SPEECH COMPRESSION USING ITU-G.729

IMPLEMETATION AND ANALYSIS

USING MATLAB

SYNDICATE MEMBERS:

NC MUDASSAR HAFEEZ (LEADER)

NC OMAIR KHALID

NC FAHD RASHEED

PC ARSLAN JAVED

PROJECT SUPERVISOR:

MAJ IMTIAZ AHMAD KHOKHAR

Dissertation submitted for partial fulfillment of requirement of MCS/NUST for the award of the B.E. degree in Telecommunication Engineering. Department of Electrical Engineering

Military College of Signals

National University of Sciences & Technology

Rawalpindi

APRIL 2004

TABLE OF CONTENTS

CHAPTER NO 4

PROBLEM AREAS OF G.729

LIST OF TABLES

LIST OF FIGURES

CHAPTER NO 1

CHAPTER 1

1.0 LITERATURE REVIEW

Before starting any project it is essential to have a thorough background knowledge of all the related aspects. Before starting our actual project work we too had a literature review to broaden our knowledge and to know the things from practical point of view. In this chapter we shall describe the topics that we reviewed during this period of phase of our project.

1.1 SIGNALS:

A signal is a form of energy that contains information. All the telecommunication is about formation, transmission and reception of signals. There are two main types of signals; analog signals and digital signals. Analog signals can be represented as current or voltages. These signals vary continuously in time domain over a specified current or voltage range, such as 4-20mA dc or 0 to 5 V dc. Analog signals represent continuously variable entities such as temperatures, pressures, or flow rates.

Digital signals are usually on/off signals. Digital signals are discrete time signals and give the value of samples at specific discrete points in time
As computer based controllers and systems understand only discrete on/off information, the conversion of analog signals to digital representation is necessary. Digital signals propagate more efficiently than analog signals,

largely because digital impulses, which are well-defined and orderly, are easier for electronic circuits to distinguish from noise, which is chaotic. This is the chief advantage of digital modes in communications.

Information in the real world is always in analog form. For our benefit we have to convert it into digital form. A device called transducer is used that converts physical energy into electrical voltage or current for transmission. Analog-to-Digital conversion is an electronic process in which a continuous variable(analog) signal is changed, into a multi-level(digital) signal.

The input to an analog-to-digital converter (ADC) consists of a voltage that varies among a theoretically infinite number of values. Examples are sine waves, the waveforms representing human speech. The output of the ADC, in contrast, has defined levels or states. The number of states is always a power of two -- that is, 2, 4, 8, 16, etc. The simplest digital signals have only two states, and are called binary. All whole numbers can be represented in binary form as strings of ones and zeros. [11]

1.2 ***CONVOLUTIONAL ENCODING:***

Convolutional coding is a major form of channel coding. Convolutional codes operate on serial data, one or a few bits at a time. There are a variety of useful convolutional, and a variety of algorithms for decoding the received coded information sequences to recover the original data. Convolutional encoding with Viterbi decoding is a FEC technique that is particularly suited to a channel in which the transmitted signal is corrupted mainly by additive white gaussian noise (AWGN). AWGN channel is a noise whose voltage distribution over time has characteristics that can be described using a Gaussian, or normal, statistical distribution, i.e. a bell curve. This voltage distribution has zero mean and a standard deviation that is a function of the signal-to-noise ratio (SNR) of the received signal. Let's assume for the moment that the received signal level is fixed. Then if the SNR is high, the standard deviation of the noise is small, and vice-versa. In digital communications, SNR is usually measured in terms of E_b/N_0, which stands for energy per bit divided by the one-sided noise density.

Let's take a moment to look at a couple of examples. Suppose that we have a system where a '1' channel bit is transmitted as a voltage of -1V, and a '0' channel bit is transmitted as a voltage of +1V. This is called bipolar non-return-to-zero (bipolar NRZ) signaling. It is also called binary "antipodal" (which means the signaling states are exact opposites of each other) signaling. The receiver comprises a comparator that decides the received channel bit is a '1' if its voltage is less than 0V, and a '0' if its voltage is greater than or equal to 0V. One would want to sample the output of the comparator in the middle of each data bit interval. We will see the behavior of our example system when E_b/N_0 is high, and then when the E_b/N_0 is lower.

The following figure shows the results of a channel simulation where one million (1×10^6) channel bits are transmitted through an AWGN channel with an E_b/N_0 level of 20 dB (i.e. the signal voltage is ten times the rms noise voltage). In this simulation, a '1' channel bit is transmitted at a level of -1V, and a '0' channel bit is transmitted at a level of +1V. The x axis of this figure corresponds to the received signal voltages, and the y axis represents the number of times each voltage level was received:

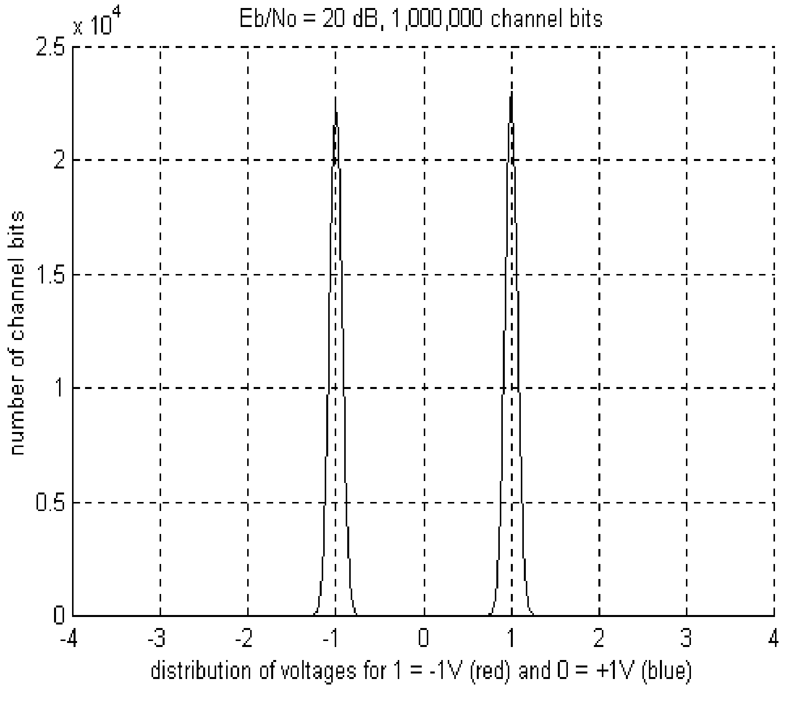


Figure 1.1 Channel Simulation Of Convolutional Encoder

Our simple receiver detects a received channel bit as a '1' if its voltage is less than 0V, and as a '0' if its voltage is greater than or equal to 0V. Such a receiver would have little difficulty correctly receiving a signal as depicted in the figure above. Very few (if any) channel bit reception errors would occur. In this example simulation with the Eb/N0 set at 20 dB, a transmitted '0' was never received as a '1', and a transmitted '1' was never received as a '0'. Convolutional codes are usually described using two parameters: the code rate and the constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length parameter, K, denotes the "length" of the convolutional encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols. Closely related to K is the parameter m, which indicates how many encoder cycles an input bit is retained and used for encoding after it first appears at the input to the convolutional encoder. The "m" parameter can be thought of as the memory length of the encoder. Viterbi decoding is one of two types of decoding algorithms used with convolutional encoding-the other type is sequential decoding. Sequential decoding has the advantage that it can perform very well with long-constraint-length convolutional codes, but it has a variable decoding time. [12]

1.3 VITERBI DECODING ALGORITHM:

The Viterbi algorithm essentially performs maximum likelihood decoding, however, it reduces the computational load by taking advantage of the special structure in the code trellis. The single most important concept to aid in understanding the Viterbi algorithm is the trellis diagram. The figure below shows the trellis diagram for a rate 1/2 K = 3 convolutional encoder, for a 15-bit message:

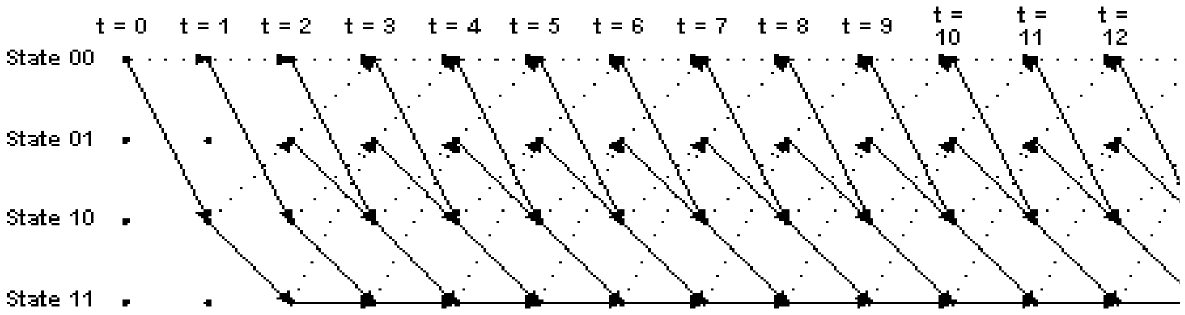


Figure 1.2 Trellis Diagram

The four possible states of the encoder are depicted as four rows of horizontal dots. There is one column of four dots for the initial state of the encoder and one for each time instant during the message. For a 15-bit message with two encoder memory flushing bits, there are 17 time instants in addition to t = 0, which represents the initial condition of the encoder. The solid lines connecting dots in the diagram represent state transitions when the input bit is a one. The dotted lines represent state transitions when the input bit is a zero. Notice the correspondence between the arrows in the trellis diagram and the state transition table discussed above. Also notice that since the initial condition of the encoder is State 002, and the two memory flushing bits are zeroes, the arrows start out at State 002 and end up at the same state.

The following diagram shows the states of the trellis that are actually reached during the encoding of our example 15-bit message

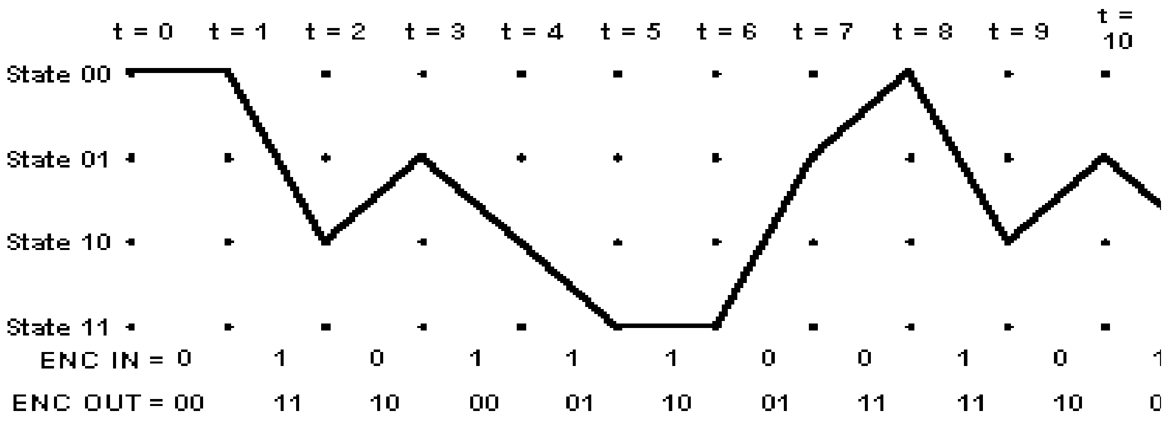


Figure 1.3 Trellis States

The advantage of Viterbi decoding compared with brute-force decoding is that the complexity of a Viterbi decoder is not a function of the number of the symbolism the codeword sequence. The algorithm involves calculating a measure of similarity, or distance, between the received signal ,at time t_1 and all the trellis paths entering each state at time t_1.The Viterbi algorithm removes from consideration those trellis paths that could not possibly be candidates for the maximum likelihood choice. When two paths enter the same state, the one having the best metric is shown; this path is called the surviving path. This section of surviving paths is performed for all the states. The decoder moves in this way to advance deeper into the trellis, making decisions by eliminating the least likely paths. The early rejection of the unlikely paths reduces the decoding complexity. The Viterbi algorithm is, in fact, maximum likelihood. The goal of selecting the optimum path can be expressed, equivalently, as choosing the codeword with the maximum likelihood metric, or as choosing the codeword with the minimum distance metric.[17]

1.4 ***EQUALIZATION:***

Equalization is the increase or decrease of signal strength for a portion of (a band of) audio frequencies. The audio we record (the sound made by instruments or voices) is *complex.* By this we mean that it is composed of energy at different audio frequencies. If we take a bass control (a simple equalizer) and turn the knob clockwise, we will get an increase in strength of the signal (or the signal component) that has lower frequencies (usually any component below about 500 Hz). Thus equalization affects the tone because it changes the level relationship of the **fundamental** and **harmonic** frequency. The tuned frequency and almost always the lowest frequency in the sounding of a pitch of a musical instrument is called as fundamental frequency. Whole number of multiples of the fundamental frequency that determines the tone rather than the pitch recognition of the instrument's sound is called harmonics.

1.4.1 ***Active Equalization*:**

This means that the designated frequencies are actually run back through an amplifier to boost just the desired frequencies. This however is extremely expensive because separate amps must be used for each range of frequencies to be altered.

1.4.2 ***Passive Equalization:***

This is accomplished in one of two ways. The first is similar to the old tone knobs found on pre-digital car radios. Through the use of resistance, the bass or the treble signals are attenuated making the other seem louder. This is done by taking the signal, to be altered and dividing it into two separate components. Using a potentiometer, a delay is created in one of the signals to move it out of phase with the other. When the two signals are recombined, the degree of phase difference between them results in a proportionate attenuation of the signal.[13]

1.5 ***MEDIUM EFFECTS:***

For transmission of any type of data we need a medium. The medium may be wired or wireless, analog or digital. Each one has its own effects on the data being transferred through it. In this section we shall describe briefly the effects of different types of media on the signal.

1.5.1 ***WIRED MEDIA:***

When a data acquisition system is transmitting signals over wires, some signal degradation is unavoidable and will occur due to noise and resistance. Noise and signal degradation due to resistance are two basic problems in signal transmission. Noise is defined as any unwanted electrical or magnetic phenomena that corrupt a message signal. Noise can be categorized into two broad categories based on the source- internal noise and external noise. While internal noise is generated by components associated with the signal itself, external noise results when natural or man-made electrical or magnetic phenomena influence the signal as it is being transmitted. Noise limits the ability to correctly identify the sent message and therefore limits information transfer. Some of the sources of internal and external noise include:

Electromagnetic interference (EMI);
Radio-frequency interference (RFI);

1.5.2 WIRELESS MEDIA:

In a wireless medium, the electrical interference between the signals can causes a signal degradation. In radio transmission, the presence of the reflecting objects creates a constantly changing environment that dissipates the signal energy in amplitude, phase and time. These random phase and amplitudes of the different multi path components cause fluctuation in the signal strength, thereby including small scale fading, signal distortion or both.

1.5.3 DIGITAL MEDIA:

In a digital media due to accumulation of noise and signal distortion along the transmission path, the regenerative repeaters are used. If the bits are lost or corrupted in the process of transmission, usually redundant bits are then transmitted.[14]

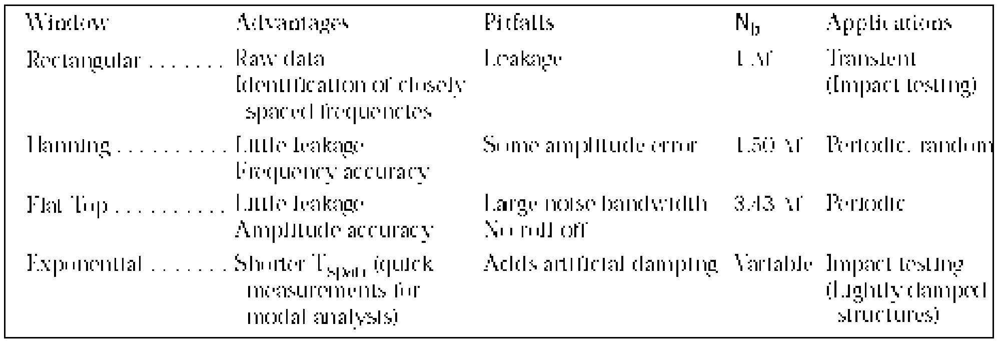
1.6 BANDWIDTH UTILIZATION:

As there is rapid change in field of telecommunication and the world is becoming a global village, more and more communication instruments are being introduced everyday in the market. The number of communication equipment might be increased but in order to utilize them, the resources such as bandwidth are used. Bandwidth can never be increased therefore it is considered a valuable resource. The process of communication takes place through engaging the bandwidth, which has to be utilized in the best way possible. With special reference to telephony, there are two different types of compression techniques being used today; one of them is waveform coding while the other is vocoding. In waveform coding the various techniques are DPCM, ADPCM, DM etc. The other technique is vocoding in which we extract certain parameters from input speech and send it to the decoder and try to reconstruct our original speech from them. Examples of vocoders are ITU-T standards G.723, G.726, G.728, G.729 etc. Yet another way of bandwidth utilization is multiplexing technique, which has the following components, FDMA, TDMA, CDMA.

a) In FDMA certain slots in the frequency spectrum are utilized for the purpose of communication and that slot is engaged by one user as long as the communication persists.
b) In TDMA, certain slots in the spectrum are utilized for a certain amount of time limit.
c) In CDMA, a code is given which helps to engage the frequency spectrum for the whole time limit. So the user is recognized by the code only.[15]

1.7 <u>WINDOWING:</u>

Fourier analysis is commonly used to estimate the spectral content of a measured signal. When choosing the appropriate window, one needs to be aware of its advantages and pitfalls in order to fit the measurement situation. The following deals with some practical considerations on the effects of windowing.



