

To the Graduate Council:

I am submitting herewith a thesis written by Olufemi Adebola Adeyemi, entitled "Analysis and Comparison of Three Techniques for the Compression of Digital Instrumentation Data". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Bruce W. Bomar

Bruce W. Bomar, Major Professor

We have read this thesis
and recommend its acceptance:

Ray Joseph
Monty Smith

Accepted for the Council:

Corninkel

Vice Provost
and Dean of The Graduate School

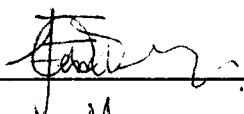
STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the library shall make it available to borrowers under rules of the library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____


May 1991

**Analysis and Comparison of Three
Techniques for the Compression
of Digital Instrumentation Data**

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Olufemi A. Adeyemi

August 1991

*Dedicated to the sweet memories of Bunmi,
who started a great race
but did not finish ...
Atinuke will ...*

ACKNOWLEDGMENTS

The author wishes to express his appreciation to his advisor and major professor, Dr. Bruce Bomar, for his guidance, suggestions and advise during the course of this graduate program. The author would also like to acknowledge the technical assistance of Dr. Roy Joseph and Dr. L.M. Smith, both members of the thesis committee.

Appreciation is also expressed to my family and my friends for their encouragement and support during my study.

The work reported herein was supported by the Arnold Engineering Development Center (AEDC), Air Force Systems Command (AFSC), through The University of Tennessee Space Institute (UTSI) Support Contract (Work Order A91W-11 under Basic Ordering Agreement A91B-01) with Sverdrup Technology Inc., AEDC Group, operating contractor of the Engine Test Facility (ETF), AEDC, AFSC, Arnold Air Force Base, TN, under Project DC97EW.

ABSTRACT

When instrumentation signals are acquired and recorded digitally it is desirable to minimize the data storage space required. This thesis analyzes and compares three methods which were thought to be promising for compressing bandlimited smoothly varying instrumentation data with stationary characteristics over some block length to minimize the quantity of digital data which must be recorded.

The first method considered adaptively varies the sample rate by factors of two using real-time fast Fourier transforms (FFTs) to determine the minimum rate that captures the equivalent spectral components of the signal. To test the procedure, representative simulated and test signals were adaptively sampled and then restored to their original sample rate to examine the amount of data compression achieved and the accuracy of the restoration. The restoration accuracy is examined as a function of the threshold used in the frequency domain to define the portion of the spectrum with significant information content.

The second method records portions of the FFTs of the original digital signal. The portion of each FFT that contains spectral components above a threshold which defines the part of the spectrum with significant information content is stored. Signals were recorded in this way to examine the data compression achieved and were restored to the time domain to determine the accuracy of the restoration as a function of the frequency domain threshold.

The third method estimates the current value of the signal by a linear prediction from previous signal samples. The difference between this estimate and the actual sample is then coded with a Huffman coding scheme and stored to achieve

storage reduction. The decoded difference signal can then be added to the predicted signal to reproduce the input signal with no error within the accuracy of the analog-to-digital converter.

All three data compression methods were compared with respect to the accuracy of reconstruction and the amount of compression attained. It was found that the method of linear prediction gives the best overall result with storage space reduction of about 2:1 on representative test signals. An important factor of merit for the third method is its ability to perfectly reproduce a signal with effectively no error within the accuracy of the analog-to-digital converter.

TABLE OF CONTENTS

CHAPTER	PAGE
INTRODUCTION	1
I. ADAPTIVE DATA SAMPLING	4
1.1 Introduction	4
1.2 ADS Implementation	4
1.3 Signal Restoration	9
1.3.1 Digital Filter Design	11
II. TRANSFORM CODING	14
2.1 Introduction	14
2.2 Transform Coding Implementation	15
2.3 Signal Reconstruction with TC	19
III. LINEAR PREDICTIVE CODING	21
3.1 Introduction	21
3.2 LPC Implementation	21
3.3 Signal Reconstruction with LPC	30
IV. RESULTS	31
4.1 Introduction	31
4.2 Signals Analyzed	31
1. Simulated Signals	32
2. Actual Signals	32
4.3 Adaptive Data Sampling scheme	33
4.4 Transform Coding Scheme	34
4.5 Linear Predictive Coding Scheme	41
V. DISCUSSION AND CONCLUSION	55
REFERENCES	62

VITA	64
----------------	----

.

FIGURE	LIST OF FIGURES	PAGE
1.1	Adaptive Sample Rate System.	5
1.2	Adaptive Sample Rate Technique.	7
1.3	Midpoint Interpolation.	10
1.4	Filter Response for Linear Interpolation.	13
2.1	Frequency Response of Representative Signal.	18
3.1	Model of LPC scheme using FIR Filter.	26
3.2	Huffman Coding Algorithm.	29
4.1	A Comparison of the Peak Errors versus Threshold for both Pure and Noisy Sine Waves using TC technique at 50% Overlap.	35
4.2	Compression Ratio versus Threshold for the first set of three Data files using TC technique at 50% Overlap.	36
4.3	Compression Ratio versus Threshold for the second set of three Data files using TC technique at 50% Overlap.	37
4.4	Peak Error versus Threshold for the first set of three Data files using TC technique at 50% Overlap.	38
4.5	Peak Error versus Threshold for the second set of three Data files using TC technique at 50% Overlap.	39
4.6	Peak Errors versus Overlap at different Threshold values T for DAT.1 file using TC technique.	40

4.7	Theoretical values of CMPR versus Overlap for TC	42
4.8	Plot of a typical CMPR versus Overlap for Data file DAT.1.	43
4.9	Compression Ratio versus Model Order for Data file DAT.4.	45
5.1	Modified Gaussian Distribution for Peak Error Control Using LPC Scheme.	57

TABLE	LIST OF TABLES	PAGE
I.	Sample Rates and Maximum Signal Bandwidth.	8
II.	Results of LPC scheme on Noisy Sine Waves.	46
III.	Results of LPC scheme on DAT.1.	49
IV.	Results of LPC scheme on DAT.2.	52

INTRODUCTION

Data compression techniques have been widely used to reduce the storage required for digitally recorded speech and images [1]-[7]*. There is now an increasing interest in applying these techniques to the compression of bandlimited smoothly varying instrumentation signals, such as digitized vibrational and strain gauge data or other transducer outputs that exhibit sinusoidal behaviour [8]. In speech and image compression, extreme accuracy is not of great importance since intelligible speech and similar images are acceptable. With instrumentation signals, however, this is not the case, since often errors greater than 1% of full scale are not acceptable.

This thesis analyzes and compares three methods of data compression that were thought to be promising for instrumentation data compression. Two of these, transform coding and linear predictive coding, have been widely used for speech and image compression [1]-[4]. The third, that of adaptively varying the sample rate, has been used primarily for instrumentation signal compression [7]. In this thesis, the three techniques are implemented and applied to both simulated and actual instrumentation data. The three techniques are compared primarily with regard to their compression performance and the peak error (due to compression only) of the signal reconstructed from the compressed data.

In Chapter I, a method of adaptively varying the sample rate by a factor of two uses the real-time fast Fourier transform (FFT) to determine the smallest sample rate. Although the process of digitally converting the sample rate of a signal from one rate to another has received considerable attention [9], the process of adaptively sampling a signal has apparently received very little attention. Page and Nelson [7] presented a method for adaptively selecting one of two sample rates

* Numbers in brackets refer to similarly numbered references in the References.

which differ by a factor of 64. However, no method is presented for converting back to a uniform sample rate. The method considered in this thesis is an extension of the method of [7] that permits the dynamic selection of a sample rate which is 1, 2, 4, 8, 16, 32, or 64 times lower than the maximum rate.

With adaptive sampling rate, the signal is initially over-sampled to guarantee that the Nyquist criterion (sampling frequency greater than twice the highest frequency content of signal) is met or exceeded. Overlapping blocks of samples are then fast Fourier transformed to determine the sample rate that is actually needed to represent the samples in the middle of the transformed block. The output sample rate is then lowered when possible by subsampling to achieve the lowest sample rate that maintains signal integrity. Representative simulated and actual test signals are adaptively sampled and then restored to their original sample rate to examine the amount of data compression achieved and the accuracy of the restoration. The restoration accuracy is examined as a function of the threshold used in the frequency domain to define the portion of the spectrum containing significant information. The restoration to a uniform sample rate from an adaptively sampled signal is accomplished through successive interpolation of the midpoints between existing samples in the time domain with repeated use of a polyphase subfilter obtained from a half-band interpolation filter [9].

In Chapter II the Transform Coding technique is discussed. It involves a linear transformation of input samples to form a space of transformed coefficients. Only transform coefficients that exceed a threshold (based on a percentage of the largest spectral component magnitude) are retained and recorded to achieve data compression. The success of the transform coding technique in achieving compression depends on the degree to which the signal can be represented by fewer dominant components in the transform domain. The restoration operation involves an in-

verse transformation of the compressed transformed sample block back to the time domain.

The transform used in this thesis is the Discrete Fourier Transform (DFT) with a block size of 1024. Pearl [10] has shown that for a long block length, ($N > 128$), the DFT approaches the performance of the optimum Karhunen-Loeve Transform (KLT). Although KLT gives the best performance in Transform Coding, it has the disadvantage of lacking, a fast algorithm for its computation. For the KLT, the transformation matrix is signal-dependent, and the computation of its elements is not simple. The DFT was selected because it is signal-independent and has a fast algorithm, thus permitting a substantial saving in computing time and allowing for real-time implementation on a low-cost single-chip digital signal processor.

Chapter III deals with data compression using Linear Predictive Coding (LPC). Here the predictor uses previous samples in a predictive filter which is used to predict the present value of the input signal. The difference between this estimate and the actual value of the signal is coded and stored. At the decoding end, a similar predictor uses the decoded difference to reconstruct the original signal. Efficient coding of the difference is essential in achieving the lowest possible storage for a given signal. Therefore the optimum Huffman Coding scheme [11] is used in coding this difference because it gives an average code word length that is close to the entropy of the difference. Although Witten *et al* [12] point out that Arithmetic Coding gives better compression, Huffman coding is easy to implement and fast. Furthermore, it is found that for typical instrumentation signals, Huffman coding achieves coding efficiencies within 0.1 bit/sample of the entropy of the differences being coded so the improvement with arithmetic coding would be negligible.

The results of the three methods are shown and compared in Chapter IV. Finally, discussions and conclusions appear in Chapter V.

CHAPTER I

ADAPTIVE DATA SAMPLING

1.1 Introduction

Instrumentation signals are sampled to meet or exceed the Nyquist criterion for signal reconstruction. With adaptive sampling, the sampling rate is adjusted to the individual data set characteristics. Where significant detail is present, the input signal is sampled at a faster rate. Sampling at a much slower rate can be utilized in areas with no significant detail. Thus, Adaptive Data Sampling (ADS) minimizes the storage and maximizes the recording time of information without significantly compromising signal integrity. The ADS method presented in this chapter continuously estimates the input signal's bandwidth and dynamically selects sample rates which are 1, 2, 4, 8, 16, 32 or 64 times lower than the maximum rate. If a lower sample rate is selected, samples are discarded in order to compress the output data.

1.2 ADS Implementation

Fig. 1.1 depicts the data sampling technique. Data from the Analog-to-Digital Converter (A/D) is buffered and Fourier transformed in overlapping blocks of 1024-points with an overlap of 50%. The spectrum from each block is then used to make a sample rate decision concerning the middle 512 samples of the block, based on the distribution of energy in the spectrum relative to the total energy in the spectrum. If no sample rate reduction is possible, all 512 samples are output; but whenever a sample rate reduction by a factor M is possible, every M^{th} sample is output giving an output block which can be as small as 8 samples.

Because of the windowing [13] which must be performed in fast Fourier

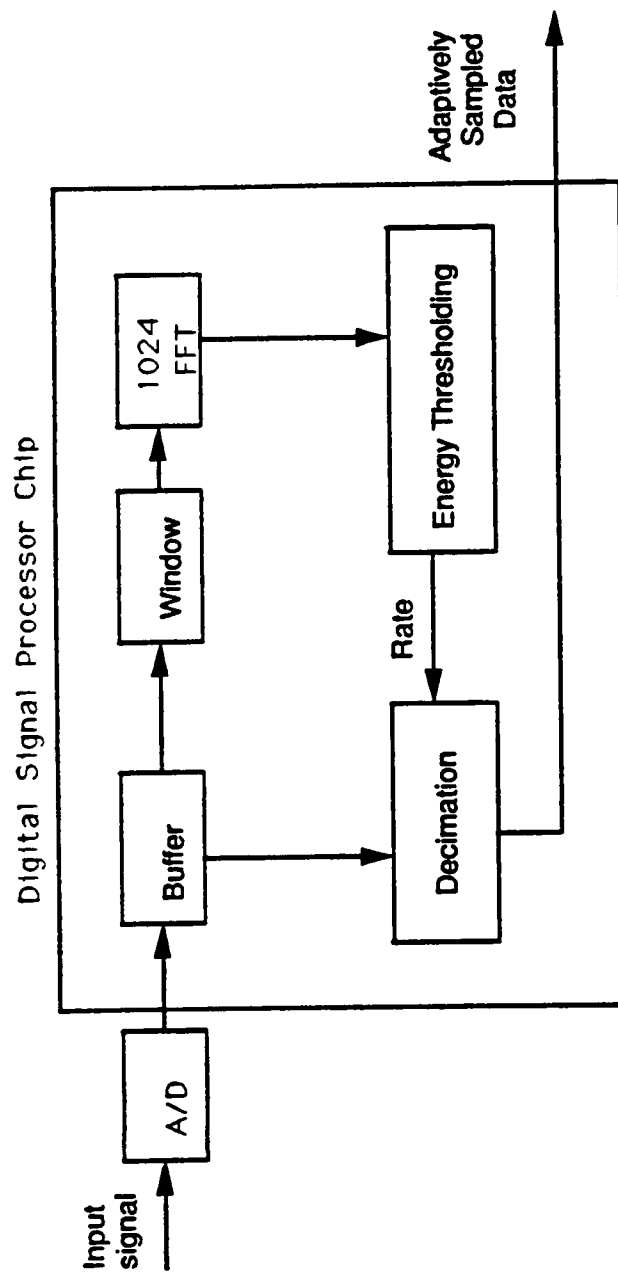


Fig. 1.1 Adaptive Sample Rate System

transform spectral analysis, data from the A/D is processed in overlapping blocks with an overlap of 50%. This window is applied to the data to reduce the discontinuity at the block ends due to the transform's periodic extension, thus minimizing spectral leakage. Despite the windowing, the 512 samples in the middle of the block, which are output at a rate based on Fourier transform's result, contribute strongly to the new sample rate as shown in Fig. 1.2.

