# ETSI ES 202 212 V1.1.1 (2003-11)

*ETSI Standard*

## Speech Processing, Transmission and Quality Aspects (STQ);
## Distributed speech recognition;
## Extended advanced front-end feature extraction algorithm;
## Compression algorithms;
## Back-end speech reconstruction algorithm

Reference

DES/STQ-00034

Keywords

performance, speech, transmission

***ETSI***

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

# Contents

# Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

# Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Speech Processing, Transmission and Quality Aspects (STQ).

# Introduction

The performance of speech recognition systems receiving speech that has been transmitted over mobile channels can be significantly degraded when compared to using an unmodified signal. The degradations are as a result of both the low bit rate speech coding and channel transmission errors. A Distributed Speech Recognition (DSR) system overcomes these problems by eliminating the speech channel and instead using an error protected data channel to send a parameterized representation of the speech, which is suitable for recognition. The processing is distributed between the terminal and the network. The terminal performs the feature parameter extraction, or the front-end of the speech recognition system. These features are transmitted over a data channel to a remote "back-end" recognizer. The end result is that the degradation in performance due to transcoding on the voice channel is removed and channel invariability is achieved.

The present document presents a standard for a front-end to ensure compatibility between the terminal and the remote recognizer. The first ETSI standard DSR front-end ES 201 108 [1] was published in February 2000 and is based on the Mel-Cepstrum representation that has been used extensively in speech recognition systems. This second standard is for an Advanced DSR front-end that provides substantially improved recognition performance in background noise. Evaluation of the performance during the selection of the present document showed an average of 53 % reduction in speech recognition error rates in noise compared to ES 201 108 [1].

For some applications, it may be necessary to reconstruct the speech waveform at the back-end. Examples include:

- Interactive Voice Response (IVR) services based on the DSR of "sensitive" information, such as banking and brokerage transactions. DSR features may be stored for future human verification purposes or to satisfy procedural requirements.
- Human verification of utterances in a speech database collected from a deployed DSR system. This database can then be used to retrain and tune models in order to improve system performance.
- Applications where machine and human recognition are mixed (e.g. human assisted dictation).

In order to enable the reconstruction of speech waveform at the back-end, additional parameters such as fundamental frequency (F0) and voicing class need to be extracted at the front-end, compressed, and transmitted. The availability of tonal parameters (F0 and voicing class) is also useful in enhancing the recognition accuracy of tonal languages, e.g. Mandarin, Cantonese, and Thai.

The present document specifies a proposed standard for an Extended Advanced Front-End (XAFE) that extends the noise-robust advanced front-end with additional parameters, viz., fundamental frequency F0 and voicing class. It also specifies the back-end speech reconstruction algorithm using the transmitted parameters.

# 1 Scope

The present document specifies algorithms for extended advanced front-end feature extraction, their transmission, back-end pitch tracking and smoothing, and back-end speech reconstruction which form part of a system for distributed speech recognition. The specification covers the following components:

a) the algorithm for advanced front-end feature extraction to create Mel-Cepstrum parameters;

b) the algorithm for extraction of additional parameters, viz., fundamental frequency F0 and voicing class;

c) the algorithm to compress these features to provide a lower data transmission rate;

d) the formatting of these features with error protection into a bitstream for transmission;

e) the decoding of the bitstream to generate the advanced front-end features at a receiver together with the associated algorithms for channel error mitigation;

f) the algorithm for pitch tracking and smoothing at the back-end to minimize pitch errors;

g) the algorithm for speech reconstruction at the back-end to synthesize intelligible speech.

NOTE: The components a), c), d) and e) are already covered by the ES 202 050 [2]. Besides these (four) components, the present document covers the components b), f) and g) to provide back-end speech reconstruction and enhanced tonal language recognition capabilities. If these capabilities are not of interest, the reader is better served by (un-extended) ES 202 050 [2].

The present document does not cover the "back-end" speech recognition algorithms that make use of the received DSR advanced front-end features.

The algorithms are defined in a mathematical form, pseudo-code, or as flow diagrams. Software implementing these algorithms written in the 'C' programming language is contained in the ZIP file es_202212v010101p0.zip which accompanies the present document. Conformance tests are not specified as part of the standard. The recognition performance of proprietary implementations of the standard can be compared with those obtained using the reference 'C' code on appropriate speech databases.

It is anticipated that the DSR bitstream will be used as a payload in other higher level protocols when deployed in specific systems supporting DSR applications. In particular, for packet data transmission, it is anticipated that the IETF AVT RTP DSR payload definition (see bibliography) will be used to transport DSR features using the frame pair format described in clause 7.

The extended advanced DSR standard is designed for use with discontinuous transmission and to support the transmission of Voice Activity information. Annex A describes a VAD algorithm that is recommended for use in conjunction with the Advanced DSR standard, however it is not part of the present document and manufacturers may choose to use an alternative VAD algorithm.

The Extended Advanced Front-End (XAFE) incorporates tonal information, viz., fundamental frequency F0 and voicing class, as additional parameters. This information can be used for enhancing the recognition accuracy of tonal languages, e.g. Mandarin, Cantonese, and Thai.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI ES 201 108: "Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms".

[2] ETSI ES 202 050: "Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms".

[3] ETSI EN 300 903: "Digital cellular telecommunications system (Phase 2+); Transmission planning aspects of the speech service in the GSM Public Land Mobile Network (PLMN) system (GSM 03.50)".

# 3 Definitions, symbols and abbreviations

## 3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

**analog-to-digital conversion:** electronic process in which a continuously variable (analog) signal is changed, without altering its essential content, into a multi-level (digital) signal

**blind equalization:** process of compensating the filtering effect that occurs in signal recording

NOTE: In the present document blind equalization is performed in the cepstral domain.

**DC-offset:** Direct Current (DC) component of the waveform signal

**discrete cosine transform:** process of transforming the log filter-bank amplitudes into cepstral coefficients

**fast fourier transform:** fast algorithm for performing the discrete Fourier transform to compute the spectrum representation of a time-domain signal

**feature compression:** process of reducing the amount of data to represent the speech features calculated in feature extraction

**feature extraction:** process of calculating a compact parametric representation of speech signal features which are relevant for speech recognition

NOTE: The feature extraction process is carried out by the front-end algorithm.

**feature vector:** set of feature parameters (coefficients) calculated by the front-end algorithm over a segment of speech waveform

**framing:** process of splitting the continuous stream of signal samples into segments of constant length to facilitate blockwise processing of the signal

**frame pair packet:** definition is specific to the present document: the combined data from two quantized feature vectors together with 4 bits of CRC

**front-end:** part of a speech recognition system which performs the process of feature extraction

**magnitude spectrum:** absolute-valued Fourier transform representation of the input signal

**multiframe:** grouping of multiple frame vectors into a larger data structure

**mel-frequency warping:** process of non-linearly modifying the frequency scale of the Fourier transform representation of the spectrum

**mel-frequency cepstral coefficients:** cepstral coefficients calculated from the mel-frequency warped Fourier transform representation of the log magnitude spectrum

**notch filtering:** filtering process in which the otherwise flat frequency response of the filter has a sharp notch at a predefined frequency

NOTE: In the present document, the notch is placed at the zero frequency, to remove the DC component of the signal.

**offset compensation:** process of removing DC offset from a signal

**power spectral density:** squared magnitude spectrum of the signal

**pre-emphasis:** filtering process in which the frequency response of the filter has emphasis at a given frequency range

NOTE: In the present document, the high-frequency range of the signal spectrum is pre-emphasized.

**sampling rate:** number of samples of an analog signal that are taken per second to represent it digitally

**SNR-dependent Waveform Processing (SWP):** processing of signal waveform with objective to emphasize high-SNR waveform portions and de-emphasize low-SNR waveform portions

**voice activity detection:** process of detecting voice activity in the signal

NOTE: In the present document one voice activity detector is used for noise estimation and a second one is used for non-speech frame dropping.

**wiener filtering:** filtering of signal by using Wiener filter (filter designed by using Wiener theory)

NOTE: In this work, objective of Wiener filtering is to de-noise signal

**windowing:** process of multiplying a waveform signal segment by a time window of given shape, to emphasize pre-defined characteristics of the signal

**zero-padding:** method of appending zero-valued samples to the end of a segment of speech samples for performing a FFT operation

## 3.2 Symbols

For the purposes of the present document, the following symbols apply:

**For feature extraction:**

| | |
|---|---|
| $bin$ | FFT frequency index |
| $c(i)$ | cepstral coefficients; used with appropriate subscript |
| $E(k)$ | filter-bank energy; used with appropriate subscript |
| $H(bin)$ or $H(k)$ | Wiener filter frequency characteristic; used with appropriate subscript |
| $h(n)$ | Wiener filter impulse response; used with appropriate subscript |
| $k$ | filter-bank band index |
| $K_{FB}$ | number of bands in filter-bank |
| $lnE$ | log-compressed energy feature appended to cepstral coefficients |
| $n$ | waveform signal time index |
| $N$ | length, (e.g. frame length, FFT length, ...); used with appropriate subscript |
| $P(bin)$ | power spectrum; used with appropriate subscript |
| $S(k)$ | log filter-bank energy; used with appropriate subscript |
| $s(n)$ | waveform signal; used with appropriate subscript |

| | |
|---|---|
| $t$ | frame time index |
| $T_{PSD}$ | number of frames used in the PSD Mean technique |
| $w(n)$ | windowing function in time domain; used with appropriate subscript |
| $W(bin)$ | frequency window |
| $X(bin)$ | FFT complex output |

**For compression:**

| | |
|---|---|
| $\mathrm{Idx}^{i,\, i+1}(t)$ | codebook index |
| $\mathrm{N}^{i,\, i+1}$ | size of the codebook (compression) |
| $\mathrm{Q}^{i,\, i+1}$ | compression codebook |
| $\mathrm{q}_j^{i,\, i+1}$ | jth codevector in the codebook $\mathrm{Q}^{i,\, i+1}$ |
| y(t) | feature vector with 14 components |

## 3.3 Abbreviations

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

| | |
|---|---|
| APM | All-Pole spectral envelope Modelling |
| AVT | Audio/Video Transport |
| BPL | Break Point Lists |
| CDE | Cepstra De-Equalization |
| CLS | CLaSsification |
| COMB | COMBined magnitudes estimate calculation |
| CRC | Cyclic Redundancy Code |
| CTM | Cepstra To Magnitudes transformation |
| DC | Direct Current |
| DCT | Discrete Cosine Transform |
| DSR | Distributed Speech Recognition |
| FB | Filter-Bank |
| FFT | Fast Fourier Transform |
| FIR | Finite Impulse Response |
| FVS | Feature Vector Selection |
| HFB | High Frequency Band |
| HOCR | High Order Cepstra Recovery |
| HSI | Harmonic Structure Initialization |
| IDCT | Inverse Discrete Cosine Transform |
| IETF | Internet Engineering Task Force |
| IVR | Interactive Voice Response |
| LBND | Low-Band Noise Detection |
| LFB | Low Frequency Band |
| LSB | Least Significant Bit |
| LSTD | Line Spectrum to Time-Domain transformation |
| MEL-FB | MEL Filter Bank |
| MF | Mel-Filtering |
| MFCC | Mel-Frequency Cepstral Coefficients |
| MSB | Most Significant Bit |
| NR | Noise Reduction |
| OLA | OverLap-Add |
| PF | PostFiltering |
| PITCH | PITCH estimation |
| PP | Pre-Processing |
| PSD | Power Spectral Density |
| PTS | Pitch Tracking and Smoothing |
| QMF | Quadrature-Mirror Filters |
| RTP | Real Time Protocol |
| SEC | Spectrum and Energy Computation |
| SFEQ | Solving Front-Equation |
| SNR | Signal to Noise Ratio |
| SS | Spectral Subtraction |

| | |
|---|---|
| STFT | Short Time Fourier Transform |
| SWP | SNR-dependent Waveform Processing |
| UPH | Unvoiced PHase |
| VAD | Voice Activity Detection (used for non-speech frame dropping) |
| VADNest | Voice Activity Detection (used for Noise estimation) |
| VADVC | Voice Activity Detection for Voicing Classification |
| VC | Voicing Class |
| VPH | Voiced Phase synthesis |
| VQ | Vector Quantizer |
| XAFE | eXtended Advanced Front-End |

# 4 System overview

This clause describes the distributed speech recognition front-end algorithm based on mel-cepstral feature extraction technique. The specification covers the computation of feature vectors from speech waveforms sampled at different rates (8 kHz, 11 kHz and 16 kHz).

The feature vectors consist of 13 static cepstral coefficients and a log-energy coefficient.

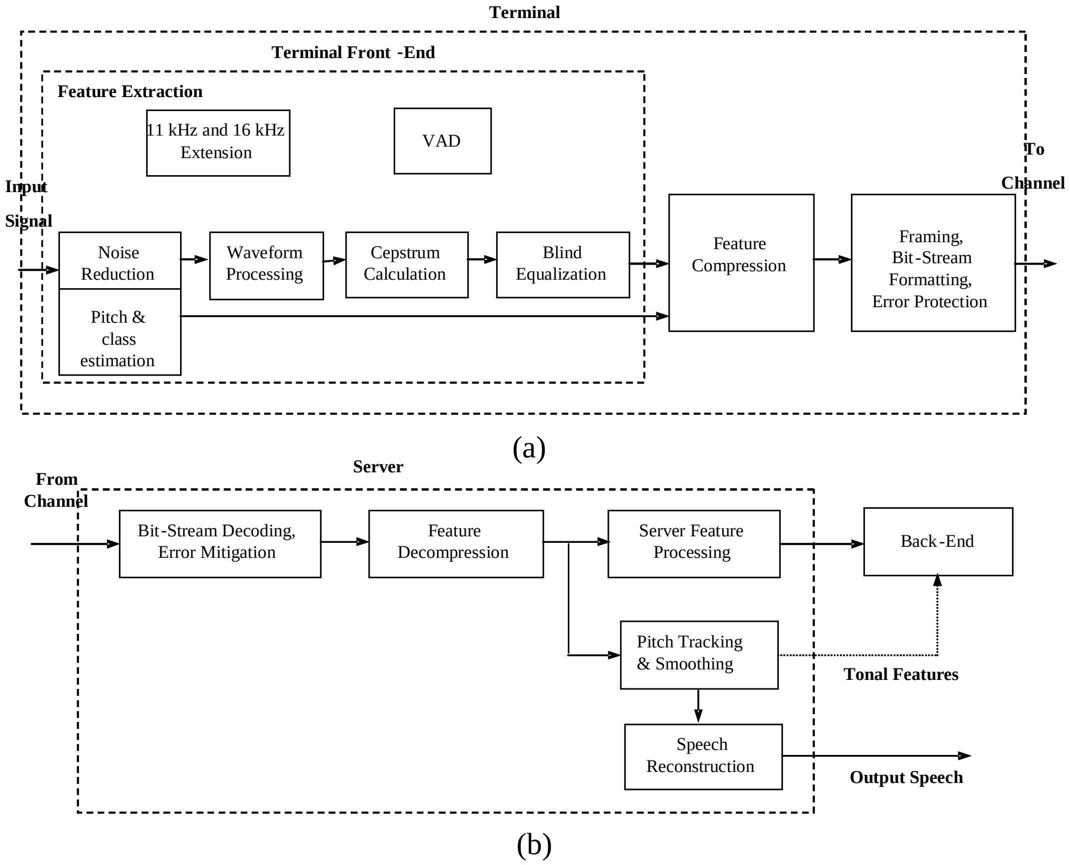
The feature extraction algorithm defined in this clause forms a generic part of the specification while clauses 4 to 6 define the feature compression and bit-stream formatting algorithms which may be used in specific applications.

The characteristics of the input audio parts of a DSR terminal will have an effect on the resulting recognition performance at the remote server. Developers of DSR speech recognition servers can assume that the DSR terminals will operate within the ranges of characteristics as specified in EN 300 903 [3]. DSR terminal developers should be aware that reduced recognition performance may be obtained if they operate outside the recommended tolerances.

Figure 4.1 shows the block scheme of the proposed front-end and its implementation in both the terminal and server sides. In the terminal part, which is shown in figure 4.1(a), speech features are computed from the input signal in the Feature Extraction part. Then, features are compressed and further processed for channel transmission.

In the Feature Extraction part, noise reduction is performed first. Then, waveform processing is applied to the de-noised signal and cepstral features are calculated. At the end, blind equalization is applied to the cepstral features. The Feature Extraction part also contains an 11 kHz and 16 kHz extension block for handling these two sampling frequencies. Voice Activity Detection (VAD) for the non-speech frame dropping is also implemented in Feature Extraction.

At the server side (see figure 4.1(b)), bit-stream decoding, error mitigation and decompression are applied. Before entering the back-end, an additional server feature processing is performed. All blocks of the proposed front-end are described in detail in the following clauses.



(a)

(b)

**Figure 4.1: Block scheme of the proposed extended front-end (a) shows blocks implemented at the terminal side and (b) shows blocks implemented at the server side**

# 5 Feature extraction description

## 5.1 Noise reduction

### 5.1.1 Two stage mel-warped Wiener filter approach

Noise reduction is based on Wiener filter theory and it is performed in two stages. Figure 5.1 shows the main components of the Noise Reduction block of the proposed front-end. The input signal is first de-noised in the first stage and the output of the first stage then enters the second stage. In the second stage, an additional, dynamic noise reduction is performed, which is dependent on the Signal-to-Noise Ratio (SNR) of the processed signal.

Noise reduction is performed on a frame-by-frame basis. After framing the input signal, the linear spectrum of each frame is estimated in the Spectrum Estimation block. In PSD Mean block (Power Spectral Density), the signal spectrum is smoothed along the time (frame) index. Then, in the WF Design block, frequency domain Wiener filter coefficients are calculated by using both the current frame spectrum estimation and the noise spectrum estimation. The noise spectrum is estimated from noise frames, which are detected by a Voice Activity Detector (VADNest). Linear Wiener filter coefficients are further smoothed along the frequency axis by using a Mel Filter-Bank, resulting in a Mel-warped frequency domain Wiener filter. The impulse response of this Mel-warped Wiener filter is obtained by applying a Mel IDCT (Mel-warped Inverse Discrete Cosine Transform). Finally, the input signal of each stage is filtered in the Apply Filter block. Notice from figure 5.1 that the input signal to the second stage is the output signal from the first stage. At the end of Noise Reduction, the DC offset of the noise-reduced signal is removed in the OFF block.

Additionally, in the second stage, the aggression of noise reduction is controlled by Gain Factorization block.

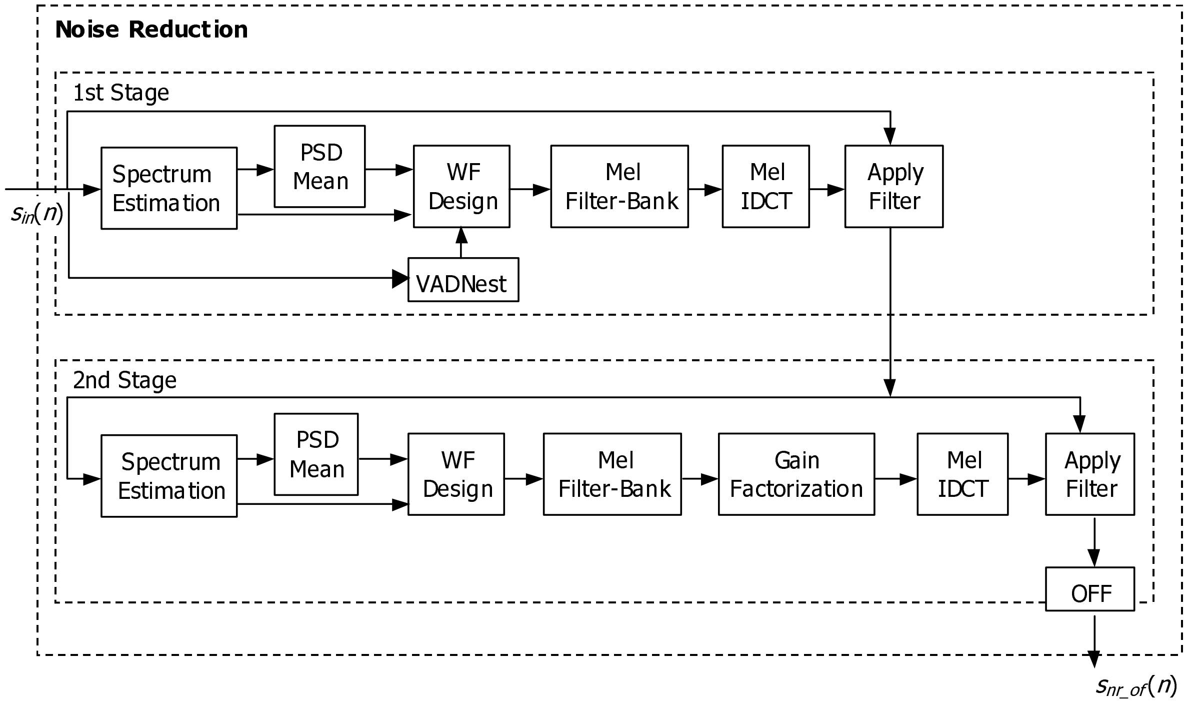


**Figure 5.1: Block scheme of noise reduction**

### 5.1.2 Buffering

The input of the noise reduction block is a 80-sample frame. A 4-frame (frame 0 to frame 3) buffer is used for each stage of the noise reduction. At each new input frame, the 2 buffers are shifted by one frame. The new input frame becomes frame 3 of the first buffer. Then the frame 1 (from position 80 to position 159 in the buffer) of the first buffer is denoised and this denoised frame becomes frame 3 of the second buffer. The frame 1 of the second buffer is denoised and this denoised frame is the output of the noise reduction block. Hence at each stage of the noise reduction block, there is a latency of 2 frames (20 ms). For each stage of the noise reduction block, the spectrum estimation is performed on the window which starts at position 60 and ends at position 259.

### 5.1.3 Spectrum estimation

Input signal is divided into overlapping frames of $N_{in}$ samples. 25 ms ($N_{in}$ = 200) frame length and 10ms (80 samples) frame shift are used. Each frame $s_{in}(n)$ is windowed by a Hanning window of length $N_{in}$, $w_{Hann}(n)$, like:

$$s_w(n) = s_{in}(n) \times w_{Hann}(n), \quad 0 \le n \le N_{in} - 1 \tag{5.1}$$

where:

$$w_{Hann}(n) = 0,5 - 0 \times 5 \times \cos\left(\frac{2 \times \pi \times (n + 0,5)}{N_{in}}\right) \tag{5.2}$$

Then, zeros are padded from the sample $N_{in}$ up to the sample $N_{FFT}$-1, where $N_{FFT}$ =256 is the Fast Fourier Transform (FFT) length:

$$s_{FFT}(n) = \begin{cases} s_w(n), & 0 \le n \le N_{in} - 1 \\ 0, & N_{in} \le n \le N_{FFT} - 1 \end{cases} \tag{5.3}$$

To get the frequency representation of each frame, the FFT is applied to $s_{FFT}(n)$ like:

$$X(bin) = FFT\{s_{FFT}(n)\} \tag{5.4}$$

where *bin* denotes the FFT frequency index.

The power spectrum of each frame, $P(bin)$ $0 \le bin \le N_{FFT}/2$ is computed by applying the power of 2 function to the FFT bins:

$$P(bin) = |X(bin)|^2, \quad 0 \le bin \le N_{FFT}/2 \tag{5.5}$$

The power spectrum $P(bin)$ is smoothed like:

$$P_{in}(bin) = \frac{P(2 \times bin) + P(2 \times bin + 1)}{2}, \quad 0 \le bin < N_{FFT}/4 \tag{5.6}$$

$$P_{in}(N_{FFT}/4) = P(N_{FFT}/2)$$

By this smoothing operation, the length of the power spectrum is reduced to $N_{SPEC} = N_{FFT}/4 + 1$.

## 5.1.4 Power spectral density mean

This module computes for each power spectrum bin $P_{in}(bin)$ the mean over the last $T_{PSD}$ frames.

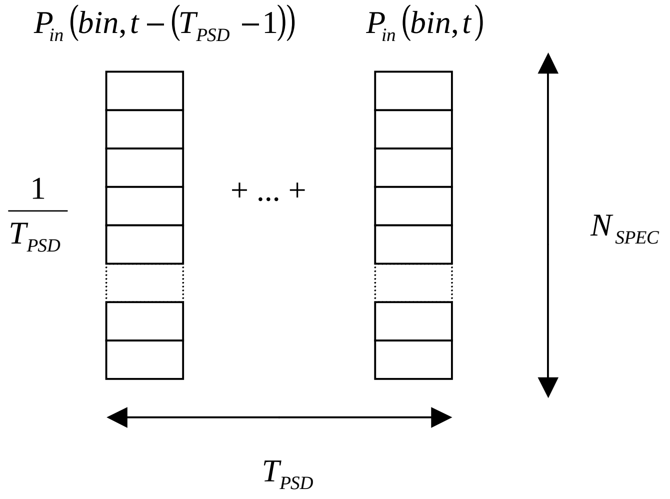


**Figure 5.2: Mean computation over the last $T_{PSD}$ frames as performed in PSD mean**

Power Spectral Density mean (PSD mean) is calculated as:

$$P_{in\_PSD}(bin, t) = \frac{1}{T_{PSD}} \sum_{i=0}^{T_{PSD}-1} P_{in}(bin, t-i), \quad \text{for} \quad 0 \le bin \le N_{SPEC} - 1 \tag{5.7}$$

where the chosen value for $T_{PSD}$ is 2 and *t* is frame (time) index. Note that throughout the present document, we use frame index *t* only if it is necessary for explanation. If the frame index is dropped, current frame is referred.

### 5.1.5 Wiener filter design

A forgetting factor $lambdaNSE$ (used in the update of the noise spectrum estimate in first stage of noise reduction) is computed for each frame depending on the frame time index $t$:

$$\begin{array}{l} if\ (t < NB\_FRAME\_THRESHOLD\_NSE) \\ then \\ \quad lambdaNSE = 1 - 1/t \\ else \\ \quad lambdaNSE = LAMBDA\_NSE \end{array} \tag{5.8}$$

where $NB\_FRAME\_THRESHOLD\_NSE$ equals 100 and $LAMBDA\_NSE$ equals 0,99.

In first stage the noise spectrum estimate is updated according to the following equation, dependent on the $flagVAD_{Nest}$ from VADNest:

$$\begin{cases} P_{noise}^{1/2}(bin,t_n) = max\left(lambdaNSE \times P_{noise}^{1/2}(bin,t_n - 1) + (1 - lambdaNSE) \times P_{in\_PSD}^{1/2}(bin,t_n), EPS\right) \\ P_{noise}^{1/2}(bin,t) = P_{noise}^{1/2}(bin,t_n) \end{cases} \tag{5.9}$$

where $EPS$ equals $\exp(-10{,}0)$, $t$ represents the current frame index, $t_n$ represents the index of the last non-speech frame and $P_{in\_PSD}(bin,t)$ is the output of the PSD Mean module. $P_{noise}^{1/2}(bin,-1)$ is initialized to $EPS$.

In the second stage the noise spectrum estimate is updated permanently according to the following equation:

$$\begin{array}{l} if\,(t < 11) \\ then \\ \qquad lambdaNSE = 1 - 1/t \\ \qquad P_{noise}(bin,t) = lambdaNSE \times P_{noise}(bin,t-1) + (1 - lambdaNSE) \times P_{in\_PSD}(bin,t) \\ else \\ \qquad upDate = 0{,}9 + 0{,}1 \times P_{in\_PSD}(bin,t) / \left(P_{in\_PSD}(bin,t) + P_{noise}(bin,t-1)\right) \\ \qquad\qquad \times \left(1 + 1/\left(1 + 0{,}1 \times P_{in\_PSD}(bin,t) / P_{noise}(bin,t-1)\right)\right) \\ \qquad P_{noise}(bin,t) = P_{noise}(bin,t-1) \times upDate \\ if\left(P_{noise}^{1/2}(bin,t) < EPS\right) \\ then \\ \qquad P_{noise}^{1/2}(bin,t) = EPS \end{array} \tag{5.10}$$

Then the noiseless signal spectrum is estimated using a "decision-directed" approach:

$$P_{den}^{1/2}(bin,t) = BETA \times P_{den3}^{1/2}(bin,t-1) + (1 - BETA) \times T\left[P_{in\_PSD}^{1/2}(bin,t) - P_{noise}^{1/2}(bin,t)\right] \tag{5.11}$$

$P_{den}^{1/2}(bin,-1)$ is initialized to 0, $BETA$ equals 0,98 and the threshold function $T$ is given by:

$$T\left[z(bin,t)\right] = \begin{cases} z(bin,t) & \text{if } \ z(bin,t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

Then the a priori SNR $\eta(bin,t)$ is computed as:

$$\eta(bin,t) = \frac{P_{den}(bin,t)}{P_{noise}(bin,t)} \tag{5.13}$$

The filter transfer function $H(bin,t)$ is obtained according to the following equation:

$$H(bin,t) = \frac{\sqrt{\eta(bin,t)}}{1+\sqrt{\eta(bin,t)}} \tag{5.14}$$

The filter transfer function $H(bin,t)$ is used to improve the estimation of the noiseless signal spectrum:

$$P_{den2}^{1/2}\left(bin,t\right) = H\left(bin,t\right)P_{in\_PSD}^{1/2}\left(bin,t\right) \tag{5.15}$$

Then an improved a priori SNR $\eta_2(bin,t)$ is obtained:

$$\eta_2(bin,t) = \max\left(\frac{P_{den2}\left(bin,t\right)}{P_{noise}\left(bin,t\right)}, \eta_{TH}{}^{2}\right) \tag{5.16}$$

where $\eta_{TH}$ equals 0,079 432 823 (value corresponding to a SNR of -22 dB).

The improved transfer function $H_2(bin,t)$ is then obtained according to the following equation:

$$H_2(bin,t) = \frac{\sqrt{\eta_2(bin,t)}}{1+\sqrt{\eta_2(bin,t)}},\ 0 \le bin \le N_{SPEC} - 1 \tag{5.17}$$

The improved transfer function $H_2(bin,t)$ is then used to calculate the noiseless signal spectrum $P_{den3}^{1/2}\left(bin,t\right)$ that will be used for the next frame in Equation (5.11):

$$P_{den3}^{1/2}\left(bin,t\right) = H_2\left(bin,t\right)P_{in}^{1/2}\left(bin,t\right) \tag{5.18}$$

## 5.1.6 VAD for noise estimation (VADNest)

A forgetting factor $lambdaLTE$ (used in the update of the long-term energy) is computed for each frame using the frame time index $t$:

```
if $\left(t < NB\_FRAME\_THRESHOLD\_LTE\right)$
then
    $lambdaLTE = 1 - 1/t$                                  (5.19)
else
    $lambdaLTE = LAMBDA\_LTE$
```

where $NB\_FRAME\_THRESHOLD\_LTE$ equals 10 and $LAMBDA\_LTE$ equals 0,97.

Then the logarithmic energy $frameEn$ of the M (M = 80) last samples of the input signal $s_{in}\left(n\right)$ is computed:

$$frameEn = 0,5 + \frac{16}{\ln 2} \times \ln\left(\frac{\left(64 + \sum_{i=0}^{M-1} s_{in}(n)^2\right)}{64}\right) \tag{5.20}$$

Then $frameEn$ is used in the update of $meanEn$:

```
if $\left( \begin{array}{l} ((frameEn - meanEn) < SNR\_THRESHOLD\_UPD\_LTE) \\ OR \\ (t < MIN\_FRAME) \end{array} \right)$
then
    if $((frameEn < meanEN) OR (t < MIN\_FRAME))$                                (5.21)
    then
        $meanEn = meanEn + (1 - lambdaLTE) \times (frameEn - meanEn)$
    else
        $meanEn = meanEn + (1 - lambdaLTEhigherE) \times (frameEn - meanEn)$
    if $(meanEn < ENERGY\_FLOOR)$ then $meanEn = ENERGY\_FLOOR$
```

where $SNR\_THRESHOLD\_UPD\_LTE$ equals 20, $ENERGY\_FLOOR$ equals 80, $MIN\_FRAME$ equals 10 and $lambdaLTEhigherE$ equals 0,99.

Then $frameEn$ and $meanEn$ are used to decide if the current frame is speech ( $flagVAD_{Nest} = 1$) or not ( $flagVAD_{Nest} = 0$ ):

```
if $(t > 4)$
then
    if $((frameEn - meanEn) > SNR\_THRESHOLD\_VAD)$
    then
        $flagVAD_{Nest} = 1$
        $nbSpeechFrame = nbSpeechFrame + 1$
    else
        if $(nbSpeechFrame > MIN\_SPEECH\_FRAME\_HANGOVER)$
        then
            $hangOver = HANGOVER$
        $nbSpeechFrame = 0$
        if $(hangOver != 0)$
        then
            $hangOver = hangOver - 1$
            $flagVAD_{Nest} = 1$
        else                                                                    (5.22)
            $flagVAD_{Nest} = 0$
```

where $SNR\_THRESHOLD\_VAD$ equals 15, $MIN\_SPEECH\_FRAME\_HANGOVER$ equals 4 and $HANGOVER$ equals 15.

$nbSpeechFrame$, $meanEn$, $flagVAD_{Nest}$ and $hangOver$ are initialized to 0. The frame time index $t$ is initialised to 0 and is incremented each frame by 1 so that it equals 1 for the first frame processed.

