Chapter 4

VECTOR QUANTIZATION TECHNIQUES

4.1 INTRODUCTION

Quantization is a process of mapping an infinite set of scalar or vector quantities by a finite set of scalar or vector quantities. Quantization has applications in the areas of signal processing, speech processing and Image processing. In speech coding, quantization is required to reduce the number of bits used to represent a sample of speech signal. When less number of bits is used to represent a sample the bit-rate, complexity and memory requirement gets reduced. Quantization results in the loss in the quality of a speech signal, which is undesirable. So a compromise must be made between the reduction in bit-rate and the quality of speech signal.

Two types of quantization techniques exist. They are scalar quantization and vector quantization. “Scalar quantization deals with the quantization of samples on a sample by sample basis”, while “vector quantization deals with quantizing the samples in groups called vectors”. Vector quantization increases the optimality of a quantizer at the cost of increased computational complexity and memory requirements.

Shannon theory states that “quantizing a vector is more effective than quantizing individual scalar values in terms of spectral distortion”. According to Shannon the dimension of a vector chosen
greatly affects the performance of quantization. Vectors of larger dimension produce better quality compared to vectors of smaller dimension and in vectors of smaller dimension the transparency in the quantization is not good at a particular bit-rate chosen [8]. This is because in vectors of smaller dimension the correlation that exists between the samples is lost and the scalar quantization itself destroys the correlation that exists between successive samples. So the quality of the quantized speech signal gets lost. Therefore, quantizing correlated data requires techniques that preserve the correlation between the samples, which is achieved by the vector quantization technique (VQ). Vector quantization is the simplification of scalar quantization. Vectors of larger dimension produce transparency in quantization at a particular bit-rate chosen. In Vector quantization the data is quantized in the form of contiguous blocks called vectors rather than individual samples. But later with the development of better coding techniques, it is made possible that transparency in quantization can also be achieved even for vectors of smaller dimension. In this thesis quantization is performed on vectors of full length and on vectors of smaller dimensions for a given bit-rate [4, 50].

An example of two dimensional vector quantizer is shown in Fig 4.1. The two dimensional region shown in Fig 4.1 is called as the voronoi region, which in turn contains several numbers of small hexagonal regions. The hexagonal regions defined by the red borders are called as the encoding regions. The green dots represent the
vectors to be quantized which fall in different hexagonal regions and the blue circles represent the codeword’s (centroids). The vectors (green dots) falling in a particular hexagonal region is best represented by the codeword (blue circle) falling in that hexagonal region [51-54].

Fig 4.1 Two Dimensional Vector Quantizer

Vector quantization technique has become a great tool with the development of non variational design algorithms like the Linde, Buzo, Gray (LBG) algorithm. On the other hand besides spectral distortion the vector quantizer is having its own limitations like the computational complexity and memory requirements required for the searching and storing of the codebooks. For applications requiring higher bit-rates the computational complexity and memory requirements increases exponentially. The block diagram of a vector quantizer is shown in Fig 4.2.
Let \( s_k = \left[ s_1, s_2, \ldots, s_N \right]^T \) be an ‘N’ dimensional vector with real valued samples in the range \( 1 \leq k \leq N \). The superscript ‘T’ in the vector \( s_k \) denotes the transpose of the vector. In vector quantization, a real valued ‘N’ dimensional input vector \( s_k \) is matched with the real valued ‘N’ dimensional codewords of the codebook \( N = 2^b \). The codeword that best matches the input vector with lowest distortion is taken and the input vector is replaced by it. The codebook consists of a finite set of codewords \( C = C_i, 1 \leq i \leq L \), where \( C_i = \left[ C_{i1}, C_{i2}, \ldots, C_{iN} \right]^T \), where ‘C’ is the codebook, ‘L’ is the length of the codebook and \( C_i \) denote the \( i^{th} \) codeword in a codebook. In Fig 4.2 \( s(n) \) represents the line spectral frequencies (LSF) to be quantized. In this work the number of line spectral frequencies per frame is ten and they are extracted from the eleven linear predictive coefficients of a frame using the matlab command poly2lsf.
4.2 LINE SPECTRAL FREQUENCIES

The parameters used in the analysis and synthesis of the speech signals are the LPC coefficients. In speech coding the quantization is not performed directly on the LPC coefficients, but the quantization is performed by transforming the LPC coefficients to other forms which ensure filter stability after quantization. Another reason for not using LPC coefficients is that LPC coefficients have a wide dynamic range and so the LPC filter easily becomes unstable after quantization. So LPC coefficients are not used for quantization. The alternative to LPC coefficients is the use of line spectral frequency (LSF) parameters which ensure filter stability after quantization. The filter stability is checked easily just by observing the order of the LSF samples in an LSF vector after quantization. If the LSF samples in a vector are in the ascending or descending order the filter stability is ensured otherwise the filter stability cannot be ensured [54-58].

The transfer function of the LPC filter is given by equation (4.1)

$$H(z) = \frac{1}{A_p(z)}$$  \hspace{1cm} (4.1)

where

$$A_p(z) = 1 + \sum_{k=1}^{p} \alpha_k z^{-k}$$  \hspace{1cm} (4.2)

In determining the line spectral frequencies, $A_p(z)$ is decomposed into two transfer functions having even and odd symmetry. This is done by finding the sum and difference of $A_p(z)$ and its conjugate.
$B_p(z)$. These transfer functions are represented by $H_{p+1}(z)$ and $Q_{p+1}(z)$ with $K_{p+1} = \pm 1$. Where $K_{p+1} = \pm 1$ gives the conditions at the glottis corresponding to the artificial boundary, $K_{p+1} = +1$ corresponds to the opening at the glottis and $K_{p+1} = -1$ corresponds to the closure at the glottis.

For $K_{p+1} = +1$ \( H_{p+1}(z) = A_p(z) - B_p(z) \) (Difference filter) \( (4.3) \)

For $K_{p+1} = -1$ \( Q_{p+1}(z) = A_p(z) + B_p(z) \) (Sum filter) \( (4.4) \)

\[
A_p(z) = \frac{1}{2} \left[ H_{p+1}(z) + Q_{p+1}(z) \right] \quad (4.5)
\]

As $K_{p+1} = \pm 1$ two roots exist and the order of equations (4.3) and (4.4) is reduced to

\[
H'(z) = \frac{H_{p+1}(z)}{(1-z)} \quad (4.6)
\]

\[
= A_0 z^p + A_1 z^{(p-1)} + \ldots + A_p
\]

and

\[
Q'(z) = \frac{Q_{p+1}(z)}{(1+z)} \quad (4.7)
\]

\[
= B_0 z^p + B_1 z^{(p-1)} + \ldots + B_p
\]

where

\[
A_0 = 1 \quad (4.8)
\]

\[
B_0 = 1 \quad (4.9)
\]

\[
A_k = (\alpha_k - \alpha_{p+1-k}) + A_{k-1} \quad (4.10)
\]
\[ B_k = (\alpha_k - \alpha_{p+1-k}) - B_{k-1} \]  
(4.11)

for \( k = 1, \ldots, p \)

The angular positions of the roots of \( H'(z) \) and \( Q'(z) \) gives us the line spectral frequencies and occurs in complex conjugate pairs. The line spectral frequencies range from \( 0 \leq \omega_i \leq \pi \).