A real analog signal $x(t)$ bandlimited to f_a kHz, sampled at a rate of f_s kHz and windowed with a sequence $w(n)$ yields a real sequence $x(n)$, where n is the time-sequence index. A 1024-point real-data FFT of the sequence produces 513 positive-frequency-domain values $X(k)$, $k = 0, 1, 2, \dots, 512$ spaced at $\frac{f_s}{1024}$ kHz from 0 to $\frac{f_s}{2}$ kHz, where k is the frequency-sequence index.

If the original signal has been oversampled to completely avoid aliasing, the oversampling factor is given as

$$\eta_o = \frac{f_s}{2f_a}. \quad (1.1)$$

By Parseval's relation, the total energy in the discrete time sequence $x(n)$ is

$$E_t = \frac{1}{1024} \left[2 \sum_{k=1}^{511} |X(k)|^2 + |X(0)|^2 + |X(512)|^2 \right]. \quad (1.2)$$

At different points in the signal spectrum, the total energy from dc to that point can be calculated and compared with the total energy E_t to determine the output sample rate.

Thus, the bandwidth of the samples in each block is determined by the energy ratio

$$E(m) = \frac{1}{1024} \frac{\left[2 \sum_{k=1}^{P(m)} |X(k)|^2 + |X(0)|^2 \right]}{E_t} \quad (1.3)$$

where $P(m)$ takes on the values shown in Table I.

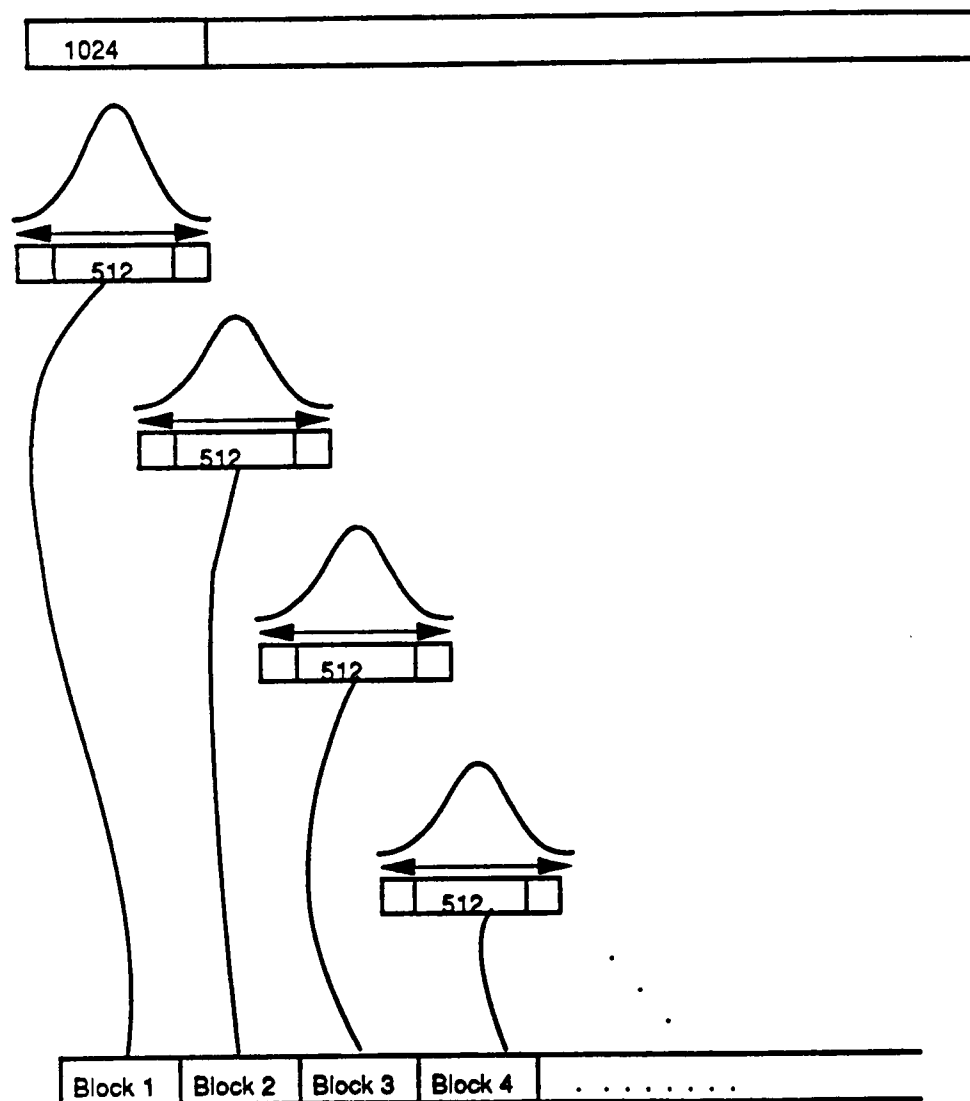


Fig. 1.2 Adaptive Sample Rate Technique

The signal bandwidth is determined by finding the m for which

$$E(m) > T$$

where T is a specified energy threshold between 0 and 1. Notice that if the sample rate is reduced by 64, the highest value of k for which $|X(k)|$ can be significant is $k = 8$. However, from Table I the signal bandwidth is only allowed to be $k = 6$ in this case to retain some oversampling. This is necessary to provide for a transition band in the interpolation filter used to restore the signal to its original sample rate. In all cases in Table I an oversampling ratio of at least 1.28 is retained.

Table I. Sample Rates and Maximum Signal Bandwidth

m	$P(m)$	Oversampling Factor η_o	Output Sample Rate, f'_s
1	6	$\frac{8}{6} = 1.33$	$\frac{f_s}{64}$
2	12	$\frac{16}{12} = 1.33$	$\frac{f_s}{32}$
3	25	$\frac{32}{25} = 1.33$	$\frac{f_s}{16}$
4	50	$\frac{64}{50} = 1.28$	$\frac{f_s}{8}$
5	100	$\frac{128}{100} = 1.28$	$\frac{f_s}{4}$
6	200	$\frac{256}{200} = 1.28$	$\frac{f_s}{2}$
7	400	$\frac{512}{400} = 1.28$	f_s

A typical value of the energy threshold T is 0.9. This value includes 90% of the input energy in the band from dc to $P(m)$.

A good window function is important in implementing the FFT spectral anal-

ysis to ensure accurate energy calculation in (1.2) and (1.3).

The N=1024 Hamming Window given by

$$w(n) = 0.54 - 0.46 \cos \left(\frac{2\pi n}{N} \right) \quad (1.5)$$

where $n = 0, 1, 2, \dots, N - 1$ is chosen for this application because of its narrow main-lobe width, low side lobes, and efficient application in time and frequency domains [13].

1.3 Signal Restoration

Uniform sample rate restoration from an adaptively sampled signal can be accomplished through successive interpolation that produces samples at the midpoint between existing samples. When a block is brought into memory on a signal processor or computer, memory locations are reserved for all samples that will exist at the sample rate to which the signal is being converted (maximum sample rate). Those portions of the adaptively sampled signal which are sampled at less than the maximum sample rate, i.e., $(\frac{1}{M}f_s)$ for $M > 1$, are placed in every M^{th} location in memory. Then, starting at the lowest sample rate in the block, samples are interpolated at the midpoint between existing samples unless the midpoint sample already exists, in which case the existing sample is retained. This process continues until all the memory locations (reserved for samples at maximum sample rate) are filled up.

The operation which must be performed accurately for signal reconstruction is the interpolation of midpoint samples. Consider what happens if zero-value samples are initially assumed at the midpoints. The effect of the zero insertion on the spectrum is the creation of a mirror image of the original spectrum with an amplitude reduction of 2 as shown in the example of Fig. 1.3. To remove the

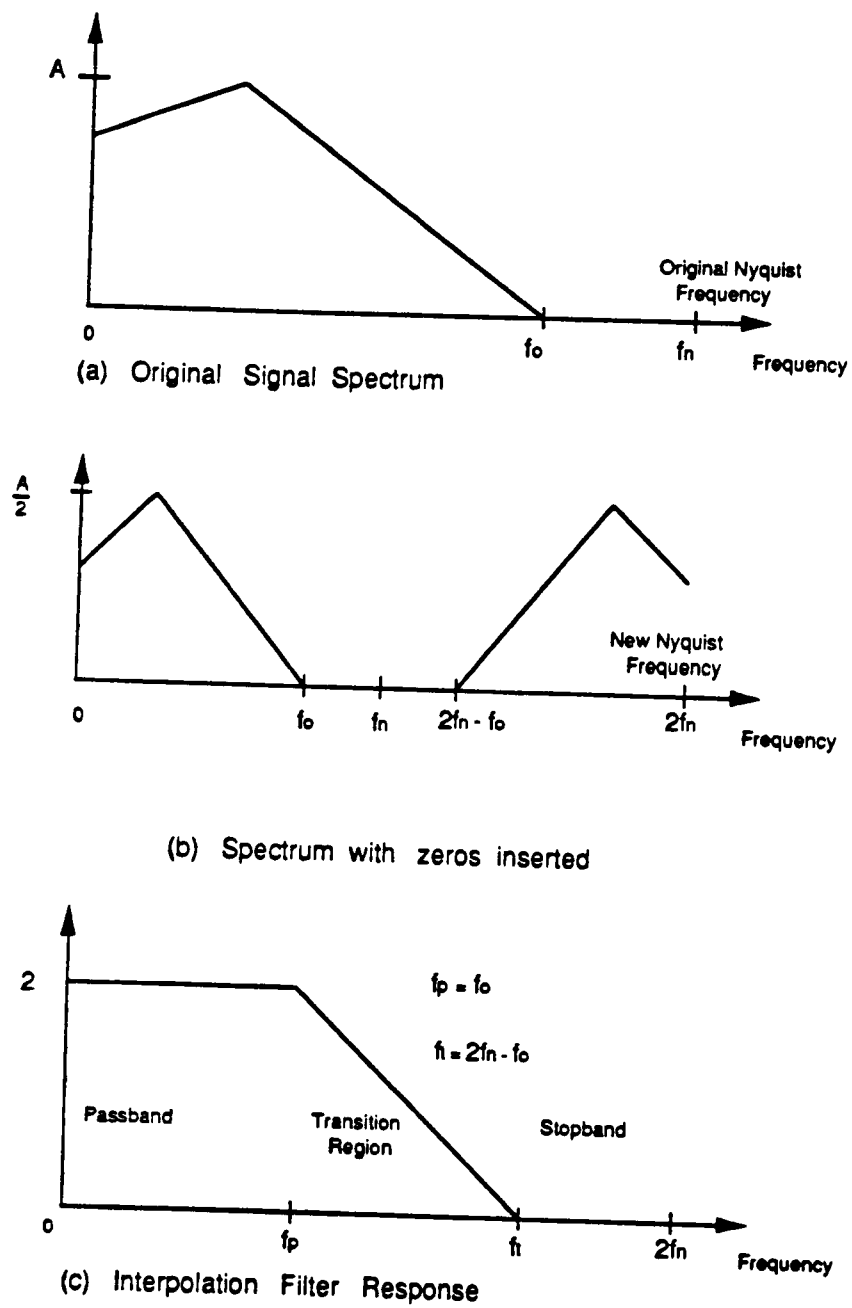


Fig. 1.3 Midpoint Interpolation

image and increase the amplitude, we need to lowpass-filter the signal with a digital filter having the response of Fig. 1.3(c) [14]. The zero insertion at midpoint, followed by lowpass filtering, reduces to computing the interpolated sample at location n using

$$x(n) = \sum_{l=0}^{L-1} g(l)x\left[n - \left(\frac{L-1}{2} - l\right)M\right] \quad (1.7)$$

where $g(l)$ are the filter coefficients, L is the filter order and M is the distance between original samples before interpolation.

The filter coefficients are the first polyphase subfilter [9, pp 79-85] of a finite impulse response half-band filter [9, pp 155-185] designed to have the response of Fig. 1.3(c). The second polyphase subfilter is simply a delayed unit sample function that retains the already-available samples.

1.3.1 Digital Filter Design

The filter coefficients $g(l)$ are readily obtained using a computer program by Parks and McClellan which is listed in the book by Parks and Burrus [15, pp 288-301] for designing linear phase filters based on Chebyshev approximation using the Remez exchange algorithm. The filter order must be $2L - 1$ to obtain a half-band filter, where L is the number of terms desired in (1.7). Filters with larger filter order (L) give better filter response but require more computation to implement the interpolation. The filter band edges and gain are as shown in Fig. 1.3c. Equal weighting of the passband and stopband errors in the Chebyshev approximation is used to obtain a half-band filter.

The output of the program will be $2L - 1$ coefficients which are the impulse response samples, $h(n)$, $n = 1, 2, \dots, 2L - 1$. The first polyphase subfilter is the odd samples of $h(n)$ giving

$$g(l) = h(2l + 1), \quad l = 0, 1, 2, \dots, L - 1. \quad (1.8)$$

Fig. 1.4 gives the response of a typical interpolation filter with $L = 18$. This can be seen to provide an excellent approximation to the desired response of Fig. 1.3(c). As long as the adaptively sampled signal is bandlimited to the passband of the interpolation filter, the filter shown in Fig. 1.4 will provide a virtually perfect reconstruction.

For each successive stage of interpolation by 2 the interpolation filter can be a more efficient, lower-order filter than for the previous stage since the distance from f_o to f_n in Fig. 1.3, and hence filter transition band, is greater. However, to simplify the interpolation process, the filter used for the first stage can also be applied for each additional stage until the whole block in memory is completely filled. The latter approach using a single filter is used in this thesis.

Note that implementing the ADS signal reconstruction on a signal processor with limited memory might prove difficult if the signal is too long. In this case, however, the signal can be reconstructed in overlapping blocks giving the same result as described earlier.

