# A COMPARISON OF ADAPTIVE PREDICTORS IN SUB-BAND CODING

by

PAUL NING

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

Master of Science

and

Bachelor of Science

at the

Massachusetts Institute of Technology

September 1987

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
August 7, 1987

Certified by [illegible]
Dennis Klatt, Senior Research Scientist
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith, Chairman
Department of Electrical Engineering and Computer Science

# A COMPARISON OF ADAPTIVE PREDICTORS IN SUB-BAND CODING

by

PAUL NING

Submitted to the Department of
Electrical Engineering and Computer Science
on August 7, 1987 in Partial Fulfillment
of the Requirements for the Degrees of
Master of Science and Bachelor of Science

**ABSTRACT**

Adaptive Differential Pulse Code Modulation (ADPCM) and Sub-Band Coding (SBC) are two efficient medium rate speech coding schemes. This study investigates the benefits of applying the adaptive predictive techniques of ADPCM to the encoding of subbands in SBC. The performances of two well-known predictors, the least-mean-square transversal and the least-squares lattice, are compared in the context of an SBC-ADPCM system.

A floating point simulation is written for the sub-band coder, which includes Quadrature Mirror Filters, dynamic bit allocation, an adaptive quantizer, and adaptive predictors. Both predictive and non-predictive versions of the coder at 16 kbps and 24 kbps are used to process four phonetically balanced sentences spoken by two males and two females. At each bit rate, the voice qualities of the non-predictive, least-mean-square, and least-squares coders are judged with objective signal-to-noise ratio measures as well as subjective listening tests.

The simulation results indicate that both predictors exhibit similar behavior in tracking the subbands. However, overall signal-to-noise ratio improvements due to prediction are very small, typically less than 1 dB. Listening tests also support the conclusion that sub-band coding quality is not significantly enhanced by either the least-mean-square or least-squares algorithms.

Thesis Supervisor: Dennis Klatt
Title: Senior Research Scientist
Department of Electrical Engineering
and Computer Science

## ACKNOWLEDGEMENTS

I would like to thank ROLM Corporation for supporting this research and providing a "Great Place to Work" for several rewarding co-op assignments. In particular, I am grateful to the members of my group, Tho Autruong, Kirk Johnson, Dan Lai, Mike Locke, and our manager, Ben Wilbanks, for their helpful comments during the course of this work.

I would also like to thank Dennis Klatt of MIT's Department of Electrical Engineering and Computer Science for supervising this study.

Most importantly, this thesis is dedicated to my parents and my brother for their love and encouragement throughout my life.

TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1 - INTRODUCTION

The advent of modern, powerful computers has made possible the practical implementation of a wide variety of digital signal processing techniques. One of the many areas of application for this technology is speech. Speech enhancement algorithms can improve voice signals degraded by noise and help to remove echo effects. Methods for time scale modification of speech enable recorded sentences to be played back at variable rates while maintaining intelligibility. Speech synthesis techniques allow machines to talk and speech recognition schemes enable them to identify spoken words. Speech coders, in general, convert one representation of speech into another. Compression algorithms are coders which aim to reduce storage requirements or lower transmission rates.

This thesis investigates a particular type of digital speech compression method. In particular, our goal is to evaluate and compare the performances of two adaptive predictors when used in a sub-band coder. We will implement different versions of the sub-band coder, process several sentences with them, and perform objective and subjective tests on the results.

Sub-band coding [1,2] is a recently developed frequency domain method for compressing speech. Adaptive prediction [3], which refers to the estimation of current signal values based on past information, is another technique used for speech compression. In this thesis, both methods are combined in the same coder. The details of our coder will be presented in Chapter 2. As an introduction, we will briefly review some of the fundamentals of digital speech processing.

## 1.1 BACKGROUND

### 1.1.1 Digital Speech

Speech signals are inherently analog i.e. continuous in both time and amplitude (Fig. 1-1). However, there are many advantages to representing such a waveform as a sequence of binary digits (bits), each of which can take on only two values. Such a signal is known as digital and is the form of data used in all modern computers. Since bits are restricted to just two discrete levels, digital signals degraded by additive noise may be exactly recovered if the noise magnitude is below a certain threshold. Analog representations do not have this property. Digital data is also easily stored, generated, and manipulated.

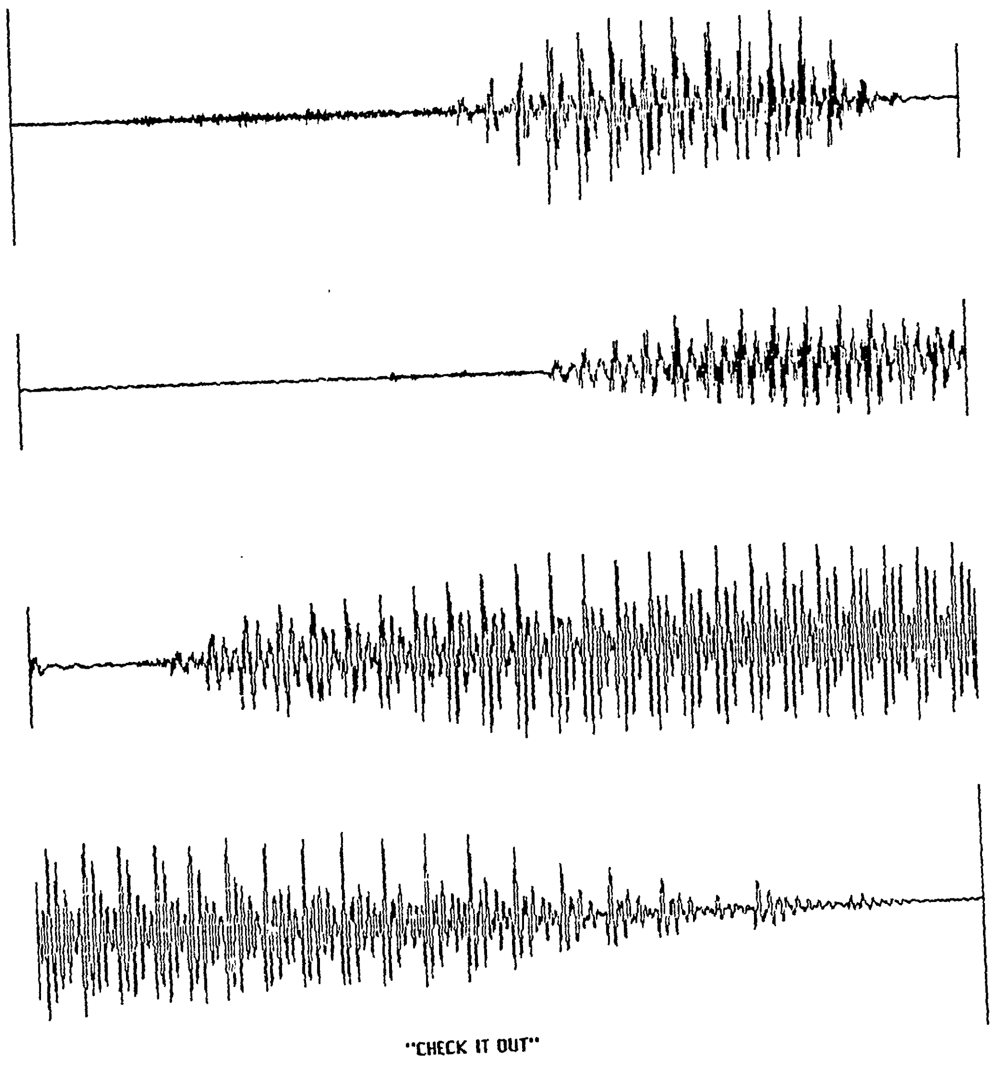
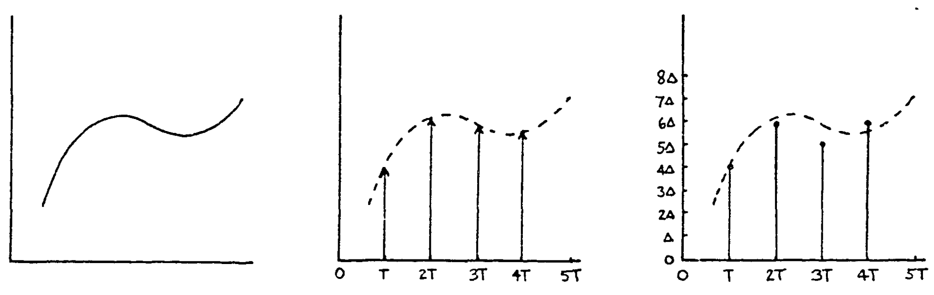


Figure 1-1. Speech Waveform

There are two distinct operations involved in digitizing a waveform - sampling and quantizing (Fig. 1-2). Sampling refers to the measurement of a signal only at specific time instances. These time instances are typically chosen to be multiples of a fundamental sampling period, T. Waveforms which have been sampled are known as discrete-time signals. Quantization constrains the amplitude of the signal to only a finite number of values. A discrete-time waveform which has also been quantized is digital.



Analog Discrete-Time (sampled) Digital (sampled and quantized)

Figure 1-2. Types of Signals

A desired feature of the analog-to-digital conversion process is that it preserves the information in the original signal i.e. one can recover the analog waveform from the digital data alone. This is, of course, not true in general since sampling throws out all values between samples and quantizing reduces the accuracy of the samples that are saved. However, it has been proven that under certain conditions the analog signal is nearly completely recoverable from the digital samples.

This is a fundamental result of signal processing theory and is known as the sampling theorem. It states that any bandlimited signal (no frequency components higher than some B Hz) can be reconstructed from periodic samples alone provided that the sampling rate is at least twice B. If the sampling rate is too low, the reconstructed output will be characterized by what is known as 'aliasing' distortion. In practice, sampling of an analog signal is preceded by a lowpass filter stage which effectively constrains the bandwidth B of the analog signal to half of the sampling rate. This stage is sometimes referred to as an anti-aliasing filter. The quantizing of the sample values introduces an approximation error which can be made as small as desired by increasing the number of quantizer levels.

For a complete and rigorous exposition of the sampling theorem and digital signal processing fundamentals, see [4] or [5].

### 1.1.2 Pulse Code Modulation

The sampling theorem guarantees that if an analog signal is sampled often enough, the samples alone are sufficient to reconstruct the signal. However, there are many ways to digitally code the samples [6]. The first and most straightforward is Pulse Code Modulation (PCM) [7], which is simply an amplitude quantizer.

In PCM, a binary code is assigned to each quantization level. Input samples are identified with one of these levels and represented by the corresponding sequence of bits. In this way, successive samples are translated to a stream of 0's and 1's. Figure 1-3 shows a three bit quantizer that distinguishes between eight different amplitude levels.

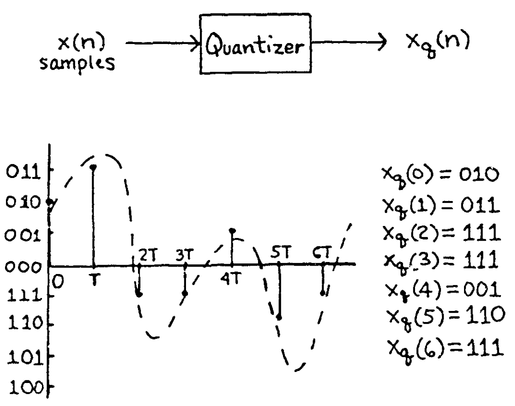


Figure 1-3. PCM

The basic quantizer has levels which are equally spaced (linear) and constant with time (fixed). The distance between adjacent levels is known as the step size. Hence, linear, fixed quantizers are those with uniform, constant step sizes. Quantizers may also be classified (Fig. 1-4) based upon whether zero is a valid level (mid-tread) or is halfway between levels (mid-rise).

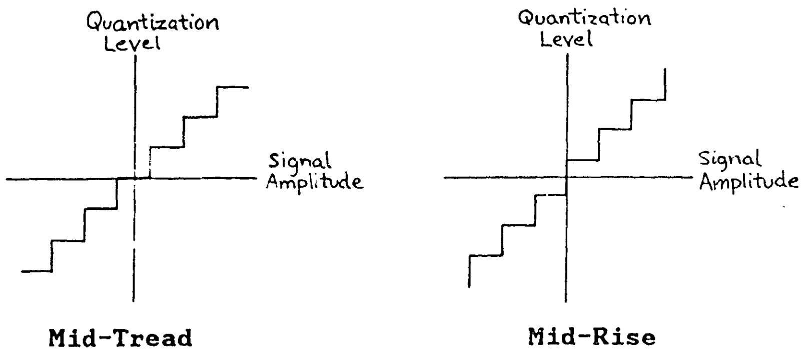
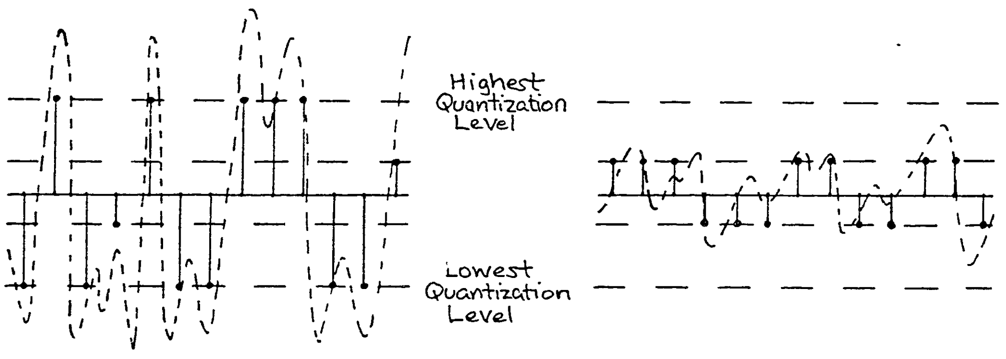


Figure 1-4. Mid-Tread and Mid-Rise Quantizers

Approximating the continuous amplitude range of a signal with a set of discrete levels can result in two types of errors - clipping and rounding (Fig. 1-5). Clipping occurs when the input falls outside the range of the highest and lowest quantizer levels. Rounding takes place when the signal falls between two levels (rounding error is sometimes referred to as granular noise).



Clipping Rounding

Figure 1-5. Quantizer Noise

PCM performance can be improved by designing quantizers which are adaptive (not fixed) or non-linear. Adaptive quantizers [8,9] are designed to take advantage of changes in signal variance over different segments of speech (the short-term statistics of speech are said to be non-stationary). In particular, step sizes are allowed to adjust with time in order to minimize clipping and rounding errors. To reduce the incidence of clipping, the step size of the quantizer should increase with the variance of the input. Likewise, granular noise can be lessened by decreasing the step size for smaller short-term variances. In effect, the quantizer adapts itself to match the input range.

Non-linear quantizers are commonly used in telephony applications. The so-called µ-law curve [6] represents a combination of linear and logarithmic relationships between quantization level and signal amplitude (Fig. 1-6). The motivation for this is based on the fact that a given amount of noise is less disturbing if the original signal level is high. Thus, for larger inputs, we may allow greater quantization noise, so we space the levels further apart. Correspondingly, for low signal amplitudes the step size is smaller to reduce the quantization noise. In terms of signal-to-(quantizing)-noise ratio or SNR, to be defined shortly, the µ-law curve maintains a relatively constant SNR over a wider range of input amplitudes, thereby increasing the dynamic range of the quantizer.

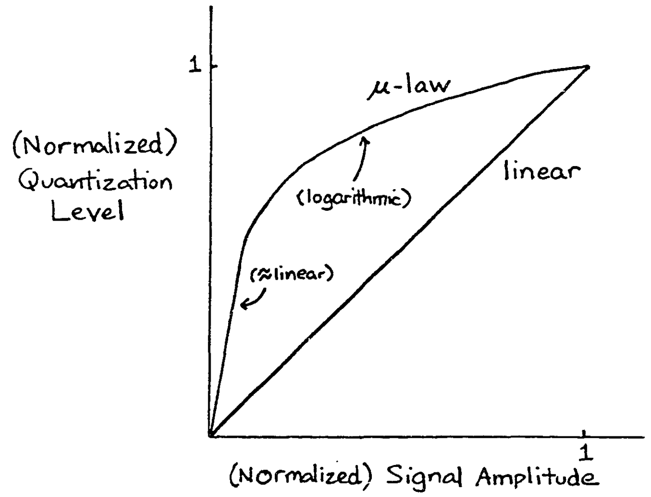


Figure 1-6. μ-law vs. linear

### 1.1.3 Coder Performance Measures

The voice quality that a speech coder produces must ultimately be judged by human listeners. However, it is useful to have some objective measures of coder performance for standardized comparisons.

The most common yardstick is the signal-to-noise ratio mentioned earlier. If x(n) is the input to a coder and y(n) is the output of the decoder, then

$$\text{SNR} = 10\log_{10} \frac{\langle x^2(n)\rangle}{\langle (x(n)-y(n))^2\rangle} \tag{1-1}$$

where <> denotes time average or expected value. The numerator and the denominator represent the powers of the original signal and coding error signal, respectively. The logarithm converts the power ratio to a decibel (dB) scale. In the case of many coders, including PCM, the error is simply the quantization noise (sub-band coders, as we shall see, also have error contributions from other sources).

Another measure of coder quality is the segmental SNR, or SSNR. The speech input is divided into contiguous segments, typically about 16 ms long [6], and a standard SNR (dB) is computed for each segment. The individual SNR's are then averaged to obtain SSNR

$$\text{SSNR} = (1/M) \sum_{i=1}^{M} \text{SNR}(i) \text{ (dB)} \tag{1-2}$$

where M is the total number of segments in the signal. The segmental SNR was developed because the standard SNR tends to obscure poor performance in segments with low input power. By averaging dB values, weak segments have a greater effect on the overall measure.

In addition to voice quality, we are also interested in a coder's information rate, or bandwidth, as measured by its bit rate. The bit rate of a coder is the number of bits per second needed to represent the input signal. In the case of PCM, the bit rate is equal to the sampling rate times the number of bits per sample used by the quantizer.

Finally, we should consider the computational requirements for the implementation of the coding and decoding algorithms. This is especially important in real-time (on-line) applications (i.e. where the processing is done as quickly as the input is received or the output needed).

The goal of speech compression algorithms is to code speech at a low bit rate and minimum computational cost while maintaining good voice quality. Of course, what is defined as 'good' is dependent upon the particular application of the coder. As to be expected, achieving this aim involves a compromise. For any particular coding scheme, the voice quality tends to decrease as the bit rate is lowered. Different algorithms can provide many levels of voice quality for the same bit rate, but better fidelity sometimes comes at the expense of higher coder complexity.

In standard telephony applications, an 8 kHz, 8 bit $\mu$-law PCM system is commonly used (the spectrum of a typical telephony channel ranges from 300 Hz to 3200 Hz so the conditions of the sampling theorem are satisfied). This yields a bit rate of 64 kbps. By using more elaborate digital coding schemes, however, significant compression below 64 kbps is possible.

### 1.1.4 Differential Pulse Code Modulation

In order to compress speech and still maintain voice quality, we must get more out of our bit resources. Adaptive quantization, for example, takes advantage of time-varying speech variance to reduce coding noise. Another important property of speech which can be exploited is correlation between samples. In other words, voice signals contain much redundancy. As a result, we can try to predict sample values by considering past data alone. Prediction is the essential concept used in Differential Pulse Code Modulation (DPCM) [10,11].

Let's see how prediction can improve coder performance. For a given n-bit quantizer and input signal variance, a good quantizer is one whose $2^n$ levels match the range of the input. If the input signal variance is reduced, the levels can be compressed, resulting in lower absolute quantization noise for the same number of bits (Fig. 1-7). DPCM exploits this idea by quantizing not the original signal but the difference between the coder input and a prediction signal generated based upon past samples. If the predictor is successful, the difference signal will have a smaller variance than the input and can be quantized with less noise. This translates to better voice quality for the same bit rate, or alternatively, comparable voice quality at a lower bit rate.

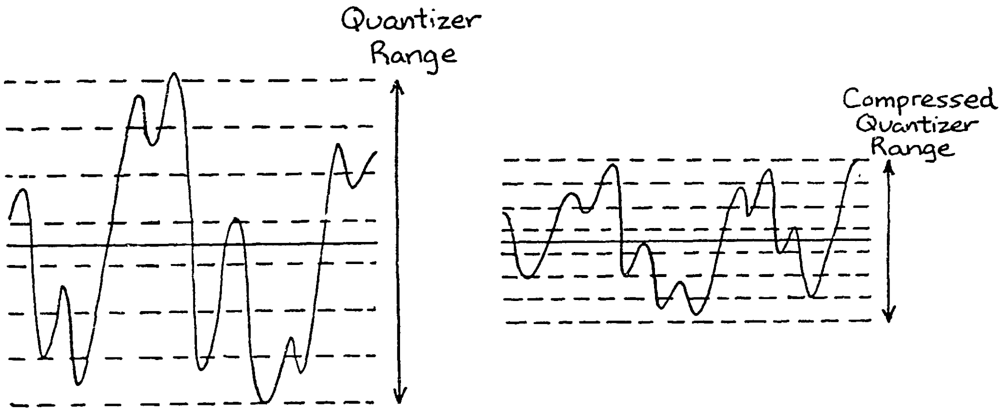


Figure 1-7. Reduced Quantization Noise With Smaller Signal Variance

Fig. 1-8 is a block diagram of the DPCM coder and decoder. The system has three main components - a quantizer which generates the pulse code, an inverse quantizer which converts the pulse code to a level, and a predictor which forms an estimate of the current input sample based on past data. The coder and decoder both use the same types of predictor and inverse quantizer so that their computations are consistent. Notice that the coder input is no longer fed directly into the quantizer as in plain PCM. Instead an error or difference signal is generated by subtracting the predicted value from the input.

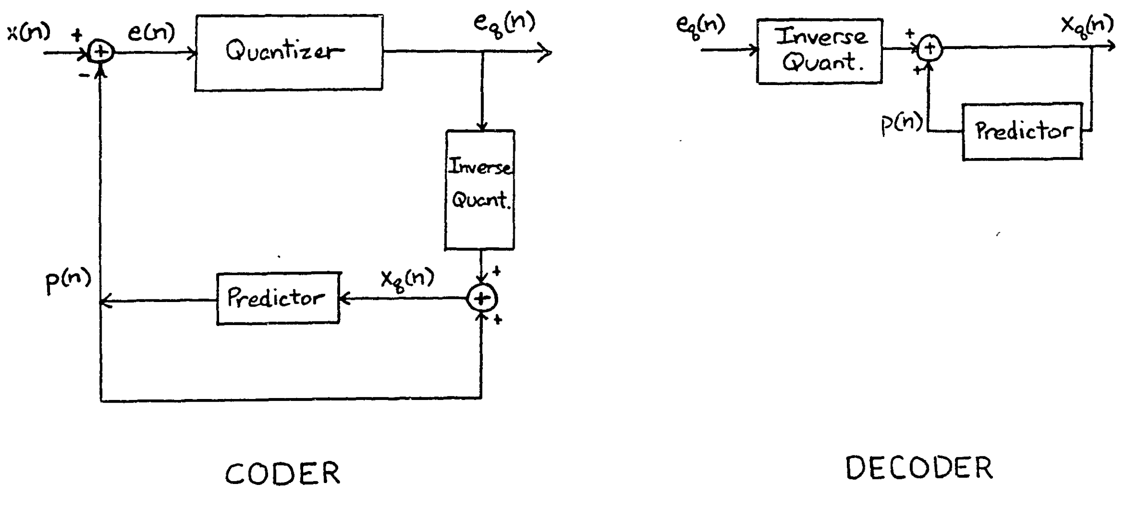


Figure 1-8. DPCM

It is instructive to note that the input to the predictor is not the coder input, x(n), but an approximate version, $x_q(n)$ , that is reconstructed from the predictor output and an error signal that has been degraded by quantization. This is best explained by considering the design of the decoder portion of the DPCM system. Since the decoder does not have access to the original error signal but only to a quantized version of it, the output of the decoder is only an approximation to the coder input. Consequently, the decoder's predictor cannot be provided with the original signal (indeed, if the decoder had x(n), we wouldn't need the coder). Moreover, we want the outputs of the two predictors to be identical. Therefore, we must also use quantized inputs to the coder's predictor.

The design of the predictor is obviously important to the success of any DPCM system. A linear Nth order predictor computes the estimate as a weighted sum of the past N samples

$$p(n) = \sum_{i=1}^{N} a_i x_q(n-i) \tag{1-3}$$

where $a_i$ is the $i^{th}$ weight and $x_q(n-i)$ is a delayed and quantized version of the input. A direct implementation of this uses a tapped delay line, or transversal, structure (Fig. 1-9) [12]. Another possible implementation uses a ladder, or lattice, configuration (Fig. 1-10) [13,14]. In the lattice, the coefficients $a_i$ do not explicitly appear as multipliers. However, the net effect of the computation is still a linear combination of the past inputs.

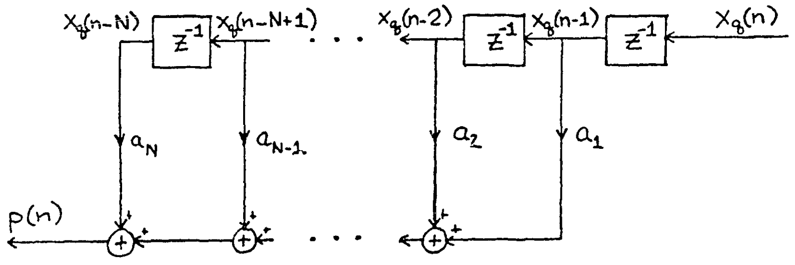


Figure 1-9. Transversal Predictor

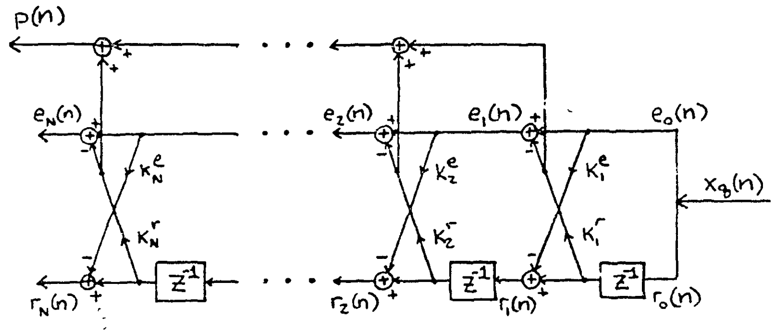


Figure 1-10. Lattice Predictor

Just as quantizers can be made to adapt to their input, predictor parameters can as well. Again, the motivation for this is that short-term speech statistics are not stationary. A given set of predictor coefficients may work adequately for some speech segments but poorly for others. A DPCM system with an adaptive predictor is known as ADPCM [3,15]. In general, an ADPCM system has both a variable quantizer and a variable predictor.

Predictor adaptation algorithms are based upon minimizing some error criterion. The two most commonly used are the least-mean-square (LMS) [16,17] and least-squares (LS) [18]

$$(1-4) \qquad \text{LMS} : E[e^2(n)] = \lim_{T\to\infty} (T+1)^{-1} \sum_{n=0}^{T} e^2(n)$$

$$(1-5) \qquad \text{LS} : \sum_{n=0}^{T} w^{T-n} e^2(n) \qquad 0 < w < 1$$

where

$$(1-6) \qquad e(n) = x(n) - p(n) = x(n) - \sum_{i=1}^{N} a_i x_q(n-i)$$

is the prediction error. The LMS update algorithm attempts to minimize the expected value of the squared prediction error. The LS method seeks to minimize an exponentially weighted sum of squared errors.

Adaptive predictors are characterized by both their implementation structure and coefficient update scheme. Thus, we may have LMS transversal, LMS lattice, LS transversal, and LS lattice predictors. They may also be classified as either forward or backward adaptive.

Forward adaptation uses actual coder inputs and is typically done on a block basis. Unquantized inputs are buffered by the coder and new coefficients are computed for the entire block of samples. The samples are then differentially quantized using these coefficients. This is done for each block of inputs. Since the predictor in the decoder only has access to quantized data, identical adaptation is possible only if the new coefficients are explicitly sent to the decoder. This adds some overhead to the bit rate needed for the coded data itself.

Backward adaptation depends only on quantized samples that are also available at the decoder, so no overhead transmission is required. Furthermore, coefficient adaptation is done on a sample by sample basis, thereby avoiding the inherent delays of input buffering. The disadvantage of backward adaptation is that it is based on reconstructed or quantized data instead of actual samples. It was found in [19] that forward predictors outperform backward predictors at the same bit rate if parameter overhead is ignored. However, when the transmission cost of the forward predictor's side information is considered, the opposite conclusion is reached.

In this study, we will implement backward adaptive least-mean-square transversal and least-squares lattice predictors. Details of their design, including coefficient update equations, will be provided in Chapter 2.

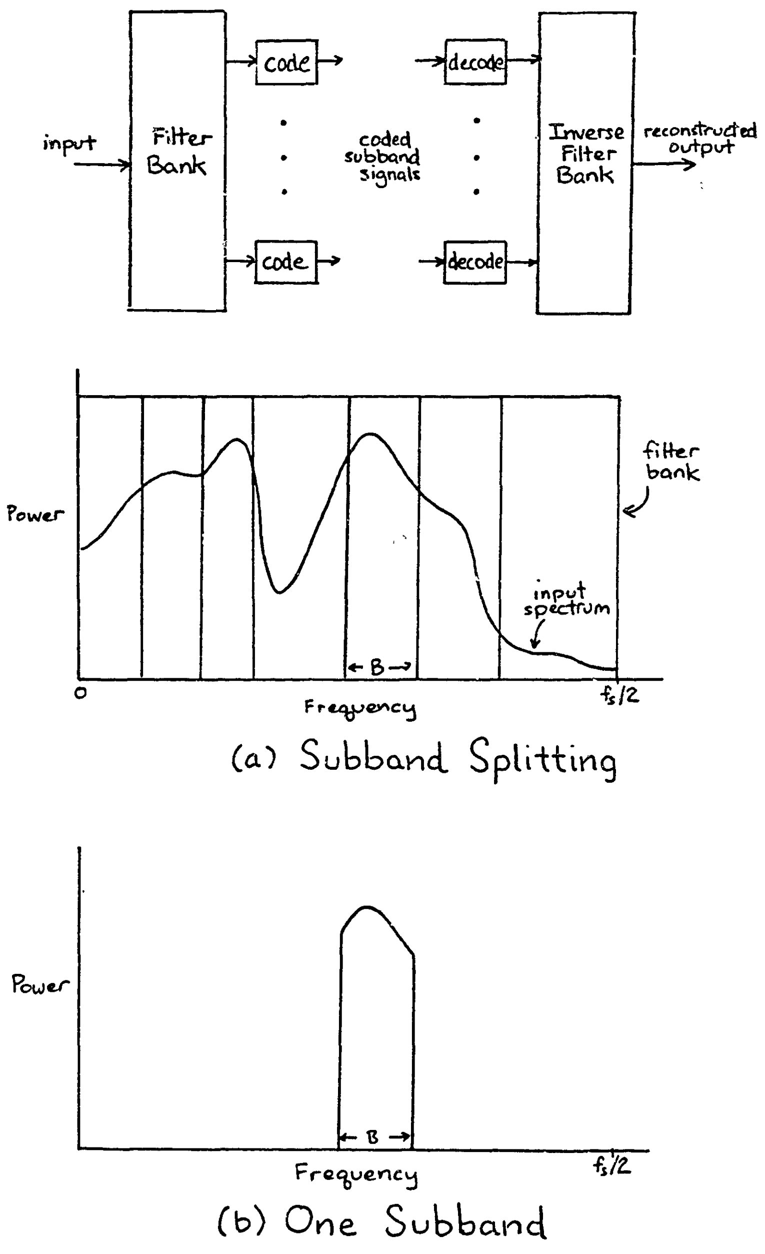
### 1.1.5 Sub-Band Coding

DPCM utilizes the redundancy of a speech signal to achieve compression. Another important characteristic of voice is that it does not have equal power at all frequencies. This fact is exploited by sub-band coding.

Originally developed in [1] and [2], sub-band coding uses a bank of bandpass filters to separate the voice signal into several frequency bands which are then individually coded (Fig. 1-11a). The signal is reconstructed by decoding each subband and passing the results through an inverse filter bank. There are two advantages to coding bands separately. Since some of the subband signals will have more power than others and are therefore more important to the overall speech quality, they can be assigned more quantization bits. Also, any quantization noise in one band does not affect the coding of others.

At first, it may seem counterproductive to split a signal into many bands before coding. After all, the more bands there are, the more samples there are to code. This is true if we assume the same sampling rate for the subbands as for the full spectrum input. In practice, this is not the case. Since each of the subbands has a smaller bandwidth than the original signal, we can sample it at a lower rate (this is known as sub-sampling or decimation [4]). For example, a decimation by 2 would mean every other output of each bandpass filter is ignored. If a subband has bandwidth B, we need only sample at 2B in order to capture all of its information. But this is strictly true only for baseband (0 Hz to B Hz) signals. Subbands are usually located at higher frequencies (Fig. 1-11b). So before we sample at 2B we must first modulate the signal down to the baseband. This can be done by multiplying the subband signal with a sinewave at an appropriate frequency.

An elegant way to obviate the modulation step is to choose the bandpass filters so that they are all of the same bandwidth B and have low and high frequency cutoffs that are integral multiples of B (Fig. 1-12). The benefits of this 'integer-band sampling' are presented in [1]. Without modulation down to the baseband, the $m^{th}$ subband, which has power from (m-1)B to mB, should normally be sampled at at least 2mB Hz in order to avoid aliasing effects. However, due to the design of the integer bands, the aliasing caused by decimation to the desired sampling rate of 2B Hz is actually advantageous (Fig. 1-13). The decimation step implicitly modulates each subband to 0 to B Hz without overlapping the aliased copies of the spectrum. Therefore, the decimated output of each bandpass filter is a baseband signal ready for coding. For a bank of k integer bandpass filters spanning 0 to $f_s/2$ Hz, where $f_s$ is the input sampling frequency, each subband has a bandwidth of $f_s/(2k)$ and can be decimated by a factor of k. Since the input has been converted into k subbands each of which has 1/k as many samples as the input, the total number of samples that need to be coded remains the same.



(a) Subband Splitting

(b) One Subband

Figure 1-11. Sub-Band Coding

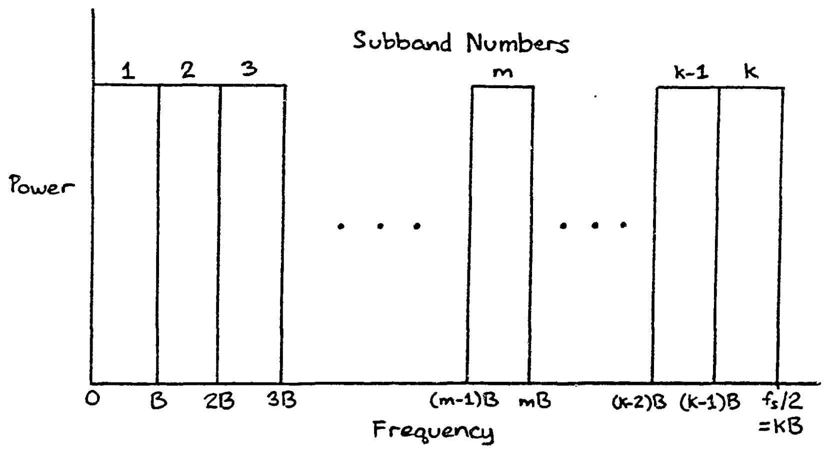


Figure 1-12. Integer-Band Sampling

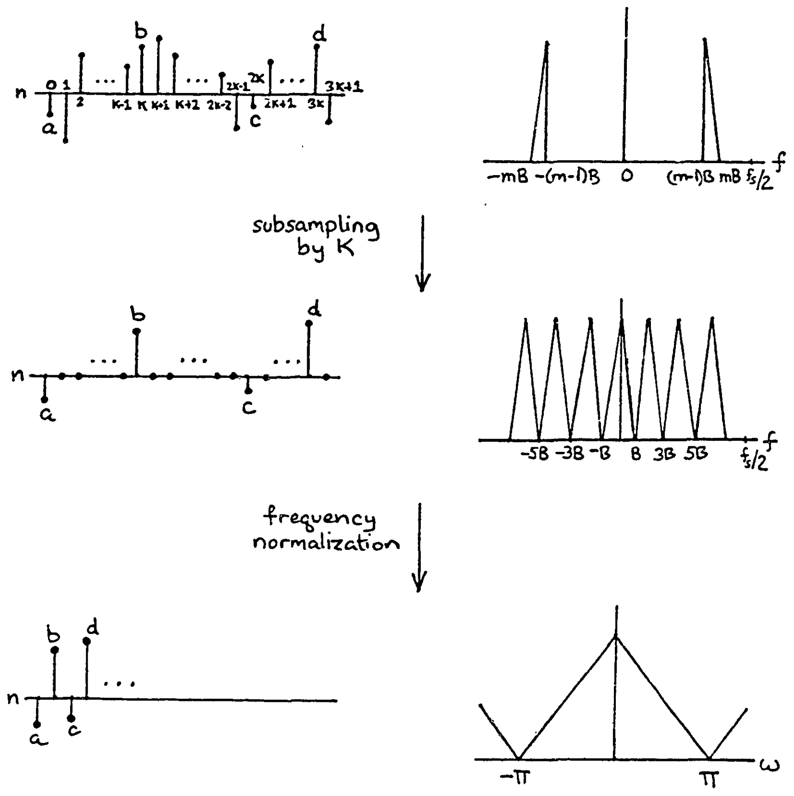


Figure 1-13. Decimation of a Subband

At the receiver end, each subband signal is decoded, upsampled to the original sampling rate, and interpolated (Fig. 1-14). This is implemented by putting k-1 zeros between every pair of decoded samples. The resulting sequence is then passed through the appropriate integer bandpass filter to complete the reconstruction of that subband. Thus the baseband version of the subband signal is effectively modulated back up to its original frequency range. The final task of the inverse filter bank is to sum the outputs of its constituent filters.

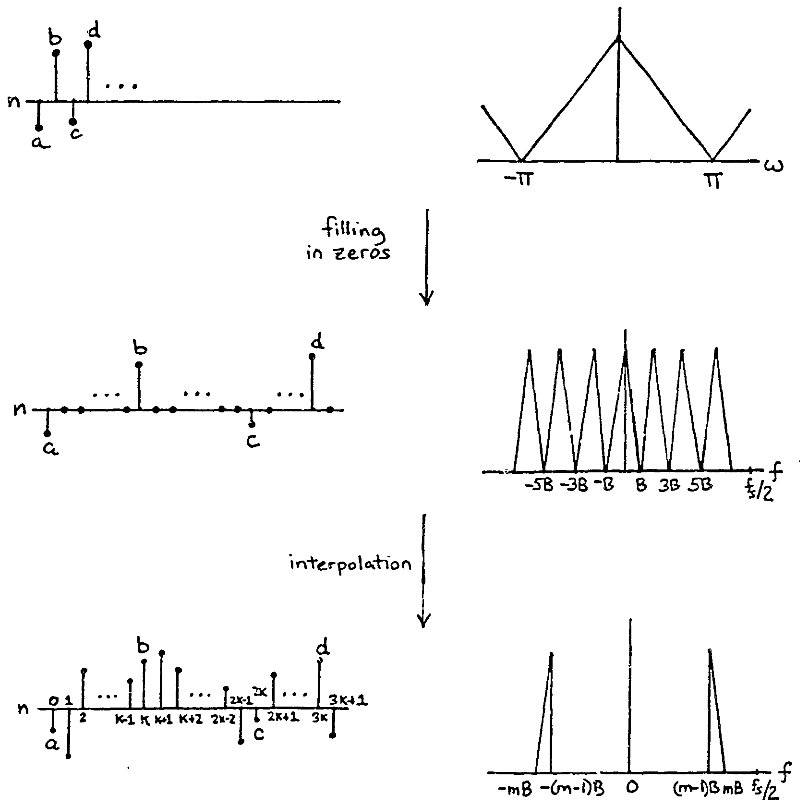


Figure 1-14. Interpolation of a Subband

So far in our discussion we have assumed that the bandpass filters have perfectly sharp cutoffs between passband and stopband. However, ideal filters are not realizable. Instead, we must deal with non-zero transition bands and their consequences. If we still want to decimate by k, each bandpass filter must have the same bandwidth $B = f_s/(2k)$. But the transition regions between bands will leave annoying notches in the composite frequency response of the coder (Fig. 1-15a). To remove the notches, each bandpass filter should have a bandwidth B'>B (Fig. 1-15b). But then the decimation rate would be $f_s/(2B')<k$, producing more samples than desired.

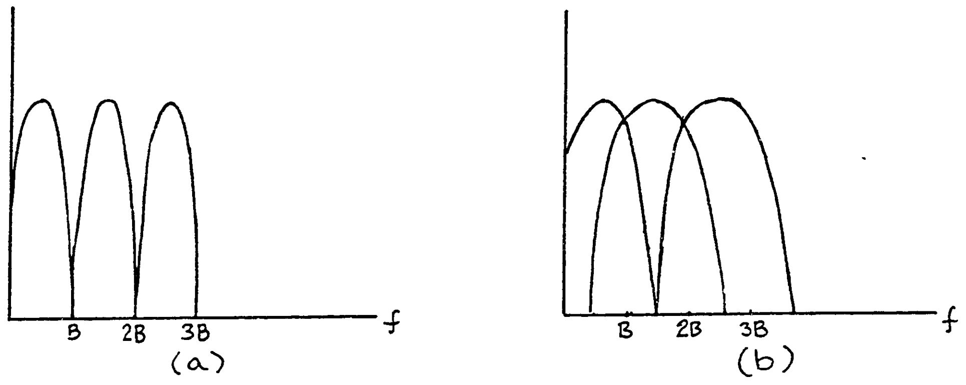


Figure 1-15. Practical Bandpass Filters

Esteban and Galand [2] solved this dilemma by developing what are known as Quadrature Mirror Filters (QMF). QMF's are specially designed bandpass filters that have bandwidth greater than B (as in Fig. 1-15b) and yet yield subband outputs that can be decimated by $k=f_s/(2B)$ without producing irreparable aliasing. This works because the aliasing caused by excessive decimation in the coder's QMF filter bank is actually cancelled out when the voice signal is reconstructed from its subbands by the receiver's inverse QMF filter bank. Since Esteban and Galand's work, many additional studies have been done on QMF's (e.g. [20-23]).

In order to take advantage of subband splitting, quantization bits should be allocated to the bands based upon average power. This may be done with a fixed allocation scheme based upon long-term speech statistics or, more effectively, with an adaptive method using short-term power calculations. The dynamic allocation of bits tracks the relative power in the subbands and makes sure that the strongest bands are always given the most bits.

Early sub-band coders (SBC) used PCM with adaptive quantization to encode the individual subband signals. At 32 kbps, SBC voice quality was determined to be comparable to the standard 64 kbps µ-law PCM [2]. However, later studies included predictors for even more compression.

### 1.1.6 Sub-Band Coding With ADPCM

The basic block diagram for an SBC-ADPCM system is shown in Figure 1-16. There are many choices to be made in the design of such a sub-band coder. Several papers have experimented with variations in the bit allocation schemes, types of predictors, number of bands, filter bank implementation, and kinds of quantizers [24-33].

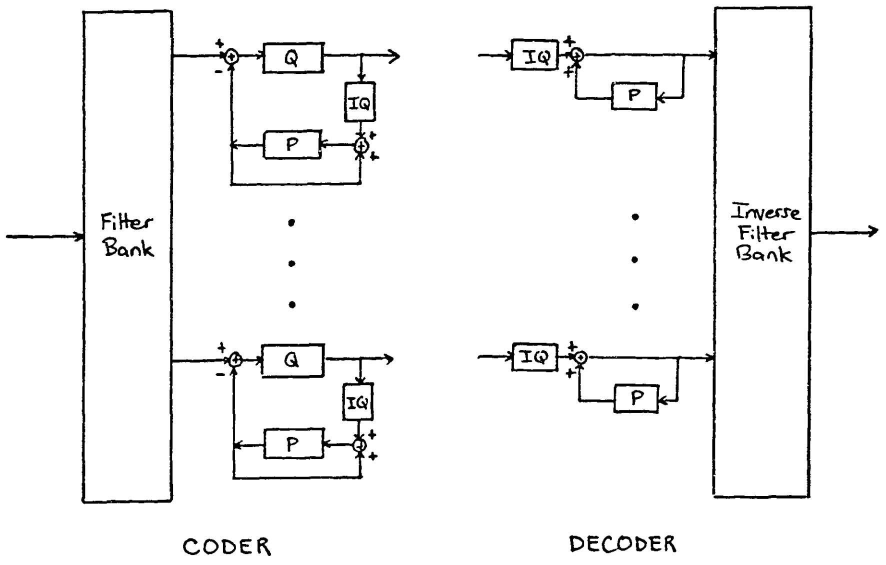
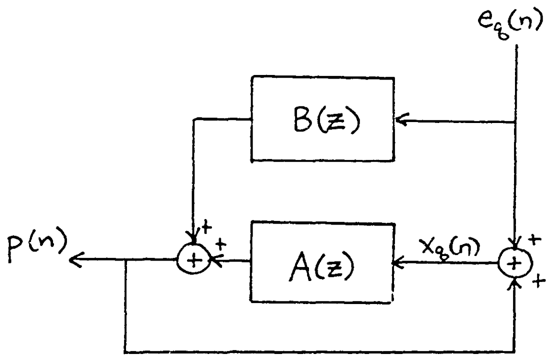


Figure 1-16. SBC-ADPCM

Galand, Daulasim, and Esteban [27] implemented an eight band sub-band coder with dynamic bit allocation and ADPCM to code the baseband (up to 1000 Hz) of a Voice Excited Predictive Coder. They employed a second order backward adaptive LMS transversal predictor to code each subband and obtained 2-12dB SNR improvement over a non-differential scheme.

In [33], Gupta and Virupaksha compared various types of sub-band coders. They considered several adaptive non-linear quantizers with dynamic bit allocation as well as differential coders with fixed bit allocation. Not included in their report was the combination of dynamic bit allocation with adaptive predictors. A four band QMF and overall bit rate of 16 kbps was selected for the study. An eighth order transversal predictor was used with some of the fixed bit allocation methods. Both constant and LMS adaptive predictor coefficients were tried. They found that adaptive quantization with dynamic bit allocation (AQ-DBA) outperformed constant and adaptive prediction with fixed bit allocation (DPCM-FBA and ADPCM-FBA) in objective SNR tests. However, subjective listening tests revealed that ADPCM-FBA was actually preferred over AQ-DBA, reinforcing the idea that SNR tests alone are not a sufficient indicator of voice quality. Also, both objective and subjective measures showed that ADPCM-FBA was much better than DPCM-FBA, indicating the benefits of adapting predictor coefficients to the input.

Hamel, Soumagne, and Le Guyader [32] simulated an SBC-ADPCM system which utilized the LMS adaptive predictor recommended by the International Telephone and Telegraph Consultative Committee (CCITT). The CCITT predictor (Fig. 1-17) is different from the ones we have discussed so far in that the output, p(n), is not a strict linear combination of past reconstructed coder inputs, r(n). In addition, there are terms corresponding to the quantized error signal, $e_q(n)$. This is the same predictor that is used in the standard 32 kbps ADPCM coder recognized by the American National Standards Institute (ANSI) [34]. The eight coefficients are updated by an LMS (gradient) method on a sample-to-sample basis. Hamel, et.al. ran their coder with an eight band QMF, dynamic allocation of bits, and adaptive quantization at 16 kbps.



$$p(n) = \sum_{i=1}^{2} a_i(n)x_q(n-i) + \sum_{i=1}^{6} b_i(n)e_q(n-i)$$

Figure 1-17. CCITT Predictor

In 1985, Soong, Cox, and Jayant [28] published a comparative study of various SBC-ADPCM building blocks. They found that in terms of segmental SNR, a least-squares lattice predictor did better than both the CCITT recommended one and a constant first order transversal structure. Furthermore, coder performance improved with the number of subbands and with the length of the QMF filters. Also, dynamic bit allocation was judged to be superior to a fixed assignment.

## 1.2 PROBLEM

The two widely used adaptive predictors, LMS transversal [17] and LS lattice [35,36], have not been compared in the context of a sub-band coder. However, numerous papers [14,37-42] have addressed their relative merits, which reflect the choice of both parameter update algorithm and implementation configuration.

Referring to Figures 1-9 and 1-10 we see that the lattice structure requires more computation per output point. Aside from this drawback, lattices have three useful properties [39]. Whereas the intermediate sums of the tapped delay line have no significance, the $m^{th}$ sum of a ladder form represents the output of the corresponding $m^{th}$ order transversal predictor. Thus, an $N^{th}$ order lattice automatically generates all the outputs of tapped delay lines of order 1 to N. This property of lattice structures allows predictor orders to be dynamically assigned [43]. A second feature of lattices is their modularity; longer lattices can be constructed by simply adding on identical stages to smaller ones. Finally, ladder forms have been found to be very insensitive to roundoff noise.

The LS lattice predictor generally converges faster and tracks sudden changes in input better than the LMS transversal [14]. The least-squares algorithm is an exact adaptation in the sense that for each new sample, optimal predictor parameters are computed which minimize the sum of the squared prediction errors up to that sample [35]. The least-mean-square algorithm, however, is a gradient search technique that updates the coefficients in the direction of the minimum average squared error. This is only an approximate solution since the actual gradient is never known [44].

More controlled comparisons of the LS and LMS methods have been conducted by implementing both in lattice form. Satorius and Shensa [39] developed lattice parameter adaptation equations based upon LS and LMS criteria. They showed that the update equations were very similar except for a variable gain factor in the LS adaptation that could be interpreted as a 'likelihood' detector. For likely data samples, the gain remains constant so parameter updates follow a fixed step size. For unexpected samples corresponding to sudden input changes, the gain variable becomes large, thereby increasing the adaptation speed for improved tracking. Medaugh and Griffiths [37] derived a similar relationship between the two sets of update equations but their simulations of convergence behavior did not indicate a preferred predictor.

The performances of different adaptive predictors have been studied for ADPCM systems. Honig and Messerschmitt [41] considered five predictors - fixed transversal, LMS transversal, LMS lattice, LS lattice, and LS lattice with pitch prediction. Pitch predictors make estimates by recalling samples from one pitch period earlier instead of just the past few samples. Their simulations of ADPCM systems with adaptive quantization and fifth and tenth order predictors showed no significant differences in SNR or root-mean-square predictor error between the four adaptive predictors. This was an unexpected result since LS lattice algorithms are supposed to converge faster [14]. One explanation they offered was that the LMS transversal predictor was quick enough to adapt well to the test waveforms, so the extra speed of the LS algorithm was not observable. In another paper, Reininger and Gibson [42] looked at an ADPCM system with adaptive quantization and an

LMS transversal, Kalman transversal (which uses a least-squares criterion), LMS lattice, or LS lattice predictor. The coder was run at 16 kbps on five sentences and the following subjective ranking (best to worst) was obtained : LS lattice, Kalman transversal, LMS lattice, LMS transversal.

## 1.3 GOALS

The purpose of this study is to compare the adaptive least-mean-square transversal and least-squares lattice predictors in the context of an SBC-ADPCM coder. In particular, we will consider 16 and 24 kbps sub-band coders with either LMS, LS, or no predictor at all. Results at these bit rates will indicate the better SBC predictor as well as LMS and LS improvements over a non-differential scheme. In addition, performance trends with respect to predictor order will be determined. Finally, cross comparisons between the two bit rates will show whether the use of a predictor can maintain the same voice quality at a bandwidth savings of 8 kbps.

## 1.4 APPROACH

The sub-band coding algorithm is implemented with and without prediction at 16 kbps and 24 kbps. These coders are then used to process several phonetically balanced sentences. Finally, the relative performances of the coders are evaluated using SNR and SSNR values as well as double-blind A-B listening tests.

## 1.5 EQUIPMENT

All simulation is done in FORTRAN on an IBM VM/SP mainframe at ROLM Corporation (Santa Clara, CA). Quadrature Mirror Filter design utilities are written in BASIC on the IBM PC-AT. A ROLM proprietary voice card does the analog-to-digital and digital-to-analog conversions of test sentences before and after SBC processing.

# CHAPTER 2 - DESIGN OF CODERS

This chapter describes the components of the speech coders that are implemented and tested.

## 2.1 OVERVIEW

All of the coders simulated are based on sub-band splitting (sub-band coders are also known as split-band voice coding schemes). These can be classified into three categories depending on the type of predictor used : least-mean-square transversal, least-squares lattice, or none.

The benefits of sub-band coding are due to two main features. The first is the use of Quadrature Mirror Filters (QMF) to separate voice into individual subbands. QMFs are designed so that the subband signals they generate can be decimated and interpolated without introducing aliasing distortion in the reconstructed signal. This permits the coding of fewer subband samples, thereby improving performance at any given bit rate. The second essential feature is the bit allocation scheme. By recognizing the fact that power is unequally distributed among subbands, coder performance can be optimized by using a correspondingly unequal assignment of bits. Intuitively, the bands that have more power are more important and are therefore allocated more bits.

It was found in [28] that coder performance improves with the use of more subbands, i.e. narrower bandpass filters. Also, a dynamic bit allocation scheme, one that redistributes bits at regular intervals, demonstrates superior results compared to a fixed assignment (this is to be expected since the short-time power spectrum of speech, in addition to being non-uniform with frequency, changes with time). With these recommendations, this study implements a 16-band QMF with bit allocation that is dynamically adjusted every 16ms frame (128 8KHz input samples). The QMF bank is an integer-band structure (Chapter 1), with 250Hz wide filters covering 0-4000Hz. As suggested in [29], the top three bands are not coded, i.e. always assigned zero bits, since telephony channels typically cut off around 3200Hz [6].

During any particular frame, each subband is allocated a certain number of bits per sample. But how those bits are actually utilized depends on the quantizing scheme for that subband. In this study, adaptive PCM and ADPCM are used.

The coders without predictors employ uniform, mid-rise quantizers with step size that is adapted on a sample-by-sample basis. The coders with predictors use the same adaptive quantizer but feed it a difference signal generated by subtracting the predictor output from the subband sample (ADPCM). As discussed in the introduction, the smaller variance of the difference signal should enhance the operation of the quantizer.

An Nth order predictor works only as well as there is correlation between current samples and the N samples immediately preceding them. Previous studies [31,32] have shown that bandpass signals of width 250Hz or 500Hz located at frequencies higher than about 1000Hz have close to zero correlation between samples (in other words, the autocorrelation function of these subband signals have negligible values at all delays except for zero). For this reason, predictors are used only in the first four subbands.

Each of the remaining sections of this chapter focuses on one of the four main components of the SBC-ADPCM system : the filter bank, bit allocation algorithm, adaptive quantizer, and adaptive predictors.

## 2.2 QUADRATURE MIRROR FILTERS

The input signal to the coders is first processed by a 16-band filter bank. Since band splitting creates several signals from only one, this naturally tends to increase the number of samples that need to be coded. In order to prevent this, each of the subband signals is decimated by a factor of 16 (only one out of every 16 samples is preserved). However, just as sampling an analog signal introduces aliased components in the frequency domain (shifted copies of the original spectrum), decimating a discrete-time signal produces aliasing as well.

More precisely, if $X(e^{j\omega})$ is the Discrete-Time Fourier Transform of a sequence $x(n)$

$$(2\text{-}1) \qquad X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

and $y(n)$ is the result of decimating $x(n)$ by a factor of M

$$(2\text{-}2) \qquad y(n) = x(nM)$$

then it can be shown that ([45])

$$(2\text{-}3) \qquad Y(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} X(e^{j(\omega-2\pi k)/M})$$

The spectrum of $y(n)$ contains M-1 shifted copies of $X(e^{j\omega})$ as well as $X(e^{j\omega})$ itself. Notice also the presence of a 1/M factor in the exponent. This results from the fact that $y(n)$ has $1/M$ as many samples as does $x(n)$.

The advantage of QMF's over general bandpass filters is that the additional spectrum copies (aliasing terms) which are introduced by decimation are cancelled out when the subband signals are combined in a related inverse QMF bank [46]. To see how this is achieved, it is instructive to examine the simplest QMF, one with only two bands.

## 2.2.1 Two-Band Quadrature Mirror Filter

A two-band QMF consists of just a lowpass filter, $H_1(e^{j\omega})$, and a highpass filter, $H_2(e^{j\omega})$ (Fig. 2-1). The impulse responses of the filters are related by the transformation

(2-4) $$h_2(n) = (-1)^n h_1(n)$$

By direct substitution of (2-4) into (2-1), it can be shown that

(2-5) $$H_2(e^{j\omega}) = H_1(e^{j(\omega+\pi)})$$

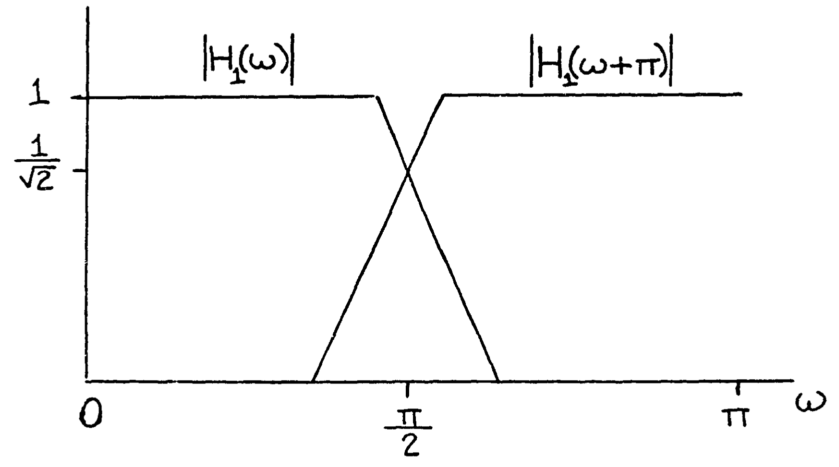


Figure 2-1. Two-Band QMF

In the two-band sub-band coder (Fig. 2-2), the outputs of the lowpass and highpass filters are each decimated by a factor of two. To reconstruct the coder input, the subband signals are upsampled by inserting a zero between every two samples, filtered again, and then added together. (For the purposes of this development, it is assumed that the subbands are perfectly coded and decoded.) Note that the inverse QMF, i.e. the bandpass filters used for reconstruction, are identical to the analysis filters with the exception of a minus sign in front of $H_2(e^{j\omega})$.

