Transcoding between QCELP 13K and G.723.1 CELP Speech Coders

by

Durodami J. Lisk

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 1999
[June 1999]

Author .....
Department of Electrical Engineering and Computer Science
May 19, 1999

Certified by. .....
Gregory W. Wornell
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Transcoding between QCELP 13K and G.723.1 CELP Speech Coders

by

Durodami J. Lisk

Submitted to the
Department of Electrical Engineering and Computer Science

May 19, 1999

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

Transcoding of speech coders occurs frequently in systems today as subsystems use different codecs for speech compression. Transcoding is traditionally done by fully decoding speech from the first coder before sending it to the second coder. The CELP speech coders, QCELP 13K and G.723.1, are evaluated under tranditional transcoding situations. An LSP-based transcoder is then designed to partially replace the decoder of the leading coder and the encoder or the following coder. This transcoder is evaluated and compared with the traditional case. Listening tests for speech with flat response show that in the traditional case, distortion is most likely due to both the digital and non-digital parts of the transcoding process. The LSP-based transcoder does not demonstrate a clear-cut improvement in "digital distortion" for flat speech. Listening tests with modified IRS response show that distortion in transcoding performance is instead likely linked with the weighting filter within the coders. Suggestions for improvements are given.

Thesis Supervisor: Gregory W. Wornell
Title: Associate Professor, Electrical Engineering and Computer Science

Acknowledgments

I wish to express my thanks to my supervisors, Sharath Manjunath and Andy DeJaco at Qualcomm Inc., and Gregory Wornell at MIT, for their guidance and help througout this thesis.

I would like to thank Qualcomm Incorporated and the MIT VI-A Program for sponsoring this thesis and for providing an opportunity to do research in San Diego. I will always cherish this experience.

I would like to thank all my friends who have been very supportive and encouraging throughout this whole process. Special thanks to my friends in Impact Campus Fellowship for their prayers. These were timely and effective.

I would like to thank my parents and family for all they have done for me all these years. My journey up to MIT and through MIT has been a result of the effort they have expended on my behalf. *Thank you very much.*

Lastly, and most importantly, I would like to thank my Father in heaven, and my Lord and Savior, Jesus Christ for the abundant grace that has been available to me througout this experience. *I love You.*

This thesis is dedicated to you all.

Contents

List of Figures

[1]Here we assume that the Nyquist Frequency is 4000 Hz

List of Tables

Chapter 1

Introduction

Communications technology today is designed to carry either speech or data. Technology differs depending on which is carried. This Masters of Engineering thesis will exclusively address speech. Even within the speech framework, there are a vast amount of standards and technology available to perform very similar tasks. Again, we will be dealing with only a very small subset of these technologies. Specifically, this thesis will address two speech coders, QCELP 13K (hereafter known as QCELP) and G.723.1. These speech coders are part of a general class of speech coders known as CELP (Code Excited Linear Prediction) coders. We will investigate situations where we have these coders in transcoding[1] situations. Tandem[2] situations using the same coder are also considered to help us better understand the transcoding situation. Speech quality through these two-stage coding situations will be analyzed and designs for improving speech quality proposed and evaluated.

Technology for carrying speech through wire and wireless media has been around for quite a while. However, the increase in the demand for these media to carry speech has necessitated the improvement in the technology available. Coders that can deliver higher quality speech with lower bit rate (in case of digital systems) are very desirable. One such algorithm that has been very successful at coding speech at low bitrates (while retaining good quality) is the class of CELP coders. These were first proposed about 15 years ago in a paper by Atal and Schroeder[1, 21]. We will concern ourselves entirely with two coders,

[1]One coder followed by the other coder.

[2]Usually this means that both coders are the same. Throughout this thesis, the word "tandem(ing)" is used loosely. The context will determine whether we are talking about the using the same coder or different coders.

QCELP and G.723.1. QCELP, Qualcomm's 13K coder[23], will be paid the most attention in this thesis. This coder is mainly used in cellular telephony (especially in Qualcomm phones) and is now the TIA[3] standard IS-733, High Rate Speech Service Option 17 for Wideband Spread Spectrum Communications Systems. This coder provides high quality speech as a result of the effective algorithm and the high full-rate bit rate. G.723.1 is an ITU[4] standard used for the internet. It supports lower bit rates than QCELP and produces lower quality speech.

As mentioned earlier, there are a lot of algorithms available and presently in use. It is therefore not surprising that situations arise when communication between two systems, using different algorithms, is necessary. One such case arises when a user needs to transmit speech from a cellular phone to an end-user on the internet. The other direction, though less likely, might also arise. In the former case, one must go from say, a QCELP coder to a G.723.1 coder and vice versa for the latter. This transcoding situation creates a possible need for efficient algorithms, if possible. The standard practice in dealing with transcoding situations is to fully decode the speech from the first coder before passing it to the following coder. However, this process causes undesired degradation in the speech quality.

This thesis addresses this transcoding problem. In particular, we seek to develop a transcoding algorithm that would allow for better quality speech without necessarily increasing complexity and delay by much (if any) and without making any significant alterations to the encoder of the leading coder and the decoder of the coder in tandem. In other words, this transcoder "black box" could take in packets from the leading coder and generate packets for the following coder. A study of the detioration of speech through the CELP coders in necessary to help facilitate this end. To aid this venture, the tandem case with the same coders is also considered in certain cases.

The rest of this thesis is organized as follows: First, we give some background information on pertinent research and results. Then we describe CELP coders, with specific attention given to QCELP and G.723.1. We then describe the transcoding problem in more depth and come up with a transcoder. This transcoder is evaluated subjectively and the results analyzed. Further efforts are then made to improve on these results. Finally, we conclude by discussing the implications of the results, conclusions and possible future work.

[3]Telecommunications Industry Association
[4]International Telecommunications Union

1.1 Background Information

We first mention some research that has been done in transcoding and then talk about some results from tests done on CELP coders.

Transcoding and Tandeming have been investigated to quite some depth for speech coders; especially for non-CELP coders. Not only has tandeming and transcoding been studied in-depth for Differential Encoding Systems like ADPCM[5]/PCM[6]/CVSD[7] [20, 16, 9, 7] papers have been proposed with algorithms that completely reduce further degradation due to tandeming/transcoding. One such paper, by Nishitani[16], studies and develops models for quantization in ADPCM/PCM coders and proposes an algorithm for distortion free speech coding through tandem connections. This is clearly the optimal case and the "Holy Grail" of tandem coding research. However this is not necessary always attainable. Quantization in the ADPCM/PCM case is a direct quantization of speech. Tandem connections in Differential Encoding Systems are, therefore, really a case of Cascaded Quantizers[7]. CELP coders, on the other hand, do not quantize speech directly, but rather quantize parameters that are capable of producing the speech, as explained in Chapter 2. Deducing an exact mathematical model for the effect of quantization is complex and probably intractable given the complex operations involved in CELP coders today. Thus, in the case of improvements on tanscoding/tandem connections between CELP coders, the "playing field" includes anything from a minor improvement to almost perfect tandem connections. The literature does not seem to have much on direct study on improving tandem/transcoding connections in CELP coders. There are some evaluative studies on tandem properties of different CELP coder algorithms. A celp coder in tandem with a non-CELP coder has been studied for some arrangements: for example, CELP/CVSD, CVSD/CELP, DM/CELP[6, 15, 5, 4].

Celp coders have also been studied in tandem conditions with other celp coders (usually, the same coder). One such study was performed by AT&T Bell Laboratories[18]. This study concluded the following with regards to tandem processing of several coders, mostly CELP coders - including QCELP under different modes:

(i) There is no statistically significant difference between performance in IRS[8] flat re-

[5]Adaptive Differential Pulse Code Modulation
[6]Pulse Code Modulation
[7]Continous Variable Slope Delta Modulation
[8]This is different from modIRS as we will see later.

sponse filter conditions.

(ii) There is statistically significant difference between tandeming and non-tandeming conditions for the QCELP coder.

(iii) This degradation is worse for lower rate coders.

Other tests[9] have been done that corroborate these results for the QCELP coder and show that there is much worse degradation in lower rate coders (e.g. IS-96A).

[9]We do not give specific details because of proprietary restrictions.

Chapter 2

Overview

Since this thesis deals entirely with CELP coders, it is essential that the reader have an understanding of these coders and in particular, QCELP and G.723.1. The first section gives a very general overview of CELP coders and some of the key algorithms involved. The latter sections deal with the specific coders, QCELP and G.723.1.

2.1 CELP Coders

As the name CELP suggests, this algorithm consists of synthesizing speech by exciting a filter, derived by linear prediction, by a set of values (from a codebook). This simple but powerful idea is what gives CELP its appeal. Speech can be represented by a model whose parameters are sufficient for reconstruction. The bit rate of the coder depends on the number of parameters sent from encoder to decoder. CELP allows for lower bit rates with good speech qualities, especially in narrowband cases.

2.1.1 Linear Prediction

Linear prediction, a concept used in countless applications, is particularly useful in CELP and especially in the case of speech, because it allows one to closely approximate the speech spectra with a very small set of values (called Linear Prediction Coefficients - LPCs). These LPCs are then used to create the synthesis filter, which when excited by the error signal (or some approximation to it), reconstructs the speech. The equation below (Equation 2.1)

summarizes linear prediction.

$$s[n] = \sum_{k=1}^{N} a_k s[n-k] + e_s[n] \tag{2.1}$$

where $s[n]$ is the signal (speech in our case), the a_k's are the LPCs and $e_s[n]$ is the error signal. Knowing $e_s[n]$ and a_k is sufficient to reconstruct, in principle, $s[n]$ exactly (except maybe at some points[1]). N is the order used. Larger N gives better approximation. A Synthesis filter for N=10 is shown below (Figure 2-1). The a_k's are determined so as to

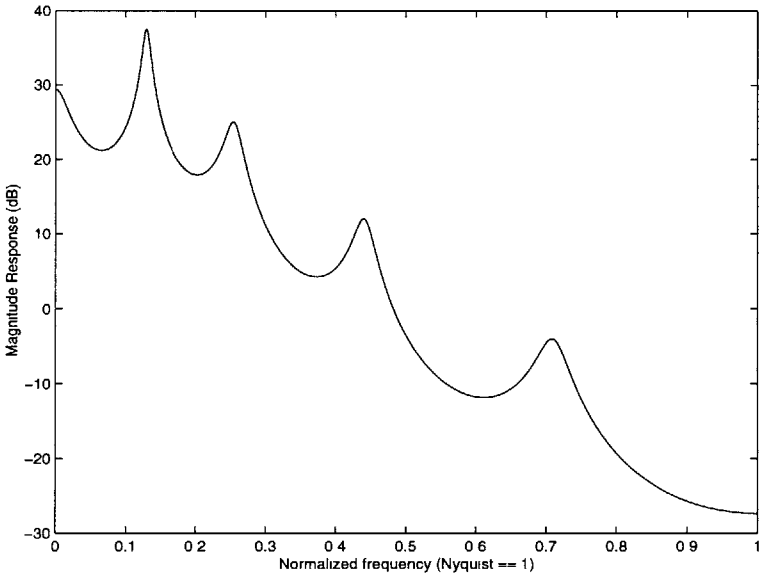

Figure 2-1: LP synthesis filter for 160 samples of a 8Khz sampled speech waveform

minimize $e_s[n]$ in the squared-error sense. The Levinson-Durbin Algorithm[19] is employed to determine the LPCs. Equation 2.1 refers to an all-pole (auto-regressive) model. A general pole-zero model can be used but isn't in general because auto-regressive models are good enough for speech. These all-pole models match speech very well at the spectral peaks but do not do that well at the valleys.

2.1.2 Code Excitation

The other half of Code Excited Linear Prediction (CELP), "Code Excited", completes the overall picture. The excitation signal used is an approximation to $e_s[n]$, the error signal.

[1]Places where the spectrum of $e_s[n]$ is zero.

This approximation is derived from a codebook of finite index length, M, with 2^M values. The determination of which of these values to use depends on the particular CELP coder.

2.1.3 Quantization

The use of linear prediction and code excitation is central to CELP coders. However, the implementation details differ. One such difference in implementation is quantization. There are many algorithms in the literature for quantization of parameters. A general class of quantization widely used is Vector Quantization. Here, instead of quantizing each parameter separately, one can take advantage of any correlation between these parameters by jointly quantizing a couple of them at a time. The LPCs however are not quantized directly. This is because it is not possible to easily guarantee the stability of LPCs across quantization. Thus, the LPCs are converted to an intermediary "stage" before quantization. Again, there are several options for this intermediary "stage": some include, arcsine of reflection coefficients, log area ratios and line spectral coefficients. Line spectral pairs (LSPs) are the most common since they are more efficient bitwise [17, 19]. LSPs also have a very nice property that stability is guaranteed if the LSPs are ordered.

Other aspects of CELP coders will be evident as we describe, in more detail, the two CELP coders, QCELP and G.723.1.

2.2 QCELP and G.723.1

These two coders, though similar in many ways, have some differences. We briefly discuss some of the main similarities and differences and some of the assumptions and simplifications we make in this thesis. We then give a more detailed description of the encoder and decoder, with particular emphasis on QCELP (discussing differences with G.723.1 where necessary). Some of the differences and similarities are shown in Table 2.1. Others will come up in the discussions below.

In our discussion of these coders, we make certain assumptions and simplifications. We only consider the 6.3kbps rate in G.723.1 (though the results also apply to the slower rate). Voice Activity Detection and Comfort Noise Generation are disabled in the G.723.1 coder. For QCELP, we, for the most part, ignore the fact that there are different rates (most of the rates use the same algorithms, so this is not a problem). The rate reduction option,

Table 2.1: Comparisons between Coders

QCELP	G.723.1
8 kHz sampled, 14 bit linear PCM speech	8 kHz sampled 16 bit linear PCM speech
10th order linear prediction	10th order linear prediction
Frame[2] size of 160 samples (20 ms)	Frame size of 240 samples (30 ms)
LPC window size[3] = 160 samples	LPC window size = 180 samples
Variable rate coder with rate reduction[4] capabilities. Rates are: Rate 1(13.3kbps), Rate 1/2(6.2kbps), Rate 1/4(2.7kbps) Rate 1/8(1kbps) and blank (0kbps)	Fixed rate coder with two rates (5.3kbps and 6.3kbps[5]).
Cyclic codebook	Algebraic codebook (5.3kbps); Maximum Likelihood Multipulse (6.3kbps).

which optimizes the initial rate decisions (these initial decisions are based on whether the speech is voiced[6] or unvoiced[7] and other energy and band characteristics), is disabled. Also, in certain cases, we do not give a detailed description of parts of the algorithm - which are usually complex - as this is not necessary. We now describe the encoder and decoder for QCELP and G.723.1.

2.3 Encoder

The encoder (block diagram shown in Figure 2-2) takes in 14 bit (16 in G.723.1), linear PCM highpass-filtered speech, previously sampled at 8 kHz and generates packets of varying size depending on rate. Frame size for QCELP 13K is 20ms or 160 samples (30ms, 240 samples for G.723.1). Each frame is subdivided into 4 subframes. Most of processing is done at the subframe level. The highpass[8] filtered speech is first processed to produce the LPCs which

[2]Speech is processed once per frame.

[3]This is the actual number of speech samples used to calculate the LPC using the Levinson-Durbin algorithm. It is not necessarily equal to the frame size nor centered within the frame.

[4]Rate reduction goes beyond the first stage Rate Determination Algorithm and tries to identify the most efficient encoding rate based on input speech statistics.

[5]It is possible to switch between rates at frame boundaries.

[6]Associated with the vocal tract. Excitation in this case is close to an impulse train.

[7]Associated with the glottis. Excitation is random noise. Unvoiced speech are sent at lower rates than voiced speech.

[8]A highpass filter removes circuit noise and DC. G.723.1 uses a first order filter that has a steeper cut-off than the second order filter QCELP uses.

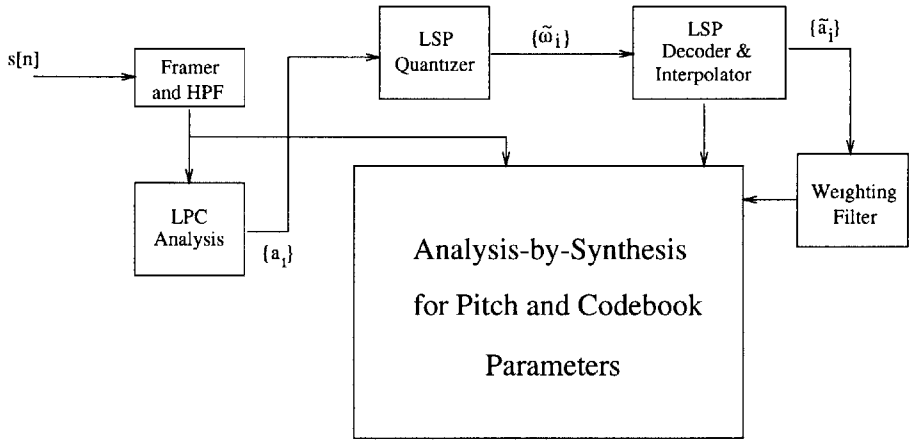

Figure 2-2: Simplified Encoder for QCELP

will be used to calculate other important parameters.

2.3.1 Linear Predictive Coefficients

A 160 sample Hamming window, centered around the middle of the fourth subframe is used (180 samples in G.723.1). The first 17 values of the autocorrelation, R_k's, are calculated. The first 11 are used to calculate LPCs and the rest are used for rate determination. The 11 autocorrelation values are fed through Durbin's recursion algorithm[19] (Appendix C) to produce the 10 LPCs. These LPC coefficient form the basis of the "prediction error filter" transfer function:

$$A(z) = 1 - a_1 z^{-1} - ... - a_{10} z^{-10} \tag{2.2}$$

2.3.2 LPCs to LSPs

The LPCs are then converted to Line Spectrum Pairs (LSPs). In the case of G.723.1, a small bandwidth expansion (7.5Hz) is performed. This is done to avoid problems caused by very small bandwidths in formant peaks that could occur in some LPC frames[2]. Conversion to LSP is done by defining new transfer functions $P(z)$ and $Q(z)$ as:

$$P(z) = A(z) + z^{-11} A(z^{-1}) = 1 + p_1 z^{-1} + ... + p_5 z^{-5} + p_5 z^{-6} + ... + p_1 z^{-10} + z^{-11} \tag{2.3}$$

$$Q(z) = A(z) - z^{-11}A(z^{-1}) = 1 + q_1 z^{-1} + ... + q_5 z^{-5} - q_5 z^{-6} - ... - q_1 z^{-10} - z^{-11} \tag{2.4}$$

where

$$\begin{aligned} p_i &= -a_i - a_{11-i},\ 1 \leq i \leq 5 \\ q_i &= -a_i + a_{11-i},\ 1 \leq i \leq 5 \end{aligned}$$

The LSPs frequencies are the ten zeros ($\omega_1 ... \omega_{10}$) which exist between $\omega = 0$ and $\omega = 1.0$ in the following equations:

$$P'(\omega) = \cos(5(\pi\omega)) + p'_1 \cos((4(\pi\omega)) + ... + p'_4 \cos(\pi\omega) + \frac{p'_5}{2} \tag{2.5}$$

$$Q'(\omega) = \cos(5(\pi\omega)) + q'_1 \cos((4(\pi\omega)) + ... + q'_4 \cos(\pi\omega) + \frac{q'_5}{2} \tag{2.6}$$

Since the formant synthesis (LPC) filter is stable, the roots of the two functions in (2.5) and (2.6), when rearranged in increasing order, alternate in the range $\omega \in (0, 1)$[22].

2.3.3 Converting the LSPs to Transmission Codes for QCELP

The 10 LSP frequencies are quantized into 32[9] bits using a vector quantizer (VQ). Five 2-dimensional vectors are used for this purpose.

Converting to Sensitivities

LSP frequencies have different sensitivities to quantization. The model described below to calculate these sensitivities is computationally efficient and was developed by Gardner[8]). These sensitivities are used in the quantization process to weight the quantization error in each LSP frequency appropriately.

First, the set of vectors of length 10, J_i, where i is the index of the LSP frequency, are obtained by long division operations on P and Q given in Equations (2.3) and (2.4). For the

[9]All Rates except Rate 1/8 use this mechanism. Rate 1/8 uses a different mechanism which includes a 1-bit quantizer and a predictor.

