Contents for Subpart 2

Subpart 2: Speech coding - HVXC

2.1 Scope

MPEG-4 parametric speech coding uses Harmonic Vector eXcitation Coding (HVXC) algorithm, where harmonic coding of LPC residual signals for voiced segments and Vector eXcitation Coding (VXC) for unvoiced segments are employed. HVXC allows coding of speech signals at 2.0 kbps and 4.0 kbps with a scalable scheme, where 2.0 kbps decoding is possible not only using the 2.0 kbps bit-stream but also using a 4.0 kbps bit-stream. HVXC also provides variable bit rate coding where a typical average bit-rate is around 1.2-1.7 kbit/s. Independent change of speed and pitch during decoding is possible, which is a powerful functionality for fast data base search. The frame length is 20 ms, and one of four different algorithmic delays, 33.5 ms, 36ms, 53.5 ms, 56 ms can be selected.

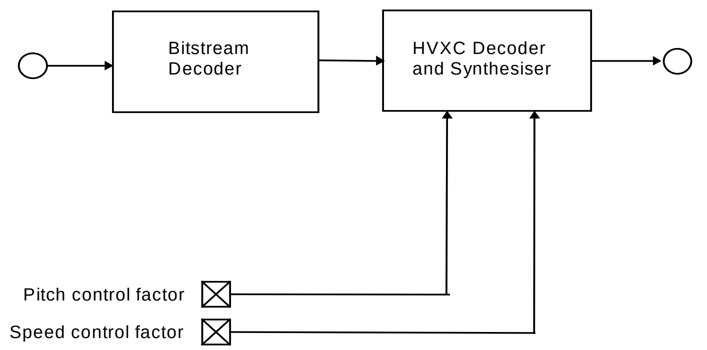


Figure 2.1.1 - Block diagram of the parametric speech decoder

2.2 Definitions

2.2.1. DFT: Discrete Fourier Transform.
2.2.2. dimension conversion: A method to convert a dimension of a vector by a combination of low pass filtering and linear interpolation.
2.2.3. excitation: A time domain signal which excites the LPC synthesis filter.
2.2.4. FFT: Fast Fourier Transform.
2.2.5. fundamental frequency: A parameter which represents signal periodicity in frequency domain.
2.2.6. harmonics: Samples of frequency spectrum at multiples of the fundamental frequency.
2.2.7. harmonic magnitude: Magnitude of each harmonic.
2.2.8. harmonic synthesis: A method to obtain a periodic excitation from harmonic magnitudes.
2.2.9. HVXC: Harmonic Vector eXcitation Coding (parametric speech coding)
2.2.10. IFFT: Inverse Fast Fourier Transform.
2.2.11. initial phase: A phase value at the onset of voiced signal in harmonic synthesis.
2.2.12. interframe prediction: A method to predict a value in the current frame from values in the previous frames. Interframe prediction is used in VQ of LSP.
2.2.13. LPC: Linear Predictive Coding.
2.2.14. LPC synthesis filter: An IIR filter whose coefficients are LPC coeffcients. This filter models the time varying vocal tract.
2.2.15. LPC residual signal: A signal filtered by the LPC inverse filter, which has a flattened frequency spectrum.
2.2.16. LSP: Line Spectral Pairs.
2.2.17. mixed voiced frame: A speech segment which has both voiced and unvoiced components.
2.2.18. pitch: A parameter which represents signal periodicity in the time domain. It is expressed in terms of the number of samples.
2.2.19. pitch control: A functionality to control the pitch of the synthesized speech signal without changing its speed.
2.2.20. postfilter: A filter to enhance the perceptual quality of the synthesized speech signal.
2.2.21. sinusoidal synthesis: A method to obtain a time domain waveform by a sum of amplitude modulated sinusoidal waveforms.
2.2.22. spectral envelope: A set of harmonic magnitudes.