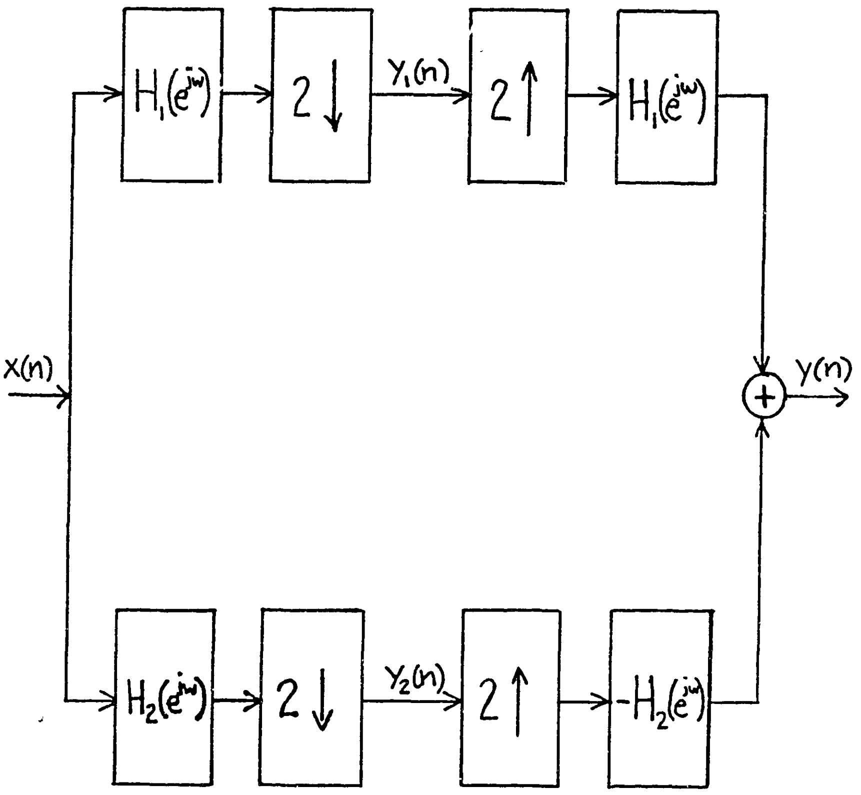


Figure 2-2. Two-Band Sub-Band Coder

In terms of Discrete-Time Fourier Transforms (DTFTs),

(2-6a) $$Y_1(e^{j\omega}) = \frac{1}{2}\,[H(e^{j\omega/2})X(e^{j\omega/2}) + H(e^{j(\omega/2-\pi)})X(e^{j(\omega/2-\pi)})]$$

(2-6b) $$Y_2(e^{j\omega}) = \frac{1}{2}\,[H(e^{j(\omega/2+\pi)})X(e^{j\omega/2}) + H(e^{j\omega/2})X(e^{j(\omega/2-\pi)})]$$

where the second term in each expression represents the aliasing caused by decimation. The upsampling of the subbands has the effect of a $\omega \rightarrow 2\omega$ transformation of the frequency response. Thus, the output $Y(e^{j\omega})$, is given by

$$(2\text{-}7)\qquad Y(e^{j\omega}) = H(e^{j\omega})Y_1(e^{j2\omega}) - H(e^{j(\omega+\pi)})Y_2(e^{j2\omega})$$

$$= \frac{1}{2}[H(e^{j\omega})X(e^{j\omega})+H(e^{j(\omega-\pi)})X(e^{j(\omega-\pi)})]H(e^{j\omega})$$

$$- \frac{1}{2}[H(e^{j(\omega+\pi)})X(e^{j\omega})+H(e^{j\omega})X(e^{j(\omega-\pi)})]H(e^{j(\omega+\pi)})$$

$$= \frac{1}{2}X(e^{j\omega})[H^2(e^{j\omega})-H^2(e^{j(\omega+\pi)})]$$

where the $2\pi$-periodicity of the DTFT is used to recognize that the aliasing terms cancel. It only remains to show that $H_1(e^{j\omega})$ can be designed so that $|H_1{}^2(e^{j\omega})-H_1{}^2(e^{j(\omega+\pi)})| \simeq 1$.

Symmetric, finite impulse response (FIR) filters with an even number of taps, M, have a characteristic frequency response of the form

$$(2\text{-}8)\qquad H(e^{j\omega}) = A(\omega)e^{-j\omega(M-1)/2}$$

where $A(\omega) = |H(e^{j\omega})|$. If $H_1(e^{j\omega})$ is such an FIR filter, then (2-7) becomes

$$(2\text{-}9)\qquad Y(e^{j\omega}) = \frac{1}{2}X(e^{j\omega})[A^2(\omega)e^{-j\omega(M-1)}+A^2(\omega+\pi)e^{-j\omega(M-1)}]$$

$$= \frac{1}{2}X(e^{j\omega})e^{-j\omega(M-1)}[A^2(\omega)+A^2(\omega+\pi)]$$

$A^2(\omega)+A^2(\omega+\pi)$ can be made close to 1 by designing the magnitude response of $h_1(n)$ to be very flat in the passband, highly attenuated in the stopband, and at half power in the middle of the transition band. The complex exponential corresponds to a real-time delay of M-1 samples.

If $H_1(e^{j\omega})$ is designed according to these specifications, then the output of the sub-band coder will be a delayed version of the input scaled by a factor of 1/2. The inverse QMF bank typically scales all of its outputs by two to correct for this attenuation.

## 2.2.2 Tree vs. Parallel Implementation

The two-band QMF be easily extended to yield $N=2^t$ bands. This can be done with either a tree or parallel structure [46].

The more straightforward implementation is the tree configuration (Fig. 2-3a). Separation into subbands is done in stages by repeated application of the two-band QMF. The first stage splits the input into a lowpass and a highpass version, which are decimated by two. Each of these signals is then split into two more bands and again decimated. The net effect of the two stages is that of a bank of four integer-band bandpass filters (Fig. 2-3b). Since each of the two-band QMFs at any stage insure aliasing cancellation for their inputs, the cascade of analysis filters does as well. Thus, a $2^t$-band QMF can be implemented with t stages of two-band QMFs.

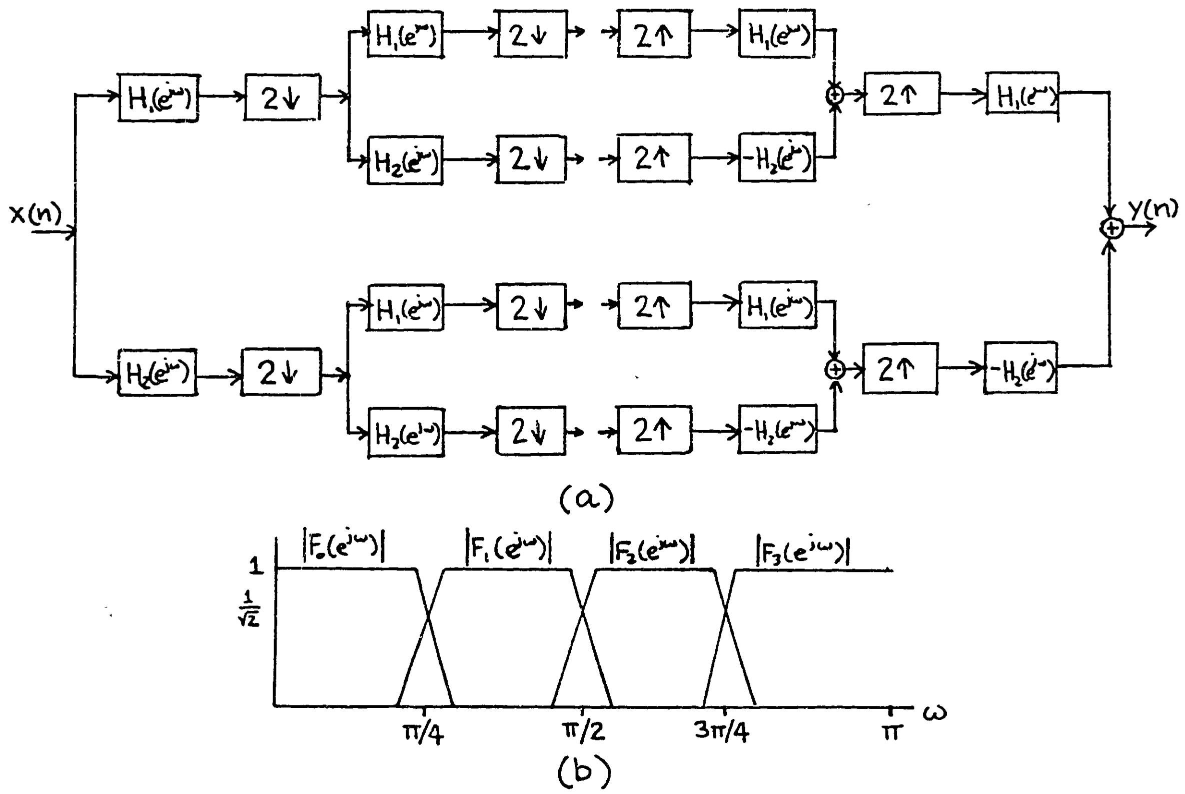


Figure 2-3. Tree Configuration (4-Band)

In Fig. 2-3, the same fundamental lowpass filter, $H_1(e^{j\omega})$, is used at each stage (in general this need not be true). It is clear that $H_1(e^{j\omega})$ separates lowpass and highpass components in the first stage, but not as obvious is how the same filter can split each of these signals into even narrower bands in the second stage.

As mentioned in Chapter 1, the decimation of an integer-band signal implicitly performs a modulation to the baseband along with a normalization of the frequency axis. This allows the use of $H_1(e^{j\omega})$ in the second stage. Fig. 2-4 illustrates this point. The subsampling step (multiplying by an impulse train) achieves modulation down to the baseband and throwing out the zero samples normalizes the sampling frequency to the decimated rate. (Roughly speaking, shrinking in the time domain corresponds to stretching in the frequency domain, and vice versa.) Thus, the baseband version of each subband is stretched out so that $H_1(e^{j\omega})$ selects half of it. A consequence of this is that the transition region of filters in the second stage are effectively twice as narrow when considered at the original sampling rate. Therefore, to achieve a specified transition bandwidth for the four net bandpass filters of the QMF analysis, the second stage filters only need about half as many taps as those in the first stage. For a t-stage QMF, the filters in stage i are typically designed to be half as long as those in stage i-1. This reduces computation costs and delay time.

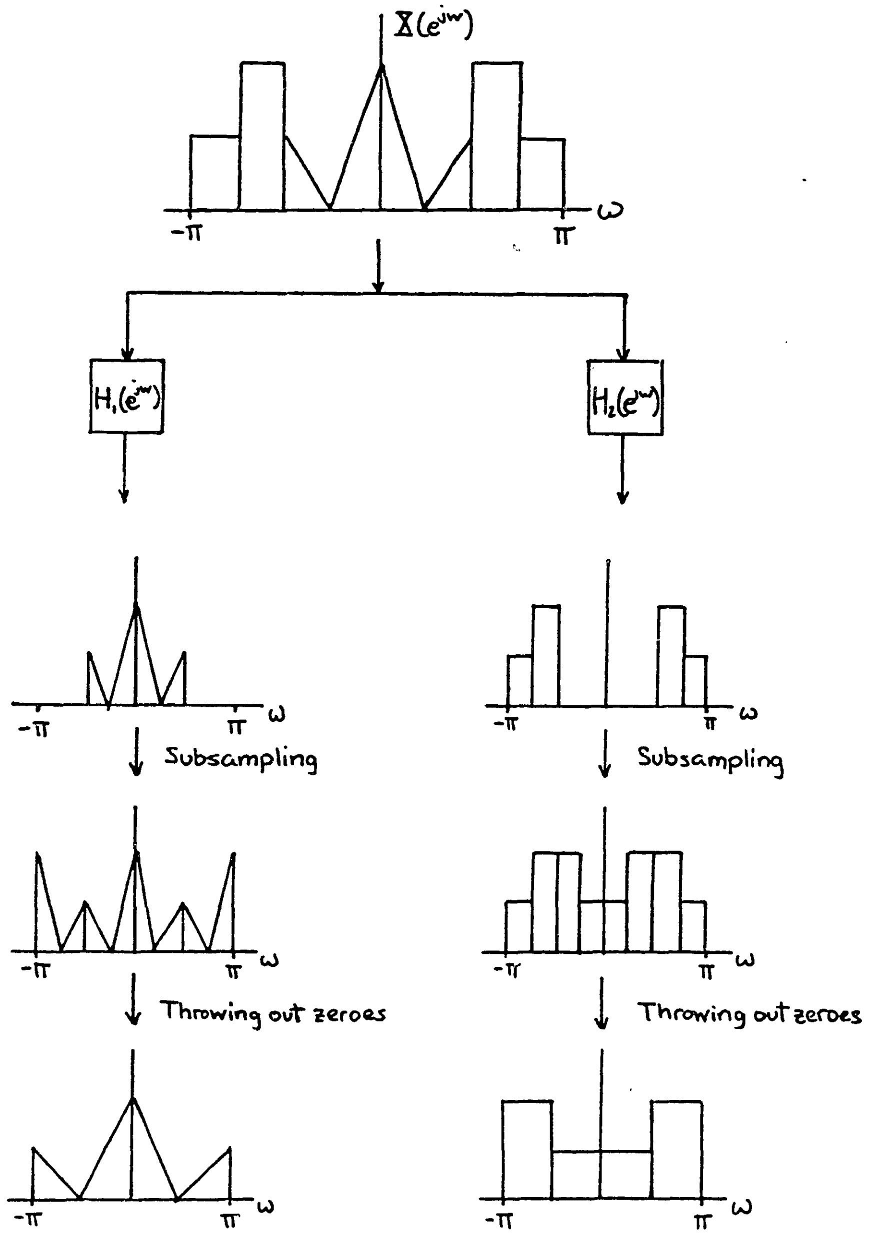


Figure 2-4. Tree QMF, Frequency Domain (4-Band)

To determine the frequency responses of the $2^t$ composite bandpass filters, we must cascade appropriate filters from each stage while taking into account the differences in sampling rates (Fig. 2-5). Each net bandpass filter is obtained by choosing a combination of t lowpass or highpass filters,

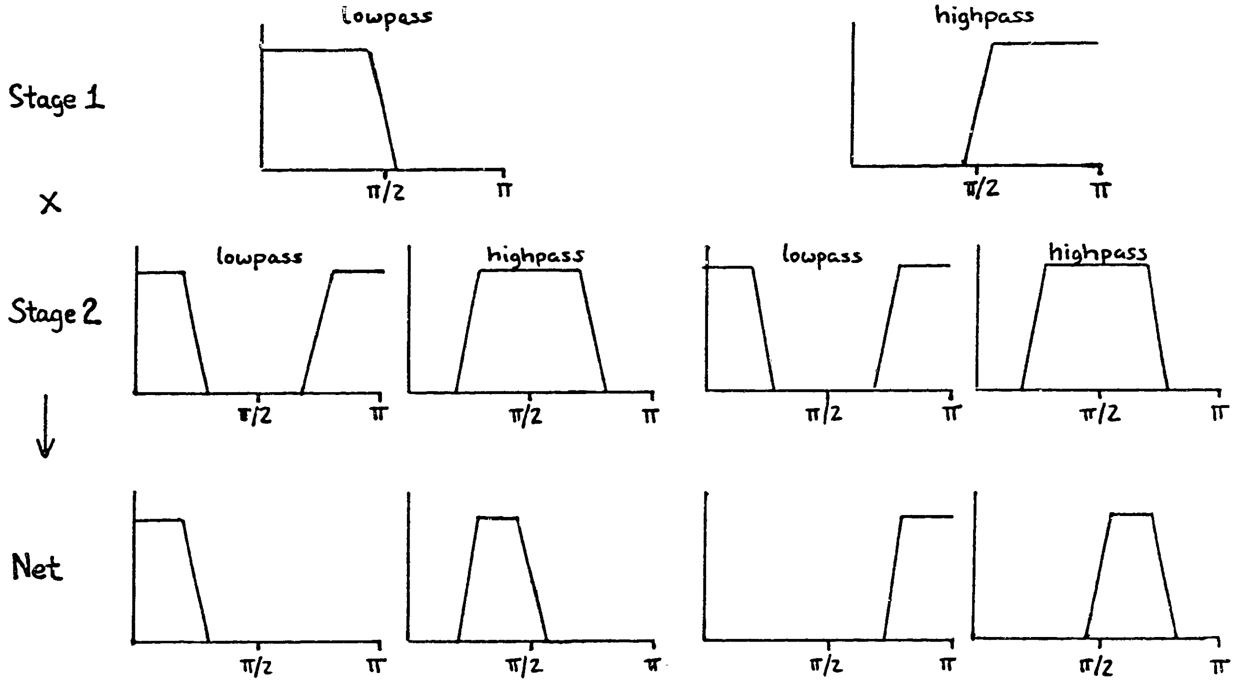


Figure 2-5. Multiplying Stage Responses

The parallel QMF (Fig. 2-6) directly implements the net bank of bandpass filters, thereby avoiding the explicit cascading of stages. The coefficients of the parallel filters are determined by recognizing that multiplication of frequency responses is equivalent to convolution of the impulse responses. However, just as decimation must be considered in the frequency domain, its effects cannot be overlooked in the time domain. In other words, the impulse responses of the half-band filters at each stage cannot be directly convolved with one another. To compensate for the different sampling rates, $2^{i-1}-1$ zeroes must be inserted between successive taps of the ith stage (upsampling by $2^{i-1}$) before convolution. It is easy to show that if $M_i$ is the length of the lowpass or highpass filter of the ith stage of a tree structure, then the impulse response of the corresponding direct parallel implementation has length

$$L = 1 + \sum_{i=1}^{t} (M_i-1)2^{i-1} \qquad (2\text{-}10)$$

Galand and Nussbaumer have found that the result of this convolution typically contains very small coefficients at the beginning and end of the impulse response [46]. This permits truncation of the impulse response to approximately $L^* = L/t$ while still achieving adequate aliasing cancellation.

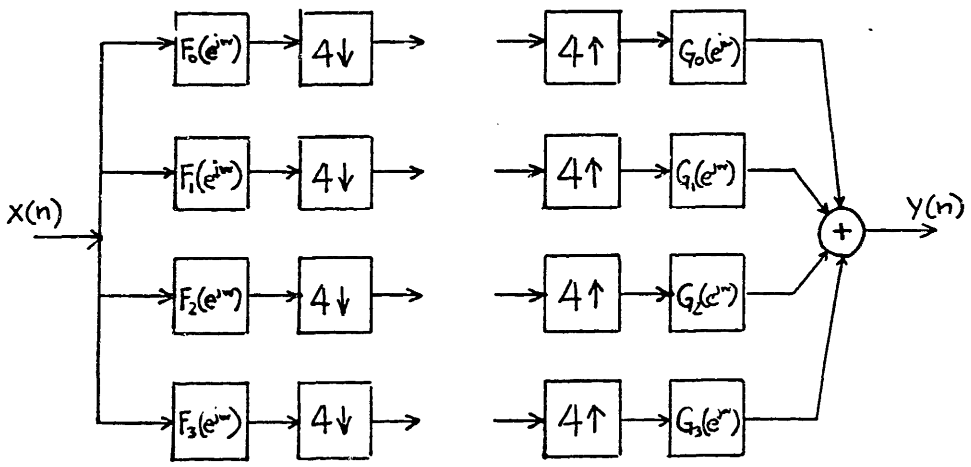


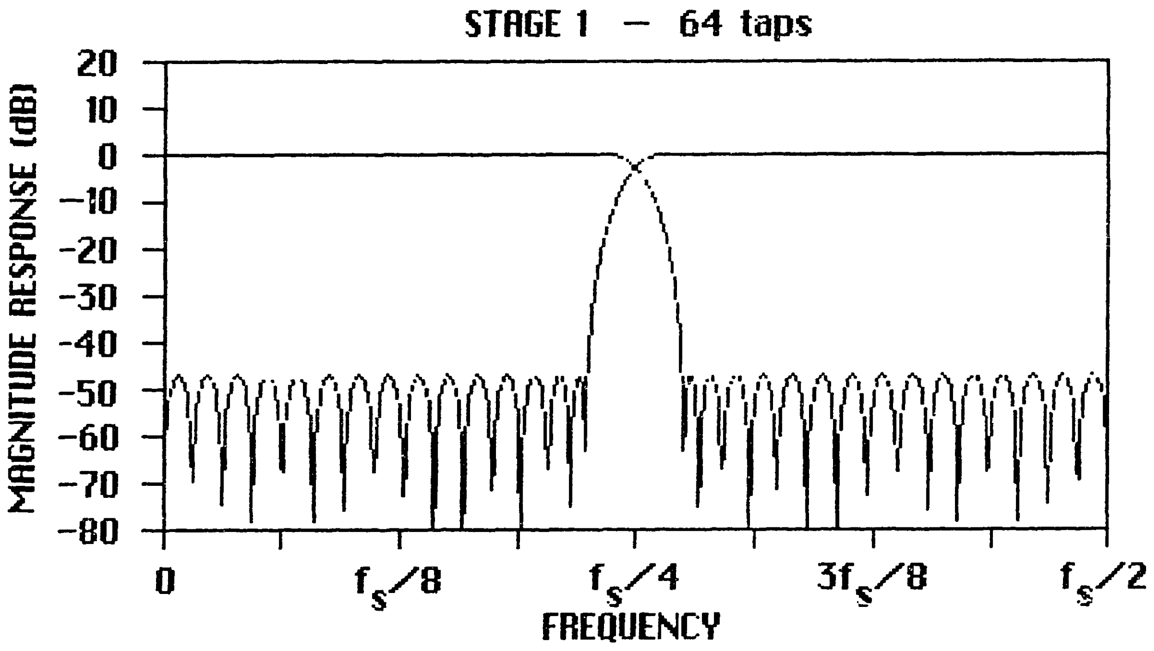
Figure 2-6. Parallel QMF (4-Band)

Parallel implementations with truncated tap lengths have shorter delays, require less storage, and are less complex than equivalent tree structures while requiring about the same number of multiplications and additions.

## 2.2.3 Filter Design

In this study a 16-band parallel QMF of length 72 taps is designed by convolving, with decimation adjustment, impulse responses corresponding to lowpass and highpass filters of an associated tree structure.

The first step in designing the filter bank is to come up with a half-band lowpass filter for each of the four stages. Coefficients of the half-band filters were derived using the Remez exchange algorithm for optimal linear phase FIR construction [5]. Filter lengths for stages 1 to 4 were 64, 32, 16, and 10 taps. The actual coefficients are listed in Appendix A. Their frequency responses, as well as those of their mirror image highpass versions, are shown in Fig. 2-7.



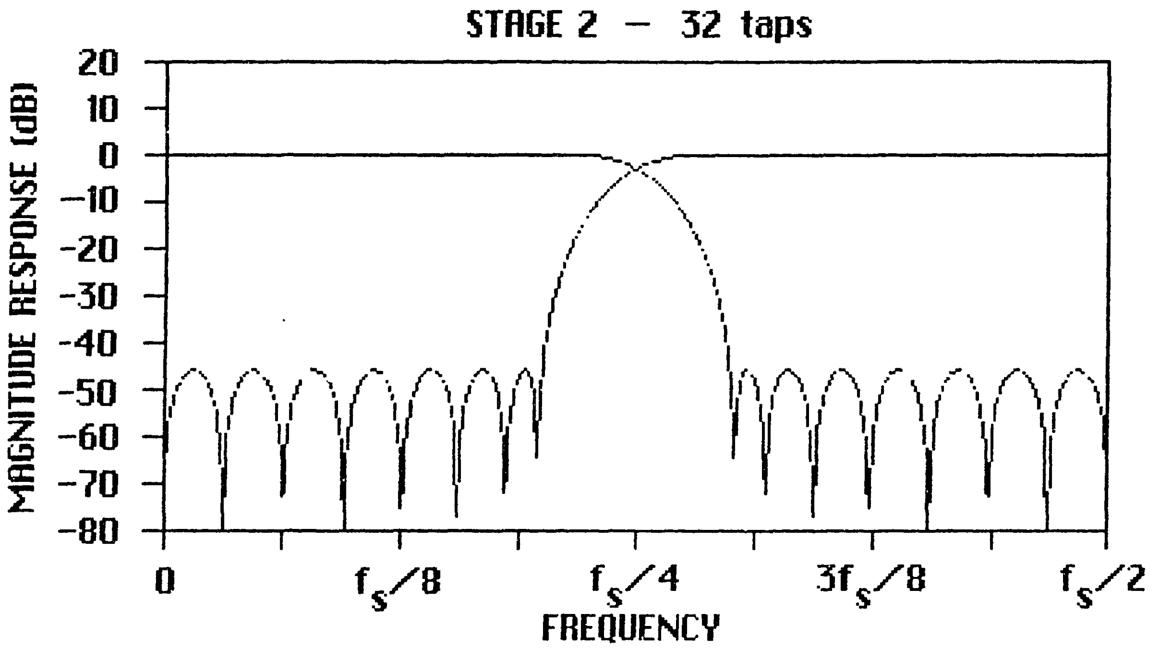
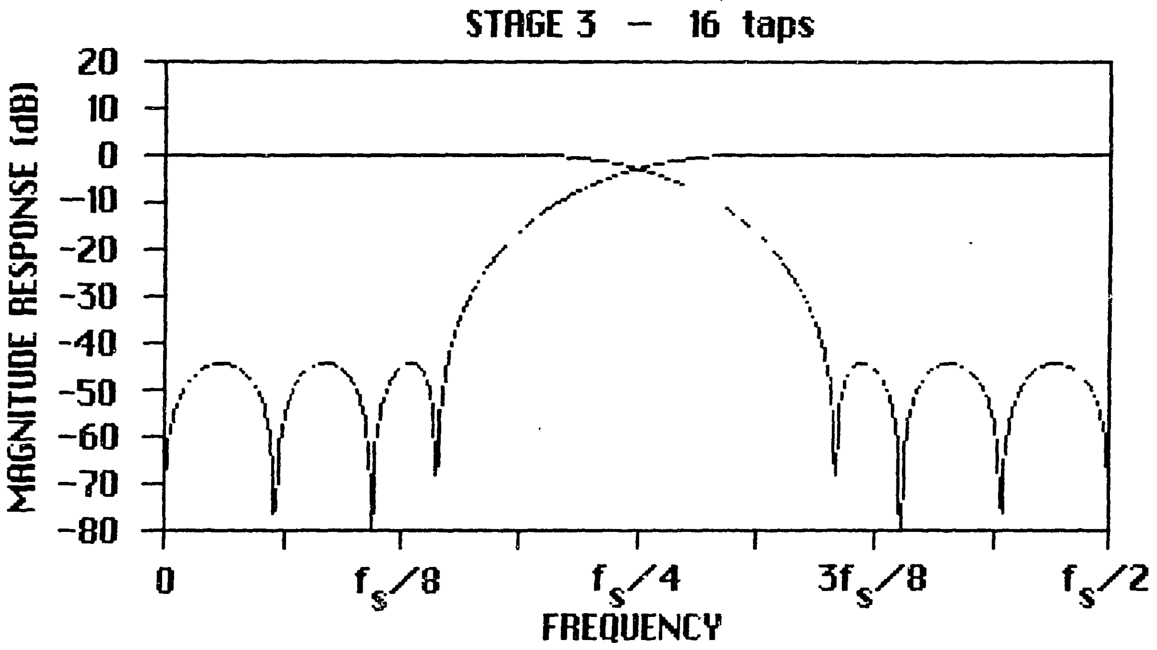


Figure 2-7. Half-Band Filters



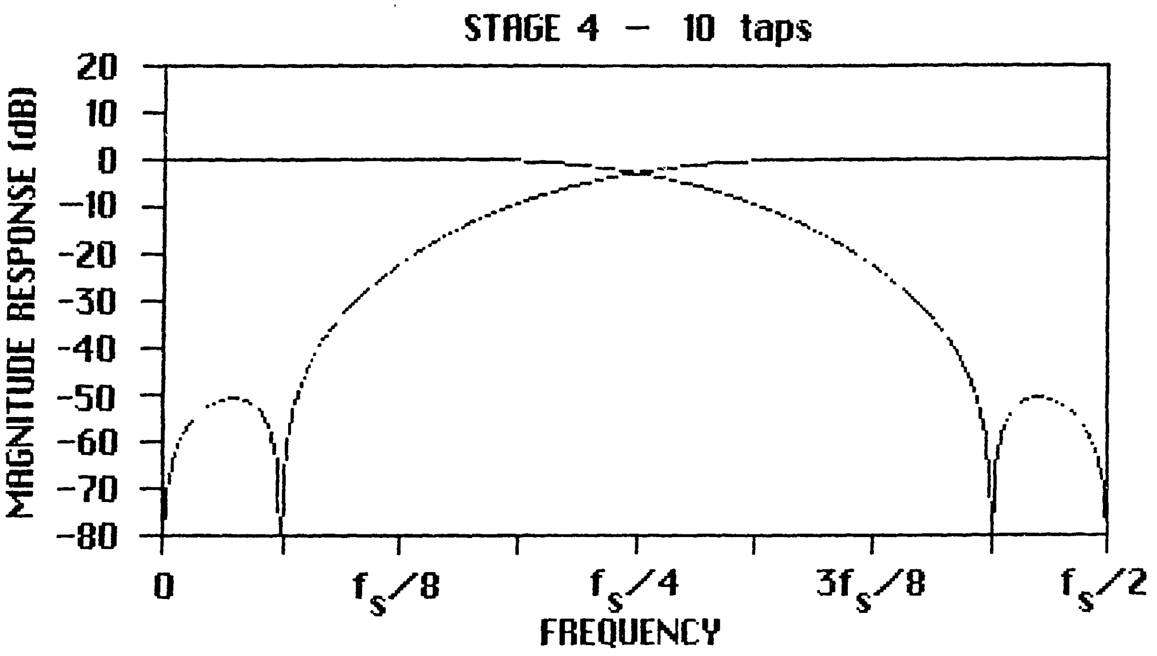
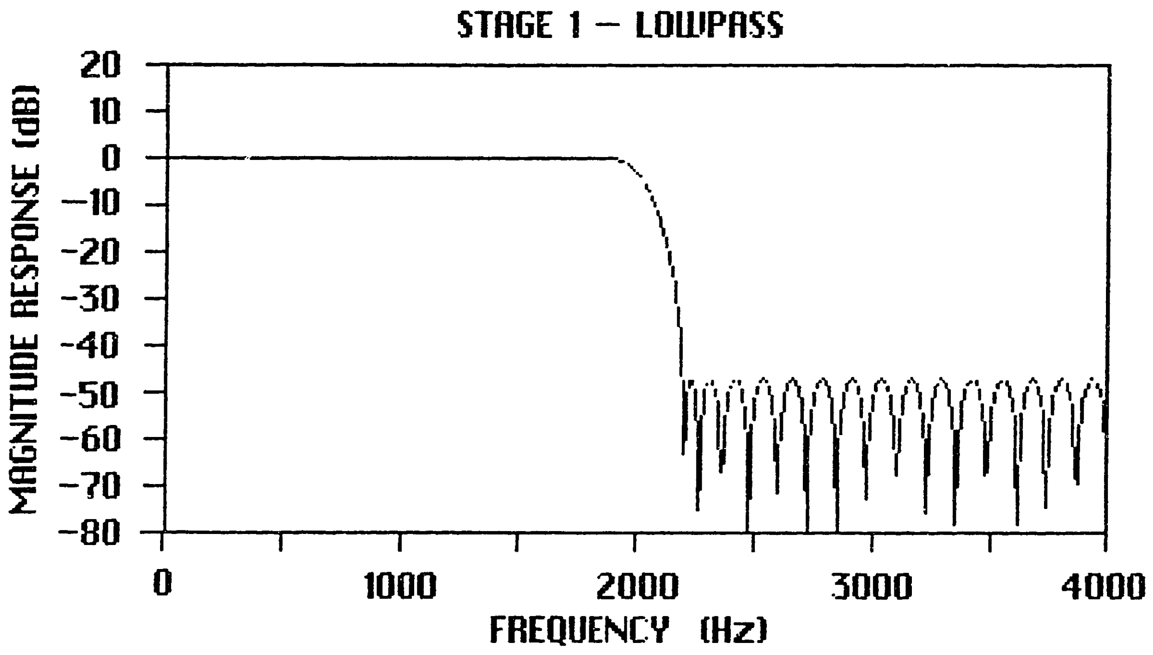


Figure 2-7. Half-Band Filters (continued)

After normalization to the sampling frequency of stage 1, the frequency responses become



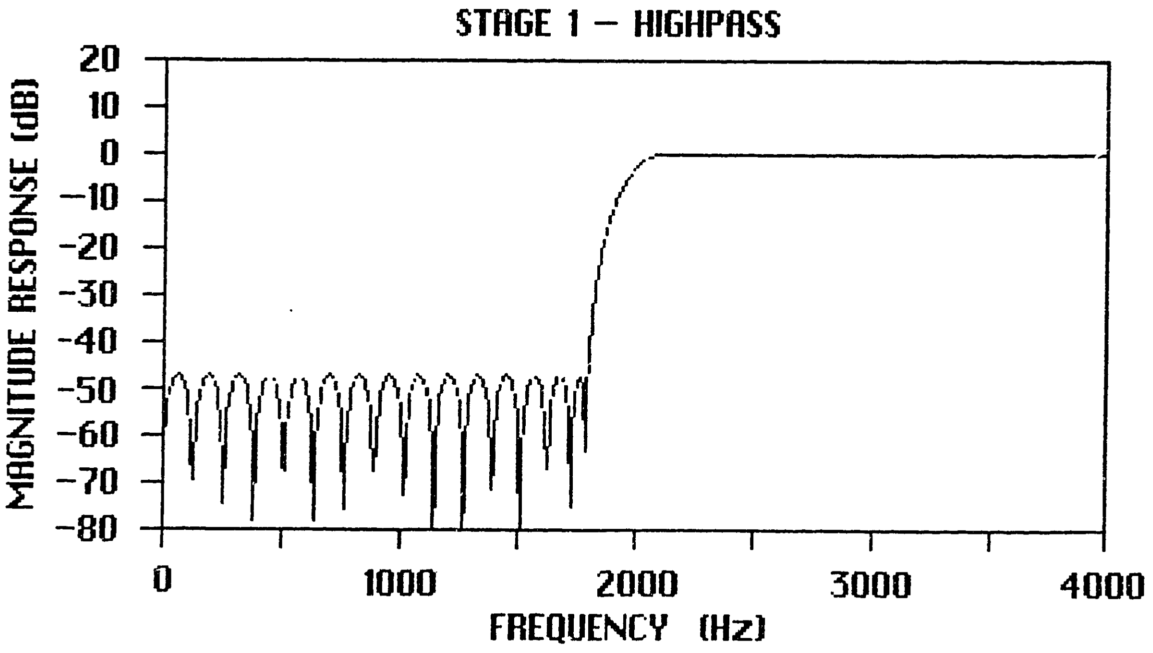
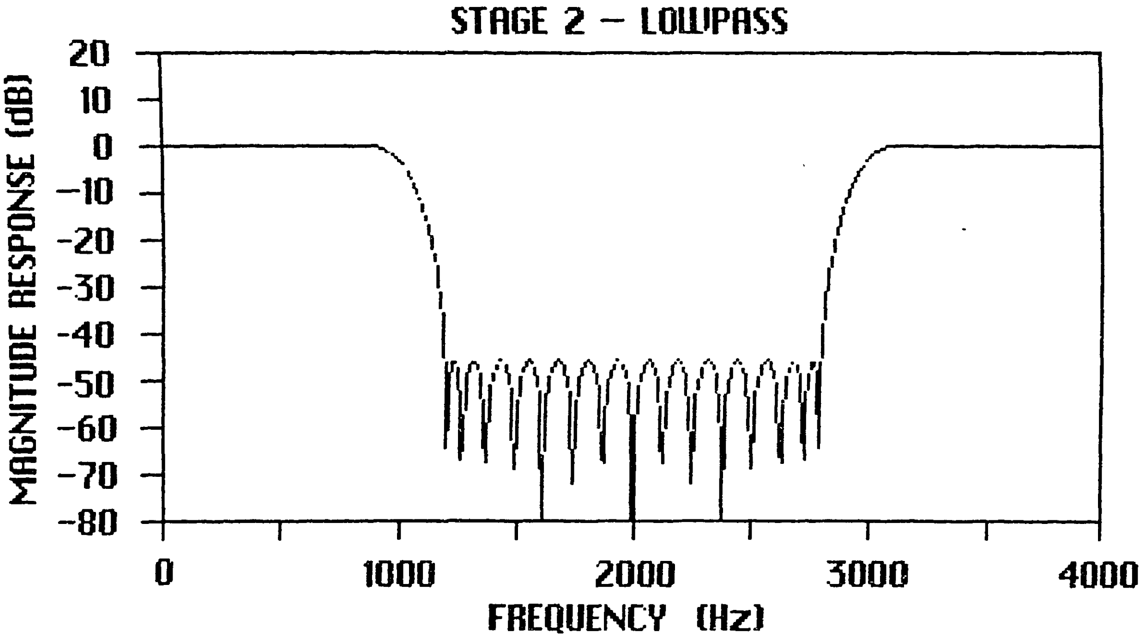


Figure 2-8a. Normalized Half-Band Filters (Stage 1)



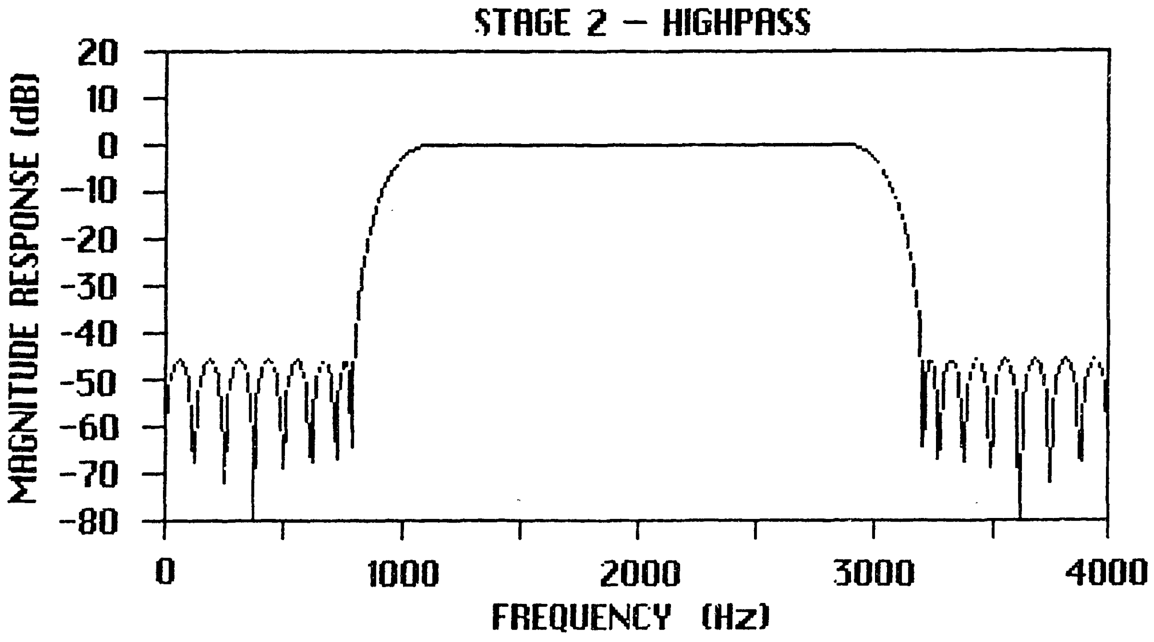
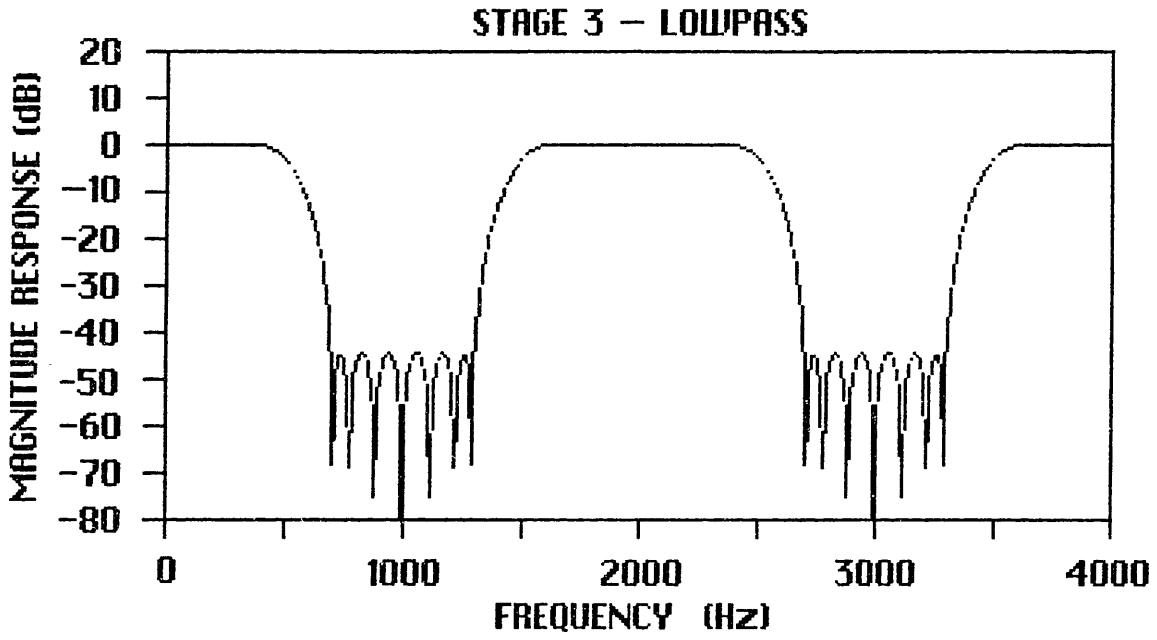


Figure 2-8b. Normalized Half-Band Filters (Stage 2)



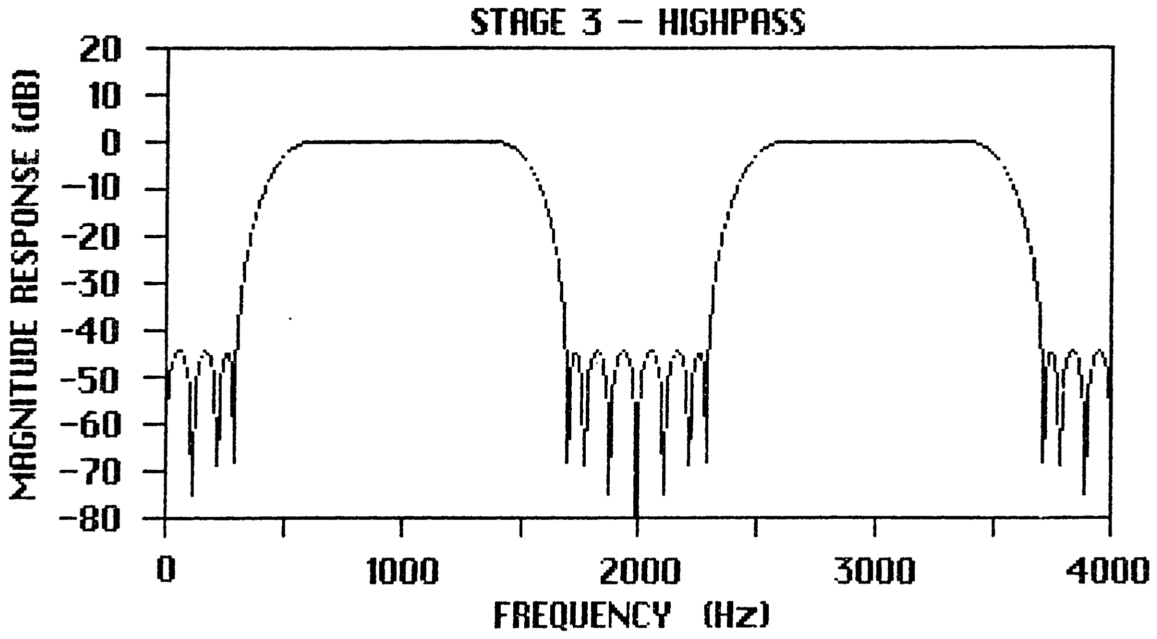
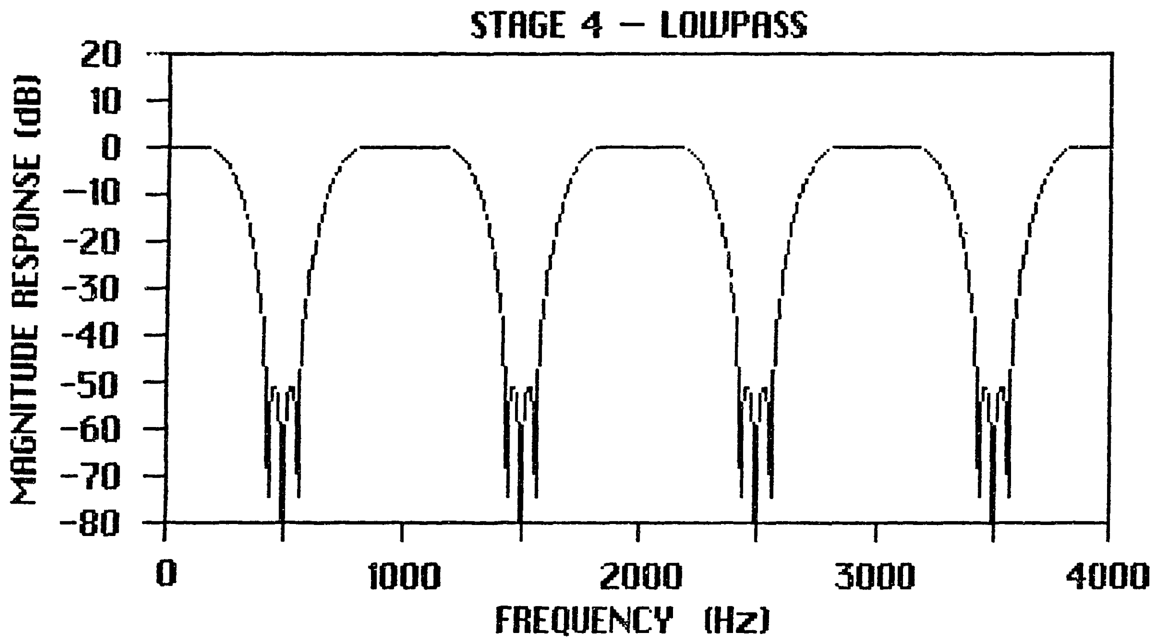


Figure 2-8c. Normalized Half-Band Filters (Stage 3)



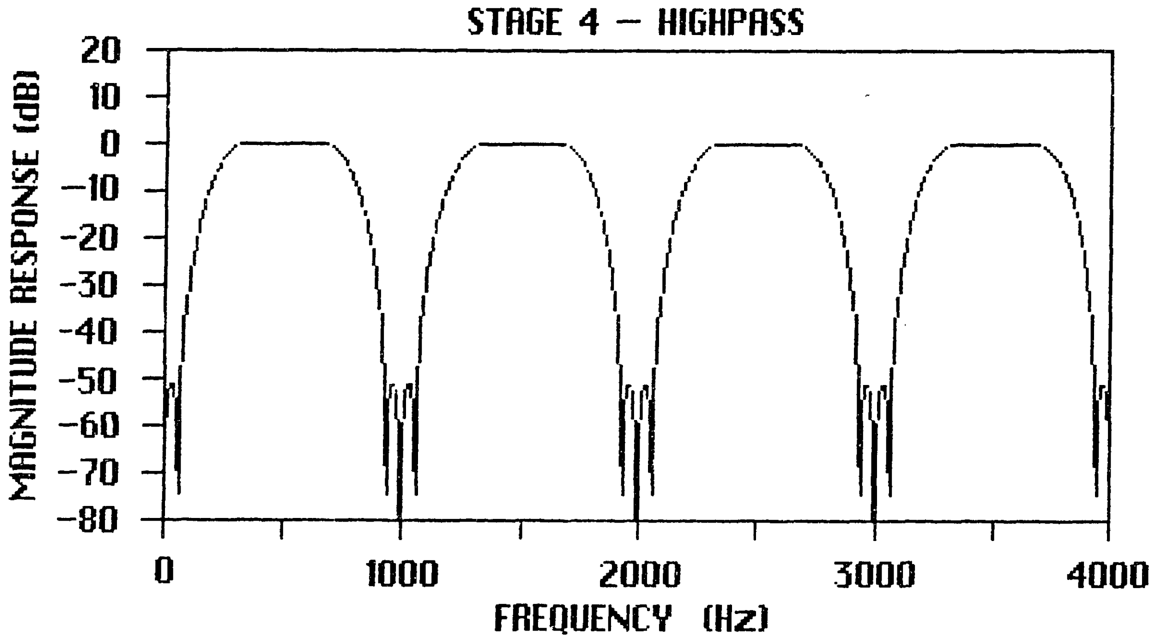
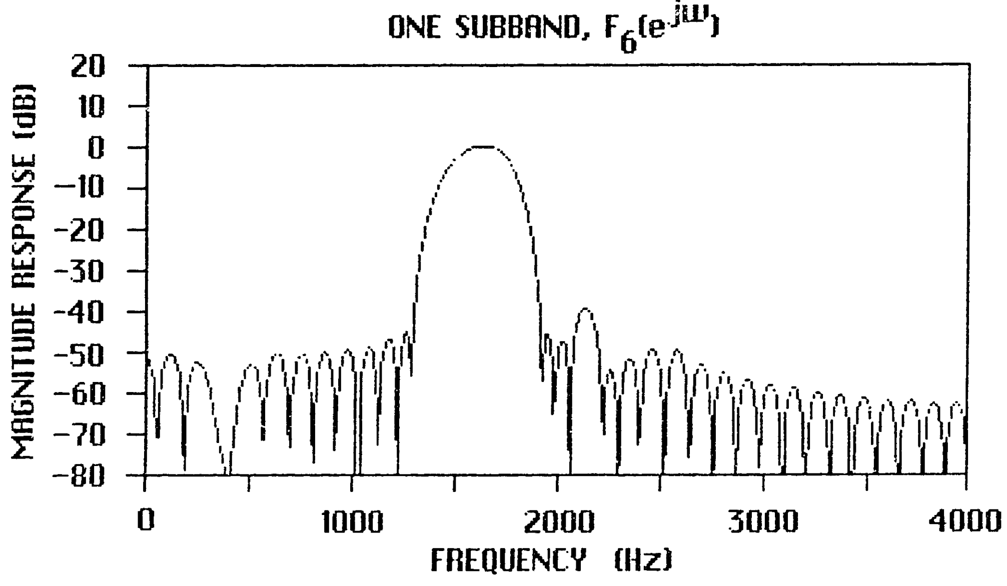


Figure 2-8d. Normalized Half-Band Filters (Stage 4)

These filters are convolved to form 16 bandpass filters each of length 258 (see equation (2-10)). The first and last 93 coefficients are truncated to leave 72 taps (see appendix A for coefficient values). The frequency responses of the resulting filters are shown in Fig. 2-9. The composite response of the QMF system, given by

$$h(n) = \sum_{k=0}^{N-1} f_k(n) * g_k(n) \tag{2-11}$$

where $f_k(n)*g_k(n)$ is the convolution of the kth bandpass impulse responses of N-band QMF and inverse QMF filter banks, respectively, is plotted in Fig. 2-10.



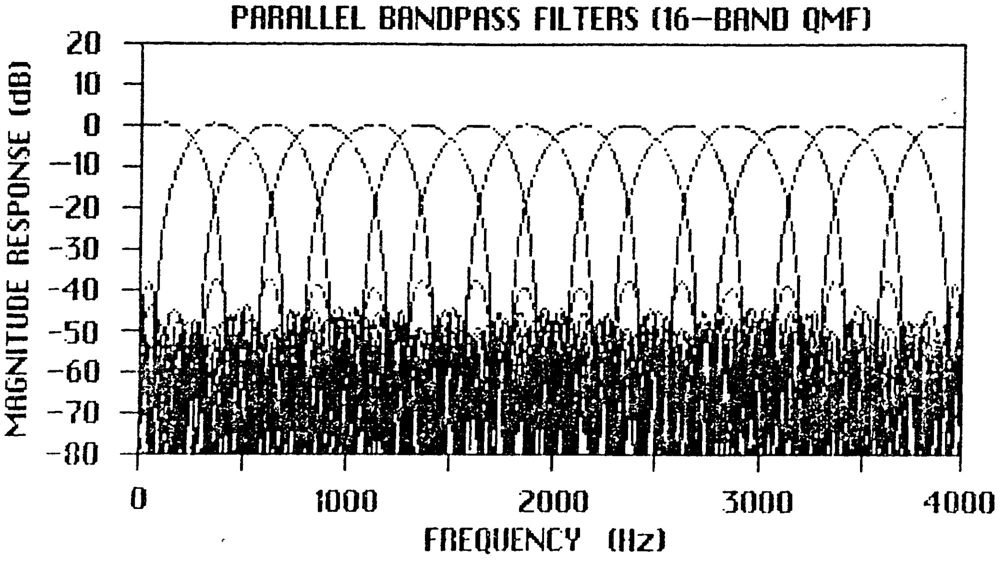


Figure 2-9. 16-Band QMF Bandpass Filters

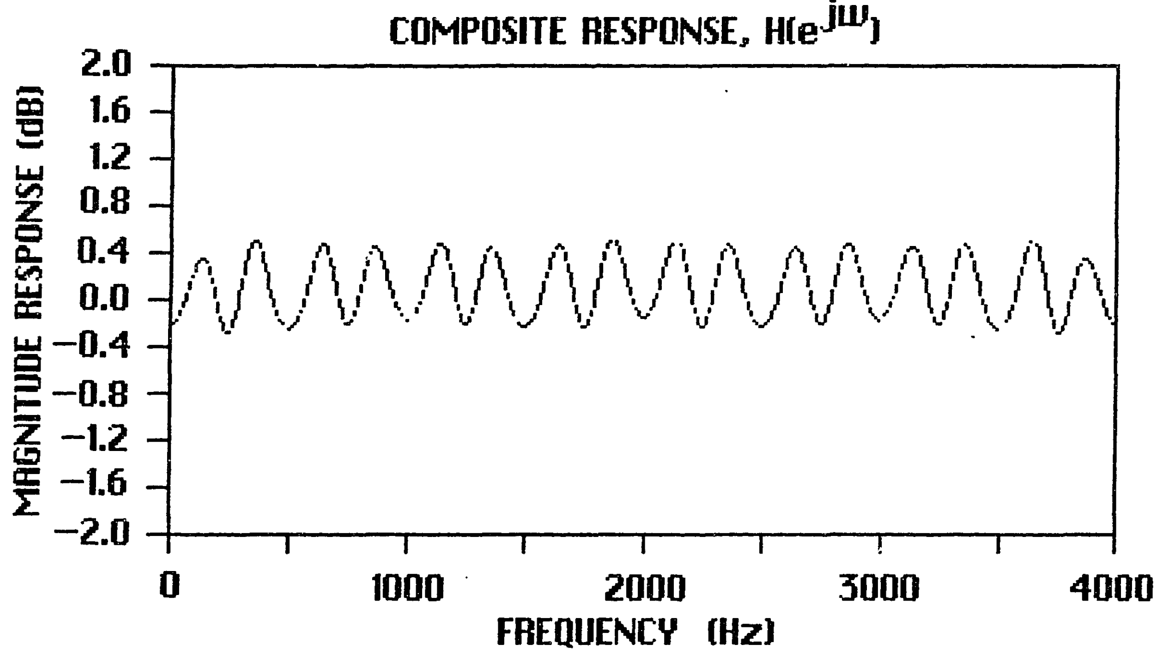


Figure 2-10. Composite Response, 16-Band QMF

The inherent symmetry of the half-band filters is manifested in certain properties of the parallel filters.

(2-12) $f_{N-1-k}(n) = (-1)^n f_k(n)$

(2-13) $g_k(n) = N(-1)^k f_k(n)$

These relationships are useful in reducing the amount of computation needed for QMF analysis and reconstruction.

## 2.3 BIT ALLOCATION

The goal of the bit allocation algorithm is to determine the number of bits available for sample encoding (from the specified coder bit rate), and to distribute them among the 13 subbands based upon relative power. Bit allocation is recalculated for every block of 128 input samples (16ms) to dynamically adjust to changes in the input power spectrum.

It is shown in [47] that the total quantization error of a subband coder is minimized with the following bit distribution

(2-14) $$n(i) = \frac{M}{N} - \frac{1}{N}\sum_{j=0}^{N-1} \log_2\sqrt{E_j} + \log_2\sqrt{E_i}$$

where $n_i$ and $E_i$ are the number of bits and energy, respectively, of the $i^{th}$ band, $0 \leq i \leq N-1$. M is the total number of bits available per subband sampling interval.

$$M = \sum_{i=0}^{N-1} n_i \tag{2-15}$$

This algorithm requires the computation of the energy in each subband. Esteban and Galand suggest a simple alternative [47]. They estimate the energy of a subband during a particular block to be proportional to the square of the maximum sample amplitude.

$$E_i = a^2 C_i^2 \quad \text{for some constant } a \tag{2-16}$$

$$\text{where} \quad C_i = \max_{j=1,p} |s_i(j)|$$

$C_i$, the so-called subband characteristic, is the largest value in the $i^{th}$ subband's current block of p samples. Substituting this into (2-14) gives

$$n(i) = \frac{M}{N} - \frac{1}{N} \sum_{j=0}^{N-1} \log_2 C_j + \log_2 C_i \tag{2-17}$$

## 2.3.1 Adjustment Algorithm

In this paper, n(i) is restricted to the range $0 \leq n(i) \leq 5$ as recommended in [48] and [49]. However, equation (2-17) does not always produce integer values in the desired range. Fig. 2-11 illustrates such a case. Clearly, an adjustment algorithm is needed to determine actual bit allocation based upon the n(i) values.