The line spectral frequencies have the following properties:

- All the roots of \( H'(z) \) and \( Q'(z) \) must lie on the unit circle which is the required condition for stability.
- The roots of \( H'(z) \) and \( Q'(z) \) are arranged in an alternate manner on the unit circle i.e., \( 0 \leq \omega_{q,0} < \omega_{p,0} < \omega_{q,1} < \omega_{p,1} \ldots \leq \pi \).

The roots of equation (4.6) is obtained using the real root method [31] and is explained in the following section.

### 4.2.1 Real Root Method

The coefficients of equations (4.6) and (4.7) are symmetrical and so the order ‘p’ of equations (4.6) and (4.7) get reduces to ‘p/2’. Then

\[ H'(z) = z^{p/2} \left[ A_0 \left( \frac{p}{2} + z^{-\frac{p}{2}} \right) + A_1 \left( z^{\frac{p}{2}-1} + z^{-\frac{p}{2}-1} \right) + \ldots + A_{\frac{p}{2}} \right] \]  
(4.12)

and

\[ Q'(z) = z^{p/2} \left[ B_0 \left( \frac{p}{2} + z^{-\frac{p}{2}} \right) + B_1 \left( z^{\frac{p}{2}-1} + z^{-\frac{p}{2}-1} \right) + \ldots + B_{\frac{p}{2}} \right] \]  
(4.13)
As the roots of $H'(z)$ and $Q'(z)$ lie on the unit circle, they are evaluated on the unit circle.

Let $Z = e^{j\omega}$ then $\cos \omega = \frac{Z^1 + Z^{-1}}{2}$ and

$$H'(z) = 2e^{j\frac{p\omega}{2}} \left[ A_0 \left( \cos \left( \frac{p}{2} \omega \right) \right) + A_1 \left( \cos \left( \frac{p-2}{2} \omega \right) \right) + \ldots + \frac{1}{2} A_{\frac{p}{2}} \right]$$

(4.14)

$$Q'(z) = 2e^{j\frac{p\omega}{2}} \left[ B_0 \left( \cos \left( \frac{p}{2} \omega \right) \right) + B_1 \left( \cos \left( \frac{p-2}{2} \omega \right) \right) + \ldots + \frac{1}{2} B_{\frac{p}{2}} \right]$$

(4.15)

When $x = \cos(\omega)$ the LSF parameters are given by

$$\text{LSF}(i) = \frac{\cos^{-1}(x_i)}{2\pi T}$$

(4.16)

Where ‘T’ is the sampling period.

**4.3 CODEBOOK DESIGN**

Vector quantization of speech signals requires the generation of codebooks. The codebooks are designed using an iterative algorithm called Linde, Buzo and Gray (LBG) algorithm. The input to the LBG algorithm is a training sequence. The training sequence is the concatenation of a set LSF vectors obtained from people of different groups and of different ages. The speech signals used to obtain training sequence must be free of background noise. The speech signals can be recorded in sound proof booths, computer rooms and open environments. In this work the speech signals are recorded in
computer rooms. In practice speech data base like TIMIT database is available for use in speech coding and speech recognition.

The codebook generation using LBG algorithm requires the generation of an initial codebook, which is the centroid or mean obtained from the training sequence. The centroid obtained is then split into two centroids or codewords using the splitting method. The iterative LBG algorithm splits these two codeword’s into four, four into eight and the process continues till the required numbers of codewords in the codebook are obtained [59-61].

4.3.1 LBG Algorithm

The flow chart of LBG algorithm is shown in Fig 4.3. The LBG algorithm is properly implemented by a recursive procedure given below:

1. Initially the codebook generation requires a training sequence of LSF parameters which is the input to LBG algorithm. The training sequence is obtained from a set of speech samples recorded from different groups of people in a computer room.

2. Let ‘R’ be the region of the training sequence.

3. Obtain an initial codebook from the training sequence, which is the centroid or mean of the training sequence and let the initial codebook be ‘C’.
4. Split the initial codebook \( C \) into a set of codewords \( C_n^+ \) and \( C_n^- \) where

\[
C_n^+ = C (1 + \varepsilon)
\]
\[ C_n^- = C(1 - \varepsilon) \]  \hspace{1cm} (4.18)

\[ \varepsilon = 0.01 \] is the minimum error to be obtained between old and new codewords.

5. Compute the difference between the training sequence and each of the codeword’s \( C_n^+ \) and \( C_n^- \) and let the difference be ‘D’.

6. Split the training sequence into two regions \( R_1 \) and \( R_2 \) depending on the difference ‘D’ between the training sequence and the codeword’s \( C_n^+ \) and \( C_n^- \). The training vectors closer to \( C_n^+ \) falls in the region \( R_1 \) and the training vectors closer to \( C_n^- \) falls in the region \( R_2 \).

7. Let the training vectors falling in the region \( R_1 \) be \( TV_1 \) and the training sequence vectors falling in the region \( R_2 \) be \( TV_2 \).

8. Obtain the new centroid or mean for \( TV_1 \) and \( TV_2 \). Let the new centroids be \( C_{R1} \) and \( C_{R2} \).

9. Replace the old centroids \( C_n^+ \) and \( C_n^- \) by the new centroids \( C_{R1} \) and \( C_{R2} \).

10. Compute the difference between the training sequence and the new centroids \( C_{R1} \) and \( C_{R2} \) and let the difference be \( D^1 \).

11. Repeat steps 5 to 10 until \[ \frac{D^1 - D}{D} < \varepsilon \]

12. Repeat steps 4 to 11 till the required number of codewords in the codebook are obtained.
where \( N = 2^b \) represents the number of codewords in the codebook and ‘\( b \)’ represents the number of bits used for codebook generation, ‘\( D \)’ represents the difference between the training sequence and the old codewords and \( D^1 \) represents the difference between the training sequence and the new codewords.

4.4 SPECTRAL DISTORTION

The quality of the speech signal is an important parameter in speech coders and is measured in terms of spectral distortion measured in decibels (dB). The spectral distortion is measured between the LPC power spectra of the quantized and unquantized speech signals. The spectral distortion is measured frame wise and the average or mean of the spectral distortion calculated over all frames is taken as the final value of the spectral distortion. For a quantizer to be transparent the mean of the spectral distortion must be less than 1 dB without any audible distortion in the reconstructed speech. But the mean of the spectral distortion is not a sufficient measure to find the performance of a quantizer, this is because the human ear is sensitive to large quantization errors that occur occasionally. So in addition to measuring the mean of the spectral distortion it is also necessary to have another measure of quality which is the percentage number of frames having a spectral distortion greater than 2dB and less than 4dB and the percentage number of frames having a spectral distortion greater than 4dB. The frames
having spectral distortion between 2 to 4dB and greater than 4dB are called as outlier frames [54].