Window	Advantages	Pitfalls	N_b	Applications
Rectangular	Raw data Identification of closely spaced frequencies	Leakage	1 Δf	Transient (Impact testing)
Hanning	Little leakage Frequency accuracy	Some amplitude error	1.50 Δf	Periodic, random
Flat Top	Little leakage Amplitude accuracy	Large noise bandwidth No roll off	3.43 Δf	Periodic
Exponential	Shorter T_{span} (quick measurements for modal analysis)	Adds artificial damping	Variable	Impact testing (Lightly damped structures)

Table 1.1 Comparison of window for FFT

Let us examine the situation with a sine wave of frequency f_0. In theory the corresponding spectrum is a peak at f_0. When a non integer number of periods is acquired, this results in signal leakage, characterized by the smearing of the spectrum. Figure 1 illustrates this phenomenon by comparing three cases, where f is the sine wave frequency and $.f$ the frequency resolution. Case 1(a) allows us to determine the ideal situation where an integer number of periods (200) is set for the signal generator. The corresponding frequency is f_0 = 508.626 Hz. In practice this case is not likely to occur, because the frequency that is being measured rarely falls on a frequency line. On the other hand, case 1(b) represents a typical situation where the leakage is clearly visible. Here f has been slightly decreased, which results in a non-integer number of periods within T_{span}. The maximum leakage is obtained in case 1(c). The answer lies in the spectrum of the window as shown in Figure 2(a). The FFT emulates a bank of parallel bandpass filters with the center frequencies
exactly centered on integer multiples of $.f$. The width and shape of each filter is identical and are given by the spectrum of the observation window shown in Figure2. Note that the filter shape is characterized by multiple lobes separated by zero values at multiples of $.f$ and that all filters in the bank overlap. When f of an applied sine corresponds exactly to a filter center-frequency [case1(a)], only that filter will respond because f corresponds to an amplitude notch of all other filters in the bank. Conversely, if f is not exactly on a frequency line [case1(b)], the energy at f is smeared over adjacent frequencies because the secondary lobes of all other filters overlap f with nonzero gain and these filters

respond in proportion to this gain. This perverse effect is maximum when $f = f_0$ $.f/2$ [case 1(c)], since the frequency f coincides with the peak of each side lobe. If side lobes could be reduced in amplitude, this error would decrease as well. This is why people have used a number of windows to weight the truncated signal such that the starting value and the ending value are zero. This produces a signal that appears periodic in T_{span}, meeting the basic assumption of the Fourier Series. Weighting avoids the sharp discontinuities induced by the Rectangular window and yields reduced amplitude side lobes as desired. Figure2(b) shows the spectrum of the well known Hanning2 window. Observe that the amplitude of the first side-lobe is reduced from –13.2 dB to –32.2 dB. More importantly, notice that the amplitudes of subsequent side-lobes fall off at 60 dB/decade as opposed to 20 dB/decade for the Rectangular window.[16]

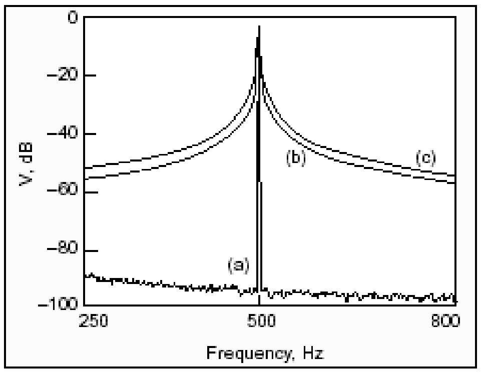


FIGURE 1.4

Rectangular window: (a) $f = f_0$; (b) $f = f_0 - .f/8$;

(c) $f = f_0 - .f/2$.

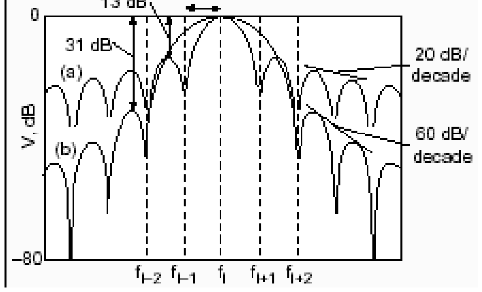


FIGURE 1.5

Comparison of spectra for the Rectangular window (a) and the Hanning window(b)

CHAPTER NO 2

CHAPTER NO.2

2.0 COMPRESSION TECHNIQUES :

Before starting our original project we studied several other compression techniques to get a better picture of speech compression. Basically coders used in the compression techniques are broadly divided in to two categories. They are

a. Vocoders
b. Waveform coders

Vocoders extract certain parameters from input speech and try to reconstruct same speech at the decoder with the help of those parameters. They don't try to reconstruct the original waveform which is intended to be done in case of waveform coders. They actually model the human speech system that comprises of vocal tract and its input from lungs.

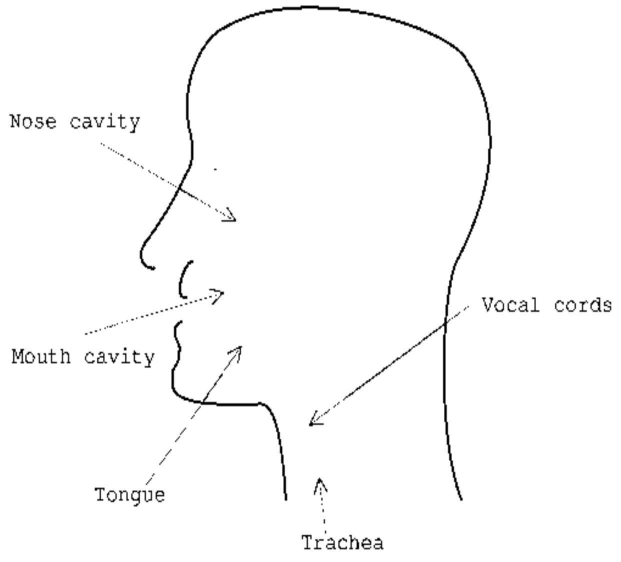


Figure 2.1 Human Voice Speech System

this family of coder includes G.723,G.729,etc which will be discussed in the later part of this chapter.

Waveform coders don't extract any parameters from input speech rather they maintain the original waveform. This family of coders includes ADPCM, Delta Modulation etc.

Vocoders have low data rates and toll speech quality whereas waveform coders have high data rates and excellent speech quality. Toll quality means good performance while being used in telephony [18] .

Besides these two major classifications of compression techniques there is yet another technique known as MPEG. We shall first describe this technique and then go to the vocoders.

2.1 MPEG:

MPEG is the "Moving Picture Experts Group", working under the joint direction of the International Standards Organization (ISO) and the International Electro-Technical Commission (IEC). This group works on standards for the coding of moving pictures and associated audio [2].

The MPEG audio standard is a high-complexity, high-compression, and high audio quality algorithm. Its advantages include high immunity to noise (interference that destroys the integrity of signals), stability, reproducibility, and the efficient implementation of audio processing functions through a computer.

2.1.1 MPEG- 1:

MPEG-1 describes the compression of audio signals using high performance perceptual coding schemes. It specifies a family of three audio coding schemes, simply called Layer-1,-2,-3, with increasing encoder complexity and performance . The three codecs are backward compatible i.e. a Layer-N decoder is able to decode bitstream data encoded in Layer-N and all Layers below N [1] .

2.1.2 COMPRESSION RATES OF DIFFERENT LAYERS:

The previous Layer 1 and Layer 2 standards would provide 4:1 or 6:1-8:1 compression ratios respectively (resulting in a bitstream of 384-192 kbps). MP3 allowed for compression down to 11:1-12:1 of the original CD bitrate, but with preserved quality. In return, it consumed more CPU time than the previous layers [4].

2.1.3 FEATURES OF MPEG1:

All Layers of MPEG1 use the same basic structure. The coding scheme can be described as "perceptual subband / transform coding". The encoder analyzes the spectral components of the audio signal by calculating a filterbank or transform and applies a psychoacoustic model to estimate the just noticeable noise-level. In its quantization and coding stage, the encoder tries to allocate the available number of data bits in a way to meet both the bitrate and masking requirements. The decoder is much less complex. Its only task is to synthesize an audio signal out of the coded spectral components.

All Layers use the same analysis filterbank (with 32 subbands). Layer-3 adds a MDCT (modified discrete cosine transform) to increase the frequency resolution. All layers use the same "header information" in their bitstream, to support the hierarchical structure of the stan dard. All layers use a bitstream structure that contains parts that are more sensitive to bit errors (header, bit allocation, scalefactors, side information) and parts that are less sensitive (data of spectral components). All Layers may use 32, 44.1 or 48 kHz sampling frequency. From Layer-1 to Layer-3 complexity increases (mainly true for the encoder),overall codec delay increases, and performance increases (sound quality per bitrate).

2.1.4 DELAYS OF MPEG1 LAYERS:

The standard gives some figures of the theoretical minimum delays:

Layer-1: 19 ms (<50 ms)

Layer-2: 35 ms (100 ms)

Layer-3: 59 ms (150 ms)

We can see that the practical values in the brackets are significantly above that. As they depend on the practical implementation and many other factors so exact figures are hard to give. So the figures in brackets are just roughly estimated values [1].

2.2__MP3:__

MP3 stands for MPEG-1 Layer 3. Layers represent a family of coding algorithms. Layer 1 has the lowest complexity compared to the other layers. Layer 2 requires a more complex encoder and decoder and is directed more towards different applications. Thus, Layer 2 is more able than Layer 1 to remove the redundancy in the signal that takes up space. Layer 3 is again more complex and further reduces redundancy and relevance from the data. . The layers are backwards compatible, which means that the decoder of Layer 3 audio should also be able to decode Layers 1 and 2. Though MP3 compression is considered lossy because some data cannot be recovered after compression, the MPEG algorithm can achieve transparent, or perceptually lossless, compression. After testing, it was concluded that expert listeners could not distinguish between coded and original audio clips even with a six to one compression ratio [4].

The phenomenon of auditory masking is of great importance in the understanding of mp3 encoder so its explanation is given below:

__2.2.1 AUDITORY MASKING:__

In a layman language masking can be explained as in the presence of a strong person the effect of the weaker person is diminished. Same is the case with frequencies. Basically the concept of auditory masking is that in the vicinity of strong tonal signal the effect of weak tonal signal is diminished , lets explain it with the help of an example. Suppose you have a strong tone with a frequency of 1000Hz. You also have a tone nearby of say 1100Hz. This second tone is 18 dB lower. You are not going to hear this second tone. It is completely masked by the first 1000Hz tone. As a matter of fact, any relatively weak sounds near a strong sound is masked. If you introduce another tone at 2000Hz also 18 dB below the first 1000Hz tone, you will hear this.You will have to turn down the 2000Hz tone to something like 45 dB below the 1000Hz tone before it will be masked by the first tone. So the further you get from a sound the less masking effect it has [5].

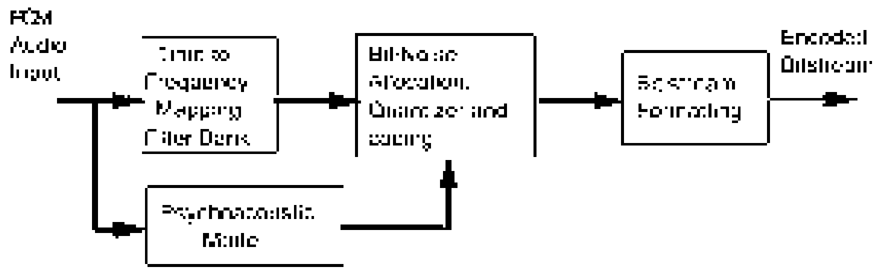

PCM Audio Input
Time to Frequency Mapping Filter Bank
Bit/Noise Allocation, Quantizer and coding
Bitstream Formatting
Encoded Bitstream
Psychoacoustic Model

FIGURE 2.2 MP3 Encoder

2.2.2 ***ENCODER OF MP3 :***

a. First, the input audio stream passes through a filter bank that divides the soundinto 32 equal width sub bands of frequency. In it we use MDCT (modified discrete cosine tranform) for the frequency domain analysis of the input signal. MDCT is designed to be performed on consecutive blocks of a larger dataset, where subsequent blocks are 50% overlapped. This overlapping, in addition to the energy-compaction qualities of the DCT, makes the MDCT especially attractive for signal compression applications.
b. After dividing the input signal in 32 sub-bands we try to figure out the frequencies in each sub-band which carry the effect of all other frequencies of that sub-band or the frequencies which mask other frequencies. We are actually interested in transmitting these frequencies which carry the information of almost the entire sub-band.

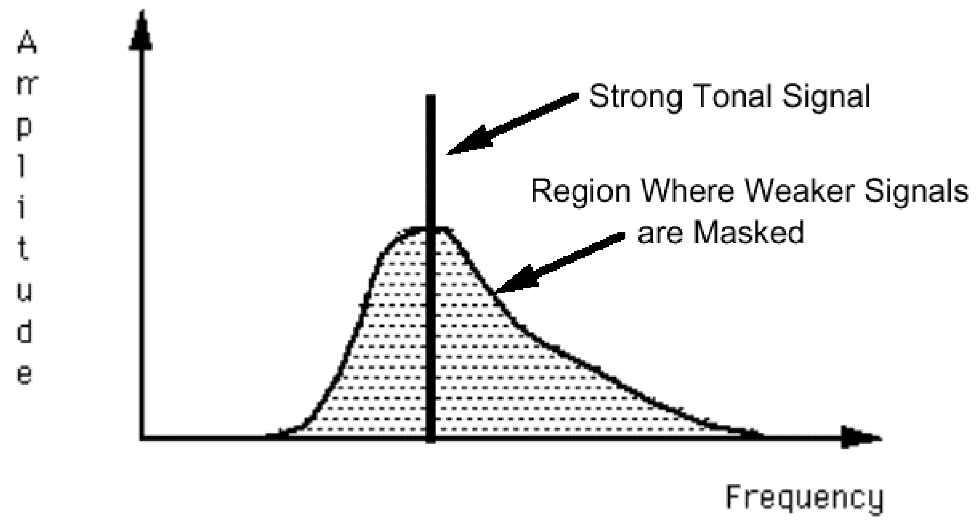


Figure 2.3 Principle of MPEG

a. The input to filter bank also passes through psycho-acoustic model that utilizes the concept of auditory masking to determine what can or cannot be heard in each sub band. The psychoacoustic model works on the concept of auditory masking and its effect on compression. The output of this block is SMR (signal to mask ratio).

b. The out puts of both filter bank and psycoacoustic model enters in bit allocation block.The bit allocation block minimizes the audibility of noise. Through an iterative algorithm, the bit allocation uses information from the psychoacoustic model to determine the number of code bits to be allocated to each sub band. This process can be described using the following formula:

   a. ***MNRdB = SNRdB - SMRdB***

   b. where:

      a. $\boldsymbol{MNR_{dB}}$ is the mask-to-noise ratio

b. ***SNR_{dB}*** is the signal-to-noise ratio, given with the MPEG audio standard in a table.

c. ***SMR_{dB}*** is the signal-to-mask ratio, derived from the psychoacoustic model

c. Then the sub bands are placed in order of lowest to highest mask-to-noise ratio, and the lowest sub band is allocated the smallest number of code bits and this process continues until no more code bits can be allocated [3].
d. The quantization step of the signal is done by looking at the values assigned to each sub-band, which means that it will quantizise high values more roughly than low ones. Quantization also causes an error, a form of noise, which has to be minimized. By giving each subband a trait called 'scalefactor' this can be done. The scalefactor is a factor which shows how much presence the sub-band will have in the final signal. The process of applying scalefactors and error reduction is done through two iteration loops in the algorithm.After the quantization, the coding step comes in where the information is tranformed into useful code. This is done through Huffman coding, an entropy coding process which is lossless.

2.2.3 <u>DECODER:</u>

When the encoded bit stream reaches at the decoder first step is the frame unpacking.Decoding is much simpler, because it only involves adding the 32 subbands back together. It is important to note that this compression algorithm is not lossless. However, it is designed to be “transparently lossy,” meaning that the information that is discarded is imperceptible to humans. A large number of tests have been performed in an attempt to decide that whether it is imperceptible to humans and almost everyone agrees that there is no perceivable difference in sound quality between an original uncompressed CD audio signal and a compressed MPEG Layer-3 audio signal.

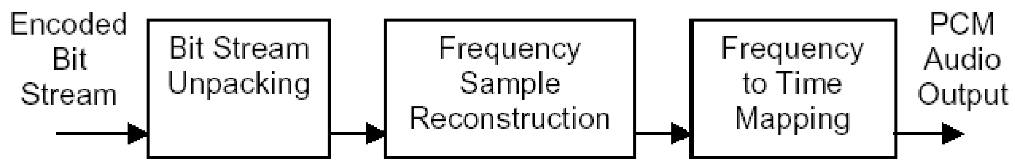


FIGURE 2.4 MP3 Decoder

2.3 <u>G.723.1:</u>

2.3.1 <u>FEATURES OF G.723.1:</u>

The coder is based on the principles of linear prediction analysis-by synthesis coding and attempts to minimise a perceptually weighted error signal.

a. It operates on a digital input signal specified with 16 bit (PCM linear) at 8 kHz sampling rate. The decoder converts the received signal after decoding back to an analogue signal.
b. The coder can be used for compressing the speech or other audio signals component of multimedia services at a very low bit rate.
c. Two bit rates are associated with 5.3 and 6.3 kbit/s, as part of the overall H.234 family of standards. The higher bit rate offers greater quality, whereby the lower offers good quality and provides system designers with additional flexibility.
d. It is possible to switch between the two rates at any frame boundary. Variable rate operation using discontinuous transmission and noise fill during non-speech intervals is possible.
e. The excitation signal for the high rate coder is Multi-Pulse Maximum Likelihood Quantization (MP-MLQ) and for the low rate coder is Algebraic-Code-Excited Linear Prediction (ACELP).
f. The coder provides high quality at low rates using limited amount of complexity.

g. The frame size is 30 ms and additional look ahead of 7.5 ms is required.Therefore the total algorithmic delay of 37.5 ms results.
h. Additional delay is caused due to processing, transmission and buffering of the multiplexing protocol.
i. Both the 5.3k bps and 6.4k bps rates are mandatory for the encoder and decoder. A G.723 frame stream may switch between the two rates at any 30 ms frame boundary.
j. Relative to the G.729/G.729A coders the G.723 coder passes DTMF tones through with less distortion.

