AN ABSTRACT OF

Hin Leong Tan for the degree of Master of Science in Electrical and Computer Engineering presented on January 17, 1984.

Title: Frequency Domain Deconvolution

Redacted for Privacy

Abstract Approved: ROY C. RATHJA

Frequency domain design of deconvolution filters is studied using Fast Fourier Transform. Filters for both the spike and the Gaussian reflector wavelet are obtained. True deconvolution filters are infinitely long IIR filters, and frequency domain analysis is an effective way of finding its optimum finite length approximation for an arbitary given filter length. The Gaussian waveform and its Discrete Fourier Transform has several desirable characteristics that make it a good choice as a reflector wavelet. Basic wavelets with zeros on the unit circle results in spike inverse filters that are quasistable and infinitely long. By circularly shifting the Gaussian FFT sequence it is possible, depending on the location of the zeros, to find Gaussian inverse filters that are much shorter than the corresponding spike inverse filters.

FREQUENCY DOMAIN DECONVOLUTION

by

HIN LEONG TAN

A Thesis

Submitted to

Oregon State University

in partial fulfillment of

the requirements for the

Degree of

Master of Science

Completed January 17, 1984

Commencement June, 1984

APPROVED:

Redacted for Privacy

Professor of Electrical and Computer Engineering in Charge of Major

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Dated Thesis is Presented: January 17, 1984

Typed by Norma A. Sherman for Hin Leong Tan

TABLE OF CONTENTS

		Page
I.	Introduction	1
	1.1 Purpose	1
	1.2 The Fast Fourier Transform	3
	1.3 The Deconvolution Model	4
	1.4 Overview	7
II.	The Relector Wavelet	9
	2.1 Introduction	9
	2.2 Spike Reflector Series	10
	2.3 Gaussian Waveform	12
	2.3.1 Continuous Fourier Transform	12
	2.3.2 Fast Fourier Transform	14
	2.3.3 Finite Sequences	15
	2.3.4 Phase	15
	2.3.5 Truncation of FFT Data	16
	2.4 Summary	24
III.	Frequency Domain Analysis	25
	3.1 Introduction	25
	3.2 Aliasing in the Time Domain	25
	3.3 Poles and Zeros of Sequences	26
	3.4 Analytic Examples	27
	3.4.1 Minimum Phase Sequence	28
	3.4.2 Maximum Phase Sequence	31
	3.4.3 Summary/Discussion of Analytic Examples	34
	3.5 Mixed Phase Sequences	36
	3.6 Estimating Delay of Output Spike	38
	3.6.1 Windowing the Sequences	38
	3.6.2 Example of Delaying Output Spike	40
	3.6.3 Determining the Optimum Window Position	46
	3.7 Quasistable Inverse Filters	47
	3.8 Circular Convolution vs. Linear Convolution	49
	3.9 Circular Inverse Filters	49
	3.10 Summary	53
IV.	Design and Implementation of the Deconvolution Filter	55
	4.1 Introduction	55
	4.2 Gaussian Reflector Series	55
	4.3 Flowchart	58
	4.4 Examples of Inverse Filters	60
	4.4.1 Spike Inverse Filters	62
	4.4.2 Gaussian Inverse Filters	67
	4.5 Circular Shift of Gaussian FFT	68
	4.6 Examples of Linear Filtering	79
	4.7 Example of Circular Inverse Filtering	80
	4.8 Summary	85

		Page
V.	Summary/Conclusion	86
	5.1 Summary of Results	86
	5.2 Additional Research	88
	References	90
	Appendix A: DFT of Gaussian Waveform Standard Deviation - 2.5	91
	Appendix B: DFT of Gaussian Waveform Standard Deviation - 5.0	92
	Appendix C: Filter Coefficients	93
	Appendix D: Program for Finding the Inverse Filter Using the Flowchart of Figure 4.1	94

LIST OF FIGURES

Figure		Page
1.1	Composite Signals	2
1.2	Linear Discrete Time System	4
2.1	FFT of Gaussian Sequences	17
2.2	Symmetrical Truncation of FFT Data	19
2.3	Error Due to Truncation of Gaussian FFT . . .	20
2.4	Truncation Error as Percentage of Peak Value	21
2.5	Effect of Truncating Gaussian DFT (Truncation Length = 93)	23
2.6	Effect of Truncating Gaussian DFT (Truncation Length = 109)	23
3.1	Minimum Phase	27
3.2	Mixed Phase	27
3.3	Maximum Phase	27
3.4	Poles/Zeros of $W_1(z)$	29
3.5	Minimum Phase Sequence	32
3.6	FFT of Sequence	32
3.7	Inverse Filter	32
3.8.1	Poles/Zeros of $W_2(z)$	33
3.8.2	Maximum Phase Sequence	35
3.8.3	Inverse Filter	35
3.9	Mixed Phase Sequence	37
3.10	FFT of Sequence	37
3.11	Inverse Filter for Spike Output	37
3.12	Pole/Zero Location	36
3.13.1	Pole/Zero Location	40
3.13.2	Mixed Phase Wavelet	42
3.14	FFT of Wavelet	42
3.15	Inverse Filter for Output Spike at $\delta(n)$	42
3.16	Inverse Filter for Output Spike at $\delta(n-20)$. .	43
3.17	Inverse Filter for Output Spike at $\delta(n-30)$	43
3.18	Inverse Filter (same as Fig. 3.15)	44
3.19	Average Magnitude as Function of Window Location	44
3.20	RMS Error	44
3.21	Inverse Filter (Same as Fig. 417)	45
3.22	Average Magnitude	45
3.23	RMS Error	45
3.24	Wavelet with a Pair of Zeros on Unit Circle	50
3.25	FFT of Wavelet	50
3.26	Inverse Filter for Spike Output	50
3.27	Sinusoid Wavelet	51
3.28	Inverse Filter for Spike Output	51

Figure		Page
4.1	Algorithm for Finding Inverse Filter	59
4.2	Pulse Sequence	63
4.3	FFT of Pulse Sequence	63
4.4	Spike Inverse Filter	63
4.5	Damped Cosine Sequence	64
4.6	FFT of Damped Cosine Sequence	64
4.7	Spike Inverse Filter	64
4.8	Chirp Sequence	65
4.9	FFT of Chirp Sequence	65
4.10	Spike Inverse Filter	65
4.11	Chirp Sequence; Extended Length	66
4.12	FFT of Chirp Sequence	66
4.13	Spike Inverse Filter	66
4.14	Pulse Sequence	69
4.15	Gaussian Inverse Filter	69
4.16	FFT of Gaussian Inverse Filter	69
4.17	Damped Cosine Sequence	70
4.18	Gaussian Inverse Filter	70
4.19	FFT of Gaussian Inverse Filter	70
4.20	Chirp Sequence	71
4.21	Gaussian Inverse Filter	71
4.22	FFT of Gaussian Inverse Filter	71
4.23	Chirp Sequence; Extended Length	72
4.24	Gaussian Inverse Filter	72
4.25	FFT of Gaussian Inverse Filter	72
4.26	Gaussian Inverse Filter for Wavelet of Figure 3.24	73
4.27	FFT of Gaussian Inverse Filter	73
4.28	Pole/Zero Location	75
4.29	Wavelet with Zeros Close to Unit Circle on Right Half of z-plane	76
4.30	FFT of Wavelet	76
4.31	Spike Inverse Filter	76
4.32	Gaussian Inverse Filter	77
4.33	FFT of Gaussian Inverse Filter	77
4.34	Gaussian Inverse Filter Obtained after Circular Shifting Gaussian FFT	78
4.35	FFT of Gaussian Inverse Filter	78
4.36	Damped Cosine Wavelet	81
4.37	Spike Inverse Filter	81
4.38	Gaussian Inverse Filter	81
4.39	Composite Damped Cosine Wavelet	82
4.40	Filtered Signal Using Spike Filter	82
4.41	Filtered Signal Using Gaussian Filter	82

Figure		Page
4.42	Composite Damped Cosine Wavelet	83
4.43	Filtered Signal Using Spike Filter	83
4.44	Filtered Signal Using Gaussian Filter	83
4.45	Composite Chirp Wavelet	84
4.46	Circular Spike Inverse Filter	84
4.47	Circular Filtered Signal	84

FREQUENCY DOMAIN DECONVOLUTION

I. INTRODUCTION

1.1 Purpose

A problem often encountered in signal processing is the separation of overlapping signals. A composite signal is the combination of two or more different signals that overlap in the time domain. If each of the different signals have different frequency content, then the extraction of each signal can be easily accomplished through bandpass filtering. The difficulty arises when the composite signal is made up of individual signals that are identical, differing only in amplitude and the time of occurrence (See Fig 1.1). Separation of these individual signals (which we shall term "basic wavelets") may not be accomplished through bandpass filtering. Such composite signals are often encountered when dealing with signals that are reflected back from echoing sources. Radar ranging and seismic explorations are two examples where composite signals of overlapping wavelets often occur.

In the analysis of such composite signals, it is often the aim to find the amplitude and time of occurrence of each wavelet. Special deconvolution (or inverse) filters have to be designed for this. Weiner least-squares inverse filters have found widespread use for such purposes [7]. Such filters use the least-square error criterion in solving for the coefficients of the impulse response of the filter. The design procedure involves the formation of a set of

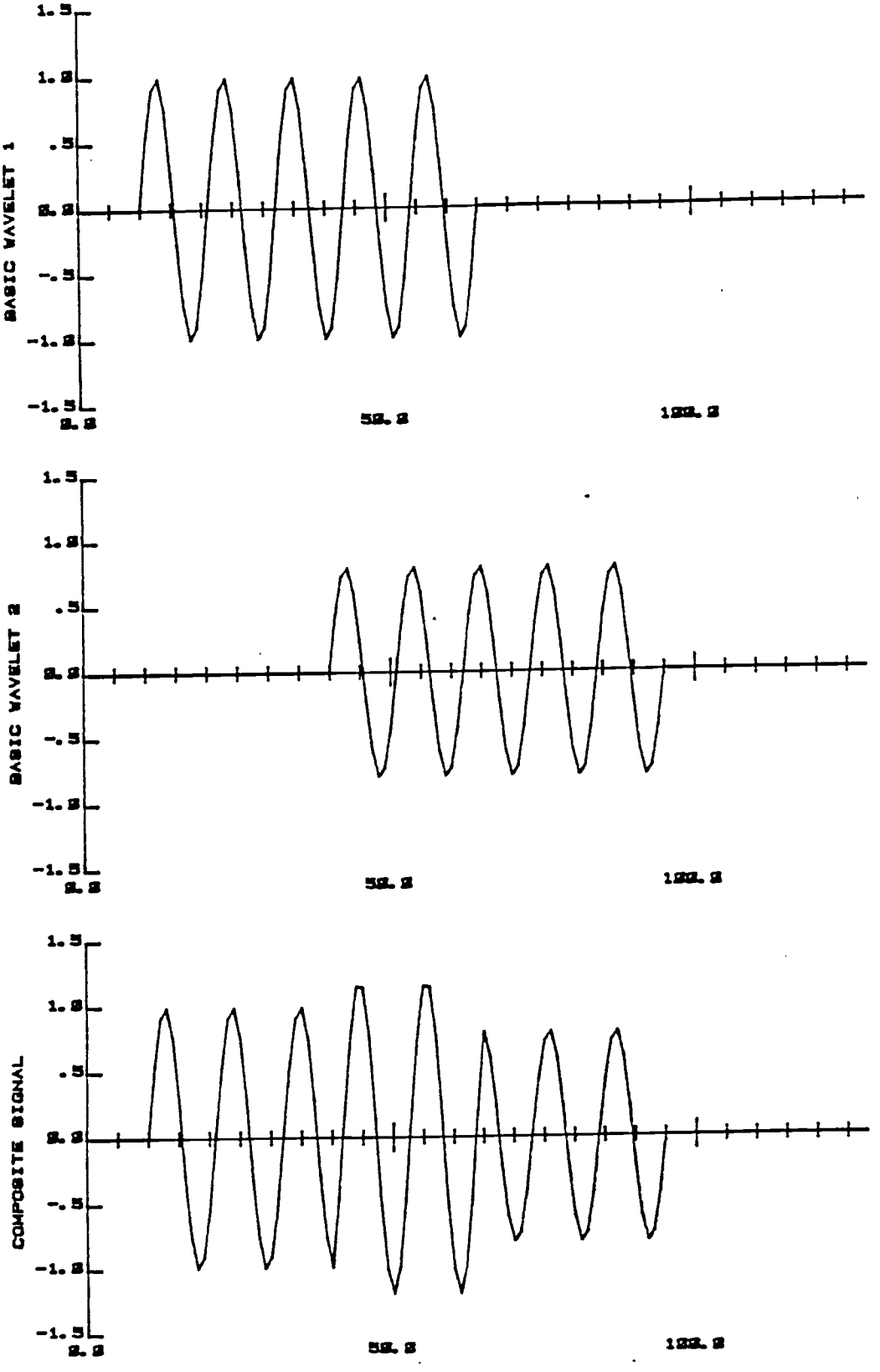


Figure 1.1 - Composite Signals

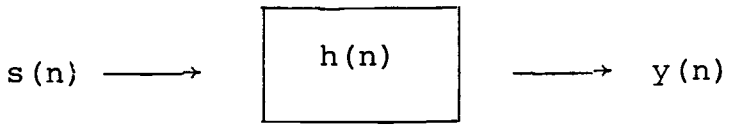
normal equations usually expressed in matrix form. The coefficients of the normal equations are the auto-correlation of the input and cross-correlation of the input and desired output of the filter [11, p. 11]. All analysis is in the time domain and phase information is generally discarded [11, p. 14].