LSP frequencies with odd index, ω_1, ω_3, etc (zeros of $P'(\omega)$), the long division is performed as

$$\frac{1 + p_1 z^{-1} p_2 z^{-2} + ... + p_2 z^{-9} + p_1 z^{-10} + z^{-11}}{1 - 2\cos(\pi\omega_i) z^{-1} + z^{-2}} = J_i(1) + J_i(2) z^{-1} + ... + J_i(10) z^{-9} \tag{2.7}$$

while for the rest (those with even indices), it is calculated as

$$\frac{1 + q_1 z^{-1} + q_2 z^{-2} + ... + q_2 z^{-9} + q_1 z^{-10} + z^{-11}}{1 - 2\cos(\pi\omega_i) z^{-1} + z^{-2}} = J_i(1) + J_i(2) z^{-1} + ... + J_i(10) z^{-9} \tag{2.8}$$

Autocorrelations of vectors J_i are computed as:

$$R_{J_i}(n) = \sum_{k=1}^{10-n} J_i(k) J_i(k+n), \; 0 \leq n < 10, 0 \leq i \leq 10 \tag{2.9}$$

Finally, the sensitivity weights for the LSP frequencies are computed by cross correlating the R_{J_i} vectors with the autocorrelation vector computed from the speech, $R(k)$ (mentioned in Section 2.3.1). The final sensitivity weights are given by:

$$SW_i = \sin^2(\pi\omega_i) \left[R(0) R_{J_i}(0) + 2.0 \sum_{k=1}^{9} R(k) R_{J_i}(k) \right], \; 1 \leq i \leq 10 \tag{2.10}$$

These weights, SW_i are used to compute the weighted squared error metrics needed to search the LSP VQ codebooks as briefly described in the next section.

Vector Quantization of LSP Frequencies

As mentioned earlier, the LSP vector is divided into 5 2-dimensional subvectors, each quantized by a VQ, whose codebook has varying sizes (6,7,7,6,6 bits respectively totalling 32 bits).

Differential vectors are used in the codebooks; i.e. the VQ codebooks contain possible values for quantized differences in the LSP frequencies, given by $\Delta\omega_j = \omega_j - \omega_{j-1}$. The ith

VQ codebook contains possible quantized values for $\Delta\omega_{2i-1}$ and $\Delta\omega_{2i}$. The five subvectors are quantized sequentially in the following manner.

The best vector for the ith codebook is determined by minimizing the sensitivity weighted error between the quantized ($\tilde{\omega}_i$) and unquantized (ω_i) LSP frequencies. This weighted error is computed as

$$\begin{aligned} error &= SW_{2i-1}\left(\omega_{2i-1}-\tilde{\omega}_{2i-1}\right)^2 + SW_{2i}\left(\omega_{2i}-\tilde{\omega}_{2i}\right)^2 \\ &= SW_{2i-1}\left(\omega_{2i-1}-\left(\tilde{\omega}_{2i-2}+\Delta\tilde{\omega}_{2i-1}\right)\right)^2 + SW_{2i}\left(\omega_{2i}-\left(\tilde{\omega}_{2i-2}+\Delta\tilde{\omega}_{2i-1}+\Delta\tilde{\omega}_{2i}\right)\right)^2 \\ &= SW_{2i-1}\left(\omega_{2i-1}-\left(\tilde{\omega}_{2i-2}+L_k\left(i,1\right)\right)\right)^2 + SW_{2i}\left(\omega_{2i}-\left(\tilde{\omega}_{2i-2}+L_k\left(i,1\right)+L_k\left(i,2\right)\right)\right)^2 \end{aligned} \tag{2.11}$$

where $\Delta\tilde{\omega}_1 = \tilde{\omega}_1$ and $L_k\left(i,j\right)$ is the jth element in the kth subvector of the ith codebook. The index of the codevector, k^*, which results in the minimum error for each subvector is selected and sent as the transmission code for that subvector.

2.3.4 Converting the LSPs to Transmission Codes for G.723.1

The G.723.1 coder does not use sensitivity calculations to quantize LSPs. Instead it uses the algorithm based on linear prediction that is briefly described below.

First, the long term DC component, $\boldsymbol{\omega}_{DC}$, is removed from the LSP vector $\boldsymbol{\omega}$ to get the new vector $\boldsymbol{\omega}'$. A first order predictor, $b = 12/32$, is then applied to the previously decoded LSP vector $\tilde{\boldsymbol{\omega}}_{n-1}$ to obtain the DC removed predicted LSP vector, $\bar{\boldsymbol{\omega}}_n$, and the residual LSP vector, $\mathbf{e}_n$ at nth frame.

$$\boldsymbol{\omega}'_n = \left[\omega'_{1,n}\ \omega'_{2,n}\ \ldots\ \omega'_{10,n}\right]^T \tag{2.12}$$

$$\bar{\boldsymbol{\omega}}'_n = \left[\bar{\omega}'_{1,n}\ \bar{\omega}'_{2,n}\ \ldots\ \bar{\omega}'_{10,n}\right]^T \tag{2.13}$$

$$\bar{\boldsymbol{\omega}}'_n = b\left[\tilde{\boldsymbol{\omega}}_{n-1}-\boldsymbol{\omega}_{DC}\right]^T \tag{2.14}$$

$$\mathbf{e}_n = \boldsymbol{\omega}'_n - \bar{\boldsymbol{\omega}}'_n \tag{2.15}$$

The unquantized LSP vector, $\boldsymbol{\omega}_n$, the quantized LSP vector, $\tilde{\boldsymbol{\omega}}_n$, the residual LSP

vector, $\mathbf{e}_n$ are divided into 3 subvectors with dimension 3, 3 and 4 respectively. Each mth subvector is vector quantized using an 8-bit codebook (256 entries). The index l of the appropriate subvector codebook entry that minimizes the error criterion $E_{l,m}$ is chosen for that subvector.

$$\boldsymbol{\omega}_m = [\omega_{1+3m}\ \omega_{2+3m}\ \ldots\ \omega_{K_m+3m}]^T,\ K_m = \begin{cases} 3, & m=0 \\ 3, & m=1 \\ 4, & m=2 \end{cases} \tag{2.16}$$

$$\tilde{\boldsymbol{\omega}}_{l,m} = [\tilde{\omega}_{1,l,m}\ \tilde{\omega}_{2,l,m}\ \ldots\ \tilde{\omega}_{K_m,l,m}]^T,\quad 0 \le m \le 2, 1 \le l \le 256 \tag{2.17}$$

$$\boldsymbol{\omega} = \boldsymbol{\omega}' + \boldsymbol{\omega}_{DC} \tag{2.18}$$

$$\tilde{\boldsymbol{\omega}}_{l,m} = \bar{\boldsymbol{\omega}}'_m + \boldsymbol{\omega}_{DC_m} + \mathbf{e}_{l,m}, \tag{2.19}$$

$$E_{l,m} = (\boldsymbol{\omega}_m - \tilde{\boldsymbol{\omega}}_{l,m})^T\, P_m\, (\boldsymbol{\omega}_m - \tilde{\boldsymbol{\omega}}_{l,m}) \tag{2.20}$$

where $e_{l,m}$ is the lth entry of the mth split residual LSP codebook and P_n is the diagonal weighting matrix, determined from the unquantized LSP coefficients:

$$\begin{aligned} P_{j,j} &= \frac{1}{min\{\omega_j - \omega_{j-1}, \omega_{j+1} - \omega_j\}},\ 2 \le j \le 9 \\ P_{1,1} &= \frac{1}{\omega_2 - \omega_1} \\ P_{10,10} &= \frac{1}{\omega_{10} - \omega_9} \end{aligned} \tag{2.21}$$

This chosen index is sent as the transmission code for the subvector.

2.3.5 Decoding LSPs and Converting to LPCs

The LSP frequencies calculated are used to determine the parameters needed to excite the LPC filter to reproduce the speech (described later in Section 2.3.6). However, for the encoder to resemble the decoder closely, it must use the quantized LSPs (since this is what the decoder sees) for it's synthesis filter. As a result, the encoder must have some of the

functionalities of the decoder in it (henceforth referred to as the encoder's decoder). One of these is the algorithm to decode LSPs back to LPCs. This algorithm is described below.

The LSP transmission codes (indexes of codebooks) are converted back to quantized LSP frequencies. For QCELP, the quantized LSPs are:

$$\begin{aligned} \tilde{\omega}_{2i-1} &= \tilde{\omega}_{2i-2} + \Delta\tilde{\omega}_{2i-2} = \tilde{\omega}_{2i-2} + L_{k^*}(i,1) \\ \tilde{\omega}_{2i} &= \tilde{\omega}_{2i-1} + \Delta\tilde{\omega}_{2i} = \tilde{\omega}_{2i-2} + + L_{k^*}(i,2) \end{aligned} \tag{2.22}$$

In case of G.723.1, the three subvectors are decoded to form $\tilde{\mathbf{e}}_n$. This is then added to the predicted vector, $\bar{\omega}'_n$, and the DC vector, ω_{DC}, to form the decoded LSP vector, $\tilde{\omega}_n$. A stability check is also performed to ensure that LSPs are ordered ($\tilde{\omega}_1 < \tilde{\omega}_2 < ... < \tilde{\omega}_{10}$) and separated by a minimum of 31.25 Hz. A simple averaging algorithm is used to modify LSP frequencies that violate this check.

These LSP frequencies are then linearly interpolated[10] to generate four LSP frequencies; one for each subframe[11]. These four LSP frequencies are then converted to LPCs, $\tilde{\mathbf{a}}$, by doing the "inverse" of equations (2.3) to (2.6). A small bandwidth expansion of 15 Hz is performed at this stage[12] for the QCELP coder. These decoded LPCs (referred to as qLPCs) form the basis of the synthesis filter given by:

$$\frac{1}{\tilde{A}(z)} = \frac{1}{1 - \tilde{a}_1 z^{-1} - ... - \tilde{a}_{10} z^{-10}} \tag{2.23}$$

2.3.6 Analysis-by-Synthesis Loop

The excitation parameters necessary to reproduce the speech at the decoder are determined by synthesizing speech for different possible excitations and choosing the best by comparing this synthesized speech with the input speech, using some weighted minimum squared error criterion (sometimes called the MPSE for Minimum Perceptual Squared Error). Again, G.723.1 and QCELP differ in the implementation of this loop. As usual, we describe the algorithm for QCELP and mention the pertinent deviations in G.723.1 wherever necessary. We also give a brief overall picture of the G.723.1 encoder at the end. First, the pitch

[10]This interpolation across time frames generally results in improved quality of synthetic speech without any additional information for transmission[2]

[11]These LSP frequencies produce stable filters[2].

[12]This is done after LSP quantization instead of before as in G.723.1.

parameters are determined (as shown in Figure 2-3). Using the periodicity of speech to

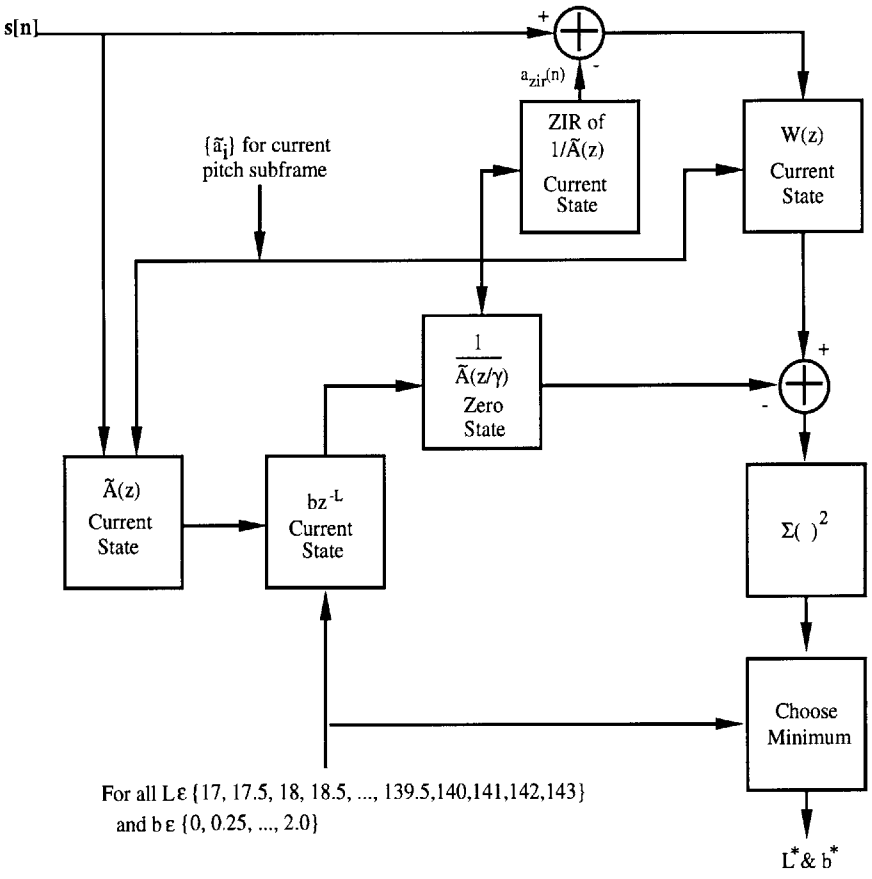

Figure 2-3: Analysis-by-Synthesis Procedure for the Pitch Parameter Search[23]

help reduce the dynamic range of the residual enhances the quality of the speech. Before we briefly describe the algorithm to generate pitch parameters, we examine the weighting filter (shown as $W(z)$ in Figure 2-3).

Weighting Filter

Weighting filters are used to help greatly reduce the perceptible quantization noise. Without these filters, this noise is evident. Reducing noise (every subframe) is done by shaping the noise spectrum. Quantization noise can be "hidden" under the formant peaks without much discernible effect. The weighting filter thus achieves its goal by "shifting" noise from the troughs to the peaks of the speech spectrum: attenuate frequencies where error is perceptually more important and amplify others[1].

A speech-dependent weighting filter is used. The general form is:

$$W(z) = \frac{\bar{A}(z/\gamma_1)}{\bar{A}(z/\gamma_2)} \tag{2.24}$$

For G.723.1, $\gamma_1 = 0.9$, $\gamma_2 = 0.5$ and the actual LPCs values are used. QCELP, on the other hand, uses $\gamma_1 = 1$, $\gamma_2 = 0.78$ and $\bar{A}(\cdot) = \tilde{A}(\cdot)$: the qLPCs are used here. QCELP makes a trade-off by choosing $\gamma = 1$[13] . In doing so, it greatly reduces complexity by collapsing the product of the synthesis filter, $1/\tilde{A}(z)$ and the weighting filter, $\tilde{A}(z)/\tilde{A}(z/\gamma_2)$, into one all-pole filter, $1/\tilde{A}(z/\gamma_2)$. G.723.1 cannot do this without taking a big hit in quality[14]. The figure below (Figure 2-4) shows weighting filters for both coders for a frame of voiced speech.

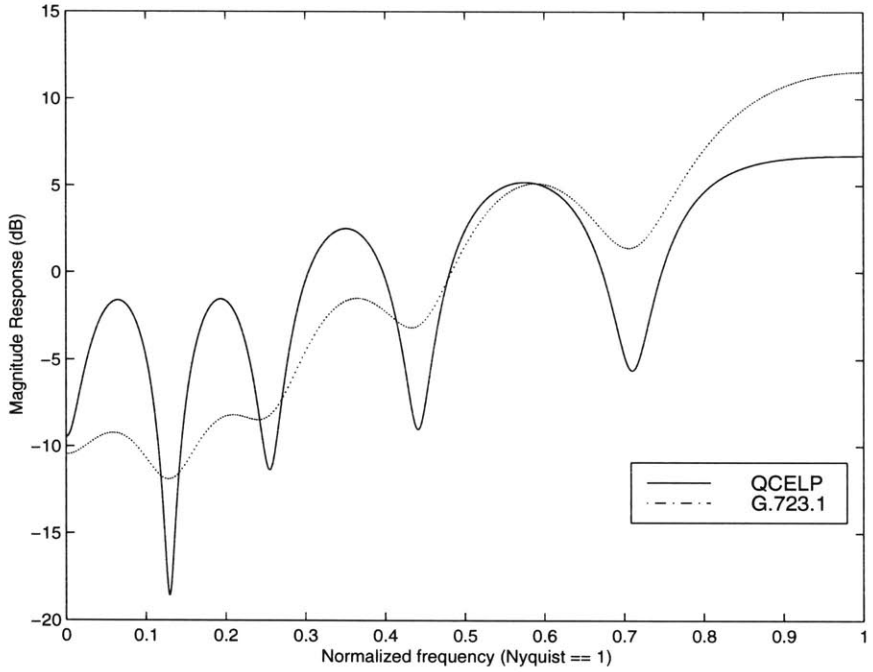

Figure 2-4: Weighting and Associated Synthesis Filters for a Frame of Voiced Speech

Periodicity and Pitch Search

A frame size of 160 samples is long enough to contain more than one period of speech (pitch for speech is typically around 130 Hz which is about 60 samples). Pitch prediction is only

[13]Thus, only using one degree of freedom.

[14]Simulations showed that using the QCELP filter for G.723.1 causes a very perceptible reduction in quality.

done for Rate 1 and Rate 1/2 packets. A simple prediction filter is derived in this manner: future speech L samples from now is predicted to be some gain, b, multiplied by the present sample. Therefore, we only need to send the error, $e[n] = p[n] - bp[n - L]$. This allows for better quantization. The pitch synthesis filter (used by the decoder and the encoders decoder) is therefore

$$\frac{1}{P(z)} = \frac{1}{1 - bz^{-L}} \tag{2.25}$$

and is excited by the prediction error, $E(z)$. Eight bits are used to represent the lag, L (ranges from 17 and 143, including some half delays) and three bits to represent the prediction gain, b (ranges from 0 to 2.0). Pitch prediction (new values for b and L) is done every subframe (40 samples). However, values for the whole frame are stored and available since L is usually greater than 40.

Figure 2-3 shows exactly how this is determined. Though some of the complexities (approximations used to enhance speed/reduce complexity) are omitted, it shows how the ZIR and ZSR responses of the synthesis filter, $1/\tilde{A}(z)$ (Equation 2.23), also update once every subframe with the new qLPCs.

Codebook Search

Once the pitch parameters are determined, the excitation parameters are then determined. Circular codebooks with 128 values are used to encode Rate 1 and Rate 1/2 frames (separate codebook for each rate). These are calculated 16 times every frame for Rate 1 and 4 times every frame for Rate 1/2. Rate 1/4 and Rate 1/8 use other mechanisms, based on energy of prediction residual and a pseudorandom generator. Only the codebook search mechanism for Rates 1 and 1/2 is briefly described here.

The codebook parameters specify the excitation to the speech filter (an approximation to $E(z)$ above). This excitation is generated by scaling a codebook vector by the codebook gain, G. The goal of the search (shown in Figure 2-5) is to find the codebook vector and gain which minimize the weighted error between input and synthesized speech. The gain is quantized by taking the log and using a combination of linear prediction, a scalar quantizer and a lookup table. The index for the value in the vector and the sign of the gain are sent after doing some straightforward manipulations.

