p^B(i) \quad i = 1,\ldots,5$

5.3.4.2 Computation of Music Detector Parameters

This section describes the computation of the music detector parameters. A difference Δ_{lsf} between $lsf_1(2)$ and $lsf_1(1)$, obtained from the LPC analysis on the last quarter of the coding frame, is computed. A running mean of $lsf_1(1)$ is computed as:

$$\overline{lsf}(1) = 0.75 \cdot \overline{lsf}(1) + 0.25 \cdot lsf_1(1) \tag{5.3.4.2-1}$$

A running mean energy, $\overline{E}$, is calculated as:

$$\overline{E} = 0.75 \cdot \overline{E} + 0.25 \cdot E \tag{5.3.4.2-2}$$

where E is the frame energy. A spectral difference SD_4 is calculated as:

$$SD_4 = \sum_{i=1}^{10} \left(k_1(i) - \overline{k}_N(i)\right)^2 \tag{5.3.4.2-3}$$

where $\overline{k}_N$ is the running mean reflection coefficients of noise/silence. The running mean of the partial residual energy $\overline{E}_N^{res}$ is updated along $\overline{k}_N$ when the input VAD is inactive as:

$$\overline{E}_N^{res} = 0.9 \cdot \overline{E}_N^{res} + 0.1 \cdot E^{res} \tag{5.3.4.2-4}$$

and

$$\overline{k}_N(i) = 0.75 \cdot \overline{k}_N(i) + 0.25 \cdot k_1(i) \qquad i = 1,\ldots,10 \tag{5.3.4.2-5}$$

A running mean of the normalized pitch correlation is given by:

$$\overline{corr}_p = 0.8 \cdot \overline{corr}_p + 0.2 \cdot \left(\frac{1}{5} \cdot \sum_{i=1}^{i=5} corr_p^B(i) \right) \tag{5.3.4.2-6}$$

A periodicity flag F_p, similar to the one in the VAD module, is calculated using $corr_p^B(\cdot)$ and different thresholds. A spectral continuity counter c_{sp} is incremented if $k_1(2) \geq 0.0$ and $\overline{corr}_p < 0.5$ and reset to 0 otherwise. A periodicity continuity counter c_{pr} is incremented each time F_p is set and reset to 0 every 32 frames. A running mean of the periodicity counter $\bar{c}_{pr}$ is updated every 32 frames as:

$$\bar{c}_{pr} = \alpha \cdot \bar{c}_{pr} + (1-\alpha) \cdot c_{pr} \tag{5.3.4.2-7}$$

where

$$\alpha = \begin{cases} 0.98 & c_{pr} > 12 \\ 0.95 & c_{pr} > 10 \\ 0.90 & \text{otherwise} \end{cases} \tag{5.3.4.2-8}$$

A very low frequency noise flag F_f is set if the initial VAD is inactive and either $lsf_1(1) < 110$ Hz or $\overline{lsf}(1) < 150$ Hz. The initial inactive VAD decision from the VAD module is corrected to an active VAD decision by comparing SD_4, E^{res}, $\overline{E}_N^{res}$, E, and $\bar{c}_{pr}$ to a set of thresholds. A noise continuity counter C_N is incremented each time the corrected VAD is inactive and is reset otherwise. A running mean of the normalized pitch correlation $\overline{corr}_p^N$ is updated if either the corrected VAD is inactive or F_f is set:

$$\overline{corr}_p^N = 0.8 \cdot \overline{corr}_p^N + 0.2 \cdot \left(\frac{1}{5} \cdot \sum_{i=1}^{i=5} corr_p^B(i) \right) \tag{5.3.4.2-9}$$

A music continuity counter c_M is adaptively incremented and decremented by comparing the MD parameters to each other and to a set of fixed thresholds, controlled by the various flags. A running mean of this counter $\bar{c}_M$ is updated as:

$$\bar{c}_M = 0.9 \cdot \bar{c}_M + 0.1 \cdot c_M \tag{5.3.4.2-10}$$

5.3.4.3 Music Detection Decision

The music detection flag F_{MUS} is set if either $\bar{c}_{pr} \geq 18$ or $\bar{c}_M > 200$. In this case, $\overline{E}_N^{res}$ is reset to 0.

5.3.4.4 Reset of tracking counters

$\bar{c}_{pr}$, c_{pr}, and c_{sp} are reset to 0 if either $E < 13$ dB or F_f is set or $c_{pr} > 50$, or $c_{sp} > 20$.

5.3.4.5 Update of pitch lag, pitch correlation and pitch gain buffers

The pitch lag buffer $l_p^B(\cdot)$ is updated on a frame basis as:

$$l_p^B(i) = l_p^B(i+2),\ i = 0,1,2 \tag{5.3.4.5-1}$$

and

$$l_p^B(3) = l_p^{ol}(0), l_p^B(4) = l_p^{ol}(1) \quad \text{(5.3.4.5-2)}$$

Similarly, the pitch correlation buffer $corr_p(\cdot)$ is updated on a frame basis as:

$$corr_p^B(i) = corr_p^B(i+2),\ i = 0,1,2 \quad \text{(5.3.4.5-3)}$$

and

$$corr_p^B(3) = R_p^{ol}(0), corr_p^B(4) = R_p^{ol}(1). \quad \text{(5.3.4.5-4)}$$

On the other hand, the pitch gain buffer $g_p^B(\cdot)$ is updated as on a subframe basis during Rate 1 and Rate 1/2 as:

$$g_p^B(i) = g_p^B(i+1),\ i = 0,1,2,3, \quad \text{(5.3.4.5-5)}$$

and

$$g_p^B(4) = g_a, \quad \text{(5.3.4.5-6)}$$

where g_a is the unquantized adaptive-codebook gain.

5.3.5 Calculation of Weighting Coefficient for 4-Subframe Structure

Routine name: `PWF_WeighingFilters`

Input:

- First element of the interpolated reflection coefficient vector, $\hat{k}_m(0)$, for $m = 0,1,2,3,4$
- Interpolated LPC prediction gain, $\hat{G}_m^{LPC}$, for $m = 0,1,2,3,4$
- Interpolated prediction coefficients, $\hat{a}_m(i)$, for $m = 0,1,2,3,4$

Output:

- Weighing filter numerator coefficients, $\hat{a}_m^{num}(i)$ for $m = 0,1,2,3,4$
- Weighing filter denominator coefficients, $\hat{a}_m^{den}(i)$, for $m = 0,1,2,3,4$

The weighting filter coefficients for the four subframes and the lookahead, indexed by $m = 0,1,2,3,4$, are calculated by the following procedure. A weighting coefficient for the numerator is given by:

$$w_m^{num} = 0.9 + \min\left(0.03 \cdot \max\left(\hat{G}_m^{LPC} - 5, 0\right) \cdot \max\left(-\hat{k}_m(0) - 0.5, 0\right), 0.04\right) \quad (5.3.5\text{-}1)$$

and a weighting coefficient for the denominator is given by

$$w_m^{den} = 0.6 + 1.25 \cdot \min\left(\hat{k}_m(0) + 0.9, 0\right). \quad (5.3.5\text{-}2)$$

The filter coefficient for the numerator are calculated by:

$$\hat{a}_m^{num}(i) = \hat{a}_m(i) \cdot \left(w_m^{num}\right)^i \quad i = 1,\ldots,10. \quad (5.3.5\text{-}3)$$

The filter coefficients for the denominator are calculated by:

$$\hat{a}_m^{den}(i) = \hat{a}_m(i) \cdot \left(w_m^{den}\right)^i \quad i = 1,\ldots,10. \quad (5.3.5\text{-}4)$$

5.3.6 Generating the Weighted Residual

Routine name: `PWF_speech_to_residu`

Input:

- Weighing filter numerator coefficients, $\hat{a}_m^{num}(i)$ for $m = 0,1,2,3,4$
- Pre-processed speech, $\widetilde{s}(n)$

Output:

- Weighted residual, $r_w(n)$

For each subframe, indexed by $m = 0,1,2,3,4$, the weighted residual signal, $r_w(n)$, is generated from the speech by the filter given by

$$H_m(z) = 1 + \sum_{i=1}^{10} \hat{a}_m^{num}(i) z^{-i} \qquad (5.3.6\text{-}1)$$

The filter parameters, $\hat{a}_m^{num}(i)$, are the filter coefficients for each of the four 5 ms subframes and the lookahead, calculated at Section 5.3.5.

5.3.7 Initial Setting of Voiced/Unvoiced Level

Function name: `CLA_NoiseUnvoicedDetect`

Input:

- Pre-processed speech, $\tilde{s}(n)$
- Weighted residual, $r_w(n)$

Output:

- Voiced/Unvoiced level (initial value), C_{VUV}
- Energies of weighted residual for two halves of the frame, $E_{rw}(0)$ and $E_{rw}(1)$
- Sharpness of weighted residual, S_{rp}^{wr}

The initial setting of voiced/unvoiced level of speech, C_{VUV}, is to the values 0, 1, or 2. The value of the parameter is further refined, and extended to include level 3, in Sections 5.3.12 and 5.3.15.

5.3.7.1 Calculation of Decision Parameters

The sharpness of the weighted residual is calculated by:

$$S_{rp}^{wr} = \frac{\frac{1}{L_F}\sum_{n=0}^{L_{frm}-1}\left|r_w(n)\right|}{\max_{n=0,\ldots,L_{frm}-1}\left|r_w(n)\right|}. \qquad (5.3.7.1\text{-}1)$$

The first order prediction coefficient of the speech on the second half of the frame is given by:

$$\breve{p} = \frac{\sum_{n=L_{frm}/2}^{L_{frm}-2}\tilde{s}(n)\cdot\tilde{s}(n+1)}{\sum_{n=L_{frm}/2}^{L_{frm}-1}\tilde{s}^2(n)} \qquad (5.3.7.1\text{-}2)$$

The zero-crossing rate of the speech signal on the second half of the frame, C_{zc}, is calculated by evaluating the number of sign changes of the speech signal over the second half of the frame, and normalizing it with the half-frame size.

The energies of the weighted residual on the first half and on the second half of the frame are given by:

$$E_{rw}(0) = \sum_{n=0}^{L_{frm}/2-1} r_w^2(n) \qquad E_{rw}(1) = \sum_{n=L_{frm}/2}^{L_{frm}-1} r_w^2(n) \qquad (5.3.7.1\text{-}3)$$

The energy of the signal of the first ¾ of the frame is calculated by:

$$E_{3/4} = \frac{1}{3/4\, L_{frm}} \sum_{n=0}^{3/4\, L_{frm}-1} \tilde{s}^2(n) \qquad \text{(5.3.7.1-4)}$$

The number of speech samples in the frame that has an absolute value less than 0.1 is counted, and denoted $C_{|0.1|}$.

5.3.7.2 Initial Setting Voiced/Unvoiced Level

Six of the 7 parameters (E_2 excluded) are compared to a set of thresholds, for an initial setting of the voiced/unvoiced level, C_{VUV}, for the frame. The initial setting is 0, 1, or 2.

5.3.8 Generating the Weighted Speech

Routine name: `PWF_residu_to_wspeech`

Input:

- Weighted residual, $r_w(n)$
- Weighing filter denominator coefficients, $\hat{a}_m^{den}(i)$, for $m = 0,1,2,3,4$
- First element of the interpolated reflection coefficient vector, $\hat{k}_m(0)$, for $m = 0,1,2,3,4$
- Spectral flatness flag, F_{flat}

Output:

- Weighted speech, $s_w(n)$
- Adaptive low-pass filter parameters, μ_m, for $m = 0,1,2,3,4$

The weighted speech, s_w, is generated from the weighted residual signal with the application of a fixed weighted synthesis filter and an adaptive low-pass filter. The transfer function of the weighted synthesis filter for each subframe, indexed by $m = 0,1,2,3,4$, is given by:

$$H_m(z) = \frac{1}{1 + \sum_{i=1}^{10} a_m^{den}(i) \cdot z^{-i}} \qquad \text{(5.3.7.2-1)}$$

The weighting filter parameters, $a_m^{den}(i)$, are the coefficients for each of the four 5 ms subframes and the lookahead calculated in Section 5.3.5. The adaptive low-pass filter is given by:

$$L(z) = 1 + \mu_m \cdot z^{-1} \qquad \text{(5.3.7.2-2)}$$

Where μ_m is given by:

$$\mu_m = \begin{cases} \max\left\{\min\left\{\hat{k}_m(0) + 0.5, 0.3\right\}, 0\right\} & F_{flat} = 0 \\ 0 & F_{flat} = 1 \end{cases} \qquad \text{(5.3.7.2-3)}$$

5.3.9 Open-Loop Pitch Detection

Routine name: `PIT_ol_2pitchs`

Input:

- Weighted speech, $s_w(n)$
- Previous frame class, C_{FR}^{prv}
- Previous frame open-loop pitch lags
- Previous frame open-loop pitch normalized correlation

Output:

- Open-loop pitch lags, $l_p^{ol}(m)$, $m = 0,1,2$
- Open-loop pitch normalized correlation, $R_p^{ol}(m)$, $m = 0,1,2$

The open-loop pitch detection module calculates 3 open-loop pitch values, for the first half of the encoding frame, the second half of the encoding frame, and the lookahead. It searches and determines the open-loop pitch lags for the look ahead and for the second half of the encoding frame. It obtains the open-loop pitch lag for the first half of the encoding frame by refining the open-loop pitch lag of the previous frame lookahead open-loop pitch lag.

5.3.9.1 Open-Loop Pitch Detection on the Look Ahead Buffer

The open-loop pitch on the lookahead buffer is estimated by selecting 4 open-loop pitch candidates on 4 intervals, selecting an initial best candidate, and selecting a final best candidate using smoothing.

5.3.9.1.1 Selection of 4 Open-Loop Pitch Candidates

The normalized correlation of the weighted speech is calculated for the interval [17,148]. The normalized correlation is given by:

$$R_p(l) = \frac{\sum_{n=0}^{79} s_w(n) \cdot s_w(n-l)}{\sqrt{\sum_{n=0}^{79} s_w^2(n) \cdot \sum_{n=0}^{79} s_w^2(n-l)}} \qquad l = 17,\ldots,148 \tag{5.3.9.1-1}$$

The maximum of the normalized correlation is found for each of the four intervals [17,33], [34,67], [68,135], and [136,148], resulting in four maximum values of $R_p(l)$. The locations of these four $R_p(l)$ are the initial estimate of the four open-loop pitch candidates. The non-uniform size of the intervals is to help the detection of potential pitch multiples, as will be discussed in the next section.

5.3.9.1.2 Correction of Open-Loop Pitch Multiples for the Lookahead

The location of the maximum of the normalized correlation $R_p(l)$ from the 4 possible candidates is selected as the initial candidate, l_{cand}^{ol}, for the open-loop pitch lag. In order to avoid pitch doubling, pitch correction is performed by inspecting the pitch candidates from the lower pitch ranges. The lookahead pitch value is corrected using the last two frames open-loop pitch lags that were calculated on the second half of the respective frame, and their 4-level reliability indicator. To avoid pitch multiples, the pitch is corrected by inspecting the pitch candidates selected from the intervals of lower pitch lag. A weighting parameter, *D*, is calculated for each lower-lag pitch candidate, based on its fitting to the previous frames' pitch contour. A flag signifies if the lower-lag pitch candidate is a possible 1/2, 1/3, or 1/4 of the initial candidate, l_{cand}^{ol}.

If the lower-lag pitch fits into the pitch contour, if it is a possible sub-multiple of the initial candidate, and if its normalized correlation $R_p(l)$ is larger than 0.25, the lower-lag pitch value replaces the initial candidate l_{cand}^{ol} as the selected open-loop pitch, $l_p^{ol}(2)$.

5.3.9.2 Open-Loop Pitch Detection on the Second Half of the Encoding Frame

Before detecting the open-loop pitch detection for the second half of the frame, the past pitch values and their corresponding reliability indicators, used for the pitch contour matching, are modified. The open-loop pitch from the previous frame lookahead, or the open-loop pitch from the current frame lookahead (computed in the last section), replaces the open-loop pitch kept from 2 frames in the past, if the pitch estimates of the last two frames were not reliable.

Following this correction, the normalized correlation and the initial 4 pitch candidates for this segment are calculated in similar fashion to Section 5.3.9.1.1, and pitch multiples are eliminated, in similar fashion to Section 5.3.9.1.2, to select an updated open-loop pitch candidate.

5.3.9.3 Final Open-Loop Pitch Correction

The final open-loop pitch for the lookahead obtained according to Section 5.3.9.1 is selected as $l_p^{ol}(2)$, and the normalized correlation is returned as:

$$R_p^{ol}(2) = R_p\left(l_p^{ol}(2)\right) \tag{5.3.9.3-1}$$

The open-loop pitch candidate for the second half of the encoding frame is selected according to Section 5.3.9.2. The open-loop pitch candidate for the first half of the encoding frame is given by the lookahead open-loop pitch value of the previous frame. The two later candidates are tested and modified to select the final open-loop pitch values, providing a smooth pitch contour and further correction of pitch sub-multiples.

5.3.9.3.1 Smoothing of the Open-Loop Pitch Candidates

The open-loop pitch of the previous two frames, the open-loop pitch candidate for the first and the second halves of the frame, the open-loop pitch for the lookahead, and their corresponding normalized correlations, are examined for smoothness and reliability. According to this examination, the open-loop pitch for the second half of the frame can be

modified to either the value of the open-loop pitch on the lookahead, or an average of the open-loop pitch on the first half of the frame and the open-loop pitch on the lookahead, or can be reselected from the initial pitch candidates that were computed according to Section 5.3.9.1.1. The open-loop pitch for the first half of the frame can be modified, according to this examination, to either the value of the average of the open-loop pitch of the previous frame and the open-loop pitch on the second half of the frame, or can be reselected from the initial pitch candidates that were computed according to Section 5.3.9.1.1.

5.3.9.3.2 Further Correction of Pitch Sub-Multiples

For voiced speech frames, open-loop pitch lag candidates of less than 50 samples for the first and second halves of the frame are further tested against an average of the open-loop pitch lag. If they are significantly smaller than the average, they are modified to either one of initial pitch candidates that were computed according to Section 5.3.9.1.1, or to the open-loop pitch of the previous frame. The average pitch lag is updated on a frame-by-frame basis using the final open-loop pitch candidate from the second half of the frame. The update is done in a recursive manner with a leakage factor of 0.75 and it is done only for voiced speech with reliable and smooth pitch contour. The average pitch lag is reset to 40 after 3 frames of inactive speech.

The refined pitch lag for the first half of the frame is returned as $l_p^{ol}(0)$ and the selected pitch lag for the second half of the frame is returned as $l_p^{ol}(1)$. The normalized correlations are returned as:

$$R_p^{ol}(0) = R_p\left(l_p^{ol}(0)\right) \tag{5.3.9.3-2}$$

$$R_p^{ol}(1) = R_p\left(l_p^{ol}(1)\right) \tag{5.3.9.3-3}$$

5.3.10 Initial Frame Classification

Routine name: `CLA_signal_classifier`

Input:

- Pre-processed speech, $\tilde{s}(n)$
- Open-loop pitch lag on the second half of the frame $l_p^{ol}(1)$
- Open-loop pitch normalized correlation on the second half of the frame, $R_p^{ol}(1)$

Output:

- Frame class decision (initial value), C_{FR}

The initial frame classification perform the initial classification of the frame into one of the 6 categories:

0 - Silence
1 - Noise-like Unvoiced
2 - Unvoiced
3 - Onset
5 - Non-stationary voiced
6 - Stationary voiced

Note that the category number 4 is not used. Also, class 1, of noise-like unvoiced, and class 6, of stationary voiced, are determined later.

5.3.10.1 Average and Variance of Open-Loop Pitch Values

An average of the open-loop pitch, $l_p^{ol}(1)$, on the last two frames and the current frame, is calculated and denoted by l_p^{avg}. The variance of the same 3 open-loop pitch values is calculated and denoted by l_p^{var}.

5.3.10.2 Noise-Suppressed Classification Parameters

5.3.10.2.1 First Order Reflection Coefficients for 4 Subframes

The frame is divided into 4 subframes of 40 samples each. For each of such subframe the first order reflection coefficient is calculated, using a Hamming window of 80 samples centered on the middle of the subframe. The first order reflection coefficient for the m^{th} subframe is denoted k_m^{cl} and is given by:

$$k_m^{cl} = \frac{\sum_{n=1}^{79} \tilde{s}_m(n) \cdot \tilde{s}_m(n-1)}{\sum_{n=0}^{79} \tilde{s}_m^2(n)} \qquad m = 0,\ldots,3 \tag{5.3.10.2-1}$$

where

$$\widetilde{s}_m(n) = \widetilde{s}(40 \cdot m - 20 + n) \cdot W_H(n) \qquad n = 0,\ldots,79 \tag{5.3.10.2-2}$$

and $W_H(n)$ is a Hamming window of size 80.

5.3.10.2.2 Noise-Suppressed Spectral Tilt, Absolute Maximum, and Open-Loop Pitch Correlation

At the next step, the frame is divided into 8 subframes of 20 samples each. For each of such subframe, indexed by m, the spectral tilt for the m^{th} subframe is defined by $T_m^{cl} = k_{\lfloor m/2 \rfloor}^{cl}$. For each of such subframe the absolute maximum value of the samples in an interval is calculated and denoted M_m^{cl}. The interval for the calculation of the absolute maximum value is related, but not identical, to the subframe, and is defined by its start and end points. The start point is given by:

$$\text{start}_\text{m} = \begin{cases} \max\left(-140, 100 - 1.5 \cdot l_p^{ol}(1)\right) + 20m & l_p^{ol}(1) > 20 \\ 80 + 20m & l_p^{ol}(1) \le 20 \end{cases} \tag{5.3.10.2-3}$$

(Negative indices relate to past samples). The end point is given by $\text{end}_\text{m} = 100 + 20m - 1$. The maximum of the absolute value is given by:

$$M_m^{cl} = \max_i \left\{ \left| s(i) \right|, start_m \le i \le end_m \right\} \tag{5.3.10.2-4}$$

For each such subframe the energy is calculated on an interval, which is related to the subframe, and is denoted E_m^{cl}. Its start and end points define the interval. The start point is given by:

$$\text{start}_\text{m} = \begin{cases} \max\left(-140, 20 - 2 \cdot l_p^{ol}(1)\right) + 20m & l_p^{ol}(1) > 20 \\ 20m & l_p^{ol}(1) \le 20 \end{cases} \tag{5.3.10.2-5}$$

The end point is given by $\text{end}_\text{m} = 20 + 20m - 1$. The energy of the segment is calculated by:

$$E_m^{cl} = 1 + \frac{\sum_{i=start_m}^{end_m} s^2(i)}{end_m - start_m} \tag{5.3.10.2-6}$$

The running means of the energy, the spectral tilt, the absolute maximum and the open-loop pitch normalized correlation ($\overline{E}_{noise}, \overline{T}_{noise}, \overline{M}_{noise}\ \overline{R}_{noise}$) are updated for each on the 20-sample subframes. For the first 6 frames, the update is performed if $T_n^{cl} < -0.4$ and $R_p^{ol}(1) < 0.5$ according to:

$$\begin{aligned}
\overline{E}_{noise} &= 0.80 \cdot \overline{E}_{noise} + 0.20 \cdot E_m^{cl} \\
\overline{T}_{noise} &= 0.75 \cdot \overline{T}_{noise} + 0.25 \cdot T_m^{cl} \\
\overline{M}_{noise} &= 0.75 \cdot \overline{M}_{noise} + 0.25 \cdot M_m^{cl} \\
\overline{R}_{noise} &= 0.75 \cdot \overline{R}_{noise} + 0.25 \cdot R_p^{ol}(1)
\end{aligned} \qquad (5.3.10.2\text{-}7)$$

For frames later than the first six frames the update is controlled by the VAD decision, and the update is performed if the $VAD = 0$ according to:

$$\begin{aligned}
\overline{E}_{noise} &= 0.999 \cdot \overline{E}_{noise} + 0.001 \cdot E_m^{cl} \\
\overline{T}_{noise} &= 0.990 \cdot \overline{T}_{noise} + 0.010 \cdot T_m^{cl} \\
\overline{M}_{noise} &= 0.990 \cdot \overline{M}_{noise} + 0.010 \cdot M_m^{cl} \\
\overline{R}_{noise} &= 0.990 \cdot \overline{R}_{noise} + 0.010 \cdot R_p^{ol}(1)
\end{aligned} \qquad (5.3.10.2\text{-}8)$$

The noise-to-signal ratio is given by:

$$NSR_m^{cl} = \min\left(0.968, \sqrt{\frac{\overline{E}_{noise}}{E_m^{cl}}}\right) \qquad (5.3.10.2\text{-}9)$$

A buffer of the last 7 values of the noise-suppressed spectral tilt and the noise-suppressed absolute maximum is maintained. (The values in the buffer can be from the previous frame or the current frame, as the subframe index progresses from the first to the last.) The updated value of each noise-suppressed parameter is calculated according to:

$$\begin{aligned}
T_{ns}^{7}(m) &= T_m^{cl} - NSR_m^{cl} \cdot \overline{T}_{noise} \\
M_{ns}^{7}(m) &= T_m^{cl} - NSR_m^{cl} \cdot \overline{M}_{noise}
\end{aligned} \qquad (5.3.10.2\text{-}10)$$

A noise-suppressed value of the open-loop pitch normalized correlation is calculated according to:

$$R_{ns}(m) = R_p^{ol}(1) - NSR_m^{cl} \cdot \overline{R}_{noise} \qquad (5.3.10.2\text{-}11)$$

5.3.10.2.3 Spectral Tilt Slope, Absolute Maximum Slope, and Slope Statistics

A first order derivative (slope) of the spectral tilt and the absolute maximum for the m^{th} subframe is calculated by:

$$\begin{aligned}
T'_{ns}(m) &= \sum_{i=1}^{7} i \cdot \left(T_{ns}^{i}(m) - T_{ns}^{0}(m)\right)/140 \\
M'_{ns}(m) &= \sum_{i=1}^{7} i \cdot \left(M_{ns}^{i}(m) - M_{ns}^{0}(m)\right)/140
\end{aligned} \qquad (5.3.10.2\text{-}12)$$

For each frame, using the noise-suppressed parameters, the minimum of the spectral tilt slope for the frame, $T'_{\min}$, the maximum of the slope of the absolute maximum, $M'_{\max}$, the sum of the slopes of the absolute maximums, $\overline{M'}$, the minimum of the spectral tilt, $T_{\min}$, and the sum of the noise suppressed spectral tilt, $\overline{T}$, are given by:

$$
\begin{aligned}
T'_{\min} &= \min_{i=0}^{7} T'_{ns}(i) \\
M'_{\max} &= \max_{i=0}^{7} M'_{ns}(i) \\
\overline{M'} &= \sum_{i=0}^{7} M'_{ns}(i) \\
\overline{T} &= \sum_{i=0}^{7} T_{ns}^{7}(i) \\
T_{\min} &= \min_{i=0}^{7} T_{ns}^{7}(i)
\end{aligned}
\qquad \text{(5.3.10.2-13)}
$$

An auto-regressive average of $T_{\min}$ is generated by $T_{\min}^{avg} = 0.75 \cdot T_{\min}^{avg} + 0.25 \cdot T_{\min}$.

5.3.10.2.4 Statistics of Open-Loop Pitch Correlation

The maximum of the noise-suppressed open-loop pitch correlation, $R_{\max}$, and the sum of noise-suppressed open-loop pitch correlation, $\overline{R}$, are given by:

$$
\begin{aligned}
R_{\max} &= \max_{i=1}^{8} R_{ns}(i) \\
\overline{R} &= \sum_{i=1}^{8} R_{ns}(i)
\end{aligned}
\qquad \text{(5.3.10.2-14)}
$$

An auto-regressive average of $\overline{R}$ is given by $\overline{R}^{avg} = 0.75 \cdot \overline{R}^{avg} + 0.25 \cdot \overline{R}$.

5.3.10.2.5 Classification Decision

The calculated parameters are compared to a set of thresholds to generate two Boolean parameters, one indicating voiced speech, and the other indicates onset speech. The Boolean parameters are further modified according to the stored Boolean parameters of the previous frame. The voiced parameter, the onset parameter, the VAD decision, and the previous frame initial classification decision, are used in a set of Boolean conditions to provide the initial frame classification decision for the current frame, C_{FM}. The initial frame classification is limited to the classes of Silence (0), Unvoiced (2), Onset (3), or Voiced (5).

5.3.11 Noise-to-Signal Ratio (NSR) Estimation

Routine name: `SNR_Calc_NSR_enc`

Input:

- Pre-processed speech, $\tilde{s}(n)$
- LSFs calculated on the last quarter of the encoding frame, $lsf_1(i)$
- Voice activity detection flag, VAD
- Frame class (initial value), C_{FM}

Output:

- Noise-to-signal measure, NSR

The noise-to-signal ratio (NSR) is a measure of the level of background noise presented to the SMV codec. The NSR is the reciprocal of the more familiar signal-to-noise ratio value, but is bounded by 1.

5.3.11.1 Signal Input to the NSR Estimation Function

The input of $\tilde{s}(n)$ to the NSR calculation function is a segment of 160 samples that includes the second half of the coding frame and the lookahead buffer.

5.3.11.2 NSR counters

Three counters are used in the NSR estimation. C_{noise} counts the number of consecutive frames where the initial frame classification is below 3 (not onset or voiced frames). C_{vad} counts the number of consecutive frames where the VAD indicates inactive voice. C_{NSR} is a modulo 1000 frame counter, which helps in resetting the NSR estimate.

5.3.11.3 NSR Calculation

The energy of the current segment is calculated by:

$$E^{NSR} = \sum_{i=0}^{L_{frm}-1} \tilde{s}^2(i) \qquad \text{(5.3.11.3-1)}$$

If the current frame is zero in modulo 1000, the returned NSR is 0, and the average energy memory, $\overline{E}_p^{NSR}$ is set to E^{NSR}. Otherwise, a spectral difference measure is calculated, as the sum of the absolute values of the current frame LSFs, $lsf_1(i)$, and the previous frame LSFs, $lsf_{1,prv}(i)$:

$$D_{lsf} = \sum_{i=1}^{10} \left| lsf_1(i) - lsf_{1,prv}(i) \right| \qquad \text{(5.3.11.3-2)}$$

An auto-regressive average of the spectral distance is calculated by

$$\overline{D}_{lsf} = 0.75 \cdot \overline{D}_{lsf} + 0.25 \cdot D_{lsf} . \quad (5.3.11.3\text{-}3)$$

An energy difference measure is calculated between the current frame energy, E^{NSR}, and the previous frame energy E_p^{NSR}, by:

$$D_E = \frac{\left| E^{NSR} - E_p^{NSR} \right|}{\max\left(E^{NSR} + E_p^{NSR}, 8000 \right)} \quad (5.3.11.3\text{-}4)$$

An auto-regressive average of the energy difference distance is calculated by

$$\overline{D}_E = 0.75 \cdot \overline{D}_E + 0.25 \cdot D_E \quad (5.3.11.3\text{-}5)$$

An SNR measure is calculated as

$$SNR = \frac{E^{NSR}}{\max\left\{ \overline{E}_p^{NSR}, 0.1 \right\}}, \quad (5.3.11.3\text{-}6)$$

and if the VAD indicates inactive voice, $\overline{E}_p^{NSR}$ is updated according to:

$$\overline{E}_p^{NSR} = \begin{cases} E^{NSR} & \overline{E}_p^{NSR} < 0.1 \\ 0.75 \cdot \overline{E}_p^{NSR} + 0.25 \cdot E^{NSR} & \overline{E}_p^{NSR} \geq 0.1 \end{cases} \quad (5.3.11.3\text{-}7)$$

The SNR, C_{vad}, C_{NSR}, D_{lsf}, $\overline{D}_{lsf}$, D_E, and $\overline{D}_E$, are compared to set of thresholds and a locally refined VAD decision is obtained. This locally refined VAD is also controlled by the average of the background noise energy $\overline{E}_{noise}$ and the modulo counter C_{NSR}. If the locally refined VAD decision indicates inactive speech, the average of the background noise is updated according to:

$$\overline{E}_{noise} = \begin{cases} E^{NSR} & \overline{E}_{noise} < 0 \\ \min\left(0.85 \cdot \overline{E}_{noise} + 0.15 \cdot E^{NSR}, E^{NSR} \right) & \overline{E}_{noise} \geq 0 \end{cases} \quad (5.3.11.3\text{-}8)$$

($\overline{E}_{noise}$ can be negative due to initialization.) The *NSR* is calculated as:

$$NSR = \min\left(\sqrt{\frac{\max\left(\overline{E}_{noise} - 16000, 0 \right)}{\max\left(E^{NSR}, 0.1 \right)}}, 1.0 \right) \quad (5.3.11.3\text{-}9)$$

5.3.12 Classification Refinement

Routine name: `CLA_Class_Correct`

Input:

- Music detection flag, F_{MUS}
- Frame class decision (initial value), C_{FR}, and previous frame class decision, C_{FR}^{prv}
- VAD decision (initial value), VAD
- Voiced/unvoiced level (initial value), C_{VUV}
- Open-loop normalized pitch correlations, $R_p^{ol}(l)$, $l = 0,1,2$
- Noise-to-signal ratio, NSR
- Interpolated reflection coefficients, $\hat{k}_m(i)$, $m = 0,1,2,3,4$
- Sharpness of weighted residual, S_{rp}^{wr}, (Section 5.3.7)
- SMV mode, M_{SMV}
- Energy of the lookahead buffer, given by $R_2(0)$.
- Energy of the weighted residual given by $E_{rw}(0) + E_{rw}(1)$, (Section 5.3.7)
- LPC gain on the last quarter of the encoding frame, G_1^{LPC}

Output:

- Frame class decision (refined), C_{FR}
- Auxiliary class decision (used mainly for signal modification), C_{SM}
- Onset flag, F_{onset}
- LPC onset flag, F_{LPC_onset}
- Noisy-voiced flag F_{noisyV}
- VAD decision (refined), VAD
- Voiced/unvoiced level (refined), C_{VUV}

5.3.12.1 Operation of the Classification Refinement

The class refinement uses previously calculated parameters, and saved values from previous frames, to refine the classification decision, the VAD decision, and the voiced/unvoiced level decision. It also sets three flags that are used later, F_{onset}, F_{LPC_onset}, and F_{noisyV}. F_{onset} are used during the rate selection and the final classification correction,

F_{LPC_onset} is used to control the LSFs smoothing, and F_{noisyV} is used to control the gain quantization, the perceptual weighting filter and the fixed-codebook search. The classification refinement procedure also sets C_{SM}, an auxiliary classification decision, used to control several aspects of the encoding algorithm, and in particular, the weighted speech modification algorithm (Section 5.3.15). C_{SM} takes the values of $-1,\ldots,4$, where an increasing number indicates an increasing level of voicing.

The decision parameters are compared to each other, and to a set of thresholds, for a sequential refinement of initial decision parameters and for the setting of new decision parameters and flags.

5.3.13 Rate Selection

Routine name: `CLA_Rate_Select`

Input:

- Music detection flag, F_{MUS}
- SMV mode, M_{SMV}
- Frame class decision (refined), C_{FR}, and previous frame class decision, C_{FR}^{prv}
- Open-loop normalized pitch correlations, $R_p^{ol}(l)$, $l = 0,1,2$
- Noise-to-signal ratio, NSR
- Sharpness of weighted residual, S_{rp}^{wr}, (Section 5.3.7)
- Energy of the weighted residual given by $E_{rw}(0) + E_{rw}(1)$, (Section 5.3.7)
- First element of third reflection coefficient vector, $\hat{k}_2(0)$
- Onset flag, F_{onset}

Output:

- SMV rate, $Rate$

5.3.13.1 Rate-Selection Algorithm

For each frame, the rate selection algorithm selects one of the 4 possible rates, Rate 1, Rate 1/2, Rate 1/4, or Rate 1/8. The selection of the rate is controlled by the SMV operating mode, which can be Mode 0 (Premium), Mode 1 (Standard), Mode 2 (Economy), or Mode 3 (Max-Capacity). The rate selection algorithm uses previously computed parameters and a set of fixed and adaptive thresholds for initial selection of the rate for each frame. The initial rate selection is later refined (Section 5.3.16).