1.2 The Fast Fourier Transform

The design of deconvolution filters may also be accomplished in the frequency domain. This method requires the use of the Discrete Fourier Transform (DFT) of a sampled signal. The Fast Fourier Transform (FFT) is an efficient algorithm for computing the DFT of a sequence. It efficiently calculates samples of the frequency spectrum. In this dissertation the FFT and DFT are used interchangably to mean the same thing. It is the efficiency and time-saving factor that has caused the FFT to find widespread application in digital signal processing. Convolution in the time domain corresponds to multiplication in the frequency domain. It is the apparent simplicity of this relationship that has prompted the author to investigate the practicability of designing deconvolution filters in the frequency domain.

1.3 The Deconvolution Model

The input-output relationship of the linear discrete time system is shown in figure 1.2.



s(n) : Input Signal

h(n) : Linear Shift-Invariant System
(Impulse Response)

y(n) : Output Signal

Figure 1.2 Linear Discrete Time System

In this investigation, only composite signals that are comprised of identical basic wavelets that differ in amplitude and time of occurrence are considered. It is often helpful to consider such signals as a convolution of a reflector wavelet and a reflector series. That is,

$$s(n) = w(n) * r(n) \qquad (1.1)$$

where

s(n) = Composite Signal

w(n) = Reflector Wavelet

r(n) = Reflector Series

* Defines Linear Convolution

NOTE: Reflector wavelet is not the same as a basic wavelet

In this model, the magnitude and time of occurrence of spikes (or pulses) in the reflector series is indicative of the magnitude and location of each wavelet in the composite signal. Each occurrence of a wavelet (as seen in the reflector series) is called an echo.

The input-output relationship of the linear system in figure 1.2 is as follows:

$$y(n) = s(n) * h(n)$$

$$= r(n) * w(n) * h(n)$$

Since the reflector series is desired at the output, we have

$$r(n) = r(n) * w(n) * h(n)$$

This implies that

$$w(n) * h(n) = \delta(n) \qquad (1.2)$$

Where

$$\delta(n) = 1 \; ; \quad n = 0$$

$$= 0 \; ; \quad \text{otherwise}$$

From equation (1.2), one observes that the task is to design a filter that when convolved with the reflector wavelet will produce a single spike at n = 0.

In the z-transform domain, equation (1.2) may be written as:

$$W(z) \cdot H(z) = 1$$

$$\therefore H(z) = \frac{1}{W(z)} \qquad (1.3)$$

where W(z) and H(z) are the z-transform of w(n) and h(n)

respectively. It can be seen that the z-transform of the required filter is the reciprocal of the z-transform of the reflector wavelet. A more detailed discussion of this is presented in Chapter 3.

Consider the case of a single echo. The signal is composed of a single basic wavelet, $w_b(n)$. The output is a reflector series with a single spike (or pulse), $r_s(n)$. Applying this to equation (1.1),

$$w_b(n) = w(n) * r_s(n) \qquad (1.4)$$

The reflector series $r_s(n)$ may be arbitarily chosen to suit a particular design purpose. If $r_s(n)$ is chosen to be a single spike at zero, $\delta(n)$ then the reflector wavelet $w(n)$ is identical to the basic wavelet $w_b(n)$. For each choice of $r_s(n)$, there is a unique reflector wavelet associated with that choice. In designing the deconvolution filter, the basic wavelet $w_b(n)$ is assumed to be known. A preferred reflector series $r_s(n)$ is chosen, and then the associated reflector wavelet is determined from equation (1.4).

For notational purposes, the FFT of each of these sequences is denoted by an uppercase letter, but indexed by 'k'. That is, the FFT of $w(n)$ is $W(k)$. In the Discrete Fourier Transform (DFT) domain, multiplication corresponds to circular convolution in the discrete time

domain. If the linear convolution operator is changed to the circular convolution operator $\circledast$, then equations (1.2) and (1.4) may be rewritten as

$$w(n) \circledast h(n) = \delta(n) \tag{1.5}$$

$$w_b(n) = w(n) \circledast r_s(n) \tag{1.6}$$

Since these are circular convolution operators, the following DFT relationships result directly from equations (1.5) and (1.6).

$$W(k) \cdot H(k) = 1 \tag{1.7}$$

$$W_b(k) = W(k) \cdot R_s(k) \tag{1.8}$$

Solving for H(k), the following is obtained

$$H(K) = \frac{R_s(k)}{W_b(k)} \tag{1.9}$$

Equation (1.9) gives a simple relationship from which the frequency spectrum of the impulse response of the required filter may be obtained.

The problem may arise that if for some value of k, $W_b(k) = 0$, then H(k) cannot be determined from equation (1.9). The solution to this is addressed in section 4.3.

1.4 Overview

Chapter 2 discusses some possible choices for the shape of the reflector wavelt, and presents the advantages of using a Gaussian wavelet.

Chapter 3 presents the theory of using the FFT in designing inverse filters. Several examples involving short sequences are used to clarify the ideas presented in this chapter.

Chapter 4 presents conclusions based on the previous chapters and applies the theory to practical examples. The algorithim for finding the inverse filter via the FFT is defined. The emphasis of this chapter is to show practical examples of inverse filtering in the frequency domain.

II. THE REFLECTOR WAVELET

2.1 Introduction

The deconvolution model presented in the previous chapter shows how the deconvolution filter may be designed based on a knowledge of the basic wavelet and a chosen reflector wavelet. The design of the filter may be done in the frequency domain through the use of the Fast Fourier Transform. As shown in equations (1.5) and (1.6), a basic requirement in the use of the FFT is for the linear convolution operator to be replaced by the circular convolution operator [1, p. 105]. Equation (1.9) gives the frequency domain relationship between the basic wavelet, reflector wavelet and the impulse response of the filter.

In the analysis of composite signals, the first task is often to extract the basic wavelet from the overlapping signals. When this is satisfactorily achieved, the next step is then to deconvolve the composite signal into a reflector series that will show the amplitude and time of arrival of each wavelet. The extraction of the basic wavelet may be accomplished through homomorphic systems for deconvolution. Such systems involve the use of a class of non-linear signal processing techniques. The complex cepstrum technique involves taking the complex logarithm of the FFT of the sampled signals. Such analysis

is generally difficult, and detailed discussion can be found in the reference texts [1, 3, 4].

It will be assumed in this dissertation that the basic wavelet is known and the task is to design a filter or filters that will extract the reflector series. A common choice for the reflector wavelet is a spike. In time domain analysis, the choice of a spike is often desired as it leads to simplifications in the normal equations [5, p. 13]. Such filters are called Wiener spike filters or Wiener inverse filters.

Besides the spike, there are a variety of alternative choices for the reflector wavelet. It will be seen in particular that the Gaussian waveform has several unique properties that gives it a distinct advantage for use as a reflector wavelet. The use of the Gaussian waveform has been suggested [2, p. 342], and part of the objective of this thesis is to determine the practicability and usefulness of using such a reflector series. Section 2.3 discusses the characteristics of a Gaussian waveform curve and the effects of its truncation.

2.2 The Spike Reflector Series

When the spike is chosen to be the reflector wavelet, then,

$$r_s(n) = \delta(n)$$

The FFT of a spike is one, and therefore

$$R_s(k) = 1 \quad \text{for all } k \quad ; \quad k = 0,1..... N-1$$

The frequency spectrum of the impulse response of the deconvolution filter is determined from equation (1.9). This equation then becomes

$$H(k) = \frac{1}{W_b(k)} \qquad (2.1)$$

where $W_b(k)$ is the FFT of the basic wavelet.

The impulse response h(n) is found from the inverse FFT of H(k). In frequency domain design, the advantage of using a spike reflector wavelet is that the frequency spectrum of the filter is found by taking the reciprocal of the frequency spectrum of the basic wavelet. This is possible except when $W_b(k) = 0$ for some value of k. The solution to this problem will be addressed in section 4.3. Since the frequency spectrum of the reflector wavelet need not be taken, computation time is saved in the design of the filter. A second advantage of choosing a spike is that it gives the best resolution compared to any other time series. The time of occurrence and magnitude of each overlapping wavelet is most easily determined from a spike reflector series.

The first disadvantage of using a spike is in the application of the designed filter. The frequency response of the filter as obtained from equation (2.1) is non-zero for all values of k. This implies that when convolution is

performed by using the FFT, every value of H(k) must be used when multiplying the input sequences with the filter. It will be shown in the following section that when the Gaussian waveform is chosen as a reflector wavelet, more than half of the values of H(k) may be approximated by zero.

The second disadvantage of using a spike is that the inverse filter may become very long when zeros of the basic wavelet lie close to the unit circle. Further discussion of this is presented in Chapter 3.

2.3 Gaussian Waveform

This section discusses the time and frequency domain characteristics of a Gaussian waveform. It will be shown that the Fourier transform of a Gaussian waveform is another Gaussian waveform. There exists an inverse relationship between the standard deviation in the time domain (σ_t) and the standard deviation (σ_ω) of the frequency spectrum. For small standard deviations, the curve falls quickly to zero, and it is this property that distinguishes it from other choices of reflector series.

2.3.1 Gaussian Waveform - Continuous Fourier Transform

The Gaussian waveform in continuous time is defined as follows:

$$f(t) = \frac{1}{\sigma_t\sqrt{2\pi}} \exp\left[-\frac{(t-\mu)^2}{2\sigma_t^2}\right]$$

σ_t = standard deviation

μ = mean

Following this definition of f(t), the continuous Fourier transform of f(t), which is F(ω), is found from the integral

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\ e^{-j\omega t} dt$$

The Fourier transform of a Gaussian function is given by the following relationship

$$e^{-\alpha t^2} \longleftrightarrow \sqrt{\frac{\pi}{\alpha}}\ \exp\left[-\frac{\omega^2}{4\alpha}\right] \qquad (2.2)$$

Where "$\longleftrightarrow$" denotes a Fourier transform pair. Letting $\alpha = \frac{1}{2\sigma_t^2}$ and using the scaling theorem of Fourier integrals, transform pair (2.2) becomes

$$\frac{1}{\sigma_t\sqrt{2\pi}}\ \exp\ -\left[\frac{t^2}{2\sigma_t^2}\right] \longleftrightarrow \exp\left[\frac{-\omega^2\sigma_t^2}{2}\right] \qquad (2.3)$$

Alternatively, the pair (2.3) may be rewritten as

$$\frac{1}{\sigma_t\sqrt{2\pi}}\ \exp\ -\left[\frac{t^2}{2\sigma_t^2}\right] \longleftrightarrow \exp\left[-\frac{\omega^2}{2\sigma_\omega^2}\right] \qquad (2.4)$$

where $\sigma_\omega = \frac{1}{\sigma_t}$

From the pair (2.3) it can be seen that the Fourier transform of a Gaussian curve is another Gaussian with a peak magnitude of 1. The peak magnitude also corresponds to the d.c. term which is the area under the f(t) curve. Furthermore, from the tranform pair (2.4), it can be seen that there exists an inverse relationship between the standard deviation (σ_t) of the time function and the standard deviation (σ_ω) of its transform

Thus a signal that is a broad Gaussian distribution will have a narrow distribution for its frequency spectrum. The transform pair for a non-zero mean Gaussian distribution may be obtained by applying the shifting theorem.

$$\frac{1}{\sigma_t\sqrt{2\pi}} \exp\left[-\frac{(t-\mu)^2}{2\sigma_t^2}\right] \longleftrightarrow \exp\left[-\frac{\omega^2\sigma_t^2}{2}\right] e^{-j\mu\omega} \quad (2.5)$$

This is the transform pair relationship for a generalized Gaussian distribution.