In order to measure objectively the distortion between the quantized and unquantized outputs, a method called the spectral distortion is often used in narrowband speech coding. For an $i^{th}$ frame, the spectral distortion (in dB), $SD_i$, is given by equation (4.19).

$$SD_i = \sqrt{\frac{1}{f_2 - f_1} \int_{f_1}^{f_2} \left[ 10 \log_{10} s_i(f) - 10 \log_{10} \hat{s}_i(f) \right]^2 \, df} \text{ dB} \quad (4.19)$$

Where $s_i(f)$ and $\hat{s}_i(f)$ are the LPC power spectra of the unquantized and quantized $i^{th}$ frame respectively. The frequency ‘$f$’ is in Hz and the frequency range is given by $f_1$ and $f_2$. The frequency range used in practice for narrowband speech coding is 0 to 4000 Hz [12, 33].

The average or mean of the spectral distortion $SD$ is given by equation (4.20)

$$SD = \frac{1}{N} \sum_{i=1}^{N} SD_i \quad (4.20)$$

The conditions for transparent speech coding are:

- The average or mean of the spectral distortion (SD) must be less than or equal to 1dB.
- There must be no outlier frames having a spectral distortion greater than 4dB.
- The number of outlier frames between 2 to 4dB must be less than 2%.

These three conditions are required to evaluate the performance of a quantizer. At a given bit-rate, an optimization process has to be
carried out so as to obtain better performance i.e., accepting a large average spectral distortion for a few outliers.

4.5 WEIGHTED LSF DISTANCE MEASUREMENT

In the design of a vector quantizer instead of using the mean squared error (MSE) distance measure the weighted LSF distance measurement is used. This is done to place emphasis on the low frequency LSFs and on the LSFs with higher power spectrum. The weights used are of two types. They are static or dynamic [54].

- Fixed or Static weights \( \{S_i\} \) : These are used to place emphasis on the low frequency LSFs in order to account for the sensitivity of human ear for low and high frequency LSFs.

- Varying or Dynamic weights \( \{W_i\} \) : These are used to place emphasis on the LSFs with high power spectrum.

The weighted distance measurement between the original and the contiguous or approximated vectors is given by equation (4.21)

\[
D_w(f, \hat{f}) = \sum_{i=1}^{10} S_i W_i (f_i - \hat{f}_i)^2
\]

(4.21)

Where \( D_w(f, \hat{f}) \) is the weighted distance measure, ‘f’ is the original vector, \( \hat{f} \) is the approximated vector and \( f_i, \hat{f}_i \) denote the \( i^{th} \) sample in the original and approximated LSF vectors respectively.

The static weights are given by equation (4.22)

\[
S_i = \begin{cases} 
1.0 & 1 \leq i \leq 8 \\
0.8 & i = 9 \\
0.4 & i = 10 
\end{cases}
\]

(4.22)
The dynamic weights are given by equation (4.23)

\[ W_i = \left[ PS(f) \right]^r \] (4.23)

Where \( PS(f) \) is the LPC power spectrum and ‘r’ is a constant whose value is taken based on experimental results and is 0.15 [54].

### 4.6 VECTOR QUANTIZATION TECHNIQUES

There exist a number of vector quantization techniques each one is having its own advantages and disadvantages. Each technique is developed to decrease the parameters like spectral distortion, computational complexity and memory requirements. The vector quantization techniques that exist are the Split Vector Quantization (SVQ) technique, Multistage Vector Quantization (MSVQ) technique, Split-Multistage Vector Quantization (S-MSVQ) technique and Switched Split Vector Quantization (SSVQ) technique. As marketability and cost of a product depends on the complexity and memory requirements, the performance of the vector quantization techniques is measured in terms of spectral distortion in decibels, computational complexity in kilo flops per frame and memory requirements in floats. The performance of a vector quantization technique mainly depends on how efficiently the codebook is generated. The codebook is generated efficiently using a large training set and using more number of bits for codebook generation. The goal involved in the design of each vector quantization technique is to make the technique to use more number of training vectors and less
number of bits for codebook generation, there by the spectral distortion, computational complexity and memory requirements get reduced. It has been observed that as the number of bits used for codebook generation decreases the computational complexity and memory requirements decreases but the spectral distortion increases. This increase in spectral distortion is reduced by increasing the number of training vectors used for codebook generation [62-71].

4.6.1 Unconstrained Vector Quantization

The block diagram of an Unconstrained Vector Quantizer (UVQ) is shown in Fig 4.4.

![Block diagram of Unconstrained Vector Quantizer](image)

Fig 4.4 Block diagram of Unconstrained Vector Quantizer

Unconstrained Vector Quantization technique is the most awful vector quantization technique used for achieving lowest distortion at a given bit-rate and dimension. In LPC-10 the order of the filter chosen
is 10 and so the length of each LSF vector is 10. In Unconstrained Vector Quantization technique the quantization is done on vectors of full length i.e., vectors containing all the ten line spectral frequencies. In Fig 4.4 $S_1$, $S_2$, $S_3$……$S_n$ are the input LSF vectors to be quantized using the Unconstrained Vector Quantizer.

The main advantage of UVQ is that it is expected to give lowest spectral distortion for a given bit-rate as the correlation that exists between the samples of a vector is preserved. But the disadvantage of UVQ is that as vectors of full length are used, at higher bit-rates the complexity and memory requirements increases in an exponential manner making it impractical for applications requiring higher bit-rates. Another problem with UVQ is that the generation of codebook becomes a difficult task on general purpose computers as the memory available with them is limited. So the number of training vectors used for codebook generation must be limited in number or the length of each vector must be reduced. In practice on general purpose computers, using UVQ technique the generation of codebook at higher bit-rates is a difficult task, even though the training data contains less number of training vectors. But the number of training vectors required to generate the codebook must be large than the number of codewords in a codebook otherwise there will be too much over fitting of the training set [54].

The computational complexity and memory requirements of a ‘b’ bit, ‘n’ dimensional vector quantizer are calculated as follows [54]:
To calculate the mean square error (MSE) between two vectors of ‘n’ dimension, ‘n’ subtractions, ‘n’ multiplications and \( n-1 \) additions are required. So a total of \( 3n-1 \) flops are required.

To search a codebook of \( 2^b \) code vectors, \( (3n-1)2^b \) flops are required in addition to the minimum distortion search requiring \( 2^b-1 \) flops.

So the number of computations made by a ‘b’ bit, ‘n’ dimensional vector quantizer is

\[
\text{Total complexity} = (3n-1)2^b + 2^b-1
\]

\[= 3n2^b -1 \text{ flops per vector.} \quad (4.24)\]

In computing the complexity each addition, multiplication and comparison is considered as one floating point operation. So a ‘b’ bit ‘n’ dimensional vector quantizer requires a codebook of \( 2^b \) code words, needs to store \( n2^b \) floating point values and computes \( (3n-1)2^b \) flops per vector. In the design of a vector quantizer if weighted distance measure is used instead of mean square error distance measure the complexity increases from \( (3n-1)2^b \) to \( 4n2^b-1 \) flops per vector.