The rate-selection algorithm sets the rate to Rate 1 if the music flag indicates a music signal. Otherwise, the algorithm selects the coding rate for the frame according to a set of fixed thresholds and conditions, where the set of thresholds and conditions depends on the dictated mode of operation for the SMV.

An initial selection of Rate 1 can be forced to Rate 1/2 by the half-rate max flag, F_{HRM}.

5.3.14 Open-Loop Pitch Quantization, Pitch Adjustment and Interpolation

If the auxiliary classification decision $C_{SM} > 0$, the open loop $l_p^{ol}(1)$ has to be quantized, adjusted and interpolated before the modified weighted speech can be generated as described in the next section. Before the quantization begins, if $l_p^{ol}(1) > 120$ and the Rate is not 1, C_{SM} will be set to zero and the frame will be classified into a type-0 frame and hence, be excluded from this quantization process. The quantization of $l_p^{ol}(1)$ is performed by searching through a pitch codebook of size 256 for Rate 1 and and of size 128 for other rates. To avoid an exhaustive search, the search range is limited to $\left[l_p^{low}, l_p^{high}\right]$ where

$$l_p^{low} = \max\left\{l_p^{ol}(1) - 5, 17\right\} \tag{5.3.14-1}$$

$$l_p^{high} = \begin{cases} \min\left\{l_p^{ol}(1) + 5, 148\right\} & Rate \text{ is Rate 1} \\ \min\left\{l_p^{ol}(1) + 5, 120\right\} & \text{otherwise} \end{cases} \tag{5.3.14-2}$$

Since the pitch table is not necessarily expressed in integer values, the search requires to upsample the normalized pitch correlation in a bandlimited fashion. As a result, the quantized pitch is chosen to be the codeword that maximizes the upsampled correlations. Based on this pitch, a pitch contour is constructed and will be used by the signal modification algorithm (Section 5.3.15). The modification algorithm attempts to modify the speech signal to match this pitch contour. If the matching is successful, the frame is classified into type 1, and type 0 otherwise. For type-1 frames, the quantized pitch value obtained by this procedure is not further modified.

5.3.14.1 Open-Loop Pitch Refinement and Quantization

Routine name: `PIT_FinePitch`

Input:

- SMV rate, $Rate$
- Pitch search range, $\left[l_p^{low}, \ldots, l_p^{high}\right]$
- Weighted speech, $s_w(n)$
- Signal modification accumulated delay, Δ_{SM}

Output:

- Quantized open-loop pitch lag at the end of the encoding frame, l_{pp}^q
- Integer part of the quantized open-loop pitch lag at the end of the encoding frame, $l_{pp,INT}^q$

The pitch refinement search starts with calculating the normalized pitch correlation on the weighted speech according to

$$R_p(l) = \frac{\sum_{n=0}^{79} s_w(n) \cdot s_w(n-l)}{\sqrt{\sum_{n=0}^{79} s_w^2(n) \cdot \sum_{n=0}^{79} s_w^2(n-l)}} \qquad l = l_p^{low} - 6, \ldots, l_p^{high} + 6 \tag{5.3.14.1-1}$$

and the optimal integer lag l_p^* is found by

$$l_p^* = \arg\max_{l=l_p^{low},\ldots,l_p^{high}} R_p(l) \tag{5.3.14.1-2}$$

Note that the ± 6 samples are needed for the upsampling of $R_p(\cdot)$ during the fractional search in the next step. After obtaining the optimal integer lag l_p^*, the fractional search begins. The search interval is $[l_p^* - 1, \ldots, l_p^* + 1]$ and $R_p(\cdot)$ is upsampled at all possible fractional pitch values within that interval. The upsampling is bandlimited and uses a series of pre-stored Sinc tables. The lag that maximizes the interpolated normalized correlation is chosen. The corresponding codebook index is denoted by I_p.

Next, the quantized pitch lag is adjusted to reduce the risk of reaching the delay limits of the signal modification algorithm, since the allowed level for the accumulated delay is $\Delta_{SM} \le 20$. The adjustment is carried out by modifying the index of the pitch quantization, I_p, according to:

$$I_p = \begin{cases} I_p + 1 & \text{Rate 1 and } \Delta_{SM} > 10 \\ I_p + 2 & \text{Rate 1 and } \Delta_{SM} > 15 \\ I_p - 1 & \text{Rate 1 and } \Delta_{SM} < -10 \\ I_p - 2 & \text{Rate 1 and } \Delta_{SM} < -15 \\ I_p + 1 & \text{Rate } \tfrac{1}{2} \text{ and } \Delta_{SM} > 10 \\ I_p - 1 & \text{Rate } \tfrac{1}{2} \text{ and } \Delta_{SM} < -10 \end{cases} \tag{5.3.14.1-3}$$

After the modification, attention is paid to ensure the modified I_p is still a valid codebook index (i.e., I_p cannot be negative or exceed the codebook size). The final pitch value, l_{pp}^q is obtained from the quantization table using the modified index. The $l_{pp,INT}^q$ is simply the rounded version of l_{pp}^q.

5.3.14.2 Pitch Interpolation

If $C_{SM} > 0$, pitch interpolation path, $Pit(n)$, is constructed, using the open-loop quantized pitch lags from previous two frames, denoted by $l_{pp}^q(frm-1)$ and $l_{pp}^q(frm-2)$ for the

previous frame and the one before, respectively. If the previous rate is Rate 1/4 or Rate 1/8, these lags are set to $l_{pp}^{q}(frm-1) = l_{pp}^{q}(frm-2) = l_{pp}^{q}$. Otherwise, if the previous frame was of type 0, $l_{pp}^{q}(frm-1) = l_{pp}^{q}(frm-2)$, and both are set to the closed-loop pitch lag of the last subframe of the calculated in Sections 5.6.6 and 5.6.7.

Routine name: `PIT_PitchInterpolat`

Input:

- Quantized open-loop pitch lag at the end of the encoding frame, l_{pp}^{q}
- Open-loop pitch lags from previous two frames, $l_{pp}^{q}(frm-1)$ and $l_{pp}^{q}(frm-2)$

Output:

- Quantized Interpolated pitch contour, $Pit(n)$

Before the pitch is interpolated, a mean pitch value is first obtained by:

$$\bar{l}_{pp}^{q} = 0.5 \cdot \left(l_{pp}^{q}(frm-2) + l_{pp}^{q} \right) \tag{5.3.14.2-1}$$

The relationships between $l_{pp}^{q}(frm-1)$, l_{pp}^{q}, and $\bar{l}_{pp}^{q}$ are tested, and if these values are close to each other, the pitch contour is set according to:

$$Pit(n) = \begin{cases} \dfrac{n \cdot l_{pp}^{q} + \left(L_{frm} - n\right) \cdot l_{pp}^{q}(-1)}{L_{frm}} & n = 0, \ldots, L_{frm} - 1 \\ l_{pp}^{q} & n = L_{frm}, \ldots, L_{frm} + L_{lkha} - 1 \end{cases} \tag{5.3.14.2-2}$$

Otherwise, the pitch contour is not interpolated and is set to:

$$Pit(n) = l_{pp}^{q} \qquad n = 0, \ldots, L_{frm} + L_{lkhd} - 1 \tag{5.3.14.2-3}$$

5.3.15 Generating Modified Weighted Speech Signal

Generate a modified weighted speech signal, $s_{mw}(n)$, by warping the current weighted speech signal, $s_w(n)$, to match the pitch contour, $Pit(n)$, for the current frame.

Input: The inputs to the weighted speech modification routine are:

- M_{SMV} : selected SMV mode
- $Rate$: fixed codec bit rate
- F_{flat} : spectral flatness flag
- NSR : background noise to the signal ratio
- $s_w(n)$: current input weighted speech signal
- $Pit(n)$: pitch track for the current frame
- C_{SM} : signal modification class

Output: The outputs from the weighted speech modification routine are:

- $s_{mw}(n)$: modified weighted speech signal
- Δ_{SM} : signal modification accumulated delay
- C_{FR} : frame class 6 determination that defines Type 1
- C_{VUV} : voiced/unvoiced level (updated)
- $g_a(m)$: pitch gain (adaptive-codebook gain) for each subframe (indexed by m) of stationary-voiced frame (class 6)

5.3.15.1 Frame based processing for $C_{SM} \le 0$

For these two classes of C_{SM}, $(C_{SM} \le 0)$, the accumulated delay will be set to zero or kept unchanged.

5.3.15.1.1 Region of Mapping of Weighted Speech to Modified Weighted Speech

If $C_{SM} = -1$ reset the accumulated delay:

$$\Delta_{SM} = 0 \tag{5.3.15.1-1}$$

Calculate the starting time point T_0 and the ending time point T_1 of the weighted speech signal to be mapped to the current frame of the modified speech signal:

$$T_0 = L_{olpit} - L_{lkhd} - L_{frm} + \Delta_{SM} \quad (5.3.15.1\text{-}2)$$

$$T_1 = T_0 + L_{frm} \quad (5.3.15.1\text{-}3)$$

5.3.15.1.2Weighted Speech Mapping to Modified Weighted Speech for One Frame

Since the delay in the region $[T_0, T_1]$ is 0 or constant, only one $Sinc$ table based on the fractional value of the delay Δ_{SM} is needed to do the signal mapping:

$$s_{mw1}(n) = \sum_{i=-10}^{10} w_s\left(f(\Delta_{SM}), i\right) \cdot s_w\left(n + i + I(T_0)\right), \qquad n = 0, \ldots, L_{frm} - 1 \quad (5.3.15.1\text{-}4)$$

where $f(\circ)$ takes the fractional part, $I(\circ)$ takes the integer part, and $w_s(f, i)$ is the Hamming weighted $Sinc$ window.

5.3.15.1.3Update of Modified Weighted Speech Buffer

Shift the old buffer:

$$s_{mw\_buff}(n) \Leftarrow s_{mw\_buff}(n + L_{frm}) \ , \quad 0 \le n < L_{mw\_buff} - L_{frm} \quad (5.3.15.1\text{-}5)$$

where $s_{mw\_buff}(n), \ n = 0, \ldots, L_{mw\_buff} - 1$ is the buffer vector of the modified weighted speech signal. Then, add the current modified weighted speech signal to the buffer:

$$s_{mw\_buff}(n + L_{mw\_buff} - L_{frm}) = s_{mw}(n) \ , \quad 0 \le n < L_{frm} \quad (5.3.15.1\text{-}6)$$

5.3.15.2 Variable Subframe Processing $C_{SM} > 0$

For $C_{SM} > 0$ the analysis and the modification of the weighted voiced speech is conducted on a variable subframe basis. The subframe size, L_{SF}, is related to pitch lag value and the location of the subframe within a frame. For each subframe, a local delay, D_L, at the end of the subframe is estimated in a fast way by maximizing the correlation between a current subframe and a modified target signal. A continuous time warping of the signal as opposed to simple shifting is then performed with a variable local delay from 0 to the estimated local delay D_L. The absolute value of the accumulated delay is limited a maximum 20 samples (2.5 ms). As a portion of the results, the estimated open-loop pitch gains and the normalized pitch correlation values will be provided.

Initialize parameters:

- Set the starting point of the first subframe within a frame to zero, $i_0 = 0$
- Set the pitch gain vector to zero, $g_a(l) = 0, \ l = 0,1,2,3$

- Set the pitch correlation vector to zero, $R_{SM}(l) = 0,\ l = 0,1,2,3$
- Set the average pitch correlation value to zero, $R_{SM\_avg} = 0$
- Set the minimum pitch correlation value to 1.0, $R_{SM\_\min} = 1.0$

5.3.15.2.1 Determining Parameters

Before searching for a local delay for each subframe, some parameters need to be estimated.

Calculate the pulse length (pulse width):

$$L_{puls} = \min\{\max\{0.2 \cdot Pit(i_0), 5\}, 30\} \tag{5.3.15.2-1}$$

Calculate the starting time, T_0, of the current subframe on the current weighted speech signal, s_w:

$$T_0 = L_{olpit} - L_{lkhd} - L_{frm} + i_0 + \Delta_{SM} \tag{5.3.15.2-2}$$

where Δ_{SM} is the accumulated delay for the last subframe.

Define the pulse peak location searching range $[T_0, T_2]$ for the current subframe, where the end time point, T_2, is calculated as:

$$T_2 = \min\{T_0 + Pit(i_0), L_{olpit} - 12\} \tag{5.3.15.2-3}$$

The pulse peak location, T_{peak}, is evaluated by maximizing the peak energy as a function of the location n:

$$E_{peak}(n) = \sum_{k=-2}^{2} s_w(n+k)^2 \quad , \quad T_0 \le n \le T_2, \tag{5.3.15.2-4}$$

That is,

$$E_{peak}(T_{peak}) = \max\{E_{peak}(n) :\ n = T_0, \ldots, T_2\} \tag{5.3.15.2-5}$$

From $n = T_0$ to $n = T_2$ the average peak energy is estimated as:

$$E_{avg}(T_0) = E_{peak}(T_0) \tag{5.3.15.2-6}$$

$$\text{if } E_{peak}(n) < 4 \cdot E_{avg}(n-1)$$
$$E_{avg}(n) = 0.875 \cdot E_{avg}(n-1) + 0.125 \cdot E_{peak}(n) \quad ; n = T_0 + 1, \ldots, T_2 \tag{5.3.15.2-7}$$

After obtaining the maximum peak energy and the average peak energy, an average energy to peak energy ratio is calculated by

$$APR = \frac{E_{avg}(T_2)}{E_{peak}(T_{peak})} \tag{5.3.15.2-8}$$

For the first subframe, some protection for evaluating the peak location is set by

1 if $i_0 = 0$ and $T_{peak} < T_0 + L_{puls}$

$$T_{peak} = T_0 + L_{puls} \quad (5.3.15.2\text{-}9)$$

2 Based on the parameters obtained above, the current subframe size, L_{sub}, and the
3 extended subframe size, L_{ex_sub}, are calculated by

4 $$L_{sub} = \min\{T_{peak} - T_0 + L_{puls}\ ,\ L_{frm} - i_0\} \quad (5.3.15.2\text{-}10)$$

5 $$L_{ex\_sub} = T_{peak} - T_0 + 2 \cdot L_{puls} \quad (5.3.15.2\text{-}11)$$

6 If $T_{peak} + 1.25 \cdot Pit(i_0) > L_{olpit} - L_{lkhd} + \Delta_{SM}$, the next possible peak location is evaluated in
7 order to judge whether the current subframe must include the remaining samples up to the
8 end of the current frame. The next peak is estimated by using the same approach as
9 described above with a new searching range of $[\underline{T}_0, \underline{T}_2]$:

10 $$\underline{T}_0 = T_{peak} + 0.5 \cdot Pit(i_0) \quad (5.3.15.2\text{-}12)$$

11 $$\underline{T}_2 = \min\{\underline{T}_0 + Pit(i_0), L_{olpit} - 12\} \quad (5.3.15.2\text{-}13)$$

12 If the newly found peak location is beyond the end of the current frame,

13 $$T'_{peak} > L_{oppit} - L_{lkhd} + \Delta_{SM} \quad (5.3.15.2\text{-}14)$$

14 The current subframe size and the extended subframe size are modified according to:

15 $$L_{sub} = L_{frm} - i_0 \quad (5.3.15.2\text{-}15)$$

16 $$L_{ex\_sub} = \max\{L_{sub},\ T'_{peak} - \underline{T}_0 + 2 \cdot L_{puls}\} \quad (5.3.15.2\text{-}16)$$

17 $$L_{ex\_sub} \Leftarrow \min\{\min\{\ L_{ex\_sub}\ ,\ L_{frm}\ - i_0\ + \max\{L_{lkhd} - \Delta_{SM} - 15,\ 0\}\},\ L_{max\_ex\_sub}\}$$

18 (5.3.15.2-17)

19 Each of the extended subframe sizes is always limited to $L_{\max\_ex\_sub} = 180$.

20 Each of the defined subframes is split into two regions where the continuous time warping
21 will be applied on the first region and the time shifting will be performed on the second
22 region. The length of the first region is defined as

23 $$L_{center} = \min\left\{L_{sub}, \max\left\{T_{peak} - T_0 - L_{puls}, {}^{L_{puls}}\!/_{2}\right\}\right\} \quad (5.3.15.2\text{-}18)$$

24

25 5.3.15.2.2Target Signal Determination for Current Local Delay Search

26 The target signal for the current local delay searching is determined according to the buffer
27 of the modified weighted speech signal and the current pitch track function.

28 Create the current pitch track function:

29 $$PitF(n) = Pit(i_0 + n) \qquad n = 0, \ldots, L_{tmp} - 1 \quad (5.3.15.2\text{-}19)$$

$$PitF(n) = PitF(L_{tmp} - 1) \qquad n = L_{tmp}, \ldots, L_{ex\_sub} - 1 \tag{5.3.15.2-20}$$

where L_{tmp} is a temporal parameter:

$$L_{tmp} = \min\left\{L_{ex\_sub}, L_{frm} - i_0 + L_{frm}/2\right\} \tag{5.3.15.2-21}$$

Calculate the target signal and memorize it in a temporal buffer extended from the past modified weighted speech signal:

$$s_{mw\_buff}\left(L_{mw\_buff} + n\right) = \sum_{i=-10}^{10} w_s\left(f\left(PitF(n)\right), i\right) \cdot s_{mw\_buff}\left(L_{mw\_buff} + n - I\left(PitF(n)\right) + i\right) \qquad n = 0, \ldots, L_{ex\_sub} - 1 \tag{5.3.15.2-22}$$

In order to simplify later computation, copy the target signal into a target signal vector:

$$T_{G\_SM}(n) = s_{mw\_buff}\left(L_{mw\_buff} + n\right) \quad n = 0, \ldots, L_{ex\_sub} - 1 \tag{5.3.15.2-23}$$

5.3.15.2.3 Searching Range Determination for Local Delay

The local delay search range is limited by the positive boundary SR_1 and the negative boundary SR_0. Two APR values that could influence the searching range are calculated first in the same way as described from by Equation (5.3.15.2-4) to (5.3.15.2-8). APR_1 is based on the weighted speech in the region $\left[\underline{T}_0, \underline{T}_2\right]$ where

$$\underline{T}_0 = T_{peak} - 0.5 \cdot Pit(i_0) \tag{5.3.15.2-24}$$

$$\underline{T}_2 = \min\left\{\underline{T}_0 + Pit(i_0), L_{olpit} - 12\right\} \tag{5.3.15.2-25}$$

APR_2 is measured on the target signal $T_{G\_SM}(n)$ with $3 \le n \le L_{ex\_sub} - 3$. The searching range will be initialized first and then some modification on it could be performed.

Initialize the boundaries:

$$SR_1 = I\left(\max\left\{\min\left\{0.075 \cdot L_{center} + 0.5, 5\right\}, 1\right\}\right) \tag{5.3.15.2-26}$$

$$SR_0 = SR_1 \tag{5.3.15.2-27}$$

Let $L_{tmp} = 4$ if $C_{SM} \ge 3$ and $L_{tmp} = 3$ otherwise. Check the influence from the two APR values by:

1
$$\begin{aligned}
&\text{if } APR_1 > 1/4 \\
&\quad SR_1 = \min\{SR_1, 3\} \\
&\quad SR_0 = \max\{SR_0, -3\} \\
&\text{else if } APR_1 > 1/12 \\
&\quad SR_1 = \min\{SR_1, 4\} \\
&\quad SR_0 = \max\{SR_0, -4\}
\end{aligned} \qquad \text{(5.3.15.2-28)}$$

2
$$\begin{aligned}
&\text{if } APR_2 > 1/3 \\
&\quad SR_1 = \min\{SR_1, L_{tmp}\} \\
&\quad SR_0 = \max\{SR_0, -L_{tmp}\} \\
&\text{else if } APR_2 > 1/6 \\
&\quad SR_1 = \min\{SR_1, L_{tmp} + 1\} \\
&\quad SR_0 = \max\{SR_0, -L_{tmp} - 1\}
\end{aligned} \qquad \text{(5.3.15.2-29)}$$

3 Further consider the influence of the subframe size. If $L_{ex\_sub} < T_{peak} - T_0 - 2$ or
4 $Pit(i_0) < 25$ or $T_{peak} \le T_{puls} + T_0$:

5
$$\begin{aligned}
&\text{if } APR_2 > 1/4 \\
&\quad SR_1 = \min\{SR_1, 1\} \\
&\quad SR_0 = \max\{SR_0, -1\} \\
&\text{else} \\
&\quad SR_1 = \min\{SR_1, 2\} \\
&\quad SR_0 = \max\{SR_0, -2\}
\end{aligned} \qquad \text{(5.3.15.2-30)}$$

6 Further consider the influence from the accumulated delay:

7
$$\begin{aligned}
&\text{if } \Delta_{SM} > 5 \\
&\quad SR_1 = \min\{SR_1, 4\} \\
&\text{if } \Delta_{SM} > 10 \\
&\quad SR_1 = \min\{SR_1, 2\} \\
&\text{if } \Delta_{SM} > 15 \\
&\quad SR_1 = \min\{SR_1, 0\}
\end{aligned} \qquad \text{(5.3.15.2-31)}$$

8

9
$$\begin{aligned}
&\text{if } \Delta_{SM} < -5 \\
&\quad SR_0 = \max\{SR_0, -4\} \\
&\text{if } \Delta_{SM} < -10 \\
&\quad SR_0 = \max\{SR_0, -2\} \\
&\text{if } \Delta_{SM} < -15 \\
&\quad SR_0 = 0
\end{aligned} \qquad \text{(5.3.15.2-32)}$$

5.3.15.2.4Local Delay Search

This section will be divided into (1) modification of the target signal, (2) searching for an integer part of the local delay, (3) searching for a fractional part of the local delay, and (4) generating a warped weighted-speech signal.

5.3.15.2.4.1 Modification of Target Signal

The target signal is modified in time domain before maximizing the correlation between the target signal and the current extended subframe by applying a weighting window:

$$\text{if } C_{SM} \geq 3$$

$$T_{G\_SM\_M}(n) = \left(0.25 + 0.75\frac{n}{L_{center}}\right) \cdot T_{G\_SM}(n)$$

$$\text{else} \qquad (5.3.15.2\text{-}33)$$

$$T_{G\_SM\_M}(n) = \left(0.5 + 0.5\frac{n}{L_{center}}\right) \cdot T_{G\_SM}(n) \qquad ;0 \leq n < L_{center}$$

$$T_{G\_SM\_M}(n) = T_{G\_SM}(n) \qquad ;L_{center} \leq n < L_{ex\_sub} - 1$$

where $T_{G\_SM\_M}$ is the modified target signal.

5.3.15.2.4.2 Integer Local Delay Search

In order to calculate the correlation values between the target signal and the current signal with different integer local delays, a signal keeping the latest accumulated delay with zero local delay is calculated first by mapping the weighted speech signal from the region $[TT_0, TT_1]$ to a signal vector $s_{w\_shift}(n), \quad n = 0, \ldots, TT_1 - TT_0 - 1$

$$s_{w\_shift}(n) = \sum_{i=-10}^{10} w_s\left(f(TT_0), i\right) \cdot s_w\left(n + I(TT_0) + i\right), \qquad ;n = 0, \ldots, TT_1 - TT_0 - 1 \qquad (5.3.15.2\text{-}34)$$

where

$$TT_0 = T_0 + SR_0 - 5 \qquad (5.3.15.2\text{-}35)$$

$$TT_1 = T_0 + SR_1 + 5 + L_{ex\_sub} + 1 \qquad (5.3.15.2\text{-}36)$$

Then, the correlation value, $R_{P\_SM\_shift}\left(k - \left(SR_0 - 5\right)\right)$, on the integer local delay of $SR_0 - 5 \leq k \leq SR_1 + 5$, is calculated by

$$R_{P\_SM\_shift}(k - (SR_0 - 5)) = \frac{\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM\_M}(n) s_{w\_shift}(n + k - (SR_0 - 5))}{\sqrt{\sum_{n=0}^{L_{ex\_sub}-1} T^2{}_{G\_SM\_M}(n) \sum_{n=0}^{L_{ex\_sub}-1} s^2{}_{w\_shift}(n + k - (SR_0 - 5))}} \qquad (5.3.15.2\text{-}37)$$

The optimal integer local delay, k_{opt}, and the corresponding correlation value are obtained by maximizing the above correlation with a minimum value condition:

if $C_{SM} \geq 3$

$$R_{P\_SM\_shift}\left(k_{opt}-(SR_0-5)\right)=\max\left\{R_{P\_SM\_shift}\left(k-(SR0-5)\right)>0.3\ ,\ SR_0 \leq k \leq SR_0\right\}$$

else

$$R_{P\_SM\_shift}\left(k_{opt}-(SR_0-5)\right)=\max\left\{R_{P\_SM\_shift}\left(k-(SR0-5)\right)>0.4\ ,\ SR_0 \leq k \leq SR_0\right\} \quad \text{(5.3.15.2-38)}$$

5.3.15.2.4.3 Fractional Local Delay Determination

All the possible fractional local delays are generated and memorized in a table, $Tab(i)$,

$$Tab(i) = k_{opt} - 0.75 + 0.1 \cdot i \qquad i = 0,\ldots,15$$

To find a best fractional delay, the correlation with integer delays is up-sampled for each fractional delay candidate:

$$R_{P\_SM\_F}\left(Tab(i)\right)=\sum_{j=-4}^{4} w_s\left(f\left(Tab(i)\right),\ j\right)\cdot R_{P\_SM\_shift}\left(I\left(Tab(i)\right)+\ j\ -\ (SR_0-5)\ \right) \quad \text{(5.3.15.2-39)}$$

The best fractional local delay, D_L, and the corresponding correlation value, $R_{P\_SM\_opt}$, will be found by maximizing the up-sampled correlation:

$$\begin{aligned} R_{P\_SM\_opt} &= R_{P\_SM\_F}(D_L) \\ &= \max\{\ R_{P\_SM\_F}(Tab(i)),\ \ i=0,1,....,15\ \} \end{aligned} \quad \text{(5.3.15.2-40)}$$

5.3.15.2.4.4 Warping Current Weighted Speech With Determined Delay

After finding the local delay, the current subframe is warped from the original weighted speech signal to a temporal modified weighted speech signal vector:

$$s'_{mw}(n)=\sum_{i=-10}^{10} w_s\left(f\left(T_0+D_L\cdot n/L_{center}\right),\ i\right)\cdot s_w\left(n+I\left(T_0+D_L\cdot n/L_{center}\right)+i\right),$$

$$;n=0,\ldots,L_{center}-1$$

$$s'_{mw}(n)=\sum_{i=-10}^{10} w_s\left(f\left(T_0+D_L\right),i\right)\cdot s_w\left(n+I\left(T_0+D_L\right)+i\right), \quad \text{(5.3.15.2-41)}$$

$$;n=L_{center},....,L_{ex\_sub}-1$$

5.3.15.2.5Further Modification of Weighted Speech for Half_Rate_Max

If the maximum rate is limited to the Rate 1/2, further signal modification could be performed on some specific transition areas by smoothing the signal with waveform interpolation.

5.3.15.2.5.1 Generating a Backward Waveform Vector

Calculate two conditions for the possible further signal modification:

$$Cond_1 = 0.25 < R_{P\_SM\_opt} < 0.7 \text{ and } Pit(i_0) > 25 \text{ and } L_{sub} > \max\{0.9 \cdot Pit(i_0), 17\} \quad \text{(5.3.15.2-42)}$$

$$Cond_2 = I(T_0 + \Delta_{SM} + Pit(i_0) + L_{ex\_sub} + 17) > L_{olpit} \quad \text{(5.3.15.2-43)}$$

If both conditions are met, calculate a following reference signal for estimating a backward waveform:

$$\begin{aligned} T'_{G\_SM\_M}(n) &= 0.8\ (0.5 + 0.5\, n / L_{center})\ s'_{mw}(n) \quad , \qquad n = 0, \ldots, L_{center} - 1 \\ T'_{G\_SM\_M}(n) &= s'_{mw}(n) \quad , \qquad n = L_{center}, \ldots, L_{ex\_sub} - 1 \end{aligned} \quad \text{(5.3.15.2-44)}$$

Then, the current backward waveform will be generated by taking a future signal vector in the interval $\lfloor TT_0 + D_{shift}\ ,\ TT_0 + D_{shift} + L_{ex\_sub} \rfloor$ of the weighted speech signal s_w, where

$$TT_0 = T_0 + \Delta_{SM} + Pit(i_0) \quad \text{(5.3.15.2-45)}$$

The shifting parameter, $-3 \le D_{shift} \le 3$, is optimized by maximizing the correlation between the future signal vector and the reference signal defined in Equation (5.3.15.2-44). The same searching procedure as described from the Sections 5.3.15.2.4.2 to 5.3.15.2.4.4 is used to obtain D_{shift} and the corresponding correlation value R_{P_shift} by setting $SR_0 = -3$, $SR_1 = 3$, and replacing $T_{G\_SM\_M}$ with the reference signal $T'_{G\_SM\_M}$ defined by Equation (5.3.15.2-44).

The future signal vector in the above determined area will be mapped to the backward waveform vector, s_{w_bkwd}, if the following condition is satisfied:

$$Cond_3 = (C_{SM} \ge 4 \text{ and } R_{P\_shift} > 0.25) \text{ or } R_{P\_shift} > 0.4 \quad \text{(5.3.15.2-46)}$$

If $Cond_3$ is met,

$$s_{w\_bkwd}(n) = \sum_{i=-10}^{10} w_s(f(TT_0 + D_{shift}), i) \cdot s_w(n + I(TT_0 + D_{shift}) + i) \qquad n = 0, \ldots, L_{ex\_sub} - 1 \quad \text{(5.3.15.2-47)}$$

5.3.15.2.5.2 Generating a Candidate Vector by Waveform Interpolation

If $Cond_1$ and $Cond_3$ are met, or $R_{P\_SM\_opt} > 0.4$, the following waveform interpolation will be performed to generate a candidate vector to possibly replace the modified weighted speech signal for the current variable subframe.

Generate a temporal vector:

if $Cond_3$

$$s_{WI}(n) = T_{G\_SM}(n) + s_{w\_bkwd}(n) \qquad n=0,\ldots,L_{ex\_sub}-1$$

else

$$s_{WI}(n) = T_{G\_SM}(n) \quad , \qquad n=0,\ldots,L_{ex\_sub}-1 \tag{5.3.15.2-48}$$

where the target signal T_{G_SM} acts as the forward waveform. The energy of the vector is normalized by a gain factor G_{tmp}:

$$G_{tmp} = \sqrt{\frac{\sum_{n=0}^{L_{ex\_sub}-1} s'_{mw}(n)^2}{\sum_{n=0}^{L_{ex\_sub}-1} s_{WI}(n)^2}} \tag{5.3.15.2-49}$$

$$s_{WI}(n) \Leftarrow s_{WI}(n) \cdot G_{tmp} \quad , \quad n=0,\ldots,L_{ex\_sub} \tag{5.3.15.2-50}$$

Set a length of an interpolation region:

if $C_{SM} \geq 3$

$$L_{tmp} = L_{sub} / 2$$

else

$$L_{tmp} = L_{sub} / 3 \tag{5.3.15.2-51}$$

Perform the interpolation at the beginning of the subframe:

$$s_{WI}(n) \Leftarrow (n \cdot s_{WI}(n) + (L_{tmp}-n) \cdot s'_{mw}(n)) / L_{tmp} \quad , \qquad n=0,1,\ldots\ldots,L_{tmp}-1 \tag{5.3.15.2-52}$$

Perform the interpolation at the end of the subframe:

$$s_{WI}(L_{sub}-n) \Leftarrow (n \cdot s_{WI}(L_{sub}-n) + (L_{tmp}-n) \cdot s'_{mw}(L_{sub}-n)) / L_{tmp} \quad , \quad n=1,2,\ldots.L_{tmp} \tag{5.3.15.2-53}$$

$$s_{WI}(n) = s'_{mw}(n) \, , \qquad n=L_{sub},\ldots\ldots,L_{ex\_sub}-1 \tag{5.3.15.2-54}$$

5.3.15.2.5.3 Making a Decision to Use Interpolated Modified Weighted Speech

If $Cond_1$ and $Cond_3$ are met, or $R_{P\_SM\_opt} > 0.4$ calculate the normalized correlation between the original modified target signal and the interpolated one:

$$R_{P\_tmp} = \frac{\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM\_M}(n) \cdot s_{WI}(n)}{\sqrt{\left(\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM\_M}(n)^2\right)\left(\sum_{n=0}^{L_{ex\_sub}-1} s_{WI}(n)^2\right)}} \tag{5.3.15.2-55}$$

The decision is made by:

$$
\begin{aligned}
&if\ \ (R_{P\_tmp} > 1.1\ \ R_{P\_SM\_opt})\ \{ \\
&\qquad R_{P\_SM\_opt} = R_{P\_tmp} \\
&\qquad s'_{mw}(n) = s_{WI}(n)\ , \quad n = 0,1,.....,L_{ex\_sub} - 1 \\
&\}
\end{aligned}
\tag{5.3.15.2-56}
$$

5.3.15.2.6Final Determination of Current Local Delay and Accumulated Delay

Correct the local delay:

$$
\begin{aligned}
&If\ \ (L_{center} > L_{sub}) \\
&\qquad D_L \Leftarrow D_L \cdot L_{sub} / L_{center}
\end{aligned}
\tag{5.3.15.2-57}
$$

If the accumulated delay is larger than the maximum limit, the local delay will be set to zero and the accumulated delay will be kept the same:

$$
\begin{aligned}
&If\ \ (\left|\Delta_{SM} + D_L\right| > 20), \\
&\qquad s'_{mw}(n) = \sum_{i=-10}^{10} w_s(f(T_0), i) \cdot s_w(I(T_0) + n + i), \qquad n = 0,1,.....,L_{ex\_sub} - 1 \\
&else \\
&\qquad \Delta_{SM} \Leftarrow \Delta_{SM} + D_L
\end{aligned}
\tag{5.3.15.2-58}
$$

5.3.15.2.7Updating Modified Weighted Speech Buffer

Shift the old buffer:

$$
s_{mw\_buff}(n) \Leftarrow s_{mw\_buff}(n + L_{sub})\ , \quad 0 \le n < L_{mw\_buff} - L_{sub}\ ,
\tag{5.3.15.2-59}
$$

where $s_{mv\_buff(n)} \quad n = 0,\ldots,L_{mw\_buff} - 1$, is the buffer vector of the modified weighted speech. Then, add the current modified weighted speech signal to the buffer:

$$
s_{mw\_buff}(n + L_{mw\_buff} - L_{sub}) = s'_{mw}(n)\ , \quad 0 \le n < L_{sub}
\tag{5.3.15.2-60}
$$

5.3.15.2.8Pitch Pre-Enhancement and Periodicity Detection

The pitch periodicity of the modified original signal is pre-enhanced according to different conditions. The pitch gain and the normalized pitch correlation are also estimated.