2.2.23. speed control: A functionality to control the speed of the synthesized speech signal without changing its pitch or phonemes.
2.2.24. SQ: Scalar Quantization.
2.2.25. unvoiced frame: A speech segment which looks like random noise with no periodicity.
2.2.26. variable bit rate: The number of bits representing a coded frame varies frame by frame over time.
2.2.27. voiced frame: A speech segment which has periodicity and a relatively high energy.
2.2.28. VQ: Vector Quantization.
2.2.29. V/UV decision: Decision whether the current frame is voiced or unvoiced or mixed voiced.
2.2.30. VXC: Vector eXcitation Coding. It is also called CELP (Coded Excitation Linear Prediction). In HVXC, no adaptive codebook is used.
2.2.31. white Gaussian noise: A noise sequence which has a Gaussian distribution.

A general glossary and list of symbols and abbreviations is located in Section 1.

2.3 Bitstream syntax

An MPEG-4 Natural Audio Object HVXC object type is transmitted in one or two Elementary Streams: The base layer stream and an optional enhancement layer stream.

The bitstream syntax is described in pseudo-C code.

2.3.1 Decoder configuration (HvxcSpecificConfig)

The decoder configuration information for HVXC object type is transmitted in the DecoderConfigDescriptor() of the base layer and the optional enhancement layer Elementary Stream (see Section 1, clause 1.6).

HVXC Base Layer -- Configuration

For HVXC object type in unscalable mode or as base layer in scalable mode the following HvxcSpecificConfig() is required:

[redacted]

```
    HVXCconfig();
}
```


HVXC Enhancement Layer -- Configuration

HVXC object type provides a 2 kbit/s base layer plus a 2 kbit/s enhancement layer scalable mode. In this scalable mode the basic layer configuration must be as follows:

```
HVXCvarMode = 0   HVXC fixed bit rate
HVXCrateMode = 0  HVXC 2 kbps
```

For the enhancement layer, there is no HvxcSpecificConfig() required:

```
HvxcSpecificConfig() {
}
```

Table 2.3.1 - Syntax of HVXCconfig()

Syntax	No. of bits	Mnemonic
HVXCconfig()		
{		
HVXCvarMode	**1**	**uimsbf**
HVXCrateMode	**2**	**uimsbf**
extensionFlag	**1**	**uimsbf**
if (extensionFlag) {		
< to be defined in MPEG-4 Version 2 >		
}		

Table 2.3.2 - HVXCvarMode

HVXCvarMode	Description
0	HVXC fixed bit rate
1	HVXC variable bit rate

Table 2.3.3 - HVXCrateMode

HVXCrateMode	HVXCrate	Description
0	2000	HVXC 2 kbit/s
1	4000	HVXC 4 kbit/s
2	3700	HVXC 3.7 kbit/s
3 (reserved)		

Table 2.3.4 - HVXC constants

NUM_SUBF1	NUM_SUBF2
2	4

2.3.2 Bitstream frame (alPduPayload)

The dynamic data for HVXC object type is transmitted as AL-PDU payload in the base layer and the optional enhancement layer Elementary Stream(see Section 1, clause 1.6).

HVXC Base Layer -- Access Unit payload

```
alPduPayload {
   HVXCframe();
}
```


HVXC Enhancement Layer -- Access Unit payload

To parse and decode the HVXC enhancement layer, information decoded from the HVXC base layer is required.

```
alPduPayload {
   HVXCenhaFrame();
}
```

Table 2.3.5 - Syntax of HVXCframe()

Syntax	No. of bits	Mnemonic

HVXCframe()
{
    if (HVXCvarMode ==0) {
        HVXCfixframe(HVXCrate)
    }
    else {
        HVXCvarframe()
    }
}		

2.3.2.1 HVXC bitstream frame

Table 2.3.6 - Syntax of HVXCfixframe(rate)

Syntax	No. of bits	Mnemonic
HVXCfixframe(rate) {		

```
    if (rate >= 2000){
        idLsp1()
        idVUV()
        idExc1()
    }
    if (rate >= 3700){
        idLsp2()
        idExc2(rate)
    }
```

Table 2.3.7 - Syntax of HVXCenhaFrame()