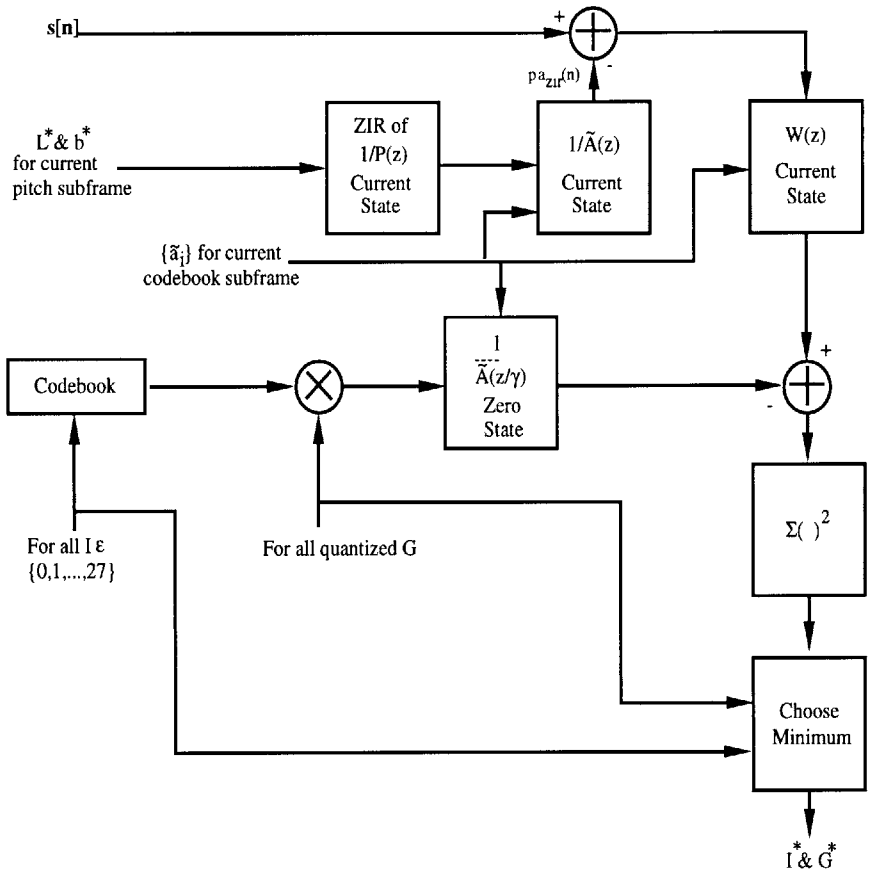

Figure 2-5: Simplified Codebook Search for QCELP[23]

Analysis-by-Synthesis for G.723.1

The actual (without any simplifications) block diagram for G.723.1 is shown in Figure 2-6). Initial examination of Figure 2-6 suggests that the algorithm is quite different from that of QCELP. This is not so: the high-level design is very similar. Below is a list of some of the minor differences:

- G.723.1 uses a fifth order predictor [15]
- A Harmonic noise Shaping filter is used to remove noise in between harmonics. This is absent in QCELP[16]. Tests[17] were run on female speech to confirm this.
- Weighting filter, as discussed earlier.

[15]Sample $p[n+L]$ is predicted using a weighed sum of 5 samples in the neighborhood of $p[n]$.

[16]Since QCELP provides very high quality speech, it does not need this filter. G.723.1, a lower rate coder, exhibits higher quality with this filter present.

[17]By author.

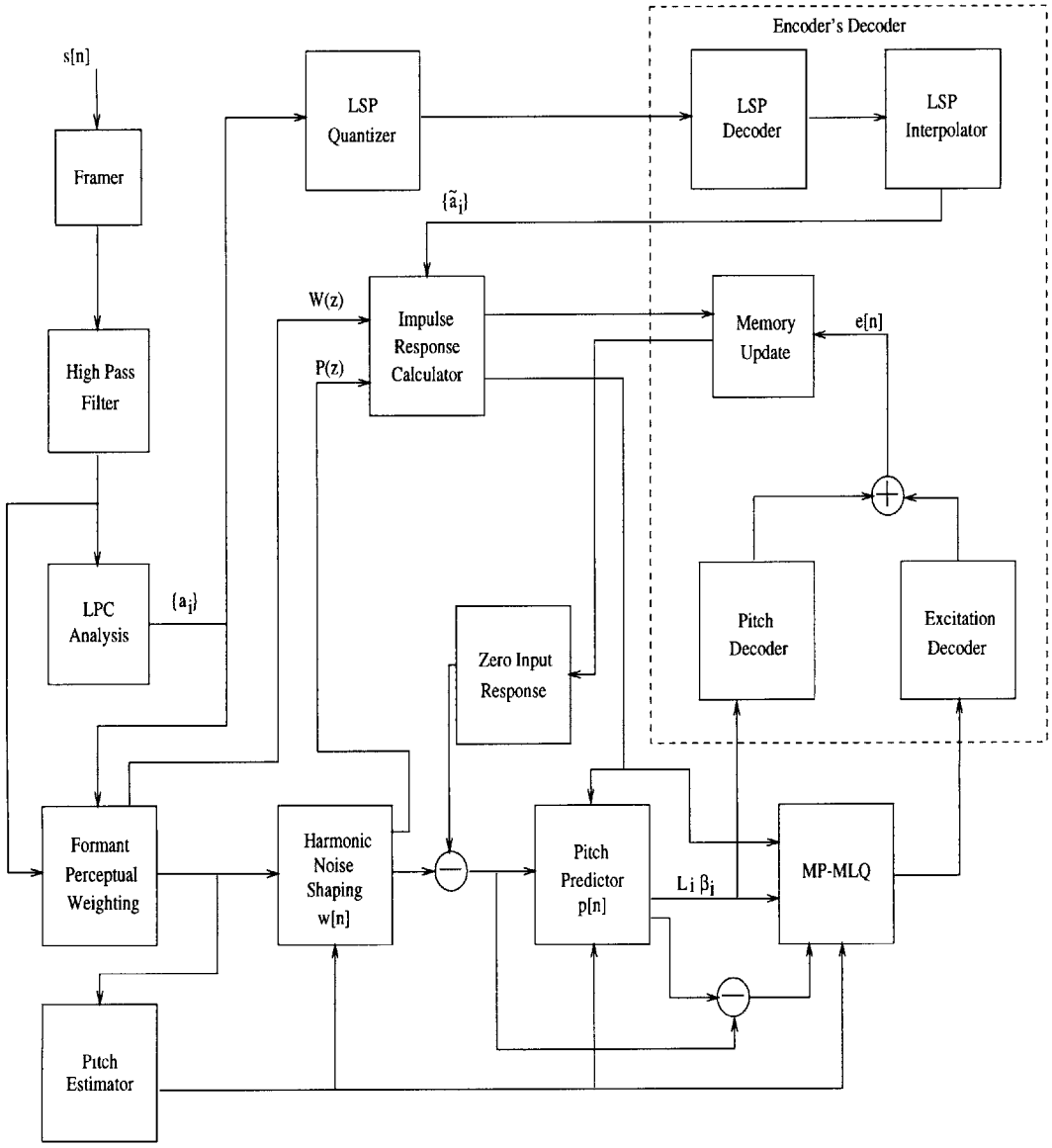

Figure 2-6: Encoder for G.723.1[12]

- Rate 6.3kbps uses MP-MLQ (Multi-pulse Maximum Likelihood Quantization). MP-MLQ uses 5 or 6 (exceptions in the cases when the pitch, L, is small) pulses of equal absolute value spaced either at even or odd time samples to represent the excitation to the pitch filter.

2.4 Decoder

The packets sent by the encoder through the channel is then decoded by the decoder to generate speech that is perceptually close to the original. The Figure below (2-7) shows the decoding algorithm for QCELP.

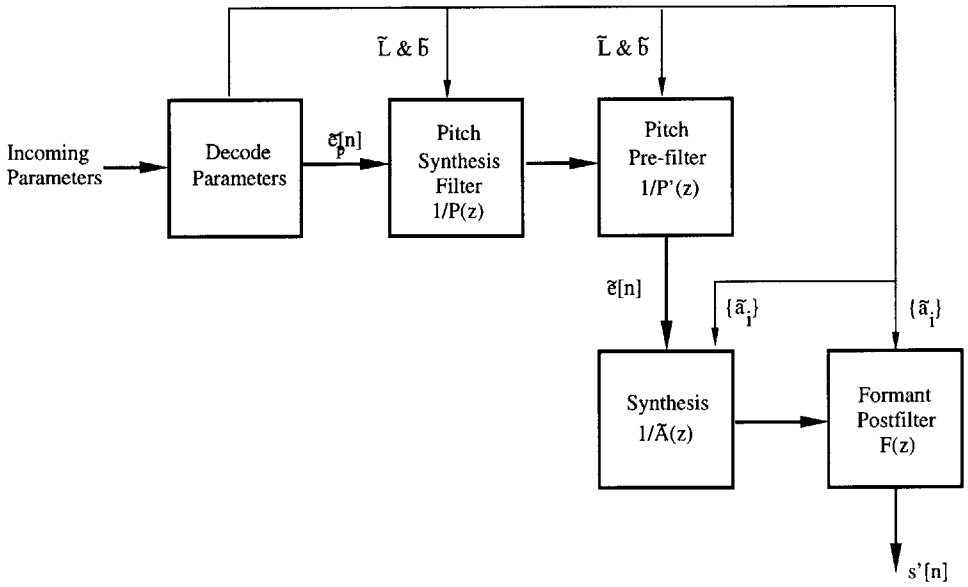

Figure 2-7: Simplified Decoder for QCELP[23]

We have seen most the decoder since the encoder uses these functionalities for its analysis. The excitation is derived from the parameters and fed to the pitch filter, which then creates the residual that excites the synthesis filter, $1/\tilde{A}(z)$. The pitch prefilter between the pitch and synthesis filters improves the quality of the signal[18]. Synthesized speech undergoes some processing before being sent to the A/D (usually includes first converting to μ-law quantization). We are mainly interested in the postfilter block.

[18]QCELP uses a simple pole filter very similar to the pitch filter: same lag but different gain. G.723.1 uses forward and backward correlation analysis to increase SNR at multiples of the pitch period.

2.4.1 Formant Postfilter

Postfiltering is done on the speech (every subframe) to improve the quality. Amongst other things, postfiltering reduces the troughs (also reduces perceptible noise) and increases the peaks (most of the perceptual information are stored in formant peaks) in the frequency response of the speech. It does this by using a conventional ARMA filter, dependent on speech, $\tilde{s}[n]$, given by

$$F(z) = \frac{\tilde{A}(z/\lambda_1)}{\tilde{A}(z/\lambda_2)} B(z) \tag{2.26}$$

Both QCELP and G.723.1 use similar equations. For QCELP, $\lambda_1 = 0.625$, $\lambda_2 = 0.775$ whereas for G.723.1, $\lambda_1 = 0.65$ and $\lambda_2 = 0.75$. B(z) is an anti-tilt filter that tries to offset the spectral tilt in $\tilde{A}(z/\lambda_1)/\tilde{A}(z/\lambda_2)$. The weighting filters, $B_Q(z)$, $B_G(z)$ for QCELP and G.723.1 respectively, are:

$$B_Q(z) = \frac{1}{1 + 0.3z^{-1}} \tag{2.27}$$

$$B_G(z) = 1 - 0.25k_1 z^{-1}$$

$$k_1 = \frac{3}{4}k_{1old} + \frac{1}{4}k \tag{2.28}$$

$$k = \frac{\sum_{k=1}^{59} \tilde{s}[n]\tilde{s}[n-1]}{\sum_{k=0}^{59} \tilde{s}[n]\tilde{s}[n]}$$

Simulations showed[19] that the the G.723.1 filter, because of it's speech-dependent tilt filter, does a much better job in terms of speech distortion[20]. Some of this "superiority" is also partly due to the fact that the QCELP postfilter (see Figure 2-8) brings down the troughs and raises the peaks much more than G.723.1 (thus, more signal distortion)[21].

[19] Run by author.

[20] This was done by observing the quality and spectrum of speech through a multistage tandem using both filters in the QCELP coder.

[21] QCELP does better at noise reduction because of this attenuation of noise in the troughs.

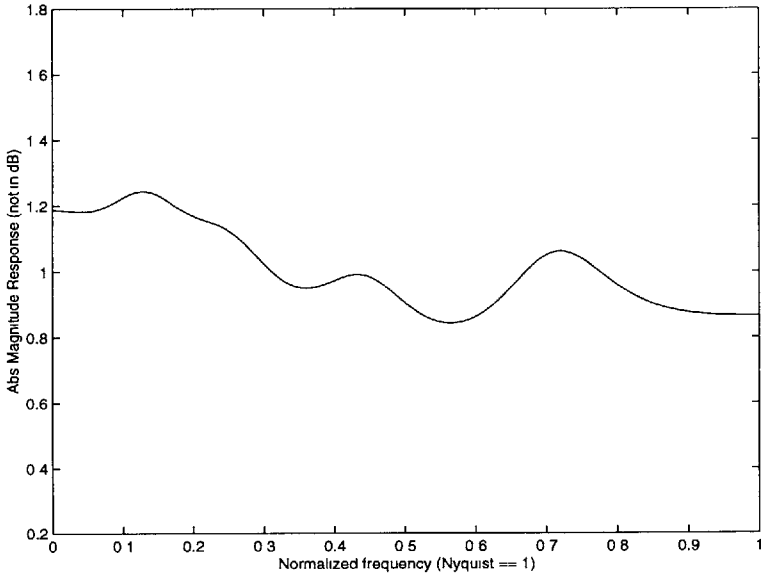

Figure 2-8: Postfilter for a frame of Voiced Speech

Chapter 3

TRANSCODING

The coders just described in the last chapter will now be investigated under tandem and transcoding situations.

3.1 The Problem Statement

As seen earlier, previous work on tandeming suggests that there is much degradation when most CELP coders are put in tandem. We will concern ourselves entirely with the two coders described earlier (QCELP and G.723.1). The diagram below illustrates the envisioned transcoder (Figure 3-1)

Coder 1 and 2 in Figure 3-1 are either QCELP and G.723.1 (the two cases are G.723.1 to QCELP and vice versa). Using such a transcoder could enhance speech quality and improve delay characteristics. In designing this transcoder, initially, we require that the encoder for coder 1 and the decoder for coder 2 be unchanged[1] . This condition is later relaxed a little to see if further improvements could be made. Also, for most of our initial analyses we use flat speech (since the previous work done seemed to mainly use this kind of speech and in cases where other characteristics were used, no significant differences were found). However, we will go beyond flat speech to consider, in particular, modified IRS (Intermediate Reference System)[2][14]. We now proceed to a closer investigation into issues involved in tandeming. In particular, we look at LPC and postfilter degradation.

[1]With the exception of the postfilter. We will consider cases with and without postfilter.
[2]ITU-T Recommendation P.830: discussed later in Section 4.2.1.

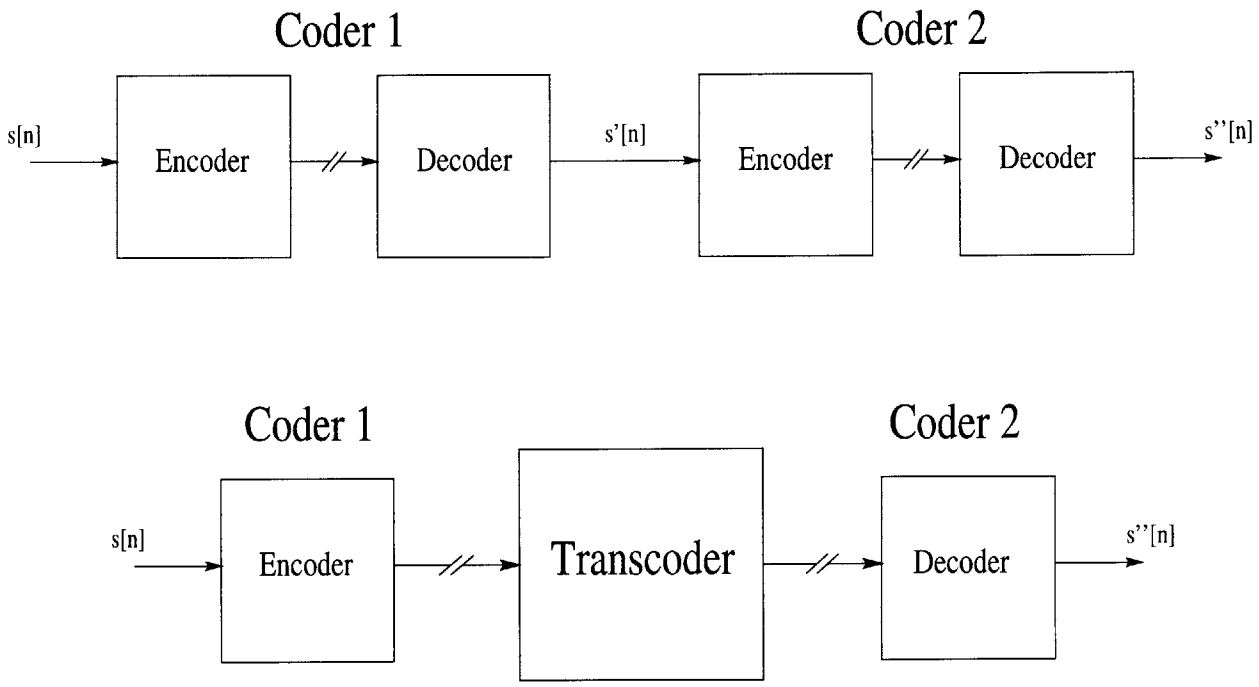

Figure 3-1: Above: Transcoding without a Transcoder. Below: Transcoding with a Transcoder

3.2 LPC Degradation

As we have seen, the LPC synthesis filter forms the "backbone" of the speech reconstruction process. When the speech goes through a tandem connection, one expects the LPC estimates to deviate from what they used to be at the first decoder. If they deviate far enough, the random codebook will not be able to recreate a speech perceptually close to the original speech. If the speech out of decoder 1 is close to the actual speech, we expect the LPC estimates in encoder 2 to produce a synthesis filter that is close to that of the first coder. The figure below (Figure 3-2) shows the behavior of a LPC synthesis[3] filter in tandeming.

The effects shown in this figure is typical of most of the frames of voiced speech. Some deviations are more extreme that others. However, there are enough extreme deviations[4] to suggest that there could be a problem. Two reasons why the LPC synthesis filter after the first stage could be different from that of the second stage are:

- The speech from decoder 1 is perceptually very different from the actual speech

[3]Using qLPCs.

[4]Further tandem stages show a worsening in these deviations. This suggests that quantization errors are probably not the sole cause of this phenomenon.

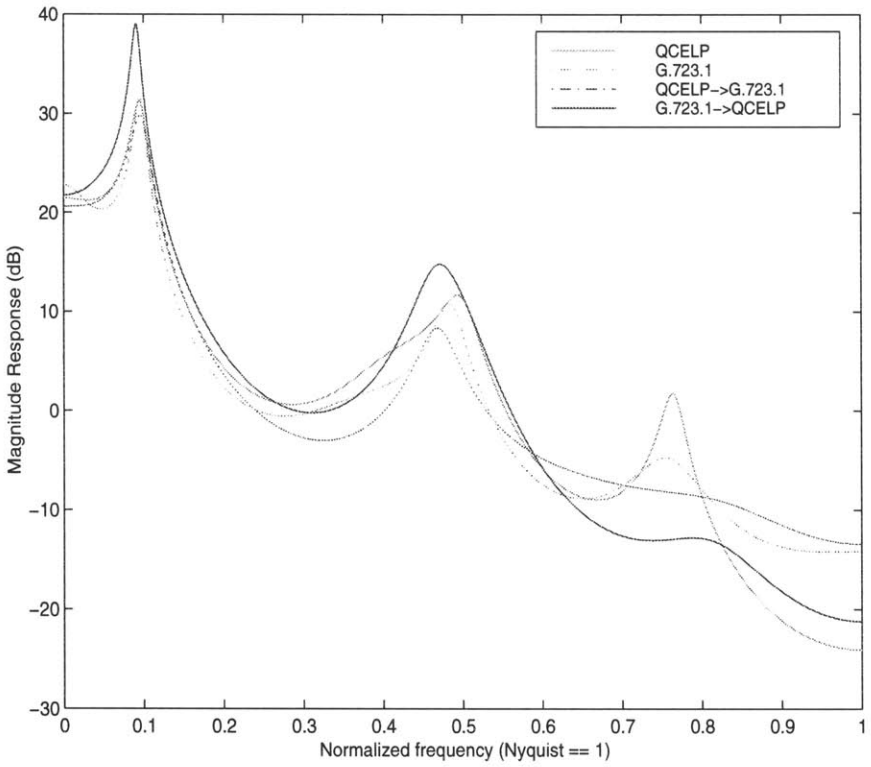

Figure 3-2: Synthesis filters for single and tandem stages

- Although the speech from decoder 1 is perceptually close to the actual speech, the spectra is different enough so that LPC estimation produces a "bad" filter.

Initial listening of speech from decoder 1 did not reveal perceptual differences. The second alternative, therefore, seems more likely. In the autocorrelation form of the LPC analysis, the LPC representation matches the magnitude of the speech spectrum but not the phase[3]. The effect of phase in human speech perception seems to be minimal; however, it is well known that phase modifications can produce dramatic changes in wave shape[3]. It is thus reasonable that the distortion could be due to phase.