### 5.1.7 Mel filter-bank

The linear-frequency Wiener filter coefficients $H_2(bin),\ 0 \le bin \le N_{SPEC}-1,$ (computed by formula (5.17)) are smoothed and transformed to the Mel-frequency scale. Mel-warped Wiener filter coefficients $H_{2\_mel}(k)$ are estimated by using triangular-shaped, half-overlapped frequency windows applied on $H_2(bin)$. To obtain the central frequencies of FB bands in terms of FFT bin indices, $bin_{centr}(k)$, the linear frequency scale $f_{lin}$ was transformed to mel scale by using the following formula:

$$MEL\{f_{lin}\} = 2\,595 \times \log_{10}(1 + f_{lin}/700) \qquad (5.23)$$

Then, the central frequency of the *k*-th band, $f_{centr}(k)$, is calculated as:

$$f_{centr}(k) = 700 \times \left(10^{\frac{f_{mel}(k)}{2595}} - 1\right), \quad \text{for } 1 \le k \le K_{FB} \qquad (5.24)$$

with $K_{FB}$ = 23 and

$$f_{mel}(k) = k \times \frac{MEL\{f_{lin\_samp}/2\}}{K_{FB}+1} \qquad (5.25)$$

where $f_{lin\_samp} = 8\,000$ is the sampling frequency. Additionally, two marginal FB bands with central frequencies $f_{centr}(0) = 0$ and $f_{centr}(K_{FB}+1) = f_{lin\_samp}/2$ are added to the $K_{FB}$ = 23 Mel FB bands for purposes of following DCT transformation to the time domain; thus, in total we calculate $K_{FB}$ + 2 = 25 Mel-warped Wiener filter coefficients. The FFT bin index corresponding to central frequencies is obtained as:

$$bin_{centr}(k) = round\left(\frac{f_{centr}(k)}{f_{lin\_samp}} \times 2 \times (N_{SPEC}-1)\right) \qquad (5.26)$$

Frequency windows $W(k,i)$ for $1 \le k \le K_{FB}$ are calculated as:

$$W(k,i) = \frac{i - bin_{centr}(k-1)}{bin_{centr}(k) - bin_{centr}(k-1)}, \quad \text{for} \quad bin_{centr}(k-1)+1 \le i \le bin_{centr}(k) \qquad (5.27a)$$

$$W(k,i) = 1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k)}, \quad \text{for} \quad bin_{centr}(k)+1 \le i \le bin_{centr}(k+1) \qquad (5.27b)$$

and $W(k,i) = 0$ for other *i*. For $k = 0$

$$W(0,i) = 1 - \frac{i}{bin_{centr}(1) - bin_{centr}(0)}, \quad \text{for} \quad 0 \le i \le bin_{centr}(1) - bin_{centr}(0) - 1 \qquad (5.27c)$$

and $W(0,i) = 0$ for other *i*. For $k = K_{FB} + 1$

$$W(K_{FB}+1,i) = \frac{i - bin_{centr}(K_{FB})}{bin_{centr}(K_{FB}+1) - bin_{centr}(K_{FB})}, \quad \text{for } bin_{centr}(K_{FB})+1 \le i \le bin_{centr}(K_{FB}+1) \qquad (5.27d)$$

and $W(K_{FB}+1,i)=0$ for other *i*. Mel-warped Wiener filter coefficients $H_{2\_mel}(k)$ for $0 \le k \le K_{FB}+1$ are computed as:

$$H_{2\_mel}(k) = \frac{1}{\sum_{i=0}^{N_{SPEC}-1} W(k,i)} \sum_{i=0}^{N_{SPEC}-1} W(k,i) \times H_2(i) \qquad (5.28)$$

### 5.1.8 Gain factorization

In this block, factorization of the Wiener filter Mel-warped coefficients (or gains), $H_{2\_mel}(k)$, is performed to control the aggression of noise reduction in the second stage.

In the first stage, de-noised frame signal energy $E_{den}(t)$, where *t* is frame index starting with 1, is calculated by using the de-noised power spectrum $P_{den3}(bin,t)$ computed by (5.18) as:

$nbSpeechFrame$ , $meanEn$ , $flagVAD_{Nest}$ and $hangOver$ are initialized to 0.

$$E_{den}(t) = \sum_{bin=0}^{N_{SPEC}-1} P_{den3}^{1/2}(bin,t) \tag{5.29}$$

In the second stage, the noise energy at the current frame index *t* is estimated by using the noise power spectrum $P_{noise}(bin,t)$ as:

$$E_{noise}(t) = \sum_{bin=0}^{N_{SPEC}-1} P_{noise}^{1/2}(bin,t) \tag{5.30}$$

Then, smoothed SNR is evaluated by using three de-noised frame energies (notice there is two frames delay between the first and the second stage) and noise energy like:

```
$Ratio = \frac{E_{den}(t-2) \times E_{den}(t-1) \times E_{den}(t)}{E_{noise}(t) \times E_{noise}(t) \times E_{noise}(t)}$
$if\,(Ratio > 0{,}0001)$
$then$                                                              (5.31)
        $SNR_{aver}(t) = 20/3 \times \log_{10}(Ratio)$
$else$
        $SNR_{aver}(t) = -100/3$
```

To decide the degree of aggression of the second stage Wiener filter for each frame, the low SNR level is tracked by using the following logic:

```
$if\ \{(SNR_{aver}(t) - SNR_{low\_track}(t-1)) < 10 \quad \text{or} \quad t < 10\}$
        calculate $\lambda_{SNR}(t)$
        $SNR_{low\_track}(t) = \lambda_{SNR}(t) \times SNR_{low\_track}(t-1) + (1 - \lambda_{SNR}(t)) \times SNR_{aver}(t)$      (5.32)
$else$
        $SNR_{low\_track}(t) = SNR_{low\_track}(t-1)$
```

with $SNR_{low\_track}$ initialized to zero. The forgetting factor $\lambda_{SNR}(t)$ is calculated by the following logic:

```
$if\ \{t < 10\}$
    $\lambda_{SNR}(t) = 1 - 1/t$
$else$
    $if\ \{SNR_{aver}(t) < SNR_{low\_track}(t)\}$                       (5.33)
        $\lambda_{SNR}(t) = 0{,}95$
    $else$
        $\lambda_{SNR}(t) = 0{,}99$
```

The intention of gain factorization is to apply more aggressive noise reduction to purely noisy frames and less aggressive noise reduction to frames also containing speech. At this point, the current SNR estimation, $SNR_{aver}(t)$ is compared to the low SNR tracked value, $SNR_{low\_track}(t)$, and the Wiener filter gain factorization coefficient $\alpha_{GF}(t)$ is updated. This is done by the following logic:

```
$if\,(E_{den}(t) > 100)$
$then$
        $if\ \ \{SNR_{aver}(t) < (SNR_{low\_track}(t) + 3{,}5)\}$
                $\alpha_{GF}(t) = \alpha_{GF}(t-1) + 0{,}15$
                $if\ \ \{\alpha_{GF}(t) > 0{,}8\}$
                        $\alpha_{GF}(t) = 0{,}8$                    (5.34)
        $else$
                $\alpha_{GF}(t) = \alpha_{GF}(t-1) - 0{,}3$
                $if\ \ \{\alpha_{GF}(t) < 0{,}1\}$
                        $\alpha_{GF}(t) = 0{,}1$
```

with $\alpha_{GF}(0) = 0{,}8$.

The second stage Wiener filter gains are multiplied by $\alpha_{GF}(t)$ like:

$$H_{2\_mel\_GF}(k,t) = (1 - \alpha_{GF}(t)) + \alpha_{GF}(t) \times H_{2\_mel}(k,t), \quad 0 \le k \le K_{FB} + 1 \tag{5.35}$$

The coefficient $\alpha_{GF}(t)$ takes values from 0,1 to 0,8, which means that the aggression of the second stage Wiener filter is reduced to 10 % for speech + noise frames and to 80 % for noise frames.

### 5.1.9 Mel IDCT

The time-domain impulse response of Wiener filter $h_{WF}(n)$ is computed from the Mel Wiener filter coefficients $H_{2\_mel}(k)$ from clause 5.1.6 (in the second stage, $H_{2\_mel\_GF}(k)$ from equation (5.35)) by using Mel-warped inverse DCT:

$$h_{WF}(n) = \sum_{k=0}^{K_{FB}+1} H_{2\_mel}(k) \times IDCT_{mel}(k,n), \qquad 0 \le n \le K_{FB} + 1 \tag{5.36}$$

where $IDCT_{mel}(k,n)$ are Mel-warped inverse DCT basis computed as follows.

First, central frequencies of each band are computed for $1 \le k \le K_{FB}$ like:

$$f_{centr}(k) = \frac{1}{\sum_{i=0}^{N_{SPEC}-1} W(k,i)} \sum_{i=0}^{N_{SPEC}-1} W(k,i) \times i \times \frac{f_{samp}}{2 \times (N_{SPEC} - 1)} \tag{5.37}$$

where $f_{samp} = 8\,000$ is sampling frequency. $f_{centr}(0) = 0$ and $f_{centr}(K_{FB} + 1) = f_{samp} / 2$. Then, Mel-warped inverse DCT basis are obtained as:

$$IDCT_{mel}(k,n) = \cos\left(\frac{2 \times \pi \times n \times f_{centr}(k)}{f_{samp}}\right) \times df(k), \;\; 0 \le k \le K_{FB} + 1, \; 0 \le n \le K_{FB} + 1 \tag{5.38}$$

where $f_{centr}(k)$ is central frequency corresponding to the Mel FB index $k$ and $df(k)$ is computed like:

$$df(k)=\frac{f_{centr}(k+1)-f_{centr}(k-1)}{f_{samp}},\quad 1\le k\le K_{FB} \tag{5.39}$$

$$df(0)=\frac{f_{centr}(1)-f_{centr}(0)}{f_{samp}}\quad \text{and}\quad df(K_{FB}+1)=\frac{f_{centr}(K_{FB}+1)-f_{centr}(K_{FB})}{f_{samp}}$$

The impulse response of Wiener filter is mirrored as:

$$h_{WF\_mirr}(n)=\begin{cases} h_{WF}(n), & 0\le n\le K_{FB}+1\\ h_{WF}(2\times(K_{FB}+1)+1-n), & K_{FB}+2\le n\le 2\times(K_{FB}+1)\end{cases} \tag{5.40}$$

## 5.1.10 Apply filter

The causal impulse response $h_{WF\_caus}(n,t)$ is obtained from $h_{WF\_mirr}(n,t)$ according to the following relations:

$$\begin{cases} h_{WF\_caus}(n,t)=h_{WF\_mirr}(n+K_{FB}+1,t), & n=0,\cdots,K_{FB}\\ h_{WF\_caus}(n,t)=h_{WF\_mirr}(n-K_{FB}-1,t), & n=K_{FB}+1,\cdots,2\times(K_{FB}+1)\end{cases} \tag{5.41}$$

The causal impulse response $h_{WF\_caus}(n,t)$ is then truncated giving $h_{WF\_trunc}(n,t)$:

$$h_{WF\_trunc}(n,t)=h_{WF\_caus}\left(n+K_{FB}+1-(FL-1)/2,t\right),\quad n=0,\cdots,FL-1 \tag{5.42}$$

where the filter length $FL$ equals 17.

The truncated impulse response is weighted by a Hanning window:

$$h_{WF\_w}(n,t)=\left\{0,5-0,5\times\cos\left(\frac{2\times\pi\times(n+0,5)}{FL}\right)\right\}\times h_{WF\_trunc}(n,t),\ 0\le n\le FL-1 \tag{5.43}$$

Then the input signal $s_{in}$ is filtered with the filter impulse response $h_{WF\_w}(n,t)$ to produce the noise-reduced signal $s_{nr}$:

$$s_{nr}(n)=\sum_{i=-(FL-1)/2}^{(FL-1)/2} h_{WF\_w}\left(i+(FL-1)/2\right)\times s_{in}(n-i),\ 0\le n\le M-1 \tag{5.44}$$

where the filter length $FL$ equals 17 and the frame shift interval $M$ equals 80.

## 5.1.11 Offset compensation

To remove the DC offset, a notch filtering operation is applied to the noise-reduced signal like:

$$s_{nr\_of}(n)=s_{nr}(n)-s_{nr}(n-1)+(1-1/1024)\times s_{nr\_of}(n-1),\quad 0\le n\le M-1 \tag{5.45}$$

where $s_{nr}(-1)$ and $s_{nr\_of}(-1)$ correspond to the last samples of the previous frame and equal 0 for the first frame, and $M=80$ is the frame shift interval.

## 5.2 Waveform Processing

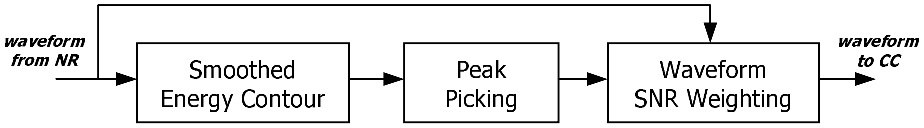


**Figure 5.3: Main components of SNR-dependent waveform processing**

SNR-dependent Waveform Processing (SWP) is applied to the noise reduced waveform that comes out from the Noise Reduction (NR) block. The noise reduction block outputs 80-sample frames that are stored in a 240-sample buffer (from sample 0 to sample 239). The waveform processing block is applied on the window that starts at sample 1 and ends at sample 200. Figure 5.3 describes the basic components of SWP. In the Smoothed Energy Contour block, the instant energy contour is computed for each input frame by using the Teager operator like:

$$E_{Teag}(n) = \left|s^2_{nr\_of}(n) - s_{nr\_of}(n-1) \times s_{nr\_of}(n+1)\right|, \quad 1 \le n < N_{in} - 1 \qquad (5.46a)$$

$$E_{Teag}(0) = \left|s^2_{nr\_of}(0) - s_{nr\_of}(0) \times s_{nr\_of}(1)\right| \qquad (5.46b)$$

and

$$E_{Teag}(N_{in}-1) = \left|s^2_{nr\_of}(N_{in}-1) - s_{nr\_of}(N_{in}-2) \times s_{nr\_of}(N_{in}-1)\right| \qquad (5.46c)$$

The energy contour is smoothed by using a simple FIR filter of length 9 like:

$$E_{Teag\_Smooth}(n) = \frac{1}{9}\sum_{i=-4}^{4} E_{Teag}(n+i) \qquad (5.47)$$

At the beginning or ending edge of $E_{Teag}(n)$, the $E_{Teag}(0)$ or $E_{Teag}(N_{in}\text{-}1)$ value is repeated, respectively.

In the Peak Picking block, maxima in the smoothed energy contour related to the fundamental frequency are found. First, the global maximum over the entire energy contour $E_{Teag\_Smooth}(n),\ 0 \le n \le N_{in} - 1$, is found. Then, maxima on both left and right sides of the global maximum are identified. Each maximum is expected to be between 25 and 80 samples away from its neighbour.

In the block Waveform SNR Weighting, a weighting function is applied to the input frame. Having the number of maxima $N_{MAX}$ of the smoothed energy contour $E_{Taeg\_Smooth}(n)$ and their positions $pos_{MAX}(n_{MAX}),\ 0 \le n_{MAX} < N_{MAX}$, a weighting function $w_{swp}(n)$ of length $N_{in}$ is constructed, which equals 1,0 for $n$ from intervals:

$$\left\langle [pos_{MAX}(n_{MAX}) - 4],\ [pos_{MAX}(n_{MAX}) - 4] + 0{,}8 \times [pos_{MAX}(n_{MAX}+1) - pos_{MAX}(n_{MAX})] \right\rangle,\ 0 \le n_{MAX} < N_{MAX}$$

and equals 0 otherwise. At the transitions (from 0,0 to 1,0 or from 1,0 to 0,0), the $w_{swp}(n)$ function has value 0,5.

Finally, the following weighting is applied to the input noise-reduced frame:

$$s_{swp}(n) = 1{,}2 \times w_{swp}(n) \times s_{nr\_of}(n) + 0{,}8 \times (1 - w_{swp}(n)) \times s_{nr\_of}(n), \quad 0 \le n \le N_{in} - 1 \qquad (5.48)$$

# 5.3 Cepstrum Calculation

This block performs cepstrum calculation. Cepstrum calculation is applied on the signal that comes out from the waveform processing block. Figure 5.4 shows main components of the Cepstrum Calculation block.

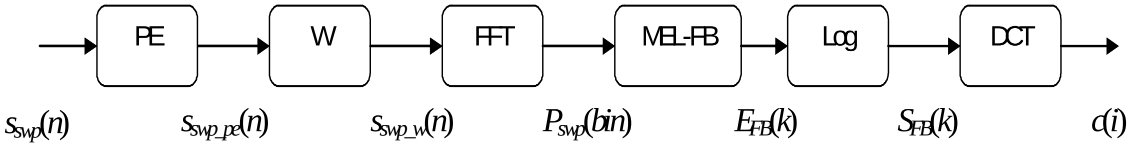


**Figure 5.4: Main components of the cepstrum calculation block**

## 5.3.1 Log energy calculation

For each frame, a log energy parameter is calculated from the de-noised signal as:

$$lnE = \begin{cases} \ln\left(E_{swp}\right) & \text{if } E_{swp} \geq E_{THRESH} \\ \ln\left(E_{THRESH}\right) & \text{otherwise} \end{cases} \tag{5.49a}$$

where $E_{THRESH}$ = exp(-50) and $E_{swp}$ is computed as:

$$E_{swp} = \sum_{n=0}^{N_{in}-1} s_{swp}(n) \times s_{swp}(n) \tag{5.49b}$$

## 5.3.2 Pre-emphasis (PE)

A pre-emphasis filter is applied to the output of the waveform processing block $s_{swp}(n)$ like:

$$s_{swp\_pe}(n) = s_{swp}(n) - 0{,}9 \times s_{swp}(n-1) \tag{5.50}$$

where $s_{swp\_of}(-1)$ is the last sample from the previous frame and equals 0 for the first frame.

## 5.3.3 Windowing (W)

A Hamming window of length $N_{in}$ =200 is applied to the output of the pre-emphasis block:

$$s_{swp\_w}(n) = \left[0{,}54 - 0{,}46 \times \cos\left(\frac{2\pi \times (n+0{,}5)}{N_{in}}\right)\right] \times s_{swp\_pe}(n), \quad 0 \leq n \leq N_{in} - 1 \tag{5.51}$$

## 5.3.4 Fourier transform (FFT) and power spectrum estimation

Each frame of $N_{in}$ samples is zero padded to form an extended frame of 256 samples. An FFT of length $N_{FFT}$ = 256 is applied to compute the complex spectrum $X_{swp}(bin)$ of the de-noised signal:

$$X_{swp}(bin) = FFT\left\{s_{swp\_w}(n)\right\} \tag{5.52}$$

Corresponding power spectrum $P_{swp}(bin)$ is calculated as:

$$P_{swp}(bin) = \left|X_{swp}(bin)\right|^2, \quad 0 \leq bin \leq N_{FFT}/2 \tag{5.53}$$

## 5.3.5 Mel Filtering (MEL-FB)

**Purpose**

The leading idea of the MEL-FB module is to recombine the information contained in the frequency-dependent representation (FFT) by regrouping it in a Mel-band representation.

The FFT-bins are linearly recombined for each Mel-band. The useful frequency band lies between $f_{start}$ and $f_{samp}$ / 2. This band is divided into $K_{FB}$ channels equidistant in the Mel frequency domain. Each channel has a triangular-shaped frequency window. Consecutive channels are half-overlapping.

**Frequencies and index**

In the FFT calculation, index value $bin = N_{FFT}$ corresponds to the frequency $f_{samp}$. The formula that accounts for the index calculation of frequencies is then:

$$index\{f\} = round\left\{\frac{f}{f_{samp}} \times N_{FFT}\right\} \tag{5.54}$$

where $round\{\cdot\}$ stands for rounding towards the nearest integer.

**Mel-function**

The Mel-function is the operator which rescales the frequency domain.

$$Mel\{x\} = \Lambda \times \log_{10}\left(1+\frac{x}{\mu}\right) = \lambda \times \ln\left(1+\frac{x}{\mu}\right), \quad \text{with} \quad \lambda = \frac{\Lambda}{\ln(10)} \tag{5.55a}$$

The inverse Mel-function is:

$$Mel^{-1}\{y\} = \mu \times \left(\exp\left(\frac{y}{\lambda}\right) - 1\right) \tag{5.55b}$$

**Central frequencies of the filters**

The central frequencies of the filters are calculated from the Mel-function, in order to have an equidistant distribution of the bands in the Mel domain.

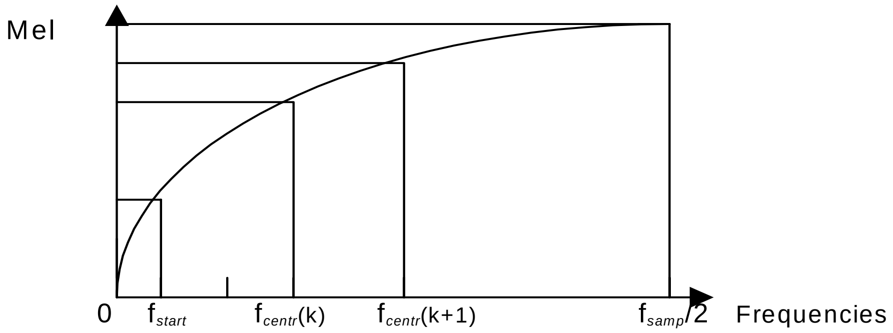


**Figure 5.5: Linear to Mel frequency mapping**

$$f_{centr}(k) = Mel^{-1}\left\{Mel\{f_{start}\} + k \times \frac{Mel\{f_{samp}/2\} - Mel\{f_{start}\}}{K_{FB}+1}\right\}, \quad 1 \le k \le K_{FB} \tag{5.56}$$

In our proposal, parameters are chosen as follows:

$$f_{start} = 64\ Hz, \quad f_{samp} = 8\ kHz$$
$$\mu = 700, \quad \Lambda = 2\,595, \quad \lambda = 1127$$
$$K_{FB} = 23$$

In terms of FFT index, the central frequencies of the filters correspond to:

$$bin_{centr}(k) = index\{f_{centr}(k)\} = round\left\{\frac{f_{centr}(k)}{f_{samp}} \times N_{FFT}\right\}, \quad 1 \le k \le K_{FB} \tag{5.57}$$

For the *k*-th Mel-band, the frequency window is divided into two parts . The former part (i.e. frequencies $f_{centr}(k-1) < f < f_{centr}(k)$) accounts for increasing weights, whereas the latter part (i.e. frequencies $f_{centr}(k) < f < f_{centr}(k+1)$) accounts for decreasing weights. Each frequency window is applied to the de-noised power spectrum $P_{swp}(bin)$ computed by (5.53). Frequency window weights for each band are calculated depending on the position of each frequency bin with respect to the corresponding band central frequency like:

if the bin *i* is from $bin_{centr}(k-1) \le i \le bin_{centr}(k)$, then:

$$W_{left}(i,k) = \frac{i - bin_{centr}(k-1) + 1}{bin_{centr}(k) - bin_{centr}(k-1) + 1}, \qquad \text{for } 1 \le k \le K_{FB} \tag{5.58}$$

if the bin *i* is from $bin_{centr}(k) < i \le bin_{centr}(k+1)$, then:

$$W_{right}(i,k) = 1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k) + 1}, \qquad \text{for } 1 \le k \le K_{FB} \tag{5.59}$$

For other situations, weights equal zero.

**Output of *MEL-FB***

The output of each Mel filter is the weighted sum of the de-noised power spectrum values $P_{swp}(bin)$ from equation (5.53) in each band. Triangular, half-overlapped windowing is used as follows:

$$E_{FB}(k) = \sum_{i=bin_{centr}(k-1)}^{bin_{centr}(k)} W_{left}(i,k) \times P_{swp}(i) + \sum_{i=bin_{centr}(k)+1}^{bin_{centr}(k+1)} W_{right}(i,k) \times P_{swp}(i), \quad \text{for } 1 \le k \le K_{FB} \tag{5.60}$$

### 5.3.6 Non-linear transformation (Log)

The output of Mel filtering is subjected to a logarithm function (natural logarithm).

$$S_{FB}(k) = \ln(E_{FB}(k)), \quad \text{for } 1 \le k \le K_{FB} \tag{5.61}$$

A flooring is applied in such a way that the log filter bank outputs cannot be smaller than -10.

### 5.3.7 Cepstral coefficients (DCT)

13 cepstral coefficients are calculated from the output of the Non-linear transformation block by applying a DCT.

$$c(i) = \sum_{k=1}^{K_{FB}} S_{FB}(k) \times \cos\left(\frac{i \times \pi}{K_{FB}} \times (k - 0{,}5)\right), \quad 0 \le i \le 12 \tag{5.62}$$

Notice that in the case of 16 kHz input signal, number of FB bands $K_{FB}$ is increased by 3 (see clause 5.5 for more details).

### 5.3.8 Cepstrum calculation output

The final feature vector consists in 14 coefficients: the log-energy coefficient *lnE* and the 13 cepstral coefficients *c*(0) to *c*(12).

The $c(0)$ coefficient is often redundant when the log-energy coefficient is used. However, the feature extraction algorithm is defined here for both energy and $c(0)$. Depending on the application, either the coefficient $c(0)$, or the log-energy coefficient, or a combination of $c(0)$ and $lnE$ may be used.

## 5.4 Blind equalization

12 cepstral coefficients ($c(1),\ldots,c(12)$) are equalized according to the following LMS algorithm:

$$weightingPar = Min\left(1, Max\left(0, \ln E - 211/64\right)\right), \tag{5.63}$$

$$stepSize = 0,008\,789\,062\,5 \times weightingPar, \tag{5.64}$$

$$c_{eq}(i) = c(i) - bias(i),\ 1 \le i \le 12 \tag{5.65}$$

$$bias(i) += stepSize \times \left(c_{eq}(i) - RefCep(i)\right),\ 1 \le i \le 12 \tag{5.66}$$

where *lnE* is the log energy of the current frame as computed by (5.49a) and the values of $bias(i)$ and $RefCep(i)$ at the initialization stage are the following:

$$bias(i) = 0,0,\quad 1 \le i \le 12,$$

$$\begin{aligned}
&RefCep(1) = -6,618\,909,\ RefCep(2) = \ \ 0,198\,269,\ RefCep(3) = -0,740\,308\\
&RefCep(4) = \ \ 0,055\,132,\ RefCep(5) = -0,227\,086,\ RefCep(6) = \ \ 0,144\,280,\\
&RefCep(7) = -0,112\,451,\ RefCep(8) = -0,146\,940,\ RefCep(9) = -0,327\,466,\\
&RefCep(10) = \ \ 0,134\,571,\ RefCep(11) = \ \ 0,027\,884,\ RefCep(12) = -0,114\,905,
\end{aligned} \tag{5.67}$$

The reference cepstrum corresponds to the cepstrum of a flat spectrum.

## 5.5 Extension to 11 kHz and 16 kHz sampling frequencies

For the 11 kHz sampling frequency, we perform downsampling from 11 kHz to 8 kHz and all front-end processing is the same as in the case of the 8 kHz sampling frequency.

For the 16 kHz sampling frequency, we extended the 8 kHz front-end as shown on figure 5.6. In this approach, the 8 kHz feature extraction part processes the signal from the Low-Frequency Band (LFB, 0 kHz to 4 kHz) and it is re-used without significant changes. The signal from the High Frequency Band (HFB, 4 kHz to 8 kHz) is processed separately and the high-frequency information is added to the low-frequency information just before transforming the log FB energies to cepstral coefficients. Additionally, the whole-band log energy parameter *lnE* is also computed by using both the low-frequency and high-frequency information.

### 5.5.1 FFT-based spectrum estimation

As it can be observed from figure 5.6, the input signal, $s_{in\_16}(n)$, is first filtered by a couple of Quadrature-Mirror Filters (QMF), $h_{LFB\_QMF}(n)$ and $h_{HFB\_QMF}(n)$, to get both the LFB and HFB signal portions:

$$s_{LFB}(n) = s_{in\_16}(n) \times h_{LFB\_QMF}(n),\ s_{HFB}(n) = s_{in\_16}(n) \times h_{HFB\_QMF}(n) \tag{5.68}$$

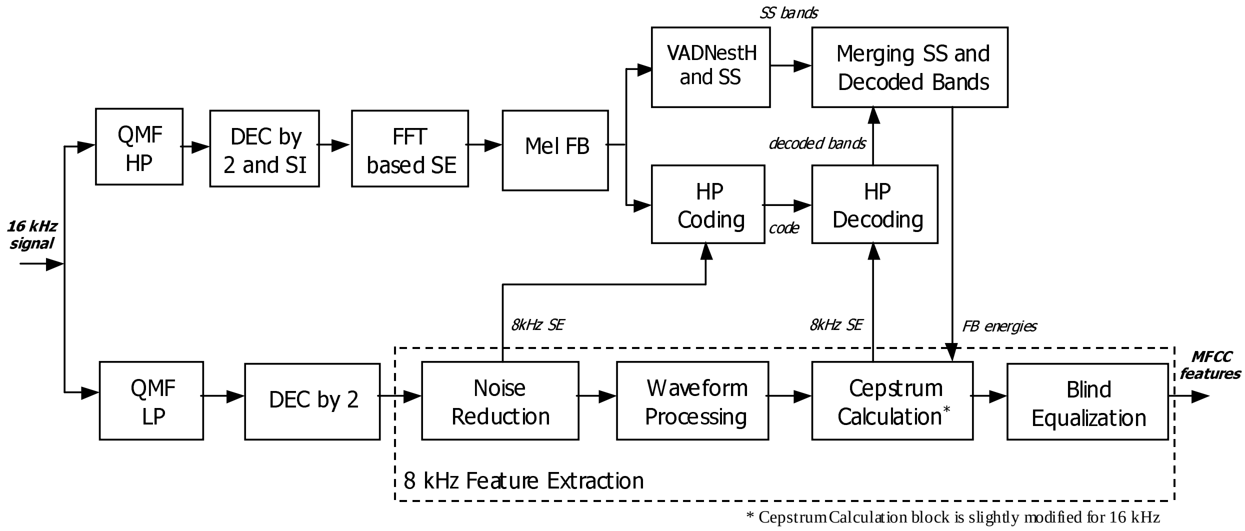


**Figure 5.6: Extension of 8 kHz front-end for 16 kHz sampling frequency**

The LFB QMF is a Finite Impulse Response (FIR) filter of length 118 from the ITU-T standard software tools library for downsampling. The HFB QMF is an FIR filter obtained from the LFB QMF by multiplying each sample of its impulse response by $(-1)^n$, where $n$ is sample index. Both LFB and HFB signals are decimated by factor 2 by choosing only every second sample from the corresponding filtered signal. Additionally, the HFB signal is frequency-inverted (spectrum inversion, SI on figure 5.6) by multiplying the HFB signal sequence by the sequence $(-1)^n$, where $n$ is the sample index. The LFB signal enters the Noise Reduction part of Feature Extraction and it is processed up to the cepstral coefficient computation in the same way as in the case of 8 kHz sampling frequency.

By downsampling and spectral-inversion, the HFB signal is shifted to the frequency range 0 kHz to 4 kHz. This shifted HFB signal $s_{SI\_HFB}(n)$ is further processed on frame-by-frame basis, where the frame length and frame shift are synchronized with the LFB processing and are the same as in the case of 8 kHz input signal (i.e. 25ms/10ms). Each frame of length $N_{in}$ = 200 is windowed by a Hamming window:

$$s_{W\_HFB}(n) = s_{SI\_HFB}(n) \times w_{Hamm}(n), \quad 0 \le n \le N_{in} - 1 \tag{5.69}$$

and zeros are padded from the sample $N_{in}$ up to the sample $N_{FFT}$ -1, where $N_{FFT}$ = 256 is the FFT length:

$$s_{W\_HFB\_FFT}(n) = \begin{cases} s_{W\_HFB}(n), & 0 \le n \le N_{in} - 1 \\ 0, & N_{in} \le n \le N_{FFT} - 1 \end{cases} \tag{5.70}$$

A smoothed HFB power spectrum, $P_{Smooth\_HFB}(bin)$, is estimated by using an FFT followed by power of 2 like:

$$X_{HFB}(bin) = FFT\{s_{W\_HFB\_FFT}(n)\} \tag{5.71}$$

$$P_{HFB}(bin) = |X_{HFB}(bin)|^2, \quad 0 \le bin \le N_{FFT}/2 \tag{5.72}$$

$$P_{Smooth\_HFB}(bin) = \frac{P_{HFB}(2 \times bin) + P_{HFB}(2 \times bin + 1)}{2}, \quad 0 \le bin < N_{FFT}/4 \tag{5.73}$$

$$P_{Smooth\_HFB}(N_{FFT}/4) = P_{HFB}(N_{FFT}/2)$$

By the smoothing operation, the length of the power spectrum is reduced to $N_{SPEC} = N_{FFT}/4 + 1$

### 5.5.2 Mel Filter-Bank

The entire high-frequency band is divided into $K_{HFB}$ = 3 Filter-Bank (FB) bands, which are equidistantly distributed in the Mel-frequency domain. Energies within the FB bands, $E_{HFB}(k)$, are estimated by using triangular-shaped, half-overlapped frequency windows applied on the HFB power spectrum. To obtain the central frequencies of FB bands in terms of FFT bin indices, $bin_{centr}(k)$, we used the following relationship between the linear and mel frequency scales:

$$f_{mel} = MEL\{f_{lin}\} = 2\,595 \times \log_{10}(1 + f_{lin}/700) \tag{5.74}$$

Then, the central frequency of the $k$-th band, $f_{centr}(k)$, is calculated as:

$$f_{centr}(k) = 700 \times \left(10^{\frac{f_{mel}(k)}{2\,595}} - 1\right), \quad 1 \le k \le K_{HFB} \tag{5.75}$$

with:

$$f_{mel}(k) = MEL\{f_{lin\_start}\} + k \times \frac{MEL\{f_{lin\_samp}/2\} - MEL\{f_{lin\_start}\}}{K_{HFB} + 1} \tag{5.76}$$

where $f_{lin\_start} = 80$ is the starting frequency and $f_{lin\_samp} = 8\,000$ is the sampling frequency. The corresponding FFT bin index is obtained as:

$$bin_{centr}(k) = round\left(\frac{f_{centr}(k)}{f_{lin\_samp}} \times 2 \times N_{SPEC}\right) \tag{5.77}$$

Having the central frequencies, $bin_{centr}(k)$, the energy within the $k$-th FB band, $E_{HFB}(k)$, is computed as:

$$\begin{aligned} E_{HFB}(k) = & \sum_{i=bin_{centr}(k-1)+1}^{bin_{centr}(k)} \frac{i - bin_{centr}(k-1)}{bin_{centr}(k) - bin_{centr}(k-1)} \times P_{Smooth\_HFB}(i) + \\ & + \sum_{i=bin_{centr}(k)+1}^{bin_{centr}(k+1)} \left(1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k)}\right) \times P_{Smooth\_HFB}(i) \end{aligned} \tag{5.78}$$

where $1 \le k \le K_{HFB}$, $bin_{centr}(0)$ and $bin_{centr}(K_{HFB}+1)$ are the FFT indices corresponding to the starting frequency $f_{lin\_start}$, and half of the sampling frequency $f_{lin\_samp}/2$.

### 5.5.3 High-frequency band coding and decoding

Before coding, the natural logarithm is applied to the HFB mel FB energies $E_{HFB}(k)$ as:

$$S_{HFB}(k) = \ln(E_{HFB}(k)), \quad 1 \le k \le K_{HFB} \tag{5.79}$$

with a flooring avoiding values of $S_{HFB}(k)$ lower than -10. The HFB log FB energies, $S_{HFB}(k)$, are coded and decoded by using three auxiliary bands computed from 2 kHz to 4 kHz frequency interval of LFB power spectrum. For coding, the auxiliary bands are calculated before applying both Noise Reduction (NR) and waveform processing (SWP) to the LFB signal. For decoding, the auxiliary bands are calculated after applying both NR and SWP to the LFB signal. Auxiliary bands are approximately logarithmically spaced in the given frequency interval.