Syntax	No. of bits	Mnemonic
HVXCenhaFrame()		
{		
idLsp2()		
idExc2(4000)		
}		

Table 2.3.8 - Syntax of idLsp1()

Syntax	No. of bits	Mnemonic
idLsp1()		
{		
LSP1	**5**	**uimsbf**
LSP2	**7**	**uimsbf**
LSP3	**5**	**uimsbf**
LSP4	**1**	**uimsbf**
}		

Table 2.3.9 - Syntax of idLsp2()

Syntax	No. of bits	Mnemonic
idLsp2()		
{		
LSP5	**8**	**uimsbf**
}		

Table 2.3.10 - Syntax of idVUV()

Syntax	No. of bits	Mnemonic
idVUV()		
{		
VUV	**2**	**uimsbf**
}		

Table 2.3.11 - VUV (for fixed bit rate mode)

VUV	Description
0	Unvoiced Speech
1	Mixed Voiced Speech-1
2	Mixed Voiced Speech-2
3	Voiced Speech

Table 2.3.12 - Syntax of idExc1()

Syntax	No. of bits	Mnemonic
idExc1()		
{		

if (VUV!=0){		
Pitch	**7**	**uimsbf**
SE_shape1	**4**	**uimsbf**
SE_shape2	**4**	**uimsbf**
SE_gain	**5**	**uimsbf**
}		
else{		
for (sf_num=0;sf_num<NUM_SUBF1;sf_num++){		
VX_shape1 [sf_num]	**6**	**uimsbf**
}		
}		
}		

Table 2.3.13 - Syntax of idExc2(rate)

Syntax	No. of bits	Mnemonic
idExc2(rate)		
{		
if (VUV!=0){		
SE_shape3	**7**	**uimsbf**
SE_shape4	**10**	**uimsbf**
SE_shape5	**9**	**uimsbf**
if (rate>=4000){		
SE_shape6	**6**	**uimsbf**
}		
}		
else{		
for (sf_num=0;sf_num<NUM_SUBF2-1;sf_num++){		
VX_shape2[sf_num]	**5**	**uimsbf**
VX_gain2[sf_num]	**3**	**uimsbf**
}		
if (rate>=4000){		
VX_shape2[3]	**5**	**uimsbf**
VX_gain2[3]	**3**	**uimsbf**
}		
}		
}		

idLsp1(), idExc1(), idVUV() are treated as base layer in case of scalable mode.
idLsp2(), idExc2() are treated as enhancement layer in case of scalable mode.

Table 2.3.14 - Syntax of HVXCvarframe()

Syntax	No. of bits	Mnemonic
HVXCvarframe()		
{		
idvarVUV()		
idvarLsp1()		
idvarExc1()		
}		

Table 2.3.15 - Syntax of idvarVUV()

Syntax	No. of bits	Mnemonic
idvarVUV()		
{		
VUV	**2**	**uimsbf**
}		

VX_gain1 [sf_num] 4 uimsbf

Table 2.3.16 - VUV (for variable bit rate mode)

VUV	Description
0	Unvoiced Speech
1	Background Noise
2	Mixed Voiced Speech
3	Voiced Speech

Table 2.3.17 - Syntax of idvarLsp1()

Syntax	No. of bits	Mnemonic
idvarLsp1()		
{		
if (VUV != 1){		
LSP1	**5**	**uimsbf**
LSP2	**7**	**uimsbf**
LSP3	**5**	**uimsbf**
LSP4	**1**	**uimsbf**
}		
}		

Table 2.3.18 - Syntax of idvarExc1()

Syntax	No. of bits	Mnemonic
idvarExc1()		
{		
if (VUV != 1){		
if (VUV != 0){		
Pitch	**7**	**uimsbf**
SE_Shape1	**4**	**uimsbf**
SE_Shape2	**4**	**uimsbf**
SE_Gain	**5**	**uimsbf**
}		
else{		
for (sf_num=0;sf_num<NUM_SUBF1;sf_num++){		
VX_gain1[sf_num]	**4**	**uimsbf**
}		
}		
}		
}		

2.4 Bitstream semantics

2.4.1 Decoder configuration (HvxcSpecificConfig)

HVXCvarMode: A flag indicating HVXC variable rate mode(Table 2.3.1).
HVXCrateMode: A 2 bit field indicating HVXC bit rate mode(Table 2.3.1).
extensionFlag: A flag indicating the presence of MPEG-4 Version 2 data (Table 2.3.1, for future use).