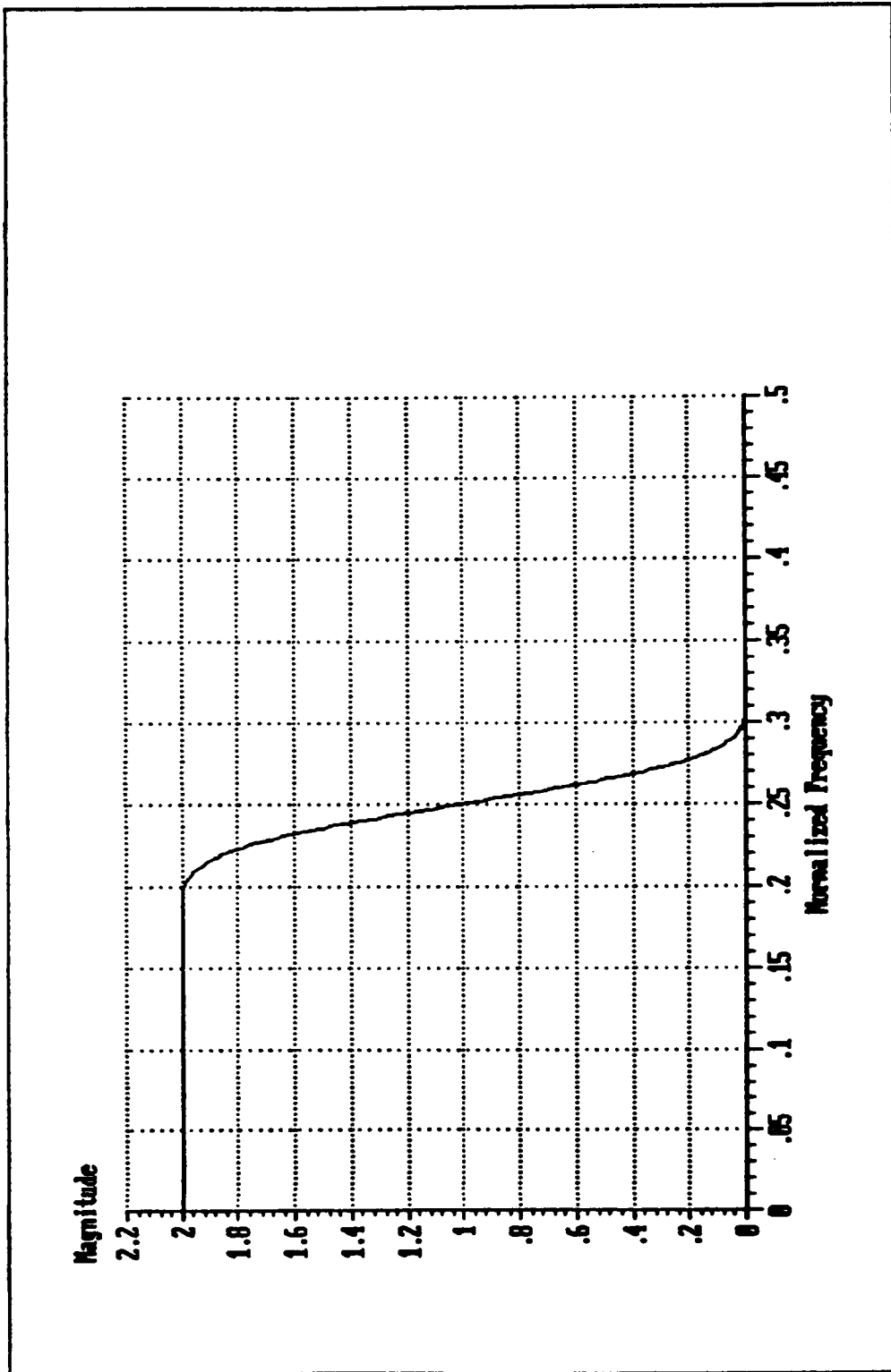


Fig. 1.4 Filter Response for Linear Interpolation

CHAPTER II

TRANSFORM CODING

2.1 Introduction

In Transform Coding (TC), each block of signal samples is transformed into a set of transform coefficients. These coefficients are then quantized independently and only the largest magnitude coefficients are stored. The goal is to store the entire block of data with fewer bits than would be needed to store the original block. Since the transformation used must be reversible, blocks of reconstructed signal samples can be obtained from an inverse transformation of the stored coefficients. Because the stored coefficients are quantized and only the dominant signal spectral characteristics are stored, a portion of the frequency content is lost, consequently, the original signal cannot be recovered exactly.

Application of TC techniques to speech and picture bandwidth compression has received considerable attention. In both applications, the number of bits required to code the transform coefficients depends upon the sensitivity of human hearing and vision to the subjective effect of the quantization error. The *bit allocation* [16] method is effective for these applications because of its ability to quantize every value in the spectral representation with different precision based on the knowledge of human auditory and visual systems. In instrumentation signals however, bit allocation is very difficult to accomplish because the frequency domain characteristics of the instrumentation signal are often not predictable and there is no way of ranking the spectral components by their importance. Therefore, in this thesis all spectral coefficients stored are quantized with the same number of bits.

Given an analog signal $x_a(t)$ bandlimited to f_a kHz and sampled to yield the sequence $x(n)$, transform coding involves the following operations. A block of N

successive samples is arranged as a vector $\mathbf{x}$, and then $\mathbf{x}$ is linearly transformed using a unitary matrix $\mathbf{A}$

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

with $\mathbf{A}^{-1} = \mathbf{A}^H$ where $\mathbf{A}^H$ is the Hermitian transpose of $\mathbf{A}$. The elements of $\mathbf{y}$ are the transform coefficients of the coding scheme.

These elements are independently quantized and subjected to an adaptive thresholding procedure based on a percentage of the largest absolute value in $\mathbf{y}$ to yield a vector $\hat{\mathbf{y}}$. This vector of quantized, thresholded, transform coefficients is stored to represent the block. Using the inverse matrix $\mathbf{A}^{-1} = \mathbf{A}^H$, a discrete time sequence $\hat{\mathbf{x}}(n)$ can be obtained from

$$\hat{\mathbf{x}} = \mathbf{A}^H \hat{\mathbf{y}}$$

where $\hat{\mathbf{x}}$ are the reconstructed discrete time samples.

2.2 Transform Coding Implementation

In implementing the TC technique, the Discrete Fourier Transform (DFT), defined as

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} w(n)x(n)e^{-j\frac{2\pi nk}{N}}, \quad k = 0, 1, 2, \dots, N-1 \quad (2.1)$$

is used because it can be executed with a fast algorithm (FFT) permitting computer implementation in $O(N \log_2 N)$ operations, resulting in a substantial saving of computer time [14]. In (2.1), $w(n)$ is the window function used. Although the DFT is suboptimum for transform coding because the $X(k)$ coefficients are not fully uncorrelated, the performance of the DFT approaches the optimum Karhunen-Loeve Transform (KLT) for block sizes N larger than about 128 [10]. The KLT is

difficult to use because it is signal dependent and in general lacks a fast algorithm for its implementation.

Data from the A/D are buffered and Fourier transformed in overlapping blocks of 1024-points. The maximum coefficient value in the spectrum is determined, and a fraction of this value is used as the threshold. All magnitude values below this threshold are then set to zero, and the remaining frequency domain values are quantized and stored as the compressed data.

For example, a 1024-point real data FFT of a section of input signal from the A/D will produce 513 frequency domain values $X(k)$, $k = 0, 1, \dots, 512$. If the largest FFT coefficient is determined as

$$X_{max} = \max |X(k)|, \quad k = 0, 1, \dots, 512 \quad (2.2)$$

then the threshold value is

$$X_{thr} = TX_{max}, \quad 0 < T \leq 1$$

Reduction in storage is achieved for each block by storing only those coefficients which exceed the threshold and effectively setting those magnitude values below the threshold to zero. Clearly, the lower the number of coefficients stored, the greater the data compression. However, as the number of stored coefficients decreases, so does the quality and integrity of the reconstructed signal. Consequently, a trade-off exists between signal integrity and data compression. Lower values of T reduces compression but gives more accurate reconstructed signals.

Windowing [13] in TC is done so that leakage from dominant spectral lobes does not result in coefficients of significant (above threshold) amplitude far away from the peak which would increase the required storage. The $N=1024$ Hamming Window (1.5, pp. 9) is again used for this application because of its narrow main lobe width and low side lobes.

Fig. 2.1 depicts the frequency spectrum of a representative signal and illustrates the procedure of setting the magnitude values below a threshold to zero.

Thresholding in the frequency domain is effectively filtering with a filter

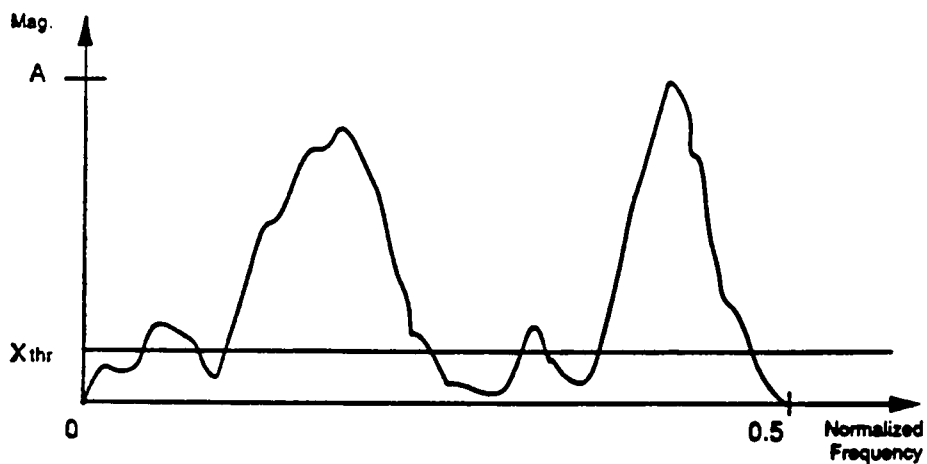
$$H(\omega) = \begin{cases} 1 & \text{where samples are retained} \\ 0 & \text{where no samples are retained} \end{cases} \quad (2.3)$$

which has an N-point unit sample response resulting in circular convolution effects (since implemented via the DFT). Overlapping the data blocks by more samples reduces the circular convolution errors by allowing more points near the ends of the reconstructed block to be discarded. This is because the circular convolution effects are greater near the block ends. However, increasing the overlap also increases the storage required and so reduces the data compression that can be achieved.

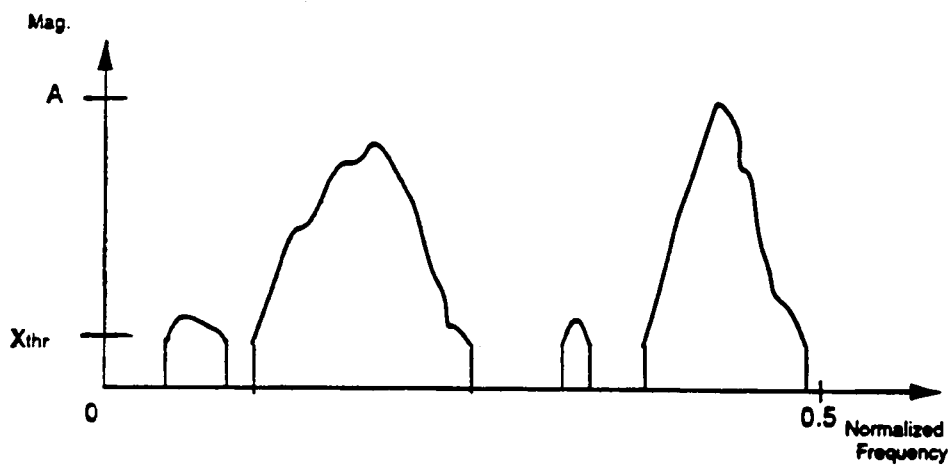
Two methods of storing the thresholded coefficients were considered and found to be comparable for actual signals:

1. Starting at bin $k = 0$ store the value of k for the first coefficient above the threshold followed by the number of consecutive coefficients above the threshold followed by the real and imaginary part of these coefficients. This process is repeated for all groups of coefficients until all above-threshold coefficients have been stored.
2. For all 513 coefficients store a value. If a coefficient is below the threshold store a 0 bit for this value. If a coefficient is above the threshold store a 1 bit followed by the real and imaginary parts of the coefficient.

Consider a situation where the frequency domain representation of a block of size N gives a nearly uniform spectrum. For this case, it is not desirable to use the TC technique because little or no compression will be achieved. Jayant and Noll [17] define a spectral flatness measure γ_x^2 as the ratio of the geometric mean



(a) Frequency Spectrum of typical Representative Signal



(b) Magnitude Thresholding of Representative Signal below T

Fig. 2.1 Frequency Response of Representative Signal

and the arithmetic mean of the samples of the power spectral density.

$$\gamma_z^2 = \frac{\lim_{N \rightarrow \infty} \left[\prod_{k=0}^{\frac{N}{2}} S_k \right]^{\frac{1}{N}}}{\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=0}^{\frac{N}{2}} S_k}$$

where S_k is the power spectral density of the block. The ratio γ_z^2 is by definition, not greater than unity. The value of unity is only obtained if all the samples in a block are equal. The spectral flatness measure is similar to the entropy of the block. If the information content over a block size N in frequency domain is averaged the entropy obtained is given by

$$H = - \sum_{k=0}^{\frac{N}{2}} \frac{|X(k)|}{s} \log \frac{|X(k)|}{s}$$

where $s = \sum_{k=0}^{\frac{N}{2}} |X(k)|$ and $X(k)$ is the Discrete Fourier transform of the signal. The value of H ranges from 0 to H_{max} where

$$H_{max} = \log\left(\frac{N}{2} + 1\right).$$

As H approaches H_{max} , little or no compression is achieved by using the TC technique. Thus, H provides a means of testing a signal to see if TC is likely to provide compression.

2.3 Signal Reconstruction with TC

In TC, restoration involves taking the inverse DFT of the compressed frequency domain data $\hat{X}(k)$ defined as

$$w(n)\hat{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{X}(k) e^{j \frac{2\pi nk}{N}}, \quad n = 0, 1, \dots, N-1. \quad (2.4)$$

This can also be computed with an FFT. Zero is used in (2.4) for those values of $\hat{X}(k)$ which were not stored due to being below the threshold. To obtain $x(n)$, the right-hand side of (2.4) is divided by the window function $w(n)$.