Calculate the normalized correlation between the current modified weighted speech signal and the signal modification target:

$$R_{P\_SM} = \frac{\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM}(n) \cdot s'_{mw}(n)}{\sqrt{\left(\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM}(n)^2\right)\left(\sum_{n=0}^{L_{ex\_sub}-1} s'_{mw}(n)^2\right)}} \quad (5.3.15.2\text{-}61)$$

Calculate the open-loop pitch gain:

$$G_{P\_SM} = \frac{\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM}(n) \cdot s'_{mw}(n)}{\sum_{n=0}^{L_{ex\_sub}-1} T_{G\_SM}(n)^2} \quad (5.3.15.2\text{-}62)$$

If $R_{P\_SM} > 0.5$ the pitch pre-enhancement will be applied and the energy will be normalized. The energy of the modified signal is saved first before adding the pre-enhancement:

$$E_{tmp} = \sum_{n=0}^{L_{ex\_sub}-1} s'_{mw}(n)^2 \quad (5.3.15.2\text{-}63)$$

Apply the pre-enhancement:

$$s'_{mw}(n) \Leftarrow s'_{mw}(n) + C_{tmp} \cdot T_{G\_SM}(n), \qquad n = 0,\ldots,L_{ex\_sub}-1 \quad (5.3.15.2\text{-}64)$$

where the enhancement coefficient C_{tmp} is

$$\begin{aligned} &if\ M_{SMV} \geq 1 \\ &\quad C_{tmp} = \min\{4\,NSR + 0.25,\ 0.4\} \cdot \min\{G_{P\_SM},\ 1.0\} \\ &else \\ &\quad C_{tmp} = \min\{4\,NSR + 0.125,\ 0.25\} \cdot \min\{G_{P\_SM},\ 1.0\} \end{aligned} \quad (5.3.15.2\text{-}65)$$

Calculate the energy normalization gain:

$$G_{tmp} = \sqrt{\frac{E_{tmp}}{\sum_{n=0}^{L_{ex\_sub}-1} s'_{mw}(n)^2}} \quad (5.3.15.2\text{-}66)$$

Apply the energy gain normalization:

$$s'_{mw}(n) \Leftarrow G_{tmp} \cdot s'_{mw}(n), \qquad n = 0,1,2,\ldots\ldots,L_{ex\_sub}-1 \quad (5.3.15.2\text{-}67)$$

Then, copy the temporal modified signal to the modified signal vector:

$$s_{mw}(i_0 + n) = s'_{mw}(n), \qquad n = 0,\ldots,L_{sub}-1 \quad (5.3.15.2\text{-}68)$$

After obtaining the above information, the temporal class information C_{VUV} could be corrected under some conditions:

$$\begin{aligned} &if \quad ((APR_1 < 1/30 \ \ or \ \ G_{P\_SM} > 0.7 \ or \ R_{P\_SM} > 0.6) \ and \\ &\quad (L_{ex\_sub} > T_{peak} - T_0 + 2) \ and \ (C_{VUV} = 2)\ ) \\ &C_{VUV} = 3 \end{aligned} \tag{5.3.15.2-69}$$

Before finishing the subframe based processing for the current variable subframe, some temporal pre-parameters are calculated and memorized as described below. These parameters obtained from the current variable subframe will be later converted for the fixed subframe.

Update the minimum pitch correlation:

$$\begin{aligned} &if \ \ (R_{P\_SM} < R_{SM\_min}), \\ &\quad R_{SM\_min} = R_{P\_SM} \end{aligned} \tag{5.3.15.2-70}$$

Pre-calculate the average pitch correlation:

$$R_{SM\_avg} \Leftarrow R_{SM\_avg} + R_{P\_SM} \cdot L_{sub} \tag{5.3.15.2-71}$$

Pre-calculate the pitch gain and the pitch correlation for each fixed subframe:

$$\begin{aligned} &if \ \ (Rate = Full\_Rate)\ , \\ &\quad L_{tmp} = \min\{i_0 + L_{sub}\ , (i+1)\cdot 40\} - \max\{i_0, i\cdot 40\} \\ &\quad g_a(i) \Leftarrow g_a(i) + G_{P\_SM} \cdot \max\{L_{tmp}, 0\} \\ &\quad R_{SM}(i) \Leftarrow R_{SM}(i) + R_{P\_SM} \cdot \max\{L_{tmp}, 0\} \\ &\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad i = 0,1,2,3 \\ &else \\ &\quad L_{tmp} = \min\{i_0 + L_{sub}\ , (i+1)\cdot 53\} - \max\{i_0, i\cdot 53\} \\ &\quad g_a(i) \Leftarrow g_a(i) + G_{P\_SM} \cdot \max\{L_{tmp}, 0\} \\ &\quad R_{SM}(i) \Leftarrow R_{SM}(i) + R_{P\_SM} \cdot \max\{L_{tmp}, 0\} \\ &\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad i = 0,1,2 \end{aligned} \tag{5.3.15.2-72}$$

Finally, the starting point for the next variable subframe is updated:

$$i_0 \Leftarrow i_0 + L_{sub} \tag{5.3.15.2-73}$$

5.3.15.3 Detecting Stationary Voiced Speech and Calculating Open-Loop Pitch Gain for each Fixed Subframe

This section describes the detection of stationary voiced speech frame (type-1 frame) and the calculation of pitch gain for each fixed subframe. The algorithm for Rate 1 is slightly different from that for Rate 1/2. The type-1 frame detection will be helped by the average pitch correlation, which is finalized by:

$$R_{SM\_avg} \Leftarrow R_{SM\_avg} / L_{frm} \tag{5.3.15.3-1}$$

5.3.15.3.1 Detecting Stationary Voiced Speech and Calculating Open-Loop Pitch Gains for Rate 1

Calculate the pitch gain and the pitch correlation for each fixed subframe:

$$R_{SM}(i) \Leftarrow R_{SM}(i)/40\,, \qquad i = 0,1,2,3$$
$$g_a(i) \Leftarrow g_a(i)/40\,, \qquad i = 0,1,2,3 \tag{5.3.15.3-2}$$

Modify pitch gain for each subframe:

$$\begin{aligned} &\textit{if } (F_{flat} = 1), \\ &\quad g_a(i) \Leftarrow \max\left\{\min\left\{g_a(i)\cdot\left(0.75 + 0.25\ R_{SM}(i)\right), 1.2\right\}, 0\right\} \\ &\qquad\qquad i = 0,1,2,3 \\ &\textit{else} \\ &\quad g_a(i) \Leftarrow \max\left\{\min\left\{g_a(i)\cdot\left(0.7 + 0.3\ R_{SM}(i)\right), 1.2\right\}, 0\right\} \\ &\qquad\qquad i = 0,1,2,3 \end{aligned} \tag{5.3.15.3-3}$$

Detect stationary voiced speech:

$$\textit{if } (R_{SM\_last} > 0.6 \textit{ and } R_{SM\_avg} > 0.6 \textit{ and } C_{SM} > 0):$$
$$G_{P\_avg} = \frac{g_a(0) + g_a(1) + g_a(2) + g_a(3)}{4} \tag{5.3.15.3-4}$$

Calculate:

$$\begin{aligned} C_1 = &\left|g_a(0) - G_{P\_avg}\right| < 0.25 \text{ and } \left|g_a(1) - G_{P\_avg}\right| < 0.25 \text{ and} \\ &\left|g_a(2) - G_{P\_avg}\right| < 0.25 \text{ and } \left|g_a(3) - G_{P\_avg}\right| < 0.25 \end{aligned} \tag{5.3.15.3-5}$$

$$C_2 = g_a(0) > 0.5 \text{ and } g_a(1) > 0.5 \text{ and } g_a(2) > 0.5 \text{ and } g_a(3) > 0.5 \tag{5.3.15.3-6}$$

$$C_3 = R_{SM}(0) > 0.75 \text{ and } R_{SM}(1) > 0.75 \text{ and } R_{SM}(2) > 0.75 \text{ and } R_{SM}(3) > 0.75 \tag{5.3.15.3-7}$$

$$\text{If } \left(C_1 \text{ and } C_2 \text{ and } R_{SM\_\min} > 0.5\right) \text{ or } \left(C_3 \text{ and } G_{P\_avg} > 0.5\right)$$
$$C_{FR} = 6 \tag{5.3.15.3-8}$$

Update last pitch correlation:

$$R_{SM\_last} = R_{SM}(3) \tag{5.3.15.3-9}$$

5.3.15.3.2 Detecting Stationary Voiced Speech and Calculating Open-Loop Pitch Gains for Rate 1/2

Calculate the pitch gain and the pitch correlation for each fixed subframe:

$$R_{SM}(i) \Leftarrow R_{SM}(i)/53\,, \qquad i = 0,1,2$$
$$g_a(i) \Leftarrow g_a(i)/53\,, \qquad i = 0,1,2 \tag{5.3.15.3-10}$$

Modify pitch gain for each subframe:

$$g_a(i) \Leftarrow \max\left\{\min\left\{g_a(i)\cdot\left(0.75 + 0.25\ R_{SM}(i)\right), 1.2\right\}, 0\right\}$$
$$i = 0,1,2 \tag{5.3.15.3-11}$$

Detect stationary voiced speech:

if $(R_{SM\_avg} > 0.6 \; and \; C_{SM} > 0)$:

$$G_{P\_avg} = \frac{g_a(0) + g_a(1) + g_a(2)}{3} \quad \text{(5.3.15.3-12)}$$

Calculate:

$$C_1 = \left| g_a(0) - G_{P\_avg} \right| < 0.25 \text{ and } \left| g_a(1) - G_{P\_avg} \right| < 0.25 \text{ and } \left| g_a(2) - G_{P\_avg} \right| < 0.25 \quad \text{(5.3.15.3-13)}$$

$$C_2 = G_P(0) > 0.4 \text{ and } G_P(1) > 0.4 \text{ and } G_P(2) > 0.4 \quad \text{(5.3.15.3-14)}$$

$$\text{If } \left(C_1 \text{ and } C_2 \text{ and } R_{SM\_\min} > 0.4 \right)$$
$$C_{FR} = 6 \quad \text{(5.3.15.3-15)}$$

Update last pitch correlation:

$$R_{SM\_last} = R_{SM}(2) \quad \text{(5.3.15.3-16)}$$

5.3.16 Final Frame Class and Type Decisions, Rate Selection, and Pitch Lag Refinement

5.3.16.1 Final Classification, Type and Rate Selection

The final decision on the frame class, the frame type, and the rate, is made according to the final values of several parameters, $Rate$, $Rate^{prv}$, C_{FR}, C_{FR}^{prv}, C_{VUV}, $l_p^{ol}(0)$, $l_p^{ol}(1)$, l_{pp}^{q}, $l_{pp}^{q}(frm-1)$, M_{SMV}, F_{MUS}, F_{HRM}. The parameters are compared to each other and to a set of fixed thresholds to obtain the final frame class, the final frame type, and the final coding rate for the frame.

The following frame rate/type transitions shall be disallowed at the SMV encoder

a) Rate 1 type-1 frame following a Rate 1/8 frame
b) Rate 1/2 type-1 frame following a Rate 1/8 frame
c) Rate 1 type-1 frame following a Rate 1/4 frame
d) Rate 1/2 type-1 frame following a Rate 1/4 frame
e) Rate 1/8 frame following a Rate 1 type-1 frame
f) Rate 1/8 frame following a Rate 1/2 type-1 frame
g) Rate 1/4 frame following a Rate 1 type-1 frame
h) Rate 1/4 frame following a Rate 1/2 type-1 frame

Transitions a, b, c, and d are disallowed by the rate-selection procedure described in Section 5.3.13. The transitions e, f, g, and h are disallowed here by using a Rate 1/2 type-0 frame in place of the first Rate 1/8 or Rate 1/4 frame that immediately follows a type-1 frame.

If the current frame is of type 1, and the previous frame is of type 1, and the pitch lag of the current frame is different from that of the previous frame by more than 15, the current frame shall be modified to a Rate 1, type-0 frame.

Table 5.3-8 and Table 5.3-9 show the different rates that are used to encode the different speech classes, C_{FM}, for the different encoding modes of SMV. The optimal rate selection for each of these classes is based on the various speech parameters described above.

C_{FM}	Rate 1/8	Rate 1/4	Rate 1/2	Rate 1
silence ($C_{FM}=0$)	√		√	
noise-like ($C_{FM}=1$)			√	√
unvoiced ($C_{FM}=2$)			√	√
onset ($C_{FM}=3$)				√
non-stat. voiced ($C_{FM}=5$)				√
stat. voiced ($C_{FM}=6$)				√

Table 5.3-8: SMV Class-Rate Map (Mode 0)

C_{FM}	Rate 1/8	Rate 1/4	Rate 1/2	Rate 1
silence ($C_{FM}=0$)	√			
noise-like ($C_{FM}=1$)		√	√	
unvoiced ($C_{FM}=2$)		√	√	
onset ($C_{FM}=3$)		√	√	√
non-stat. voiced ($C_{FM}=5$)			√	√
stat. voiced ($C_{FM}=6$)			√	√

Table 5.3-9: SMV Class-Rate Map (Modes 1, 2 and 3)

5.3.16.2 Pitch Lags Refinement

The open-loop pitch lags are modified if the rate is changed from Rate 1 to Rate 1/2 by the F_{HRM} signal, and if $C_{SM} > 0$, according to:

$$l_p^{ol}(0) = \begin{cases} l_p^{ol}(0) & 30 \le l_p^{ol}(0) \\ 2 \cdot l_p^{ol}(0) & l_p^{ol}(0) < 30 \end{cases} \quad (5.3.16\text{-}1)$$

$$l_p^{ol}(1) = \begin{cases} l_p^{ol}(1) & 30 \le l_p^{ol}(1) \\ 2 \cdot l_p^{ol}(1) & l_p^{ol}(1) < 30 \end{cases} \quad (5.3.16\text{-}2)$$

The subframe lags to be used as initial search points for the closed-loop pitch search for Rate 1 are set according to:

$$l_p^{SF}(0) = l_p^{SF}(1) = l_p^{ol}(0), \; l_p^{SF}(2) = l_p^{SF}(3) = l_p^{ol}(1) \quad (5.3.16\text{-}3)$$

For all other rates the subframe lags to be used as initial search points for the closed-loop pitch search are set according to:

$$l_p^{SF}(0) = l_p^{ol}(0), \; l_p^{SF}(1) = l_p^{ol}(1). \quad (5.3.16\text{-}4)$$

If $C_{SM} > 0$, the last subframe lag to be used as an initial search point is further modified to the integer part of the quantized open-loop pitch at the end of the encoding frame. For Rate 1 $l_p^{SF}(3) = l_{pp,INT}^{q}$, while for all other rates $l_p^{SF}(1) = l_{pp,INT}^{q}$.

5.3.17 Adaptive-codebook Gain De-emphasis for Type-1 Frame (Rate 1 and Rate 1/2)

The adaptive-codebook gain vector $g_a(m)$ for type-1 frame will be slightly attenuated from their calculated values. The gains for the 4 subframes of Rate 1 or the 3 subframes of Rate 1/2 is modified by the following steps:

If F_{noisyV} is 1

$$g_a(m) = 0.95 \cdot g_a(m) \qquad m = 1,2,3,[4] \tag{5.3.17-1}$$

else if F_{noisyV} is 2

$$g_a(m) = 0.92 \cdot g_a(m) \qquad m = 1,2,3,[4] \tag{5.3.17-2}$$

Furthermore, if F_{flat} is 1

$$g_a(m) = \alpha \cdot g_a(m) \qquad m = 1,2,3,[4] \tag{5.3.17-3}$$

where α is 0.95 if M_{SMV} is 0, otherwise, it is 0.97.

5.3.18 Adaptive-codebook Gain Quantization for Type-1 Frame (Rate 1 and Rate 1/2)

Routine name: `GEQ_Quant_PitchGain`

Input:

- Unquantized adaptive-codebook gain vector, $g_a(m),\ m = 1,2,3,[4]$
- Quantization table for the selected rate, $G_a^{Rate}(i,m),\ m = 1,2,3,[4]$

Output:

- Quantized adaptive-codebook gain vector, $g_a(m),\ m = 1,2,3,[4]$

The adaptive-codebook gain vector $g_a(m)$ for Type-1 frame is quantized to 6 and 4 bits for Rate 1 and Rate 1/2, respectively. The best vector for $g_a(m)$ is found as the vector that satisfies:

$$\arg\min_{i=0,\ldots,N_j-1}\left\{\sum_{m=1}^{M_j}\left(g_a(m)-G_a^{Rate}(i,m)\right)^2\right\} \qquad (5.3.18\text{-}1)$$

For Rate 1 $M_1 = 4$ and $N_1 = 64$, and for Rate 1/2 $M_2 = 3$ and $N_2 = 16$.

The quantization of the adaptive-codebook gains for type-1 frames is performed on a frame basis prior to any subframe processing.

5.3.19 Generating Modified Speech from Modified Weighted Speech

Routine name: `PWF_wspeech_to_speech`

Input:

- Modified weighted speech, $s_{wm}(n)$
- Weighing filter numerator coefficients, $\hat{a}_m^{num}(i)$, for $m = 0,1,2,3,4$
- Weighing filter denominator coefficients, $\hat{a}_m^{den}(i)$, for $m = 0,1,2,3,4$
- Adaptive low-pass filter parameters, μ_m, for $m = 0,1,2,3,4$

Output:

- Modified speech, $s_{\mathrm{mod}}(n)$

The modified speech, $s_{\mathrm{mod}}(n)$, is generated from the modified weighted speech, $s_{wm}(n)$, by the inverse weighting filter and the inverse of the adaptive low-pass filtering. The transfer function of the inverse of the fixed weighted synthesis filter for each 5 ms subframe, indexed by $m = 0,1,2,3$, is given by:

$$H_m(z) = \frac{1 + \sum_{i=1}^{10} a_m^{den}(i) \cdot z^{-i}}{1 + \sum_{i=1}^{10} a_m^{num}(i) \cdot z^{-i}} \qquad \text{(5.3.16.2-1)}$$

The filter parameters, $a_m^{den}(i)$ and $a_m^{num}(i)$ for $m = 0,1,2,3$, are the weighted filter parameters for each of the four 5 ms subframes, calculated in Section 5.3.5. The inverse adaptive low-pass filter is given by:

$$L_m(z) = \frac{1}{1 + \mu_m \cdot z^{-1}} \qquad \text{(5.3.16.2-2)}$$

where μ_m, for $m = 0,1,2,3,4$, was calculated in Section 5.3.8.

5.3.20 LSFs Smoothing Parameter

Routine name: `SMO_lsf_smooth_est`

Input:

- Voice activity decision, VAD
- Previous frame voice activity decision, VAD^{prv}
- First element of reflection coefficient vector for the second quarter of the encoding frame, $k_1(0)$
- LSFs vector calculated at the fourth quarter of the encoding frame, $lsf_1(i)$

Output:

- LSFs smooth factor, β_{SMO}^{LSF}

An LSFs smoothing factor is calculated, based on the VAD, the previous VAD, the first reflection coefficient corresponding to the second quarter of the encoding frame, and two spectral distortion measures that are calculated by the LSFs smoothing routine. The first spectral distortion measure is calculated between the current frame LSFs, $lsf_1(i)$, and the previous frame LSFs, $lsf_{1,prv}(i)$:

$$SD_1^{SMO} = \sum_{i=1}^{10} \left(lsf_1(i) - lsf_{1,prv}(i) \right)^2 \qquad (5.3.16.2\text{-}1)$$

The second spectral distortion is calculated by finding the difference between the current frame LSFs, $lsf_1(i)$, and an average of the LSFs, $\overline{lsf}^{SMO}$:

$$sd^{SMO} = \sum_{i=1}^{10} \left(lsf_1(i) - \overline{lsf}^{SMO}(i) \right)^2 \qquad (5.3.16.2\text{-}2)$$

The sum of sd^{SMO} over the current frame and the last 3 frames gives the second spectral distortion measure, SD_2^{SMO}. The input parameters and the two spectral distortion measures are tested and compared to a set of thresholds, to generate a noise stationarity index, I_{nst}, which has a value at the range $[0,\ldots,5]$. The LSFs smoothing factor is given by:

$$\beta_{SMO}^{LSF} = SMO_{tab}[I_{nst}] \qquad (5.3.16.2\text{-}3)$$

where the values in the LSF smoothing table, SMO_{tab}, are:

$$SMO_{tab} = \left[0, 0, \frac{0.9}{16}, \frac{0.9 \cdot 4}{16}, \frac{0.9 \cdot 9}{16}, 0.9\right] . \quad (5.3.16.2\text{-}4)$$

5.3.21 LSFs Smoothing

The average of the LSFs is updated according to:

$$\overline{lsf}^{SMO}(i) = \beta_{LSF}^{SMO} \cdot \overline{lsf}^{SMO}(i) + \left(1 - \beta_{LSF}^{SMO}\right) \cdot lsf_1(i) \qquad i = 1,\ldots,10 \tag{5.3.16.2-1}$$

For the SMV operation Mode 3, or if the SMV in a Half-Rate-Max Mode, or if the flag F_{LPC_onset} is on, the LSFs that are quantized and transmitted, $\underline{lsf}_1(i)$ are generated according to:

$$\underline{lsf}_1(i) = \beta_{LSF}^{SMO} \cdot \underline{lsf}_{1,prv}(i) + \left(1 - \beta_{lSF}^{SMO}\right) \cdot \left(0{,}75 \cdot lsf_2(i) + 0.25 \cdot lsf_1(i)\right) \qquad i = 1,\ldots,10 \tag{5.3.16.2-2}$$

where $\underline{lsf}_{1,prv}$ are the previous frame smoothed LSFs.

For all other cases, the LSFs that are quantized and transmitted, $\underline{lsf}_1(i)$ are generated according to:

$$\underline{lsf}_1(i) = \beta_{LSF}^{SMO} \cdot \underline{lsf}_{1,prv}(i) + \left(1 - \beta_{LSF}^{SMO}\right) \cdot lsf_1(i) \qquad i = 1,\ldots,10 \tag{5.3.16.2-3}$$

5.3.22 LSFs Quantization

Routine name: `LSF_Q_lsfqnt`

Input:

- Smoothed LSFs, $\underline{lsf}_1(i)$
- SMV rate, $Rate$

Output:

- Quantized LSFs, $lsf_1^q(i)$

The LSFs are quantized with 25 bits, 21 bits, 20 bits, and 11 bits, for the Rate 1, the Rate 1/2, the Rate 1/4, and the Rate 1/8, respectively. The quantization is performed using a weighted-mean-squared error measure, and a multi-stage vector quantization scheme.

5.3.22.1 Calculation of LSFs Quantization Weights

The LSFs quantization weights are given by:

$$w_{LSF}(i) = \left(P_{LP}^2\left(\underline{lsf}_1(i)\right)\right)^{-0.2} \tag{5.3.22.1-1}$$

Where $P_{LP}^2\left(\underline{lsf}_1(i)\right)$ is the LP power spectrum at the $\underline{lsf}_1(i)$, which can be calculated, for example, by:

$$P_{LP}^2\left(\underline{lsf}_1(i)\right) = 4\cdot\left(2-2\cos\left(2\pi\cdot\underline{lsf}_1(i)\right)\right)\left(\cos\left(5\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+p_1'\cdot\cos\left(4\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+ p_2'\cdot\cos\left(3\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+p_3'\cdot\cos\left(2\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+p_4'\cdot\cos\left(2\pi\cdot\underline{lsf}_1(i)\right)+p_5'/2\right)$$
$$i = 2,4,6,8,10$$

(5.3.22.1-2)

$$P_{LP}^2\left(\underline{lsf}_1(i)\right) = 4\cdot\left(2+2\cos\left(2\pi\cdot\underline{lsf}_1(i)\right)\right)\left(\cos\left(5\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+q_1'\cdot\cos\left(4\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+ q_2'\cdot\cos\left(3\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+q_3'\cdot\cos\left(2\cdot 2\pi\cdot\underline{lsf}_1(i)\right)+q_4'\cdot\cos\left(2\pi\cdot\underline{lsf}_1(i)\right)+q_5'/2\right)$$
$$i = 1,3,5,7,9$$

(5.3.22.1-3)

where $\left[q_1', q_2', q_3', q_4', q_5'\right]$ and $\left[p_1', p_2', p_3', p_4', p_5'\right]$ are as calculated in Section 5.3.1.2.4 for the fourth quarter of the encoding frame.

The power spectrum can be also calculated from the LSFs.

The weighting parameters are further modified according to

$$w_{LSF}(i) \Leftarrow \left(1.64-0.16\cdot i\right)\cdot w_{LSF}(i) \qquad i = 4,\ldots,10 \tag{5.3.22.1-4}$$

5.3.22.2 LSFs Mean Removal

A vector that represents the mean value of the LSFs is subtracted from the LSFs, generating the mean-removed LSFs vector:

$$\underline{lsf'}_1(i) = \underline{lsf}_1(i) - M_{LSF}(i) \qquad i = 1,\ldots,10 \tag{5.3.22.2-1}$$

5.3.22.3 LSFs Prediction

The LSFs are predicted using a moving average (MA) prediction scheme. The prediction error, ε_k, is generated by:

$$\varepsilon_j(i) = \underline{lsf'}_1(i) - \sum_{p=1}^{O_{Rate}^{MA}} b_{j,p}^{Rate}(i) \cdot \varepsilon_{j^*,p}^{q}(i) \qquad i = 1\ldots,10 \tag{5.3.22.3-1}$$

where O_{Rate}^{MA}, the order of the MA prediction, is 2 for Rate 1 and 4 for all other rates. The prediction coefficients $b_{j,p}^{Rate}(i)$, for each of the possible switch prediction j ($j = 0,1$ for Rate 1 and Rate 1/2, and $j = 0$ Rate 1/4 and Rate 1/8), multiply the previously chosen and quantized prediction error (index by j^*) for O_{Rate}^{MA} frames (indexed by p). Two prediction-error vectors, ε_1 and ε_2, are generated for Rate 1 and Rate 1/2, and a single prediction vector, ε_1, is generated for Rate 1/4 and Rate 1/8. Different prediction coefficients are used for each rate, except of the prediction coefficients for Rate 1/4, which use the second set of prediction coefficients of Rate 1/2.

5.3.22.4 Quantization of the LSFs Prediction Error

The LSFs prediction error is quantized using a multi-stage vector quantization scheme. The number of bits used for each rate is different, but the quantization tables are shared between all rates and the quantization schemes are similar. A generic quantization scheme is depicted in the following diagram, while the LSFs quantization for each rate is described in Sections 5.3.22.4.1 to 5.3.22.4.4.

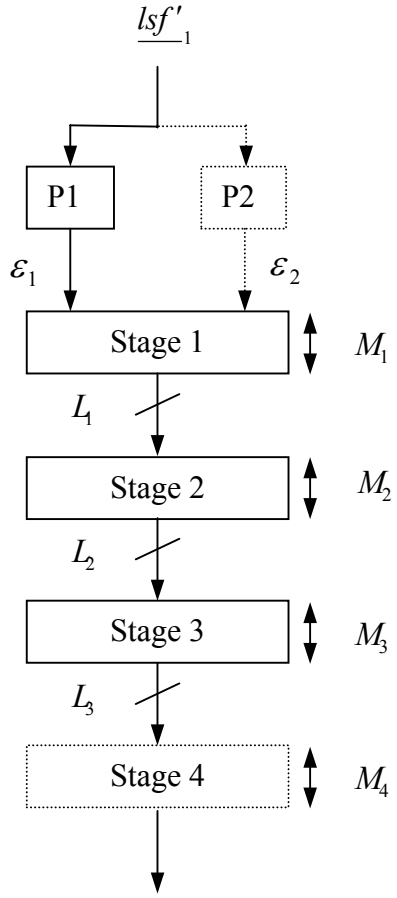

Figure 5.3-3: A Schematic Diagram of (Switched) Predictive Multi-Stage LSFs Quantization

The input to the first stage is the LSFs prediction error, and the input for each consecutive stage is the quantization error of the previous stage. Each stage has several inputs. Since two predictors are used for Rate 1 and Rate 1/2, stage 1 has two inputs for these rates. Since a single predictor is used for Rate 1/4 and Rate 1/8, stage 1 has a single input for these rates. The number of inputs for stage 2 is L_1, the number of inputs for stage 3 is L_2, and the number of inputs for stage 4 is L_3. The number of code vectors in the j^{th} stage is M_j. For the j^{th} stage, denoting the input vector by $I_{LSF}^{j,l}(i)$, where l is the index of the input vector, the codebook vectors $C_{LSF}^{j}(i)$, the output vectors are chosen as the L_j best combination of an input vector and a code vector, minimizing the weighted mean squared error measure:

$$WMSE_{LSF}^{j,l,r} = \sum_{i=1}^{10} w_{LSF}(i)\left(I_{LSF}^{j,l}(i) - C_{LSF}^{j,r}(i)\right)^2 \qquad (5.3.22.4\text{-}1)$$

The output vectors from the j^{th} stage are generated by the L_j best combination of an input vector and a codebook entry, and are given by

$$I_{LSF}^{j+1,l'}(i) = I_{LSF}^{j,l^*}(i) - C_{LSF}^{j,r^*}(i) \qquad i = 1,\ldots,10 \tag{5.3.22.4-2}$$

where l' is counting the input vectors to $j^{th}+1$ stage, and l^* and r^* denote the chosen input vector and the chosen codebook vector, respectively.

5.3.22.4.1LSFs Quantization for Rate 1

Two predictors are used for Rate 1, and both ε_1 and ε_2 are the inputs to stage 1. The following table summarizes the parameters used for Rate 1:

M_1	M_2	M_3	M_4	L_1	L_2	L_3
128	128	64	16	8	7	5

Table 5.3-10: Parameters for Rate 1 LSFs Quantization

The total number of bits used for LSFs quantization for Rate 1 is 25, comprising of 1 bit for the switched prediction, 7 bits for the first stage, 7 bits for the second stage, 6 bits for the third stage, and 4 bits for the fourth stage.

5.3.22.4.2LSFs Quantization for Rate 1/2

Two predictors are used for Rate 1/2, and both ε_1 and ε_2 are the inputs to stage 1. The following table summarizes the parameters used for Rate 1/2:

M_1	M_2	M_3	L_1	L_2
128	128	64	9	7

Table 5.3-11: Parameters for Rate 1/2 LSFs Quantization

The total number of bits used for LSFs quantization for Rate 1/2 is 21, comprising, 7 bits for the first stage, 7 bits for the second stage, and 6 bits for the third stage.

5.3.22.4.3LSFs Quantization for Rate 1/4

A single predictor is used for Rate 1/4, and only ε_1 is the input to stage 1. The predictor is identical to one of the two predictors of Rate 1/2. The following table summarizes the parameters used for Rate 1/4:

M_1	M_2	M_3	L_1	L_2
128	128	64	9	7

Table 5.3-12: Parameters for Rate 1/4 LSFs Quantization

The total number of bits used for LSFs quantization for Rate 1/4 is 20, comprising of 7 bits for the first stage, 7 bits for the second stage, and 6 bits for the third stage.

5.3.22.4.4LSFs Quantization for Rate 1/8

A single predictor is used for Rate 1/8, and only ε_1 is the input to stage 1. The following table summarizes the parameters used for Rate 1/8:

M_1	M_2	M_3	L_1	L_2
16	16	8	16	16

Table 5.3-13: Parameters for Rate 1/8 LSFs Quantization

The code vectors for each of the 3 stages are generated as an average of two code vectors in the respective stage used for the LSFs quantization of the other rates. The total number of bits used for LSFs quantization for Rate 1/8 is 11, comprising of 4 bits for the first stage, 4 bits for the second stage, and 3 bits for the third stage.

5.3.23 Interpolation of Quantized and Unquantized LSFs

The quantized LSFs are interpolated for each subframe. For Rate 1 type-0 frames the interpolation contour is quantized with 2 bits. For all other cases, no bits are transmitted for the interpolation contour, and the LSFs are linearly interpolated between the previous frame quantized LSFs and the current frame quantized LSFs. The interpolated LSFs for each subframe are converted to the quantized prediction coefficients for each subframe, $a_j^q(i)$. The range of j is $[1,2,3,4]$ for Rate 1 frames, $[1,2]$ for Rate 1/2 type-0 frames, and $[1,2,3]$ for Rate 1/2 type-1 frames. The quantized LSFs are converted to a set of quantized prediction coefficients, $\hat{a}_m^q(i)$.

5.3.23.1 Interpolation of Quantized LSFs for Rate 1 Type-0 Frames

For Rate 1 type-0 frames the encoder searches for the best interpolation path and quantizes the best interpolation path with 2 bits. The search is carried by comparing the 4 possible interpolations between the current frame quantized LSFs, $lsf_1^q(i)$, and the previous frame quantized LSFs, $lsf_{1,prv}^q(i)$, with respect the LSFs with were calculated of the second quarter of the coding frame, $lsf_0(i)$.

5.3.23.1.1 Calculating of Distance Weights

A set of weights, w_{INTR}, is calculated from $lsf_0(i)$ according to:

$$\begin{aligned} w_{INTR}(1) &= \left(1 - lsf_0(1)\right) \cdot \left(1 - \left(lsf_0(2) - lsf_0(1)\right)\right) \\ w_{INTR}(i) &= \left(1 - lsf_0(i)\right) \cdot \left(1 - \min\left(lsf_0(i+1) + lsf_0(i), lsf_0(i) + lsf_0(i-1)\right)\right) \qquad i = 2,\ldots,9 \\ w_{INTR}(10) &= \left(1 - lsf_0(10)\right) \cdot \left(1 - \left(lsf_0(10) - lsf_0(9)\right)\right) \end{aligned} \tag{5.3.23.1-1}$$

5.3.23.1.2 Determination of Best Interpolation Path

The best interpolation path is determined by calculating the four possible weighted averages between $lsf_1^q(i)$ and $lsf_{1,prv}^q(i)$, and finding the one that is closest to $lsf_0(i)$. The weighted average, $\overline{lsf}_0^{j,q}$, for $j = 1,\ldots,4$ is calculated by:

$$\overline{lsf}_0^{j,q}(i) = C_j \cdot lsf_1^q(i) + \left(1 - C_j\right) \cdot lsf_{1,prv}^q(i) \qquad i = 1,\ldots,10 \tag{5.3.23.1-2}$$

where $C_1 = 0.5$, $C_2 = 0.6$, $C_3 = 0.7$, and $C_4 = 0.8$. For each j, the distance between $\overline{lsf}_o^{j,q}(i)$ and $lsf_0(i)$ is calculated as:

$$D_{INTR}^j = \sum_{i=1}^{10} \left| \overline{lsf}_0^{j,q}(i) - lsf_0(i) \right| \cdot w_{INTR}(i) \tag{5.3.23.1-3}$$

The C_{j^*} that minimizes D_{INTR}^j is chosen, and the interpolated LSFs for each of the 4 subframes, $\hat{lsf}_m^q(i)$ are obtained by:

$$\hat{lsf}_m^q(i) = \begin{cases} 0.5 \cdot C_{j^*} \cdot lsf_1^q(i) + \left(1 - 0.5 \cdot C_{j^*}\right) \cdot lsf_{1,prv}^q(i) & m = 0 \\ C_{j^*} \cdot lsf_1^q(i) + \left(1 - C_{j^*}\right) \cdot lsf_{1,prv}^q(i) & m = 1 \\ 0.5 \cdot \left(1 + C_{j^*}\right) \cdot lsf_1^q(i) + 0.5 \cdot \left(1 - C_{j^*}\right) \cdot lsf_{1,prv}^q(i) & m = 2 \\ lsf_1^q(i) & m = 3 \end{cases} \quad i = 1,\ldots,10$$

(5.3.23.1-4)

5.3.23.2 Interpolation of Quantized LSFs for Rate 1 Type-1 frames, Rate 1/4 frames, and Rate 1/8 Frames

For Rate 1 type-1 frames, as well as for Rate 1/4 and Rate 1/8 frames, the LSFs are linearly interpolated for each subframe, and the LSFs for each of the 4 subframes, $\hat{lsf}_m^q(i)$ are obtained by:

$$\hat{lsf}_m^q(i) = \begin{cases} 0.25 \cdot lsf_1^q(i) + 0.75 \cdot lsf_{1,prv}^q(i) & m = 0 \\ 0.5 \cdot lsf_1^q(i) + 0.5 \cdot lsf_{1,prv}^q(i) & m = 1 \\ 0.75 \cdot lsf_1^q(i) + 0.25 \cdot lsf_{1,prv}^q(i) & m = 2 \\ lsf_1^q(i) & m = 3 \end{cases} \quad i = 1,\ldots,10 \qquad \text{(5.3.23.2-1)}$$

5.3.23.3 Interpolation of Quantized and Unquantized LSFs for Rate 1/2 Type-0 Frames

For Rate 1/2 type-0 frames the LSFs are linearly interpolated for each subframe, and the LSFs for each of the 2 subframes, $\hat{lsf}_m^q(i)$ are obtained by:

$$\hat{lsf}_m^q(i) = \begin{cases} 0.375 \cdot lsf_1^q(i) + 0.625 \cdot lsf_{1,prv}^q(i) & m = 0 \\ 0.875 \cdot lsf_1^q(i) + 0.125 \cdot lsf_{1,prv}^q(i) & m = 1 \end{cases} \quad i = 1,\ldots,10 \qquad \text{(5.3.23.3-1)}$$

The unquantized LSFs are interpolated according to:

$$\hat{lsf}_m(i) = \begin{cases} 0.75 \cdot lsf_0(i) + 0.25 \cdot lsf_{1,prv}(i) & m = 0 \\ 0.75 \cdot lsf_1^q(i) + 0.25 \cdot lsf_0(i) & m = 1 \end{cases} \quad i = 1,\ldots,10$$

The interpolated unquantized LSFs are converted to interpolated unquantized prediction coefficients $\hat{a}_m(i)$.

