3.1 INTRODUCTION:

The main definition of quantization is: it is the division of a large quantity into a discrete number of small parts, often assumed to be integral multiples of a common quantity. Quantization of signals plays a major role in various applications in the areas of signal processing, speech processing and Image processing.

Quantization is required to reduce the number of bits used to represent a sample of speech signal which is known as speech coding. However during this process, some of the parameters like Bit-rate, complexity and memory requirement also get reduced.

Therefore Quantization of the signal results in the loss in the quality of a speech signal, which is undesirable. Hence the researchers have to compromise either with the reduction in bit-rate or with the quality of speech signal.

There are two types of quantizers. They are Non-uniform quantizer and uniform quantizer. Non-uniform quantizer is the one in which the difference between the quantization levels are not uniform, while Uniform quantizer is the one in which the difference between the quantization levels is uniform.

3.2 TYPES OF QUANTIZATIONS:

There are mainly two types of quantizations.

1) Scalar Quantization
2) Vector Quantization

3.2.1 Scalar Quantization:

Quantization [8] is an essential component of speech coding systems. Scalar quantization is the process by which the signal samples are independently quantized
(Sample by Sample basis). The quantization process is based on the probability density function of the signal samples. An N-level scalar quantizer can be treated as a one-dimensional mapping of the input range $R$ onto an index in a mapping table (or codebook) $C$. Thus

$$Q: R \rightarrow C \quad C \subset R$$ (3.1)

The receiver uses this index to reconstruct an approximation to the input level. So, to design the Scalar quantizer, the quantizers are matched to the distribution of the source samples, which may or may not be known in advance. If the distribution is not known in advance, an empirical choice may be made through a Gaussian or Laplacian distribution.

3.2.2 Scalar Quantization of LPC parameters:

Scalar quantization [5, 10, 14] techniques have been reported in the literature for quantizing the LPC parameters. Many of the techniques use either uniform or non-uniform quantizers. If the Non-uniform quantizers are used, then the quantization distortion is less.

Quantizers are designed by using Lloyd algorithm, which will help us to obtain the training data set of the database. However Lloyd algorithm is similar to that of Linde-Buzo-Gray (LBG) algorithm, which is applied to individual LPC parameters. The main aim is to quantize the LPC parameters with less distortion so that they are applicable for speech coding applications.

Scalar quantization of the LPC coefficients [79-81] is usually not done directly because; if there is a small amount of quantization error of the individual LPC coefficients then it can produce relatively large spectral errors and they may result in instability of the all-pole filter $H(z)$. So to avoid spectral errors, the researchers have to concentrate on using more number of bits to quantize.

To overcome all these problems, it is necessary to transform the LPC coefficients to any other representations so that it can result in stability of the all-pole filter after LPC quantization. One of these techniques can be used such as the
Reflection Coefficient (RC) representation, the Arcsine Reflection Coefficient (ASRC) representation, the Log-Area Ratio (LAR) representation and the Line Spectral Frequency (LSF) representation. The scalar quantization of LPC parameters in terms of these representations are described below.

3.3 VECTOR QUANTIZATION:

Vector quantization [10, 11] is a process by which the elements of a vector are quantized as groups called vectors. Vector quantization increases the optimality of a quantizer and there is an increase of computational complexity and memory requirements. Vector quantization is more efficient than scalar quantization in terms of error at a given bit rate. The central component of a Vector Quantizer (VQ) is a codebook C of size N x k, which maps the k-dimensional space $R^k$ onto the reproduction vectors also called code vectors or code words

$$Q : R^k \rightarrow C, \quad C = (Y_1, Y_2, ..., Y_N)^T, \quad Y_i \in R^k$$  \hspace{1cm} (3.2)

Generally, codebook contains finite number of vectors, $Y_i: i = 1, 2, ..., N$. The codebook vectors are selected by following the procedure of clustering or training process which gives rise to training sequence of high bit rate vectors. In this process a vector $x$ is formed by coding the input samples by handling them as blocks of k samples.