If indeed the LPC analysis is producing the distortion, a transcoder could utilize the LPCs (in the form of LSPs) from the first coder and convert them to LPCs for the second coder. In doing so, we could potentially avoid calculating the LPCs from the synthesized speech from the first decoder. This idea is discussed in more detail and implemented in Section 3.4.

3.3 Formant Postfilter Degradation

In addition to LPC degradation, the formant postfilter could be introducing some undesired effects that show up in tandem situations. As was discussed earlier, the formant postfilter (also known as short-term postfilter) reduces the spectral valleys (troughs) and increases the peaks. This postfiltering further adds to the noise reduction introduced by the weighting filter and in some cases ensures that we do indeed have good control of the noise spectrum. This is because the weighting filter acts based on the expectation that weighted quantization error will be white noise but it cannot guarantee that all the time[24]. The postfilter, however, does not discriminate between noise and signal. As a result, significant signal distortion could occur. The postfilter for QCELP, being more severe than that of G.723.1, performs poorly in the multiple tandem case. Why not remove the postfilter altogether? As mentioned earlier, the postfilter gives us better control of output noise[24]. Removing the postfilter could cause some unwanted perceptual noise in tandeming. These issues and others are investigated in the upcoming sections.

3.4 Transcoding by LSP Interpolation

Given the possible degradations just discussed, we design the following transcoder. Line spectral pairs from the first coder are used to produce LSPs for the packets sent out to the decoder of the second coder. Given the complexity of CELP coders and the need for speech to produce other information (rate decision, codebook parameter etc), we only consider the case where we do most of the decoding process of coder 1 and then most of the encoding process of coder 2[5]. In particular we do the following (see Figure 3-3):

- Pass the decoded LSPs from decoder 1 to encoder 2.
- Have the option of turning off the postfilter in decoder 1[6].

There are certain issues that need to be addressed in the implementation of this algorithm.

[5]Most of the decoding process of coder 1 needs to be done to produce synthetic speech which is necessary for the encoding process.

[6]We will also do tests with the postfilter on, and make comparisons.

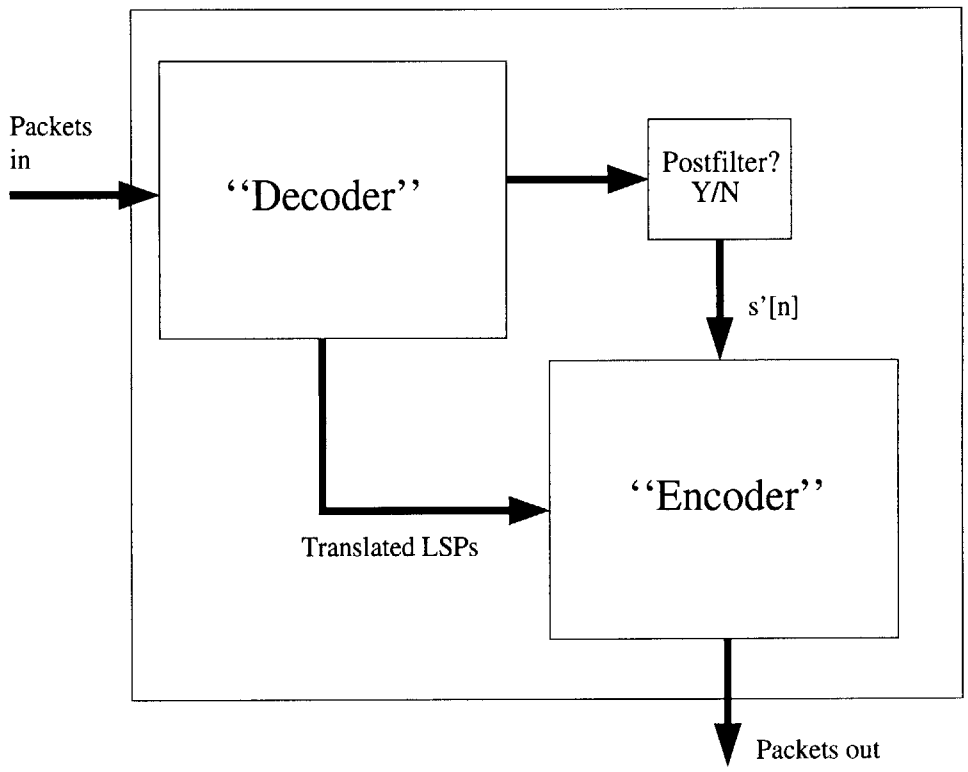

Figure 3-3: General Transcoder with LSP interpolation

3.4.1 Appropriateness of LSP translation

Coder 1 and Coder 2 (for e.g., G.723.1, and QCELP) have different frame sizes: G723.1 has a 30 ms frame and QCELP has a 20ms frame. As a result, direct reuse of LSPs is not feasible since packets are not aligned. LSPs need to be interpolated before passed on to next coder. In the case of G.723.1 to QCELP, the transcoder needs to produce three packets for every two incoming packets. Therefore, we must also buffer up packets when going from QCELP to G.723.1 (or in general, coders of smaller frame size to coders of larger frame size). Issues of delay are addressed later in Section 3.6.

Linear interpolation of LSPs introduce errors in the sense that the new LSPs derived (for the new frame size) are not exactly matched of the speech segment they describe – there is a mismatch. We will use the word "mismatch" to generally refer to cases where we do not use the parameters directly produced from the decoded speech by the normal means – calculating the autocorrelations, LPCs, LSPs, etc. We should also note that the windowsizes that are used by G.723.1 and QCELP differ – by 20 samples. This will introduce more inaccuracy, albeit very small, to our translation process.

The specific details of implementation is dealt with in Section 3.5. Once the LSPs have been interpolated to suit encoder 2, there are other issues that need to be considered. They are discussed below.

3.4.2 LSP and LPC mismatch for QCELP

We have decided that, for now, we need decoded speech to be passed to second encoder, since this speech is used by the encoder to make rate decisions, produce LPCs, and possibly to perform other complex functions that vary among CELP coder. Its most important use (and that's the most important reason why we have it) is in the analysis-by-synthesis loop. Since this speech is available, we could conceivably use some combination of parameters produced from this speech in conjunction with our imported LSPs for better quality. The fact that we are going to be using translated LSPs with the decoded speech implies that there is going to be some kind of mismatch between these LPCs – and parameters derived from these LPCs – and the decoded speech (which usually produces but, in this case, did not produce all of these parameters). The parameters of most concern here are the autocorrelation coefficients from the speech and the LPCs that are produced from these coefficients. These parameters could be derived from either the decoded speech or from the imported LSPs (other parameters[7] can only be derived from the one source; decoded speech). Ignoring the complications of rate and voice/unvoiced decisions, LPCs, autocorrelation coefficients and LSPs are mainly used in the encoder for these purposes:

(i) LSPs are quantized and sent as packets.

(ii) These quantized LSPs are used to produce LPCs (qLPCs) used for most of the filters in analysis-by-synthesis loop[8].

(iii) Where applicable (depending on coders), LSPs, LPCs and autocorrelation coefficients are used to calculate the sensitivities that are used for quantization of these LSPs (see Section 2.3.3).

It is pretty clear in the first and second cases that the use of the "imported" LSPs are desirable (that is the motivation for importing LSPs). This is our first mismatch. Using

[7]Energy in certain bands, SNR, amongst others.

[8]Remember, in G.723.1 this is not true of the weighting filter: unquantized LPCs are used.

imported LSPs for the first case implies using it for the second case; otherwise, the encoder's decoder will be not be mirroring the decoder.

In the third case, however, (only applicable to QCELP and thus pertinent to the G.723.1-to-QCELP tandem), there are various options available which could yield potentially different results. We briefly discuss the the viable options and resort to experimentation (by listening) to decide which is best.

Deriving Parameters Purely from Decoded Speech

In this case, all the parameters passed to the sensitivity calculator are derived from the decoded speech. The translated LSPs are not used at all in this stage. The implications are twofold: we do not save anything by way of complexity and we need more memory to carry around two separate copies of LSPs. The only mismatch in this case is that we will be quantizing translated LSPs (tLSPs) using sensitivities generated by parameters from the decoded speech.

Deriving Parameters Purely from Imported LSPs

In this case, all the parameters passed to the sensitivity calculator are derived from imported LSPs (tLSPs). Therefore, LPC-to-LSP calculations do not need to be carried out on the decoded speech. However, we need to somehow calculate all the parameters necessary for sensitivity calculation ($P(z)$, $Q(z)$ and Rs, the autocorrelations). $P(z)$ and $Q(z)$ can be calculated from the LPCs calculated from tLSPs (call these LPCs, the qtLPCs). The R_k's are more complicated to calculate. Eleven autocorrelation coefficients are used to calculate PARCORs and LPCs. There is no way to reproduce all of these 11 autocorrelation coefficients given the 10 LPCs. However, given $R[0]$ the first autocorrelation coefficient, we can calculate the other 10 (see the algorithm developed to do this in Appendix C), which are multiples of $R[0]$. Fortunately, only the relative magnitudes of these autocorrelations are important in using the sensitivities for quantization.

Deriving Parameters by Doing a Combination of the Two

Here, tLSPs are used, in place of the LSPs generated by speech, for analysis-by-synthesis, for transmission, and for sensitivity calculations. All other parameters used for sensitivity

calculations are derived from the decoded speech. As a result, LPC-to-LSP calculations do not need to be carried out.

Empirical Analysis

These three alternatives are tested by listening to speech. The algorithm below (Section 3.5) is used to carry out this test. The necessary adjustments are made so all three alternatives can be tested. After several listening tests, it is very obvious that the first option is the outright winner. The other options produce speech that have some perceptible quantization noise. This suggests that using decoded speech to generate parameters for sensitivity calculations is the best approach. Therefore parameters from that speech must be the optimal set of values for the calculations. One setback with the use of this alternative, as mentioned earlier, is the need to carry around two sets of LSPs (tLSPs and the LSPs derived from speech). Since there are only 10 LSPs, this should not be a problem.

3.4.3 LPC and LSP mismatch for G.723.1

Though the issues in G.723.1 are somewhat different from those of QCELP, the results are similar. In the case of G.723.1, LSPs are quantized via a simple method that does not depend on sensitivities. However, there is one area in which we can choose between using LPCs calculated from the tLSPs (qtLPCs) and those calculated directly from the decoded speech. Empirical analysis shows that using the LPC from the speech results in better quality speech.

3.5 Implementation of LSP transcoder

Taking the results of the analyses of these mismatches into consideration, we implement the transcoders for both the G.723.1-to-QCELP and QCELP-to-G.723.1. We will give a detailed description of the latter. To avoid too much repetition, we'll only discuss the ways in which former differs from the latter.

3.5.1 Transcoder for QCELP to G.723.1

The transcoder needs to take in 20ms QCELP packets and send out 30ms G.723.1 packets. It does this by decoding the incoming packets to produce speech (just like in the QCELP

decoder), while producing appropriate LSPs for an encoder very similar to that of G.723.1 (see Figure 3-4). We use already available C simulations for the separate coders to simulate

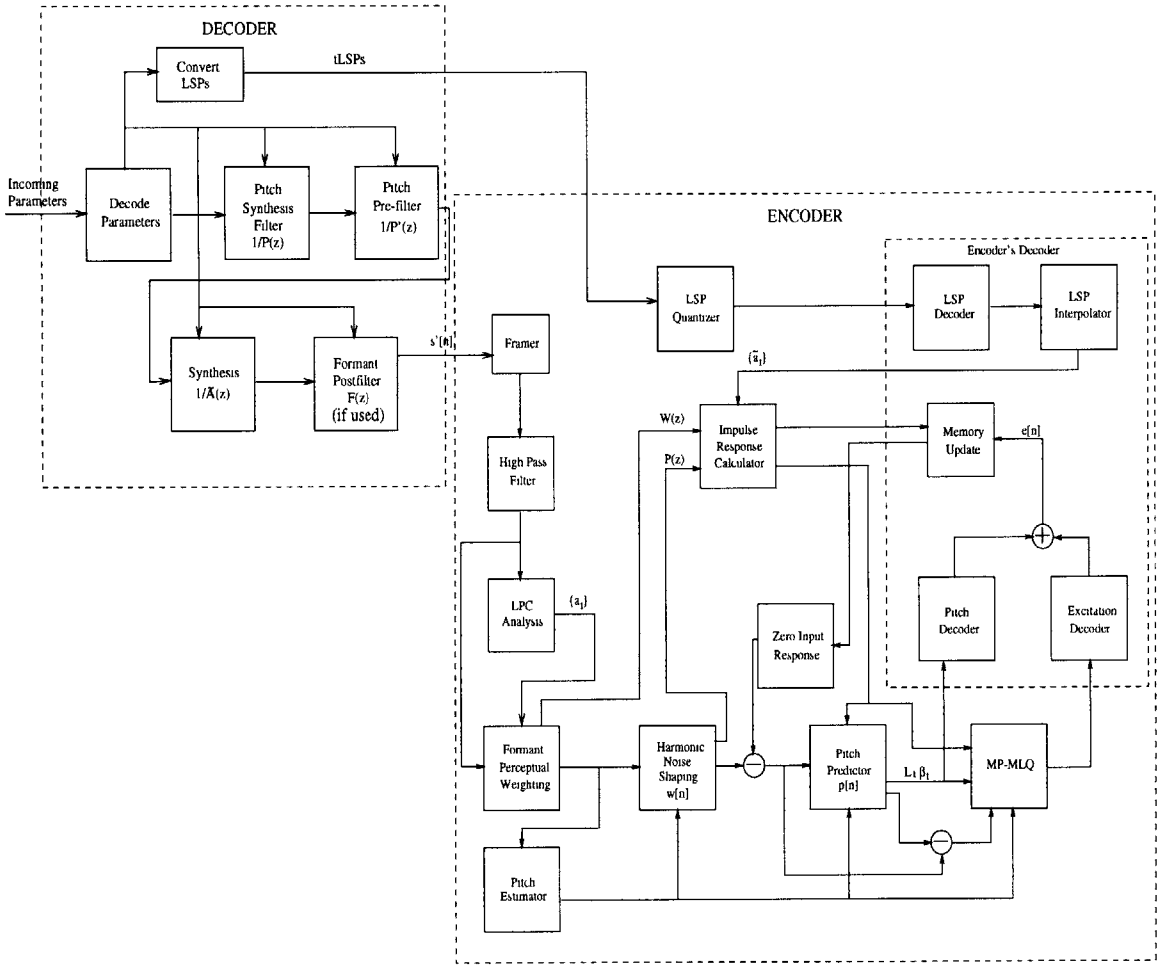

Figure 3-4: Transcoder for QCELP to G.723.1

this transcoder. The version implemented is off-line (not real time). We describe this implementation and mention what needs to be done to write a real-time simulation for this transcoder.

Doing the transcoder off-line (which is sufficient for listening test for quality changes) is not difficult (nor too tedious) given the C simulations for different coders. In particular, the following needs to be done:

(i) Use the QCELP C simulation for the decoder to produce the speech (either postfiltered or non-postfiltered version).

(ii) At the same time, produce decoded LSPs from the decoder. These are LSPs that

have been unquantized from the incoming packets.

(iii) Take these 20 ms LSPs for all frames and do an off-line interpolation to produce 30ms LSPs for G.723.1.

(iv) Pass these new LSPs into the encoder of G.723.1 using the decoded speech from QCELP making sure we use the right parameters at the right place (see Section 3.4.2).

In (i) and (ii), all we needed to do was turn off the postfilter when necessary and add a few lines of code to print out the decoded LSPs to a file.

Part (iii) is the most involved. LSPs are represented in fixed point in G.723.1 and floating point in QCELP. Therefore, the appropriate conversion needs to be made. Linear interpolation is done as follows:

$$tLSP_j^G = \beta qLSP_{i-1}^Q + (1-\beta)qLSP_i^Q \tag{3.1}$$

where $tLSP_j^G$ is the calculated LSP for the jth 30ms frame for G.723.1, $qLSP_{i-1}^Q$ and $qLSP_i^Q$ are the 20ms frame quantized LSPs from QCELP closest to the speech signal around which $tLSP_j^G$ is calculated (See Figure 3-5). β is the appropriate weighting fraction. The matlab

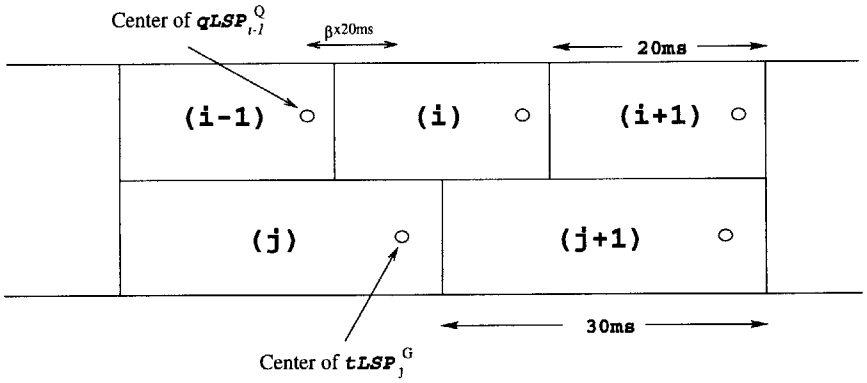

Figure 3-5: Interpolation of LSPs

code in Appendix A.1 gives a detailed implementation. In some cases, there might not be enough information to calculate tLSPs for the last frame. In this case, we just drop the last frame. End effects are negligible so too much care does not need to be taken when determining LSPs for the last frame.

In part (iv), we need to pass in the LSPs to the G.723.1 encoder. Since we need the LPCs calculated from the decoded speech for the weighting filter (as discussed in Section 3.4.3 above), Levinson-Durbin must be run. The tLSPs are now passed in as the LSPs for the encoder instead of doing the LPC to LSP calculations. However, the tLSPs are not passed "as is" to the encoder but are derived from the qtLPCs (calculated from the tLSPs) by doing an LPC to LSP calculation. This is necessary because the G.723.1 algorithm does bandwidth expansion on LPCs before LSPs are calculated. Failure to do this tLSP→qtLPC →new tLSP caused a clicking noise in one of the frames of a test file containing female speech.

Doing the above in real-time is not conceptually more difficult but a little trickier. Since we are going from a smaller frame size to a larger frame size, we need to buffer up one frame of LSP values every other frame (going from 20ms to 30ms), on average, so we can do interpolation. We also need to buffer speech for LPC calculation (for weighting filter) since the autocorrelation window is centered about the fourth subframe (see Section 2.3.1). Calculating β is straightforward. We could do it exactly as is in the matlab code (Appendix A.1); but that requires keeping track of the number of frames that have gone by since the beginning. This could become impractically large. It is easily seen that β is periodic with period 2 where the 2 unique values are $0.4375, 0.9375$. We can simply store these values and use a counter that goes from 1 to 2 to achieve our goal. We will give a more in-depth analysis of some issues involved in real-time implementation when we discuss delay.

3.5.2 Transcoder for G.723.1 to QCELP

Again, this transcoder was not implemented in real-time. The steps are very similar to that of the previous transcoder. Decoded LSPs are printed from G.723.1, converted to tLSPs for QCELP and then imported to the encoder. The interpolation of LSPs is a follows:

$$tLSP_j^Q = \gamma qLSP_{i-1}^G + (1-\gamma)qLSP_i^G \tag{3.2}$$

where γ, like β, lies between 0 and 1. Matlab code for exact implementation is in Appendix A.2. tLSPs are imported "as is" to form the qtLPCs used for the analysis-by-synthesis. LSPs and LPCs used for calculating the sensitivities are derived from the decoded speech. The encoder C simulation is slightly altered to take these into account.

Just as before, we can talk about the real-time implementation of this transcoder. In this case we are going from a larger frame size to a smaller one. However we will still need to buffer up one frame of LSP values in some cases. Again, we wait till the next section to give a more in-depth analysis of all issues involved. Calculation of γ is straightforward since it periodic with period 3, where the 3 unique values are $0.5625, 0.0625, 1.0625$.

3.6 Delay Analysis

In this section, we discuss delay issues of speech in real-time implementation of the transcoders.