2.3.2 - Gaussian Distribution - Fast Fourier Transform

The FFT of a function is the discrete version of the continuous Fourier transform. It is essentially the same as the continuous transform except for three distinct differences. First, the magnitude of the continuous transform is scaled down by a factor of $\frac{1}{\Delta T}$, where ΔT is the sampling period. Second, the FFT is a periodic version of the continuous transform, with periodicity of 2π. Third, the FFT is discrete.

Letting the sampling frequency equal one, then

$$t = n\Delta T = n$$

The discrete Gaussian curve becomes:

$$f(n\Delta t) = f(n) = \frac{1}{\sigma_t\sqrt{2\pi}} \exp\left[\frac{(n-\mu)^2}{2\sigma_t^2}\right] \quad (2.6)$$

Where $\Delta T = 1$

and $n = \pm 1, \pm 2 \ldots \pm M$

for a sequence of length $2M - 1$

If σ_t is greater than 2, then σ_ω is small compared to π and the frequency spectrum may be considered to be bandlimited to within the value of $\pm\pi$. Thus, no aliasing occurs and the FFT is effectively a sampled version of the continuous Fourier transform.

2.3.3 Gaussian Waveform: Finite Sequences

The Gaussian waveform is an infinite two-sided sequence. The sequence has to be truncated in order to take the FFT. Depending on the truncation window length, a small amount of high frequency noise will be added to the frequency spectrum. However, because the Gaussian curve by nature falls very rapidly to zero, noise effect due to truncation is extremely small. By maintaining the peak of the curve at the center of the truncation window, the noise may be kept at a minimum. Thus it may be necessary to shift the sequence before truncation.

2.3.4 Gaussian Distribution - Phase

Time-shifting the Gaussian sequence does not affect the magnitude of the FFT but results in a linear phase. For a sequence of length N, time shifting it by u points to the right results in the following relationship

$$f(n-u) \longleftrightarrow F(k) \exp\left(-j \frac{2\pi}{N} \cdot u\right) \qquad (2.7)$$

Where $f(n) \longleftrightarrow F(k)$ denotes the FFT transform pair

Figure 2.1 shows the FFT of two Gaussian sequences, each of length N. Only the magnitude of the frequency spectrum is shown, as the phase is linear and may be easily deduced from equation (2.7).

2.3.5 Gaussian Distribution - Truncation of FFT Data

The FFT of a Gaussian sequence is another Gaussian sequence that is periodic with period 2π. It has been shown (equation 2.4) that the standard deviation (σ_t) of the time sequence and the standard deviation (σ_ω) of its FFT is inversely related. Figure 2.1 shows the FFT of two Gaussian sequences with different standard deviations (σ_t) of 1.0 and 2.5. It is clear from this plot that the broader sequence with a standard deviation (σ_t) of 2.5 has a corresponding FFT magnitude spectrum that is narrower. The sequence falls off very rapidly to zero and for values of N greater than 30, the magnitude is almost zero. Appendix A is a listing of the exact values of the magnitude spectrum for a Gaussian sequence with a standard deviation (σ_t) of 2.5. For values of N greater than 32 and less than 99, the magnitude falls to the order of less than 1×10^{4}. Appendix B gives the magnitude spectrum listing for a Gaussian sequence of standard deviation (σ_t) equal to 5.0. The magnitude falls even more quickly to zero, and for values of N greater than 17 and less than 112, the order of

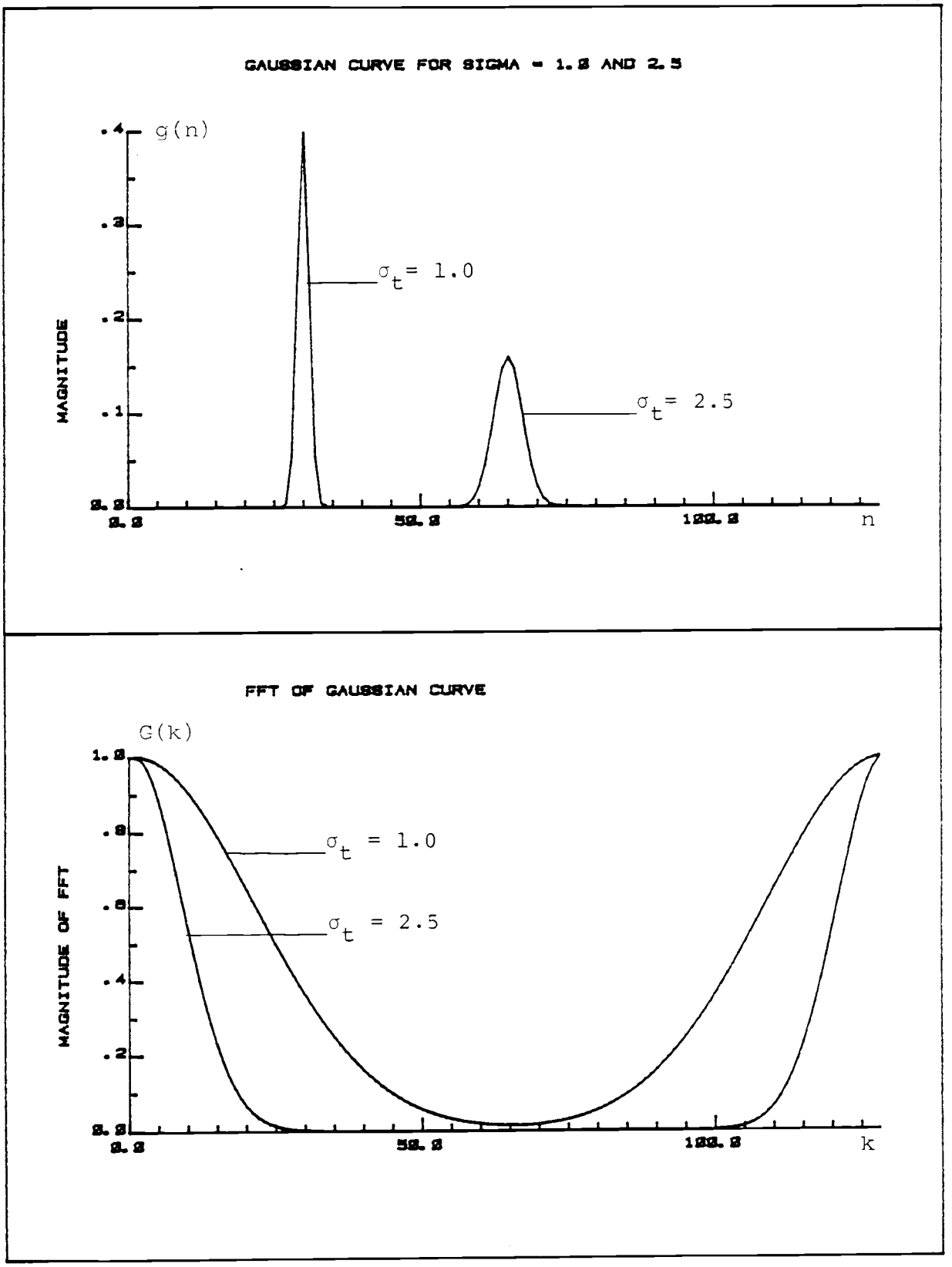


Figure 2.1 : FFT of Gaussian Sequences

magnitude is less than 1×10^{-4}. Double precision accuracy was used for these calculations and it may be assumed that the precision for the FFT magnitude calculations is about one part in 10^{16}. In Appendix B, the values of the magnitude spectrum actually fall below 1×10^{-16}, but because of rounding errors due to the finite accuracy of the machine, this cannot be valid.

Heuristically, the small numbers in the magnitude spectrum may be approximated to zero without significantly affecting the original time sequence recovered by taking the inverse FFT of the transformed sequence. This assumption is made on the basis that most of the information is contained in the main lobe of the Gaussian FFT and the extremely small values may be approximated to zero without losing any information. This approximation is equivalent to the truncation of data. The values of the Gaussian FFT are truncated (approximated to zero) symmetrically about N = 64. Ideally, this is the location of the minimum magnitude for a 128-point FFT sequence. The truncation length, M, is symmetrical about this mid-point.

FFT of Gaussian Sequence

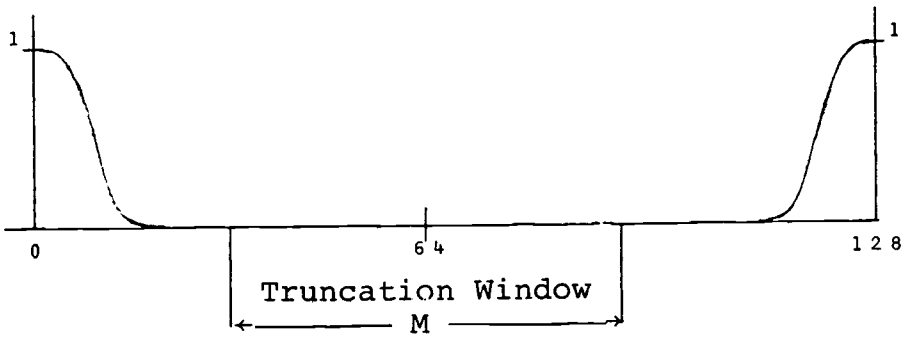


Figure 2.2

Symmetrical Truncation of FFT Data

The inverse FFT was taken of the truncated sequence and compared with the original time sequence. The root mean square (RMS) error was computed for the difference between the two and plotted (Figure 2.3) as a function of the truncation length, M. The error increases with increasing truncation length, but decreases with increasing standard deviation, σ_t. For a Gaussian sequence with $\sigma_t = 2.5$, truncation of 80 data points (out of a sequence of length 128) results in an RMS error of less than 1.6×10^{-4}. The same truncation of 80 points for a Gaussian sequence of $\sigma_t = 5.0$ results in an RMS error of less than 5.4×10^{-10}. The plot shows that a sequence with $\sigma_t = 5.0$ would permit truncation of 100 out of 128 data points with a resulting RMS error of less than 1×10^{-4}.