5.3.23.4 Interpolation of Quantized and Unquantized LSFs for Rate 1/2 Type-1 Frames

For Rate 1/2 type-1 frames the LSFs are linearly interpolated for each subframe, and the LSFs for each of the 3 subframes, $\hat{lsf}_m^q(i)$ are obtained by:

$$\hat{lsf}_m^q(i) = \begin{cases} 0.290625 \cdot lsf_1^q(i) + 0.709375 \cdot lsf_{1,prv}^q(i) & m = 0 \\ 0.621875 \cdot lsf_1^q(i) + 0.378125 \cdot lsf_{1,prv}^q(i) & m = 1 \\ 0.953125 \cdot lsf_1^q(i) + 0.046875 \cdot lsf_{1,prv}^q(i) & m = 2 \end{cases} \quad i = 1,\ldots,10$$

(5.3.23.4-1)

The unquantized LSFs are interpolated according to:

$$\hat{lsf}_m(i) = \begin{cases} 0.58125 \cdot lsf_0(i) + 0.41875 \cdot lsf_{1,prv}(i) & m = 0 \\ 0.25 \cdot lsf_1(i) + 0.75 \cdot lsf_0(i) & m = 1 \\ 0.9125 \cdot lsf_1(i) + 0.0875 \cdot lsf_0(i) & m = 2 \end{cases} \qquad i = 1,\ldots,10$$

(5.3.23.4-2)

The interpolated unquantized LSFs are converted to interpolated unquantized prediction coefficients $\hat{a}_m(i)$.

5.3.24 Recalculating of Weighting Filter Coefficients for Rate 1/2

Since Rate 1/2 frames have 3 subframes for type-1 frame and 2 subframes for type-0 frame, the weighing filter coefficients for each subframe are recalculated. For type-1 frames, the interpolated first element of the reflection coefficients are recalculated as:

$$\begin{aligned}
\hat{k}_0(0) &\Leftarrow 0.8375 \cdot \hat{k}_0(0) + 0.1625 \cdot \hat{k}_1(0) \\
\hat{k}_1(0) &\Leftarrow 0.5 \cdot \hat{k}_1(0) + 0.5 \cdot \hat{k}_2(0) \\
\hat{k}_2(0) &\Leftarrow 0.8375 \cdot \hat{k}_2(0) + 0.1625 \cdot \hat{k}_3(0)
\end{aligned} \tag{5.3.24-1}$$

Similarly, the interpolated LPC prediction gains are recalculated as:

$$\begin{aligned}
\hat{G}_0^{LPC} &\Leftarrow 0.8375 \cdot \hat{G}_0^{LPC} + 0.1625 \cdot \hat{G}_1^{LPC} \\
\hat{G}_1^{LPC} &\Leftarrow 0.5 \cdot \hat{G}_1^{LPC} + 0.5 \cdot \hat{G}_2^{LPC} \\
\hat{G}_2^{LPC} &\Leftarrow 0.8375 \cdot \hat{G}_2^{LPC} + 0.1625 \cdot \hat{G}_3^{LPC}
\end{aligned} \tag{5.3.24-2}$$

For type-0 frames, the interpolated first element of the reflection coefficients are recalculated as:

$$\begin{aligned}
\hat{k}_0(0) &\Leftarrow 0.5 \cdot \hat{k}_0(0) + 0.5 \cdot \hat{k}_1(0) \\
\hat{k}_1(0) &\Leftarrow 0.5 \cdot \hat{k}_2(0) + 0.5 \cdot \hat{k}_3(0)
\end{aligned} \tag{5.3.24-3}$$

Similarly, the interpolated LPC gain is recalculated as:

$$\begin{aligned}
\hat{G}_0^{LPC} &\Leftarrow 0.5 \cdot \hat{G}_0^{LPC} + 0.5 \cdot \hat{G}_1^{LPC} \\
\hat{G}_1^{LPC} &\Leftarrow 0.5 \cdot \hat{G}_2^{LPC} + 0.5 \cdot \hat{G}_3^{LPC}
\end{aligned} \tag{5.3.24-4}$$

The weighing filter coefficients, $\hat{a}_m^{num}(i)$ and, $\hat{a}_m^{den}(i)$ for each subframe are recalculated for each subframe, using the recalculated and interpolated prediction coefficients, $\hat{a}_m(i)$, first element of reflection coefficients, $\hat{k}_m(0)$, and LPC gains, $\hat{G}_m^{LPC}$, as described in Section 5.3.5.

5.3.25 Identifying Long-Term Spectral Characteristic

Routine name: `CLA_Identify_Input`

Input:

- Frame type, $Type$
- Interpolated quantized prediction coefficients for the first subframe, $\hat{a}_0^q(i)$
- First element of quantized LSF vector, $lsf_1^q(1)$
- Quantized adaptive-codebook gain for the last subframe of the previous frame, g_a^{prv}
- SMV rate, $Rate$

Output:

- Spectral flatness flag, F_{flat}
- LPC prediction gain from quantized parameters of the first subframe, $G_{0,q}^{LPC}$

The identification of the long-term spectral characteristics is performed by the analysis of several quantized spectral and pitch characteristics over a long time period.

The quantized prediction coefficients of the first subframe are converted to quantized reflection coefficients, $\hat{k}_{0,q}(i)$, and the quantized LPC prediction gain is calculated by:

$$G_{0,q}^{LPC} = -10 \cdot \log_{10}\left(\prod_{i=1}^{10}\left(1 - k_{0,q}^2(i)\right)\right) \tag{5.3.25-1}$$

A counter of the number of consecutive Rate 1/8 frames, $C_{1/8}$, is obtained. A counter of the type-1 frames, C_{type1}, is maintained and used in the calculation of a long term parametric average, $\overline{X}$ as follows:

$$X = 2.5 \cdot \left(\min\left(\frac{G_{0,q}^{LPC}}{35}, 1\right) - 2 \cdot k_{0,q}(1) + \min\left(g_a^{prv}, 1\right) + 15 \cdot lsf_1^q(1)\right) \tag{5.3.25-2}$$

$$\overline{X} = \begin{cases} 0.75 \cdot X & C_{type1} = 0 \\ 0.9 \cdot \overline{X} + 0.1 \cdot X & 0 < C_{type1} < 10 \\ 0.99 \cdot \overline{X} + 0.01 \cdot X & 10 \le C_{type1} < 100 \\ 0.998 \cdot \overline{X} + 0.002 \cdot X & 100 \le C_{type1} \end{cases} \tag{5.3.25-3}$$

C_{type1} is updated after the calculation of Equation (5.3.25-2) and Equation (5.3.25-3) above, and is limited to 10000. The final long-term spectral characteristic flag, F_{flat}, is updated only if the frame type is 0, $C_{1/8} > 10$, and $C_{type1} > 200$, as follows:

$$F_{flat} = \begin{cases} 1 & \overline{X} > 5.8 \\ 0 & \overline{X} < 5.3 \end{cases} \quad \text{(5.3.25-4)}$$

If none of the conditions is met, F_{flat} is not changed from its previous value.

5.4 Excitation Coding at Rate 1/8

5.4.1 Excitation and Unquantized Gain Determination

Routine name: GCB_gauss_excit

Input:

- Seed, $seed$
- Energies of weighted residual for two halves of the frame, $E_{rw}(0)$ and $E_{rw}(1)$

Output:

- Unquantized fixed-codebook gain, g_c
- Random excitation, $r(n)$

A Gaussian random excitation $r(n)$ is generated using the random noise generator described in Section 5.6.19. For this rate, the seed used is a signed 16-bit static variable initialized to 0x0E76. The seed is re-set to its initial value whenever the rate is not the Rate 1/8. The gain g_c is determined as follows:

$$g_c = 0.7 \cdot \sqrt{\frac{\max(E_{ref} - 960, 0)}{\sum_{n=0}^{L_{frm}} e^2(n)}} \qquad (5.4.1\text{-}1)$$

where E_{ref} is set according to the rates of the previous two frames. If either previous two frames was not coded using Rate 1/8:

$$E_{ref} = E_{wr}(0) + E_{wr}(1) \qquad (5.4.1\text{-}2)$$

otherwise

$$E_{ref} = 0.5 \cdot E_{ref} + 0.5 \cdot (E_{wr}(0) + E_{wr}(1)) \qquad (5.4.1\text{-}3)$$

5.4.2 Gain quantization

Routine name: GEQ_gainNSQMA_1_5

Input:

- Random Excitation, $r(n)$
- Unquantized fixed-codebook gain, g_c

Output:

- Quantized fixed-codebook gain, g_c

A single fixed-codebook gain g_c is quantized, using a prediction scheme, with 5 bits per frame. The quantized $\hat{g}_c$ is found as the entry that satisfies:

$$\arg\min\left\{\left|g_c - \hat{g}_c\right|\right\} \text{ such that } \hat{g}_c \le 5 g_{mem} \tag{5.4.2-1}$$

where g_{mem} is the estimate of the average Rate 1/8 frame gain, and is updated after the above quantization for each Rate 1/8 frame as

$$g_{mem} = \max\left(round\left(\max\left(g_r, 100\right)\right), round\left(0.05 \cdot round\left(\max\left(g_r, 100\right)\right) + 0.95 \cdot g_{mem}\right)\right) \tag{5.4.2-2}$$

The entries of the codebook contain the correction factor for the predicted fixed-codebook gain. The prediction of the fixed-codebook gain is based on a 4th order MA prediction of the fixed-codebook energy.

The relation between the correction factor γ and the quantized fixed-codebook gain is given by

$$\hat{g}_c = \gamma \cdot \tilde{g}_c \tag{5.4.2-3}$$

where $\hat{g}_c$ is the quantized fixed-codebook gain, and $\tilde{g}_c$ is the predicted fixed-codebook gain. The predicted fixed-codebook gain is based on a 4th order MA prediction of the fixed-codebook energy, and is given by

$$\tilde{g}_c = 10^{\frac{1}{20}\left(\tilde{E} - E_c + \overline{E}\right)} \tag{5.4.2-4}$$

where the $\overline{E} = 30\text{dB}$ is the mean energy,

$$E_c = 10\log_{10}\left(\frac{1}{L_{frm}}\sum_{n=0}^{L_{frm}-1} c_{unf}^{fix}(n)^2\right) \tag{5.4.2-5}$$

and

$$\tilde{E} = \sum_{i=1}^{4} b_i \cdot \left(20\log_{10}\gamma_{frm-i}^{q}\right) \tag{5.4.2-6}$$

where the prediction coefficients of the MA prediction are $\{b_1, b_2, b_3, b_4\} = \{0.7, 0.6, 0.4, 0.2\}$, and γ_{frm-i}^{q} are the quantized correction factors for the previous 4 frames.

To avoid production of a NULL (all 1s) Rate 1/8 packet at the encoder, the Rate 1/8 gain search shall be restricted to the first 31 entries of the table when the quantized LSF indices are all 1s.

5.5 Excitation Coding at Rate 1/4

Noise Excited Linear Prediction (NELP) is used for Rate 1/4 frames. A pseudo-random noise is colored using a set of filters, and scaled appropriately by gains in 10 subframes of 16 samples each to model the excitation.

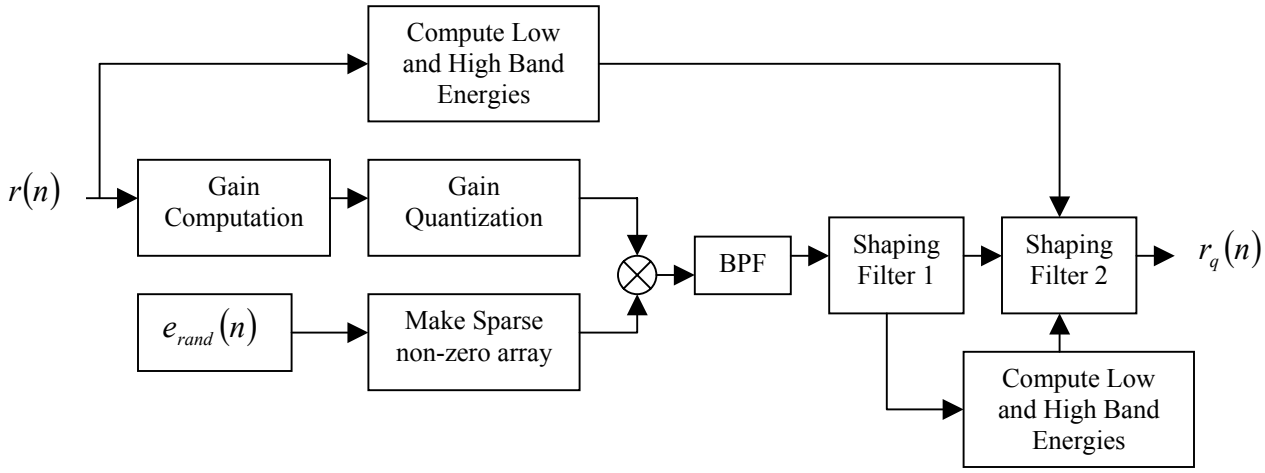

Figure 5.5-1: Excitation Coding for Rate 1/4 Frames

5.5.1 Computation and Quantization of Gains

Routine name: `nelp_encoder`

Input:

- Pre-processed speech, $\tilde{s}(n)$
- Quantized LSFs, $lsf_1^q(i)$

Output:

- Quantized excitation, $r_q(n)$

The pre-processed speech, $\tilde{s}(n)$, is converted to the residual signal $r(n)$ using the quantized LSFs, $lsf_1^q(i)$ (by converting the LSFs to LPCs, and zero filtering the pre-processed speech with the LPCs). The energy E_i of the residual signal is computed for each of the 10 subframes $(0 \le i < 10)$ as

$$E_i = \frac{1}{16} \sum_{n=16i}^{16i+15} r^2(n) \qquad (5.5.1\text{-}1)$$

The 10 subframe gains G_i are computed from the above energies as

$$G_i = \sqrt{E_i} \tag{5.5.1-2}$$

Two normalization factors for two splits $(0 \le j < 2)$ of the above 10 gains are determined by

$$G'[j] = \sqrt{\frac{1}{5}\sum_{i=0}^{4} G_{j*5+i}^2} \tag{5.5.1-3}$$

The above two normalization factors are vector quantized using a two dimensional codebook $UVG1CB[32][2]$ to provide an index $i\hat{G}'$ as

$$i\hat{G}' = \arg\min_k \left(\sum_{j=0}^{1} (\log 10(G'[j]) - UVG1CB[k][j])^2 \right) \quad , \quad 0 \le \mathrm{k} < 32 \tag{5.5.1-4}$$

The quantized normalization factors are then computed for $(0 \le j < 2)$ using

$$\hat{G}'[j] = 10^{UVG1CB[i\hat{G}'][j]} \tag{5.5.1-5}$$

The 10 subframe gains are then normalized using the quantized normalization factors according to

$$G''[2j+i] = \frac{G[2j+i]}{\hat{G}'[j]} \quad , 0 \le i < 5; 0 \le j < 2 \tag{5.5.1-6}$$

The 10 normalized gains are then vector quantized using a 3-dimensional codebook $UVG2CB[2][64][5]$ to provide two indices $iG''[0]$, and $iG''[1]$ such that

$$i\hat{G}''[j] = \arg\min_k \left(\sum_{i=0}^{4} (G''[2j+i] - UVG2CB[j][k][i])^2 \right) \quad , \quad 0 \le \mathrm{k} < 64; 0 \le \mathrm{j} < 2 \tag{5.5.1-7}$$

The quantized, normalized gains are then given by

$$\hat{G}''[2j+i] = UVG2CB[j][i\hat{G}''][i] \quad , \quad 0 \le i < 5; 0 \le j < 2 \tag{5.5.1-8}$$

The quantized gains used to scale the NELP excitation then are

$$\hat{G}[2j+i] = \hat{G}'[j]\hat{G}''[2j+i] \quad , \quad 0 \le \mathrm{i} < 5; \ 0 \le \mathrm{j} < 2 \tag{5.5.1-9}$$

5.5.2 Random number generation

A pseudo-random vector e_{rand} of 160 samples is generated as follows,

$$NELPseed = (521 \times NELPseed + 259)/32768 \tag{5.5.2-1}$$

$$e_{rand}(n) = NELPseed \quad , \quad 0 \le n < 160 \tag{5.5.2-2}$$

For Rate 1/4, the seed is initialized at the beginning of each frame to a signed 16-bit number that is derived from the bits representing the LSF.

5.5.3 Creation of sparse non-zero excitation

The above pseudo-random vector $e_{rand}(n)$ is divided into 10 subframes, and each subframe is made sparse by retaining the 4 largest absolute magnitude samples, and zeroing out the remaining 12. Each of these 10 subframes of the sparse vectors is denoted by $e_{sparse}(i,n)$, $0 \le i < 10$; $0 \le n < 16$. These sparse vectors are then scaled by their appropriate gains to create 160 samples of a sparse, excitation signal

$$r_1(16i+n) = \sqrt{3}\hat{G}(i)e_{sparse}(i,n), \quad 0 \le i < 10; 0 \le n < 16 \tag{5.5.3-1}$$

5.5.4 Shaping the excitation

The above excitation $r_1(n)$ is filtered by a 12th order bandpass filter, in order to remove the low band and high components to yield the signal $r_2(n)$. This bandpass filter is given by the following transfer function

$$BP(z) = \frac{BP_{n0} + BP_{n1}z^{-1} + ... + BP_{n12}z^{-12}}{1 + BP_{d1}z^{-1} + ... + BP_{d12}z^{-12}} \tag{5.5.4-1}$$

where the coefficients BP_{ni}, and BP_{di} are given in the tables **bp1_num_coef**, and **bp1_den_coef** respectively in the C-code.

$r_2(n)$ is then filtered by the 10th order shaping filter $SF1(z)$ to give the signal $r_3(n)$. This shaping filter is given by the following transfer function

$$SF1(z) = \frac{SF1_{n0} + SF1_{n1}z^{-1} + ... + SF1_{n12}z^{-10}}{1 + SF1_{d1}z^{-1} + ... + SF1_{d12}z^{-10}} \tag{5.5.4-2}$$

where the coefficients $SF1_{ni}$, and $SF1_{di}$ are given in the tables **shape1_num_coef**, and **shape1_den_coef** respectively in the C-code.

$r_3(n)$ is then scaled according to the following equation to yield

$$r_4(n) = \left(\sqrt{\frac{\sum_{n=0}^{159} [r(n)]^2}{\sum_{n=0}^{159} [r_3(n)]^2}} \right) r_3(n) \tag{5.5.4-3}$$

$r(n)$ and $r_4(n)$ are filtered by the 10th order lowpass filter $LP(z)$ to give $r_L(n)$ and $r_{4L}(n)$. Similarly, $r(n)$ and $r_4(n)$ are filtered by the 10th order highpass filter $HP(z)$ to give $r_H(n)$ and $r_{4H}(n)$.

$$LP(z) = \frac{LP_{n0} + LP_{n1}z^{-1} + ... + LP_{n12}z^{-10}}{1 + LP_{d1}z^{-1} + ... + LP_{d12}z^{-10}} \tag{5.5.4-4}$$

$$HP(z) = \frac{HP_{n0} + HP_{n1}z^{-1} + ... + HP_{n12}z^{-10}}{1 + HP_{d1}z^{-1} + ... + HP_{d12}z^{-10}} \tag{5.5.4-5}$$

where the coefficients LP_{ni}, and LP_{di} are given in the tables **txlpf1_num_coef**, and **txlpf1_den_coef** respectively in the C-code, and the coefficients HP_{ni}, and HP_{di} are given in the tables **txhpf1_num_coef**, and **txhpf1_den_coef** respectively in the C-code.

Two factors R_L, and R_H are computed as follows

$$R_L = 10\log 10\left(\frac{\sum_{n=0}^{n=159}[r_L(n)]^2}{\sum_{n=0}^{n=159}[r_{4L}(n)]^2}\right) \quad (5.5.4\text{-}6)$$

and

$$R_H = 10\log 10\left(\frac{\sum_{n=0}^{n=159}[r_H(n)]^2}{\sum_{n=0}^{n=159}[r_{4H}(n)]^2}\right) \quad (5.5.4\text{-}7)$$

A filter index F_{idx} is computed as follows from R_L, and R_H, and sent to the decoder.

$$F_{idx} = \begin{cases} 1 & ; \quad R_L < -3 \\ 2 & ; \quad R_L \geq -3 \text{ and } R_H < -3 \\ 0 & ; \quad otherwise \end{cases} \quad (5.5.4\text{-}8)$$

The shaping filter $SF(z)$ is determined from F_{idx} as follows

$$SF(z) = \begin{cases} SF1(z) & ; \quad F_{idx} = 0 \\ SF1(z)SF2(z) & ; \quad F_{idx} = 1 \\ SF1(z)SF3(z) & ; \quad F_{idx} = 2 \end{cases} \quad (5.5.4\text{-}9)$$

where, $SF2(z)$, $SF3(z)$ are

$$SF2(z) = \frac{SF2_{n0} + SF2_{n1}z^{-1} + ... + SF2_{n12}z^{-10}}{1 + SF2_{d1}z^{-1} + ... + SF2_{d12}z^{-10}} \quad (5.5.4\text{-}10)$$

$$SF3(z) = \frac{SF3_{n0} + SF3_{n1}z^{-1} + ... + SF3_{n12}z^{-10}}{1 + SF3_{d1}z^{-1} + ... + SF3_{d12}z^{-10}} \quad (5.5.4\text{-}11)$$

where the coefficients $SF2_{ni}$, $SF2_{di}$, $SF3_{ni}$, and $SF3_{di}$ are given in the tables **shape2_num_coef**, **shape2_den_coef,** **shape3_num_coef**, and **shape3_den_coef** respectively in the C-code.

Finally, $r_3(n)$ is filtered by $SF(z)$ to provide the quantized formant residual $r_q(n)$ for the Rate 1/4 frame.

5.6 Excitation Coding at Rate 1 and Rate 1/2

5.6.1 Introduction to Excitation Coding for Rate 1 and Rate 1/2

The coding of the excitation for Rate 1 and Rate 1/2 is performed for each subframe. Since the coding approach for both rates is the same, the subframe processing loop in the SMV encoder is shared between Rate 1 and Rate 1/2, while some internal sections of the subframe processing loop are distinct for each rate and for each frame type.

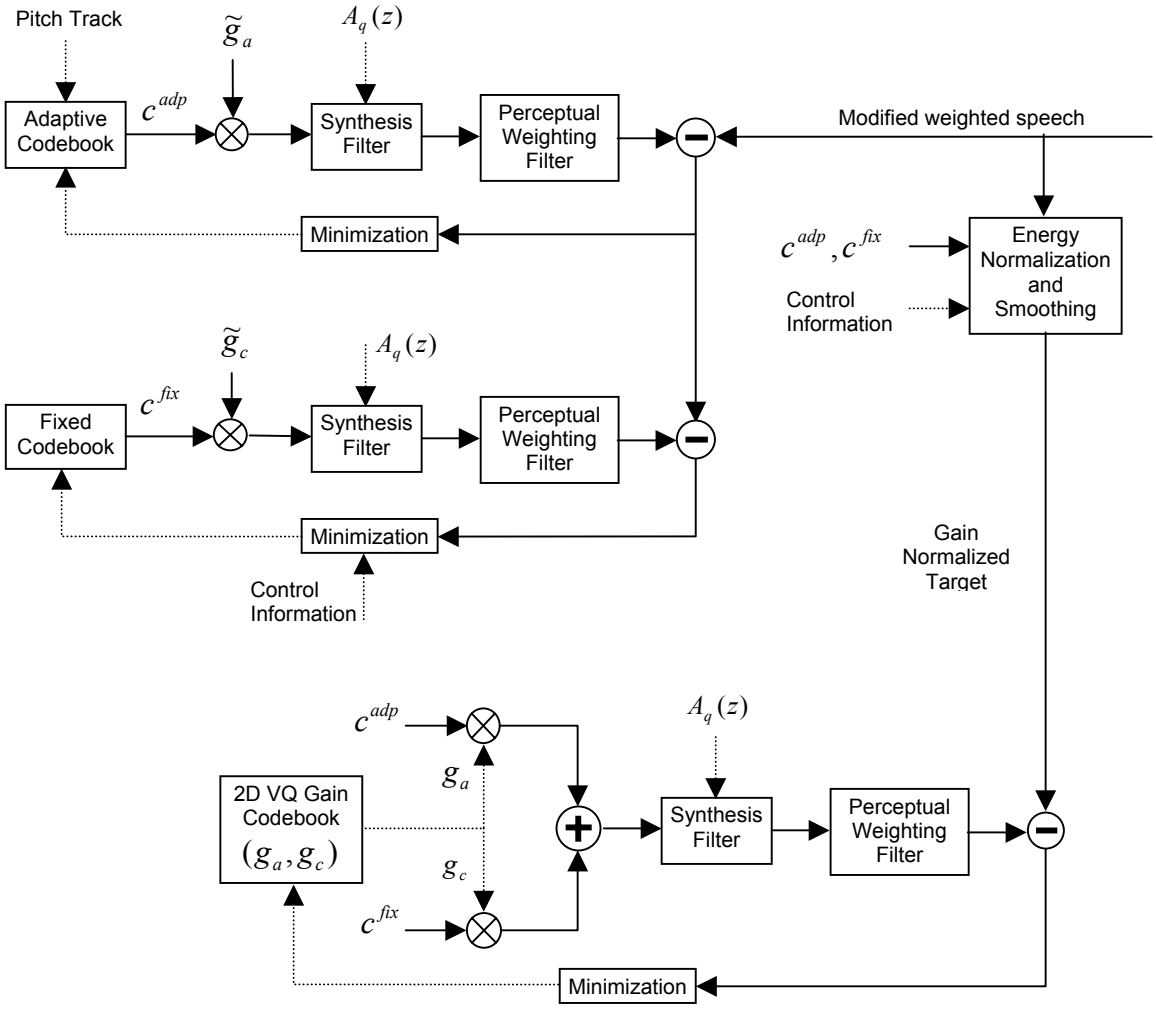

Figure 5.6-1: Subframe Excitation Coding for Rate 1 and Rate 1/2 Type-0 Frames

The subframe excitation coding for type-0 frames is depicted in Figure 5.6-1. The subframe processing includes a search for an adaptive-codebook contribution, a fixed-codebook contribution, and the joint quantization of the adaptive-codebook gain and the fixed-codebook gain, using gain-normalized target signal. The choice of the codebook vector and

the quantization of the gains are completed for each subframe before the next subframe is processed.

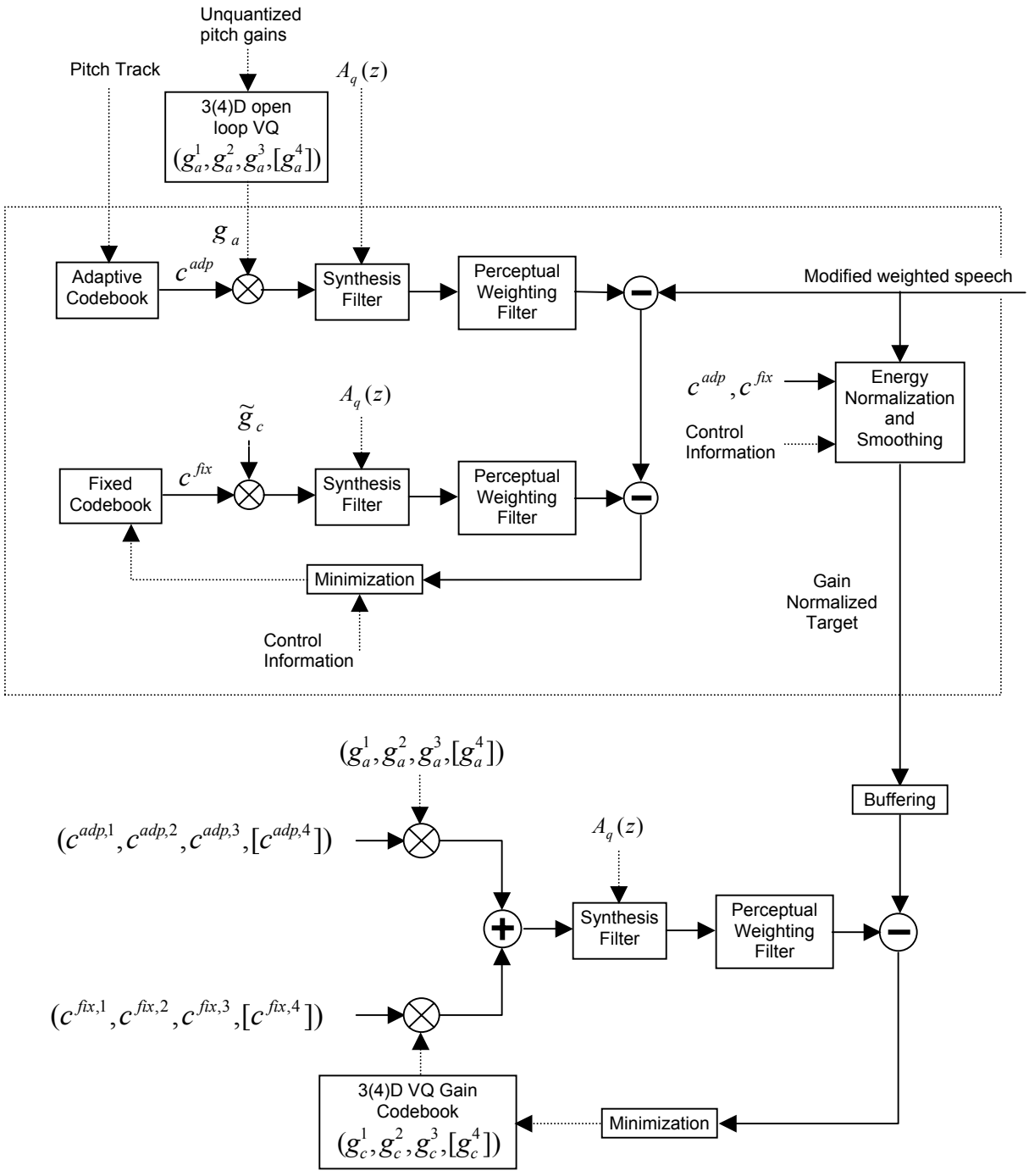

Figure 5.6-2: Subframe excitation coding for Rate 1 and Rate 1/2 type-1 frames

The subframe excitation coding for type-1 frames is depicted in Figure 5.6-2. Before the subframe is processed, the adaptive-codebook gains for all of the subframes are jointly quantized. The subframe processing includes only the search for the fixed-codebook contribution, since the adaptive-codebook contribution is obtained from the interpolated pitch contour. After the fixed-codebook contribution is determined for all subframes, the fixed-codebook gains are jointly quantized, using gain-normalized target signal.

Before the subframe processing loop is executed, the number of subframes is set, according to the rate and the frame type. For Rate 1 the number of subframes is 4, for Rate 1/2 type-0 frames the number of subframes is 2, and for Rate 1/2 type-1 frames the number of subframes is 3. At the beginning of the subframe loop the number of samples in the subframe, L_{SF} , is set, according to the following table:

	Rate 1		Rate 1/2	
	Type-0 frames	Type-1 frames	Type-0 frames	Type-1 frames
1st subframe	40	40	80	53
2nd subframe	40	40	80	53
3rd subframe	40	40	–	54
4th subframe	40	40	–	–

Table 5.6-1: L_{SF} - Number of Samples in a Subframe for Rate 1 and Rate 1/2

Sections 5.6.2 to 5.6.17 describe the subframe processing for Rate 1 and Rate 1/2, and are repeated for all 2,3, or 4 subframes. Note that the subframe processing loop is performed even when the selected rate is Rate 1/4 or Rate 1/8; however, the loop is performed without the excitation search and is needed only to keep the proper memory updates. For some of the symbols in the description of the excitation in the following description, the index of the subframe is omitted for simplicity.

5.6.2 Calculation of Impulse Response of the Combined Synthesis and Weighting Filter

Routine name: `LPC_ImpulseResonse`

Input:

- Weighing filter numerator coefficients for the subframe, $\hat{a}_m^{num}(i)$
- Weighing filter denominator coefficients for the subframe, $\hat{a}_m^{den}(i)$
- Interpolated quantized prediction coefficients for the subframe, $\hat{a}_m^{q}(i)$

Output:

- Impulse response of the combined filter for the subframe, $h_{imp}(n)$

The synthesis and weighing filter are combined in the form:

$$F(z) = \frac{A^{num}(z)}{A(z)A^{den}(z)} \tag{5.6.2-1}$$

The filter $A^{num}(z)$ uses the coefficients $\hat{a}_m^{num}(i)$ and the filter $A^{den}(z)$ uses the coefficients $\hat{a}_m^{den}(i)$, calculated according to Section 5.3.5 for Rate 1 and according to Section 5.3.24 for Rate 1/2. The filter $A(z)$ uses the interpolated quantized prediction coefficients, $\hat{a}_m^{q}(i)$, obtained according to Section 5.3.23. The impulse response of the combined filter for the length of the subframe is calculated and stored in $h_{imp}(n)$.

5.6.3 Generating the Target Signal

Routine name: `PRC_TargetSignal`

Input:

- Weighing filter numerator coefficients for the subframe, $\hat{a}_m^{num}(i)$
- Weighing filter denominator coefficients for the subframe, $\hat{a}_m^{den}(i)$
- Interpolated quantized prediction coefficients for the subframe, $\hat{a}_m^{q}(i)$
- Modified speech, $s_{\mathrm{mod}}(n)$

Output:

- Target signal for closed-loop pitch search or pitch interpolation, $T_{gs}(n)$
- Residual modified signal, $r_{\mathrm{mod}}(n)$, concatenated to the adaptive codebook, $C^{adp}(n)$

For each subframe (with a size defined by the rate and by the frame type), the modified speech signal, $s_{\mathrm{mod}}(n)$ (Section 5.3.19), is filtered by the weighting filter to generate the target signal for the closed-loop search (type-0 frames) and the pitch interpolation (type-1 frames). The zero-input response of the previous frame is removed from the synthesis filter. The filtering is performed in 2 steps, and result in the modified residual signal, as well as the target signal.

At the first step, the residual signal $r_{\mathrm{mod}}(n)$ is generated from the modified speech $s_{\mathrm{mod}}(n)$ by the all-zero filter $A(z)$, where the filter coefficients are the interpolated prediction coefficients for each subframe. The residual for the current subframe is concatenated to the adaptive codebook from the previous subframes, to generate the adaptive codebook, $C^{adp}(n)$, for the current subframe.