3.6.1 QCELP to G.723.1

Under the normal case of no transcoder, the delay of a speech sample from input at the first encoder to output at the second decoder is as follows[9]:

(i) First, the input speech has to be buffered in order to calculate the LPCs. QCELP uses 160 sample LSPs centered around the middle of the 4th subframe (139-140 sample). This requires an extra 7.5 ms of speech to be buffered (in addition to 20 ms frame of speech). Thus, there is a total delay of 27.5 ms.

(ii) There is an algorithmic delay in the encoder. This delay must be less than 20 ms since frames are sent out at this rate. This delay is usually close to this maximum value.

(iii) Transmission delay.

(iv) Decoding delay at the QCELP decoder. This delay must again be strictly less than 20 ms. It is usually much less than this.

(v) Delay in going from linear to μ-law quantization and from going from μ-law through D/A.

(vi) Delay in going through A/D to μ-law and then to linear quantization.

[9]For simplicity we assume this speech sample is the first in the frame. If this is not the case, although the total delay is still the same, there is less delay on the encoder side and equivalently more delay on the decoder sides for that specific speech sample.

(vii) In G.723.1 encoder, just as in QCELP, an extra 7.5 ms (corresponding to a 180 sample LPC window centered about the 210th sample) buffer delay is needed. Thus, there is a total delay of 37.5 ms.

(viii) Encoding delay which must be less than 30 ms.

(ix) Transmission delay.

(x) Decoding delay, again less than 30 ms.

In case of the transcoder, most of the delays are similar. However, the delay in the transcoder itself is different from that in the QCELP decoder and the G.723.1 encoder which it replaces. We make the assumption that it takes about 2 ms to decode about 20 ms of speech[10](3 ms for 30 ms and so on). We want to know what is the maximum allowable time for encoding. Here, encoding involves calculating the tLSPs and doing all the other encoder algorithms as discussed before. Since the transcoder has to generate packets once every 30 ms, and it takes 3 ms to decode a 30 ms, one would expect the maximum encoding time to be 27 ms. The following figures, Figures 3-6, 3-7, 3-8 show the first 180 ms of a real-time implementation of the transcoder using this maximum encoding time. Packets 1, 2, 3, etc. represent incoming QCELP packets whereas Packets A, B, C, etc. represent outgoing G.723.1 packets. Encoding can only begin if there is enough speech to calculate the LPC window (requires 37.5 ms of speech each 30 ms frame) and the tLSPs (the next speech frame is sometimes required to do this interpolation). Figure 3-9 shows this visually.

We notice a couple things from the figures showing the real-time implementation time diagram (Figures 3-6, 3-7, 3-8). First, steady state is achieved at the start of the 4th incoming packet (Figure 3-7). Secondly, the outgoing packets are not sent as soon as they are ready. This is in order to be compatible with the decoder requirement of a packet coming approximately every 30 ms. We could send these packets asynchronously as long as we make sure that, given that the first packet gets there at time t_i ms (and the decoder starts decoding it), the jth packet is there before time $t_i + 30(j-1)$ ms.

We now compare the delay between the normal case and the the case of the transcoder. In this comparison, we initially ignore the delay in converting to μ-law and back, the delay

[10]This is approximately correct for present DSPs. It takes much longer to encode because of the different searches performed.

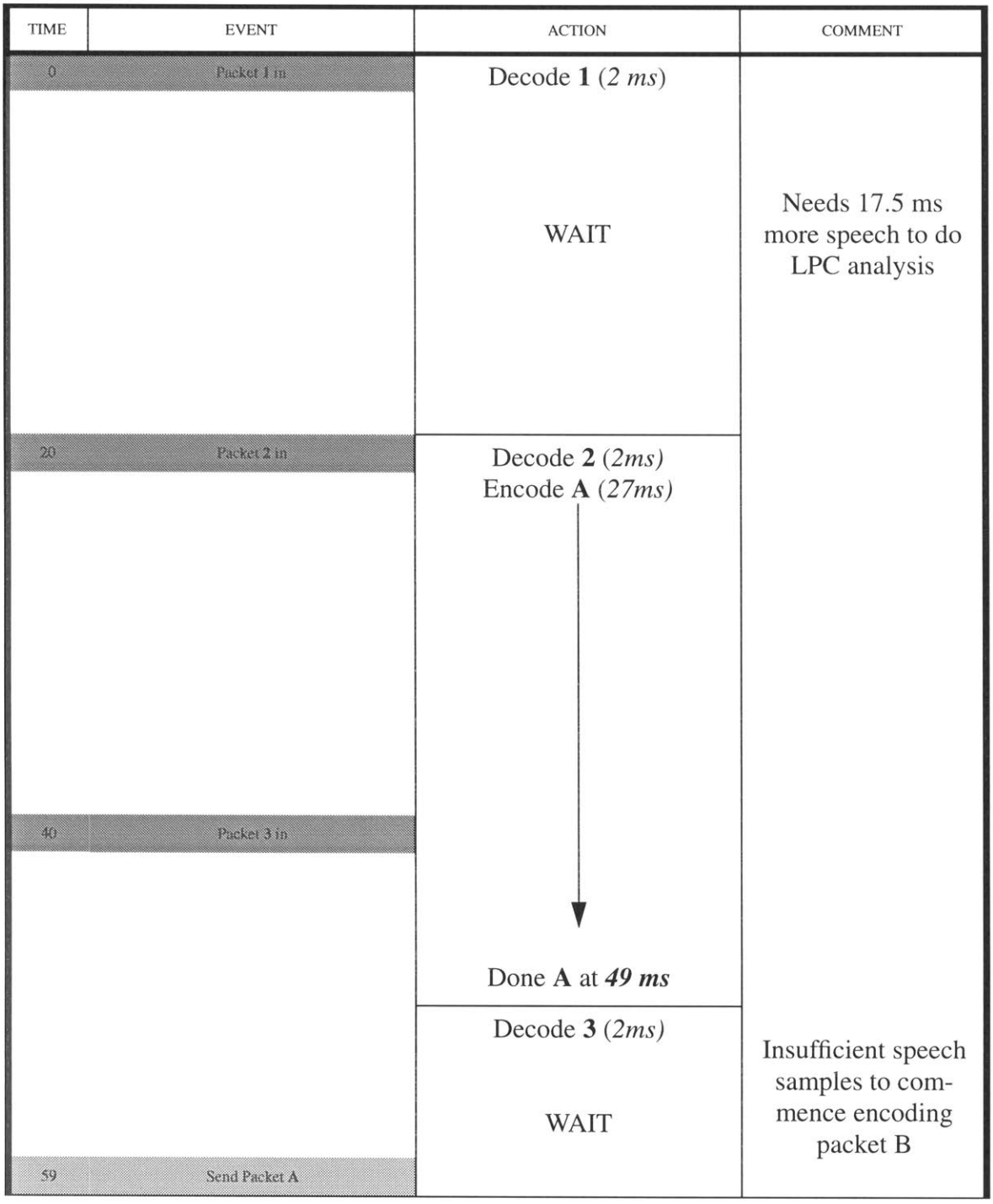

Figure 3-6: 0-60ms of the QCELP-to-G.723.1 transcoder

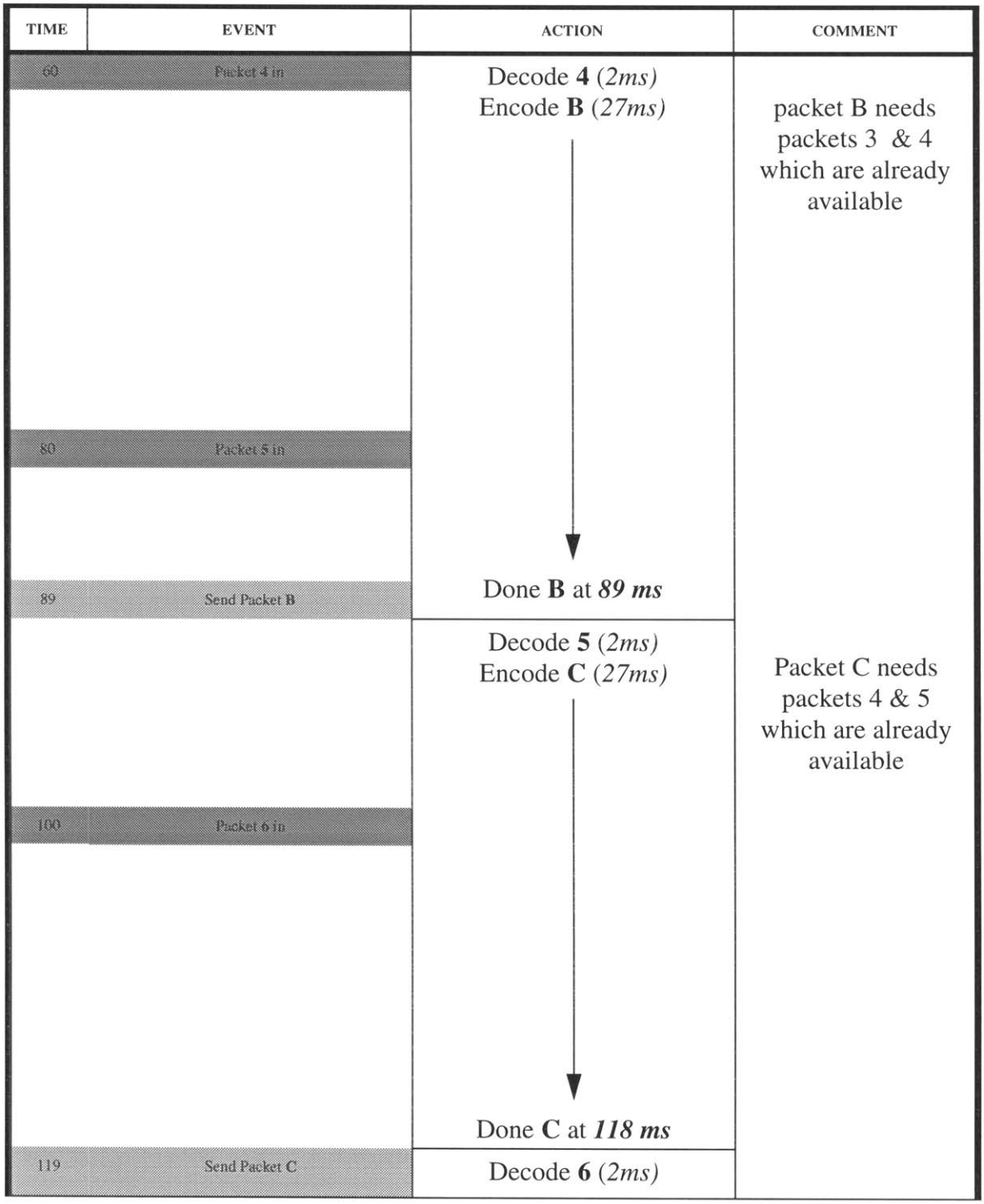

Figure 3-7: 60-120ms of the QCELP-to-G.723.1 transcoder

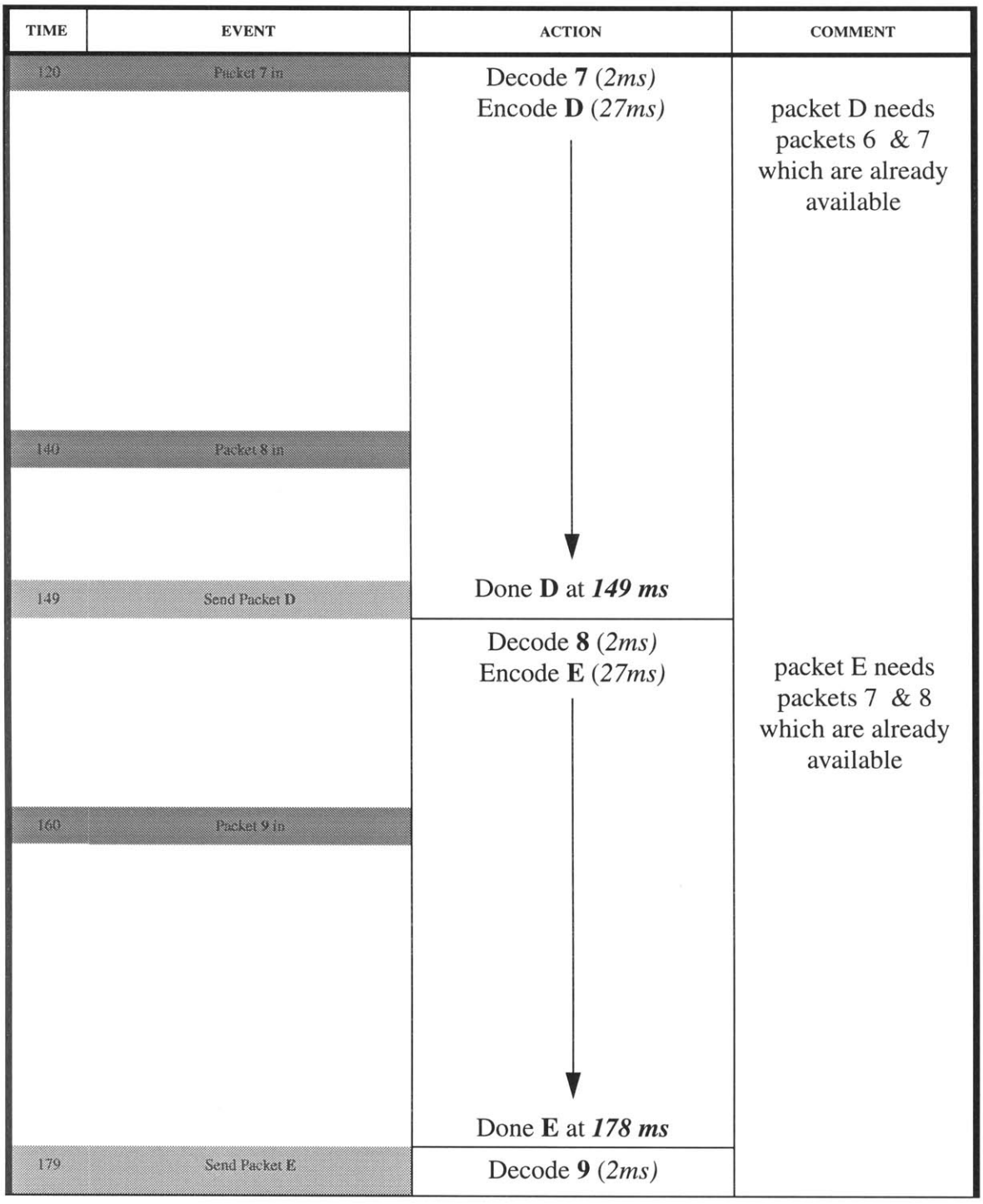

Figure 3-8: 120-180ms of the QCELP-to-G.723.1 transcoder

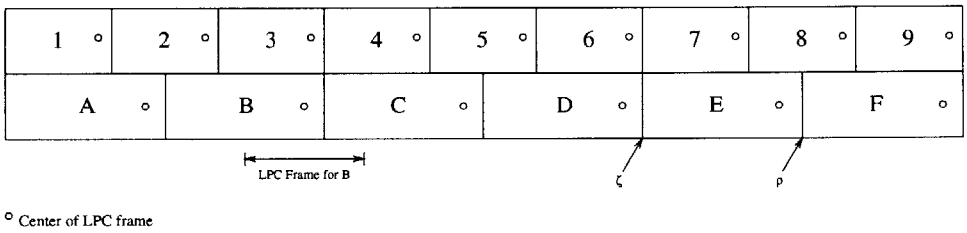

Figure 3-9: Incoming and outgoing packets in QCELP-to-G.723.1 transcoder

in A/D and D/A and the transmission delays (items (v),(vi),(iii) and (ix)). We also take the encoder time for G.723.1 to be 27 ms (the maximum allowable time for the encoder portion of transcoder[11]). The encoder time for QCELP is taken to be 18 ms (which happens to be the maximum allowable delay while transcoding from G.723.1 to QCELP as we will see in the next section). With these numbers, the "total" delay in the normal case is $27.5 + 18 + 2 + 37.5 + 27 + 3 = 115$ ms. The delay in the transcoder case (in steady state) is slightly trickier. Consider the first sample of speech in Packet 7 (shown as ζ in Figure 3-9). This corresponds to outgoing Packet E. From Figure 3-8, we see that Packet E is sent out $179 - 120 = 59$ ms after Packet 7 arrives. Thus the total delay time is $27.5 + 18 + 59 + 3 = 107.5$ ms. Note that even if we had considered different samples, we would still get the same total delay. For example, consider the 80th sample of Packet 8 (corresponding to the 240th sample of Packet E) shown as ρ in Figure 3-9. Figure 3-8 shows the transcoder delay to be $179 - 140 = 39$ ms. However, since the 240th sample is 160 more samples behind the first sample in its packet than the 80th sample is behind the first in its packet, we have an extra 20 ms on the decoder side (corresponding to speech reconstructed at 8000 Hz). Therefore, the total delay is still 107.5 ms. Thus, the transcoder saves 7% in delay. The percentage increases when we consider μ-law, A/D, D/A delays (not present in transcoder) and faster encoding times. The percentage decreases when we factor in transmission delays.

3.6.2 G.723.1 to QCELP

This case is very similar to the previously discussed transcoder. In particular, all the delays itemized above are the same; but the order is reversed so that G.723.1 goes first.

The transcoder implementation with a maximum encoding time of 18ms (20 ms minus

[11] We expect the encoding times between transcoder and the normal G.723.1 encoder to be similar since we are doing very similar computations.

2ms for decoding 20ms frames) is shown in Figures 3-10, 3-11 and 3-12. Again, Packets 1, 2, 3, etc. represent incoming packets (this time, G.723.1) and Packets A, B, C, etc., outgoing packets. Figure 3-13 gives a visual representation of packet dependence. Similar observations are made from the real-time diagrams in Figures 3-10, 3-11 and 3-12. Steady state is achieved at the beginning of incoming Packet 3.

The total delay for the normal case with the same assumptions is the same as before; 115 ms. We now calculate the delay in the transcoder. Consider the first sample in Packet 3, shown by ξ in Figure 3-13. This corresponds to first sample in outgoing Packet D. Figure 3-12 clearly shows the transcoder delay is $101 - 60 = 41$ ms. The total delay is therefore $37.5 + 27 + 41 + 2 = 107.5$ ms as before.

Thus, in both cases, we have similar delay savings. This 7.5 ms comes from the fact that, in steady state, we no longer have to wait, on average, for the extra 7.5 ms of speech to do LPC analysis.