2.4.2 Bitstream frame (alPduPayload)

LSP1: This 5 bits field represents the index of the first stage LSP quantization (base layer, Tables 2.3.8 and 2.3.17).
LSP2: This 7 bits field represents the index of the second stage LSP quantization (base layer, Tables 2.3.8 and 2.3.17).

LSP3: This 5 bits field represents the index of the second stage LSP quantization (base layer, Tables 2.3.8 and 2.3.17).
LSP4: This 1 bit field represents the flag whether a interframe prediction is used or not in the second stage LSP quantization (base layer, Tables 2.3.8 and 2.3.17).
LSP5: This 8 bits field represents the index of the third stage LSP quantization (enhancement layer, Table 2.3.9).
VUV: This 2 bits field represents V/UV decision mode. It should be noted that this field has a different meaning according to HVXC variable rate mode(Tables 2.3.10 and 2.3.15)
Pitch: This 7 bits field represents the index of the linearly quantized pitch lag ranging from 20 to 147 samples(Tables 2.3.12 and 2.3.18).
SE_shape1: This 4 bits field represents the index of the spectral envelope shape (base layer, Tables 2.3.12 and 2.3.18).
SE_shape2: This 4 bits field represents the index of the spectral envelope shape (base layer, Tables 2.3.12 and 2.3.18).
SE_gain: This 5 bits field represents the index of the spectral envelope gain (base layer, Tables 2.3.12 and 2.3.18).
VX_shape1[sf_num]: This 6 bits field represents the index of the sf_num-th subframe's VXC shape (base layer, Tables 2.3.12 and 2.3.18).
VX_gain1[sf_num]: This 4 bits field represents the index of the sf_num-th subframe's VXC gain (base layer, Tables 2.3.12 and 2.3.18).
SE_shape3: This 7 bits field represents the index of the spectral envelope shape (enhancement layer, Table 2.3.13).
SE_shape4: This 10 bits field represents the index of the spectral envelope shape (enhancement layer, Table 2.3.13).
SE_shape5: This 9 bits field represents the index of the spectral envelope shape (enhancement layer, Table 2.3.13).
SE_shape6: This 6 bits field represents the index of the spectral envelope shape (enhancement layer, Table 2.3.13).
VX_shape2[sf_num]: This 5 bits field represents the index of the sf_num-th subframe's VXC shape (enhancement layer, Table 2.3.13).
VX_gain2[sf_num]: This 3 bits field represents the index of the sf_num-th subframe's VXC gain (enhancement layer, Table 2.3.13).

2.5 HVXC decoder tools

2.5.1 Overview

HVXC provides an efficient coding scheme for Linear Predictive Coding (LPC) residuals based on harmonic and stochastic vector representation. Vector quantization (VQ) of the spectral envelope of LPC residuals with a weighted distortion measure is used when the signal is voiced. Vector excitation coding (VXC) is used when the signal is unvoiced. The major algorithmic features are:

- Weighted VQ of variable dimension spectral vector.
- A fast harmonic synthesis algorithm by IFFT.
- Interpolative coder parameters for speed/pitch control

Also, functional features include:

- As low as 33.5 ms of total algorithmic delay
- 2.0-4.0 kbps scalable mode
- Variable bit rate coding for rates less than 2.0 kbps

2.5.1.1 Framing structure and block diagram of the decoder

Figure 2.5.1 shows the overall structure of the HVXC decoder. The HVXC decoder tools allow decoding of speech signals at 2 kbit/s and higher, up to 4 kbit/s. HVXC decoder tools also allow decoding with variable bit rate mode at an average bit rate of around 1.2-1.7 kbps. The basic decoding process is composed of four steps; de-quantization of parameters, generation of excitation signals for voiced frames by sinusoidal synthesis (harmonic synthesis) and noise component addition, generation of excitation signals for unvoiced frames by codebook look-up, and LPC synthesis. To enhance the synthesized speech quality spectral post-filtering is used.