The three auxiliary log FB energies for coding are computed from the input signal power spectrum $P_{in}(bin)$, $0 \le bin < N_{SPEC}$, calculated in the first stage of Noise Reduction block (see equation (5.6) in clause 5.1.2) as:

$$S_{LFB\_aux}(1) = \ln\left(\sum_{bin=33}^{38} P_{in}(bin)\right), \; S_{LFB\_aux}(2) = \ln\left(\sum_{bin=39}^{48} P_{in}(bin)\right) \text{ and} \tag{5.80}$$

$$S_{LFB\_aux}(3) = \ln\left(\sum_{bin=49}^{64} P_{in}(bin)\right)$$

with flooring that avoids values of $S_{LFB\_aux}(k)$ lower than -10. Then, coding is performed as:

$$Code(k,l) = S_{LFB\_aux}(k) - S_{HFB}(l), \quad 1 \le k,l \le K_{HFB} \tag{5.81}$$

The three auxiliary bands for decoding are computed from the de-noised power spectrum $P_{swp}(bin)$, $0 \le bin \le N_{FFT}/2$, calculated in the Cepstrum Calculation block (see clause 5.3.5) as:

$$S_{swp\_LFB\_aux}(1) = \ln\left(\frac{1}{2}\sum_{bin=66}^{76} P_{swp}(bin)\right), \; S_{swp\_LFB\_aux}(2) = \ln\left(\frac{1}{2}\sum_{bin=77}^{96} P_{swp}(bin)\right), \text{ and} \tag{5.82}$$

$$S_{swp\_LFB\_aux}(3) = \ln\left(\frac{1}{2}\sum_{bin=97}^{128} P_{swp}(bin)\right)$$

with flooring that avoids values of $S_{swp\_LFB\_aux}(k)$ lower than -10. The decoded HFB bands, $S_{code\_HFB}(k)$, are obtained by using the code $Code(k,l)$ and the three de-noised auxiliary LFB log FB energies $S_{swp\_LFB\_aux}(k)$ like:

$$S_{code\_HFB}(k) = \sum_{l=1}^{K_{HFB}} w_{code}(l)\left(S_{swp\_LFB\_aux}(l) - Code(l,k)\right), \quad 1 \le k \le K_{HFB} \tag{5.83}$$

where $w_{code}(l)$ is a frequency-dependent weighting with:

$$\sum_{l=1}^{K_{HFB}} w_{code}(l) = 1 \tag{5.84}$$

In the current implementation, frequency weights are $w_{code}(1) = 0{,}1, w_{code}(2) = 0{,}2, w_{code}(3) = 0{,}7$

### 5.5.4 VAD for noise estimation and spectral subtraction in high-frequency bands

A simple, energy-based Voice Activity Detector for Noise estimation (VADNestH) is designed for noise estimation in the HFB signal. A forgetting factor for a) updating the noise estimation and b) tracking the low log energy level is computed for each frame *t* according to the logic:

```
if  $\{t < 100\}$
    $\lambda_{NSE}(t) = 1 - 1/t$
else                                                        (5.85)
    $\lambda_{NSE}(t) = 0{,}99$
```

The low log energy level is tracked by using the following logic:

```
if  $\{[(E_{\log}(t) - E_{\log\_low\_track}(t-1)) < 1{,}2]$ or $[t < 10]\}$
    if  $\{t < 10\}$
        $E_{\log\_low\_track}(t) = \lambda_{NSE}(t) \times E_{\log\_low\_track}(t-1) + (1 - \lambda_{NSE}(t)) \times E_{\log}(t)$
    else                                                    (5.86)
        if  $\{E_{\log}(t) < E_{\log\_low\_track}(t-1)\}$
            $E_{\log\_low\_track}(t) = 0{,}98 \times E_{\log\_low\_track}(t-1) + (1 - 0{,}98) \times E_{\log}(t)$
        else
            $E_{\log\_low\_track}(t) = 0{,}995 \times E_{\log\_low\_track}(t-1) + (1 - 0{,}995) \times E_{\log}(t)$
```

where $E_{\log\_low\_track}$ is initialized to 0 and the log energy $E_{\log}(t)$ is computed like:

$$E(t) = \sum_{k=1}^{K_{HFB}} E_{HFB}(t,k) \qquad (5.87a)$$

$$E_{\log}(t) = \begin{cases} \ln(E(t)) & \text{for } E(t) > 0{,}001 \\ \ln(0{,}001) & \text{for } E(t) \le 0{,}001 \end{cases} \qquad (5.87b)$$

VADNestH flag $flagVAD_{NestH}(t)$ is updated by using the current frame log energy $E_{\log}(t)$ and the low log energy level $E_{\log\_low\_track}(t)$ as follows:

```
if  $\{(E_{\log}(t) - E_{\log\_low\_track}(t)) > 2{,}2\}$
    $flagVAD_{NestH}(t) = 1$
    $nbSpeechFrame(t) = nbSpeechFrame(t-1) + 1$
else
    if  $\{nbSpeechFrame(t-1) > 4\}$
        $hangOver(t) = 5$
    $nbSpeechFrame(t) = 0$                                   (5.88)
    if  $\{hangOver(t)\ != 0\}$
        $hangOver(t+1) = hangOver(t) - 1$
        $flagVAD_{NestH}(t) = 1$
    else
        $flagVAD_{NestH}(t) = 0$
```

VADNestH flag is used for estimating the HFB noise spectrum in terms of FB energies like:

$$\begin{array}{l} if \ \ \{flagVAD_{NestH}(t) = 0\} \\ \quad \hat{N}_{HFB}(k,t) = \lambda_{NSE}(t) \times E_{HFB}(k,t) + (1 - \lambda_{NSE}(t)) \times \hat{N}_{HFB}(k,t-1), \quad 1 \le k \le K_{HFB}, \end{array} \qquad (5.89)$$

where *t* is the frame index and the noise FB energy vector is initialized to a zero vector.

Spectral subtraction is performed like:

$$E_{SS\_HFB}(k) = \max\{E_{HFB}(k) - \alpha \times \hat{N}_{HFB}(k),\ \beta \times E_{HFB}(k)\}, \quad 1 \le k \le K_{HFB} \qquad (5.90)$$

where $\alpha$ = 1,5 and $\beta$ = 0,1 were set empirically.

### 5.5.5 Merging spectral subtraction bands with decoded bands

In the Cepstrum Calculation block, log FB energies from both LFB and HFB are joined and cepstral coefficients representing the entire frequency band are calculated. It is obvious that the noise reduction performed on the LFB signal is more complex than the Spectral Subtraction (SS) algorithm applied on HFB FB bands, and thus FB energies resulting from these two processes are not entirely compatible. To reduce the differences between the FB energies from the HFB and LFB, the SS HFB log FB energies are used in combination with the HFB log FB energies resulting from the coding scheme described in clause 5.5.3.

First, rough pre-emphasis correction and log non-linearity are applied on HFB energies resulting from spectral subtraction like:

$$S_{SS\_HFB}(k) = \ln((1 + a_{pre}) \times E_{SS\_HFB}(k)) \quad 1 \le k \le K_{FB} \qquad (5.91)$$

where $a_{pre} = 0{,}9$ is pre-emphasis constant. The HFB log FB energies $S_{HFB}(k)$ are then obtained by combining both $S_{SS\_HFB}(k)$ and $S_{code\_HFB}(k)$, like:

$$S_{HFB}(k) = \lambda_{merge} \times S_{code\_HFB}(k) + (1 - \lambda_{merge}) \times S_{SS\_HFB}(k), \quad 1 \le k \le K_{HFB} \qquad (5.92)$$

where $\lambda_{merge} = 0{,}7$ is an empirically set constant.

For each frame, a cepstrum is calculated from a vector of log FB energies that is formed by appending the three HFB log FB energies to the LFB log FB energies. Before joining the LFB and HFB log FB energies, the transition between the last LFB band $S_{FB}(K_{FB})$ (computed as in clause 5.3.7) and the first HFB $S_{HFB}(1)$ is smoothed by modifying the two transition log energies like:

$$S'_{FB}(K_{FB}) = 0{,}6 \times S_{FB}(K_{FB}) + 0{,}4 \times S_{aver} \qquad (5.93a)$$

and

$$S'_{HFB}(1) = 0{,}6 \times S_{HFB}(1) + 0{,}4 \times S_{aver} \qquad (5.93b)$$

where

$$S_{aver} = \frac{S_{FB}(K_{FB}) + S_{HFB}(1)}{2} \qquad (5.93c)$$

Finally, the log FB energy vector for cepstrum calculation $S_{cep}(k),\ 1 \le k \le K_{FB} + K_{HFB}$, is formed like:

$$S_{cep}(k) = \{S_{FB}(1), S_{FB}(2), \ldots, S_{FB}(K_{FB} - 1), S'_{FB}(K_{FB}), S'_{HFB}(1), S_{HFB}(2), S_{HFB}(3)\} \qquad (5.94)$$

## 5.5.6 Log energy calculation for 16 kHz

Log energy parameter is computed by using information from both the LFB and HFB. We used the HFB log FB energies, $S_{HFB}(k)$, to modify the log energy parameter. First, we computed the HFB energy $E_{HFB}$ by using pre-emphasis corrected, de-noised HFB log FB energies like:

$$E_{HFB} = \sum_{k=1}^{K_{HFB}} \exp(S_{HFB}(k) - preem\_corr) \qquad (5.95)$$

where:

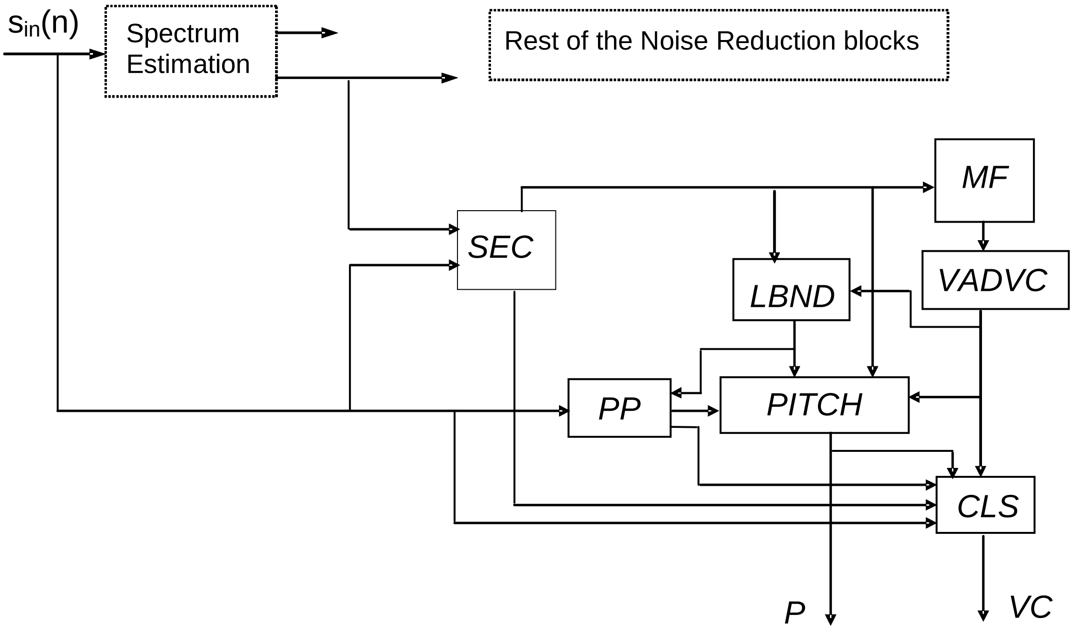
$$preem\_corr = \ln(1 + a_{pre}) \qquad (5.96)$$

and $a_{pre}$ = 0,9 is the pre-emphasis constant. Then, the energy parameter is computed as the natural logarithm of the sum of the de-noised LFB energy $E_{swp}$ and the de-noised HFB energy $E_{HFB}$:

$$lnE = \ln(E_{swp} + E_{HFB}) \qquad (5.97)$$

## 5.6 Pitch and class estimation

As indicated in figure 4.1, estimation of pitch and voicing class parameters is embedded inside the noise reduction block (clause 5.1). A block diagram for pitch and class estimation is shown in figure 5.7. The "spectrum estimation" block at the top-left corner of figure 5.7 represents the block with the same name in figure 5.1. The input to this block, viz., $s_{in}(n)$, and one of the outputs from this block, viz., $X(bin)$ (Eq. 5.4), form the inputs to the estimation of the Pitch (P) and Voicing Class (VC) parameters.



| | |
|---|---|
| CLS | CLaSsification |
| LBND | Low-Band Noise Detection |
| MF | Mel-Filtering |
| PITCH | PITCH estimation |
| PP | Pre-Processing for pitch and class estimation |
| SEC | Spectrum and Energy Computation |
| VADVC | Voice Activity Detection for Voicing Classification |

**Figure 5.7: Block scheme for pitch and class estimation**

### 5.6.1 Spectrum and energy computation

The input to the SEC block is the input speech $s_{in}(n)$ and X(bin), bin = 0, 1,…, $N_{FFT}$ -1, where X(bin) represents the complex short-time Fourier Transform of $s_{in}(n)$. As a first step, X(0) is set to 0 to remove any DC offset. Then, the following quantities are computed: power spectrum *pbin*, pre-emphasized power spectrum $pbin_{pe}$, frame energy *E*, logarithm of frame energy log*E*, and average spectral value $s_w(1)$.

The power spectrum is computed as

$$pbin_k = \mathrm{Re}(X(k))^2 + \mathrm{Im}(X(k))^2$$

The pre-emphasized power spectrum is computed as:

$$pbin_{pe,k} = pbin_k \times \left( \left|1 - 0{,}97 \times \cos(k\pi/128)\right|^2 + \left|\sin(k\pi/128)\right|^2 \right)$$

The frame energy is computed as:

$$E = \sum_{n=0}^{N-1} s_{in}^2(n) - \frac{1}{N}\left(\sum_{n=0}^{N-1} s_{in}(n)\right)^2$$

The log-energy is computed as log*E* = log(*E*). A floor is used in the energy calculation, which makes sure that the result for log*E* is not less than -50. The floor value for *E* (lower limit for the argument of ln) is approximately 2e-22.

The average spectral value is computed as:

$$s_w(1) = \frac{1}{N_{FFT}} \sum_{k=0}^{N_{FFT}-1} X(k)$$

The power spectrum *pbin* is fed into the *MF* block, mel-filtered as described in clause 5.3.5, and the mel-filter outputs $fbank_i$, $i$ = 1, …, 23 are fed into the *VADVC* block. The pre-emphasized power spectrum $pbin_{pe}$ is fed into the *LBND* block. The frame energy *E* is fed into the *LBND* block and the *CLS* block. The short-time Fourier transform *X*(*bin*), the power spectrum *pbin*, the log-energy log*E*, and the average spectral value $S_w(1)$ are fed into the *PITCH* block. Furthermore, the input speech signal $S_{in}$(n) is fed into the *PP* block and the *CLS* block.

## 5.6.2 Voice Activity Detection for Voicing Classification (VADVC)

The input to the Voice Activity Detection (VAD) block is the mel-filter output $fbank_i$, $i$ = 1, …, 23. The outputs of the *VAD* block are the *vad_flag* and *hangover_flag*. The *vad_flag*, if TRUE, indicates that the current frame is a speech frame. The *hangover_flag*, if TRUE, indicates that the current frame is likely to be a speech frame because it follows a speech segment. The operation of the VAD block is described below with reference to figure 5.8.

In the following, we denote the mel-filter output for the $m^{th}$ frame and $i^{th}$ channel by *F*(*m*,*i*), and when the specific channel is not important, the mel-filter output for the $m^{th}$ frame by *F*(*m*). Using these values as input, the channel energy estimator provides a smoothed estimate of the channel energies as follows:

$$E_{ch}(m,i) = \max\{E_{\min}, \alpha_{ch}(m)E_{ch}(m-1,i) + (1-\alpha_{ch}(m))(\lambda_i F(m,i))\};\ i = 1, 2, \ldots, 23 \qquad (5.98)$$

where $E_{ch}(m,i)$ is the smoothed channel energy estimate for the $m^{th}$ frame and the $i^{th}$ channel, $E_{min}$ is the minimum allowable channel energy, $\{\lambda_i$, $i$ = 1, 2, …, 23} are the correction factors to compensate for the effect of the pre-emphasis filter and the varying widths of the triangular weighting windows used in mel-filtering, and $\alpha_{ch}(m)$ is the channel energy smoothing factor defined as:

$$\alpha_{ch}(m) = \begin{cases} 0{,}00; & m = 1 \\ 0{,}45; & m > 1 \end{cases} \qquad (5.99)$$

The minimum channel energy $E_{min}$ is 5 000 for 8 kHz, 6 400 for 11 kHz, and 10 000 for 16 kHz sampling frequency respectively. he value of the correction factor $\lambda_i$ is given by the $i^{th}$ value in the 23-element table: {0,3333, 0,3333, 0,2857, 0,2857, 0,2857, 0,2500, 0,2500, 0,2222, 0,2000, 0,2000, 0,2000, 0,1818, 0,1667, 0,1538, 0,1429, 0,1429, 0,1333, 0,1176, 0,1111, 0,1111, 0,1000, 0,0909, 0,0870}.

From the channel energy estimate, the peak-to-average ratio for the current frame *m*, denoted by *P2A*(*m*) is estimated at the peak-to-average ratio estimator as follows:

$$P2A(m) = 10\log_{10}\left(\frac{\max(E_{ch}(m,i)\big|_{i=5}^{23})}{(1/23)\sum_{i=1}^{23} E_{ch}(m,i)}\right) \qquad (5.100)$$

Similar to the channel energy estimate, the channel noise energy estimate (defined below) is initialized as follows:

```
if (($m \le$ INIT_FRAMES) OR (fupdate_flag == TRUE))
{
    if (P2A(m) < PEAK_TO_AVE_THLD)
    {
$E_n(m,i) = \begin{cases} E_{ch}(m,i);\ m = 1,\ 1 \le i \le 23; \\ 0{,}7E_n(m-1,i) + 0{,}3E_{ch}(m,i);\ 2 \le m \le INIT\_FRAMES,\ 1 \le i \le 23; \end{cases}$
    }
    else
    {
  $E_n(m,i) = E_{\min};1 \le i \le 23;$
    }
}
```

(5.101)

where $E_n(m,i)$ is the smoothed noise energy estimate for the $m^{th}$ frame and the $i^{th}$ channel, *INIT_FRAMES* is the number of initial frames which are assumed to be noise-only frames, and *fupdate_flag* is the forced update flag defined later. The value of *INIT_FRAMES* = 10, and that of *PEAK_TO_AVE_THLD* = 10.0. Initially, *fupdate_flag* is set to FALSE.

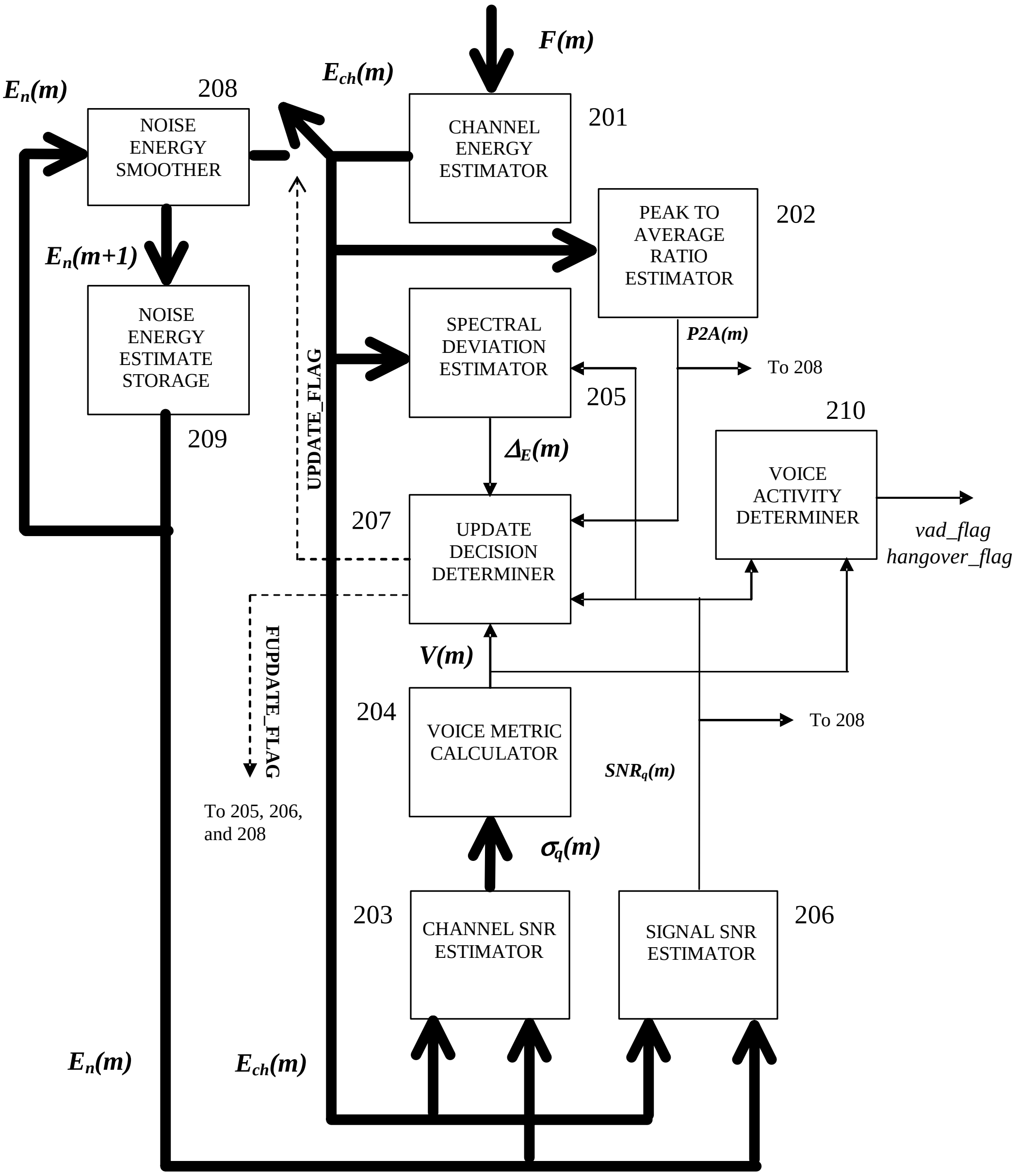


**Figure 5.8: Block diagram of the voice activity detection (VADVC) algorithm**

The channel energy estimate $E_{ch}(m)$ and the channel noise energy estimate $E_n(m)$ are used to estimate the quantized channel signal-to-noise ratio (SNR) indices at the channel SNR estimator as:

$$\sigma_q(m,i) = \max(0, \min(89, \mathrm{round}(10\log_{10}\left(\frac{E_{ch}(m,i)}{E_n(m,i)}\right) \Big/ 0{,}375)));1 \le i \le 23 \qquad (5.102)$$

where the values $\{\sigma_q(m, i), i = 1, 2, \ldots, 23\}$are constrained to be between 0 and 89 both inclusive.

From the channel SNR estimate $\sigma_q(m)$ for the current frame, the voice metric $V(m)$ for the current frame is computed at the voice metric calculator as the sum:

$$V(m) = \sum_{i=1}^{23} v(\sigma_q(i)) \tag{5.103}$$

where $v(k)$ is the $k^{\text{th}}$ value of the 90-element voice metric table $v$ defined as: $v$ = {1,1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4, 5,5,5,6,6,7,7,7,8,8,9,9,10,10,11,12,12,13,13,14,15,15,16,17,17,18,19,20,20,21,22,23,24,24,25,26,27,28,28,29,30,31,32, 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}.

The channel energy estimate $E_{ch}(m)$ is also used as input to the spectral deviation estimator, which estimates the spectral deviation $\Delta_E(m)$ for the current frame as follows. First, the log energy spectrum is estimated as:

$$E_{dB}(m,i) = 10\log_{10}(E_{ch}(m,i))\ ;\ i = 1, 2, \ldots, 23 \tag{5.104}$$

Next, the spectral deviation $\Delta_E(m)$ is estimated as the sum of the absolute difference between the current log energy spectrum and an average long-term log energy spectrum denoted by $\overline{E}_{dB}(m)$, that is:

$$\Delta_E(m) = \sum_{i=1}^{23} \left| E_{dB}(m,i) - \overline{E}_{dB}(m,i) \right| \tag{5.105}$$

The average long-term log energy spectrum is initialized as follows:

```
if ((m ≤ INIT_FRAMES) OR (fupdate_flag == TRUE))
```

$$\overline{E}_{dB}(m,i) = E_{dB}(m,i);\ 1 \le i \le 23 \tag{5.106}$$

The average long-term log energy spectrum is updated as follows:

$$\overline{E}_{dB}(m+1,i) = \begin{cases} 0{,}9\overline{E}_{dB}(m,i) + 0{,}1E_{dB}(m,i);\ V(m) > SIG\_THLD(m) \\ 0{,}7\overline{E}_{dB}(m,i) + 0{,}3E_{dB}(m,i);\ V(m) \le SIG\_THLD(m) \end{cases} \tag{5.107}$$

where the parameter *SIG_THLD*($m$) depends on the quantized signal SNR described next. The initial value of *SIG_THLD* is 217.

The speech signal SNR is estimated at the signal SNR estimator as follows. First, the total noise energy of the current frame $E_{tn}(m)$ is computed as the sum of the channel noise energies, that is:

$$E_{tn}(m) = \sum_{i=1}^{23} E_n(m,i) \tag{5.108}$$

Next, the instantaneous total signal energy $E_{ts,inst}(m)$ is computed as follows:

```
if (V(m) > SIG_THLD(m))
    $E_{ts,inst}(m) = \sum_{i=1}^{23} \max(E_{ch}(m,i), E_n(m,i))$;
else
    $E_{ts,inst}(m) = E_{tn}(m)$;
end                                                    (5.109)
```

Initialization of $E_{ts,inst}(m)$ is performed as follows:

```
if ((m ≤ INIT_FRAMES) OR (fupdate_flag == TRUE))
```

$$E_{ts,inst}(m) = INIT\_SIG\_ENRG; \qquad (5.110)$$

where the value of *INIT_SIG_ENRG* = 1,0E+09 for 8 kHz, 1,67E+09 for 11 kHz, and 3,0E+09 for 16 kHz respectively.

Once the total instantaneous signal energy and the total noise energy are computed, the instantaneous signal-to-noise ratio of the current frame denoted by $SNR_{inst}(m)$ is computed as:

$$SNR_{inst} = \max(0{,}0,\ 10 \log_{10}(E_{ts,inst}(m) / E_{tn}(m))) \qquad (5.111)$$

From the instantaneous SNR, the smoothed SNR is estimated as:

```
if (($m \le INIT\_FRAMES$) OR ($fupdate\_flag$ == TRUE))
    $SNR(m) = SNR_{inst}(m)$;
else
{
    if ($V(m) > SIG\_THLD(m)$)
    {
        $SNR(m) = \beta\ SNR(m-1) + (1-\beta)\ SNR_{inst}(m)$;
        $\beta = \min(\beta+0.003, HI\_BETA)$;
    }
    else
        $\beta = \max(\beta-0.003, LO\_BETA)$;
}                                                        (5.112)
```

The lower and upper limits of the smoothing factor $\beta$ are respectively *LO_BETA* = 0,950 and *HI_BETA* = 0,998. Initially, the value of $\beta$ is set at *LO_BETA*. The signal SNR is then quantized to 20 different values as:

$$SNR_q(m) = \max(0,\min(\mathrm{round}(SNR(m)/1{,}5),19)). \qquad (5.113)$$

The quantized signal SNR is used to determine different threshold values. For example, the signal threshold for the next frame $SIG\_THLD(m+1)$ is determined using $SNR_q(m)$ as an index into the 20-element table {36, 43, 52, 62, 73, 86, 101, 117, 134, 153, 173, 194, 217, 242, 268, 295, 295, 295, 295, 295}.

At this point, the voice metric $V(m)$, the spectral deviation $\Delta_E(m)$, the peak-to-average ratio $P2A(m)$, and the quantized signal SNR $SNR_q(m)$ are input to an update decision determiner. The logic shown below in pseudo-code demonstrates how the noise estimate update decision is made and also how a forced update decision is made (a forced update mechanism allows the voice activity detector to recover from wrong classification of background noise as speech whenever there is a sudden increase in background noise level).

First, the update threshold for the current *frame* $UPDATE\_THLD(m)$ is determined using $SNR_q(m)$ as an index into a 20-element table given by {31, 32, 33, 34, 35, 36, 37, 37, 37, 37, 37, 37, 37, 37, 37, 38, 38, 38, 38, 38}. The update decision determination process begins by clearing the update flag (*update_flag*) and the forced update flag (*fupdate_flag*). These flags are set if certain conditions are satisfied as illustrated by the pseudo-code below. The initial value of *update_cnt* is set to 0.

```
update_flag = FALSE;
fupdate_flag = FALSE;
if ((m > INIT_FRAMES) AND (V(m) < UPDATE_THLD(m)) AND
    (P2A(m) < PEAK_TO_AVE_THLD)
{
    update_flag = TRUE;
    update_cnt = 0;
}
else
{
    if ((P2A(m) < PEAK_TO_AVE_THLD) AND ($\Delta_E$(m) < DEV_THLD))
    {
        update_cnt = update_cnt + 1;
        if (update_cnt ≥ UPDATE_CNT_THLD)
        {
            update_flag = TRUE;
            fupdate_flag = TRUE;
        }
    }
}                                                        (5.114)
```

In order to avoid long term "creeping" of the update counter (*update_cnt*) setting the forced update flag (fupdate_flag) falsely in the above pseudo-code, an hysteresis logic is implemented as shown below. Initial values of *last_update_cnt* and *hyster_cnt* are set to 0.

```
    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;                    (5.115)
```

The values of different constants used above are as follows: *DEV_THLD* = 70, *UPDATE_CNT_THLD* = 500, and *HYSTER_CNT_THLD* = 9. Whenever the above referenced update flag is set for a given frame, the channel noise estimate for the next frame is updated in the noise energy smoother as follows:

$$E_n(m+1,i) = 0{,}9E_n(m,i) + 0{,}1E_{ch}(m,i))\,;\ i = 1, 2, \ldots, 23 \qquad (5.116)$$

The updated channel noise estimate is stored in noise energy estimate storage for all future frames until the next update occurs. The output of the noise energy estimate storage $E_n(m)$ is used as an input to the channel SNR estimator as described earlier.

Next, we describe the operation of the voice activity determiner, which uses the voice metric $V(m)$ and the quantized signal SNR value $SNR_q(m)$ as inputs. For the first *INIT_FRAMES* frames, the outputs of the voice activity determiner, viz., *vad_flag* and *hangover_flag* are set to FALSE since these frames are assumed to be noise-only frames. For the following frames, the voice activity determiner operates by testing if the voice metric exceeds the voice metric threshold $V_{th}$. If the output of this test is TRUE, then the current frame is declared "voice-active". Otherwise, the hangover count variable (*hangover_count*) is tested to find out if it is greater than or equal to zero. If the output of this test is TRUE, then also the current frame is declared "voice-active". If the outputs of both tests are FALSE, then the current frame is declared "voice-inactive". The "hangover" mechanism is generally used to cover slowly decaying speech that might otherwise be classified as noise, and to bridge over small gaps or pauses in speech. It is activated if the number of consecutive "voice-active" frames (counted by the *burst_count* variable) is at least equal to $B_{cnt}$, the burst count threshold. To activate the mechanism, the number of hangover frames is set to $H_{cnt}$, the hangover count threshold. The pseudo-code for the voice activity determiner is shown below. To begin with, the voice metric threshold $V_{th}$, the hangover count threshold $H_{cnt}$, and the burst count threshold $B_{cnt}$ are initialized to 56, 28 and 6 respectively. Furthermore, the variables hangover_count and burst_count are both initialized to 0.

```
if (V(m) > Vth(m))
    {
        vad_local = TRUE;
        burst_count = burst_count + 1;
        if (burst_count >= Bcnt(m))
            hangover_count = Hcnt(m);
    }
    else
    {
        vad_local = FALSE:
        burst_count = 0;
    }

if ((vad_local == TRUE) OR (hangover_count > 0))
        vad_flag = TRUE;
    else
        vad_flag = FALSE;

if ((vad_local == FALSE) && (hangover_count > 0))
{
        hangover_flag = TRUE;
        hangover_count = hangover_count - 1;
    }
    else
                                      hangover_flag = FALSE;             (5.117)
```

As a final step, the quantized SNR value is used to determine the voice metric threshold $V_{th}$, the hangover count threshold $H_{cnt}$, and the burst count threshold $B_{cnt}$ for the next frame as:

$$V_{th}(m+1) = V_{table}[SNR_q(m)],\ H_{cnt}(m+1) = H_{table}[SNR_q(m)],\ B_{cnt}(m+1) = B_{table}[SNR_q(m)], \quad (5.118)$$

where $SNR_q(m)$ is used as an index into the respective tables. These tables are defined by: $V_{table}$ = {32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 55, 56, 57, 57, 58, 58, 58, 58}, $H_{table}$ = {54, 52, 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16}, and $B_{table}$ = {2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6}.

## 5.6.3 Low-band noise detection

In the scope of clause 5.6.3, the following symbolic notations for some constants are used if not stated differently in the text:

FFTL = 256 - FFT dimension;

$f_s$ = 8 - sampling rate of the input speech data in kHz.

The input to the low-band noise detection (*LBND*) block are the pre-emphasized power spectrum $pbin_{pe,k}$, $k=0,\dots,FFTL/2$, from the *SEC* block, the *vad_flag* and the frame energy $E$. The output of the LBND block is *lbn_flag* indicating (if TRUE) that the current frame contains background noise in the low frequency band.