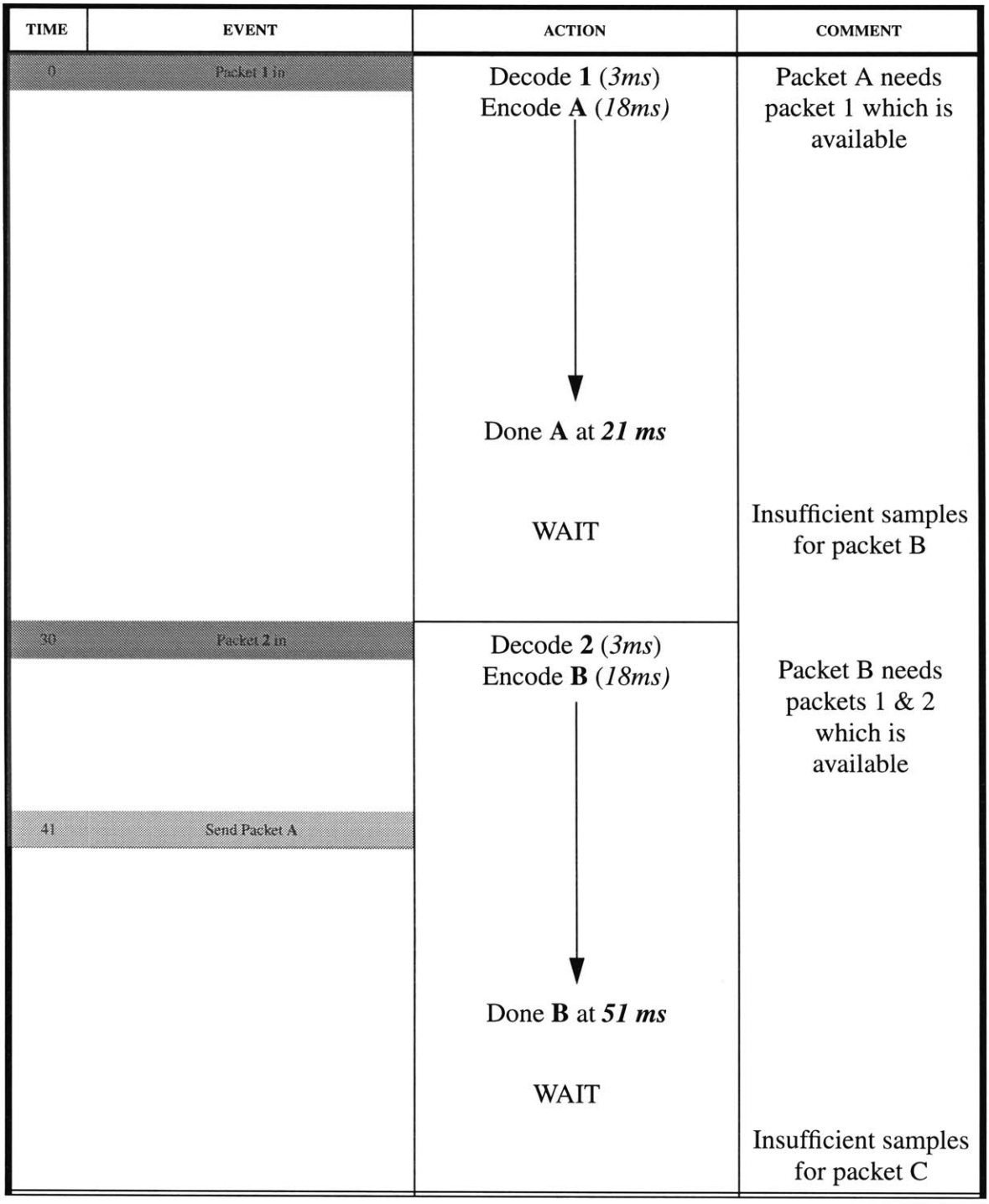

Figure 3-10: 0-60ms of the G.723.1-to-QCELP transcoder

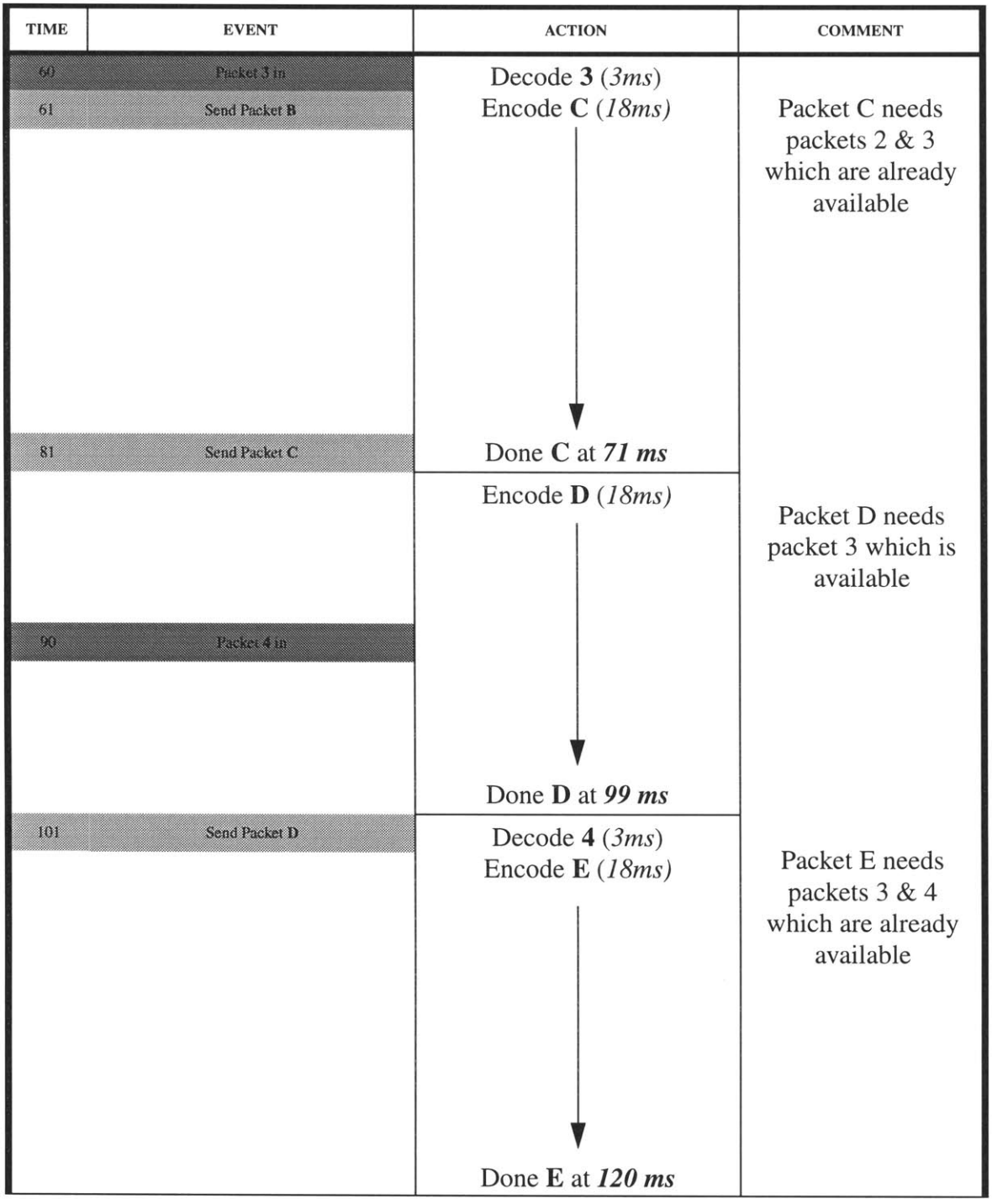

Figure 3-11: 60-120ms of the G.723.1-to-QCELP transcoder

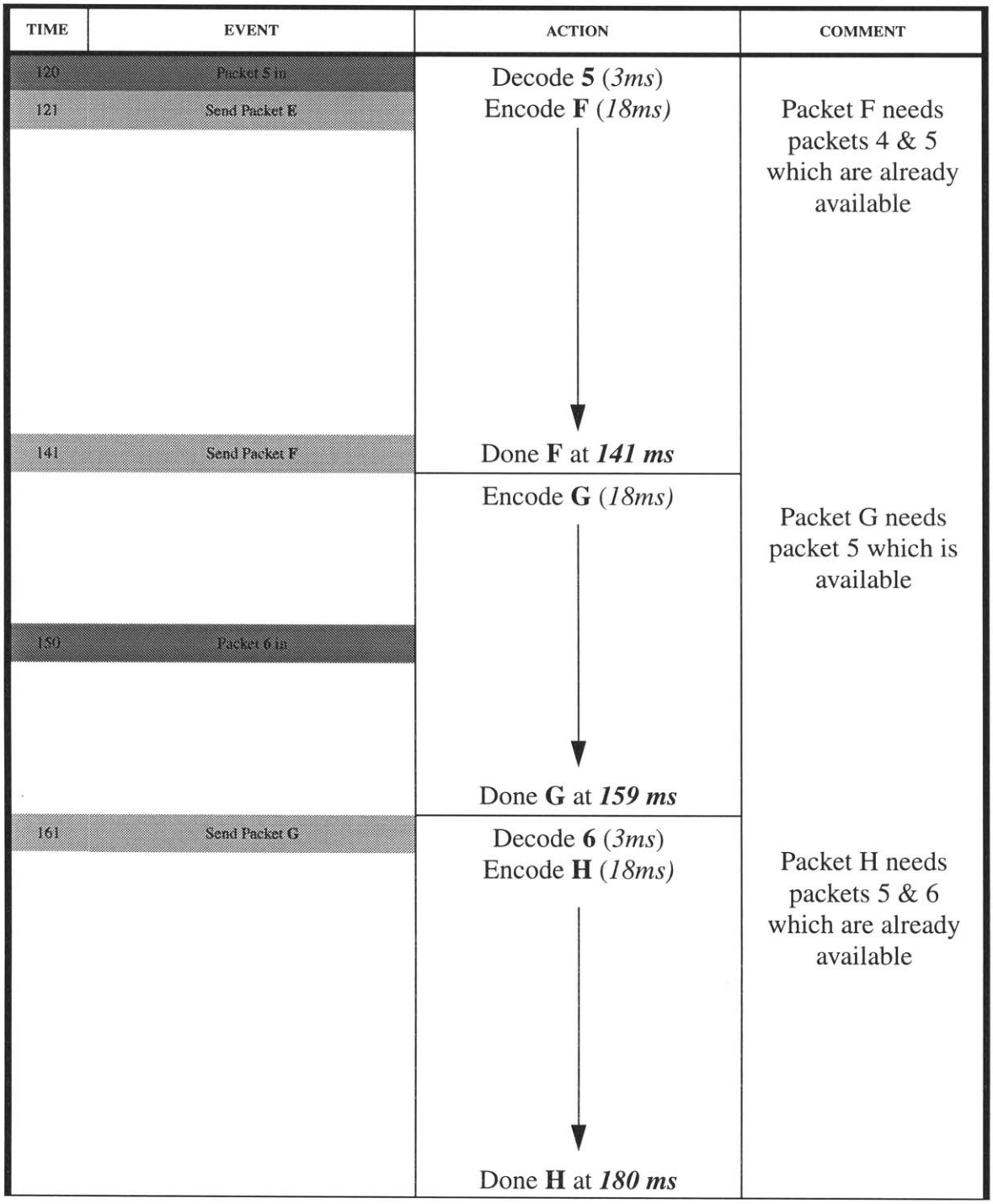

Figure 3-12: 120-180ms of the G.723.1-to-QCELP transcoder

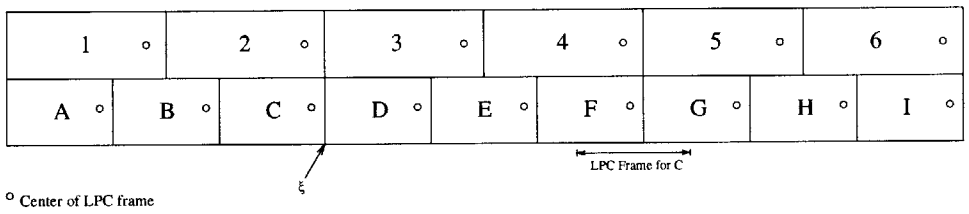

Figure 3-13: Incoming and outgoing packets in G.723.1-to-QCELP transcoder

Chapter 4

Empirical Testing of Transcoder

The transcoder is tested to see if there are any quality improvements over the normal case. Quality testing in speech is quite arbitrary in some ways as there is no very objective way of measuring the perceptual quality of speech. Empirical testing is done by listening to a variety of speech files. Mean Opinion Score (MOS) is generally accepted as a means of rating the quality of speech coders. Here, several listeners are asked to rate the speech on some comparative scale (e.g. EXCELLENT, GOOD, SATISFACTORY, NOT SATISFACTORY, POOR) which is then assigned a numerical scale (say 1-5, 5 being excellent). Averaging these scores over different speech waveforms gives us the MOS.

Several speech files are used to test the quality of the transcoders. Listening is done primarily by the author and secondarily by his supervisors. If initial listening produced promising results, a MOS test will be set up and Mean Opinion Scores calculated for the cases with and without transcoders.

Because QCELP is such a good coder, it is sometimes hard to hear much degradation in the tamdem case with G.723.1. As a result, measuring improvements is difficult. As a result, for listening purposes only, we will also use a lower rate (and lower quality) speech coder so we will have enough degradation from which to measure any improvements. The coder used here is the Variable Rate Speech Service Option for the Globalstar Communication System (Globalstar^{TM}, a proprietary of Qualcomm®). We will not discuss the details of this coder[1]. However, it is a very good substitute for QCELP, with a lower bit rate.

[1]For proprietary reasons.

4.1 Listening for Flat Speech

Here, we use speech files (male and female) with "flat"[2] response.

As mentioned earlier, transcoding is done between G.723.1 and QCELP or Globalstar, a lower rate and quality coder.

First we discuss the distortions observed in the G.723.1 to QCELP tandem.

4.1.1 QCELP and G.723.1 in tandem

When these two coders are in tandem (in whatever order), we notice the following kinds of distortions[3]:

- muffling (bad high frequency reproduction).
- some low frequency distortion.

There are also some other observations that are specific to the use of postfilters and to the order in which these coders are placed in tandem. If QCELP is the leading coder and we use a postfilter, there is some slight added frequency distortion over the case where we turn off the postfilter. On the flip side, turning off the postfilter causes there to be slightly more quantization noise in some speech samples. When G.723.1 is the leading coder, turning off postfilter has no significant perceptible effect.

All these distortions, however, do not amount to a lot and without a trained ear, it is not very easy to detect. This seems to contradict the fact that there is considerable distortion in tandem situation. We deal with this issue a little later. For now, we turn to the case where we use Globalstar^{TM} (from now on, referred to as GS) in place of QCELP.

4.1.2 GS and G.723.1 in tandem

Similar results are observed in this case but the distortions are worse and very perceptible. This gives us room for improvement. We now listen to these speech samples using the transcoder.

[2]This is the speech spectrum produced from a hi-fi microphone.

[3]The comparison is always made with the lower quality coder; in this case, G.723.1.

GS and G.723.1 using transcoder

When G.723.1 is the leading coder, we still observe frequency distortion. In most cases it is similar to the tandem case (without transcoder). However, in a few phrases, there is some small improvement. One such phrase was "a man in a blue sweater", spoken by a male speaker.

When GS is the leading coder, similar overall lack of improvement is observed. Again, there is some improvement in a phrase, "in the rear of the ground floor was a large passage", by a male speaker. Here, omitting the postfilter increases the quality slightly even though GS postfilter supresses noise more.

4.1.3 Analysis

The results just mentioned do not seem very promising. Apart from the scattered observations of slight improvement, there is no strong evidence that the transcoder is beneficial. However, as mentioned earlier (Section 4.1.1) the original distortions observed (esp. in the QCELP case) is much lower than expected. This is probably due to the fact that our simulations are not a very accurate representation of reality. In reality, there are linear to μ-law (or A-law) quantizers, A/D and D/As as mentioned before (Section 3.6). These stages introduce distortion that is not modelled by our simulations. In that light, our transcoder will probably improve speech quality; though, because it is done digitally and not because of LSP translation.

4.2 Listening for Modified IRS Speech

Flat speech, as mentioned earlier, is unfiltered speech. This is generally what has been used to evaluate coders in the past. However, flat speech is not necessarily a true representation of what happens in real digital systems. The ITU has put forward recommendations for subjective performance assessments[13, 14]. These involve using what is called an Intermediate Reference System (or more recently, the Modified Interference Reference System or modIRS). We briefly describe what these are and then discuss the results of these tandemed coders in response to modIRS speech.

4.2.1 An Intermediate Reference System

From the late 1970s to the late 1980s, a specification for an Intermediate Reference System (IRS) was developed which is now ITU Recommendation P.48[13]. This intermediate reference system is used to define loudness ratings and to help standardize performance among different equipments. Of all the requirements and specifications of this IRS, we will only be concerned with the send and receive frequency characteristics. These IRS characteristics come from extensive series of measurements made on analog telephones in the early 1970s [14]. However, for the loudness balance purposes for which the IRS was designed, it was necessary to include a 300-3400Hz bandpass filter, know as the SRAEN filter [14]. These specifications, however, do not represent modern digital filters and have therefore been modified to form what is now known as the modified IRS (modIRS). The characteristics are given in Table B.1 in the Appendix and plotted in Figure 4-1 below. Tolerance on the nomimal points given in the table for send and receive are ± 2.5 dB between 200 Hz and 3400 Hz with a roll-off of at least 15 dB/octave below 200 Hz and an appropriate anti-aliasing filter above 3400 Hz.

4.2.2 QCELP/GS and G.723.1 in tandem

Now we use speech that has been filtered though a modified IRS filter similar to one described in the previous section (Section 4.2.1). When QCELP or GS are put in tandem with G.723.1, using the transcoder or otherwise, we make the striking observation that distortion is much less than in the case with flat speech. Distortion is much less in this case than in the case with flat speech!

4.2.3 Summary

Our goal was to determine which one of items (i) and (ii) below is true:

(i) LPC degradation[5] is not the main case of distortion.

(ii) LPC degradation is the main cause of degradation.

[5]By this we mean the degradation in LPCs caused by doing Levinson-Durbin on decoded speech in coder 2.

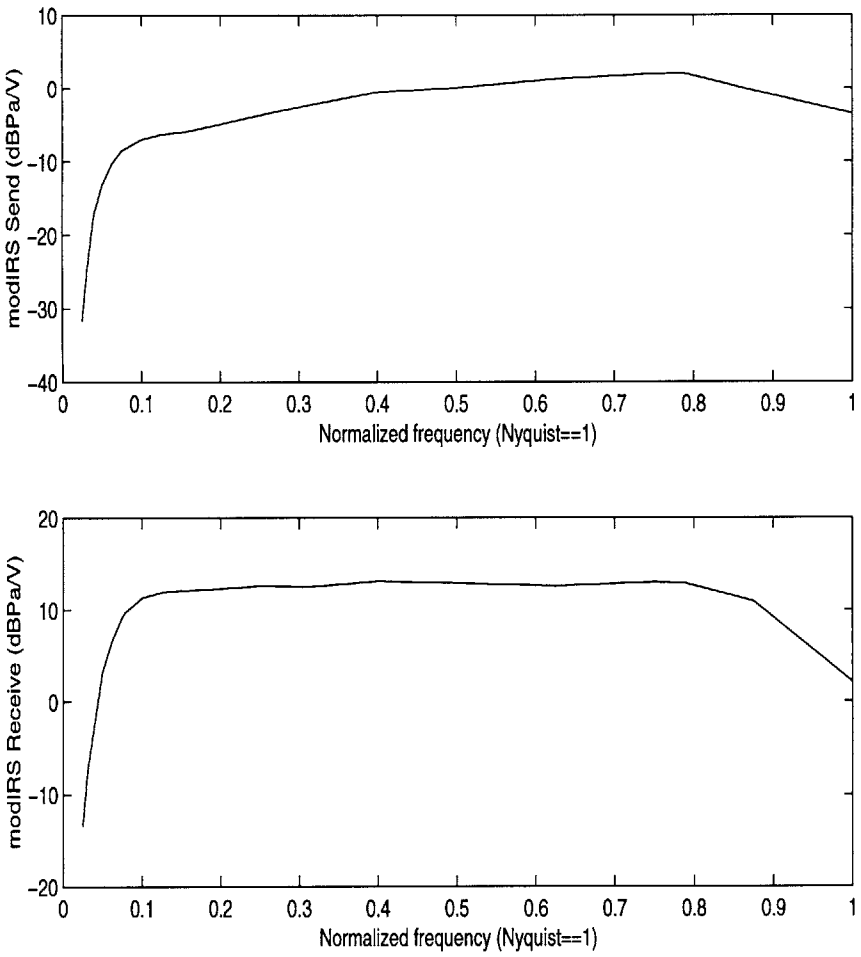

Figure 4-1: Send and Receive Characteristics for modified IRS [4][14]

If (i) is true, then we need to determine what is the main possible cause of degradation and what makes modIRS speech do much better in tandeming. If (ii) is true, for our results to be consistent, it would mean that:

- The transcoder does not improve LPC degradation.
- There is not much LPC degradation in modIRS speech.

In light of the results from this chapter, we now investigate the apparent discrepancy between flat and modIRS speech. We aim to find out which of (i) and (ii) is true and why.

Chapter 5

Further Investigation of Tandem Degradation

In this chapter we seek to determine the most probable cause of degradation and to propose possible solutions.

5.1 Is It Really the LPC?

We mentioned in the last chapter that if the LPC degradation was indeed the cause of distortion, then our transcoder must not be rectifying this LPC degradation we observe in transcoding. Indeed, we notice that if we plot the LPCs in the encoder of coder 2 for both cases, with and without transcoding, there is no systematic "improvement" in the synthesis filter[1].