At the second step, the target signal $T_{gs}(n)$ is generated from the residual signal $r_{\mathrm{mod}}(n)$ by the combined filter:

$$F(z) = \frac{A^{num}(z)}{A(z)A^{den}(z)} \tag{5.6.3-1}$$

The zero-input response removal is performed by the manipulation of the states of the $A(z)$ and $A^{den}(z)$ filters.

5.6.4 Generating the Ideal Excitation

Routine name: `PRC_Ideal_Excit`

Input:

- Weighing filter numerator coefficients for the subframe, $\hat{a}_m^{num}(i)$
- Weighing filter denominator coefficients for the subframe, $\hat{a}_m^{den}(i)$
- Interpolated quantized prediction coefficients for the subframe, $\hat{a}_m^{q}(i)$
- Target signal for closed-loop pitch search or pitch interpolation, $T_{gs}(n)$

Output:

- Ideal excitation, $I_{ext}(n)$

The ideal excitation $I_{ext}(n)$ is generated from the target signal $T_{gs}(n)$ as the zero-state response of the filter:

$$\hat{F}(z) = \frac{A(z)A_{den}(z)}{A_{num}(z)} \tag{5.6.4-1}$$

5.6.5 Adaptive-codebook Contribution for Type-1 Frames

Routine name: `LTP_PP_pitch_ext`

Input:

- Pitch contour, $Pit(n)$
- Target signal for closed-loop pitch search or pitch interpolation, $T_{gs}(n)$
- Adaptive codebook, $C^{adp}(n)$
- Impulse response of the combined filter for the subframe, $h_{imp}(n)$

Output:

- Adaptive-codebook contribution, $c_{unf}^{adp}(n)$
- Filtered adaptive-codebook contribution, $c_{f}^{adp}(n)$
- Fractional pitch for the subframe, l_{FRC}^{p}
- Integer pitch for the subframe, l_{INT}^{p}
- Normalized closed-loop pitch correlation, R_{p}^{wgt}

For type-1 frames, the adaptive-codebook vector contribution for the subframe is generated using the interpolated pitch contour, $Pit(n)$ (Section 5.3.14.2). The filtered adaptive-codebook contribution is obtained by the convolution of the adaptive-codebook vector, and the normalized adaptive-codebook correlation is derived.

5.6.5.1 The Adaptive-codebook Vector Contribution

The adaptive codebook vector contribution for the subframe, $c_{unf}^{adp}(n)$, is obtained by pitch-filtering the adaptive codebook $C^{adp}(n)$ according to:

$$c_{unf}^{adp}(n) = C^{adp}(n) = C^{adp}(n - Pit(n)) \quad n = 0,\ldots,L_{SF} - 1 \tag{5.6.5.1-1}$$

Since $Pit(\cdot)$ can have fractional values, a set of Sinc windows (sampled at 21 different phases) is used to obtain $c_{unf}^{adp}(n)$ values at fractional time locations. The fractional pitch for the subframe, l_{FRC}^{p}, is set to $Pit(L_{SF}/2)$ and l_{INT}^{p} is set to the integer part of the fractional pitch, i.e. $I(l_{FRC}^{p})$.

5.6.5.2 The Filtered Adaptive Vector Contribution and Normalized Pitch Correlation

The filtered adaptive-codebook vector, $c_f^{adp}(n)$, is obtained by a zero-state convolution of $c_{unf}^{adp}(n)$ with $h_{imp}(n)$. The normalized pitch correlation is obtained by:

$$R_p^{wgt} = \frac{\max\left(\sum_{n=0}^{L_{SF}-1} T_{gs}(n) \cdot c_f^{adp}(n)\ ,\ 0\right)}{\sqrt{\sum_{n=0}^{L_{SF}-1}\left(T_{gs}(n)\right)^2 \cdot \sum_{n=0}^{L_{SF}-1}\left(c_f^{adp}(n)\right)^2}} \qquad (5.6.5.2\text{-}1)$$

5.6.6 Adaptive-codebook Contribution For Rate 1 Type-0 Frames

Routine name: `LTP_close_8_5k_pitch`

Input:

- Target signal for closed-loop pitch search or pitch interpolation, $T_{gs}(n)$
- Adaptive codebook, $C^{adp}(n)$
- Impulse response of the combined filter for the subframe, $h_{imp}(n)$
- Integer open-loop pitch for the subframe, l_p^{SF}
- Signal modification classifier, C_{SM}
- Fractional pitch lag of previous subframe, $l_{FRC}^{p,prv}$

Output:

- Adaptive-codebook contribution, $c_{unf}^{adp}(n)$
- Filtered adaptive-codebook contribution, $c_f^{adp}(n)$
- Fractional pitch for the subframe, l_{FRC}^{p}
- Integer pitch for the subframe, l_{INT}^{p}
- Normalized closed-loop pitch correlation, R_p^{wgt}
- Adaptive-codebook gain (closed-loop pitch gain), g_a

For Rate 1 type-0 frames the pitch is represented by 8 bits for the first and third subframes, and by 5-bit differential coding for the second and the fourth subframes. The closed-loop pitch search interval is $l_p^{SF} \pm 3$ for the first subframe and $l_p^{SF} \pm 3$ for the third subframe. For the second and the fourth subframes the closed-loop pitch search interval is $l_{FRC}^{p,prv} \pm 4.328125$ around the closed-loop pitch of the previous subframe, as defined in the differential-pitch codebook. For the fourth subframe, if $C_{SM} > 0$, the closed-loop pitch search interval is $\left[l_{FRC}^{p,prv}(2) - 4.328125, \ldots, l_{FRC}^{p,prv}(2) + 4.328125\right] \cap \left[l_p^{SF} - 1, \ldots, l_p^{SF} + 1\right]$ where $l_{FRC}^{p}(2)$ denotes the closed-loop pitch of the third subframe. (If the intersection is found to be an empty set, a single point, $l_{FRC}^{p,prv}(2) - 4.328125$ or $l_{FRC}^{p,prv}(2) + 4.328125$ is used.) Note that the resulting search pitch interval is further constrained so that it is always a subset of the allowable pitch interval $[17, \ldots, 148]$.

5.6.6.1 Normalized Cross Correlation Between the Target and the Filtered Excitation

The normalized correlations between the filtered adaptive codebook and target signal at the integer values of the search interval are computed. The number of normalized correlation values is the number of integer points in the interval, with extra 5 integer points in each direction for further interpolation. For each lag l in the search interval, the filtered adaptive codebook, $C_{flt,l}^{adp}(n)$ is obtained by convolving the adaptive codebook $C^{adp}(n)$ with $h_{imp}(n)$, using a recursive filtering procedure. The normalized correlation for each lag, $R_{nrm}^{adp}(l)$, is given by:

$$R_{nrm}^{adp}(l) = \frac{\sum_{n=0}^{L_{SF}-1} T_{gs}(n) \cdot C_{flt,l}^{adp}(n)}{\sqrt{\sum_{n=0}^{L_{SF}-1} \left(C_{flt,l}^{adp}(n)\right)^2}} \qquad (5.6.6.1\text{-}1)$$

where $L_{SF} = 40$.

5.6.6.2 Fractional Pitch Search and Quantization

The integer closed-loop pitch is obtained as the lag l that maximizes the normalized correlation, $R_{nrm}^{adp}(l)$. For the last subframe, if the maximum of the normalized correlation is negative, and $C_{SM} > 0$, the integer closed-loop pitch is set to the open-loop pitch lag.

The fractional pitch is searched on the search interval of $[-0.9,\ldots,0.9]$ around the integer closed-loop pitch lag. The allowable fractional pitch lag values on that interval are defined by the direct pitch quantization table for the first and the third subframes, and by the differential quantization table for the second and the fourth subframes. The fractional pitch is obtained by a band-limited interpolation of the normalized correlation $R_{nrm}^{adp}(l)$, calculated in Section 5.6.6.1, at the allowable fractional pitch lags on the search interval, and choosing the pitch by the lag that maximizes the interpolated normalized correlation. The adaptive-codebook contribution is constructed in the vector $c_{unf}^{adp}(n)$. l_{INT}^{p} is simply the rounded version of l_{FRC}^{p}.

5.6.6.3 Adaptive-codebook Gain (Closed-Loop Pitch Gain) and Normalized Correlation

The adaptive-codebook vector $c_{unf}^{adp}(n)$ is convolved with $h_{imp}(n)$, to generate the filtered adaptive-codebook vector $c_{f}^{adp}(n)$. The closed-loop pitch gain is given by:

$$g_a = \frac{\sum_{n=0}^{L_{SF}-1} T_{gs}(n) \cdot c_{f}^{adp}(n)}{\sum_{n=0}^{L_{SF}-1} c_{f}^{adp}(n) \cdot c_{f}^{adp}(n)} \qquad (5.6.6.3\text{-}1)$$

where $L_{SF} = 40$. The closed-loop normalized correlation is given by:

$$R_p^{wgt} = \frac{\sum_{n=0}^{L_{SF}-1} T_{gs}(n) \cdot c_f^{adp}(n)}{\sqrt{\sum_{n=0}^{L_{SF}-1} c_f^{adp}(n) \cdot c_f^{adp}(n) \cdot \sum_{n=0}^{L_{SF}-1} T_{gs}(n) \cdot T_{gs}(n)}} \quad (5.6.6.3\text{-}2)$$

5.6.7 Adaptive-codebook Contribution For Rate 1/2 Type-0 Frames

Routine name: `LTP_close_7b_pitch`

Input:

- Target signal for closed-loop pitch search or pitch interpolation, $T_{gs}(n)$
- Adaptive codebook, $C^{adp}(n)$
- Impulse response of the combined filter for the subframe, $h_{imp}(n)$
- Integer open-loop pitch for the subframe, l_p^{SF}
- Signal modification classifier, C_{SM}

Output:

- Adaptive-codebook contribution, $c_{unf}^{adp}(n)$
- Filtered adaptive-codebook contribution, $c_f^{adp}(n)$
- Fractional pitch for the subframe, l_{FRC}^{p}
- Integer pitch for the subframe, l_{INT}^{p}
- Normalized closed-loop pitch correlation, R_p^{wgt}
- Adaptive-codebook gain, g_a

For Rate 1/2 type-0 frames the pitch lag is represented by 7 bits for the first and the second subframes, which represent the integer pitch lags on the range $[21,\ldots,148]$. The closed-loop pitch search for Rate 1/2 type-0 is similar to the closed-loop pitch search for Rate 1 type-0 frames, except only integer pitch values are searched. The closed-loop search interval is $l_p^{SF} \pm 3$ around the open-loop pitch lags. If the open-loop pitch is smaller than 21, it is multiplied by 2 so that it fits into the allowable pitch range. For the second subframe, if $C_{SM} > 0$ and if $l_p^{SF}(1)$ is larger than 60, the closed-loop pitch search interval is reduced to $l_p^{SF}(1) \pm 1$.

5.6.7.1 Normalized Cross Correlation Between the Target and the Filtered Excitation

The normalized cross correlation for each lag, $R_{nrm}^{adp}(l)$, between the filtered adaptive codebook and target signal at the integer values of the search interval are computed, as described for Rate 1 type-0 frames in Section 5.6.6.

5.6.7.2 Integer Pitch Search and Quantization

The integer closed-loop pitch lag l_{INT}^{p} is obtained as the lag that maximizes the normalized correlation. For the last subframe, if the maximum normalized correlation is negative and $C_{SM} > 0$, the integer closed-loop pitch lag is set to the open-loop pitch. The adaptive-codebook contribution, $c_{unf}^{adp}(n)$, is constructed according to the integer closed-loop pitch l_{INT}^{p}. l_{FRC}^{p} in this case is set to l_{INT}^{p}.

5.6.7.3 Adaptive-codebook Gain (Closed-Loop Pitch Gain) and Normalized Correlation

Similar to Rate 1 type-0 frames, the adaptive-codebook vector $c_{unf}^{adp}(n)$ is convolved with $h_{imp}(n)$, to generate the filtered adaptive-codebook vector $c_{f}^{adp}(n)$. The closed-loop pitch gain, g_a, and the normalized pitch correlation is calculated according to Equations (5.6.6.3-1) and (5.6.6.3-2) in Section 5.6.6.3, with $L_{SF} = 80$.

5.6.8 Scaling of Adaptive-Codebook Gain

In order to enhance the perceptual performance of the fixed-codebook search, the adaptive-codebook gain is scaled down before it is used to generate the target signal for the fixed-codebook search, $T_g(n)$. In this way, $T_g(n)$ would have a small amount of periodicity, which encourages placing of pulses around high-energy segments during the pulse-codebook search (i.e., around the pulse spike in the case of voiced speech).

For type-1 frames, if $F_{noisyV} < 2$, the open-loop pitch gain, g_a, is scaled down according to:

$$g_a \Leftarrow g_a \cdot \left(0.3 \cdot R_p^{wgt} + 0.7\right) \tag{5.6.8-1}$$

For type-0 frames, the closed-loop pitch gain, g_a, is scaled down according to:

$$g_a \Leftarrow g_a \cdot \left(0.5 \cdot R_p^{wgt} + 0.5\right) \tag{5.6.8-2}$$

5.6.9 Update of Target and Ideal Excitation

The target for the fixed-codebook search, $T_g(n)$, is generated from the target for the adaptive-codebook search, $T_{gs}(n)$, the filtered adaptive-codebook contribution, $c_f^{adp}(n)$, and the closed-loop pitch gain, g_a, according to:

$$T_g(n) = T_{gs}(n) - g_a \cdot c_f^{adp}(n) . \tag{5.6.9-1}$$

The ideal excitation is updated, using the unfiltered adaptive-codebook contribution, $c_{unf}^{adp}(n)$, and the closed-loop pitch gain, g_a, according to:

$$I_{ext}(n) \Leftarrow I_{ext}(n) - g_a \cdot c_{unf}^{addp}(n) . \tag{5.6.9-2}$$

5.6.10 Calculation of Fixed-codebook Smoothing Parameters

Routine name: `SMO_excitation_analysis`

Input:

- Modified speech, $s_{\mathrm{mod}}(n)$
- Ideal excitation, $I_{ext}(n)$
- Fractional pitch for the subframe, l_{FRC}^{p}
- Voice activity decision, VAD
- First element of interpolated reflection coefficient vector for the sub frame, $\hat{k}_m(0)$

Output:

- Subframe mode factor, C_{sub_mod}
- Gain normalization factor, β_E

The routine generates two factors for the subframe, one used for sub-codebook selection, and the other for gain normalization. A counter of the consecutive inactive voiced subframes (according to the frame VAD decision), C_{VAD}, is kept and is limited to 8. An internal counter of the number of noise updates, C_{update_noise}, is obtained and limited by 20. An internal counter of the number of speech updates, C_{update_speech}, are obtained and limited by 60. Using the noise update counter and the speech update counter, the maximum of the SNR, $SNR_{\max}$, is updated, using an average of the maximum of the absolute values during noise segments, $\overline{N}_{\max}$ and the average of the maximum of the absolute values during speech segments, $\overline{S}_{\max}$. The maximum of the SNR is used to set a level of adaptation, C_{adpt}.

The normalized correlation is calculated on the modified speech for the integer portion of the pitch lag, $\lfloor l_{FRC}^{p} \rfloor$, and for $\lfloor l_{FRC}^{p} \rfloor + 1$ are calculated according to:

$$R_1 = \frac{\sum_{n=0}^{79} s_{\mathrm{mod}}(n) \cdot s_{\mathrm{mod}}\left(n - \lfloor l_{FRC}^{p} \rfloor\right)}{\sqrt{\sum_{n=0}^{79} s_{\mathrm{mod}}^2(n) \cdot s_{\mathrm{mod}}(n) \sum_{n=0}^{79} s_{\mathrm{mod}}^2\left(n - \lfloor l_{FRC}^{p} \rfloor\right)}} \tag{5.6.10-1}$$

$$R_2 = \frac{\sum_{n=0}^{79} s_{\mathrm{mod}}(n) \cdot s_{\mathrm{mod}}\left(n - \lfloor l_{FRC}^{p} \rfloor +\right)}{\sqrt{\sum_{n=0}^{79} s_{\mathrm{mod}}^2(n) \cdot s_{\mathrm{mod}}(n) \sum_{n=0}^{79} s_{\mathrm{mod}}^2\left(n - \lfloor l_{FRC}^{p} \rfloor + 1\right)}} \tag{5.6.10-2}$$

(If $\lfloor l_{FRC}^{p} \rfloor = 148$, the second correlation is calculated for the lag 147.)

The maximum between both correlations is kept, and is denoted R_{L^p}. A moving average of R_{L^p} is obtained by:

$$\overline{R}_{L^p} = 0.9 \cdot \overline{R}_{L^p} + 0.1 \cdot R_{L^p} \tag{5.6.10-3}$$

A buffer of maximum of absolute values in the subframe, and a buffer of sum of absolute values in the frame, for the last 15 subframes, are maintained and undated. The buffer of maximum absolute values is divided into 5 groups, each of 3 subframes, and the derivative of the maximum of absolute sum for each group is obtained.

Using the maximum of absolute values for the current subframe, the level of adaptation C_{adpt}, a tracking level of the noise, the average pitch correlation R_{L^p}, and the VAD decision, a binary speech mode, F_{speech}, is obtained.

Update of the noise or the speech parameters is performed, using the parameters described above for a decision between the update of noise parameters or the update of speech parameters, and the rate of the updates.

The maximum of the absolute value of the ideal excitation subframe is obtained, and denoted by $I_{\max}$. The ratio between $I_{\max}$ and a long term average of it, $\bar{I}_{\max}$, is denoted $R_{\max}$. The speech mode, F_{speech}, and $R_{\max}$ are used to generate the binary excitation mode factor, C_{ext_mod}, and the gain smoothing factor, β_E.

The parameters described above are used for the proper update of the long-term average $\bar{I}_{\max}$.

For each subframe, the frame class C_{FR} is multiplied by C_{ext_mod} to generate the subframe-modified class, C_{sub_mod}. Further, if $F_{noisyV} = 2$, C_{sub_mod} is also set to 2.

5.6.11 Principles and Algorithms for Pulse Codebook Search

The fixed-codebook contribution is obtained by a search through several sub-codebooks (with the exception of Rate 1 type-1 frames), and by choosing the best contribution (pulse location and signs, or random vector) and the sub-codebook that contains that best contribution. Three sub-codebooks of pulse structure are used for Rate 1 type-0 frames. A single codebook of pulse structure is used for Rate 1 type-1 frame. Two sub-codebooks of pulse structure and a Gaussian sub-codebook are used for Rate 1/2 type-0 frames. Three sub-codebooks of pulse structure are used for Rate 1/2 type-1 frames.

5.6.11.1 Analysis-by-Synthesis Approach for Fixed-Codebook Contribution

The sub-codebooks are searched for the best fixed-codebook contribution. The best sub-codebook, and the best combination of pulses (or random excitation) in the chosen sub-codebook, are selected for the fixed-codebook contribution. The selection between the sub-codebooks is based on a distance measure that combines the minimization of the mean squared error between the filtered excitation and the target signal (known as closed-loop analysis-by-synthesis approach) and a set of perceptual criteria.

The analysis-by-synthesis approach require finding the fixed-codebook excitation vector, c, which minimizes the mean-square error measure (the time domain index, n, is omitted in the following equations):

$$\varepsilon = \left\| T_g - gHc \right\|^2 \tag{5.6.11.1-1}$$

where H is the lower-triangular matrix, where each column includes the impulse response, with the first element on the diagonal. It is well known that by solving Equation (5.6.11.1-1) for the optimal gain and substituting the optimal gain into Equation (5.6.11.1-1), it can be shown that the minimizing ε_i is equivalent to maximizing:

$$\tilde{\varepsilon} = \frac{\left(\left(T_g \right)^T Hc \right)^2}{\left\| Hc \right\|^2} = \frac{\left(\left(H^T T_g \right)^T c \right)^2}{c^T H^T Hc} \tag{5.6.11.1-2}$$

If we define $b_{Tg} = \left(H^T T_g \right)^T$ and $y = Hc$:

$$\tilde{\varepsilon} = \frac{\left(b_{Tg} c \right)^2}{y^T y} \ . \tag{5.6.11.1-3}$$

5.6.11.2 Principles of Iterative Search Procedure for Pulse Sub-Codebooks

The fixed-codebook search for the pulse sub-codebooks (or a codebook) is based on an iterative algorithm. The algorithm searches for the best pulse position of each pulse or a pair of pulses, while keeping all the other pulses at their previously determined positions. After the locations for all the pulses in a particular sub-codebook are searched, the procedure can be repeated for further refinement of the pulse locations. The repeated refinement is called “a turn”. During turn 0, the initial locations for the pulses in the sub-codebook are determined.

The fast-search iterative algorithm is based on the linearity of the elements in the numerator and the denominator. Let the vector c^- to include all the other pulses in the sub-codebook, except of the i^{th} pulse, i.e., $c_i = c^- + p_i$, where p_i is a vector that includes

a single pulse at the location j_i and zeros elsewhere. Define $y^- = Hc^-$, then $y = y^- + Hp_i$. Equation (5.6.11.1-3), for the additional i^{th} pulse, can be written as:

$$\tilde{\varepsilon}_i = \frac{\left(b_{Tg}c\right)^2}{\|y\|^2} = \frac{\left[b_{Tg}\left(c^- + p_i\right)\right]^2}{\left(y^- + Hp_i\right)^T\left(y^- + Hp_i\right)} = \frac{\left[\left(b_{Tg}c\right)^- + b_{Tg}p_i\right]^2}{\|y^-\|^2 + 2\left(y^-\right)^T Hp_i + \|Hp_i\|^2} \quad (5.6.11.2\text{-}1)$$

The fast search algorithm is based on the pre-calculation of the terms $b_{Tg}p_i$, Hp_i, and $\|Hp_i\|^2$ for any possible pulse location in the subframe. The evaluation of numerator and the denominator of $\tilde{\varepsilon}_i$ requires only the calculation of the correlation between y^- and Hp_i, and 3 additions.

5.6.11.3 Updates of Pulse Contributions

When the best pulse is chosen, its contribution should be added to the overall pulse contribution, prior to the search for the next pulse to create the terms y_i^{update} and $\left(b_{Tg}c\right)_i^{update}$. (If all the pulses in the sub-codebook are included, $y_i^{update} = y$ and $\left(b_{Tg}c\right)_i^{update} = \left(b_{Tg}c\right)$). On the other hand, at each search for a new possible location for a specific pulse, the contribution of that pulse at its old location should be removed. The addition or the subtraction of the pulse contribution is done, simply, by:

$$y_i^{update} = y^- + Hp_i \quad (5.6.11.3\text{-}1)$$

$$y^- = y_i^{update} - Hp_i \quad (5.6.11.3\text{-}2)$$

$$\left(b_{Tg}c\right)_i^{update} = \left(b_{Tg}c\right)^- + b_{Tg}p_i \quad (5.6.11.3\text{-}3)$$

$$\left(b_{Tg}c\right)^- = \left(b_{Tg}c\right)_i^{update} - b_{Tg}p_i \quad (5.6.11.3\text{-}4)$$

5.6.11.4 Concept of Pitch Enhancement for Pulse Codebooks

Each pulse in the pulse codebook is called a main pulse. Each main pulse can be repeated within the subframe, with an appropriate gain factor, on intervals corresponding to the pitch lag of the subframe. Pulse insertion after the main pulse is called forward pitch enhancement, and pulse insertion before the main pulse is called backward pitch enhancement. The pre-calculation of $b_{Tg}p_i$, Hp_i, and $\|Hp_i\|^2$ include both the forward pitch enhancement and the backward pitch enhancement. Note, in particular, that if only forward pitch enhancement was used, a single vector, which includes the forward pitch-enhanced impulse response of a pulse at the beginning of the frame, would be sufficient to represent Hp_i with a shift of that vector, and the term $\left(y^-\right)^T Hp_i$ could be easily calculated. However, since the backward pitch enhancement includes non-causal contributions, up to

4 different vectors are needed to be pre-calculated to represent Hp_i, depending on the pitch lag of the subframe. (The maximal subframe length is 80, but in that case the minimum pitch lag is 21, hence the maximal number of different vectors is 4.) The appropriate representative vector is selected according to the index of the main pulse and the pitch lag of the subframe. The representative vectors are denoted by $H_p^m \quad m = 0,\ldots,3$ where m indicates the number of backward pitch enhancement pulses. Note that not all H_p^m need to be calculated, and the number depends on the subframe size and the pitch lag for the subframe

5.6.11.5 Pre-Calculation Procedure

Routine name: `FCS_PreCalc_FBCB`

Input:

- Target signal, $T_g(n)$
- Impulse response vector (possibly modified), $\hat{h}_{imp}(n)$
- Integer pitch lag for the subframe, l_{INT}^p
- Modified pitch gain, $\hat{g}_a$ (see Section 5.6.13.2 and 5.6.14.2)

The pitch enhancement coefficients are calculated by:

$$P_{mag}(m) = \left(\hat{g}_a\right)^m \quad m = 0,\ldots,4 \tag{5.6.11.5-1}$$

The representative vectors for Hp_i are calculated as follows. The vector H_p^0 is obtained by:

$$H_p^0(j) = \sum_{m=0}^{4} P_{mag}(m) \cdot \hat{h}_{imp}\left(j - m \cdot l_{INT}^p\right) \quad j = 0,\ldots,L_{SF}-1 \tag{5.6.11.5-2}$$

where $\hat{h}_{imp}(j)$ is a causal impulse response. The vectors $H_p^m \quad m = 1,\ldots,4$ are obtained recursively using:

$$H_p^m(j) = H_p^{m-1}\left(j - l_{INT}^p\right) + P_{mag}(m) \cdot \hat{h}_{imp}(j) \quad j = 0,\ldots,L_{SF}-1 \tag{5.6.11.5-3}$$

Note that $H_p^m \quad m = 1,\ldots,4$ are not evaluated if $m \cdot l_{INT}^p \geq L_{SF}$.

An auxiliary function, targeted for computation reduction during the search, is calculated as:

$$P_m(j) = m \quad \text{for} \quad m \cdot l_{INT}^p \leq j < (m+1) \cdot l_{INT}^p \quad j = 0,\ldots,L_{SF}-1 \tag{5.6.11.5-4}$$

The weighted filtered target signal is obtained by:

$$b_{Tg}(j) = \sum_{n=j}^{L_{SF}-1} T_g(n) \cdot \hat{h}_{imp}(n-j) \quad j = 0,\ldots,L_{SF}-1 \tag{5.6.11.5-5}$$

The pitch enhanced $b_{Tg}p_i$ is calculated for each index in the subframe (p_i such that $j_i = i$) by:

$$b_{Tg}p_i = \sum_{m=-5}^{5} P_{mag}(|m|) \cdot b_{Tg}(i + m \cdot l_{INT}^{p}), \; i = 0,\ldots,L_{SF}-1 \quad ;0 \le i + m \cdot l_{INT}^{p} < L_{SF} \; . \tag{5.6.11.5-6}$$

The signs of $b_{Tg}p_i$ are saved in a sign array for each i in the subframe according to:

$$\begin{aligned} S_i &= 1 \qquad b_{Tg}p_i > 0 \\ S_i &= -1 \qquad b_{Tg}p_i \le 0 \end{aligned} \tag{5.6.11.5-7}$$

The pitch enhanced $b_{Tg}p_i$ is then modified to its absolute value for each i in the subframe.

The energy of the pitch-enhanced and filtered pulse contribution, $\|Hp_i\|^2$, is obtained for each index in the subframe (p_i such that $j_i = i$) by:

$$\|Hp_i\|^2 = \sum_{l=0}^{L_{SF}-i-1+P_m(i)\cdot l_{INT}^{p}} \left(H_p^{P_m(i)}(l)\right)^2 \qquad i = 0,\ldots,L_{SF}-1 \tag{5.6.11.5-8}$$

using a recursive calculation procedure.

An alternate method of computing the filtered codebook energy factor, $\|y\|^2$ in equation 5.6.11.2-1, uses the correlation matrix of the impulse response, Φ, whose elements are defined by

$$\phi(i,j) = \sum_{l=MAX\left(\left(i-P_m(i).l_{INT}^{p}\right),\left(j-P_m(j).l_{INT}^{p}\right)\right)}^{L_{SF}-1} H_p^{P_m(i)}(l-i) H_p^{P_m(j)}(l-j) \;\; ; \; 0 \le i < L_{SF} \text{ and } 0 \le j < L_{SF} \tag{5.6.11.5-9}$$

5.6.11.6 An Iterative Algorithm Based on Position Search for a Single Pulse

Routine name: `FCS_Fast_Search_FBCB`

The iterative algorithm based on position search for a single pulse achieves an initial positioning or re-evaluation of all the pulse locations in the sub-codebook by searching for the best location of a single pulse at a time. The algorithm can perform any number of turns for further refinements of all pulse locations. At each turn, the best location for each pulse in the sub-codebook is searched while keeping the previous locations for all the other pulses in the sub-codebook. At the first turn the pulses are added sequentially, where the first pulse is positioned independently of the other pulses, the second pulse is added to the sub-codebook vector while retaining the first pulse, and so on, until the last pulse in the sub-codebook is positioned.

5.6.11.6.1Initialization of the Search Parameters

If the iterative algorithm is called for an initial positioning of the first pulse in the sub-codebook (first turn), the search parameters, $\left(b_{Tg}c\right)^{update}$ and y^{update}, are initialized to:

$$\left(b_{Tg}c\right)^{update} = 0 \tag{5.6.11.6-1}$$

$$y^{update}(j) = 0 \qquad j = 0,\ldots,L_{SF}-1 \tag{5.6.11.6-2}$$

If the iterative algorithm is called for a refinement of previously set of positions, the search parameters are initialized to:

$$\left(b_{Tg}c\right)^{update} = \sum_{P_i} b_{Tg}\,p_i \tag{5.6.11.6-3}$$

$$y^{update}(j) = \sum_{p_i} S_i \cdot H_p^{P_m(i)}\left(j - i + P_m(i)\cdot l_{INT}^p\right) \qquad j = 0,\ldots,L_{SF}-1 \tag{5.6.11.6-4}$$

where i runs over all the previously chosen pulses in the sub-codebook.

5.6.11.6.2The Search Procedure

The search procedure is structured of three nested loops. The external loop progresses the turns for further refinement of an already established set of pulse locations. The intermediate loop progresses sequentially from the first pulse to the last pulse in the sub-codebook, and refines the location of each individual pulse (tested pulse). Before the internal loop is entered, the contribution of the tested pulse in its previous location is removed from the search parameters as described in Section 5.6.11.3 by Equations (5.6.11.3-2) and (5.6.11.3-4).

In the internal search loop, $\widetilde{\varepsilon}_i$ is evaluated for all the possible track locations for the tested pulse. The error measure $\widetilde{\varepsilon}_i$ is evaluated using the procedure described in Section 5.6.11.2 by Equation (5.6.11.2-1). The location i^* that maximizes $\widetilde{\varepsilon}_i$ is selected.

At the end of inner loop, the contribution of the tested pulse at the selected position is added to the search parameters as described in Section 5.6.11.3 by Equations (5.6.11.3-1) and (5.6.11.3-3).

Alternatively, the filtered codebook energy factor, $\|y\|^2$ in Equation 5.6.11.2-1 can be computed as

$$\|y\|^2 = \sum_{k=0}^{N_p-1}\phi(p_k,p_k) + 2\sum_{k=0}^{N_p-2}\sum_{l=k+1}^{N_p-1} S_k S_l \phi(p_k,p_l) \tag{5.6.11.6-5}$$

where N_p is the number of pulses in the code-vector. It can be noted that the above equation is a general equation and the actual computation can be efficiently performed using a set of partial summations as the pulses are searched iteratively.

5.6.11.6.3Building the Pulse Excitation

After the final turn of sub-codebook refinement has been completed, the pulse excitation, including the forward and the backward pitch enhancement pulse, is constructed,

according to the selected pulse locations, the subframe pitch lag l_{INT}^{p}, and the pitch enhancement coefficients P_{mag}.

5.6.11.7 An Iterative Algorithm Based on Joint Position Search for Two Pulses

Routine name: `FCS_Fast_Search_FBCB2`

The iterative algorithm that is based on a joint position search for two pulses achieves an initial positioning or re-evaluation of all the pulse locations in the sub-codebook by jointly searching for the best locations of two pulses at a time. The algorithm can perform any number of turns for further refinements of all pulse locations. At each turn, the best locations for two pulses in the sub-codebook are searched while keeping the previous locations for all the other pulses in the sub-codebook. At the first turn pairs of pulses are added to the fixed-codebook contribution sequentially, where the first pair of pulses is positioned independently of the other pulses, the second pair is added, and so on, until the last pair in the sub-codebook is positioned.

Note that each turn in the joint search for two pulses is of higher computational complexity than is the search for a single pulse, but pre-search procedure can be used to limit the search complexity while keeping the advantages of the joint search.

5.6.11.7.1 Initialization of the Search Parameters

The initialization of the search parameters, for the case of the first turn or any sequential turn is performed exactly as described in Section 5.6.11.6.1 above.

5.6.11.7.2 Pre-Search for 2-Pulse Sub-Codebook

To reduce the search complexity in the case of 2-pulse sub-codebook (used for Rate 1/2 frames, see Section 5.6.14.1), a low-complexity pre-search procedure is performed to identify a set of candidates for further search. The number of pre-search candidates depends on the frame type and the subframe pitch lag, l_{INT}^{p}, according to:

$$N_{pre} = \begin{cases} 16 & \text{type-1 frame, } l_{INT}^{p} < 30 \\ 19 & \text{type-1 frame, } l_{INT}^{p} \geq 30 \\ 18 & \text{type-0 frame, } l_{INT}^{p} < 45 \\ 22 & \text{type-0 frame, } l_{INT}^{p} \geq 45 \end{cases} \qquad (5.6.11.7\text{-}1)$$

The pre-search identifies the N_{pre} pulse locations in each of the two tracks that maximize the error criterion:

$$\widehat{\varepsilon}_i = \frac{\left(b_{Tg}\, p_i\right)^2}{\left\| Hp_i \right\|^2} \qquad (5.6.11.7\text{-}2)$$

5.6.11.7.3 Pre-Search for 3-Pulse Sub-Codebook

The pre-search for the 3-pulse sub-codebook positions only the main pulse by maximizing the same error criterion of Equation (5.6.11.7-2) above. After the main pulse is positioned, the tracks for the second and the third pulse are dynamically allocated. See Section 5.6.14.1 for the details of the 3-pulse dynamic tracks. Note that the main pulse location is not modified any further.

5.6.11.7.4The Search Procedure

The search procedure is structured of four nested loops. The external loop progresses the turns for further refinement of an already established set of pulse locations. The second loop progresses on an arranged set of two pulses in the sub-codebook at a time, and refines the locations of the two pulses in the set (tested pulse pair). Before the internal loop is entered, for turn 1 or higher, the contributions of the tested pulse pair in their previous locations are removed from the search parameters in Section 5.6.11.3 by Equations (5.6.11.3-2) and (5.6.11.3-4).

The two internal search loops are performed on either all the possible track locations for the tested pulse pair, or on the pre-selected possible track locations for the tested pulse pair. The order of the two loops is structured such that the internal loop is performed for the pulse with the larger number of possible track locations. At the outer loop for the first pulse, which runs on all possible locations for this pulse, indexed by i_1, the search parameters are temporarily updated with the update procedure described in Section 5.6.11.2 by Equation (5.6.11.2-1), using $b_{Tg}p_{i_1}$, Hp_{i_1}, and $\left\|Hp_{i_1}\right\|^2$. In the inner loop for the second pulse, indexed i_2, the search parameters are updated using $b_{Tg}p_{i_2}$, Hp_{i_2}, and $\left\|Hp_{i_2}\right\|^2$. The error measure $\tilde{\varepsilon}_{i_1,i_2}$ is evaluated using the temporarily updated search parameters, and the locations i_1^* and i_2^* that maximize $\tilde{\varepsilon}_{i_1,i_2}$ are selected.