The *LBND* code maintains an internal state variable *LH_Ratio* which is initialized to 1,9. The operation of the *LBND* block is described by the following pseudo code wherein the *cut_idx* parameter is defined as:

$$cut\_idx = floor\left(380 \times FFTL / \left(1\,000 \times f_s\right)\right) \quad (5.119)$$

```
if (vad_flag == FALSE)
{
    if (2E/FFTL < 500)
        cur_ratio = 0;
    else
    {
low_max = $\max_{1 \le k \le cut\_idx} pbin_{pe,k}$;
high_max = $\max_{cut\_idx < k \le FFTL/2} pbin_{pe,k}$;
    if (high_max == 0)
        cur_ratio = 10;
    else
        cur_ratio = low_max / high_max
    }
LH_Ratio = $0,99 \times LH\_Ratio + 0,01 \times cur\_ratio$;
}

if (LH_Ratio > 1,9)
    lbn_flag = TRUE;
else
                                        lbn_flag = FALSE;                    (5.120)
```

## 5.6.4 Pre-Processing for pitch and class estimation

The input to the Pre-Processing (PP) block is the input signal $s_{in}$ and the *lbn_flag* from the Low-Band Noise Detection (LBND) block. The outputs of the PP block are the low-pass filtered, downsampled speech signal $s_{lpds}$ which is fed into the Pitch estimation block (PITCH) and the high-pass filtered upper-band signal $s_{ub}$ which is fed into the Classification block (CLS). The low-pass and high-pass filtering are performed using pole-zero filters with the generic form shown below:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \dots + b_M x(n-M) - a_1 y(n-1) - a_2 y(n-2) - \dots - a_M y(n-M) \quad (5.121)$$

where $x$ is the input, $y$ is the output, *M* is order of the filter, $b_0, b_1, ..., b_M$ are the coefficients of the numerator polynomial defining the zeros, and $1, a_1, a_2, ..., a_M$ are the coefficients of the denominator polynomial defining the poles. The filter coefficients used are shown in table 5.1. The low-pass filtered speech is first decimated by a factor *DSMP*, where *DSMP* is 4. The latest (2 × *MAX_PITCH* / *DSMP*) samples referred to as the *low-pass filtered extended downsampled frame* is fed into the *PITCH* block. The value of the *MAX_PITCH* parameter is 160.

**Table 5.1: Filter coefficients used in the pre-processing block**

| Filter details | Filter Coefficients |
|---|---|
| low-pass filter numerator coefficients<br>filter order - 7<br>*lbn_flag* = FALSE | 0,0003405377<br>0,0018389033<br>0,0038821292<br>0,0037459142<br>0,0010216130<br>-0,0010216130<br>-0,0008853979<br>-0,0002043226 |
| low-pass filter denominator coefficients;<br>filter order - 7<br>*lbn_flag* = FALSE | 1,00000000<br>-4,47943480<br>8,88015848<br>-10,05821568<br>6,99836861<br>-2,98181953<br>0,71850318<br>-0,07538083 |
| low-pass filter numerator coefficients<br>filter order - 6<br>*lbn_flag* = TRUE | 0,00034054<br>0,00204323<br>0,00510806<br>0,00681075<br>0,00510806<br>0,00204323<br>0,00034054 |
| low-pass filter denominator coefficients<br>filter order - 6<br>*lbn_flag* = TRUE | 1,00000000<br>-3,57943480<br>5,65866717<br>-4,96541523<br>2,52949491<br>-0,70527411<br>0,08375648 |
| high-pass filter numerator coefficients<br>filter order - 6 | 0,14773250<br>-0,88639500<br>2,21598750<br>-2,95464999<br>2,21598749<br>-0,88639500<br>0,14773250 |
| high-pass filter denominator coefficients<br>filter order - 6 | 1,00000000<br>-2,37972104<br>2,91040657<br>-2,05513144<br>0,87792390<br>-0,20986545<br>0,02183157 |

### 5.6.5 Pitch estimation

In the scope of clause 5.6.5, the following symbolic notations for some constants and variables are used if not stated differently in the text:

$FFTL$ = 256 - FFT dimension;

N = 200 - frame size;

$f_s$ = 8 - sampling rate of the input speech data in kHz;

*stft(n) = X(n)* - Short Time Fourier Transform (STFT) spectrum given by (5.4);

*pbin(n) =* $pbin_n$ - power spectrum computed in the *SEC* block.

A flowchart of the pitch estimation process is shown on figure 5.9. Pitch frequency (*F0*) candidates are generated sequentially in high, middle and low frequency intervals (*search ranges*). The candidates generated for a search range are added to the candidates generated earlier and an attempt is made to determine a pitch estimate among the candidates. If the pitch estimate is not determined, the next search range is processed. Otherwise certain internal variables, which represent the pitch estimation history information are updated. At output, the pitch estimate is converted from the frequency to time representation or is set to 0 indicating an unvoiced frame.

### 5.6.5.1 Dirichlet interpolation

Frequency resolution of the discrete complex spectrum in the diapason [0 kHz, 4 kHz] is doubled by the interpolation of the STFT (5.4) by Dirichlet kernel. The interpolated STFT is calculated as follows:

$$\begin{aligned}
&istft(2n) = stft(n) \\
&\mathrm{Re}[istft(2n+1)] = \mathrm{s_w}(1) - \sum_{k=0}^{LDK-1} D(k)\times\{\mathrm{Im}[stft(n-k)] - \mathrm{Im}[stft(n+1+k)]\} \\
&\mathrm{Im}[istft(2n+1)] = \sum_{k=0}^{LDK-1} D(k)\times\{\mathrm{Re}[stft(n-k)] - \mathrm{Re}[stft(n+1+k)]\} \\
&n = 0,1,\ldots,N4\,kHz
\end{aligned} \tag{5.122}$$

where:

*(N4kHz-1)* is the index of the FFT point representing 4kHz frequency;

$$D(k) = \frac{1}{FFTL}\bigg/ tg\left(\frac{\pi}{FFTL}\times(k+0{,}5)\right); \tag{5.123}$$

*LDK= 8*

In (5.122), an *stft(i)* value corresponding to a negative value of *i<0* is replaced by the complex conjugate *stft×(-i)* associated with *-i*.

The number of *istft* samples computed and used further is *FFTIL = 2 × N4 kHz - 1*. The *istf* vector is used for the processing of the current and the next frames.

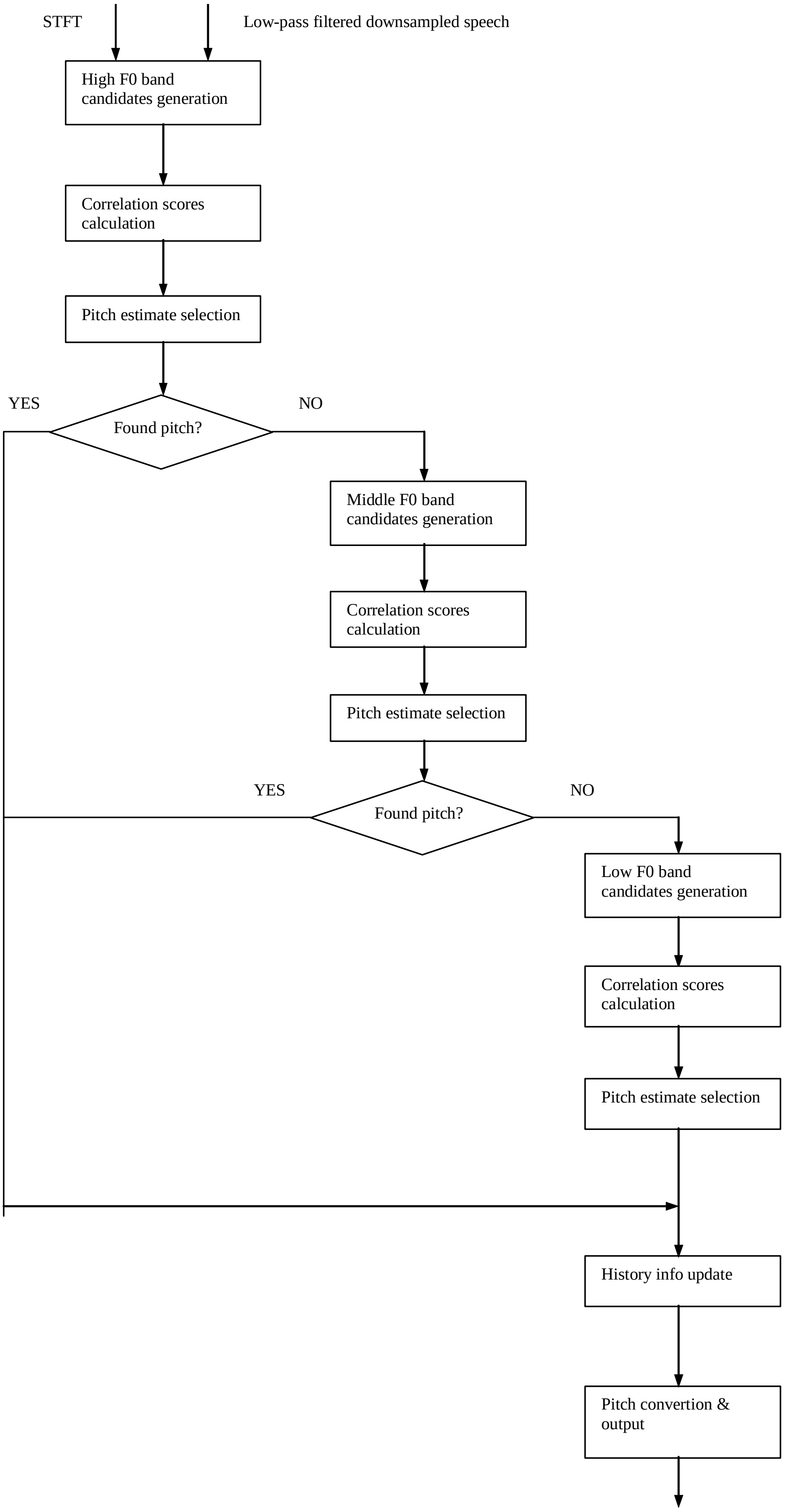

STFT
Low-pass filtered downsampled speech
High F0 band candidates generation
Correlation scores calculation
Pitch estimate selection
YES
NO
Found pitch?
Middle F0 band candidates generation
Correlation scores calculation
Pitch estimate selection
YES
NO
Found pitch?
Low F0 band candidates generation
Correlation scores calculation
Pitch estimate selection
History info update
Pitch convertion & output

**Figure 5.9: Pitch estimation flowchart**

### 5.6.5.2 Non-speech and low-energy frames

If the frame either has been classified by the VADVC block as a non-speech frame or its log-energy value is less than a predefined threshold $log\ E < 13{,}6$ then the pitch frequency F0 estimate is set to 0 and the final step of history information update is performed as described further.

### 5.6.5.3 Search ranges specification and processing

The entire search diapason for pitch frequency is defined as *SR* = [52 Hz, 420 Hz]. If a variable *StableTrackF0* (which is described below) has a non-zero value then *SR* is narrowed as follows:

*SR = SR 3 [0,666 × StableTrackF0, 2,2 × StableTrackF0].*

Three slightly overlapping search ranges are specified:

*SR1 = SR 3 [52 Hz, 120 Hz];*

*SR2 = SR 3 [100 Hz, 210 Hz];*

*SR3 = SR 3 [200 Hz, 420 Hz].*

The processing stages described in clauses 5.6.5.3 to 5.6.5.7 are performed consequently for the three search ranges in the order SR3, SR2, SR1. If there are differences specific to a certain search range they are explained in the relevant clause. It might happen that some of the search ranges are empty. No processing is performed for an empty search range.

### 5.6.5.4 Spectral peaks determination

This stage is performed only twice: first time for the *SR3* and *SR2* ranges, and a second time for *SR1*.

When the processing is being performed for *SR3*/*SR2* search interval, power spectrum with doubled frequency resolution is computed as follows:

$$ps(n) = \begin{cases} pbin_{n/2}, & for\, even\ \ n \\ \mathrm{Re}[istft(n)]^2 + \mathrm{Im}[istft(n)]^2, & for\, odd\ \ n \end{cases} \tag{5.124}$$

When the processing is being performed for SR1 search interval, an STFT corresponding to a double frame is approximated as follows:

$$istft_2(n) = istft(n) + \exp(-j \times \frac{\pi \times n \times M}{FFTIL}) \times istft_{prev}(n) \tag{5.125}$$

where $istft_{prev}$ is the Dirichlet interpolated STFT of the previous frame. Then power spectrum is computed as:

$$ps(n) = \mathrm{Re}[istft_2(n)]^2 + \mathrm{Im}[istft_2(n)]^2 \tag{5.126}$$

In (5.124) to (5.126), n = 0, 1, …, FFTIL - 1 corresponding to the frequency interval [0, 4kHz].

Smoothing by 3-tap symmetric filter is applied to the power spectrum:

$$\begin{aligned} &sps(n) = 0{,}625 \times ps(n) + 0{,}1875 \times [ps(n-1) + ps(n+1)],\ \ n = 1,..., FFTIL - 2 \\ &sps(0) = ps(0),\quad sps(FFTIL - 1) = ps(FFTIL - 1) \end{aligned} \tag{5.127}$$

The values of the smoothed power spectrum *sps(n)* are analysed within the range n∈ [$N_0$+2, FFTIL-3] and all local maxima are determined. $N_0$ is set to $300 \times 2FFTL/(1\,000 \times f_s)$ if low band noise has been detected at that frame. Otherwise $N_0$ = 0. That is, if low band noise is present then the spectral components residing at frequencies lower than 300 Hz are not analysed. A value *sps(n)* is considered as a local maximum if the following condition is TRUE

$$sps(n) > sps(n-1) \wedge sps(n) > sps(n+1) \wedge [sps(n-1) \geq sps(n-2) \vee sps(n+1) \geq sps(n+2)$$

Let $\{(A_k, n_k), k = 1,\ldots,Npeaks\}$ be a list of all the local maxima (representing spectral peaks) sorted in ascending order of their frequencies where $A_k = sps(n_k)$.

**Scaling down of high frequency peaks**

The entire range [0, FFTIL] of the frequency index is divided into three equal sub-intervals, and the maximal values $\mathrm{Amax}_1$, $\mathrm{Amax}_2$ and $\mathrm{Amax}_3$ of $\mathrm{A}_\mathrm{k}$ is found in the low, middle and high sub-intervals correspondingly. The value $Amax_j$ $(j = 2,3)$ is evaluated against a threshold $THR_j = A\max_1 \times \rho_j^{\,2}$. If $Amax_j > THR_j$ then all the $\mathrm{A}_\mathrm{k}$ associated with j-th interval are multiplied by factor $THR_j/Amax_j$ . The following parameter values are used $\rho_2 = 0{,}65,\ \rho_3 = 0{,}45$.

If the number of the peaks (the local maxima) exceeds 30 then the peaks with amplitudes less than $0{,}001^2 \times \max A_k$ are discarded from the peaks list. If the number of remaining peaks is still exceeds 30 then all the high frequency peaks starting from the peak #31 are discarded. The total number $Npeaks$ of the peaks is updated as needed.

The peaks are sorted in descending order of their amplitudes. If the number of peaks is greater than 20 then only 20 first peaks are selected for further processed, and the number $Npeaks$ is set to 20.

Location and amplitude of each peak is refined by fitting parabola through the corresponding local maximum and the two neighbouring samples of the power spectrum $sps$.

$$\begin{aligned}
&loc_k = n_k - 0{,}5 \times b/a \\
&refA_k = sps(n_k + 1) + 0{,}25 \times b \times (loc_k - n_k), \\
&where \\
&\qquad a = sps(n_k - 1) - 2A_k + sps(n_k + 1),\ and \\
&\qquad b = sps(n_k + 1) - sps(n_k - 1)
\end{aligned} \tag{5.128}$$

Then the peak locations $loc_k$ are converted to Hz units and the square roots are taken from the peak amplitudes:

$$\begin{aligned}
PF_k &= loc_k \times 1\,000 \times f_s /(2 \times FFTL) \\
PA_k &= \sqrt{refA_k}
\end{aligned} \tag{5.129}$$

The sequence $\{PA_k, PF_k, k=1,\ldots,Npeaks\}$ represents magnitude spectrum peaks.

Scaling down of high frequency peaks procedure is applied to this peaks sequence as described above except for that this time $\rho_j$ is used for the threshold $\mathrm{THR}_\mathrm{j}$ computation instead of $\rho_j^{\,2}$.

If $Npeaks > 7$ the final attempt to reduce the number of peaks is done as follows. If a number $N1$ exists so that $\sum_{k=1}^{N1} PA_k \le 0{,}95 \times \sum_{k=1}^{Npeaks} PA_k$ then only $N1$ starting peaks are taken. Otherwise the peaks are scanned from the end of the list towards the beginning and all the peaks with amplitudes less than $0{,}406 \times PA_7$ are put out. The number $Npeaks$ of peaks is updated.

The peak amplitudes are normalized:

$$NPA_k = PA_k \Big/ \sum_{i=1}^{Npeaks} PA_i \tag{5.130}$$

### 5.6.5.5 F0 Candidates generation

Pitch candidates are selected among the local maxima of a piecewise constant *utility function* $U(F0)$:

$$U(F0)=\sum_i NPA_i \times I(PF_i/F0)$$

$$where$$

$$I(r)=\begin{cases}1, & |r|\le D1\\ 0{,}5, & D1<|r|\le D2\\ 0, & D2<|r|<0{,}5\end{cases} \tag{5.131}$$

$$I(r+1)=I(r)$$

$$D1=65/512,\ \ D2=100/512$$

Lower $F0min$ and upper $F0max$ limits for F0 are defined as the left and the right edges respectively of the processed search range $SRi,\ i=1,2,3$.

First, a partial utility function is built including only contributions of a few highest peaks. The partial utility function is represented by a list of break points. Then all local maxima locations of the partial utility function are determined. Finally, the values of the whole utility function at the local maxima are computed.

**Building partial utility function**

$NPprelim$ peaks are selected from the top of the peaks list. $NPprelim = min(Npeaks, 7)$. A counter variable is initialized BPCount = 0. For each peak $(NPA_k, PF_k), k=1,\ldots,NPprelim$, a list $\text{BPL}_\text{k}$ of the utility function break points is collected as described below.

The maximal and minimal dividers of the peak frequency are calculated:

$$N_{\min}=ceil\left[\max(0,\frac{PF_k}{F0\max}-D1)\right] \qquad N_{\max}=floor\left(\frac{PF_k}{F0\min}+D2\right) \tag{5.132}$$

The counter BPCount is updated $BPCount = BPCount + N_{max} - N_{min} + 1$ and compared against a predefined threshold BPLimit:

$$BPLimit=\begin{cases}60 & for\ SR1\\ 30 & for\ SR2\\ 20 & for\ SR3\end{cases} \tag{5.133}$$

If the counter value exceeds the threshold then the entire peaks processing is terminated, and no more break point lists are built. Otherwise the processing of the k-th peak continues. Index $n$ scans the range $[Nmin, Nmax]$ in the reverse order $n = Nmax, Nmax\text{-}1, \ldots, Nmin$ each time generating four new breakpoints in the list, each break point is given by its frequency value BPF and amplitude value BPA:

$$\begin{aligned}BPF_{4(n-1)+1}&=PF_k/(n+D2) & BPA_{4(n-1)+1}&=0{,}5\times PA_k\\ BPF_{4(n-1)+2}&=PF_k/(n+D1) & BPA_{4(n-1)+2}&=0{,}5\times PA_k\\ BPF_{4(n-1)+3}&=PF_k/(n-D1) & BPA_{4(n-1)+3}&=-0{,}5\times PA_k\\ BPF_{4(n-1)+4}&=PF_k/(n-D2) & BPA_{4(n-1)+4}&=-0{,}5\times PA_k\end{aligned} \tag{5.134}$$

Note that the break points in the list are ordered in the increasing order of the frequency.

If the list is not empty and $BPF_1 < F0\min$ then the beginning of the list is modified as follows. The first k = max(1, m - 2) elements are discarded where $m=\min i:\{BPF_i > F0\min\}$. The new head of the list (former element #m-1) is set to: $BPF = F0\min,\ BPA=\sum_{j=1}^{k+1} BPA_j$.

If the list is not empty and there are elements (at the tail) with $BPF \geq F0\max$, that elements are deleted from the list.

Finally, if $F0\max > PF_k / D2$ then one or two elements are appended at the end of the list depending on certain conditions as described below. Two frequency values are calculated: $F1 = PF_k / D2$ and $F2 = PF_k / D1$.

*if* ( $F2 < F0\min$ )

*One element is appended*: $BPF = F0\min, \quad BPA = PF_k$

*else if* ( $F1 < F0\min < F2 \leq F0\max$ )

*Two elements are appended*: $BPF = F0\min, \quad BPA = 0{,}5PF_k$ and $BPF = F2, \quad BPA = 0{,}5PF_k$

*else if* ( $F1 < F0\min \wedge F2 > F0\max$ )

*One element is appended*: $BPF = F0\min, \quad BPA = 0{,}5PF_k$

*else if* ( $F1 \geq F0\min \wedge F2 \leq F0\max$ )

*Two elements are appended*: $BPF = F1, \quad BPA = 0{,}5PF_k$ and $BPF = F2, \quad BPA = 0{,}5PF_k$

*else if* ( $F1 \geq F0\min \wedge F2 > F0\max$ )

*One element is appended*: $BPF = F1, \quad BPA = 0{,}5PF_k$

All the Break Point Lists $\{BPL_k\}$ are merged together into one array $U_{partial}=\{(BPF_n, BPA_n)\}$ preserving the frequency ascending order, and the amplitudes of the break points are modified as:

$$BPA_n = BPA_n + BPA_{n-1}, n = 2, 3, \ldots$$

If the last break point frequency is less than *F0max* then a new terminating element *(BPF = F0max, BPA = 0)* is appended to the array. Further we will refer to the number of elements in the $U_{partial}$ array as *NBP*.

**Preliminary candidates determination**

*NCprelim* break points are determined which are the highest in amplitude local maxima among the elements of the $U_{partial}$ array, where *NCprelim = min(4,NBP)*. These break points being sorted in the descended order of amplitude form a *list of preliminary candidates*. If a variable *StableTrackF0* (which is described in clause 5.6.5.8) has a non-zero value then an additional break point *BPad* is sought which is the highest in amplitude local maximum among the $U_{partial}$ array elements having frequency in the range [*StableTrackF0/1,22, StableTrackF0 × 1,22*]. If such the break point is found then the amplitude associated with it is increased by 0,06 and compared against the amplitudes of the preliminary candidates list members. If the modified amplitude is greater than the amplitude of at least one of the preliminary candidates then *BPad* is inserted into the preliminary candidate list so that the list elements order is preserved, and the last list member is put out. Finally, the frequency value for each candidate is modified as:

$$BF_n = 0{,}5 \times (BF_n + BF_{n+1})$$

If $n < NBP$ where $n$ is the index of the break point in the $U_{partial}$ array.

**Candidate amplitudes refinement**

For each preliminary candidate the amplitude value is recomputed in accordance to formula (5.131) wherein F0 is substituted by the frequency value associated with that candidate and the summation is performed over all the *Npeaks* spectral peaks.

**Final candidates determination**

*NC* (final) candidates are selected from the preliminary candidates, *NC = min(2,Nprelim)*. For the selection purpose a compare function is defined for a pair (*F1,A1*) and (*F2,A2*) of candidates given by their frequencies *Fi* and amplitudes *Ai*. Let *F1 < F2*. The first candidate is declared to be better than the second one if the following condition is satisfied:

$$A1 > A2 + 0{,}06 \vee (A1 > A2 \wedge 1{,}17 \times F1 > F2) \qquad (5.135)$$

otherwise the second candidate is considered as the best between the two.

*NC* best candidates are determined, sorted in descending order of their quality, and form a final candidates list. If the pitch estimate *PrevF0* obtained at the previous frame has non-zero value then the preliminary candidates are determined having frequency values within the interval [Prev*F0/1,22, PrevF0 × 1,22*]. If such preliminary candidates exist then one of them having the maximal amplitude is declared as an additional candidate. The amplitude *a* of the additional candidate is increased by 0,06 (*b = a + 0,06*), and compared against the amplitudes of the final candidates list members. If a member exists with amplitude less than *b* then the last member of the final candidates list is replaced by the additional candidate.

Below the amplitudes associated with the candidates are referred to as Spectral Scores (SS).

## 5.6.5.6 Computing correlation scores

Correlation score is computed for each pitch candidate. The input for correlation score calculation stage comprises the low-pass filtered extended downsampled frame (clause 5.6.4) and the candidate pitch frequency F0. Here we designate the low-pass filtered extended downsampled frame by *u(n)* and assume that the origin *n = 0* is associated with the sample #NDS counting from the end of the vector *u*, so that the preceding to it samples have negative index values. NDS is the length of downsampled frame *NDS = 200/DSMP* where DSMP is the downsampling factor (clause 5.6.4).

Candidate pitch frequency is converted to a time-domain lag:

$$\tau = \frac{8\,000}{F0 \times DSMP} \qquad (5.136)$$

An integer lag is calculated by rounding the lag value to the upper integer number $i\tau = ceil(\tau)$.
Analysis window length is calculated:

$$LW = floor\left(\frac{75}{DSMP}\right) \qquad (5.137)$$

**Offset and length parameters calculation**

Offset *O* and length *Len* parameters are calculated to be used by further processing, besides two following cases are treated differently.

**Case 1:**

$$i\tau \le LW$$

$$O = i\tau + \arg\max_{0 \le t \le NDS - LW - i\tau} E(t),$$

where:

$$E(t) = \sum_{n=t}^{t+LW+i\tau-1} u(n)^2$$

$$Len = LW + i\tau$$

**Case 2:**

$$i\tau > LW$$

Two vectors are extracted from the signal $u$:

$u1=\{u(t0), u(t0 + 1), \ldots, u(t0 + i\tau - 1)\}$ *and* $u2=\{ u(t0 - i\tau), u(t0 + 1 - i\tau), \ldots, u(t0 - 1)\}$,

where:

$$t0 = \begin{cases} NDS/2, \; if \;\; i\tau < NDS/2 \\ NDS - i\tau, \quad\quad otherwise \end{cases}$$

An auxiliary offset $ofs$ is determined as:

$$ofs = \arg\max_{0 \le t \le i\tau - 1} E(t)$$

where:

$$E(t) = \sum_{n=0}^{LW-1} (u(n0 + t \bmod i\tau + n)^2 + u(n0 + t \bmod i\tau + n - i\tau)^2),$$

$$t0 = \begin{cases} NDS/2, \; if \;\; i\tau < NDS/2 \\ NDS - i\tau, \quad\quad otherwise \end{cases}$$

If ofs+LW$\le i\tau$ then O = t0 + ofs and Len = LW.

Otherwise two sets of the offset and length parameters are prepared:

$\{O1 = t0 + ofs, Len1 = i\tau - ofs\}$ and $(O2 = t0, Len2 = LW - Len1\}$.

**Correlator**

Input parameters for this block are $O$, $Len$ and $i\tau$

Three vectors are extracted from $u$:

$X = \{u(O), u(O + 1), \ldots, u(O + Len - 1)\}^T$

$Y = \{u(O - i\tau), u(O - i\tau + 1), \ldots, u(O - i\tau + Len - 1)\}^T$

$Z = \{u(O - i\tau + 1), u(O - i\tau + 2), \ldots, u(O - i\tau + Len)\}^T$

For each vector the sum of the coordinates is computed: $\Sigma X$, $\Sigma Y$ and $\Sigma Z$. The following inner products are computed also: $X^TX$, $Y^TY$, $Z^TZ$, $X^TY$, $X^TZ$ and $Y^TZ$.

Where there are two sets of the offset and length parameters ($O1$, $Len1$) and ($O2$, $Len2$), the correlator block is applied twice, one time for each set, and the corresponding output values (the sums and the inner products) are summed.

**DC removal**

The inner products computed by the correlator are modified as follows:

$\mathbf{X^TX = X^TX - (\Sigma X)^2/LW}$

$\mathbf{Y^TY = Y^TY - (\Sigma Y)^2/LW}$

$\mathbf{Z^TZ = Z^TZ - (\Sigma Z)^2/LW}$

$\mathbf{X^TY = X^TY - \Sigma X \times \Sigma Y/LW}$

$\mathbf{X^TZ = X^TZ - \Sigma X \times \Sigma Z/LW}$

$\mathbf{Y^TZ = Y^TZ - \Sigma Y \times \Sigma Z/LW}$

**Interpolation**

Correlation score $CS$ is computed by the following interpolation formula:

$$CS = \frac{\beta \times X^T Z + \alpha \times Y^T Z}{\sqrt{Z^T Z \times \left(\beta^2 \times X^T X + 2\alpha\beta \times X^T Y + \alpha^2 \times Y^T Y\right)}} \tag{5.138}$$

where:

$$\alpha = i\tau - \tau,\ \beta = 1 - \alpha\ .$$

Finally, $CS$ value is truncated if it falls outside the interval [0, 1].

$$CS = \max(CS,0),$$

$$CS = \min(CS,1).$$

### 5.6.5.7 Pitch estimate selection

Input to this stage is the set of pitch candidates. Each candidate ($F0_k$, $SS_k$, $CS_k$) is represented by the corresponding pitch frequency $F0_k$, spectral score (the utility function value) $SS_k$ and correlation score $CS_k$. The block outputs a pitch estimate (F0, SS, CS) which either is selected among the candidates or indicates that that the frame represents unvoiced speech in which case F0 is set to 0.

Pitch estimate selection block might be entered several (at most 3) times during the processing of one frame. It is entered after pitch candidates generation is performed for each pitch search interval SRi. Each time the list of pitch candidates which is fed into the block is updated appropriately to include all the pitch candidates detected so far. Thus the list passed into this block after the processing of SR3 search range includes the candidates found within this range, typically two candidates. If one of the candidates is selected as the pitch estimate then the pitch estimation process terminates and the control flows to the history information update block (described in clause 5.6.5.8). Otherwise the candidates generated within the SR2 range are combined with the ones found within SR3 and the combined list (typically containing four candidates) is fed into pitch estimate selection block. If no pitch estimate is selected at this time the block is entered again after SR1 range is processed. At this time the candidate list contains the candidates generated in all the three ranges (typically 6 candidates). A variable $EPT$ which is fed to the block along with the candidates list indicates whether the list contains candidates generated for all the three search ranges ($EPT = 1$) or not ($EPT = 0$).

The selection process is shown on the flow-chart of figure 5.10.

The candidates are sorted at step 100 in descending order of their $F0$ values. Then at step 110 the candidates are scanned sequentially until a candidate of *class 1* is found, or all the candidates are tested. A candidate is defined to be of class 1 if the $CS$ and $SS$ values associated with the candidate satisfy the following condition:

$$(CS \geq C1 \text{ AND } SS \geq S1) \text{ OR } (SS \geq S11 \text{ AND } SS + CS \geq C\,S1) \qquad \text{(Class 1 condition)}$$

where:

$$C1 = 0{,}79,\ S1 = 0{,}78,\ S11 = 0{,}68 \text{ and } CS1 = 1{,}6.$$

At step 130 the flow branches. If a class 1 candidate is found it is selected to be a *preferred candidate*, and the control is passed to step 140 performing a *Find Best in Vicinity* procedure described by the following. Those candidates among the ones following in the list the preferred candidate are checked to determine those ones which are close in terms of $F0$ to the preferred candidate. Two values $F0_1$ and $F0_2$ are defined to be close to each other if:

$$(F01 < 1{,}2 \times F02 \text{ AND } F02 < 1{,}2 \times F01) \qquad \text{(Closeness condition).}$$

A plurality of *better* candidates is determined among the close candidates. A better candidate must have a higher SS and a higher CS values than those of the preferred candidate respectively. If at least one better candidate exists then the *best* candidate is determined among the better candidates. The best candidate is characterized by that there is no other better candidate, which has a higher SS and a higher CS values than those of the best candidate respectively. The best candidate is selected to be a preferred candidate instead of the former one. If no better candidate is found the preferred candidate remains the same.

At step 150 the candidates following the preferred candidate are scanned one by one until either a candidate of class 1 is found whose scores $SS_{candidate}$ and $CS_{candidate}$ satisfy following condition:

$$SS_{candidate} + CS_{candidate} \geq SS_{preferred} + CS_{preferred} + 0{,}18$$

or all the candidates are scanned. If a candidate is found which meets the above condition it is selected to be the preferred candidate and Find Best in Vicinity procedure is applied. Otherwise the control is passed directly to step 180, where the $EPT$ variable value is tested. If $EPT$ indicates that all the pitch search ranges have been processed the pitch estimate is set to the preferred candidate. Otherwise the following condition is tested:

$$SS_{preferred} \geq 0{,}95 \text{ AND } CS_{preferred} \geq 0{,}95$$

If the condition is satisfied the pitch estimate is set to the preferred candidate, otherwise the pitch frequency $F0$ is set to 0 indicating that no pitch is detected.

Returning to the conditional branching step 130, if no class 1 candidate is found then at step 120 it is checked if the $StableTrackF0$ variable has non-zero value in which case the control is passed to step 210, otherwise step 270 is performed.

At step 210 a reference fundamental frequency value $F0_{ref}$ is set to $StableTrackF0$. Then at step 220 the candidates are scanned sequentially until either a candidate of a *class 2* is found or all the candidates are tested. A candidate is defined to be of class 2 if the frequency and the score values associated with it satisfy the condition:

$$(CS > C2 \text{ AND } SS > S2) \text{ AND } (1/1{,}22 < |F0/F0ref\,| < 1{,}22 \qquad \text{(Class 2 condition)}$$

where $C2 = 0{,}7$, $S2 = 0{,}7$. If no class 2 candidate is found then the pitch estimate is set to 0 at step 240. Otherwise, the class 2 candidate is chosen to be the preferred candidate and Find Best in Vicinity procedure is applied at step 250. Then at step 260 the pitch estimate is set to the preferred candidate.

Returning to the conditional branching step 120, if $StableTrackF0 = 0$ then control is passed to step 270 where a *Continuous Pitch Condition*:

$$PrevF0 > 0 \text{ AND } StablePitchCount > 1$$

is tested (StablePitchCount variable is described below in clause 5.9.8) If the condition is satisfied then at step 280 the frequency reference value $F0_{ref}$ is set to $PrevF0$ and the class 2 candidate search is performed at step 290. If a class 2 candidate is found (test step 300) then it is selected as the preferred candidate, Find Best In Vicinity procedure is applied at step 310, and the pitch estimate is set to the preferred candidate at step 320. Otherwise, the processing proceeds with step 330 likewise it happens if Continuous Pitch Condition test of step 270 fails.