However, the following experiment shows that LPC degradation is not the most likely culprit. We consider two QCELP coders in tandem. Careful listening tests of flat speech reveal some frequency distortion (loss of high frequency component) in the speech through this tandem. We then perform a similar experiment but use all the parameters from the first coder and pass them to the second coder. The second coder therefore uses the LPCs, LSPs and autocorrelation coefficients from the first coder (rather than calculating these afresh from speech) [2]. This second coder uses the speech from the first coder only for the

[1] In the sense that the synthesis filter in coder 2 approaches that of coder 1.

[2] This is done off-line. In fact, this experiment cannot be done in real-life tandem situations as one does not have access to any information from the first stage, except for the decoded speech.

purposes of comparing it to synthesized speech to find the optimal excitation parameters. Listening tests reveal that there is still some distortion in the resulting speech. If LPCs were the main cause of degradation, we would have noticed a greater reduction in distortion that is observed.

All these observations – difference between flat and modIRS results, the same coder tandem case with imported parameters – seem to suggest that there is still some other cause of distortion that is not necessarily tied to the tandem connection, but that is inherent in the coders themselves. We must remember that these coders are not "perfect coders" and therefore there will always be some sort of perceptual distortion, albeit very little and hardly detectable in some coders like QCELP. Is there some distortion beyond what should be expected given the bit rate limitation?

One major difference between flat and modIRS speech is that dynamic range in the frequency domain is much less for modIRS. This leads us to postulate that the weighting filter used in the analysis-by-synthesis stage is not sufficiently favoring high frequencies in the case of flat speech.

5.2 Is It Really The Weighting Filter?

The weighting filter for QCELP is given by:

$$W(z) = \frac{\tilde{A}(z)}{\tilde{A}(z/\gamma)} \tag{5.1}$$

Below is a picture (Figure 5-1) of the weighting filter with different values of γ for a frame of speech (the actual value used in the coder is 0.78).

As γ increases from zero to one, an all-pass filter, we notice that the tilt in the filter decreases. Smaller values of γ weigh the higher frequencies increasingly more than the lower frequencies.

Using some of these different values of γ, we plot the narrow band spectrum of a portion of speech (male voice saying, "call her on the phone") for the two stage QCELP tandem with no postfilters in both stages [3].

Figures 5-2 and 5-5 above show that there is quite a loss of information in the high

[3]So we have a fairer playing field for the different weighting filters

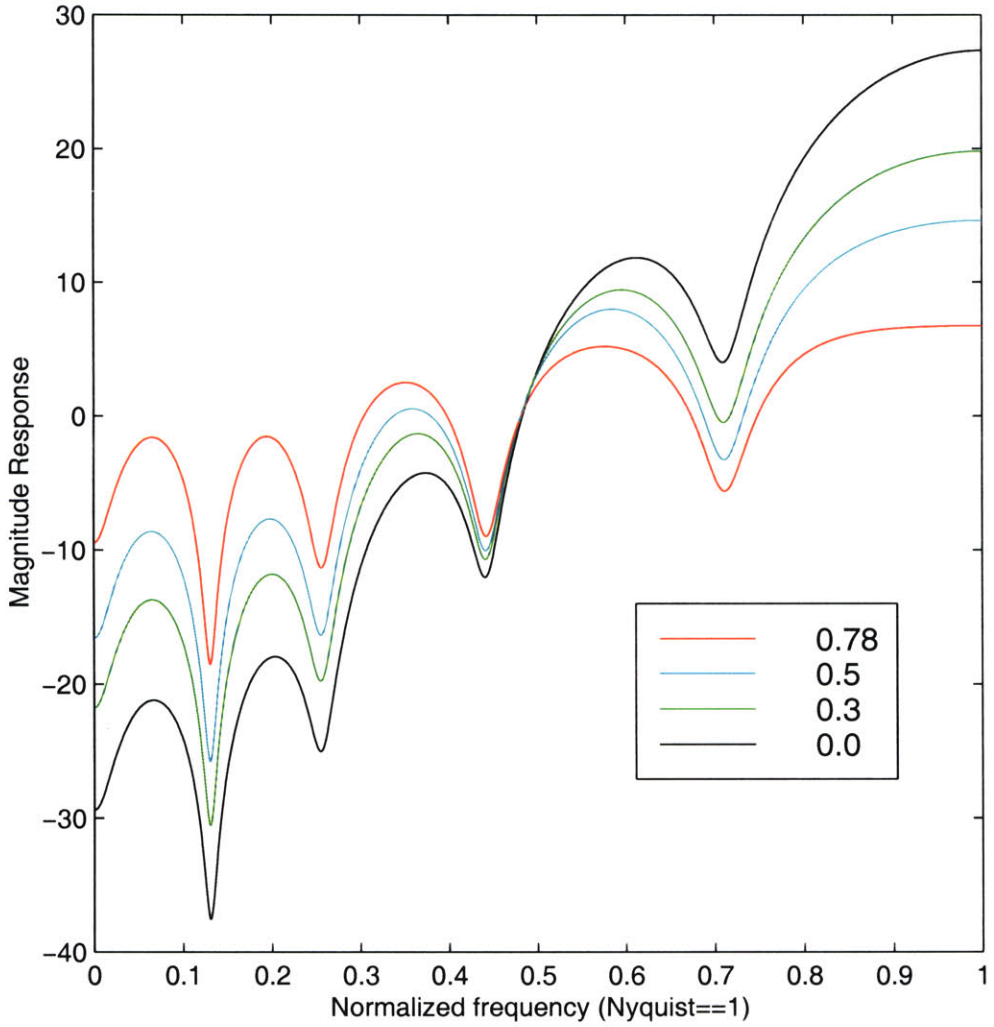

Figure 5-1: QCELP Weighting Filter for different γ

frequencies for the 2-stage tandem. One such frequency band is between 2500 Hz and 4000 Hz and between 10.05 and 10.15 secs. However, as we decrease the value of γ and thus, increase the dynamic range of the weighting filter, this loss of information is reduced greatly. Listening tests however reveal that there is more quantization noise with lower γ (expected since γ is optimized mainly to remove quantization noise). The recovery of spectral information, therefore, does not necessarily mean that our signal-to-noise ratio increased at these high frequencies. However, the noise observed seems to be more low frequency, which might suggest a improvement.

To continue with our experiments, we now do some comparisons between the weighting filters for flat speech and those for modIRS speech. In particular, we plot the QCELP

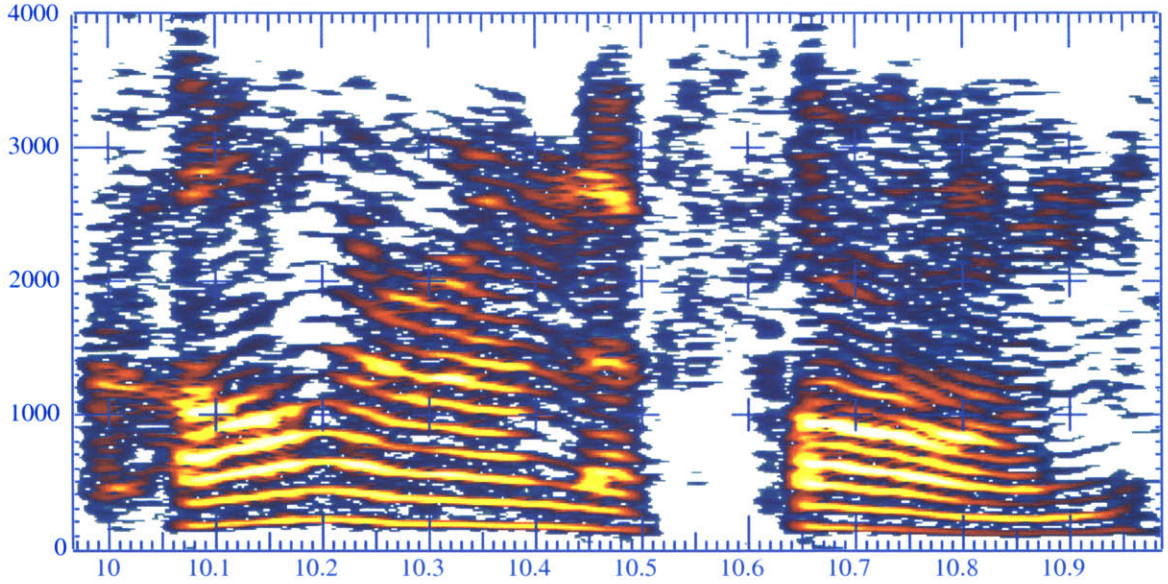

Figure 5-2: Spectrum of Speech for 2-stage QCELP Tandem with $\gamma = 0.78$

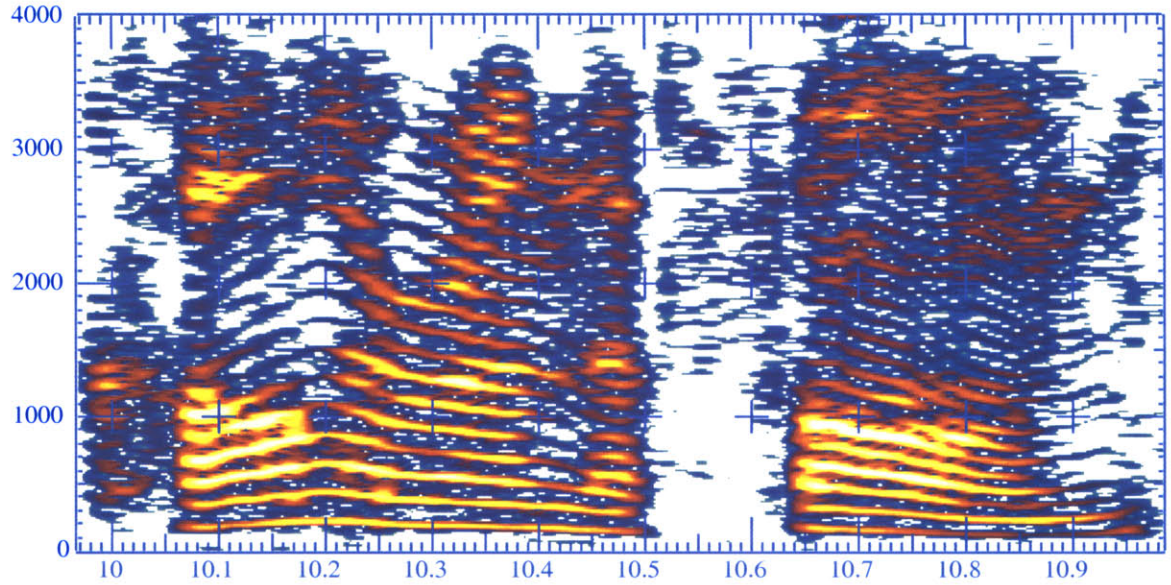

Figure 5-3: Spectrum of Speech for 2-stage QCELP Tandem with $\gamma = 0.3$

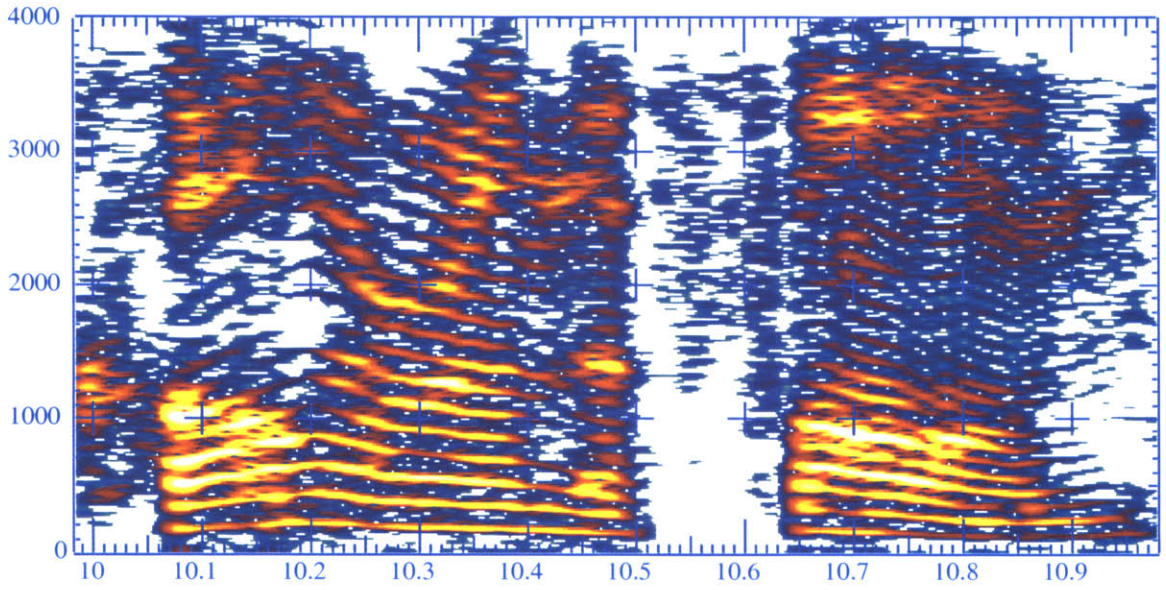

Figure 5-4: Spectrum of Speech for 2-stage QCELP Tandem with $\gamma = 0$

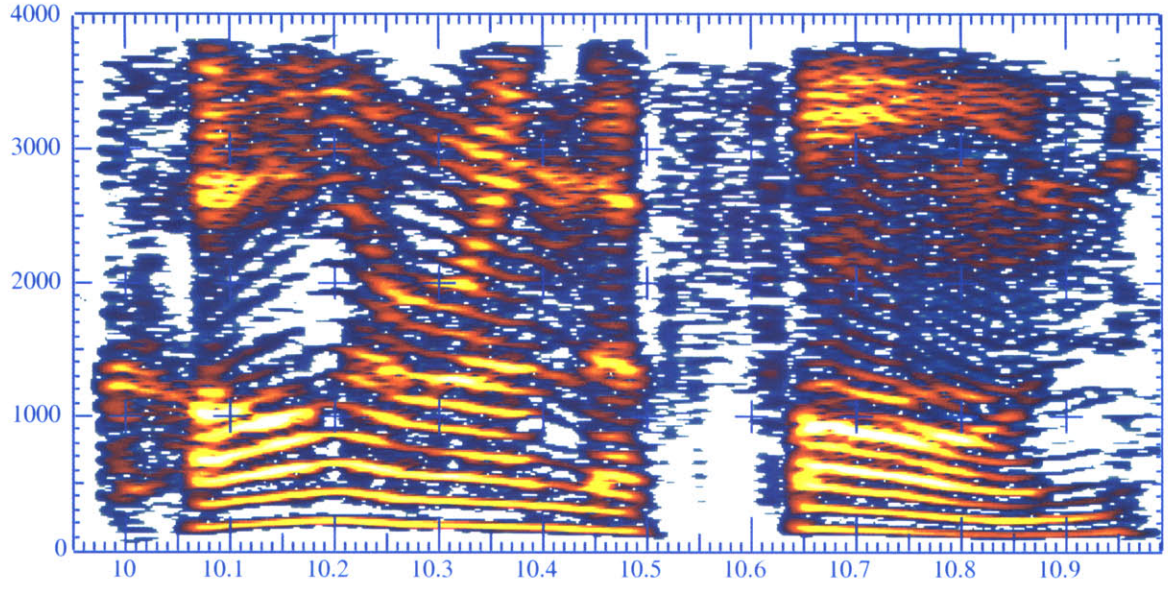

Figure 5-5: Spectrum of Speech

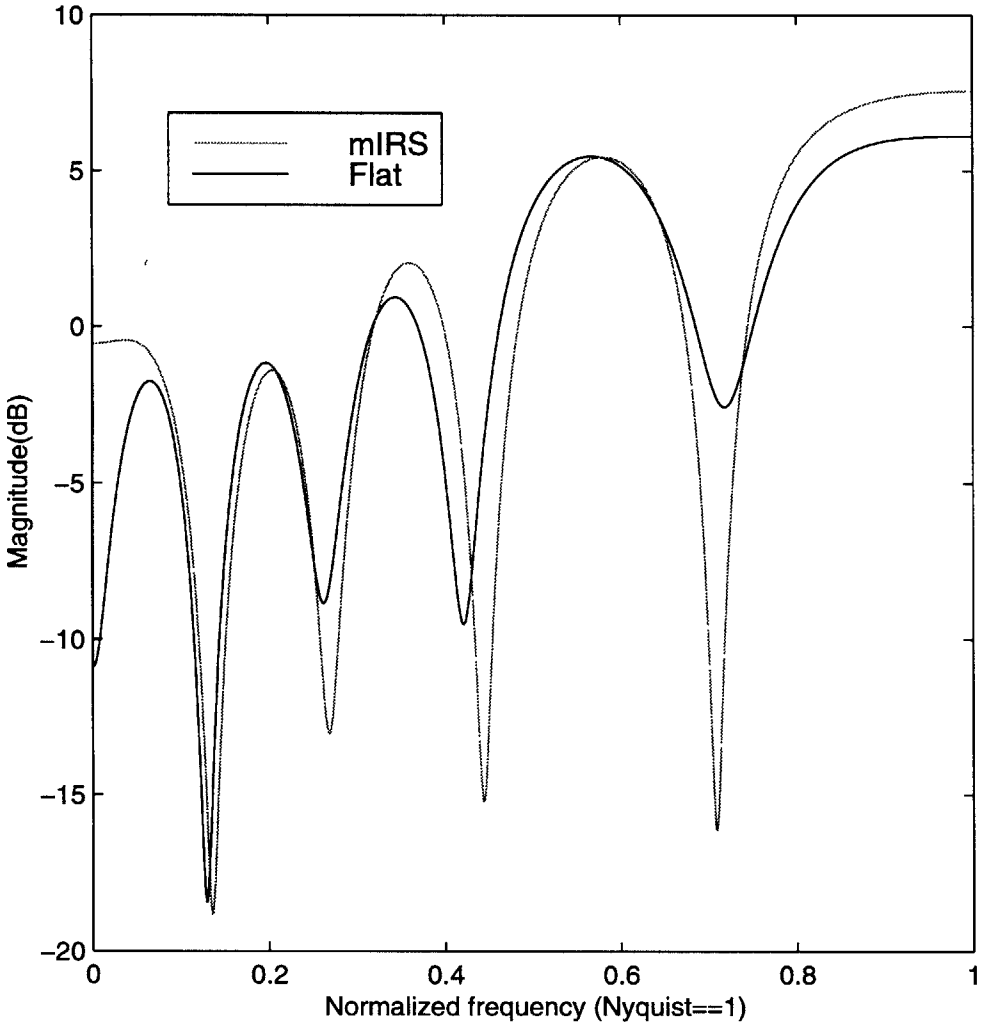

Figure 5-6: Weighting Filter for Flat and ModIRS Speech

weighting filter (with $\gamma = 0.78$) for several frames of speech. A typical plot[4] for a frame of voiced speech is shown below in Figure 5-6.

We notice from Figure 5-6 that the magnitude of the weighting filter for modIRS filtered input speech is higher than that of flat speech at very low and very high frequencies. We concentrate on the low frequency. We believe that the weighting filter for flat speech gives too much weighting to low frequencies and therefore does not leave enough bits for the higher frequencies. Below is yet another diagram (Figure 5-7) to help us see the differences between flat and modIRS weighting filters. To get these plots, we take the average of LPC filters for all frames (1,163) of a speech. Analyzing Figure 5-7 reveals that, at low

[4]All the plots do not ressemble this but the majority of the plots are similar to this one, as far as the issues being discussed are concerned.