After the inner two loops are completed, in order to prepare the search parameters for the next pair search, the contribution of the tested pulse pair at their newly selected positions is added to the previous search parameters as described in Section 5.6.11.3 by Equations (5.6.11.3-1) and (5.6.11.3-3).

Alternatively, the filtered codebook energy factor, $\|y\|^2$ in Equation 5.6.11.2-1 can be computed as

$$\|y\|^2 = \sum_{k=0}^{N_p-1}\phi(p_k,p_k)+2\sum_{k=0}^{N_p-2}\sum_{l=k+1}^{N_p-1}S_kS_l\phi(p_k,p_l) \qquad \text{(5.6.11.7-3)}$$

where N_p is the number of pulses in the code-vector. It can be noted that the above equation is a general equation and the actual computation can be efficiently performed using a set of partial summations as the pulses are searched iteratively.

5.6.11.7.5Building the Pulse Excitation

After the final turn of sub-codebook refinement has been completed the pulse excitation, including the forward and the backward pitch enhancement pulse, is constructed, according to the selected pulse locations, the subframe integer pitch lag l_{INT}^p, and the pitch enhancement coefficients P_{mag}.

5.6.11.8 Joint Position Search for Five Pulses

Routine name: `FCS_Full_Search_FBCB_5p`

For Rate 1/2 type-1 frames, a special 5-pulse sub-codebook, described in Table 5.6-8, is used. The locations for the 5 pulses are searched with a joint positioning search. The joint positioning search includes 5 nested loops, which test all possible locations for the 5 pulses. For each pulse, indexed by i, the contribution to search parameters is added are temporarily with the update procedure described in Section 5.6.11.2 by Equation (5.6.11.2-1), using $b_{Tg} p_i$, Hp_i, and $\|Hp_i\|^2$. The error measure $\widetilde{\varepsilon}_{i_1,i_2,i_3,i_4,i_5}$ is evaluated using the updated search parameters, and the locations i_1^*, i_2^*, i_3^*, i_4^* and i_5^* that maximize $\widetilde{\varepsilon}_{i_1,i_2,i_3,i_4,i_5}$ are selected.

5.6.12 Gaussian Codebook Search

Routine name: GCB_gauss_cb

The Gaussian codebook is searched in a closed-loop manner in order to minimize the perceptual mean-squared error between the original and the reconstructed speech. The Gaussian excitation vector is constructed from two orthogonal basis vectors, where the first basis vector contains all of the even sample points in the sub-frame, and the second basis vector contains all of the odd sample points; hence, each basis vector is of dimension 40, and the same pseudo Gaussian codebook is used for the two basis vectors. For each basis vector, 54 candidates are considered, and 1 bit is used to specify the sign. This results in 11664 entries, which are specified by 13 bits plus the unused index space of the 2-pulse codebook, i.e. 3472 entries.

In order to reduce the complexity two candidates are pre-selected in an open-loop approach for each basis vector, by maximizing the cross correlation functions $\left|R_1^G(l)\right|$ and $\left|R_2^G(l)\right|$, respectively:

$$R_1^G(l) = \sum_{n=0}^{39} I_{exc}(2\cdot n)\cdot c_{Ga,unf}^{l}(n) \qquad l = 0,...,53 \tag{5.6.12-1}$$

$$R_2^G(l) = \sum_{n=0}^{39} I_{exc}(2\cdot n+1)\cdot c_{Ga,unf}^{l}(n) \qquad l = 0,...,53 \tag{5.6.12-2}$$

where $I_{exc}(n)$ is the ideal excitation after removal of the contribution from the adaptive codebook, and $c_{Ga,unf}^{l}(n)\ \ l = 0,...,53$ is the candidate excitation vectors from the Gaussian codebook. The same codebook is used for both basis vectors. The signs of the candidate vectors $sign\left(R_1^G(l)\right)$ and $sign\left(R_2^G(l)\right)$ are determined as the sign of the correlation functions $R_1^G(l)$ and $R_2^G(l)$ for the respective candidate vectors. The final candidate is determined among the four possible combinations of the pre-selected candidates for the two basis vectors by maximizing

$$R^G(l_1,l_2) = \frac{\left(\sum_{j=0}^{L_{SF}-1} T_g(n)\cdot c_{Ga,f}^{l_1,l_2}(n)\right)^2}{\sum_{j=0}^{L_{SF}-1}\left(c_{Ga,f}^{l_1,l_2}(n)\right)^2} \qquad l_1 = L_1(0), L_1(1) \quad l_2 = L_2(0), L_2(1) \tag{5.6.12-3}$$

where $L_1(0), L_1(1)$ and $L_2(0), L_2(1)$ specifies the candidate vectors for the two basis vectors and $_{Ga,f}^{l_1,l_2}(n)$ is the result of the convolution between $h_{imp}(n)$ and the Gaussian code vector $c_{Ga,unf}^{l_1,l_2}(n)$ reconstructed according to:

$$c_{Ga,unf}^{l_1,l_2}(n) = \begin{cases} sign\left(R_1^G(l_1)\right)\cdot c_{Ga,unf}^{l_1}\left(\dfrac{n}{2}\right) & n \text{ even} \\ sign\left(R_2^G(l_2)\right)\cdot c_{Ga,unf}^{l_2}\left(\dfrac{n-1}{2}\right) & n \text{ odd} \end{cases} \qquad n = 0,...,L_{SF}-1 \tag{5.6.12-4}.$$

5.6.13 Fixed-codebook Contribution for Rate 1

5.6.13.1 Fixed-codebook Structure for Rate 1

The fixed-codebook contribution for Rate 1 type-0 frames comprises 3 sub-codebooks, each of 5 pulses. The 3 sub-codebooks differ in the tracks for the 5 pulses, and only one of the 3 sub-codebooks is selected and used. The fixed-codebook contribution for Rate 1 type-1 frames is a single codebook of 8 pulses.

The following tables give the tracks for the three 5-pulse sub-codebooks for type-0 frames, and the tracks for the single 8-pulse codebook for type-1 frames.

	Track locations in a 40-sample subframe (type-0 frames)	# of bits
Pulse 1	1, 3, 6, 8, 11, 13, 16, 18, 21, 23, 26, 28, 31, 33, 36, 38	4
Pulse 2	4, 9, 14, 19, 24, 29, 34, 39	3
Pulse 3	1, 3, 6, 8, 11, 13, 16, 18, 21, 23, 26, 28, 31, 33, 36, 38	4
Pulse 4	4, 9, 14, 19, 24, 29, 34, 39	3
Pulse 5	0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 37	4

Table 5.6-2: Tracks for the 1st 5-Pulse Sub-Codebook for Rate 1 Type-0 Frames

Since Pulse 1 and Pulse 3 share the same track, the sign information for both tracks can be represented with a single bit (using the order of the pulse locations). Since Pulse 2 and Pulse 4 also share the same track, their sign information can also be represented with a single bit. The sign of Pulse 5 is represented by a single bit. The number of bits for the representation of the 1st 5-pulse sub-codebook for Rate 1 type-0 frames is 21.

	Track locations in a 40-sample subframe (type-0 frames)	# of bits
Pulse 1	0, 1, 2, 3, 4, 6, 8, 10	3
Pulse 2	5, 9, 13, 16, 19, 22, 25, 27	3
Pulse 3	7, 11, 15, 18, 21, 24, 28, 32	3
Pulse 4	12, 14, 17, 20, 23, 26, 30, 34	3
Pulse 5	29, 31, 33, 35, 36, 37, 38, 39	3

Table 5.6-3: Tracks for the 2nd 5-Pulse Sub-Codebook for Rate 1 Type-0 Frames

Since each pulse has an individual sign, the number of bits used to represent the 2nd 5-pulse sub-codebook for Rate 1 type-0 frames is 20.

	Track locations in a 40-sample subframe (type-0 frames)	# of bits
Pulse 1	0, 1, 2, 3, 4, 5, 6, 7	3
Pulse 2	8, 9, 10, 11, 12, 13, 14, 15	3
Pulse 3	16, 17, 18, 19, 20, 21, 22, 23	3
Pulse 4	24, 25, 26, 27, 28, 29, 30, 31	3

Pulse 5	32, 33, 34, 35, 36, 37, 38, 39	3

Table 5.6-4: Tracks for the 3rd 5-Pulse Sub-Codebook for Rate 1 Type-0 Frames

Since each pulse has an individual sign, the number of bits used to represent the 2nd 5-pulse sub-codebook for Rate 1 type-0 frames is 20.

	Track locations in a 40-sample subframe (type-1 frames)	# of bits
Pulse 1	0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30,32, 35, 37	4
Pulse 2	1, 6, 11, 16, 21, 26, 31, 36	3
Pulse 3	3, 8, 13, 18, 23, 28, 33, 38	3
Pulse 4	4, 9, 14, 19, 24, 29, 34, 39	3
Pulse 5	0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 37	4
Pulse 6	1, 6, 11, 16, 21, 26, 31, 36	3
Pulse 7	3, 8, 13, 18, 23, 28, 33, 38	3
Pulse 8	4, 9, 14, 19, 24, 29, 34, 39	3

Table 5.6-5: Tracks for the 8-Pulse Codebook for Rate 1 Type-1 Frames

Since the each pulse shares a track with another pulse, the sign information for both pulses can be represented with a single bit, and the number of bits used to represent the 8-pulse codebook for Rate 1 type-1 frames is 30 bits.

5.6.13.2 Fixed-codebook Search for Rate 1

Routine name: `FCS_cdbk_search_8500BSP`

Input:

- Target signal, $T_g(n)$
- Impulse response, $h_{imp}(n)$
- Ideal excitation, $I_{ext}(n)$
- Integer pitch lag, l_{INT}^{p}
- Pitch gain, g_a (for type-1 frames the pitch gain is quantized pitch gain for the subframe, for type-0 frames the pitch gain is the quantized pitch gain of the previous subframe)
- Subframe-modified class, C_{sub_mod}
- Noise-to-signal ratio, NSR
- Normalized pitch correlation, R_p^{wgt}

Output:

- Unfiltered fixed-codebook excitation, $c_{unf}^{fix}(n)$

For Rate 1, the impulse response is not modified, and $\hat{h}_{imp}(n)$ used in Section 5.6.11.5 is identical to $h_{imp}(n)$, as defined in Section 5.6.2.

5.6.13.2.1Pitch Enhancement Gain and Pre-Calculation of Search Parameters

The pitch gain used for pitch enhancement is obtained as:

$$\hat{g}_a = \begin{cases} 0.75 \cdot \min\{g_a, 1\} & l_{INT}^p < 20 \text{ and } g_a > 0.5 \\ g_a & \text{otherwise} \end{cases} \quad (5.6.13.2\text{-}1)$$

$\hat{g}_a$ is further modified according to the frame type according to:

$$\hat{g}_a \Leftarrow \begin{cases} \min\{\hat{g}_a, 0.8\} & \text{type-0 frame} \\ \min\{\hat{g}_a, 1.0\} & \text{type-1 frame} \cdot \end{cases} \quad (5.6.13.2\text{-}2)$$

.The search parameters are pre-calculated, as described in Section 5.6.11.5.

5.6.13.2.2Fixed-codebook Search for Type-0 Frames

The two 5-pulse sub-codebooks, described in Table 5.6-2 and Table 5.6-3, are searched, each with a single search turn, using the iterative search algorithm based on a position search for a single pulse at a time, described in Section 5.6.11.6. The sub-codebook that results in the maximum error criterion (minimum error) $\tilde{\varepsilon}$ is selected, with its combination of 5 pulses.

The criterion is then weighted, based on the sharpness of the ideal excitation, the signal-to-noise ratio, and the subframe-modified class.

The third 5-pulse sub-codebook described in Table 5.6-4 is searched, using the iterative search algorithm based on a position search for a single pulse at a time, described in Section 5.6.11.6, and its error criterion is compared to the weighted criterion from the previous step. If the criterion from the third sub-codebook is larger than the weighted criterion, the third sub-codebook is chosen; otherwise, the best of the two first sub-codebooks is chosen.

The pulse locations in the selected sub-codebook are further refined with 3 more turns, using the iterative search algorithm based a on joint position search for two pulses, described in Section 5.6.11.7. If the 1st 5-pulse sub-codebook, described in Table 5.6-2, is chosen, the redundancy in the representation of two pulses with two signs on identical tracks is removed between track1 1 and track 3 and between track 2 and track 4, by exchanging the pulse locations according to the individual pulse signs, after the final refinement turn.

The fixed-codebook contribution, $c_{unf}^{fix}(n)$, is reconstructed by placing the pulses at the selected positions with the selected signs, and the addition of pitch enhancements pulses.

5.6.13.2.3Fixed-codebook Search for Type-1 Frames

The 8-pulse codebook is first searched with 2 turns of the iterative search algorithm that is based on a joint position search for two pulses described in Section 5.6.11.7. The pulse locations are further refined with 3 more turns of the iterative search algorithm based on position search for a single pulse, described in Section 5.6.11.6. The redundancy in the representation of two pulses with two signs on identical tracks is removed between track 1 and track 5, track 2 and track 5, track 3 and track 6, and track 4 and track 8, by exchanging the pulse locations according to the individual pulse signs, after the final refinement turn.

5.6.14 Fixed-codebook Contribution for Rate 1/2

5.6.14.1 Fixed-codebook Structure for Rate 1/2

The fixed-codebook contribution for Rate 1/2 type-0 frames comprises 3 sub-codebooks, one of 2 pulses, the second of 3 pulses, and the third of random excitation. The fixed-codebook contribution for Rate 1/2 type-1 frames also comprises 3 sub-codebooks; the first of 2 pulses, the second is of 3 pulses, and the third of 5 pulses.

The following tables give the tracks for the 2-pulse sub-codebook for type-0 frames, and the tracks for the 2-pulse and the 5-pulse sub-codebook for type-1 frames.

	Track locations in an 80-sample subframe (type-0 frames)	# of bits
Pulse 1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79	6.5
Pulse 2	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79	6.5

Table 5.6-6: Tracks for the 2-Pulse Sub-Codebook for Rate 1/2 Type-0 Frames

Since $\log_2(80) = 6.322... < 6.5$, the locations of both pulses are combined and coded with $2 \cdot 6.5 = 13$ bits. Moreover, since Pulse 1 and Pulse 2 share the same track, the sign information for both tracks can be represented with a single bit. The number of bits for the representation of the 2-pulse sub-codebook for Rate 1/2 type-0 frames is 14.

	Track locations in a 53/54-sample subframe (type-1 frames)	# of bits
Pulse 1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52	5
Pulse 2	1, 3, 5, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51	5

Table 5.6-7: Tracks for the 2-Pulse Sub-Codebook for Rate 1/2 type-1 frames

The signs for each pulse are represented individually, and the total number of bits for the representation of the 2-pulse sub-codebook for Rate 1/2 type-1 frames is 12.

	Track locations in a 53/54-sample subframe (type-1 frames)	# of bits
Pulse 1	0, 15, 30, 45	2

Pulse 2	0, 5	1
Pulse 3	10, 20	1
Pulse 4	25, 35	1
Pulse 5	40, 50	1

Table 5.6-8: Tracks for the 5-pulse Sub-Codebook for Rate 1/2 Type-1 Frames

The signs for each pulse are represented individually, and the total number of bits for the representation of the 5-pulse sub-codebook for Rate 1/2 type-1 frames is 11.

The 13-bit 3-pulse sub-codebook for type-0 frames and the 11-bit 3-pulse sub-codebook for type-1 frames are constructed dynamically, where the tracks for the second and the third pulse depend on the location of the first pulse. The following tables give the fixed tracks for the first pulse for both sub-codebooks.

	Track locations in a 80-sample subframe (type-0 frames)	# of bits
Pulse 1	0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75	4

Table 5.6-9: Track for the First Pulse in the 3-Pulse Sub-Codebook for Rate 1/2 type-0 Frames

	Track locations in a 53/54-sample subframe (type-1 frames)	# of bits
Pulse 1	3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48	4

Table 5.6-10: Track for the First Pulse in the 3-Pulse Sub-Codebook for Rate 1/2 Type-1 Frames

The tracks for the 2nd pulse and the 3rd pulse are constructed in a small interval around the location of the first pulse. The length of the interval is 16 samples for type-0 frames and 8 samples for type-1 frames. The interval is positioned with the first pulse in the middle, but if the first pulse is closer to the subframe boundaries, the interval is adjusted to be fully included in the subframe. The 2nd pulse and the 3rd pulse are located on alternating even and odd locations to the left of the first pulse with reverse odd and even locations to the right of the first pulse.

Since 3 bits are used for the 2nd pulse and the 3rd pulse for type-0 frames, and since the signs of each pulse are represented individually, the total number of bits for the representation of the 3-pulse sub-codebook for Rate 1/2 type-0 frames is 13.

Since 2 bits are used for the 2nd pulse and the 3rd pulse for type-1 frames, and since the signs of each pulse are represented individually, the total number of bits for the representation of the 3-pulse sub-codebook for Rate 1/2 type-1 frames is 11.

5.6.14.2 Fixed-codebook Search for Rate 1/2

Routine name: `FCS_cdbk_search_4000BPS`

Input:

- Adaptive codebook, $C^{adp}(n)$
- Quantized and interpolated prediction coefficients, $\hat{a}_m^q(n)$
- Target signal, $T_g(n)$
- Impulse response, $h_{imp}(n)$
- Ideal excitation, $I_{ext}(n)$
- Integer pitch lag, l_{INT}^p
- Pitch gain, g_a (for type-1 frames the pitch gain is quantized pitch gain for the subframe, for type-0 frames the pitch gain is the quantized pitch gain of the previous subframe)
- Subframe-modified class, C_{sub_mod}
- Noise-to-signal ratio, NSR
- Normalized pitch correlation, R_p^{wgt}
- Previous frame rate, $Rate^{prv}$
- LPC prediction gain from quantized parameters of the first subframe, $G_{0,q}^{LPC}$

Output:

- Unfiltered fixed-codebook excitation, $c_{unf}^{fix}(n)$

The fixed-codebook search for Rate 1/2 incorporates three codebook enhancements, in addition to the pitch enhancement described in Section 5.6.11.4. The codebook enhancement is incorporated into the impulse response, by modifying $h_{imp}(n)$ (Section 5.6.2) to form $\hat{h}_{imp}(n)$, used in Section 5.6.11.

5.6.14.2.1Parameters of Formant Correction Filter and Stability Index

A weighing parameter, α^{corr}, is set according to the LPC prediction gain from quantized parameters of the first subframe, $G_{0,q}^{LPC}$ (Section 5.3.25)

$$\alpha^{corr} = \begin{cases} 0.0125 & G_{0,q}^{LPC} < 10 \\ 0.025 & 10 \le G_{0,q}^{LPC} < 20 \\ 0.05 & 20 \le G_{0,q}^{LPC} \end{cases} \quad (5.6.14.2\text{-}1)$$

The parameters for an all-pole filter are obtained (for the sub-frame indexed by m) from the quantized and interpolated prediction coefficients $\hat{a}_m^q(j)$ by:

$$a^{corr}(j) = \hat{a}_m^q(j) \cdot \left(\alpha^{corr}\right)^j \qquad j = 1,\ldots,10 \tag{5.6.14.2-2}$$

At the first subframe, a stability index for the entire frame is calculated by:

$$S_t = 1 - \frac{\left|G_{0,q}^{LPC} - G_{0,q}^{LPC,prv}\right|}{\left|G_{0,q}^{LPC} + G_{0,q}^{LPC,prv}\right|} \tag{5.6.14.2-3}$$

where $G_{0,q}^{LPC,prv}$ is $G_{0,q}^{LPC}$ of the previous frame. If the coding rate is not Rate 1/2, S_t is set to 0.

5.6.14.2.2Pulse Addition Based on Adaptive-codebook Correlations

The weighted quantized prediction coefficients $\tilde{a}_m^q(j)$ (for the sub-frame indexed by m) are obtained from the quantized and interpolated prediction coefficients $\hat{a}_m^q(j)$ according to:

$$\tilde{a}_m^q(j) = (0.85)^j \cdot \hat{a}_m^q(j) \qquad j = 1,\ldots,10 \tag{5.6.14.2-4}$$

The 170 samples of the previously quantized and reconstructed excitation are filtered by the filter:

$$\tilde{W}(z) = \frac{1}{\tilde{A}^q(z)} \tag{5.6.14.2-5}$$

where $\tilde{a}_m^q(j)$ are the coefficients of $\tilde{A}^q(z)$, to generate the weighted quantized excitation $c_w^q(n)$. Note the previously quantized and reconstruction excitation for type-1 frames is comprised only of samples from the previous frames, while the previously quantized and reconstructed excitation for type-0 is comprised of samples from previous frames, as well as samples from previous subframes of the current frame.

A set of correlation lags on the previously quantized and reconstructed excitation is constructed. If $17 < \frac{l_{INT}^p}{3} < L_{SF}$, the lags:

$$L^C = \left\lfloor \frac{j \cdot l_{INT}^p}{3} + 0.5 \right\rfloor \qquad j = 1,\ldots,5 \tag{5.6.14.2-6}$$

are included in the set of correlation lags. Similarly, if $17 < \frac{l_{INT}^p}{4} < L_{SF}$, the lags:

$$L^C = \left\lfloor \frac{j \cdot l_{INT}^p}{4} + 0.5 \right\rfloor \qquad j = 1,\ldots,5 \tag{5.6.14.2-7}$$

are included in the set of correlation lags. The set is then limited to include lags that are smaller than $l_{INT}^p - 6$ and smaller than L_{SF}. The number of correlation lags is denoted N_w^C, and the correlation lags by L_m^C, $m = 1,\ldots,N_w^C$. The normalized correlation on the weighted previously reconstructed excitation is calculated by:

$$R_w^C(m) = \frac{\sum_{n=0}^{79} c_w^q(90+n)\cdot c_w^q\left(90+n-L_m^C\right)}{\frac{1}{2}\left(\sum_{n=0}^{79} c_w^q(90+n)\cdot c_w^q(90+n)+\sum_{n=0}^{79} c_w^q\left(90+n-L_m^C\right)\cdot c_w^q\left(90+n-L_m^C\right)\right)}$$
$$m = 1,\ldots,N_w^C \qquad (5.6.14.2\text{-}8)$$

The normalized correlations are further limited to the range $0 \le R_w^C(m) \le 1$.

If the current frame is type-1 and the previous frame was also Rate 1/2 type-1 frame, the correlations are normalized by:

$$R_w^C(m) \Leftarrow 0.5\cdot R_w^C(m)\cdot S_t\cdot\min\left(\hat{g}_p,1\right) \qquad m=1,\ldots,N_w^C \qquad (5.6.14.2\text{-}9)$$

Otherwise, the correlation is normalized by

$$R_w^C(m) \Leftarrow 0.25\cdot R_w^C(m)\cdot S_t\cdot\min\left(\hat{g}_p,0.75\right) \qquad m=1,\ldots,N_w^C \qquad (5.6.14.2\text{-}10)$$

5.6.14.2.3Generating the Modified Impulse Response

The modified impulse response $\hat{h}_{imp}(n)$ is generated from the impulse response $h_{imp}(n)$. The correlation enhancement is performed by

$$h'_{imp}(n) = h_{imp} + \sum_{m=1}^{N_w^C} R_w^C(m)\cdot h_{imp}\left(n-L_m^C\right) \qquad n=0,\ldots,L_{SF}-1 \qquad (5.6.14.2\text{-}11)$$

where $h_{imp}(n)$ is zero for $i<0$.

The formant correction is performed by the filtering of $h'_{imp}(n)$ by the zero-state all-pole filter:

$$H^{corr}(z) = \frac{1}{A^{corr}(z)} \qquad (5.6.14.2\text{-}12)$$

where the coefficients of $A^{corr}(z)$ were calculated in Section 5.6.14.2.1.

The last modification of the impulse response is obtained by the convolution of output of the formant correction filter with a constant 20-sample dispersion vector, to generate $\hat{h}_{imp}(n)$.

5.6.14.2.4Pitch Enhancement Gain and Pre-Calculation of Search Parameters

The pitch gain used for pitch enhancement is obtained as:

$$\hat{g}_a = \begin{cases} \min\{g_a,1\} & \text{type-1 frame} \\ 0.75 & \text{type-0 frame} \end{cases} \qquad (5.6.14.2\text{-}13)$$

The search parameters are pre-calculated, as described in Section 5.6.11.5.

1

2 5.6.14.2.5Fixed-codebook Search for Type-0 Frames

3 The 2-pulse sub-codebook, described in Table 5.6-6, is searched with a first search turn
4 with the iterative search algorithm based on joint position search for two pulses (Section
5 5.6.11.7), and a second turn with iterative search algorithm based on position search for a
6 single pulse (Section 5.6.11.6). The redundancy in the representation of two pulses with
7 two signs on identical tracks is removed between track 1 and track 2, by exchanging the
8 pulse locations according to the individual pulse signs.

9 The criterion $\widetilde{\varepsilon}$ from the 2-pulse sub-codebook is weighted, as a function of the pitch lag
10 and the normalized pitch correlation.

11 Before the 3-pulse sub-codebook is searched, the modified pitch gain $\hat{g}_a$ is set to 0.25, and
12 search parameters are pre-calculated again, as described in Section 5.6.11.5. The 3-pulse
13 sub-codebook is then searched, using a single turn of the iterative search algorithm based
14 on joint position search for two pulses.

15 The maximum between the weighted criterion from the 2-pulse sub-codebook and the
16 criterion from the 3-pulse sub-codebook selects which sub-codebook is chosen at this
17 stage, with its best pulse locations and signs.

18 The chosen criterion is weighted again, using the subframe-modified class, the sharpness
19 of the ideal excitation, the normalized pitch correlation, the information of the pre-selected
20 sub-codebook (3-pulse or 2-pulse), the noise-to-signal ratio, and the pitch lag.

21 The Gaussian sub-codebook is searched last (Section 5.6.12), using the original impulse
22 response, $h_{imp}(n)$ (Section 5.6.2).

23 The weighted criterion from the previous search is compared to the criterion from the
24 Gaussian sub-codebook, and the best sub-codebook is selected.

25 The fixed-codebook contribution, $c_{unf}^{fix}(n)$, is reconstructed. If the 2-pulse sub-codebook or
26 the 3-pulse sub-codebook were selected, the codebook enhancement contributions are
27 added to the reconstructed fixed-codebook contribution.

28

29 5.6.14.2.6Fixed-codebook Search for Type-1 Frames

30 The 2-pulse sub-codebooks, described in Table 5.6-7, is searched with a first search turn
31 with the iterative search algorithm based on a joint position search for two pulses (Section
32 5.6.11.7), and a second turn with iterative search algorithm based on a position search for
33 a single pulse (Section 5.6.11.6).

34 The 3-pulse sub-codebook is searched next, using a single turn of the iterative search
35 algorithm based on a joint position search for two pulses (Section 5.6.11.7).

36 The maximum between the criterion from the 2-pulse sub-codebook and the criterion from
37 the 3-pulse sub-codebook selects which sub-codebook is chosen at this stage, with its best
38 pulse locations and signs.

39 The chosen criterion is then weighted, using the subframe-modified class, the sharpness of
40 the ideal excitation, and the noise-to-signal ratio.

41 The 5-pulse sub-codebook is searched last using the joint position search described in
42 Section 5.6.11.8.

43 The weighted criterion from the previous search is compared to the criterion from the 5-
44 pulse sub-codebook, and the best sub-codebook is selected.

The fixed-codebook contribution, $c_{unf}^{fix}(n)$, is reconstructed, and the codebook enhancement contributions are added to the reconstructed fixed-codebook contribution.

5.6.15 Filtering of Fixed-Codebook Contribution

The fixed-codebook vector, $c_{unf}^{fix}(n)$, is convolved with the impulse response $h_{imp}(n)$ to generate the filtered fixed-codebook vector, $c_f^{fix}(n)$.

5.6.16 Gain Re-optimization and Normalization for Rate 1/2 and Rate 1

Routine name: `GEQ_gain_reopt_2`

Input:

- Filtered adaptive-codebook vector, $c_f^{adp}(n)$
- Filtered fixed-codebook vector, $c_f^{fix}(n)$
- Target vector, $T_{gs}(n)$

Output:

- Quantized fixed-codebook gain, g_c

The purpose of this module is to modify the target signal prior to the gain quantization. First a re-optimization process takes places as follows. If the normalized pitch correlation is below 0.75 and the frame is of type 0, the adaptive and fixed-codebook gains are jointly optimized by minimizing the WMSE between the original and the reconstructed speech according to

$$\{g_a, g_c\} = \arg\min\left\{\sum_{n=0}^{L_{SF}-1}\left(T_{gs}(n) - \left(\left(g_a c_f^{adp}(n)\right) + \left(g_c c_f^{fix}(n)\right)\right)\right)^2\right\} \quad (5.6.16\text{-}1)$$

This results in the following estimates of the two gains:

$$g_a = \frac{R_{a,t}\cdot R_{c,c} - R_{a,c}\cdot R_{c,t}}{R_{a,a}\cdot R_{c,c} - R_{a,c}\cdot R_{a,c}} \quad (5.6.16\text{-}2)$$

and

$$g_c = \frac{R_{c,t} - g_a R_{a,c}}{R_{c,c}}, \quad (5.6.16\text{-}3)$$

where

$$R_{a,t} = \sum_{n=0}^{L_{SF}-1} c_f^{fix}(n)\cdot T_{gs}(n), \quad (5.6.16\text{-}4)$$

$$R_{c,t} = \sum_{n=0}^{L_{SF}-1} c_f^{fix}(n)\cdot T_{gs}(n), \quad (5.6.16\text{-}5)$$

$$R_{a,a} = \sum_{n=0}^{L_{SF}-1} \left(c_f^{adp}(n)\right)^2, \quad (5.6.16\text{-}6)$$

$$R_{c,c} = \sum_{n=0}^{L_{SF}-1} \left(c_f^{fix}(n)\right)^2, \quad (5.6.16\text{-}7)$$

$$R_{a,c} = \sum_{n=0}^{L_{SF}-1} c_f^{adp}(n)\cdot c_f^{fix}(n). \quad (5.6.16\text{-}8)$$

where L_{SF} is the subframe length.

If the normalized pitch correlation is above 0.75 or the frame is of type 1, the adaptive and fixed-codebook gains are disjointedly optimized. The adaptive-codebook gain was already calculated according to Equation (5.6.6.3-1). Subsequently, the fixed-codebook gain is estimated according to

$$g_c = \arg\min\left\{ \sum_{n=0}^{L_{SF}-1} \left(T_g(n) - \left(g_c \cdot c_f^{fix}(n) \right) \right)^2 \right\}, \quad \text{(5.6.16-9)}$$

where

$$T_g(n) = T_{gs}(n) - g_a \cdot c_f^{adp}(n). \quad \text{(5.6.16-10)}$$

The gain is given by

$$g_c = \frac{\sum_{n=0}^{L_{SF}-1} T_g(n) \cdot \left(c_f^{fix}(n) \right)}{\sum_{n=0}^{L_{SF}-1} \left(c_f^{fix}(n) \right)^2} \quad \text{(5.6.16-11)}$$

Second, a gain normalization process takes place as follows. First, we describe the gain normalization process for type-0 frames. The target energy of both the quantized excitation and reconstructed speech is estimated according to the smoothing factor β_E (Section 5.6.10).

Routine name: `PRC_GainsNorm_Gc_Gp`

Input:

- Filtered adaptive-codebook vector, $c_f^{adp}(n)$
- Filtered fixed-codebook vector, $c_f^{fix}(n)$
- Unfiltered adaptive-codebook vector, $c_{unf}^{adp}(n)$
- Unfiltered fixed-codebook vector, $c_{unf}^{fix}(n)$
- Target vector, $T_{gs}(n)$
- Noise-to-signal ratio, NSR
- Gain smoothing factor, β_E
- Excitation mode factor, C_{ext_mod}
- Weighted residual signal, $r_w(n)$
- Residual of modified signal, $r_{\mathrm{mod}}(n)$

Output:

- Normalized gains, g_a and g_c

5.6.16.1 Normalization for $NSR > 0.125$

A scaling factor g_s for the codebook gains is estimated in order to match the energy targets. It is based on the smoothed open and closed loop energy targets.

It should be noted that the smoothing is variable and may be zero. The open loop scaling factor is given by

$$g_s = 0.75 \cdot \sqrt{\frac{\max(\widetilde{E}_e - 5,0)}{\sum_{n=0}^{L_{SF}-1} \left(g_a \cdot c_{unf}^{adp}(n) + g_c \cdot c_{unf}^{fix}(n)\right)^2}}, \quad (5.6.16.1\text{-}1)$$

where it is bounded by $g_s < 1.2 / g_a$ and the target energy of the quantized excitation (also referred as the open loop target energy) is given by

$$\widetilde{E}_e = \beta_E \cdot \widetilde{E}_e + (1-\beta_E) \cdot \sum_{n=0}^{L_{SF}-1} r_w(n)^2 . \quad (5.6.16.1\text{-}2)$$

The target energy of the reconstructed speech (also referred as the closed loop target energy) is given by

$$\widetilde{E}_s = \beta_E \cdot \widetilde{E}_s + (1-\beta_E) \cdot \sum_{n=0}^{L_{SF}-1} T_{gs}(n)^2 . \quad (5.6.16.1\text{-}3)$$

5.6.16.2 Normalization for $NSR \le 0.125$

The determination of g_s is derived as follows. First, an open-loop scaling factor and a closed-loop scaling factor are computed as:

$$g_{ol} = \sqrt{\frac{\max(\widetilde{E}_e - 5,0)}{\sum_{n=0}^{L_{SF}-1} \left(g_a \cdot c_{unf}^{adp}(n) + g_c \cdot c_{unf}^{fix}(n)\right)^2}}, \quad (5.6.16.2\text{-}1)$$

and

$$g_{cl} = \sqrt{\frac{\max(\widetilde{E}_s - 5,0)}{\sum_{n=0}^{L_{SF}-1} \left(g_a \cdot c_f^{adp}(n) + g_c \cdot c_f^{fix}(n)\right)^2}}, \quad (5.6.16.2\text{-}2)$$

Here,

$$\widetilde{E}_e = \beta_E \cdot \widetilde{E}_e + (1-\beta_E) \cdot \sum_{n=0}^{L_{SF}-1} r_{\mathrm{mod}}(n)^2 \quad (5.6.16.2\text{-}3)$$

and

$$\widetilde{E}_s = \beta_E \cdot \widetilde{E}_s + (1-\beta_E) \cdot \sum_{n=0}^{L_{SF}-1} T_{gs}(n)^2 \quad . \tag{5.6.16.2-4}$$

An attenuation of the LPC filter is also calculated as:

$$g^{LPC} = \sqrt{\frac{\sum_{n=0}^{L_{SF}-1} r_{\mathrm{mod}}(n)^2}{\sum_{n=0}^{L_{SF}-1} T_{gs}(n)^2}}, \tag{5.6.16.2-5}$$

where g^{LPC} is bounded by $g^{LPC} < 0.8$.

Then g_s is calculated in the following steps if the flag C_{ext_mod} is not 0 as:

$$g_s = \min\left\{\left(\frac{g^{LPC}}{0.8}\right) \cdot g_{ol} + \left(1 - \frac{g^{LPC}}{0.8}\right) \cdot g_{cl}, g_{cl}\right\}, \tag{5.6.16.2-6}$$

Then it is refined in two steps:

$$g_s = \min\{0.7 \cdot g_s, \frac{1.2}{g_a}\} \tag{5.6.16.2-7}$$

and $g_s = \min\{1 + g^{LPC}, g_s\}$.

Otherwise, $g_s = 0.75 \cdot \min\{g_{ol}, g_{cl}\}$. It is further refined in this case as $g_s = \min\{g_s, \frac{1.2}{g_a}\}$

In any case g_s is always lower-bounded by 1.0.