The Vector quantizer encoder scans the codebook for an entry $y_i$ and that serves best for approximation of the current input vector $X_t$ at time $t$. In the standard approach of vector quantization, the encoder is designed to minimize the distortion $D$ to get the optimal estimated vector $X_t$;

$$X_t = \min_{Y_i \in C} D(X_t, Y_i)$$  \hspace{1cm} (3.3)

This is therefore referred as nearest neighbor encoding. The particular index $i$ thus derived constitutes the VQ representation of $X$. So to reconstruct the information at the receiver, the index of the code vector is transmitted. Therefore it is very clear
that codebook ‘C’ must be located in both the transmitter and the receiver. The
receiver performs a table lookup and obtains a quantized copy of the input vector. The
code rate or simply the rate of a vector quantizer in bits per component is thus

$$r = \frac{\log_2 N}{K}$$  \hspace{1cm} (3.4)

This gives the number of bits per vector component which is used to represent
the input vector. If the codebook is well designed then it gives an indication of the
accuracy or precision that is achievable with the vector quantizer. Since \(N = 2^K\) the
researchers have to even concentrate on encoding search complexity of the code book,
however codebook storage size increases exponentially with dimension ‘\(K\)’ and rate
‘\(r\)’.

So, training procedures require a affluent combination of source material to
perform vector quantization. The training set may differ with microphones, acoustic
background noise, different languages and even the gender. Most of the recent
methods overcome this drawback of expensive computational requirements by
performing deterministic annealing and genetic optimization methods.

A VQ is nothing but an approximate. The idea is similar to that of “rounding-off” (say to the nearest integer). An example of a 1-dimensional VQ is shown below:

```
00 | 01 | 10 | 11
-4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4
```

Figure: 3.1 One dimensional Vector Quantizer

Here, every number between -4 and -2 is approximated by -3. Every number
between -2 and 0 are approximated by -1. Every number between 0 and 2 are
approximated by +1. Every number greater than 2 is approximated by +3. Note that
the approximate values are uniquely represented by 2 bits. This is a one dimensional,
2-bit VQ. It has a rate of 2 bits/dimension.
A two dimensional VQ is shown in the Figure: 3.2. It can be observed that each pair of numbers available in a particular region is approximated by a red star associated with that particular region.

From the Figure: 3.2 it can be observed that there are 16 regions and 16 red stars and each red star uniquely represented by 4 bits. Thus, this can be represented as a 2-dimensional, 4-bit VQ and its rate is referred as 2 bits/dimension. From the Figure: 3.2 the red stars are called code vectors and the regions defined by the blue borders are called encoding regions. The set of all code vectors is called the codebook [11, 14] and the set of all encoding regions is called the partition of the space.

3.3.1 VQ Design Problem:

The VQ design problem [11] can be stated as follows: A vector source with its statistical properties known, given a distortion measure, and given the number of code vectors, find a codebook (the set of all red stars) and a partition (the set of blue lines) which result in the smallest average distortion.
Assuming that there is a training sequence \((T)\) consisting of ‘\(M\)’ source vectors:

\[
T = (X_1, X_2, \ldots, X_M)
\]

(3.5)

Usually the training sequence is obtained from large database. If the data base is a speech signal, then the training sequence can be obtained by recording several speech conversations.

Let ‘\(m\)’ be the source vectors and assuming it as large and if the value of \(m\) is large then the statistical properties of the source vectors can be considered as the training sequence.

For a \(K\)-dimensional source vector, e.g.

\[
X_m = (X_{(m,1)}, X_{(m,2)}, \ldots, \ldots, X_{(m,K)})
\]

(3.6)

Where \(m = 1, 2, 3, \ldots, M\)

Let \(N\) be the number of code vectors and let

\[
C = \{C_1, C_2, \ldots, C_N\}
\]

(3.7)

Where, \(C_1, C_2, \ldots, C_N\) represents the number of codebooks and each code vector is \(K\)-dimensional, e.g.,

\[
C_n = \{C_{(n,1)}, C_{(n,2)}, \ldots, C_{(n,1)}\}, \quad n = 1, 2, 3, \ldots, N
\]

(3.8)

Let \(S_n\) be the encoding region associated with code vector \(C_n\) and let

\[
p = \{S_1, S_2, \ldots, S_N\}
\]

(3.9)

\(S_1, S_2, \ldots, S_N\), denote the partition of the space. If the source vector \(X_m\) is in the encoding region \(S_N\), then its approximation (denoted by \(Q(X_m)\) is \(C_n\)

\[
Q(X_m) = C_n \quad \text{if} \quad X_m \in S_n
\]

(3.10)
Assuming a squared-error distortion measure, the average distortion can be given as:

$$D_{\text{ave}} = \frac{1}{MK} \sum_{m=1}^{M} \|X_m - Q(X_m)\|$$

(3.11)

Where $\|e\|^2 = e_1^2 + e_2^2 + \cdots + e_k^2$. The design problem can be succinctly stated as follows: Given $T$ and $N$, find $C$ and $P$ such that $D_{\text{ave}}$ is minimized.