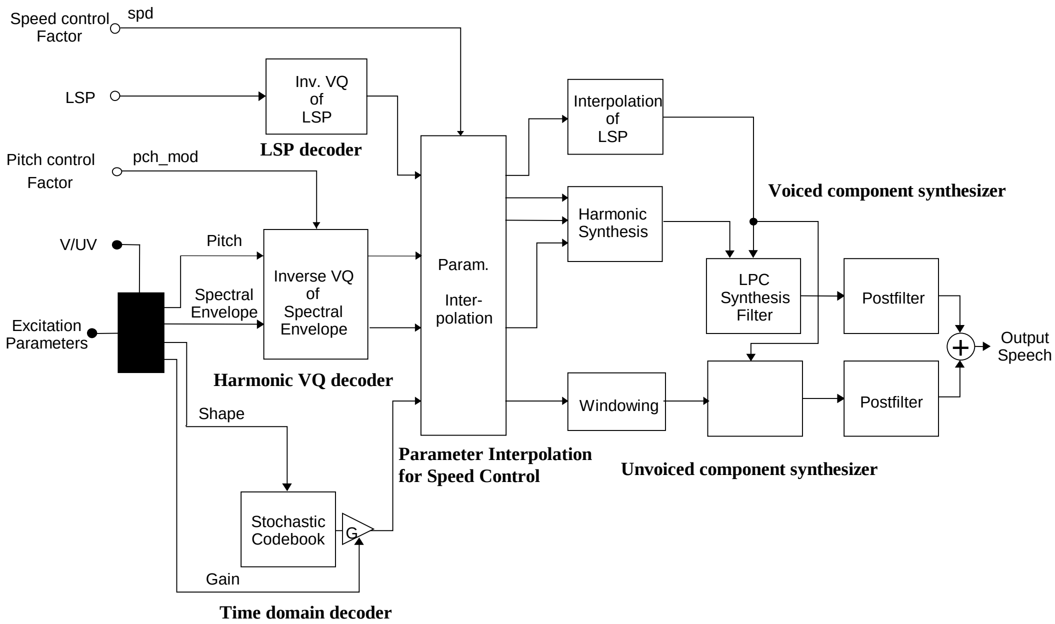


Figure 2.5.1 - Blockdiagram of the HVXC decoder

2.5.1.2 Delay mode

HVXC coder/decoder supports low/normal delay encode/decode mode, allowing any combinations of delay mode at 2.0-4.0 kbps with a scalable scheme.The figure below shows the framing structure of each delay mode. The frame length is 20 ms for all the delay modes. For example, use of low delay encode and low delay decode mode results in a total coder/decoder delay of 33.5 ms.
In the encoder, the algorithmic delay could be selected to be either 26 ms or 46 ms. When 46 ms delay is selected, one frame look ahead is used for pitch detection. When 26 ms delay is selected, only the current frame is used for pitch detection. For both cases, syntax is common, all the quantizers are common, and bitstreams are compatible.
In the decoder the algorithmic delay can be selected to be either 10 ms (normal delay mode) or 7.5 ms (low delay mode). When 7.5 ms delay is selected, the decoder frame interval is shifted by 2.5 ms (20 samples) compared to the 10 ms delay mode. In this case, the excitation generation and LPC synthesis phase is shifted by 2.5 ms. Again, for both cases, syntax is common, all the quantizers are common, and bitstreams are compatible.

In summary, any independent choice of encoder/decoder delay from the following combination is possible:
Encoder delay: 26 ms or 46 ms
Decoder delay: 10 ms or 7.5 ms
One or some combinations of the delay mode shall be supported depending on the application.