The output of this filter is x[n], [illegible]

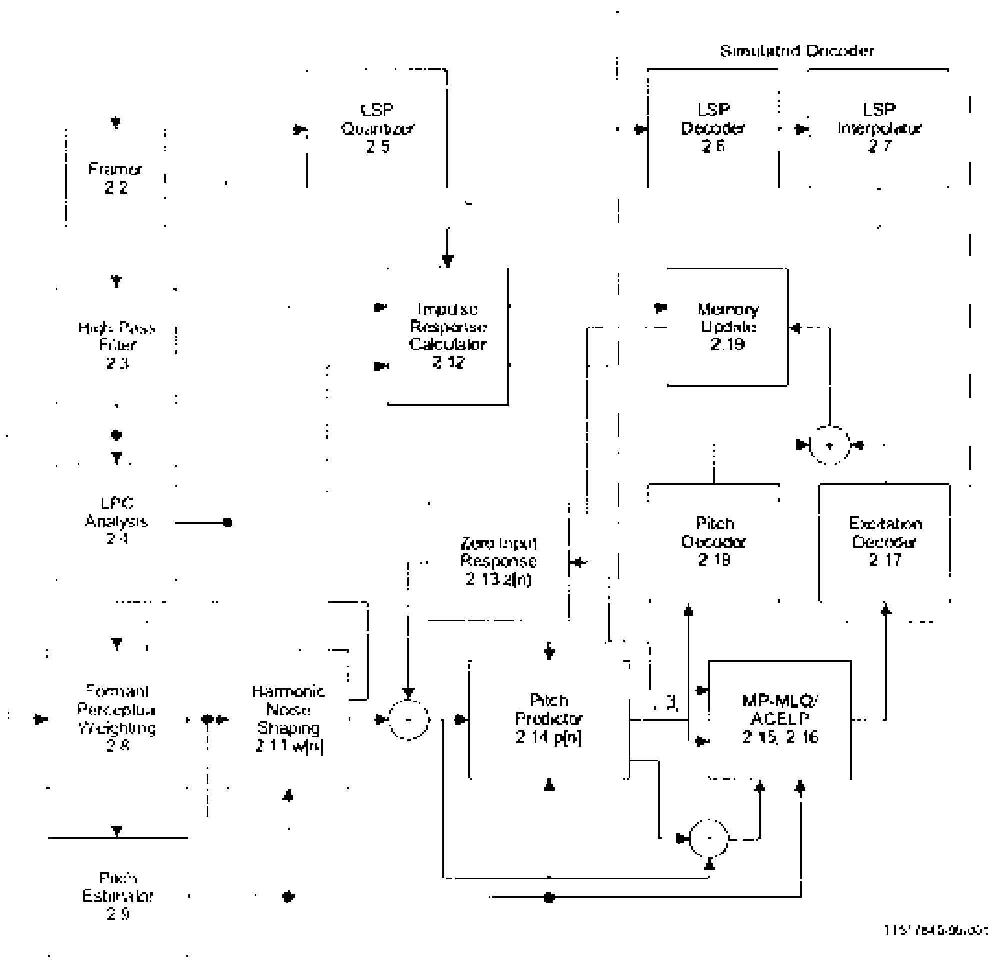


Figure 2.5 G.723.1 ENCODER:

2.3.2 OPERATION OF ENCODER:

The encoder operates on frames of 240 samples (30ms) each. Each frame is first highpass filtered to remove DC component and then divided into 4 subframes of 60 samples (7.5ms) each. For every subframe, a 10th order Linear Prediction Coder (LPC) filter is computed. The LPC filter for the last subframe is quantized using a Predictive Split Vector Quantizer (PSVQ). The unquantized LPC coefficients are used to construct the short-term perceptual weighting filter. The weighting filter is used to filter the entire frame and to obtain the perceptually weighted speech signal. Every two subframes the open loop pitch period (pitch estimation in the range from 18 to 142 samples) is computed using the weighted speech signal. From this point speech is processed at subframe basis. The estimated pitch period is used to construct the harmonic noise shaping filter. The combination of LPC synthesis filter, perceptual weighting filter, and the harmonic noise shaping filter is used to create an impulse response. Using this impulse response and the pitch period estimation a closed loop pitch predictor (5th order) is computed. The pitch period is computed as a small differential value around the open loop pitch estimate. Both the pitch period and the differential value are transmitted to the decoder. Finally the non-periodic component of the excitation is approximated. For high rate, Multi-Pulse Maximum Likelihood Quantization (MP-MLQ) excitation is used, and for lower bit rate, an algebraic-code-excitation (ACELP) is used.

The major differences between the two rates are in the pulse positions and amplitude coding. Also at the lower rate 170 codebook entries are always used for the gain vector of the long term predictor.

Parameters coded	**Subframe 0**	**Subframe 1**	**Subframe 2**	**Subframe 3**	**Total**
LPC indices					24
Adaptive codebook lags	7	2	7	2	18
All the gains combined	12	12	12	12	48
Pulse positions	20	18	20	18	73
Pulse signs	6	5	6	5	22
Grid index	1	1	1	1	4
Total					189

TABLE 2.1 ***BIT ALLOCATION OF 6.4 Kb/s CODING TABLE:***

Parameters coded	**Subframe 0**	**Subframe 1**	**Subframe 2**	**Subframe 3**	**Total**
LPC indices					24
Adaptive codebook lags	7	2	7	2	18
All the gains combined	12	12	12	12	48
[1]Pulse positions	12	12	12	12	48
Pulse signs	4	4	4	4	16
Grid index	1	1	1	1	4
Total					158

TABLE 2.2 ***BIT ALLOCATION OF 5.3Kb/s CODING ALGORITHM:***

[1]

2.3.3 G.723.1 DECODER:

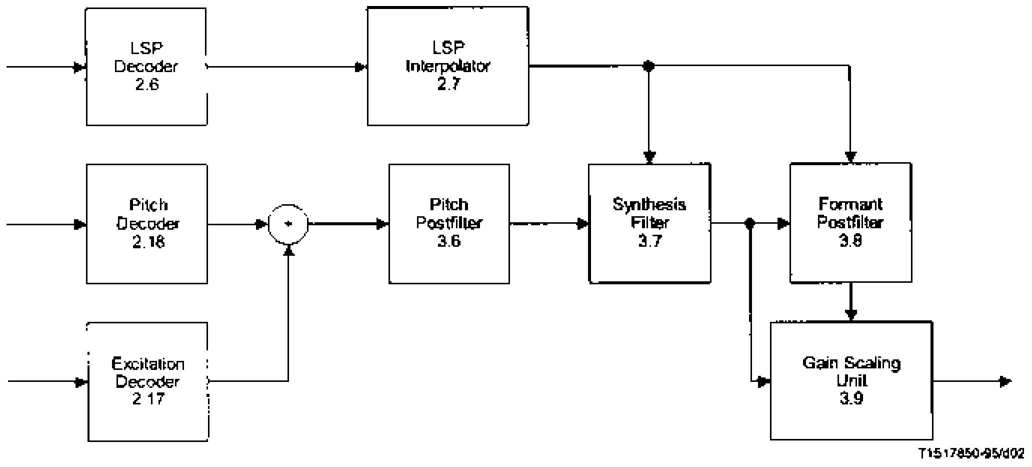


FIGURE 2.6 G.723.1 Decoder

The decoder operates on a frame by frame basis. First the LPC coefficients are reconstructed out of the decoded LSP coefficients and therefore the synthesis filter can be rebuilt. Second for every subframe both the adaptive and the fixed codebook excitation are decoded and fed to the synthesis filter. The adaptive postfilter consists of a formant and a forward-backward pitch postfilter to improve the quality of synthesized speech. The quality is improvement is obtained by increasing the SNR at multiples of the pitch period. The excitation signal is input to the pitch postfilter, which in turn is input to the synthesis filter. Afterwards the output of the synthesis filter is input to the formant postfilter (also requiring the LPC coefficients). A gain scaling unit maintains the energy at the input level of the formant postfilter.

2.3.4 ANALYSIS OF G.723.1:

PARAMETER	**OBJECTIVES**	**PERFORMANCE**
Speech quality in error free conditions(includes both clean speech and background noise)	Not worse than 32 kb/s G.726	Equivalent to 32 kb/s G.726 in clean speech but worse in case of background noise
Speech performance with bit errors for an input signal nominal level of -26 db with respect to the overload point	=< 0.5 MOS degradation	Met objective
Speech quality dependency on the input signal level between –36 db and –16 db with respect to the overload point	Not worse than 32 kb/s G.726	Met objective
Quality dependency on speakers	Not worse than 32 kb/s G.726	Met objective
Capability to transmit music	No annoying effects generated	No formal tests, sounds a little scratchy
Capability to transmit signaling info. Tones	Capability to transmit DTMF	Sensitive to DTMF imperfections
Tandeming capability for the speech coder	2 asynchronous codings with a total distortion of <= 4 asynchronous 32 kb/s G.726	Better than objective

TABLE 2.3 Analysis of G.723.1

2.4 <u>G.729:</u>

2.4.1 <u>FEATURES:</u>

a. Compresses 8 kHz CODEC or linear audio data to 8 kbps.
b. Operates on 10ms frames with short algorithm delays.
c. Short-term synthesis filter is based on a 10th order Linear Prediction (LP) filter.
d. Long-term, or pitch synthesis, filter is implemented using the adaptive-code book approach [6].
e. There is a look ahead 5 ms in g.729 forming a total algorithmic delay of 15 ms.

2.4.2 <u>COMPARISON BETWEEN G.723.1 AND G.729:</u>

Comparing the overall performance of G.723.1 at 6.3kbit/s and G.729 there is no great deal of difference. The real differences are in bit rate and delay. G.723.1 has several advantages with respect to bit rate either 6.4 or 5.33 kbit/s. Whereby G.729 and G.729 Annex A have the advantage of lower delay. Furthermore the G.723.1 has a small advantage in that its floating point version is already a standard. G.723.1 is used in the Intel Internet phone and G.729 was modified for digital cellular telephony. If we add one additional frame for the transmission delay the one-way total coder delay become 97.5ms for G.723.1 versus 35ms for G.729.

The quality judgement is based on their relative scores compared to 32kbit/s G.726. No marked differences have been observed for tandems, noisy backgrounds, input level, or speaker dependency. G.729 is marginally better in several of these cases. Therefore their selection is only based on the decision of bit rate, complexity and delay. These differences can also be described in terms of a tabular form

ATTRIBUTE	G.723.1	G.729	G.729 ANNEX **A**
Bit rate(s) Silence Compression	5.3 and 6.4 kbit/s Annex A	8 kbit/s Annex B	8 kbit/s Annex B
Delay Frame size Look ahead Total codec delay	30 ms 7.5 ms 67.5 ms	10 ms 5 ms 25 ms	10 ms 5 ms 25ms
Frame Length	20/24 bytes	10 bytes	10 bytes
Complexity MIPS RAM	16 2200 words	20 3000 words	10.5 2000 word

TABLE 2.4 Comparison of different standards

2.4.3 <u>PERFORMANCE VS COMPLEXITY:</u>

Here is a comparison of different ITU standards in terms of their performance and complexity. The comparison is shown in the tabular form below:

CODEC	DATA RATE (KBPS)	MOS	DELAY	MIPS
G.726 MULTIRATE ADPCM	16 - 40	2.0 - 4.3	0.125	6.5
G.723 MP-MLQ ACELP	5.3, 6.3	4.1	70	25
G.728 LD-CELP	16.0	4.1	2	37.5
G.729 CS-ACELP	8.0	4.1	20	34
G.729a CS-ACELP	8.0	3.4	20	17
MP3	64-56	3.6-3.8	59	35

TABLE 2.5 Complexity vs Performance

2.4.4 <u>CONCLUSION:</u>

After noticing the features and comparison of above mentioned speech compression techniques we have chosen G.729 giving the best output at 8 kb/s with minimum possible delay and better sound quality.

CHAPTER NO 3

CHAPTER NO 3

3.0 Introduction:

This Recommendation (ITU-T G.729) contains the description of an algorithm for the compression of speech and audio signals at 8 kbit/s using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP). The recommendation belongs to the H.324 family of standards. The option for using discontinuous transmission and noise fill during non-speech intervals can be incorporated as per requirement.

The coder is optimized to compress speech while maintaining a high quality at the mentioned rate using minimal complexity and delay. It is designed to operate with a digital signal, hence the analogue speech input is first digitized by performing telephone bandwidth filtering (Recommendation G.712) to remove the unwanted spectral components of human speech, then sampling it at 8000 Hz (in accordance with the Nyquist criterion), followed by a conversion to 16-bit linear PCM. This 16-bit digital signal is than used as input to the encoder.

The output of the decoder is again 16-bit linear PCM and should be converted back to an analogue(artificially reconstructed) signal by similar means(retracing the initial steps used for obtaining the digital signal). Other input/output characteristics, such as those specified by Recommendation G.711 for 64 kbit/s PCM(with logarithmic companding) data, should be converted to 16-bit linear PCM before inputting it to the encoder, or from 16-bit linear PCM to the appropriate format after decoding. The bitstream from the encoder to the decoder is completely defined within this Recommendation.

The basic idea behind the encoding-decoding process is the CS-ACELP analysis-by-synthesis methodology, the analysis procedure results in the extraction of all the necessary parameters(components/properties) of speech that are than coded in an appropriate manner for transmission whereas the synthesis procedure consists of artificially reconstructing the speech from the analyzed parameters. All this being easier said than done while achieving maximum compression and maintaining good quality.

In this chapter the CS-ACELP encoder and decoder have been discussed with an aim to create a familiarity of the reader with their basic operation. The later chapters describe the software that defines this coder in 16 bit fixed-point arithmetic and also the detailed step by step description of the encoder/decoder principles.

3.0.1 INPUT LIMITATIONS:

As described earlier the coder has been optimized to compress speech at 8Kbps while maintaining high quality, hence human speech is the most suitable input for the ADC (analogue to digital converter). None the less music and other audio signals can also be fed to the ADC but since that procedure involves telephone bandwidth filtering so there is a possibility of losing integral parts of the spectrum, moreover since the CS-ACELP coder uses the analysis methodology to extract the various parameters of human voice so it is not as efficient in case of audio or music signals because these signals contain properties and possess a spectrum different from human voice. In short the input should be confined to human speech for maximum compression and optimal quality.

3.0.2 General description of the coder/decoder operation

The CS-ACELP coder is based on the Code-Excited Linear-Prediction (CELP) coding model. The coder operates on speech frames of 10 ms corresponding to 80 samples at a sampling rate of 8000 samples per second. For every 10 ms frame, the speech signal is ‘analyzed’ to extract the parameters of the CELP model (linear-prediction filter coefficients, adaptive and fixed-codebook indices and gains), which shall all be discussed in detail in the following chapter. These parameters are encoded and transmitted. The bit allocation of the coder parameters is shown in Table 1,the procedure as to how these parameters have been assigned the specific number of bits shall be elaborated later .

Parameter	Codeword	Subframe 1	Subframe 2	Total per frame
Line spectrum pairs	$L0$, $L1$, $L2$, $L3$			18
Adaptive-codebook delay	$P1$, $P2$	8	5	13
Pitch-delay parity	$P0$	1		1
Fixed-codebook index	$C1$, $C2$	13	13	26
Fixed-codebook sign	$S1$, $S2$	4	4	8
Codebook gains (stage 1)	$GA1$, $GA2$	3	3	6
Codebook gains (stage 2)	$GB1$, $GB2$	4	4	8
Total				80

TABLE 3.1 Bit allocation of the 8 kbit/s CS-ACELP algorithm (10 ms frame)

At the decoder, these parameters are used to retrieve the ‘excitation’ and the ‘synthesis filter’ parameters. The speech is reconstructed by filtering this excitation through the short-term synthesis filter, as is shown in Figure 1. The short-term synthesis filter is based on a 10th order Linear Prediction (LP) filter. The long-term, or pitch synthesis filter is implemented using the so-called adaptive-codebook approach. After computing the reconstructed speech, it is further enhanced by a post filter.

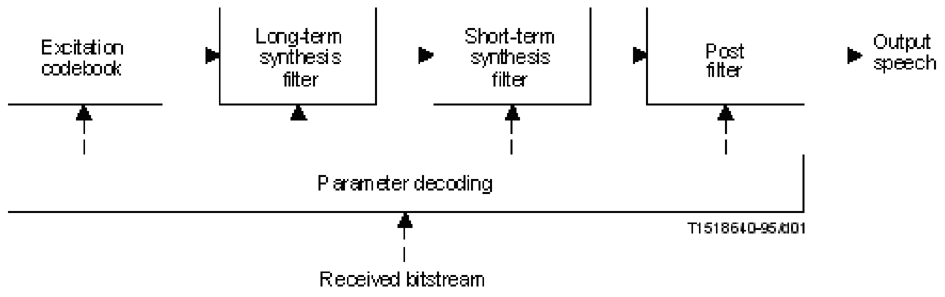


Figure 3.1Block Diagram of Celp Sybthesis model

3.0.3 <u>**Encoder overview:**</u>

The encoding principle is shown in Figure 2. The input signal is high-pass filtered to remove the DC component and scaled in the pre-processing block. The pre-processed signal serves as the input signal for all subsequent analysis. Linear predictive (LP) analysis is done once per 10 ms frame to compute the LP filter coefficients. These coefficients are converted to Line Spectrum Pairs (LSP) and quantized using predictive two-stage Vector Quantization (VQ) with 18 bits. The excitation signal is chosen by using an analysis-by-synthesis search procedure in which the error between the original and reconstructed speech is minimized according to a perceptually weighted distortion measure. This is done by filtering the error signal with a perceptual weighting filter, whose coefficients are derived from the unquantized LP filter. The amount of perceptual weighting is made adaptive to improve the performance for input signals with a flat frequency-response.