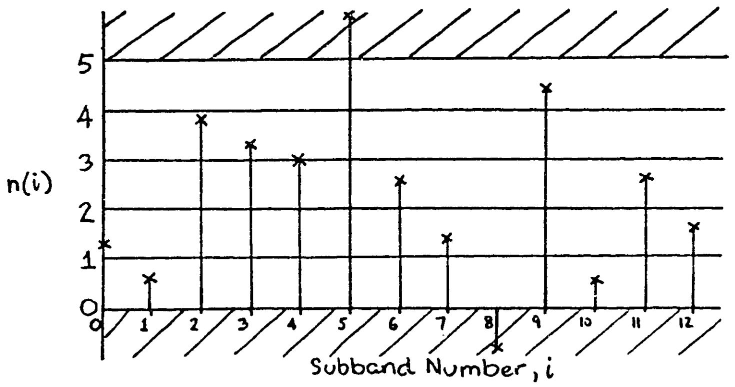


Figure 2-11. n(i) vs. i

The first step is to come up with an initial allocation by assigning 5 bits to bands where n(i) > 5, 0 bits where n(i) < 0, and rounding down the others to integers. Fig. 2-12, for example, shows the preliminary assignments for the distribution of Fig. 2-11.

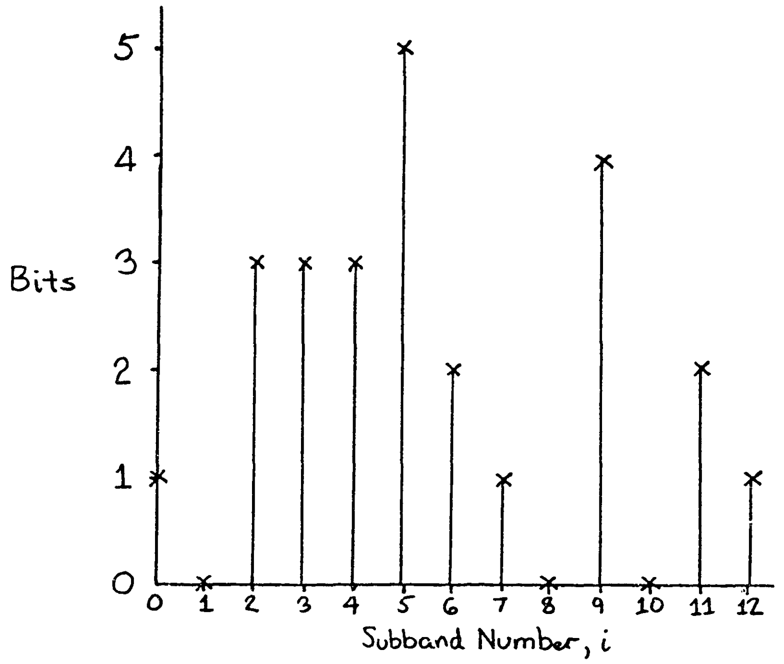


Figure 2-12. Initial Allocation

The next step is to check that the total number of bits distributed matches the number that are available. If so, then the allocation is done. If not, then a reassignment is necessary.

The adjustment algorithm used is related to that suggested by Jayant and Noll [6,p.530]. The basic idea is to add or subtract the same constant from all of the n(i) computed from (2-17). In this manner, more or less bits can be allocated while still maintaining the relative sizes of the n(i). Graphically, this corresponds to shifting the vertical scale of Fig. 2-11 up or down until the bit allocation equals the available supply. For example, if the initial calculation gives out too many, then all of the n(i) are uniformly reduced (the vertical scale is raised) until enough bits are removed from the bands.

The adjustment begins by determining how close the initial allocation comes to the available bit rate. If there are extra bits left to distribute, the scale is lowered by the minimum amount needed to increase the bits given to any band by one. Fig. 2-13 demonstrates the effect of this shift on Fig. 2-11. The bit assignment is then revised based upon this new scale. If this gives additional bits to more than one band, the lower bands (which tend to have more energy) receive them first, until all the extra bits are used up. If the highest band is reached and more bits are still available for distribution, the scale is lowered again and the process repeated. Clearly, this algorithm must terminate unless assigning the maximum of 5 bits to every band is not sufficient to exhaust the bit resources. This special case is tested for before the adjustment loop is entered.

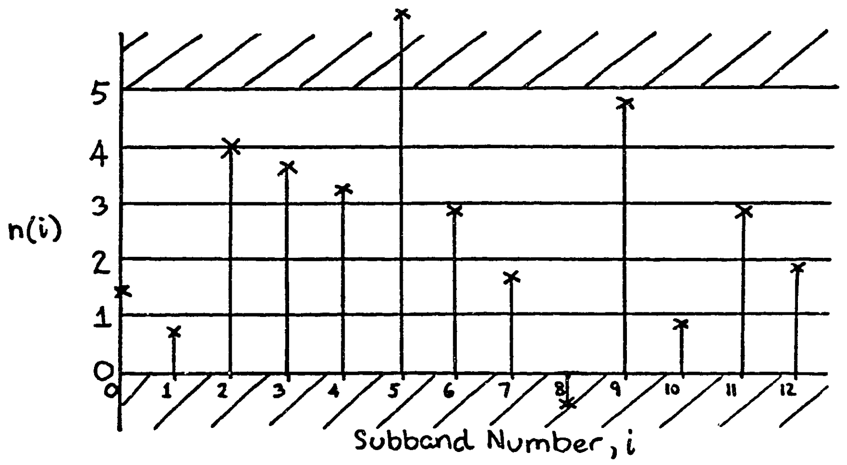


Figure 2-13. Effect of Lowered Scale

A similar strategy is used when the initial bit allocation assigns more bits than are available. When this happens, the vertical scale is raised by the minimum needed to take bits away from the subbands. If more than one band is affected by the shift, bits are first subtracted from the higher frequencies. Additional shifts are performed if necessary. This continues until enough bits have been subtracted from the bands, or until all bands have no bits remaining, whichever comes first.

An important consideration in the design of the adjustment algorithm is that it should always converge. The proposed routine insures this by providing a means for increasing or decreasing the bit allocation by one bit at a time.

### 2.3.2 Average Bits Per Sample

Both quantized data and overhead information contribute to the bit rate of a coder. In a sub-band coding system, overhead is needed by the transmitter to somehow convey to the receiver the number of bits used to code each band.

In some designs [27,47,50], the subband characteristics, C(i), are sent to the receiver, which repeats the transmitter's bit allocation calculation. In those coders, the C(i) are also used to direct the adaptation of the quantizer. However, in this design, quantizer adaptation is independent of the characteristics (see Section 2.4), so actual bit allocation results can be sent instead of the C(i). This is desirable because the latter consumes more overhead. Since subband characteristics are selected channel samples, they can take on many different values and, hence, accurately quantizing them for transmission requires many bits. On the other hand, actual bit assignments (0,1,2,3,4,5) can be exactly specified with only three bits each. In the simulated coder, this results in an overhead cost of

$$(13 \text{ bands})(3 \text{ bits per band})/(16 \text{ ms}) = 2.4375 \text{ kbps}$$

Therefore, the average number of bits available for quantizing each subband sample is given by

$$(2\text{-}18)\quad B_{avg} = \left(\frac{(\text{SBC bit rate}) - 2.4375 \text{ kbps}}{\text{number of coded bands}}\right)\left(\frac{1}{\text{subband sampling rate}}\right)$$

where the number of coded bands is 13 and the subband sampling rate is 1/16 of the 8 kHz input rate, or 0.5 kHz. Thus, this implementation allows 3.3 bits per sample at 24 kbps and 2.1 bits per sample at 16 kbps.

### 2.3.3 Bit Allocation With Prediction

For reasons that will become clear in Sections 2.4 and 2.5, proper adaptation of the predictors requires that subbands which use differential coding must be allocated at least two bits.

The bit allocation algorithm previously described does not distinguish between bands that use prediction and those that do not. As a result, some predicted bands may be assigned less than two bits. This condition is checked for and, if found, corrected by drawing bits from the higher frequencies. This may not be possible, however, if the higher bands run out of bits. When this occurs, execution continues but a warning message is issued.

## 2.4 ADAPTIVE QUANTIZER

Each subband has a distinct quantizer with an associated step size and number of bits. The adaptive quantizer implemented in this thesis is a uniform, mid-rise design based upon that of Hamel, Soumagne, and Le Guyader [32].

The choice of quantizer involves two main issues. The first is its method of adaptation, namely, how the step size is updated. The second is its behavior in the presence of dynamic bit allocation.

### 2.4.1 Step Size Adaptation

For the purpose of discussing the step size update scheme, assume for now that the bit allocation is always five.

Fig. 2-14 illustrates the 5-bit quantizer. The 32 levels occur at odd multiples of Δ/2 where Δ is the step size. Each level is used to represent the range of input values of width Δ centered around it. For example, level 3 corresponds to 2.5Δ and covers inputs from 2Δ to 3Δ (the lower bound, 2Δ, is included but not the upper bound, 3Δ). All inputs greater than or equal to 15Δ are assigned to the highest level, 16. Similarly, all inputs less than or equal to -15Δ are represented by the -16 level.

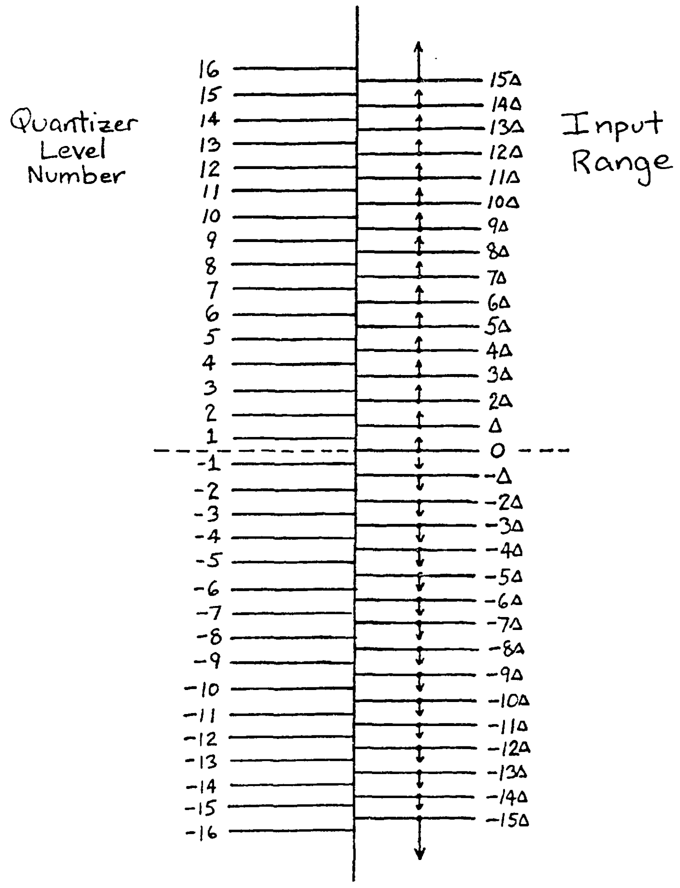


Figure 2-14. 5-bit Quantizer

Notice that positive and negative inputs are symmetrically rounded away from zero. This prevents any bias in the quantizer operation, i.e. inputs with average value zero will be quantized to an average of zero as well. This is not the case, for example, if inputs are always rounded down to obtain quantized levels; such a scheme would produce a negative bias. An exception to this symmetry is the handling of zero inputs. An ideal procedure would be to randomly pick either level 1 or -1. However, since zero inputs are rare, level 1 is selected in all instances.

Step sizes are adapted using Jayant's method [8]. After each sample, the step size is multiplied by an update factor determined by the value of the previous output level.

(2-19) $\Delta(i) = \Delta(i-1) \times M(|\text{level}(i-1)|)$

where M is known as the multiplier function. The motivation behind this is the observation that high output levels indicate too small a step size whereas low outputs suggest too large a step. M is greater than 1 for high levels and less than 1 for levels close to zero. Recommended values of M for a 5-bit quantizer are given in Table 2-1 [32].

| \|level\| | M(\|level\|) |
|---|---|
| 16 | 1.70 |
| 15 | 1.62 |
| 14 | 1.54 |
| 13 | 1.46 |
| 12 | 1.39 |
| 11 | 1.31 |
| 10 | 1.24 |
| 9 | 1.18 |
| 8 | 1.12 |
| 7 | 1.06 |
| 6 | 1.01 |
| 5 | .96 |
| 4 | .93 |
| 3 | .90 |
| 2 | .88 |
| 1 | .87 |

Table 2-1. Recommended 5-bit Quantizer Multiplier Values

Notice that step sizes can increase faster than they can decrease. This is attributed to the similar behavior of speech signals.

Maximum and minimum values for the step size, $\Delta_{max}$ and $\Delta_{min}$, are usually included as bounds on the adaptation process. In addition, an initial value, $\Delta_{start}$, is needed. These values depend on the power of the inputs and are determined in Chapter 3.

### 2.4.2 Effects of Variable Bit Allocation

Since a quantizer is characterized by, among other things, the number of bits it uses, and bits are allocated dynamically in the sub-band coder, some procedure must be developed to easily switch between 0, 1, 2, 3, 4, and 5 -bit quantizers. Furthermore, it is desirable that the transitions somehow preserve the step size adaptation process.

Hamel, Soumagne, and Le Guyader [32] present an elegant solution to this problem using the 5-bit quantizer of Fig. 2-14 as a basis. Quantizers of fewer bits are simply defined as level subsets of the 32 5-bit levels. The fundamental step size is retained between transitions and is updated with the same multipliers used by the 5-bit quantizer (Fig. 2-15). Inputs are still symmetrically rounded away from zero, i.e. values which fall between two levels are assigned to the one that is further from zero.

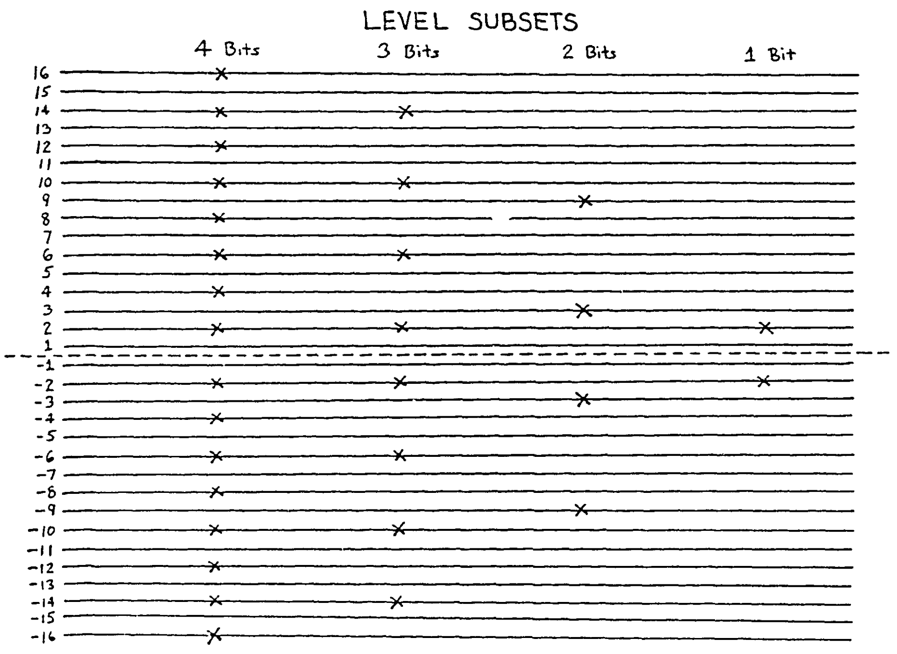


Figure 2-15. Variable Bit Quantizers

An example will make this method clearer. Assume that after a block of 8 subband samples has just been coded with 5 bits, the step size is 2.0. Now consider the effect of only 3 bits being assigned to that subband for the next block. The step size will initially remain 2.0. If the next quantizer input is, say, 15.3, then this would put it between levels 2 and 3 of the 3-bit quantizer (corresponding to levels 6 and 10, respectively, of the original 5-bit quantizer). This is rounded to level 3 and the step size is multiplied by M(10)=1.24 to obtain the new step size, 2.48.

The 0-bit quantizer does not change Δ at all and always gives an output of 0, which is not associated with any level.

### 2.4.3 Sample-to-Sample vs. Block Adaptation

Jayant's adaptation scheme updates step sizes from sample-to-sample. An alternate method, known as Block Companded Pulse Code Modulation (BCPCM), was developed by Galand [2,27,29,47,49].

In BCPCM, the step size is constant within each block of subband samples and is given by

(2-20) $$\Delta = C/2^{b-1}$$

where C and b are the characteristic and bit allocation, respectively, of that subband (Fig. 2-16). In this way, the range of the quantizer has the same limits as the range of the subband signal over that block.

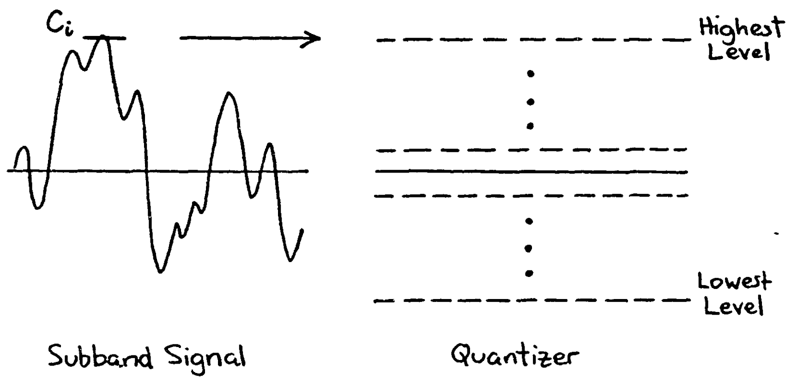


Figure 2-16. Block Companded PCM

In this study, Jayant's sample-to-sample technique was chosen over Galand's BCPCM because of the presence of predictors. In differential coding, the quantizer input is not the subband signal but instead, the prediction error. If BCPCM is used, the quantizer's range, which is modeled from the subband characteristic, will probably be larger than that of its input, the difference signal.

## 2.5 ADAPTIVE PREDICTORS

The two adaptive predictors compared in this study are the least-mean-square transversal and the least-squares lattice. Each generates predicted values that are linear combinations of past samples.

### 2.5.1 LMS Transversal

The LMS transversal predictor [12] is conceptually very simple (Fig. 2-17). The tapped delay line structure produces a weighted sum of past inputs at every sampling period.

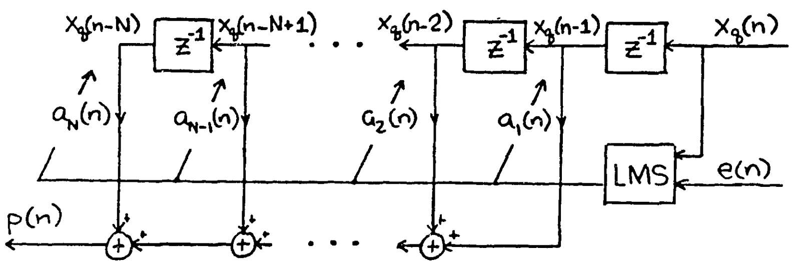


Figure 2-17. LMS Transversal Predictor

For each new sample, the coefficients are updated in a direction which tends to minimize the mean-square error of the prediction.

(2-21) $$a_m(n+1) = a_m(n) - c\frac{\partial}{\partial a_m(n)}\langle e^2(n)\rangle$$

where c is some positive number that controls the rate of adaptation.

The form of (2-21) makes intuitive sense. If the derivative of $\langle e^2(n)\rangle$ with respect to $a_m(n)$ is positive, then the mean-square error increases with larger $a_m(n)$. Hence, the coefficient should be decreased to improve performance. (Since the derivative represents the gradient of the mean-square error function, the LMS algorithm is said to use a gradient search to find the optimal coefficients.)

In practice, (2-21) cannot be used without modification because the gradient of the mean-square error is never known exactly. Instead, the derivative of the instantaneous square error is used as an approximation

(2-22) $$\frac{\partial}{\partial a_m(n)}\langle e^2(n)\rangle \simeq \frac{\partial}{\partial a_m(n)}e^2(n)$$

This can be expressed as

(2-23) $$\begin{aligned}\frac{\partial}{\partial a_m(n)}e^2(n) &= 2e(n)\frac{\partial}{\partial a_m(n)}e(n)\\ &= 2e(n)\frac{\partial}{\partial a_m(n)}\left(x(n) - \sum_{i=1}^{N} a_i(n)x_q(n-i)\right)\\ &= -2e(n)x_q(n-m)\end{aligned}$$

Also, for stable adaptation, Widrow and Stearns [51] show that c should be inversely proportional to the product of the input power and the order of prediction

(2-24) $$c = \frac{g/2}{N\langle x_q^2(n)\rangle} \quad \text{for some positive gain } g$$

Substituting (2-23) and (2-24) into (2-21) yields

(2-25) $$a_m(n+1) = a_m(n) + \frac{g}{N}\,\frac{e(n)x_q(n-m)}{\langle x_q^2(n)\rangle}$$

The input signal power is highly variable over time. To track such changes, $\langle x_q^2(n)\rangle$ is calculated as an exponentially weighted sum decaying backwards in time

(2-26) $$\langle x_q^2(n)\rangle_T = (1-\alpha)\sum_{i=0}^{\infty} \alpha^i x_q^2(T-i)$$

Furthermore, a constant bias, $\beta$, is added to the power estimate so that c does not blow up during moments of silence. The complete expression for LMS adaptation is then

(2-27) $$a_m(n+1) = a_m(n) + \frac{g}{N}\,\frac{e(n)x_q(n-m)}{(1-\alpha)\sum_{i=0}^{\infty}\alpha^i x_q^2(n-i) + \beta}$$

With the exception of the 1/N factor, this is identical to the LMS update equation used by Cohn and Melsa in their full band ADPCM system [12].

### 2.5.2 LS Lattice

The LS lattice predictor (Fig. 2-18) differs from the LMS transversal both in structure of implementation and in criterion of adaptation. Whereas the LMS variables have obvious interpretations as linear predictor coefficients, the meaning of the LS variables is less apparent. In fact, predictor coefficients are never explicitly computed in the LS lattice implementation.

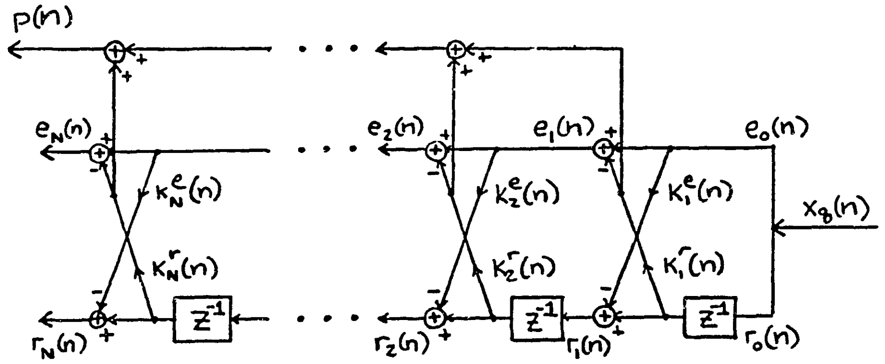


Figure 2-18. LS Lattice Predictor

A formal derivation of the lattice variable interdependencies and the least-squares recursive update equations is given in [35]. This section describes the significance attached to these variables and their recursions.

### 2.5.2.1 Forward and Backward Prediction

The lattice, which consists of two rows interconnected by weighted cross-paths, resembles a ladder with rungs (hence the alternative name, ladder form). The predictor output is the sum of values taken from the cross-paths.

The variables in the top row, $e_m(n)$, are referred to as the forward prediction errors and are associated with the forward predictor coefficients, $a_i(n)$.

$$(2\text{-}28) \qquad e_m(n) = x_q(n) - \sum_{i=1}^{m} a_i(n)x_q(n-i)$$

These are the errors in predicting $x_q(n)$ using the m forward coefficients of time n. This definition can be generalized to

$$(2\text{-}29) \qquad e_m(t,n) = x_q(t) - \sum_{i=1}^{m} a_i(n)x_q(t-i)$$

where $e_m(t,n)$ is the error in predicting $x_q(t)$ with $a_i(n)$.

The variables in the bottom row, $r_m(n)$, are known as the backward prediction errors. (In contrast to forward prediction, backward prediction involves the estimation of sample values based upon future inputs.)

$$(2\text{-}30) \qquad r_m(n) = x_q(n-m) - \sum_{i=1}^{m} b_i(n)x_q(n-i+1)$$

where $b_i(n)$ are the backward coefficients. In general,

$$(2\text{-}31) \qquad r_m(t,n) = x_q(t-m) - \sum_{i=1}^{m} b_i(n)x_q(t-i+1)$$

represents the error in predicting $x_q(t-m)$ as a weighted sum of the m samples immediately following it.

### 2.5.2.2 Recursions

It can be shown that the $a_i(n)$ and $b_i(n)$ which minimize the accumulated errors

$$(2\text{-}32a) \qquad \sum_{t=0}^{n} w^{n-t} e_m^{\,2}(t,n) \qquad 0 < w < 1$$

(2-32b) $$\sum_{t=0}^{n} w^{n-t} r_m^2(t,n) \qquad 0 < w < 1$$

also give prediction errors that obey the order recursions

(2-33a) $e_{m+1}(t,n) = e_m(t,n) - K^r_{m+1}(n) r_m(t-1,n-1)$

(2-33b) $r_{m+1}(t,n) = r_m(t-1,n-1) - K^e_{m+1}(n) e_m(t,n)$

$$\text{where } K^e_{m+1}(n) = \frac{C_{m+1}(n)}{R^e_m(n)} = \frac{\sum_{t=1}^{n} w^{n-t} r_m(t-1,n-1) e_m(t,n)}{\sum_{t=1}^{n} w^{n-t} e_m^2(t,n)}$$

$$\text{and } K^r_{m+1}(n) = \frac{C_{m+1}(n)}{R^r_m(n-1)} = \frac{\sum_{t=1}^{n} w^{n-t} r_m(t-1,n-1) e_m(t,n)}{\sum_{t=1}^{n} w^{n-t} r_m^2(t-1,n-1)}$$

are referred to as the reflection coefficients. By recognizing that $r_m(n) = r_m(n,n)$ and $e_m(n) = e_m(n,n)$, and evaluating (2-33) for t=n, this becomes

(2-34a) $e_{m+1}(n) = e_m(n) - K^r_{m+1}(n) r_m(n-1)$

(2-34b) $r_{m+1}(n) = r_m(n-1) - K^e_{m+1}(n) e_m(n)$

It is this set of order recursions that gives rise to the ladder structure.

The numerator and denominator of the reflection coefficients are exponentially weighted sums that may be recursively calculated in time as follows

(2-35a) $C_{m+1}(n) = wC_{m+1}(n-1) + r_m(n-1,n-1) e_m(n,n)$

$= wC_{m+1}(n-1) + r_m(n-1) e_m(n)$

(2-36a) $R^e_{m+1}(n) = wR^e_{m+1}(n-1) + e^2_{m+1}(n,n)$

$= wR^e_{m+1}(n-1) + e^2_{m+1}(n)$

(2-37a) $R^r_{m+1}(n) = wR^r_{m+1}(n-1) + r^2_{m+1}(n,n)$

$= wR^r_{m+1}(n-1) + r^2_{m+1}(n)$

$C_m(n)$, which is the numerator of both reflection coefficients, is called the partial correlation (PARCOR) variable since it (exponentially) averages the product of the forward prediction error at time n and the backward prediction error at time n-1. $R_m{}^r(n)$ is roughly proportional to the short-time variance of the backward error and, similarly, $R_m{}^e(n)$, measures the forward error variance.

An important feature of the least-squares algorithm is the modification of the time update equations (2-35a)-(2-37a) by the inclusion of a gain variable, $\gamma_m(n)$.

(2-35b) $$C_{m+1}(n) = wC_{m+1}(n-1) + \frac{r_m(n-1)e_m(n)}{1-\gamma_{m-1}(n-1)}$$

(2-36b) $$R^e_{m+1}(n) = wR^e_{m+1}(n-1) + \frac{e^2_{m+1}(n)}{1-\gamma_m(n-1)}$$

(2-37b) $$R^r_{m+1}(n) = wR^r_{m+1}(n-1) + \frac{r^2_{m+1}(n)}{1-\gamma_m(n)}$$

(2-38) where $$\gamma_m(n) = \gamma_{m-1}(n) + \frac{r^2_m(n)}{R^r_m(n)}$$

$\gamma$ can be interpreted as a likelihood detector. It measures the deviation of data from an expected Gaussian distribution. For data that fits the assumed statistics, $\gamma$ is close to zero, so it has little effect on the update. For very unlikely data, $\gamma$ will approach 1, which can dramatically increase the update term. In this way, sudden changes in inputs can be tracked very quickly. The presence of $\gamma$ turns out to be the main difference between lattice implementations of the LS and LMS algorithms [37,39].

### 2.5.2.3 Time Initialization

Before the first input sample is received, the forward and backward error variances at each of the N stages of the predictor are set to some small positive number, $\delta$, so that the corresponding reflection coefficients are initially defined. Also, the PARCOR variable starts at zero at each stage.

### 2.5.2.4 Order Initialization

The 0th order errors, $e_0(n)$ and $r_0(n)$, are always set to the value of each new input sample. This makes sense since the error of a 0th order predictor is simply the signal being predicted. Correspondingly, the error variances, $R_0{}^e(n)$ and $R_0{}^r(n)$, are updated with the square of the input.

The likelihood variable, $\gamma$, is defined to be zero at stage -1.

### 2.5.2.5 Predictor Output

The Nth order forward prediction error is, from equation (2-34a),

(2-39) $$e_N(n) = e_{N-1}(n) - K^r_N(n)r_{N-1}(n-1)$$

Repeated decomposition of the $e_m(n)$ terms eventually gives

(2-40) $$e_N(n) = e_0(n) - \sum_{m=1}^{N} K_m^r(n) r_{m-1}(n-1)$$

$$= x_q(n) - \sum_{m=1}^{N} K_m^r(n) r_{m-1}(n-1)$$

But the prediction error is just the difference between the input and the predictor output and so

(2-41) $$p(n) = \sum_{m=1}^{N} K_m^r(n) r_{m-1}(n-1)$$

Thus, the LS lattice comes up with p(n) without actually storing past input samples or forward predictor coefficients.

**2.5.2.6 Equation Summary**

The basic equations for the LS lattice predictor are summarized in Table 2-2.

| TIME | ORDER |
|---|---|
| **Initialize** | |
| $R_m^e(0)=R_m^r(0)=\delta$ | $e_0(n)=r_0(n)=x_q(n)$ |
| $C_m(-1)=0$ | $\gamma_{-1}(n-1)=0$ |
| **Iterate** | |
| $C_{m+1}(n) = wC_{m+1}(n-1) + \dfrac{r_m(n-1)e_m(n)}{1-\gamma_{m-1}(n-1)}$ | $\gamma_m(n) = \gamma_{m-1}(n) + \dfrac{r_m^2(n)}{R_m^r(n)}$ |
| $R_{m+1}^e(n) = wR_{m+1}^e(n-1) + \dfrac{e_{m+1}^2(n)}{1-\gamma_m(n-1)}$ | $e_{m+1}(n) = e_m(n) - K_{m+1}^r(n)r_m(n-1)$ |
| $R_{m+1}^r(n) = wR_{m+1}^r(n-1) + \dfrac{r_{m+1}^2(n)}{1-\gamma_m(n)}$ | $r_{m+1}(n) = r_m(n-1) - K_{m+1}^e(n)e_m(n)$ |
| | where $K_{m+1}^e(n) = \dfrac{C_{m+1}(n)}{R_m^e(n)}$ |
| | and $K_{m+1}^r(n) = \dfrac{C_{m+1}(n)}{R_m^r(n-1)}$ |

Output

$$p(n) = \sum_{m=1}^{N} K_m^r(n)r_{m-1}(n-1)$$

Table 2-2. LS Lattice Equations

## 2.5.3 Prediction and Bit Allocation

The LMS predictor uses the error signal in updating its coefficients (equation (2-27)). However, only a quantized version of the error signal is available (see Fig. 1-8), thereby making LMS performance highly sensitive to quantizer accuracy.

In the extreme case of 0 bit allocation, the quantized error signal is always zero, indicating to the LMS algorithm that its prediction is perfect, regardless of its actual performance. Similarly, a 1-bit quantizer has an output that is always decreasing in magnitude (the step size multiplier is .88) so the error signal is represented as continuously improving even if it is not. For this reason, bands which use LMS prediction should always be given at least two bits for quantization.

The LS lattice does not depend on the error signal as directly as the LMS transversal and is therefore not as sensitive to 0 or 1-bit quantizer degradation. However, for better representation of past inputs, which the predictor sees as the sum of its output and the quantized error, at least two bits are required for the LS predicted bands as well.

# CHAPTER 3 - SIMULATION RESULTS

The speech coding algorithms described in Chapter 2 are implemented in FORTRAN and used to process voice samples on an IBM VM/SP mainframe computer (see Appendix C for program listings). The results of the simulations are presented below. The next chapter interprets and discusses these data.

## 3.1 TEST OVERVIEW

Several aspects of the sub-band coders are evaluated. In conjunction with the ultimate test of how good the processed sentences sound, some diagnostic data is obtained to help analyze the behavior of the coder components.

Four phonetically balanced sentences [53], two spoken by males and two by females, are used as the basis for the simulations.

| | | |
|---|---|---|
| Female Speakers | | |
| | DARCI : | The frosty air passed through the coat. |
| | BETH : | The small pup gnawed a hole in the sock. |
| Male Speakers | | |
| | GLENN : | The meal was cooked before the bell rang. |
| | MIKE : | Hoist the load to your left shoulder. |

(In the rest of this report, these sentences will be referred to by the names of the speakers.)

The next two sections give performance data on the filter bank and the bit allocation algorithm. This is followed by three sections dealing with the quantizer and predictors, including the determination of some of their parameters. Finally, objective and subjective measures are given for overall coder quality using these parameters.

## 3.2 QMF PERFORMANCE

To isolate the performance of the Quadrature Mirror Filters, the sub-band coder is run on the four test sentences without any quantization. This is done by passing the output of the analysis QMF directly to the reconstruction QMF.

Since there is no quantization, the bit rate of the coder is not meaningful in this situation. SNR (signal-to-noise ratio) and SSNR (segmental signal-to-noise ratio) values as defined in equations (1-1) and (1-2) are obtained for each sentence

| | DARCI | BETH | GLENN | MIKE |
|---|---|---|---|---|
| SNR (dB) | 30.83 | 30.99 | 30.55 | 29.38 |
| SSNR (dB) | 26.51 | 25.96 | 26.47 | 27.22 |

Table 3-1. Isolated QMF Performance

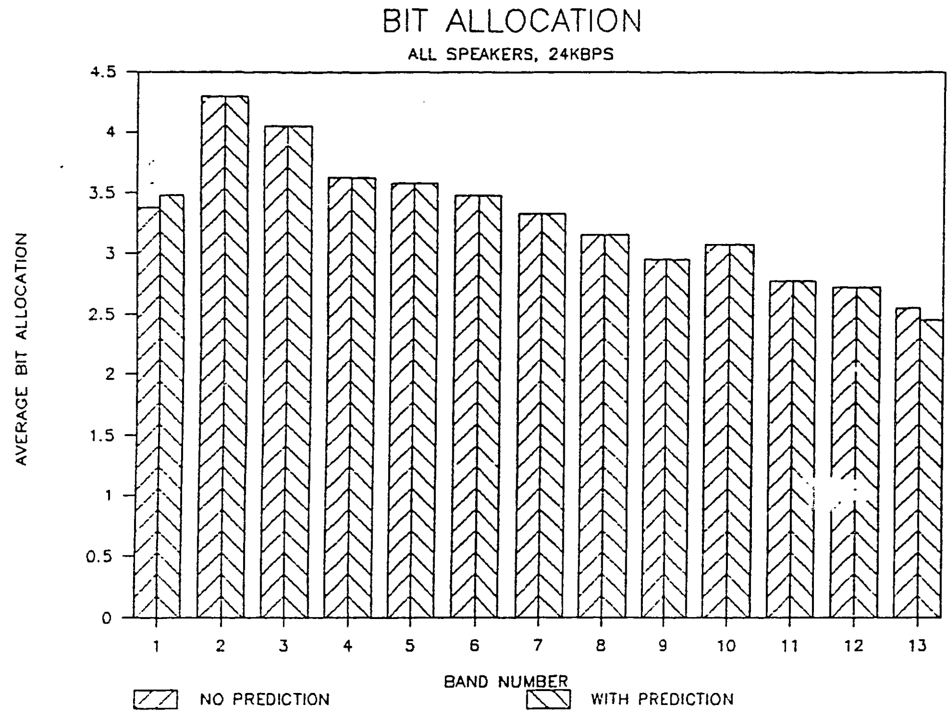
Informal listening tests indicate that the filter bank alone introduces very little distortion to the original sentences.

## 3.3 BIT ALLOCATION STATISTICS

The bit allocation algorithm distributes available bits to the subbands on the basis of relative power. As explained in Section 2.3, the power of each band is not actually calculated. Instead, the power is estimated to be proportional to the square of the maximum sample amplitude, C(i). These so-called characteristics determine the relative bit distribution for each 16 ms block of input samples.

In the presence of prediction, the allocation is adjusted to make sure that the first four subbands have at least two bits. At 24 kbps, the adjustment is minor. At 16 kbps, these bands are more likely to get fewer than two bits, so the modification is more pronounced.

Simulations demonstrate that mean bit allocations for the test sentences are similar. Distributions averaged over all four speakers at 16 kbps and 24 kbps are illustrated in Fig. 3-1. Complete bit allocation statistics for individual sentences, as well as minimum, maximum, and average subband characteristic values, are given in Appendix D.



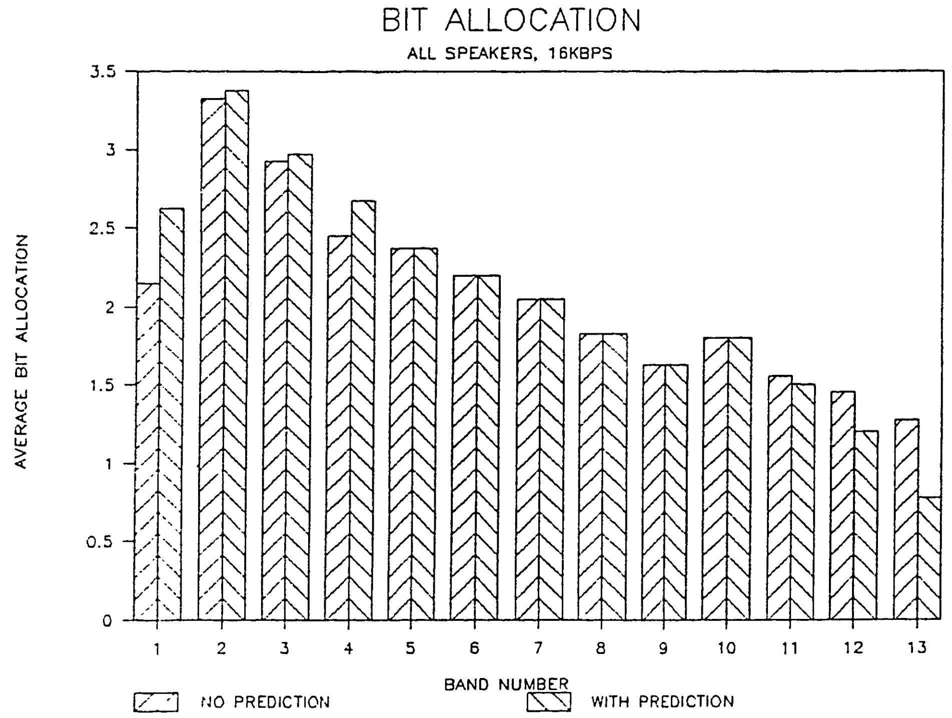


**Figure 3-1. Average Bit Allocation (four sentences)**

Notice that the lower bands, especially numbers 2 and 3 (250 to 750 Hz), typically get the most bits. This is where the voice spectrum has the most power.

## 3.4 QUANTIZER PARAMETERS

The adaptive quantizer is a 5-bit (32-level) uniform mid-rise design with level subsets used for smaller bit allocations. Adaptation of the step size is handled by Jayant's multiplier rule (see Section 2.4).

The values of the multipliers, M(|level|), are taken directly from those recommended in [32]. The only other parameters necessary to completely specify the quantizer are the maximum, minimum, and starting step sizes ($\Delta_{max}$, $\Delta_{min}$, $\Delta_{start}$). The selection of these values is, of course, dependent upon the expected range of the inputs.

For the four sentences in the test set, values of $\Delta_{max} = 512$ and $\Delta_{min} = .1$ were experimentally determined to provide sufficient range to code the subband samples. $\Delta_{start}$ is chosen to be equal to $\Delta_{min}$ since the beginnings of the test sentences are close to silence.

To determine the degradation in voice quality introduced by quantization, independent of the effectiveness of the bit allocation algorithm, the test sentences are coded without prediction using the maximum of 5 bits per band. This may be achieved by running the sub-band coder at 35 kbps or greater (see equation (2-18)). The results are given in Table 3-2.

| | DARCI | BETH | GLENN | MIKE |
|---|---|---|---|---|
| SNR (dB) | 18.52 | 18.91 | 14.02 | 13.45 |
| SSNR (dB) | 14.39 | 15.69 | 15.21 | 14.10 |

Table 3-2. Coder Performance With Constant Bit Allocation of 5

These values represent the best that the quantizer can do without prediction. Comparing these numbers with those of Table 3-1 shows a loss of more than 10 db in both SNR and SSNR.

## 3.5 PREDICTOR OPTIMIZATION

Each of the two predictors has several parameters that can affect its performance. In order to make a fair comparison between them, parameters that are optimal in some sense should be found for both.

### 3.5.1 Figure of Merit

Parameter optimization requires a specific measure of predictor quality. Such a figure of merit should not only indicate input tracking ability but also consider the relative importance of the subbands.

Since a good predictor yields a difference signal that tends to be much smaller than the input, a reasonable tracking measure is

(3-1) $$r = \frac{\text{NRMS}}{\text{SRMS}} \times 100$$

where NRMS is the root-mean-square noise (prediction error) and SRMS is the root-mean-square signal (predictor input). Thus, r represents the percentage reduction in signal RMS.

In the sub-band coder, predictor performance should be weighted more heavily in the bands with more power. This is incorporated in the expression

(3-2) $$R = \frac{\sum_{i=1}^{4} (r_i)(\text{SRMS}_i)}{\sum_{i=1}^{4} \text{SRMS}_i}$$

where $r_i$ and $\text{SRMS}_i$ are the percentage RMS reduction and signal RMS, respectively, of band i. Only the first four subbands are considered since those are the ones which use prediction.

In order to optimize the predictors to all four test sentences, the figure of merit which is minimized is

(3-3) $$R_{avg} = \frac{1}{4} (R_{DARCI} + R_{BETH} + R_{GLENN} + R_{MIKE})$$

Although predictor inputs are quantized in the SBC-ADPCM system, the figure of merit is calculated from exact subband samples. In this manner, the optimization process is not tied to any particular bit rate. Quantization effects are treated separately in a later section.

### 3.5.2 LMS Optimization

The LMS transversal has four parameters which may be adjusted to improve performance. These are the order of prediction (N), the gain of the update term (g), the exponential weighting factor for input power estimation ($\alpha$), and the constant bias term added to the power estimate ($\beta$).

Minimizing $R_{avg}$ produces the optimal values N=6, g=.65, $\alpha$=.72, and $\beta$=3000. Table 3-3 lists the RMS values that are achieved.

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| Band, i | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ |
| 1 | 19.77 | 67.90 | 8.91 | 33.00 | 12.48 | 33.39 | 31.94 | 73.30 |
| 2 | 97.18 | 197.97 | 28.66 | 68.12 | 93.06 | 151.86 | 333.89 | 447.74 |
| 3 | 146.76 | 266.92 | 32.03 | 65.69 | 106.21 | 133.34 | 279.45 | 320.73 |
| 4 | 255.75 | 430.06 | 48.82 | 97.61 | 73.31 | 85.94 | 211.99 | 229.83 |
| | R = 53.95% | | R = 44.78% | | R = 70.47% | | R = 80.00% | |

$R_{avg}$ = 62.30%

Table 3-3. Optimal Reduction of Signal RMS (LMS Transversal)

To determine how LMS performance varies with order, the optimal g, $\alpha$, and $\beta$ are run with different orders on the test set (Note that for orders other than 6, these three parameters may no longer be optimal. However, experiments show them to be close.) The figure of merit is calculated in each case.

| Order | $R_{avg}$(%) |
|---|---|
| 1 | 101.04 |
| 2 | 81.67 |
| 3 | 74.95 |
| 4 | 65.64 |
| 5 | 63.35 |
| 6 | 62.30 |
| 7 | 63.06 |
| 8 | 64.01 |
| 9 | 64.92 |
| 10 | 65.70 |

Table 3-4. RMS Reduction for Different Orders (LMS Transversal)

### 3.5.3 LS Optimization

Just as in LMS optimization, parameter values are sought which minimize $R_{avg}$. In this case, there are three variables : the prediction order (N), the initial value of the forward and backward prediction error variances ($\delta$), and the exponential weighting factor of the time recursions (w). The optimal values turn out to be N=6, $\delta$=1, and w=.95. These reduce the signal RMS by about 35%, on the average (see below).

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| Band, i | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ |
| 1 | 16.21 | 67.90 | 5.92 | 33.00 | 9.44 | 33.39 | 26.64 | 73.30 |
| 2 | 117.68 | 197.97 | 36.30 | 68.12 | 111.39 | 151.86 | 365.85 | 447.74 |
| 3 | 128.84 | 266.92 | 40.41 | 65.69 | 103.87 | 133.34 | 267.45 | 320.73 |
| 4 | 272.92 | 430.06 | 50.72 | 97.61 | 78.41 | 85.94 | 215.75 | 229.83 |
| | R = 55.63% | | R = 50.43% | | R = 74.93% | | R = 81.72% | |

$R_{avg}$ = 65.68%

Table 3-5. Optimal Reduction of Signal RMS (LS Lattice)

The same values of $\delta$ and w are then used with different N to demonstrate LS behavior at other orders.

| Order | $R_{avg}$(%) |
|---|---|
| 1 | 87.77 |
| 2 | 79.65 |
| 3 | 74.44 |
| 4 | 69.39 |
| 5 | 66.82 |
| 6 | 65.68 |
| 7 | 65.92 |
| 8 | 66.05 |
| 9 | 66.79 |
| 10 | 67.25 |

Table 3-6. RMS Reduction for Different Orders (LS Lattice)

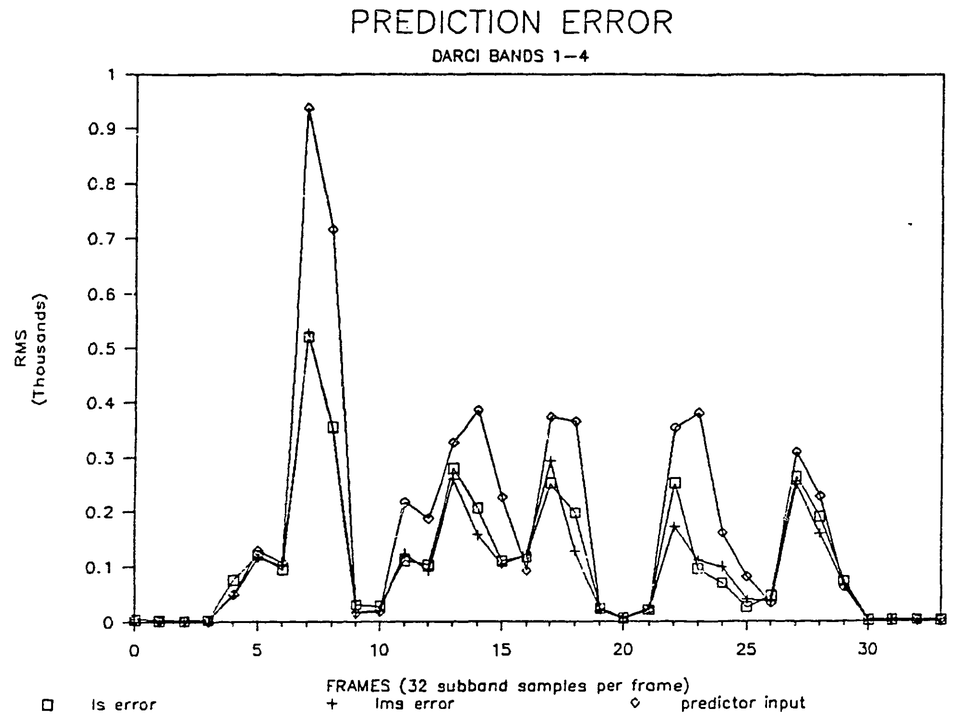
It should be mentioned that in the optimization process as well as in the operational coder, every subband's initial block of eight least-squares outputs is set to zero. This is done because simulations reveal consistently poor prediction for these samples and better performance thereafter.

### 3.5.4 Frame-to-Frame Predictor Performance

So far, predictor performance has only been considered over entire sentences. To see how inputs are tracked on smaller time scales, RMS data is computed over sentence segments.

In [41], Honig and Messerschmitt conduct a frame-to-frame analysis of prediction error RMS in an ADPCM system using a frame length of 500 samples. In SBC-ADPCM, predictors are applied to decimated subbands instead of full-band signals. Hence, a similar analysis is done here for frames of 32 subband samples (equivalent to 512 coder inputs).

Figure 3-2 shows frame-to-frame RMS values of the predictor input along with LMS and LS prediction errors over all four bands for each of the test sentences. Plots for individual subbands are given in Appendix D.



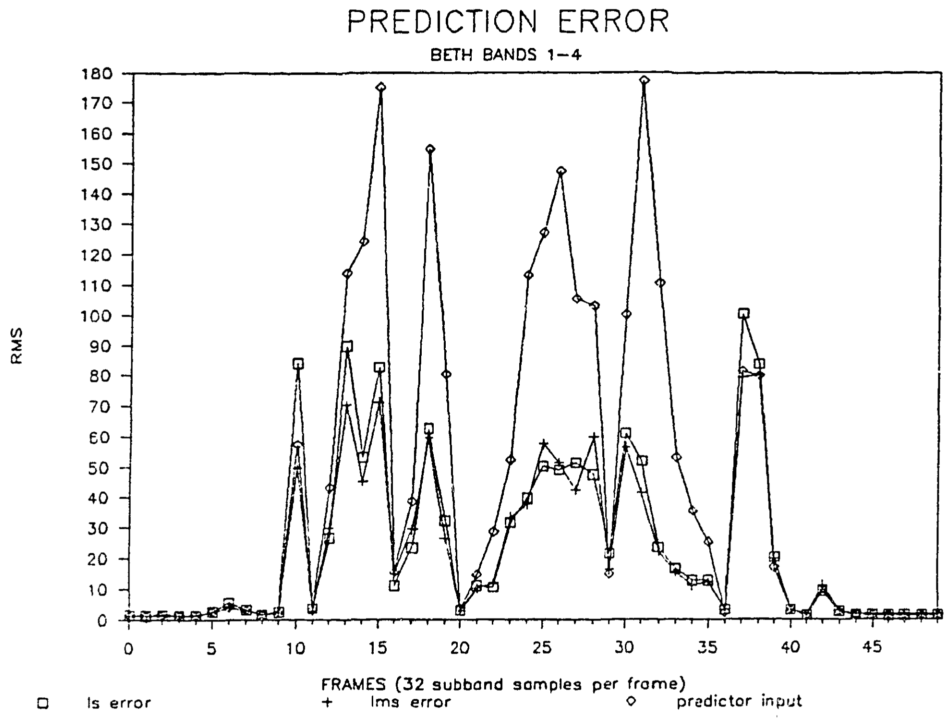
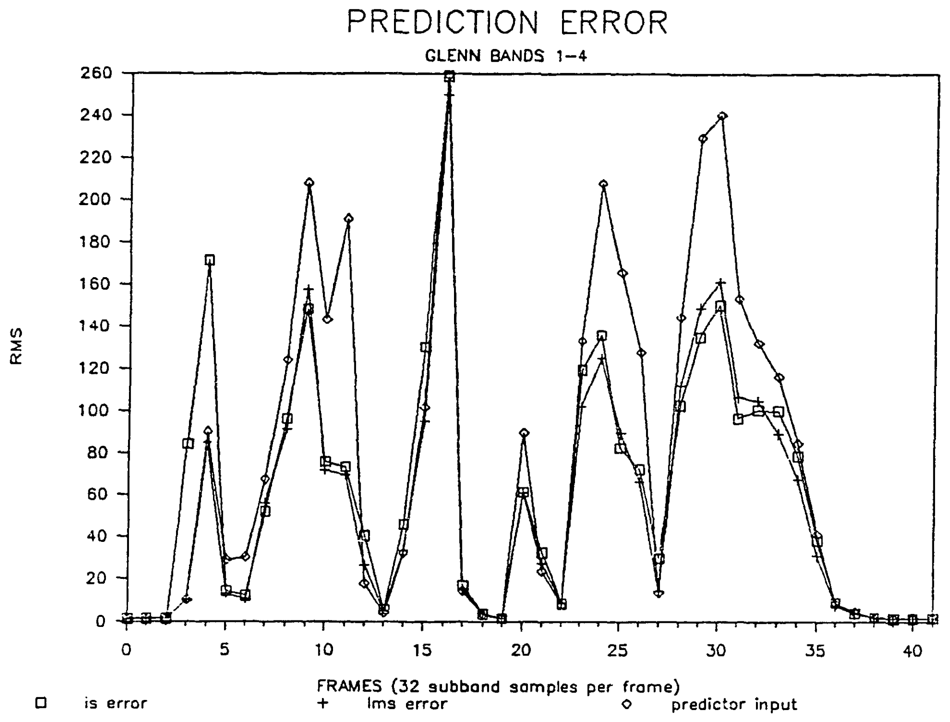


Figure 3-2. Frame-to-Frame Prediction Error



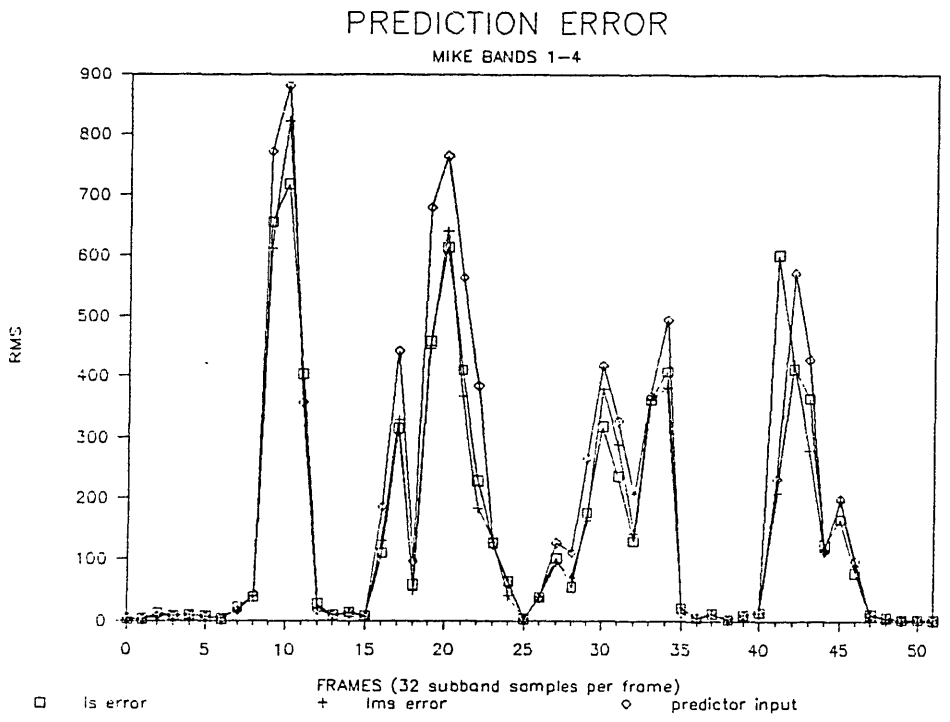


**Figure 3-2. Frame-to-Frame Prediction Error (continued)**

## 3.6 PREDICTOR PERFORMANCE WITH QUANTIZATION

In the previous section, optimal LMS and LS parameters are determined with unquantized inputs. The following data shows how the predictors perform with these parameters at 16 kbps and 24 kbps.