3.3.2 Optimality Criteria:

If $C$ and $P$ are the solution to the above minimization problem, then it must satisfy the following two criteria.

**Nearest Neighbor Condition**

$$S_n = \{X: \|X - C_n\|^2 \leq \|X - C_{n'}\|^2 \forall n' = 1, 2, \ldots, N\}$$

(3.12)

As per the condition the encoding region $S_n$ should consist of all vectors which are closer to $C_n$.

**Centroid Condition**

$$C_n = \frac{\sum_{X_m \in S_n} X_m}{\sum_{X_m \in S_n} 1}$$

(3.13)

As per the condition, the code vector $C_n$ should be the average of all training vectors which are in encoding region $S_n$. During the implementation process, it is ensured that at least one training vector will fit into each encoding region so that the denominator value in the above equation will never be zero.
3.4 CODEBOOK GENERATION ALGORITHMS:

3.4.1 LBG Algorithm:

The LBG Vector quantization design algorithm [12] is an iterative algorithm which can solve the above said two optimality criteria. The algorithm requires an initial codebook \( C^{(0)} \). This initial codebook contains High bit rate vectors. These high bit rate vectors are then split into two. The iterative algorithm is run with these two code vectors as the initial codebooks. These two code vectors are split into four and the process is repeated until the desired number of code vectors is obtained. The algorithm is given below.

3.4.2 LBG Algorithm design:

1) Given \( T \) Fixed \( \varepsilon >0 \) to be a small number
2) Let \( N=1 \) and

\[
C_{1}^{*} = \frac{1}{M} \sum_{m=1}^{M} X_{m} \quad (3.14)
\]

Calculate

\[
D_{\text{ave}}^{*} = \frac{1}{MK} \sum_{m=1}^{M} \| X_{m} - C_{1}^{*} \|^{2} \quad (3.15)
\]

3) Splitting For \( I=1,2,\ldots,N \) and set

\[
C_{i}^{(0)} = (1+\varepsilon)C_{i}^{*} \quad (3.16)
\]

\[
C_{N+i}^{(0)} = (1-\varepsilon)C_{i}^{*} \quad (3.17)
\]

Set \( N=2N \)

4) Iteration Let \( D_{\text{ave}}^{0} = D_{\text{ave}}^{*} \). Set the iteration index \( i=0 \)
a) For m=1, 2,…, M, find the minimum value of \( \| X_m - C_n \|^2 \).

The value of n=1, 2,…,N and Let n* will be the index to achieve the minimum Set

Therefore \( Q(X_m) = C_n \)

\[(3.18)\]

b) For n=1,2,…..,N , now the code vector can be updated

\[
C_n^{(i+1)} = \frac{\sum_{Q(X_m)=C_n} X_m}{Q(X_m)} = C_n^i
\]

\[(3.19)\]

c) Set i = i + 1

d) Find the value of

\[
D_{ave} = \frac{1}{MK} \sum_{m=1}^{M} \| X_m - Q(X_m) \|^2
\]

\[(3.20)\]

e) If \( (D_{ave}^{(i-1)} - D_{ave}^{(i)})/D_{ave}^{(i-1)} > \varepsilon \), perform the operation step (a)

f) else set \( D_{ave} = D_{ave}^{(i)} \).

For n=1, 2,……, N find set \( C_n^* \) as the final code vectors

g) To get the desired no. of code vectors repeat steps (c) and (d)

Continue the process by considering the value \( \varepsilon = 0.001 \).

3.4.3 Codebook Generation:

Codebook is generated from a set of training data (LSF) which is obtained by collecting a set of LSF vectors. LBG algorithm [Linde, Buzo and Gray, 1980] [12], is used for clustering L training vectors into a set of M codebook vectors.

The implementation procedure of LBG algorithm is given below:

1. Design a single vector codebook; which is the centroid of the complete set of training vectors and hence there is no iteration.

2. Split each current codebook \( Y_n \) into two parts \( Y_n^+ \) and \( Y_n^- \) according to the rule
\[ Y^+_n = Y_n(1+\varepsilon) \]  
\[ Y^-_n = Y_n(1-\varepsilon) \]

Where the value of \( n \) can vary from 1 to the current size of the codebook. \( \varepsilon \) is a splitting parameter (choose \( \varepsilon=0.01 \)).

3. Each training vector is analyzed to get the codeword which is close to Nearest-Neighbor Search in the current codebook by looking into similarity measurement parameter and assign that vector to the corresponding cell which is allied with the closest codeword.