As the standard deviation increases, the peak of the Gaussian sequence decreases. RMS error of the same magnitude

Fig. 2.3 : Error Due To Truncation Of Gaussian FFT

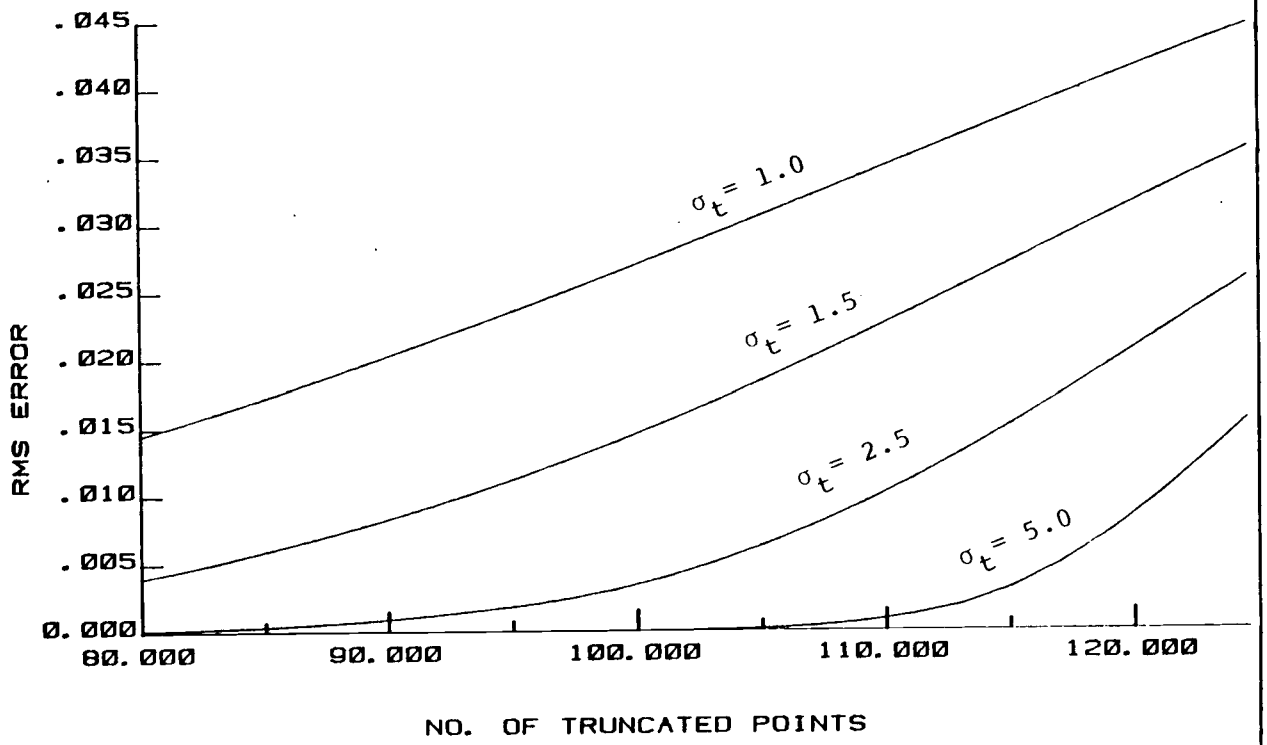
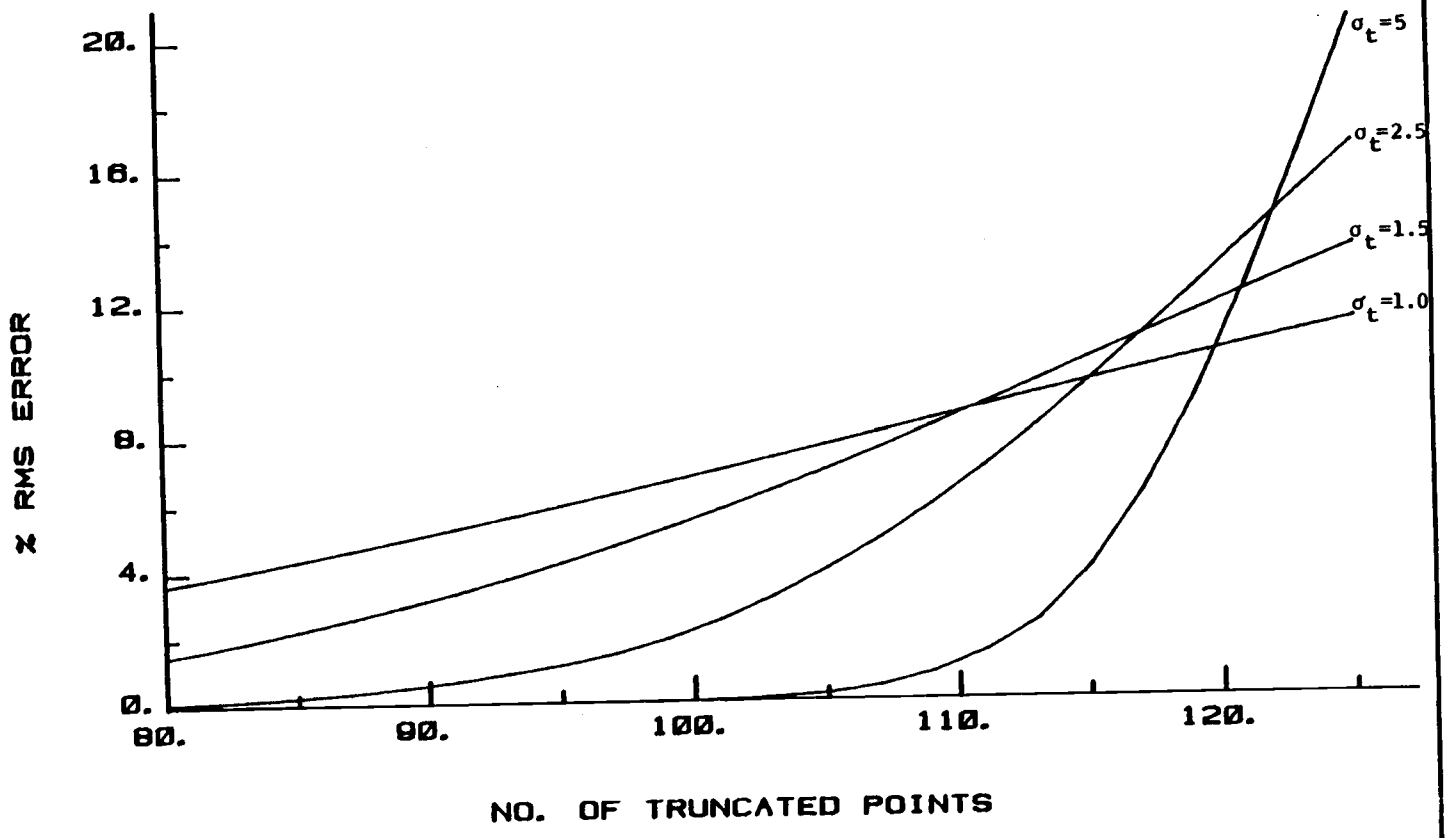


Fig. 2.4 : Truncation Error As Percentage Of Peak Value



will be more dominant in sequences with lower peaks than those of higher peaks. Figure 2.4 is a plot of the RMS error as a percentage of the peak of the time sequence. For a given truncation length, the percentage error does not continuously decrease with increasing standard deviation σ_t. The percentage error gives a measure of the effect of the error upon the peak of the waveform. This would be an important conideration when selecting a suitable standard deviation for a reflector series.

Figures 2.5 and 2.6 show the effect of truncation on the FFT sequence. Notice that the side lobe errors increase and become more prominent for larger truncation lengths. The standard deviation for both sequences was 2.5. In figure 2.5, the truncation length was 93 which resulted in an RMS error of about 1.5×10^{-3} and an RMS percentage error (with respect to the peak) of about 0.8%. In Figure 2.6, the truncation length was 109 which resulted in an RMS error of about 9.5×10^{-3} and an RMS percentage error of about 5.0%. Notice that although the RMS percentage error is only about 5.0%, the maximum error of the first side lobe is about 13% of the peak.

The magnitude of the side lobes can be reduced by proper choice of windowing. For instance if the Hanning window is used instead of the rectangular window for truncation, the peak amplitude of the first side lobe can be

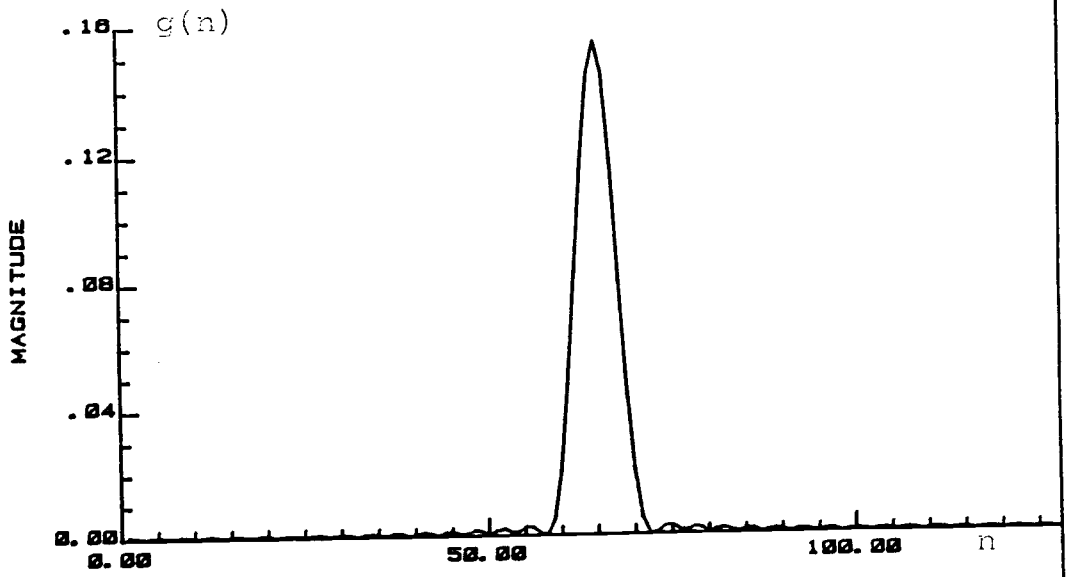


Fig. 2.5 : Effect of Truncating Gaussian DFT.
Truncation Length = 93

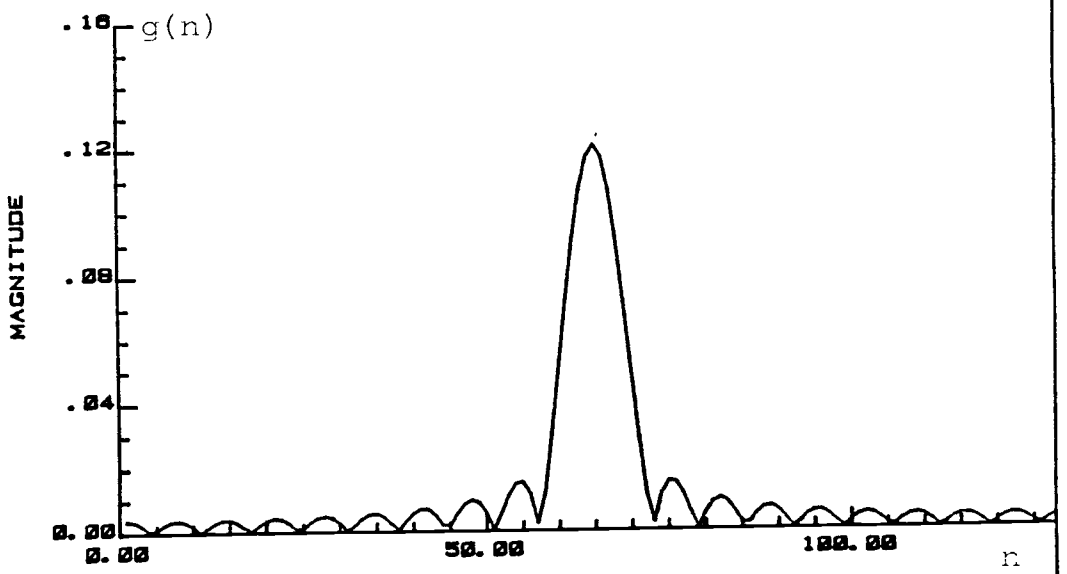


Fig. 2.6 : Effect of Truncating Gaussian DFT.
Truncation Length = 109

reduced by a factor of about 8. However, this is achieved only at the expense of a slightly widened Gaussian waveform. Further discussion on windowing can be found in the reference texts [1, p. 239-255].

2.4 Summary

The FFT of a Gaussian waveform is a Gaussian waveform. A unique characteristic of the Gaussian spectrum is the existence of a large number of points close to zero in the central region of the spectrum. This region may be approximated to zero without introducing significant error.

III. FREQUENCY DOMAIN ANALYSIS

3.1 Introduction

The FFT is an evaluation of the z-transform of a sequence at discrete points evenly spaced on the unit circle. If the reflector series is chosen to be a unit spike at zero, then, as derived in equation (1.3), the z-transform of the inverse filter is the reciprocal of the z-transform of the basic wavelet. That is

$$H(z) = \frac{1}{W(z)} \qquad (1.3)$$

where H(z) and W(z) are the z-transforms of the inverse filter and basic wavelet, respectively. Since W(z) is a polynomial in z^{-1}, it contains zeros, and consequently H(z) contains poles. This implies that the true inverse filter is an infinitely long IIR filter. This chapter discusses the problems of aliasing when the FFT frequency domain method is used to find an approximation to the infinite filter. It also addresses the problem of finding a suitable delay of the spike output when non-minimum phase sequences are encounted.