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| Band, i | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ |
| 1 | 27.89 | 67.90 | 9.63 | 33.00 | 15.30 | 33.39 | 46.94 | 73.30 |
| 2 | 104.04 | 197.97 | 29.22 | 68.12 | 94.89 | 151.86 | 337.21 | 447.74 |
| 3 | 155.45 | 266.92 | 33.18 | 65.69 | 105.85 | 133.34 | 282.04 | 320.73 |
| 4 | 253.22 | 430.06 | 49.35 | 97.61 | 74.16 | 85.94 | 217.27 | 229.83 |
| | R = 56.15% | | R = 45.90% | | R = 71.74% | | R = 82.44% | |

$R_{avg} = 64.06\%$

Table 3-7a. Signal RMS Reduction With 16 kbps Quantization (LMS Transversal)

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| Band, i | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ |
| 1 | 28.27 | 67.90 | 8.68 | 33.00 | 17.21 | 33.39 | 46.98 | 73.30 |
| 2 | 112.04 | 197.97 | 35.03 | 68.12 | 115.20 | 151.86 | 347.03 | 447.74 |
| 3 | 153.53 | 266.92 | 35.64 | 65.69 | 104.79 | 133.34 | 272.46 | 320.73 |
| 4 | 282.09 | 430.06 | 51.37 | 97.61 | 82.90 | 85.94 | 219.75 | 229.83 |
| | R = 59.82% | | R = 49.44% | | R = 79.13% | | R = 82.70% | |

$R_{avg} = 67.77\%$

Table 3-7b. Signal RMS Reduction With 16 kbps Quantization (LS Lattice)

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| Band, i | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ |
| 1 | 30.78 | 67.90 | 9.45 | 33.00 | 15.25 | 33.39 | 40.73 | 73.30 |
| 2 | 95.33 | 197.97 | 28.68 | 68.12 | 93.93 | 151.86 | 334.64 | 447.74 |
| 3 | 149.44 | 266.92 | 33.11 | 65.69 | 104.80 | 133.34 | 281.70 | 320.73 |
| 4 | 275.21 | 430.06 | 50.72 | 97.61 | 72.25 | 85.94 | 211.07 | 229.83 |
| | R = 57.20% | | R = 46.12% | | R = 70.76% | | R = 81.01% | |

$R_{avg} = 63.77\%$

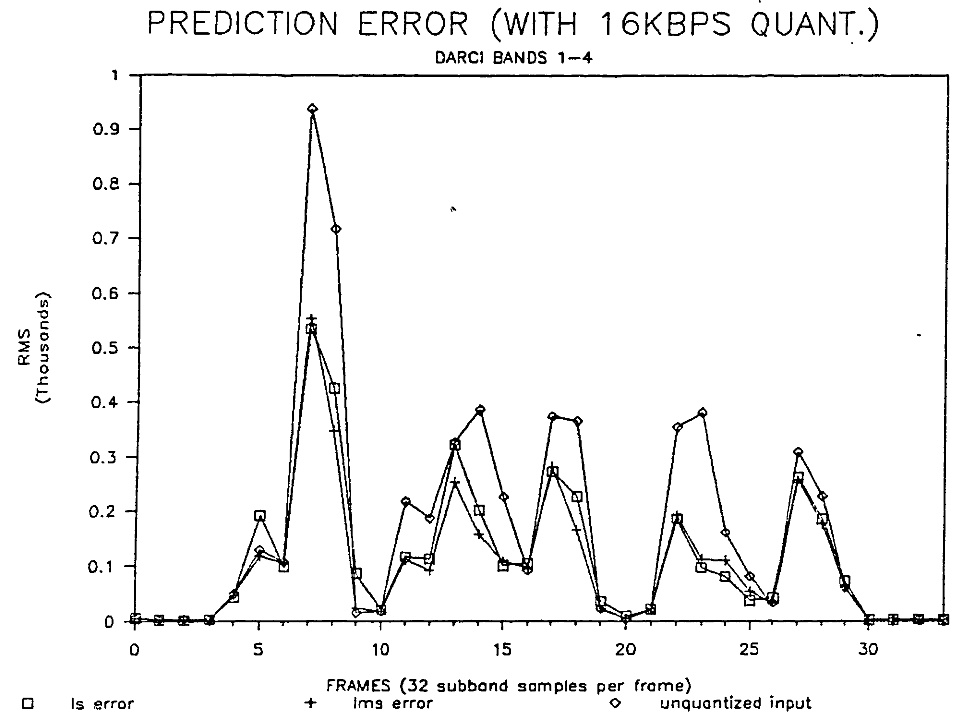
Table 3-8a. Signal RMS Reduction With 24 kbps Quantization (LMS Transversal)

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| Band, i | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ | $NRMS_i$ | $SRMS_i$ |
| 1 | 29.33 | 67.90 | 7.02 | 33.00 | 13.39 | 33.39 | 38.79 | 73.30 |
| 2 | 96.24 | 197.97 | 35.04 | 68.12 | 111.60 | 151.86 | 336.86 | 447.74 |
| 3 | 127.71 | 266.92 | 34.30 | 65.69 | 103.48 | 133.34 | 269.01 | 320.73 |
| 4 | 265.94 | 430.06 | 50.47 | 97.61 | 80.18 | 85.94 | 211.33 | 229.83 |
| | R = 53.93% | | R = 47.97% | | R = 76.30% | | R = 79.88% | |

$R_{avg} = 64.52\%$

Table 3-8b. Signal RMS Reduction With 24 kbps Quantization (LS Lattice)

A frame-to-frame RMS analysis identical to that for prediction without quantization is done for 16 kbps and 24 kbps. Since results at the two bit rates turn out to be very similar, only one set is plotted (Fig. 3-3).



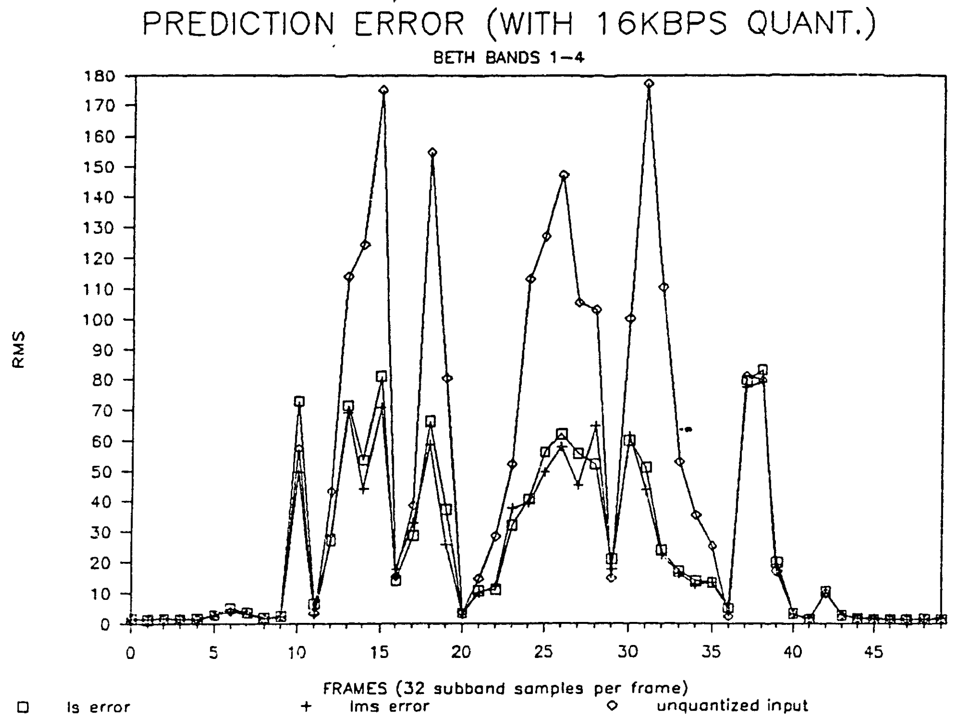
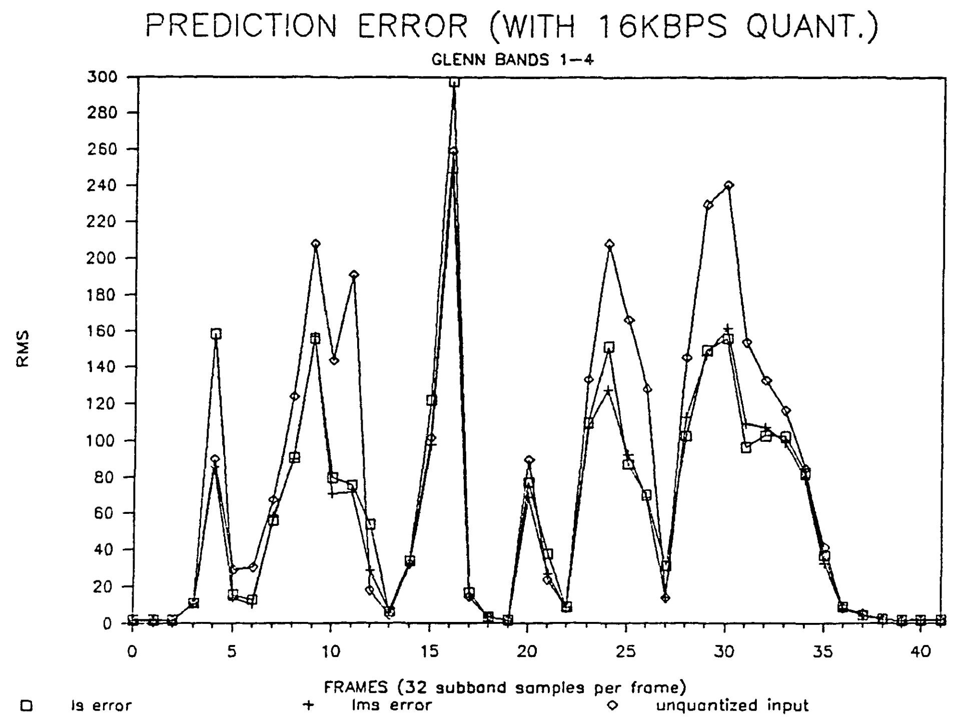


**Figure 3-3. Frame-to-Frame Prediction Error With 16 Kbps Quantization**



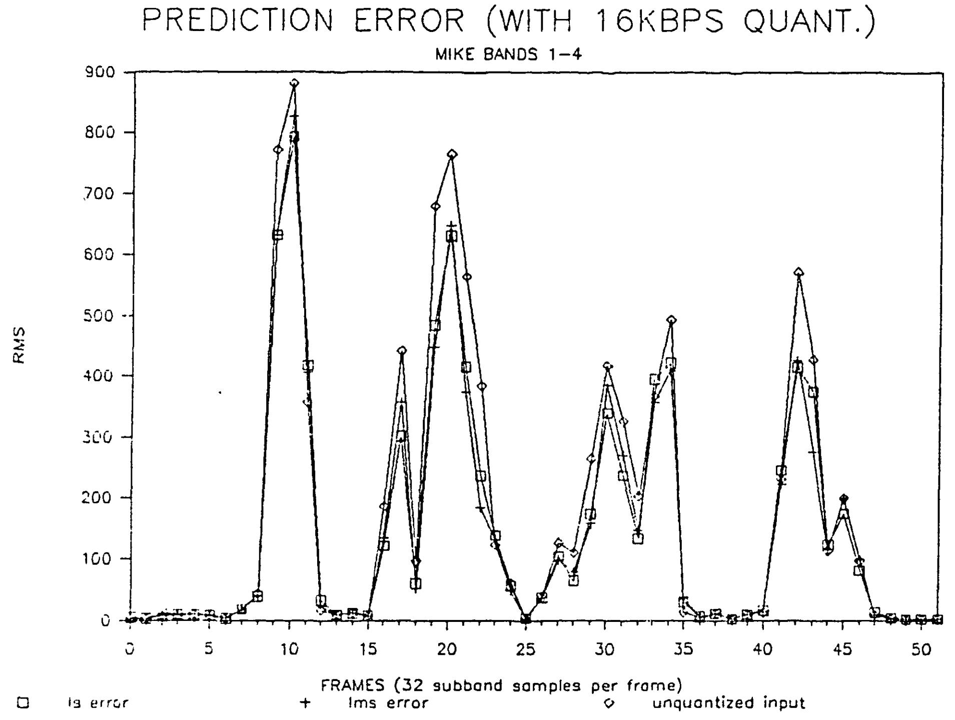


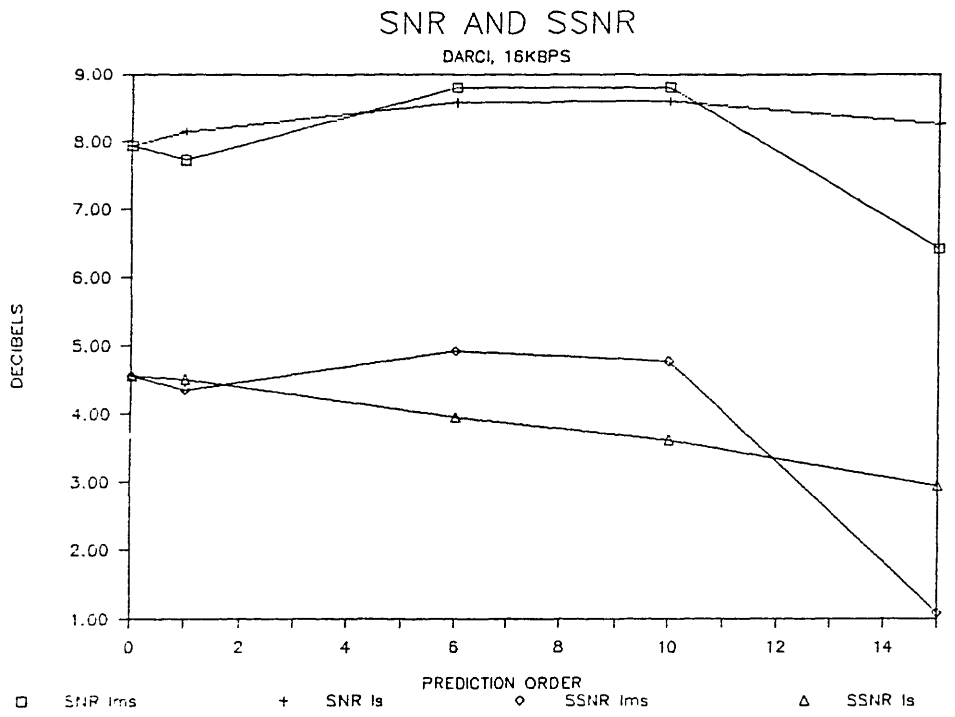
**Figure 3-3. Frame-to-Frame Prediction Error With 16 Kbps Quantization (continued)**

## 3.7 CODER PERFORMANCE

The final group of tests involves judging the voice quality of the processed sentences at 16 kbps and 24 kbps. Three versions of the sub-band coder, one without prediction, one using LMS transversal, and one using LS lattice, are each run on the test set. Each of the predictive coders are simulated at orders 1, 6, 10, and 15 using the optimal parameters determined in Section 3.5.

### 3.7.1 SNR and SSNR Performances

The following four figures are plots of SNR and SSNR vs. prediction order for DARCI, BETH, GLENN, and MIKE at 16 kbps and 24 kbps. For all of the figures, an order of 0 indicates no prediction. The SNR and SSNR values are listed in Appendix D.



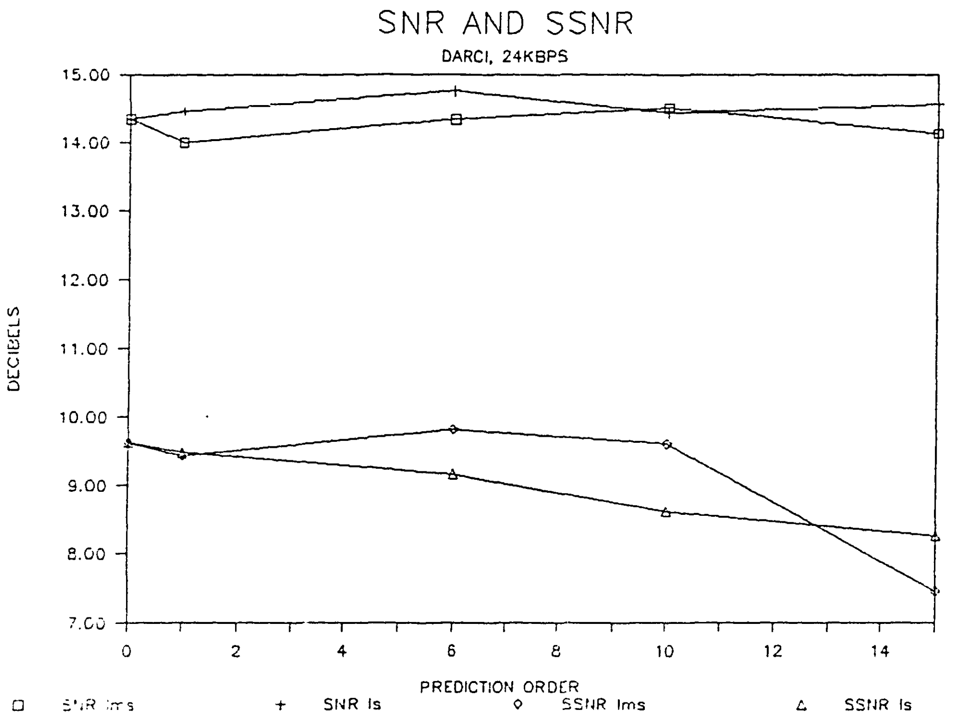
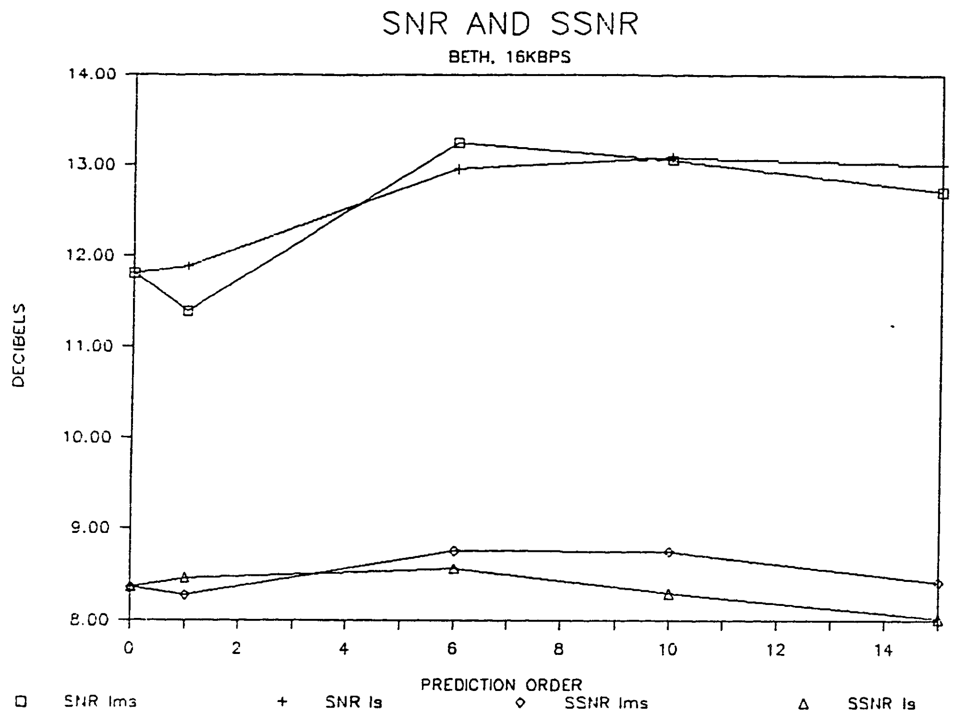


**Figure 3-4. SNR and SSNR (DARCI)**



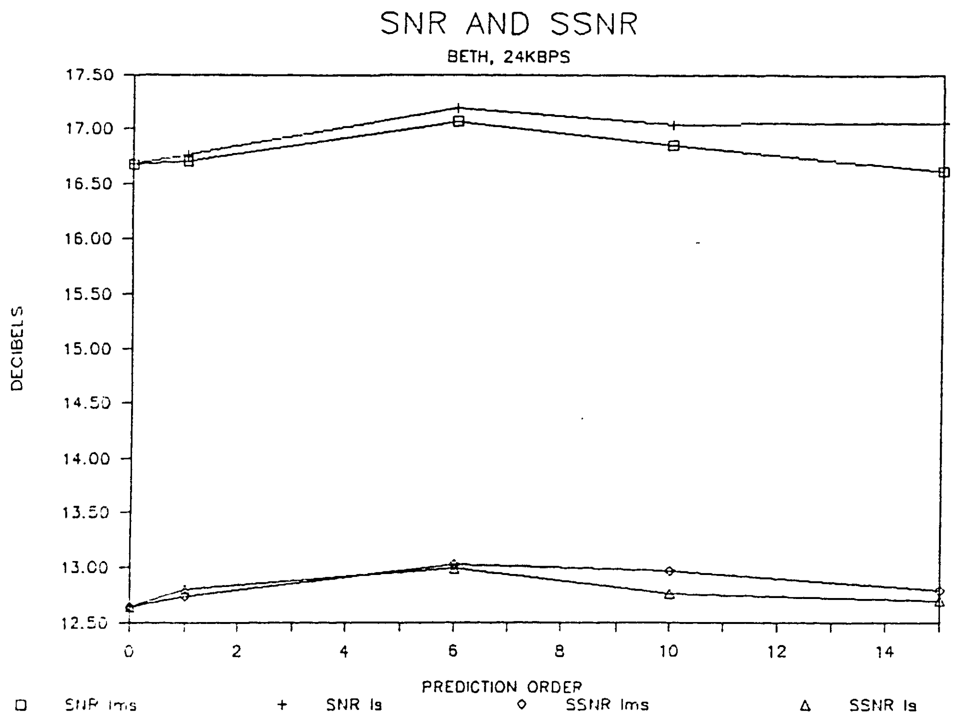
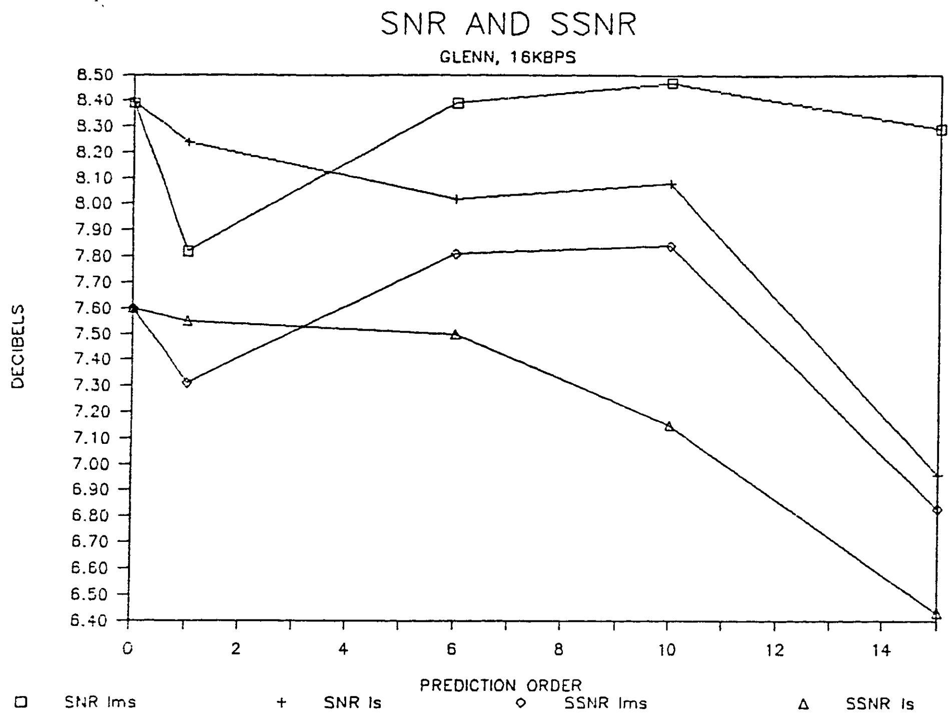


**Figure 3-5. SNR and SSNR (BETH)**



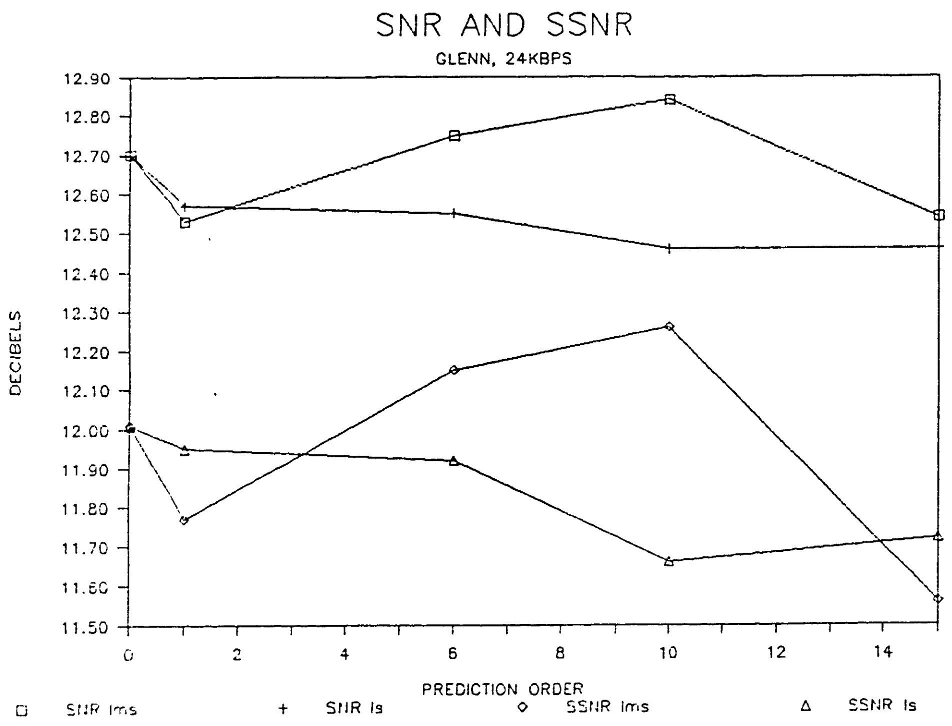
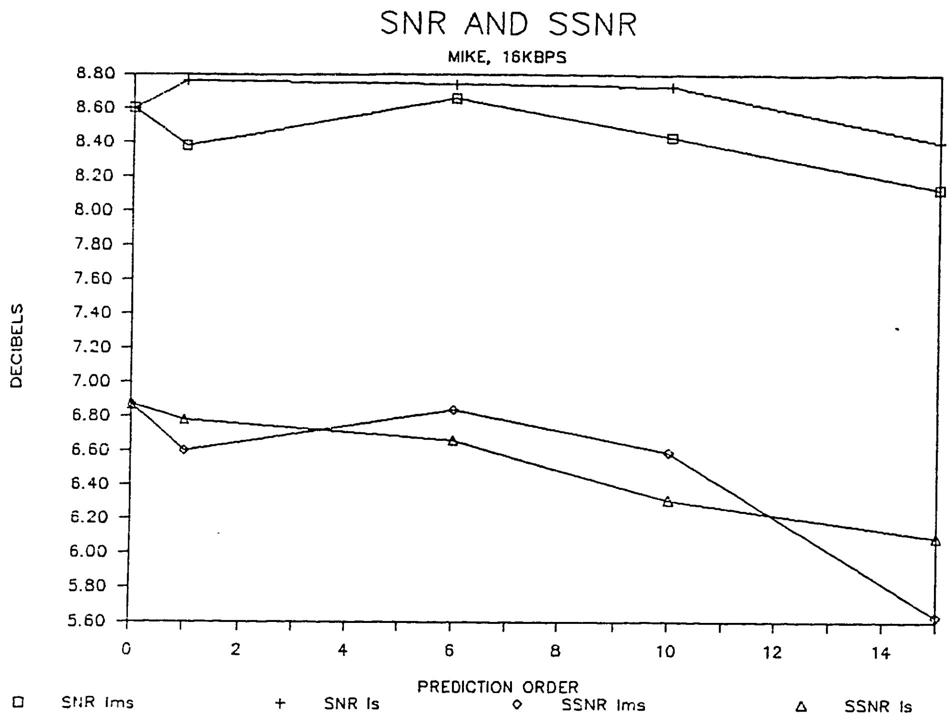


**Figure 3-6. SNR and SSNR (GLENN)**



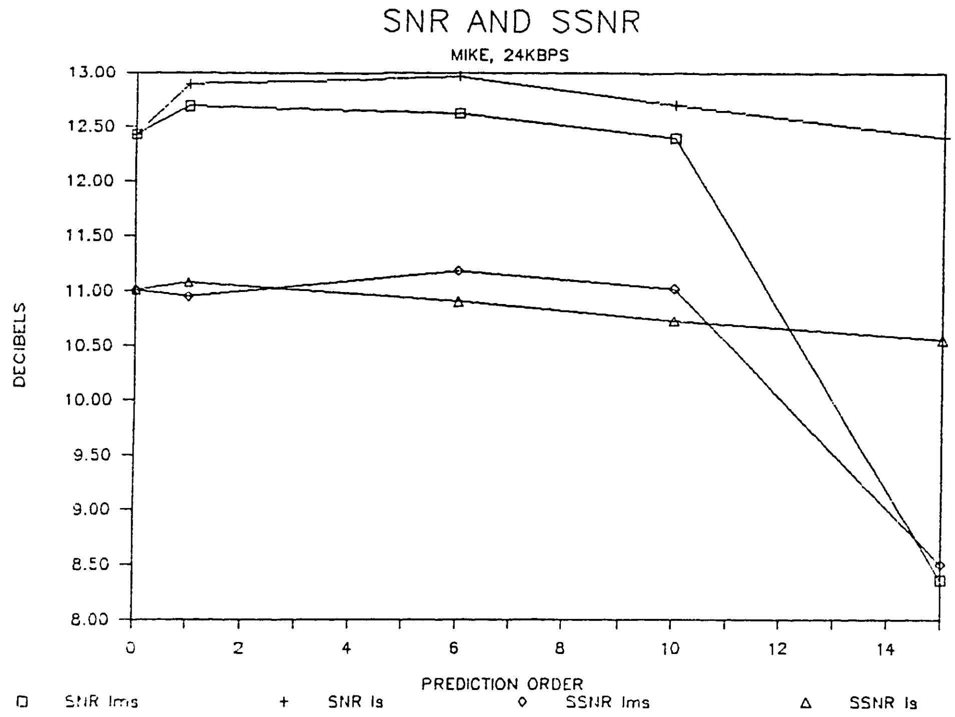
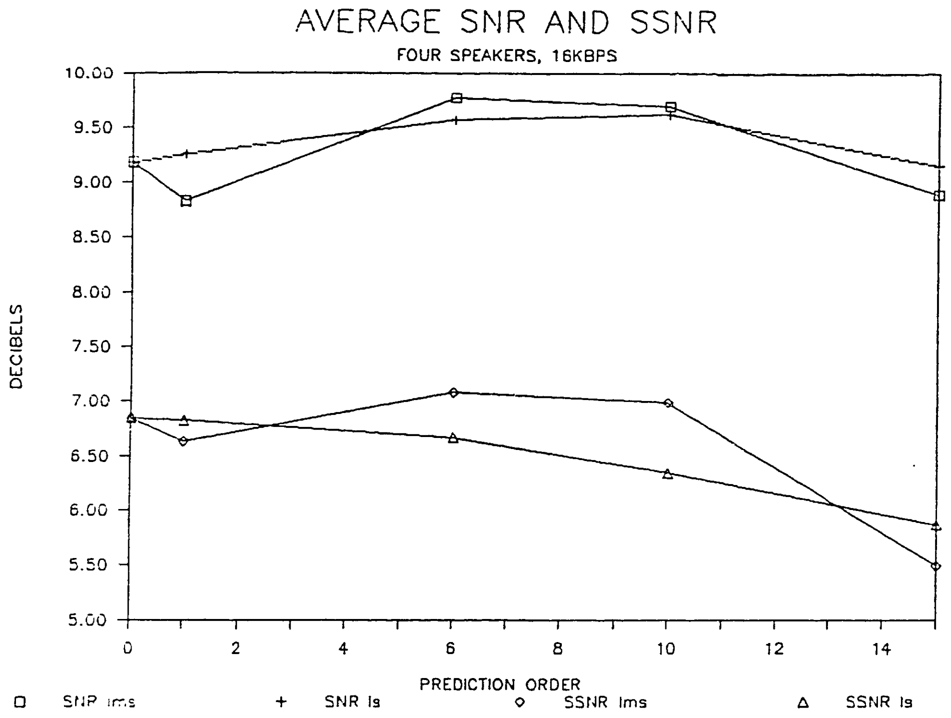


**Figure 3-7. SNR and SSNR (MIKE)**

**The objective measures for all four sentences are averaged and graphed in Fig. 3-8.**



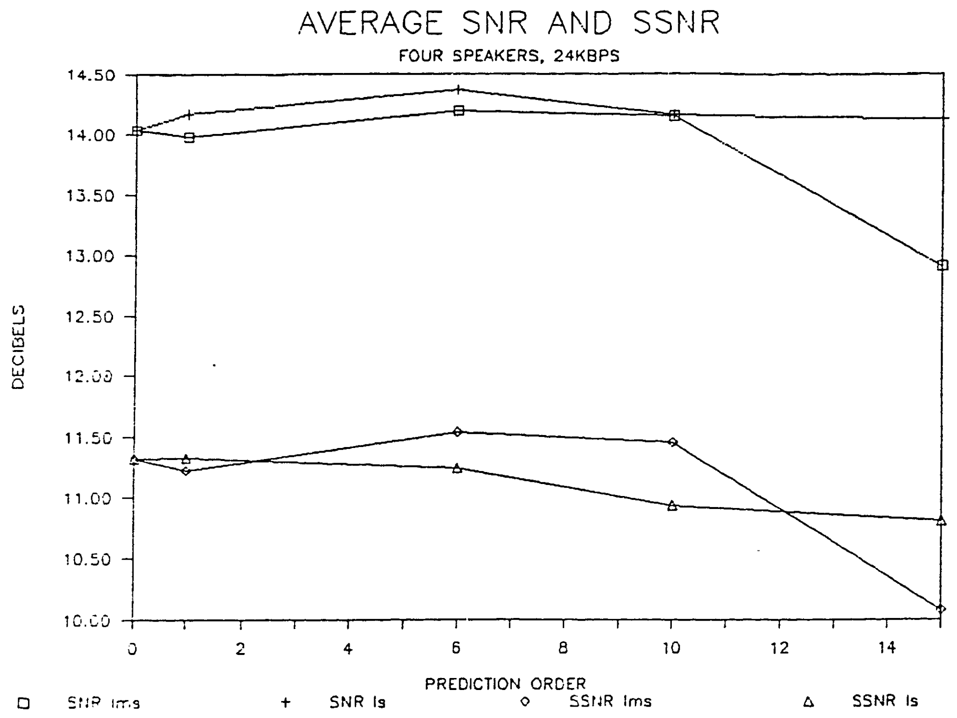


**Figure 3-8. SNR and SSNR (all speakers)**

### 3.7.2 Listening Tests

Subjective preference tests are conducted in which the relative qualities of SBC without prediction, SBC with optimal LMS prediction, and SBC with optimal LS prediction are judged at 16 kbps and 24 kbps.

#### 3.7.2.1 Test Format

In order to include as many different listeners as possible, the test is designed to be brief. Consequently, only two sentences are used; one male (MIKE) and one female (BETH). Six versions of each are recorded : no prediction, optimal LMS transversal, and optimal LS lattice at 16 kbps and 24 kbps. These are denoted by (NAME)(BIT RATE).(PREDICTOR), e.g. BETH16.no refers to BETH processed at 16 kbps with no prediction. Each subject hears the twelve pairs of sentences

| | A | B |
|---|---|---|
| 1 | BETH16.no | BETH16.lms |
| 2 | BETH16.no | BETH16.ls |
| 3 | BETH16.lms | BETH16.ls |
| 4 | BETH24.no | BETH24.lms |
| 5 | BETH24.no | BETH24.ls |
| 6 | BETH24.lms | BETH24.ls |
| 7 | MIKE16.no | MIKE16.lms |
| 8 | MIKE16.no | MIKE16.ls |
| 9 | MIKE16.lms | MIKE16.ls |
| 10 | MIKE24.no | MIKE24.lms |
| 11 | MIKE24.no | MIKE24.ls |
| 12 | MIKE24.lms | MIKE24.ls |

Table 3-9. Listening Test Pairs

and for each pair is asked to select one of five options indicating which one sounds better

| | |
|---|---|
| 1 | Phrase "A" Clearly Better |
| 2 | Phrase "A" Slightly Better |
| 3 | No Preference |
| 4 | Phrase "B" Slightly Better |
| 5 | Phrase "B" Clearly Better |

Table 3-10. Listening Test Response Options

For every listener, the order of the 12 pairs is randomly chosen, as is the order of the sentences within each pair. Also, the subjects are not told which versions are being played. Such a test, known as double-blind, attempts to protect against bias in the comparisons.

### 3.7.2.2 Test Results

A total of 60 people participate in the listening test, producing the following distribution of responses.

| 16 kbps | A | B | A>>B | A>B | A=B | A<B | A<<B |
|---|---|---|---|---|---|---|---|
| | lms | no | 5% | 22% | 58% | 12% | 3% |
| | ls | no | 4% | 28% | 52% | 13% | 2% |
| | ls | lms | 7% | 16% | 59% | 16% | 2% |

| 24 kbps | A | B | A>>B | A>B | A=B | A<B | A<<B |
|---|---|---|---|---|---|---|---|
| | lms | no | 1% | 14% | 70% | 14% | 1% |
| | ls | no | 1% | 17% | 67% | 12% | 2% |
| | ls | lms | 1% | 15% | 64% | 19% | 1% |

Table 3-11. Listening Test Results

A total of 120 data points are gathered for each of the six A-B comparisons (two sentences, 60 listeners).

# CHAPTER 4 - DISCUSSION

The simulation results in this study allow several observations to be made about the SBC-ADPCM system. These concern the degradation introduced by quantization, the behavior of the predictors, and the effects of differential coding on voice quality.

## 4.1 SUB-BAND CODING WITHOUT PREDICTION

SBC without prediction employs only adaptive quantization to code the subband signals. Table 4-1 summarizes the SNR and SSNR values obtained by coding the test sentences with different bit rates.

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| QUANTIZATION | SNR | SSNR | SNR | SSNR | SNR | SSNR | SNR | SSNR |
| none | 30.83 | 26.51 | 30.99 | 25.96 | 30.55 | 26.47 | 29.38 | 27.22 |
| ≥ 35 kbps | 18.52 | 14.39 | 18.91 | 15.69 | 14.02 | 15.21 | 13.45 | 14.10 |
| 24 kbps | 14.34 | 9.63 | 16.68 | 12.64 | 12.70 | 12.01 | 12.43 | 11.01 |
| 16 kbps | 7.94 | 4.55 | 11.81 | 8.36 | 8.39 | 7.60 | 8.60 | 6.87 |

(all values in dB)

Table 4-1. Quantization Degradation (No Prediction)

It is evident that quantization, even at the maximum of 5 bits per band (≥ 35 kbps), introduces a loss of 10 dB or more in both SNR and SSNR. Further reduction to 24 kbps produces a smaller distortion of about 1-5 dB. Finally, another 5 dB is lost in going down to 16 kbps.

The author's personal evaluation of the different bit rates is that voice quality is good (acceptable for voice store-and-forward applications) down to 24 kbps but at 16 kbps, the coded sentences sound noticeably poor.

The SNR results are generally lower than some of the values reported in the literature. For example, at 16 kbps, Barnwell [24] achieves about 15 dB and Gupta and Virupaksha [33], 17 dB. At 32 kbps, Esteban and Galand [47] measure 25 dB. These performances are 3-11 dB better than those in Table 4-1. However, the 16 kbps sub-band coder of Crochiere, Webber, and Flanagan [1] gives an SNR of 11.1 dB, which is comparable to the values found in this study.

The disparity in SNR is caused by the use of different quantizers. In both [24] and [33], the quantizer has a non-uniform step size which is optimized to the probability density function of the input [55]. In [47], Esteban and Galand employ block companded pulse code modulation (BCPCM) (see Section 2.4). In this thesis as well as in [1], the quantizer updates its step size based upon Jayant's simple multiplier rule [8].

To confirm the benefits of using another quantizer, the BCPCM method is simulated on the subbands of the four test sentences. Table 4-2 gives the SNR and SSNR results.

| | DARCI | | BETH | | GLENN | | MIKE | |
|---|---|---|---|---|---|---|---|---|
| QUANTIZATION | SNR | SSNR | SNR | SSNR | SNR | SSNR | SNR | SSNR |
| ≥ 35 kbps | 24.37 | 19.22 | 23.83 | 18.98 | 20.68 | 19.41 | 20.95 | 18.55 |
| 24 kbps | 21.57 | 16.76 | 22.61 | 17.78 | 19.18 | 17.49 | 19.39 | 16.82 |
| 16 kbps | 14.40 | 10.70 | 17.81 | 13.24 | 12.94 | 11.80 | 13.02 | 11.27 |

(all values in dB)

Table 4-2. BCPCM Quantization Degradation (No Prediction)

Thus, the SNR and SSNR values of BCPCM are better than those of Jayant's quantizer by about 5 dB.

## 4.2 OPTIMAL PREDICTORS

The predictor parameters are optimized to the four test sentences by minimizing a figure of merit, $R_{avg}$. Tables 3-3 and 3-5 demonstrate, however, that contributions to $R_{avg}$ from the different sentences have a large variance. This indicates that conclusions reached from this data might not be valid for other speech samples. With this qualification in mind, the following observations can be made.

### 4.2.1 Optimal Order

The optimal order for both the LMS transversal and LS lattice predictors is 6. Higher orders are not as effective in improving the figure of merit, $R_{avg}$. This suggests that the subband signals may not have significant correlation at delays beyond 6.

### 4.2.2 LMS vs. LS

Comparisons of the two optimal predictors indicate very similar performances. In terms of $R_{avg}$, the LMS transversal does only slightly better (3.38%). Tables 3-3 and 3-5 show that BETH and GLENN contribute the most to this discrepancy. The frame-to-frame RMS plots for these sentences (Fig. 3-2) reveal that the least-squares error occasionally overshoots the input during sudden increases in input RMS. Otherwise, the LS and LMS algorithms follow similar tracking patterns.

### 4.2.3 Cases of Best Prediction

Both predictors do significantly better with the female speakers than with the males - $R_{avg}$ for DARCI and BETH is 15-30% less than that for GLENN and MIKE. A probable reason for this discrepancy is that female voices have higher fundamental frequencies, which increases the chance of having only one harmonic in a subband [56].

Another interesting observation is that for all four sentences, the percentage RMS reduction, defined in equation (3-1), is greatest in the first band. This agrees with the fact that there is more correlation in lowpass signals than in higher bandpass ones [54, p.178].

### 4.2.4 Prediction With Quantization

The predictors are optimized using exact subband inputs. When quantization is introduced, the adverse effect on predictor performance turns out to be negligible. In fact, in some instances there is even a small improvement (Table 4-3). Frame-to-frame plots also show that quantization effects are insignificant (Figs. 3-2 and 3-3).

| | | DARCI | BETH | GLENN | MIKE | |
|---|---|---|---|---|---|---|
| | | R(%) | R(%) | R(%) | R(%) | $R_{avg}$(%) |
| LMS | no quantization | 53.95 | 44.78 | 70.47 | 80.00 | 62.30 |
| | 24 kbps | 57.20 | 46.12 | 70.76 | 81.01 | 63.77 |
| | 16 kbps | 56.15 | 45.90 | 71.74 | 82.44 | 64.06 |
| LS | no quantization | 55.63 | 50.43 | 74.93 | 81.72 | 65.68 |
| | 24 kbps | 53.93 | 47.97 | 76.30 | 79.88 | 64.52 |
| | 16 kbps | 59.82 | 49.44 | 79.13 | 82.70 | 67.77 |

Table 4-3. Quantization Effects on Prediction

## 4.3 SUB-BAND CODING WITH PREDICTION

### 4.3.1 Trends With Order

Fig. 3-8, which combines the performances of all four speakers, shows two clear trends.

One is that additional orders of prediction beyond 10 degrade coder performance. This may be caused by the introduction of uncorrelated samples to the prediction calculation. Another possible contribution is the fact that the LMS and LS parameters are optimal for order 6 and not necessarily for others.

The other notable trend is that the LMS curve drops more sharply than that of the LS. This is because the gradient estimate used in the LMS update equation becomes noisier as the prediction order increases. [17].

### 4.3.2 Objective Prediction Gain

The changes in SNR and SSNR resulting from the addition of 6th order optimal predictors to a non-differential sub-band coder are shown below.

| | DARCI | | BETH | | GLENN | | MIKE | | AVERAGE | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ΔSNR | ΔSSNR | ΔSNR | ΔSSNR | ΔSNR | ΔSSNR | ΔSNR | ΔSSNR | ΔSNR | ΔSSNR |
| 16 kbps | | | | | | | | | | |
| LMS | +.86 | +.37 | +1.44 | +.39 | 0 | +.21 | +.06 | -.03 | +.59 | +.24 |
| LS | +.64 | -.60 | +1.15 | +.20 | -.37 | -.10 | +.14 | -.21 | +.39 | -.18 |
| 24 kbps | | | | | | | | | | |
| LMS | 0 | +.19 | +.39 | +.38 | +.05 | +.14 | +.20 | +.18 | +.16 | +.22 |
| LS | +.42 | -.46 | +.51 | +.35 | -.15 | -.09 | +.54 | -.10 | +.33 | -.08 |

(all values in dB)

Table 4-4. SNR and SSNR Prediction Gains

An unexpected result is that some of the gains are negative. Indeed, Figs. 3-4 to 3-7 demonstrate that the optimal predictors do not always give higher SNR and SSNR values than those of non-optimal orders. A possible explanation for this is that additional distortion is introduced by the adaptive quantizer. In other words, a (greater) reduction in the RMS of the quantizer input does not in itself guarantee better coding. Perhaps the quantizer step size does not adapt well to some types of differential signals. As of yet, this idea has not been experimentally confirmed.

All of the improvements in Table 4-4 are rather small and typically less than 1 dB. The following argument demonstrates that this is a direct consequence of using prediction in only the first four bands. By building a simple model of the SBC-ADPCM system, an expression for prediction gain in terms of subband RMS reduction can be derived.

Suppose that the power of the SBC-ADPCM input, $<s_{in}^2>$, is equal to the sum of the powers of its 13 subband signals, all of which have the same power, $<s^2>$ (this isn't usually true, of course, but it simplifies these calculations). Similarly, assume that the total quantization noise power, $<n_q^2>$, is just 13 times the noise in one band, $<n^2>$. Then

(4-1a) $<s_{in}^2> = 13 <s^2>$ and

(4-1b) $<n_q^2> = 13 <n^2>$

If the predictor in each of the first four bands manages to reduce the subband signal RMS to r times its value, and this causes the quantizing noise RMS to be attenuated by the same factor, then

(4-2) $n_{RMS,p} = (r)(n_{RMS})$

where $n_{RMS,p}$ and $n_{RMS}$ are the noise RMS's in each predicted and non-predicted subband, respectively. Thus, the signal-to-noise ratio of the coder is given by

(4-3) $$SNR = \frac{<s_{in}^2>}{9n_{RMS}^2 + 4n_{RMS,p}^2} = \frac{<s_{in}^2>}{9<n^2> + 4r^2<n^2>} = \frac{13}{9 + 4r^2}\,\frac{<s_{in}^2>}{13<n^2>}$$

Therefore, the prediction gain in dB is just

(4-4) $$\text{PREDICTION GAIN} = 10\log_{10}\frac{13}{9 + 4r^2}$$

By plugging in r = 65%, which is about the average RMS reduction achieved by the optimal predictors in this study, (4-4) yields an expected gain of .85 dB. This is comparable to the gains actually obtained.

Extending this argument to the case of predicting in all 13 bands changes the gain expression to

(4-5) $$\text{PREDICTION GAIN} = 10\log_{10}\frac{1}{r^2}$$

For the same value of r, the gain is now 3.74 dB.

As mentioned in Chapter 2, however, no prediction is used in the higher bands because there is not enough correlation at those frequencies.

In another study [27], Galand, et.al. cite prediction gains of 2-12 dB using an LMS transversal predictor in sub-band coding. Their coder, however, operated on signals bandlimited to 1000 Hz, allowing prediction in all subbands. Furthermore, their QMF filters were 125 Hz wide, half the width of those in this simulation, thereby providing more correlation in each band. In addition, they coded only sustained voiced sounds, which are easier to predict than phonetically balanced sentences.

### 4.3.3 Subjective Prediction Gain

The results of the listening test (Table 3-11) support the conclusion that the prediction gain is minimal.

At 16 kbps, the majority of the responses indicate no preference among no prediction, LMS prediction, and LS prediction. However, a noticeable minority perceive a slight improvement with the LMS and LS algorithms over no prediction.

At 24 kbps, the overwhelming judgment is that all three versions are indistinguishable.

# CHAPTER 5 - CONCLUSIONS

The primary goal of this research was to compare the performances of the least-mean-square transversal and least-squares lattice predictors in the context of sub-band coding. Objective improvements over non-differential SBC and trends with respect to prediction order were to be measured. In addition, the feasibility of using predictive techniques to lower the bit rate from 24 kbps to 16 kbps while preserving voice quality was to be determined.

The LS lattice predictor generally tracks sudden changes in input better than the LMS transversal [14]. This property is particularly important in real-time speech coding since voice waveforms can change very quickly. Despite the theoretical advantage of the least-squares algorithm, the results of this study show that the two predictors, when optimized to the test sentences, have very similar behavior in tracking the subband signals.

For both LMS and LS algorithms the optimal prediction order is 6, which achieves maximum SNR and SSNR gains of 1.44 dB and .39 dB, respectively. Higher orders tend to perform slightly worse, especially with LMS. However, both objective and subjective criteria clearly indicate that any improvements over no prediction are marginal, at best, and do not permit a savings of 8 kbps.

A simple model of an SBC-ADPCM system (Section 4.3.2) demonstrates that the low values of prediction gain may be attributed to the application of predictors in only the first four subbands. This design decision is based upon the findings of Ramstad [31] and Hamel et.al. [32] that higher bands do not have as much correlation. Although this thesis does not confirm these claims, they are supported by studies which show that the prediction gain of coders with prediction in all bands is still only about 2 dB [24,28]. Another reason why prediction is not as successful at higher bands is that fewer bits are usually available, which increases the quantization noise of the predictor input.

The overall performance of the simulated coder is lower than previously published values by about 3-11 dB [24,33,47]. This is a consequence of selecting a simple but inferior quantizer. Substitution of a different quantizer achieves an SNR improvement of approximately 5 dB (Section 4.1). This result, in conjunction with the small prediction gains, suggests that the design of the quantizer is more important to overall coder quality than is the use of differential encoding.

As a final note, the findings of this study must be qualified by the size of evaluation data base. Although the four test sentences are phonetically balanced and include male and female voices of different volumes, they are not necessarily representative of a larger sample of speech. In order to make the conclusions of this thesis statistically more significant, many more sentences should be processed and analyzed.

## REFERENCES

[1] R.E. Crochiere, S.A. Webber, J.L. Flanagan, "Digital Coding of Speech in Sub-Bands," *Bell System Technical Journal*, October 1976, pp. 1069-1085.

[2] D. Esteban, C. Galand, "Application of Quadrature Mirror Filters to Split-Band Coding," *Proc. IEEE ICASSP*, 1977.

[3] B.S. Atal, M.R. Schroeder, "Adaptive Predictive Coding of Speech Signals," *Bell System Technical Journal*, October 1970, pp. 1973-1986.

[4] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

[5] L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

[6] N.S. Jayant, Peter Noll, *Digital Coding of Waveforms*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1984.

[7] B.M. Oliver, J.R. Pierce, and C.E. Shannon, "The Philosophy of PCM," *Proc. IRE*, Vol. 36, November 1948, pp. 1324-1331.

[8] N.S. Jayant, "Adaptive Quantization with a One Word Memory," *Bell System Technical Journal*, September 1973, pp. 1119-1144.

[9] P. Cummiskey, N.S. Jayant, and J.L. Flanagan, "Adaptive Quantization in Differential PCM Coding of Speech," *Bell System Technical Journal*, Vol. 52, No. 7, September 1973, pp. 1105-1118.

[10] P. Elias, "Predictive Coding," *IRE Trans. on Information Theory*, Vol. IT-1, No. 1, March 1955, pp. 16-33.

[11] R.A. McDonald, "Signal-to-Noise Performance and Idle Channel Performance of Differential Pulse Code Modulation Systems with Particular Applications to Voice Signals," *Bell System Technical Journal*, Vol. 45, No. 7, September 1966, pp. 1123-1151.

[12] D. Cohn, J. Melsa, "The Residual Encoder - An Improved ADPCM System for Speech Digitization," *IEEE Trans. on Comm.*, Vol. COM-23, No. 9, September 1974, pp. 935-941.

[13] M. Morf, A. Vieira, and D.T. Lee, "Ladder Forms for Identification and Speech Processing," *Proc. IEEE Conference on Decision and Control*, 1977, pp. 1074-1078.

[14] B. Friedlander, "Lattice Filters for Adaptive Processing," *Proc. IEEE*, Vol. 70, No. 8, August 1982, pp. 829-867.

[15] J.D. Gibson, "Adaptive Prediction in Speech Differential Encoding Systems," *IEEE Proc. on Communications*, Vol. COM-23, September 1975, pp. 935-941.

[16] B. Widrow and M. Hoff, Jr., "Adaptive Switching Circuits," *IRE WESCON Conv. Rec.*, pt. 4, 1960, pp. 96-104.

[17] B. Widrow, J.M. McCool, M.G. Larimore, and C.R. Johnson, Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proc. IEEE*, Vol. 64, No. 8, August 1976, pp. 1151-1162.

[18] M. Morf, D.T. Lee, J.R. Nickolls, and A. Vieira, "A Classification of Algorithms for ARMA Models and Ladder Realizations," *Proc. IEEE ICASSP*, 1977, pp. 13-19.

[19] J.D. Gibson and L.C. Sauter, "Experimental Comparison of Forward and Backward Adaptive Prediction in DPCM," *Proc. IEEE ICASSP*, 1980, pp. 508-511.

[20] G. Pirani and V. Zingarelli, "An Analytical Formula for the Design of Quadrature Mirror Filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 3, June 1984, pp. 645-648.

[21] V.K. Jain and R.E. Crochiere, "Quadrature Mirror Filter Design in the Time Domain," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 2, April 1984, pp. 353-361.

[22] P.C. Millar, "Recursive Quadrature Mirror Filters - Criteria Specification and Design Method," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-33, No. 2, April 1985, pp. 413-420.

[23] R.V. Cox, "The Design of Uniformly and Nonuniformly Spaced Pseudoquadrature Mirror Filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-34, No. 5, October 1986, pp. 1090-1096.

[24] Thomas P. Barnwell, III, "Subband Coder Design Incorporating Recursive Quadrature Filters and Optimum ADPCM Coders," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-30, No. 5, October 1982, pp. 751-765.

[25] R.E. Crochiere, "A Novel Approach for Implementing Pitch Prediction in Sub-Band Coding," *Proc. IEEE ICASSP*, 1979, pp. 526-529.

[26] A.J. Barabell, R.E. Crochiere, "Sub-Band Coder Design Incorporating Quadrature Filters and Pitch Prediction," *Proc. IEEE ICASSP*, 1979, pp. 530-533.

[27] C. Galand, K. Daulasim, D. Esteban, "Adaptive Predictive Coding of Base-Band Speech Signals," *Proc. IEEE ICASSP*, 1982, pp. 220-223.

[28] F.K. Soong, R.V. Cox, N.S. Jayant, "Subband Coding of Speech Using Backward Adaptive Prediction and Bit Allocation," *Proc. IEEE ICASSP*, 1985, pp. 1672-1675.

[29] C. Galand and D. Esteban, "16kbps Sub-Band Coder Incorporating Variable Overhead Information," *Proc. IEEE ICASSP*, 1982, pp. 1684-1687.

[30] R.S. Cheung and R.L. Winslow, "High Quality 16 kb/s Voice Transmission: the Subband Coder Approach," *Proc. IEEE ICASSP*, 1980, pp. 319-322.

[31] T.A. Ramstad, "Considerations on Quantization and Dynamic Bit-Allocation in Subband Coders," *Proc. IEEE ICASSP*, 1986, pp. 841-844.

[32] P. Hamel, J. Soumagne, and A. Le Guyader, "A New Dynamic Bit Allocation Scheme for Sub-Band Coding," *Proc. IEEE ICASSP*, 1985, pp. 1676-1679.

[33] V. Gupta and K. Virupaksha, "Performance Evaluation of Adaptive Quantizers for a 16-kbit/s Sub-Band Coder," *Proc. IEEE ICASSP*, 1982, pp. 1688-1691.

[34] CCITT Report, Study Group XVIII, Temp. Doc. 18, Draft Recommendation G.7zz, Geneva, November 21-25, 1983.

[35] D.T.L. Lee, M. Morf, B. Friedlander, "Recursive Least-Squares Ladder Estimation Algorithms," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, No. 3, June 1981, pp. 627-641.

[36] M. Morf and D.T. Lee, "Recursive Least Squares Ladder Forms for Fast Parameter Tracking," *Proc. IEEE Conference on Decision and Control*, 1978, pp. 1362-1367

[37] R.S. Medaugh and L.J. Griffiths, "A Comparison of Two Fast Linear Predictors," *Proc. IEEE ICASSP*, 1981, pp. 293-296.

[38] D.D. Falconer and L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization," *IEEE Trans. on Communications*, Vol. COM-26, October 1978, pp. 1439-1446.

[39] E.H. Satorius and M.J. Shensa, "Recursive Lattice Filters - A Brief Overview," *Proc. IEEE Conference on Decision and Control*, 1980, pp. 955-959.

[40] E.H. Satorius and J.D. Pack, "Application of Least Squares Lattice Algorithms to Adaptive Equalization," *IEEE Trans. on Communications*, Vol. COM-29, No. 2, February 1981, pp. 136-142.

[41] M.L. Honig and D.G. Messerschmitt, "Comparison of Adaptive Linear Prediction Algorithms in ADPCM," *IEEE Trans. on Communications*, Vol. COM-30, No. 7, July 1982, pp. 1775-1785.

[42] R.C. Reininger and J.D. Gibson, "Backward Adaptive Lattice and Transversal Predictors for ADPCM," *Proc. IEEE ICASSP*, 1984.

[43] F.L. Kitson and K.A. Zeger, "A Real-Time ADPCM Encoder Using Variable Order Prediction," *Proc. IEEE ICASSP*, 1986, pp. 825-828.

[44] B. Widrow, et. al., "Adaptive Noise Cancelling : Principles and Applications," *Proc. IEEE*, Vol. 63, No. 12, December 1975, pp. 1692-1719.

[45] R.E. Crochiere and L.R. Rabiner, "Interpolation and Decimation of Digital Signals - A Tutorial Review," *Proc. IEEE*, Vol. 69, No. 3, March 1981, pp. 300-331.

[46] Claude R. Galand and Henri J. Nussbaumer, "New Quadrature Mirror Filter Structures," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, No. 3, June 1984, pp. 522-531.

[47] D. Esteban, C. Galand, "32 kbps CCITT Compatible Split Band Coding Scheme," *Proc. IEEE ICASSP*, 1978, pp. 320-325.

[48] D. Esteban and C. Galand, "Multiport Implementation of Real Time 16kbps Sub-Band Coder," *Proc. IEEE ICASSP*, 1981.