CHAPTER III

LINEAR PREDICTIVE CODING

3.1 Introduction

In Linear Predictive Coding (LPC), the current value of the signal at each sample point is predicted by a fixed linear combination of some of the previous values. The difference between the estimated and true value of the signal is then coded and stored along with the coefficients used to perform the linear prediction. During reconstruction, the decoded difference is added to the predicted signal to reproduce the original signal perfectly; thus, no error is introduced.

In predictive coding, compression is achieved by only storing the difference between the predicted and actual data, rather than storing the true data itself. For good predictions, the dynamic range of the difference is much less than that of the original data. Unlike adaptive data sampling and transform coding, the accuracy of the LPC scheme is essentially perfect (to within computational accuracy) because no part of the input signal is discarded. The parameters of the LPC are optimized to obtain an efficient prediction in the sense that the mean squared error between the predicted and true values of the signal is minimized and therefore compression is maximized (minimum number of bits required to store the differences).

3.2 LPC Implementation

Let an input signal $x(t)$ be sampled to produce the sequence $x(n)$. The predictor forms an estimate of the signal's present value based on a linear combination of the last m signal samples using

$$\hat{x}(n) = \sum_{i=1}^m a_i x(n-i). \quad (3.1)$$

The predicted value $\hat{x}(n)$ is next subtracted from the original signal value $x(n)$ to form the difference

$$\epsilon(n) = x(n) - \hat{x}(n). \quad (3.2)$$

In LPC, samples are grouped into blocks of N successive samples each. For each such block, the first m samples, $x(0)$, $x(1)$, $\dots$, $x(m-1)$, are stored in their original form. It is then desired to minimize the prediction error over the remainder of the block. This error is defined as

$$\begin{aligned} E &= \sum_{n=m}^{N-1} \epsilon^2(n) \\ &= \sum_{n=m}^{N-1} \left[x(n) - \sum_{i=1}^m a_i x(n-i) \right]^2. \end{aligned} \quad (3.3)$$

This error can be minimized by equating the derivative of E to zero

$$\frac{\partial E}{\partial a_k} = \sum_{n=m}^{N-1} 2 \left(\left[x(n) - \sum_{i=1}^m a_i x(n-i) \right] \left[-x(n-k) \right] \right). \quad (3.4)$$

Equating the derivative to zero gives

$$\sum_{n=m}^{N-1} \sum_{i=1}^m a_i x(n-i) x(n-k) = \sum_{n=m}^{N-1} x(n) x(n-k) \quad (3.5)$$

for $k=1, 2, \dots, m$.

Thus, (3.5) can be rewritten as

$$\sum_{i=1}^m a_i \sum_{n=m}^{N-1} x(n-i) x(n-k) = \sum_{n=m}^{N-1} x(n) x(n-k) \quad (3.6)$$

for $k=1, 2, \dots, m$.

If

$$\Phi_{ki} \equiv \sum_{l=m}^{N-1} x(l-i) x(l-k) \quad (3.7)$$

then,

$$\Phi = \begin{pmatrix} \Phi_{11} & \Phi_{12} & \dots & \Phi_{1m} \\ \Phi_{21} & \Phi_{22} & \dots & \Phi_{2m} \\ \vdots & \vdots & \dots & \vdots \\ \Phi_{m1} & \Phi_{m2} & \dots & \Phi_{mm} \end{pmatrix}$$

Note that Φ is a symmetric matrix.

If

$$v_k = \sum_{l=m}^{N-1} x(l)x(l-k) \quad (3.8)$$

$$\mathbf{V} = [v_1, v_2, \dots, v_m]^t$$

$$\mathbf{A} = [a_1, a_2, \dots, a_m]^t$$

then, (3.6) can be represented in matrix-vector form as

$$\Phi \mathbf{A} = \mathbf{V}$$

$$\mathbf{A} = \Phi^{-1} \mathbf{V}. \quad (3.9)$$

Substituting for values of k and i in (3.7) gives

$$\begin{aligned} \Phi_{12} &= \sum_{l=m}^{N-1} x(l-1)x(l-2) \\ &= x(m-1)x(m-2) + x(m)x(m-1) \\ &\quad + x(m+1)x(m) + \dots + x(N-2)x(N-3) \end{aligned} \quad (3.10)$$

also

$$\begin{aligned} \Phi_{13} &= \sum_{l=m}^{N-1} x(l-1)x(l-3) \\ &= x(m-1)x(m-3) + x(m)x(m-2) \\ &\quad + x(m+1)x(m-1) + \dots + x(N-2)x(N-4) \end{aligned} \quad (3.11)$$

and

$$\begin{aligned}
\Phi_{23} &= \sum_{l=m}^{N-1} x(l-2)x(l-3) \\
&= x(m-2)x(m-3) + x(m-1)x(m-2) \\
&\quad + x(m)x(m-1) + \dots + x(N-3)x(N-4).
\end{aligned} \tag{3.12}$$

The equations for Φ can be generalized to

$$\Phi_{ki} = \Phi_{k-1,i-1} + x(m-k)x(m-i) - x(N-k)x(N-i). \tag{3.13}$$

Thus, only the elements of the first row in Φ need to be computed using (3.7) and the rest can be found from (3.13) and the symmetric nature of Φ .

Also, substituting for values of k and l in equation (3.8) yields

$$\begin{aligned}
v_1 &= \sum_{l=m}^{N-1} x(l)x(l-1) \\
&= x(m)x(m-1) + x(m+1)x(m) + \dots + x(N-1)x(N-2) \\
&= \Phi_{12} - x(m-1)x(m-2) + x(N-1)x(N-2)
\end{aligned} \tag{3.14}$$

and

$$\begin{aligned}
v_2 &= \sum_{l=m}^{N-1} x(l)x(l-2) \\
&= x(m)x(m-2) + x(m+1)x(m-1) + \dots + x(N-1)x(N-3) \\
&= \Phi_{13} - x(m-1)x(m-3) + x(N-1)x(N-3)
\end{aligned} \tag{3.15}$$

Therefore, v_k can be generalized to

$$v_k = \Phi_{1,k+1} - x(m-1)x(m-(k+1)) + x(N-1)x(N-(k+1)) \tag{3.16}$$

so only the value for v_m needs to be computed independently of the values of Φ .

Thus, the procedure for computing Φ and V is to use (3.7) to calculate the first row of Φ and use the symmetry property and (3.13) to form other components of Φ . Values of V are determined from (3.16) except for v_m which is computed using (3.8). The predictor coefficients a_i can then be calculated from (3.9).

The differences between the predicted and actual signal values are coded using the Huffman Coding scheme. This is a near-optimum source coding scheme because it gives an average code word length l [11] of

$$H \leq l \leq H + 0.086 + P_{max} \quad (3.17)$$

where H is the source entropy given by

$$H = - \sum_k P_k \log_2 P_k$$

where P_k is the probability of the k^{th} value of $\epsilon(n)$, and

$$P_{max} = \max_k \{P_k\}.$$

For actual signals there is usually enough spread in the values of $\epsilon(n)$ for P_{max} to be small; usually less than 0.1. Thus, the Huffman coding scheme achieves coding efficiencies within 0.2 bits/sample of the optimum entropy value. Since this is an upper bound, actual efficiencies are usually even better with values around 0.1 bits/sample above the entropy common.

For the duration of a block, the difference $\epsilon(n)$ in (3.2) can be viewed as the output of a FIR filter since

$$\begin{aligned} \epsilon(n) &= x(n) - \hat{x}(n) \\ &= x(n) - \sum_{i=1}^m a_i x(n-i) \end{aligned}$$

$$\begin{aligned}
&= x(n) + \sum_{i=1}^m b_i x(n-i) \\
&= \sum_{i=0}^m b_i x(n-i)
\end{aligned} \tag{3.18}$$

where $b_i = -a_i$ and $b_0 = 1$.

Taking the z-transform of (3.18) gives

$$\epsilon(z) = \sum_{i=0}^m b_i z^{-i} X(z). \tag{3.19}$$

If

$$H(z) = \sum_{i=0}^m b_i z^{-i}$$

then

$$\epsilon(z) = H(z)X(z) \tag{3.20}$$

and

$$H(z) = \frac{\epsilon(z)}{X(z)}$$

or

$$X(z) = \frac{1}{H(z)} \epsilon(z). \tag{3.21}$$

With this approach, the LPC operations over a block can be viewed in the simple way shown in Fig. (3.1). Both the $H(z)$ and $\frac{1}{H(z)}$ filters are started with the first m samples of the block as their initial conditions.

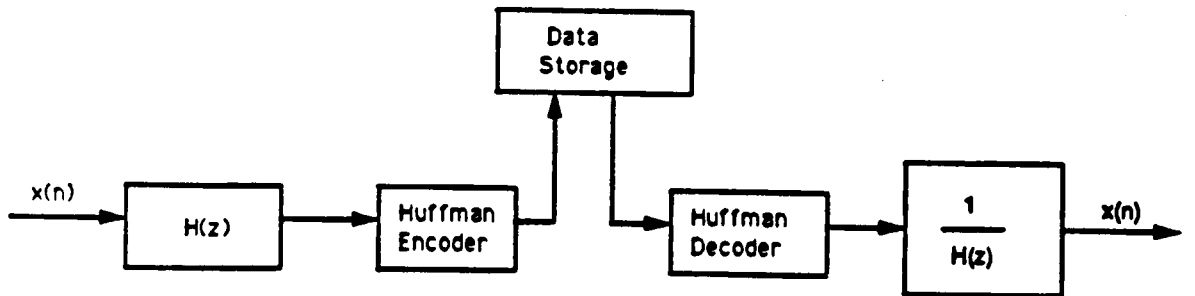


Fig. 3.1 Model of LPC scheme using FIR Filter

The filter $H(z)$ in Fig. (3.1) is a whitening filter [17] i.e., a filter with a flat error distribution with respect to frequency. Thus, $\epsilon(n)$ is approximately a white noise process. For actual signals it has also been found that $\epsilon(n)$ has a roughly Gaussian amplitude distribution with zero mean. As a signal becomes more noisy, the standard deviation of the Gaussian distribution tends to increase.

With Huffman coding, it is necessary for the decoder to know the probability distribution of the block being decoded so it can generate the same code used by the encoder. By assuming a zero-mean Gaussian distribution, both encoder and decoder can generate the same code for a given block by knowing a single number - the standard deviation of $\epsilon(n)$ for the block.

The choice of the Gaussian model will not affect the accuracy of the reconstructed signal no matter what the actual distribution of the difference is. However, inefficient coding of the differences will result if the actual probability distribution of the signal differs significantly from a Gaussian since the code word length will not be optimal.

When coding a block of differences, the length n_k of the k^{th} code word would ideally satisfy

$$n_k = \log_2\left(\frac{1}{P_k}\right) \quad (3.22)$$

where P_k is the probability of the k^{th} difference. In this way, the average number of bits given by

$$B_a = \sum_k P_k n_k$$

becomes equal to the entropy of the block. The Huffman coding algorithm uses the P_k 's to generate a set of code words that approximately satisfy (3.22).

The following is a brief discussion of how to implement Huffman coding. For more details on this subject the reader is referred to [11] and [18].

The Huffman coding algorithm starts by placing the values of the P_k in decreasing order on the leaves of a tree as shown in Fig. 3.2. In the figure, these probabilities range from 0.3 down to 0.05. The algorithm for generating the code is to complete the tree in the following way. The two smallest probabilities on the tree are summed to form a new node which is labeled with the sum. The branches from the new node to the nodes from which it was summed are labeled with the branch to the less probable node receiving a 1 and the other a 0.

On the figure, the first time this is done is in summing 0.05 and 0.15 to form the 0.2 node. The nodes that were summed are removed from further consideration and the process is repeated for the next two smallest probabilities. From the figure, these are the two nodes labeled 0.15 which are summed to form the 0.3 probability node. Next, the 0.2 leaf node and the 0.2 node formed from the first sum are chosen and summed to form the 0.4 probability node. This process continues until a sum gives a probability of 1.0 which identifies the root node.

The code word for the k^{th} difference is the bit pattern on the tree from the root node to the leaf node corresponding to P_k . For example, the code word for for the $P_k=0.05$ difference is 111 while that for $P_k=0.3$ difference is 01.

A computer algorithm for forming the Huffman code can be described by defining the following terms associated with the tree. If two nodes are adjacent on a path from the root to the leaf, the one that is closer to the root is the *parent* of the other, which is called the *child* of the parent. Two nodes with a common parent are called *siblings*. The algorithm for generating the Huffman code is given below.

1. Create two lists. The first is a list T of the tree nodes which initially contains only the leaves of the tree which are assigned the source probabilities P_k . The second is a list L of the source probabilities P_k .

2. Take the two smallest probabilities in L , make the corresponding nodes siblings, generate an intermediate node as their parent, label the link from parent to the more probable child with *zero* and the other with *one*, assign the sum of the two sibling probabilities to the parent, add the parent node to T , delete the two sibling probabilities from L , and add the parent probability to L .
3. Repeat step 2 until only one probability remains in L . This one remaining probability should be unity and identifies the root node of the tree. The bit pattern from the root node to any leaf is the Huffman code for that leaf.

A computer program for efficiently generating the Huffman code using this algorithm is given in [18, pp. 275-283].

The decoder used to recover a difference from its Huffman code uses the same tree as the encoder. As the decoder receives each bit, it traverses the tree starting at the root node. Whenever a leaf node is encountered, a difference has been decoded and decoding resumes at the root node.

3.3 Signal Reconstruction with LPC

Restoration of the original signal with LPC involves decoding the Huffman Codes to get the difference $\epsilon(n)$. This is then added to the predicted values $\hat{x}(n)$ to obtain a perfect (to within calculation accuracy) reconstruction of $x(n)$ using equation (3.2). The predictor used during reconstruction is identical to the one employed during initial estimation of the predicted signal. Although no peak error is introduced with LPC, it is possible to allow for some acceptable percentage of peak error by discarding (zeroing) differences less than an error threshold. This would permit further compression.

CHAPTER IV

RESULTS

4.1 Introduction

The three data compression methods discussed in earlier chapters were implemented with computer programs. Representative simulated signals and test data from strain gauge measurements were compressed and restored to their original sample rate to examine the amount of data compression and the accuracy of restoration. For each of the methods, the compression ratio, defined as the ratio of the total number of bits in a sample block after compression to the total number of bits in the same block before compression, was calculated. A compression ratio (CMPR) of 1.0 indicates that no data reduction was achieved. The smaller the CMPR value relative to 1.0, the better the data reduction obtained. The peak error, defined as the maximum absolute value of the difference between the original and restored data in a sample block, was also determined and used as a measure of the accuracy of restoration. The peak error can also be measured as percentage of the full scale amplitude of the signals used.