At step 330 the candidates are scanned sequentially until a candidate of *class 3* is found or all the candidates are tested. A candidate is defined to be of class 3 if the scores associated with it satisfy the condition:

$$(CS \geq C3 \text{ OR } SS \geq S3) \qquad \text{(Class 3 condition)}$$

where, $C3 = 0{,}85$, $S3 = 0{,}82$. If no class 3 candidate is found then the pitch frequency is set to 0. Otherwise, the class 3 candidate is selected as the preferred candidate, and Find Best in Vicinity procedure is applied at step 360. Then at step 370 the pitch estimate is set to the preferred candidate.

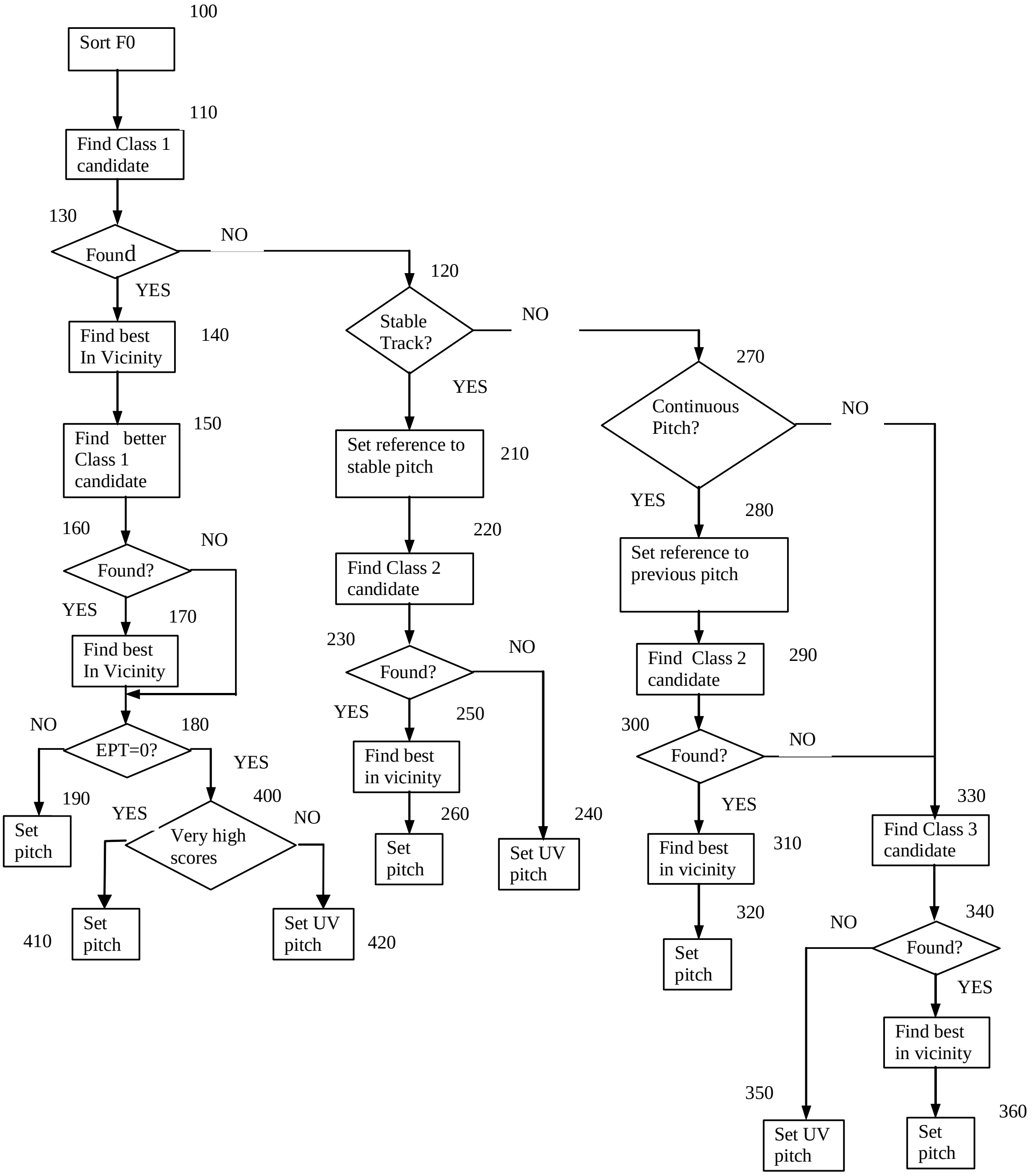


**Figure 5.10: Pitch estimate selection**

### 5.6.5.8 History information update

The pitch estimator maintains following variables holding information on the estimation process history: *PrevF0, StableTrackF0, StablePitchCount* and *DistFromStableTrack*.

The variables are initialized as follows:

*PrevF0 = 0, StablePitchCount = 0, DistFromStableTrack = 1 000, StableTrackF0 = 0.*

The variables are updated at each frame after pitch estimation processing is completed and the pitch frequency estimate *F0* is set. The update process is described by the following pseudo code section.

```
if (F0 > 0  AND  PrevF0 > 0  AND 1/1.22 < |F0/PrevF0| < 1.22)
    StablePitchCount = StablePitchCount + 1;
else
    StablePitchCount = 0;
if (StablePitchCount ≥ 6)
{
    DistFromStableTrack = 0;
    StableTrackF0 = F0;
}
else if (DistFromStableTrack  ≤ 2 )
{
    if (StableTrackF0 > 0  AND 1/1.22 < |F0/StableTrackF0| < 1.22)
    {
        DistFromStableTrack = 0;
        StableTrackF0 = F0;
    }
    else
        DistFromStableTrack = DistFromStableTrack + 1;
}
else {
    StableTrackF0 = 0;
    DistFromStableTrack = DistFromStableTrack + 1;
}

PrevF0 = F0;
```

### 5.6.5.9 Output pitch value

The pitch frequency estimate *F0* is converted to an output pitch value *P* representing pitch period duration measured in sampling intervals.

$$P = \begin{cases} 0 \; if \;\; F0 = 0 \\ 8\,000 / F0 \;\; otherwise \end{cases} \qquad (5.139)$$

## 5.6.6 Classification

The inputs to the classification block are the *vad_flag* and *hangover_flag* from the *VAD* block, the frame energy *E* from the *SEC* block, the input signal $s_{in}$, the upper-band signal $s_{ub}$ from the *PP* block, and the pitch period estimate *P* from the *PITCH* block. The output of the classification block is the voicing class *VC*, which is one of the output parameters of the front-end.

The voicing class *VC* is estimated from the different inputs to the classification block as follows. From the upper-band signal $s_{ub}$ and the frame energy *E*, the upper-band energy fraction $EF_{ub}$ is computed as:

$$EF_{ub} = \frac{\sum_{i=1}^{N} s_{ub}(i)^2}{E} \qquad (5.140)$$

From the offset-free input signal $s_{of}$, the zero-crossing measure *ZCM* is computed as follows:

$$ZCM = \frac{1}{2(N-1)} \sum_{i=2}^{N} |\, \mathrm{sgn}[s_{of}(i)] - \mathrm{sgn}[s_{of}(i-1)] \,| \qquad (5.141)$$

where:

$$\mathrm{sgn}[s_{of}(i)] = \begin{cases} +1, \; s_{of}(i) \ge 0 \\ -1, \; s_{of}(i) < 0 \end{cases} \qquad (5.142)$$

The logic used by the classification block is illustrated by the pseudo-code below.

```
if (vad_flag == FALSE)
    VC = "non-speech";
else if (P == 0)
    VC = "unvoiced";
else if ((hangover_flag == TRUE) || ($EF_{ub}$ ≤ EF_UB_THLD) || (ZCM >= ZCM_THLD))
    VC = "mixed-voiced";
else
    VC = "fully-voiced";
end
```

The upper-band energy fraction threshold EF_UB_THLD is 0.0018 and the zero-crossing measure threshold ZCM_THLD is 0,4375.

# 6 Feature compression

## 6.1 Introduction

This clause describes the distributed speech recognition front-end feature vector compression algorithm. The algorithm makes use of the parameters from the front-end feature extraction algorithm of clause 5. Its purpose is to reduce the number of bits needed to represent each front-end feature vector.

## 6.2 Compression algorithm description

### 6.2.1 Input

The compression algorithm is designed to take the feature parameters for each short-time analysis frame of speech data as they are available and as specified in clause 5.4.

Fourteen of the sixteen parameters are compressed using a Vector Quantizer (VQ). The input parameters for the VQ are the first twelve static Mel cepstral coefficients:

$$\mathbf{c_{eq}}(t) = \left[c_{eq}(1,t), c_{eq}(2,t), \ldots, c_{eq}(12,t)\right]^T \qquad (6.1)$$

where $t$ denotes the frame index, plus the zeroth cepstral coefficient $c(0)$ and a log energy term $lnE(t)$ as defined in clause 5.3.2. The final input to the compression algorithm is the VAD flag. These parameters are formatted as:

$$y(t) = \begin{bmatrix} \mathbf{c_{eq}}(t) \\ VAD(t) \\ c(0,t) \\ lnE(t) \end{bmatrix} \qquad (6.2)$$

The remaining two parameters, viz., pitch period and class, are compressed jointly using absolute and differential scalar quantization techniques.

### 6.2.2 Vector quantization

The feature vector $\boldsymbol{y}(t)$ is directly quantized with a split vector quantizer. The 14 coefficients (c(1) to c(12), c(0) and $lnE$) are grouped into pairs, and each pair is quantized using its own VQ codebook. The resulting set of index values is then used to represent the speech frame. Coefficient pairings (by front-end parameter) are shown in table 6.1, along with the codebook size used for each pair. The VAD flag is transmitted as a single bit. $c(1)$ to $c(10)$ are quantized with 6 bits per pair, while $c(11)$ and $c(12)$ are quantized with 5 bits. The closest VQ centroid is found using a weighted Euclidean distance to determine the index:

$$d_j^{i,i+1} = \begin{bmatrix} y_i(t) \\ y_{i+1}(t) \end{bmatrix} - q_j^{i,i+1} \qquad (6.3)$$

$$idx^{i,i+1}(t) = \underset{0 \le j \le \left(N^{i,i+1}-1\right)}{\text{argmin}} \left\{\left(d_j^{i,i+1}\right) W^{i,i+1} \left(d_j^{i,i+1}\right)\right\}, \quad i = \{0, 2, 4...12\} \tag{6.4}$$

where $q_j^{i,i+1}$ denotes the *j*th codevector in the codebook $Q^{i,i+1}$, $N^{i,i+1}$ is the size of the codebook, $W^{i,i+1}$ is the (possibly identity) weight matrix to be applied for the codebook $Q^{i,i+1}$, and $idx^{i,i+1}(t)$ denotes the codebook index chosen to represent the vector $\left[y_i(t), y_{i+1}(t)\right]^T$. The indices are then retained for transmission to the back-end.

**Table 6.1: Split vector quantization feature pairings**

| | Size | Weight Matrix | | |
|---|---|---|---|---|
| **Codebook** | **($N^{I,I+1}$)** | **($W^{I,I+1}$)** | **Element 1** | **Element 2** |
| $Q^{0,1}$ | 64 | I | c(1) | c(2) |
| $Q^{2,3}$ | 64 | I | c(3) | c(4) |
| $Q^{4,5}$ | 64 | I | c(5) | c(6) |
| $Q^{6,7}$ | 64 | I | c(7) | c(8) |
| $Q^{8,9}$ | 64 | I | c(9) | c(10) |
| $Q^{10,11}$ | 32 | I | c(11) | c(12) |
| $Q^{12,13}$ | 256 | Non-identity | c(0) | lnE |

Two sets of VQ codebooks are defined; one is used for speech sampled at 8 kHz or 11 kHz while the other for speech sampled at 16 kHz. The numeric values of these codebooks and weights are specified as part of the software implementing the standard. The weights used (to one decimal place of numeric accuracy) are:

8 kHz or 11 kHz sampling rate $$W^{12,13} = \begin{bmatrix} 10\,498{,}7 & 0 \\ 0 & 15{,}1 \end{bmatrix}$$

16 kHz sampling rate $$W^{12,13} = \begin{bmatrix} 10\,617{,}2 & 0 \\ 0 & 21{,}8 \end{bmatrix}$$

## 6.2.3 Pitch and class quantization

The pitch period of a frame can range from 19 samples to 140 samples (both inclusive) at 8 kHz sampling rate. The voicing class of a frame can be one of the following four: *non-speech*, *unvoiced speech*, *mixed-voiced speech*, and (fully) *voiced speech*. The class information of a frame is represented jointly using the pitch and class indices. The pitch information of alternate frames is quantized absolutely using 7 bits or differentially using 5 bits.

### 6.2.3.1 Class quantization

When the voicing class of a frame is non-speech or unvoiced speech, the pitch index of the corresponding frame is chosen to be *zero*, i.e. all-zero codeword either 5 bits or 7 bits long. For non-speech, the 1-bit class index is chosen as 0, and for unvoiced speech, the class index is chosen as 1. For such frames, the pitch period is indeterminate.

When the voicing class of a frame is mixed-voiced speech or (fully) voiced speech, the pitch index of the corresponding frame is chosen to be some index other than zero, either 5 bits or 7 bits long. For mixed-voiced speech, the 1-bit class index is chosen as 0, and for (fully) voiced speech, the class index is chosen as 1. For such frames, the pitch index specifies the pitch period as discussed under clause 5.2.3.2.

Thus the pitch and class indices of a frame jointly determine the voicing class of the frame as illustrated in table 6.2.

**Table 6.2: Class quantization**

| Voicing Class (VC) | Pitch index (Pidx) | Class index (Cidx) |
|---|---|---|
| Non-speech | 0 | 0 |
| Unvoiced-speech | 0 | 1 |
| Mixed-voiced speech | > 0 | 0 |
| Fully-voiced speech | > 0 | 1 |

### 6.2.3.2 Pitch quantization

The pitch period of an even-numbered frame (with the starting frame numbered zero), or equivalently, the first frame of each frame pair is quantized absolutely using 7 bits. Out of the 128 indices ranging from 0 to 127, the index 0 is reserved for indicating that the voicing class is non-speech or unvoiced speech as discussed under clause 5.2.3.1. The remaining 127 indices are assigned in increasing order to 127 quantization levels that span the range from 19 to 140 uniformly in the log-domain. Given the pitch period of the frame, the quantization level that is closest to the pitch period in the Euclidean sense and the corresponding index are chosen:

$$Pidx(m) = \underset{1 \le j \le 127}{\arg\min} (P(m) - q_j)^2 \qquad (6.5)$$

where $P(m)$ is the pitch period of the $m^{\text{th}}$ frame ($m$ even), $q_j$ is the $j^{\text{th}}$ quantization level, and $Pidx(m)$ is the pitch quantization index for the $m^{\text{th}}$ frame.

The pitch period of an odd-numbered frame (with the starting frame numbered zero), or equivalently, the second frame of each frame pair is quantized differentially using 5 bits. Out of the 32 indices ranging from 0 to 31, the index 0 is reserved for indicating that the voicing class is non-speech or unvoiced speech as discussed under clause 6.2.3.1. The remaining 31 indices are assigned in increasing order to 31 quantization levels, which are chosen depending on which of the three preceding quantized pitch periods serves as the reference (for differential quantization) and what its value is. The choice of the reference pitch period and the 31 quantization levels for different situations are illustrated in table 6.3. With reference to the table, a quantized pitch period value with a non-zero index may be *reliable* or *unreliable* to serve as a reference. An absolutely quantized pitch period value is always considered reliable. A differentially quantized pitch period value is considered reliable only if the reference value used for its quantization is the quantized pitch period value of the preceding frame. In table 6.3, the different quantization levels are specified as a factor that multiplies the chosen reference value. If any quantization level falls outside the pitch range of 19 to 140, then it is limited to the appropriate boundary value.

**Table 6.3: Choice of reference and quantization levels for differential quantization**

| **Pitch indices of preceding 3 frames** | | | **Choice of reference pitch period and 31 quantization levels for (*m*+1)th frame** |
|---|---|---|---|
| **Pidx(m-2)** | **Pidx(m-1)** | **Pidx(m)** | |
| 0 | 0<br>OR<br>> 0 but unreliable | 0 | No suitable reference is available. Use 5-bit absolute quantization.<br>The 31 quantization levels are chosen to span the range from 19 to 140 uniformly in the log-domain. |
| Do not care | Don't care | > 0 | The quantized pitch period value of the *m*th frame is chosen as the reference.<br>Out of the 31 quantization levels, 27 are chosen to cover the range from (0,8163 × reference) to (1,2250 × reference) uniformly in the log-domain.<br>The other 4 levels depend on the reference value as follows:<br>19 ≤ reference ≤ 30 - (2,00, 3,00, 4,00, 5,00) × reference<br>30 < reference ≤ 60 - (1,50, 2,00, 2,50, 3,00) × reference<br>60 < reference ≤ 95 - (0,50, 0,67, 1,50, 2,00) × reference<br>95 < reference ≤ 140 - (0,25, 0,33, 0,50, 0,67) × reference |
| Do not care | > 0<br>Reliable | 0 | The quantized pitch period value of the (*m*-1)th frame is chosen as the reference.<br>The choice of quantization levels is the same as shown in the row below. |
| > 0 | 0<br>OR<br>> 0 but unreliable | 0 | The quantized pitch period value of the (*m*-2)th frame is chosen as the reference.<br>Out of the 31 quantization levels, 25 are chosen to cover the range from (0,7781 × reference) to (1,2852 × reference) uniformly in the log-domain.<br>The other 6 levels depend on the reference value as follows:<br>19 ≤ reference ≤ 30 - (1,50, 2,00, 2,50, 3,00, 4,00, 5,00) × reference<br>30 < reference ≤ 60 - (0,67, 1,50, 2,00, 2,50, 3,00, 4,00) × reference<br>60 < reference ≤ 95 - (0,33, 0,50, 0,67, 1,50, 1,75, 2,00) × reference<br>95 < reference ≤ 140 - (0,20, 0,25, 0,33, 0,50, 0,67, 1,50) × reference |

The 31 indices used for differential quantization are assigned in increasing order to the 31 quantization levels. Given the pitch period of the frame, the quantization level that is closest to the pitch period in the Euclidean sense and the corresponding index are chosen:

$$Pidx(m+1) = \underset{1 \le j \le 31}{\arg\min} (P(m+1) - q_j)^2 \qquad (6.6)$$

where $P(m+1)$ is the pitch period of the (m+1)th frame ($m$ even), $q_j$ is the $j$th quantization level, and $Pidx(m+1)$ is the pitch quantization index for the (m+1)th frame.

# 7 Framing, bit-stream formatting and error protection

## 7.1 Introduction

This clause describes the format of the bitstream used to transmit the compressed feature vectors. The frame structure used and the error protection that is applied to the bitstream is defined. The basic unit for transmission consists of a pair of speech frames and associated error protection bits with the format defined in clause 7.2.4. This frame pair unit can be used either for circuit data systems or packet data systems such as the IETF Real-Time Protocols (RTP). For circuit data transmission a multiframe format is defined consisting of 12 frame pairs in each multiframe and is described in clauses 7.2.1 to 7.2.3. The formats for DSR transmission using RTP are defined in the IETF Audio Video Transport, Internet-Draft (see bibliography) where the number of frame pairs sent per payload is flexible and can be designed for a particular application.

## 7.2 Algorithm description

### 7.2.1 Multiframe format

In order to reduce the transmission overhead, each multiframe message packages speech features from multiple short-time analysis frames. A multiframe, as shown in table 7.1, consists of a synchronization sequence, a header field, and a stream of frame packets.

**Table 7.1: Multiframe format**

| Sync Sequence | Header Field | Frame Packet Stream |
|---|---|---|
| <- 2 octets -> | <- 4 octets -> | <- 162 octets -> |
| <- 168 octets -> | | |

In order to improve the error robustness of the protocol, the multiframe has a fixed length (168 octets). A multiframe represents 240 ms of speech, resulting in a data rate of 5 600 bits/s.

In the specification that follows, octets are transmitted in ascending numerical order; inside an octet, bit 1 is the first bit to be transmitted. When a field is contained within a single octet, the lowest-numbered bit of the field represents the lowest-order value (or the least significant bit). When a field spans more than one octet, the lowest-numbered bit in the first octet represents the lowest-order value (LSB), and the highest-numbered bit in the last octet represents the highest-order value (MSB). An exception to this field mapping convention is made for the cyclic redundancy code (CRC) fields. For these fields, the lowest numbered bit of the octet is the highest-order term of the polynomial representing the field. In simple stream formatting diagrams (e.g. table 7.1) fields are transmitted left to right.

### 7.2.2 Synchronization sequence

Each multiframe begins with the 16-bit synchronization sequence 0 × 87B2 (sent LSB first, as shown in table 7.2).

The inverse synchronization sequence 0 × 784D can be used for synchronous channels requiring rate adaptation. Each multiframe may be preceded or followed by one or more inverse synchronization sequences. The inverse synchronization is not required if a multiframe is immediately followed by the synchronization sequence for the next multiframe.

**Table 7.2: Multiframe synchronization sequence**

| Bit | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Octet |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 2 |

### 7.2.3 Header field

Following the synchronization sequence, a header field is transmitted. Due to the critical nature of the data in this field, it is represented in a (31, 16) extended systematic codeword. This code will support 16-bits of data and has an error correction capability for up to three bit errors, an error detection capability for up to seven bit errors, or a combination of both error detection and correction.

Ordering of the message data and parity bits is shown in table 7.3, and definition of the fields appears in table 7.4. The 4 bit multiframe counter gives each multiframe a modulo-16 index. The counter value for the first multiframe is "0001". The multiframe counter is incremented by one for each successive multiframe until the final multiframe. The final multiframe is indicated by zeros in the frame packet stream (see clause 7.2.4).

NOTE: The remaining nine bits which are currently undefined are left for future expansion. A fixed length field has been chosen for the header in order to improve error robustness and mitigation capability.

**Table 7.3: Header field format**

| Bit | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Octet |
|---|---|---|---|---|---|---|---|---|---|
| | Ext | MframeCnt | | | | feType | SampRate | | 1 |
| | EXP8 | EXP7 | EXP6 | EXP5 | EXP4 | EXP3 | EXP2 | EXP1 | 2 |
| | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | 3 |
| | P16 | P15 | P14 | P13 | P12 | P11 | P10 | P9 | 4 |

**Table 7.4: Header field definitions**

| Field | No. Bits | Meaning | Code | Indicator |
|---|---|---|---|---|
| SampRate | 2 | sampling rate | 00 | 8 kHz |
| | | | 01 | 11 kHz |
| | | | 10 | undefined |
| | | | 11 | 16 kHz |
| FeType | 1 | Front-end specification | 0 | standard |
| | | | 1 | noise robust |
| MframeCnt | 4 | multiframe counter | xxxx | Modulo-16 number |
| Ext | 1 | Extended front-end | 0 | Not extended (4 800 bps) |
| | | | 1 | Extended (5 600 bps) |
| EXP1 - EXP8 | 8 | Expansion bits (TBD) | 0 | (zero pad) |
| P1 - P16 | 16 | Cyclic code parity bits | (see below) | |

The generator polynomial used is:

$$g_1(X) = 1 + X^{8} + X^{12} + X^{14} + X^{15} \tag{7.1}$$

The proposed (31, 16) code is extended, with the addition of an (even) overall parity check bit, to 32 bits. The parity bits of the codeword are generated using the calculation:

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \\ P_{15} \\ P_{16} \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1 \\ 1&1&0&0&0&0&0&0&1&1&0&0&1&1&1&0 \\ 1&1&1&0&0&0&0&0&1&1&1&0&1&1&0&1 \\ 0&1&1&1&0&0&0&0&0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&0&0&1&0&1&1&0&0&0&0 \\ 0&1&0&1&1&1&0&0&0&1&0&1&1&0&0&0 \\ 0&0&1&0&1&1&1&0&0&0&1&0&1&1&0&0 \\ 0&0&0&1&0&1&1&1&0&0&0&1&0&1&1&0 \\ 1&0&0&0&1&0&1&1&0&0&0&0&0&0&0&1 \\ 0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&1 \\ 0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&1 \\ 0&0&0&0&1&0&0&0&1&0&1&1&0&0&0&1 \\ 0&0&0&0&0&1&0&0&0&1&0&1&1&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1 \end{bmatrix}^{T} \times \begin{bmatrix} SampRate1 \\ SampRate2 \\ feType \\ MFrameCnt1 \\ MFrameCnt2 \\ MFrameCnt3 \\ MFrameCnt4 \\ Ext \\ EXP1 \\ EXP2 \\ EXP3 \\ EXP4 \\ EXP5 \\ EXP6 \\ EXP7 \\ EXP8 \end{bmatrix} \tag{7.2}$$

where $T$ denotes the matrix transpose.

### 7.2.4 Frame packet stream

Each 10 ms frame from the front-end is represented by the codebook indices specified in clause 6.2.2, the pitch index and class index specified in clause 6.2.3, and the VAD flag. The indices and the VAD flag for a pair of frames are formatted according to table 7.5.

NOTE: The exact alignment with octet boundaries will vary from frame pair to frame pair.

**Table 7.5: Frame information for *m*$^{th}$ and (m+1)$^{th}$ frames**

<table>
<tr><th>Bit</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>Octet</th></tr>
<tr><td></td><td colspan="2">Idx$^{2,3}$(m)</td><td colspan="6">Idx$^{0,1}$(m)</td><td>1</td></tr>
<tr><td></td><td colspan="4">Idx$^{4,5}$(m)</td><td colspan="4">Idx$^{2,3}$(m) (cont)</td><td>2</td></tr>
<tr><td></td><td colspan="6">Idx$^{6,7}$(m)</td><td colspan="2">Idx$^{4,5}$(m) (cont)</td><td>3</td></tr>
<tr><td></td><td>Idx$^{10,11}$(m)</td><td>VAD(m)</td><td colspan="6">Idx$^{8,9}$(m)</td><td>4</td></tr>
<tr><td></td><td colspan="4">Idx $^{12,13}$(m)</td><td colspan="4">Idx $^{10,11}$(m) (cont)</td><td>5</td></tr>
<tr><td></td><td colspan="4">Idx$^{0,1}$(m+1)</td><td colspan="4">Idx $^{12,13}$(m) (cont)</td><td>6</td></tr>
<tr><td></td><td colspan="6">Idx$^{2,3}$(m+1)</td><td colspan="2">Idx$^{0,1}$(m+1) (cont)</td><td>7</td></tr>
<tr><td></td><td colspan="2">Idx$^{6,7}$(m+1)</td><td colspan="6">Idx$^{4,5}$(m+1)</td><td>8</td></tr>
<tr><td></td><td colspan="4">Idx$^{8,9}$(m+1)</td><td colspan="4">Idx$^{6,7}$(m+1) (cont)</td><td>9</td></tr>
<tr><td></td><td colspan="5">Idx$^{10,11}$(m+1)</td><td>VAD(m+1)</td><td colspan="2">Idx$^{8,9}$(m+1) (cont)</td><td>10</td></tr>
<tr><td></td><td colspan="8">Idx $^{12,13}$(m)</td><td>11</td></tr>
<tr><td></td><td colspan="4">Pidx(m)</td><td colspan="4">CRC(m,m+1)</td><td>12</td></tr>
<tr><td></td><td colspan="5">Pidx(m+1)</td><td colspan="3">Pidx(m) (cont)</td><td>13</td></tr>
<tr><td></td><td colspan="4"></td><td colspan="2">PC-CRC(m,m+1)</td><td>Cidx(m+1)</td><td>Cidx(m)</td><td>14</td></tr>
</table>

The codebook indices for each frame take up 44 bits. After two frames worth of codebook indices, or 88 bits, a 4-bit CRC ( $g(X) = 1 + X + X^4$ ) calculated on these 88 bits immediately follows it. The pitch indices of the first frame (7 bits) and the second frame (5 bits) of the frame pair then follow. The class indices of the two frames in the frame pair worth 1 bit each next follow. Finally, a 2-bit CRC (denoted by PC-CRC) calculated on the pitch and class bits (total: 14 bits) of the frame pair using the binary polynomial $g(X) = 1 + X + X^2$ is included. The total number of bits in frame pair packet is therefore 44 + 44 + 4 + 7 + 5 + 1 + 1 + 2 = 108, or 13,5 octets. Twelve of these frame pair packets are combined to fill the 162 octet (1 296 bit) feature stream. When the feature stream is combined with the overhead of the synchronization sequence and the header, the resulting format requires a data rate of 5 600 bits/s.

All trailing frames within a final multiframe that contain no valid speech data will be set to all zeros.

# 8 Bit-stream decoding and error mitigation

## 8.1 Introduction

This clause describes the algorithms used to decode the received bitstream to regenerate the speech feature vectors. It also covers the error mitigation algorithms that are used to minimize the consequences of transmission errors on the performance of a speech recognizer and/or a speech reconstructor.

## 8.2 Algorithm description

### 8.2.1 Synchronization sequence detection

The method used to achieve synchronization is not specified in the present document. The detection of the start of a multiframe may be done by the correlation of the incoming bit stream with the synchronization flag. The output of the correlator may be compared with a correlation threshold (the value of which is not specified in this definition). Whenever the output is equal to or greater than the threshold, the receiver should decide that a flag has been detected. For increased reliability in the presence of errors the header field may also be used to assist the synchronization method.

### 8.2.2 Header decoding

The decoder used for the header field is not specified in the present document. When the channel can be guaranteed to be error-free, the systematic codeword structure allows for simple extraction of the message bits from the codeword. In the presence of errors, the code may be used to provide either error correction, error detection, or a combination of both moderate error correction capability and error detection capability.

In the presence of errors, the decoding of the frame packet stream in a multiframe is not started until at least two headers have been received in agreement with each other. Multiframes are buffered for decoding until this has occurred. The header block in each received multiframe has its cyclic error correction code decoded and the "common information carrying bits" are extracted. With the header defined in the present document the "common information carrying bits" consist of SampRate, FeType, Ext, and EXP1 - EXP8 (expansion bits).

NOTE: The use of EXP1 - EXP8 depends on the type of information they may carry in the future. Only those bits which do not change between each multiframe are used in the check of agreement described above.

Once the common information carrying bits have been determined then these are used for all the multiframes in a contiguous sequence of multiframes.

### 8.2.3 Feature decompression

Codebook, pitch, and class indices and the VAD flag are extracted from the frame packet stream, with optional checking of CRC and PC-CRC. (Back-end handling of frames failing the CRC and PC-CRC check is specified in clause 8.2.4.) Using the codebook indices received, estimates of the front-end features are extracted with a VQ codebook lookup:

$$\begin{bmatrix} \hat{y}_i(t) \\ \hat{y}_{i+1}(t) \end{bmatrix} = q^{i,i+1}_{idx^{i,i+1}(m)} \qquad i = \{0,\ 2,\ 4 \ldots 12\} \qquad (8.1)$$

From the pitch and class indices, the voicing class feature is extracted as specified in table 6.2. For non-speech and unvoiced frames, the pitch period is indeterminate. For a mixed-voiced or (fully) voiced frame, the pitch period is estimated from the pitch index as follows. For a frame with absolute pitch quantization (*m* even), the pitch index directly specifies the quantized pitch period. For a frame with differential pitch quantization (*m* odd), the pitch index specifies the factor by which the reference has to be multiplied. The reference, which can be the quantized pitch period value of any one of the preceding three frames, is obtained using the rules of table 6.3. If no suitable reference is available (Row 1 of table 6.3), then the pitch index directly specifies the quantized pitch period.

### 8.2.4 Error mitigation

#### 8.2.4.1 Detection of frames received with errors

When transmitted over an error prone channel then the received bitstream may contain errors. Two methods are used to determine if a frame pair packet has been received with errors:

- CRC and PC-CRC: The CRC recomputed from the codebook indices of the received frame pair packet data does not match the received CRC for the frame pair, or, the PC-CRC recomputed from the pitch and class indices of the received frame pair packet data does not match the received PC-CRC for the frame pair, or both.
- Data consistency: A heuristic algorithm to determine whether or not the decoded parameters for each of the two speech vectors in a frame pair packet are consistent. The details of this algorithm are described below.

The parameters corresponding to each index, idx$^{i,\ i+1}$, of the two frames within a frame packet pair are compared to determine if either of the indices are likely to have been received with errors:

$$badindexflag_i = \begin{cases} 1 & \text{if} \quad \left(y_i(t+1) - y_i(t) > T_i\right) \ OR \ \left(y_{i+1}(t+1) - y_{i+1}(t) > T_{i+1}\right) \\ 0 & \qquad\qquad otherwise \end{cases} \quad \text{i = \{0,2....12\}} \qquad (8.2)$$

The thresholds $T_i$ have been determined based on measurements of error free speech. A voting algorithm is applied to determine if the whole frame pair packet is to be treated as if it had been received with transmission errors. The frame pair packet is classified as received with error if:

$$\sum_{i=0,2,...12} badindexflag_i \geq 2 \qquad (8.3)$$

The data consistency check for erroneous data is only applied when frame pair packets failing the CRC test are detected. It is applied to the frame pair packet received before the one failing the CRC test and successively to frames after one failing the CRC test until one is found that passes the data consistency test. The details of this algorithm are shown in the flow charts of figures 8.1 and 8.2.

### 8.2.4.2 Substitution of parameter values for frames received with errors

The parameters from the last speech vector received without errors before a sequence of one or more "bad" frame pair packets and those from the first good speech vector received without errors afterwards are used to determine replacement vectors to substitute for those received with errors. If there are B consecutive bad frame pairs (corresponding to 2B speech vectors) then the first B speech vectors are replaced by a copy of the last good speech vector before the error and the last B speech vectors are replaced by a copy of the first good speech vector received after the error. It should be noted that the speech vector includes the 12 static cepstral coefficients, the zeroth cepstral coefficient, the log energy term and the VAD flag, and all are therefore replaced together. In the presence of errors, the decoding of the frame packet stream in a multiframe is not started until at least two headers have been received in agreement with each other. Multiframes are buffered for decoding.

### 8.2.4.3 Modification of parameter values for frames received with errors

The log*E*, pitch, and class parameters of frames received with errors are modified as follows after the substitution step described in clause 8.2.4.2. This modification step affects only back-end speech reconstruction - it does not affect speech recognition.

First, a 3-point median filter is applied to the log*E* parameter. The median value of the log*E* parameters of the preceding, current, and succeeding frames replaces the log*E* parameter of the current frame. The median filter is switched on only after the first frame error has been detected. In other words, there is no median filtering for an error-free channel.

Second, the log*E*, pitch, and class parameters of frames received with errors are modified according to the runlength of errors. Let the runlength of errors be 2B frames. If 2B is less than or equal to 4, no parameter modification is done. In this case, because of the substitution step in clause 8.2.4.2, the first B frames receive their parameters from the good frame on the left (before the error) and the next B frames receive their parameters from the good frame on the right (after the error).