3.2 Aliasing in the Time Domain

It is a well known fact that any continuous signal has to be sampled at a minimum of twice its highest frequency to prevent aliasing. This is the Nyquist frequency. There exists an analagous relationship between samples of the

Fourier transform and the discrete time domain sequences. It is relevant to recall here that the FFT is a sampling of a continuous function.

Let x(n) be an infinite sequence with a corresponding z-transform of X(z). It can be shown [1, p. 129, problem #22] that when N samples of X(z) are taken along the unit circle at intervals of $\frac{2\pi}{N}$, and the inverse FFT computed, the resulting sequence, x'(n) is an aliased version of x(n).

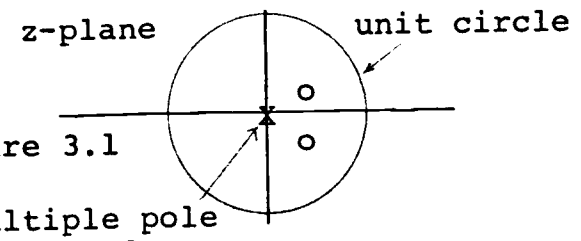
$$x'(n) = x(n + mN) \qquad m = 0,1,2...\infty \qquad (3.1)$$

If x(n) tends to zero for $n \geq N$, then it may be approximated by a finite length sequence.

3.3 The Poles and Zeros of Sequences

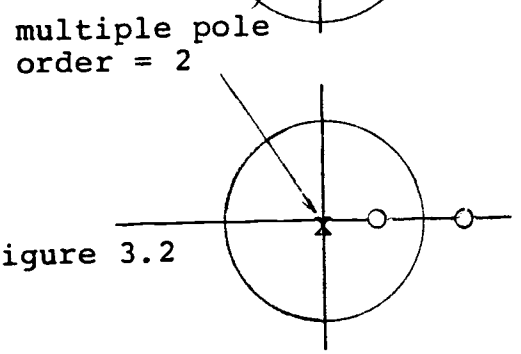
In dealing with finite length sequences (samples of wavelets, etc.) of length greater than one, the z-transform is always characterized by poles at the origin or infinity and zeros distributed anywhere on the z-plane. For a sequence of length N which starts at n = 0, the corresponding z-transform has a multiple pole of order N-1 at the origin, and N-1 zeros on the z-plane. It is convenient to characterize sequences by the location of the zeros on the z-plane. Sequences with all the zeros located within the unit circle are termed minimum phase sequences,

and those with all the zeros located outside the unit circle are termed maximum phase sequences. Mixed phase sequences are those with zeros both inside and outside the circle. It is worthwhile to note also that for sequences with real coefficients, complex valued zeros always occur in conjugate pairs.



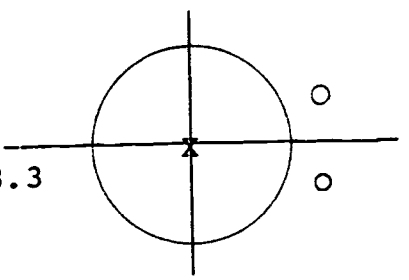
Minimum Phase

Figure 3.1



Mixed Phase

Figure 3.2



Maximum Phase

Figure 3.3

3.4 Analytic Examples

The purpose of this section is to show two analytic examples of inverse filters. The true inverse filter is obtained from equation (1.3) and is infinitely long.

Using the FFT, an aliased version of the infinite filter is obtained. The following two examples will illustrate this fact using a minimum phase and a maximum phase sequence. The examples are an adaption of one of the problems found in the reference text [1, p. 132].

3.4.1 Minimum Phase Sequence

The objective is to design an inverse filter for a spike reflector series for the following wavelet

$$w_1(n) = \delta(n) - 0.5\,\delta(n - 32) \qquad (3.2)$$

The z transform of this wavelet is

$$W_1(z) = 1 - 0.5\,z^{-32} = z^{-32}(z^{32} - 0.5)$$

The zeros are obtained by equating

$$(Z^{32} - 0.5) = 0$$

and hence solving for the zeros.

$$z = \sqrt[32]{0.5}\,\exp\,(j\frac{2\pi}{32}\,k);k = 0, 1, \ldots 31$$

$$= 0.9786\,\exp\,(j\frac{2\pi}{32}\,k);k = 0, 1, \ldots 31$$

This corresponds to 32 zeros inside the unit circle evenly spaced at an angle of $\frac{2\pi}{32}$ radius apart, on a radius of 0.9786.

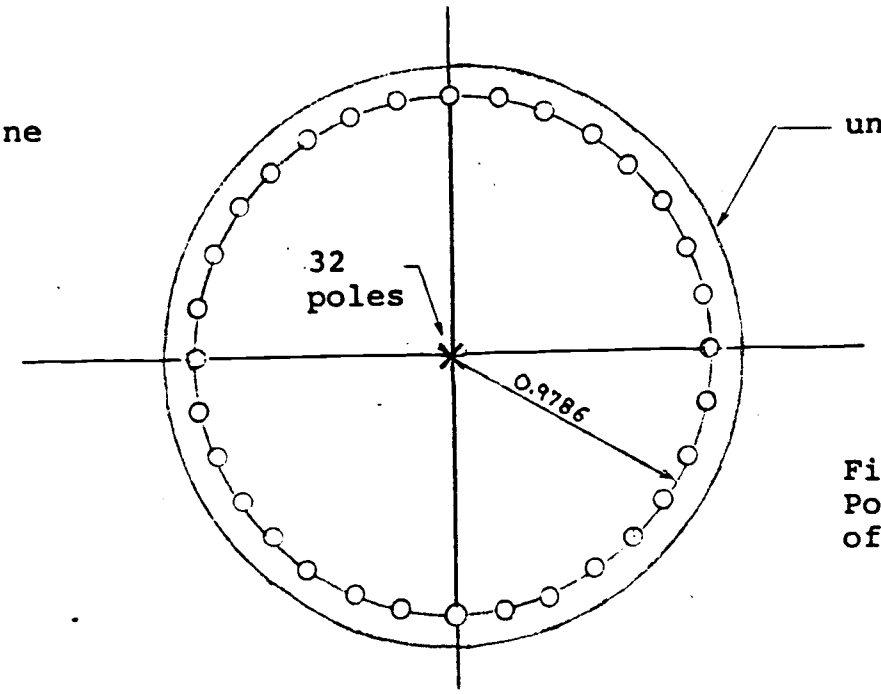


Figure 3.4
Poles/Zeros
of $W_1(z)$

Case 1 Inverse filter obtained using equation (1.3).

The true inverse filter obtained from the z-transform is,

$$H_1(z) = \frac{1}{W_1(z)}$$

$$= \frac{1}{1 - 0.5\, z^{-32}} \qquad (3.3)$$

Using the identity

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1 - x} \qquad |x| < 1$$

and letting $x = 0.5\, z^{-32}$ then

$$H_1(z) = \sum_{n=0}^{\infty} (0.5)^n z^{-32n}$$

Taking the inverse z-transform, this corresponds to a sequence

$$h_1(n) = \sum_{k=0}^{\infty} 0.5^k\, \delta(n - k \cdot 32)$$

Case 2 Inverse filter obtained using FFT method;

equation (2.1)

First, define

$$W_N^k = \exp\left[-j\,\frac{2\pi}{N}k\right]$$

The FFT of the sequence (3.2) is

$$W_1(k) = \sum_{n=0}^{N-1} [\delta(n) - 0.5\,\delta(n-32)]e^{-j\frac{2\pi}{N}nk}$$

$$= 1 - 0.5\,W_N^{32k} \qquad \text{where } N = 128$$

The inverse filter is found by taking the reciprocal of W(k)

$$H(k) = \frac{1}{W_1(k)} \qquad k = 0,1,\ldots 127$$

$$= \frac{1}{1 - 0.5W_N^{32k}}$$

$$= \sum_{n=0}^{\infty} \left[0.5\,W_N^{32k}\right]^n \qquad k = 0,1,\ldots 127$$

$$= \sum_{n=0}^{\infty} (0.5)^n\, W_N^{32nk}$$

By a replacement of variables and partition of the above summation into four sections, it may be shown that

$$H(k) = \frac{16}{15}\left[\sum_{n=0}^{3}(0.5)^n\, W_n^{32kn}\right]$$

Taking the inverse FFT, this corresponds to a sequence

$$h_2(n) = \frac{16}{15}\,[\delta(n) + 0.5\,\delta(n - 32) + 0.25\,\delta(n - 64)$$

$$+ .125\,\delta(n - 96)]$$

Comparing $h_2(n)$ with $h_1(n)$ it can be seen that $h_2(n)$ is an aliased version of $h_1(n)$. Specifically, the relationship is

$$h_2(n) = \sum_{k=0}^{\infty} h_1(n + kN) \quad \text{where } N = 128$$

Figure 3.5 is the minimum phase sequence and figure 3.6 is the corresponding FFT of the sequence. Note the 32 peaks corresponding to the 32 poles and zeros on the z-plane. Figure 3.7 is the resulting filter found by taking the reciprocal of the FFT.

3.4.2 Maximum Phase Sequence

The sequence of equation (3.2) is modified to yield a maximum phase sequence

$$w_2(n) = \delta(n) - 2\,\delta(n - 32) \qquad (3.5)$$

The analysis is similar to the previous sequence. The z-transform corresponds to

$$W_2(z) = 1 - 2\,z^{-32}$$

with corresponding zeros at

$$z = \sqrt[32]{2}\,\exp\left(j\frac{2\pi}{32}k\right)\;;\; k = 0, 1, \ldots 31$$