For different thresholds that determine those coefficients set to zero in transform coding and the end of the significant spectral content in adaptive data sampling, the computer programs compute the peak error, compression ratio and the total number of bits required for data storage. The linear predictive coding scheme was tried for different block lengths and model orders to determine the model order and block size that gives the best data compression.

4.2 Signals Analyzed

Several simulated signals and many test signals were compressed with the

three methods discussed in this thesis and the results were analyzed. All signals were 12-bit data spanning the full scale of ± 2047 . All data were analyzed in files of size 32768 samples. To reduce the computing time required, only the first 16384 samples of a file were compressed and analyzed using adaptive data sampling and transform coding. However, with linear predictive coding the compression ratio often continued to improve as the block size increased beyond 16384 so all 32768 samples were processed in analyzing LPC.

1. Simulated Signals

A number of simulated signals were analyzed with the three methods discussed in this thesis. Included were:

1. Low frequency sine waves with an amplitude of 1000 and a normalized frequency of 0.1.
2. Low frequency sine waves with Gaussian noise added. The Gaussian noise added to the sine wave has a mean value of zero and a standard deviation of 30.
3. Amplitude modulated sine waves.
4. Sum of sine waves.

The results of only the first two simulated signals are discussed in this chapter since the results are representative of those obtained from the other signals.

2. Actual Signals

The results of analyzing six different data files that represent about 20 Mbytes of strain gauge measurements are also presented. The data files are labeled DAT.1, DAT.2, ... DAT.6. The analyses were carried out on much more data than the six representative data files presented in this chapter.

4.3 Adaptive Data Sampling scheme

A FORTRAN program was written that implements ADS. A fixed block length of 1024 was used. For different threshold values T ranging from 0.7 to 0.98, the program computes the CMPR, peak error and the total number of bits required for data storage. Using ADS on the simulated and test signals gives the following results.

For threshold values T between 0.7 and 0.98, a peak error of 0.882 (full scale 2047, or about 0.043% peak error) with a CMPR of 0.502 was obtained with the low frequency sine waves. With Gaussian noise added to the sine waves, the peak error for $T=0.98$ was 0.539 (about 0.02% peak error) and the CMPR jumped up to 1.002. This value represents the maximum CMPR value for the ADS scheme and it denotes no data compression (the 0.002 over unity comes from the storage required to indicate the sample rate for each block). At $T=0.85$, a CMPR of 0.502 which is comparable to that obtained for the pure sine waves was obtained for the noisy sine waves, but due to aliasing of broadband noise in the signal, the peak error was 1542.2. This is a peak error of about 75% and it is certainly not acceptable for instrumentation signals. The ADS gives good performance on both simulated signals considered, but as the signal becomes noisy, the ADS scheme fails to subsample the signals for high thresholds. When a lower threshold permits subsampling, unacceptable peak errors are obtained due to aliasing of the noise.

For the actual signals analyzed, the results followed the trend obtained for the noisy sine waves. When subsampling is not possible on most of the data blocks, i.e., CMPR's of near unity were recorded, acceptable peak errors of between 0.001% and 0.004% were obtained. But when the signals are subsampled, peak errors of between 1.7% and 6% were obtained. For some applications these would be acceptable errors, but for many applications they would not.

4.4 Transform Coding Scheme

A FORTRAN program was also written to implement TC. A fixed block length of 1024 was used. For different threshold values T ranging from 0.005 to 0.3, and for different numbers of samples of overlap, the program computes the CMPR, peak error and the total number of bits required for data storage. Analyzing the simulated and actual signals with TC gives the following results.

Fig. 4.1 shows the comparison between the peak errors between the pure sine wave and the noisy sine wave. Due to noise, the peak error obtained for the noisy signal is higher than that of the pure sine wave. Six representative signals from the strain gauge measurements were analyzed with the TC scheme. The results of the analysis were separated into two sets of three data files each. Fig. 4.2 shows the plot of the CMPR's versus threshold for the first set of three data files overlapped by 512 samples. When the threshold T is close to zero, the number of samples stored is higher due to the overlap. But as the threshold value increases away from zero i.e., $T=0.075$, data reduction of about 80% is obtained when the signal is overlapped by 512 samples, although the peak error at this stage is unacceptable (between 1.5% and 2.5%). Fig. 4.3 also shows the CMPR's for the second set of three data files. The same characteristics observed in Fig. 4.2 are also repeated.

Figures 4.4 and 4.5 shows the peak errors versus the threshold T for the actual data files at an overlap of 512 samples. From the plots, the peak error increases as the threshold increases. For some of the files considered, acceptable peak errors of about 1% of full scale were obtained at threshold $T=0.01$. Although these peak errors were acceptable, the TC method stored more samples at this threshold resulting in a higher CMPR value. Fig. 4.6 shows peak errors versus overlap at different thresholds for the actual data file DAT.1. In general, as the number of overlap samples increases, the peak error decreases. For the transform

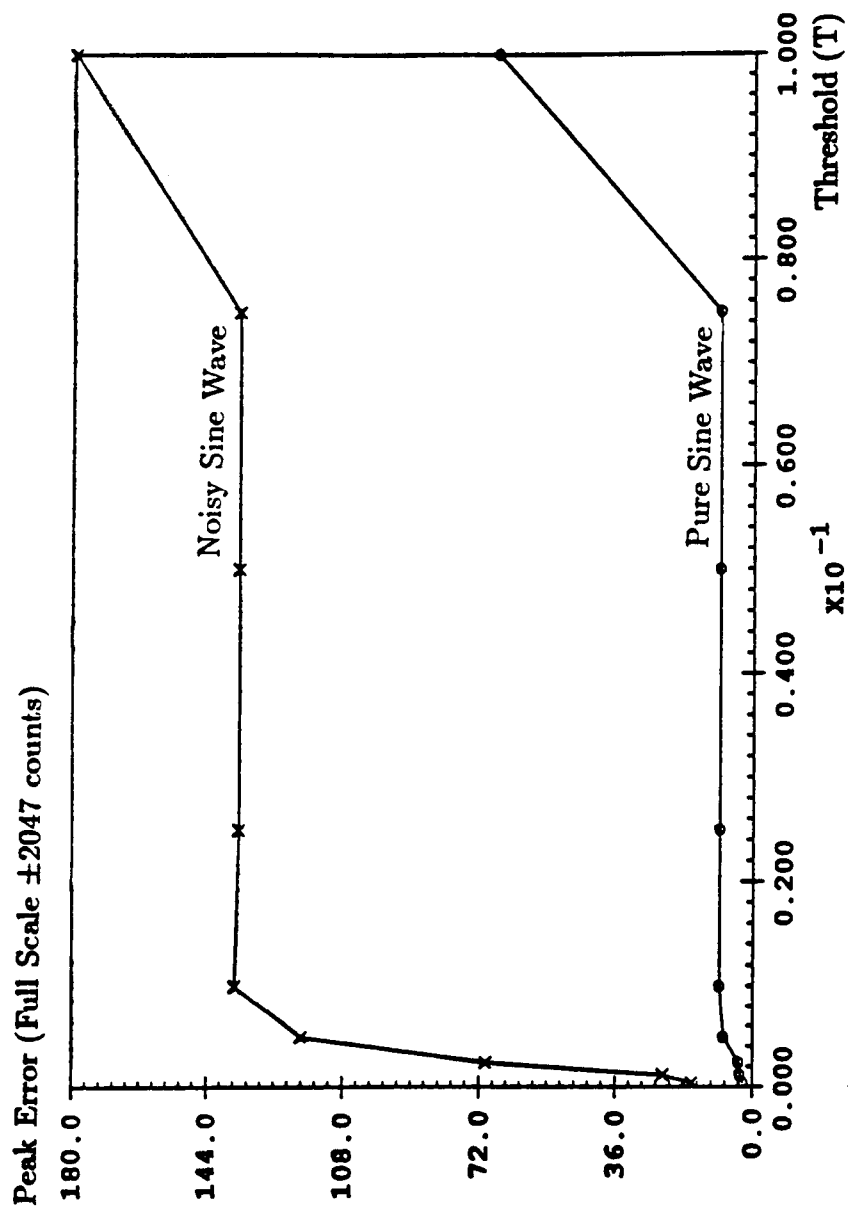


Fig. 4.1 A comparison of the Peak Errors versus Threshold for both Pure and Noisy Sine Waves using the TC technique at 50% Overlap

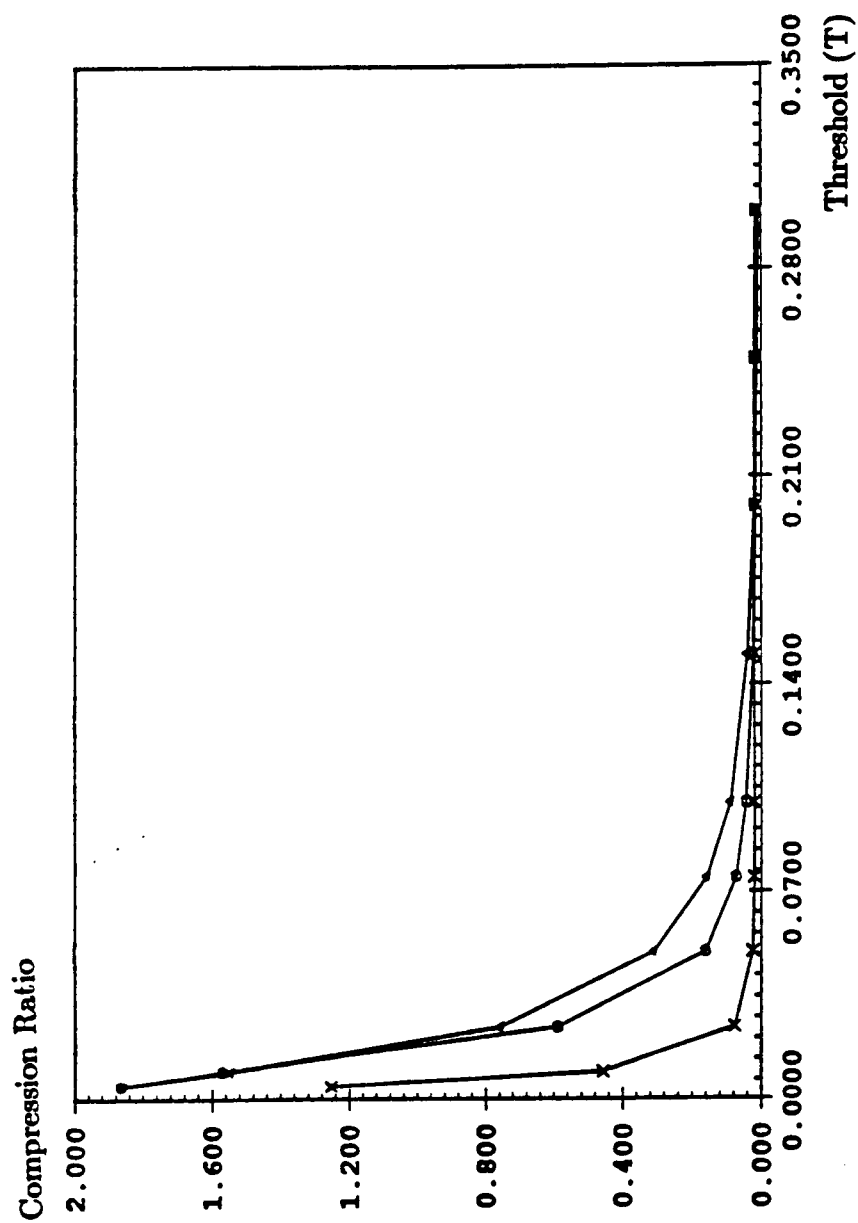


Fig. 4.2 Compression Ratio versus Threshold for the first set of three Data files using TC technique at 50% Overlap

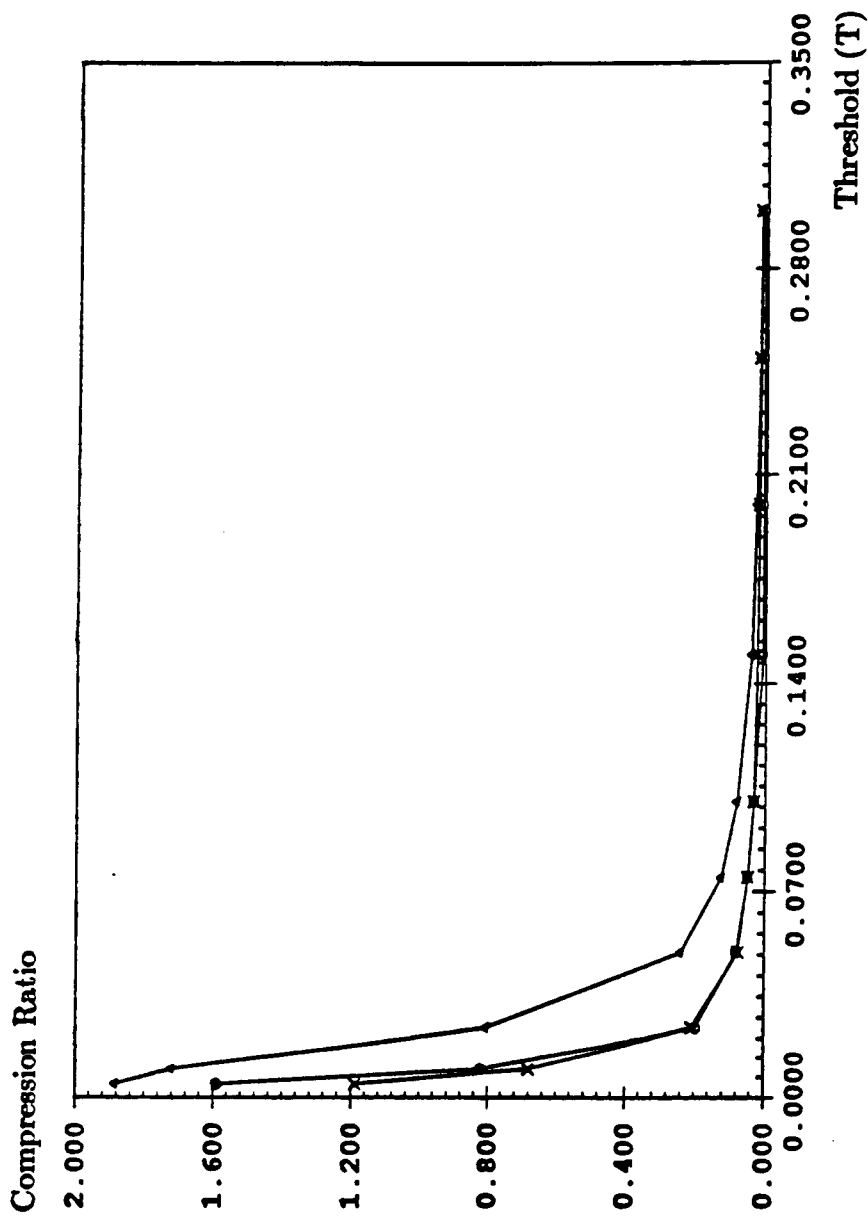


Fig. 4.3 Compression Ratio versus Threshold for the second set of three Data files using TC technique at 50% Overlap