The excitation parameters (fixed and adaptive-codebook parameters) are determined per subframe of 5 ms (40 samples) each. The quantized and unquantized LP filter coefficients are used for the second subframe, while in the first subframe interpolated LP filter coefficients are used (both quantized and unquantized). An open-loop pitch delay is estimated once per 10 ms frame based on the perceptually weighted speech signal. Then the following operations are repeated for each subframe. The target signal is computed by filtering the LP residual through the weighted synthesis filter. The initial states of these filters are updated by filtering the error between LP residual and excitation. This is equivalent to the common approach of subtracting the zero-input response of the weighted synthesis filter from the weighted speech signal. The impulse response $h(n)$ of the weighted synthesis filter is computed. Closed-loop pitch analysis is then done (to find the adaptive-codebook delay and gain), using the target signal and impulse response, by searching around the value of the open-loop pitch delay. A fractional pitch delay with 1/3 resolution is used. The pitch delay is encoded with 8 bits in the first subframe and differentially encoded with 5 bits in the second subframe. The target signal $x(n)$ is updated by subtracting the (filtered) adaptive-codebook contribution, and this new target, $x'(n)$, is used in the fixed-codebook search to find the optimum excitation. An algebraic codebook with 17 bits is used for the fixed-codebook excitation. The gains of the adaptive and fixed-codebook contributions are vector quantized with 7 bits, (with MA prediction applied to the fixed-codebook gain). Finally, the filter memories are updated using the determined

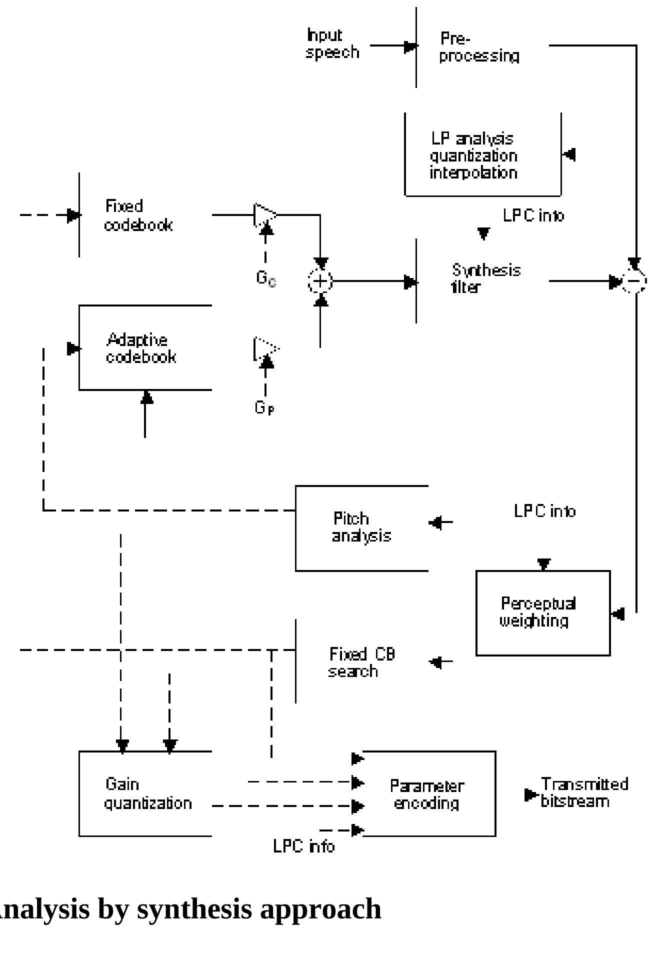


Figure 3.2 Analysis by synthesis approach

3.0.4 Decoder overview:

The decoder principle is shown in Figure 3. First, the parameter's indices are extracted from the received bitstream. These indices are decoded to obtain the coder parameters corresponding to a 10 ms speech frame. These parameters are the LSP coefficients, the two fractional pitch delays, the two fixed-codebook vectors, and the two sets of adaptive and fixed-codebook gains. The LSP coefficients are interpolated and converted to LP filter coefficients for each subframe. Then, for each 5 ms subframe the following steps are done:

- the excitation is constructed by adding the adaptive and fixed-codebook vectors scaled by their respective gains;
- the speech is reconstructed by filtering the excitation through the LP synthesis filter;
- the reconstructed speech signal is passed through a post-processing stage, which includes an adaptive postfilter based on the long-term and short-term synthesis filters, followed by a high-pass filter and scaling operation.

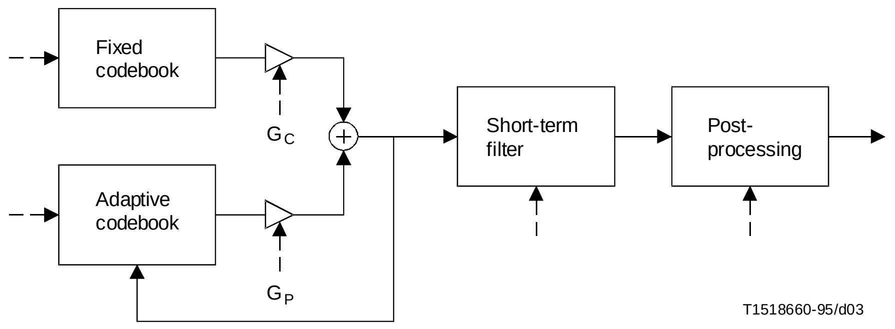


FIGURE 3

DECODER BLOCK DIAGRAM

3.0.5 Delay:

As mentioned earlier the coder uses 10 ms frames. In addition, there is a look-ahead of 5 ms, resulting in a total algorithmic delay of 15 ms. All additional delays in a practical operation of this coder are due to:

1) Time required for the processing of encoding and decoding operations.
2) Time taken during the transmission and reception over the communication link
3) Delay arising due to multiplexing operations i.e. when combining audio data with other data.

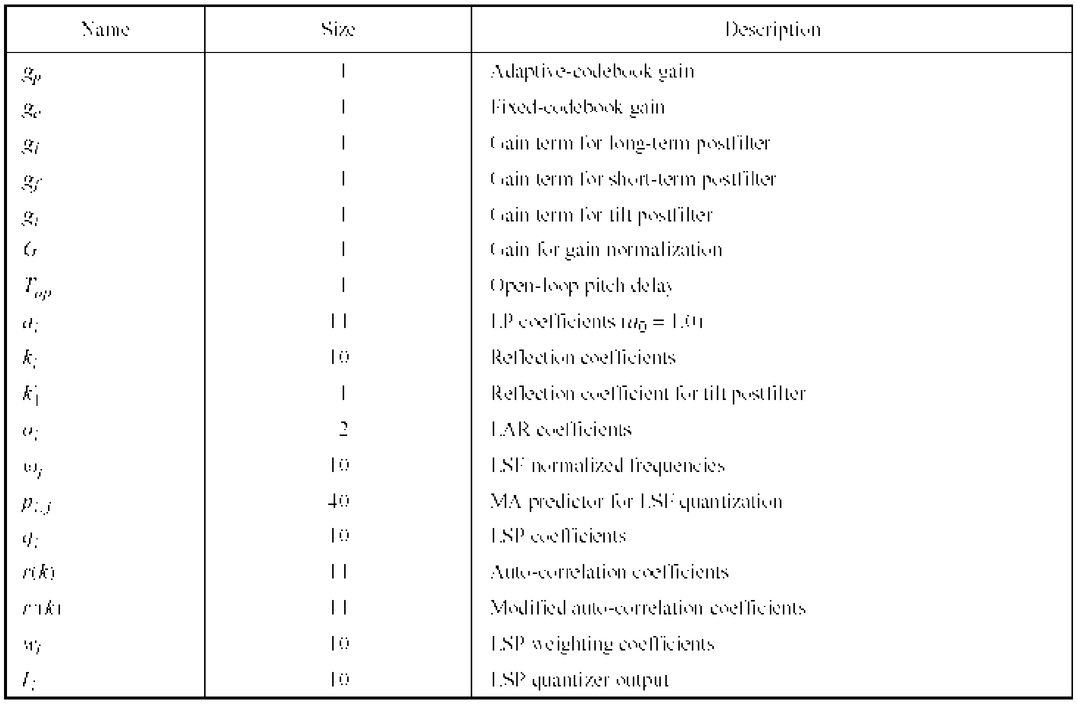
Below are some useful tables in which the description of the symbols, variables, constants and acronyms used in this recommendation(ITU-T G.729) have been given.

Name	Reference	Description
$1/\hat{A}(z)$	Equation (2)	LP synthesis filter
${}^{H}h1^{(z)}$	Equation (1)	Input high-pass filter
${}^{H}p^{(z)}$	Equation (78)	Long-term postfilter
${}^{H}f^{(z)}$	Equation (84)	Short-term postfilter
${}^{H}t^{(z)}$	Equation (86)	Tilt-compensation filter
${}^{H}h2^{(z)}$	Equation (91)	Output high-pass filter
$P(z)$	Equation (46)	Pre-filter for fixed codebook
$W(z)$	Equation (27)	Weighting filter

TABLE 3. 2 Glossary of most relevant symbols, ITU-T recommendation G.729

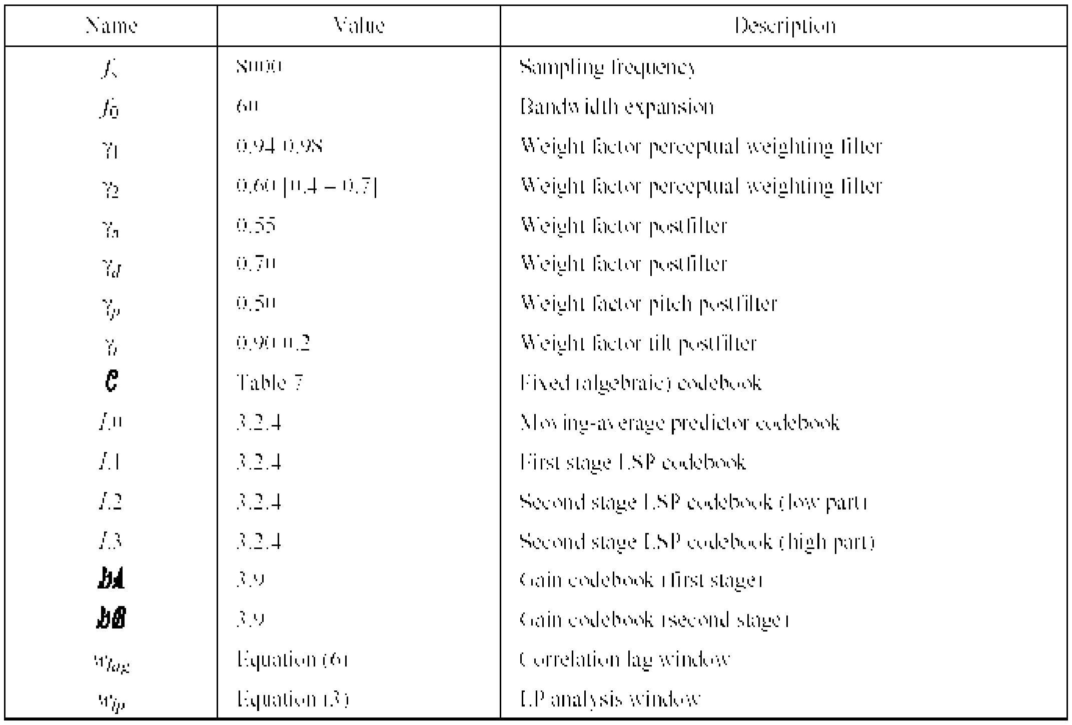
Name	Reference	Description
$c(n)$	3.8	Fixed-codebook contribution
$d(n)$	3.8.1	Correlation between target signal and $h(n)$
$ew(n)$	3.10	Error signal
$h(n)$	3.5	Impulse response of weighting and synthesis filters
$r(n)$	3.6	Residual signal
$s(n)$	3.1	Pre-processed speech signal
$\hat{s}(n)$	4.1.6	Reconstructed speech signal
$s'(n)$	3.2.1	Windowed speech signal
$sf(n)$	4.2	Postfiltered output
$sf'(n)$	4.2	Gain-scaled postfiltered output
$sw(n)$	3.6	Weighted speech signal
$x(n)$	3.6	Target signal
$x'(n)$	3.8.1	Second target signal
$u(n)$	3.10	Excitation to LP synthesis filter
$v(n)$	3.7.1	Adaptive-codebook contribution
$y(n)$	3.7.3	Convolution $v(n) * h(n)$
$z(n)$	3.9	Convolution $c(n) * h(n)$

TABLE 3 .3 Glossary of most relevant signals, ITU-T recommendation G.729



Name	Size	Description
g_p	1	Adaptive-codebook gain
g_c	1	Fixed-codebook gain
g_l	1	Gain term for long-term postfilter
g_f	1	Gain term for short-term postfilter
g_t	1	Gain term for tilt postfilter
G	1	Gain for gain normalization
T_{op}	1	Open-loop pitch delay
a_i	11	LP coefficients ($a_0 = 1.0$)
k_i	10	Reflection coefficients
k_1	1	Reflection coefficient for tilt postfilter
o_i	2	LAR coefficients
ω_i	10	LSF normalized frequencies
$p_{i,j}$	40	MA predictor for LSF quantization
q_i	10	LSP coefficients
$r(k)$	11	Auto-correlation coefficients
$r'(k)$	11	Modified auto-correlation coefficients
w_i	10	LSP weighting coefficients
l_i	10	LSP quantizer output

***TABLE 3.4 Glossary of most relevant variables*, ITU-T recommendation G.729**



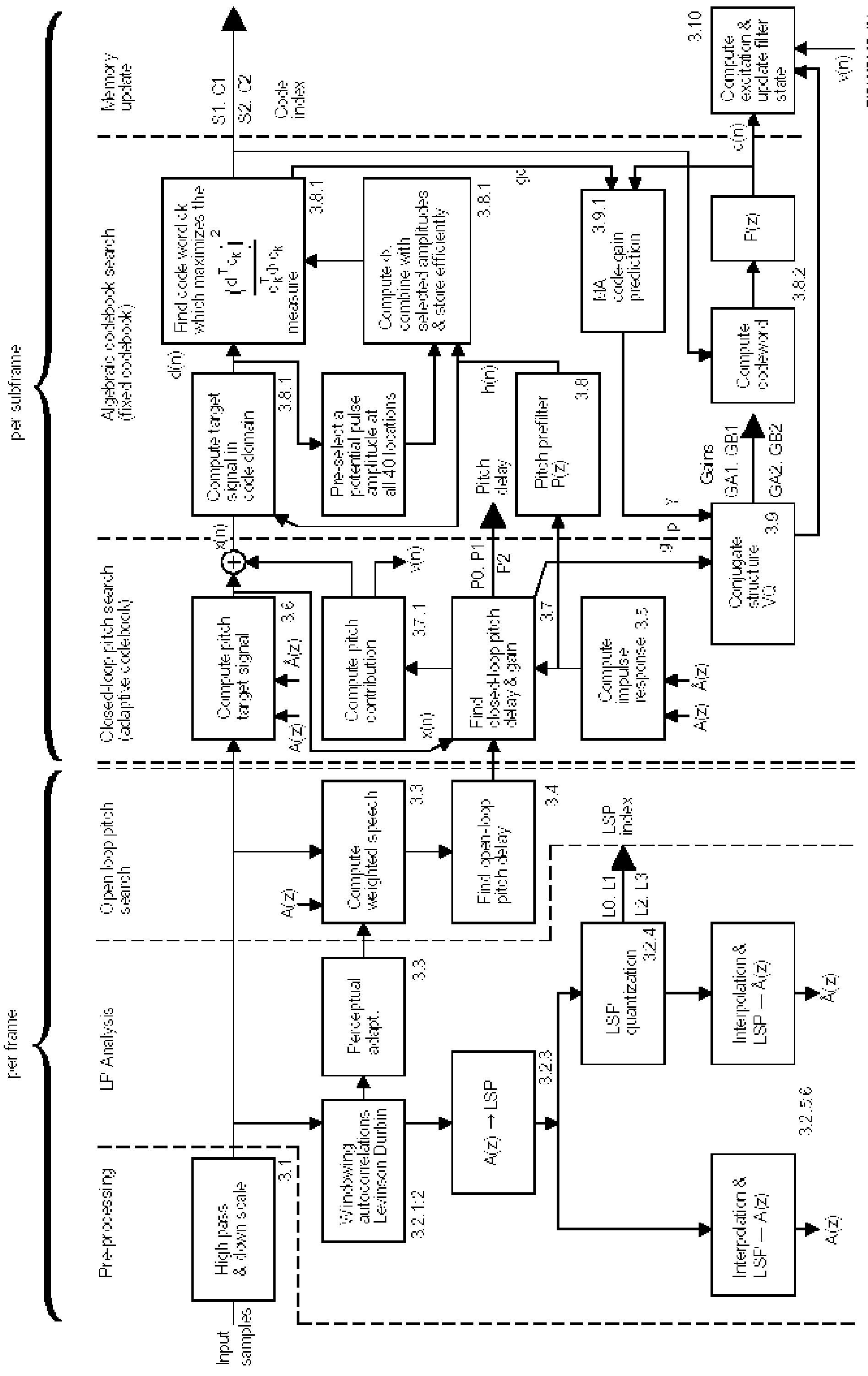
Name	Value	Description
f_s	8000	Sampling frequency
f_0	60	Bandwidth expansion
γ_1	0.94 0.98	Weight factor perceptual weighting filter
γ_2	0.60 [0.4 – 0.7]	Weight factor perceptual weighting filter
γ_n	0.55	Weight factor postfilter
γ_d	0.70	Weight factor postfilter
γ_p	0.50	Weight factor pitch postfilter
γ_t	0.90 0.2	Weight factor tilt postfilter
$\mathcal{C}$	Table 7	Fixed (algebraic) codebook
$L0$	3.2.4	Moving-average predictor codebook
$L1$	3.2.4	First stage LSP codebook
$L2$	3.2.4	Second stage LSP codebook (low part)
$L3$	3.2.4	Second stage LSP codebook (high part)
$\mathcal{GA}$	3.9	Gain codebook (first stage)
$\mathcal{GB}$	3.9	Gain codebook (second stage)
w_{lag}	Equation (6)	Correlation lag window
w_{lp}	Equation (3)	LP analysis window

<u>**TABLE 3.5Glossary of most relevant constants, ITU-T recommendation G.729**</u>

Acronym	**Description**
CELP	Code-Excited Linear-Prediction
CS-ACELP	Conjugate-Structure Algebraic-CELP
MA	Moving Average
MSB	Most Significant Bit
MSE	Mean-Squared Error
LAR	Log Area Ratio
LP	Linear Prediction
LSP	Line Spectral Pair
LSF	Line Spectral Frequency
VQ	Vector quantization

TABLE 3.6 Glossary of acronyms, ITU-T recommendation G.729

Now we will see a more detailed description of the encoder/decoder operation along with their diagrams so that the reader can have an idea as to how the process of speech compression takes place.



per frame
per subframe
Pre-processing
LP Analysis
Open loop pitch search
Closed-loop pitch search (adaptive codebook)
Algebraic codebook search (fixed codebook)
Memory update
Input samples
High pass & down scale
3.1
Windowing autocorrelations Levinson Durbin
3.2.1:2
Perceptual adapt.
3.3
A(z) → LSP
3.2.3
LSP quantization
3.2.4
Interpolation & LSP → A(z)
Interpolation & LSP → Â(z)
3.2.5:6
LSP index
L0, L1
L2, L3
Compute weighted speech
3.3
Find open-loop pitch delay
3.4
Compute pitch target signal
3.6
Compute pitch contribution
3.7.1
Find closed-loop pitch delay & gain
3.7
Compute impulse response
3.5
Conjugate structure VQ
3.9
Pitch delay
P0, P1
P2
Compute target signal in code domain
3.8.1
Pre-select a potential pulse amplitude at all 40 locations
Pitch prefilter P(z)
3.8
Compute c(n), combine with selected amplitudes & store efficiently
3.8.1
Find code word ck which maximizes the measure
3.8.1
MA code-gain prediction
3.9.1
Compute codeword
3.8.2
F(z)
Compute excitation & update filter state
3.10
Gains
GA1, GB1
GA2, GB2
S1, C1
S2, C2
Code index
x(n)
y(n)
d(n)
h(n)
c(n)
v(n)
gc
gp
T1518670-95/d04

Figure 3.3 BLOCK DIAGRAM OF THE FUNCTIONAL ENCODER.ITU-T BLOCK DIAGRAM OF THE FUNCTIONAL ENCODER.ITU-T G.729

As seen in the functional diagram of the encoder(ITU-T recommendation G.729), it consists of six portions i.e.