$$= 1.0219\,\exp\left(j\frac{2\pi}{32}k\right)\;;\; k = 0, 1, \ldots 31$$

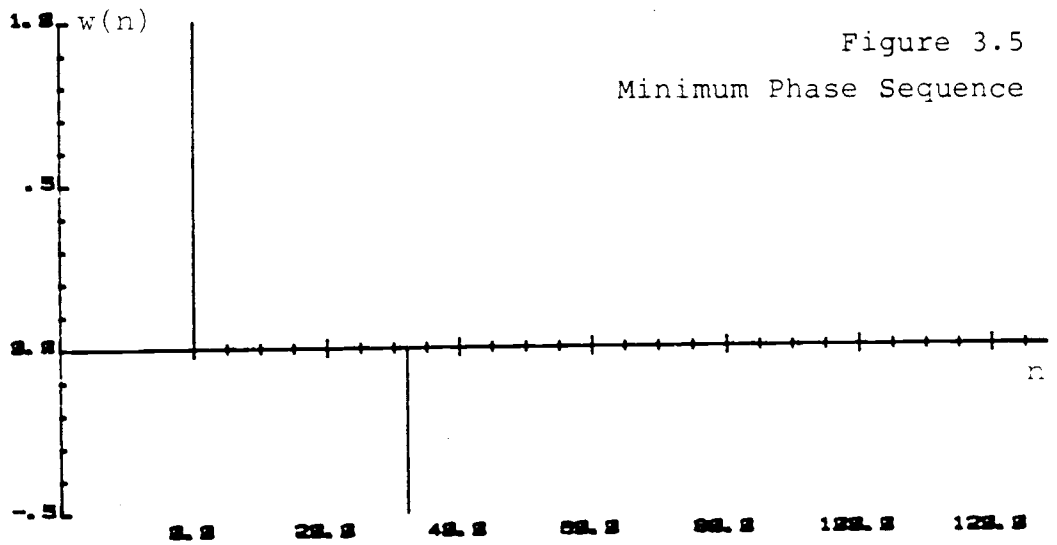


Figure 3.5
Minimum Phase Sequence

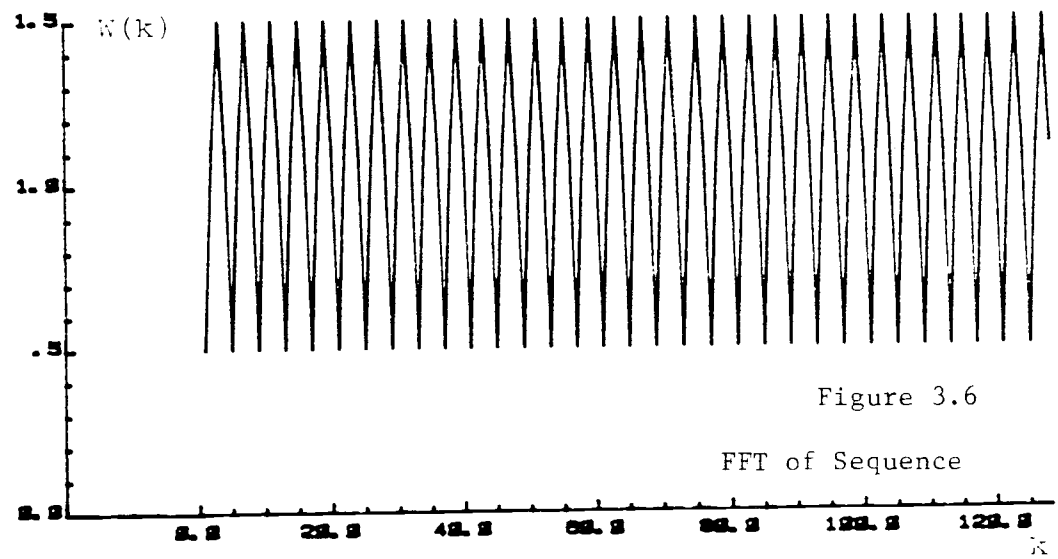


Figure 3.6
FFT of Sequence

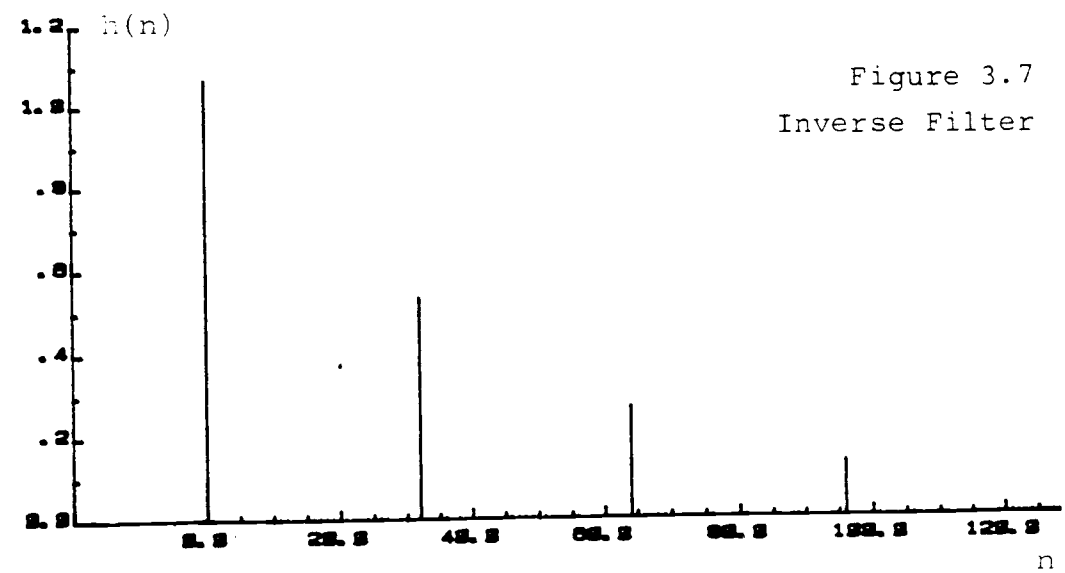


Figure 3.7
Inverse Filter

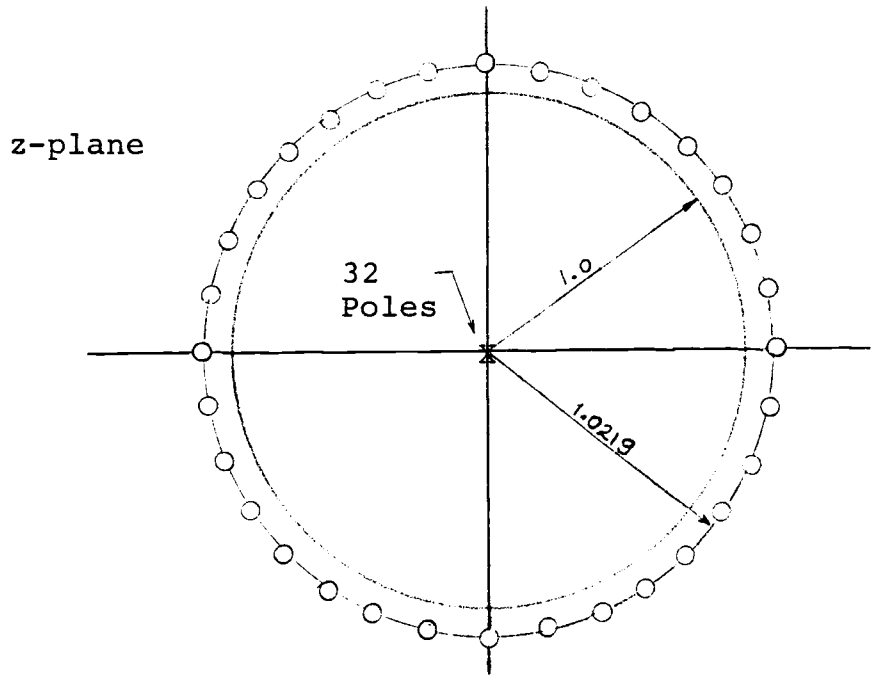


Figure 3.8.1 Poles/Zeros of $W_2(z)$

Note that the zeros lie outside the unit circle; hence for stability, the inverse filter must be a left sided sequence. In this particular example, it can be shown that the z-transform of the inverse filter is identical to that of equation (3.4) except that it is a left sided sequence and the summation index n is from $-\infty$ to 1. Figure 3.8.2 shows the maximum phase sequence of equation (3.5). Taking the reciprocal of the FFT of the sequence gives the inverse filter shown in figure 3.8.3. This is identical to the filter of figure 3.7 except that it is a "left sided" sequence.

Note that in FFT analysis, all sequences are periodic and left-sided stable sequences show up

as those with large magnitude beginning from the right end and decaying towards the left end of the sequence.

3.4.3 Summary/Discussion of Analytic Examples

It has been shown that the use of the FFT in designing the inverse filter results in an aliased version of the true inverse filter. This is due to the fact that FFT analysis involves sampling the frequency spectrum of an infinite filter which results in aliasing in the time domain. However, if the sequence length N is sufficiently large so that the filter coefficients approach zero within the length of N, then aliasing is minimal and practically may be considered neglible. The rate of decay of the filter coefficients is dependent on the location of the zeros of the wavelet on the z-plane. The further the zeros are from the unit circle, the faster the rate of decay. IIR inverse filters that decay rapidly to zero may be approximated by finite length sequences.

Maximum phase sequences have zeros located outside the unit circlewhich become the corresponding poles of the inverse filter. Since the poles are outside the unit circle, stability requires that the sequence be left sided. Approximation of such sequences by a finite length requires that the output spike be delayed. The estimation of delay will be discussed in a later section.