The computational complexity of an Unconstrained Vector Quantizer is given by equation (4.25)

\[
\text{Complexity}_{UVQ} = (4n2^b -1) \text{ flops/frame} \quad (4.25)
\]

where

\( n \) is the dimension of the vector
\( b \) is the number of bits allocated to the vector quantizer.

The memory requirement of an Unconstrained Vector Quantizer is given by equation (4.26)

\[
\text{Memory}_{UVQ} = n^2 b \text{ floats}
\]  

(4.26)

### 4.6.2 Product Code Vector Quantization

Exhaustive search vector quantizers achieve lowest distortion at the expense of complexity and memory requirements at higher bit-rates. So to make the vector quantizers more practical for vectors of larger dimension and higher bit-rates structural constraints are imposed on the design of a vector quantizer or codebook. One way of achieving this is by decomposing the codebook into a Cartesian product of smaller codebooks i.e., \( C = C_1 * C_2 * C_3 \ldots \ldots * C_m \). The advantage with smaller codebooks is that the computational complexity and memory requirements are reduced to a very great extent. This is because the number of bits used for codebook generation is divided among the sets of the decomposed codebook [12, 18]. Examples of product code vector quantization techniques are Split Vector Quantization (SVQ), Multistage Vector Quantization (MSVQ), Split-Multistage Vector Quantization (S-MSVQ), Switched Split Vector Quantization (SSVQ). In this thesis two product code vector quantization techniques are proposed. They are: Switched Multistage Vector Quantization (SWMSVQ) and Multi Switched Split Vector Quantization (MSSVQ) techniques [54, 72].
4.6.2.1 Split Vector Quantization

The main disadvantage of Unconstrained Vector Quantizer is high complexity, memory requirements and the generation of codebook is a difficult task as vectors of full length are used for quantization without any structural constraint. As a result more number of training vectors and bits cannot be used for codebook generation. With these constraints the quantizer cannot produce better quality quantized outputs. So to improve the performance of Unconstrained Vector Quantization technique a well known technique called Split Vector Quantization has been developed. The concept behind Split Vector Quantization is that the vectors of larger dimensions are split into vectors of smaller dimensions and the bits allocated to the quantizer are divided among the splits (parts). Due to splitting, the dimension of a vector gets decreased hence more number of training vectors and bits are used for codebook generation. As a result the performance of quantization is increased, the complexity and memory requirements are reduced. But the main disadvantage with this technique is that, due to splitting the linear and non linear dependencies that exist between the samples of a vector is lost and the shape of the quantizer cells is affected. As a result the spectral distortion increases slightly. This increase in spectral distortion is compensated by increasing the number of training vectors and using more number of bits for codebook generation. The number of splits in this type of quantizer must be limited in number otherwise the vector quantizer will act as a scalar quantizer.
In Split Vector Quantization the training sequence used for codebook generation is split into vectors of smaller dimension. And each split of the training sequence is used to generate separate sub codebooks, there by independent vector quantizers exist and the bits must be allocated to each of them. As a result less number of bits is available at each quantizer, the computational complexity and memory requirements gets reduced as they depend on the number of bits allocated to the quantizer and on the dimension of the vector to be quantized. The block diagram of a three part Split vector quantizer is shown in Fig 4.5.

![Fig 4.5 Block diagram of three part Split Vector Quantizer](image)

From Fig 4.5 it is observed that a vector $S_1$ of dimension ‘$n$’ is quantized by splitting it into sub-vectors $S_{11}, S_{12}, S_{13}$ of smaller dimensions. Each of these sub-vectors are quantized using their respective codebooks. In this work the order of the filter taken is 10
and so the LSF vector contain 10 samples and these 10 samples are split into three parts of 3, 3, 4 samples [54, 73-75].

From results of Split Vector Quantization technique it is proved that the computational complexity and memory requirements gets decreased compared to Unconstrained Vector Quantization technique. So Split Vector Quantization technique is proved to be superior to Unconstrained Vector Quantization technique in terms of the computational complexity and memory requirements.

In an ‘n’ dimensional Split Vector Quantizer of ‘sp’ splits and ‘b’ bits per vector. The vector space R^n is split into ‘sp’ subspaces or splits or parts of lower dimension, then the dimension of a subspace is

\[ n_i \text{ where } n = \sum_{i=1}^{sp} n_i. \]

The number of independent quantizers is equal to the number of splits and the bits used for quantization are divided among the splits. When \( b_i \) is the bits allocated to each split of the vector quantizer then the total bits allocated to the quantizer is

\[ b = \sum_{i=1}^{sp} b_i. \] [54].

The computational complexity of a Split Vector Quantizer is given by equation (4.27)

\[
\text{Complexity}_{SVQ} = \sum_{i=1}^{sp} \left( 4n_i 2^{b_i} - 1 \right)
\] (4.27)
where

\( n_i \) is the dimension of a sub-vector in \( i^{th} \) split

\( b_i \) is the number of bits allocated to the \( i^{th} \) split of a quantizer

\( s_p \) is the number of splits.

The Memory requirements of a Split Vector Quantizer is given by equation (4.28)

\[
\text{Memory}_{SVQ} = \sum_{i=1}^{s_p} n_i^2 b_i
\]  

(4.28)

### 4.6.2.2 Multistage Vector Quantization

Multistage Vector Quantization is a modification of Unconstrained Vector Quantization technique. It is also called as Multistep, Residual or Cascaded Vector Quantization. Multistage Vector Quantization (MSVQ) technique preserves all the features of Unconstrained Vector Quantization technique while decreasing the computational complexity, memory requirements and spectral distortion. Multistage Vector Quantization technique shows significant improvement in the quality of the speech signal by decreasing the spectral distortion, but the computational complexity and memory requirements are high compared to Split Vector Quantization technique. This is because Split Vector Quantization technique deals with the vectors of lower dimensions while Unconstrained and Multistage Vector Quantization techniques deal with vectors of larger dimensions.
Multistage Vector Quantizer is a cascaded connection of several vector quantizers, where the output of one stage is given as an input to the next stage and the bits used for quantization are divided among the stages connected in cascade [12, 14]. As a result less number of bits are available at each stage due to which the computational complexity and memory requirements get reduced. The generation of codebooks at different stages of a three stage MSVQ is shown in Fig 4.6.

![Diagram](image_url)

**Fig 4.6 Generation of codebooks at different stages of Multistage Vector Quantizer**

From Fig 4.6 it is observed that the codebook at the first stage is generated by taking the training sequence as an input. At the second stage the codebook is generated using the quantization errors of the first stage, likewise the codebook at the third stage is generated using
the quantization errors of the second stage. This process is continued for the required number of stages [76-80].

The block diagram of a three stage Multistage Vector Quantizer is shown in Fig 4.7.