Pre-processing.
LP Analysis.
Open loop pitch search.
Closed loop pitch search.
Algebraic codebook search.
Memory update.

Now we shall have a closer look at each of the blocks contained within these portions so as to have a better understanding of the encoder operation.

3.1 Pre-Processing:

The pre-processing block gets it input in the form of 16 bit PCM. Its main function is to first high pass this input to remove the DC component and any low frequency components that might be present and secondly to divide the input by a factor of two so that while implementing the code in 'C' the variables should not get overflowed, this is known as fixed point implementation[7]. As given in the ITU-T recommendation a second order pole/zero filter with a cutoff freq of 140 Hz is used. Both the scaling and high-pass filtering are combined by dividing the coefficients at the numerator of this filter by 2. The resulting filter is given by:

$$H_{h1}(z) = \frac{0.46363718 - 0.92724705z^{-1} + 0.46363718z^{-2}}{1 - 1.90594 65z^{-1} + 0.9114024z^{-2}} \quad \textbf{(1)}$$

The input signal when passed through this filter will be known as s(n) and be used in all the subsequent coder operations.

3.2 LP Analysis:

The LP Analysis is performed once per frame, there are three steps involved in calculating the LP coefficients, the first is the windowing.

3.2.1 Windowing

The LP analysis window is a combination of two widely used windows namely the hamming window and the second one being the cosine cycle function[7]. The window is given by:

$$w_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{399}\right) & n = 0,\dots,199 \\ \cos\left(\frac{2\pi(n-200)}{159}\right) & n = 200,\dots,239 \end{cases} \qquad \textbf{(2)}$$

However it is dependent on the developer to use a window which has smaller width of main-lobe and greater distance between the main lobe and first side lobe than the one recommended[8].

Another important thing to note is that the windowing is not just performed on a single speech frame rather the LP analysis window uses 120 samples from the past speech frame, 80 from the present one and 40 samples from the future frame[7]. The use of the future samples results into an extra delay of 5ms at the encoder stage[7]. the windowing procedure is illustrated in the figure below.

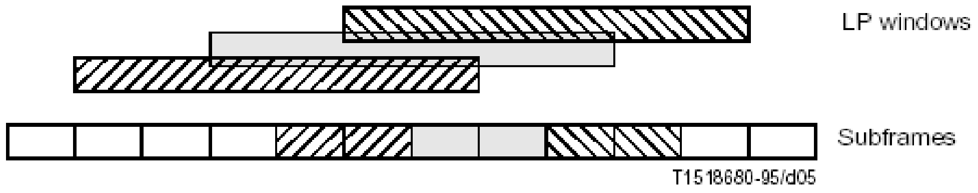


Figure 3.5Windowing procedure in LP Analysis.

The windowed speech is given by[7]:

$$s'(n) = w_{lp}(n)\, s(n) \quad n = 0,\dots,239 \qquad (3)$$

3.2.2 ***Autocorrelation:***

The next step in the LP analysis is performing autocorrelation of the windowed speech[7],

$$r(k) = \sum_{n=k}^{239} s'(n)\, s'(n-k) \qquad k = 0,\dots,10$$

(4)

The boundaries of the autocorrelation coefficients are set in a manner to avoid problems from low level input and also noise margin is added to them[7]. The modified autocorrelation coefficients are given by:

$$r'(0) = 1.0001\, r(0)$$
$$r'(k) = w_{lag}(k)\, r(k) \qquad k = 1,\dots,10$$

(5)

3.2.3 ***Levinson Durbin Algorithm***

The third and final step in calculating the LP coefficients is to solve the modified autocorrelation coefficients set of equations[7], given as

$$\sum_{i=1}^{10} a_i r'(|i-k|) = -r'(k) \qquad k = 1,\dots,10$$

(6)

with help of the Levinson-Durbin algorithm. The final solution is given as $a_j = a_j^{[10]}$, $j = 0, ...,10$, with $a_0 = 1$. These shall be known as the LPC coefficients for all further calcutaions[7].

3.2.4 ***LP to LSP conversion***

The next step in the LP analysis is the determination of the LSP coefficients from the LP coefficients. The LP filter coefficients a_i, $i = 0,...10$ are converted to Line Spectral Pair (LSP) coefficients for quantization and interpolation purposes[7]. For a 10th order LP filter, the LSP coefficients are defined as the roots of the sum and difference polynomials:

$$F_1'(z) = A(z) + z^{-11}A(z^{-1}) \qquad \textbf{(7)}$$

and:

$$F_2'(z) = A(z) - z^{-11}A(z^{-1}) \qquad \textbf{(8)}$$

respectively. The polynomial $F_1'(z)$ is symmetric, and $F_2'(z)$ is anti-symmetric. It can be proven that all roots of these polynomials are on the unit circle and they alternate each other. $F_1'(z)$ has a root $z = -1$ ($\omega = \pi$) and $F_2'(z)$ has a root $z = 1$ ($w = 0$). These two roots are eliminated by defining the new polynomials[7]:

$$F_1(z) = F_1'(z) / (1 + z^{-1}) \qquad \textbf{(9)}$$

and:

$$F_2(z) = F_2'(z) / (1 - z^{-1}) \qquad \textbf{(10)}$$

Each polynomial has five conjugate roots on the unit circle ($e^{\pm j\omega_i}$), and they can be written as:

$$F_1(z) = \prod_{i=1,3,\ldots,9} (1 - 2q_i z^{-1} + z^{-2})$$ **(11)**

$$F_2(z) = \prod_{i=2,4,\ldots,10} (1 - 2q_i z^{-1} + z^{-2})$$ **(12)**

where $q_i = \cos(\omega_i)$ [7]. The coefficients ω_i are the Line Spectral Frequencies (LSF) and they satisfy the ordering property $0 < \omega_i < \omega_2 < \ldots < \omega_{10} < \pi$. The coefficients q_i are referred to as the LSP coefficients in the cosine domain[7].
Since both polynomials $F_1(z)$ and $F_2(z)$ are symmetric only the first five coefficients of each polynomial need to be computed. The coefficients of these polynomials are found by the recursive relations:

$$\begin{aligned} f_1(i+1) &= a_{i+1} + a_{10-i} - f_1(i) \qquad i = 0,\ldots,4 \\ f_2(i+1) &= a_{i+1} - a_{10-i} + f_2(i) \qquad i = 0,\ldots,4 \end{aligned}$$ **(13)**

where $f_1(0) = f_2(0) = 1.0$. The LSP coefficients are found by evaluating the polynomials $F_1(z)$ and $F_2(z)$ at 60 points equally spaced between 0 and π and checking for sign changes. A sign change signifies the existence of a root and the sign change interval is then divided four times to allow better tracking of the root. The Chebyshev polynomials are used to evaluate $F_1(z)$ and $F_2(z)$. In this method the roots are found directly in the cosine domain[7]. The polynomials $F_1(z)$ or $F_2(z)$, evaluated at $z = e^{j\omega}$, can be written as:

$$F(\omega) = 2e^{-j5\omega}\, C(x)$$

(14)

with:

$$C(x) = T_5(x) + f(1)T_4(x) + f(2)T_3(x) + f(3)T_2(x) + f(4)T_1(x) + f(5)/2$$ **(15)**

where $T_m(x) = \cos(m\omega)$ is the mth order Chebyshev polynomial, and $f(i)$, $i = 1,...,5$, are the coefficients of either $F_1(z)$ or $F_2(z)$, computed using Equation (11).

The polynomial $C(x)$ is evaluated at a certain value of $x = \cos(\omega)$ using the recursive relation:

```
for k = 4 down to 1
        $b_k = 2xb_{k+1} - b_{k+2} + f(5 - k)$
end
$C(x) = xb_1 - b_2 + f(5)/2$
```

with initial values $b_5 = 1$ and $b_6 = 0$.

3.2.5 *Quantization of the LSP coefficients*

The next block in the LP analysis is the quantization of the LSP coefficients. The LSP coefficients q_i are quantized using the LSF representation ω_i in the normalized frequency domain $[0, \pi]$[7]; that is:

$$\omega_i = \arccos(q_i) \qquad i = 1,...,10$$

(16)

The predicted LSF coefficients of the current frame are matched with the computed ones and their difference is quantized in two steps according to the ITU-T recommendation[7]. The first step involves the quantization of the computed vector with a 10 dimensional(10 values that range between 0-3.1) codebook L1 having 128 different combinations and represented by 7 bits, index number of the vector having the minimum difference with the computed one is selected and the vector is thus quantized to discrete values and the index number assigned can also be used at the decoder side to get the quantized LSP coefficients[7].

The second stage consists of a split vector quantization using two 5 dimensional codebooks L2 and L3 having 32 combinations each and represented by 5 bits(each) [7]. This second stage is implemented to reduce the error between the quantized and original vector. The entries of the two codebooks range between –1<0<1. The quantized vector is divided into two equal parts, with first part being compared with entries of codebook L2 and second part being compared with entries of codebook L3. The index number of the vector that resembles the difference between the quantized and computed vector most closely is chosen and same procedure is carried out for the second part. Thus we can greatly reduce the error between the quantized and computed coefficients if the values of the codebook entries are added together in proper manner i.e. the codebook L2 added to the first half of the quantized vector L1 and the codebook L3 added to the second half of the vector L1.

3.2.6 Interpolation of the LSP coefficients

The final step in the LP analysis is interpolation of the LSP coefficients and determining the quantized LPC coefficients. The quantized (and unquantized) LP coefficients are used for the second sub frame. For the first sub frame, the quantized (and unquantized) LP coefficients are obtained by linear interpolation of the corresponding parameters in the adjacent sub frames[7]. The interpolation is done on the LSP coefficients in the cosine domain. Let $q_i^{(current)}$ be the LSP coefficients computed for the current 10 ms frame, and $q_i^{(previous)}$ the LSP coefficients computed in the previous 10 ms frame. The (unquantized) interpolated LSP coefficients in each of the two sub frames are given by[7]:

$$\begin{aligned} &Subframe\ 1:\ q_i^{(1)} = 0.5q_i^{(previous)} + 0.5q_i^{(current)} && i = 1,\ldots,10 \\ &Subframe\ 2:\ q_i^{(2)} = q_i^{(current)} && i = 1,\ldots,10 \end{aligned} \qquad \textbf{(17)}$$

The same interpolation procedure is used for the interpolation of the quantized LSP coefficients by substituting q_i by $\hat{q}_i$ in Equation (17) [7].

Once the LSP coefficients are quantized and interpolated, they are converted back to the LP coefficients a_i. This conversion is done as follows. The coefficients of $F_1(z)$ and $F_2(z)$ are found by expanding Equations (11) and (12) knowing the quantized and interpolated LSP coefficients. The coefficients $f_1(i)$, $i = 1,...,5$, are computed from q_i using the recursive relation:

```
for $i = 1$ to $5$
    $f_1(i) = -2q_{2i-1} f_1(i-1) + 2f_1(i-2)$
    for $j = i-1$ down to $1$
        $f_1^{(i)}(j) = f_1^{(i-1)}(j) - 2q_{2i-1} f_1^{(i-1)}(j-1) + f_1^{(i-1)}(j-2)$
    end
end
```

with initial values $f_1(0) = 1$ and $f_1(-1) = 0$. The coefficients $f_2(i)$ are computed similarly by replacing q_{2i-1} by q_{2i}[7].

Once the coefficients $f_1(i)$ and $f_2(i)$ are found, $F_1(z)$ and $F_2(z)$ are multiplied by $1 + z^{-1}$ and $1 - z^{-1}$, respectively, to obtain $F_1'(z)$ and $F_2'(z)$[7]; that is:

$$\begin{aligned} f_1'(i) &= f_1(i) + f_1(i-1) \qquad & i &= 1.....5 \\ f_2'(i) &= f_2(i) - f_2(i-1) \qquad & i &= 1.....5 \end{aligned} \tag{18}$$

Finally the LP coefficients are computed from $f_1'(i)$ and $f_2'(i)$ by[7]:

$$a_i = \begin{cases} 0.5f_1'(i) + 0.5f_2'(i) & i = 1.....5 \\ 0.5f_1'(11-i) - 0.5f_2'(11-i) & i = 6.....10 \end{cases} \tag{19}$$

This is directly derived from the relation $A(z) = (F_1'(z) + F_2'(z))/2$, and because $F_1'(z)$ and $F_2'(z)$ are symmetric and anti-symmetric polynomials, respectively.

3.2.7 Perceptual Adaptation

A small but integral part of the LP analysis is the perceptual adaptation, this block consists of the weights that shall be used in the next block for weighting the speech signal and also the unquantized LPC coefficients[7]. The weighting filter is given by[7]:

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} = \frac{1 + \sum_{i=1}^{10} \gamma_1^i a_i z^{-i}}{1 + \sum_{i=1}^{10} \gamma_2^i a_i z^{-i}} \qquad \textbf{(20)}$$

The variables $\gamma 1$ and $\gamma 2$ are made functions of the spectral shape of the input signal to make the weighting more efficient [7]. The reflection coefficients obtained as a by product of the Levinson-Durbin algorithm can be used to give the spectral shape.

3.3 Open Loop Pitch Search:

In the first block of this section the previously computed weights are used to weight the original speech signal and also the LPC coefficients[7].

3.3.1 Open loop Pitch delay

This weighted speech is then used to find the open loop pitch delay which is a close estimate for delay given to the excitation signal. This open-loop pitch analysis is done once per frame (10 ms) [7]. The open-loop pitch estimation uses the weighted speech signal $sw(n)$ and is done as follows:
In the first step, three maxima of the correlation:

$$R(k) = \sum_{n=0}^{79} sw(n)sw(n-k) \qquad (21)$$

are found in the following three ranges[7]:

$i = 1: 80.....143$

$i = 2: 40.....79$

$i = 3: 20.....39$

The retained maxima $R(t_i)$, $i = 1,...,3$, are normalized through[7]:

$$R'(t_i) = \frac{R(t_i)}{\sqrt{\sum_n sw^2(n-t_i)}} \qquad i = 1.....3$$

(22)

The winner among the three normalized correlations is selected by favoring the delays with the values in the lower range. This is done by weighting the normalized correlations corresponding to the longer delays. The best open-loop delay T_{op} is determined as follows:

```
$T_{op} = t_1$
$R'(T_{op}) = R'(t_1)$
$if\ R'(t_2) \geq 0.85R'(T_{op})$
        $R'(T_{op}) = R'(t_2)$
        $T_{op} = t_2$
$end$
$if\ R'(t_3) \geq 0.85R'(T_{op})$
        $R'(T_{op}) = R'(t_3)$
        $T_{op} = t_3$
$end$
```

This procedure of dividing the delay range into three sections and favoring the smaller values is used to avoid choosing pitch multiples[7]. The T_{op} calculated here will be used to find the exact delay given to the excitation signal in the next portions.

3.4 Closed Loop Search

In this section the exact delay for the excitation signal is found. Now we will discuss the blocks that are contained within this section.

3.4.1 Computation of the impulse response

The impulse response $h(n)$ of the weighted synthesis filter $W(z)/\hat{A}(z)$ is needed for the search of adaptive and fixed codebooks[7]. The impulse response $h(n)$ is computed for each sub frame by filtering a signal consisting of the coefficients of the filter $A(z/\gamma 1)$ extended by zeros through the two filters $1/\hat{A}(z)$ and $1/A(z/\gamma 2)$. [7]

3.4.2 Computation of the target signal

The procedure for computing the target signal, used in this Recommendation, is the filtering of the LP residual signal $r(n)$ through the combination of synthesis filter $1/\hat{A}(z)$ and the weighting filter $A(z/\gamma 1)/A(z/\gamma 2)$ [7]. After determining the excitation for the sub frame, the initial states of these filters are updated by filtering the difference between the residual and excitation signals.

The residual signal $r(n)$, which is needed for finding the target vector is also used in the adaptive-codebook search to extend the past excitation buffer. This simplifies the adaptive-codebook search procedure for delays less than the sub frame size of 40.