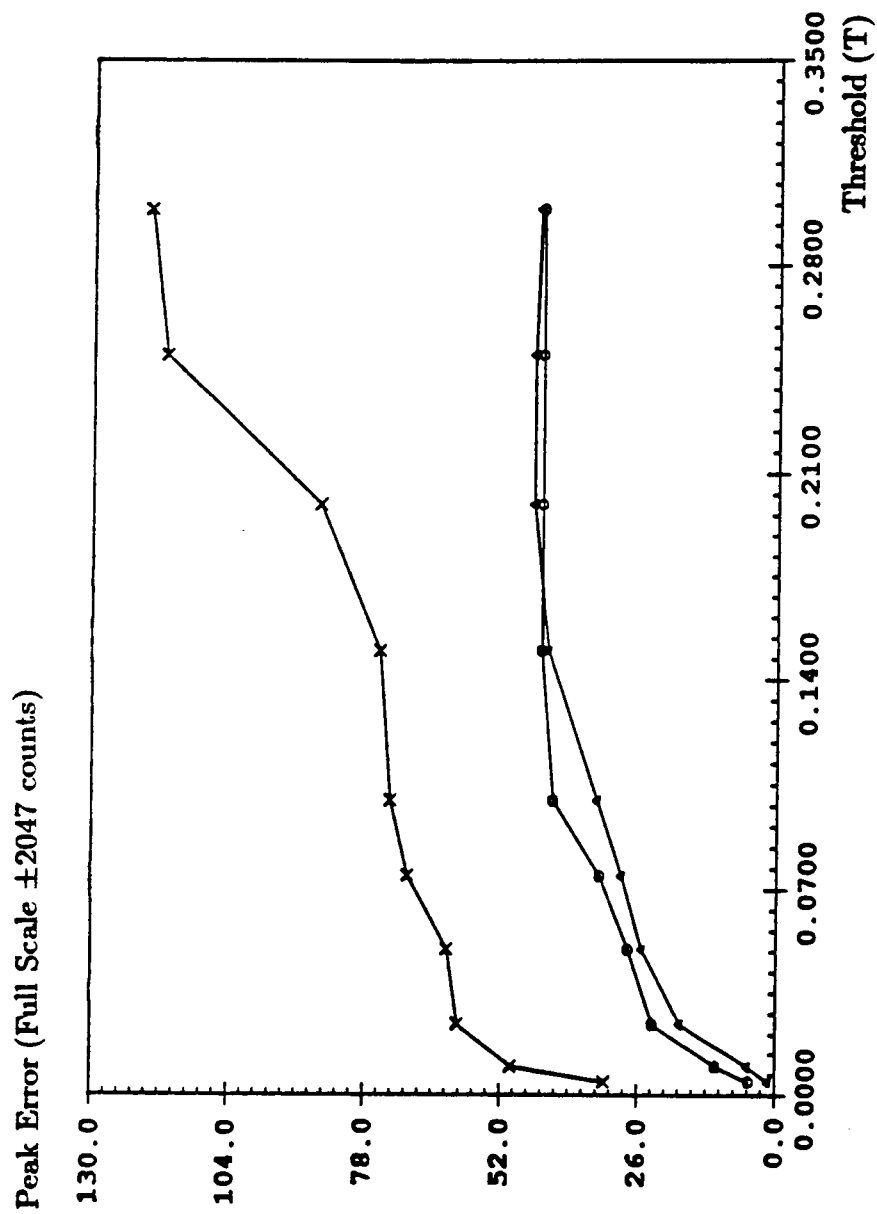


Fig. 4.4 Peak Error versus Threshold for the first set of three Data files using TC technique at 50% Overlap

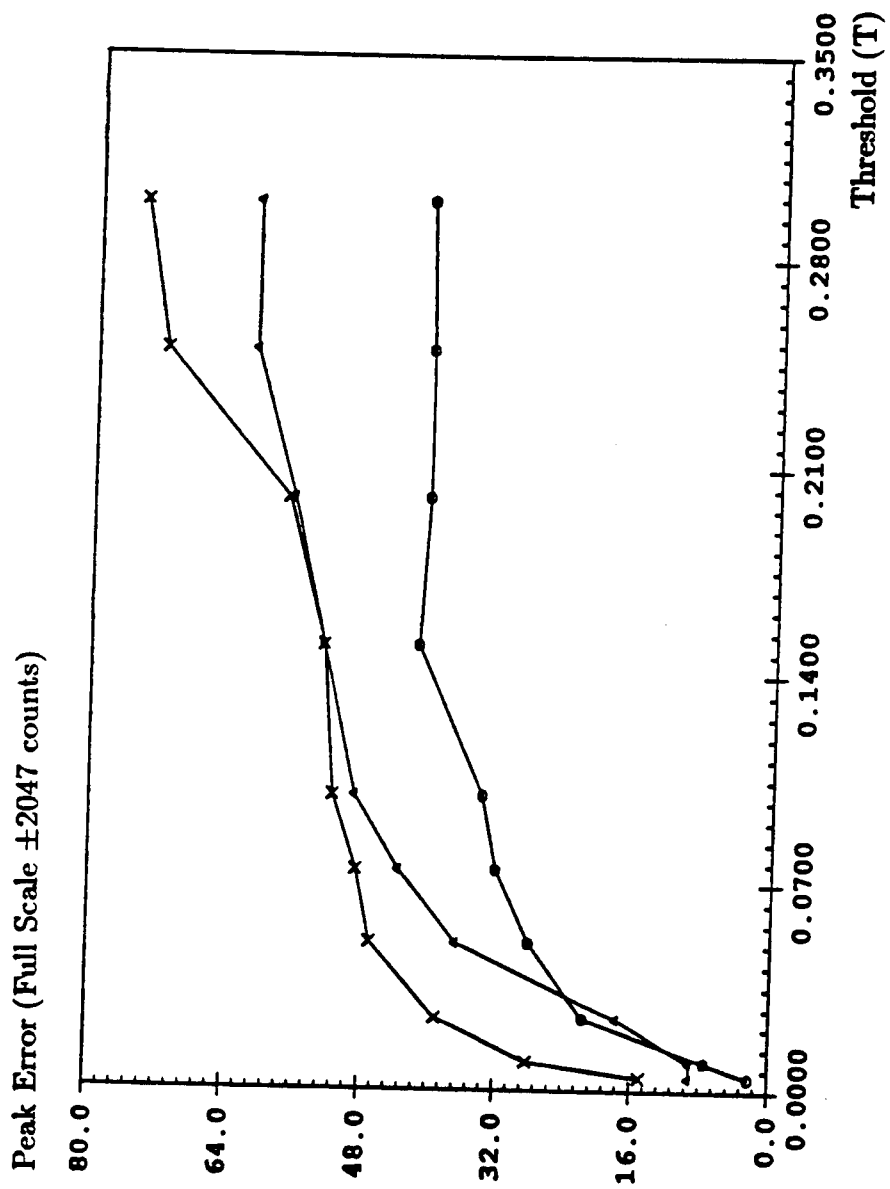


Fig. 4.5 Peak Error versus Threshold for the second set of three Data files using TC technique at 50% Overlap

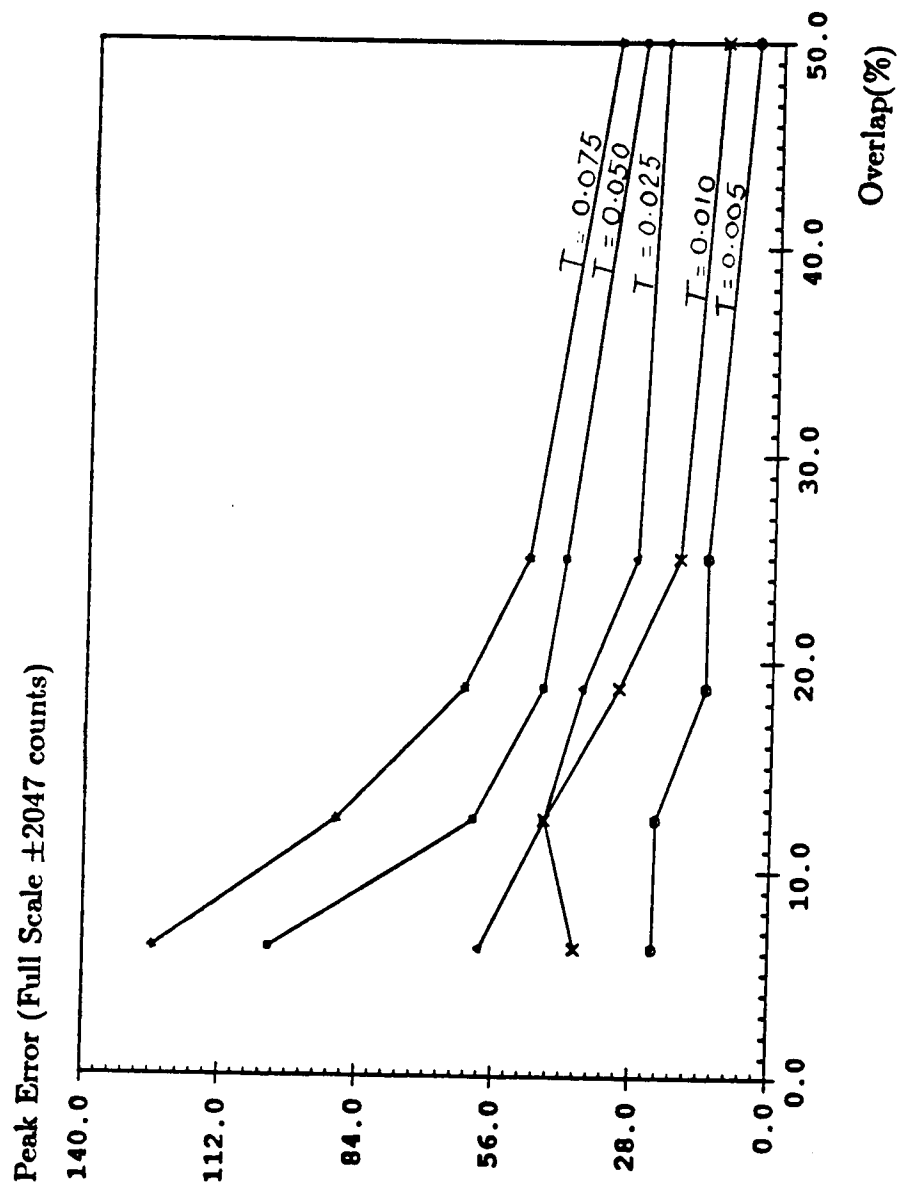


Fig. 4.6 Peak Errors versus Overlap at different Threshold values T for DAT.1 file using TC technique

coding scheme, the relationship between the compression ratio and overlap can be represented by

$$CMPR = \frac{1024}{1024 - OVERLAP} \quad (4.1).$$

A plot of (4.1) for different values of overlap, shown in Fig. 4.7, shows that as the number of samples of overlap increases towards 1024, an infinite amount of data will be stored. Fig. 4.8 shows a typical CMPR versus overlap for one of the actual data files. The plot can be seen to be comparable to the plot obtained from (4.1). An overlap of 50% was selected for TC implementation because ratio goes up very fast when the overlap is greater than 50% (as shown in Fig. 4.7).