4. Find the centroid of the training vectors assigned to that cell to update the codeword in each cell.

5. Iteration 1: repeat steps 3 and 4 until the average distance falls below a preset threshold level.

6. Iteration 2: repeat steps 2, 3 and 4 until a codebook size of \( M \) is achieved.

### 3.4.4 Flow Chart for LBG Algorithm:

The LBG algorithm [12] is used to design codebook of \( M \) stages. The algorithm starts with a single large code book and the process starts by splitting the codebook into two groups of vectors as per the splitting technique on the code and this process of splitting will continue till the desired vector codebooks of size \( M \) are obtained.

The flow diagram represented in Figure: 3.3 describe the sequence of steps followed in designing the codebook using LBG algorithm. The purpose of "Cluster vectors" is the nearest-neighbor search procedure which assigns each training vector to a cluster associated with the closest codeword.

As mentioned in the flow chart "Find centroids" is the centroid update procedure and "Compute D (distortion)" is the summation of the distances of all training vectors in the nearest-neighbor search and is helpful to determine whether the Procedure has been converged or not.
Figure: 3.3 Flow diagram of the LBG algorithm
(Adapted from Rabiner and Juang, 1993)
3.5 MULTI STAGE VECTOR QUANTIZATION:

Several techniques can be employed in calculating the codebooks in MSVQ design [17]. The simplest method is to train the codebooks sequentially. The codebook for the first stage is computed in a traditional manner using, e.g., GLA and the training data is quantized with the obtained one-stage vector quantizer. The resulting quantization error vectors are used as the training data for the second stage. This is repeated for all stages, with each new codebook trained using the error between the original and the reconstructed vectors including all the previous stages.

The Multi Stage Vector Quantizer (MSVQ) is a type of product-code vector quantizer [91-96] which reduces the complexity of a vector quantizer, but at the cost of lower performance. In a 2-stage vector quantization [100], the LPC parameter vector (in some suitable representation such as the LSF representation) is quantized by the first-stage vector quantizer and the error vector ‘e’ (which is the difference between the input and output vectors of the first stage) is quantized by the second-stage vector quantizer.

The final quantized version of the LPC vector is obtained by summing the outputs of the two stages. To minimize the complexity of the 2-stage vector quantizer, the bits available for LPC quantization are divided equally between the two stages. Selection of a proper distortion measure is the most important issue in the design and operation of a vector quantizer.

Since the spectral distortion is used here for evaluating LPC quantization performance, ideally it should be used to design the vector quantizer. However, it is very difficult to design a vector quantizer using this distortion measure. Therefore, simpler distance measures (such as the Euclidean and the weighted Euclidean distance measures) between the original and quantized LPC parameter vectors (in some suitable representation such as the LSF representation) are used to design the LPC vector quantizer.
So, to find the best LPC parametric representation for the Euclidean distance measure, the study of the 2-stage vector quantizer with the distance measure in the following three domains: the LSF domain, the arcsine reflection coefficient domain and the log-area ratio domain is done. The 2-stage vector quantizer performs better with the LSF representation than with the other two representations. The Euclidean distance measure used for vector quantization in the preceding section provides equal weights to individual components of the LSF vector, which obviously are not proportional to their spectral sensitivities. Paliwal and Atal [5, 10, 14] have proposed a weighted Euclidean distance measure in the LSF domain which tries to assign weights to individual LSFs according to their spectral sensitivities. The weighted Euclidean distance measure between the test LSF vector \( f \) and the reference LSF vector \( \hat{f} \) is given by

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

(3.23)

Where \( f_i \) and \( \hat{f}_i \) are the \( i \)th LSFs in the test and reference vector, respectively, and \( C_i \) and \( W_i \) are the weights assigned to the \( i \)th LSF. These are given by

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

(3.24)

And

\[
W_i = [P(f_i)^r] \quad \text{For all values of } 1 \leq i \leq 10
\]

Where \( P(f) \) is defined as the LPC power spectrum which is associated with the test vector. Power spectrum has function of frequency \( f \) and \( r \) is an empirical constant which controls the relative weights given to different LSFs and is determined experimentally. A typical value of \( r = 0.15 \) can be considered to get the best performance. Hence it is concluded that a 2-stage vector quantizer can achieve transparent quantization of LPC parameters with the weighted LSF distance measure is about 25 bits/frame and it can provide an average spectral distortion of about 1 dB. The three stage multi-stage vector quantizer block diagram is shown in Figure 3.5. If \( X \) is input vector and \( e1 \) is the error vector and is the input for the 2nd stage and
similarly $e_2$ is the error vector for $2^{nd}$ stage and is the input for the $3^{rd}$ stage. Thus multi stage vector quantization is constructed.