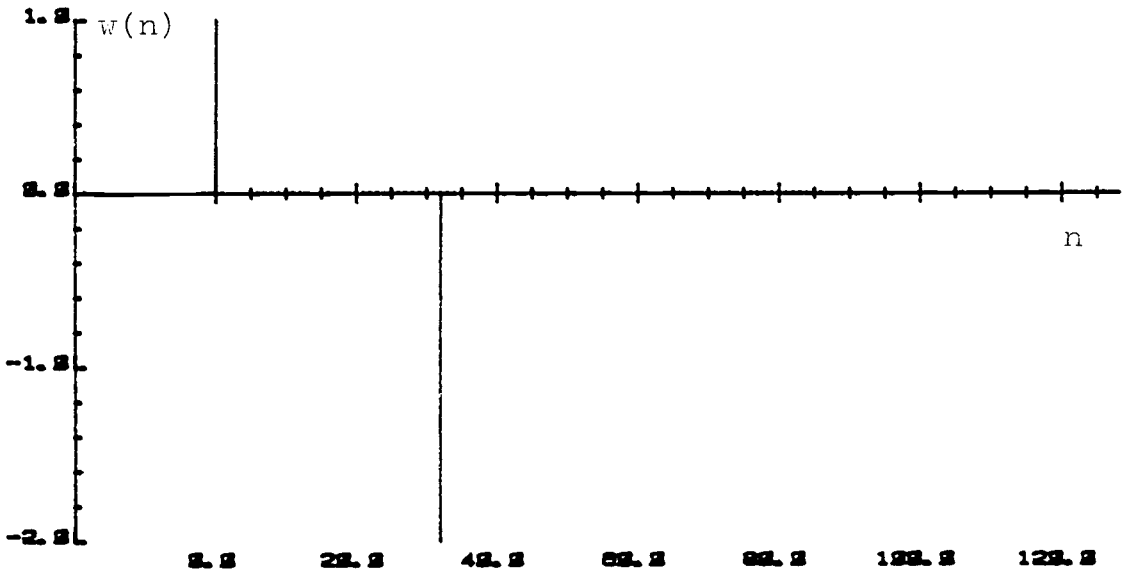


Figure 3.8.2 Maximum Phase Sequence

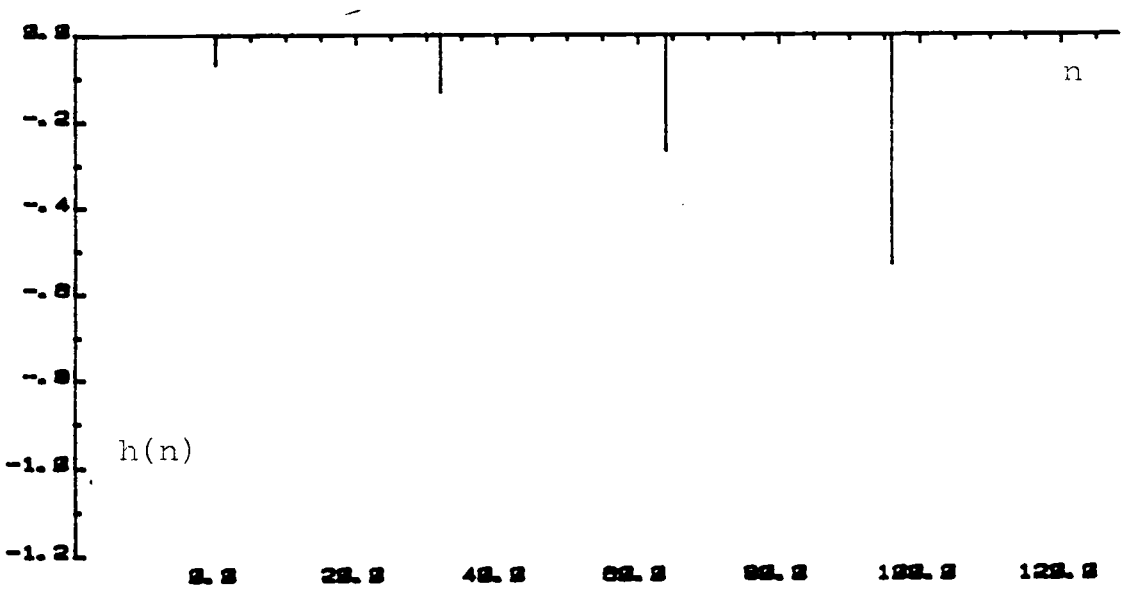


Figure 3.8.3 Inverse Filter

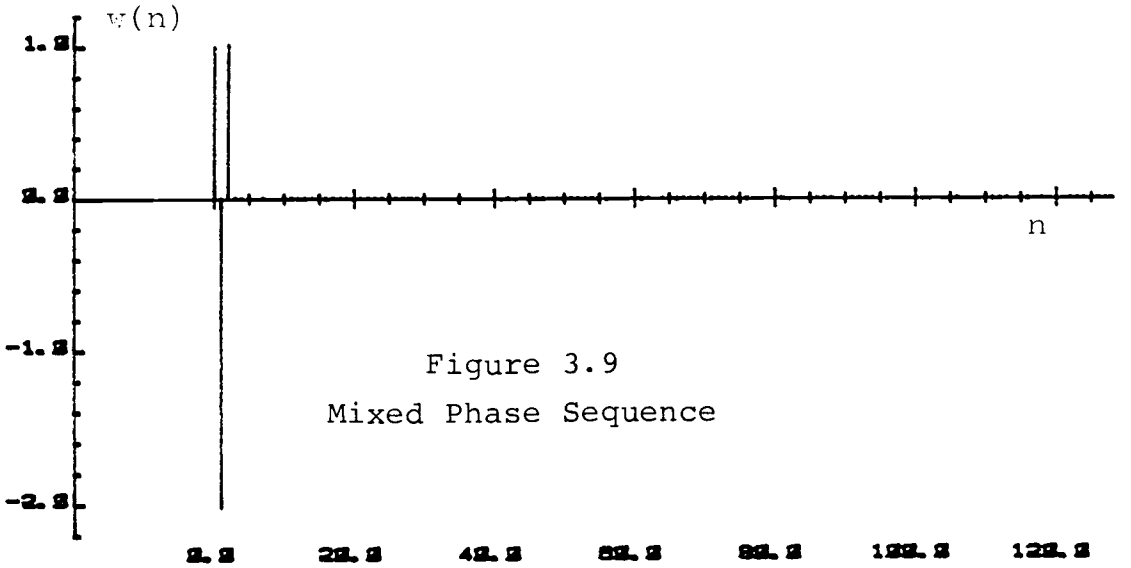


Figure 3.9
Mixed Phase Sequence

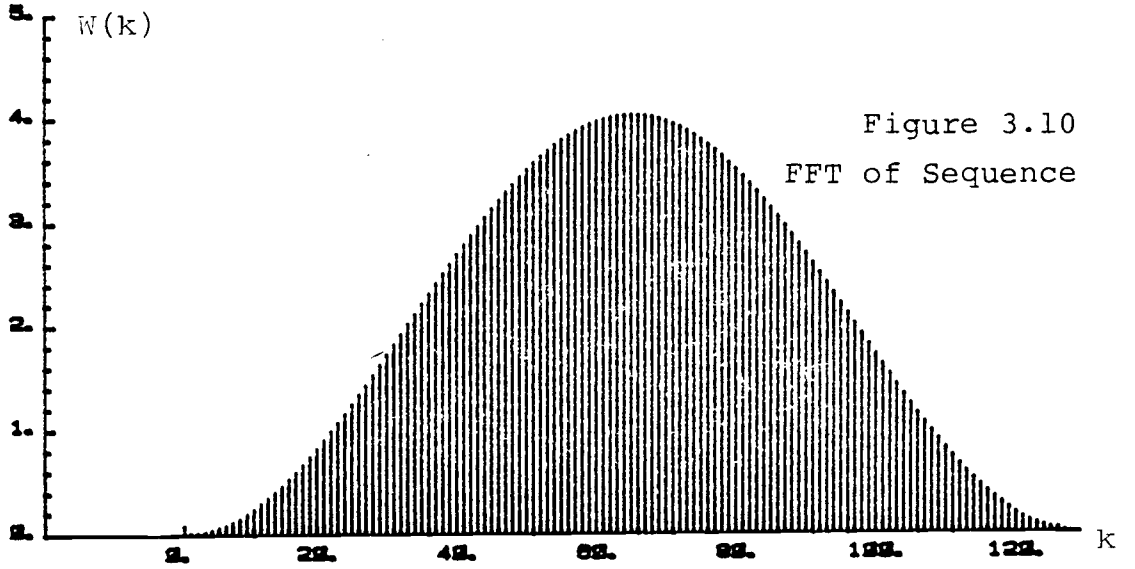


Figure 3.10
FFT of Sequence

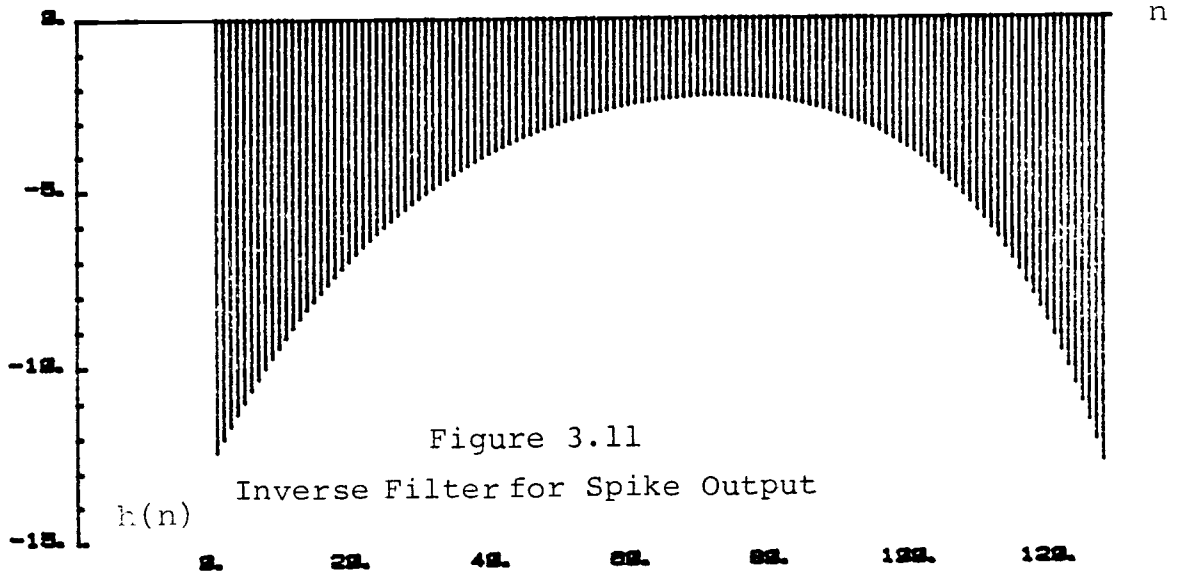


Figure 3.11
Inverse Filter for Spike Output

3.5 Mixed Phase Sequences

In practice, wavelets with zeros inside and outside the unit circle may be encountered. The inverse filters of such mixed phase sequences will have corresponding poles inside and outside the circle. For stability, this implies that the resulting filter is a combination of left and right sided sequences. In frequency domain analysis using the FFT, such two-sided filters show up as sequences with large co-efficients that begin at both ends but decay towards the center.

Figure 3.9 shows a three-point mixed phase sequence. The mathematical representation of the sequence is

$$w(n) = \delta(n) - 2.02\delta(n - 1) + 1.0185\ \delta(n - 2)$$

and

$$W(z) = z^{-2}(z^2 - 2.02z + 1)$$

The corresponding zeros are at

$$Z_1 = 0.97 \quad \text{and} \quad Z_2 = 1.05$$

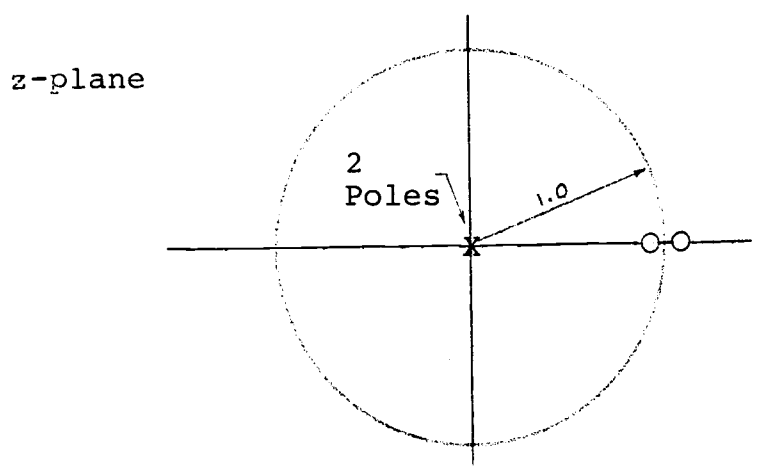


Figure 3.12 Pole/Zero Location

Figure 3.12 shows the pole/zero location and figure 3.10 shows the corresponding FFT of the sequence. Notice that the zeros are located very close to the unit circle and the damping factor for the corresponding poles is low. For a sequence of length N = 128, as seen in figure 3.11, the left and right sided sequences are aliased at the central portion. Aliasing may be avoided by extending the sequence.

3.6 Estimating Delay of Output Spike

Inverse filters are IIR filters. Frequency domain deconvolution (involving the use of the FFT) results in filters that are an aliased version of the IIR filters. However, by using a sufficiently long sequence, aliasing is minimized and for most practical considerations may be considered negligible. If the IIR inverse filter is sufficiently damped, it may be approximated by a finite length filter. The purpose of this section is to discuss the approximation method and determine the required output delay (if necessary).

3.6.1 Windowing the Sequences

The truncation of a long sequence is most easily understood in terms of windowing. For an FFT filter sequence of length N, one may choose to implement a filter

of length L, where L is less than N. This may be thought of as windowing the sequence of length N with a rectangular window of length L. The question then is to decide over which region of the sequence to place the window. For minimum error of truncation, one would intuitively place the window over the region where the average magnitude of the coefficients are maximum. This would simply involve a search through the sequence to find the maximum average magnitude based on a length L. A more detailed discussion of this is presented in Section 3.6.2. Remember that in dealing with the FFT, the sequences are periodic.

Once the optimum window position has been determined, the sequence has to be circularly shifted so that the window begins on the left end of the sequence. This is necessary because the objective is to perform linear instead of circular convolution. The number of right-shifts required to place the window on the left end of the sequence corresponds to the delay of the output for the given filter length. It should be noted at this point that for linear convolution with the FFT, the length of the windowed filter coefficients L, plus the length of the wavelet must be less than N.

3.6.2 Example of Delaying Output Spike

To clarify the concepts presented in the previous section, it is appropriate at this point to illustrate by example the method of determining the output delay. Figure 3.13.2 shows a mixed phase sequence.

Mathematically the sequence is represented as:

$$\omega(n) = \delta(n) - 0.25\ \delta(n-1) - 3\ \delta(n-2) - 1\ \delta(n-3)$$

The corresponding z-transform is

$$W(z) = z^{-3}\ (z^3 - 0.25z^2 - 3Z - 1)$$

The zeros of W(z) are located at

$$z_1 = 2$$

$$z_2 = -0.3596$$

$$z_3 = -1.3904$$

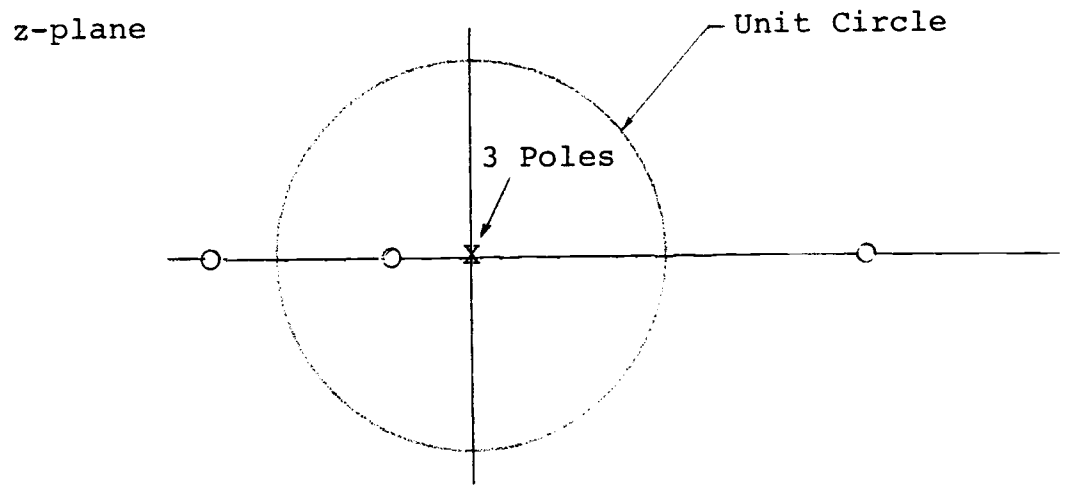


Figure 3.13.1

Pole/Zero Location

Figure 3.14 is the FFT of the wavelet. Notice the two peaks and valleys as characterized by the location of the zeros.