4.5 Linear Predictive Coding Scheme

Within the accuracy of the A/D converter used in sampling the analog signal, LPC gives no errors. Therefore the technique was evaluated to determine a moderate predictor model order and block size that gives best possible data reduction. For the simulated signals, compression ratios of about 0.223 were obtained for the pure sine wave. This high compression was obtained because the pure sine wave is smooth and the LPC scheme predicts the signal almost perfectly. In effect only the predictor coefficients and the first m samples of each sample block need to be stored. With the noisy sine waves, a CMPR of about 0.6 was obtained for model order (m) between 2 and 60. Table II shows the compression ratios as a function of the block size and model order for the noisy sine wave. Tables III-IV shows the CMPR's as they relate to the model order m and block sizes for two of the six test data files. Results for the other files are similar. From these tables, it is clear that as model order m and block size increases, better data reduction is obtained. For a model order of 30 and a block size of 16384, data reduction of between 42% and 50% are obtained. Because of the computation involved in calculating

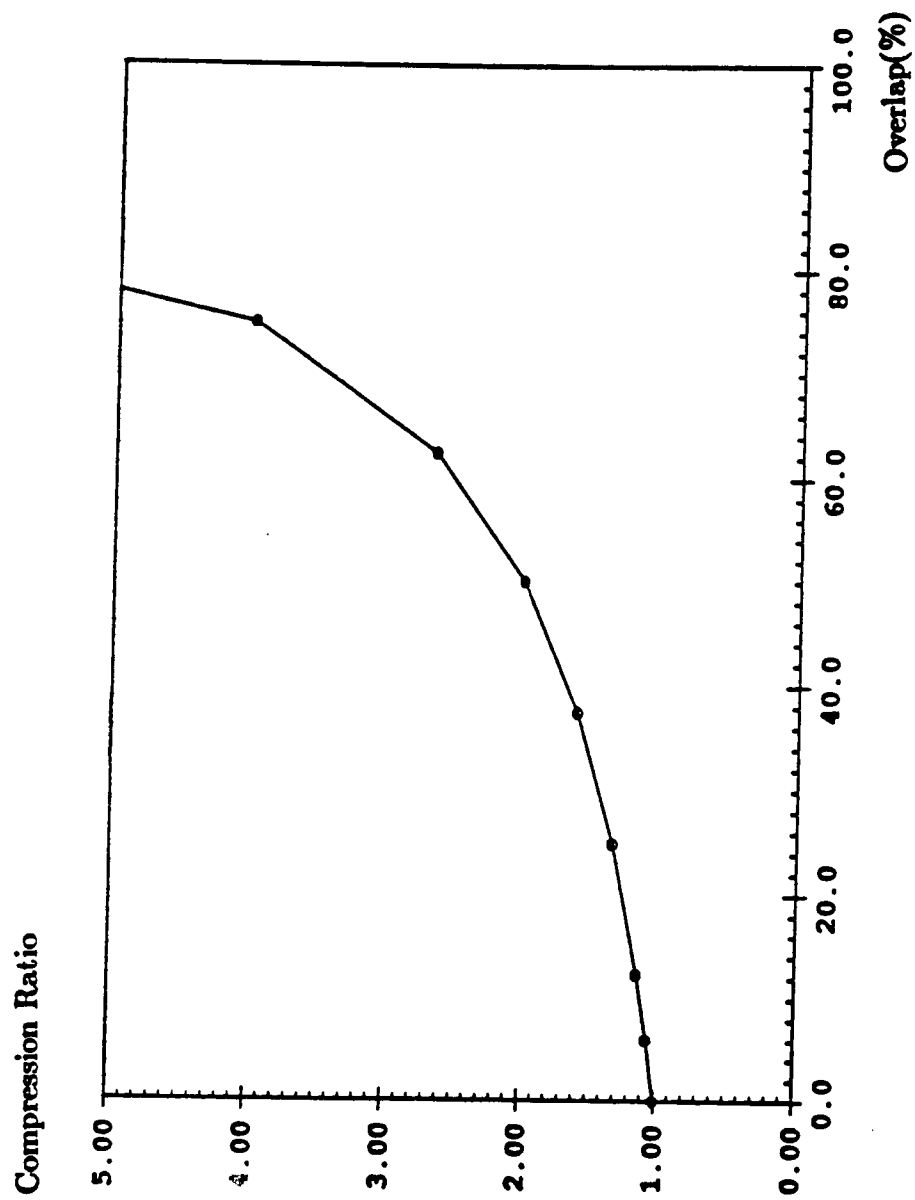


Fig. 4.7 Theoretical values of CMPR versus Overlap for TC

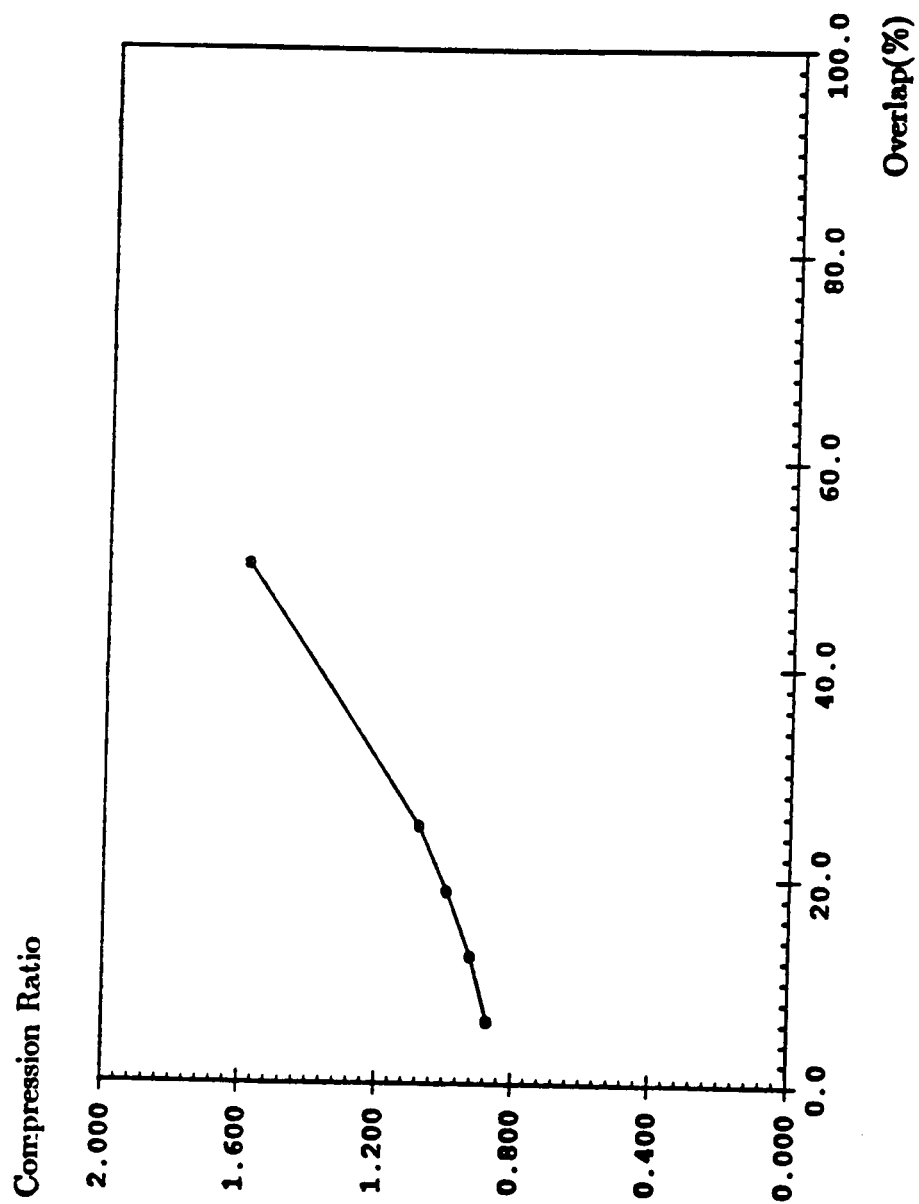


Fig. 4.8 Plot of a typical CMPR versus Overlap for Data file DAT.1

the predictor coefficients when the model order is high, a compromise has to be reached between the data reduction and the computation when selecting the appropriate model order. Fig. 4.9 shows how the CMPR changes as a function of model order for file DAT.4 and block lengths of 16384 and 32768. The other data files have a similar plot. From this plot it is clear that there is negligible improvement in CMPR for model orders above about 30.

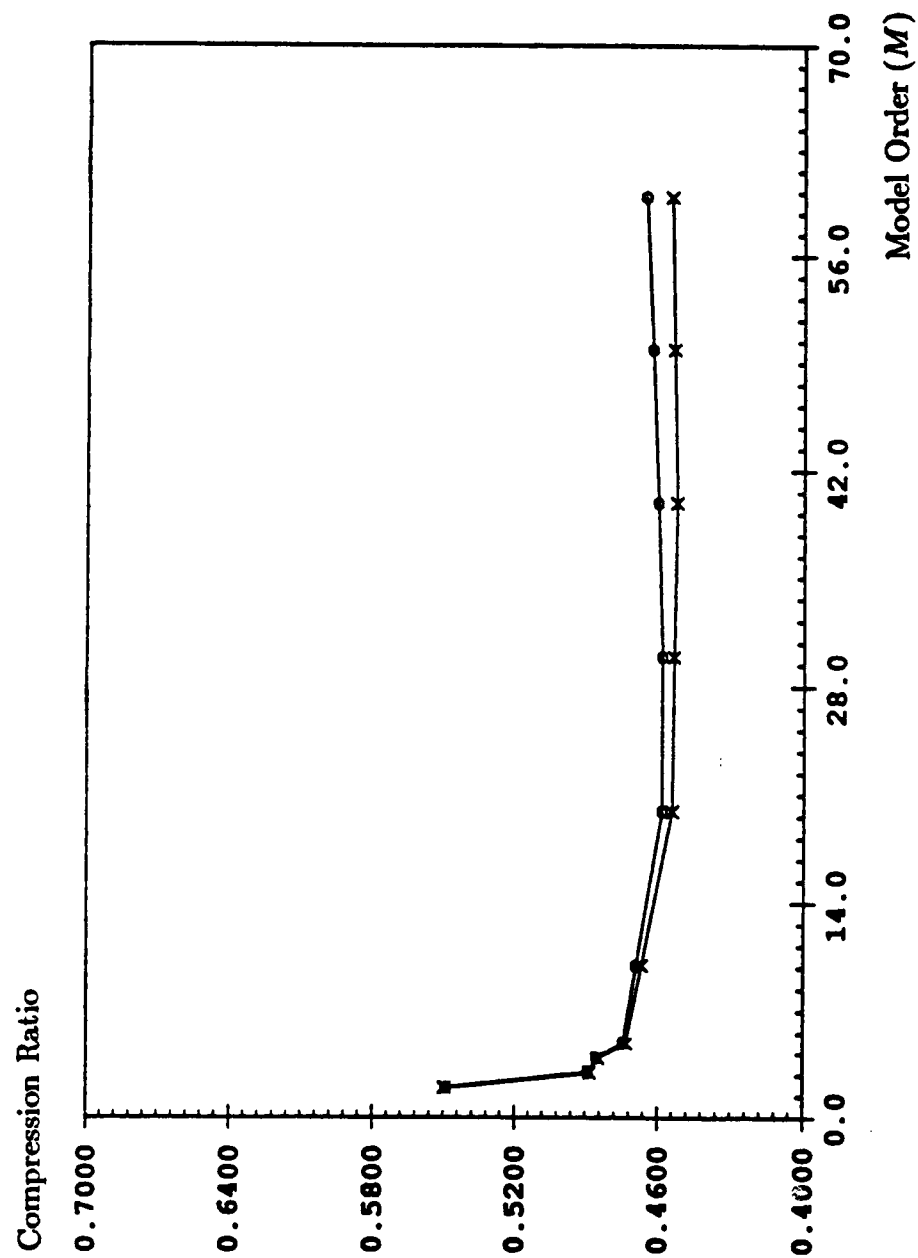


Fig. 4.9 Compression Ratio versus Model Order for Data file DAT.4

Table II. Results of LPC scheme on Noisy Sine Waves

Model Order (m)	Block Size	Compression Ratio
2	512	0.6996663
	1024	0.6903890
	2048	0.6856282
	4096	0.6833089
	8192	0.6821899
	16384	0.6816406
	32768	0.6813354
3	512	0.6647949
	1024	0.6522827
	2048	0.6461182
	4096	0.6431071
	8192	0.6415609
	16384	0.6407878
	32768	0.6404012
4	512	0.6516724
	1024	0.6363525
	2048	0.6287639
	4096	0.6249593
	8192	0.6229858
	16384	0.6221110
	32768	0.6216431
5	512	0.6483154
	1024	0.6301473
	2048	0.6210327
	4096	0.6164958
	8192	0.6141968
	16384	0.6131185
	32768	0.6122640

Table II. Contd.

Model Order (m)	Block Size	Compression Ratio
10	512	0.6686198
	1024	0.6358235
	2048	0.6192830
	4096	0.6110433
	8192	0.6069946
	16384	0.6048584
	32768	0.6039022
20	512	0.7220459
	1024	0.6595662
	2048	0.6281942
	4096	0.6125488
	8192	0.6046143
	16384	0.6007894
	32768	0.5986531
30	512	0.7786865
	1024	0.6865845
	2048	0.6404216
	4096	0.6173503
	8192	0.6055705
	16384	0.5999349
	32768	0.5971069
40	512	0.8371379
	1024	0.7149251
	2048	0.6542358
	4096	0.6233724
	8192	0.6082560
	16384	0.6006063
	32768	0.5968221

Table II. Contd.

Model Order (m)	Block Size	Compression Ratio
50	512	0.8952230
	1024	0.7441610
	2048	0.6681315
	4096	0.6298421
	8192	0.6112468
	16384	0.6015422
	32768	0.5967000
60	512	0.9532267
	1024	0.7729492
	2048	0.6822103
	4096	0.6366170
	8192	0.6143799
	16384	0.6028849
	32768	0.5969849

Table III. Results of LPC scheme on DAT.1

Model Order (m)	Block Size	Compression Ratio
2	512	0.5078939
	1024	0.4981893
	2048	0.4933878
	4096	0.4910278
	8192	0.4898885
	16384	0.4894613
	32768	0.4891968
3	512	0.5022990
	1024	0.4894002
	2048	0.4834188
	4096	0.4800619
	8192	0.4786581
	16384	0.4780070
	32768	0.4774373
4	512	0.5073853
	1024	0.4915771
	2048	0.4841512
	4096	0.4801636
	8192	0.4782918
	16384	0.4774373
	32768	0.4768880
5	512	0.5074260
	1024	0.4886881
	2048	0.4792074
	4096	0.4745687
	8192	0.4724731
	16384	0.4713135
	32768	0.4707438

Table III. Contd.

Model Order (m)	Block Size	Compression Ratio
10	512	0.5276082
	1024	0.4932658
	2048	0.4762166
	4096	0.4677124
	8192	0.4635620
	16384	0.4615072
	32768	0.4604289
20	512	0.5775554
	1024	0.5124308
	2048	0.4801025
	4096	0.4632162
	8192	0.4550781
	16384	0.4522705
	32768	0.4503174
30	512	0.6328939
	1024	0.5365601
	2048	0.4884847
	4096	0.4647420
	8192	0.4525553
	16384	0.4466146
	32768	0.4435628
40	512	0.6869710
	1024	0.5586345
	2048	0.4954834
	4096	0.4634399
	8192	0.4475708
	16384	0.4397990
	32768	0.4357503

Table III. Contd.

Model Order (m)	Block Size	Compression Ratio
50	512	0.7432861
	1024	0.5838826
	2048	0.5042521
	4096	0.4647013
	8192	0.4446615
	16384	0.4347738
	32768	0.4297689
60	512	0.8034668
	1024	0.6129354
	2048	0.5177816
	4096	0.4700928
	8192	0.4463297
	16384	0.4344889
	32768	0.4287516*

* small values of compression ratio indicates good data reduction.