Therefore, the equations that represent the error vector are

$$e_1 = X - \hat{X}_1 \quad (3.25)$$

$$\hat{e}_1 = Q[e_1] \quad (3.26)$$

and

$$\hat{X} = \hat{X}_i + \sum_{i=1}^{m-1} \hat{e}_i \quad (3.27)$$

Where, $m$ is the number of stages used to design MSVQ.

![Figure: 3.4 Codebook generation for different stages of MSVQ.](image)
In a 3-stage MSVQ every stage is an independent individual vector quantizer, therefore, the bits needed to quantize at each stage will be allocated based on the total number of bits. So after having a split in the total bits, each vector quantizer will operate at a lower bit rate. Therefore it requires less memory and computational requirements.

So, comparing with the total size (in number of floating point values) of the codebooks of an $m$-stage, $n$-dimensional MSVQ operating at $b$ bits/vector, with that of an equivalent unconstrained vector quantizer of the same bit rate and dimensionality:

$$\sum_{i=1}^{m} n2^{b_i} \leq n2^b$$  \hspace{1cm} (3.28)

Where, $b_i$ is the number of bits allotted to the $i^{th}$ stage vector quantizer and

$$b = \sum_{i=1}^{m} b_i$$  \hspace{1cm} (3.29)
The sequential training algorithm, however, fails to efficiently exploit the inter-stage dependencies in the codebook optimization. One approach for improving the performance of the sequential training is the iterative algorithm referred to as the joint design of the stage codebooks [8]. The error vectors are calculated as the error between the original vector and the multistage reproduction vector including all the stages except the current stage that the codebook is re-optimized for. The simultaneous joint design algorithm proposed in [9] offers yet another step towards better performance and faster convergence. The computational complexity of MSVQ depends on the search algorithm applied in encoding. An optimal search requires that the distortion measure is calculated for every possible vector combination and thus the complexity is the same as in the full-search VQ. On the other hand, the simplest, but sub-optimal, search algorithm for MSVQ is the sequential search, where at each stage the subsequent stages are disregarded. After the first stage, the residual from the previous stages is quantized at each stage independently of the other stages.

However, better trade-off between performance and complexity can be obtained with more advanced search algorithms. According to [6] the $M$-$L$ search, in which the $M$ best vector combinations are searched at each of the $L$ stages, achieves performance close to that of the optimal search with a relatively low complexity. Compared to sequential training and search, the improvement in SD achieved using the simultaneous joint optimization with the $M$-$L$ search algorithm was approximately 0.1 dB in our simulations for 2-stage VQs with the number of bits ranging from 10 to 16. Approximately half of the improvement resulted from $M$-$L$ search while the other half came from the optimization procedure as can be seen from Figure 3.5.

The Multi Stage quantizers [97] discussed from now on is all designed using the simultaneous joint optimization procedure with $M$-$L$ search. The parameter $M$ is set to 8. Since the aim of MSVQ is to reduce the complexity and the memory requirements, the most common bit allocation schemes, for example the ones used in [2], distribute the bits approximately equally between stages. The largest size of an individual codebook was restricted to 8 bits (256 code vectors) and the
codebook of any particular stage was not allowed to be larger than the codebook of the previous stage.

The following observations were made based on our simulation results. First of all, to minimize the coding distortion the number of stages should be kept as low as possible. It was found useless to add more stages before the largest allowable codebook size is reached at each stage. For example, the 2-stage quantizer with two 8-bit codebooks performed better than the 16-bit quantizers with 3 stages. On the other hand, the 14-bit 2-stage quantizer with bit allocation 7-7 can be performed, however, it will be slightly worse than the 14-bit 3-stage quantizers where an 8-bit codebook was used at the first stage but still better than the other 14-bit 3-stage quantizers where smaller than 8-bit codebooks were used at the first stage.

These trends consistently apply for all simulation results with each quality measure. This suggests that the available bits should be allocated so that the maximum number of bits is allocated to each stage except possibly for the last stage, where the bit allocation representation x-y-z is interpreted so that the first-stage codebook consists of 2x vectors while the second-stage codebook has 2y vectors and the third-stage codebook 2z vectors. Perhaps the most important observation is MSVQ is very robust to changes in bit allocation. Better performance was always obtained with any of i-bit quantizers than with any of j-bit quantizer if i > j.