The LP residual is given by[7]:

$$r(n) = s(n) + \sum_{i=1}^{10} a_i s(n - i) \qquad n = 0,\ldots,39$$

(23)

3.4.3 Adaptive Codebook Search

The adaptive-codebook parameters (or pitch parameters) are the delay and gain[7]. In the adaptive-codebook approach for implementing the pitch filter, the excitation is repeated for delays less than the subframe length[7]. In the search stage, the excitation is extended by the LP residual to simplify the closed-loop search. The adaptive-codebook search is done every (5 ms) subframe. In the first subframe, a fractional pitch delay *T*1 is used with a resolution of 1/3 in the range of [19 1/3 , 84 2/3] and integers only in the range [85, 143]. For the second subframe, a delay *T*2 with a resolution of 1/3 is always used in the range *int*(*T*1) − 5 2/3 , *int*(*T*1) + 4 2/3, where *int*(*T*1) is the integer part of the fractional pitch delay *T*1 of the first sub frame[7]. This range is adapted for the cases where *T*1 straddles the boundaries of the delay range. For each sub frame the optimal delay is determined using closed-loop analysis that minimizes the weighted mean-squared error[7]. In the first sub frame the delay *T*1 is found by searching a small range (six samples) of delay values around the open loop delay *Top.*

The search boundaries *tmin* and *tmax* are defined by[7]:

```
$t_{min} = T_{op} - 3$
$if\ t_{min} < 20\ then\ t_{min} = 20$
$t_{max} = t_{min} + 6$
 $if\ t_{max} > 143\ then$
      $t_{max} = 143$
      $t_{min} = t_{max} - 6$
$end$
```

Similarly for the second sub frame, closed-loop pitch analysis is done around the pitch selected in the first subframe to find the optimal delay *T*2.

The closed-loop pitch search minimizes the mean-squared weighted error between the original and reconstructed speech.

This is achieved by maximizing the term:

$$R(k) = \frac{\sum_{n=0}^{39} x(n)\, y_k(n)}{\sqrt{\sum_{n=0}^{39} y_k(n)\, y_k(n)}} \tag{24}$$

where $x(n)$ is the target signal and $yk(n)$ is the past filtered excitation at delay k [past excitation convolved with $h(n)$] [7]. Note that the search range is limited around a pre-selected value, which is the open-loop pitch *Top* for the first sub frame, and *T*1 for the second sub frame.

3.4.4 Computation of Adaptive Codebook Vector

Once the pitch delay has been determined, the adaptive-codebook vector $v(n)$ is computed by interpolating the past excitation signal $u(n)$ at the given integer delay k and fraction t[7]:

$$v(n) = \sum_{i=0}^{9} u(n-k+i) b_{30}(t+3i) + \sum_{i=0}^{9} u(n-k+1+i) b_{30}(3-t+3i) \quad n = 0,\dots,39 \quad t = 0, 1, 2 \tag{25}$$

The interpolation filter *b*30 is based on a Hamming windowed *sinc* functions truncated at ± 29 and padded with zeros at ± 30 [*b*30(30) = 0]. The filter has a cut-off frequency (–3 dB) at 3600 Hz in the over-sampled domain[7].

3.5 Algebraic(Fixed) Codebook Search

The fixed codebook is based on an algebraic codebook structure using an Interleaved Single-Pulse Permutation (ISPP) design[7]. In this codebook, each codebook vector contains four non zero pulses. Each pulse can have either the amplitudes +1 or −1, and can assume the positions given in Table 7[7].

Pulse	Sign	Positions
i0	S0: ± 1	M0: 0, 5, 10, 15, 20, 25, 30, 35
i1	S1: ± 1	M1: 1, 6, 11, 16, 21, 26, 31, 36
i2	S2: ± 1	M2: 2, 7, 12, 17, 22, 27, 32, 37
i3	S3: ± 1	M3: 3, 8, 13, 18, 23, 28, 33, 38
		4, 9, 14, 19, 24, 29, 34, 39

Table 3.7, G.729

The codebook vector *c*(*n*) is constructed by taking a zero vector of dimension 40, and putting the four unit pulses at the found locations, multiplied with their corresponding sign[7]:

$$c(n) = s_0\delta(n - m_0) + s_1\delta(n - m_1) + s_2\delta(n - m_2) + s_3\delta(n - m_3) \quad n = 0,...,39 \tag{26}$$

where δ(0) is a unit pulse.

For delays less than 40, the codebook *c*(*n*) of Equation (26) is modified according to[7]:

$$c(n) = \begin{cases} c(n) & n = 0,...,T - 1 \\ c(n) + \beta c(n - T) & n = T,...,39 \end{cases} \tag{26-1}$$

This modification is incorporated in the fixed-codebook search by modifying the impulse response *h*(*n*) according to:

$$h(n) = \begin{cases} h(n) & n = 0,...,T - 1 \\ h(n) + \beta h(n - T) & n = T,...,39 \end{cases}$$

Where β is the pitch gain and T is the integer component of pitch delay of the current sub-frame.

3.5.1 Fixed-codebook search procedure

The fixed codebook is searched by minimizing the mean-squared error between the weighted input speech *sw*(*n*) and the weighted reconstructed speech. The target signal used in the closed-loop pitch search is updated by subtracting the adaptive-codebook contribution[7]. That is:

$$x'(n) = x(n) - g_p y(n) \qquad n = 0,\ldots,39 \tag{27}$$

where *y*(*n*) is the filtered adaptive-codebook vector and *gp* the adaptive-codebook gain.

The computation for the fixed codebook vector involves a tedious number of loops and in this ITU-T recommendation a focused search method is adopted in which a pre-computed threshold is tested before entering the last loop. Once the fixed codebook vector is found it is transmitted along with the other parameters, the components of the fixed codebook vector are the positions and signs of the excitation signal.

3.5.2 Codeword computation of the fixed codebook:

The pulse positions of the pulses *i*0, *i*1 and *i*2, are encoded with 3 bits each, while the position of i3 is encoded with 4 bits[7]. Each pulse amplitude is encoded with 1 bit. This gives a total of 17 bits for the 4 pulses. By defining $s = 1$ if the sign is positive and $s = 0$ if the sign is negative, the sign codeword 'S' and fixed-codebook codeword 'C' are obtained from[7]:

$$S = s0 + 2s1 + 4s2 + 8s3$$

$$C = (m_0/5) + 8(m_1/5) + 64(m_2/5) + 512(2(m_3/5) + jx) \quad (28)$$

where *jx* = 0 if *m*3 = 3,8,...,38, and *jx* = 1 if *m*3 = 4,9...,39.

3.5.3 Quantization of the gains

Other parameters to be sent are the codebook gains for the excitation signal. The adaptive-codebook gain (pitch gain) and the fixed-codebook gain are vector quantized using 7 bits[7]. The gain codebook search is done by minimizing the mean squared weighted error between original and reconstructed speech which is given by:

$$E = x^t x + g_p^2 y^t y + g_c^2 z^t z - 2g_p x^t y - 2g_c x^t z + 2g_p g_c y^t z \quad \textbf{(29)}$$

where ***x*** is the target vector (see 3.6), ***y*** is the filtered adaptive-codebook vector of Equation (44), and ***z*** is the fixed codebook vector convolved with *h*(*n*),

$$z(n) = \sum_{i=0}^{n} c(i)h(n-i) \qquad n = 0,\ldots,39 \quad \textbf{(30)}$$

3.5.4 Codebook search for gain quantization

The adaptive-codebook gain, *gp*, and the factor γ are vector quantized using a two-stage conjugate structured codebook[7]. The first stage consists of a 3 bit two-dimensional codebook ϑA, and the second stage consists of a 4 bit two-dimensional codebook ϑB . The first element in each codebook represents the quantized adaptive-codebook gain *gp* ^ , and the second element represents the quantized fixed-codebook gain correction factor γ^. Given codebook indices GA and GB for ϑA and ϑB, respectively, the quantized adaptive-codebook gain is given by:

$$\hat{g}p = \vartheta A_1(GA) + \vartheta B_1(GB) \quad \textbf{(31)}$$

and the quantized fixed-codebook gain by:

$$\hat{g}c = gc'\,\hat{\gamma} = gc'\,(\,\vartheta A_2\,(GA) + \vartheta B_2\,(GB)) \qquad \textbf{(32)}$$

This conjugate structure simplifies the codebook search, by applying a pre-selection process[7].

3.6 Memory update

An update of the states of the synthesis and weighting filters is needed to compute the target signal in the next subframe[7].
After the two gains are quantized, the excitation signal, *u*(*n*), in the present subframe is obtained using:

$$u(n) = \hat{g}_p v(n) + \hat{g}_c c(n) \qquad n = 0,\ldots,39 \qquad \textbf{(33)}$$

where *gp*^ and *gc*^ are the quantized adaptive and fixed-codebook gains, respectively, *v*(*n*) is the adaptive-codebook vector (interpolated past excitation), and *c*(*n*) is the fixed-codebook vector including harmonic enhancement. The states of the filters can be updated by filtering the signal *r*(*n*)-*u*(*n*) (difference between residual and excitation) through the filters 1/*Â*(*z*) and *A*(z/γ1)/*A*(*z*/γ2) for the 40 sample subframe and saving the states of the filters. This would require three filter operations. A simpler approach, which requires only one filter operation, is as follows. The locally reconstructed speech *s*^(*n*) is computed by filtering the excitation signal through 1/*Â*(*z*). The output of the filter due to the input *r*(*n*) − *u*(*n*) is equivalent to *e*(*n*) = *s*(*n*) − *s*^(*n*). So the states of the synthesis filter 1/*Â*(*z*) are given by *e*(*n*), *n* = 30,...,39. Updating the states of the filter *A*(*z*/γ1)/*A*(*z*/γ2) can be done by filtering the error signal *e*(*n*) through this filter to find the perceptually weighted error *ew*(*n*). However, the signal *ew*(*n*) can be equivalently found by:

$$ew(n) = x(n) - \hat{g}_p y(n) - \hat{g}_c z(n) \qquad \textbf{(34)}$$

Since the signals $x(n)$, $y(n)$ and $z(n)$ are available, the states of the weighting filter are updated by computing $ew(n)$ as in Equation (32) for $n = 30,...,39$. This saves two filter operations[7].

3.7 Decoder operations

The decoder block diagram has already been shown in figure 3.2[7], the coded parameters sent by the encoder (LP coefficients, codebook vectors, gains) are first decoded and these decoded parameters are used to reconstruct the speech. Once the encoder functionality has been understood than the reconstruction process seems quite easy.

A list of transmitted parameters is given in table 3.8 [7].

Symbol	Description	Bits
$L0$	Switched MA predictor of LSP quantizer	1
$L1$	First stage vector of quantizer	7
$L2$	Second stage lower vector of LSP quantizer	5
$L3$	Second stage higher vector of LSP quantizer	5
$P1$	Pitch delay first sub frame	8
$P0$	Parity bit for pitch delay	1
$C1$	Fixed codebook first sub frame	13
$S1$	Signs of fixed-codebook pulses 1st sub frame	4
$GA1$	Gain codebook (stage 1) 1st sub frame	3
$GB1$	Gain codebook (stage 2) 1st sub frame	4
$P2$	Pitch delay second sub frame	5
$C2$	Fixed codebook 2nd sub frame	13
$S2$	Signs of fixed-codebook pulses 2nd sub frame	4
$GA2$	Gain codebook (stage 1) 2nd sub frame	3
$GB2$	Gain codebook (stage 2) 2nd sub frame	4

Table 3.8 Bit allocation of G.729

3.7.1 Decoding of LP filter parameters

The received indices $L0$, $L1$, $L2$ and $L3$ of the LSP quantizer are used to reconstruct the quantized LSP coefficients using the procedure described in the LP analysis[7]. These quantized LSP coefficients are then converted back into their normalized value ωi, hence the value of qi can be found from Eq-16.The initial interpolation procedure of Eq-17 is used to obtain two sets of interpolated LSP coefficients (corresponding to two subframes). For each subframe, the interpolated LSP coefficients are converted to LP filter coefficients ai, which are used for synthesizing the reconstructed speech in the subframe.

3.7.2 Decoding the adaptive-codebook vector

The adaptive codebook index P1 is used to find the integer and fractional parts of the pitch delay T1. int(T1) will be referred to as the integer part and fracT1 as the fractional part[7],

```
$if\ P1 < 197$
    $int(T_1) = (P1 + 2)/3 + 19$
    $frac = P1 - 3\ int(T_1) + 58$
$else$
    $int(T_1) = P1 - 112$
    $frac = 0$
$end$
```

The integer and fractional part of $T2$ are obtained from $P2$ and *tmin*, where *tmin* is derived from $T1$ as follows:

```
$t_{min} = int(T_1) - 5$
$if\ t_{min} < 20\ then\ t_{min} = 20$
$t_{max} = t_{min} + 9$
$if\ t_{max} > 143\ then$
    $t_{max} = 143$
    $t_{min} = t_{max} - 9$
$end$
```

Now *T*2 is decoded using:

$$int(T_2) = (P2 + 2)/3 - 1 + t_{min}$$

$$frac = P2 - 2 - 3\,((P2 + 2)/3 - 1)$$

The adaptive-codebook vector *v*(*n*) is found by interpolating the past excitation *u*(*n*) (at the pitch delay) using Equation (25).

3.7.3 Decoding of the fixed-codebook vector

The received fixed-codebook index *C* is used to extract the positions of the excitation pulses. The pulse signs are obtained from *S*. This is done by reversing the initial process. Once the pulse positions and signs are decoded the fixed codebook vector *c*(*n*) is constructed using Equation (26). If the integer part of the pitch delay *T* is less than the subframe size 40, *c*(*n*) is modified according to Equation (26-1) [7].

3.7.4 Decoding of the adaptive and fixed-codebook gains

The received gain-codebook index gives the adaptive-codebook gain $\hat{g}_p$ and the fixed-codebook gain correction factor $\hat{\gamma}$. This procedure is described in detail in the encoder section. The estimated fixed-codebook gain *gc'* is found using the gain prediction method. The fixed-codebook vector is obtained from the product of the quantized gain correction factor with this predicted gain Equation (31). The adaptive-codebook gain is reconstructed using Equation (32) [7].

3.7.5 Reconstructing the Speech

The excitation *u*(*n*) [see Equation (33)] is input to the LP synthesis filter. The reconstructed speech for the subframe is given by:

$$\hat{s}(n) = u(n) - \sum_{i=1}^{10} \hat{a}_i \hat{s}(n-i) \qquad n = 0,\dots,39$$

(35)

where *âi* are the interpolated LP filter coefficients for the current subframe. The reconstructed speech *s*^(*n*) is then processed by the post processor[7].

3.8 Post processor

The post processor performs the following functions[7]

- Post-filtering.
- High pass filtering.
- Up scaling.

3.8.1 Long term post-filter

It is given by[7]:

$$H_p(z) = \frac{1}{1+\gamma_p g_l}\left(1+\gamma_p g_l z^{-T}\right)$$

(36)

where T is the pitch delay, and g_l is the gain coefficient. Note that g_l is bounded by 1, and it is set to zero if the long-term prediction gain is less than 3 dB. The factor γ_p controls the amount of long-term postfiltering and has the value of $\gamma_p = 0.5$. The long-term delay and gain are computed from the residual signal ř(n) obtained by filtering the speech $\hat{s}(n)$ through $\hat{A}(z/\gamma_n)$, which is the numerator of the short-term postfilter[7].

$$\hat{r}(n) = \hat{s}(n) + \sum_{i=1}^{10} \gamma_n^i \hat{a}_i \hat{s}(n - i) \quad \textbf{(37)}$$

The long-term delay is computed using a two-pass procedure. The first pass selects the best integer $T0$ in the range [$int(T1) - 1$, $int(T1) + 1$], where $int(T1)$ is the integer part of the (transmitted) pitch delay $T1$ in the first subframe. The best integer delay is the one that maximizes the correlation[7].

$$R(k) = \sum_{n=0}^{39} \hat{r}(n)\hat{r}(n - k) \quad \textbf{(38)}$$

The second pass chooses the best fractional delay T with resolution 1/8 around $T0$. This is done by finding the delay with the highest pseudo-normalized correlation,

$$R'(k) = \frac{\sum_{n=0}^{39} \hat{r}(n)\hat{r}_k(n)}{\sqrt{\sum_{n=0}^{39} \hat{r}_k(n)\hat{r}_k(n)}} \quad \textbf{(39)}$$

where *rk^* (n) is the residual signal at delay k. Once the optimal delay T is found, the corresponding correlation $R'(T)$ is normalized with the square-root of the energy of *r^*(n). The squared value of this normalized correlation is used to determine if the long-term postfilter should be disabled. This is done by setting $gl = 0$ if:

$$\frac{R'(T)^2}{\sum_{n=0}^{39} \hat{r}(n)\hat{r}(n)} < 0.5$$

(40)

Otherwise the value of g_l is computed from:

$$g_l = \frac{\sum_{n=0}^{39} \hat{r}(n)\hat{r}_k(n)}{\sum_{n=0}^{39} \hat{r}_k(n)\hat{r}_k(n)} \qquad \text{bounded by } 0 \le g_l \le 1.0$$

(41)

The non-integer delayed signal *rk^* (*n*) is first computed using an interpolation filter of length 33. After the selection of *T*, *rk^* (*n*) is recomputed with a longer interpolation filter of length 129. The new signal replaces the previous one only if the longer filter increases the value of *R′*(*T*).

3.8.2 Short-term postfilter

The short-term postfilter is given by[7]:

$$H_f(z) = \frac{1}{g_f}\frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)} = \frac{1}{g_f}\frac{1 + \sum_{i=1}^{10} \gamma_n^i \hat{a}_i z^{-i}}{1 + \sum_{i=1}^{10} \gamma_d^i \hat{a}_i z^{-i}}$$

(42)

where $\hat{A}(z)$ is the received quantized LP inverse filter (LP analysis is not done at the decoder) and the factors γ_n and γ_d control the amount of short-term postfiltering, and are set to $\gamma_n = 0.55$, and $\gamma_d = 0.7$. The gain term g_f is calculated on the truncated impulse response $h_f(n)$ of the filter $\hat{A}(z/\gamma_n)/\hat{A}(z/\gamma_d)$ and is given by:

$$g_f = \sum_{n=0}^{19} |h_f(n)|$$

(43)

3.8.3 High pass filtering and Upscaling

A high-pass filter with a cut-off frequency of 100 Hz is applied to the reconstructed postfiltered speech $sf'(n)$. The filter is given by:

$$H_{h2}(z) = \frac{0.93980581 - 1.8795834z^{-1} + 0.93980581z^{-2}}{1 - 1.9330735z^{-1} + 0.93589199z^{-2}} \quad \textbf{(44)}$$

The filtered signal is multiplied by a factor 2 to restore the input signal level[7].