---

**Fig 4.7** Block diagram of three stage Multistage Vector Quantizer

The implementation of Multistage Vector Quantizer requires the design of vector quantizers at each stage. In Multistage Vector Quantization the input vector ‘s’ to be quantized is passed through the first stage of the vector quantizer so as to obtain the quantized version of the input vector i.e., ‘\( \hat{s}_1 \)’. The quantization error or residual error \( e_1 = s - \hat{s}_1 \) at the first stage is the difference of the input vector and the quantized version of the input vector. The quantization error at the first stage is given as an input to the second stage vector quantizer to obtain the quantized version of the error vector \( \hat{e}_1 \) at the
first stage. Likewise the quantization error at the second stage \( e_2 = e_1 - \hat{e}_1 \) is given as input to the third stage vector quantizer to obtain the quantized version of the error vector at the second stage i.e., \( \hat{e}_2 \) and this process continues for the required number of stages. Finally the decoder takes the indices \( I_i \) from each quantizer stage and adds the corresponding codewords to obtain the quantized version of the input vector i.e., \( \hat{s} = \hat{s}_1 + \hat{e}_1 + \hat{e}_2 \) [54].

In a Multistage Vector Quantizer each stage acts as an independent vector quantizer and the total bits available for vector quantization are divided among the stages. Then the complexity of a stage equals the complexity of Unconstrained Vector Quantizer and is \( 4n2^{b_j} - 1 \), where \( b_j \) is the number of bits allocated to the \( j^{th} \) stage. Likewise the memory requirements of a stage in Multistage Vector Quantizer is \( n^{2^{b_j}} \).

The computational complexity of a Multistage Vector Quantizer is given by equation (4.29)

\[
\text{Complexity}_{\text{MSVQ}} = \sum_{j=1}^{P} \left( 4n2^{b_j} - 1 \right)
\]

where

\( n \) is the dimension of the vector

\( b_j \) is the number of bits allocated to the \( j^{th} \) stage

\( P \) is the number of stages
The Memory requirements of a Multistage Vector Quantizer is given by equation (4.30)

$$\text{Memory}_{\text{MSVQ}} = \sum_{j=1}^{p} n^2_j b^j$$

(4.30)

### 4.6.2.3 Split-Multistage Vector Quantization

In order to improve the performance of Multistage Vector Quantization and Split Vector Quantization techniques a hybrid product code vector quantization technique, called Split-Multistage Vector Quantization (S-MSVQ) technique is developed. Split-Multistage Vector Quantization technique is a hybrid of Split Vector Quantization and Multistage Vector Quantization techniques. Split-Multistage Vector Quantization technique offer lowest spectral distortion, complexity and memory requirements than the Unconstrained Vector Quantization, Multistage Vector Quantization and Split Vector Quantization techniques [73-80]. The decrease in spectral distortion is due to summing of the quantized errors at each stage.

In Split-Multistage Vector Quantization the dimension of vectors to be quantized is reduced by means of splitting and the bits allocated to the quantizer are divided among the stages and splits of each stage. As a result the dimension of the vectors to be quantized and the bits at each stage & split are reduced which decreases the complexity and memory requirements compared to Unconstrained Vector Quantization, Multistage Vector Quantization and Split Vector Quantization techniques.
The generation of the codebooks at each stage of the Split-Multistage Vector Quantizer is similar to the codebooks generation at each stage of the Multistage Vector Quantizer. But the difference is that each stage of the Split-Multistage Vector Quantizer involves the generation of several sub codebooks. The number of sub codebooks generated at each stage is equal to the number of splits at that stage. In this work, Split-Multistage Vector Quantizer with three parts (splits) and three stages have been developed. The performance of quantization depends on the number of stages and on the number of splits at each stage. As the number of stages increases the quality of the quantized output is increased, but there must be a limit on the number of stages and on the number of splits at each stage as the number of bits at each stage is limited.

The allocation of the bits to a stage is shown in Table 4.1.

<table>
<thead>
<tr>
<th>Number of bits / frame</th>
<th>Number of stages</th>
<th>Bits allocation at each stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>3</td>
<td>8,8,8</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>8,8,7</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>8,7,7</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3,3,3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3,3,2</td>
</tr>
</tbody>
</table>
The allocation of the bits at each split of a stage is shown in Table 4.2.

Table 4.2 Allocation of bits to each split of a stage

<table>
<thead>
<tr>
<th>Number of bits at each stage</th>
<th>Number of splits / stage</th>
<th>Bits allocation to each split</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>2,3,3</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2,2,3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2,2,2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1,1,1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0,1,1</td>
</tr>
</tbody>
</table>

From Tables 4.1 and 4.2 it is observed that the minimum number of bits at each stage with three parts must be at least three. So with three parts (splits) and three stages, in Split-Multistage Vector Quantizer the number of bits at a frame cannot be reduced below 9 bits.

The block diagram of a Split-Multistage Vector Quantizer with three parts and three stages is shown in Fig 4.8. The block diagram is similar to three stage Multistage Vector Quantizer except for the splits at each stage. In Split-Multistage Vector Quantizer each split is treated as a separate vector quantizer and the vectors at each split are quantized independently. The quantization mechanism involved in Split-Multistage Vector Quantizer is similar to the quantization process involved in Multistage Vector Quantizer, except that in Split-
Multistage Vector Quantizer at each stage the sub-vectors are quantized independently.

Split-Multistage Vector Quantizer is a hybrid of Split Vector Quantizer and Multistage Vector Quantizer. The equations for complexity and memory requirements are derived from the complexity and memory requirement equations of a Split Vector Quantizer and Multistage Vector Quantizer. Equations (4.31) and (4.32) below are obtained from the equations (4.29) and (4.30) by including with a summation term having limits from 1 to \( sp \), where ‘sp’ denote the splits at each stage.
The computational complexity of a Split-Multistage Vector Quantizer is given by equation (4.31)

\[
\text{Complexity } \text{S-MSVQ} = \sum_{j=1}^{p} \sum_{i=1}^{sp} \left( 4n_{ji} \cdot 2^{b_{ji}} - 1 \right)
\]  

(4.31)

where

- \( n_{ji} \) is the dimension of a sub-vector in \( j^{th} \) stage \( i^{th} \) split
- \( b_{ji} \) is the number of bits allocated to the \( j^{th} \) stage and \( i^{th} \) split of a quantizer
- \( p \) is the number of stages
- \( sp \) is the number of splits.

The Memory requirements of a Split-Multistage Vector Quantizer is given by equation (4.32)

\[
\text{Memory } \text{S-MSVQ} = \sum_{j=1}^{p} \sum_{i=1}^{sp} n_{ji} \cdot 2^{b_{ji}}
\]  

(4.32)

4.6.2.4 Switched Split Vector Quantization

Switched Split Vector Quantization (SSVQ) is one of the latest vector quantization techniques and is developed to improve the performance of Split Vector Quantization technique. Switched Split Vector Quantization technique is a hybrid of Switch Vector Quantization and Split Vector Quantization techniques and is used to exploit the linear and non-linear dependencies that exist between the splits of a Split Vector Quantizer. In Switched Split Vector Quantizer initially the Switch Vector Quantizer partitions the entire vector space
into voronoi regions and exploits the dependencies that exist across all dimensions of the vector space. Then the Split Vector Quantizer is designed for each of the voronoi regions. As the Split Vector Quantizer is adapted to the local statistics of the voronoi region the sub optimality’s of the Split Vector Quantizer is localized. The function of Switch Vector Quantizer is to perform vector quantization by switching among the codebooks connected in parallel. Switched Split Vector Quantizer can be implemented in two ways they are hard decision scheme and soft decision scheme.