[49] C. Galand and D. Esteban, "16kbps Real Time QMF Sub-Band Coding Implementation," *Proc. IEEE ICASSP*, 1980, pp. 332-335.

[50] C. Galand and D. Esteban, "Multirate Sub-Band Coder with Embedded Bit Stream: Application to Digital TASI," *Proc. IEEE ICASSP*, 1983, pp.1284-1287.

[51] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

[52] Ben Wilbanks, private communication, April 14, 1987.

[53] "IEEE Recommended Practice for Speech Quality Measurements," *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-17, No. 3, September 1969, pp. 225-246.

[54] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1978.

[55] J. Max, "Quantizing for Minimum Distortion," *IRE Trans. on Information Theory*, Vol. IT-6, March 1960, pp. 7-12.

[56] Dennis Klatt, private communication, August 3, 1987.

# APPENDIX A - QMF COEFFICIENTS

This appendix gives the impulse responses of the 16 parallel QMF bandpass filters discussed in Chapter 2. In addition, it lists the coefficients of the half-band filters from the corresponding tree structure.

## A.1 HALF-BAND FILTERS

The parallel QMF bank is generated from four half-band lowpass filters, h1(n), h2(n), h3(n), and h4(n), from stages 1 to 4, respectively, of a tree configuration. Each is a symmetric FIR filter with an even number of taps. Because of decimation effects (see Section 2.2), the number of coefficients may be reduced by about half at each stage.

Stage 1 - 64 taps

| | | | | | |
|---|---|---|---|---|---|
| h1(0) | = | .0008710111 | h1(32) | = | .4575616 |
| h1(1) | = | -.002555511 | h1(33) | = | .1417411 |
| h1(2) | = | -.0004363039 | h1(34) | = | -.09642129 |
| h1(3) | = | .002222395 | h1(35) | = | -.05521201 |
| h1(4) | = | .0002690002 | h1(36) | = | .05530956 |
| h1(5) | = | -.003230121 | h1(37) | = | .03117935 |
| h1(6) | = | -.0001858207 | h1(38) | = | -.03876234 |
| h1(7) | = | .004394296 | h1(39) | = | -.01976451 |
| h1(8) | = | -.00007973524 | h1(40) | = | .02942451 |
| h1(9) | = | -.005825718 | h1(41) | = | .01309416 |
| h1(10) | = | .0005297959 | h1(42) | = | -.02318487 |
| h1(11) | = | .007542193 | h1(43) | = | -.008769508 |
| h1(12) | = | -.001228547 | h1(44) | = | .01858054 |
| h1(13) | = | -.009589027 | h1(45) | = | .005808109 |
| h1(14) | = | .002256642 | h1(46) | = | -.01496923 |
| h1(15) | = | .01203045 | h1(47) | = | -.003727316 |
| h1(16) | = | -.003727316 | h1(48) | = | .01203045 |
| h1(17) | = | -.01496923 | h1(49) | = | .002256642 |
| h1(18) | = | .005808109 | h1(50) | = | -.009589027 |
| h1(19) | = | .01858054 | h1(51) | = | -.001228547 |
| h1(20) | = | -.008769508 | h1(52) | = | .007542193 |
| h1(21) | = | -.02318487 | h1(53) | = | .0005297959 |

| | | | |
|---|---|---|---|
| h1(22) = | .01309416 | h1(54) = | -.005825718 |
| h1(23) = | .02942451 | h1(55) = | -.00007973524 |
| h1(24) = | -.01976451 | h1(56) = | .004394296 |
| h1(25) = | -.03876234 | h1(57) = | -.0001858207 |
| h1(26) = | .03117935 | h1(58) = | -.003230121 |
| h1(27) = | .05530956 | h1(59) = | .0002690002 |
| h1(28) = | -.05521201 | h1(60) = | .002222395 |
| h1(29) = | -.09642129 | h1(61) = | -.0004363039 |
| h1(30) = | .1417411 | h1(62) = | -.002555511 |
| h1(31) = | .4575616 | h1(63) = | .0008710111 |

## Stage 2 - 32 taps

| | | | |
|---|---|---|---|
| h2(0) = | .001125614 | h2(16) = | .4644493 |
| h2(1) = | -.00459779 | h2(17) = | .1321532 |
| h2(2) = | -.0007166802 | h2(18) = | -.0999044 |
| h2(3) = | .007749691 | h2(19) = | -.0438903 |
| h2(4) = | -.0002332937 | h2(20) = | .05505545 |
| h2(5) = | -.01375898 | h2(21) = | .01920389 |
| h2(6) = | .002733096 | h2(22) = | -.03485896 |
| h2(7) = | .02235853 | h2(23) = | -.008170245 |
| h2(8) = | -.008170245 | h2(24) = | .02235853 |
| h2(9) = | -.03485896 | h2(25) = | .002733096 |
| h2(10) = | .01920389 | h2(26) = | -.01375898 |
| h2(11) = | .05505545 | h2(27) = | -.0002332937 |
| h2(12) = | -.0438903 | h2(28) = | .007749691 |
| h2(13) = | -.0999044 | h2(29) = | -.0007166802 |
| h2(14) = | .1321532 | h2(30) = | -.00459779 |
| h2(15) = | .4644493 | h2(31) = | .001125614 |

## Stage 3 - 16 taps

| | | | |
|---|---|---|---|
| h3(0) = | .002005465 | h3(8) = | .4766322 |
| h3(1) = | -.01261604 | h3(9) = | .1096776 |
| h3(2) = | .0006893994 | h3(10) = | -.096956 |
| h3(3) = | .03872023 | h3(11) = | -.0196975 |
| h3(4) = | -.0196975 | h3(12) = | .03872023 |
| h3(5) = | -.096956 | h3(13) = | .0006893994 |
| h3(6) = | .1096776 | h3(14) = | -.01261604 |
| h3(7) = | .4766322 | h3(15) = | .002005465 |

## Stage 4 - 10 taps

| | | | |
|---|---|---|---|
| h4(0) = | .01219983 | h4(5) = | .4890657 |
| h4(1) = | -.001423909 | h4(6) = | .07888518 |
| h4(2) = | -.07799588 | h4(7) = | -.07799588 |
| h4(3) = | .07888518 | h4(8) = | -.001423909 |
| h4(4) = | .4890657 | h4(9) = | .01219983 |

## A.2 PARALLEL BANDPASS FILTERS

These 16 filters split signals into their subband components. Each is a truncated version of a 258-tap filter obtained by convolving (frequency normalized) half-band filters.

### f0(n) - 0 to 250 Hz

| | | | |
|---|---|---|---|
| f0(0) = | 1.053927E-03 | f0(36) = | .0710003 |
| f0(1) = | 1.841017E-03 | f0(37) = | 6.973641E-02 |
| f0(2) = | 2.063785E-03 | f0(38) = | 6.720698E-02 |
| f0(3) = | 2.345845E-03 | f0(39) = | 6.351141E-02 |
| f0(4) = | 1.506928E-03 | f0(40) = | 5.874324E-02 |
| f0(5) = | 1.367166E-03 | f0(41) = | 5.310814E-02 |
| f0(6) = | 7.006013E-04 | f0(42) = | 4.672493E-02 |
| f0(7) = | 2.67798E-04 | f0(43) = | 3.989666E-02 |
| f0(8) = | -1.244974E-03 | f0(44) = | 3.276445E-02 |
| f0(9) = | -2.206876E-03 | f0(45) = | 2.570829E-02 |
| f0(10) = | -3.523216E-03 | f0(46) = | 1.885466E-02 |
| f0(11) = | -4.592768E-03 | f0(47) = | 1.258937E-02 |
| f0(12) = | -6.402049E-03 | f0(48) = | 6.863879E-03 |
| f0(13) = | -7.726114E-03 | f0(49) = | 1.987042E-03 |
| f0(14) = | -8.812724E-03 | f0(50) = | -2.034144E-03 |
| f0(15) = | -9.322696E-03 | f0(51) = | -5.012196E-03 |
| f0(16) = | -9.782126E-03 | f0(52) = | -7.254503E-03 |
| f0(17) = | -9.669026E-03 | f0(53) = | -8.83648E-03 |
| f0(18) = | -8.836473E-03 | f0(54) = | -9.669022E-03 |
| f0(19) = | -7.254504E-03 | f0(55) = | -9.782125E-03 |
| f0(20) = | -5.01219E-03 | f0(56) = | -9.322696E-03 |
| f0(21) = | -2.034141E-03 | f0(57) = | -8.812728E-03 |
| f0(22) = | 1.987044E-03 | f0(58) = | -7.726108E-03 |
| f0(23) = | 6.86388E-03 | f0(59) = | -6.402048E-03 |
| f0(24) = | 1.258937E-02 | f0(60) = | -4.592766E-03 |
| f0(25) = | 1.885467E-02 | f0(61) = | -3.523215E-03 |
| f0(26) = | 2.570829E-02 | f0(62) = | -2.206874E-03 |
| f0(27) = | 3.276445E-02 | f0(63) = | -1.244974E-03 |
| f0(28) = | 3.989666E-02 | f0(64) = | 2.677977E-04 |
| f0(29) = | 4.672494E-02 | f0(65) = | 7.006018E-04 |
| f0(30) = | 5.310812E-02 | f0(66) = | 1.367166E-03 |
| f0(31) = | 5.874324E-02 | f0(67) = | 1.506929E-03 |
| f0(32) = | 6.351142E-02 | f0(68) = | 2.345844E-03 |
| f0(33) = | .067207 | f0(69) = | 2.063786E-03 |
| f0(34) = | 6.973638E-02 | f0(70) = | 1.841016E-03 |
| f0(35) = | .0710003 | f0(71) = | 1.053927E-03 |

### f1(n) - 250 to 500 Hz

| | | | |
|---|---|---|---|
| f1(0) = | 1.378243E-03 | f1(36) = | -1.226034E-02 |
| f1(1) = | 1.812966E-03 | f1(37) = | -3.541158E-02 |
| f1(2) = | 1.783216E-03 | f1(38) = | -.0545745 |
| f1(3) = | 1.991476E-03 | f1(39) = | -.0677396 |
| f1(4) = | 1.387367E-03 | f1(40) = | -7.368946E-02 |

| | | | |
|---|---|---|---|
| f1(5) | = 3.827771E-04 | f1(41) | = -7.219621E-02 |
| f1(6) | = -4.39708E-04 | f1(42) | = -6.400691E-02 |
| f1(7) | = -1.283091E-03 | f1(43) | = -5.064121E-02 |
| f1(8) | = -1.361171E-03 | f1(44) | = -3.413434E-02 |
| f1(9) | = -1.952853E-03 | f1(45) | = -.0166961 |
| f1(10) | = -1.66957E-03 | f1(46) | = -4.631148E-04 |
| f1(11) | = -1.58957E-03 | f1(47) | = 1.289227E-02 |
| f1(12) | = -7.449485E-04 | f1(48) | = 2.225322E-02 |
| f1(13) | = -7.163259E-04 | f1(49) | = 2.744549E-02 |
| f1(14) | = -1.431951E-03 | f1(50) | = 2.850179E-02 |
| f1(15) | = -3.52007E-03 | f1(51) | = 2.643269E-02 |
| f1(16) | = -6.858762E-03 | f1(52) | = 2.188209E-02 |
| f1(17) | = -1.119207E-02 | f1(53) | = .0165853 |
| f1(18) | = -1.658529E-02 | f1(54) | = 1.119206E-02 |
| f1(19) | = -.0218821 | f1(55) | = 6.858761E-03 |
| f1(20) | = -2.643268E-02 | f1(56) | = 3.52007E-03 |
| f1(21) | = -.0285018 | f1(57) | = 1.43195E-03 |
| f1(22) | = -2.744548E-02 | f1(58) | = 7.163236E-04 |
| f1(23) | = -2.225322E-02 | f1(59) | = 7.449468E-04 |
| f1(24) | = -1.289227E-02 | f1(60) | = 1.589569E-03 |
| f1(25) | = 4.631233E-04 | f1(61) | = 1.66957E-03 |
| f1(26) | = .0166961 | f1(62) | = 1.952853E-03 |
| f1(27) | = 3.413435E-02 | f1(63) | = 1.36117E-03 |
| f1(28) | = 5.064122E-02 | f1(64) | = 1.283091E-03 |
| f1(29) | = 6.400694E-02 | f1(65) | = 4.397079E-04 |
| f1(30) | = 7.219621E-02 | f1(66) | = -3.827774E-04 |
| f1(31) | = 7.368946E-02 | f1(67) | = -1.387367E-03 |
| f1(32) | = .0677396 | f1(68) | = -1.991475E-03 |
| f1(33) | = 5.457451E-02 | f1(69) | = -1.783217E-03 |
| f1(34) | = 3.541155E-02 | f1(70) | = -1.812965E-03 |
| f1(35) | = 1.226033E-02 | f1(71) | = -1.378243E-03 |

<u>f2(n) - 500 to 750 Hz</u>

| | | | |
|---|---|---|---|
| f2(0) | = 3.29006E-04 | f2(36) | = .079559 |
| f2(1) | = 1.478735E-03 | f2(37) | = 6.077661E-02 |
| f2(2) | = 2.525595E-03 | f2(38) | = 2.811491E-02 |
| f2(3) | = 3.132475E-03 | f2(39) | = -9.849666E-03 |
| f2(4) | = 2.072552E-03 | f2(40) | = -4.351676E-02 |
| f2(5) | = 8.003403E-04 | f2(41) | = -.0646829 |
| f2(6) | = -1.449775E-03 | f2(42) | = -6.899008E-02 |
| f2(7) | = -2.877092E-03 | f2(43) | = -5.683186E-02 |
| f2(8) | = -3.464928E-03 | f2(44) | = -.0329204 |
| f2(9) | = -2.881519E-03 | f2(45) | = -4.698264E-03 |
| f2(10) | = -6.880043E-04 | f2(46) | = .0201566 |
| f2(11) | = 1.339663E-03 | f2(47) | = 3.592261E-02 |
| f2(12) | = 3.083792E-03 | f2(48) | = 4.007546E-02 |
| f2(13) | = 2.334802E-03 | f2(49) | = .0339758 |
| f2(14) | = -7.400271E-04 | f2(50) | = 2.120152E-02 |
| f2(15) | = -5.686532E-03 | f2(51) | = 6.733697E-03 |
| f2(16) | = -1.096119E-02 | f2(52) | = -5.237205E-03 |
| f2(17) | = -1.369183E-02 | f2(53) | = -1.236606E-02 |
| f2(18) | = -1.236606E-02 | f2(54) | = -1.369183E-02 |

| | | | |
|---|---|---|---|
| f2(19) = | -5.237201E-03 | f2(55) = | -1.096119E-02 |
| f2(20) = | .0067337 | f2(56) = | -5.686531E-03 |
| f2(21) = | 2.120153E-02 | f2(57) = | -7.400276E-04 |
| f2(22) = | 3.397578E-02 | f2(58) = | 2.334805E-03 |
| f2(23) = | 4.007546E-02 | f2(59) = | 3.083793E-03 |
| f2(24) = | .0359226 | f2(60) = | 1.339663E-03 |
| f2(25) = | .0201566 | f2(61) = | -6.88005E-04 |
| f2(26) = | -4.698269E-03 | f2(62) = | -2.881518E-03 |
| f2(27) = | -3.292041E-02 | f2(63) = | -3.464929E-03 |
| f2(28) = | -5.683185E-02 | f2(64) = | -2.877092E-03 |
| f2(29) = | -.0689901 | f2(65) = | -1.449774E-03 |
| f2(30) = | -.0646829 | f2(66) = | 8.003408E-04 |
| f2(31) = | -4.351676E-02 | f2(67) = | 2.072552E-03 |
| f2(32) = | -9.849658E-03 | f2(68) = | 3.132475E-03 |
| f2(33) = | 2.811493E-02 | f2(69) = | 2.525596E-03 |
| f2(34) = | 6.077661E-02 | f2(70) = | 1.478734E-03 |
| f2(35) = | 7.955902E-02 | f2(71) = | 3.290059E-04 |

<u>f3(n) - 750 to 1000 Hz</u>

| | | | |
|---|---|---|---|
| f3(0) = | 9.509069E-04 | f3(36) = | -2.781808E-02 |
| f3(1) = | 2.198731E-03 | f3(37) = | -7.006082E-02 |
| f3(2) = | 2.444318E-03 | f3(38) = | -7.894783E-02 |
| f3(3) = | 2.260582E-03 | f3(39) = | -5.131538E-02 |
| f3(4) = | 4.705371E-04 | f3(40) = | -1.910019E-03 |
| f3(5) = | -1.719627E-03 | f3(41) = | 4.484323E-02 |
| f3(6) = | -3.007549E-03 | f3(42) = | 6.749694E-02 |
| f3(7) = | -3.568226E-03 | f3(43) = | 5.794703E-02 |
| f3(8) = | -2.238392E-03 | f3(44) = | 2.418963E-02 |
| f3(9) = | -7.424812E-04 | f3(45) = | -1.535784E-02 |
| f3(10) = | 1.619965E-04 | f3(46) = | -4.196692E-02 |
| f3(11) = | 7.741133E-04 | f3(47) = | -4.594031E-02 |
| f3(12) = | 9.169444E-04 | f3(48) = | -2.957347E-02 |
| f3(13) = | 2.671274E-03 | f3(49) = | -4.335095E-03 |
| f3(14) = | 5.72768E-03 | f3(50) = | 1.660142E-02 |
| f3(15) = | 8.344388E-03 | f3(51) = | 2.524926E-02 |
| f3(16) = | 8.021514E-03 | f3(52) = | 2.088413E-02 |
| f3(17) = | 1.661141E-03 | f3(53) = | 9.528255E-03 |
| f3(18) = | -9.528253E-03 | f3(54) = | -1.661145E-03 |
| f3(19) = | -2.088413E-02 | f3(55) = | -8.021516E-03 |
| f3(20) = | -2.524926E-02 | f3(56) = | -8.344388E-03 |
| f3(21) = | -1.660141E-02 | f3(57) = | -5.727682E-03 |
| f3(22) = | 4.335092E-03 | f3(58) = | -2.671273E-03 |
| f3(23) = | 2.957347E-02 | f3(59) = | -9.169448E-04 |
| f3(24) = | .0459403 | f3(60) = | -7.741124E-04 |
| f3(25) = | 4.196692E-02 | f3(61) = | -1.619955E-04 |
| f3(26) = | 1.535783E-02 | f3(62) = | 7.424804E-04 |
| f3(27) = | -2.418964E-02 | f3(63) = | 2.238393E-03 |
| f3(28) = | -5.794703E-02 | f3(64) = | 3.568226E-03 |
| f3(29) = | -6.749693E-02 | f3(65) = | 3.007549E-03 |
| f3(30) = | -4.484322E-02 | f3(66) = | 1.719626E-03 |
| f3(31) = | 1.910017E-03 | f3(67) = | -4.705373E-04 |
| f3(32) = | 5.131537E-02 | f3(68) = | -2.260582E-03 |

| | | | |
|---|---|---|---|
| f3(33) = | 7.894785E-02 | f3(69) = | -2.444318E-03 |
| f3(34) = | 7.006081E-02 | f3(70) = | -2.19873E-03 |
| f3(35) = | 2.781808E-02 | f3(71) = | -9.509069E-04 |

### f4(n) - 1000 to 1250 Hz

| | | | |
|---|---|---|---|
| f4(0) = | -1.547387E-03 | f4(36) = | 7.412682E-02 |
| f4(1) = | -9.403325E-04 | f4(37) = | 2.046397E-02 |
| f4(2) = | 1.705107E-03 | f4(38) = | -4.672247E-02 |
| f4(3) = | 3.284409E-03 | f4(39) = | -7.794309E-02 |
| f4(4) = | 1.735107E-03 | f4(40) = | -5.212579E-02 |
| f4(5) = | -1.119945E-03 | f4(41) = | 8.692244E-03 |
| f4(6) = | -3.47952E-03 | f4(42) = | 5.818448E-02 |
| f4(7) = | -2.938148E-03 | f4(43) = | 6.195537E-02 |
| f4(8) = | 3.449191E-04 | f4(44) = | 2.227494E-02 |
| f4(9) = | 2.521271E-03 | f4(45) = | -2.725728E-02 |
| f4(10) = | 1.901988E-03 | f4(46) = | -5.028862E-02 |
| f4(11) = | -8.933055E-04 | f4(47) = | -3.505078E-02 |
| f4(12) = | -2.754217E-03 | f4(48) = | 8.673328E-04 |
| f4(13) = | 2.375553E-04 | f4(49) = | 2.841566E-02 |
| f4(14) = | 5.720124E-03 | f4(50) = | .0303573 |
| f4(15) = | 7.077011E-03 | f4(51) = | 1.172394E-02 |
| f4(16) = | 3.851522E-04 | f4(52) = | -9.341259E-03 |
| f4(17) = | -1.148032E-02 | f4(53) = | -1.787843E-02 |
| f4(18) = | -1.787842E-02 | f4(54) = | -1.148031E-02 |
| f4(19) = | -9.341258E-03 | f4(55) = | 3.851526E-04 |
| f4(20) = | 1.172395E-02 | f4(56) = | 7.07701E-03 |
| f4(21) = | 3.035731E-02 | f4(57) = | 5.720122E-03 |
| f4(22) = | 2.841565E-02 | f4(58) = | 2.375556E-04 |
| f4(23) = | 8.673323E-04 | f4(59) = | -2.754216E-03 |
| f4(24) = | -3.505078E-02 | f4(60) = | -8.933063E-04 |
| f4(25) = | -5.028862E-02 | f4(61) = | 1.901992E-03 |
| f4(26) = | -2.725726E-02 | f4(62) = | 2.521272E-03 |
| f4(27) = | 2.227494E-02 | f4(63) = | 3.44919E-04 |
| f4(28) = | 6.195538E-02 | f4(64) = | -2.938148E-03 |
| f4(29) = | 5.818448E-02 | f4(65) = | -3.479522E-03 |
| f4(30) = | 8.692224E-03 | f4(66) = | -1.119944E-03 |
| f4(31) = | -.0521258 | f4(67) = | 1.735107E-03 |
| f4(32) = | -7.794309E-02 | f4(68) = | 3.284409E-03 |
| f4(33) = | -4.672246E-02 | f4(69) = | 1.705107E-03 |
| f4(34) = | .020464 | f4(70) = | -9.403322E-04 |
| f4(35) = | .0741268 | f4(71) = | -1.547388E-03 |

### f5(n) - 1250 to 1500 Hz

| | | | |
|---|---|---|---|
| f5(0) = | -1.077484E-03 | f5(36) = | -4.235782E-02 |
| f5(1) = | 2.451781E-04 | f5(37) = | -8.116985E-02 |
| f5(2) = | 2.213019E-03 | f5(38) = | -.0331005 |
| f5(3) = | 2.810156E-03 | f5(39) = | 4.783839E-02 |
| f5(4) = | 1.064582E-04 | f5(40) = | .0745233 |
| f5(5) = | -3.16625E-03 | f5(41) = | 2.244972E-02 |
| f5(6) = | -3.140288E-03 | f5(42) = | -4.794444E-02 |

| | | | |
|---|---|---|---|
| f5(7) | = -2.999361E-04 | f5(43) | = -.0624358 |
| f5(8) | = 2.552159E-03 | f5(44) | = -1.264966E-02 |
| f5(9) | = 3.117862E-03 | f5(45) | = 4.266636E-02 |
| f5(10) | = 1.250884E-03 | f5(46) | = 4.728455E-02 |
| f5(11) | = -5.942435E-04 | f5(47) | = 5.206136E-03 |
| f5(12) | = -1.790009E-03 | f5(48) | = -3.342525E-02 |
| f5(13) | = -3.968694E-03 | f5(49) | = -3.186685E-02 |
| f5(14) | = -4.86335E-03 | f5(50) | = -8.331285E-04 |
| f5(15) | = 2.444199E-04 | f5(51) | = .0227665 |
| f5(16) | = 9.572948E-03 | f5(52) | = 1.873709E-02 |
| f5(17) | = 1.267051E-02 | f5(53) | = -6.951063E-04 |
| f5(18) | = 6.951014E-04 | f5(54) | = -1.267051E-02 |
| f5(19) | = -1.873709E-02 | f5(55) | = -9.572948E-03 |
| f5(20) | = -.0227665 | f5(56) | = -2.444193E-04 |
| f5(21) | = 8.331333E-04 | f5(57) | = 4.863352E-03 |
| f5(22) | = 3.186685E-02 | f5(58) | = 3.968693E-03 |
| f5(23) | = 3.342525E-02 | f5(59) | = 1.790007E-03 |
| f5(24) | = -5.206137E-03 | f5(60) | = 5.942436E-04 |
| f5(25) | = -4.728456E-02 | f5(61) | = -1.250887E-03 |
| f5(26) | = -4.266633E-02 | f5(62) | = -3.117863E-03 |
| f5(27) | = 1.264966E-02 | f5(63) | = -2.552159E-03 |
| f5(28) | = 6.243581E-02 | f5(64) | = 2.99936E-04 |
| f5(29) | = 4.794443E-02 | f5(65) | = 3.140289E-03 |
| f5(30) | = -2.244973E-02 | f5(66) | = 3.166249E-03 |
| f5(31) | = -.0745233 | f5(67) | = -1.064581E-04 |
| f5(32) | = -4.783838E-02 | f5(68) | = -2.810156E-03 |
| f5(33) | = 3.310052E-02 | f5(69) | = -2.213019E-03 |
| f5(34) | = 8.116981E-02 | f5(70) | = -2.451778E-04 |
| f5(35) | = 4.235782E-02 | f5(71) | = 1.077484E-03 |

## f6(n) - 1500 to 1750 Hz

| | | | |
|---|---|---|---|
| f6(0) | = -6.432538E-04 | f6(36) | = 6.598325E-02 |
| f6(1) | = -1.883458E-03 | f6(37) | = -2.667495E-02 |
| f6(2) | = 4.271868E-04 | f6(38) | = -8.000176E-02 |
| f6(3) | = 3.065246E-03 | f6(39) | = -2.054875E-02 |
| f6(4) | = 1.134626E-03 | f6(40) | = 6.387653E-02 |
| f6(5) | = -2.995892E-03 | f6(41) | = .0551661 |
| f6(6) | = -2.849876E-03 | f6(42) | = -2.748879E-02 |
| f6(7) | = 1.537185E-03 | f6(43) | = -.0645323 |
| f6(8) | = 3.155873E-03 | f6(44) | = -1.098223E-02 |
| f6(9) | = -5.798835E-04 | f6(45) | = 4.958126E-02 |
| f6(10) | = -2.939913E-03 | f6(46) | = 3.548085E-02 |
| f6(11) | = 4.104597E-04 | f6(47) | = -2.221282E-02 |
| f6(12) | = 2.487659E-03 | f6(48) | = -3.979581E-02 |
| f6(13) | = -2.353335E-03 | f6(49) | = -2.705459E-03 |
| f6(14) | = -5.192677E-03 | f6(50) | = 2.888529E-02 |
| f6(15) | = 3.052733E-03 | f6(51) | = 1.603027E-02 |
| f6(16) | = 1.108669E-02 | f6(52) | = -1.293634E-02 |
| f6(17) | = 1.369201E-03 | f6(53) | = -1.678643E-02 |
| f6(18) | = -1.678643E-02 | f6(54) | = 1.369201E-03 |
| f6(19) | = -1.293635E-02 | f6(55) | = 1.108669E-02 |
| f6(20) | = 1.603027E-02 | f6(56) | = 3.052734E-03 |

| | | | |
|---|---|---|---|
| f6(21) = | .0288853 | f6(57) = | -5.192679E-03 |
| f6(22) = | -2.705466E-03 | f6(58) = | -2.353336E-03 |
| f6(23) = | -.0397958 | f6(59) = | 2.487661E-03 |
| f6(24) = | -2.221282E-02 | f6(60) = | 4.104598E-04 |
| f6(25) = | 3.548085E-02 | f6(61) = | -2.939915E-03 |
| f6(26) = | 4.958124E-02 | f6(62) = | -5.798837E-04 |
| f6(27) = | -1.098222E-02 | f6(63) = | 3.155871E-03 |
| f6(28) = | -.0645323 | f6(64) = | 1.537185E-03 |
| f6(29) = | -2.748879E-02 | f6(65) = | -2.849877E-03 |
| f6(30) = | 5.516611E-02 | f6(66) = | -2.995891E-03 |
| f6(31) = | 6.387653E-02 | f6(67) = | 1.134625E-03 |
| f6(32) = | -2.054874E-02 | f6(68) = | 3.065246E-03 |
| f6(33) = | -8.000178E-02 | f6(69) = | 4.271869E-04 |
| f6(34) = | -2.667493E-02 | f6(70) = | -1.883458E-03 |
| f6(35) = | 6.598325E-02 | f6(71) = | -6.43254E-04 |

<u>f7(n) - 1750 to 2000 Hz</u>

| | | | |
|---|---|---|---|
| f7(0) = | -1.331422E-03 | f7(36) = | -5.516587E-02 |
| f7(1) = | -1.441063E-03 | f7(37) = | -6.484276E-02 |
| f7(2) = | 1.728595E-03 | f7(38) = | 4.209616E-02 |
| f7(3) = | 3.060151E-03 | f7(39) = | 7.001066E-02 |
| f7(4) = | -6.35885E-04 | f7(40) = | -2.711672E-02 |
| f7(5) = | -3.31489E-03 | f7(41) = | -7.002083E-02 |
| f7(6) = | -2.920494E-05 | f7(42) = | 1.210748E-02 |
| f7(7) = | 3.459125E-03 | f7(43) = | 6.501156E-02 |
| f7(8) = | 1.365859E-03 | f7(44) = | 1.13572E-03 |
| f7(9) = | -1.986201E-03 | f7(45) = | -5.576929E-02 |
| f7(10) = | -1.81656E-03 | f7(46) = | -1.093587E-02 |
| f7(11) = | -1.80667E-04 | f7(47) = | 4.401698E-02 |
| f7(12) = | 1.912382E-03 | f7(48) = | 1.662893E-02 |
| f7(13) = | 4.147828E-03 | f7(49) = | -3.150684E-02 |
| f7(14) = | 1.639933E-04 | f7(50) = | -1.812277E-02 |
| f7(15) = | -8.536899E-03 | f7(51) = | .0201946 |
| f7(16) = | -4.304379E-03 | f7(52) = | 1.652823E-02 |
| f7(17) = | 1.301442E-02 | f7(53) = | -1.101988E-02 |
| f7(18) = | 1.101988E-02 | f7(54) = | -1.301442E-02 |
| f7(19) = | -1.652823E-02 | f7(55) = | 4.304378E-03 |
| f7(20) = | -.0201946 | f7(56) = | 8.536899E-03 |
| f7(21) = | 1.812278E-02 | f7(57) = | -1.639954E-04 |
| f7(22) = | 3.150683E-02 | f7(58) = | -4.147826E-03 |
| f7(23) = | -1.662893E-02 | f7(59) = | -1.912382E-03 |
| f7(24) = | -4.401699E-02 | f7(60) = | 1.806667E-04 |
| f7(25) = | 1.093587E-02 | f7(61) = | 1.816562E-03 |
| f7(26) = | 5.576927E-02 | f7(62) = | 1.986202E-03 |
| f7(27) = | -1.135718E-03 | f7(63) = | -1.365858E-03 |
| f7(28) = | -6.501156E-02 | f7(64) = | -3.459125E-03 |
| f7(29) = | -1.210748E-02 | f7(65) = | 2.920572E-05 |
| f7(30) = | 7.002083E-02 | f7(66) = | 3.31489E-03 |
| f7(31) = | 2.711673E-02 | f7(67) = | 6.358855E-04 |
| f7(32) = | -7.001066E-02 | f7(68) = | -3.060151E-03 |
| f7(33) = | -4.209617E-02 | f7(69) = | -1.728594E-03 |
| f7(34) = | 6.484275E-02 | f7(70) = | 1.441063E-03 |

| | | | |
|---|---|---|---|
| f7(35) = | 5.516587E-02 | f7(71) = | 1.331422E-03 |

f8(n) - 2000 to 2250 Hz

| | | | |
|---|---|---|---|
| f8(0) = | -1.331422E-03 | f8(36) = | -5.516587E-02 |
| f8(1) = | 1.441063E-03 | f8(37) = | 6.484276E-02 |
| f8(2) = | 1.728595E-03 | f8(38) = | 4.209616E-02 |
| f8(3) = | -3.060151E-03 | f8(39) = | -7.001066E-02 |
| f8(4) = | -6.35885E-04 | f8(40) = | -2.711672E-02 |
| f8(5) = | 3.31489E-03 | f8(41) = | 7.002083E-02 |
| f8(6) = | -2.920494E-05 | f8(42) = | 1.210748E-02 |
| f8(7) = | -3.459125E-03 | f8(43) = | -6.501156E-02 |
| f8(8) = | 1.365859E-03 | f8(44) = | 1.13572E-03 |
| f8(9) = | 1.986201E-03 | f8(45) = | 5.576929E-02 |
| f8(10) = | -1.81656E-03 | f8(46) = | -1.093587E-02 |
| f8(11) = | 1.80667E-04 | f8(47) = | -4.401698E-02 |
| f8(12) = | 1.912382E-03 | f8(48) = | 1.662893E-02 |
| f8(13) = | -4.147828E-03 | f8(49) = | 3.150684E-02 |
| f8(14) = | 1.639933E-04 | f8(50) = | -1.812277E-02 |
| f8(15) = | 8.536899E-03 | f8(51) = | -.0201946 |
| f8(16) = | -4.304379E-03 | f8(52) = | 1.652823E-02 |
| f8(17) = | -1.301442E-02 | f8(53) = | 1.101988E-02 |
| f8(18) = | 1.101988E-02 | f8(54) = | -1.301442E-02 |
| f8(19) = | 1.652823E-02 | f8(55) = | -4.304378E-03 |
| f8(20) = | -.0201946 | f8(56) = | 8.536899E-03 |
| f8(21) = | -1.812278E-02 | f8(57) = | 1.639954E-04 |
| f8(22) = | 3.150683E-02 | f8(58) = | -4.147826E-03 |
| f8(23) = | 1.662893E-02 | f8(59) = | 1.912382E-03 |
| f8(24) = | -4.401699E-02 | f8(60) = | 1.806667E-04 |
| f8(25) = | -1.093587E-02 | f8(61) = | -1.816562E-03 |
| f8(26) = | 5.576927E-02 | f8(62) = | 1.986202E-03 |
| f8(27) = | 1.135718E-03 | f8(63) = | 1.365858E-03 |
| f8(28) = | -6.501156E-02 | f8(64) = | -3.459125E-03 |
| f8(29) = | 1.210748E-02 | f8(65) = | -2.920572E-05 |
| f8(30) = | 7.002083E-02 | f8(66) = | 3.31489E-03 |
| f8(31) = | -2.711673E-02 | f8(67) = | -6.358855E-04 |
| f8(32) = | -7.001066E-02 | f8(68) = | -3.060151E-03 |
| f8(33) = | 4.209617E-02 | f8(69) = | 1.728594E-03 |
| f8(34) = | 6.484275E-02 | f8(70) = | 1.441063E-03 |
| f8(35) = | -5.516587E-02 | f8(71) = | -1.331422E-03 |

f9(n) - 2250 to 2500 Hz

| | | | |
|---|---|---|---|
| f9(0) = | -6.432538E-04 | f9(36) = | 6.598325E-02 |
| f9(1) = | 1.883458E-03 | f9(37) = | 2.667495E-02 |
| f9(2) = | 4.271868E-04 | f9(38) = | -8.000176E-02 |
| f9(3) = | -3.065246E-03 | f9(39) = | 2.054875E-02 |
| f9(4) = | 1.134626E-03 | f9(40) = | 6.387653E-02 |
| f9(5) = | 2.995892E-03 | f9(41) = | -.0551661 |
| f9(6) = | -2.849876E-03 | f9(42) = | -2.748879E-02 |
| f9(7) = | -1.537185E-03 | f9(43) = | .0645323 |
| f9(8) = | 3.155873E-03 | f9(44) = | -1.098223E-02 |

```
f9(9)  =  5.798835E-04              f9(45) = -4.958126E-02
f9(10) = -2.939913E-03              f9(46) =  3.548085E-02
f9(11) = -4.104597E-04              f9(47) =  2.221282E-02
f9(12) =  2.487659E-03              f9(48) = -3.979581E-02
f9(13) =  2.353335E-03              f9(49) =  2.705459E-03
f9(14) = -5.192677E-03              f9(50) =  2.888529E-02
f9(15) = -3.052733E-03              f9(51) = -1.603027E-02
f9(16) =  1.108669E-02              f9(52) = -1.293634E-02
f9(17) = -1.369201E-03              f9(53) =  1.678643E-02
f9(18) = -1.678643E-02              f9(54) =  1.369201E-03
f9(19) =  1.293635E-02              f9(55) = -1.108669E-02
f9(20) =  1.603027E-02              f9(56) =  3.052734E-03
f9(21) = -.0288853                  f9(57) =  5.192679E-03
f9(22) = -2.705466E-03              f9(58) = -2.353336E-03
f9(23) =  .0397958                  f9(59) = -2.487661E-03
f9(24) = -2.221282E-02              f9(60) =  4.104598E-04
f9(25) = -3.548085E-02              f9(61) =  2.939915E-03
f9(26) =  4.958124E-02              f9(62) = -5.798837E-04
f9(27) =  1.098222E-02              f9(63) = -3.155871E-03
f9(28) = -.0645323                  f9(64) =  1.537185E-03
f9(29) =  2.748879E-02              f9(65) =  2.849877E-03
f9(30) =  5.516611E-02              f9(66) = -2.995891E-03
f9(31) = -6.387653E-02              f9(67) = -1.134625E-03
f9(32) = -2.054874E-02              f9(68) =  3.065246E-03
f9(33) =  8.000178E-02              f9(69) = -4.271869E-04
f9(34) = -2.667493E-02              f9(70) = -1.883458E-03
f9(35) = -6.598325E-02              f9(71) =  6.43254E-04
```

<u>f10(n) - 2500 to 2750 Hz</u>

```
f10(0)  = -1.077484E-03             f10(36) = -4.235782E-02
f10(1)  = -2.451781E-04             f10(37) =  8.116985E-02
f10(2)  =  2.213019E-03             f10(38) = -.0331005
f10(3)  = -2.810156E-03             f10(39) = -4.783839E-02
f10(4)  =  1.064582E-04             f10(40) =  .0745233
f10(5)  =  3.16625E-03              f10(41) = -2.244972E-02
f10(6)  = -3.140288E-03             f10(42) = -4.794444E-02
f10(7)  =  2.999361E-04             f10(43) =  .0624358
f10(8)  =  2.552159E-03             f10(44) = -1.264966E-02
f10(9)  = -3.117862E-03             f10(45) = -4.266636E-02
f10(10) =  1.250884E-03             f10(46) =  4.728455E-02
f10(11) =  5.942435E-04             f10(47) = -5.206136E-03
f10(12) = -1.790009E-03             f10(48) = -3.342525E-02
f10(13) =  3.968694E-03             f10(49) =  3.186685E-02
f10(14) = -4.86335E-03              f10(50) = -8.331285E-04
f10(15) = -2.444199E-04             f10(51) = -.0227665
f10(16) =  9.572948E-03             f10(52) =  1.873709E-02
f10(17) = -1.267051E-02             f10(53) =  6.951063E-04
f10(18) =  6.951014E-04             f10(54) = -1.267051E-02
f10(19) =  1.873709E-02             f10(55) =  9.572948E-03
f10(20) = -.0227665                 f10(56) = -2.444193E-04
f10(21) = -8.331333E-04             f10(57) = -4.863352E-03
f10(22) =  3.186685E-02             f10(58) =  3.968693E-03
```

| | | | |
|---|---|---|---|
| f10(23) | = -3.342525E-02 | f10(59) | = -1.790007E-03 |
| f10(24) | = -5.206137E-03 | f10(60) | = 5.942436E-04 |
| f10(25) | = 4.728456E-02 | f10(61) | = 1.250887E-03 |
| f10(26) | = -4.266633E-02 | f10(62) | = -3.117863E-03 |
| f10(27) | = -1.264966E-02 | f10(63) | = 2.552159E-03 |
| f10(28) | = 6.243581E-02 | f10(64) | = 2.99936E-04 |
| f10(29) | = -4.794443E-02 | f10(65) | = -3.140289E-03 |
| f10(30) | = -2.244973E-02 | f10(66) | = 3.166249E-03 |
| f10(31) | = .0745233 | f10(67) | = 1.064581E-04 |
| f10(32) | = -4.783838E-02 | f10(68) | = -2.810156E-03 |
| f10(33) | = -3.310052E-02 | f10(69) | = 2.213019E-03 |
| f10(34) | = 8.116981E-02 | f10(70) | = -2.451778E-04 |
| f10(35) | = -4.235782E-02 | f10(71) | = -1.077484E-03 |

<u>f11(n) - 2750 to 3000 Hz</u>

| | | | |
|---|---|---|---|
| f11(0) | = -1.547387E-03 | f11(36) | = 7.412682E-02 |
| f11(1) | = 9.403325E-04 | f11(37) | = -2.046397E-02 |
| f11(2) | = 1.705107E-03 | f11(38) | = -4.672247E-02 |
| f11(3) | = -3.284409E-03 | f11(39) | = 7.794309E-02 |
| f11(4) | = 1.735107E-03 | f11(40) | = -5.212579E-02 |
| f11(5) | = 1.119945E-03 | f11(41) | = -8.692244E-03 |
| f11(6) | = -3.47952E-03 | f11(42) | = 5.818448E-02 |
| f11(7) | = 2.938148E-03 | f11(43) | = -6.195537E-02 |
| f11(8) | = 3.449191E-04 | f11(44) | = 2.227494E-02 |
| f11(9) | = -2.521271E-03 | f11(45) | = 2.725728E-02 |
| f11(10) | = 1.901988E-03 | f11(46) | = -5.028862E-02 |
| f11(11) | = 8.933055E-04 | f11(47) | = 3.505078E-02 |
| f11(12) | = -2.754217E-03 | f11(48) | = 8.673328E-04 |
| f11(13) | = -2.375553E-04 | f11(49) | = -2.841566E-02 |
| f11(14) | = 5.720124E-03 | f11(50) | = .0303573 |
| f11(15) | = -7.077011E-03 | f11(51) | = -1.172394E-02 |
| f11(16) | = 3.851522E-04 | f11(52) | = -9.341259E-03 |
| f11(17) | = 1.148032E-02 | f11(53) | = 1.787843E-02 |
| f11(18) | = -1.787842E-02 | f11(54) | = -1.148031E-02 |
| f11(19) | = 9.341258E-03 | f11(55) | = -3.851526E-04 |
| f11(20) | = 1.172395E-02 | f11(56) | = 7.07701E-03 |
| f11(21) | = -3.035731E-02 | f11(57) | = -5.720122E-03 |
| f11(22) | = 2.841565E-02 | f11(58) | = 2.375556E-04 |
| f11(23) | = -8.673323E-04 | f11(59) | = 2.754216E-03 |
| f11(24) | = -3.505078E-02 | f11(60) | = -8.933063E-04 |
| f11(25) | = 5.028862E-02 | f11(61) | = -1.901992E-03 |
| f11(26) | = -2.725726E-02 | f11(62) | = 2.521272E-03 |
| f11(27) | = -2.227494E-02 | f11(63) | = -3.44919E-04 |
| f11(28) | = 6.195538E-02 | f11(64) | = -2.938148E-03 |
| f11(29) | = -5.818448E-02 | f11(65) | = 3.479522E-03 |
| f11(30) | = 8.692224E-03 | f11(66) | = -1.119944E-03 |
| f11(31) | = .0521258 | f11(67) | = -1.735107E-03 |
| f11(32) | = -7.794309E-02 | f11(68) | = 3.284409E-03 |
| f11(33) | = 4.672246E-02 | f11(69) | = -1.705107E-03 |
| f11(34) | = .020464 | f11(70) | = -9.403322E-04 |
| f11(35) | = -.0741268 | f11(71) | = 1.547388E-03 |

<u>f12(n) - 3000 to 3250 Hz</u>

| | | | |
|---|---|---|---|
| f12(0) = | 9.509069E-04 | f12(36) = | -2.781808E-02 |
| f12(1) = | -2.198731E-03 | f12(37) = | 7.006082E-02 |
| f12(2) = | 2.444318E-03 | f12(38) = | -7.894783E-02 |
| f12(3) = | -2.260582E-03 | f12(39) = | 5.131538E-02 |
| f12(4) = | 4.705371E-04 | f12(40) = | -1.910019E-03 |
| f12(5) = | 1.719627E-03 | f12(41) = | -4.484323E-02 |
| f12(6) = | -3.007549E-03 | f12(42) = | 6.749694E-02 |
| f12(7) = | 3.568226E-03 | f12(43) = | -5.794703E-02 |
| f12(8) = | -2.238392E-03 | f12(44) = | 2.418963E-02 |
| f12(9) = | 7.424812E-04 | f12(45) = | 1.535784E-02 |
| f12(10) = | 1.619965E-04 | f12(46) = | -4.196692E-02 |
| f12(11) = | -7.741133E-04 | f12(47) = | 4.594031E-02 |
| f12(12) = | 9.169444E-04 | f12(48) = | -2.957347E-02 |
| f12(13) = | -2.671274E-03 | f12(49) = | 4.335095E-03 |
| f12(14) = | 5.72768E-03 | f12(50) = | 1.660142E-02 |
| f12(15) = | -8.344388E-03 | f12(51) = | -2.524926E-02 |
| f12(16) = | 8.021514E-03 | f12(52) = | 2.088413E-02 |
| f12(17) = | -1.661141E-03 | f12(53) = | -9.528255E-03 |
| f12(18) = | -9.528253E-03 | f12(54) = | -1.661145E-03 |
| f12(19) = | 2.088413E-02 | f12(55) = | 8.021516E-03 |
| f12(20) = | -2.524926E-02 | f12(56) = | -8.344388E-03 |
| f12(21) = | 1.660141E-02 | f12(57) = | 5.727682E-03 |
| f12(22) = | 4.335092E-03 | f12(58) = | -2.671273E-03 |
| f12(23) = | -2.957347E-02 | f12(59) = | 9.169448E-04 |
| f12(24) = | .0459403 | f12(60) = | -7.741124E-04 |
| f12(25) = | -4.196692E-02 | f12(61) = | 1.619955E-04 |
| f12(26) = | 1.535783E-02 | f12(62) = | 7.424804E-04 |
| f12(27) = | 2.418964E-02 | f12(63) = | -2.238393E-03 |
| f12(28) = | -5.794703E-02 | f12(64) = | 3.568226E-03 |
| f12(29) = | 6.749693E-02 | f12(65) = | -3.007549E-03 |
| f12(30) = | -4.484322E-02 | f12(66) = | 1.719626E-03 |
| f12(31) = | -1.910017E-03 | f12(67) = | 4.705373E-04 |
| f12(32) = | 5.131537E-02 | f12(68) = | -2.260582E-03 |
| f12(33) = | -7.894785E-02 | f12(69) = | 2.444318E-03 |
| f12(34) = | 7.006081E-02 | f12(70) = | -2.19873E-03 |
| f12(35) = | -2.781808E-02 | f12(71) = | 9.509069E-04 |

<u>f13(n) - 3250 to 3500 Hz</u>

| | | | |
|---|---|---|---|
| f13(0) = | 3.29006E-04 | f13(36) = | .079559 |
| f13(1) = | -1.478735E-03 | f13(37) = | -6.077661E-02 |
| f13(2) = | 2.525595E-03 | f13(38) = | 2.811491E-02 |
| f13(3) = | -3.132475E-03 | f13(39) = | 9.849666E-03 |
| f13(4) = | 2.072552E-03 | f13(40) = | -4.351676E-02 |
| f13(5) = | -8.003403E-04 | f13(41) = | .0646829 |
| f13(6) = | -1.449775E-03 | f13(42) = | -6.899008E-02 |
| f13(7) = | 2.877092E-03 | f13(43) = | 5.683186E-02 |
| f13(8) = | -3.464928E-03 | f13(44) = | -.0329204 |
| f13(9) = | 2.881519E-03 | f13(45) = | 4.698264E-03 |
| f13(10) = | -6.880043E-04 | f13(46) = | .0201566 |

| | | | |
|---|---|---|---|
| f13(11) = | -1.339663E-03 | f13(47) = | -3.592261E-02 |
| f13(12) = | 3.083792E-03 | f13(48) = | 4.007546E-02 |
| f13(13) = | -2.334802E-03 | f13(49) = | -.0339758 |
| f13(14) = | -7.400271E-04 | f13(50) = | 2.120152E-02 |
| f13(15) = | 5.686532E-03 | f13(51) = | -6.733697E-03 |
| f13(16) = | -1.096119E-02 | f13(52) = | -5.237205E-03 |
| f13(17) = | 1.369183E-02 | f13(53) = | 1.236606E-02 |
| f13(18) = | -1.236606E-02 | f13(54) = | -1.369183E-02 |
| f13(19) = | 5.237201E-03 | f13(55) = | 1.096119E-02 |
| f13(20) = | .0067337 | f13(56) = | -5.686531E-03 |
| f13(21) = | -2.120153E-02 | f13(57) = | 7.400276E-04 |
| f13(22) = | 3.397578E-02 | f13(58) = | 2.334805E-03 |
| f13(23) = | -4.007546E-02 | f13(59) = | -3.083793E-03 |
| f13(24) = | .0359226 | f13(60) = | 1.339663E-03 |
| f13(25) = | -.0201566 | f13(61) = | 6.88005E-04 |
| f13(26) = | -4.698269E-03 | f13(62) = | -2.881518E-03 |
| f13(27) = | 3.292041E-02 | f13(63) = | 3.464929E-03 |
| f13(28) = | -5.683185E-02 | f13(64) = | -2.877092E-03 |
| f13(29) = | .0689901 | f13(65) = | 1.449774E-03 |
| f13(30) = | -.0646829 | f13(66) = | 8.003408E-04 |
| f13(31) = | 4.351676E-02 | f13(67) = | -2.072552E-03 |
| f13(32) = | -9.849658E-03 | f13(68) = | 3.132475E-03 |
| f13(33) = | -2.811493E-02 | f13(69) = | -2.525596E-03 |
| f13(34) = | 6.077661E-02 | f13(70) = | 1.478734E-03 |
| f13(35) = | -7.955902E-02 | f13(71) = | -3.290059E-04 |

## f14(n) - 3500 to 3750 Hz

| | | | |
|---|---|---|---|
| f14(0) = | 1.378243E-03 | f14(36) = | -1.226034E-02 |
| f14(1) = | -1.812966E-03 | f14(37) = | 3.541158E-02 |
| f14(2) = | 1.783216E-03 | f14(38) = | -.0545745 |
| f14(3) = | -1.991476E-03 | f14(39) = | .0677396 |
| f14(4) = | 1.387367E-03 | f14(40) = | -7.368946E-02 |
| f14(5) = | -3.827771E-04 | f14(41) = | 7.219621E-02 |
| f14(6) = | -4.39708E-04 | f14(42) = | -6.400691E-02 |
| f14(7) = | 1.283091E-03 | f14(43) = | 5.064121E-02 |
| f14(8) = | -1.361171E-03 | f14(44) = | -3.413434E-02 |
| f14(9) = | 1.952853E-03 | f14(45) = | .0166961 |
| f14(10) = | -1.66957E-03 | f14(46) = | -4.631148E-04 |
| f14(11) = | 1.58957E-03 | f14(47) = | -1.289227E-02 |
| f14(12) = | -7.449485E-04 | f14(48) = | 2.225322E-02 |
| f14(13) = | 7.163259E-04 | f14(49) = | -2.744549E-02 |
| f14(14) = | -1.431951E-03 | f14(50) = | 2.850179E-02 |
| f14(15) = | 3.52007E-03 | f14(51) = | -2.643269E-02 |
| f14(16) = | -6.858762E-03 | f14(52) = | 2.188209E-02 |
| f14(17) = | 1.119207E-02 | f14(53) = | -.0165853 |
| f14(18) = | -1.658529E-02 | f14(54) = | 1.119206E-02 |
| f14(19) = | .0218821 | f14(55) = | -6.858761E-03 |
| f14(20) = | -2.643268E-02 | f14(56) = | 3.52007E-03 |
| f14(21) = | .0285018 | f14(57) = | -1.43195E-03 |
| f14(22) = | -2.744548E-02 | f14(58) = | 7.163236E-04 |
| f14(23) = | 2.225322E-02 | f14(59) = | -7.449468E-04 |
| f14(24) = | -1.289227E-02 | f14(60) = | 1.589569E-03 |

| | | | |
|---|---|---|---|
| f14(25) = | -4.631233E-04 | f14(61) = | -1.66957E-03 |
| f14(26) = | .0166961 | f14(62) = | 1.952853E-03 |
| f14(27) = | -3.413435E-02 | f14(63) = | -1.36117E-03 |
| f14(28) = | 5.064122E-02 | f14(64) = | 1.283091E-03 |
| f14(29) = | -6.400694E-02 | f14(65) = | -4.397079E-04 |
| f14(30) = | 7.219621E-02 | f14(66) = | -3.827774E-04 |
| f14(31) = | -7.368946E-02 | f14(67) = | 1.387367E-03 |
| f14(32) = | .0677396 | f14(68) = | -1.991475E-03 |
| f14(33) = | -5.457451E-02 | f14(69) = | 1.783217E-03 |
| f14(34) = | 3.541155E-02 | f14(70) = | -1.812965E-03 |
| f14(35) = | -1.226033E-02 | f14(71) = | 1.378243E-03 |

<u>f15(n) - 3750 to 4000 Hz</u>

| | | | |
|---|---|---|---|
| f15(0) = | 1.053927E-03 | f15(36) = | .0710003 |
| f15(1) = | -1.841017E-03 | f15(37) = | -6.973641E-02 |
| f15(2) = | 2.063785E-03 | f15(38) = | 6.720698E-02 |
| f15(3) = | -2.345845E-03 | f15(39) = | -6.351141E-02 |
| f15(4) = | 1.506928E-03 | f15(40) = | 5.874324E-02 |
| f15(5) = | -1.367166E-03 | f15(41) = | -5.310814E-02 |
| f15(6) = | 7.006013E-04 | f15(42) = | 4.672493E-02 |
| f15(7) = | -2.67798E-04 | f15(43) = | -3.989666E-02 |
| f15(8) = | -1.244974E-03 | f15(44) = | 3.276445E-02 |
| f15(9) = | 2.206876E-03 | f15(45) = | -2.570829E-02 |
| f15(10) = | -3.523216E-03 | f15(46) = | 1.885466E-02 |
| f15(11) = | 4.592768E-03 | f15(47) = | -1.258937E-02 |
| f15(12) = | -6.402049E-03 | f15(48) = | 6.863879E-03 |
| f15(13) = | 7.726114E-03 | f15(49) = | -1.987042E-03 |
| f15(14) = | -8.812724E-03 | f15(50) = | -2.034144E-03 |
| f15(15) = | 9.322696E-03 | f15(51) = | 5.012196E-03 |
| f15(16) = | -9.782126E-03 | f15(52) = | -7.254503E-03 |
| f15(17) = | 9.669026E-03 | f15(53) = | 8.83648E-03 |
| f15(18) = | -8.836473E-03 | f15(54) = | -9.669022E-03 |
| f15(19) = | 7.254504E-03 | f15(55) = | 9.782125E-03 |
| f15(20) = | -5.01219E-03 | f15(56) = | -9.322696E-03 |
| f15(21) = | 2.034141E-03 | f15(57) = | 8.812728E-03 |
| f15(22) = | 1.987044E-03 | f15(58) = | -7.726108E-03 |
| f15(23) = | -6.86388E-03 | f15(59) = | 6.402048E-03 |
| f15(24) = | 1.258937E-02 | f15(60) = | -4.592766E-03 |
| f15(25) = | -1.885467E-02 | f15(61) = | 3.523215E-03 |
| f15(26) = | 2.570829E-02 | f15(62) = | -2.206874E-03 |
| f15(27) = | -3.276445E-02 | f15(63) = | 1.244974E-03 |
| f15(28) = | 3.989666E-02 | f15(64) = | 2.677977E-04 |
| f15(29) = | -4.672494E-02 | f15(65) = | -7.006018E-04 |
| f15(30) = | 5.310812E-02 | f15(66) = | 1.367166E-03 |
| f15(31) = | -5.874324E-02 | f15(67) = | -1.506929E-03 |
| f15(32) = | 6.351142E-02 | f15(68) = | 2.345844E-03 |
| f15(33) = | -.067207 | f15(69) = | -2.063786E-03 |
| f15(34) = | 6.973638E-02 | f15(70) = | 1.841016E-03 |
| f15(35) = | -.0710003 | f15(71) = | -1.053927E-03 |

# APPENDIX B - SOFTWARE DESIGN

Computer simulation was used to compare the coding algorithms presented in this thesis. The software can be classified into two categories : those programs that actually perform the SBC-ADPCM processing and those utilities that help develop and evaluate the coders.

With the exception of the QMF utilities, FQMFBANK and TRUNCATE, all code was written in FORTRAN and run on an IBM VM/SP mainframe. The QMF programs were used during the initial phase of this project and were written in BASIC for an IBM AT personal computer.

Listings of the actual programs are included in Appendix C. The remainder of this appendix describes the organization and approach taken in their development.

## B.1 MAIN LINE

Programs: SBC, CODN, CODLMS, CODLS, DECN, DECLMS, DECLS

The main line is a top level view of the sub-band coding process. A block diagram of the SBC process is shown in Fig. B-1. Decompositions of CODN, CODLMS, CODLS, DECN, DECLMS, and DECLS are in Fig. B-2. This standard top-down approach allows the details of module implementation to be filled in after main line coding, provided that the modules have been adequately specified.

### B.1.1 Walkthrough

The first task of the SBC main program is to prompt the user for coder bit rate, choice of predictor, and, if a predictor is selected, the order of the prediction desired.

SBC takes 8 KHz inputs and processes them 128 samples at a time (16ms blocks). These are first passed through the analyzing Quadrature Mirror Filter Bank (QMF), the coefficients of which are read in by INIT, producing 16 channels of 8 samples each (recall that each of the subbands is decimated by 16 before being quantized). Only the first 13 bands, covering 0 to 3250Hz, are actually coded. The last three bands are assumed to be zero at the inverse QMF bank. This is in recognition of telephony bandwidth constraints (typically around 3200Hz).