CHAPTER NO.4

CHAPTER NO 4

4.0 Problem Areas

During the implementation phase of the project we found that certain blocks of the encoder cannot be implemented the way they have been described in the official recommendation. Either the procedure has not been made clear or it takes so much time to compute that the delays reach beyond the limit. In order to get a solution a comprehensive study was done and in the end we were able to modify some of the major blocks of the encoder. The block of vector quantization of LSF was modified during our project. All the related study and development of the new technique for vector quantization is described below:

4.1Vector Quantization

VQ is nothing more than an approximation. The idea is similar to that of ``rounding-off'' (say to the nearest integer). An example of a 1-dimensional VQ is shown below [9]:

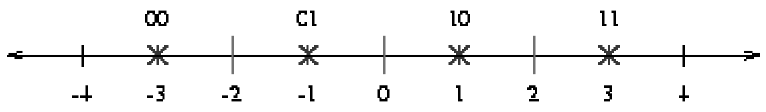


Figure 4.1 1 Dimensional VQ

Here, every number less than -2 are 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 are approximated by +3. Note that the approximate values are uniquely represented by 2 bits. This is a 1-dimensional, 2-bit VQ. It has a rate of 2 bits/dimension [9].
In Scalar Quantization one represents the values by fixed subset of representative values. For examples, if you have 16 bit values and send only 8 most signifcant

bits, you get an approximation of the original data at the expense of precision [9]. In this case the fixed subset is all the 16-bit numbers divisable by 256, i.e 0, 256, 512.

In Vector Quantization you represent not individual values but (usually small) arrays of them. A typical example is a color map: a color picture can be represented by a 2D array of triplets (RGB values). In most pictures those triplets do not cover the whole RGB space but tend to concetrate in certain areas. For example, the picture of a forest will typically have a lot of green. One can select a relatively small subset (typically 256 elements) of representative colors, i.e RGB triplets, and then approximate each triplet by the representative of that small set. In case of 256 one can use 1 byte instead of 3 for each pixel [9].

Having understood the basic concept of VQ it is not difficult now to forsee that VQ can be used in a same manner for large data sets, especially when consecutive points are correlated in some way, which is exactly the case with speech parameters. CELP speech compression algorithms use those subset "codebooks" and use them to quantize exciation vectors for linear prediction -- hence the name CELP which stands for Codebook Excited Linear Prediction.

However it should be kept in mind that Vector Quantization, just like Scalar Quantization, is a lossy compression but the losses can be reduced to such a limit that they can either be ignored or fall within an acceptable threshold level.

4.2 <u>The Process of Vector Quantization</u>

A vector quantizer maps *k-dimensional* vectors in the vector space R^k into a finite set of vectors $Y = \{y_i: i = 1, 2, ..., N\}$. Each vector y_i *is* called a code vector or a *codeword*. and the set of all the codewords is called a *codebook*. Associated with each codeword, y_i, is a nearest neighbor region called *Voronoi* region, and it is defined by [10][12]:

$$V_i = \left\{x \in R^k : \left\|x - y_i\right\| \leq \left\|x - y_j\right\|, for\,all\ j \neq i\right\} \qquad \text{(Eq-4.1)}$$

As an example we take vectors in the two dimensional case without loss of generality. Figure 1 shows some vectors in space. Associated with each cluster of vectors is a representative codeword. Each codeword resides in its own Voronoi region. These regions are separated with imaginary lines in figure 1 for illustration. Given an input vector, the codeword that is chosen to represent it is the one in the same Voronoi region.

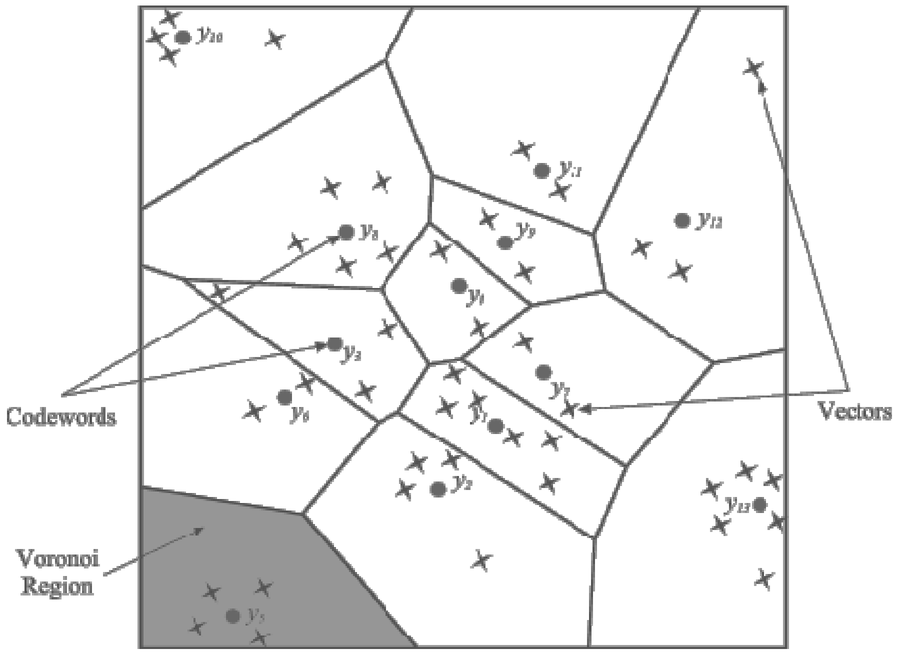


Figure 4.2 : Codewords in 2-dimensional space. Input vectors are marked with an x, codewords are marked with red circles, and the Voronoi regions are separated with boundary lines [10].

The representative codeword is determined to be the closest in Euclidean distance from the input vector.

4.3 How Does VQ Work In Compression?

A vector quantizer is composed of two operations. The first is the encoder, and the second is the decoder. The encoder takes an input vector and outputs the index of the codeword that offers the lowest distortion. In this case the lowest distortion is found by evaluating the Euclidean distance between the input vector and each codeword in the codebook. Once the closest codeword is found, the index of that codeword is sent through a channel (the channel could be a computer storage, communications channel, and so on). [11].

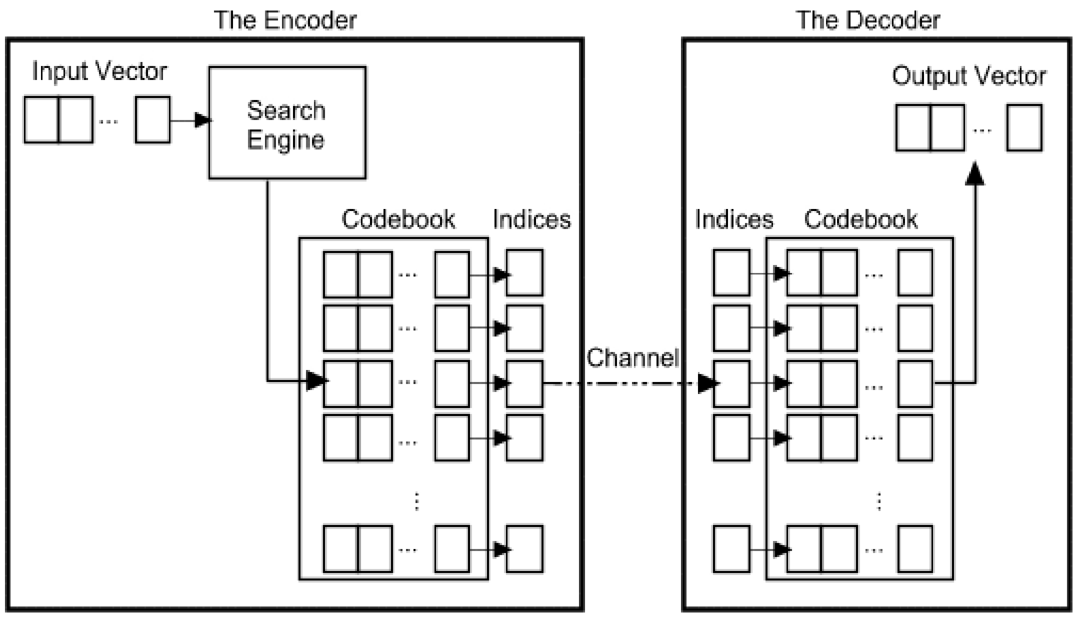


Figure 4.3: The Encoder and decoder in a vector quantizer. Given an input vector, the closest codeword is found and the index of the codeword is sent through the channel. The decoder receives the index of the codeword, and outputs the codeword.

4.4Vector quantization Methods Used In The Implementation of G.729

As we have seen that vector quantization is a very versatile technique and should be implemented according to the nature of the data being quantized. During our attempts to vector quantize the LSF coefficients we had to go through a number of methods for its implementation so that the error could be reduced to a minimum acceptable value. The ITU-T recommendation [7] is not specific about the method to be adopted for the calculation of the quantized LSF coefficients neither does it specify the data for the codebook indices, hence we had the tedious task of not only coming up with the optimal vector quantization methodology but also to form the codebook. However we would like to admit that converting the LSF coefficients into the normalized values [7] helped us a lot because the values were not only ranging between 0-3.1 but they were always in ascending order. The 7bit value of the codebook index implied that the set of vectors to be used for quantization could atmost be 128 and since the LSF coefficients were 10 dimensional hence our codebook vectors also had to be 10 dimensional. Also the normalized vectors had a starting range of 0-1.2 and a terminating range of 2.8-3.1, hence we had the ease of concentrating the starting and ending points of our 128 codebook vectors.

4.4.1 Quantization Using Random Vectors

Our first attempt at quantizing the LSF coefficients was with totally random vectors i.e. the vectors had random slopes and random values as shown in fig 4.4,

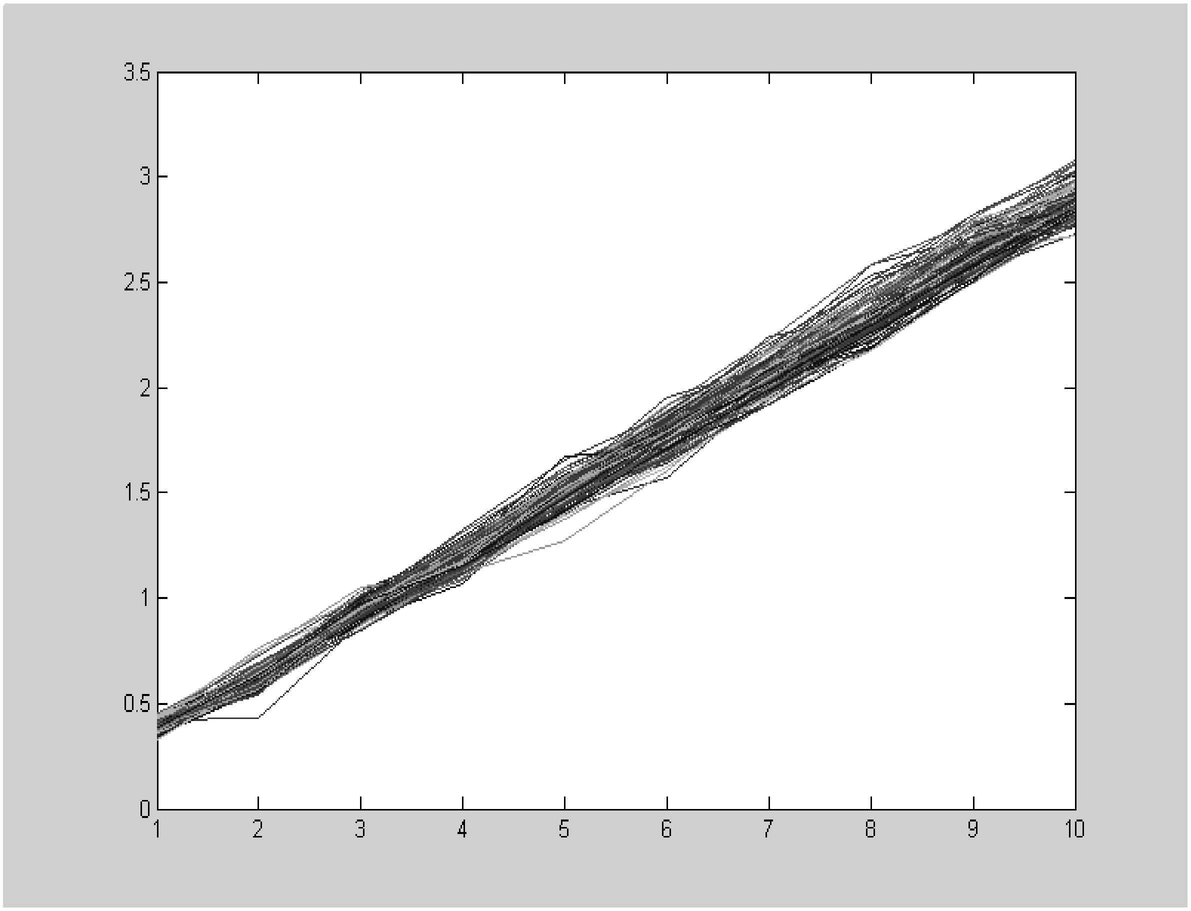


Fig 4.4 Random Vectors

The resultant error from this method had a range of –.3<0<+.3 which was not at all acceptable and resulted in huge differences while reconstructing the speech signal. Hence this method had to be rejected and we started to look for a better way to quantize the LSF coefficients.

4.4.2 Quantization Using Same Slope Vectors

In our second attempt we tried to quantize the LSF coefficients with a set of vectors that had the same upward slope, as shown in fig 4.5,

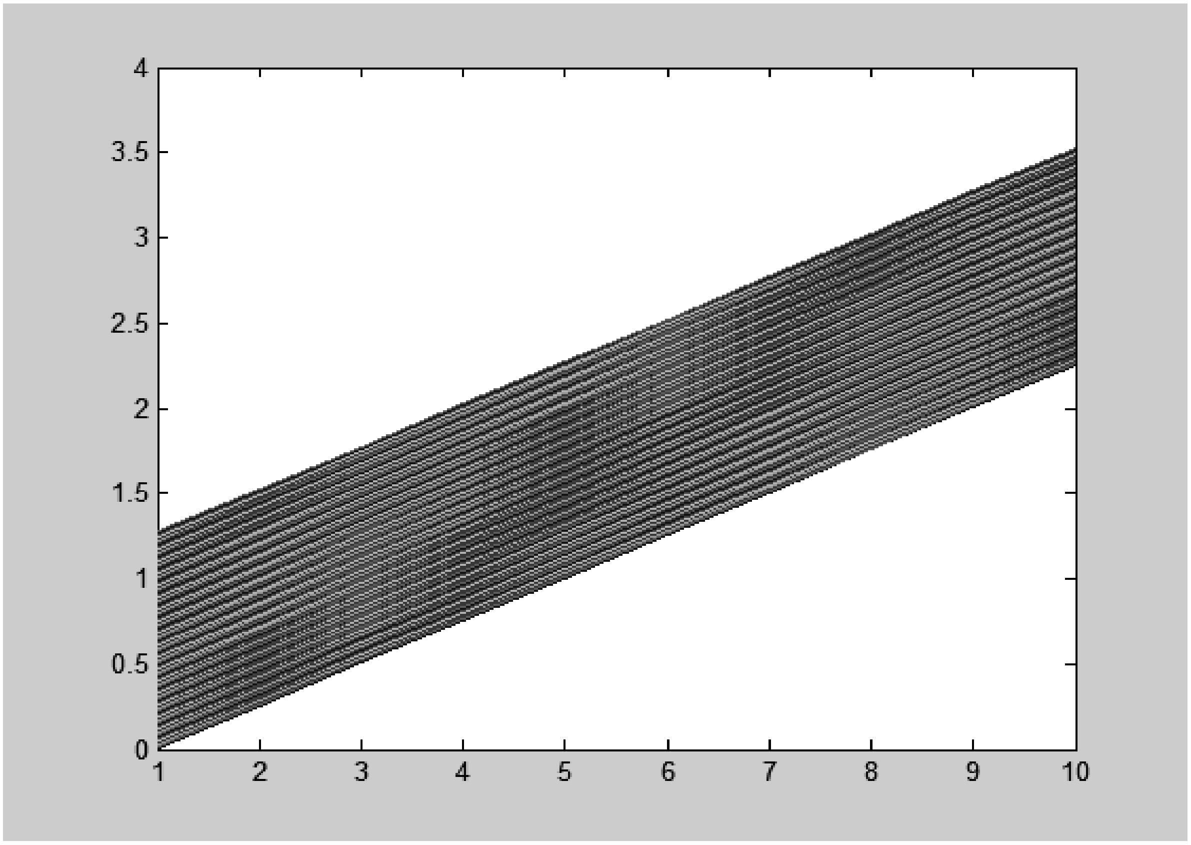


Fig 4.5 Vectors with same slope

Fixing the slope did help in minimizing the error but still not to the acceptable level. While reconstructing the speech it was found that it didn't meet the required level of quality. The error range for this method was -.2<0<+.2

Having failed in our second attempt we were quite frustrated, however we did not lose hope and continued our search to come up with a better codebook.