In hard decision scheme each vector to be quantized is quantized in only one of the codebooks connected in parallel. The selection of a codebook for quantization depends on the nearest codeword selected in the initial codebook. An initial codebook is one which is designed for the selection of a switch. The initial codebook is generated from the training vectors used for the generation of the codebooks connected in parallel. The number of codewords or centroids in the initial codebook is equal to the number of switches or number of codebooks connected in parallel.

In soft decision scheme each vector is quantized in all the codebooks connected in parallel and is done by switching from one codebook to the other. In the soft decision scheme there is no need of the initial codebook as the input vector is quantized in all the codebooks connected in parallel.

The first step in the design of SSVQ is to design an initial codebook with centroids equal to the number of switches chosen. Secondly the
training vectors used for the generation of initial codebook are divided into groups based on the nearest neighbor criterion. The training vectors that are nearer to a centroid in the initial codebook must form in one group. The number of groups is equal to the number of centroids in the initial codebook.

Fig 4.9 Codebook training in Switched Split Vector Quantizer

In hard decision scheme the training of a codebook is shown in Fig 4.9. In Fig 4.9 \( \{C_i\}_{i=1}^m \) corresponds to the centroids or codewords in the initial codebook, ‘s’ corresponds to the vector to be quantized and \( i=1, 2, \ldots, m \) corresponds to the number of switching directions.

The block diagram of a Switched Split Vector Quantizer is shown in Fig 4.10. From Fig 4.10 it is observed that a Split Vector Quantizer is connected at each switch of SSVQ. The vector ‘s’ to be quantized is switched to one of the quantizers connected in parallel. The switch
selection depends on the nearest neighbor criterion between the input vector and the centroids of the initial codebook. In Switched Split Vector Quantizer using hard decision scheme the bits used for the design of the quantizer are allocated to the switches and to the Quantizer chosen for vector quantization. In soft decision scheme bits are allocated to the switches and to all the quantizers connected in parallel. The number of bits allocated to the switches is equal to $\log_2 m$ where ‘$m$’ corresponds to the number of switching directions. If ‘$B$’ bits are used to design Switched Split Vector Quantizer, then the number of bits allocated to the vector quantizers is $\left(B - \log_2 m \right)$.

![Block diagram of Switched Split Vector Quantizer](image)

**Fig 4.10** Block diagram of Switched Split Vector Quantizer

While implementing SSVQ using soft decision scheme, the input vector is quantized in all the Vector Quantizers connected in parallel and the quantized vector that gives minimum distortion with the input
vector is taken as the quantized version of the input vector. It is expected that using soft decision scheme there is a slight decrease in the spectral distortion as the input vector is quantized using more than one vector quantizer, computational complexity and memory requirements compared to hard decision scheme as the bits allocated to the vector quantizer are distributed to all the quantizers connected in parallel. So the number of bits at each quantizer is less in soft decision scheme compared to the bits available at a quantizer in hard decision scheme. As the number of available bits at each independent quantizer decreases the complexity and memory requirements decreases. The problem with SSVQ soft decision scheme is that using soft decision scheme the reduction in bit-rate is inversely proportional to the number of switches, number of quantizers connected in parallel and number of splits, so it is not used.

In this work a 2-switch 3-part Switched Split Vector Quantizer using hard decision scheme is implemented and its performance is compared with other product code vector quantization schemes. From results it is observed that Switched Split Vector Quantization using hard decision scheme is having less spectral distortion and computational complexity compared to Split Vector Quantization and Multistage Vector Quantization techniques, but compared to Split-Multistage Vector Quantization technique the spectral distortion and computational complexity are high. The memory requirements of Switched Split Vector Quantization using hard decision scheme is better than Multistage Vector Quantization technique but is more
than Split Vector Quantization technique and Split-Multistage Vector Quantization techniques [54, 81-85].

The equations for complexity and memory requirements of switched split vector quantizer is obtained from the complexity and memory requirement equations of a split vector quantizer by taking into account the complexity and memory requirements of a switch vector quantizer [54].

The computational complexity of Switch Vector Quantizer is given by equation (4.33)

$$\text{Complexity}_{\text{SWITCH}} = 4n^2 \cdot b_m - 1$$  \hspace{1cm} (4.33)

The computational complexity of Switched Split Vector Quantizer using hard decision scheme is given by equation (4.34)

$$\text{Complexity}_{\text{SSVQ HARD}} = \left(4n^2 \cdot b_m - 1\right) + \sum_{i=1}^{sp} \left(4n_{i} \cdot 2^{b_i} - 1\right)$$  \hspace{1cm} (4.34)

where

- $n$ is the dimension of a vector
- $n_{i}$ is the dimension of a sub-vector in $i^{th}$ split
- $b_m$ is the number of bits allocated to the switch vector quantizer
- $b_i$ is the number of bits allocated to the $i^{th}$ split of a quantizer
- $m = 2^m$ is the number of switching directions
- $sp$ is the number of splits.

In computing the complexity and memory requirements of soft decision scheme a summation term with limits from 1 to $Pl$ is used.
This is used to add the complexity and memory requirements of all the vector quantizers connected in parallel.

The computational complexity of Switched Split Vector Quantizer using soft decision scheme is given by equation (4.35)

\[
\text{Complexity}_{\text{SSVQ SOFT}} = \left( 4n^2 b^m - 1 \right) + \sum_{k=1}^{\text{Pl}} \sum_{i=1}^{\text{sp}} \left( 4n_{ki} b^{ki} - 1 \right) \tag{4.35}
\]

where

- \( n_{ki} \) is the dimension of a sub-vector in \( k^{\text{th}} \) codebook and \( i^{\text{th}} \) split
- \( b_{ki} \) is the number of bits allocated to the \( k^{\text{th}} \) codebook \( i^{\text{th}} \) split of a quantizer
- \( \text{Pl} \) is the number of codebooks connected in parallel at each stage

The Memory requirements of Switch Vector Quantizer is given by equation (4.36)

\[
\text{Memory}_{\text{SWITCH}} = n^2 b^m \tag{4.36}
\]

The Memory requirements of Switched Split Vector Quantizer using hard decision scheme is given by equation (4.37)

\[
\text{Memory}_{\text{SSVQ HARD}} = n^2 b^m + 2^m \sum_{i=1}^{\text{sp}} n^i 2 b^i \tag{4.37}
\]

The Memory requirements of Switched Split Vector Quantizer using soft decision scheme is given by equation (4.38)

\[
\text{Memory}_{\text{SSVQ SOFT}} = n^2 b^m + 2^m \sum_{k=1}^{\text{Pl}} \sum_{i=1}^{\text{sp}} n_{ki} 2^{ki} b^{ki} \tag{4.38}
\]
4.7 RESULTS AND DISCUSSION

This section deals with comparing the results of 3-part (split) SVQ, 3-stage MSVQ, 3-part 3-stage S-MSVQ and 2-switch 3-part SSVQ using hard decision scheme in terms of spectral distortion, computational complexity, memory requirements and number of unstable frames for 24 to 20 bits per frame. The spectral distortion is measured in decibels (dB), computational complexity in flops per frame, memory requirements in floats (1 float = 4 bytes, 1 byte = 8 bits). Frames having average spectral distortion greater than 1dB are considered as outlier frames, there must be no outlier frames having a spectral distortion greater than 4 dB and the number of outlier frames between 2 to 4 dB must be less than 2%. For transparent coding the average spectral distortion must be less than 1 dB. The Database used for this experiment is a TIMIT database.