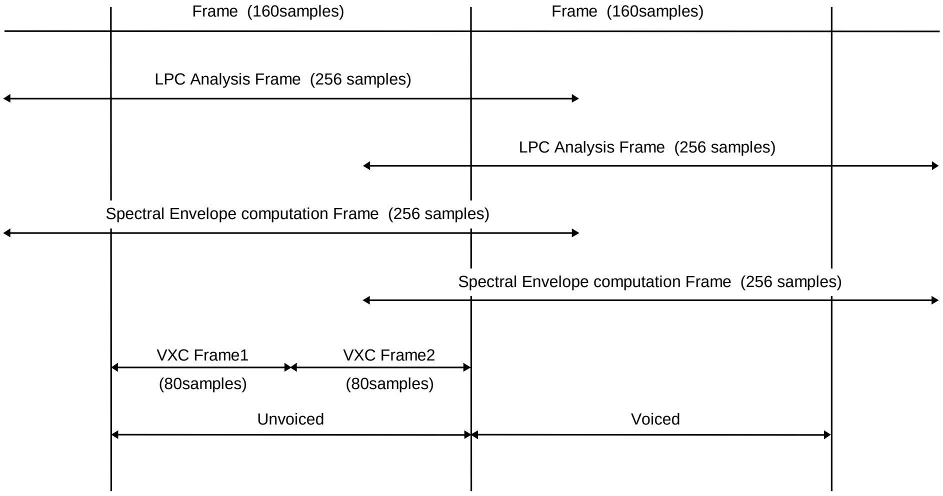
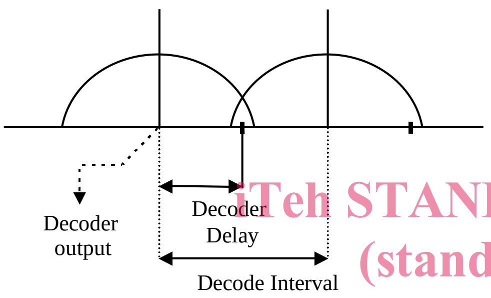
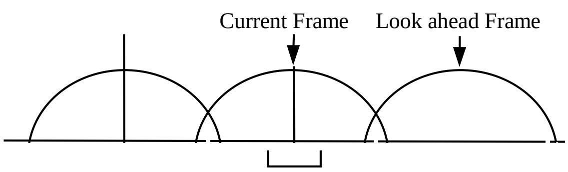


Figure 2.5.2 - Framing structure of HVXC

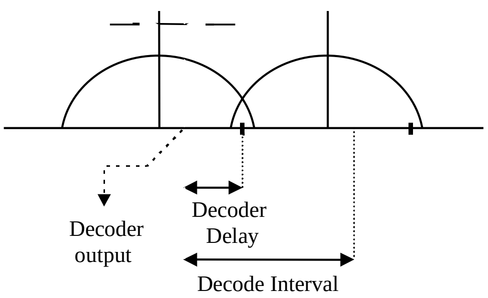
Normal Delay Decoder (10 ms)

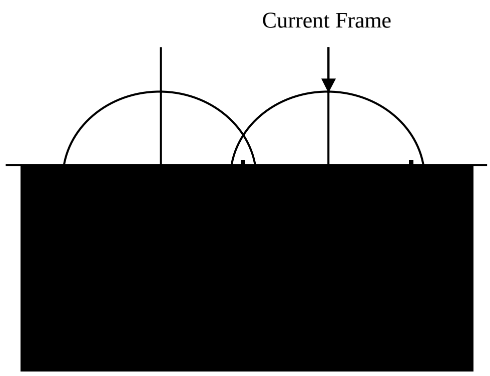


Normal Delay Encoder (46 ms)



Low Delay Decoder (7.5 ms)





2.5.2 LSP decoder

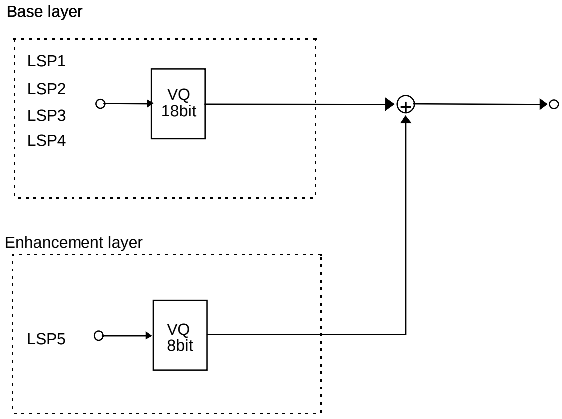


Figure 2.5.4 - LSP decoder

2.5.2.1 Tool description

For the quantization of the LSP parameters, a multistage quantizer structure is used and the output vectors from each stage have to be summed up to obtain the LSP parameters.