Based on the final scaling factor the unquantized gains are modified according to

$$g'_a = g_s \cdot g_a, \tag{5.6.16.2-8}$$

$$g'_c = g_s \cdot g_c, \tag{5.6.16.2-9}$$

and the target is corrected accordingly to

$$T_{gs}(n) = g'_a \cdot c_f^{adp}(n) + g'_c \cdot c_f^{fix}(n) \tag{5.6.16.2-10}$$

5.6.16.3 Fixed-Codebook Gain Normalization for Type-1 Frames and $NSR > 0.25$

Routine name: `PRC_GainNorm_Gc`

Input:

- Filtered adaptive-codebook vector, $c_f^{adp}(n)$
- Filtered fixed-codebook vector, $c_f^{fix}(n)$

- Unfiltered adaptive-codebook vector, $c_{unf}^{adp}(n)$
- Unfiltered fixed-codebook vector, $c_{unf}^{fix}(n)$
- Target vector, $T_g(n)$
- Residual of modified signal, $r_{\mathrm{mod}}(n)$

Output:

- Normalized gain, g_c

The different scale factors are computed as:

$$g_{ol} = \sqrt{\frac{\sum_{n=0}^{L_{SF}-1} r_{\mathrm{mod}}^2(n)}{\sum_{n=0}^{L_{SF}-1} \left(g_c \cdot c_{unf}^{fix}(n)\right)^2}} \qquad (5.6.16.3\text{-}1)$$

and

$$g_{cl} = \sqrt{\frac{\sum_{n=0}^{L_{SF}-1} T_g^2(n)}{\sum_{n=0}^{L_{SF}-1} \left(g_c \cdot c_f^{fix}(n)\right)^2}} \qquad (5.6.16.3\text{-}2)$$

g^{LPC} is computed:

$$g^{LPC} = \sqrt{\frac{\sum_{n=0}^{L_{SF}-1} r_{\mathrm{mod}}(n)^2}{\sum_{n=0}^{L_{SF}-1} T_g(n)^2}} \qquad (5.6.16.3\text{-}3)$$

The gain-scaling factor g_s is computed as:

$$g_s = 0.75 \cdot g^{LPC} \cdot g_{ol} + (1 - 0.75 \cdot g^{LPC}) \cdot g_{cl} \qquad (5.6.16.3\text{-}4)$$

It is refined as:

$$g_s = \min(1 + 0.75 \cdot g^{LPC}, \max(1.0, 0.5 \cdot g_s)) \qquad (5.6.16.3\text{-}5)$$

Based on the final scaling factor, the unquantized gains are modified according to:

$$g_a' = g_a, \qquad (5.6.16.3\text{-}6)$$

$$g_c' = g_s \cdot g_c, \qquad (5.6.16.3\text{-}7)$$

and the target is corrected accordingly to

$$T_g(n) = g'_a \cdot c_f^{adp}(n) + g'_c \cdot c_f^{fix}(n) \quad \text{(5.6.16.3-8)}$$

5.6.17 Adaptive-Codebook and Fixed-Codebook Gain Quantization for Type-0 Frames

Routine name: `GEQ_gainVQMA_2`

Input:

- Filtered adaptive-codebook vector, $c_f^{adp}(n)$
- Filtered fixed-codebook vector, $c_f^{fix}(n)$
- Unfiltered adaptive-codebook vector, $c_{unf}^{adp}(n)$
- Unfiltered fixed-codebook vector, $c_{unf}^{fix}(n)$
- Target vector, $T_g(n)$

Output:

- Quantized gains, g_a and g_c

For type-0, the adaptive and fixed-codebook gains are jointly vector quantized with 7 bits per subframe. The 2-dimensional codebook is searched exhaustively for the entry that minimizes the MSE between the target and the reconstructed speech signal, i.e. minimizing

$$E = \sum_{n=0}^{L_{SF}-1} \left(T_g(n) - \left(g_a c_f^{adp}(n) + g_c c_f^{fix}(n)\right)\right)^2 \qquad (5.6.17\text{-}1)$$

where L_{SF} is 40 and 80 for Rate 1 and Rate 1/2 respectively. The quantized adaptive and fixed-codebook gains are derived from the 7 bits codebook. The entries of the codebook contain the adaptive-codebook gain and the correction factor for the predicted fixed-codebook gain.

The prediction of the fixed-codebook gain is based on a 4th and 2nd order MA prediction of the fixed-codebook energy for Rate 1 and Rate 1/2 respectively.

The relation between the correction factor γ^m and the quantized fixed-codebook gain is given by

$$g_c = \gamma^m \cdot \tilde{g}_c \qquad (5.6.17\text{-}2)$$

where g_c is the quantized fixed-codebook gain and $\tilde{g}_c$ is the predicted fixed-codebook gain. The predicted fixed-codebook gain is given by

$$\tilde{g}_c = 10^{\frac{1}{20}\left(\tilde{E}_k - E_c + \overline{E}\right)}, \qquad (5.6.17\text{-}3)$$

where the $\overline{E} = 30\text{dB}$ is the mean energy,

$$E_c = 10\log_{10}\left(\frac{1}{L_{SF}}\sum_{n=0}^{L_{SF}-1} c_{unf}^{fix}(n)^2\right), \quad (5.6.17\text{-}4)$$

and

$$\widetilde{E} = \sum_{i=1}^{p} b_i \cdot \left(20\log_{10}\left(\gamma^{m-i}\right)\right), \quad (5.6.17\text{-}5)$$

where p is 4 and 2 for Rate 1 and Rate 1/2 respectively. The prediction coefficients of the MA prediction are $\{b_1, b_2, b_3, b_4\} = \{0.7, 0.6, 0.4, 0.2\}$ and $\{b_1, b_2\} = \{0.6, 0.3\}$ for Rate 1 and Rate 1/2 respectively.

5.6.18 Fixed-Codebook Gain Quantization for Type-1 Frames

Routine name: `GEQ_gainVQMA_3,GEQ_gainVQMA_4`

Input:

- Filtered adaptive-codebook vector, $c_f^{adp}(n)$
- Filtered fixed-codebook vector, $c_f^{fix}(n)$
- Unfiltered adaptive-codebook vector, $c_{unf}^{adp}(n)$
- Unfiltered fixed-codebook vector, $c_{unf}^{fix}(n)$
- Target vector, T_g
- Quantized gains, $g_a(m),\ m = 1,2,3,[4]$

Output:

- Quantized gains, $g_c(m),\ m = 1,2,3,[4]$

This section describes the fixed-codebook gain quantization for Rate 1 and Rate 1/2. For type-1 frames the adaptive-codebook gain quantization is described in Section 5.3.18. The fixed-codebook gain vector $g_c(m)$ for type-1 frame is quantized to 10 and 8 bits for Rate 1 and Rate 1/2, respectively. The quantization is predictive delayed quantization; i.e. it takes place after all of subframes have been processed with unquantized fixed-codebook gains. The excitation signals, the target signals, and the quantized adaptive- codebook gains, are buffered during the subframe processing and used to perform delayed joint quantization of the fixed-codebook gain vector.

For Rate 1, the 4-dimensional codebook is searched in order to minimize

$$E = \sum_{m=1}^{4}\sum_{n=0}^{40}\left(\left(T_g^m(n) - g_a(m)c_f^{adp,m}(n)\right) - g_c(m)c_f^{fix,m}(n)\right)^2, \quad (5.6.18\text{-}1)$$

where the quantized adaptive-codebook gains $\{g_a(m)\}$ originate from the original frame based processing, and $\{T_g^m(n)\}$, $\{c_f^{adp,m}(n)\}$, and $\{c_f^{fix,m}(n)\}$ are buffered during the subframe processing. Note that the superscript m is the m^{th} subframe.

Similarly for Rate 1/2, we have

$$E = \alpha_1 \sum_{n=0}^{52} \left(\left(T_g^1(n) - g_a(1) \cdot c_f^{adp,1}(n) \right) - g_c(1) \cdot c_f^{fix,1}(n) \right)^2 + \alpha_2 \sum_{n=0}^{52} \left(\left(T_g^2(n) - g_a(2) \cdot c_f^{adp,2}(n) \right) - g_c(2) \cdot c_f^{fix,2}(n) \right)^2 + \alpha_3 \sum_{n=0}^{53} \left(\left(T_g^3(n) - g_a(3) \cdot c_f^{adp,3}(n) \right) - g_c(3) \cdot c_f^{fix,3}(n) \right)^2 , \quad (5.6.18\text{-}2)$$

where the quantized adaptive-codebook gains $\{g_a(m)\}$ originate from the original frame based processing, and $\{T_g^m(n)\}$, $\{c_f^{adp,m}(n)\}$, and $\{c_f^{fix,m}(n)\}$ are buffered during the subframe processing and

$$\alpha_m = \frac{\sum_{n=0}^{L_{sf}(m)} \left(T_g^m(n) \right)^2}{\sum_{n=0}^{L_{frm}} T_g(n)^2} . \quad (5.6.18\text{-}3)$$

For Rate 1, the fixed-codebook gains $\{g_c(1), g_c(2), g_c(3), g_c(4)\}$ are derived from a 2-stage 10 bit codebook (7 and 3 bits respectively), where the entries of the codebook contain a 4-dimensional correction factor for the predicted fixed-codebook gains. For Rate 1/2, the fixed-codebook gains $\{g_c(1), g_c(2), g_c(3)\}$ are derived from an 8-bit codebook where the entries of the codebook contain a 3-dimensional correction factor for the predicted fixed-codebook gains. The prediction of the fixed-codebook gains is based on an MA prediction of the fixed-codebook energy.

The relation between the correction factors γ^m and the quantized fixed-codebook gains is given by

$$g_c^m = \gamma^m \cdot \tilde{g}_c^m \quad (5.6.18\text{-}4)$$

where g_c^m is the quantized fixed-codebook gain and $\tilde{g}_c^m$ is the predicted fixed-codebook gain of the m^{th} subframe of current frame.

The predicted fixed-codebook gains are based on an MA prediction of the fixed-codebook energy, and it is given by

$$\tilde{g}_c^m = 10^{\frac{1}{20}\left(\tilde{E}^m - E_c^m + \overline{E} \right)} \quad (5.6.18\text{-}5)$$

where the $\overline{E} = 34\text{dB}$ is the mean energy,

$$E_c^m = 10 \log_{10} \left(\frac{1}{L_{SF}} \sum_{n=0}^{L_{SF}-1} \left(c_{unf}^{fix,j}(n) \right)^2 \right), \quad (5.6.18\text{-}6)$$

and

$$\tilde{E}^m = \sum_{i=1}^{N} b_{m,i} \cdot \left(20 \log_{10} \left(\gamma_{prev}^{N+1-i} \right) \right), \quad (5.6.18\text{-}7)$$

where N is 4 and 3 for Rate 1 and Rate 1/2 respectively. The predictor coefficients for Rate 1 are

$$\{b_{\mathrm{m},1}, b_{m,2}, b_{m,3}, b_{m,4}\} = \begin{cases} \{0.7, 0.6, 0.4, 0.2\} & m = 1 \ (1^{\mathrm{st}} \text{ subframe}) \\ \{0.4, 0.2, 0.1, 0.05\} & m = 2 \ (2^{\mathrm{nd}} \text{ subframe}) \\ \{0.3, 0.2, 0.075, 0.025\} & m = 3 \ (3^{\mathrm{rd}} \text{ subframe}) \\ \{0.2, 0.075, 0.025, 0.0\} & m = 4 \ (4^{\mathrm{th}} \text{ subframe}) \end{cases}, \quad (5.6.18\text{-}8)$$

and the predictor coefficients for Rate 1/2 are

$$\{b_{m,1}, b_{m,2}, b_{m,3}\} = \begin{cases} \{0.6, 0.30, 0.100\} & m = 1 \ (1^{\mathrm{st}} \text{ subframe}) \\ \{0.4, 0.25, 0.100\} & m = 2 \ (2^{\mathrm{nd}} \text{ subframe}) \\ \{0.3, 0.15, 0.075\} & m = 3 \ (3^{\mathrm{rd}} \text{ subframe}) \end{cases}. \quad (5.6.18\text{-}9)$$

Consequently, the prediction of the energies of the 4/3 subframes is based on the same past memory.

5.6.19 Random Number Generator

Routine name: `GCB_quickran`

Input:

- Seed, $seed$

Output:

- Uniformly distributed variable, $rand(seed)$
- Updated Seed, $seed$

$$r(n) = \sum_{i=1}^{12} rand(seed)$$

where $rand()$ generates a zero mean, uniformly distributed variable. For each sample, $seed$ is updated as follows:

$$seed = extract\_l(seed \cdot 25173 + 13849), \quad (5.6.19\text{-}1)$$

where $extract\_l(.)$ extracts the 16 least significant bits of a 32 bit argument.

6 SMV DECODER

6.1 Rate and Frame Type Retrieval and Frame Error Detection

6.1.1 Handling NULL Rate 1/8 Frame

If the decoder receives an all 1's (null) Rate 1/8 packet, frame erasure processing is performed for the current frame. (The encoder bit-packing routine ensures that, for 1/8 rate, an all 1's bit stream is replaced by a valid bit stream that includes a single 0.)

6.1.2 Handling All Zero Frames

If the decoder receives an all 0's packet (for all rates), the current frame is processed as frame erasure. (The encoder bit-packing routine ensures that, for all rates, an all 0's bit stream is replaced by a valid bit stream that includes a single 1.)

6.1.3 Handling 'Rate 1 with Bit Errors' Frame

If the decoder receives 'Rate 1 with Bit Errors' frame, it will process the frame as an erasure (Insufficient Frame Quality) as in Section 6.8.

6.1.4 Channel Rate-Determination Algorithm Error Detection

The multiplex sub-layer occasionally provides bad rates to the SMV decoder due to errors in the Channel rate-determination algorithm (CRDA).

Due to constraints imposed on the SMV encoder the following frame rate, and type transitions between successive frames are illegal:

a) Type-1 Rate 1 frame following a Rate 1/8 frame
b) Type-1 Rate 1/2 frame following a Rate 1/8 frame
c) Type-1 Rate 1 frame following a Rate 1/4 frame
d) Type-1 Rate 1/2 frame following a Rate 1/4 frame
e) Rate 1/8 frame following a Type-1 Rate 1 frame
f) Rate 1/8 frame following a Type-1 Rate 1/2 frame
g) Rate 1/4 frame following a Type-1 Rate 1 frame
h) Rate 1/4 frame following a Type-1 Rate 1/2 frame

Once the frame rate and the type of the current frame have been determined from the packet, a check for the above transitions shall be made. If any of the above transitions is detected, the current frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8. In addition, if the transitions of a, b, c, or d are detected, the following decoder memories are reset to their initialization values:

i. Excitation memory
ii. Frame Erasure Concealment excitation memory
iii. LPC synthesis memory
iv. Postfilter synthesis memory
v. Past fixed codebook energy
vi. Previous frame LSFs

Further, if the transitions of a, b, c, or d are detected, the fixed codebook gain moving average vector quantization memories are extended by their old values.

If the Rate Sanity Flag in the Rate 1 SMV packet is 1, the frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8.

If the Rate Sanity Flag in the Rate 1/4 SMV packet is 1, the frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8.

If the two shaping filter bits (SF_ IDX[0] and SF_IDX[1]) in the Rate 1/4 SMV packet are 1, the frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8.

If the current frame is a type 1, Rate 1 frame following a type 1 (Rate 1 or Rate 1/2) frame, and the pitch lag of the current frame is different from the pitch lag of the previous frame by more than 15, the frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8.

If the current frame is a Type-1, Rate 1/2 frame following a Type-1 (Rate 1 or Rate 1/2) frame, and the pitch lag of the current frame is greater than 40, and is different from the pitch lag of the previous frame by more than 15, the frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8.

If the current frame is a Rate 1/8 frame whose gain factor g_r is such that

$$g_r > 5g_{mem} \tag{6.1.4-1}$$

the frame shall be declared a bad frame. Frame erasure processing should be performed as described in Section 6.8, where g_{mem} is the estimate of the average frame gain for Rate 1/8, and is updated after the above test in each Rate 1/8 frame as

$$g_{mem} = \max\left(round\left(\max\left(g_r,100\right)\right), round\left(0.05\cdot round\left(\max\left(g_r,100\right)\right)+0.95\cdot g_{mem}\right)\right) \tag{6.1.4-2}$$

After the fixed codebook excitation, $c_{unf}^{fix}(n)$, and the fixed-codebook gain, g_c, have been determined as described in Section 6.7.4 and 6.7.6 for each subframe of fifty type-0 frames (Rate 1 or Rate 1/2) that immediately follow a type-1 frame, the following test for unusually high fixed-codebook gain shall be performed:

$$E_{nrm}^{fix} = \frac{g_c E_{unf}^{fix}}{MAX\left(0.01, \sqrt{10^{\left(\bar{E}_{MAX}/10-8\right)}}\right)}, \tag{6.1.4-3}$$

where E_{unf}^{fix} is the energy of the unfiltered fixed codebook excitation, and $\bar{E}_{MAX}$ is the maximum LPC synthesized speech frame energy in decibels in the last fifty type-1 frames.

For Rate 1, if $\left\{-2.34\left(G^{LPC}\right)^3+170.16\left(G^{LPC}\right)^2-4143.1\left(G^{LPC}\right)+36754\right\}<E_{nrm}^{fix}$ and $G^{LPC}<35$, or $E_{nrm}^{fix}>21000$, the subframe gain g_c should be reset to zero, and the FCB gain moving average VQ memory should be reset to the initial values to avoid high energy speech artifacts.

For Rate 1/2, if $\left\{-1.84\left(G^{LPC}\right)^3+162.8\left(G^{LPC}\right)^2-4796\left(G^{LPC}\right)+47520\right\}<E_{nrm}^{fix}$ and $G^{LPC}<35$, or $E_{nrm}^{fix}>18000$ the subframe gain g_c should be reset to zero, and the FCB gain moving average VQ memory should be reset to the initial values to avoid high energy speech artifacts.

For Rate 1/2 type-0 frames, if an index in the unused index space for the 2-pulse codebook (with the additional Gaussian codebook and the additional TTY/TTD/DATA/DTMF flag) is detected, the frame is erased.

6.2 General Structure of the Decoder

The decoder comprises 3 elements:

1. Excitation generation
2. LPC synthesis filter
3. Post filtering

The operation of each element is different for each rate. In addition, a frame erasure concealment algorithm is used to generate the decoder output during single or multiple frame erasure.

Figure 6.2-1 describes the general structure of the decoder, without the frame concealment module. The specific details of decoder operations are described in the next sections.

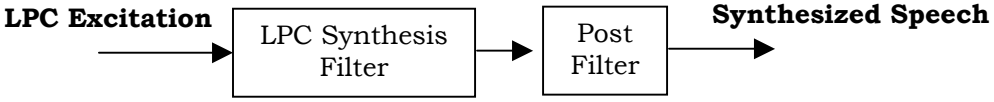

Figure 6.2-1: General Decoder Structure

6.3 Decoding of the LSFs

The indices into the LSFs multi-stage tables shall be retrieved from the bitstream, according to the transmitted rate. The quantized prediction error, $\varepsilon^q(i)$, shall be constructed by the sum of the entries in the quantization tables, as specified in Section 5.3.22.4. The reconstructed LSFs shall be obtained from the combination of the MA prediction, the decoded prediction error obtained from the decoded indices, and the mean vector of the LSFs by:

$$\underline{lsf}^q(i) = \varepsilon^q(i) + \sum_{p=1}^{O_{Rate}^{MA}} b_{j,p}^{Rate}(i) \cdot \varepsilon_p^q(i) \; + M_{LSF}(i) \qquad i = 1\ldots,10 \tag{6.1.4-1}$$

where j is the decoded or default predictor according to the rate, and O_{Rate}^{MA}, the order of the MA prediction, is 2 for Rate 1 and 4 for all other rates

6.4 Interpolation of Decoded LSFs

The LSFs shall be interpolated the same way specified for the encoder in Section 5.3.23. For Rate 1 type-0 frames, the interpolation path shall be decoded from the bitstream, and the decoded interpolation path shall be used.

6.5 Excitation Decoding for Rate 1/8

The decoding of Rate 1/8 is depicted in Figure 6.5-1. The gain index shall be decoded from the bitstream input to the decoder, and the gain correction factor shall be obtained from the quantization table. The gain value for the frame shall be calculated from the correction factor and the predicted energy, as described in Section 5.4.2. The excitation shall be generated by a random vector generator, described in Section 5.6.19. The excitation shall be multiplied by the gain, to generate the excitation to the LPC synthesis filter.

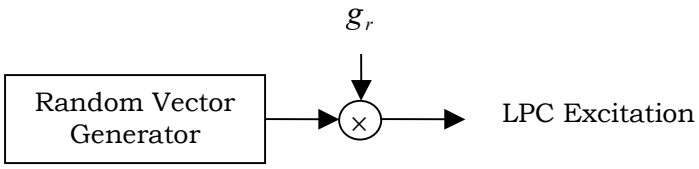

Figure 6.5-1: Excitation Decoding for Rate 1/8

6.6 Decoding of Rate 1/4

The decoding of Rate 1/4 is depicted in.

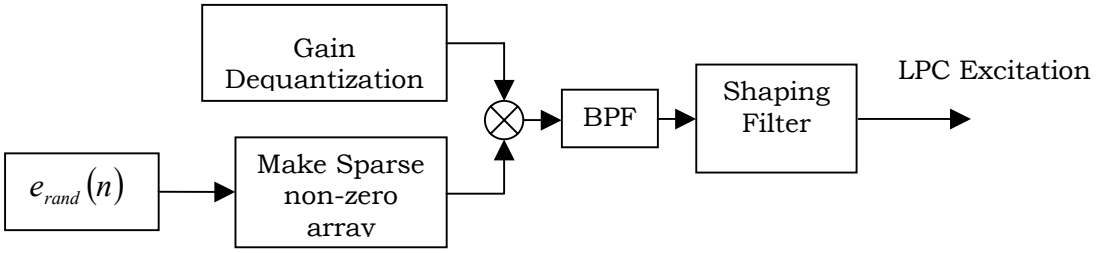

Figure 6.6-1: Excitation Decoding for Rate 1/4

The gain indices $i\hat{G}'$, and $i\hat{G}''[j], 0 \le j < 2$, and the shaping filter index F_{idx} are decoded from the bitstream input to the decoder. The gains are dequantized using Equations (5.5.1-5), (5.5.1-8) and (5.5.1-9). A pseudo-random vector, $e_{rand}(n)$, is generated as described in Section 5.5.2. As described in Section 5.5.3, this pseudo-random vector is used to generate a gain scaled sparse signal, $r_1(n)$, which is then shaped by the bandpass filter $BP(z)$, and the shaping filter $SF(z)$ to yield the Rate 1/4 frame LPC excitation. As shown in Equation (5.5.4-9), the index of the filter, F_{idx} is used to establish $SF(z)$. The Rate 1/8 frame excitation is then filtered by the LP synthesis filter, and the post-processing filter to generate speech.

6.7 Decoding of Rate 1 and Rate 1/2

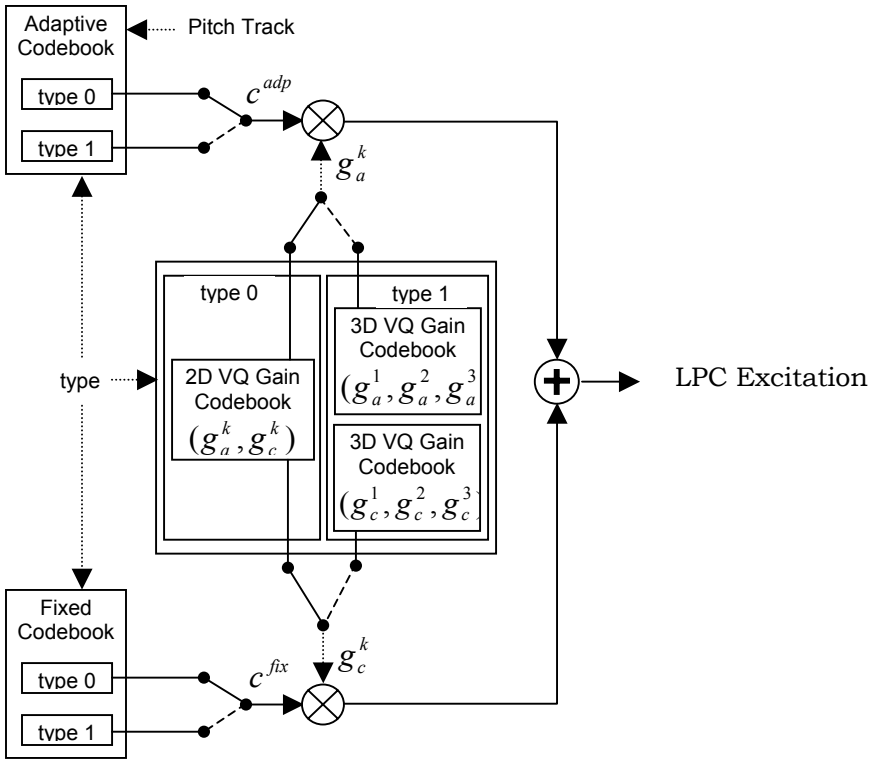

Figure 6.7-1: The SMV Decoder for Rate 1 and Rate 1/2

Figure 6.7-1 describes the excitation generation for Rate 1 and Rate 1/2. The excitation is constructed from the fixed-codebook contribution that is multiplied by the fixed-codebook gain, which is added to the adaptive-codebook contribution that multiplied by the adaptive codebook gain.

For each subframe, the fixed-codebook contribution is constructed from the pulse locations and signs, or from the Gaussian codebook indices. The adaptive-codebook contribution for type-0 frames is constructed from the pitch lag values for each subframe. For type-1 frames the adaptive-codebook contribution is obtained from the interpolated pitch contour. The gains for type-0 frames are obtained from the joint table for the fixed-codebook gain and the adaptive-codebook gain. For type-1 frames, the adaptive-codebook gains for the whole frame are obtained from the table for the adaptive-codebook gains, and the fixed-codebook gains for the whole frame are obtained from the table for fixed-codebook gains.

6.7.1 Identifying Long –Term Spectral Characteristics

The identification of the long-term spectral characteristics shall be performed as done at the encoder (Section 5.3.25).

6.7.2 Pitch Decoding (Adaptive-codebook Contribution)

For the case of type-1 frames, the pitch track and pitch interpolation shall be performed as done in the encoder (Section 5.3.14.2) using the decoded frame pitch lag and the previous frame pitch lags.

The adaptive-codebook vector contribution for the subframe, $c_{unf}^{adp}(n)$, shall be obtained from the adaptive codebook $C^{adp}(n)$ with fractional interpolation of $1/20$ of a sample, using a 21-phase sampled Sinc window. The fractional pitch for the subframe, l_{FRC}^{p}, and the integer pitch for the subframe l_{INT}^{p}, shall be obtained from the interpolated pitch contour in the middle of the subframe. The fractional pitch is the value of the pitch contour at the middle of the subframe, and the integer pitch is the integer part of the fractional pitch.

For the case of type-0 frames (Rate 1 and Rate 1/2], the adaptive-codebook contribution shall be derived from the decoded subframe pitch lags in the same way as in the encoder (Sections 5.6.6 and 5.6.7).

6.7.3 Adaptive-codebook Gain Decoding for Rate 1 and Rate 1/2 (Type-1)

Using the decoded index, the adaptive-codebook gains $g_a(m)$ shall be obtained directly from the quantization tables.

6.7.4 Fixed-codebook Excitation Decoding for Rate 1 and Rate 1/2

Using the received indices comprising the information about the codebook type and pulse positions and signs, along with the relevant decoded information such as reconstructed LP coefficients, adaptive-codebook gains (current for type-1 frames, previous for type-0 frames), pitch lags, and fixed-codebook contribution $c_{unf}^{fix}(n)$ shall be reconstructed as it was done in the encoder (Sections 5.6.13 and 5.6.14).

6.7.5 Fixed-codebook Excitation Contribution for Rate 1/8 and Rate 1/4

The fixed-codebook contribution shall be obtained using a Gaussian random excitation. The seed for the random number generator shall be obtained and updated as was done at the encoder (Section 5.5 and 5.6.19)

6.7.6 Gain Decoding

The adaptive-codebook gain $g_a(m)$ for Rate 1 and Rate 1/2 (type 0) shall be obtained from the relevant tables using the decoded index. The fixed-codebook gains $g_c(m)$ for Rate 1, Rate 1/2 and Rate 1/8 shall be obtained using the proper prediction. The prediction shall be complemented with the decoded prediction errors as it was done at the encoder (Sections 5.4.2, 5.6.17 and 5.6.18) in order to obtain the final $g_c(m)$.

6.7.7 Signal-to-Noise Ratio (SNR) at the Decoder

An estimate of the SNR at the decoder is calculated as:

$$SNR_{dec} = E_{dec}^{S} - E_{dec}^{N} \tag{6.7.7-1}$$

where

$$E_{dec}^{S} = \max\left\{0.95 \cdot E_{dec}^{S} + 0.05 \cdot E_{dec}, E_{dec}\right\}, \tag{6.7.7-2}$$

$$E_{dec}^{N} = \min\left\{0.95 \cdot E_{dec}^{N} + 0.05 \cdot E_{dec}, E_{dec}\right\} \tag{6.7.7-3}$$

and

$$E_{dec} = 10 \cdot \log 10\left(\frac{1}{L_{frm}} \sum_{i=0}^{L_{frm}-1} syn(i)^2\right). \tag{6.7.7-4}$$

$syn(\cdot)$ is the synthesized speech at the decoder. Note that E_{dec}^{S} is updated during Rate 1 or Rate 1/2 and E_{dec}^{N} is updated during Rate 1/8.

6.8 Frame Erasure Concealment

A bad frame indicator for the frame $bfi(frm)$ is set if an erased frame indicator was received. A counter N_{bfi} counts the number of consecutive bad frames. It is reset to 0 when a good frame is received. A counter N_{gfi} counts the number of consecutive good frames after a bad frame has been received. N_{gfi} is saturated to 4 frames.

The decoder rate of operation and frame type for the current frame are set as follows if $bfi(frm)$ is 1:

$$Rate(frm) = Rate(frm-1) \tag{6.7.7-1}$$

$$Type(frm) = Type(frm-1) \tag{6.7.7-2}$$

6.8.1 LSF Extrapolation

If $bfi(frm) = 1$, use LSF parameters of previous frame i.e. $\underline{lsf}^{frm}(i) = \underline{lsf}^{frm-1}(i), \quad i = 1,\ldots,10$. The state variables of the LSF prediction are recomputed with an attenuation as:

$$\varepsilon_j(i) = 0.6 \cdot (\underline{lsf}'(i) - \sum_{p=1}^{O_{Rate}^{MA}} b_{j,p}^{Rate}(i) \cdot \varepsilon_{j^*,p}^{q}(i)\ ) \qquad i = 1\ldots,10 \tag{6.8.1-1}$$

where $\underline{lsf}'$ is the mean-removed $\underline{lsf}$.

Special care is catered for good frames that are subsequent to a bad frame if F_c is set and the following is true:

if $1 \le N_{gfi} \le 2$ and $lsf(i) - lsf(i-1) < 75\ \text{Hz}$, set the minimum spacing between consecutive LSFs to 125 Hz, and set an attenuation flag F_a to 1. If $lsf(i) - lsf(i-1) \ge 75\ \text{Hz}$, set the minimum spacing between consecutive LSFs to 100 Hz. Otherwise, the minimum spacing is set to 50 Hz.

A caution flag F_c is set for the subsequent frame if the following is true:

$$bfi(frm) = 1 \text{ and } \left(k(1) > 0.4 \text{ or } \left(\text{Rate} = 1/8 \text{ and } SNR_{dec} > 50\ \text{dB}\right)\right) \tag{6.8.1-2}$$

where $k(1)$ is the first reflection coefficient obtained from the decoded linear prediction coefficients. In this case the state variables of LSF prediction are prepared for the next frame, and are reset to:

$$\varepsilon_j(i) = \underline{lsf}'(i) - \sum_{p=1}^{O_{Rate}^{MA}} b_{j,p}^{Rate}(i) \cdot \varepsilon_p^{q}(i) \qquad i = 1\ldots,10 \tag{6.8.1-3}$$

where $\underline{lsf}'(i)$ is the mean-removed vector of a preset $\underline{lsf}^{FLAT}(i)$ derived from a flat spectrum.

The caution flag F_c is reset to 0 if $N_{gfi} > 3$.

6.8.2 Pitch Extrapolation

Different pitch extrapolation algorithms are devised for type-0 and type-1 frames.

For Type-0: if $bfi(frm)$ is 1,

$$l_p(m) = l_p(m-1) \quad \text{if } k(1) < 0$$
$$l_p(m) = l_p(m-1) + 1 \quad \text{otherwise} \tag{6.8.2-1}$$

where $k(1)$ is the first reflection coefficient from the decoded prediction coefficients.

For type 1, if $bfi(frm)$ is 1,

If $\left|l_{pp}^{q}(frm-1) - l_{pp}^{q}(frm-2)\right| < 4$ and $\left|l_{pp}^{q}(frm-2) - l_{pp}^{q}(frm-3)\right| < 4$:

$$l_{pp}^{q}(0) = \min\left(147, \max\left(17, 3 \cdot A + B\right)\right), \tag{6.8.2-2}$$

otherwise

$$l_{pp}^{q}(frm) = l_{pp}^{q}(frm-1). \tag{6.8.2-3}$$

Where A and B are obtained through linear extrapolation as:

$$A = 0.5 \cdot \left(l_{pp}^{q}(frm-1) - l_{pp}^{q}(frm-2)\right) \tag{6.8.2-4}$$

and

$$B = \frac{l_{pp}^{q}(frm-3) + l_{pp}^{q}(frm-2) + l_{pp}^{q}(frm-1) - 3 \cdot A}{3} \tag{6.8.2-5}$$

A special feature is introduced to correct the pitch track and the ACB buffer for isolated frame errors. This takes place if $bfi(frm)$ is 0 and $bfi(frm-1)$ is 1 and $bfi(fm-2)$ is 0. The pitch track correction happens only the current frame and the previous frame are of type 1. In this case, we use the correctly received $l_{pp}^{q}(frm)$ to correct $l_{pp}^{q}(frm-1)$ via linear interpolation as:

$$l_{pp}^{q}(frm-1) = \left(\frac{l_{pp}^{q}(frm) - l_{pp}^{q}(frm-2) \cdot 159}{319}\right) + l_{pp}^{q}(frm-2). \tag{6.8.2-6}$$

The correction of the adaptive-codebook buffer happens for either frame type if both the current and previous frame is either Rate 1 or Rate 1/2 and $g_a > 0.5$. The correction of the adaptive-codebook buffer is basically the re-computing of the excitation of the preceding bad frame using the corrected pitch lag and tracks. The gains and fixed-codebook excitations remain as they were extrapolated during the preceding bad frame.

6.8.3 Gain Extrapolation

For type-1 frames:

$$g_a(m) = 0.95 \quad \text{if } N_{bfi} = 1$$
$$g_a(m) = \alpha \cdot g_a(m\text{-}1) \quad \text{if } N_{bfi} > 1 \tag{6.8.3-1}$$

where α is an attenuation factor that is a function of N_{bfi}. An average adaptive-codebook gain $\overline{g}_a$ over the preceding frame is computed.

$$g_c(m) = 0.0 \quad \text{if } \overline{g}_a > 0.6$$
$$g_c(m) = \beta \cdot \sqrt{\frac{E_{sc}(m-1)}{E_c(m)}} \quad \text{otherwise} \tag{6.8.3-2}$$

where β is an attenuation factor that is a function of N_{bfi}, $E_{sc}(m-1)$ is the gain scaled fixed-codebook excitation energy from the preceding subframe, and $E_c(m)$ is fixed-codebook excitation energy of the current subframe.