Table 4.3 Spectral distortion, Complexity and Memory requirements for a 3-part Split Vector Quantizer

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>SD(dB)</th>
<th>Percentage of outliers</th>
<th>Complexity</th>
<th>ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2-4 dB</td>
<td>&gt;4dB</td>
<td>(Kflops/frame)</td>
</tr>
<tr>
<td>24(8+8+8)</td>
<td>0.29858287663694</td>
<td>0</td>
<td>0</td>
<td>10.23</td>
</tr>
<tr>
<td>23(7+8+8)</td>
<td>0.32633867154817</td>
<td>0</td>
<td>0</td>
<td>8.701</td>
</tr>
<tr>
<td>22(7+7+8)</td>
<td>0.33288557718973</td>
<td>0</td>
<td>0</td>
<td>7.165</td>
</tr>
<tr>
<td>21(7+7+7)</td>
<td>0.33308663504386</td>
<td>0</td>
<td>0</td>
<td>5.117</td>
</tr>
<tr>
<td>20(6+7+7)</td>
<td>0.56602665050782</td>
<td>0</td>
<td>0</td>
<td>4.342</td>
</tr>
</tbody>
</table>
Table 4.4 Spectral distortion of a 3-part Split Vector Quantizer for different priorities

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>SD(dB)</th>
<th>Bits / frame</th>
<th>SD(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23(7+8+8)</td>
<td>0.3188212811235</td>
<td>23(8+7+8)</td>
<td>0.39490702824897</td>
</tr>
<tr>
<td>22(7+7+8)</td>
<td>0.33288557718973</td>
<td>22(7+8+7)</td>
<td>0.33653566762748</td>
</tr>
<tr>
<td>20(6+7+7)</td>
<td>0.56602665050782</td>
<td>20(7+7+6)</td>
<td>0.56696568993259</td>
</tr>
</tbody>
</table>

The bits allocated to each split of a 3-part SVQ are shown in the brackets of the first column of Table 4.3. The number of splits taken is 3 with 3, 3, 4 dimensions each. The allocation of the bits to a particular split depends on the priority given to that split. Splits with high frequency LSFs are given more priority. The order of priority given to the splits is split 1 < split 2 < split 3. From Table 4.4 it can be observed that when the priority is altered the spectral distortion is slightly increasing.

Table 4.5 Spectral distortion, Complexity and Memory requirements for a 3-stage Multistage Vector Quantizer

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>SD(dB)</th>
<th>Percentage of outliers</th>
<th>Complexity (kflops/frame)</th>
<th>ROM (Floats)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2-4 dB</td>
<td>&gt;4dB</td>
<td></td>
</tr>
<tr>
<td>24(8+8+8)</td>
<td>0.123725346044532</td>
<td>0</td>
<td>0</td>
<td>30.71</td>
</tr>
<tr>
<td>23(8+8+7)</td>
<td>0.1263567038350300</td>
<td>0</td>
<td>0</td>
<td>25.59</td>
</tr>
<tr>
<td>22(8+7+7)</td>
<td>0.1302083212346840</td>
<td>0</td>
<td>0</td>
<td>20.47</td>
</tr>
<tr>
<td>21(7+7+7)</td>
<td>0.1402569528743440</td>
<td>0</td>
<td>0</td>
<td>15.35</td>
</tr>
<tr>
<td>20(7+7+6)</td>
<td>0.1437777179119740</td>
<td>0</td>
<td>0</td>
<td>12.79</td>
</tr>
</tbody>
</table>
Table 4.6 Spectral distortion of a 3-stage Multistage Vector Quantizer for different priorities

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>SD(dB)</th>
<th>Bits / frame</th>
<th>SD(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23(8+8+7)</td>
<td>0.1263</td>
<td>23(7+8+8)</td>
<td>0.1277</td>
</tr>
<tr>
<td>22(8+7+7)</td>
<td>0.1302</td>
<td>22(7+7+8)</td>
<td>0.1386</td>
</tr>
<tr>
<td>20(7+7+6)</td>
<td>0.1438</td>
<td>20(6+7+7)</td>
<td>0.1464</td>
</tr>
</tbody>
</table>

The bits allocated to each stage of a 3-stage MSVQ are shown in the brackets of the first column of Table 4.5. The priority given to the stages during the allocation of bits is stage 1 > stage 2 > stage 3. Stages 2 and 3 are given less priority due to the quantization of errors. It can be observed from Table 4.6 if the priority is not maintained the spectral distortion is increasing.

Table 4.7 Spectral distortion, Complexity and Memory requirements for a 3-part 3-stage Split-Multistage Vector Quantizer

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>SD(dB)</th>
<th>Percentage of outliers</th>
<th>Complexity (kflops/frame)</th>
<th>ROM (Floats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>24(8+8+8)</td>
<td>0.1769</td>
<td>0 0</td>
<td>0.807</td>
<td>204</td>
</tr>
<tr>
<td>23(8+8+7)</td>
<td>0.1782</td>
<td>0 0</td>
<td>0.759</td>
<td>192</td>
</tr>
<tr>
<td>22(8+7+7)</td>
<td>0.1807</td>
<td>0 0</td>
<td>0.711</td>
<td>180</td>
</tr>
<tr>
<td>21(7+7+7)</td>
<td>0.1877</td>
<td>0 0</td>
<td>0.663</td>
<td>168</td>
</tr>
<tr>
<td>20(7+7+6)</td>
<td>0.1881</td>
<td>0 0</td>
<td>0.599</td>
<td>152</td>
</tr>
</tbody>
</table>

In Table 4.7 the numbers in the brackets of 24 (8+8+8) denote the bits allocated to each stage of a 3-part 3-stage S-MSVQ. The allocation
of the bits to a particular stage and split depends on the priority given
to that stage and split. The bits allocated to a stage are divided among
the splits of that stage.