The next step is determine the maximum amplitude of each subband signal over that block, known as the subband characteristic, in preparation for the bit allocation computation (see CMPCHR, DAB, and PDAB in Section B.3).

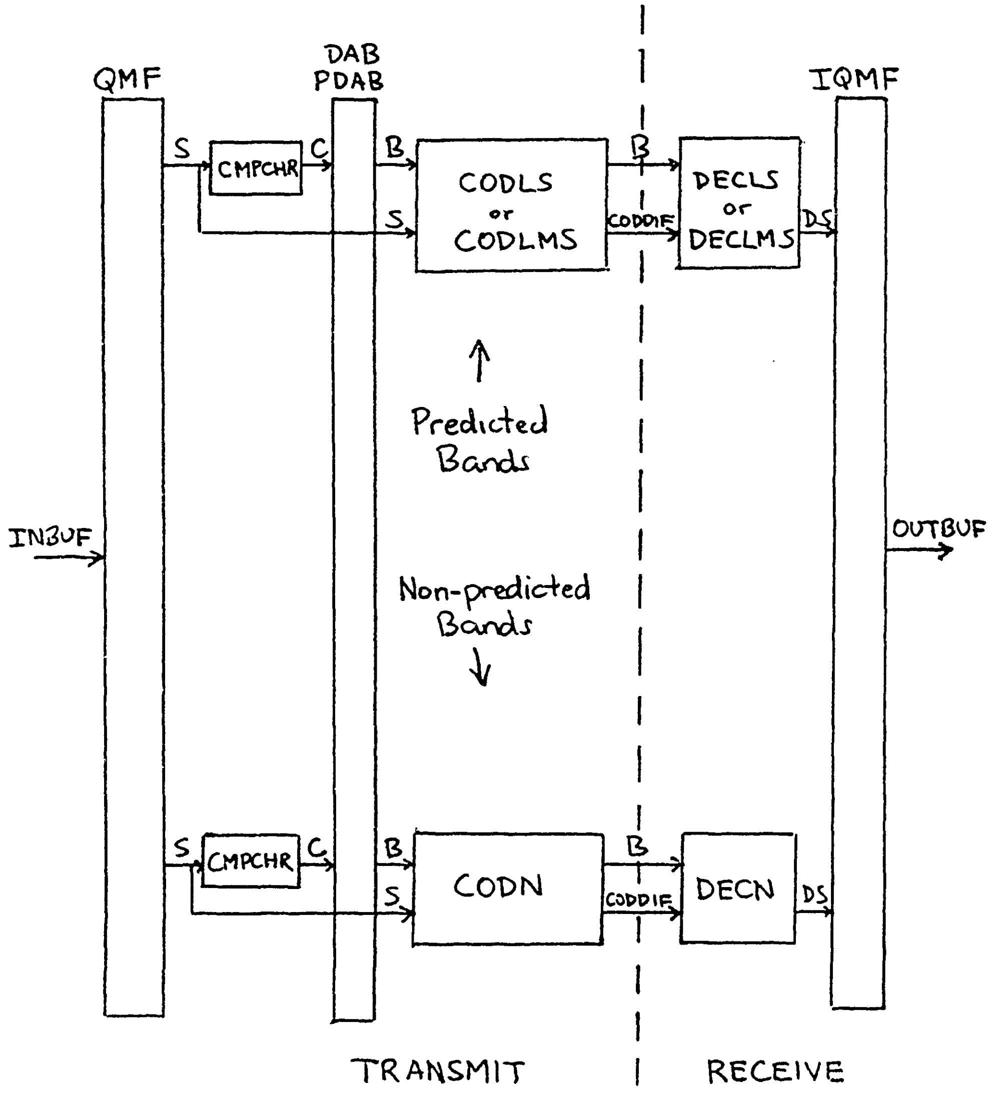


Figure B-1. SBC Block Diagram

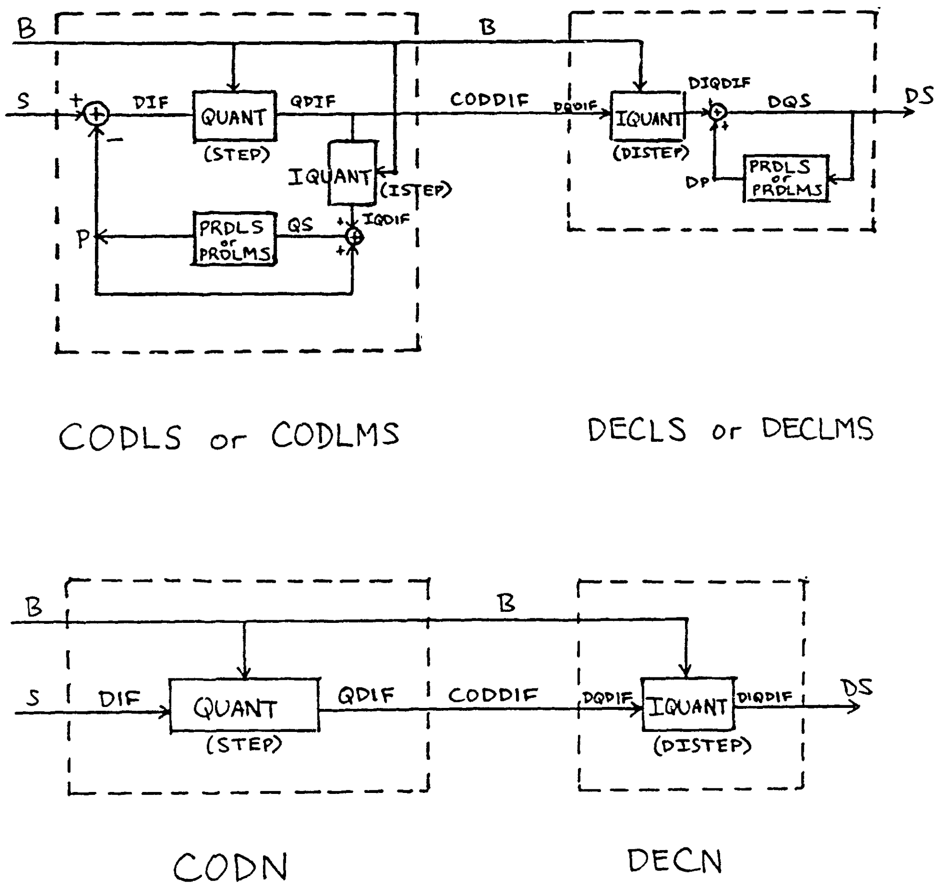


**Figure B-2. COD*, DEC* Block Diagrams**

Using the output of the dynamic bit allocation module, and the type of prediction selected, the 13 sets of 8 subband samples are coded by either CODN (plain adaptive quantizer), CODLMS (adaptive quantization with LMS transversal predictor), or CODLS (adaptive quantization with LS lattice predictor).

Once all 13 bands have been coded, each is then processed with the appropriate decoding block (DECN, DECLMS, or DECLS). The only data that these blocks share with the corresponding coding modules are the coded subband samples and the bit allocation information.

The final task for SBC is to call the inverse QMF routine, IQMF, to reconstruct the 16ms block from the 13 sets of 8 decoded samples. The entire block of 128 output samples is then written to file.

Successive blocks of samples are read in and processed in the same way. This continues until the end of the input file is reached.

### B.1.2 Variables

Input blocks come in 2-byte (16-bit) integer format. Output values are rounded to a the same format when written to file. However, internal calculations are made primarily with real, floating point values (e.g. all of the filter bank coefficients are real). Finite-length register limitations of practical hardware implementations are not considered. The only exceptions to this are the integer variables used to represent bit allocation and quantized subband data. This is necessary in order to simulate the effects of different coder bit rates.

Just as in actual voice store-and-forward systems, the internal variables of the coding and decoding halves are kept separate. In fact, the receiver has access only to the encoded data and not to any of the internal variables used during coding. (If we assume no transmission errors then corresponding data in the transmitter and receiver will turn out to be identical but, in general, this may not be true.) In this implementation, the names of variables used by the receiver half are derived by prefixing the analogous transmitter variable names with a 'd'.

Most of the variables declared in the SBC main routine are in the form of labelled COMMON statements. This FORTRAN construct simply identifies the global variables, those that subroutines can access even if they are not explicitly passed as arguments. Since the subroutine modules generally operate on many variables (consider, for example, CODLS), it is not practical to pass all of these variables in subroutine calls. Instead, each subroutine simply repeats the COMMON declaration for the labelled block that it needs and automatically can read and write to those same memory locations. For example, the adaptive quantizer declares 'COMMON /QNT/STEP(13)' in order to use the latest step sizes. Note that the availability of labels for COMMON areas of memory is particularly helpful here since different subroutines need different groups of variables.

One consequence of making a distinction between transmitter and receiver variables is that it is not practical to use COMMON in the QUANT, IQUANT, PRDLMS, and PRDLS routines. These modules appear in both the coder and decoder halves of the system and must be allowed to access the two sets of variables. Passing the appropriate set of arguments at each call was chosen over always reading both types of COMMON variables and determining which to use every time.

## B.2 QUADRATURE MIRROR FILTER BANK

Programs: INIT, QMF, IQMF

INIT is called by SBC to read in the QMF coefficients from a file. Only tap values for the first eight analysis bandpass filters, $f_0(n)$ to $f_7(n)$, are needed since the impulse responses of the other eight filters are related to these by

(B-1) $f_{15-k}(n) = (-1)^n f_k(n) \qquad 0 \le k \le 15$

Note that the code uses slightly different indices than (B-1) since arrays in FORTRAN must start with index 1 instead of 0.

The analysis bandpass filters (implemented by the QMF subroutine) all operate on the same tapped delay line. Since the output of the analysis bank is decimated by 16, computation can be reduced by shifting 16 samples at a time into the delay line and only calculating the output point that is saved.

The inverse QMF bank (implemented by IQMF), which adds the outputs of its 16 bandpass filters, is more complicated than the analyzing bank since the different subband signals require multiple tapped delay lines. However, some simplification is possible with the following observations about the reconstruction bandpass filters $g_k(n)$.

(B-2)
$$\begin{aligned} g_k(n) &= 16(-1)^k f_k(n) \\ g_{15-k}(n) &= 16(-1)^{15-k} f_{15-k}(n) \\ &= 16(-1)^{k+1}(-1)^n f_k(n) \\ &= 16(-1)^{n+1}(-1)^k f_k(n) \\ &= (-1)^{n+1} g_k(n) \end{aligned}$$

Thus,

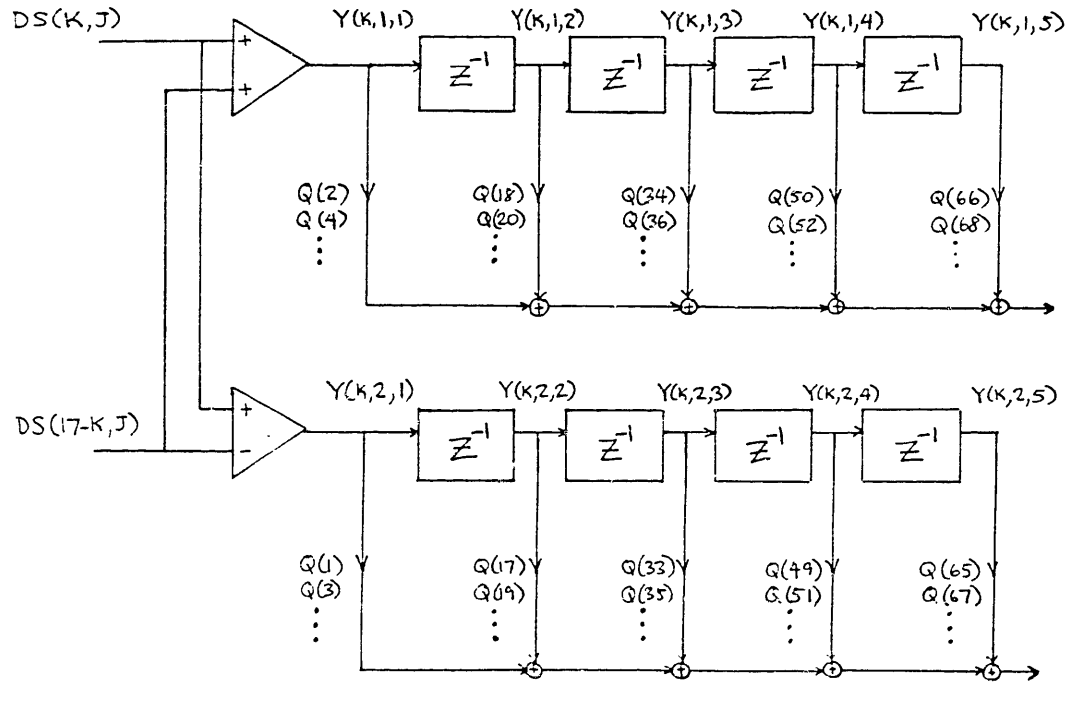
(B-3)
$$g_{15-k}(n) = \begin{cases} -g_k(n) & \text{for even } n \\ g_k(n) & \text{for odd } n \end{cases}$$

and so the effect of adding the outputs of two mirror image filters, $g_k(n)$ and $g_{15-k}(n)$, is

$$\text{(B-4)}\quad g_k(n)*s_k(n)+g_{15-k}(n)*s_{15-k}(n) = \begin{cases} g_k(n)*[s_k(n)-s_{15-k}(n)] & \text{for even } n \\ g_k(n)*[s_k(n)+s_{15-k}(n)] & \text{for odd } n \end{cases}$$

The inverse QMF must also upsample the subband signals before filtering. One way to do this is to insert 15 zeroes between every two samples. A better alternative is to recognize that this is equivalent to shifting the decimated subband signals down a tapped delay line and, for each shift, compute 16 outputs by multiplying the contents of the delay line with an appropriate subset of the filter impulse response. These subsets pick out every 16 of the 72 filter coefficients, ignoring those that would have been multiplied with zero if the subband were explicitly upsampled before filtering.

Fig. B-3 illustrates this technique for a symmetric pair of subbands, including the separate delay lines for odd and even indexed coefficients. Notice that the parity of the diagram is opposite to that of equation (B-4), reflecting the use of FORTRAN arrays that begin with 1, not 0. Thus, in the actual implementation, even indexed coefficients operate on a delay line corresponding to the sum of the subband signals while odd indexed coefficients use the difference.



(LABELS REFER TO FORTRAN VARIABLES)

Figure B-3. Inverse QMF Strategy

As part of the reconstruction process, IQMF scales all outputs by 16. It also makes sure that values outside the range of a 2-byte integer (the format of OUTBUF) are clipped to the maximum positive (32767) or negative (-32768) value.

## B.3 BIT ALLOCATION

Programs: CMPCHR, DAB, PDAB

The bit allocation code consists of three subroutines - CMPCHR, which selects the maximum sample amplitude of a subband block; DAB, which determines the bit assignments based upon the characteristics and bit rate; and PDAB, which modifies the bit distribution in the presence of prediction.

CMPCHR is straightforward. It simply finds the maximum amplitude of the eight subband samples in a given block.

DAB implements the algorithm described in Section 2.3, with the aid of two functions, RDOWN and BLEVEL. RDOWN rounds down a real number to the largest integer not greater than it. BLEVEL converts the n(i) computed from equation (2-17) to an integral bit assignment between 0 and 5, inclusive.

A special case that must be considered is a band with characteristic value 0. Since $\log_2 0$ is not computable, these bands cannot be included in the calculation of the n(i)

(B-5) $$n(i) = \frac{\text{BITS}}{P} - \frac{1}{P}\sum_{j=0}^{P-1} \log_2 C_j + \log_2 C_i$$

where P is the number of bands with nonzero power and BITS is the total number of bits available for sample quantization per subband sampling period,

(B-6) BITS = RDOWN(2×(KBPS-2.4375)).

KBPS is the coder bit rate expressed in $10^3$ bits/second.

Based on the values of P and BITS, it is determined whether or not there are enough nonzero bands to use up all the available bits. If not, then each of the P bands is assigned 5 bits and the allocation is complete. Otherwise, the calculation of the n(i) and initial bit distribution proceed normally. The remainder of DAB contains sections that deal with the separate cases of having extra bits left to allocate and having too many given out.

If there are additional bits available for distribution, all of the n(i)'s are increased by the minimum amount, SHIFT, needed to add at least one to the total allocation. To determine SHIFT, each band is associated with a scale change, DELTA, representing the minimum needed to add one bit to that band. SHIFT is then just the minimum of the DELTA's. If SHIFT is 0, then all nonzero bands already have the maximum 5 bits; an infinite loop is avoided by terminating execution when this happens. If SHIFT is greater than zero but still not enough to use up the extra bits, a new value of SHIFT is computed for the updated n(i).

The program flow for the case of the bands being given too many bits proceeds in the same manner. Whenever the current allocation equals the total allowed by the coder rate, the temporary assignments are written to B(i). Bands which had zero power (flagged by CZERO), are always given 0 bits.

The third and final subroutine whi[illegible] [illegible]etermines bit allocation, PDAB, is called by the main program, SBC, if the coder is run with prediction. PDAB looks at the output of DAB and checks to see that all bands with prediction have at least two bits. Each predicted band, i, that needs more is allowed to take one at a time from any band higher in frequency. Bits are taken from different bands before one band loses two. If the predicted band still has less than two but all higher frequencies are exhausted of bits, then a warning message is issued.

## B.4 ADAPTIVE QUANTIZER

Programs: QUANT, LEVELS, IQUANT

QUANT and IQUANT are the quantizer and inverse quantizer subroutines used by CODN, DECN, CODLMS, DECLMS, CODLS, and DECLS. LEVELS is called by QUANT to help determine level assignments for different bit allocations.

QUANT converts a sample amplitude to a level number. It first checks to see if the bit allocation is zero. If so, then no further calculation is necessary. If not, then the input is quantized to a 5-bit level as a preliminary result. LEVELS is then called to convert the 5-bit level to the appropriate n-bit level (n = 1, 2, 3, 4, or 5). Finally, the step size is updated according to the level of the output, making sure that $\Delta$ always stays between $\Delta_{min}$ and $\Delta_{max}$, inclusive.

LEVELS utilizes an 8x4 matrix, QTABLE, to store the level subset information. QTABLE(i,j) represents the 5-bit level corresponding to the ith j-bit quantizer level. If the j-bit quantizer does not have the ith value then the table entry is set to 17, which is out of the 5-bit range. This is to insure that the correct n-bit level is selected by branching on QTABLE entries (see code). LEVELS also checks that n-bit levels lie between $-2^{n-1}$ and $2^{n-1}$, inclusive. For example, a 5-bit value of 16 should be clipped to 4 in a 3-bit quantizer.

IQUANT reverses the quantization process by converting a level number to an actual sample amplitude by scaling with the step size. Since different bit counts require somewhat different scaling, IQUANT first translates all levels to the corresponding 5-bit level (using QTABLE) as an intermediate step. This result can then be scaled independently of the actual number of bits. Just as QUANT updates STEP, IQUANT updates ISTEP. Although STEP and ISTEP should be equal after each sample is processed, they are separately defined and handled to simulate actual coder operation.

## B.5 ADAPTIVE PREDICTORS

Programs: PRDLMS, PRDLS

### B.5.1 LMS Transversal

PRDLMS maintains 13 by 30 ar ays to hold the predictor coefficients and tapped delay line values for up to 13 subbands with LMS prediction of up to 30 orders each. The implementation of this predictor is straightforward.

First the current signal power approximation is updated. Because of the exponentially decaying nature of estimate, PWRQS can be recursively computed as

(B-7) $\langle x_q^2(n)\rangle_{T+1} = \alpha\langle x_q^2(n)\rangle_T + (1-\alpha)\{x_q^2(T+1)\}$

where $\langle x_q^2(n)\rangle_T$ is defined in equation (2-26).

This estimate can then be plugged into the coefficient update formula (2-27) to obtain the new tap weights. Finally, the latest sample is shifted in and used to calculate the prediction of the next input.

It should be noted that the LMS coefficients are initialized to zero before the first block of voice samples is processed, thereby forcing the predictor output to start at zero.

### B.5.2 LS Lattice

The implementation of this predictor is complicated by the nature of the recursion equations. Since both order updates and time updates are necessary, and many cross dependencies exist, some care must be exercised in their coding.

Fig. B-4 illustrates the interdependencies of the lattice variables in the time and order dimensions.

From this, a consistent variable update order may be arrived at. A valid one is

| | | | | | | |
|---|---|---|---|---|---|---|
| | $C_{m+1}(n)$ → | $\gamma_m(n)$ → | $e_{m+1}(n)$ → | $r_{m+1}(n)$ → | $R^e_{m+1}(n)$ → | $R^r_{m+1}(n)$ |
| FORTRAN name : | PCOR | G | E | R | VARE | VARR |

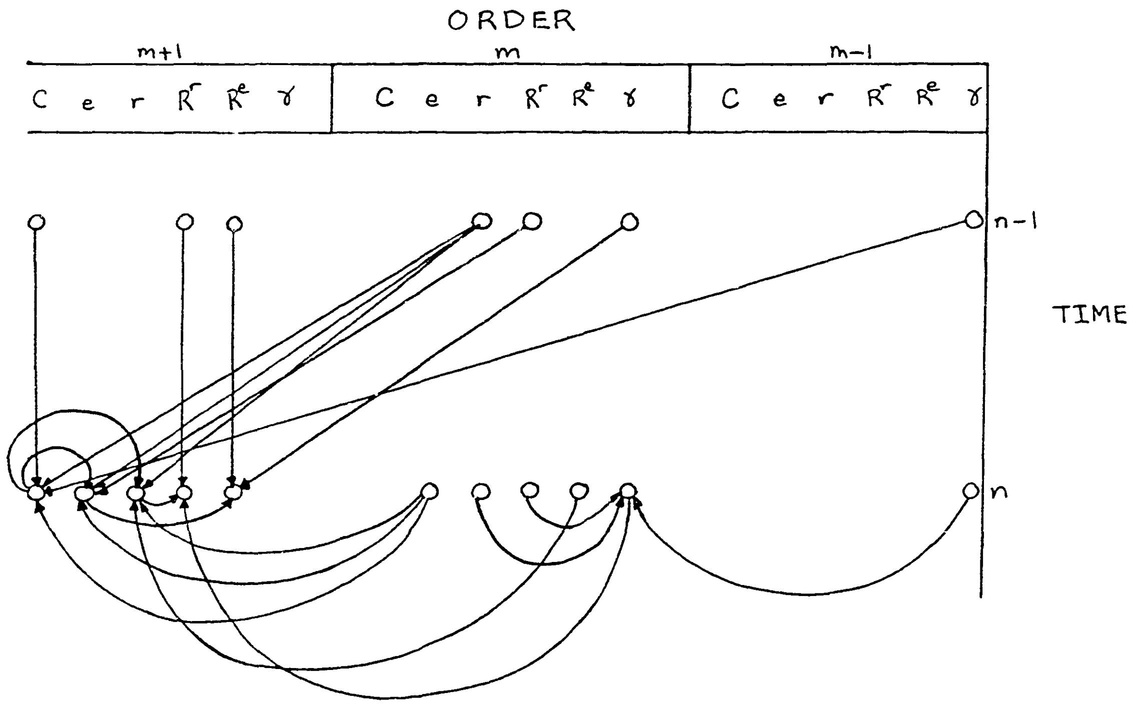


Figure B-4. LS Lattice Variable Dependencies

Notice that values of r, $R_m^r$, and $\gamma$ at time n-1 are also needed for the update of other variables. In order to avoid writing over these past values, additional arrays RM1, VARRM1, and GM1 are kept which hold delayed versions of R, VARR, and G, respectively. The order of variable update then becomes

$$\text{PCOR} \rightarrow \text{GM1} \rightarrow \text{G} \rightarrow \text{E} \rightarrow \text{RM1} \rightarrow \text{R} \rightarrow \text{VARE} \rightarrow \text{VARRM1} \rightarrow \text{VARR}$$

PRDLS has three sections. The first performs the initialization of some of the lattice variables. The second goes through the heart of the least-squares recursions in the order just developed. To avoid division by zero, the likelihood variable, $\gamma$, is not allowed to equal 1. The third and final section computes the prediction of the next input.

## B.6 UTILITIES

### B.6.1 Parallel QMF Bank Generator

Programs: FQMFBANK

This BASIC routine takes as input up to five lowpass filters representing the half band filters of a tree QMF. These impulse responses are then normalized in frequency by interspersing $2^{i-1}-1$ zeroes between every two coefficients of the ith stage. Finally, the normalized impulse responses are convolved to get the coefficients of the corresponding parallel QMF bandpass filters. In addition, the highpass filters of each stage are generated from the lowpass inputs (they are mirror images reflected about $\pi/2$ in the frequency domain), and the composite response

$$\text{(B-8)} \qquad h(n) = \sum_{k=0}^{15} f_k(n) * g_k(n)$$

is determined, as well. For h(n), the program does not compute (B-8) but an equivalent expression derivable from QMF symmetry properties.

$$\text{(B-9)} \qquad h(n) = \begin{cases} 0 & \text{for even } n \\ 2\sum_{k=0}^{7} (-1)^k \left[\sum_{t=0}^{L-1} f_k(t) f_k(n-t)\right] & \text{for odd } n \end{cases}$$

where L is the filter length

Each of the analysis filters, $f_k(n)$, is formed by choosing a lowpass or highpass filter from each stage of the associated tree structure. For example, the first bandpass of a four-stage QMF, $f_0(n)$, is the convolution of the four (frequency normalized) lowpass filters from each stage. If we label the choice of lowpass or highpass with 0 and 1, respectively, we can say $f_0(n)$ is derived from 0000 (stage 1 choice, ... , stage 4 choice). Similarly, the second bandpass filter, $f_1(n)$, is obtained from 0001. It is tempting to conclude from these examples that $f_k(n)$ is derived from the base two representation of k. However, this is not the case.

Fig. B-5 shows the positions of the (normalized) frequency responses of the lowpass and highpass filters of a 4-stage QMF (0=lowpass, 1=highpass).

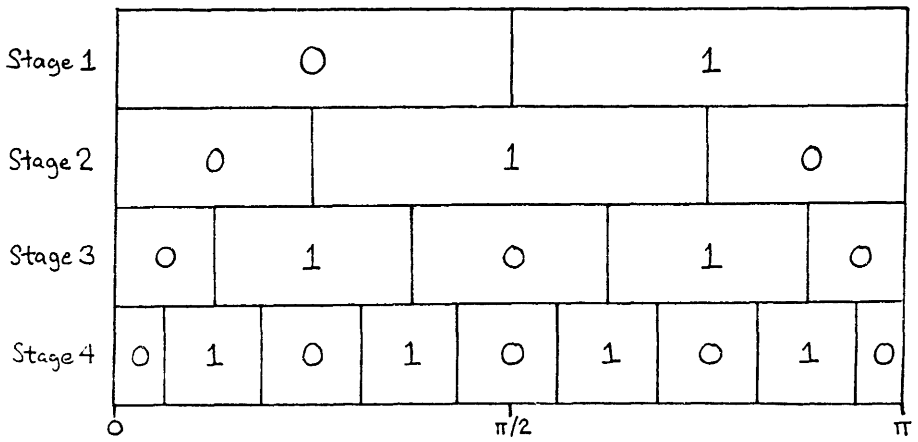


Fig. B-5 Lowpass and Highpass Ranges, 4-Stage QMF

Table B-1 lists the indices, k, of the 16 parallel bandpass filters, their binary representations, and the actual code corresponding to the lowpass/highpass filters from which they were generated.

| Bandpass Index, k | Binary(k) | Code(k) |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

Table B-1. Filter Index Representations

The pattern of code(k) turns out to be a well-known sequence called the Gray code [52]. It has the property of exactly one bit changing between successive entries. The FQMFBANK program determines which filters need to be convolved for bandpass $f_k(n)$ by first converting k to binary and then from binary to Gray code.

### B.6.2 Parallel QMF Truncation

Programs: TRUNCATE

The first step in generating parallel QMF's from a tree structure is to convolve the impulse responses of the filters from each stage. This is handled by FQMFBANK. The next step is to throw away leading and trailing coefficients of the convolution result, which are usually very small. This is done by TRUNCATE.

Bandpass filters are read from the files f*.cof and their shortened versions are written to tf*.cof. After each filter is processed, the composite response of the truncated QMF is updated using equation (B-9). When all input files are completed, the composite response is written to tg.cof.

### B.6.3 Signal-to-Noise Ratio Calculation

Program: SNR

This program computes the signal-to-noise ratio of a voice file, f2, with respect to another voice file, f1. F1 is interpreted as the pure signal component of f2. Any variation from f1 is considered noise. This comes in handy when a voice sample is coded and decoded and some quality measure of the process is desired. By using the original voice sample as f1 and the processed version as f2, the fidelity of algorithm can be judged.

In addition to the names of the two files, SNR needs as input a positive integer representing the delay between the two files. To enable corresponding samples to be compared when nonzero delay values are specified, an additional buffer of 128 samples for f2 must be used. By always having the current and next block of f2 available, the samples in the current block of f1 can be matched with f2 samples at delays up to 128 (Fig. B-6).

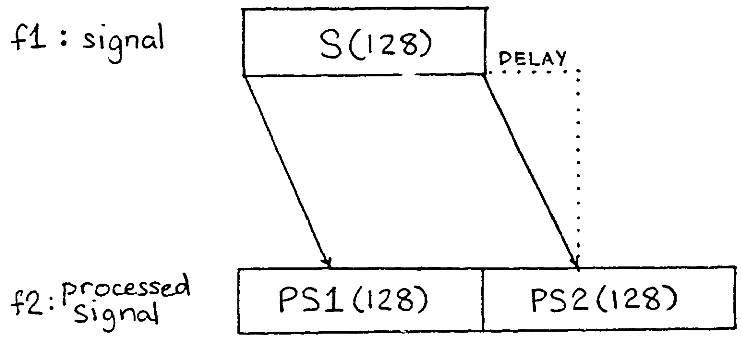


Figure B-6. SNR File Buffers

## B.6.4 Segmental Signal-to-Noise Ratio Calculation

Program: SSNR

The segmental SNR calculation simply averages the SNR (dB) values obtained for each 16ms frame. It is essentially the same program as SNR.

## APPENDIX C - PROGRAM LISTINGS

The following programs are included in this appendix.

### SBC-ADPCM PROGRAMS

| Description | Name | Language |
|---|---|---|
| Main Line | SBC | FORTRAN |
| Subroutines: | | |
| Read block from input file | INBLK | FORTRAN |
| Write block to ouput file | OUTBLK | FORTRAN |
| Code subband (no pred.) | CODN | FORTRAN |
| Decode subband (no pred.) | DECN | FORTRAN |
| Code subband with LMS | CODLMS | FORTRAN |
| Decode subband with LMS | DECLMS | FORTRAN |
| Code subband with LS | CODLS | FORTRAN |
| Decode subband with LS | DECLS | FORTRAN |
| QMF initialization | INIT | FORTRAN |
| QMF filter bank | QMF | FORTRAN |
| Inverse QMF filter bank | IQMF | FORTRAN |
| Characteristic computation | CMPCHR | FORTRAN |
| Dynamic allocation of bits | DAB | FORTRAN |
| Adjust bit alloc. for pred. | PDAB | FORTRAN |
| Adaptive quantizer | QUANT | FORTRAN |
| Level conversion for QUANT | LEVELS | FORTRAN |
| Inverse adaptive quantizer | IQUANT | FORTRAN |
| LMS transversal predictor | PRDLMS | FORTRAN |
| LS lattice predictor | PRDLS | FORTRAN |