When the bitrate is 2 kbps, the LSPs of the current frame, which are coded by split and two-stage vector quantization,are decoded using a two-stage decoding process. At 4 kbps, a 10-dimensional vector quantizer, which has an 8 bit codebook, is added to the bottom of the LSP quantizer scheme of 2.0 kbps coder. The bits needed for the LSPs are increased from 18bits/20msec to 26bits/20msec.

Table 2.5.1 - Configuration of the multistage LSP VQ

1st stage	10 LSP VQ	5 bits
2nd stage	(5+5) LSP VQ	(7+5+1) bits
3rd stage	10 LSP VQ	8 bits

2.5.2.2 Definitions

Definitions of constants

$LPCORDER$: LPC analysis order (=10)

$dim[][]$: dimensions for the split vector quantization

min_gap : minimum distance between adjacent LSP coefficients (base layer, =4.0/256.0)

$ratio\_predict$: LSP interframe prediction ratio (=0.7)

$THRSLD\_L$: minimum distance between adjacent LSP coefficients
(low frequency part of enhancement layer, =0.020)

$THRSLD\_M$: minimum distance between adjacent LSP coefficients
(middle frequency part of enhancement layer, =0.020)

$THRSLD\_H$: minimum distance between adjacent LSP coefficients
(high frequency part of enhancement layer, =0.020)

Definitions of variables

$qLsp[]$: quantized LSP parameters

$LSP1$: the index of the first stage LSP quantization (base layer)

$LSP2$, $LSP3$: the indices of the second LSP quantization (base layer)

$LSP4$: the flag showing whether a interframe prediction is used or not (base layer)

$LSP5$: the index of the third LSP quantization (enhancement layer)

$lsp\_tbl[][][]$: look-up tables for the first stage decoding process

$d\_tbl[][][]$: look-up tables for the second stage decoding process of the VQ without interframe prediction

$pd\_tbl[][][]$: look-up tables for the second stage decoding process of the VQ with interframe prediction.

$vqLsp[][]$: look-up table for the enhancement layer
$sign$: sign of code vector for the second stage decoding process
idx : unpacked index for the second stage decoding process
$lsp\_predict[]$: the LSPs predicted from $lsp\_previous[]$ and $lsp\_first[]$
$lsp\_previous[]$: the LSPs decoded at the previous frame
$lsp\_current[]$: the LSPs decoded at the current frame
$lsp\_first[]$: the LSPs decoded at the first stage decoding process

2.5.2.3 Decoding process

The decoding process of the LSP parameters for the base layer (2.0 kbps) is the same as that of the narrowband CELP. The decoding process is as described below.

Converting indices to LSPs

The LSPs of the current frame ($lsp\_current[]$), which are coded by split and two-stage vector quantization, are decoded with a two-stage decoding process. The dimension of each vector is described in the tables below. The **LSP1** and **LSP2, LSP3** represent indices for the first and the second stage respectively.

Table 2.5.2 - Dimension of the first stage LSP vector

Split Vector Index: i	Vector Dimension: $dim[0][i]$
0	10

Table 2.5.3 - Dimension of the second stage LSP vector

Split Vector Index: i	Vector Dimension: $dim[1][i]$
0	5
1	5

In the first stage, the LSP vector of the first stage $lsp\_first[]$ is decoded simply by looking up the table $lsp\_tbl[][][]$. (The table $lsp\_tbl[][]$ is shown in Annex 2.D)

```
for (i = 0; i < dim[0][0]; i++) {
   lsp_first[i] = lsp_tbl[0][LSP1][i];
}
```

In the second stage, there are two types of decoding processs, namely, decoding process of VQ without interframe prediction and VQ with interframe prediction. The flag **LSP4** indicates which process should be selected.