Table 4.8 Spectral distortion, Complexity and Memory requirements
for a 2-switch 3-part Switched Split Vector Quantizer using
hard decision scheme

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>SD(dB)</th>
<th>Percentage of outliers</th>
<th>Complexity (kflops/frame)</th>
<th>ROM (Floats)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2-4 dB</td>
<td>&gt;4dB</td>
<td></td>
</tr>
<tr>
<td>24(1+7+8+8)</td>
<td>0.15197244909408</td>
<td>0</td>
<td>0</td>
<td>8.78</td>
</tr>
<tr>
<td>23(1+7+7+8)</td>
<td>0.17666924153180</td>
<td>0</td>
<td>0</td>
<td>7.244</td>
</tr>
<tr>
<td>22(1+7+7+7)</td>
<td>0.17983792575196</td>
<td>0</td>
<td>0</td>
<td>5.196</td>
</tr>
<tr>
<td>21(1+6+7+7)</td>
<td>0.18376366181869</td>
<td>0</td>
<td>0</td>
<td>4.428</td>
</tr>
<tr>
<td>20(1+6+6+7)</td>
<td>0.18376732214826</td>
<td>0</td>
<td>0</td>
<td>3.66</td>
</tr>
</tbody>
</table>

In Table 4.8 the numbers in the brackets of 24 (1+7+8+8) denote the bits allocated to the switches and splits of a 2-switch 3-part SSVQ using hard decision scheme. The number of bits allocated to a switch will depend on the number of switches chosen. The number of bits allocated to the switches is equal to $\log_2$ number of switches. In this technique 2 switches are taken so the number of bits allocated is 1, which is the first number in bracket. The allocation of the bits to the splits of a quantizer is similar to the allocation of bits for the splits in SVQ. In this technique two quantizers are connected in parallel and the bits are allocated to the quantizer selected.
Table 4.9 gives the number of unstable frames for 3-part SVQ, 3-part 3-stage S-MSVQ and 2-switch 3-part SSVQ using hard decision scheme. The instability is due to independent quantization of the sub-vectors. This instability of frames is high at lower bit-rates. A frame is said to be unstable when the line spectral frequencies of a vector doesn’t obey the ordering property.

<table>
<thead>
<tr>
<th>Bits / frame</th>
<th>No of unstable frames</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVQ</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig 4.11 Spectral Distortion for SVQ, MSVQ, S-MSVQ and SSVQ hard decision scheme
The Fig 4.11 gives a plot of the spectral distortion of 3-part Split Vector Quantizer, 3-stage Multistage Vector Quantizer, 3-part 3-stage Split-Multistage Vector Quantizer, and 2-switch 3-part Switched Split Vector Quantizer using hard decision scheme for 24 to 20 bits per frame. The spectral distortion is measured in dB.

![Complexity for SVQ, MSVQ, S-MSVQ, SSVQ HARD DECISION SCHEME](image)

Fig 4.12 Computational complexity for SVQ, MSVQ, S-MSVQ and SSVQ using hard decision scheme

The Fig 4.12 gives a plot of the computational complexity of SVQ, MSVQ, S-MSVQ, SSVQ using hard decision scheme from 24 to 9 bits / frame. The computational complexity is measured in k flops / frame.
The Fig 4.13 gives a plot of the memory requirements of SVQ, MSVQ, S-MSVQ, SSVQ using hard decision scheme from 24 to 9 bits per frame. The memory requirements is measured in Floats (1 Float = 4 Bytes and 1 Byte = 8 bits).
Fig 4.14 Number of outlier frames having spectral distortion between 2 to 4 dB for SVQ, MSVQ, S-MSVQ and SSVQ hard decision scheme

Fig 4.14 gives a plot of the number of outlier frames lying between 2 and 4 dB. In practice for transparent quantization the number of outlier frames lying between 2 to 4 db must be less than 2%. But from Fig 4.14 it can be observed that the number of outlier frames is zero. So one can say the vector quantization is done in a good manner and transparency in quantization is achieved.
Fig 4.15  Number of outlier frames having spectral distortion greater than 4 dB for SVQ, MSVQ, S-MSVQ and SSVQ using hard decision scheme

Fig 4.15 gives a plot of the number of outlier frames greater than 4 dB. In practice for transparent quantization the number of outlier frames greater than 4 dB must zero. From Fig 4.15 it can be observed that the number of outlier frames greater than 4dB is zero. So transparency in quantization is achieved.

Tables 4.3, 4.5, 4.7 and 4.8 shows the spectral distortion measured in decibels, computational complexity measured in kilo flops per frame and memory requirements measured in floats for a 3-part Split Vector Quantizer, 3-stage Multistage Vector Quantizer, 3-part 3-stage Split-Multistage Vector Quantizer and 2-switch 3-part
Switched Split Vector Quantizer using hard decision scheme from 24 to 20 bits/frame. From Tables 4.3, 4.5, 4.7 and 4.8 and from Figs 4.11 to 4.13 it is concluded that 3-stage Multistage Vector Quantizer is having less spectral distortion compared to 3-part Split Vector Quantizer, 3-stage 3-part Split-Multistage Vector Quantizer and 2-switch 3-part Switched Split Vector Quantizer using hard decision scheme. Also it can be observed that 3-part 3-stage Split-Multistage Vector Quantizer is having less computational complexity and memory requirements compared to 3-part Split Vector Quantizer, 3-stage Multistage Vector Quantizer and 2-switch 3-part Switched Split Vector Quantizer using hard decision scheme. The lowest spectral distortion for MSVQ is due to the addition of the quantized error vectors at each stage to the quantized input vector at the first stage and due to the preservation of correlation that exists between samples of a vector. The decrease in the computational complexity and memory requirements for S-MSVQ is due to the reduced dimension of vectors to be quantized and due to the less availability of bits at each stage and at each split of the vector quantizer.

Table 4.9 gives the number of unstable frames at a particular bit-rate for a 3-part Split Vector Quantizer, 3-part 3-stage Split-Multistage Vector Quantizer and 2-switch 3-part Switched Split Vector Quantizer using hard decision scheme. The instability is due to independent quantization of the sub-vectors (LSF) at each split. In practice the number of unstable frames increases with a decrease in the bit-rate which can be observed for 2-switch 3-part Switched Split
Vector Quantizer using hard decision scheme of Table 4.9. If there are unstable frames after vector quantization the LPC synthesis filter cannot be stable as the ascending order property of the LSF coefficients in an LSF vector gets disturbed. So to prevent this, an ordering constraint is imposed on the LSF parameters of each frame which avoids the problem of unstable frames.

4.8 CONCLUSION

From results it is concluded that 3-stage Multistage Vector Quantizer is having less spectral distortion when compared to Split Vector Quantizer, Split-Multistage Vector Quantizer and Switched Split Vector Quantizer using hard decision. The 3-part 3-stage Split-Multistage Vector Quantizer is providing better tradeoff between bit-rate and computational complexity, memory requirements when compared to Split Vector Quantizer, Multistage Vector Quantizer, Switched Split Vector Quantizer using hard decision scheme. Also from results it is observed that the memory requirements of Switched Split Vector Quantizer using hard decision scheme is more compared to Split Vector Quantizer and this becomes the drawback of Switched Split Vector Quantizer.