4.4.3 Quantization Using Different(ascending) Slope Vectors

In the third attempt we used vectors having different slopes with random values however all of them were still in the ascending order. As shown in fig 4.6,

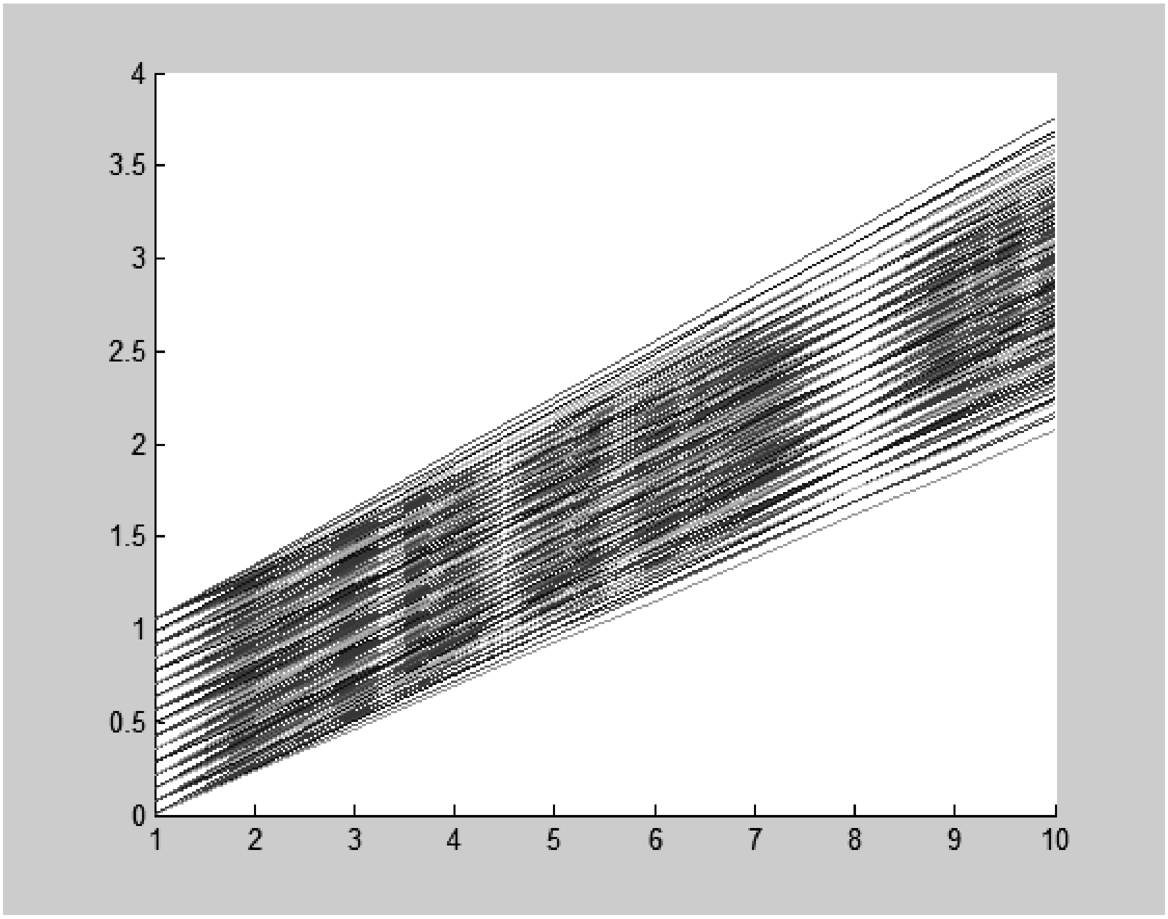


Fig 4.6 Vectors with different slope

Our attempts finally came through and we were able to reduce the error between the original and quantized LSF coefficients to an acceptable value that did not result in huge losses while reconstructing the speech. The range of error for this method is -.1<0<+.1 but mostly the error lies in the region of -.05<0<+.05.

4.4.4 Error Vector Quantization

Another step added to the vector Quantization of LSF coefficients is the quantization of the error vector [7], the procedure for error vector quantization has been carried out for all of the methods mentioned above. The only difference being in the ranges of error for each method.

After the vector has been quantized than a 10 dimensional error vector is formed by comparing the values of the original vector with that of the quantized one, this arror vector is broken down into two equal parts and put in two different codebooks[7] having 32 entries each. The error codebooks are first trained by putting in values of error resulting from the quantization of different sort of input vectors. After the codebooks have been filled than any incoming vector is first quantized and than the error of that vector is also quantized with the two error codebooks. The indices of these error codebooks are sent along with the index number of the quantized vector and at the reception end these error values are added to the quantized vector values so that the difference between the original and reconstructed is minimized.

The range of error vectors that we used is between $-.1<0<+.1$ and the figure 4.7 shows their structure

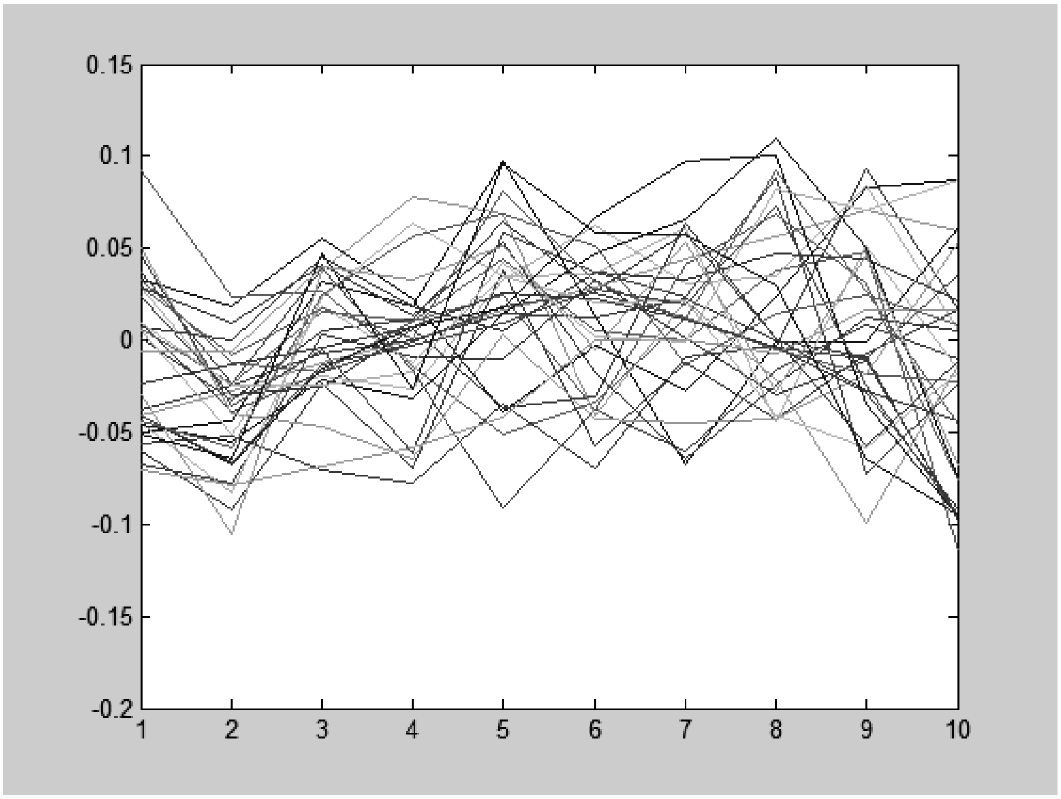


Fig 4.7 Error Vectors

This is the two stage vector quantization that we have used to quantize and transmit the LSF coefficients, after using a number of input sequences we found that the methodolgy being used is accurate and the reconstructed speech is of acceptable quality.

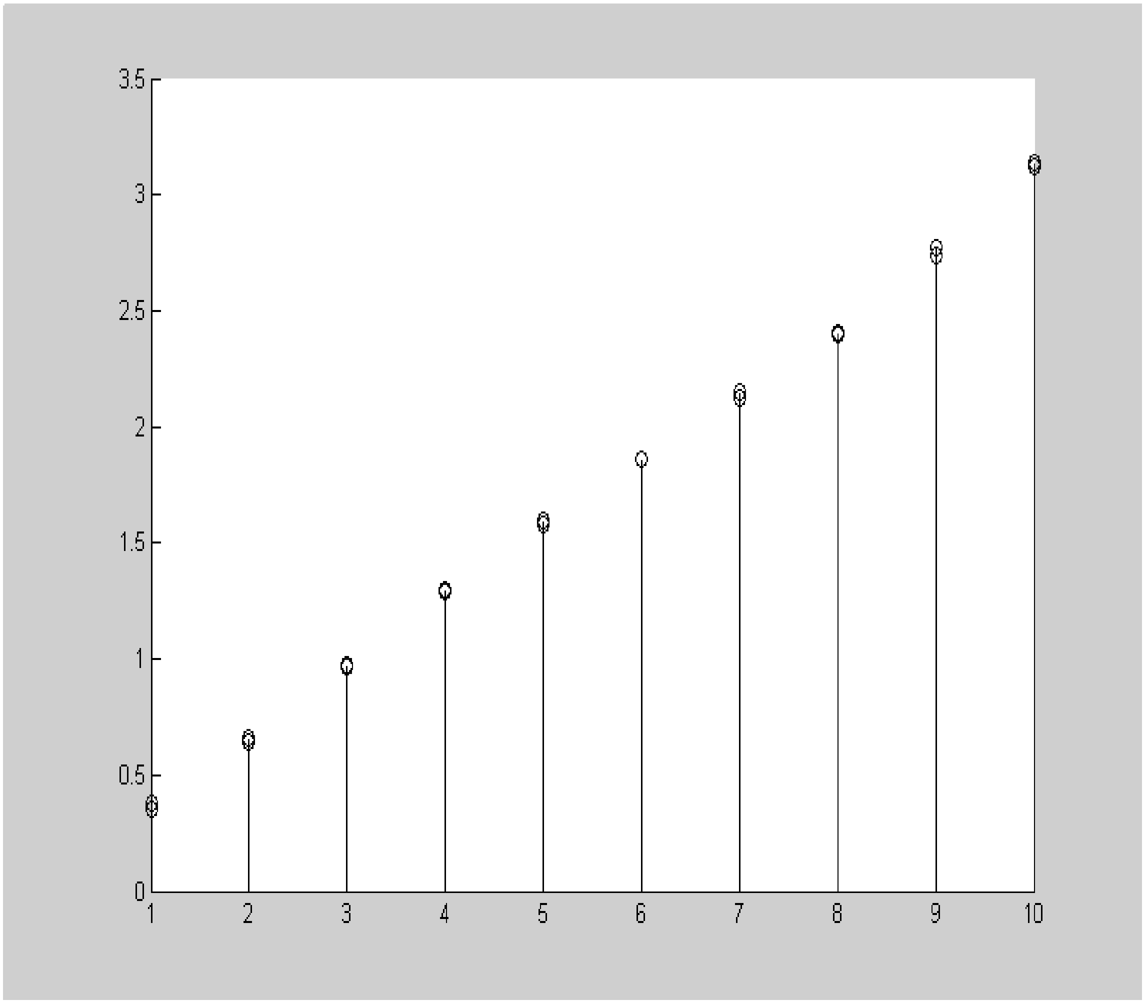


Fig 4.8 Reconstructed Vectors

in the above figure we can see that there is a very little error between the original and the reconstructed LSF vector.

4.5 WINDOWING

Once the vector quantization was solved we moved towards windowing. In this case, at first we tried different combination of the already known windows like Hamming, Hanning, rectangular etc. but at the end we had to actually modify the basic equations of these windows to get the one that could best optimize the issue in G.729.

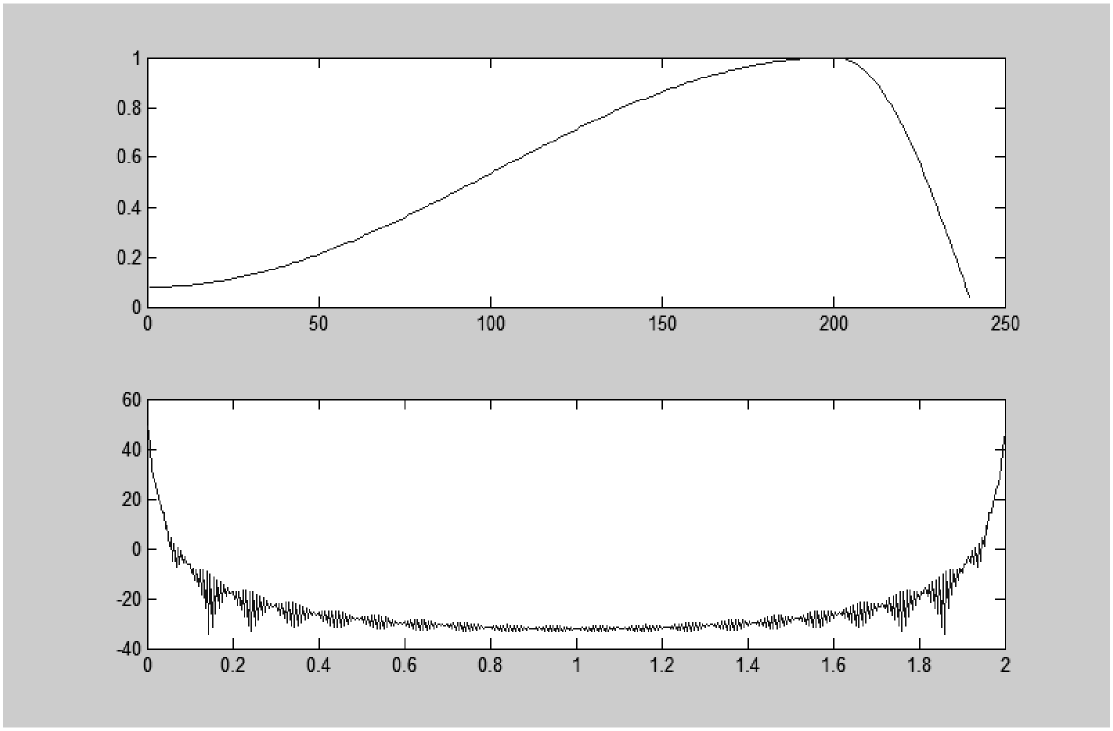


Fig 4.9 Original Window Used

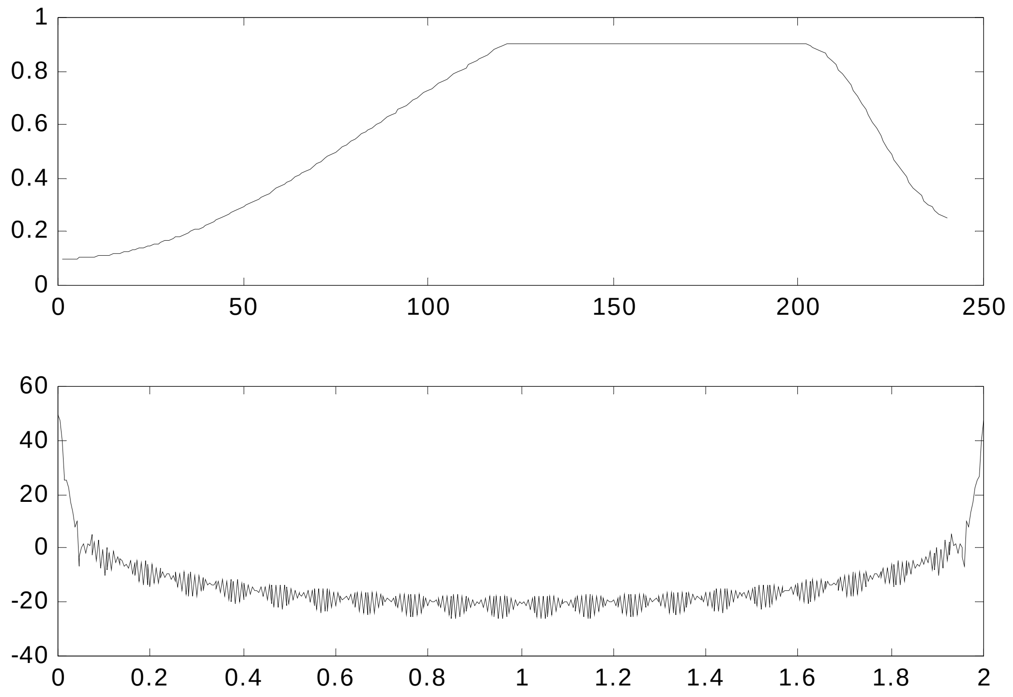


Fig 4.10 New Window Designed

In the above two windows we can see that in the firs case the main lobe in the frequency domain was ever increasing which results in distortions at the present samples which is not desired.

But in the second case the window we have designed has a very narrow main lobe which means that the samples of the present frame will not be distorted during the process of windowing.

4.5 Fixed Code Book Search:

The second problem area was the search procedure of fixed code book. According to the procedure given in the recommendation, we can only find the positions and signs of pulses for the fixed code book vector once we perform at least four nested loop search of length 8 twice in each sub frame. This makes a total of nearly four thousand loops for each sub frame. With this much iterations it is almost impossible to make an efficient coding. Unfortunately this area of search in G.729 has been made extremely secrete and there's no help from matlab or any other source.

4.6 Recommendations:

Most of the research work in the field of vocoder modeling is focused on this issue because search procedure in G.729 has not yet been optimized. This will not only solve the problem of G.729 but also al other standards that make use of the fixed code book. Therefore, we suggest the people who are interested in the field of speech compression to think on this issue and try to optimize the fixed code book procedure.

4.7 CONCLUSION:

The above mentioned were the only major implementations constraints of G.729. We put in our best effort to give an optimal solution to all of them but we managed to solve windowing and vector quantization blocks only. This solution was provided due to full resolve and commitment of the group, had there been a bit more time for us, we might had been able to solve fixed codebook search procedures as well.

REFERENCES:

[1] B.Fette, “High quality secure voice communication” Speech technology journal, pp 40-48, Oct, 1989

[2] Randy Goldberg and Lance Riek. A practical handbook of speech coders(1989), chapter 2: pp 1-28, Chapter 4: pp 1-14

[3] Mark Nelson and Jean loup Gaily. Speech Compression, The Data Compression Book (1995) pp 289-319

[4] Khalid Sayood. Introduction to Data Compression (2000) pp 497-509

[5] Richard Wolfson, Jay Pasachoff. Physics for scientists and Engineers (1995) pp 376-377

[6] McKellen, http://www.omega.co.uk/literature/transactions/volume2/analogsignal.html

[7] Bernard Sklar, Digital Communication Fundamentals and Applications,2nd Edition

[8] Bernard Sklar, Digital Communication Fundamentals and Applications,2nd Edition

[9] B.P.Lathi, Modern Digital and Analog Communication systems, 3rd edition

[10] Oppenhiem, Willinsky, Nawab, Signals and Systems

[11] Douglas, http://www.omega.com

[12] Bernard Sklar, Digital Communication Fundamentals and Applications,2nd Edition