For Rate 1 and Rate 1/2 type-0 frames, the gains are extrapolated as follows:

If $N_{bfi}=1$, an average adaptive-codebook gain $\overline{g}_a$ over the preceding subframes is computed in the first frame:

$$\overline{g}_a = \frac{1}{L}\sum_{i=1}^{L} G_a(8-i), \tag{6.8.3-3}$$

where G_a is buffer containing the last 8 g_a and L is a function of the pitch lag. A factor $F_a(m,0)$ measuring the proportion of the adaptive-codebook excitation energy to the total excitation energy is computed on a subframe basis.

$$\begin{cases} g_a = 0.95 & k(1) < 0.4 \text{ and } E_f(frm-1) \text{ and } \max(F_a(m,-1)) > 0.7 \\ g_a = \min(0.95, \overline{g}_a) & \left(k(1) < 0.4 \text{ and } E_f(frm-1) \text{ and } \max(F_a(m,-1)) > 0.7\right) \text{or } F_{ons} = 1 \\ g_a = 0.2 & \text{otherwise} \end{cases} \tag{6.8.3-4}$$

where $E_f(frm-1)$ is the energy of the previous frame and F_{ons} is a flag indicating an onset at the previous frame. For other subframes, $g_a(m) = g_a(m-1)$.

If $N_{bfi} > 1$

$$g_a(m) = \alpha \cdot g_a(m-1), \tag{6.8.3-5}$$

where α is a function of N_{bfi}.

The fixed-codebook gain is given as

$$g_c(m) = \beta \cdot \sqrt{\frac{E_{sc}(m-1)}{E_c(m)}}, \tag{6.8.3-6}$$

where β is function of both N_{bfi} and $g_a(m)$. F_{ons} is set if there is an increase of at least 13 dB between the current and the previous 2 frames or an increase of at least 7 dB in LPC gain between the current and previous frame.

For Rate 1/4,

$$\hat{G}[j] = \beta \cdot \hat{G}_{old}[j] \qquad 0 \le j < 10 \tag{6.8.3-7}$$

where β is a function of N_{bfi}, and $\hat{G}_{old}[j]$ is the set gains for the previous Rate 1/4 frame.

For Rate 1/8,

$$g_c(m) = \beta \cdot \sqrt{\frac{E_{sc}(m-1)}{E_c(m)}} \tag{6.8.3-8}$$

where β is a function of N_{bfi}.

6.8.4 Excitation Generation

A Gaussian random excitation $r(n)$ is generated using the random noise generator described in Section 5.6.19. $r(n)$ is generated regardless of rate in the case of bad frame, The seed used, is a signed 16-bit static variable initialized to 21845. The update of the seed is described in Section 5.6.19. No reset of this seed is performed.

6.9 Post-Processing

Routine name: `PPR_post_process`

Input:

- Interpolated prediction coefficients, $\hat{a}_m^q(i)$
- Decoded pitch lag value, $l_p(m)$
- Synthesized speech, $\hat{s}(n)$

Output:

- Postfiltered synthesized speech, $s'_p(n)$

Post-processing is made up of three functions: adaptive postfiltering, high-pass filtering and signal up-scaling. The adaptive postfilter is a cascade of three filters: a pitch postfilter $H_p(z)$, a tilt compensation filter $H_t(z)$, and a short-term postfilter $H_s(z)$, followed by and adaptive gain control (AGC). The postfilter is updated every subframe. The postfiltering process is organized as follows. First the synthesized speech $\hat{s}(n)$ is filtered through a filter $A(z/\gamma_{num})$ to produce a weighted residual speech signal $\hat{r}_w(n)$. The signal $\hat{r}_w(n)$ is used to compute the pitch delay l_{pf} and a gain g_{pf}. The signal $\hat{r}_w(n)$ is filtered through the pitch postfilter $H_p(z)$ to produce the signal $r'_w(n)$. $r'_w(n)$ is passed through a tilt compensation filter $H_t(z)$, leading to a signal $r_t(n)$. $r_t(n)$ is filtered through a synthesis filter $1/A(z/\gamma_{den})$ to give a signal $\hat{s}_p(n)$.

Finally, an AGC is then applied between the output signal $\hat{s}_p(n)$ and $\hat{s}(n)$ resulting in the signal $s'_p(n)$. The high-pass filtering and scaling are performed on $s'_p(n)$.

6.9.1 Weighted residual signal

The synthesized signal $\hat{s}(n)$ is filtered through a filter

$$A\left(\frac{z}{\gamma_n}\right) = 1 + \sum_{i=1}^{10} a_i \gamma_{num}^i z^{-i} \qquad (6.9.1\text{-}1)$$

to produce a weighted residual speech signal $\hat{r}_w(n)$, where $\{a_i\}$ are the interpolated LP coefficients and γ_{num} is a adaptive weighting factor, updated each subframe as:

$$\gamma_{num} = 0.75 \cdot \gamma_{num} + 0.25 \cdot \gamma_0$$

where γ_0 is set to 0.63 for Rate 1, and 0.6 otherwise.

6.9.2 Pitch postfilter

The pitch postfilter is given by:

$$H_p(z) = \frac{g_N}{1 + g_{pf} z^{-l_{pf}}} \tag{6.9.2-1}$$

where l_{pf} is pitch postfilter delay and g_{pf} is pitch postfilter gain and g_N is an AGC gain.

The pitch postfilter delay l_{pf} is obtained as the delay that maximizes the normalized correlation $R_{pf}(D)$. D are the delays in the range of $\lfloor l_p - 1.5, l_p + 1.5 \rfloor$ with 1/8 resolution, l_p is the decoded subframe pitch lag. $R_{pf}(D)$ is given by

$$R_{pf}(D) = \frac{\sum_{n=0}^{L_{SF}} \hat{r}_w(n) \hat{r}_w^D(n)}{\sum_{n=0}^{L_{SF}} \hat{r}_w^D(n) \hat{r}_w^D(n)} \tag{6.9.2-2}$$

where $\hat{r}_w^D(n)$ is the weighted residual at delay D. The pitch postfilter gain g_{pf} is obtained in the following by first computing g_{pf} as:

$$g_{pf} = \frac{\sum_{n=0}^{L_{SF}} \hat{r}_w(n) v(n)}{\sum_{n=0}^{L_{SF}} v(n) v(n)} \tag{6.9.2-3}$$

where $v(n)$ is the optimum contribution from $\hat{r}_w(n)$ at the optimum delay D. Then, g_{pf} is limited to be greater than 1 followed by a scaling as:

$$g_{pf} = \alpha \cdot g_{pf} \tag{6.9.2-4}$$

where α is 0.4 if Rate is 1, or 0.5 if Rate is 1/2, or 0 otherwise.

After the extraction of $v(n)$, $v(n)$ is updated as:

$$v(n) = r_w(n) + g_{pf} \cdot v(n), n = 0,.., L_{SF} - 1 \tag{6.9.2-5}$$

The AGC gain g_N is obtained as

$$g_N = \sqrt{\frac{\sum_{n=0}^{L_{SF}} \hat{r}_w(n) \hat{r}_w(n)}{\sum_{n=0}^{L_{SF}} v(n) v(n)}} \tag{6.9.2-6}$$

The final output $r_w'(n)$ is calculated as:

$$r_w'(n) = g_N \cdot v(n), n = 0,.., L_{SF} - 1 \tag{6.9.2-7}$$

The pitch postfilter is only applied if Rate is 1 or ½. In the case of $bfi(0)$ or N_{bfi} is 1, the optimum search for the delay D is not done but rather D is set to the decoded subframe pitch lag l_p.

6.9.3 Tilt Compensation

The tilt compensation operation consists of the following steps. First, the impulse response $h_{pf}(n)$ of the pole-zero filter

$$\frac{A\left(z/\gamma_{num}\right)}{A\left(z/\gamma_{den}\right)}, \tag{6.9.3-1}$$

truncated to 22 samples, is calculated where $\gamma_{den} = 0.75$. Second, the first reflection coefficient $k_{pf}(1)$ of $h_{pf}(n)$ is computed as:

$$k_{pf}(1) = -\frac{\sum_{j=0}^{18} h_{pf}(j)h_{pf}(j+1)}{\sum_{j=0}^{20} h_{pf}(j)h_{pf}(j)} \tag{6.9.3-2}$$

A tilt factor t_f is obtained form $k_{pf}(1)$ as:

$$t_f = \max(\min(0.5 \cdot k_{pf}(1),0),-0.1) \tag{6.9.3-3}$$

Then t_f is modified if $k_{pf}(1)$ is negative as:

$$\begin{aligned} t_f &= t_f - 0.1, \quad \text{if } F_{flat} = 0 \\ t_f &= t_f - 0.05, \quad \text{otherwise} \end{aligned} \tag{6.9.3-4}$$

Finally, the tilt compensation leads to the output $r_t(n)$ as:

$$r_t(n) = r'_w(n) + t_f \cdot r'_w(n-1), n = 0,..,L_{SF} - 1 \tag{6.9.3-5}$$

6.9.4 Short-Term Postfilter

The output of the tilt compensation filter $r_t(n)$ is passed through a synthesis filter

$$\frac{1}{A\left(z/\gamma_d\right)} \tag{6.9.4-1}$$

to produced the synthesized post-filtered signal $\hat{s}_p(n)$, where $\gamma_d = 0.75$.

6.9.5 Adaptive Gain Control

Adaptive gain control is used to compensate for gain differences between the reconstructed speech $\hat{s}(n)$ and the posfiltered signal $\hat{s}_p(n)$. The gain-scaling factor G_f for the current subframe is calculated as:

$$G_f = \sqrt{\frac{\sum_{j=0}^{L_{SF}} \hat{s}(j)\hat{s}(j)}{\sum_{j=0}^{L_{SF}} \hat{s}_p(j)\hat{s}_p(j)}}. \quad (6.9.5\text{-}1)$$

Then G_f is modified as:

$$\begin{aligned} G_f &= 1.1 \cdot G_f, \quad \text{if } F_{flat} = 0 \\ G_f &= 1.15 \cdot G_f, \quad \text{otherwise} \end{aligned} \quad (6.9.5\text{-}2)$$

The gain scaled post-filtered signal $s'_p(n)$ is obtained as

$$s'_p(n) = g_f(n) \cdot \hat{s}_p(n), n = 0,..,L_{SF} - 1 \quad (6.9.5\text{-}3)$$

where $g_f(n)$ is updated on a sample-by-sample basis as

$$g_f(n) = 0.9 \cdot g_f(n) + 0.1 \cdot G_f \quad (6.9.5\text{-}4)$$

6.9.6 High-pass filtering and up-scaling

A high-pass filter at cutoff frequency of 80 Hz is applied to the gain scaled post-filtered signal $s'_p(n)$. The filter is given by:

$$H_{pf}^{h}(z) = \frac{2.0 \cdot \left(0.92727435 - 1.8544941 z^{-1} + 0.92727435 z^{-2}\right)}{1 - 1.9059465 z^{-1} + 0.9114024 z^{-2}} \quad (6.9.6\text{-}1)$$

The factor 2 in the numerator is for the embedded up-scaling by 2.

1

7 SMV DATA CAPABILITIES

The SMV provides the capability for passing data by replacing some or all of the speech allocated bits by general user data bits. This capability is called in-band or in-bitstream data. In this document, TTY/TDD Baudot Code (45.45 baud and 50 baud) and DTMF are the only data payloads that are defined and supported by the SMV standard.

In-band TTY/TDD Baudot Code is mandatory. All SMV implementations shall include in-band TTY/TDD capability. However, in-band DTMF is not always mandatory. Depending upon applications such as wireless local loop (fixed wireless) or mobility, in-band DTMF can be optional. In any case, in-band and out-of-band DTMF shall not be transmitted simultaneously.

For mobility applications, in-band DTMF signaling shall not be used.

For fixed wireless (also known as wireless local loop) applications, sending DTMF in the forward link is common (e.g., answering machine access). In order to guarantee that in-band DTMF can be decoded correctly in both forward and reverse link, implementing in-band DTMF decoder is mandatory in both directions (forward and reverse), whereas implementing the in-band DTMF encoder is optional.

In-band data may be transported in one of three different ways:

1. Rate 1/2 packet with data (61bits/packet = 3050 bps)
2. Rate 1 packet with data (144 bits/packet = 7200 bps)
3. Rate 1 packet with Rate 1/2 speech and data in remaining bits (64 bits/packet = 3200 bps)

The data rate noted above is the data net rate, due to overhead needed for the detection of the special data packets.

The TTY/TDD and DTMF features define their own error recovery schemes. For general data applications, however, the in-band data channel is a raw bit stream and does not currently define protocols for handshaking or schemes for error detection/correction. The decoder, however, is capable of automatically detecting the in-band data by decoding the packet. No out-of-band signaling is required to send in-band data from the encoder to the decoder.

When the encoder sends in-band data to the decoder, the data is packed in either Rate 1/2 frames or Rate 1 frames and they contain a unique bit pattern that can be identified by the decoder to indicate that the packet contains data. The decoder then reads the header bits to determine which type of data is in the frame and calls the appropriate algorithm to regenerate the signal at the decoder's output, or to send the data to an appropriate application.

The bit allocation for Rate 1/2 data frames is summarized in Table 7-1.

Bit Positions	# of Bits	Rate 1/2 Data Frame
1	1	SVS (frame type)
2 – 4	3	Data Header
5 – 65	61	Data Block
66 - 80	15	Illegal Codeword

Table 7-1: Bit Allocation for Rate 1/2 Data Frames

Rate 1 packets can have one of two different data formats. The first is for transmitting data along with speech parameters (Data + Speech). The second contains only data (Data + Auxiliary Data). The bit allocations for both Rate 1 data formats are shown in Table 7-2.

Bit Positions	# of Bits	Rate 1 Data + Speech Frames	Rate 1 Data + Auxiliary Data Frames
1	1	SVS (frame type)	SVS (frame type)
2 – 81	80	Rate 1/2 Speech Frame	Auxiliary Data
82 – 84	3	Data Header	Data Header
85 – 148	64	Data Block	Data Block
149 – 170	22	Illegal Codeword	Illegal Codeword

Table 7-2: Bit Allocation for Rate 1 Data Frames

For all 3 in-band data formats, the SVS (frame type) field is set to zero and a codeword is put in the packet that is considered illegal for speech frames. A packet is identified as containing data if these to fields are set to the values below. The values of the SVS (frame type) bit and illegal codewords for each of the in-band data formats are in Table 7-3.

	Rate 1/2 Data	Data + Speech	Data + Auxiliary Data
SVS (frame type)	0	0	0
Illegal Codeword	0x7FC8 (15 bits)	0x2002EA (22 bits)	0x2002EA (22 bits)

Table 7-3: SVS (frame type) and Illegal Codeword Values

Whereas the SVS and illegal codeword identify the packet as containing data, the Data Header is a 3-bit field, which identifies the data format and payload. Table 7-4 contains a description of the Data Header Field for Rate 1 and Rate 1/2 packets.

Data Header Value	Rate 1 (Bits 82-84)	Rate 1/2 (Bits 2-4)
0	Reserved	Reserved
1	Data + Speech	Data
2	Reserved	TTY/TDD
3	Reserved	DTMF
4	Data + Auxiliary Data	Reserved
5 – 7	Reserved	Reserved

Table 7-4: Data Header Description

7.1 TTY/TDD Payload Format

The TTY/TDD payload shall be transported using the Rate 1/2 packet data format only. Rate 1 data formats shall not be used for this application. For TTY/TDD payloads, the Data Header field shall be set to 2 (see Table 7-4) and the first 8 bits of the data field shall correspond to the TTY Type field. The Baudot code is currently the only TTY Type supported by this standard. Table 7.1-1 summarizes the valid values for the TTY Type field.

TTY Type	Description
0	Reserved
1	Baudot Code
2 – 255	Reserved

Table 7.1-1: TTY Type Field

7.1.1 Baudot Code Payload

Payloads for TTY/TDD Baudot codes at 45.45 baud and 50 baud are defined for SMV. For transmitting the Baudot code, the Data Field shall be set to 2 and TTY Type shall be set to 1. For Baudot, the bits in the Rate 1/2 packet are defined as in Table 7.1-2.

Rate 1/2 Packet Bit Positions	# of Bits	Field	Value
1	1	SVS (frame type)	0
2 – 4	3	Data Header	2
5 – 12	8	TTY Type	1
13 – 20	8	TTY Header	See Table 7.3-1
21 – 36	16	TTY Character	See Table 7.3-2
37 – 52	16	TTY Rate	See Table 7.3-3
53 – 65	13	Reserved	N/A
66 - 80	15	Illegal Codeword	0x7FC8

Table 7.1-2: Rate 1/2 Packet for TTY/TDD Baudot Code Payload

7.2 DTMF Payload Format

The DTMF payload shall be transported using the Rate 1/2 packet data format only. Rate 1 data formats shall not be used for this application. Table 7.2-1 shows a Rate 1/2 packet with the format for a DTMF payload. The Data Header field shall be set to 3 (see Table 7-4) and the first 16 bits of the data field shall correspond to a 16-bit signed value DTMF digit, as specified in Table 7.2-2.

Rate 1/2 Packets Bit Positions	# of Bits	Field	Value
1	1	SVS (frame type)	0
2 – 4	3	Data Header	3
5 – 20	16	DTMF Digit	See Table 7.2-2
21 – 65	45	Reserved	N/A
66 – 80	15	Illegal Codeword	0x7FC8

Table 7.2-1: Rate 1/2 Packet with DTMF Payload

Data Field	DTMF Signal
1	1
2	2
3	4
5	5
6	6
7	7
8	8
9	9
10	0
11	*
12	#
13	A
14	B
15	C
16	D
-1	no signal detected

Table 7.2-2: DTMF Data Values

7.3 TTY/TDD Operation

The following feature provides a method for reliably transporting the 45.45 bps and 50 bps Baudot code in the SMV, making digital wireless telephony accessible to TTY/TDD users. The feature supports voice carryover/hearing carryover (VCO/HCO). VCO allows a TTY/TDD user to switch between receiving TTY and talking into the phone. Similarly, HCO allows a user to switch between transmitting TTY characters and picking up the phone to listen. When Baudot tones are not present, the vocoder operates as usual; there is no modification or added delay to the voice path when speech is present.

The Baudot code is specified in ITU-T Recommendation V.18. The Baudot code is a carrierless, binary FSK signaling scheme. A 1400 Hz. tone is used to signal a logical "1" and a 1800 Hz. tone is used to signal a logical "0". A 45.45 bps TTY bit has a duration of 22 ms. ± 0.4 and a 50 bps bit has a duration of 20 ms. ± 0.4, and a character consists of 1 start bit, 5 data bits, and 1.5 – 2 stop bits. When a character is not being transmitted, silence, or a noisy equivalent, is transmitted. Hence, a TTY character spans a minimum of 6 speech processing frames. When the TTY encoder processing detects a character, it sends the character and its header (see Section 7.3.1 for a description of the header) to the decoder over a minimum of 6 consecutive frames and a maximum of 16 frames. Because of channel impairments, the decoder may not receive all of the packets sent by the encoder because of frame erasures and random bit errors. The decoder uses the redundancy to correct any corrupted TTY information. Once the decoder recognizes the TTY character being sent, the decoder's TTY repeater regenerates the Baudot tones corresponding to that

character. The TTY processing in the encoder processes the received PCM input one frame at a time and labels each frame as either NON_TTY, TTY_ONSET, or as a TTY character. The vocoder is in one of two states, TTY_MODE or NON_TTY_MODE. In the absence of Baudot tones, the encoder

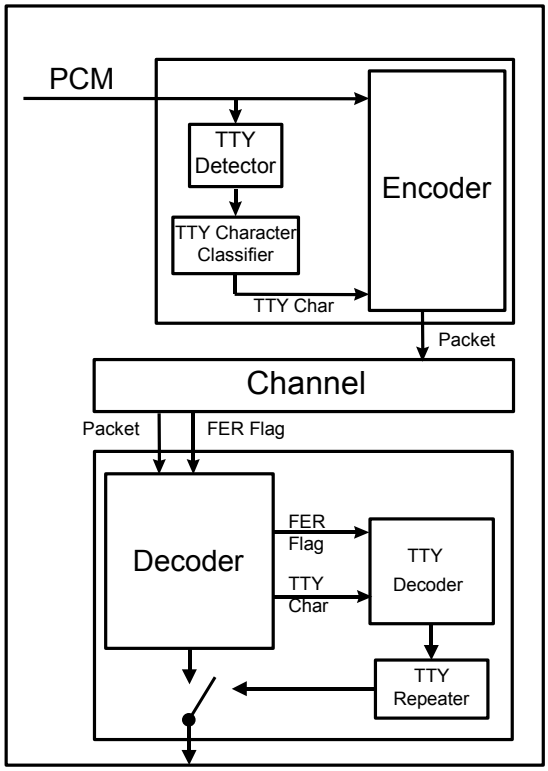

Figure 7.3-1: TTY/TDD Processing Block Diagram

and decoder are in the NON_TTY_MODE and the encoder and decoder process the frame as speech. When Baudot tones are present, the encoder and decoder enter TTY_MODE and process the TTY information as described below. When the vocoder stops detecting TTY information, it resumes processing speech.
It is recommended that a mechanism exist to disable the TTY/TDD feature in the vocoder.

7.3.1 TTY Header and Character Format

The TTY information passed from the encoder to the decoder contains a header, character information, and the baud rate. When the encoder is transmitting a TTY character, the header contains a sequence number to distinguish that character from the characters immediately before and after. The same TTY information is transmitted for each instance of a character for a minimum of 6 frames and a maximum of 16 frames. The header cycles through its range of valid values, one value for each instance of a character. During the TTY_ONSET state, the encoder sends the decoder a TTY_SILENCE message in the header and character field. The TTY header uses 2 bits in an 8-bit field (see Table 7.3-1). The TTY character uses 5 bits in a 16-bit field (see Table 7.3-2). The TTY baud rate uses 1 bit in a 16-bit field (see Table 7.3-3).

Range	Description

0	TTY_SILENCE
1 – 3	TTY Counter
4 – 255	Reserved

Table 7.3-1: Range of Values for TTY/TDD Baudot Header Field

Range	Description
4	TTY_SILENCE (when TTY Header = 0)
0 – 31	TTY Character
32 – 65535	Reserved

Table 7.3-2: Range of Values for TTY/TDD Baudot Character Field

Range	Description
0	45.45 Baud
1	50 Baud
2 – 65535	Reserved

Table 7.3-3: Range of Values for TTY/TDD Baud Rate Field

7.3.2 TTY Encoder Processing

The TTY encoder processing takes the larger task of detecting TTY characters and divides it into a series of smaller tasks, creating different levels of detection. It is through this divide and conquer approach that the tty_enc() routine has low complexity in the absence of Baudot tones.

The first level of detection is to divide the 160 samples in the speech frame into 10 blocks of 16 samples. These blocks are called detection intervals, or dits. Each dit is classified as NON_TTY, LOGIC_0, or LOGIC_1.

The next level of detection is to determine if there are enough LOGIC_0 or LOGIC_1 dits in a row to form a TTY bit. The transition from a "0" bit to a "1" bit is used to determine the onset of a TTY character and the TTY_SILENCE message is sent to the decoder. The TTY_SILENCE message shall continue to be sent until a TTY character is detected, or until a NON_TTY frame is detected. When enough bits are detected to form a character, the rate of the character is determined and the TTY character information is sent to the decoder.

7.3.2.1 TTY Encoder Inputs

- TTY character information from the previous frame (header, TTY character, and baud rate).
- 160 PCM samples from the output of the high pass filter.

7.3.2.2 Dit Classification

The 160 samples from the high pass filter's output are converted into 10 dits. For every block of 16 samples, it computes the spectral energy at the mark and space frequencies using a 16-point DFT at 1400 Hz and 1800 Hz with a rectangular window. The ratio of the maximum energy between the mark and space energy and the total energy is compared to a threshold; i.e.,

$$\frac{\max(mark\_energy, space\_energy)}{total\_energy} > \text{THRESH}$$

If that threshold is exceeded, the dit is labeled as either a mark or a space, whichever one has the greater energy. If the threshold is not met, the dit is labeled as NON_TTY.

A second threshold is applied to the total energy of the dit. If the total energy is less than the threshold, the dit is classified as NON_TTY. This minimum energy threshold prevents the TTY detector from being overly sensitive to low level tones that may have been unintentionally reflected back into the TTY detector. The threshold is set at approximately -50 dBm.

7.3.2.3 Dits to Bits

The dits are used to form a TTY bit. Dits are classified as LOGIC_0, LOGIC_1, or UNKNOWN. A nominal bit consists of 11 dits for 45.45 baud and 10 dits for 50 baud. A bit is not required to have a continuous run of LOGIC_0's or LOGIC_1's in order to be detected. Spurious UNKNOWN detections are permitted, up to a threshold.

A TTY bit is searched by looking at the dits within a variable length sliding window. The window varies in length from 8 dits to 13 dits. The number of LOGIC_0's, LOGIC_1's, and UNKNOWN dit detections are counted in the window. The dits in the search window must meet the following criteria in order to detect a TTY bit:

- Minimum of 6 LOGIC_0 (LOGIC_1) dits
- Maximum of 2 LOGIC_1 (LOGIC_0) dits
- Maximum of 5 UNKNOWN dits

Heuristics are also applied so that the sliding window is centered over the TTY bit being detected. If all of the thresholds are met, a "0" ("1") TTY bit is declared, and the overall length of the bit and the number of bad dits within the window are recorded. If the dits in the search window do not meet the criteria for a TTY bit, a gap between TTY bits is declared and the gap is recorded. The window is slid by one dit and the search is repeated.
The length of the search window is allowed to vary. The length is adjusted depending on the previous character's baud rate, and where the mark/space transitions occur.

Two history buffers are maintained to record the detected TTY bits and gaps. The array tty_bit_hist[] maintains a history of the bits that are detected and tty_bit_len_hist[] stores the lengths, in dits, of the gaps and TTY bits that are detected. Both arrays are 9 elements long, there is one element for the start bit, five for the data bits, one for the stop bit, and one for the memory bit, the bit before the start bit, as depicted in Figure 7.3-2~~Figure 7.3-2~~. The last element in the array is used to record partially detected TTY bits.

0	1	2	3	4	5	6	7	8
Memory Bit	Start Bit	LSB Data 0	Data 1	Data 2	Data 3	MSB Data 4	Stop Bit	Next Bit

Figure 7.3-2: TTY Bit History Buffer

When a TTY bit or a gap is detected, its value is recorded in tty_bit_hist[] and its length is stored in tty_bit_len_hist[]. The length is used by get_tty_char() to threshold the length of a candidate character, and by tty_rate() to determine if a detected character is 45.45 baud or 50 baud. Since the length of the memory bit is irrelevant, tty_bit_len_hist[0] is used to count the number of NON_TTY and TTY_SILENCE dits that were detected within the TTY bits.
Because the speech frames may not coincide with the boundaries of the TTY bits, it is possible that a bit may straddle two speech frames. It is possible, therefore, that the sliding window may contain only a partial bit within a frame. These partial detections are recorded in element 8 of the TTY bit history buffers, labeled "Next Bit" in Figure 7.3-2.

7.3.2.4 TTY Character Classification

The routine get_tty_char() checks if the detected bits form a TTY character. The following conditions must be met in order for a character to be declared.

- The bit preceding the start bit must not be a “0”.
- The start bit must be a “0”
- The stop bit must be a “1”
- The length of the candidate character, in dits, must be within a maximum and minimum threshold.
- The number of bad dit detections within the character must be within a threshold.

If all of the conditions are met, a character is declared.

Once a character is found, the character information and its header are sent to the decoder a minimum of 6 frames and a maximum of 16 frames. The constant FRAMING_HANGOVER dictates the maximum number of times the information for the same character is sent. If a new character is framed before FRAMING_HANGOVER is reached, the information for the old character is terminated and the new information is sent to the decoder.

7.3.2.5 TTY Baud Rate Determination

After a character has been detected, tty_rate() determines the baud rate of the character by counting the length, in dits, of the start bit and the five data bits. The length of the stop bit is not used because its length is too variable.

A character with a length of 63 dits or greater is declared 45.45 baud, otherwise it is declared 50 baud. A hangover of three characters is maintained before tty_rate() will switch from one baud rate to the other. That is to say, if the baud rate changes in the middle of a call, three consecutive characters must be detected at the new baud rate in order for tty_rate() to declare the new baud rate.

7.3.2.6 TTY State Machine

The routine get_tty_state() is responsible for changing the state of the TTY encoder processing. There are 3 states, NON_TTY_MODE, TTY_ONSET, and TTY_MODE. Get_tty_state() is responsible for determining NON_TTY_MODE and TTY_ONSET.
Changing TTY state from NON_TTY_MODE to TTY_ONSET requires that a “0” bit is followed by a “1” bit. This rule requires the presence of both the space tone and the mark tone and for the tones to be the correct duration. This test must be met in order to declare TTY_ONSET for the first time.
NON_TTY_MODE is declared by get_tty_state() whenever a non-TTY bit is detected or when a “0” bit occupies the bit preceding the start bit.

7.3.3 TTY/TDD Decoder Processing

The TTY decoder processing must recognize when TTY/TDD information is in the packet, recover from channel impairments to decode the TTY character being sent, and regenerate the Baudot tones corresponding to that character.

When the decoder is in NON_TTY_MODE, the packet is decoded in the usual manner for speech. When the decoder makes the transition to TTY_MODE, it mutes its output until it receives TTY character information or until it transitions back to NON_TTY_MODE.

7.3.3.1 TTY Decoder Inputs

- TTY Information (header, TTY character, baud rate).

- Bad frame indicator

7.3.3.2 Decoding the TTY/TDD Information

The task of detecting the presence of TTY information, recovering from frame and bit errors, and decoding the TTY character is performed in tty_dec(). If the routine detects that TTY information is being sent, the tty_dec() flag is set to non-zero and the PCM buffer is filled with the appropriate Baudot tones. If TTY is not detected, the flag is set to zero and the PCM buffer is returned unmodified.

The routine labels each frame as NON_TTY, TTY_SILENCE, FER, or a TTY character, and maintains a history buffer of these classifications for 11 frames: 9 frames of lookahead, 1 current frame, and 1 frame of lookback (see Table 7.3-4). The most recent packet enters the buffer at location 0, but the decision for the current frame is based on the contents of element 9. The buffer is updated at the end of each frame, shifting its contents to the right by one. The buffer is initialized to NON_TTY.

Frame:	0	1	2	3	4	5	6	7	8	9	10
Description:	Lookahead									Current Frame	Look-back

Table 7.3-4: tty_dec() History Buffer

At the start of each frame, the most recent information is sanity checked to see if it is consistent with TTY information. If the frame erasure flag is set, the frame is labeled FER, otherwise the TTY/TDD information is checked to see if the header and TTY character fields fall within the allowed range of values. If all of the tests pass, Frame 0 is labeled with the TTY character in the history buffer.

The TTY decoder processing can reliably regenerate the TTY characters despite channel impairments because the character information is transmitted a minimum of 6 times from the encoder. Errors are corrected by a voting process. The current frame and nine frames of lookahead are used to determine the correct TTY header, character, and baud rate. Errors are replaced with the winner of the voting process.

Voting is conducted under the following conditions:

- Any time the current frame is labeled as FER.
- Every time the current frame contains information for the start of a new character. Because a new character must contain a minimum of 6 frames of the same information, a vote is taken to verify that the information is present before it will generate the tones for that character. Once a character wins the vote, any frame errors, bit errors, or other inconsistencies are corrected in the frame window where the character information is expected, i.e. the current frame and the adjacent frames of lookahead will contain the same header and character information.
- Any time the current frame contains TTY_SILENCE or a TTY character, and the frame of lookback contains NON_TTY. This makes it harder for the decoder to erroneously go into TTY_MODE, thus preventing false alarms when speech is present.

Once tty_dec() makes its decision on the current frame, tty_dec() calls tty_gen() to generate the appropriate PCM samples.

7.3.3.3 Baudot Generator

Once the current frame is labeled by tty_dec(), tty_gen() is called to fill the PCM buffer with the appropriate Baudot tones. In the case of NON_TTY, the PCM buffer is returned unmodified. In the case of TTY_SILENCE, the PCM is muted.

Generating TTY characters is more involved because one character spans many frames, so tty_gen() must generate the Baudot tones one subframe at a time. When a TTY character needs to be regenerated, tty_gen() puts a subframe's worth of samples in the PCM buffer. It keeps track of which bit it is in the middle of generating and the number of samples left to generate for that bit, so that the next time it is called, it can pick up where it left off. Once tty_gen() begins to generate a character, it will generate the entire character before it will generate the next character. This is done so that the repeater will only generate valid TTY characters.

There exists logic in tty_gen() to detect when the next character arrives before the current one is finished. If the next character arrives before the current one can be regenerated, the bit duration is reduced by 4 samples and a minimum of 1 stop bit is generated. Otherwise, nominal bit lengths are generated, i.e. 22 ms bits for 45.45 baud and 20 ms bits for 50 baud, and a minimum of 1.5 stop bits is generated for 45.45 baud, and a minimum of 1.75 stop bits is generated for 50 baud.

There exists a provision in the ITU-T Recommendation V.18 for the TTY/TDD device to extend its stop bit in order to prevent a TTY/TDD device from detecting its own echo. This routine will extend the stop bit a maximum of 300 ms if a TTY character is followed by silence. If a new character arrives before 300 ms has elapsed, the extended stop bit is terminated and the new character is generated immediately.

The tones themselves are generated by tone_gen(). Before tty_gen() returns, it updates the decoder's lookback field in the TTY history buffer with the information corresponding to the last samples generated. For example, if tty_gen() finished generating a character in the middle of the subframe and started generating silence, the lookback field is updated with TTY_SILENCE.

7.3.3.4 Tone Generator

The routine tone_gen() is a sine wave generator. Given a frequency and the number of samples, it will generate the PCM samples by using a 2 tap marginally stable IIR filter. The filter implements the trigonometric identity:

$$\cos(k\omega) = 2 \cdot \cos(\omega) \cdot \cos\big((k-1)\omega\big) - \cos\big((k-2)\omega\big)$$

It is a zero excitation filter, using only its past 2 samples and the cosine of the frequency to be generated, to produce the next sample.

7.4 DTMF

7.4.1 DTMF Detector

The DTMF (Dual Tone Multiple Frequency) detector detects the presence and the information content of incoming DTMF tone signals. Its row and its column frequency as shown in Table 7.4-1 uniquely identify each key on the keypad. It outputs a 16-bit value associated with each of the 16 possible DTMF tone combinations.

		Column Frequency Group			
		1209 Hz	1336 Hz	1477 Hz	1633 Hz
Row Frequency Group	697 Hz	1	2	3	A
	770 Hz	4	5	6	B
	852 Hz	7	8	9	C
	941 Hz	*	0	#	D

Table 7.4-1: Touch-tone telephone keypad:

A row and a column tone is associated with each digit.

7.4.1.1 Description of the DTMF Detector

The DTMF detector consists of three main functions:

1. AGC – normalizes the input frame.
2. Goertzel – performs spectral analysis at the 8 frequencies and at the second harmonic of the column frequencies using an 80 point Goertzel algorithm. The Goertzel algorithm provides an efficient means of calculating the DFT energy at a specific frequency (see Oppenheim and Schafer, Digital Signal Processing, Prentice-Hall) In addition a 160 point Goertzel algorithm is used at the first and fourth row frequency in order to satisfy the requirements found in Bellcore document GR-506-CORE LSSGR: Signaling and Analog Interfaces Issue 1, June 1996.
3. Check– checks the spectral energies to determine if a valid DTMF tone is detected.

7.4.1.2 INPUT

- Pointer to 16 bit linear time series for current 20 msec. frame and 10 msec. look-ahead.

7.4.1.3 OUTPUT

- 16 bit valid DTMF tone digit if a tone is detected or 0xffff to indicate no DTMF present.