For a runlength greater than 4 but less than or equal to 24, the parameter modification is done as follows. The parameters of the first two frames and last two frames are not modified. From the 3rd frame to the B$^{th}$ frame, the log*E* parameter is decreased linearly from left to right by 2 per frame. The value of the log*E* parameter is however not allowed to go below 4,7. If these frames are (fully) voiced, then they are modified to mixed-voiced frames. The pitch parameters are not changed. From the (2B-2)$^{th}$ frame to (B+1)$^{th}$ frame (both inclusive), the log*E* parameter is decreased linearly from right to left by 2 per frame with a floor value of 4,7. Fully voiced frames are modified to mixed-voiced frames and the pitch parameters are not modified.

If the runlength of errors is greater than 24, then the first 12 and the last 12 frames are handled exactly as above. The remaining (2B-12) frames in the middle are modified as follows. The log*E* parameter is set to 4,7, the class parameter is set to "unvoiced", and the pitch parameter is indeterminate.

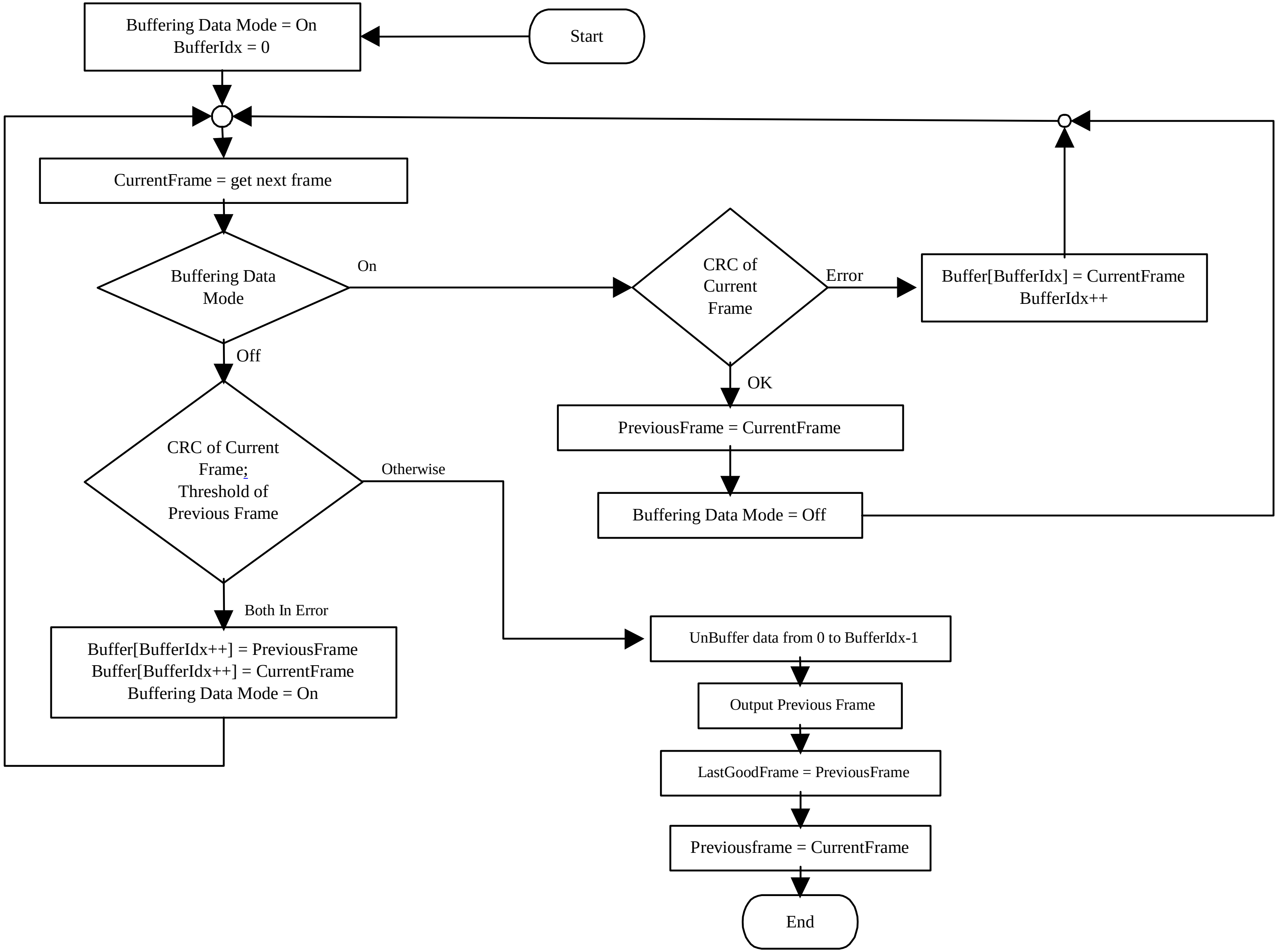

Start
Buffering Data Mode = On
BufferIdx = 0
CurrentFrame = get next frame
Buffering Data Mode
On
CRC of Current Frame
Error
Buffer[BufferIdx] = CurrentFrame
BufferIdx++
OK
PreviousFrame = CurrentFrame
Buffering Data Mode = Off
Off
CRC of Current Frame;
Threshold of Previous Frame
Otherwise
Both In Error
Buffer[BufferIdx++] = PreviousFrame
Buffer[BufferIdx++] = CurrentFrame
Buffering Data Mode = On
UnBuffer data from 0 to BufferIdx-1
Output Previous Frame
LastGoodFrame = PreviousFrame
Previousframe = CurrentFrame
End

**Figure 8.1: Error mitigation initialization flow chart**

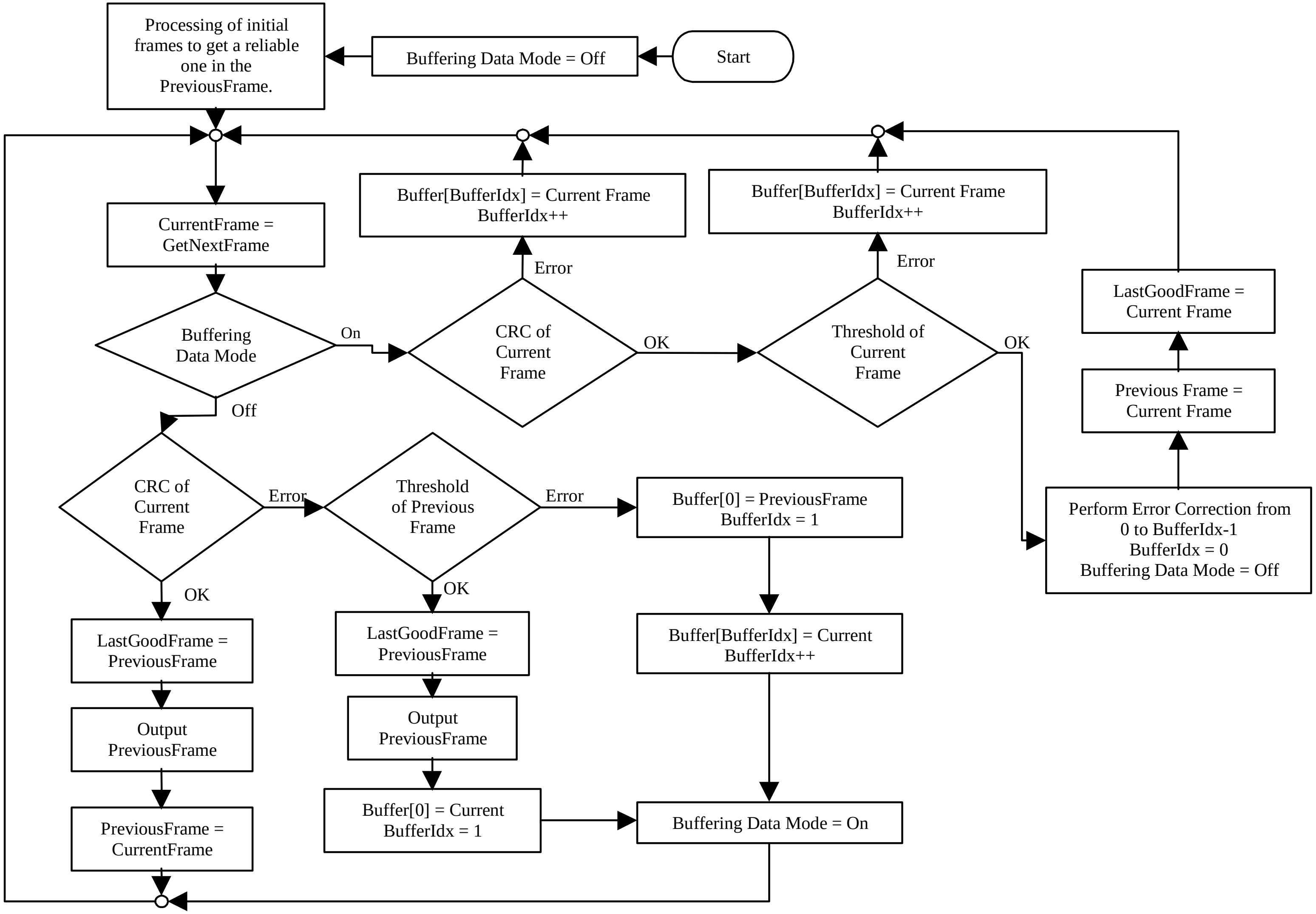

Start
Buffering Data Mode = Off
Processing of initial frames to get a reliable one in the PreviousFrame.
CurrentFrame = GetNextFrame
Buffering Data Mode
On
Off
CRC of Current Frame
Error
OK
Buffer[BufferIdx] = Current Frame
BufferIdx++
Threshold of Current Frame
Error
OK
Buffer[BufferIdx] = Current Frame
BufferIdx++
Perform Error Correction from 0 to BufferIdx-1
BufferIdx = 0
Buffering Data Mode = Off
Previous Frame = Current Frame
LastGoodFrame = Current Frame
CRC of Current Frame
Error
OK
Threshold of Previous Frame
Error
OK
LastGoodFrame = PreviousFrame
Output PreviousFrame
PreviousFrame = CurrentFrame
LastGoodFrame = PreviousFrame
Output PreviousFrame
Buffer[0] = Current
BufferIdx = 1
Buffer[0] = PreviousFrame
BufferIdx = 1
Buffer[BufferIdx] = Current
BufferIdx++
Buffering Data Mode = On

**Figure 8.2: Main error mitigation flow chart**

# 9 Server feature processing

$lnE$ and $c(0)$ combination, derivatives calculation and feature vector selection (FVS) processing are performed at the server side. $c(0)$, $c(1)$, …, $c(12)$, $lnE$ are received in the back-end. $c(0)$ is combined with $lnE$ then the first and second order derivatives of $c(1)$, …, $c(12)$, $lnE\,\&\,c(0)$ are calculated resulting in a 39 dimensional feature vector. A feature vector selection procedure is then performed according to the VAD information transmitted.

## 9.1 lnE and c(0) combination

$c(0)$ and $lnE$ are combined in the following way:

$$lnE\,\&\,c(0) = 0,6 \times c(0)/23 + 0,4 \times lnE \tag{9.1}$$

## 9.2 Derivatives calculation

First and second derivatives are computed on a 9-frame window. Velocity and acceleration components are computed according the following formulas:

$$\begin{aligned} vel(i,t) = &-1,0\times c(i,t-4) - 0,75\times c(i,t-3) - 0,50\times c(i,t-2) - 0,25\times c(i,t-1) \\ &+ 0,25\times c(i,t+1) + 0,50\times c(i,t+2) + 0,75\times c(i,t+3) + 1,0\times c(i,t+4), \\ &1 \le i \le 12 \end{aligned} \tag{9.2}$$

$$\begin{aligned} acc(i,t) = &\,1,0\times c(i,t-4) + 0,25\times c(i,t-3) - 0,285\,714\times c(i,t-2) \\ &- 0,607\,143\times c(i,t-1) - 0,714\,286\times c(i,t) - 0,607\,143\times c(i,t+1) \\ &- 0,285\,714\times c(i,t+2) + 0,25\times c(i,t+3) + 1,0\times c(i,t+4), \\ &1 \le i \le 12 \end{aligned} \tag{9.3}$$

where $t$ is the frame time index.

The same formulae are applied to obtain $lnE\,\&\,c(0)$ velocity and acceleration components.

## 9.3 Feature vector selection

A FVS algorithm is used to select the feature vectors that are sent to the recognizer. All the feature vectors are computed and the feature vectors that are sent to the back-end recognizer are those corresponding to speech frames, as detected by a VAD module (described in annex A).

# 10 Server side speech reconstruction

## 10.1 Introduction

This clause describes the server side speech reconstruction algorithm. Speech is reconstructed from feature vectors that have been decoded from the received bit stream and error-mitigated. Each feature vector consists of the following 16 parameters - 13 Mel-Frequency Cepstral Coefficients (MFCC) $C_0$ through $C_{12}$, the log-energy parameter log*E*, the pitch period value *P*, and the voicing class *VC*. The reconstructed speech is in digitized form and is provided at a sampling rate of 8 kHz regardless of the sampling rate of the input speech from which the feature vectors have been extracted.

The specification also covers a pitch tracking and smoothing algorithm, which is applied to the pitch (and class) parameters before they are used for speech reconstruction.

In clause 10, the following symbolic notations are used for some constants if not stated differently in the text:

N = 200 - frame length in samples;

M = 80 - frame shift in samples;

$f_s$ = 8 - sampling rate of synthesized speech signal in kHz;

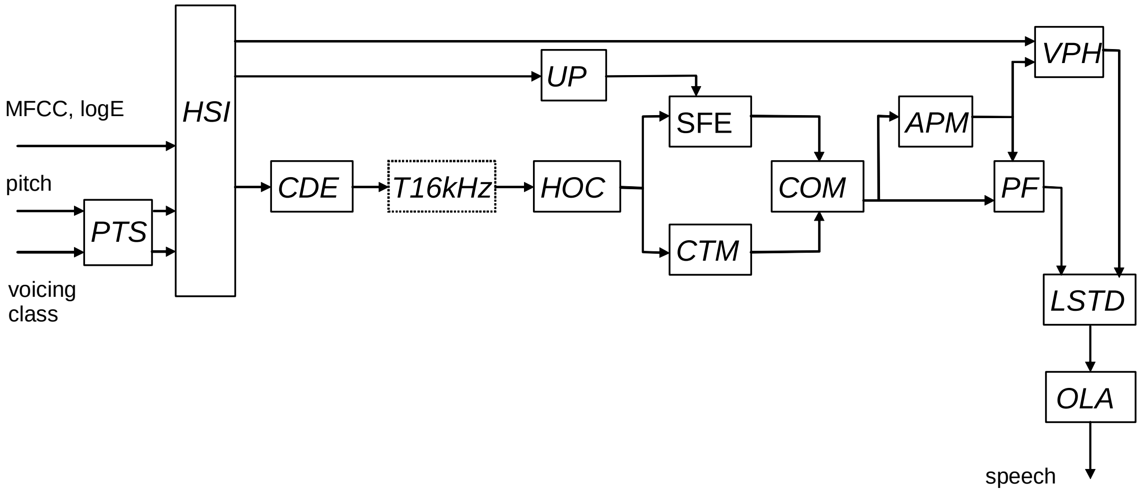
FFTL = 256 - FFT dimension.

## 10.2 Algorithm description

The reconstruction algorithm synthesizes one frame of speech signal from each MFCC vector and the corresponding log*E*, pitch and voicing class parameters. Frame synthesis is based on a harmonic model representation. The model parameters, viz., harmonic frequencies, magnitudes, and phases, are estimated for each frame and a complex spectrum (STFT) of the frame is computed. The complex spectrum is then transformed to time-domain representation and overlap-added with part of the speech signal already synthesized.

### 10.2.1 Speech reconstruction block diagram

Speech reconstruction block diagram is shown in figure 10.1.



| | |
|---|---|
| APM | All-Pole spectral envelope Modelling |
| CDE | Cepstra De-Equalization |
| COMB | Combined magnitudes estimate calculation |
| CTM | Cepstra To Magnitudes transformation |
| HOCR | High Order Cepstra Recovery |
| HSI | Harmonic Structure Initialization |
| LSTD | Line Spectrum to Time-Domain transformation |
| OLA | Overlap-add |
| PF | PostFiltering |
| PTS | Pitch Tracking and Smoothing |
| SFEQ | Solving Front-End eQuation |
| T16kHz | feature Transformation at 16kHz |
| UPH | Unvoiced Phase synthesis |
| VPH | Voiced Phase synthesis |

**Figure 10.1: Speech reconstruction block diagram**

## 10.2.2 Pitch Tracking and Smoothing

The input to the Pitch Tracking and Smoothing block (PTS) is a set of successive pitch period values $P[n]$, log energy values $logE[n]$ and voicing class values $VC[n]$. (Zero pitch period indicates either an unvoiced frame or non-speech frame.) The outputs are the corrected values $p_{fixed}\,[n]$ of pitch period and $vc_{fixed}\,[n]$ of voicing class.

Pitch processing is done in three stages. Then the voicing class value correction is performed.

The three stages of pitch processing require three working buffers to hold the pitch values of successive frames and possibly the log-energy of the frames (for the first stage only). Each stage introduces further delay (look-ahead) in the output pitch value. The buffer length $L$ (an integer number of frames) is the sum of the number of look-ahead frames (the delay) $D$, the number of backward frames (the history) $H$, plus one, which is the current output frame at that stage (i.e. $L=D+H+1$). Each stage produces a new output value, which is pushed at the top (at the end) of the next stage buffer. All other values in the buffer are pushed one frame backwards, with the oldest value discarded. This configuration is described in figure 10.2.

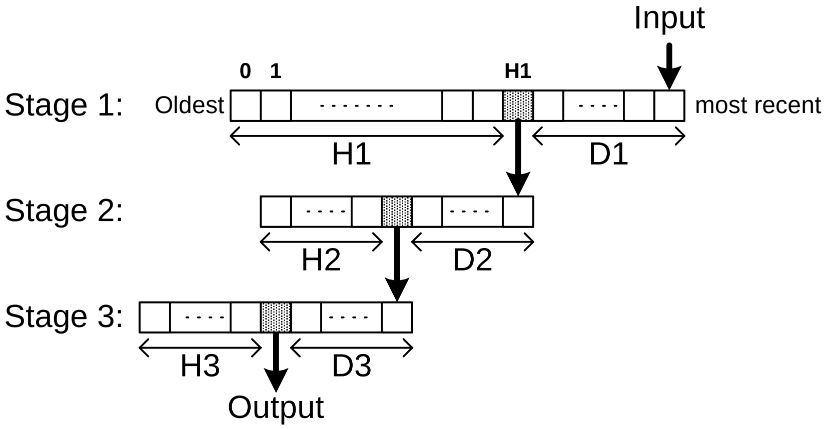


**Figure 10.2: Buffers of the three-stage pitch tracking and smoothing algorithm**

The total look-ahead (in frames) required for the correction of current pitch value, and therefore the delay introduced by the PTS block is: *D = D1 + D2 + D3*. The delay and history values used are:

**First stage: D1 = 8, H1 = 10 (therefore L1 = 19);**

**Second stage: D2 = H2 = 1 (therefore L2 = 3);**

**Third stage: D3 = H3 = 2 (therefore L3 = 5).**

And the total delay is **11 frames**.

All the three stage buffers are initialized by zero values. Each coordinate of the energy buffer used at the first stage is initialized by -50.

In the description of the three-stage pitch tracking algorithm the terms "voiced frame" and "unvoiced frame" are redefined. A frame is referred to as voiced frame if it is either of "*fully voiced*" or of "*mixed-voiced*" class. A frame is referred to as unvoiced if it is of "*unvoiced*" or "*non-speech*" class.

### 10.2.2.1 First stage - gross pitch error correction

Let *p[n], n=0,1,...,L1-1* be the pitch period values of the *first stage buffer*, such that *p[L1-1]* is the most recent value (the new input pitch), and *p[0]* is the oldest value. A pitch value of zero indicates an unvoiced frame. Similarly, there is a buffer of the same length holding the energy values.

The output pitch of the first stage has a delay of *D1* frames compared to the most recent frame in the buffer. The processed frame has *D1* frames look-ahead and *H1* backwards frames. A new pitch value $P_{out}$ associated with the location *n=H1* in the buffer has to be calculated and pushed to the second stage pitch tracking.

If the frame is unvoiced (i.e. *p[H1]==0*) then $P_{out}=0$ as well.

If the frame is voiced, but there are unvoiced frame at both sides (i.e. *p[H1]!=0, p[H1-1]==p[H1+1]==0*), then $P_{out}=0$.

If the frame is voiced, and is a member of a voiced segment of only two frames, then the similarity between the pitch values of the two voiced frames is examined as described below. If they are *similar*, then no change is made to the pitch value, i.e. $P_{out}=p[H1]$. Otherwise, the frame is reclassified as unvoiced, $P_{out}=0$.

In the remaining cases, the output pitch value $P_{out}$ will be assigned the value *p[H1]*, or it may be assigned an integer multiplication or integer divide of *p[H1]*. To do this, first the voiced segment in which the frame *H1* is located in is identified. This voiced segment can extend *D1* frames ahead and *H1* frames backwards at the most. It will be shorter if there are unvoiced frames in the buffer. Then, a *reference* pitch value is extracted using the information from the neighbouring frames in the voiced segment. Finally, the output pitch value of the first stage is identified.

**Similarity measure**

Two (positive) pitch periods $P_1$ and $P_2$ are declared as similar if for a given *similarity factor* $\rho > 1$ the following is true:

$$\rho \times P_1 \geq P_2 \geq P_1 / \rho$$

A similarity factor of 1,28 is used to check the similarity of two pitch periods of successive frames (i.e. 10 ms apart). A factor of 1,4 is used for pitch periods that are two frames apart (20 ms).

**Relevant frames identification**

The voiced segment in which the current frame (in position *H1*) is located and its pitch and energy values are copied to a temporary buffer. The pitch values of this segment are notified by *q[n], n = 0, 1, …, N - 1* and the corresponding log-energy values as *e[n], n = 0, 1, …, N - 1.* Here *N* is the number of frames in the voiced segment. (Note that $2 < N \leq L1$). Figure 10.3 describes the indexing of the voiced segment. "U" represents an unvoiced frame, and "V" a voiced frame. Location *K* in the voiced segment now represents the current examined frame (*p[H1]*, for which a first stage output pitch value must be calculated):

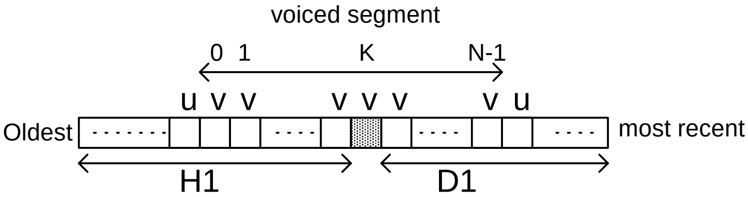


**Figure 10.3: Location of a voiced segment within the first stage buffer**

The purpose of the following process is to identify the set of frames that have *similar* pitch values, and their total energy is the greatest. To do that, the *N* pitch values are sorted according to ascending pitch values. The sorted pitch values are then divided into groups. A group contains one or more consecutive sorted pitch periods, such that neighbouring pitch values are *similar* (with the similarity factor 1,28) in the sense defined above. The pitch values are processed from the smallest to the largest. When the similarity is violated between the consecutive sorted pitch values, the previous group is closed and a new group is opened.

For each group, the total energy of all frames in the group is calculated. The group that has the biggest total energy is selected. All other frames that are not within the selected group are marked as deleted in the original (unsorted) voiced segment temporary buffer *q*.

**Reference pitch value calculation**

One or more pitch tracks are identified in the voiced segment (represented by the buffers *q* and *e*). The tracking is done only on the frames that were not deleted by the relevant frames identification process. If frame *K* (examined frame of the stage 1) was not deleted, it will be included in one of the pitch tracks. A pitch track is defined as a set of successive undeleted voiced frames, whose neighbouring pitch values are *similar* in the above specified sense. The energy of each pitch track is the sum of the log-energy of all its frames

After all the pitch tracks are identified, the one with the biggest energy is examined. The *reference* pitch $P_{ref}$ is defined as the pitch value in the selected track that is closest to position *K*. If the selected pitch track includes frame *K*, it means that the reference pitch is exactly the pitch value of the examined frame (meaning it will not change at the first stage of processing).

**First stage output calculation**

Let $p_1$ and $p_2$ be two positive numbers. We define the distance measure *Dist($p_1$,$p_2$)* in the following way:

$$Dist(p_1, p_2) = \left| \frac{p_1 - p_2}{p_1 + p_2} \right|$$

Given a reference pitch value $P_{ref}$ and the pitch value of the current examined frame *p[H1]*, the new pitch value $P_{out}$ is calculated as specified by the following pseudo code:

```
INTEGER SCALING
{
    $if\ (P_{ref} == p[H1])$
        $P_{out} == p[H1])$ ;
$elseif\ (P_{ref} > p[H1])$
    {
        $Q = ceil(P_{ref} / p[H1])$ ;
        $M = \arg\min_{m=1,...,Q} Dist(P_{ref}, m \times p[H1])$ ;
        $P_{out} = M \times p[H1]$ ;
    }
$else$
    {
        $Q = ceil(p[H1]/P_{ref})$ ;
        $M = \arg\min_{m=1,...,Q} Dist(m \times P_{ref}, p[H1])$ ;
        $P_{out} = p[H1]/M$ ;
    }
    $if\ (M == 2)$
    {
        $if\ (P_{ref} > p[H1])$
        {
            $if\ (1{,}4 \times Dist(P_{ref}, 2p[H1]) > Dist(P_{ref}, p[H1]))$
            $P_{out} = p[H1]$ ;
        }
    $if\ (P_{ref} < p[H1])$
        {
            $if\ (1{,}4 \times Dist(2P_{ref}, p[H1]) > Dist(P_{ref}, p[H1]))$
            $P_{out} = p[H1]$ ;
        }
    }
}
```

### 10.2.2.2 Second stage - voiced/unvoiced decision and other corrections

Let *p[n], n = 0, 1, 2 (L2 = 3)* be the pitch period values of the *second stage buffer*, such that *p[2]* is the most recent value (the new output of the first stage), and *p[0]* is the oldest value. An output value will be associated with the middle location *n=1* in the buffer, and will be marked $P_{out}$.

$P_{out}$ will be assigned the value of *p[1]*, unless one of the following occurs:

If all three frames are voiced, and *p[2]* is *similar* to *p[0]*, then we examine the middle value *p[1]*. If it is not *similar* (with $\rho = 1{,}28$) to the average of *p[2]* and *p[0]*, the output value $P_{out}$ will receive this average value instead of *p[1]*.

If *p[0]* and *p[2]* are voiced and *similar*, and if *p[1]* is unvoiced, then the output frame will be voiced with a pitch $P_{out}$ equal to average of *p[0]* and *p[2]*. Here the similarity is evaluated using a similarity factor of $\rho = 1{,}28$ instead of 1,4, even though the pitch values to be compared are two frames apart.

1) If the oldest frame in the buffer is unvoiced (*p[0]==0*) and the two other frames are voiced, or if the most recent frame is unvoiced (*p[2]==0*) and the two other frames are voiced, then the *similarity* between the two voiced frames is examined. If they are not *similar*, then the output frame will be unvoiced, i.e. $P_{out}=0$.

### 10.2.2.3 Third stage - smoothing

Let *p[n], n = 0, 1, ..., L3 - 1* be the pitch period values of the *third stage buffer*, such that *p[L3-1]* is the most recent value (the new output of the second stage), and *p[0]* is the oldest value. *L3* is odd. An output value will be associated with the middle location *(L3-1)/2* in the buffer, and will be marked $p_{fixed}$.

If there is an unvoiced frame in the middle location (i.e. *p[(L3-1)/2]==0*) then the output frame is also unvoiced and $p_{fixed}=0$. Otherwise, a filtering operation is performed by weighting a modified version of all the pitch values in the buffer as described below.

A new set of pitch values *q[n], n = 0, 1, …, L3 - 1* is derived from the current values *p[n]* in the third stage buffer, according to the following rules:

1) *q[(L3-1)/2] = p[(L3-1)/2].*

2) For each n, if *p[n]==0 (unvoiced frame)* then *q[n] = p[(L3-1)/2].*

3) All other pitch values are multiplied by an integer or divided by an integer, such that they become as close as possible to the value of the middle frame *p[(L3-1)/2]*. That is, *q[n] = M×p[n]* or *q[n]=p[n]/M* where M is an integer greater or equal one. The exact calculation of the new value is done as is described by the pseudo code titled INTEGER SCALING in the clause 10.2.2.1 above wherein the variables substitution should be done as: $P_{ref}$ *by p[(L3-1)/2], p[H1] by p[n], and* $P_{out}$ *by q[n].*

The final output pitch is calculated in the following way:

$$p_{fixed} = \sum_{n=0}^{L3-1} q[n] \times h[n]$$

where:

*h[0]=1/9, h[1]=2/9, h[2]=3/9, h[3]=2/9, h[4]=1/9,* (L3 = 5).

### 10.2.2.4 Voicing class correction

The input for the voicing class correction are three voicing class values *VC[n-1], VC[n]* and *VC[n+1]* associated with three consecutive frames, and pitch values before and after the tracking procedure associated with the middle frame *n* and marked as *P* and $p_{fixed}$ correspondingly. The output of this processing step is a corrected voicing class value $vc_{fixed}$ associated with the middle frame *n*. *VC[n-1]* is initialized by zero when the very first frame is processed. The processing is described by the following pseudo code:

```
{
    if (VC[n-1]=="mixed-voiced" AND VC[n]=="fully-voiced" AND VC[n+1] != "fully-voiced")
        $vc_{fixed}$ = "mixed-voiced";
    else
        $vc_{fixed}$ = VC[n];

    if (P == 0  AND  $p_{fixed}$ != 0)
        $vc_{fixed}$ = "mixed-voiced";
    elseif (P != 0  AND  pfixed == 0)
        $vc_{fixed}$ = "unvoiced";
}
```

## 10.2.3 Harmonic Structure Initialization

Inputs for the Harmonic Structure Initialization (HSI) block are the pitch value $p=p_{fixed}$ and the voicing class value $vc_{fixed}$ corresponding to the current frame being synthesized. The HSI block produces modified values of the input parameters and array(s) of harmonic-elements.

The reconstruction algorithm treats non-speech frames and unvoiced frames in the same way. Consequently the voicing class value is modified as:

```
if (vc_fixed == "non-speech")                                        (10.1)
    vc = "unvoiced";
else
    vc = vc_fixed;
```

The modified voicing class *vc* has one of the three possible values: "*fully-voiced*", "*mixed-voiced*", and "*unvoiced*". Accordingly we refer to the frame being synthesized as fully-voiced, mixed-voiced or unvoiced.

For a fully-voiced frame an array $VH = \{H_k, k=1,\ldots,N_v\}$ of harmonics is allocated. Each harmonic $H_k = (f_k, A_k, \varphi_k)$ is represented by a normalized frequency $f_k$, magnitude $A_k$ and phase $\varphi_k$ values. The number of harmonics $N_v$ is:

$$N_v = floor(p/2) \qquad (10.2)$$

The normalized frequency $f_k$ associated with $k$-th harmonic is set to:

$$f_k = k/p \qquad (10.3)$$

For an unvoiced frame an array $UH = \{H_k, k=1,\ldots,N_u\}$ of harmonics is allocated. The number of harmonics $N_u$ is:

$$N_u = FFTL/2 - 1 \qquad (10.4)$$

The normalized frequency associated with k-th harmonic is set to:

$$f_k = k/FFTL \qquad (10.5)$$

For a mixed-voiced frame both *VH* and *UH* arrays are allocated.

The *HSI* block does not set values of the harmonic magnitudes and phases. This is a subject of the further processing.

The elements of the *VH*-array will be henceforth referred to as voiced harmonics, and the elements of the *VU*-array as unvoiced harmonics.

## 10.2.4 Unvoiced phase synthesis

The input for the Unvoiced Phase synthesis (UPH) block is the UH array of unvoiced harmonics. Thus the block is entered only if the *vc*_variable value is either "unvoiced" or "mixed-voiced". The block sets phase values $\{\varphi_k, k=1,\ldots,N_u\}$ associated with the array elements (unvoiced harmonics). The phase values are obtained by a generator of pseudo random uniformly distributed numbers, and they are scaled to fit into the interval [0π, 2π]. A new vector of phase values is generated each time the UPH block is entered.

## 10.2.5 Cepstra de-equalization

This block inverts the blind equalization transform (clause 5.4) performed at front-end. Twelve cepstra coefficients $C_k$, k=1, …, 12, are modified as described by the pseudo code shown below:

```
weightingPar = min(1,max(0, logE - 211/64));
stepSize = 0,0087890625 weightingPar;
new_bias(i) = 0,999 bias(i) + stepSize (C_i - RefCep(i)), i=1,…,12;
```

*$C_i$ += bias(i), i=1,…,12;*

*bias(i) = new_bias(i), i=1,…,12.*

where *logE* is log-energy value of the current frame from the decoded feature vector; *bias* and *RefCep* vectors are initialized as described in clause 5.4.

## 10.2.6 Transformation of features extracted at 16 kHz

This processing step is performed only if the features have been extracted from the input speech sampled at 16 kHz. The function of this block is to convert the features (cepstra and log-energy) to the ones representing [0 kHz, 4 kHz] frequency band corresponding to the sampling rate of 8 kHz.

First, the vector of cepstra coefficients undergoes 26-dimensional IDCT:

$$val_k = \frac{2}{26}\sum_{n=0}^{12} C_n \times \cos\left(\frac{\pi}{26}\times n \times (k-0{,}5)\right), k = 1,...,26 \qquad (10.6)$$

Then the first 23 obtained values are used to produce a modified cepstra vector by means of 23-dimensional DCT:

$$C_k = \sum_{i=1}^{23} val_i \cos\left(\frac{\pi\times k}{23}\times(i-0{,}5)\right),\ k = 0,...,12 \qquad (10.7)$$

Finally the last three values $val_{24}$, $val_{25}$ and $val_{26}$ are used to modify the log-energy as specified below:

```
{
    del = ln(1.9);
    E = exp(logE);
    fixE = exp(val24- del)  + exp(val25- del) + exp(val26- del);
    if (E > fixE)
    {
        E = E - fixE;
        logE =  max(-50, ln(E));
    }
}
```

## 10.2.7 Harmonic magnitudes reconstruction

Harmonic magnitudes reconstruction is done in three major steps. An estimate $A^E$ of the magnitudes vector is obtained in the *SFEQ* block. Another estimate $A^I$ of the magnitudes vector is obtained in the *CTM* block. Then a final estimate $A$ is calculated in the *COMB* block by combining $A^E$ with $A^I$.