### UTILITIES

| Description | Name | Language |
|---|---|---|
| Parallel QMF Bank Generator | FQMFBANK | BASIC |
| Parallel QMF Truncation | TRUNCATE | BASIC |
| SNR computation | SNR | FORTRAN |
| Segmental SNR computation | SSNR | FORTRAN |
| Subroutine: | | |
| INBLK routine for SNR and SSNR | IN | FORTRAN |

```
C xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx C
C                                                                      C
C PROGRAM : SBC                                                        C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : MAIN LINE OF SUB-BAND CODER.  PROCESSES INPUT FILE     C
C               WITH OR WITHOUT PREDICTION AND WRITES RESULT TO        C
C               OUTPUT FILE.  ALSO KEEPS TEST STATISTICS ON THE        C
C               CODER'S PERFORMANCE.                                   C
C                                                                      C
C CALLS : INIT, INBLK, QMF, CMPCHR, DAB, PDAB, CODN, DECN, CODLMS,     C
C         DECLMS, CODLS, DECLS, IQMF, OUTBLK.                          C
C                                                                      C
C KEY VARIABLES : KBPS - CODER BIT RATE IN 1000 BITS/SECOND            C
C                 WHICHP - PREDICTOR SELECTOR                          C
C                 ORDER - ORDER OF PREDICTION                          C
C                 PHIGH - HIGHEST BAND WHICH USES A PREDICTOR          C
C                 INBUF - INPUT BUFFER                                 C
C                 OUTBUF - OUTPUT BUFFER                               C
C                 B - NUMBER OF BITS ALLOCATED                         C
C                 CODDIF - CODED DIFFERENCE SIGNAL                     C
C                 S - SUBBAND SIGNAL                                   C
C                 DS - DECODED SUBBAND SIGNAL                          C
C                 C - SUBBAND CHARACTERISTIC                           C
C                 BAVG - AVERAGE BIT ALLOCATION                        C
C                 CAVG - AVERAGE CHARACTERISTIC                        C
C                 CMAX - MAXIMUM CHARACTERISTIC                        C
C                 CMIN - MINIMUM CHARACTERISTIC                        C
C                 K - SUBBAND NUMBER (1 TO 16)                         C
C                                                                      C
C COMMON LABELS : /BPF/ - INPUT AND OUTPUT BANDPASS FILTERS; USED BY   C
C                         INIT, QMF, IQMF                              C
C                 /CODE/ - CODING BLOCKS; USED BY CODN, CODLS, CODLMS  C
C                 /DECODE/ - DECODING BLOCKS; USED BY DECN, DECLS,     C
C                            DECLMS                                    C
C                 /QNT/ - QUANTIZER STEP SIZE; USED BY QUANT, CODN,    C
C                         CODLS, CODLMS                                C
C                 /IQNT/ - INVERSE QUANTIZER STEP SIZE (CODING         C
C                          BLOCKS); USED BY CODLS, CODLMS              C
C                 /DIQNT/ - INVERSE QUANTIZER STEP SIZE (DECODING      C
C                           BLOCKS); USED BY DECN, DECLS, DECLMS       C
C                 /LMS/ - LMS PREDICTOR VARIABLES (CODING BLOCK);      C
C                         USED BY CODLMS                               C
C                 /DLMS/ - LMS PREDICTOR VARIABLES (DECODING BLOCK);   C
C                          USED BY DECLMS                              C
C                 /LS/ - LS PREDICTOR VARIABLES (CODING BLOCK);        C
C                        USED BY CODLS                                 C
C                 /DLS/ - LS PREDICTOR VARIABLES (DECODING BLOCK);     C
C                         USED BY DECLS                                C
C                 /TEST/ - DIAGNOSTIC VARIABLES FOR FRAME-TO-FRAME     C
C                          AND CUMULATIVE PREDICTION PERFORMANCE       C
```

```
C                          STATISTICS; USED BY CODLS, CODLMS                  C
C                                                                             C
C *************************************************************************** C


C
C DECLARATION OF VARIABLES
C
      INTEGER*2 KBPS, WHICHP, ORDER, K, T, PHIGH
      INTEGER*2 INBUF(128), OUTBUF(128), B(13), CODDIF(13,8)
      REAL S(16,8), DS(16,8), C(16)
      REAL BAVG(13), CAVG(13), CMAX(13), CMIN(13)

      DATA INBUF/128*0/, OUTBUF/128*0/, B/13*0/, CODDIF/104*0/
      DATA S/128*0.0/, DS/128*0.0/, C/16*0.0/, PHIGH/4/
      DATA BAVG/13*0.0/, CAVG/13*0.0/, T/0/

      COMMON /BPF/X(72), Y(8,2,5), Q(8,72)
      REAL X, Y, Q

      COMMON /CODE/DIF(13), IQDIF(13), P(13), QS(13), QDIF(13)
      REAL DIF, IQDIF, P, QS
      INTEGER*2 QDIF

      COMMON /DECODE/DIQDIF(13), DP(13), DQS(13), DQDIF(13)
      REAL DIQDIF, DP, DQS
      INTEGER*2 DQDIF

      COMMON /QNT/STEP(13) /IQNT/ISTEP(13) /DIQNT/DISTEP(13)
      REAL STEP, ISTEP, DISTEP

      COMMON /LMS/LMSQS(13,30), A(13,30), PWRQS(13)
      REAL LMSQS, A, PWRQS

      COMMON /DLMS/DLMSQS(13,30), DA(13,30), DPWRQS(13)
      REAL DLMSQS, DA, DPWRQS

      COMMON /LS/PCOR(13,30), VARE(13,30), VARR(13,30), VARRM1(13,30)
      COMMON /LS/G(13,30), GM1(13,30), E(13,30), R(13,30), RM1(13,30)
      COMMON /LS/LSON(13)
      REAL PCOR, VARE, VARR, VARRM1, G, GM1, E, R, RM1
      INTEGER LSON

      COMMON /DLS/DPCOR(13,30), DVARE(13,30), DVARR(13,30)
      COMMON /DLS/DVARR1(13,30), DG(13,30), DGM1(13,30), DE(13,30)
      COMMON /DLS/DR(13,30), DRM1(13,30)
      COMMON /DLS/DLSON(13)
      REAL DPCOR, DVARE, DVARR, DVARR1, DG, DGM1, DE, DR, DRM1
      INTEGER DLSON

      COMMON /TEST/MSN(13), MSS(13), NRMS(13), SRMS(13), N(13)
      COMMON /TEST/MSNF(13), MSSF(13), NRMSF(13), SRMSF(13), NF(13)
      REAL MSN, MSS, NRMS, SRMS
      REAL MSNF, MSSF, NRMSF, SRMSF
      INTEGER N
      INTEGER NF


C
C PROMPT FOR CODER BIT RATE, TYPE, AND PREDICTOR ORDER
C
      PRINT 100
 100  FORMAT(' BIT RATE')
      READ (5,*) KBPS

      PRINT 200
 200  FORMAT(' CHOICE OF PREDICTOR (0=NONE,1=LMS,2=LS)')
      READ (5,*) WHICHP

      IF (WHICHP.EQ.0) GO TO 5
      PRINT 300
 300  FORMAT (' ORDER OF PREDICTOR')
      READ (5,*) ORDER


C
C INITIALIZATION OF QMF COEFFICIENTS
C
 5    CALL INIT


C
C MAIN LOOP - PROCESS VOICE FILE BLOCK BY BLOCK
C

C READ 128 INPUT SAMPLES FROM FILE NUMBER 3
 10   CONTINUE
      CALL INBLK(INBUF,ISTAT)
      IF (ISTAT.EQ.1) GO TO 1000

C DIVIDE INPUT SIGNAL INTO SUBBAND SIGNALS
      CALL QMF(INBUF,S)

C COMPUTE MAXIMUM AMPLITUDE CHARACTERISTIC FOR EACH BAND
      CALL CMPCHR(S,C)

C ALLOCATE BIT RESOURCES
      CALL DAB(C,KBPS,B)
      IF (WHICHP.EQ.0) GO TO 12
      CALL PDAB(B,PHIGH)

C UPDATE DIAGNOSTIC VARIABLES
 12   DO 16 I = 1,13
      IF (T.EQ.0) GO TO 13
      CMIN(I) = AMIN1(C(I),CMIN(I))
      CMAX(I) = AMAX1(C(I),CMAX(I))
      GO TO 14
 13   CMIN(I) = C(I)
```

```
      CMAX(I) = C(I)
 14   CAVG(I) = (T/(T+1.0))*CAVG(I) + (1.0/(T+1.0))*C(I)
 16   BAVG(I) = (T/(T+1.0))*BAVG(I) + (1.0/(T+1.0))*B(I)
      T = T + 1

C CODE AND DECODE EACH SUBBAND SIGNAL
      DO 90 K = 1, 13
      IF ((WHICHP.EQ.2).AND.(K.LE.PHIGH)) GO TO 20
      IF ((WHICHP.EQ.1).AND.(K.LE.PHIGH)) GO TO 30
      CALL CODN(S,K,B,CODDIF)
      CALL DECN(CODDIF,B,K,DS)
      GO TO 90
 20   CALL CODLS(S,K,B,CODDIF,ORDER)
      CALL DECLS(CODDIF,B,K,DS,ORDER)
      GO TO 90
 30   CALL CODLMS(S,K,B,CODDIF,ORDER)
      CALL DECLMS(CODDIF,B,K,DS,ORDER)
 90   CONTINUE

C RECOMBINE DECODED SUBBAND SIGNALS
      CALL IQMF(DS,OUTBUF)

C WRITE 128 PROCESSED VOICE SAMPLES TO FILE NUMBER 9
      CALL OUTBLK(OUTBUF,ISTAT)
      IF (ISTAT.EQ.1) GO TO 1000

C SIGNAL COMPLETION OF ONE BLOCK
      PRINT 400
 400  FORMAT (' .')
      GO TO 10


C
C PRINT DIAGNOSTICS
C
 1000 DO 17 I = 1,13
 17   PRINT *, I, CMIN(I), CAVG(I), CMAX(I), BAVG(I)
      DO 18 L = 1,PHIGH
 18   PRINT *, L, S(L,8), P(L), NRMS(L), SRMS(L)


      STOP
      END


C
C INITIAL VALUES FOR LABELLED COMMON VARIABLES
C
      BLOCK DATA

      COMMON /BPF/X(72), Y(8,2,5), Q(8,72)
      REAL X/72*0.0/, Y/80*0.0/

      COMMON /CODE/DIF(13), IQDIF(13), P(13), QS(13), QDIF(13)
```

```
REAL DIF/13*0.0/, IQDIF/13*0.0/, P/13*0.0/, QS/13*0.0/
INTEGER*2 QDIF/13*0/

COMMON /DECODE/DIQDIF(13), DP(13), DQS(13), DQDIF(13)
REAL DIQDIF/13*0.0/, DP/13*0.0/, DQS/13*0.0/
INTEGER*2 DQDIF/13*0/

COMMON /QNT/STEP(13) /IQNT/ISTEP(13) /DIQNT/DISTEP(13)
REAL STEP/13*0.1/, ISTEP/13*0.1/, DISTEP/13*0.1/

COMMON /LMS/LMSQS(13,30), A(13,30), PWRQS(13)
REAL LMSQS/390*0.0/, A/390*0.0/, PWRQS/13*0.0/

COMMON /DLMS/DLMSQS(13,30), DA(13,30), DPWRQS(13)
REAL DLMSQS/390*0.0/, DA/390*0.0/, DPWRQS/13*0.0/

COMMON /LS/PCOR(13,30), VARE(13,30), VARR(13,30), VARRM1(13,30)
COMMON /LS/G(13,30), GM1(13,30), E(13,30), R(13,30), RM1(13,30)
COMMON /LS/LSON(13)
REAL PCOR/390*0.0/, VARE/390*1.0/, VARR/390*1.0/
REAL VARRM1/390*1.0/, G/390*0.0/
REAL GM1/390*0.0/, E/390*0.0/, R/390*0.0/, RM1/390*0.0/
INTEGER LSON/13*0/

COMMON /DLS/DPCOR(13,30), DVARE(13,30), DVARR(13,30)
COMMON /DLS/DVARR1(13,30), DG(13,30), DGM1(13,30), DE(13,30)
COMMON /DLS/DR(13,30), DRM1(13,30)
COMMON /DLS/DLSON(13)
REAL DPCOR/390*0.0/, DVARE/390*1.0/, DVARR/390*1.0/
REAL DVARR1/390*1.0/, DG/390*0.0/
REAL DGM1/390*0.0/, DE/390*0.0/, DR/390*0.0/, DRM1/390*0.0/
INTEGER DLSON/13*0/

COMMON /TEST/MSN(13), MSS(13), NRMS(13), SRMS(13), N(13)
COMMON /TEST/MSNF(13), MSSF(13), NRMSF(13), SRMSF(13), NF(13)
REAL MSN/13*0./, MSS/13*0./, NRMS/13*0./, SRMS/13*0./
REAL MSNF/13*0./, MSSF/13*0./, NRMSF/13*0./, SRMSF/13*0./
INTEGER N/13*0/
INTEGER NF/13*0/


END
```

```
C ********************************************************************** C
C                                                                        C
C PROGRAM : INBLK                                                        C
C                                                                        C
C BY : DAN LAI                                                           C
C                                                                        C
C DATE : 7/24/87                                                         C
C                                                                        C
C DESCRIPTION : SUBROUTINE TO READ 128 SAMPLES FROM INPUT FILE           C
C                                                                        C
C CALLED BY : SBC                                                        C
C                                                                        C
C KEY VARIABLES : INBUF - INPUT BUFFER                                   C
C                 ISTAT - I/O STATUS                                     C
C                                                                        C
C ********************************************************************** C

      SUBROUTINE INBLK(INBUF,ISTAT)

      INTEGER*2 INBUF(128)

      READ (8,100,END=200,ERR=200) (INBUF(I),I=1,128)
      ISTAT = 0
      RETURN
100   FORMAT(128(Z4))
200   ISTAT = 1

      RETURN
      END
```

```
C ************************************************************ C
C                                                              C
C PROGRAM : OUTBLK                                             C
C                                                              C
C BY : DAN LAI                                                 C
C                                                              C
C DATE : 7/24/87                                               C
C                                                              C
C DESCRIPTION : SUBROUTINE TO WRITE 128 SAMPLES TO OUTPUT FILE C
C                                                              C
C CALLED BY : SBC                                              C
C                                                              C
C KEY VARIABLES : OUTBUF - OUTPUT BUFFER                       C
C                 ISTAT - I/O STATUS                           C
C                                                              C
C ************************************************************ C

      SUBROUTINE OUTBLK(OUTBUF,ISTAT)

      INTEGER*2 OUTBUF(128)

      WRITE (9,300,ERR=400) (OUTBUF(I),I=1,128)
      ISTAT = 0
      RETURN
300   FORMAT(128(Z4))
400   ISTAT = 1

      RETURN
      END
```

```
C ******************************************************************** C
C                                                                      C
C PROGRAM : CODN                                                       C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : SUBROUTINE TO CODE A BLOCK OF SUBBAND SAMPLES          C
C               WITHOUT PREDICTION                                     C
C                                                                      C
C CALLED BY : SBC                                                      C
C                                                                      C
C CALLS : QUANT                                                        C
C                                                                      C
C COMMON LABELS ACCESSED : /CODE/, /QNT/                               C
C                                                                      C
C KEY VARIABLES : DIF - DIFFERENCE SIGNAL                              C
C                 QDIF - QUANTIZED DIFFERENCE SIGNAL                   C
C                 STEP - QUANTIZER STEP SIZE                           C
C                 S - SUBBAND SIGNAL                                   C
C                 K - SUBBAND NUMBER                                   C
C                 B - NUMBER OF BITS ALLOCATED                         C
C                 CODDIF - CODED DIFFERENCE SIGNAL (ARRAY OF QDIF      C
C                          VALUES)                                     C
C                                                                      C
C ******************************************************************** C

      SUBROUTINE CODN(S,K,B,CODDIF)

      COMMON /CODE/DIF(13), IQDIF(13), P(13), QS(13), QDIF(13)
      REAL DIF, IQDIF, P, QS
      INTEGER*2 QDIF

      COMMON /QNT/STEP(13)
      REAL STEP

      REAL S(16,8)
      INTEGER*2 K, B(13), CODDIF(13,8)

      DO 10 J = 1,8
      DIF(K) = S(K,J)
      CALL QUANT(K,DIF,B,STEP,QDIF)
 10   CODDIF(K,J) = QDIF(K)

      RETURN
      END
```

```
C ******************************************************************* C
C                                                                     C
C PROGRAM : DECN                                                      C
C                                                                     C
C BY : PAUL NING                                                      C
C                                                                     C
C DATE : 7/24/87                                                      C
C                                                                     C
C DESCRIPTION : SUBROUTINE TO DECODE A BLOCK OF SUBBAND SAMPLES       C
C               WITHOUT PREDICTION                                    C
C                                                                     C
C CALLED BY : SBC                                                     C
C                                                                     C
C CALLS : IQUANT                                                      C
C                                                                     C
C COMMON LABELS ACCESSED : /DECODE/, /DIQNT/                          C
C                                                                     C
C KEY VARIABLES : DIQDIF - DECODER'S INVERSE QUANTIZED DIFFERENCE     C
C                          SIGNAL                                     C
C                 DQDIF - DECODER'S QUANTIZED DIFFERENCE SIGNAL       C
C                 DISTEP - DECODER'S INVERSE QUANTIZER STEP SIZE      C
C                 CODDIF - CODED DIFFERENCE SIGNAL (ARRAY OF DQDIF    C
C                          VALUES)                                    C
C                 B - NUMBER OF BITS ALLOCATED                        C
C                 K - SUBBAND NUMBER                                  C
C                 DS - DECODED SUBBAND SIGNAL (ARRAY OF DIQDIF VALUES) C
C                                                                     C
C ******************************************************************* C

      SUBROUTINE DECN(CODDIF,B,K,DS)

      COMMON /DECODE/DIQDIF(13), DP(13), DQS(13), DQDIF(13)
      REAL DIQDIF, DP, DQS
      INTEGER*2 DQDIF

      COMMON /DIQNT/DISTEP(13)
      REAL DISTEP

      INTEGER*2 CODDIF(13,8), B(13), K
      REAL DS(16,8)

      DO 10 J = 1,8
      DQDIF(K) = CODDIF(K,J)
      CALL IQUANT(K,DQDIF,B,DISTEP,DIQDIF)
10    DS(K,J) = DIQDIF(K)

      RETURN
      END
```

```
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX C
C                                                                         C
C PROGRAM : CODLMS                                                        C
C                                                                         C
C BY : PAUL NING                                                          C
C                                                                         C
C DATE : 7/24/87                                                          C
C                                                                         C
C DESCRIPTION : SUBROUTINE TO CODE A BLOCK OF SUBBAND SAMPLES             C
C               WITH LMS PREDICTION.  ALSO MAINTAINS PREDICTOR            C
C               PERFORMANCE DIAGNOSTICS.                                  C
C                                                                         C
C CALLED BY : SBC                                                         C
C                                                                         C
C CALLS : QUANT, IQUANT, PRDLMS                                           C
C                                                                         C
C COMMON LABELS ACCESSED : /CODE/, /QNT/, /IQNT/, /LMS/, /TEST/           C
C                                                                         C
C KEY VARIABLES : DIF - DIFFERENCE SIGNAL                                 C
C                 IQDIF - INVERSE QUANTIZED DIFFERENCE SIGNAL             C
C                 P - PREDICTOR OUTPUT                                    C
C                 QS - SUM OF PREDICTOR OUTPUT AND INVERSE                C
C                      QUANTIZED DIFFERENCE (ERROR) SIGNAL                C
C                 QDIF - QUANTIZED DIFFERENCE SIGNAL                      C
C                 STEP - QUANTIZER STEP SIZE                              C
C                 ISTEP - INVERSE QUANTIZER STEP SIZE                     C
C                 LMSQS - TAPPED DELAY LINE VALUES OF QS                  C
C                 A - LMS PREDICTOR COEFFICIENTS                          C
C                 PWRQS - POWER ESTIMATE OF QS                            C
C                 S - SUBBAND SIGNAL                                      C
C                 K - SUBBAND NUMBER                                      C
C                 B - NUMBER OF BITS ALLOCATED                            C
C                 CODDIF - CODED DIFFERENCE SIGNAL (ARRAY OF QDIF         C
C                          VALUES)                                        C
C                 ORDER - ORDER OF PREDICTION                             C
C                 MSN - MEAN SQUARE VALUE OF NOISE (PREDICTION ERROR)     C
C                 MSS - MEAN SQUARE VALUE OF SUBBAND SIGNAL               C
C                 NRMS - SQUARE ROOT OF MSN                               C
C                 SRMS - SQUARE ROOT OF MSS                               C
C                 N - NUMBER OF SUBBAND SAMPLES SINCE START OF FILE       C
C                 MSNF - MEAN SQUARE VALUE OF NOISE DURING FRAME OF       C
C                        32 SUBBAND SAMPLES (4 INPUT BLOCKS)              C
C                 MSSF - MEAN SQUARE VALUE OF SUBBAND SIGNAL DURING       C
C                        FRAME OF 32 SAMPLES                              C
C                 NRMSF - SQUARE ROOT OF MSNF                             C
C                 SRMSF - SQUARE ROOT OF MSSF                             C
C                 NF - NUMBER OF SUBBAND SAMPLES SINCE START OF           C
C                      CURRENT FRAME                                      C
C                 FILNOF - FILE NUMBER TO WHICH FRAME-TO-FRAME            C
C                          DIAGNOSTICS ARE RECORDED                       C
C                                                                         C
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX C
```

```
      SUBROUTINE CODLMS(S,K,B,CODDIF,ORDER)

      COMMON /CODE/DIF(13), IQDIF(13), P(13), QS(13), QDIF(13)
      REAL DIF, IQDIF, P, QS
      INTEGER*2 QDIF

      COMMON /QNT/STEP(13) /IQNT/ISTEP(13)
      REAL STEP, ISTEP

      COMMON /LMS/LMSQS(13,30), A(13,30), PWRQS(13)
      REAL LMSQS, A, PWRQS

      REAL S(16,8)
      INTEGER*2 K, B(13), CODDIF(13,8), ORDER

      COMMON /TEST/MSN(13), MSS(13), NRMS(13), SRMS(13), N(13)
      COMMON /TEST/MSNF(13), MSSF(13), NRMSF(13), SRMSF(13), NF(13)
      REAL MSN, MSS, NRMS, SRMS
      REAL MSNF, MSSF, NRMSF, SRMSF
      INTEGER N
      INTEGER NF, FILNOF


C
C PROCESS 8 SUBBAND SAMPLES
C
      DO 10 J = 1,8
      DIF(K) = S(K,J) - P(K)


C
C UPDATE PREDICTION DIAGNOSTICS
C
      MSN(K) = (N(K)/(N(K)+1.0))*MSN(K) + (1.0/(N(K)+1.0))*DIF(K)**2
      MSS(K) = (N(K)/(N(K)+1.0))*MSS(K) + (1.0/(N(K)+1.0))*S(K,J)**2
      NRMS(K) = SQRT(MSN(K))
      SRMS(K) = SQRT(MSS(K))
      N(K) = N(K) + 1


C
C UPDATE FRAME-TO-FRAME PREDICTION DIAGNOSTICS
C
      MSNF(K) = (NF(K)/(NF(K)+1.0))*MSNF(K) +
     X          (1.0/(NF(K)+1.0))*DIF(K)**2
      MSSF(K) = (NF(K)/(NF(K)+1.0))*MSSF(K) +
     X          (1.0/(NF(K)+1.0))*S(K,J)**2
      NRMSF(K) = SQRT(MSNF(K))
      SRMSF(K) = SQRT(MSSF(K))
      NF(K) = NF(K) + 1
      IF (NF(K).LT.32) GO TO 20
      NF(K) = 0
      MSNF(K) = 0.0
      MSSF(K) = 0.0
      FILNOF = 10 + K
      WRITE (FILNOF,*) NRMSF(K), SRMSF(K)


C
C CODE A SAMPLE
C
 20   CALL QUANT(K,DIF,B,STEP,QDIF)
      CODDIF(K,J) = QDIF(K)
      CALL IQUANT(K,QDIF,B,ISTEP,IQDIF)
      QS(K) = IQDIF(K) + P(K)
 10   CALL PRDLMS(QS,LMSQS,A,PWRQS,K,ORDER,IQDIF,P)


      RETURN
      END
```

```
C ****************************************************************** C
C                                                                    C
C PROGRAM : DECLMS                                                   C
C                                                                    C
C BY : PAUL NING                                                     C
C                                                                    C
C DATE : 7/24/87                                                     C
C                                                                    C
C DESCRIPTION : SUBROUTINE TO DECODE A BLOCK OF SUBBAND SAMPLES      C
C               WITH LMS PREDICTION                                  C
C                                                                    C
C CALLED BY : SBC                                                    C
C                                                                    C
C CALLS : IQUANT, PRDLMS                                             C
C                                                                    C
C COMMON LABELS ACCESSED : /DECODE/, /DIQNT/, /DLMS/                 C
C                                                                    C
C KEY VARIABLES : DIQDIF - DECODER'S INVERSE QUANTIZED DIFFERENCE    C
C                          SIGNAL                                    C
C                 DP - DECODER'S PREDICTOR OUTPUT                    C
C                 DQS - DECODER'S SUM OF PREDICTOR OUTPUT AND INVERSE C
C                       QUANTIZED DIFFERENCE SIGNAL                  C
C                 DQDIF - DECODER'S QUANTIZED DIFFERENCE SIGNAL      C
C                 DISTEP - DECODER'S INVERSE QUANTIZER STEP SIZE     C
C                 DLMSQS - TAPPED DELAY LINE VALUES OF DQS           C
C                 DA - DECODER'S LMS PREDICTOR COEFFICIENTS          C
C                 DPWRQS - POWER ESTIMATE OF DQS                     C
C                 CODDIF - CODED DIFFERENCE SIGNAL (ARRAY OF DQDIF   C
C                          VALUES)                                   C
C                 B - NUMBER OF BITS ALLOCATED                       C
C                 K - SUBBAND NUMBER                                 C
C                 ORDER - ORDER OF PREDICTION                        C
C                 DS - DECODED SUBBAND SIGNAL (ARRAY OF DQS VALUES)  C
C                                                                    C
C ****************************************************************** C

      SUBROUTINE DECLMS(CODDIF,B,K,DS,ORDER)

      COMMON /DECODE/DIQDIF(13), DP(13), DQS(13), DQDIF(13)
      REAL DIQDIF, DP, DQS
      INTEGER*2 DQDIF

      COMMON /DIQNT/DISTEP(13)
      REAL DISTEP

      COMMON /DLMS/DLMSQS(13,30), DA(13,30), DPWRQS(13)
      REAL DLMSQS, DA, DPWRQS

      INTEGER*2 CODDIF(13,8), B(13), K, ORDER
      REAL DS(16,8)

C
```

```
C DECODE 8 SUBBAND SAMPLES
C
      DO 10 J = 1,8
      DQDIF(K) = CODDIF(K,J)
      CALL IQUANT(K,DQDIF,B,DISTEP,DIQDIF)
      DQS(K) = DIQDIF(K) + DP(K)
      DS(K,J) = DQS(K)
 10   CALL PRDLMS(DQS,DLMSQS,DA,DPWRQS,K,ORDER,DIQDIF,DP)


      RETURN
      END
```

```
C xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx C
C                                                                         C
C PROGRAM : CODLS                                                         C
C                                                                         C
C BY : PAUL NING                                                          C
C                                                                         C
C DATE : 7/24/87                                                          C
C                                                                         C
C DESCRIPTION : SUBROUTINE TO CODE A BLOCK OF SUBBAND SAMPLES             C
C               WITH LS PREDICTION.  ALSO MAINTAINS PREDICTOR             C
C               PERFORMANCE DIAGNOSTICS.                                  C
C                                                                         C
C CALLED BY : SBC                                                         C
C                                                                         C
C CALLS : QUANT, IQUANT, PRDLS                                            C
C                                                                         C
C COMMON LABELS ACCESSED : /CODE/, /QNT/, /IQNT/, /LS/, /TEST/            C
C                                                                         C
C KEY VARIABLES : DIF - DIFFERENCE SIGNAL                                 C
C                 IQDIF - INVERSE QUANTIZED DIFFERENCE SIGNAL             C
C                 P - PREDICTOR OUTPUT                                    C
C                 QS - SUM OF PREDICTOR OUTPUT AND INVERSE                C
C                      QUANTIZED DIFFERENCE (ERROR) SIGNAL                C
C                 QDIF - QUANTIZED DIFFERENCE SIGNAL                      C
C                 STEP - QUANTIZER STEP SIZE                              C
C                 ISTEP - INVERSE QUANTIZER STEP SIZE                     C
C                 PCOR - PARTIAL CORRELATION                              C
C                 VARE - VARIANCE OF FORWARD PREDICTION ERROR             C
C                 VARR - VARIANCE OF BACKWARD PREDICTION ERROR            C
C                 VARRM1 - ONE SAMPLE DELAY OF VARR                       C
C                 G - LIKELIHOOD VARIABLE                                 C
C                 GM1 - ONE SAMPLE DELAY OF G                             C
C                 E - FORWARD PREDICTION ERROR                            C
C                 R - BACKWARD PREDICTION ERROR                           C
C                 RM1 - ONE SAMPLE DELAY OF R                             C
C                 LSON - ENABLE FOR LS OUTPUT                             C
C                 S - SUBBAND SIGNAL                                      C
C                 K - SUBBAND NUMBER                                      C
C                 B - NUMBER OF BITS ALLOCATED                            C
C                 CODDIF - CODED DIFFERENCE SIGNAL (ARRAY OF QDIF         C
C                          VALUES)                                        C
C                 ORDER - ORDER OF PREDICTION                             C
C                 MSN - MEAN SQUARE VALUE OF NOISE (PREDICTION ERROR)     C
C                 MSS - MEAN SQUARE VALUE OF SUBBAND SIGNAL               C
C                 NRMS - SQUARE ROOT OF MSN                               C
C                 SRMS - SQUARE ROOT OF MSS                               C
C                 N - NUMBER OF SUBBAND SAMPLES SINCE START OF FILE       C
C                 MSNF - MEAN SQUARE VALUE OF NOISE DURING FRAME OF       C
C                        32 SUBBAND SAMPLES (4 INPUT BLOCKS)              C
C                 MSSF - MEAN SQUARE VALUE OF SUBBAND SIGNAL DURING       C
C                        FRAME OF 32 SAMPLES                              C
C                 NRMSF - SQUARE ROOT OF MSNF                             C
C                 SRMSF - SQUARE ROOT OF MSSF                             C
C                 NF - NUMBER OF SUBBAND SAMPLES SINCE START OF           C
```

```
C                         CURRENT FRAME                                     C
C                 FILNOF - FILE NUMBER TO WHICH FRAME-TO-FRAME              C
C                         DIAGNOSTICS ARE RECORDED                          C
C                                                                           C
C xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx C


      SUBROUTINE CODLS(S,K,B,CODDIF,ORDER)

      COMMON /CODE/DIF(13), IQDIF(13), P(13), QS(13), QDIF(13)
      REAL DIF, IQDIF, P, QS
      INTEGER*2 QDIF

      COMMON /QNT/STEP(13) /IQNT/ISTEP(13)
      REAL STEP, ISTEP

      COMMON /LS/PCOR(13,30), VARE(13,30), VARR(13,30), VARRM1(13,30)
      COMMON /LS/G(13,30), GM1(13,30), E(13,30), R(13,30), RM1(13,30)
      COMMON /LS/LSON(13)
      REAL PCOR, VARE, VARR, VARRM1, G, GM1, E, R, RM1
      INTEGER LSON

      REAL S(16,8)
      INTEGER*2 K, B(13), CODDIF(13,8), ORDER

      COMMON /TEST/MSN(13), MSS(13), NRMS(13), SRMS(13), N(13)
      COMMON /TEST/MSNF(13), MSSF(13), NRMSF(13), SRMSF(13), NF(13)
      REAL MSN, MSS, NRMS, SRMS
      REAL MSNF, MSSF, NRMSF, SRMSF
      INTEGER N
      INTEGER NF, FILNOF


C
C PROCESS 8 SUBBAND SAMPLES; PREDICTOR OUTPUT ENABLED FOR LSON = 1
C
      DO 10 J = 1,8
      IF (LSON(K).EQ.0) P(K) = 0
      DIF(K) = S(K,J) - P(K)


C
C UPDATE PREDICTION DIAGNOSTICS
C
      MSN(K) = (N(K)/(N(K)+1.0))*MSN(K) + (1.0/(N(K)+1.0))*DIF(K)**2
      MSS(K) = (N(K)/(N(K)+1.0))*MSS(K) + (1.0/(N(K)+1.0))*S(K,J)**2
      NRMS(K) = SQRT(MSN(K))
      SRMS(K) = SQRT(MSS(K))
      N(K) = N(K) + 1


C
C UPDATE FRAME-TO-FRAME PREDICTION DIAGNOSTICS
C
```

```
      MSNF(K) = (NF(K)/(NF(K)+1.0))*MSNF(K) +
     X          (1.0/(NF(K)+1.0))*DIF(K)**2
      MSSF(K) = (NF(K)/(NF(K)+1.0))*MSSF(K) +
     X          (1.0/(NF(K)+1.0))*S(K,J)**2
      NRMSF(K) = SQRT(MSNF(K))
      SRMSF(K) = SQRT(MSSF(K))
      NF(K) = NF(K) + 1
      IF (NF(K).LT.32) GO TO 20
      NF(K) = 0
      MSNF(K) = 0.0
      MSSF(K) = 0.0
      FILNOF = 14 + K
      WRITE (FILNOF,*) NRMSF(K), SRMSF(K)


C
C CODE A SAMPLE
C
 20   CALL QUANT(K,DIF,B,STEP,QDIF)
      CODDIF(K,J) = QDIF(K)
      CALL IQUANT(K,QDIF,B,ISTEP,IQDIF)
      QS(K) = IQDIF(K) + P(K)
 10   CALL PRDLS(QS,PCOR,VARE,VARR,VARRM1,G,GM1,E,R,RM1,K,ORDER,P)


C
C ENABLE PREDICTOR OUTPUT AFTER FIRST 8 SUBBAND SAMPLES
C
      LSON(K) = 1


      RETURN
      END
```

```
C ******************************************************************** C
C                                                                      C
C PROGRAM : DECLS                                                      C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : SUBROUTINE TO DECODE A BLOCK OF SUBBAND SAMPLES        C
C               WITH LS PREDICTION                                     C
C                                                                      C
C CALLED BY : SBC                                                      C
C                                                                      C
C CALLS : IQUANT, PRDLS                                                C
C                                                                      C
C COMMON LABELS ACCESSED : /DECODE/, /DIQNT/, /DLS/                    C
C                                                                      C
C KEY VARIABLES : DIQDIF - DECODER'S INVERSE QUANTIZED DIFFERENCE      C
C                          SIGNAL                                      C
C                 DP - DECODER'S PREDICTOR OUTPUT                      C
C                 DQS - DECODER'S SUM OF PREDICTOR OUTPUT AND INVERSE  C
C                       QUANTIZED DIFFERENCE SIGNAL                    C
C                 DQDIF - DECODER'S QUANTIZED DIFFERENCE SIGNAL        C
C                 DISTEP - DECODER'S INVERSE QUANTIZER STEP SIZE       C
C                 DPCOR - PARTIAL CORRELATION (DECODER)                C
C                 DVARE - VARIANCE OF FORWARD PREDICTION ERROR         C
C                         (DECODER)                                    C
C                 DVARR - VARIANCE OF BACKWARD PREDICTION ERROR        C
C                         (DECODER)                                    C
C                 DVARRM1 - ONE SAMPLE DELAY OF DVARR                  C
C                 DG - LIKELIHOOD VARIABLE (DECODER)                   C
C                 DGM1 - ONE SAMPLE DELAY OF DG                        C
C                 DE - FORWARD PREDICTION ERROR (DECODER)              C
C                 DR - BACKWARD PREDICTION ERROR (DECODER)             C
C                 DRM1 - ONE SAMPLE DELAY OF DR                        C
C                 DLSON - ENABLE FOR LS OUTPUT (DECODER)               C
C                 CODDIF - CODED DIFFERENCE SIGNAL (ARRAY OF DQDIF     C
C                          VALUES)                                     C
C                 B - NUMBER OF BITS ALLOCATED                         C
C                 K - SUBBAND NUMBER                                   C
C                 ORDER - ORDER OF PREDICTION                          C
C                 DS - DECODED SUBBAND SIGNAL (ARRAY OF DQS VALUES)    C
C                                                                      C
C ******************************************************************** C


      SUBROUTINE DECLS(CODDIF,B,K,DS,ORDER)

      COMMON /DECODE/DIQDIF(13), DP(13), DQS(13), DQDIF(13)
      REAL DIQDIF, DP, DQS
      INTEGER*2 DQDIF

      COMMON /DIQNT/DISTEP(13)
      REAL DISTEP
      COMMON /DLS/DPCOR(13,30), DVARE(13,30), DVARR(13,30)
      COMMON /DLS/DVARR1(13,30), DG(13,30), DGM1(13,30), DE(13,30)
      COMMON /DLS/DR(13,30), DRM1(13,30)
      COMMON /DLS/DLSON(13)
      REAL DPCOR, DVARE, DVARR, DVARR1, DG, DE, DR, DRM1
      INTEGER DLSON

      INTEGER*2 CODDIF(13,8), B(13), K, ORDER
      REAL DS(16,8)


C
C DECODE 8 SUBBAND SAMPLES
C
      DO 10 J = 1,8
      IF (DLSON(K).EQ.0) DP(K) = 0
      DQDIF(K) = CODDIF(K,J)
      CALL IQUANT(K,DQDIF,B,DISTEP,DIQDIF)
      DQS(K) = DIQDIF(K) + DP(K)
      DS(K,J) = DQS(K)
 10   CALL
     XPRDLS(DQS,DPCOR,DVARE,DVARR,DVARR1,DG,DGM1,DE,DR,DRM1,K,ORDER,DP)


C
C ENABLE PREDICTOR OUTPUT AFTER FIRST 8 SUBBAND SAMPLES
C
      DLSON(K) = 1


      RETURN
      END
```

```
C ******************************************************************** C
C                                                                      C
C PROGRAM : INIT                                                       C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : SUBROUTINE TO READ IN QMF COEFFICIENTS FROM A FILE     C
C                                                                      C
C CALLED BY : SBC                                                      C
C                                                                      C
C COMMON LABELS ACCESSED : /BPF/                                       C
C                                                                      C
C KEY VARIABLE : Q - QMF COEFFICIENTS                                  C
C                                                                      C
C ******************************************************************** C

      SUBROUTINE INIT

      COMMON /BPF/X(72), Y(8,2,5), Q(8,72)
      REAL X, Y, Q


C
C READ COEFFICIENTS FROM FILE NUMBER 10
C
      DO 10 I = 1,8
      DO 10 J = 1,72
 10   READ (10,*) Q(I,J)

      RETURN
      END
```

```
C ****************************************************************** C
C                                                                    C
C PROGRAM : QMF                                                      C
C                                                                    C
C BY : PAUL NING                                                     C
C                                                                    C
C DATE : 7/24/87                                                     C
C                                                                    C
C DESCRIPTION : SUBROUTINE TO PRODUCE 16 SUBBAND SIGNALS FROM        C
C               A BLOCK OF INPUT SAMPLES                             C
C                                                                    C
C CALLED BY : SBC                                                    C
C                                                                    C
C COMMON LABELS ACCESSED : /BPF/                                     C
C                                                                    C
C KEY VARIABLES : X - ANALYSIS QMF TAPPED DELAY LINE                 C
C                 Q - QMF COEFFICIENTS                               C
C                 INBUF - INPUT BUFFER                               C
C                 S - SUBBAND SIGNAL                                 C
C                                                                    C
C ****************************************************************** C


      SUBROUTINE QMF(INBUF,S)

      COMMON /BPF/X(72), Y(8,2,5), Q(8,72)
      REAL X, Y, Q

      INTEGER*2 INBUF(128), T
      REAL S(16,8), TERM


C
C GENERATE 8 SAMPLES FOR EACH SUBBAND FROM THE 128 INPUTS
C
      DO 30 I = 1,8


C
C        SHIFT IN 16 SAMPLES FROM INBUF
C
         DO 20 J = 1,16
              DO 10 L = 1,71
 10           X(73-L) = X(73-L-1)
 20      X(1) = INBUF(16*(I-1)+J)


C
C        COMPUTE ONE SUBBAND SAMPLE FOR EACH BAND
C
         DO 30 K = 1,8
         S(K,I) = 0
         S(17-K,I) = 0
              DO 30 T =1,72
```

```
                TERM = Q(K,T)*X(T)
                S(K,I) = S(K,I) + TERM
30              S(17-K,I) = S(17-K,I) + TERM*(-1)**(T-1)

           RETURN
           END
```

```
C ********************************************************************** C
C                                                                        C
C PROGRAM : IQMF                                                         C
C                                                                        C
C BY : PAUL NING                                                         C
C                                                                        C
C DATE : 7/24/87                                                         C
C                                                                        C
C DESCRIPTION : RECONSTRUCT A BLOCK OF OUTPUT SAMPLES FROM SUBBANDS      C
C                                                                        C
C CALLED BY : SBC                                                        C
C                                                                        C
C COMMON LABELS ACCESSED : /BPF/                                         C
C                                                                        C
C KEY VARIABLES : Y - RECONSTRUCTION QMF TAPPED DELAY LINES              C
C                 Q - QMF COEFFICIENTS                                   C
C                 OUTBUF - OUTPUT BUFFER                                 C
C                 DS - DECODED SUBBAND SIGNALS                           C
C                                                                        C
C ********************************************************************** C

      SUBROUTINE IQMF(DS,OUTBUF)

      COMMON /BPF/X(72), Y(8,2,5), Q(8,72)
      REAL X, Y, Q

      INTEGER*2 OUTBUF(128), T
      REAL DS(16,8), TERM, OUT


C
C PROCESS 8 DECODED SAMPLES FROM EACH SUBBAND
C
      DO 40 I = 1,8
C
C        SHIFT IN THE CURRENT SAMPLE SUMS AND DIFFERENCES
C        (K IS THE SUBBAND NUMBER)
C
         DO 30 K = 1,8
C
C              J IS AN INDEX ONTO THE DELAY LINES OF SUMS AND DIFFERENCES
C
               DO 20 J = 1,4
C
C                   L DISTINGUISHES THE TWO DELAY LINES
C
                    DO 20 L = 1,2
 20                 Y(K,L,6-J) = Y(K,L,6-J-1)

               Y(K,1,1) = DS(K,I) + DS(17-K,I)
 30            Y(K,2,1) = DS(K,I) - DS(17-K,I)
```

```
C
C        COMPUTE 16 OUTPUTS FOR CURRENT I VALUE
C
         DO 40 T = 1,16
         TERM = 0
C
C             ADD CONTRIBUTIONS FROM EACH PAIR OF SUBBANDS
C
              DO 10 K = 1,8
C
C                  GO DOWN APPROPRIATE TAPPED DELAY LINE
C
                   DO 10 J = 1,5
                   M = T + 16*(J-1)
C
C                  USE 'SUM' FOR T EVEN, 'DIFFERENCE' FOR T ODD
C
                   IF (M.LE.72)
     XTERM = TERM + (-1)**(K-1)*Q(K,M)*Y(K,1.5+.5*(-1)**(T-1),J)
 10                CONTINUE
C
C        AS PART OF INTERPOLATION, OUTPUT MUST BE SCALED UP BY 16;
C        ALSO CHECK FOR CLIPPING OF INTEGER*2 RANGE
C
         OUT = 16*TERM
         IF (OUT.GT.32767) OUT = 32767
         IF (OUT.LT.(-32768)) OUT = -32768
 40      OUTBUF(16*(I-1) + T) = OUT

      RETURN
      END
```

```
C ****************************************************************** C
C                                                                    C
C PROGRAM : CMPCHR                                                   C
C                                                                    C
C BY : PAUL NING                                                     C
C                                                                    C
C DATE : 7/24/87                                                     C
C                                                                    C
C DESCRIPTION : SUBROUTINE TO COMPUTE SUBBAND CHARACTERISTICS        C
C                                                                    C
C CALLED BY : SBC                                                    C
C                                                                    C
C KEY VARIABLES : S - SUBBAND SIGNAL                                 C
C                 C - SUBBAND CHARACTERISTIC                         C
C                                                                    C
C ****************************************************************** C

      SUBROUTINE CMPCHR(S,C)

      REAL S(16,8), C(16)

      DO 10 I = 1,16
      C(I) = ABS(S(I,1))
      DO 10 J = 2,8
 10   IF (ABS(S(I,J)).GT.C(I)) C(I) = ABS(S(I,J))

      RETURN
      END
```

```
C ************************************************************************ C
C                                                                          C
C PROGRAM : DAB                                                            C
C                                                                          C
C BY : PAUL NING                                                           C
C                                                                          C
C DATE : 7/24/87                                                           C
C                                                                          C
C DESCRIPTION : SUBROUTINE TO ALLOCATE BITS TO 13 SUBBANDS BASED UPON      C
C               THEIR CHARACTERISTICS                                      C
C                                                                          C
C CALLED BY : SBC                                                          C
C                                                                          C
C CALLS : (FUNCTIONS) RDOWN, BLEVEL                                        C
C                                                                          C
C KEY VARIABLES : C - SUBBAND CHARACTERISTIC                               C
C                 KBPS - CODER BIT RATE IN 1000 BITS/SECOND                C
C                 B - NUMBER OF BITS ALLOCATED                             C
C                 DELTA - MINIMUM SCALE ADJUSTMENT NEEDED TO               C
C                         INCREASE/DECREASE BIT ALLOCATION OF A            C
C                         BAND BY 1                                        C
C                 N - VALUES COMPUTED FROM BIT ALLOCATION EQUATION         C
C                 SHIFT - MINIMUM OF THE DELTAS                            C
C                 BITS - NUMBER OF BITS AVAILABLE EVERY SUBBAND            C
C                        SAMPLING INTERVAL                                 C
C                 CZERO - FLAG WHICH INDICATES BANDS WITH C=0              C
C                 TEMPB - TEMPORARY BIT ALLOCATION FOR NONZERO BANDS       C
C                 P - NUMBER OF NONZERO BANDS (CZERO=0)                    C
C                                                                          C
C ************************************************************************ C


      SUBROUTINE DAB(C,KBPS,B)

      REAL C(16)
      INTEGER*2 KBPS, B(13)

      REAL DELTA(13), SUMLOG, N(13), LOGC(13), SHIFT
      INTEGER*2 BITS, CZERO(13), TEMPB(13)
      INTEGER*2 RDOWN, BLEVEL, P, POSFND, NEWB


C
C INITIALIZATION
C
      SUMLOG = 0.0
      SHIFT = 0.0
      P = 13
      DO 170 I = 1,13
 170  CZERO(I) = 0


C
C COMPUTE AVAILABLE BITS PER SUBBAND SAMPLING INTERVAL
```

```
C
      BITS = RDOWN(2*(KBPS-2.4375))


C
C DETECT C(I) = 0 TO AVOID LOG(0) ERRORS;
C ALSO COMPUTE LOGC(I) FOR C(I) NOT ZERO;
C P IS THE NUMBER OF SUBBANDS WITH NONZERO POWER
C
      J = 1
      DO 20 I = 1,13
      IF (C(I).NE.0) GO TO 10
      CZERO(I) = 1
      P = P - 1
      GO TO 20
 10   LOGC(J) = LOG(C(I))/LOG(2.0)
      SUMLOG = SUMLOG + LOGC(J)
      J = J + 1
 20   CONTINUE


C
C IF P IS TOO SMALL TO USE UP BITS, GIVE ALL NONZERO BANDS 5 BITS;
C IF P IS ZERO, NO BITS ARE ALLOCATED
C
      IF (P.GE.(BITS/5.)) GO TO 40
      IF (P.EQ.0) GO TO 5000
      DO 30 J = 1,P
 30   TEMPB(J) = 5
      GO TO 1000


C
C FOR THE P NONZERO BANDS, CALCULATE N(J)
C
 40   DO 50 J = 1,P
 50   N(J) = (1./P)*(2*KBPS - 4.875 - SUMLOG) + LOGC(J)


C
C CALCULATE INITIAL BIT ALLOCATION ESTIMATE, TEMPB(J),
C FOR NONZERO BANDS; UPDATE AVAILABLE BITS
C
      DO 60 J = 1,P
      TEMPB(J) = BLEVEL(N(J))
 60   BITS = BITS - TEMPB(J)


C
C BRANCH BASED UPON HOW MANY BITS ARE LEFT
C
      IF (BITS.EQ.0) GO TO 1000
      IF (BITS.GT.0) GO TO 70
      IF (BITS.LT.0) GO TO 120
```

```
C                   - EXTRA BITS TO ALLOCATE -
C FOR EACH N(J), FIND THE MINIMUM DISPLACEMENT TO GAIN A BIT
C (EXCEPT WHEN N(J)>=5, IN WHICH CASE DEFINE DELTA(J)=0)
C POSFND DETECTS NONZERO DELTA(J)'S
C
 70   POSFND = 0

      DO 100 J = 1,P
      IF (N(J).GE.5) DELTA(J) = 0
      IF (N(J).LT.0) DELTA(J) = -N(J) + 1
      IF ((N(J).LT.5).AND.(N(J).GE.0)) DELTA(J) = RDOWN(N(J)+1) - N(J)
C
C     UPDATE SHIFT = MINIMUM OF THE DELTA(J)'S
C     POSFND IS SET TO 1 WHEN A NONZERO DELTA(J) IS FOUND
C
 80   IF (POSFND.EQ.1) GO TO 90
      IF (DELTA(J).EQ.0) GO TO 100
      SHIFT = DELTA(J)
      POSFND = 1
      GO TO 100
 90   IF (DELTA(J).NE.0) SHIFT = AMIN1(DELTA(J),SHIFT)
 100  CONTINUE

C
C     IF ALL BANDS HAVE MAXIMUM 5 BITS ALREADY, THEN QUIT ADDING BITS
C
      IF (SHIFT.EQ.0) GO TO 1000

C
C     INCREASE N(J) BY AMOUNT OF SHIFT AND ADD BITS BEGINNING AT J=1
C
      DO 110 J = 1,P
      N(J) = N(J) + SHIFT
      NEWB = BLEVEL(N(J))
      IF (NEWB.GT.TEMPB(J)) BITS = BITS - 1
      TEMPB(J) = NEWB
      IF (BITS.EQ.0) GO TO 1000
 110  CONTINUE
      GO TO 70


C                   - TOO MANY BITS ALLOCATED -
C FOR EACH N(J), FIND THE MINIMUM DISPLACEMENT TO LOSE A BIT
C (EXCEPT WHEN N(J)<1, IN WHICH CASE DEFINE DELTA(J)=0)
C POSFND DETECTS NONZERO DELTA(J)'S
C
 120  POSFND = 0

      DO 150 J = 1,P
      IF (N(J).GE.5) DELTA(J) = N(J) - 5 + .001
      IF (N(J).LT.1) DELTA(J) = 0
      IF ((N(J).LT.5).AND.(N(J).GE.1))
```

```
     X     DELTA(J) = N(J) - RDOWN(N(J)) + .001
C
C     UPDATE SHIFT = MINIMUM OF THE DELTA(J)'S
C     POSFND IS SET TO 1 WHEN A NONZERO DELTA(J) IS FOUND
C
 130  IF (POSFND.EQ.1) GO TO 140
      IF (DELTA(J).EQ.0) GO TO 150
      SHIFT = DELTA(J)
      POSFND = 1
      GO TO 150
 140  IF (DELTA(J).NE.0) SHIFT = AMIN1(DELTA(J),SHIFT)
 150  CONTINUE

C
C     IF ALL BANDS HAVE MINIMUM 0 BITS ALREADY, THEN QUIT TAKING BITS
C
      IF (SHIFT.EQ.0) GO TO 1000

C
C     DECREASE N(J) BY SHIFT AND SUBTRACT BITS STARTING AT HIGH BANDS
C
      DO 160 J = 1,P
      K = P + 1 - J
      N(K) = N(K) - SHIFT
      NEWB = BLEVEL(N(K))
      IF (NEWB.LT.TEMPB(K)) BITS = BITS + 1
      TEMPB(K) = NEWB
      IF (BITS.EQ.0) GO TO 1000
 160  CONTINUE
      GO TO 120

C
C ALLOCATION COMPLETE - WRITE TEMPB(J) TO B(I)
C
 1000 J = 1
      DO 2000 I = 1,13
      IF (CZERO(I).EQ.1) GO TO 1500
      B(I) = TEMPB(J)
      J = J + 1
      GO TO 2000
 1500 B(I) = 0
 2000 CONTINUE


 5000 RETURN
      END
```

```
C
C DEFINE FUNCTION TO ROUND DOWN A REAL, R, TO MAX INTEGER I<=R
C
      INTEGER FUNCTION RDOWN*2 (R)
      REAL R
      I = INT(R)
C
C     NOTE THAT HFIX(FLOAT()) CONVERTS INTEGER*4 TO INTEGER*2
C
      IF (R.EQ.I) RDOWN = HFIX(FLOAT(I))
      IF (R.NE.I) RDOWN = HFIX(FLOAT(INT(I - .5 + SIGN(.5,R))))
      RETURN
      END


C
C DEFINE FUNCTION TO FIND BIT LEVEL OF INPUT ON A 0 TO 5 SCALE
C
      INTEGER FUNCTION BLEVEL*2 (R)
      REAL R
      INTEGER*2 RDOWN
      IF (R.GE.5) BLEVEL = 5
      IF (R.LE.0) BLEVEL = 0
      IF ((R.LT.5).AND.(R.GT.0)) BLEVEL = RDOWN(R)
      RETURN
      END
```

```
C ********************************************************************** C
C                                                                         C
C PROGRAM : PDAB                                                          C
C                                                                         C
C BY : PAUL NING                                                          C
C                                                                         C
C DATE : 7/24/87                                                          C
C                                                                         C
C DESCRIPTION : SUBROUTINE TO ADJUST BIT ALLOCATION IN THE PRESENCE      C
C               OF PREDICTION.  A WARNING MESSAGE IS ISSUED IF THIS      C
C               IS NOT POSSIBLE.                                          C
C                                                                         C
C CALLED BY : SBC                                                         C
C                                                                         C
C KEY VARIABLES : B - NUMBER OF BITS ALLOCATED                            C
C                 PHIGH - HIGHEST BAND WHICH USES A PREDICTOR             C
C                 NEXT - FIRST BAND FROM WHICH BITS CAN BE TAKEN          C
C                 BZERO - FLAG TO INDICATE ZERO BIT ALLOCATION            C
C                                                                         C
C ********************************************************************** C


      SUBROUTINE PDAB(B,PHIGH)

      INTEGER*2 B(13), PHIGH
      INTEGER NEXT, BZERO

C
C FOR BANDS WITH PREDICTOR, MAKE SURE BIT ALLOCATION IS AT LEAST 2
C THIS IS NOT POSSIBLE WHEN HIGHER BANDS RUN OUT OF BITS
C
      DO 10 I = 1,PHIGH
      NEXT = I + 1
      IF (NEXT.EQ.14) GO TO 10
      IF (B(I).GE.2) GO TO 10
C
C     ADD BITS TO B(I) UNTIL IT GETS 2, OR BITS RUN OUT AT HIGHER BANDS
C
 60   BZERO = 0
      DO 20 J = NEXT,13
      K = 13 + NEXT - J
      IF (B(K).EQ.0) GO TO 50
      B(K) = B(K) - 1
      B(I) = B(I) + 1
      IF (B(I).EQ.2) GO TO 10
      GO TO 20
 50   BZERO = BZERO + 1
 20   CONTINUE
      IF (BZERO.EQ.(13-I)) GO TO 30
      GO TO 60
 10   CONTINUE
      GO TO 40
```

```
C
C ISSUE WARNING MESSAGE
C
 30    PRINT 100
100    FORMAT (' WARNING : PREDICTED BAND WITH LESS THAN 2 BITS')


 40    RETURN
       END
```

```
C ************************************************************************ C
C                                                                          C
C PROGRAM : QUANT                                                          C
C                                                                          C
C BY : PAUL NING                                                           C
C                                                                          C
C DATE : 7/24/87                                                           C
C                                                                          C
C DESCRIPTION : SUBROUTINE TO CONVERT AN INPUT VALUE TO A QUANTIZER        C
C               LEVEL NUMBER.  ALSO UPDATES STEP SIZE.                     C
C                                                                          C
C CALLED BY : CODN, CODLMS, CODLS                                          C
C                                                                          C
C CALLS : (FUNCTION - SEE DAB SUBROUTINE LISTING) RDOWN,                   C
C         (SUBROUTINE) LEVELS                                              C
C                                                                          C
C KEY VARIABLES : DIF - SUBBAND DIFFERENCE SIGNAL                          C
C                 STEP - QUANTIZER STEP SIZE                               C
C                 K - SUBBAND NUMBER                                       C
C                 B - NUMBER OF BITS ALLOCATED                             C
C                 QDIF - QUANTIZED DIFFERENCE SIGNAL                       C
C                 QLEVEL - B-BIT QUANTIZER LEVEL                           C
C                 LEVEL - INITIAL 5-BIT QUANTIZER LEVEL                    C
C                 NLEVEL - NEW 5-BIT LEVEL CORRESPONDING TO QLEVEL         C
C                 M - STEP SIZE MULTIPLIERS                                C
C                 MAX - MAXIMUM STEP SIZE                                  C
C                 MIN - MINIMUM STEP SIZE                                  C
C                                                                          C
C ************************************************************************ C


      SUBROUTINE QUANT(K,DIF,B,STEP,QDIF)

      REAL DIF(13), STEP(13)
      INTEGER*2 K, B(13), QDIF(13)

      INTEGER*2 QLEVEL, LEVEL, NLEVEL, RDOWN
      REAL M(16), MAX, MIN

      DATA M/.87, .88, .90, .93, .96, 1.01, 1.06, 1.12,
     X       1.18, 1.24, 1.31, 1.39, 1.46, 1.54, 1.62, 1.70/
      DATA MAX/512.0/, MIN/.10/


C
C CHECK FOR ZERO BIT QUANTIZER
C
      IF (B(K).EQ.0) GO TO 10


C
C CONVERT DIF(K) TO 5-BIT, -16 TO 16 SCALE
C
      IF (DIF(K).NE.0)
     X        LEVEL = HFIX(SIGN(RDOWN(ABS(DIF(K))/STEP(K))+1.0,DIF(K)))
      IF (DIF(K).EQ.0) LEVEL = 1
      IF (LEVEL.GT.16) LEVEL = 16
      IF (LEVEL.LT.-16) LEVEL = -16


C
C CONVERT TO APPROPRIATE N-BIT LEVEL
C
      CALL LEVELS(B(K),LEVEL,QLEVEL,NLEVEL)
      QDIF(K) = QLEVEL


C
C UPDATE STEP SIZE
C
      STEP(K) = M(ABS(NLEVEL+0.0))*STEP(K)
      IF (STEP(K).GT.MAX) GO TO 20
      IF (STEP(K).LT.MIN) GO TO 30
      GO TO 10
 20   PRINT *, K
      PRINT 100
 100  FORMAT (' MAXIMUM STEP SIZE REACHED')
      STEP(K) = MAX
      GO TO 10
 30   STEP(K) = MIN


 10   RETURN
      END
```

```
C ********************************************************************* C
C                                                                       C
C PROGRAM : LEVELS                                                      C
C                                                                       C
C BY : PAUL NING                                                        C
C                                                                       C
C DATE : 7/24/87                                                        C
C                                                                       C
C DESCRIPTION : SUBROUTINE TO QUANTIZE 5-BIT LEVEL NUMBER TO            C
C               4-, 3-, 2-, OR 1-BIT LEVEL NUMBERS.                     C
C                                                                       C
C CALLED BY : QUANT                                                     C
C                                                                       C
C KEY VARIABLES : NBIT - NUMBER OF BITS                                 C
C                 LEVEL - INITIAL 5-BIT QUANTIZER LEVEL                 C
C                 QLEVEL - NBIT QUANTIZER LEVEL                         C
C                 NLEVEL - NEW 5-BIT LEVEL CORRESPONDING TO QLEVEL      C
C                 QTABLE - MATRIX WITH LEVEL SUBSET DATA                C
C                                                                       C
C ********************************************************************* C

      SUBROUTINE LEVELS (NBIT,LEVEL,QLEVEL,NLEVEL)

      INTEGER*2 NBIT, LEVEL, QLEVEL, NLEVEL, QTABLE(8,4)

      DATA QTABLE/2,7*17, 3,9,6*17, 2,6,10,14,4*17, 2,4,6,8,10,12,14,16/


C
C CHECK FOR TRIVIAL CASE NBIT = 5
C
      IF (NBIT.NE.5) GO TO 5
      QLEVEL = LEVEL
      NLEVEL = LEVEL
      GO TO 200


C
C BRANCH ON QTABLE ENTRIES
C
 5    IF (ABS(LEVEL+0.0).GT.QTABLE(7,NBIT)) GO TO 10
      IF (ABS(LEVEL+0.0).GT.QTABLE(6,NBIT)) GO TO 20
      IF (ABS(LEVEL+0.0).GT.QTABLE(5,NBIT)) GO TO 30
      IF (ABS(LEVEL+0.0).GT.QTABLE(4,NBIT)) GO TO 40
      IF (ABS(LEVEL+0.0).GT.QTABLE(3,NBIT)) GO TO 50
      IF (ABS(LEVEL+0.0).GT.QTABLE(2,NBIT)) GO TO 60
      IF (ABS(LEVEL+0.0).GT.QTABLE(1,NBIT)) GO TO 70
      QLEVEL = HFIX(SIGN(1.0,LEVEL+0.0))
      NLEVEL = HFIX(SIGN(QTABLE(1,NBIT)+0.0,LEVEL+0.0))
      GO TO 160
 10   QLEVEL = HFIX(SIGN(8.0,LEVEL+0.0))
      NLEVEL = HFIX(SIGN(QTABLE(8,NBIT)+0.0,LEVEL+0.0))
      GO TO 160
 20    QLEVEL = HFIX(SIGN(7.0,LEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(7,NBIT)+0.0,LEVEL+0.0))
       GO TO 160
 30    QLEVEL = HFIX(SIGN(6.0,LEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(6,NBIT)+0.0,LEVEL+0.0))
       GO TO 160
 40    QLEVEL = HFIX(SIGN(5.0,LEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(5,NBIT)+0.0,LEVEL+0.0))
       GO TO 160
 50    QLEVEL = HFIX(SIGN(4.0,LEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(4,NBIT)+0.0,LEVEL+0.0))
       GO TO 160
 60    QLEVEL = HFIX(SIGN(3.0,LEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(3,NBIT)+0.0,LEVEL+0.0))
       GO TO 160
 70    QLEVEL = HFIX(SIGN(2.0,LEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(2,NBIT)+0.0,LEVEL+0.0))
       GO TO 160


C
C CHECK THAT LEVEL ASSIGNMENT IS WITHIN N-BIT RANGE
C
 160   IF (ABS(QLEVEL+0.0).LE.2**(NBIT-1)) GO TO 200
       QLEVEL = HFIX(SIGN(2.0**(NBIT-1),QLEVEL+0.0))
       NLEVEL = HFIX(SIGN(QTABLE(2.0**(NBIT-1),NBIT)+0.0,QLEVEL+0.0))


 200   RETURN
       END
```

```
C ************************************************************************ C
C                                                                          C
C PROGRAM : IQUANT                                                         C
C                                                                          C
C BY : PAUL NING                                                           C
C                                                                          C
C DATE : 7/24/87                                                           C
C                                                                          C
C DESCRIPTION : SUBROUTINE TO CONVERT A LEVEL NUMBER TO A SIGNAL           C
C               AMPLITUDE.  ALSO UPDATES STEP SIZE OF INVERSE              C
C               QUANTIZER.                                                 C
C                                                                          C
C CALLED BY : CODLMS, CODLS, DECN, DECLMS, DECLS                           C
C                                                                          C
C KEY VARIABLES : K - SUBBAND NUMBER                                       C
C                 QDIF - QUANTIZED DIFFERENCE SIGNAL                       C
C                 B - NUMBER OF BITS ALLOCATED                             C
C                 ISTEP - STEP SIZE OF INVERSE QUANTIZER                   C
C                 IQDIF - INVERSE QUANTIZED DIFFERENCE SIGNAL              C
C                 LEVEL - B-BIT LEVEL NUMBER                               C
C                 QTABLE - MATRIX WITH LEVEL SUBSET DATA                   C
C                 M - STEP SIZE MULTIPLIERS                                C
C                 MAX - MAXIMUM STEP SIZE                                  C
C                 MIN - MINIMUM STEP SIZE                                  C
C                                                                          C
C ************************************************************************ C

      SUBROUTINE IQUANT(K,QDIF,B,ISTEP,IQDIF)

      INTEGER*2 K, QDIF(13), B(13)
      REAL ISTEP(13), IQDIF(13)

      INTEGER*2 LEVEL, QTABLE(8,4)
      REAL M(16), MAX, MIN

      DATA M/.87, .88, .90, .93, .96, 1.01, 1.06, 1.12,
     X       1.18, 1.24, 1.31, 1.39, 1.46, 1.54, 1.62, 1.70/
      DATA MAX/512.0/, MIN/.10/

      DATA QTABLE/2,7*17, 3,9,6*17, 2,6,10,14,4*17, 2,4,6,8,10,12,14,16/


C
C CHECK FOR ZERO BIT QUANTIZER
C
      IF (B(K).NE.0) GO TO 5
      IQDIF(K) = 0
      GO TO 30


C
C CONVERT QDIF TO 5-BIT (-16,16) LEVEL
C
```

```
 5    IF (B(K).NE.5) GO TO 10
      LEVEL = QDIF(K)
      GO TO 20
 10   LEVEL = HFIX(SIGN(QTABLE(HFIX(ABS(QDIF(K)+0.0)),B(K))+0.0,
     X                     QDIF(K)+0.0))


C
C SCALE LEVEL BY ISTEP SIZE
C
 20   IQDIF(K) = (LEVEL - SIGN(0.5,LEVEL+0.0)) * ISTEP(K)


C
C UPDATE ISTEP SIZE
C
      ISTEP(K) = M(ABS(LEVEL+0.0)) * ISTEP(K)
      IF (ISTEP(K).GT.MAX) ISTEP(K) = MAX
      IF (ISTEP(K).LT.MIN) ISTEP(K) = MIN


 30   RETURN
      END
```

```
C ****************************************************************** C
C                                                                    C
C PROGRAM : PRDLMS                                                   C
C                                                                    C
C BY : PAUL NING                                                     C
C                                                                    C
C DATE : 7/24/87                                                     C
C                                                                    C
C DESCRIPTION : SUBROUTINE TO COMPUTE SINGLE OUTPUT OF LEAST-MEAN-   C
C               SQUARE TRANSVERSAL PREDICTOR.                        C
C                                                                    C
C CALLED BY : CODLMS, DECLMS                                         C
C                                                                    C
C KEY VARIABLES : LMSQS - TAPPED DELAY LINE VALUES OF QS             C
C                 A - LMS PREDICTOR COEFFICIENTS                     C
C                 PWRQS - POWER ESTIMATE OF QS                       C
C                 QS - SUM OF PREDICTOR OUTPUT AND INVERSE           C
C                      QUANTIZED DIFFERENCE (ERROR) SIGNAL           C
C                 IQDIF - INVERSE QUANTIZED DIFFERENCE SIGNAL        C
C                 P - PREDICTOR OUTPUT                               C
C                 K - SUBBAND NUMBER                                 C
C                 ORDER - ORDER OF PREDICTION                        C
C                 ALPHA - EXPONENTIAL WEIGHT FOR PWRQS COMPUTATION   C
C                 BETA - CONSTANT BIAS TERM TO ADD TO PWRQS          C
C                 GAIN - GAIN OF UPDATE TERM                         C
C                 G - GAIN ADJUSTED TO ORDER OF PREDICTION;          C
C                     G = GAIN FOR ORDER = 10                        C
C                                                                    C
C ****************************************************************** C

      SUBROUTINE PRDLMS(QS,LMSQS,A,PWRQS,K,ORDER,IQDIF,P)

      REAL LMSQS(13,30), A(13,30), PWRQS(13)
      REAL QS(13), IQDIF(13), P(13)
      INTEGER*2 K, ORDER

      REAL ALPHA, BETA, GAIN, G

      DATA ALPHA/0.720/, BETA/3000./, GAIN/.065/


C
C ADJUST GAIN TO ORDER
C
      G = GAIN * (10.0/ORDER)


C
C UPDATE POWER ESTIMATE, PWRQS(K)
C
      PWRQS(K) = ALPHA*PWRQS(K) + (1.0-ALPHA)*QS(K)**2
```

```
C
C UPDATE COEFFICIENTS
C
      DO 10 I = 1,ORDER
 10   A(K,I) = A(K,I) + G*IQDIF(K)*LMSQS(K,I)/(PWRQS(K)+BETA)


C
C SHIFT IN QS(K) AND CALCULATE TAPPED DELAY LINE OUTPUT, P(K)
C
      P(K) = 0
      DO 20 I = 1,ORDER
      J = ORDER + 1 - I
      IF (J.EQ.1) LMSQS(K,J) = QS(K)
      IF (J.NE.1) LMSQS(K,J) = LMSQS(K,J-1)
 20   P(K) = P(K) + A(K,J)*LMSQS(K,J)


      RETURN
      END
```

```
C ******************************************************************** C
C                                                                      C
C PROGRAM : PRDLS                                                      C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : SUBROUTINE TO COMPUTE SINGLE OUTPUT OF LEAST-SQUARES   C
C               LATTICE PREDICTOR.                                     C
C                                                                      C
C CALLED BY : CODLS, DECLS                                             C
C                                                                      C
C KEY VARIABLES : PCOR - PARTIAL CORRELATION                           C
C                 VARE - VARIANCE OF FORWARD PREDICTION ERROR          C
C                 VARR - VARIANCE OF BACKWARD PREDICTION ERROR         C
C                 VARRM1 - ONE SAMPLE DELAY OF VARR                    C
C                 G - LIKELIHOOD VARIABLE                              C
C                 GM1 - ONE SAMPLE DELAY OF G                          C
C                 E - FORWARD PREDICTION ERROR                         C
C                 R - BACKWARD PREDICTION ERROR                        C
C                 RM1 - ONE SAMPLE DELAY OF R                          C
C                 QS - SUM OF PREDICTOR OUTPUT AND INVERSE             C
C                      QUANTIZED DIFFERENCE (ERROR) SIGNAL             C
C                 P - PREDICTOR OUTPUT                                 C
C                 W - EXPONENTIAL WEIGHT FOR TIME UPDATES              C
C                 K - SUBBAND NUMBER                                   C
C                 ORDER - ORDER OF PREDICTION                          C
C                                                                      C
C ******************************************************************** C

      SUBROUTINE PRDLS(QS,PCOR,VARE,VARR,VARRM1,G,GM1,E,R,RM1,K,ORDER,P)

      REAL PCOR(13,30), VARE(13,30), VARR(13,30), VARRM1(13,30)
      REAL G(13,30), GM1(13,30), E(13,30), R(13,30), RM1(13,30)
      REAL QS(13), P(13), W
      INTEGER*2 K, ORDER

      DATA W/.950/


C
C INITIALIZE STAGE 0, CORRESPONDING TO ARRAY INDEX 1
C
      E(K,1) = QS(K)
      RM1(K,1) = R(K,1)
      R(K,1) = QS(K)
      VARE(K,1) = W*VARE(K,1) + QS(K)**2
      VARRM1(K,1) = VARR(K,1)
      VARR(K,1) = VARE(K,1)


C
```

```
C ITERATE THROUGH STAGES I = 1 TO ORDER,
C     WHICH CORRESPOND TO ARRAY INDICES 2 TO ORDER+1;
C     THE MAXIMUM ORDER IS 29
C
      DO 10 I = 1,ORDER
      IF (I.NE.1)
     X  PCOR(K,I+1) = W*PCOR(K,I+1) + RM1(K,I)*E(K,I)/(1.0-GM1(K,I-1))
      IF (I.EQ.1)
     X  PCOR(K,I+1) = W*PCOR(K,I+1) + RM1(K,I)*E(K,I)
      GM1(K,I) = G(K,I)
      IF (I.NE.1) G(K,I) = G(K,I-1) + (R(K,I)**2)/VARR(K,I)
      IF (I.EQ.1) G(K,I) = (R(K,I)**2)/VARR(K,I)
      IF (G(K,I).LT.1) GO TO 30
      PRINT 300
 300  FORMAT (' LS LIKELIHOOD VARIABLE >= 1, SET TO .9999999')
      G(K,I) = .9999999
 30   E(K,I+1) = E(K,I) - (PCOR(K,I+1)/VARRM1(K,I))*RM1(K,I)
      RM1(K,I+1) = R(K,I+1)
      R(K,I+1) = RM1(K,I) - (PCOR(K,I+1)/VARE(K,I))*E(K,I)
      VARE(K,I+1) = W*VARE(K,I+1) + (E(K,I+1)**2)/(1-GM1(K,I))
      VARRM1(K,I+1) = VARR(K,I+1)
 10   VARR(K,I+1) = W*VARR(K,I+1) + (R(K,I+1)**2)/(1-G(K,I))


C
C CALCULATE PREDICTOR OUTPUT USING R, NOT RM1, SO WE USE CURRENT INPUT
C     (WE WANT TO PREDICT THE NEXT INPUT, NOT THE CURRENT ONE)
C
      P(K) = 0
      DO 20 I = 1,ORDER
 20   P(K) = P(K) + (PCOR(K,I+1)/VARRM1(K,I))*R(K,I)


      RETURN
      END
```

```
10 REM  ************************************************************
20 REM  *                                                          *
30 REM  * PROGRAM : FQMFBANK                                       *
40 REM  *                                                          *
50 REM  * BY : PAUL NING                                           *
60 REM  *                                                          *
70 REM  * DATE : 7/24/87                                           *
80 REM  *                                                          *
90 REM  * DESCRIPTION : PROGRAM TO GENERATE PARALLEL FILTER BANK   *
100 REM *               BANDPASS COEFFICIENTS FROM LOWPASS FILTERS *
110 REM *               OF CORRESPONDING TREE STRUCTURE.  ALSO     *
120 REM *               COMPUTES THE COMPOSITE FREQUENCY RESPONSE  *
130 REM *               OF THE PARALLEL QMF.                       *
140 REM *                                                          *
150 REM * CALLS : SUBROUTINE TO CONVERT TO BINARY AND GRAY CODES   *
160 REM *                                                          *
170 REM * KEY VARIABLES : M - TAP LENGTHS OF LOWPASS FILTERS       *
180 REM *                 C - ARRAY OF COEFFICIENTS FOR FREQUENCY  *
190 REM *                     NORMALIZED LOWPASS AND HIGHPASS FILTERS *
200 REM *                 STAGES - NUMBER OF STAGES                *
210 REM *                 N - TAP LENGTHS OF FREQUENCY NORMALIZED  *
220 REM *                     HALF-BAND FILTERS                    *
230 REM *                 G - COMPOSITE IMPULSE RESPONSE           *
240 REM *                 B - GRAY CODE ARRAY                      *
250 REM *                 BTWO - BASE TWO (BINARY) ARRAY           *
260 REM *                 X, Y, Z - ARRAYS FOR HANDLING CONVOLUTIONS *
270 REM *                           (X*Y=Z)                        *
280 REM *                                                          *
290 REM * FORMAT OF COEFFICIENT FILES :                            *
300 REM *                                                          *
310 REM *         THIS FORMAT HANDLES A GENERAL POLE-ZERO COMPLEX  *
320 REM *         IMPULSE RESPONSE.  THE FIRST LINE OF THE FILE IS *
330 REM *         THE NUMBER OF ZEROES IN THE FREQUENCY RESPONSE.  *
340 REM *         THIS IS FOLLOWED BY THE REAL AND IMAGINARY PARTS, *
350 REM *         RESPECTIVELY, OF THESE COEFFICIENTS.  SIMILARLY, *
360 REM *         THE NUMBER OF POLES IS GIVEN, FOLLOWED BY THEIR  *
370 REM *         COEFFICIENTS IN COMPLEX FORM.                    *
380 REM *                                                          *
390 REM *         THIS PROGRAM DEALS ONLY WITH REAL FIR FILTERS.   *
400 REM *         IN THE FORMAT JUST DESCRIBED, THE NUMBER OF TAPS *
410 REM *         IS GIVEN AT THE BEGINNING OF THE FILE, AND THE   *
420 REM *         COEFFICIENTS (EACH FOLLOWED BY A ZERO IMAGINARY  *
430 REM *         PART) ARE LISTED NEXT.  THE FILE ENDS WITH A     *
440 REM *         ZERO REPRESENTING THE NUMBER OF POLES.           *
450 REM *                                                          *
460 REM ************************************************************
470 CLS
480 PRINT "***** Parallel QMF Bank Generator *****"
490 REM *
500 REM * maximum 5 stages, maximum 128 taps per stage
510 REM *
520 DIM F$(5), N(5), C(5,2,128)
530 REM *
540 REM * get coefficient filenames
```

```
550 REM *
560 INPUT "How many stages in analogous tree structure "; STAGES
570 FOR S = 1 TO STAGES
580 PRINT "Coefficient file for lowpass filter of stage";S;"-<fn>.cof."
590 INPUT "     Enter <fn> ";F$(S)
600 NEXT S
610 REM *
620 REM * make coefficient arrays and get filter lengths
630 REM *
640 FOR S = 1 TO STAGES
650 OPEN F$(S)+".cof" FOR INPUT AS #1
660 OPEN F$(S)+"mir.cof" FOR OUTPUT AS #2
670 INPUT #1,M(S)
680 PRINT #2,M(S)
690 REM *
700 REM * initialize arrays (original and mirror) to all zeroes
710 REM *
720 N(S) = 2^(S-1)*(M(S)-1)+1
730 FOR I = 0 TO N(S)-1
740 C(S,0,I) = 0
750 C(S,1,I) = 0
760 NEXT I
770 REM *
780 REM * read coefficients and write to correct array positions
790 REM *
800 FOR J = 0 TO M(S)-1
810 INPUT #1,C(S,0,J*(2^(S-1)))
820 INPUT #1,IMAGJUNK
830 C(S,1,J*(2^(S-1))) = (-1)^J*C(S,0,J*(2^(S-1)))
840 PRINT #2, C(S,1,J*(2^(S-1))), IMAGJUNK
850 NEXT J
860 PRINT #2,0
870 CLOSE #1,#2
880 NEXT S
890 REM *
900 REM * prepare for composite response calculation, g(u)
910 REM *
920 L = 1
930 FOR S = 1 TO STAGES
940 L = L + N(S) - 1
950 NEXT S
960 DIM G(2*L-1)
970 REM  *
980 REM  *
990 REM  *
1000 REM * convolve the stages to get bandpass filters
1010 REM * (only half of the them need explicit convolution,
1020 REM *   the other half are mirror images)
1030 REM *
1040 REM *
1050 REM *
1060 FOR K = 0 TO 2^(STAGES-1)-1
1070 DIM B(STAGES)
1080 DIM BTWO(STAGES)
```

```
1090 GOSUB 2000
1100 FOR R = STAGES-1 TO 0 STEP -1
1110 PRINT B(R);
1120 NEXT R
1130 PRINT
1140 REM *
1150 REM * loop to convolve one stage at a time, x(j)*y(j)=z(j)
1160 REM *       x = latest convolved result
1170 REM *       y = current stage response
1180 REM *       z = new convolved result
1190 REM *
1200 LENGTH = 1
1210 DIM X(LENGTH)
1220 X(0) = 1
1230 FOR S = 1 TO STAGES
1240 PRINT S
1250 DIM Y(N(S)),Z(LENGTH+N(S)-1)
1260 FOR I = 0 TO N(S)-1
1270 Y(I) = C(S,B(STAGES-S),I)
1280 NEXT I
1290 FOR J = 0 TO LENGTH+N(S)-2
1300 Z(J) = 0
1310 FOR A = 0 TO LENGTH-1
1320 IF J-A < N(S) AND J-A >= 0 THEN Z(J) = Z(J) + X(A)*Y(J-A)
1330 NEXT A
1340 NEXT J
1350 ERASE X
1360 REM *
1370 REM * write z to x in preparation for another convolution
1380 REM *
1390 LENGTH = LENGTH+N(S)-1
1400 DIM X(LENGTH)
1410 FOR T = 0 TO LENGTH-1
1420 X(T) = Z(T)
1430 NEXT T
1440 ERASE Y,Z
1450 NEXT S
1460 REM *
1470 REM * write bandpass coefficients to files
1480 REM *
1490 NUM$ = STR$(K)
1500 NUMMIR$ = STR$(2^STAGES-1-K)
1510 D$ = MID$(NUM$,2,2)
1520 DMIR$ = MID$(NUMMIR$,2,2)
1530 F1$ = "f" + D$
1540 N1$ = F1$ + ".cof"
1550 F2$ = "f" + DMIR$
1560 PRINT
1570 PRINT F1$,F2$
1580 N2$ = F2$ + ".cof"
1590 OPEN N1$ FOR OUTPUT AS #1
1600 OPEN N2$ FOR OUTPUT AS #2
1610 PRINT #1, LENGTH
1620 PRINT #2, LENGTH
```

```
1630 FOR T = 0 TO LENGTH-1
1640 PRINT #1, X(T), 0
1650 PRINT #2, (-1)^T*X(T), 0
1660 NEXT T
1670 PRINT #1, 0
1680 PRINT #2, 0
1690 CLOSE #1,#2
1700 REM *
1710 REM * update g(u)
1720 REM *
1730 FOR U = 1 TO 2*L-3 STEP 2
1740 SUM = 0
1750 FOR T = 0 TO L-1
1760 IF U-T < L AND U-T >= 0 THEN SUM = SUM + X(T)*X(U-T)
1770 NEXT T
1780 G(U) = G(U) + 2*((-1)^K)*SUM
1790 NEXT U
1800 REM *
1810 REM * get ready for next bandpass calculation
1820 REM *
1830 ERASE B,X,BTWO
1840 NEXT K
1850 REM *
1860 REM * write g(u) to file
1870 REM *
1880 OPEN "g.cof" FOR OUTPUT AS #1
1890 PRINT #1, 2*L-1
1900 FOR U = 0 TO 2*L-2
1910 PRINT #1, G(U), 0
1920 NEXT U
1930 PRINT #1, 0
1940 CLOSE #1
1950 SYSTEM
1960 END
1970 REM *
1980 REM *
1990 REM *
2000 REM * subroutine to convert to base two (btwo) and gray code (b)
2010 REM *
2020 REM *
2030 REM *
2040 D = K
2050 PAR = 0
2060 FOR R = STAGES-1 TO 0 STEP -1
2070 IF D >= 2^R THEN BTWO(R)=1: D=D-2^R ELSE BTWO(R)=0
2080 IF PAR=0 THEN B(R)=BTWO(R): PAR=B(R) ELSE B(R)=1-BTWO(R):PAR=1-B(R)
2090 NEXT R
2100 RETURN
```

```
10 REM ***********************************************************
20 REM  *                                                        *
30 REM  * PROGRAM : TRUNCATE                                     *
40 REM  *                                                        *
50 REM  * BY : PAUL NING                                         *
60 REM  *                                                        *
70 REM  * DATE : 7/24/87                                         *
80 REM  *                                                        *
90 REM  * DESCRIPTION : PROGRAM TO TRUNCATE IMPULSE RESPONSES OF *
100 REM *               PARALLEL BANDPASS FILTERS GENERATED BY   *
110 REM *               FQMFBANK.  ALSO CALCULATES COEFFICIENTS  *
120 REM *               OF NEW COMPOSITE RESPONSE.               *
130 REM *                                                        *
140 REM * KEY VARIABLES : K - NUMBER OF BANDPASS FILTERS         *
150 REM *                 N - ORDER OF TRUNCATED FILTERS         *
160 REM *                 L - ORDER OF FILTERS BEFORE TRUNCATION *
170 REM *                                                        *
180 REM * FORMAT OF COEFFICIENT FILES :                          *
190 REM *                                                        *
200 REM *           THIS FORMAT HANDLES A GENERAL POLE-ZERO COMPLEX  *
210 REM *           IMPULSE REPONSE.  THE FIRST LINE OF THE FILE IS  *
220 REM *           THE NUMBER OF ZEROES IN THE FREQUENCY RESPONSE.  *
230 REM *           THIS IS FOLLOWED BY THE REAL AND IMAGINARY PARTS, *
240 REM *           RESPECTIVELY, OF THESE COEFFICIENTS.  SIMILARLY, *
250 REM *           THE NUMBER OF POLES IS GIVEN, FOLLOWED BY THEIR  *
260 REM *           COEFFICIENTS IN COMPLEX FORM.                *
270 REM *                                                        *
280 REM *           THIS PROGRAM DEALS ONLY WITH REAL FIR FILTERS.   *
290 REM *           IN THE FORMAT JUST DESCRIBED, THE NUMBER OF TAPS *
300 REM *           IS GIVEN AT THE BEGINNING OF THE FILE, AND THE   *
310 REM *           COEFFICIENTS (EACH FOLLOWED BY A ZERO IMAGINARY  *
320 REM *           PART) ARE LISTED NEXT.  THE FILE ENDS WITH A     *
330 REM *           ZERO REPRESENTING THE NUMBER OF POLES.       *
340 REM *                                                        *
350 REM * INPUT BANDPASS FILES : F0.COF, F1.COF, ... , F(K-1).COF    *
360 REM *                                                        *
370 REM * TRUNCATED BANDPASS FILES : TF0.COF, TF1.COF, ... ,TF(K-1).COF*
380 REM *                                                        *
390 REM * NEW COMPOSITE RESPONSE FILE : TG.COF                   *
400 REM *                                                        *
410 REM ***********************************************************
420 CLS
430 PRINT "***** Parallel QMF Truncation Program *****"
440 INPUT "How many bandpass filters are there in the QMF bank ";K
450 INPUT "Truncate the filters to what order ";N
460 REM *
470 REM * read in half of bandpass filters and truncate
480 REM *
490 DIM TG(2*N-1)
500 FOR I = 0 TO K/2-1
510 DIM X(N)
520 OPEN "f"+MID$(STR$(I),2,2)+".cof" FOR INPUT AS #1
530 INPUT #1, L
540 FOR J = 0 TO L-1
```

```
550 INPUT #1,C,IMAGJUNK
560 IF J>(L-N)/2-1 AND J<(L+N)/2 THEN X(J-(L-N)/2) = C
570 NEXT J
580 CLOSE #1
590 REM *
600 REM * update tg(u)
610 REM *
620 FOR U = 1 TO 2*N-3 STEP 2
630 SUM = 0
640 FOR T = 0 TO N-1
650 IF U-T < N AND U-T >= 0 THEN SUM = SUM + X(T)*X(U-T)
660 NEXT T
670 TG(U) = TG(U) + 2*((-1)^I)*SUM
680 NEXT U
690 REM *
700 REM * write truncated coefficients to files
710 REM *
720 OPEN "tf"+MID$(STR$(I),2,2)+".cof" FOR OUTPUT AS #1
730 OPEN "tf"+MID$(STR$(K-I-1),2,2)+".cof" FOR OUTPUT AS #2
740 PRINT #1, N
750 PRINT #2, N
760 FOR J = 0 TO N-1
770 PRINT #1, X(J), 0
780 PRINT #2, (-1)^J*X(J), 0
790 NEXT J
800 PRINT #1, 0
810 PRINT #2, 0
820 CLOSE #1, #2
830 REM *
840 REM * get ready for next bandpass calculations
850 REM *
860 ERASE X
870 NEXT I
880 REM *
890 REM * write truncated tg(u) to file
900 REM *
910 OPEN "tg.cof" FOR OUTPUT AS #1
920 PRINT #1, 2*N-1
930 FOR U = 0 TO 2*N-2
940 PRINT #1, TG(U), 0
950 NEXT U
960 PRINT #1, 0
970 CLOSE #1
980 SYSTEM
990 END
```

```
C ******************************************************************** C
C                                                                      C
C PROGRAM : SNR                                                        C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : PROGRAM TO COMPUTE SIGNAL-TO-NOISE RATIO FROM TWO      C
C               FILES, ASSUMING THE FIRST IS THE PURE SIGNAL AND       C
C               THE SECOND IS THE SIGNAL PLUS NOISE.                   C
C                                                                      C
C CALLS : IN                                                           C
C                                                                      C
C KEY VARIABLES : DELAY - DELAY BETWEEN ORIGINAL AND PROCESSED FILES   C
C                 S - BUFFER FOR FIRST FILE (SIGNAL)                   C
C                 PS1 - BUFFER #1 FOR SECOND FILE (PROCESSED SIGNAL)   C
C                 PS2 - BUFFER #2 FOR SECOND FILE                      C
C                 PWRS - SIGNAL POWER                                  C
C                 PWRN - NOISE POWER                                   C
C                 SNR - SIGNAL-TO-NOISE RATIO                          C
C                                                                      C
C ******************************************************************** C


      INTEGER*2 DELAY, S(128), PS1(128), PS2(128)
      REAL*8 PWRS, PWRN, SNR
      REAL R
      DATA PWRS/0.0/, PWRN/0.0/


C
C PROMPT FOR DELAY VALUE
C
      PRINT 100
 100  FORMAT (' DELAY ( < 128 )')
      READ (5,*) DELAY


C
C READ INITIAL BLOCK FROM SECOND FILE (FILE NUMBER = 9)
C
      CALL IN(9,PS1,ISTAT)
      IF (ISTAT.EQ.1) GO TO 300


C
C LOOP TO READ BLOCKS FROM FIRST FILE (FILE NUMBER = 8) AND SECOND FILE
C
 10   CALL IN(8,S,ISTAT)
      IF (ISTAT.EQ.1) GO TO 300
      CALL IN(9,PS2,ISTAT)
      IF (ISTAT.EQ.1) GO TO 300
C
```

```
C     PROCESS 128 SAMPLES
C
      DO 50 I = 1,128
      PWRS = PWRS + S(I)**2
      J = DELAY + I
      IF (J.GT.128) GO TO 20
      PWRN = PWRN + (S(I)-PS1(J))**2
      GO TO 50
 20   PWRN = PWRN + (S(I)-PS2(J-128))**2
 50   CONTINUE
C
C     PUT ALL PS2 DATA INTO PS1 SO PS2 CAN BE USED AGAIN
C
      DO 30 I = 1,128
 30   PS1(I) = PS2(I)
      GO TO 10


C
C COMPUTE SNR
C
 300  IF ((PWRS.EQ.0).OR.(PWRN.EQ.0)) GO TO 40
      R = PWRS/PWRN
      SNR = 10*LOG10(R)
      PRINT *, SNR
      GO TO 60


C
C PRINT MESSAGE IF SIGNAL OR NOISE POWER IS ZERO
C
 40   PRINT 500
 500  FORMAT(' SNR OUT OF BOUNDS')


 60   STOP
      END
```

```
C *********************************************************************** C
C                                                                         C
C PROGRAM : SSNR                                                          C
C                                                                         C
C BY : PAUL NING                                                          C
C                                                                         C
C DATE : 7/24/87                                                          C
C                                                                         C
C DESCRIPTION : PROGRAM TO COMPUTE SEGMENTAL SIGNAL-TO-NOISE RATIO       C
C               FROM TWO FILES, ASSUMING THE FIRST IS THE PURE SIGNAL    C
C               AND THE SECOND IS THE SIGNAL PLUS NOISE.  SEGMENTS       C
C               ARE 16 MS LONG (128 SAMPLES).                            C
C                                                                         C
C CALLS : IN                                                              C
C                                                                         C
C KEY VARIABLES : DELAY - DELAY BETWEEN ORIGINAL AND PROCESSED FILES     C
C                 S - BUFFER FOR FIRST FILE (SIGNAL)                      C
C                 PS1 - BUFFER #1 FOR SECOND FILE (PROCESSED SIGNAL)      C
C                 PS2 - BUFFER #2 FOR SECOND FILE                         C
C                 PWRS - SIGNAL POWER (RECOMPUTED EACH SEGMENT)           C
C                 PWRN - NOISE POWER (RECOMPUTED EACH SEGMENT)            C
C                 SSNR - SEGMENTAL SIGNAL-TO-NOISE RATIO                  C
C                                                                         C
C *********************************************************************** C


      INTEGER*2 DELAY, S(128), PS1(128), PS2(128)
      REAL*8 PWRS, PWRN, SSNR
      REAL R, T
      DATA PWRS/0.0/, PWRN/0.0/, SSNR/0.0/, T/0.0/


C
C PROMPT FOR DELAY VALUE
C
      PRINT 100
 100  FORMAT (' DELAY ( < 128 )')
      READ (5,*) DELAY


C
C READ INITIAL BLOCK FROM SECOND FILE (FILE NUMBER = 9)
C
      CALL IN(9,PS1,ISTAT)
      IF (ISTAT.EQ.1) GO TO 300


C
C LOOP TO READ BLOCKS FROM FIRST FILE (FILE NUMBER = 8) AND SECOND FILE
C
 10   CALL IN(8,S,ISTAT)
      IF (ISTAT.EQ.1) GO TO 300
      CALL IN(9,PS2,ISTAT)
      IF (ISTAT.EQ.1) GO TO 300
```

```
C
C     PROCESS 128 SAMPLES
C
      DO 50 I = 1,128
      PWRS = PWRS + S(I)**2
      J = DELAY + I
      IF (J.GT.128) GO TO 20
      PWRN = PWRN + (S(I)-PS1(J))**2
      GO TO 50
 20   PWRN = PWRN + (S(I)-PS2(J-128))**2
 50   CONTINUE
C
C     UPDATE SSNR AND RESET PWRS, PWRN
C
      IF ((PWRS.EQ.0).OR.(PWRN.EQ.0)) GO TO 40
      R = PWRS/PWRN
      SSNR = (T/(T+1))*SSNR + (1/(T+1))*10*LOG10(R)
      PWRS = 0
      PWRN = 0
C
C     PUT ALL PS2 DATA INTO PS1 SO PS2 CAN BE USED AGAIN
C
      DO 30 I = 1,128
 30   PS1(I) = PS2(I)
      T = T + 1
      GO TO 10


C
C PRINT SSNR RESULT
C
 300  PRINT *, SSNR
      GO TO 60


C
C PRINT MESSAGE IF SIGNAL OR NOISE POWER IS ZERO
C
 40   PRINT 500
 500  FORMAT(' SSNR OUT OF BOUNDS')


 60   STOP
      END
```

```
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX C
C                                                                      C
C PROGRAM : IN                                                         C
C                                                                      C
C BY : PAUL NING                                                       C
C                                                                      C
C DATE : 7/24/87                                                       C
C                                                                      C
C DESCRIPTION : SUBROUTINE TO READ IN BLOCKS OF 128 SAMPLES.           C
C               IDENTICAL TO INBLK EXCEPT FOR THE FLEXIBILITY OF       C
C               SPECIFYING THE INPUT FILE NUMBER.                      C
C                                                                      C
C                                                                      C
C CALLED BY : SNR, SSNR                                                C
C                                                                      C
C KEY VARIABLES : INBUF - INPUT BUFFER                                 C
C                 ISTAT - I/O STATUS                                   C
C                 N - INPUT FILE NUMBER                                C
C                                                                      C
C XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX C

      SUBROUTINE IN(N,INBUF,ISTAT)

      INTEGER*2 INBUF(128)

      READ (N,100,END=200,ERR=200) (INBUF(I),I=1,128)
      ISTAT = 0
      RETURN
100   FORMAT(128(Z4))
200   ISTAT = 1

      RETURN
      END
```