Table IV. Results of LPC scheme on DAT.2

Model Order (m)	Block Size	Compression Ratio
2	512	0.6817220
	1024	0.6736653
	2048	0.6698608
	4096	0.6690674
	8192	0.6701660
	16384	0.6757202
	32768	0.6936849
3	512	0.5764160
	1024	0.5641887
	2048	0.5580851
	4096	0.5552571
	8192	00.5537313
	16384	0.5533447
	32768	0.5539144
4	512	0.5741577
	1024	0.5588176
	2048	0.5512492
	4096	0.5474650
	8192	0.5458171
	16384	0.5450643
	32768	0.5455933
5	512	0.5770264
	1024	0.5587361
	2048	0.5495605
	4096	0.5451457
	8192	0.5430095
	16384	0.5424194
	32768	0.5435994

Table IV. Contd.

Model Order (m)	Block Size	Compression Ratio
10	512	0.5889689
	1024	0.5558879
	2048	0.5394694
	4096	0.5315349
	8192	0.5279338
	16384	0.5265910
	32768	0.5292562
20	512	0.6295166
	1024	0.5654297
	2048	0.5339559
	4096	0.5192057
	8192	0.5125122
	16384	0.5104167
	32768	0.5156250
30	512	0.6845703
	1024	0.5903117
	2048	0.5434774
	4096	0.5206706
	8192	0.5097656
	16384	0.5054932
	32768	0.5064901
40	512	0.7361450
	1024	0.6112061
	2048	0.5484009
	4096	0.5177612
	8192	0.5028279
	16384	0.4965006
	32768	0.4955241

Table IV. Contd.

Model Order (m)	Block Size	Compression Ratio
50	512	0.7950643
	1024	0.6393229
	2048	0.5613200
	4096	0.5230510
	8192	0.5041911
	16384	0.4959513
	32768	0.4932861
60	512	0.8547363
	1024	0.6682129
	2048	0.5749715
	4096	0.5291138
	8192	0.5063884
	16384	0.4958903
	32768	0.4916789

CHAPTER V

DISCUSSION AND CONCLUSION

Three methods of data compression for instrumentation signals have been analyzed and tested. Adaptive data sampling (ADS) was observed to perform efficiently on low frequency (substantially oversampled) signals, but as the signal frequency increases or the signal becomes noisy, its performance degrades drastically. In this discussion, clean signals are considered to have root mean square (RMS) noise levels well below 4% of the RMS signal level, otherwise, the signal is considered noisy as indicated by the low frequency sine waves with Gaussian noise added (pp. 32). For high frequency signals or where noise is present, if the threshold is chosen to prevent aliasing of the noise, little data compression can be achieved. For noisy signals with a threshold that achieves data compression, large peak errors are encountered due to aliasing of the noise. The performance of ADS might be improved by lowpass filtering the signal prior to decimation to prevent aliasing of the spectral components above the new Nyquist frequency. However, errors relative to the original data will still be present because of the spectral components discarded in the filtering process. ADS will give acceptable error only when there are no spectral contents above the Nyquist rate that will exist following decimation. Unfortunately, it has been found that this is rarely the case with noisy signals. If lowpass filtering is to be performed prior to decimation, an efficient technique for implementing the filter in the frequency domain with a bandwidth adjustable by factors of two is described in [21].

Transform coding (TC) also performed well with clean smooth signals. However, to achieve low peak errors with noisy signals TC must retain many Fourier coefficients resulting in little or no compression. This characteristic of requiring

many more coefficients in the presence of noise makes the TC scheme an ineffective data compression method for instrumentation signals where low peak errors are desired. Where large peak errors in the neighborhood of 10% of full scale are acceptable, TC appears to be an effective method and gives excellent compression ratios.

One way in which the accuracy of TC might be improved is through the inclusion of transition samples in the "ideal" filtering imposed by the TC thresholding operation. Whenever a band of samples is to be zeroed because it is below threshold, one sample at each end of the zeroed band would be multiplied by 0.5 rather than being zeroed to provide a filter transition band. Although this was not attempted in the thesis, it could reduce the circular convolution errors and represents an area of possible future study. In the presence of large noise levels it is doubtful that this method would be useful because the spectrum crosses above and below the threshold so many times that including the transition spectral samples would require recording almost all spectral coefficients.

Linear predictive coding (LPC) was found to be the best of the three methods considered. This technique is error-free, within the accuracy of the A/D converter, and data compressions of about 50% were obtained. The compression achieved by the LPC scheme can also be improved by zeroing differences $\epsilon(n)$ that are less than the acceptable peak error. Although this was not tried in the thesis, it would allow improved compression at the cost of controlled peak errors that do not exceed a prescribed maximum. Since all differences within the allowed error band between $-\epsilon_{max}$ and ϵ_{max} become zero, the probability distribution of the difference will become as shown in Fig. 5.1 (the distribution is actually discrete but is shown continuous for convenience). The code word assigned to a difference of 0 will then have very few bits resulting in improved compression. However, since

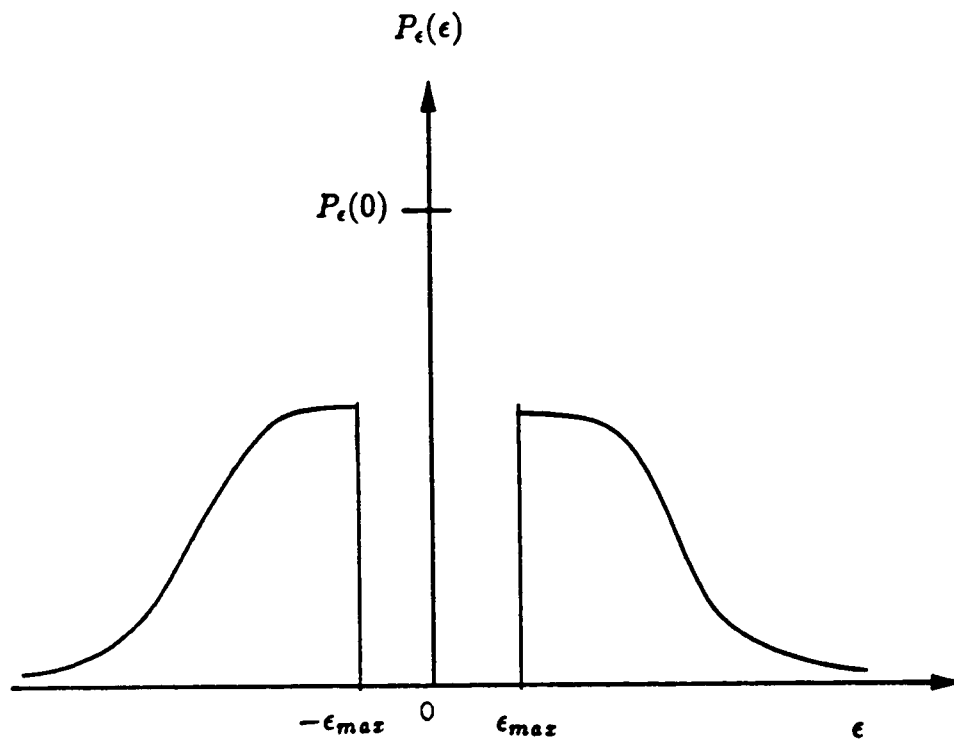


Fig. 5.1 Modified Gaussian Distribution for Peak Error Control Using LPC Scheme

P_{max} of equation (3.17, pp. 25) is now the increased probability of the zero difference, (3.17) predicts that Huffman coding may be less efficient. Therefore, the use of arithmetic coding [12] should be considered in this case. Notice also that whenever a nonzero difference is forced to zero, the decoder will have an incorrect value for use in predicting future samples. Since there is nothing that can be done about this at the decoding end, it will be necessary to have the encoder use a value that is in error by the same amount when it predicts future samples. In this way the encoder and decoder will always be predicting based on the same data. The modification of LPC to give improved compression with controlled peak errors represents another area of possible future study.

One issue which should be considered in the practical implementation of LPC is recovery from read errors when reading the compressed data from disk or tape. With Huffman coding, the beginning of one data sample is identified by the termination of decoding for the previous sample. Therefore, in the event of a read error it becomes impossible to continue decoding all subsequent data. One solution to this problem is to form all the compressed data for one data block in computer memory prior to writing it to disk or tape. The size of the compressed data block can then be determined and the number of bytes occupied by the block (or equivalently the number of disk sectors or tape blocks occupied) would be written to disk or tape followed by the compressed data. In this way a read error in one data block would not propagate to subsequent blocks since the length information would allow the remainder of a bad block to be skipped. With this approach, it may also be desirable to record the length information with some redundancy to be sure that the length information is not lost due to a read error. The maximum amount of data which would be lost due to a read error is then limited to the size of one data block and this would be a factor in determining the data block size

used.

Another issue of interest in a practical implementation of LPC is whether the data compression can take place in real time. The main computational load of LPC is in the inner products used to form the first row of the Φ matrix (equation 3.7, p.22) and the m^{th} element of the v vector (equation 3.8, p.23). Fortunately, processors are now readily available that can form inner products at rates in excess of 10 million floating-point multiplies/adds per second [22]. By implementing LPC on such a processor, a real-time implementation appears to be feasible for many applications.

Another technique may be the combination of the LPC and ADS for signals known to be bandlimited. This would utilize a decimation approach following a lowpass filtering with a cut-off frequency commensurate with the maximum frequency of interest.

In summary, the adaptive data sampling scheme performed efficiently on smooth low frequency signals, but for high frequency signals or noisy signals, the ADS scheme achieves little data compression. The TC scheme is very effective and gives excellent compression ratios where peak error in the neighborhood of 10% is acceptable. For data compression where low peak error is important, the TC scheme becomes ineffective.

Linear predictive coding was found to be the best of the three methods considered, giving error-free (within the computational accuracy and the accuracy of the A/D) compression of about 50%. It should be possible to further improve this compression by allowing controlled errors that do not exceed a prescribed maximum.

It should be pointed out that the techniques described herein have equal applicability to the compression of data acquired at sample rates well above the

Nyquist rate and for which a rate considerably higher than the Nyquist rate is to be maintained. For example, if a system acquires data at a rate C ($C \gg 2$) times the intended bandwidth f_b of the system (f_b the design maximum expected frequency) to improve peak amplitude accuracy, then for signals with frequencies far less than f_b these techniques may be applied to compress the data and yet maintain a sample rate of Cf , where f is the actual signal frequency. Similar compression performance will still be obtained but aliasing issues will disappear resulting in higher reconstruction accuracy.

REFERENCES

REFERENCES

1. R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech," IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-25, No. 4, August 1977, pp. 299-309.
2. P.A. Wintz, "Transform Picture Image," Proc. IEEE, vol. 60, No. 7, July 1972, pp. 809-820.
3. G.B. Anderson and T.S. Huang, "Piecewise Fourier Transformation for Picture Bandwidth Compression," IEEE Trans. on Comm. Tech., vol. COM-19, No. 2, April 1971, pp. 133-140.
4. S. Singhal, C.-T. Chen, and D. Le Gall, "Source Coding of Speech and Video Signals," Proc. IEEE, vol. 78, No. 7, July 1990, pp. 1233-1249.
5. B.S. Atal and M.R. Schroeder, "Adaptive Predictive Coding of Speech Signals," Bell System Tech. J., vol. 45, October 1970, pp. 1973-1986.
6. B.S. Atal, "Predictive Coding of Speech at Low Bit Rates," IEEE Trans. on Comm., vol. COM-30, No. 4, April 1982, pp. 600-614.
7. R.W. Page and N.W. Nelson, "Adaptive Sample Rate: A First-Generation Automatic Time Base," Hewlett-Packard Journal, Feb. 1988, pp.23-25.
8. Proceedings of the Scientific Data Compression Workshop. NASA CP-3025, May 1988.
9. R.E. Crochiere and L.R. Rabiner, Multirate Digital Signal Processing. Englewood Cliffs, N.J. : Prentice-Hall, 1983.
10. J. Pearl, "On Coding and Filtering Stationary Signals by Discrete Fourier Transforms," IEEE Trans. on Info. Theory, March 1973, pp. 229-232.
11. R.G. Gallager, "Variations on a Theme by Huffman," IEEE Trans. on Info. Theory, vol. IT-24, No. 26, Nov. 1978, pp. 668-674.

12. I.H. Witten, R.M. Neal and J.G. Cleary, "Arithmetic Coding for Data Compression," Comm. of the ACM, vol. 30, No. 6, June 1987, pp. 520-540.
13. F.J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," Proc. IEEE, vol. 66, No. 1, Jan. 1978, pp. 51-84.
14. J.G Proakis and D.G. Manolakis, Introduction to Digital Signal Processing. Macmillan Publishing Company, N.Y. 1988.
15. T.W. Parks and C.S. Burrus, Digital Filter Design. John Wiley, N.Y. 1987.
16. E.F. Schroeder, H.-J. Platte, and Krahé, "Stereo Audio Coding with CD-Quality and 256Kbit/sec," IEEE Trans. on Cons. Elec., vol. CE-33, No. 4, Nov. 1987, pp. 512-518.
17. N.S Jayant and P. Noll, Digital Coding of Waveforms. Englewood Cliffs, N.J. : Prentice Hall, 1984.
18. A.M. Tenenbaum and M.J. Augenstein, Data Structures Using Pascal. Englewood Cliffs, N.J. : Prentice Hall, 1981.
19. T.J. Lynch, Data Compression Techniques and Applications. Van Nostrand Reinhold, NY., 1985.
20. A. Habibi, "Comparison of the n^{th} -order DPCM Encoder With Linear Transformations and Block Quantization Techniques," IEEE Trans. on Comm. Tech., vol. COM-19, No. 6, Dec. 1971, pp. 948-956.
21. B.W. Bomar, L.M. Smith, and W.R Wysor, "A System for Accurate Multiresolution Spectral Analysis of Sampled Data," to be submitted for publication in IEEE Trans. on Signal Processing.
22. Motorola Inc. DSP96002 IEEE Floating-Point Dual-Port Processor User's Manual. Motorola Publication DSP96002UM/AD. Phoenix, AZ. 1989.

VITA

Adeyemi Olufemi Adebola was born on November 14, 1962. He attended Ibadan Grammar School, Ibadan Nigeria and graduated in July, 1979. Two years later, he entered the University of Lagos, Lagos Nigeria. There he received a Bachelor of Science degree in Electrical Engineering in July 1987. In August 1989, he accepted a graduate research assistanship from the University of Tennessee Space Institute, Tullahoma where he began studies toward a Master of Science degree in Electrical/Computer Engineering. This degree was awarded in August 1991. For now the future is bright and he his looking to God for the best that is yet to come.