### 10.2.7.1 High order cepstra recovery

The harmonic magnitudes are estimated from the Mel-Frequency Cepstral Coefficients (MFCC) and the pitch period value (clauses 10.2.7.2 to 10.2.7.4). At the front-end, only 13 of the 23 possible MFCC's are computed (clause 5.3.7), compressed, and transmitted to the back-end. The remaining 10 values, $C_{13}$ through $C_{22}$, referred to as high order cepstra here, are simply discarded, i.e. not computed. Clearly, if these missing values are available, the harmonic magnitudes can be estimated more accurately. The HOCR block attempts to at least partially recover the missing high order cepstral information for voiced frames (both mixed and fully voiced). This recovery process continues further within the Solving Front-Equation (SFEQ) block as described below in clause 10.2.7.2. For unvoiced frames, the high order cepstra are not recovered.

The recovery of high order cepstra is achieved through lookup table (table 10.1) using the pitch period as a parameter. Table 10.1 was generated by analysing a large speech database and computing the average value of (uncompressed) high order cepstra over all frames with pitch values falling in the appropriate range.

### Table 10.1: High order cepstra for different pitch ranges

| Pitch range | $C_{13}$ thru $C_{22}$ | Pitch range | $C_{13}$ thru $C_{22}$ | Pitch range | $C_{13}$ thru $C_{22}$ | Pitch range | $C_{13}$ thru $C_{22}$ |
|---|---|---|---|---|---|---|---|
| p ≤ 26 | -5,111350E-01 | 38 < p ≤ 39 | -1,031216E+00 | 50 < p ≤ 54 | 4,467755E-01 | 71 < p ≤ 72 | -7,373724E-01 |
| | -1,682880E+00 | | -1,387326E+00 | | -2,535201E-01 | | -1,685859E+00 |
| | -3,716587E-01 | | -1,014192E+00 | | 7,538735E-01 | | -3,222678E-01 |
| | -7,956616E-01 | | -1,288828E+00 | | 5,603248E-01 | | -9,107897E-01 |
| | -7,253695E-03 | | -1,319227E+00 | | 7,922218E-01 | | 2,935433E-01 |
| | -5,274537E-01 | | -1,078165E+00 | | 3,434679E-01 | | -5,313740E-01 |
| | 9,280691E-04 | | -3,695266E-01 | | 4,104464E-01 | | 4,481341E-01 |
| | -2,563041E-01 | | -1,856345E-01 | | -1,230457E-01 | | -2,842423E-01 |
| | -1,049254E-01 | | 4,743951E-01 | | -1,280315E-01 | | -1,526781E-01 |
| | -9,817168E-02 | | 5,453367E-01 | | -1,211750E-01 | | -2,500107E-01 |
| 26 < p ≤ 32 | -1,323581E+00 | 39 < p ≤ 40 | -7,697338E-01 | 54 < p ≤ 62 | 6,339330E-02 | 72 < p ≤ 74 | -6,542689E-01 |
| | -1,247226E+00 | | -1,251034E+00 | | -7,212541E-01 | | -1,688334E+00 |
| | 8,918094E-01 | | -1,135184E+00 | | 5,986097E-01 | | -1,748565E-01 |
| | 6,301045E-01 | | -1,052677E+00 | | 1,459474E-01 | | -9,630367E-01 |
| | 2,640953E-01 | | -1,081295E+00 | | 6,876847E-01 | | 2,920569E-01 |
| | -6,120602E-01 | | -1,276117E+00 | | -4,344984E-02 | | -6,694176E-01 |
| | -1,029995E+00 | | -8,835811E-01 | | 2,450704E-01 | | 3,618038E-01 |
| | -1,210108E+00 | | -4,264293E-01 | | -1,760258E-01 | | -2,193661E-01 |
| | -7,136748E-01 | | 2,759056E-01 | | -3,539870E-03 | | -8,691479E-02 |
| | -2,458055E-01 | | 3,279340E-01 | | -7,837202E-02 | | -1,523485E-01 |
| 32 < p ≤ 34 | -3,166838E+00 | 40 < p <41 | -2,970808E-01 | 62 < p ≤ 66 | -2,380725E-01 | 74 < p ≤ 76 | -3,450635E-01 |
| | -3,976374E+00 | | -1,177779E+00 | | -1,641640E+00 | | -1,905594E+00 |
| | -2,099192E+00 | | -7,915491E-01 | | 1,450078E-01 | | -6,137879E-02 |
| | -5,804268E-01 | | -1,044372E+00 | | -7,527372E-01 | | -1,113471E+00 |
| | 4,614631E-01 | | -8,211824E-01 | | 3,593675E-01 | | 2,747527E-01 |
| | 4,824880E-01 | | -1,355624E+00 | | -4,426172E-01 | | -6,160255E-01 |
| | 7,639357E-01 | | -1,054223E+00 | | 1,779412E-02 | | 1,056195E-01 |
| | -3,386363E-02 | | -6,738636E-01 | | -2,862400E-01 | | -2,321364E-01 |
| | -6,201262E-01 | | 1,521423E-02 | | -7,476118E-02 | | -3,847001E-02 |
| | -7,372425E-01 | | 9,342021E-02 | | -5,290803E-02 | | -9,724520E-02 |
| 34 < p ≤ 35 | -3,018169E+00 | 41 < p ≤ 42 | -1,576688E-01 | 66 < p ≤ 68 | -3,377764E-01 | 76 < p ≤ 77 | 2,806036E-02 |
| | -3,911408E+00 | | -1,062970E+00 | | -2,151872E+00 | | -2,085802E+00 |
| | -2,720349E+00 | | -6,441808E-01 | | -1,180943E-01 | | -4,639831E-02 |
| | -1,107410E+00 | | -6,141125E-01 | | -1,035271E+00 | | -1,303672E+00 |
| | 2,002102E-01 | | -7,753426E-01 | | 3,817170E-01 | | 1,851366E-01 |
| | 7,917436E- | | -1,160622E+00 | | -5,135021E-01 | | -6,901463E-01 |
| | 011,441889E+00 | | -1,042945E+00 | | 2,217322E-01 | | -9,140391E-03 |
| | 7,677763E-01 | | -7,988926E-01 | | -2,720239E-01 | | -2,332839E-01 |
| | -3,245252E-02 | | -3,823192E-01 | | -1,189329E-01 | | -9,564089E-02 |
| | -7,143410E-01 | | -1,765679E-01 | | -1,244790E-01 | | -1,168974E-01 |
| 35 < p ≤ 36 | -2,260784E+00 | 42 < p ≤ 43 | -2,594792E-01 | 68 < p ≤ 69 | -1,775208E-01 | 77 < p ≤ 78 | 8,053611E-02 |
| | -3,289034E+00 | | -9,725035E-01 | | -2,086558E+00 | | -2,152415E+00 |
| | -2,556978E+00 | | -4,955449E-01 | | -2,195775E-01 | | 7,933393E-02 |
| | -1,653956E+00 | | -3,837078E-01 | | -9,837000E-01 | | -1,489653E+00 |
| | -1,588058E-01 | | -5,113737E-01 | | 3,482551E-01 | | 2,179069E-01 |
| | 3,966002E-01 | | -1,020689E+00 | | -4,620659E-01 | | -8,265848E-01 |
| | 1,494472E+00 | | -8,800513E-01 | | 2,664061E-01 | | -5,724430E-02 |
| | 8,604176E-01 | | -9,256434E-01 | | -2,996481E-01 | | -2,088230E-01 |
| | 1,893507E-01 | | -5,710840E-01 | | -9,481932E-02 | | -9,954191E-02 |
| | -3,483856E-01 | | -2,608341E-01 | | -1,516739E-01 | | -8,906914E-02 |
| 36 < p ≤ 37 | -1,802585E+00 | 43 < p ≤ 46 | 1,150858E-01 | 69 < p ≤ 70 | -1,539969E-01 | 78 < p ≤ 79 | 7,484316E-02 |
| | -2,144211E+00 | | -6,361938E-01 | | -1,986363E+00 | | -2,018542E+00 |
| | -2,228024E+00 | | 2,567051E-01 | | -3,533201E-01 | | -5,265641E-02 |
| | -1,802318E+00 | | -2,648086E-01 | | -9,162003E-01 | | -1,365789E+00 |
| | -1,032504E+00 | | -4,371306E-01 | | 3,157739E-01 | | 2,166845E-01 |
| | 5,535706E-03 | | -1,010725E+00 | | -3,801906E-01 | | -9,570920E-01 |
| | 9,357433E-01 | | -7,759937E-01 | | 2,569408E-01 | | -1,540541E-01 |
| | 6,810726E-01 | | -6,455466E-01 | | -2,515628E-01 | | -2,568645E-01 |
| | 3,568225E-01 | | -2,855171E-01 | | -1,431256E-01 | | -7,194232E-02 |

| Pitch range | $C_{13}$ thru $C_{22}$ | Pitch range | $C_{13}$ thru $C_{22}$ | Pitch range | $C_{13}$ thru $C_{22}$ | Pitch range | $C_{13}$ thru $C_{22}$ |
|---|---|---|---|---|---|---|---|
| | 1,610291E-01 | | -7,813629E-02 | | -2,086413E-01 | | -2,474382E-02 |
| $37 < p \le 38$ | -1,227172E+00 | $46 < p \le 50$ | 5,119228E-01 | $70 < p \le 71$ | -3,404706E-01 | $79 < p \le 80$ | -1,306538E-01 |
| | -1,603199E+00 | | -3,679310E-01 | | -1,925969E+00 | | -1,829025E+00 |
| | -1,504956E+00 | | 6,489079E-01 | | -3,744814E-01 | | -7,194354E-02 |
| | -1,772818E+00 | | 1,279952E-01 | | -8,535586E-01 | | -1,013687E+00 |
| | -1,395420E+00 | | 2,239187E-01 | | 2,496247E-01 | | 2,636875E-01 |
| | -6,263873E-01 | | -3,094574E-01 | | -4,021760E-01 | | -6,979883E-01 |
| | 3,036422E-01 | | -2,643344E-01 | | 3,560743E-01 | | -1,266116E-01 |
| | 1,871070E-01 | | -4,557250E-01 | | -2,202438E-01 | | -2,186688E-01 |
| | 4,406141E-01 | | -2,296919E-01 | | -1,582776E-01 | | -9,609150E-02 |
| | 5,066580E-01 | | -1,546537E-01 | | -2,290248E-01 | | -1,777760E-02 |
| | | | | | | $80 < p$ | -5,111350E-01 |
| | | | | | | | -1,682880E+00 |
| | | | | | | | -3,716587E-01 |
| | | | | | | | -7,956616E-01 |
| | | | | | | | -7,253695E-03 |
| | | | | | | | -5,274537E-01 |
| | | | | | | | 9,280691E-04 |
| | | | | | | | -2,563041E-01 |
| | | | | | | | -1,049254E-01 |
| | | | | | | | -9,817168E-02 |

### 10.2.7.2 Solving front-end equation

The inputs for the SFEQ block are the MFCC vector $C$, an array $HA=\{H_k, k=1,\ldots,N_h\}$ of harmonics and a boolean indicator *voiced_flag*. If current frame is of fully-voiced class then $VH$ array is fed into the block ($HA=VH$) and the indicator is set to *voiced_flag = TRUE*. If current frame is of unvoiced class then $UH$ array is passed to the block ($HA=UH$) and the indicator is set to *voiced_flag = FALSE*. If the frame is of mixed-voiced class then the block is entered twice, one time with ($HA=VH$, *voiced_flag=TRUE* ) and another time with ($HA=UH$, *voiced_flag=FALSE* ). The SFEQ block outputs an estimate $A^E = \{A_k^E, k=1,\ldots,N_h\}$ of harmonic magnitudes.

A sequence of processing steps is carried out as described below.

**Step 1. Original bins calculation**

23-dimensional Inverse Discrete Cosine Transform (IDCT) followed by the exponent operation is applied to the low order cepstra vector $LOC = \{C_k, k=0,\ldots,12\}$ resulting in an *original bins vector* $B^{org} = \{b_k^{org}, k=1,\ldots,23\}$

$$b_k^{org} = \exp\left(\frac{2}{23}\sum_{n=0}^{12} C_n \times \cos\left(\frac{\pi}{23}\times n \times (k-0{,}5)\right)\right) \qquad (10.8)$$

If the features have been extracted from the input speech signal sampled at 16 kHz the original bins are modified as follows:

$$b_k^{org} = \sqrt{b_k^{org} \times MFS_k},\ k = 1,\ldots,23 \qquad (10.9)$$

where $MFS_k$ is a sum of the weights of k-th Mel-filter given by (5.58), (5.59).

**Step 2. Basis vectors calculation**

For each harmonic, the (normalized) frequency $f_k$ value is converted to the nearest FFT index $fidx_k$

$$fidx_k = round(f_k \times FFTL), k = 1, N_h \qquad (10.10)$$

A binary *grid* vector $G = \{g_n, n = 0,...,FFTL/2\}$ is computed in two steps:

1) $$g_n = 0,\ n=0,\ldots,FFTL/2 \qquad (10.11a)$$

2) $$g_{fidx_k} = 1, k = 1,..., N_h \qquad (10.11b)$$

23 prototype basis vectors PBV$_k$, k=1,23, are calculated. A prototype basis vector $PBV_k = \{pbv_i^k, i = 0,..., FFTL/2\}$ is derived from the triangular weighting window associated with k-th frequency channel of the Mel-filters bank given by (5.58), (5.59), clause 5.3.5.

$$pbv_i^k = g_i \times \left(0{,}4 \times \mu_i^k + 0{,}6 \times {\mu_i^k}^2\right)$$
$$\text{where } \mu_i^k = \begin{cases} 0,\ if\ \ i < cbin_{k-1}\ and\ i > cbin_{k+1} \\ \dfrac{i - cbin_{k-1} + 1}{cbin_k - cbin_{k-1} + 1},\ if\ \ cbin_{k-1} \le i \le cbin_k \\ 1 - \dfrac{i - cbin_k}{cbin_{k+1} - cbin_k + 1},\ if\ \ cbin_k + 1 \le i \le cbin_{k+1} \end{cases} \qquad (10.12)$$

$cbin_k$ is a shortcut notation for $bin_{center}(k)$ given by (5.57).

(Note that in *k*-th prototype basis vector only coordinates $pbv_{fidx_n}^k, n = 1,..., N_h$ may have non-zero values.) A *basis vector* $BV_k = \{bv, n = 1,..., N_h\}$ is derived from each prototype basis vector $PBV_k$ by selecting only those coordinates having the indexes $fidx_n$ as follows:

$$BV_k = \{bv_n^k = pbv_{fidx_n}^k, n = 1,..., N_h\},\ k = 1,...,23 \qquad (10.13)$$

**Step 3. Basis bin vectors and matrix calculation**

Each basis vector $BV_k$ is converted to a (in general) complex valued vector $LS_k = \{ls_i^k, i = 0,..., N_h\}$ as specified by the following pseudo code:

```
{
$LS_k = BV_k;$
 if (voiced_flag == FALSE)
     $ls_n^k = bv_n^k \times \exp(j \times \varphi_n) \times peph(f_n)), n = 1,..., N_h$ ;
}
```

where:

$\varphi_n$ is a phase associated with n-th unvoiced harmonic as described in clause 10.2.4; and

*peph* is phase frequency characteristic of the preemphasis operator:

$$peph(f) = \frac{1 - PE\cos(2\pi \times f) + j \times PE\sin(2\pi \times f)}{\sqrt{1 - 2PE\cos(2\pi \times f) + PE^2}},\quad PE = 0{,}9 \qquad (10.14)$$

Note that if *voiced_flag* is TRUE the coordinates of the *LS*-vectors have real values.

Each $LS_k$ vector is further converted to a synthetic magnitude spectrum vector $SM_k = \{sm_i^k, i = 0,..., FFTL/2 - 1\}$ by convolution with Fourier transformed Hamming window function followed by absolute value operation as follows:

$$sm_i^k = \left| \sum_{n=1}^{N_h} ls_n^k \times HWT(f_n - i/FFTL) \right| \qquad (10.15)$$

where:

$$HWT(f)=\begin{cases}0{,}54\Delta(f)+0{,}23\times\left[\Delta\left(f-\dfrac{1}{N-1}\right)+\Delta\left(f+\dfrac{1}{N-1}\right)\right], & if\ \ |f|\le WT\_BW,\\ 0,\ if\ \ |f|>WT\_BW\end{cases} \quad (10.16)$$

$$\Delta(f)=\begin{cases}0,\ if\ \ f=0\\ \sin(\pi\times f\times N)/\sin(\pi\times f)\end{cases} \quad (10.17)$$

$$WT\_BW=100/8\,000 \quad (10.18)$$

$N$ =200 is frame length.

Mel-filtering operation given by formula (5.60) is applied to each synthetic magnitude spectrum vector $SM_k$, (in (5.60) $P_{swi}(i)$ is substituted by $sm_i^k$), and a 23-dimensional *basis bins vector* $BB_k=\{bb_i^k, i=1,...,23\}^{\mathrm{T}}$ is obtained. We see the basis bins vectors as column vectors.

A 23-by-23 *basis bins matrix BB* which has the vectors $\mathrm{BB_k}$ as its columns is constructed:

$$BB=\begin{bmatrix}BB_1 & BB_2 & ... & BB_{23}\end{bmatrix} \quad (10.19)$$

**Step 4. Equation matrix calculation**

A 23-by-23 symmetric *equation matrix EM* is computed as follows:

$$EM\ =BB^T\times BB+0{,}001\times\lambda\times E \quad (10.20)$$

where $\lambda=sum(diag(BB^T\times BB))/23$ is an average of the main diagonal elements of the matrix $BB^TBB$ and $E$ is unit 23 by 23 matrix.

In order to reduce the computational complexity of the further processing in the reference implementation, the LU-decomposition is applied to the equation matrix $EM$, and the LU representation is stored.

**Step 5. Initialization of iterative process**

Iteration counter is set:

$$it\_count = 1$$

**Step 6. High bins calculation**

This step is carried out only if *voiced_flag* = TRUE, and is skipped otherwise.

23-dimensional IDCT followed by the exponent operation is applied to the high order cepstra vector $HOC = \{C_k, k=13,\ldots,22\}$ output from the HOCR block. The transform results in a *high bins vector* $B^{high}=\{b_k^{high}, k=1,...,23\}$

$$b_k^{high}=\exp\left(\frac{2}{23}\sum_{n=13}^{22}C_n\times\cos\left(\frac{\pi}{23}\times n\times(k-0{,}5)\right)\right) \quad (10.21)$$

If the features have been extracted from an input speech signal sampled at 16 kHz and the first iteration is being performed (*it_count ==1*) then the transform given by 10.9 (Step 1) is applied to the high bin values.

**Step 7. Reference bins calculation**

A 23-dimensional reference bins vector $B^{ref} = \{b_k^{ref}, k = 1,...,23\}$ is computed as follows:

```
if (voiced_flag == TRUE)
{
    /* coordinatewise multiplication of $B^{org}$ and $B^{high}$ vectors */
    $b_k^{ref} = b_k^{hig} \times b_k^{org}, k = 1,...,23$ ;
}
else
{
    /* $B^{org}$ is taken as $B^{ref}$ */
    $B^{ref}$ = $B^{org}$ ;
}
```

**Step 8. Basis coefficients calculation**

A *right side vector* is computed by multiplication of the transposed basis bins matrix by the reference bins vector:

$$V = BB^T \times B^{ref} \tag{10.22}$$

A set of linear equations specified in matrix notation as:

$$EM \times \gamma = V \tag{10.23}$$

is solved and a *basis coefficients vector* $\gamma = \{\gamma_k, k = 1,...,23\}^T$ is obtained:

$$\gamma = EM^{-1} \times V \tag{10.24}$$

In the reference implementation the equations (10.23) are solved using the LU-decomposition representation of the $EM$ matrix computed at step 4.

Negative basis coefficients if any are replaced by zero:

$$\gamma_k = \max(0, \gamma_k), k = 1,...,23 \tag{10.25}$$

**Step 9. Control branching**

The control branching step is described by the following pseudo code:

```
if (voiced_flag == FALSE  OR  it_count == 3)
{
    go to Step 12;
}
/* Otherwise the processing proceeds with the next step 10. */
```

**Step 10. Output bins calculation**

First, an *output bins vector* $B^{out} = \{b_k^{out}, k = 1,...,23\}^T$ is calculated by the multiplication of the transposed basis bins matrix with the basis coefficients vector:

$$B^{out} = BB^T \times \gamma \tag{10.26}$$

Then each zero-valued coordinate of this vector (if any) is replaced by a regularization value:

$$\eta = 0{,}005 \times \sum_{k=1}^{23} b_k^{out} \Big/ 23 \tag{10.27}$$

as shown by the following pseudo code instructions being performed for $k=1,\ldots,23$:

```
if $(b_k^{out} == 0)$
    $b_k^{out} = \eta$;
```

**Step 11. High order cepstra refinement**

Truncated logarithm operation described in clause 5.3.6 is applied to the coordinates of the output bins vector:

$$lB^{out} = \{lb_k = \max(-50, \ln b_k^{out}), k = 1,...,23\} \tag{10.28}$$

Discrete Cosine Transform (DCT) is applied to the $lB^{out}$ vector, besides only 10 last values are calculated out of 23:

$$C_k^{out} = \sum_{i=1}^{23} lb_i \cos\left(\frac{\pi \times k}{23} \times (i - 0{,}5)\right),\ k = 13,...,22 \tag{10.29}$$

which are considered as new estimate of the high order cepstra (HOC). Current high order cepstra values are replaced by these ten coefficients:

$$HOC = \{C_k^{out}, k = 13,...,22\} \tag{10.30}$$

The iteration counter $it\_count$ is incremented and control is passed to Step 6.

**Step 12. Harmonic magnitude estimates calculation**

The vector $A^E = \{A_k^E, k = 1,..., N_h\}$ of harmonic magnitude estimates is computed as a linear combination of the basis vectors (computed at step 2) weighted by the basis coefficients (computed at step 8):

$$A^E = \sum_{n=1}^{N_h} \gamma_n \times BV_n \tag{10.31a}$$

Finally, the obtained vector is modified in order to cancel the effect of the high frequency preemphasis done in the front-end:

$$\begin{aligned} &A_k^E = A_k^E \big/ MagPemp_k,\ \ k = 1,..., N_h \\ &where\ \ MagPemp_k = \sqrt{1 + 0{,}9^2 - 2 \times 0{,}9 \cos(2\pi \times f_k)} \end{aligned} \tag{10.31b}$$

($f_k$ are harmonic normalized frequencies)

### 10.2.7.3 Cepstra to magnitudes transformation

From the pitch period and voicing class parameters, the frequencies $f_k$, $k=1,\ldots,N_v$ of voiced harmonics and the frequencies $f_k$, $k=1,\ldots,N_u$ of unvoiced harmonics are computed in clause 10.2.3. One method to estimate the magnitudes at these frequencies from the mel-frequency cepstral coefficients $C_0, C_1, \ldots, C_{12}$ is described in clause 10.2.7.2. In this clause, a second method for transforming cepstra to magnitudes is specified.

As a first step, the high order cepstra are recovered as described in clause 10.2.7.1 for voiced frames to form the complete cepstra $C_0, C_1, \ldots, C_{22}$. For unvoiced frames, the high order cepstra are not recovered. From the cepstra of each frame, a fixed cepstra are subtracted as follows: $D_i = C_i - F_i$, $i = 0, 1, \ldots, 12$ for unvoiced frames and $i = 0, 1, \ldots, 22$ for voiced frames. The fixed Cepstral values $F_i$ are shown in table 10.2. The modified cepstra $D_i$, $i = 0, 1, \ldots, 12$ (or 22) are used in the estimation of the harmonic magnitudes as described below. To estimate the harmonic magnitude $A^I_k$ at harmonic frequency $f_k$ , the harmonic frequency $f_k$ is first transformed to a corresponding mel-frequency $m_k$ using equation (5.55a) as follows:

$$m_k = 2\,595 \times \log_{10}\left(1 + \frac{f_k}{700}\right) \tag{10.32}$$

The mel-frequency $m_k$ is then transformed to an index $j_k$ with the help of table 10.3. In the table, (integer) index values from 0 to 24 and corresponding mel-frequencies are shown. Let the mel-frequencies given in the table 10.3 be denoted by $M_0,\ldots,M_J,\ldots,M_{24}$. Given a harmonic mel-frequency $m_k$, it is first bounded so that it does not exceed $M_{24}$. Then, the index $J$ (in the range from 1 to 24) is found such that $m_k \leq M_J$. The (possibly non-integer) index value $j_k$ corresponding to $m_k$ is then calculated as:

$$j_k = M_{J-1} + ((m_k - M_{J-1})/(M_J - M_{J-1})) \qquad (10.33)$$

From the index $j_k$, another index $l_k$ is computed as follows:

$$l_k = \begin{cases} 0{,}5; & \text{if } j_k < 0{,}5 \\ 23{,}5; & \text{if } j_k > 23{,}5 \\ j_k; & \text{otherwise} \end{cases} \qquad (10.34)$$

From the modified cepstra $D_i$, $i = 0, 1,\ldots, 12$ (or 22), and the index $l_k$, the log-magnitude estimate $a_k$ is obtained as:

$$a_k = \frac{D_0}{23} + \frac{2}{23}\sum_{i=1}^{Max\_i} D_i \cos((l_k - 0{,}5) \times i \times (\pi/23)) \qquad (10.35)$$

where, $Max\_i$ is 12 or 22 depending on whether the frame is unvoiced or voiced respectively. From $a_k$, the harmonic magnitude estimate $A^I{}_k$ is obtained as follows:

$$B_k = \begin{cases} \exp(a_k) \times 2 \times (m_k/(M_0 + M_1)); & \text{if } j_k < 0{,}5 \\ \exp(a_k) \times 2 \times (24 - j_k); & \text{if } j_k > 23{,}5 \\ \exp(a_k); & \text{otherwise} \end{cases} \qquad (10.36a)$$

$$A^I{}_k = \sqrt{B_k} \qquad (10.36b)$$

The above method (10.33 through 10.36) is applied to each harmonic frequency to estimate the harmonic magnitudes $A^I{}_k$ for $k = 1, 2,\ldots, N_u$ (or $N_v$).

**Table 10.2: Fixed cepstral values**

| **Fixed Cepstral values $F_0$ through $F_{22}$** |
|---:|
| 2,5245156e+01 |
| -3,1339415e+01 |
| -5,0421652e+00 |
| -3,9743845e+00 |
| -1,5154464e+00 |
| -1,3563063e+00 |
| -5,6955354e-01 |
| -7,1809975e-01 |
| -5,5995365e-01 |
| -6,2237629e-01 |
| -5,3362716e-02 |
| -1,6299096e-01 |
| -2,5138527e-01 |
| -8,1102386e-02 |
| -2,1767279e-01 |
| 9,1988824e-02 |
| 1,8607947e-01 |
| 9,6931091e-02 |
| 9,9251014e-02 |
| 4,1572605e-02 |
| 2,6646199e-02 |
| -7,0223354e-02 |
| -2,2043307e-02 |

**Table 10.3: Index values and corresponding mel-frequencies**

| Index value | Mel-Frequencies |
|---|---:|
| 0 | 9,6383e+01 |
| 1 | 1,8517e+02 |
| 2 | 2,6747e+02 |
| 3 | 3,4416e+02 |
| 4 | 4,5023e+02 |
| 5 | 5,1577e+02 |
| 6 | 6,0745e+02 |
| 7 | 6,9222e+02 |
| 8 | 7,7107e+02 |
| 9 | 8,6828e+02 |
| 10 | 9,5777e+02 |
| 11 | 1,0407e+03 |
| 12 | 1,1179e+03 |
| 13 | 1,2075e+03 |
| 14 | 1,2906e+03 |
| 15 | 1,3827e+03 |
| 16 | 1,4679e+03 |
| 17 | 1,5472e+03 |
| 18 | 1,6330e+03 |
| 19 | 1,7238e+03 |
| 20 | 1,8078e+03 |
| 21 | 1,8859e+03 |
| 22 | 1,9766e+03 |
| 23 | 2,0605e+03 |
| 24 | 2,1461e+03 |

## 10.2.7.4 Combined magnitudes estimate calculation

This block calculates a final combined estimate $A = \{An, n = 1,..., N_h)$ of harmonic magnitudes from the estimates $A^E = \{A_n^E, n = 1,..., N_h\}$ and $A^I = \{A_n^I, n = 1,..., N_h\}$ obtained in SFEQ block (clause 10.2.7.2) and CTM block (clause 10.2.7.3) correspondingly. Voiced and unvoiced harmonic arrays are treated slightly differently.

### 10.2.7.4.1 Combined magnitude estimate for unvoiced harmonics

Vector $A^E$ is scaled so that its squared norm is equal to the squared norm of the $A^I$ vector as is specified by the pseudo code:

```
{
    $E^E = \sum_{n-1}^{N_h} A_n^{E^2};$
    if (EE == 0)
        sc = 0;
    else
    {
        $E^I = \sum_{n=1}^{N_h} A_n^{I^2};$
        $sc = \sqrt{E^I / E^E;}$
        $A^E = sc \times A^I;$
    }
}
```

The magnitudes AE and AI are mixed:

$$A = 0{,}9 \times A^E + 0{,}1 \times A^I \tag{10.37}$$

#### 10.2.7.4.2 Combined magnitude estimate for voiced harmonics

Vector $A^E$ is scaled and then mixed with the $A^I$ vector using a pitch dependent mixing proportion.

**Scaling**

Scaling is performed differently for long and short pitch period values.

If the pitch value $p$ is less than 55 samples then $A^E$ vector is scaled exactly as is described in clause 10.2.7.4.1. Otherwise (if $p \geq 55$) the scaling procedure described below is carried out.

Two scaling factors $sc_{low}$ and $sc_{high}$ are calculated in frequency bands [0, 1 200 Hz] and [1 200 Hz, $F_{Nyquist}$] respectively.

$$sc_{low} = \sqrt{\frac{\sum_{n=1}^{L} {A_n^I}^2}{\sum_{n=1}^{L} {A_n^E}^2}}, \quad sc_{high} = \sqrt{\frac{\sum_{n=L+1}^{Nh} {A_n^I}^2}{\sum_{n=L+1}^{Nh} {A_n^E}^2}} \qquad (10.38)$$

where $L = floor(1\,200 \times p/(1\,000 \times f_s))$. A scaling factor is set to 0 if the denominator of the corresponding expression is equal to zero.

Then the harmonic magnitudes $A_n^E$ are modified as specified by the following pseudo code section being executed for:

$$n=1,\ldots,H_h.$$

```
{
    fHz = $f_n \times f_s \times 1\,000$; /* harmonic frequency in Hz units */
    if ( $fHz \leq 200$ )
        $A_n^E = A_n^E \times sc_{low}$;
    elseif ( $fHz \geq 2\,500$
        $A_n^E = A_n^E \times sc_{high}$;
    else
    {
        $\lambda = (2\,500 - fHz)/(2\,500 - 200)$;
        $sc = \lambda \times sc_{low} + (1 - \lambda) \times sc_{high}$;
        $A_n^E = A_n^E \times sc$;
    }
}
```

**Mixing**

Mixture parameter values $\chi_n$ as a function of $p_n$ values are specified by table 10.4.

**Table 10.4: Magnitude mixture parameter *vs*. pitch**

| **N** | $p_n$ | $\chi_n$ | **n** | $p_n$ | $\chi_n$ |
|---|---|---|---|---|---|
| 1 | 22,5 | 0,0459 | 14 | 87,5 | 0,8740 |
| 2 | 27,5 | 0,0765 | 15 | 92,5 | 0,8586 |
| 3 | 32,5 | 0,1124 | 16 | 97,5 | 0,8306 |
| 4 | 37,5 | 0,1384 | 17 | 102,5 | 0,8299 |
| 5 | 42,5 | 0,1869 | 18 | 107,5 | 0,8496 |
| 6 | 47,5 | 0,2858 | 19 | 112,5 | 0,8346 |
| 7 | 52,5 | 0,4309 | 20 | 117,5 | 0,7617 |
| 8 | 57,5 | 0,5676 | 21 | 122,5 | 0,7336 |
| 9 | 62,5 | 0,6458 | 22 | 127,5 | 0,6321 |
| 10 | 67,5 | 0,6779 | 23 | 132,5 | 0,5522 |
| 11 | 72,5 | 0,7009 | 24 | 137,5 | 0,4016 |
| 12 | 77,5 | 0,7646 | 25 | 142,5 | 0,3306 |
| 12 | 82,5 | 0,8347 | 26 | 147,5 | 0,2909 |

The mixture parameter value $\chi$ to be used for mixing the magnitude vectors is determined by linear interpolation between the values given by the table as described by the following pseudo code:

```
{
    if ( $p \le p_1$ )
        $\chi = \chi_1$;
    elseif ( $p \ge p_{26}$ )
        $\chi = \chi_{26}$;
    else
    {
        Find n such that $p_n \le p < p_{n+1}$;
        $\rho = (p_{n+1} - p)/(p_{n+1} - p_n)$;
        $\chi = \rho \times \chi_n + (1-\rho) \times \chi_{n+1}$;
    }
    $A = \chi \times A^E + (1-\chi) \times A^I$;
}
```

## 10.2.8 All-pole spectral envelope modelling

Given the harmonic magnitudes estimate, $A_k$, $k$ = 1, 2,…, $N_v$ , of a voiced frame, an all-pole model is derived from the magnitudes as specified in this clause. The all-pole model parameters $a_j$, $j$ = 1, 2,…, $J$ are used for postfiltering (clause 10.2.9) and harmonic phase synthesis (clause 10.2.10). The model order $J$ is 10.