## APPENDIX D - TEST DATA

This appendix contains some simulation data associated with the tests discussed in Chapter 3.

### D.1 BIT ALLOCATION

Tables D-1 to D-4 show subband characteristic ranges and average bit allocations for each of the four test sentences at the two bit rates, 16 kbps and 24 kbps.

DARCI

| Band, i | $C(i)_{min}$ | $C(i)_{avg}$ | $C(i)_{max}$ | $B_{avg}$ (no pred.) 16 kbps | 24 kbps | $B_{avg}$ (with pred.) 16 kbps | 24 kbps |
|---|---|---|---|---|---|---|---|
| 1 | .7 | 60.4 | 284.0 | 1.6 | 2.8 | 2.3 | 3.0 |
| 2 | 1.5 | 195.6 | 1141.0 | 2.8 | 3.9 | 2.9 | 3.9 |
| 3 | 1.0 | 245.8 | 2094.6 | 2.7 | 4.0 | 2.8 | 4.0 |
| 4 | .5 | 314.7 | 3493.4 | 2.4 | 3.7 | 2.6 | 3.7 |
| 5 | 1.0 | 197.4 | 2016.9 | 2.4 | 3.7 | 2.4 | 3.7 |
| 6 | 1.0 | 177.6 | 2259.8 | 2.3 | 3.5 | 2.3 | 3.5 |
| 7 | .8 | 190.5 | 1580.3 | 2.3 | 3.5 | 2.3 | 3.5 |
| 8 | .7 | 164.0 | 1336.0 | 2.3 | 3.5 | 2.3 | 3.5 |
| 9 | .6 | 118.3 | 921.6 | 1.8 | 3.2 | 1.8 | 3.2 |
| 10 | 1.3 | 100.1 | 957.1 | 1.8 | 3.1 | 1.8 | 3.1 |
| 11 | .6 | 91.4 | 793.2 | 1.6 | 2.8 | 1.5 | 2.8 |
| 12 | .5 | 84.5 | 734.5 | 1.6 | 2.8 | 1.3 | 2.8 |
| 13 | .6 | 60.0 | 363.0 | 1.3 | 2.5 | .7 | 2.3 |

Table D-1. Subband Characteristics and Bit Allocation (DARCI)

BETH

| Band, i | $C(i)_{min}$ | $C(i)_{avg}$ | $C(i)_{max}$ | $B_{avg}$ (no pred.) 16 kbps | 24 kbps | $B_{avg}$ (with pred.) 16 kbps | 24 kbps |
|---|---|---|---|---|---|---|---|
| 1 | 1.7 | 32.6 | 165.0 | 2.8 | 4.0 | 3.1 | 4.0 |
| 2 | 1.2 | 67.3 | 320.5 | 3.4 | 4.4 | 3.4 | 4.4 |
| 3 | 1.0 | 65.3 | 318.4 | 3.2 | 4.2 | 3.2 | 4.2 |
| 4 | .5 | 80.4 | 500.7 | 2.7 | 3.7 | 2.9 | 3.7 |
| 5 | .7 | 72.4 | 711.5 | 2.6 | 3.7 | 2.6 | 3.7 |
| 6 | .3 | 54.0 | 542.2 | 2.2 | 3.5 | 2.2 | 3.5 |
| 7 | .4 | 31.5 | 463.6 | 2.0 | 3.4 | 2.0 | 3.4 |
| 8 | .6 | 16.6 | 192.7 | 1.7 | 3.1 | 1.7 | 3.1 |
| 9 | .5 | 9.9 | 70.0 | 1.4 | 2.7 | 1.4 | 2.7 |
| 10 | .9 | 7.9 | 72.8 | 1.4 | 2.7 | 1.4 | 2.7 |
| 11 | .4 | 7.5 | 56.9 | 1.2 | 2.4 | 1.2 | 2.4 |
| 12 | .4 | 8.2 | 43.0 | 1.1 | 2.5 | 1.0 | 2.5 |
| 13 | .4 | 10.0 | 80.0 | 1.2 | 2.6 | .9 | 2.6 |

Table D-2. Subband Characteristics and Bit Allocation (BETH)

GLENN

| Band, i | $C(i)_{min}$ | $C(i)_{avg}$ | $C(i)_{max}$ | $B_{avg}$ (no pred.) 16 kbps | 24 kbps | $B_{avg}$ (with pred.) 16 kbps | 24 kbps |
|---|---|---|---|---|---|---|---|
| 1 | .5 | 35.1 | 137.2 | 2.2 | 3.4 | 2.6 | 3.5 |
| 2 | 1.7 | 172.0 | 785.1 | 3.5 | 4.5 | 3.6 | 4.5 |
| 3 | 1.0 | 140.9 | 791.8 | 3.1 | 4.1 | 3.1 | 4.1 |
| 4 | .6 | 91.0 | 562.8 | 2.4 | 3.6 | 2.6 | 3.6 |
| 5 | .7 | 73.7 | 659.7 | 2.2 | 3.4 | 2.2 | 3.4 |
| 6 | .5 | 86.2 | 881.2 | 2.2 | 3.5 | 2.2 | 3.5 |
| 7 | .5 | 75.1 | 945.7 | 2.0 | 3.2 | 2.0 | 3.2 |
| 8 | .4 | 54.0 | 794.5 | 1.5 | 2.9 | 1.5 | 2.9 |
| 9 | .6 | 49.8 | 596.4 | 1.6 | 2.9 | 1.6 | 2.9 |
| 10 | 1.2 | 47.6 | 529.3 | 1.9 | 3.2 | 1.9 | 3.2 |
| 11 | .6 | 58.6 | 758.4 | 1.7 | 2.9 | 1.6 | 2.9 |
| 12 | .6 | 46.2 | 506.6 | 1.6 | 2.8 | 1.3 | 2.8 |
| 13 | .5 | 33.9 | 580.5 | 1.3 | 2.5 | .8 | 2.4 |

Table D-3. Subband Characteristics and Bit Allocation (GLENN)

MIKE

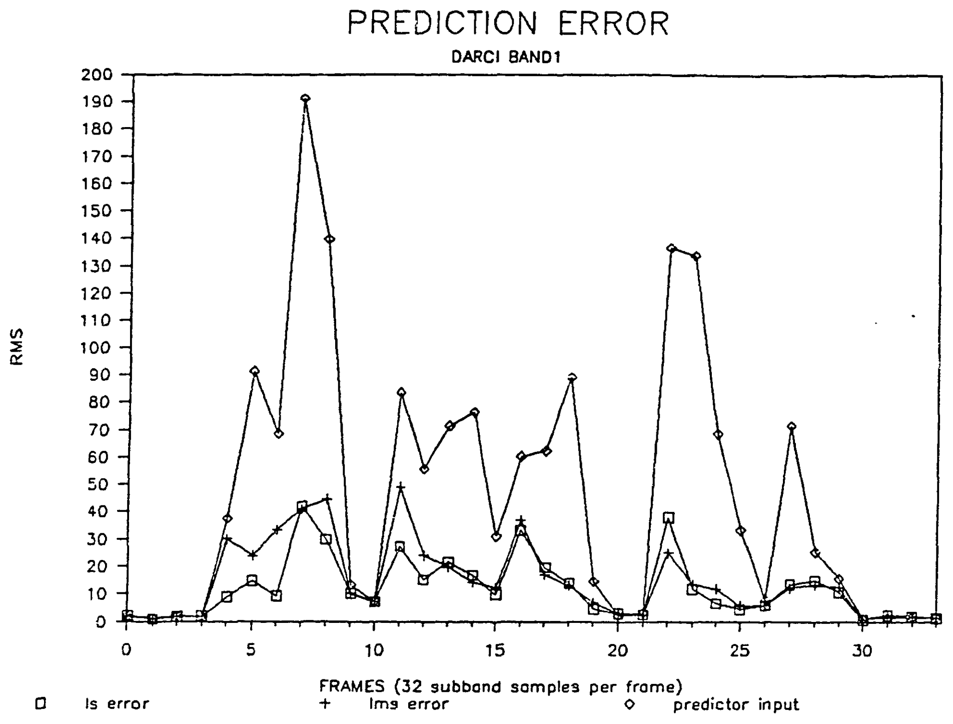
| Band, i | $C(i)_{min}$ | $C(i)_{avg}$ | $C(i)_{max}$ | $B_{avg}$ (no pred.) 16 kbps | 24 kbps | $B_{avg}$ (with pred.) 16 kbps | 24 kbps |
|---|---|---|---|---|---|---|---|
| 1 | 1.7 | 80.0 | 381.5 | 2.0 | 3.3 | 2.5 | 3.4 |
| 2 | 2.2 | 484.8 | 2789.5 | 3.6 | 4.4 | 3.6 | 4.4 |
| 3 | 1.1 | 276.9 | 2500.4 | 2.7 | 3.9 | 2.8 | 3.9 |
| 4 | .4 | 189.8 | 2464.6 | 2.3 | 3.5 | 2.6 | 3.5 |
| 5 | .9 | 217.7 | 4040.1 | 2.3 | 3.5 | 2.3 | 3.5 |
| 6 | 1.0 | 154.5 | 2246.3 | 2.1 | 3.4 | 2.1 | 3.4 |
| 7 | .7 | 112.5 | 1591.8 | 1.9 | 3.2 | 1.9 | 3.2 |
| 8 | .8 | 101.9 | 1938.1 | 1.8 | 3.1 | 1.8 | 3.1 |
| 9 | .6 | 111.5 | 1928.4 | 1.7 | 3.0 | 1.7 | 3.0 |
| 10 | 1.1 | 113.4 | 1253.5 | 2.1 | 3.3 | 2.1 | 3.3 |
| 11 | .6 | 85.9 | 955.3 | 1.7 | 3.0 | 1.7 | 3.0 |
| 12 | .8 | 67.0 | 401.1 | 1.5 | 2.8 | 1.2 | 2.8 |
| 13 | .6 | 66.1 | 594.8 | 1.3 | 2.6 | .7 | 2.5 |

Table D-4. Subband Characteristics and Bit Allocation (MIKE)

It can be seen that the four sentences are of somewhat different volumes. DARCI and MIKE are louder than BETH and GLENN. By presenting this variety, the robustness of the test set is enhanced.

## D.2 FRAME-TO-FRAME PREDICTOR PERFORMANCE

As an indication of how well the LMS transversal and LS lattice track their inputs, RMS values are calculated for the subband signals and prediction errors over successive frames of 32 subband samples. Graphs of these values for the four bands of the four test sentences are shown below. Combined plots of all four subbands for each sentence are given in Chapter 3.



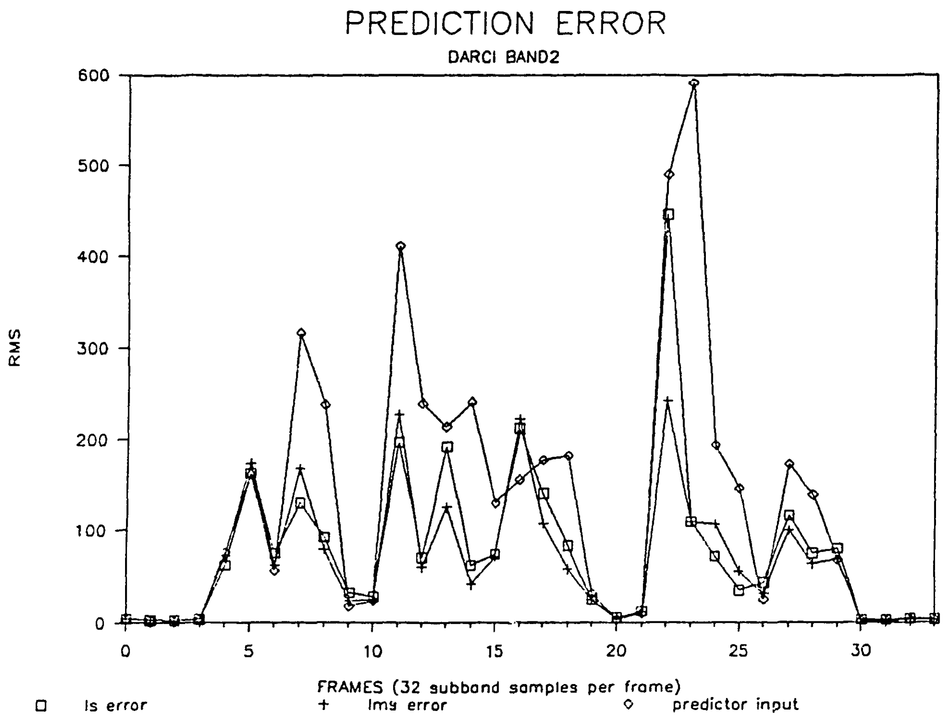
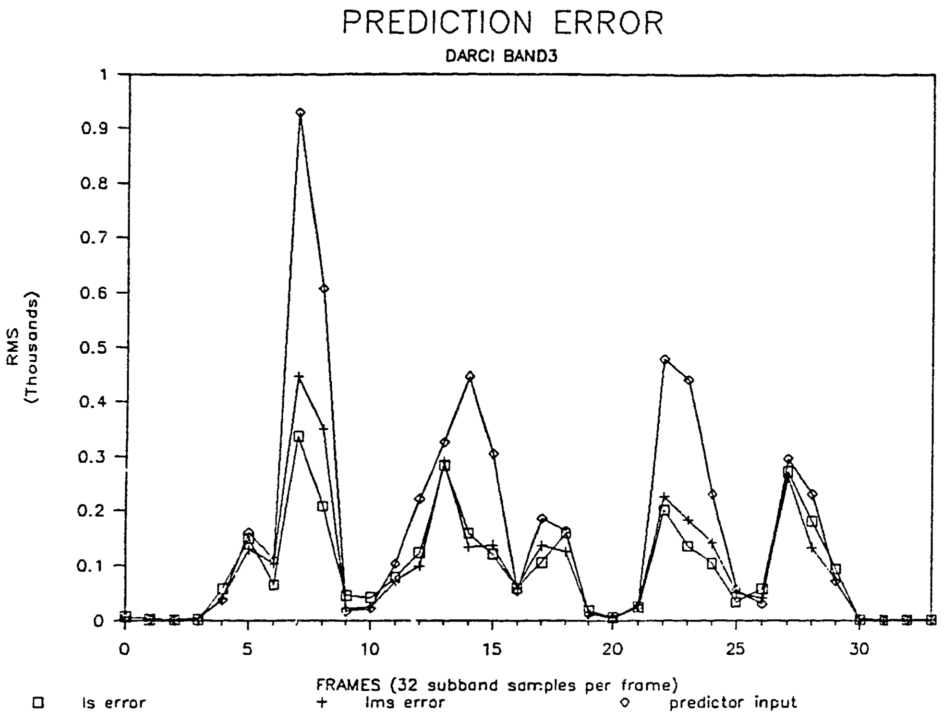


**Figure D-1. Frame-to-Frame Prediction Error : DARCI**



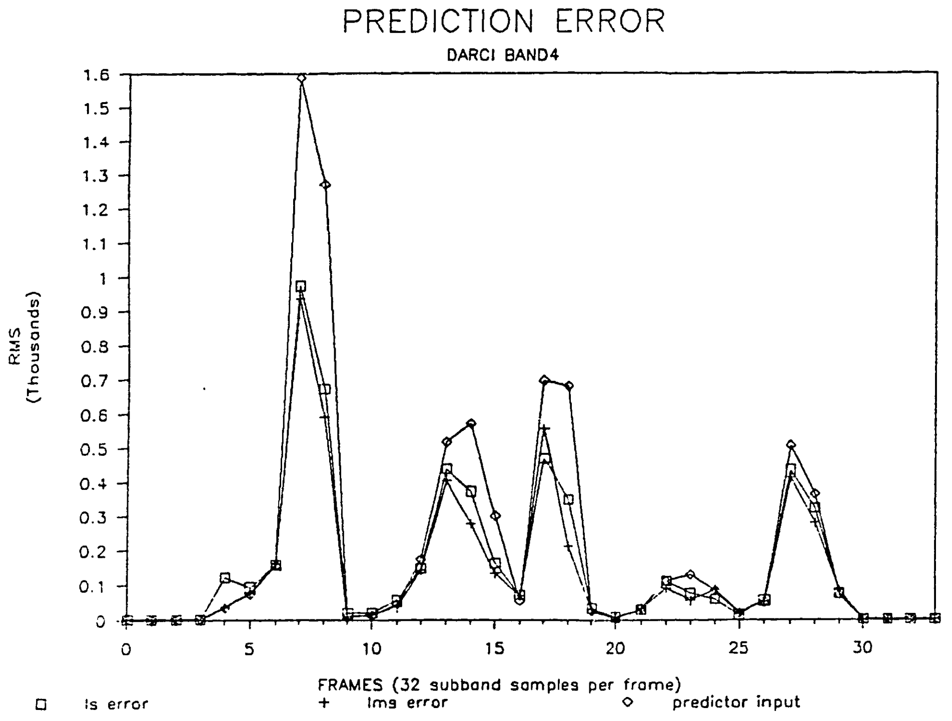
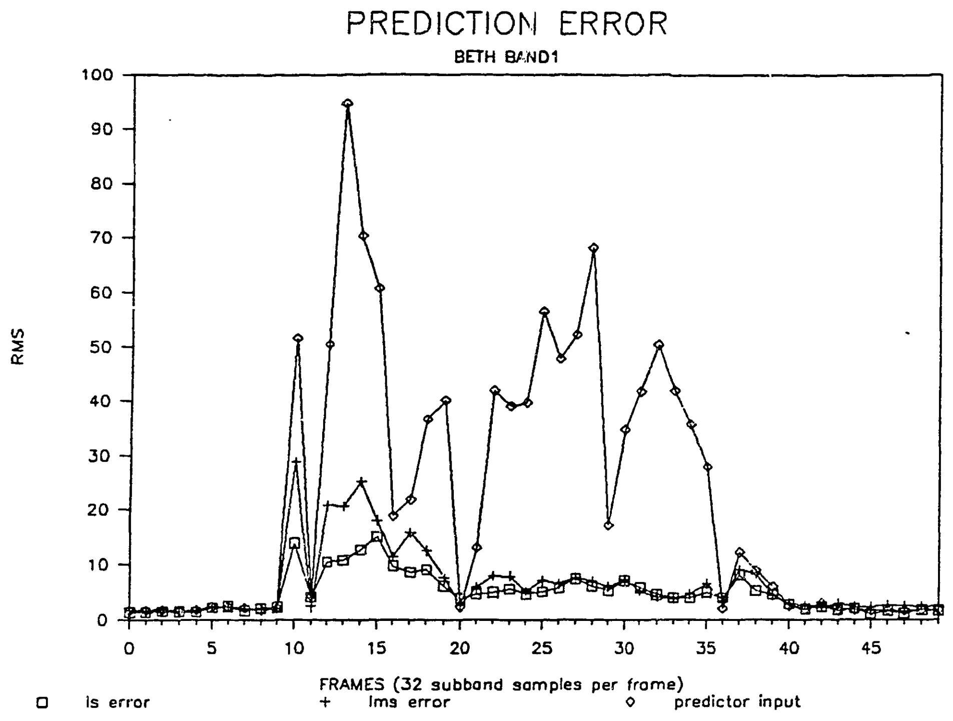


Figure D-1. Frame-to-Frame Prediction Error : DARCI (continued)



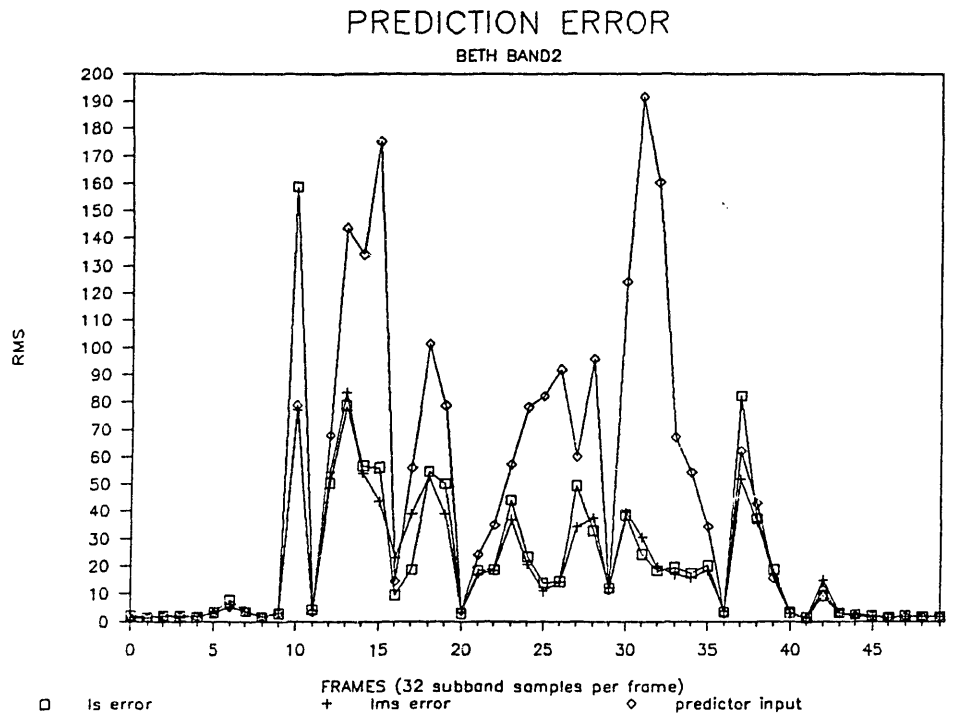
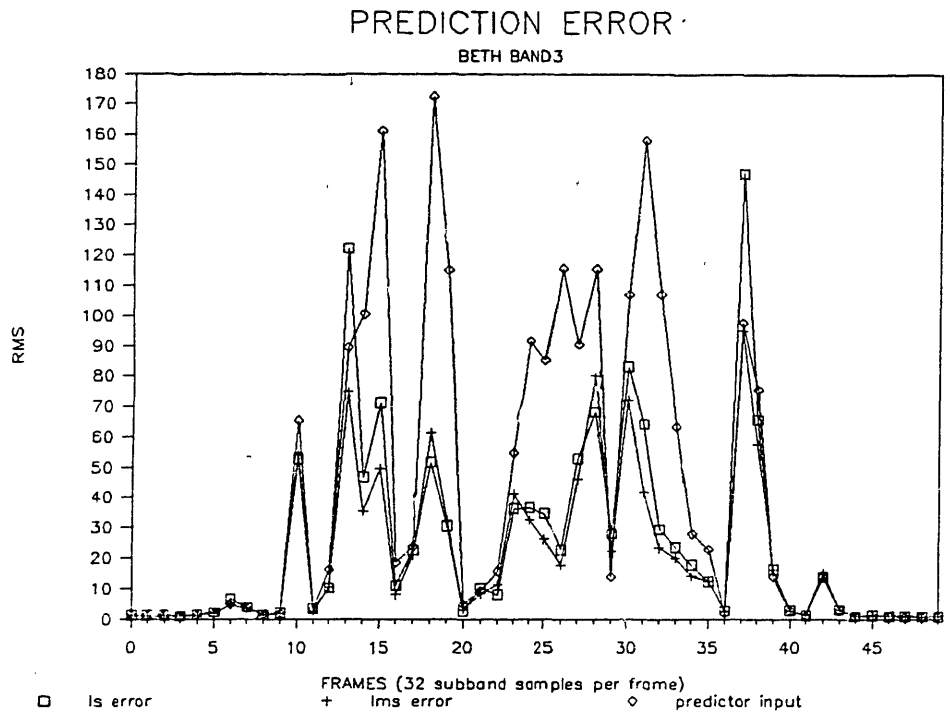


**Figure D-2. Frame-to-Frame Prediction Error : BETH**



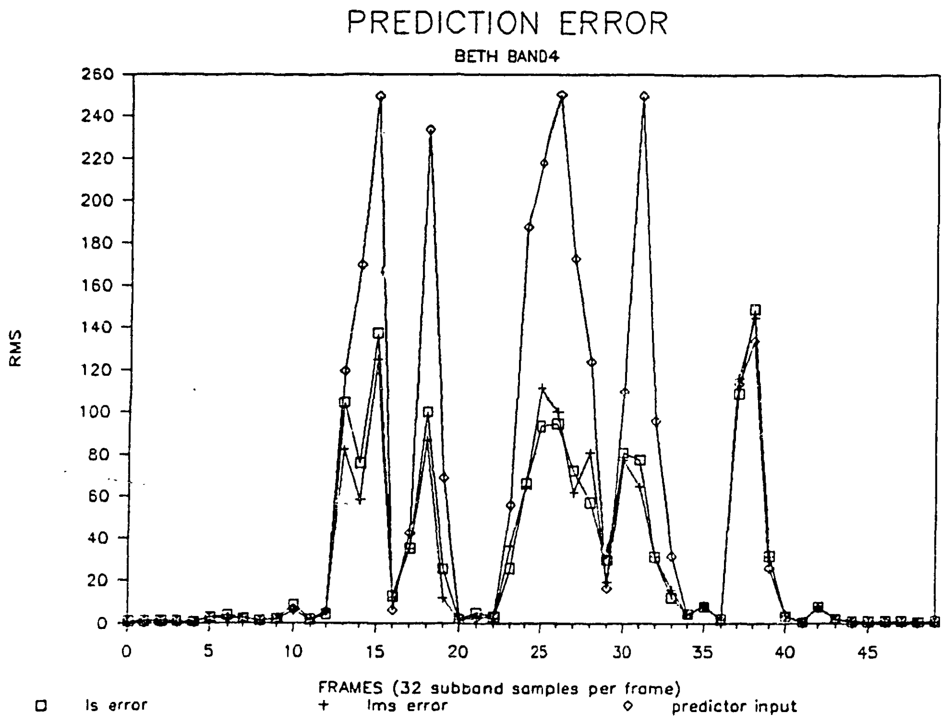
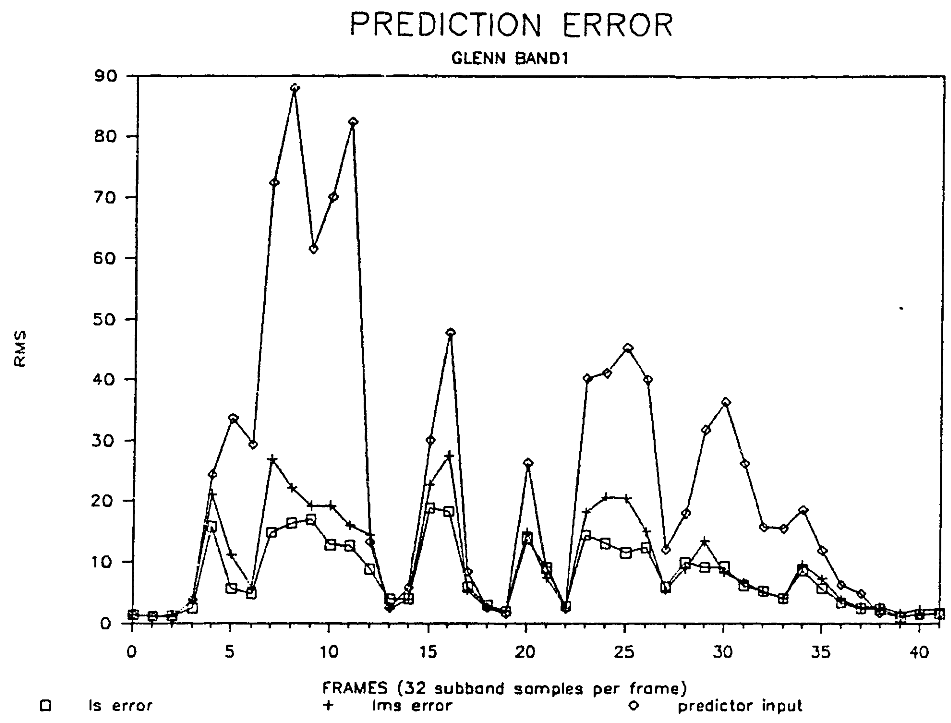


**Figure D-2. Frame-to-Frame Prediction Error : BETH (continued)**



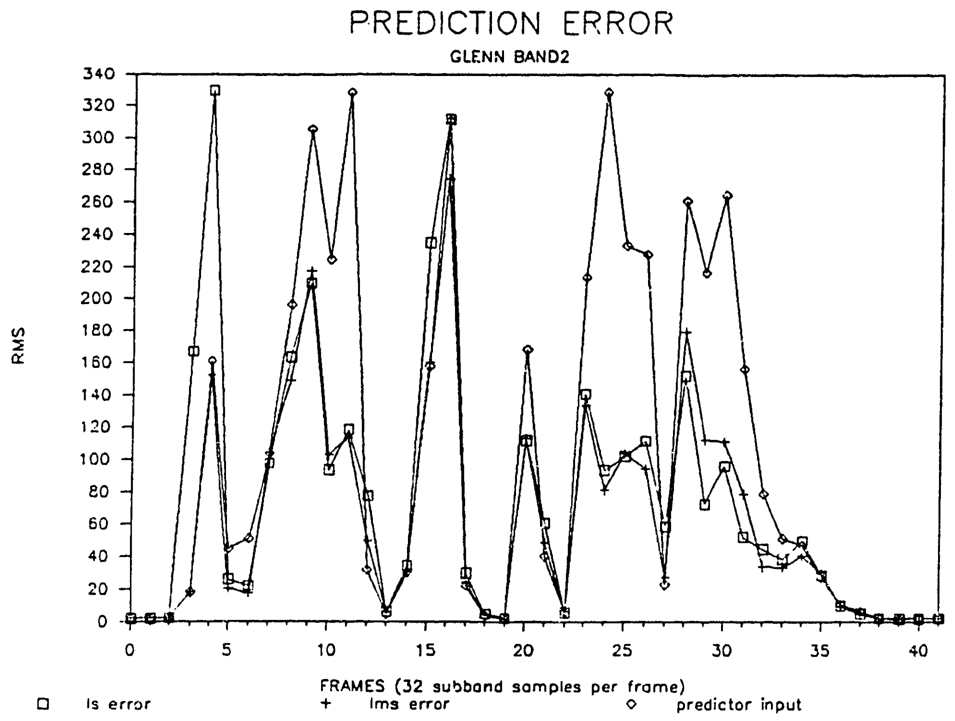
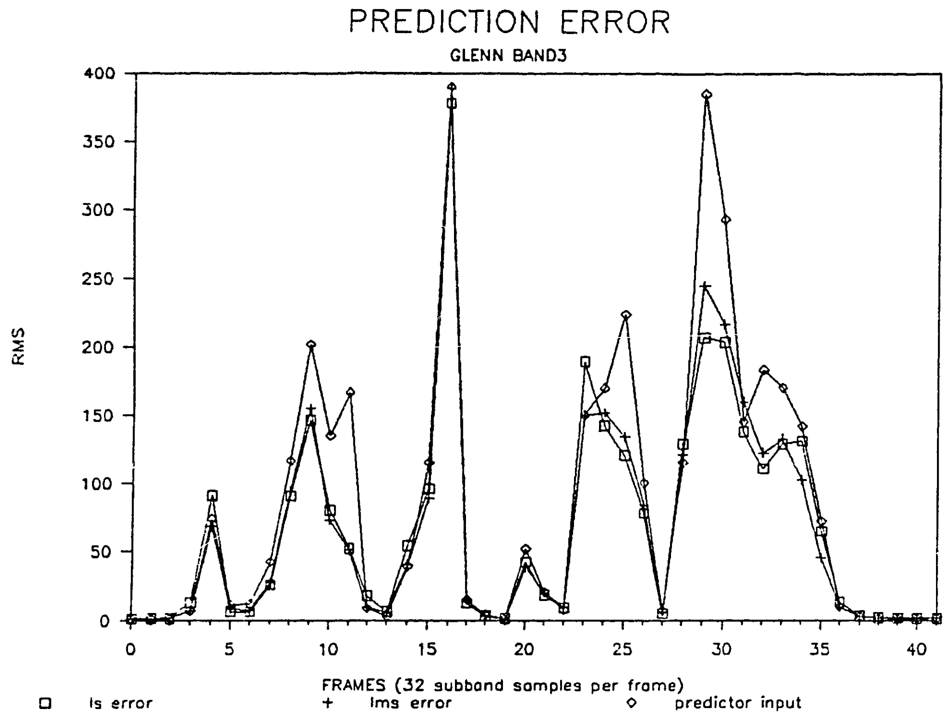


**Figure D-3. Frame-to-Frame Prediction Error : GLENN**



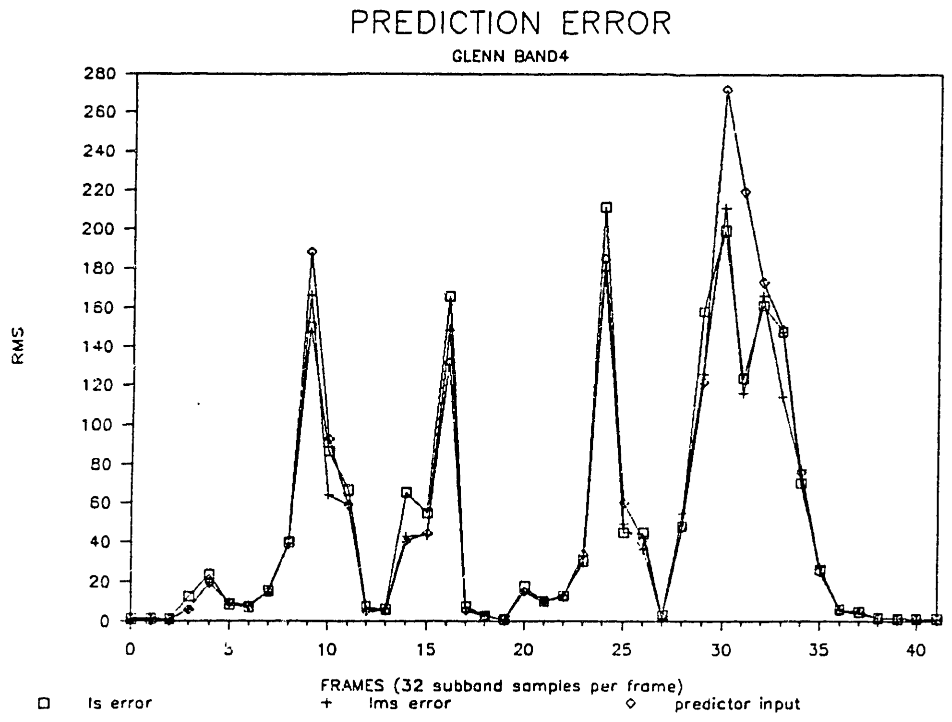
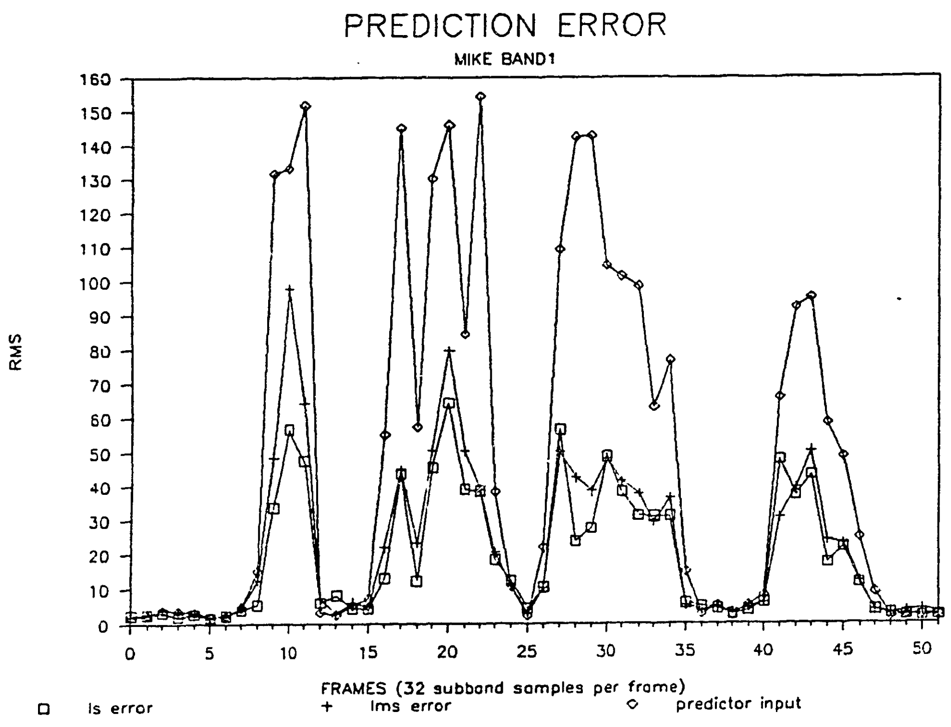


**Figure D-3. Frame-to-Frame Prediction Error : GLENN (continued)**



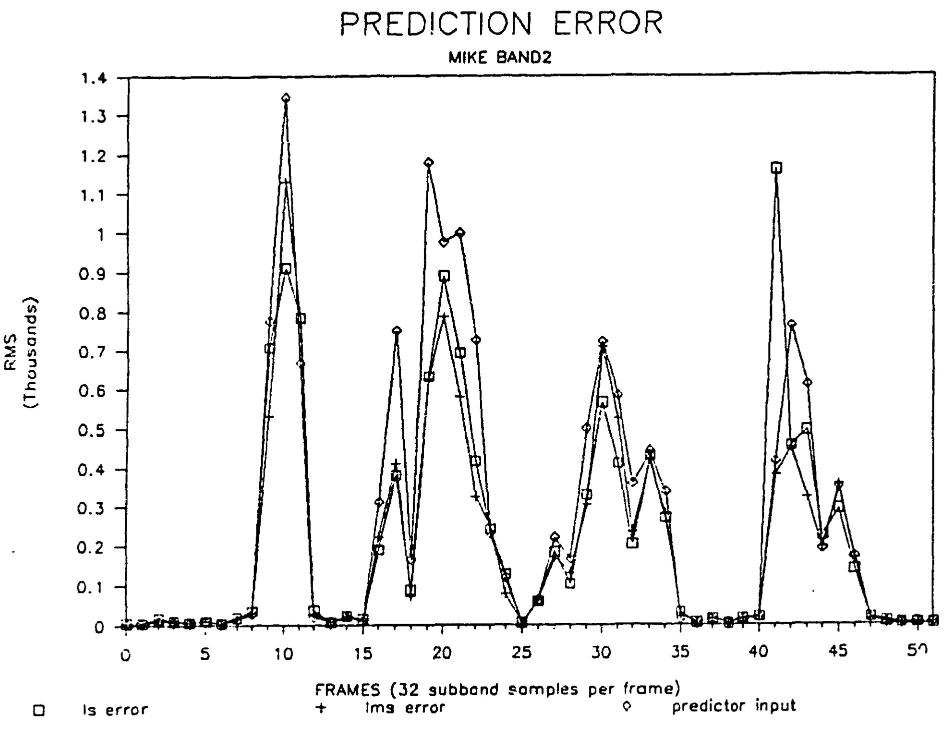
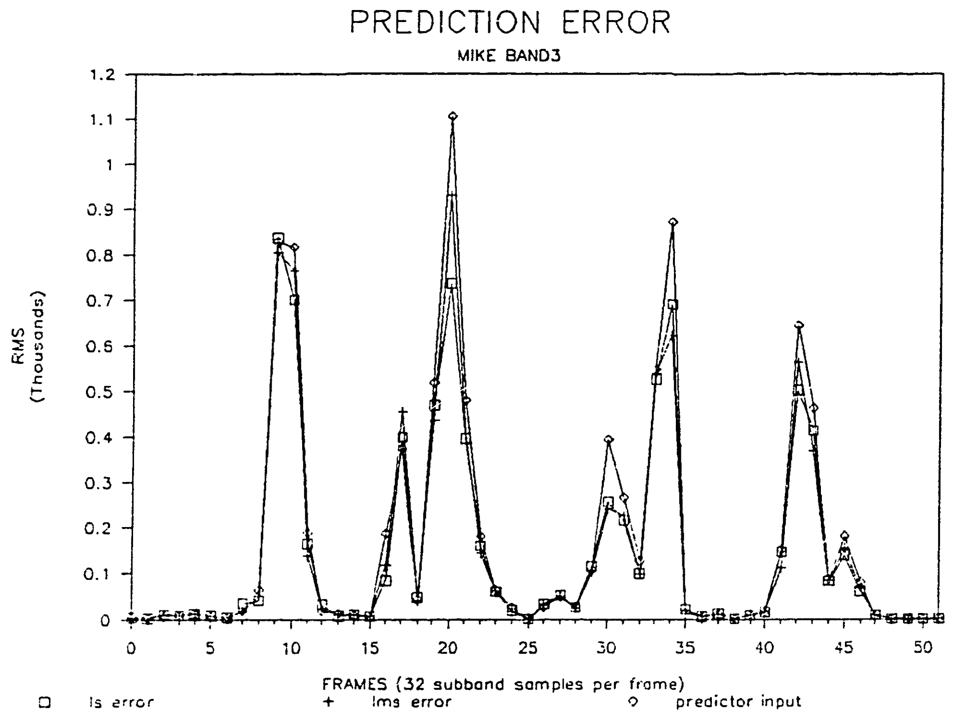


**Figure D-4. Frame-to-Frame Prediction Error : MIKE**



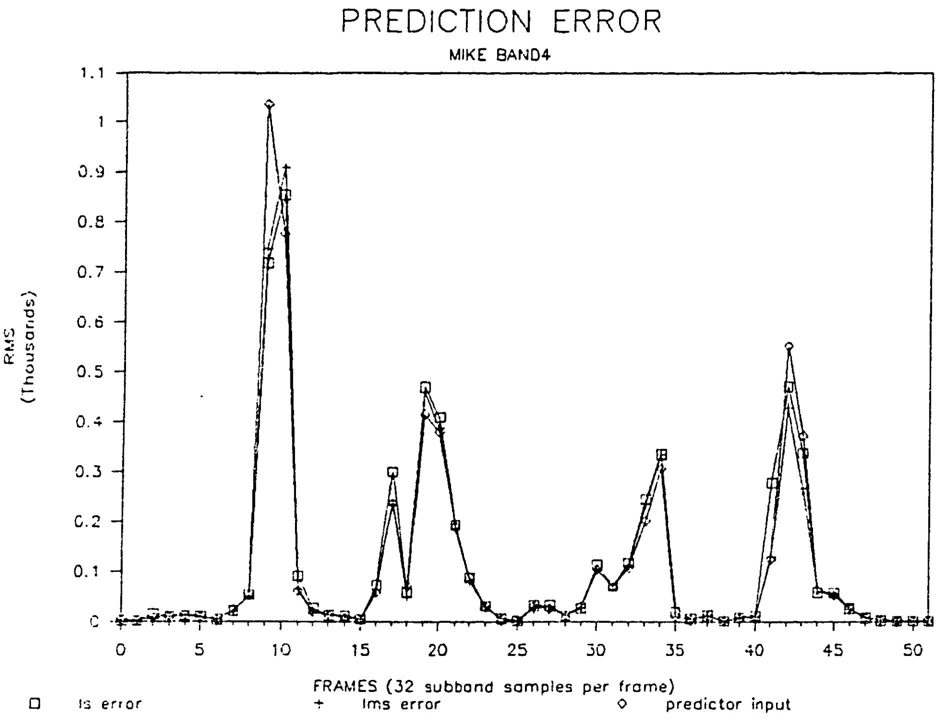


**Figure D-4. Frame-to-Frame Prediction Error : MIKE (continued)**

## D.3 SNR AND SSNR PERFORMANCES

The following tables list objective measures of voice quality for the four sentences as processed by different versions of the sub-band coder. This data is graphed in Figs. 3-4 to 3-7.

| | | 16 kbps | | 24 kbps | |
|---|---|---|---|---|---|
| Predictor | Order | SNR(dB) | SSNR(dB) | SNR(dB) | SSNR(dB) |
| none | 0 | 7.94 | 4.55 | 14.34 | 9.63 |
| LMS | 1 | 7.73 | 4.35 | 14.00 | 9.44 |
| | 6 | 8.80 | 4.92 | 14.34 | 9.82 |
| | 10 | 8.81 | 4.77 | 14.51 | 9.61 |
| | 15 | 6.43 | 1.08 | 14.13 | 7.46 |
| LS | 1 | 8.15 | 4.50 | 14.46 | 9.48 |
| | 6 | 8.58 | 3.95 | 14.76 | 9.17 |
| | 10 | 8.60 | 3.62 | 14.43 | 8.62 |
| | 15 | 8.27 | 2.94 | 14.56 | 8.26 |

Table D-5. SNR and SSNR Performances (DARCI)

| | | 16 kbps | | 24 kbps | |
|---|---|---|---|---|---|
| Predictor | Order | SNR(dB) | SSNR(dB) | SNR(dB) | SSNR(dB) |
| none | 0 | 11.81 | 8.36 | 16.68 | 12.64 |
| LMS | 1 | 11.39 | 8.27 | 16.71 | 12.74 |
| | 6 | 13.25 | 8.75 | 17.07 | 13.02 |
| | 10 | 13.06 | 8.74 | 16.85 | 12.97 |
| | 15 | 12.70 | 8.41 | 16.62 | 12.79 |
| LS | 1 | 11.89 | 8.46 | 16.77 | 12.80 |
| | 6 | 12.96 | 8.56 | 17.19 | 12.99 |
| | 10 | 13.09 | 8.29 | 17.04 | 12.76 |
| | 15 | 13.00 | 8.01 | 17.06 | 12.70 |

Table D-6. SNR and SSNR Performances (BETH)

| Predictor | Order | 16 kbps SNR(dB) | 16 kbps SSNR(dB) | 24 kbps SNR(dB) | 24 kbps SSNR(dB) |
|---|---|---|---|---|---|
| none | 0 | 8.39 | 7.60 | 12.70 | 12.01 |
| LMS | 1 | 7.82 | 7.31 | 12.53 | 11.77 |
| | 6 | 8.39 | 7.81 | 12.75 | 12.15 |
| | 10 | 8.47 | 7.84 | 12.84 | 12.26 |
| | 15 | 8.29 | 6.83 | 12.54 | 11.56 |
| LS | 1 | 8.24 | 7.55 | 12.57 | 11.95 |
| | 6 | 8.02 | 7.50 | 12.55 | 11.92 |
| | 10 | 8.08 | 7.15 | 12.46 | 11.66 |
| | 15 | 6.96 | 6.43 | 12.46 | 11.72 |

Table D-7. SNR and SSNR Performances (GLENN)

| Predictor | Order | 16 kbps SNR(dB) | 16 kbps SSNR(dB) | 24 kbps SNR(dB) | 24 kbps SSNR(dB) |
|---|---|---|---|---|---|
| none | 0 | 8.60 | 6.87 | 12.43 | 11.01 |
| LMS | 1 | 8.38 | 6.60 | 12.69 | 10.95 |
| | 6 | 8.66 | 6.84 | 12.63 | 11.19 |
| | 10 | 8.43 | 6.59 | 12.40 | 11.02 |
| | 15 | 8.13 | 5.63 | 8.36 | 8.50 |
| LS | 1 | 8.76 | 6.78 | 12.89 | 11.08 |
| | 6 | 8.74 | 6.66 | 12.97 | 10.91 |
| | 10 | 8.73 | 6.31 | 12.70 | 10.72 |
| | 15 | 8.40 | 6.09 | 12.41 | 10.55 |

Table D-8. SNR and SSNR Performances (MIKE)