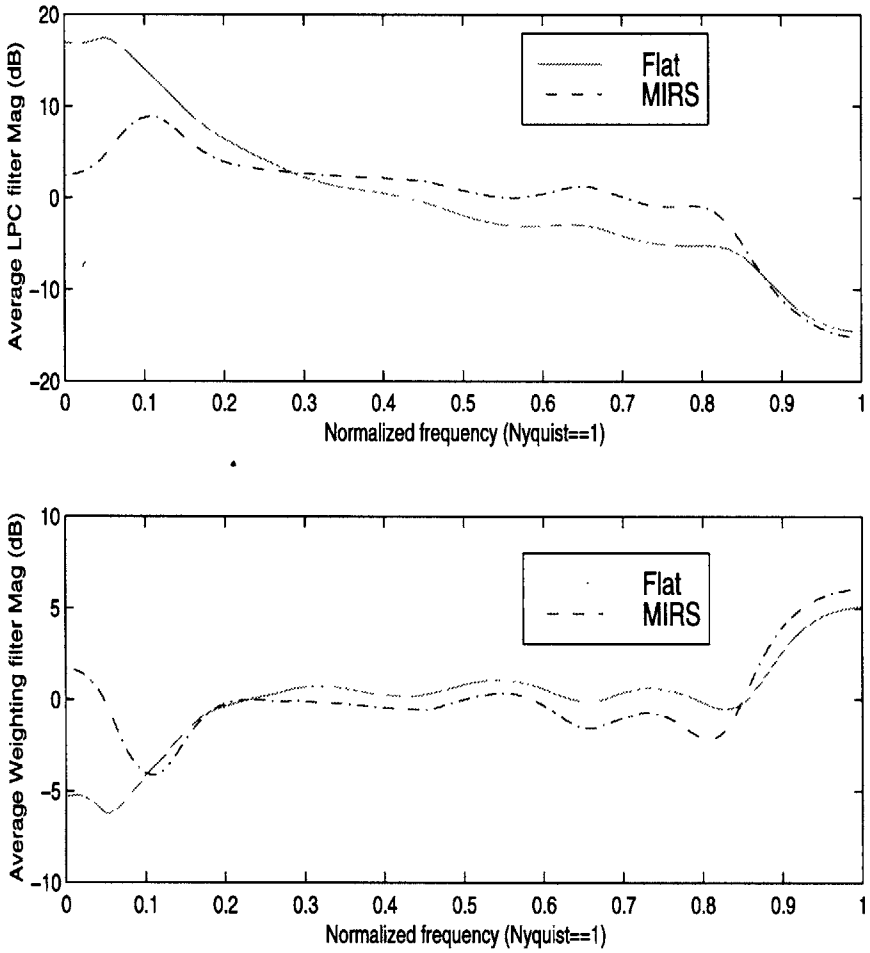

Figure 5-7: Average LPC and Weighting Filter for QCELP for A Speech File

frequencies, there is a about a 13 dB difference in the LPC filters of flat and modIRS but only about a 6 dB difference in weighting filters.

All these experiments and figures just described undergird the hypothesis that the weighting filter for QCELP is giving too much weight to the lower frequencies and not enough to higher frequencies. We do not claim that all the degradation is due to the weighting filter and thus degradation can be removed by merely "fixing" the weighting filter. We expect the bitrate limitation to have some contribution to degradation. However, an appropriate "fix" to the weighting filter should improve performance for QCELP and thus improve performance for the transcoding situation, even if it is just by a little.

5.2.1 Weighting Filter for G.723.1

For G.723.1, a pole-zero filter is used. Similar tests were performed as in the case of QCELP and the results, though somewhat positive, were not as illuminating as those of QCELP. This is because G.723.1 is a lower quality coder and is therefore very sensitive to the kind of changes we made above to the weighting filter. Quantization noise increases and makes it hard to make decisions on quality.

5.3 Proposed Solutions

Here, we discuss possible solutions to the weighting filter "problem" and evaluate some of these solutions to see if they improve performance. However, since our goal was not to do much alterations in the algorithms of the coders, we do not go into much detail.

5.3.1 First Convert to ModIRS

A simple attempt to solve the distortion problem with flat speech is to first convert the speech to modIRS, process this and then convert back to flat. We implemented this by using 20th order rational modIRS filter[5] (see Appendix D).

Listening tests for QCELP coders in tandem and for the GS and G.723.1 show that there is improvement when we first convert to modIRS and then convert back to flat speech. In speech segments where there is noticeable distortion (for e.g., the male voice saying "call her on the phone, tell her the news"), we get back the high frequencies. In some cases we get back a tad bit more high frequencies that we want[6].

However, there are drawbacks to using this method:

(i) One must detect when we have flat speech and when we have modIRS (we will discuss briefly later whether flat speech actually occurs in real life modern digital systems). There is actually no easy way of telling with accuracy whether a speech is flat or modIRS since we have speech waveforms that already have lots of high frequency components (and thus can masquerade as modIRS speech).

[5]These filters could be fine tuned more so as to give speech that is approximately centered around zero.

[6]Also, using the inverse of our modIRS filter creates a speech waveform that, even though it sounds good, does not have the natural shape of speech waveforms. Reducing the magnitude of the filter at very low frequencies helps to alleviate this effect.

(ii) Adding these conversion filters will cause delay. If the order is kept low enough, this delay should not be large.

5.3.2 Frequency-Dependent Filters

A fix for the problem with using the modIRS filter (item (i) above) to filter speech is to design a filter that is somewhat frequency dependent. In other words, if the speech has a frequency response that has too much low frequency component, then we want to reduce that and increase the high frequency component – reduce the dynamic range. We do not go into the details of how one one designs such a filter. Good filters, which have realizable approximate inverses, that are not hard to implement might be difficult to design.

5.3.3 Different Weighting Filter

Instead of the above two propositions, one could design a totally new weighting filter. The ARMA weighting filter is quite popular and does a great job at reducing quantization noise. However, it seems to be less than optimal when given speech with large low frequency energy components. Again we do not go into any involved discussions about better weighting filters. There have been some research done on using LSPs instead of LPCs for postfilters[10]. Some of this could be applied to weighting filters.

5.4 Further Discussion

Our previous discussions support the hypothesis that the weighting filters in the coders cause enough distortion in tandem coding: sufficient that the quick fix mentioned in Section 5.3.1 shows some improvement. As we mentioned earlier, our original goal was not to make changes in the algorithms of the coders[7]. Making alterations in the weighting filter could be done only in the transcoder, but that still leaves the weighting filter in encoder 1. Any improvements will therefore be reduced. Also, there are some, though not very well-defined, relationships between the weighting filters and the postfilters. Changing the weighting filter significantly in the transcoder will necessitate reoptimization of the postfilter in decoder 2 – again, meddling with the coder algorithms.

[7]Or more specifically, we do not want to change encoder 1 or decoder 2.

When we considered our original transcoder, we synthesized the decoded speech (sometimes without postfilter). Another possibility would be to only decode as far as the residual (error signal used to excite LPC synthesis filter) and use that for speech synthesis in the encoding part of the transcoder. This immediately raises issues of how to calculate autocorrelations, band energies etc.; important parameters necessary for rate decisions in our coders. However, for academic purposes, we could still investigate if using the residual improves the quality of speech coded at full rate (no rate decisions made). One big motivation for using the residual is the improvement we get in delay. This improvement could be especially big when transcoding from QCELP to G.723.1 (in 6.3 kbps mode) because of the efficient algorithms that can be used to generate transmission parameters. The results, however, were not encouraging. There was actually a detioration in quality in some cases. This serves only to reinforce our view that decoding speech all the way, with the possible exception of the postfilter, is necessary for the transcoding process.

Before we conclude, we revisit the modified IRS response. If this response describes most digital systems today, then transcoding distortions should be minimal in real life systems. Consequently, the results with flat speech would be irrelevant. Besides the fact that speech with flat response has generally been used – and is still used today – to characterize the quality of coders, there are two reasons why using flat response is still important:

- Not all systems follow modified IRS response exactly. G.723.1 is used on the internet and it is not evident how applicable modified IRS is in this case.
- Some networks apply a filter to increase the low frequency components so the response is closer to the flat response.

Chapter 6

Conclusion and Future Work

To recapitulate, this thesis sought to improve transcoding performance between QCELP and G.723.1 coders. This is done by introducing a transcoder that takes packets from the leading encoder and generates packets sent to the following decoder. In other words, the transcoder replaces the decoder of the leading coder and the encoder of the following coder. The goal was to choose a transcoder such that the quality of speech improves over the case where we just decode the speech from the first coder and pass this speech to the second coder. The resulting designs and observations are summarized in the following paragraphs.

First, previous work in quality assessment in tandem situations (which is correlated to transcoding situations) showed that there was significant distortion; even for good coders like QCELP. We were not able to reproduce this level of distortion[1]. However, this can be attributed to the fact that we do not simulate conversion to μ-law and we do not account for distortion in going to analog and back to digital (A/D and D/A). However, using Globalstar^{TM}, a lower rate coder, much like QCELP in algorithm, we could generate enough distortion, from which we could measure improvement.

Now, we could design a transcoder and measure it's performance. Because of the complexity of CELP coders today, we need to generate speech as an intermediary process in this transcoder. This decoded speech is needed for the determination of parameters that are needed for rate determination. It's other equally important use is in the analysis-by-synthesis loop needed for the generation of the excitation parameters. Our transcoder involved using the LSPs from parameters from encoder 1. We explored decoding the speech

[1]Though we did not do a MOS test to compare our results, initial listening, though quite subjective, strongly suggests that there is less distortion

in transcoder with and without the postfilter. The results for flat response, as determined by listening tests, are briefly summarized below:

- In general, quality improvement was isolated to specific phrases of speech. Overall, there was not any significant improvement.
- In cases where G.723.1 was the following coder, the choice of keeping the postfilter in the transcoder decreases the quality of speech. However, in some cases, there is a very slight increase in quantization noise.

Previous work on transcoding suggest that there is no statistical difference between flat response and IRS response in tandem situations. Recently, modified IRS (different from the IRS used in previous work) has become popular. Speech with this response showed very little degradation in transcoding. This led us to believe that the algorithms in the coders deal better with speech with modified IRS response. Experiments with the weighting filter streghtened this belief and one test that converted back flat speech first to modified IRS showed promise in terms of speech quality. However, changing the weighting filter poses problems because it requires altering the algorithms of the coders involved, as significant changes to the weighting filter will require changes to postfilter and possibly, to the codebooks used.

To conclude, this thesis has shown that in cases where there is appreciable transcoding distortion, an LSP based transcoder does not improve overall quality significantly, though it does show an improvement in end to end delay. However, appropriate changes to the algorithms of these coders can improve quality in transcoding situations.

Finally, we make suggestions for future work. The list below describes these suggestions:

- MOS tests should be done to reinforce some of the results put forth in this thesis.
- A more thorough investigation into the effects of the weighting filter and a design of better filters that improve tandem performance of flat speech.
- Other CELP coders, especially those with better tandem performances, should be studied to help shed more light on the behaviour of CELP coders in tandem.

Appendix A

A.1 Matlab Code for QCELP to G.723.1

```
function g723_tlsp = q2g_lsp(qcelp_lsp);
% This converts LSPs from QCELP to G.723.1 just as the file q2g_lpc.m
% This is the program that was used.  This is definitely not the
% most efficient way (see thesis for a more efficent way)
% We are very sloppy around the edges (beginning and ending frames).


j = 2;
i = 1;
last_value_j = 140;
while(i < length(input)*4)
value_i = i*240-30;
value_j = j*160-20;
if (value_i < value_j)
    output(i,:) = (value_i-last_value_j)*qcelp_lsp(j,:)/160+
                              (value_j-value_i)*qcelp_lsp(j-1,:)/160;
    i = i+1;
else
    if (j == length(input))
        break;
    end
    j = j+1;
    last_value_j = value_j;
end
end
g723_tlsp = round(output*32768); % These lsps are not directly used but
                                 % converted to LPCs and back to LSPs
 % as mentioned in thesis.
```

A.2 Maltab Code for G.723.1 to QCELP

```
function qcelp_tlsp = g2q_lsp(g723_lsp);
% This converts LSPs from G.723.1 to QCELP.
% This is the program that was used.  This is definitely not the
% most efficient way (see thesis for a more efficent way)
% We are very sloppy around the edges (beginning and ending frames).


j = 1;
i = 1;
last_value_j = 0;
while(i < 2*length(g723_lsp))
    value_i = i*160-20;
    value_j = j*240-30;
    if (value_i < value_j)
       if (j == 1)
          output(i,:) = input(j,:); % Set the 1st frame of QCELP
                                    % to that of G.723.1
       else
          output(i,:) = (value_i-last_value_j)*g723_lsp(j,:)/240+
                               (value_j-value_i)*g723_lsp(j-1,:)/240;
       end
       i = i+1;
    else
       if ((j == length(g723_lsp)) & (value_i ~= value_j) )
          break;
       end
       j = j+1;
       last_value_j = value_j;
    end
end
qcelp_tlsp = output/32768;
```

Appendix B

B.1 Modified IRS Filter Charateristics

Table B.1: Send and Receive Characteristics for modified IRS[14]

Frequency (Hz)	Modified IRS Send (dBV/Pa)	Modified IRS Receive (dBV/Pa)
100	-31.7	-13.4
125	-24.7	-7.4
160	-17.2	-2.4
200	-13.3	3.2
250	-10.3	6.7
300	-8.5	9.2
315	-8.3	9.7
400	-7.0	11.3
500	-6.3	11.9
600	-6.0	12.1
630	-5.9	12.1
800	-4.9	12.3
1000	-3.7	12.6
1250	-2.3	12.5
1600	-0.5	13.1
2000	0.1	12.9
2500	1.3	12.6
3000	2.0	13.0
3150	2.1	12.9
3500	-0.3	10.9
4000	-3.5	2.1
5000	-9.0	-11.7

Appendix C

C.1 Durbin's Recursive Algorithm[19]

$$
\begin{aligned}
E^{(0)} &= R(0) \\
k_i &= \left[R(i) - \sum_{j=1}^{i-1} a_j^{(j-1)} R(i-j) \right] / E^{(i-1)} \qquad 1 \le i \le 10 \\
a_i^{(i)} &= k_i \\
a_j^{(i)} &= a_j^{(i-1)} - k_i a_{i-j}^{(i-1)} \qquad 1 \le j \le i-1 \\
E^{(i)} &= (1 - k_i^2) E^{(i-1)}
\end{aligned}
$$

where i goes from 1 to 10 and the final solution is

$$a_j = a_j^{(10)} \qquad 1 \le j \le 10$$

C.2 Calculation Autocorrelation Coefficients from LPCs

The set of PARCOR (partial correlation) coefficients can be obtained from the set of LPC coefficients, a_j, using the following backward recursion[19]:

$$\begin{aligned} k_i &= a_i^{(i)} \\ a_j^{(i-1)} &= \frac{a_j^{(i)} + a_i^{(i)} a_{i-j}^{(i)}}{1 - k_i^2} \qquad 1 \le j \le i-1 \end{aligned}$$

where i goes from 10 down to 1 and we initially set

$$a_j^{(10)} = a_j \qquad 1 \le j \le 10 \tag{C.1}$$

The $a_i^{(j)}$'s and k_i's calculated above and then used to calculate the $R(i)$'s by doing part of the reverse of the Durbin algorithm. We can only calculate the $R(i)$'s relative to $R(0)$, which we set equal to 1.

$$\begin{aligned} E^{(0)} &= R(0) \\ R(i) &= E^{(i-1)} k_i + \sum_{j=1}^{i-1} a_j^{(j-1)} R(i-j) \\ E^{(i)} &= (1 - k_i^2) E^{(i-1)} \end{aligned}$$

where i goes from 1 to 10.

Appendix D

D.1 Modified IRS Filter Coefficients

$$W(z) = \frac{\sum_{n=0}^{20} b_n z^{-n}}{\sum_{n=0}^{20} a^n z^{-n}}$$

where a = {1.0000,0.1534,-0.3498,0.0333,-0.0045,-0.2782,0.0473,0.0425,
-0.0972,0.1592,-0.0440,0.2004,0.0596,-0.1521,-0.0605,-0.0036,
-0.0901,0.0417,0.0891,0.0123,0.0013}

b = {0.7438, -0.3261, -0.5755,0.2100,-0.0343,-0.2296,0.1613,0.0322,
-0.1199,0.1776,-0.0835,0.1336,-0.0131,-0.2013,0.0397,0.0432,
0.0577,0.0823,0.0654,-0.0390,-0.0088}.

References

[1] Atal, B. S. and Schroeder, M. R. Code-Excited Linear Prediction (CELP) : High-Quality Speech at Very Low Frequencies. In *Proceedings of the IEEE International Conference on Communications*, v 1, pp 937-940, March 1985.

[2] Atal, B. S., Cox, R. V. and Kroon, P. Spectral Quantization and Interpolation for CELP Coders. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, v 1, pp 69-72, May, 1989.

[3] Barnwell III, T. P. and Schafer, R. W. Improving the Performance of LPC-CVSD Tandem Connections by Phase Modification. In *IEEE International Conference on Acoutics, Speech and Signal Processing*, pp 433-436, May, 1977.

[4] Barnwell, T. P., Schafer, R. W. and Bush, A. M. Evaluation of LPC/CVSD Tandem Connections. In *3rd IEEE International Conference on Acoutic, Speech and Signal Processing*, pp 326-329, April 1978.

[5] Bergeron, L. E. Spectral Enhancement Procedure for the Wideband/Narrowband Tandem. In *3rd IEEE International Conference on Acoustics, Speech and Signal Processing*, pp 330-333, April 1978.

[6] Cheung, R. S. Application of CVSD with Delayed Decision to Narrowband/Wideband Tandem. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp 437-439, May 1977.

[7] Fischer, T. R. On the Tandem Connection of Differential Encoding Systems: The Case of Cascaded Quantizers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp 1300-1303, 1983.

[8] Gardner, William R. Sensitivity Weighted Vector Quantization of Line Spectral Pair Frequencies. Qualcomm, Inc., U.S. Patent 5,704,001, Dec 30 1997.

[9] Hangartner, R. D. and Jain, V. K. 32Kbs ADPCM/PCM Transcoder using TI-320 DSP Microprocessor. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp 1421-1424, 1985.

[10] McLoughlin, I. V. and Chance, R. J. LSP-Based Speech Modification for Intelligibility Enhancement. In *Proceedings of International Conference on Digital Signal Processing*, v2, Jul 2-4, 1997.

[11] ITU-T Recommendation G.723.1: C Reference Code, Test Signals and Test Sequences for the Fixed Point 5.3 and 6.3 kbits/s Dual Rate Speech Coder. ITU, March 1996.

[12] ITU-T Recommendation G.723.1: Dual Rate Speech Coder for Multimedia Communications at 5.3 and 6.3 kbit/s. ITU, March 1996.

[13] ITU-T Recommendation P.48: Specification for an Intermediate Reference System. ITU, 1988.

[14] ITU-T Recommendation P.830: Subjective Performance Assessment of Telephone-Band and Wideband Digital Codecs. ITU, Feb, 1996.

[15] McVay, J. W. and Gibson, J. D. Analyses and Experiments on the DM into LPC Tandem. In *IEEE Transactions on Acoustic, Speech and Signal Processing* v ASSP n3, pp 436-439, June 1982.

[16] Nishitani, T. Tandem Transcoding without Distortion Accumulation. In *IEEE Transactions on Communications* v COM-34 n 3, pp 278-284, March 1986.

[17] O'Shaughnessy, D. Speech Communication: Human and Machine. Addison-Wesley, 1987.

[18] Perkins, Mark E. Test Results: CDMA Coder Assessment. AT&T Bell Laboratories, August 16, 1995.

[19] Rabiner, Lawrence R. and Schafer, Ronald W. Digital Processing of Speech Signals. Prentice-Hall, NJ, 1978.

[20] Ramamoorthy, V. On Tandem Coding of Speech. In *Proceedings of the IEEE International Conference on Communications*, pp 1229-1233, 1995

[21] Schroeder, M. R. and Atal, B. S. In *Proceedings of IEEE International Conference on Communications*, pp 48.1, May, 1984.

[22] Soong, Frank K., Juang, Bing-Hwang. In *International Conference on Acoustics, Speech and Signal Processing*, pp 1.10.1-1.10.4, 1984.

[23] TR-45 Committee. TIA/EIA/IS-733: High Rate Speech Service Option 17 for Wideband Spread Spectrum Communication Systems. TIA, March, 1998.

[24] Wang, B. and He, Y. Adaptive Postfilter in 16kbps LD-CELP Speech Coder. In *International Conference on Signal Proceedings*, v 1, pp 678-681, Oct 14-18, 1996.

[25] Yang, J., Tang, K., Feng, C., Zhang X. and Ling, N. Code conversion algorithm between 16kbps CVSD and 64kpbs PCM. In *Tien Tzu Hsueh Pao/Acta Electronica Sinica* v 22 n 4, pp 72-75, April 1994.