The magnitudes are first normalized as specified by the pseudo-code below so that the largest normalized value is 1.

```
    if (max($A_k$) > 0)
        $B_k$ = $A_k$ / max($A_k$);  k = 1, 2,…, $N_v$
    else
        $a_j$ = 0;  j = 1, 2,…, J
```

From the normalized magnitudes, a set of interpolated magnitudes is derived. The size of the interpolated vector is given by $K = (N_v - 1) \times F + 1$, where the interpolation factor $F$ is a function of $N_v$ as shown in table 10.5. The interpolated vector is obtained by introducing ($F$ - 1) additional magnitudes through linear interpolation between each consecutive pair of the original magnitudes. When $F$ = 1, i.e. when $N_v \ge 25$, there is no interpolation and $K = N_v$. The interpolated vector is specified as follows:

$$G_k = \begin{cases} B_{1+(k-1)/F}; & k = 1, F+1, 2F+1, ..., (N_v - 1)F + 1 \\ B_j + \dfrac{k-(j-1)F-1}{F}(B_{j+1} - B_j); & (j-1)F + 1 < k < jF + 1, \;\; j = 1, 2, ..., N_v - 1 \end{cases} \quad (10.39)$$

where, $G_k$, $k = 1, 2,\ldots, K = (N_v - 1) \times F + 1$ represent the interpolated magnitude vector. Each of the interpolated magnitudes is then assigned a normalized frequency in the range from 0 to π, viz., $\omega_k = k \times \pi / (K+1)$, $k = 1, 2,\ldots, K$. The interpolated vector is next augmented by two additional magnitude values corresponding to $\omega_k = 0$ (DC) and $\omega_k = \pi$. The length of the augmented, interpolated vector is thus $K + 2$. This vector is still denoted by $G_k$, but the subscript $k$ now ranges from 0 to $K + 1 = (N_v - 1) \times F + 2$. The values of $G_0$ and $G_{K+1}$ are obtained as shown in the pseudo-code below.

```
    if (F == 1)
{
    $G_{K+1} = G_K$;
if ($G_2 > 1.2\ G_1$)
        $G_0 = 0.8\ G_1$;
    else if ($G_2 < 0.8\ G_1$)
        $G_0 = 1.2\ G_1$;
    else
        $G_0 = G_1$;
}
else
{
    $G_{K+1} = 2.0\ (G_K - G_{K-1})$;
    $G_0 = 2.0\ (G_1 - G_2)$;
}
```

**Table 10.5: Interpolation factor *vs*. number of harmonics**

| Number of voiced harmonics | Interpolation factor |
|---|---|
| $N_v < 12$ | 4 |
| $12 \le N_v < 16$ | 3 |
| $16 \le N_v < 25$ | 2 |
| $25 \le N_v$ | 1 |

From the augmented, interpolated vector $G_k$, $k = 0, 1,\ldots, K+1$, a pseudo-autocorrelation function $R_j$ is computed using the cosine transform as follows:

$$R_j = G_0 + (-1)^j G_{K+1} + 2\sum_{k=1}^{K} G_i \cos(\omega_k \times j);\ j = 0, 1,\ldots, J \tag{10.40}$$

From the pseudo-autocorrelation coefficients $R_j$, $j = 0, 1,\ldots, J$, the all-pole model parameters $a_j$, $j = 1, 2,\ldots, J$ are obtained through the well known Levinson-Durbin recursion as the solution of the normal equations:

$$\sum_{j=1}^{J} a_j \times R_{i-j} = R_i;\ 1 \le i \le J \tag{10.41}$$

For the case when $F = 1$, i.e. when $N_v \ge 25$, the all-pole model parameters derived as above represent the final values. For other cases when $F > 1$, the model parameters are further refined as specified below. The spectral envelope defined by the all-pole model parameters is given by:

$$H(\omega) = \frac{1}{\left|1 + a_1 e^{-j\omega} + a_2 e^{-j2\omega} + \ldots + a_J e^{-jJ\omega}\right|^2} \tag{10.42}$$

where, the $e^{j\omega}$ represents a complex exponential at frequency ω. The spectral envelope given by (10.42) is sampled at all the frequencies $\omega_k = k\pi / (K+1)$, $k = 0, 1,\ldots, K+1$ to obtain the modelled magnitudes $H_k$, $k = 0, 1,\ldots, K+1$. The maximum of the modelled magnitudes at frequencies corresponding to the original estimated magnitudes is then used to normalize the modelled magnitudes as follows:

$$L_k = H_k \big/ \max(H_k \;|\; k = 1, F+1, 2F+1, \ldots, (N_v - 1)F + 1);\quad k = 0, 1, \ldots,\ K+1 \tag{10.43}$$

Next, scale factors $S_k$ , $k$ = 0, 1,…, $K$ + 1 are computed as follows:

$$S_k = \begin{cases} 1; & k = 0 \text{ and } k = K+1 \\ G_k / L_k; & k = 1, F+1, 2F+1, ..., (N_v - 1)F + 1 \\ S_{(j-1)F+1} + \dfrac{k-(j-1)F-1}{F}(S_{jF+1} - S_{(j-1)F+1}); & (j-1)F+1 < k < jF+1; \quad j = 1, 2, ..., (N_v - 1) \end{cases} \tag{10.44}$$

The normalized, modelled magnitudes are then multiplied by the appropriate scale factors to obtain a new set of magnitudes $M_k = L_k \times S_k$, $k$ = 0, 1,…, $K$+1. This set of magnitudes is used to compute a new pseudo-autocorrelation function using (10.40) and subsequently a new set of all-pole model parameters as a solution (10.41) as the final values.

## 10.2.9 Postfiltering

Postfiltering is applied to the harmonic magnitudes $A_k$ , $k$ = 1, 2,…, $N_v$ of a voiced frame to emphasize the formants in the speech signal using the all-pole model parameters $a_j$, $j$ = 1, 2,…, $J$ as specified below.

From the number of voiced harmonics $N_v$, the interpolation factor $F$ from table 10.5 and the interpolated vector size $K = (N_v - 1) \times F + 1$ are first determined. Then, a weighting factor $U_k$ is computed for each harmonic as follows:

$$\theta_k = ((k-1) \times F + 1) \times (\pi / (K+1)) \tag{10.45a}$$

$$\mathrm{Re}1_k = 1 + \sum_{j=1}^{J} a_j \times \alpha^j \times \cos(j \times \theta_k); \; k = 1, 2, \ldots, N_v \tag{10.45b}$$

$$\mathrm{Im}1_k = -\sum_{j=1}^{J} a_j \times \alpha^j \times \sin(j \times \theta_k); \; k = 1, 2, \ldots, N_v \tag{10.45c}$$

$$\mathrm{Re}\,2_k = 1 + \sum_{j=1}^{J} a_j \times \beta^j \times \cos(j \times \theta_k); \; k = 1, 2, \ldots, N_v \tag{10.45d}$$

$$\mathrm{Im}\,2_k = -\sum_{j=1}^{J} a_j \times \beta^j \times \sin(j \times \theta_k); \; k = 1, 2, \ldots, N_v \tag{10.45e}$$

$$\mathrm{Re}\,3_k = 1 - \mu \times \cos(\theta_k); \; k = 1, 2, \ldots, N_v \tag{10.45f}$$

$$\mathrm{Im}\,3_k = \mu \times \sin(\theta_k); \; k = 1, 2, \ldots, N_v \tag{10.45g}$$

$$U_k = \frac{[(\mathrm{Re}\,2_k)^2 + (\mathrm{Im}\,2_k)^2] \times \sqrt{[(\mathrm{Re}\,3_k)^2 + (\mathrm{Im}\,3_k)^2]}}{[(\mathrm{Re}1_k)^2 + (\mathrm{Im}1_k)^2]}; \; k = 1, 2, \ldots, N_v \tag{10.45h}$$

The values of α, β, and μ are respectively 0,95, 0,75 and 0,5. The weights are then normalized and bounded as follows:

$$V_k = U_k \Bigg/ \left( \frac{1}{N_v} \sum_{k=1}^{N_v} U_k{}^4 \right)^{0.25} \; ; \; k = 1, 2, \ldots, N_v \tag{10.46a}$$

$$W_k = \max(0{,}5, \min(1{,}5, V_k)); \; k = 1, 2, \ldots, N_v \tag{10.46b}$$

Postfiltering is applied to the harmonic magnitudes as follows. It is ensured that the energy in the harmonics before and after postfiltering remains the same.

$$B_k = \begin{cases} A_k \times W_k; \; if \;\; \theta_k \geq 0{,}05 \times \pi \\ A_k; \; otherwise \end{cases} \tag{10.47a}$$

$$\rho = \sqrt{\sum_{k=1}^{N_v} W_k{}^2 \Big/ \sum_{k=1}^{N_v} B_k{}^2} \qquad (10.47b)$$

$$P_k = \rho \times B_k\ ;\ k = 1, 2, \ldots, N_v \qquad (10.47c)$$

where $P_k$, $k = 1, 2, \ldots, N_v$ represent the postfiltered harmonic magnitudes.

## 10.2.10 Voiced phase synthesis

The harmonic phases $\varphi_k$, $k = 1, 2, \ldots, N_v$ of a voiced frame with harmonic cyclic frequencies $\omega_k = 2\pi f_k$, $k = 1, 2, \ldots, N_v$ are specified as follows. Each harmonic phase $\varphi_k$ is made up of three components: a linear phase component $\varphi_{k,lin}$, an excitation phase component $\varphi_{k,exc}$, and an envelope phase component $\varphi_{k,env}$.

The linear phase component is computed as follows:

$$\varphi_{k,lin} = \begin{cases} 0;\ \ if\ \ previous\ \ frame\ is\ unvoiced \\ (\varphi_{1,lin,prev} \times RF + \omega_{1,ave} \times M) \times k;\ otherwise \end{cases} \qquad (10.48)$$

where:

- $\varphi_{1,lin,prev}$ represents the linear phase component of the fundamental phase of the previous frame;

*RF* represents a rational factor of the *R1/R2*, where *R1*,*R2* ∈ {1,2,3,4}, such that the jump given by $\left| p \times R1 - p_{prev} \times R2 \right| \big/ p \times R1$ between the previous pitch period ($p_{prev}$) and current pitch period ($p$) is minimized;

$\omega_{1,ave}$ is the weighted sum of the fundamental (cyclic) frequency of the previous and current frames given by $\omega_{1,ave} = (\omega_{1,prev} \times RF + \omega_1)/2$, and *M* is the frame shift in samples.

Note that $p_{prev}$ and $\varphi_{1,lin,prev}$ are initialized to 0 (meaning the previous frame is unvoiced) when the very first frame is being processed.

The excitation phase component is determined using table 10.6 as follows. Given a harmonic frequency $\omega_k$, it is first transformed into an integer index $I_k = round(256 \times \omega_k / \pi)$, the corresponding value $T[I_k]$ from table 10.6 is looked up, and un-normalized to obtain $\varphi_{k,exc} = T[I_k] \times \pi$.

The envelope phase component is computed using the all-pole mode parameters, $a_j$, $j = 1, 2, \ldots, J$, as follows. From the number of voiced harmonics $N_v$, the interpolation factor $F$ from table 10.5 and the interpolated vector size $K = (N_v - 1) \cdot F + 1$ are first determined. Then the envelope phase component is computed as:

$$\theta_k = ((k-1) \times F + 1) \times (\pi / (K+1))\ ;\ k = 1, 2, \ldots, N_v \qquad (10.49a)$$

$$\mathrm{Re}_k = 1 + \sum_{j=1}^{J} a_j \cos(j \times \theta_k)\ ;\ k = 1, 2, \ldots, N_v \qquad (10.49b)$$

$$\mathrm{Im}_k = -\sum_{j=1}^{J} a_j \sin(j \times \theta_k)\ ;\ k = 1, 2, \ldots, N_v \qquad (10.49c)$$

$$\varphi_{k,env} = (-2) \times \tan^{-1}(\mathrm{Im}_k / \mathrm{Re}_k)\ ;\ k = 1, 2, \ldots, N_v \qquad (10.49d)$$

The excitation and envelope components of the phases are added and any linear component is removed as follows:

$$\varphi_{k,sum} = \varphi_{k,exc} + \varphi_{k,env};\ \ k = 1, 2, \ldots, N_v \qquad (10.50a)$$

$$\varphi_{k,add} = \varphi_{k,sum} - k \times \varphi_{1,sum}; \quad k = 1, 2, ..., N_v \qquad (10.50b)$$

The linear phase component and the additional phase component are added to obtain the harmonic phases for the voiced frame as follows:

$$\varphi_k = \varphi_{k,lin} + \varphi_{k,add}; \quad k = 1, 2, ..., N_v \qquad (10.51)$$

**Table 10.6: Normalized excitation phases**

| Index | Normalized phase | Index | Normalized phase | Index | Normalized phase | Index | Normalized phase |
|---|---|---|---|---|---|---|---|
| 0 | 0,000000 | 64 | 0,806122 | 128 | -0,428986 | 192 | 0,231750 |
| 1 | 0,577271 | 65 | 0,761841 | 129 | -0,449249 | 193 | 0,219360 |
| 2 | 0,471039 | 66 | 0,707184 | 130 | -0,476257 | 194 | 0,211182 |
| 3 | 0,402039 | 67 | 0,649353 | 131 | -0,512085 | 195 | 0,207703 |
| 4 | 0,341461 | 68 | 0,595245 | 132 | -0,555054 | 196 | 0,209747 |
| 5 | 0,282104 | 69 | 0,553375 | 133 | -0,601379 | 197 | 0,215332 |
| 6 | 0,221069 | 70 | 0,535004 | 134 | -0,646881 | 198 | 0,217590 |
| 7 | 0,157074 | 71 | 0,551025 | 135 | -0,687469 | 199 | 0,208527 |
| 8 | 0,089905 | 72 | 0,593689 | 136 | -0,720123 | 200 | 0,184631 |
| 9 | 0,019989 | 73 | 0,629669 | 137 | -0,743896 | 201 | 0,147583 |
| 10 | -0,051819 | 74 | 0,641205 | 138 | -0,760712 | 202 | 0,101593 |
| 11 | -0,124237 | 75 | 0,637146 | 139 | -0,774292 | 203 | 0,051697 |
| 12 | -0,195770 | 76 | 0,630432 | 140 | -0,786865 | 204 | 0,002960 |
| 13 | -0,264679 | 77 | 0,626068 | 141 | -0,796417 | 205 | -0,039154 |
| 14 | -0,328705 | 78 | 0,618439 | 142 | -0,797058 | 206 | -0,068756 |
| 15 | -0,385162 | 79 | 0,597534 | 143 | -0,782288 | 207 | -0,080597 |
| 16 | -0,430573 | 80 | 0,558716 | 144 | -0,753052 | 208 | -0,073730 |
| 17 | -0,460846 | 81 | 0,504242 | 145 | -0,723755 | 209 | -0,055573 |
| 18 | -0,472351 | 82 | 0,439545 | 146 | -0,710052 | 210 | -0,038666 |
| 19 | -0,464783 | 83 | 0,371796 | 147 | -0,714722 | 211 | -0,030792 |
| 20 | -0,444977 | 84 | 0,314423 | 148 | -0,731720 | 212 | -0,033630 |
| 21 | -0,425323 | 85 | 0,322479 | 149 | -0,753998 | 213 | -0,047180 |
| 22 | -0,415466 | 86 | 0,692352 | 150 | -0,776672 | 214 | -0,072174 |
| 23 | -0,418579 | 87 | 0,820557 | 151 | -0,797760 | 215 | -0,109039 |
| 24 | -0,433502 | 88 | 0,775940 | 152 | -0,817749 | 216 | -0,156860 |
| 25 | -0,457764 | 89 | 0,703735 | 153 | -0,838562 | 217 | -0,213318 |
| 26 | -0,488617 | 90 | 0,625885 | 154 | -0,861664 | 218 | -0,275146 |
| 27 | -0,523315 | 91 | 0,549744 | 155 | -0,887115 | 219 | -0,338562 |
| 28 | -0,559174 | 92 | 0,479889 | 156 | -0,913971 | 220 | -0,398956 |
| 29 | -0,593689 | 93 | 0,420258 | 157 | -0,941437 | 221 | -0,450836 |
| 30 | -0,625031 | 94 | 0,374023 | 158 | -0,969849 | 222 | -0,487793 |
| 31 | -0,652130 | 95 | 0,341888 | 159 | 0,999176 | 223 | -0,505707 |
| 32 | -0,674835 | 96 | 0,319366 | 160 | 0,963562 | 224 | -0,510162 |
| 33 | -0,693390 | 97 | 0,297546 | 161 | 0,922089 | 225 | -0,518524 |
| 34 | -0,707428 | 98 | 0,268768 | 162 | 0,875092 | 226 | -0,545410 |
| 35 | -0,715729 | 99 | 0,230896 | 163 | 0,824432 | 227 | -0,592499 |
| 36 | -0,717133 | 100 | 0,186066 | 164 | 0,773285 | 228 | -0,654510 |
| 37 | -0,713837 | 101 | 0,137939 | 165 | 0,726074 | 229 | -0,725586 |
| 38 | -0,713104 | 102 | 0,090027 | 166 | 0,688934 | 230 | -0,801025 |
| 39 | -0,723785 | 103 | 0,045288 | 167 | 0,669617 | 231 | -0,877136 |
| 40 | -0,750366 | 104 | 0,005859 | 168 | 0,674377 | 232 | -0,950897 |
| 41 | -0,791931 | 105 | -0,026398 | 169 | 0,698090 | 233 | 0,980316 |
| 42 | -0,845093 | 106 | -0,049316 | 170 | 0,719421 | 234 | 0,918762 |
| 43 | -0,905945 | 107 | -0,059448 | 171 | 0,721069 | 235 | 0,866211 |
| 44 | -0,970825 | 108 | -0,052521 | 172 | 0,702698 | 236 | 0,824219 |
| 45 | 0,963654 | 109 | -0,028687 | 173 | 0,671631 | 237 | 0,795319 |
| 46 | 0,901123 | 110 | -0,000732 | 174 | 0,634674 | 238 | 0,786377 |
| 47 | 0,846222 | 111 | 0,012024 | 175 | 0,596527 | 239 | 0,810913 |
| 48 | 0,805481 | 112 | 0,001312 | 176 | 0,559784 | 240 | 0,872406 |
| 49 | 0,788788 | 113 | -0,028900 | 177 | 0,525757 | 241 | 0,925385 |
| 50 | 0,807312 | 114 | -0,070801 | 178 | 0,494995 | 242 | 0,926483 |
| 51 | 0,857269 | 115 | -0,117004 | 179 | 0,468231 | 243 | 0,882111 |
| 52 | 0,904724 | 116 | -0,160583 | 180 | 0,446991 | 244 | 0,808807 |
| 53 | 0,922668 | 117 | -0,194824 | 181 | 0,433105 | 245 | 0,716248 |
| 54 | 0,913757 | 118 | -0,214020 | 182 | 0,427216 | 246 | 0,608063 |
| 55 | 0,888916 | 119 | -0,217743 | 183 | 0,426483 | 247 | 0,480927 |

| Index | Normalized phase | Index | Normalized phase | Index | Normalized phase | Index | Normalized phase |
|---|---|---|---|---|---|---|---|
| 56 | 0,856750 | 120 | -0,215424 | 184 | 0,424225 | 248 | 0,310974 |
| 57 | 0,823730 | 121 | -0,221161 | 185 | 0,414124 | 249 | -0,054810 |
| 58 | 0,796082 | 122 | -0,241730 | 186 | 0,393951 | 250 | -0,554077 |
| 59 | 0,781250 | 123 | -0,274475 | 187 | 0,365723 | 251 | -0,763275 |
| 60 | 0,786346 | 124 | -0,313202 | 188 | 0,333374 | 252 | -0,904968 |
| 61 | 0,809631 | 125 | -0,351440 | 189 | 0,301086 | 253 | 0,977448 |
| 62 | 0,831787 | 126 | -0,384247 | 190 | 0,272278 | 254 | 0,884125 |
| 63 | 0,831818 | 127 | -0,409363 | 191 | 0,249054 | 255 | 0,849152 |
| | | | | | | 256 | 0,999969 |

## 10.2.11 Line spectrum to time-domain transformation

This block transforms a line spectrum of the frame represented by an array $H=\{H_n=(f_n, A_n, \varphi_n), n=1,...,N_h\}$ of harmonics to a time-domain speech signal segment. If the frame is of fully-voiced class as indicated by $vc == "fully\text{-}voiced"$ the array $H$ is set to $VH$. In case of unvoiced frame ($vc == "unvoiced"$) $H$ is set to $UH$. In the case of mixed-voiced frame the arrays of voiced and unvoiced harmonics are combined as described in the following clause.

### 10.2.11.1 Mixed-voiced frames processing

This step is performed for the mixed-voiced frames only as indicated by $vc == "mixed\_voiced"$. The input to the step are the array $VH=\{H_n^v=(f_n^v, A_n^v, \varphi_n^v), n=1,...,N_v\}$ of voiced harmonics and the array $UH=\{H_n^u=(f_n^u, A_n^u, \varphi_n^u), n=1,...,N_u\}$ of unvoiced harmonics. The output is a combined array $H=\{H_n=(f_n, A_n, \varphi_n), n=1,...,N_h\}$ of harmonics. The combined array contains the voiced harmonics associated with frequencies lower than 1 200 Hz and the unvoiced harmonics associated with frequencies higher than 1 200 Hz. The processing is described by the following pseudo code:

```
{
    v_last = $ceil(1\,200/(1\,000\times p))$; /* index of the last voiced harmonic to be taken */
    u_first = $ceil(1\,200/(1\,000\times FFTL))+1$; /* index of the first unvoiced harmonic to be taken */
    $sc = \sqrt{\frac{\sum_{n=v\_last+1}^{N_v} {A_n^v}^2}{\sum_{n=u\_first}^{N_u} A_n^u}}$; /* compute magnitude scaling factor */
    $H_n = H_n^v,\ n=1,...,v\_last$;
    $f_{v\_last+n-u\_first+1} = f_n^u,\quad \varphi_{v\_last+n-u\_first+1} = \varphi_n^u,\ n=u\_first,...,N_u$;
    $A_{v\_last+n-u\_first+1} = sc\times A_n^u,\ n=u\_first,...,N_u$;
    $N_h = v\_last+N_u-u\_first+1$;
}
```

### 10.2.11.2 Filtering very high-frequency harmonics

At this step the harmonics associated with the frequencies close enough to the Nyquist frequency (if any) are filtered out. Those elements of the harmonics array which satisfy the condition:

$$round(f\times FFTL) > round(0{,}93\times FFTL/2) \qquad (10.52)$$

are eliminated and the number $N_h$ of harmonics is updated appropriately.

### 10.2.11.3 Energy normalization

A synthetic complex discrete spectrum is calculated:

$$sd_i = \sum_{n=1}^{N_h} A_n \times \exp(j \times \varphi_n) \times \Delta(f_n - i/FFTL), i = 0,..., FFTL/2 \qquad (10.53)$$

by convolution of the line spectrum with truncated Dirichlet kernel:

$$\Delta(f) = \begin{cases} 0, \ if \quad f = 0 \ or \ |f| > WT\_BW \\ \sin(\pi \times f \times N)/\sin(\pi \times f), \ otherwise \end{cases} \qquad (10.54)$$

where WT_BW is given by (10.18). Then the frame energy estimate $E_e$ is calculated:

$$E_e = \frac{1}{FFTL}\left(|sd_0|^2 + |sd_{FFTL/2}|^2 + 2 \times \sum_{i=1}^{FFTL/2-1} |sd_i|^2\right) \qquad (10.55)$$

If the energy estimate is nonzero a normalization factor *NF* is computed using the logE parameter extracted from the decoded feature vector:

$$NF = \sqrt{\exp(\log E)/E_e}\,, \qquad (10.56)$$

otherwise the normalization factor is set to zero *NF = 0*.

The harmonic magnitudes are scaled:

$$A_n = NF \times A_n, \ n = 1,..., N_h \qquad (10.57)$$

### 10.2.11.4 STFT spectrum synthesis

A synthetic complex discrete spectrum *s_stft* is calculated like in (10.53) but Fourier transform of *2M* (*M = 80* is frame shift) samples long Hann window is used instead of the Dirichlet kernel:

$$s\_stft_i = \sum_{n=1}^{N_h} A_n \times \exp(j \times \varphi_n) \times HnWT(f_n - i/FFTL), i = 0,..., FFTL/2, \qquad (10.58)$$

$$HWT(f) = \begin{cases} 0{,}50\Delta(f) + 0{,}25 \times \left[\Delta\left(f - \frac{1}{2M-1}\right) + \Delta\left(f + \frac{1}{2M-1}\right)\right], \ if \ |f| \le WT\_BW, \\ 0, \ if \ |f| > WT\_BW \end{cases} \qquad (10.59)$$

where $\Delta$ is given by (10.17), and *WT_BW* by (10.18).

### 10.2.11.5 Inverse FFT

An inverse FFT is applied to the synthetic STFT spectrum resulting in FFTL-dimensional vector $S_{syn} = \{s_n^{syn}, n = 0,..., FFTL-1\}$ with real coordinates which is used as a time-domain representation of current frame:

$$s_n^{syn} = \frac{1}{FFTL}\sum_{i=0}^{FFTL-1} s\_stft_i \times \exp(j \times i \times n \frac{2\pi}{FFTL}) \qquad (10.60)$$

In (10.60) $s\_stft_i = s\_stft^*_{FFTL-i-1} \ if \ i \ge FFTL/2$

## 10.2.12 Overlap-Add

The input to the Overlap-Add block (OLA) is the synthesized time-domain frame $S^{syn}$ . The OLA block outputs an M = 80 samples long segment of speech which is appended to the already synthesized part of the speech signal. The OLA block maintains a pair of M samples long buffers: $BUF^{out} = \{buf_k^{out}, k = 1,...,M\}$ and $BUF^{ola} = \{buf_k^{ola}, k = 1,...,M\}$.

Each coordinate of $BUF^{ola}$ is initialized by zero values when the very first frame is processed. $BUF^{ola}$ preserves its contents in between invocations of the OLA block. The procedure performed in the OLA block is specified by the following pseudo code:

```
{
    $buf_{k+1}^{ola} = buf_{k+1}^{ola} + s\_stft_{FFTL-M+k}, \ k = 1,...,M-1$ ; /* overlap-add */
    $buf_k^{out} = buf_k^{ola}, \ k = 1,...,M$ ; /* copy OLA buffer to OUT buffer */
    $buf_k^{ola} = s\_stft_{k-1}, \ k = 1,...,M$ ; /* prepare for the next frame */
    Output $BUF^{out}$ ;
}
```

# Annex A (informative): Voice Activity Detection (VAD)

## A.1 Introduction

The voice activity detector has two stages - a frame-by-frame detection stage consisting of three measurements, and a decision stage in which the pattern of measurements, stored in a circular buffer, is analysed to indicate speech likelihood. The final decision from this second stage is applied retrospectively to the earliest frame in the buffer, so providing a short look-ahead facility. A hangover facility is also provided, with hangover duration related to speech likelihood.

## A.2 Stage 1 - Detection

In non-stationary noise, long-term energy thresholds are not a reliable indicator of speech. Similarly, in high noise conditions the structure of the speech (e.g. harmonics) cannot be wholly relied upon as an indicator as they may be corrupted by noise, or structured noises may confuse the detector.

This voice activity detector uses a noise-robust characteristic of the speech, namely the energy acceleration associated with voice onset. This acceleration is measured in three ways: across the whole spectrum, over a sub-region of the spectrum likely to contain the fundamental pitch, and by the "acceleration" of the variance of values within the lower half of the spectrum of each frame.

The generally higher SNRs of the sub-region detector make it highly noise robust, but it is vulnerable to high-pass microphones, speaker changes and band-limited noise. Consequently it cannot be relied upon in all circumstances and is treated as an augmentation of the whole spectrum measure rather than as a substitute. The variance measure detects structure within the lower half of the spectrum, making it sensitive to voiced speech. This complements the whole spectrum measure, which is better able to detect unvoiced and plosive speech.

All three measurements take their raw input from the linear-frequency Wiener filter coefficients generated by the first stage of the double Wiener filter, as described in clause 5.1.4. Each measurement uses a different aspect of this data.

**Measurement 1 - Whole spectrum**

The whole-spectrum detector uses the Mel-warped Wiener filter coefficients generated by the first stage of the double Wiener filter (see clause 5.1.6). A single input value is obtained by squaring the sum of the Mel filter banks.

The detector applies each of the following steps to the input from each frame, as described below:

1. If Frame < 15 AND Acceleration < 2,5, Tracker = MAX(Tracker, Input);

2. If Input < Tracker × UpperBound and Input > Tracker × LowerBound;

   Tracker = $a$ × Tracker + (1 - $a$) × Input;

3. If Input < Tracker × Floor, Tracker = $b$ × Tracker + (1 - $b$) × Input;

4. If Input > Tracker × Threshold, output TRUE else output FALSE.

Where $a$ = 0,8 and $b$ = 0,97, *UpperBound* is 150 % and *LowerBound* 75 %. *Floor* is 50 % and *Threshold* is 165 %. *Input* is as described above. While *Acceleration* could be calculated using the double-differentiation of successive inputs, it is estimated here by tracking the ratio of two rolling averages of the inputs. (The ratio of fast and slow-adapting rolling averages reflects the acceleration of successive inputs. The contribution rates for the averages used here were (0 × mean + 1 × input) and ((Frame - 1) × mean + 1 × input) / Frame respectively, making the acceleration measure increasingly sensitive over the first 15 frames).

Step one initializes the noise estimate *Tracker*. The acceleration measure prevents *Tracker* being updated if speech occurs within the lead-in of *Frame* < 15 frames. Step two updates *Tracker* if the current input is similar to the noise estimate. Step three is a failsafe for those instances where there is speech or an uncharacteristically large noise within the first few frames, causing the resulting erroneously high noise estimate to decay. Note there is no update if the value is greater than *UpperBound*, or between *LowerBound* and *Floor*. Step four returns true if the current input is more than 165 % larger than *Tracker*. The ratio of the instantaneous input to the short-term mean *Tracker* is a function of the acceleration of successive inputs.

**Measurement 2 - Spectral sub-bands**

The sub-band detector uses the average of the second, third and fourth Mel-filter bands described in Measurement 1 as its input. The detector then applies each of the following steps to the input from each frame, as described below:

1. Input = *p* × CurrentInput + (1 - *p*) × PreviousInput;

2. If Frame < 15, Tracker = MAX(Tracker, Input);

3. If Input < Tracker × UpperBound and Input > Tracker × LowerBound;

   Tracker = *a* × Tracker + (1 - *a*) × Input;

4. If Input < Tracker × Floor, Tracker = *b* × Tracker + (1 - *b*) × Input;

5. If Input > Tracker × Threshold, output TRUE else output FALSE.

Here, *p* = 0,75. All other parameters are the same as for Measurement 1 except *Threshold*, which equals 3,25.

**Measurement 3 - Spectral Variance**

The variance of the values comprising the whole frequency range of the linear-frequency Wiener filter coefficients for each frame is used as an input. The variance is calculated as:

$$\frac{1}{N_{SPEC}}\sum_{i=0}^{N_{SPEC}-1}\left(H_2(bin)\right)^2-\left(\sum_{i=0}^{N_{SPEC}-1}H_2(bin)\right)^2 / N_{SPEC}^2 \qquad \text{(A.1)}$$

where $N_{SPEC} = N_{FFT}/4$, and $H_2(bin)$ are the values of the linear-frequency Wiener filter coefficients as calculated by equation (5.17). The detector takes the maximum input value of the first 15 frames and then proceeds in the same manner as steps 2-4 of Measurement 1, to give a true/false output.

# A.3 Stage 2 - VAD Logic

The three measures discussed above are presented to a VAD decision algorithm. Successive inputs are presented to a buffer, which provides contextual analysis. This introduces a frame delay equal to the length of the buffer minus 1.

For an *N = 7* frame buffer, the most recent input is stored at position *N*. The decision logic applies each of the following steps, which are explained in detail below:

```
$V_N$ = Measurement 1 OR Measurement 2 OR Measurement 3

$M = MAX\left\{\begin{array}{ll} C++, & V_i = TRUE \\ C=0, & V_i = FALSE \end{array}, 1<i<N\right\}\Big|_C$

If $M>=S_P$ AND $T<L_S$, $T=L_S$

If $M>=S_L$ AND $F>F_S$, $T=L_M$ else $T=L_L$

If $M<S_P$ AND $T>0$, T--

If T>0 output TRUE else output FALSE

Frame++, Shift buffer left and return to step 1
```

Step 1: Input $V_N$ is true if any of the three measurements returns true.

Step 2: The algorithm searches for the longest single sequence of "true" values in the buffer; a counter C is incremented if the next buffer value is true, and is reset to 0 if it is false. The maximum value of C over the whole buffer is then taken as M. So for the sequence T T F T T T F, M would equal 3.

Step 3: If there are $S_P$ = 3 or more contiguous "true" values, the buffer is judged to contain "possible" speech. A short timer $T$ of $L_S$ = 5 frames is activated if no hangover is already present.

Step 4: If there are $S_L$ = 4 or more contiguous "true" values, the buffer is judged to contain "likely" speech. A medium timer $T$ of $L_M$ = 23 frames is activated if the current frame $F$ is outside an initial lead-in safety period $F_S$. Otherwise, a failsafe long timer $T$ of $L_L$ = 40 frames is used as the early presence of speech in the utterance may cause the initial noise estimate of the VAD to be too high.

Step 5: If there are less than $S_P$ = 3 contiguous "true" values, decrement the timer.

Step 6: If the timer is greater than 0, output a "true" speech decision.

Step 7: In preparation for the next frame, left-shift the buffer to accommodate the next input.

The output speech decision is applied to the frame being ejected from the buffer:

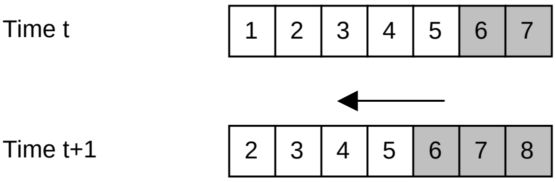


**Figure A.1**

At time t, only the current and previous inputs were "True". Consequently when the buffer is shifted, frame #1 will be marked as False. At time t + 1, a third "true" input has been received. Consequently when the buffer is shifted, frame #2 will be marked as True.

Assuming only these three inputs are "True", the full output sequence will be:

F T T T T T T T T T T T T T T T T F F F F F F…

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23…

Where green is the buffer lead in/out, red marks the positions of the actual original inputs, and blue is the timer hangover. The buffer length and hangover timers can be adjusted to suit needs, although the buffer should always be ≥ $S_L$. Once all frames in the utterance have been input, the buffer shifts until empty.

# Annex B (informative): Bibliography

IETF Audio Video Transport, Internet-Draft: "RTP Payload Format for ETSI ES 201 108 Distributed Speech Recognition Encoding" http://www.ietf.org/internet-drafts/draft-ietf-avt-dsr-05.txt.

# History

| Document history | | | |
|---|---|---|---|
| V1.1.1 | August 2003 | Membership Approval Procedure | MV 20031024: 2003-08-26 to 2003-10-24 |
| V1.1.1 | November 2003 | Publication | |
| | | | |
| | | | |
| | | | |