Table 2.5.4 - Decoding process for the second stage

LSP Index: **LSP4**	Decoding process
1	VQ with interframe prediction

Decoding process of VQ without interframe prediction

In order to obtain LSPs of the current frame $lsp\_current[]$, the decoded vectors in the second stage are added to the decoded first stage LSP vector $lsp\_first[]$. The MSB of the **LSP2** and **LSP3** represents the sign of the decoded vector, and the remaining bits represent the index for the table $d\_tbl[][][]$. (The table $d\_tbl[][][]$ is shown in Annex 2.D)

```
sign = LSP2>>6;
idx = LSP2&0x3f;
if (sign == 0) {
   for (i = 0; i < dim[1][0]; i++) {
      lsp_current[i] = lsp_first[i] + d_tbl[0][idx][i];
```

```
      }
   }
   else {
      for (i = 0; i < dim[1][0]; i++) {
         lsp_current[i] = lsp_first[i] - d_tbl[0][idx][i];
      }
   }
   sign = LSP3>>4;
   idx = LSP3&0x0f;
   if (sign == 0) {
      for (i = 0; i < dim[1][1]; i++) {
         lsp_current[dim[1][0]+i] = lsp_first[dim[1][0]+i] + d_tbl[1][idx][i];
      }
   }
   else {
      for (i = 0; i < dim[1][1]; i++) {
         lsp_current[dim[1][0]+i] = lsp_first[dim[1][0]+i] - d_tbl[1][idx][i];
      }
   }
```

Decoding process of VQ with interframe prediction

In order to obtain LSPs of the current frame $lsp\_current[]$, the decoded vectors of the second stage are added to the LSP vector $lsp\_predict[]$ which are predicted from the decoded LSPs of the previous frame $lsp\_previous[]$ and the decoded first stage LSP vector $lsp\_first[]$. As with the decoding process of VQ without interframe prediction, the MSB of the **LSP2** and **LSP3** represents the sign of the decoded vector, and the remaining bits represent the index for the table $pd\_tbl[][][]$. (The table $pd\_tbl[][][]$ is shown in Annex 2.D)

```
   for (i = 0; i < LPCORDER; i++) {
      lsp_predict[i] = (1-ratio_predict)*lsp_first[i]
                       + ratio_predict*lsp_previous[i]
   }

   sign = LSP2>>6;
   idx = LSP2&0x3f;
   if (sign == 0) {
      for (i = 0; i < dim[1][0]; i++) {
         lsp_current[i] = lsp_predict[i] + pd_tbl[0][idx][i];
      }
   }
   else {
      for (i = 0; i < dim[1][0]; i++) {
         lsp_current[i] = lsp_predict[i] - pd_tbl[0][idx][i];
      }
   }
   sign = LSP3>>4;
   idx = LSP3&0x0f;
   if (sign == 0) {
      for (i = 0; i < dim[1][1]; i++) {
         lsp_current[dim[1][0]+i] = lsp_predict[dim[1][0]+i] + pd_tbl[1][idx][i];
      }
   }
   else {
      for (i = 0; i < dim[1][1]; i++) {
         lsp_current[dim[1][0]+i] = lsp_predict[dim[1][0]+i] - pd_tbl[1][idx][i];
      }
   }
```


Stabilization of LSPs

The decoded LSPs $lsp\_current[]$ are stabilized in order to ensure stability of the LPC synthesis filter which is derived from the decoded LSPs. The decoded LSPs are arranged in ascending order, having a minimum distance between adjacent coefficients.

```
   for (i = 0; i < LPCORDER; i++) {
      if (lsp_current[i] < min_gap) lsp_current[i] = min_gap;
   }
   for (i = 0; i < LPCORDER-1; i++) {
      if (lsp_current[i+1]-lsp_current[i] < min_gap) {
```