Figure 3.15 is the corresponding zero-delay inverse filter for a spike at $\delta(n)$. Because the poles of $\frac{1}{W(z)}$ are both inside and outside the unit circle, the resulting filter is a two sided sequence. In this example, it can be seen that aliasing is negligible.

This zero-delay filter can only be implemented by circularly convolving it with the wavelet. In most cases, circular convolution is not practical. It is thus the objective to seek a solution so that linear convolution instead of circular convolution is performed. This is achieved by sufficiently delaying the output so that the filter coefficients begin on the left end of the sequence, and then approximating the sequence with a finite length filter.

Figure 3.16 and Figure 3.17 show the corresponding inverse filters designed for delayed outputs. Special attention should be given to the fact that figures 3.15, 3.16 and 3.17 are related by circular shifts of the original zero-delay filter. The number of right-shifts equal the number of delays introduced at the output.

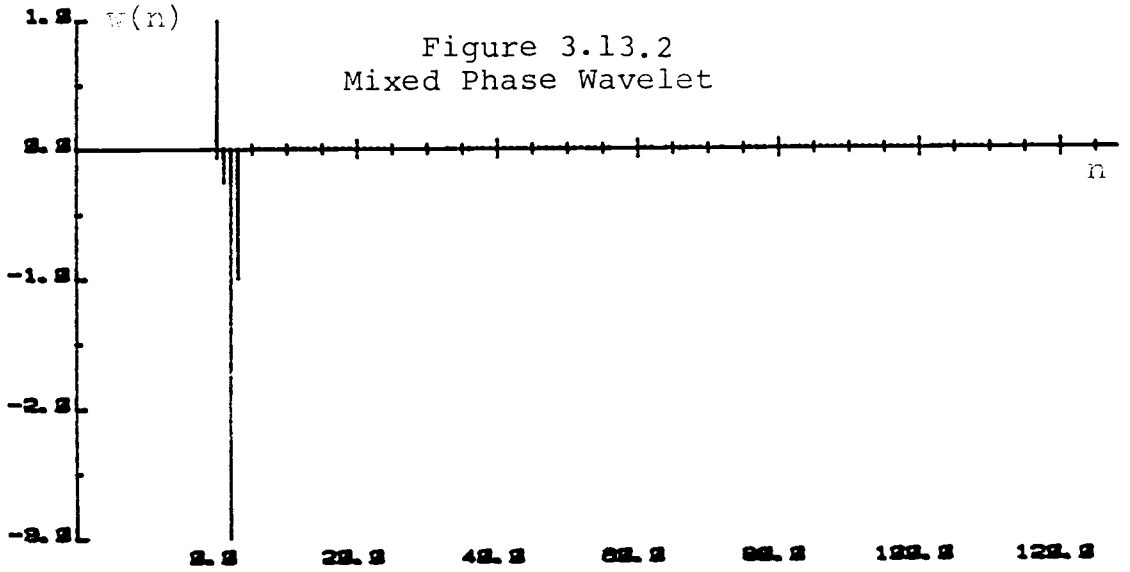


Figure 3.13.2
Mixed Phase Wavelet

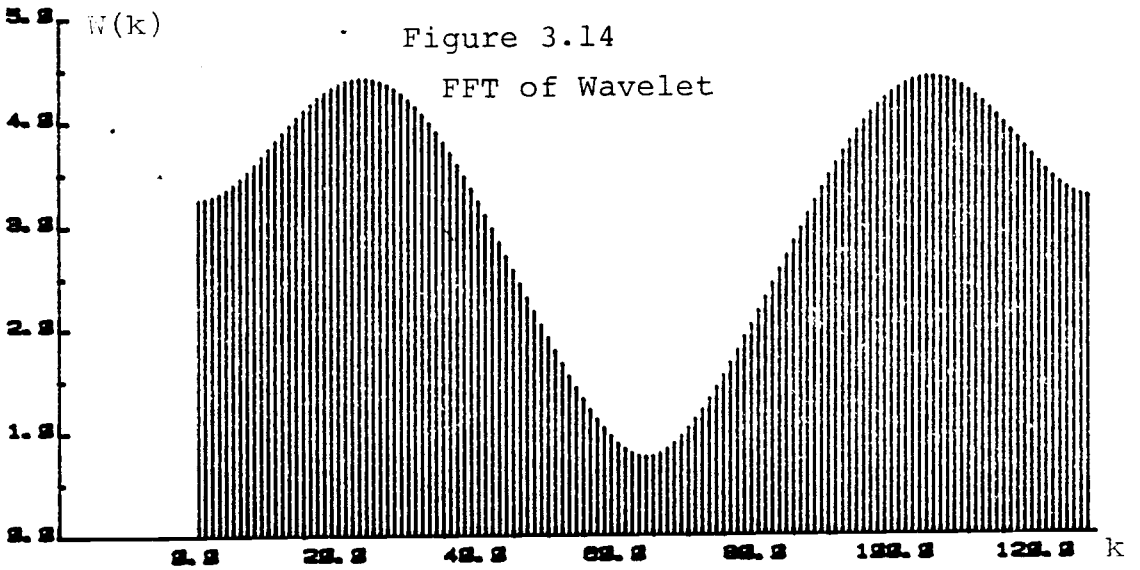


Figure 3.14
FFT of Wavelet

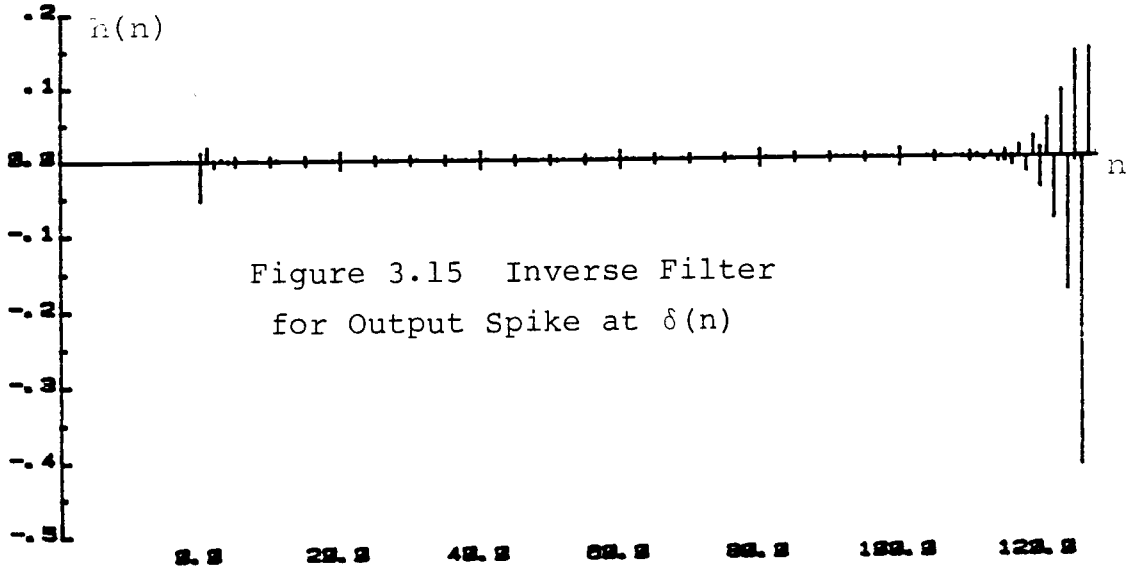
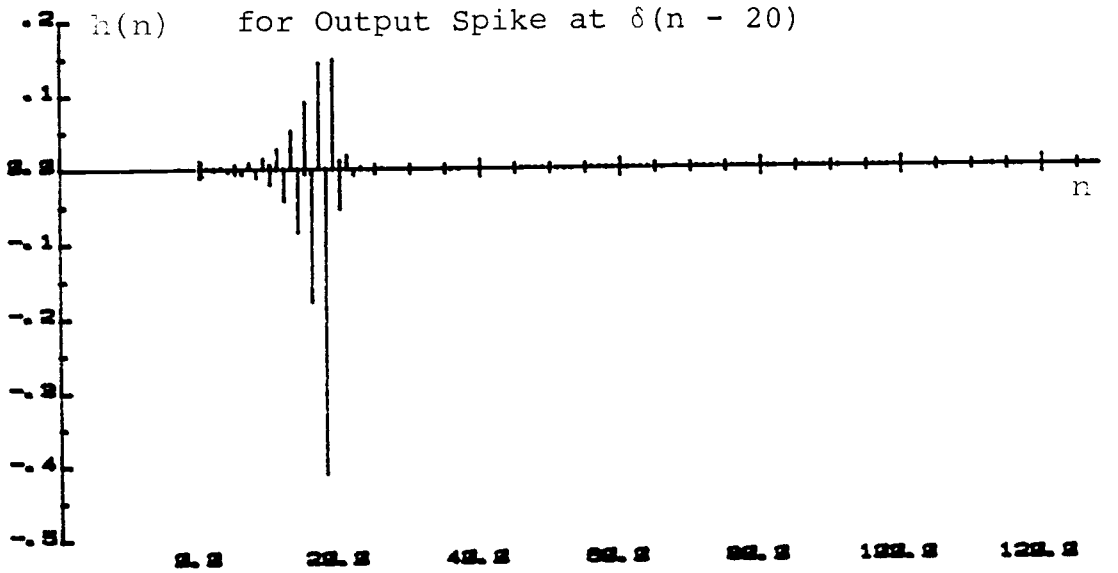


Figure 3.15 Inverse Filter
for Output Spike at $\delta(n)$

Figure 3.16 Inverse Filter
for Output Spike at $\delta(n - 20)$



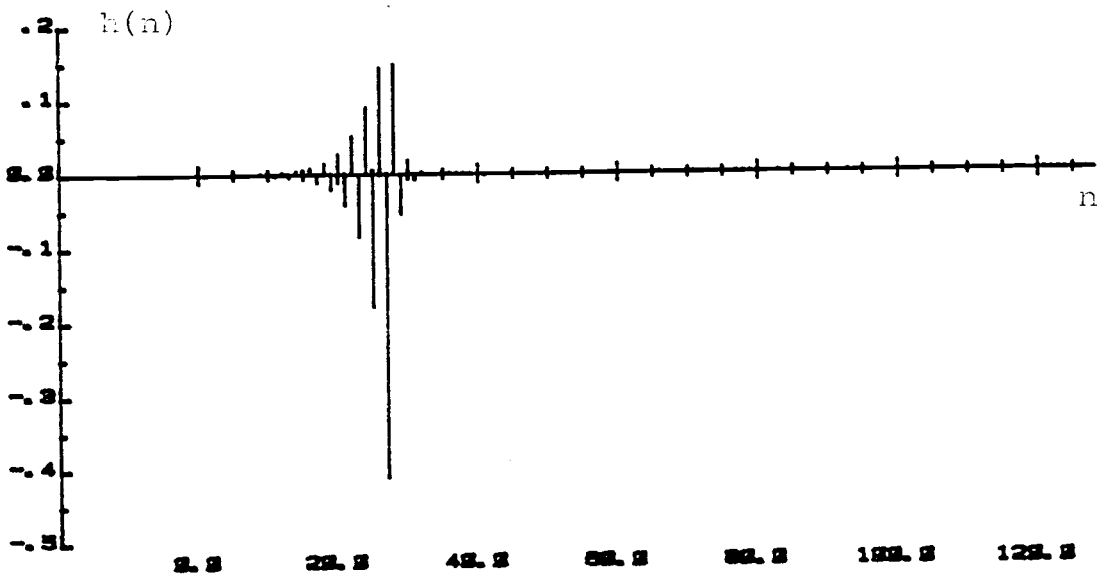


Figure 3.17 Inverse Filter
for Output spike at $\delta(n - 30)$

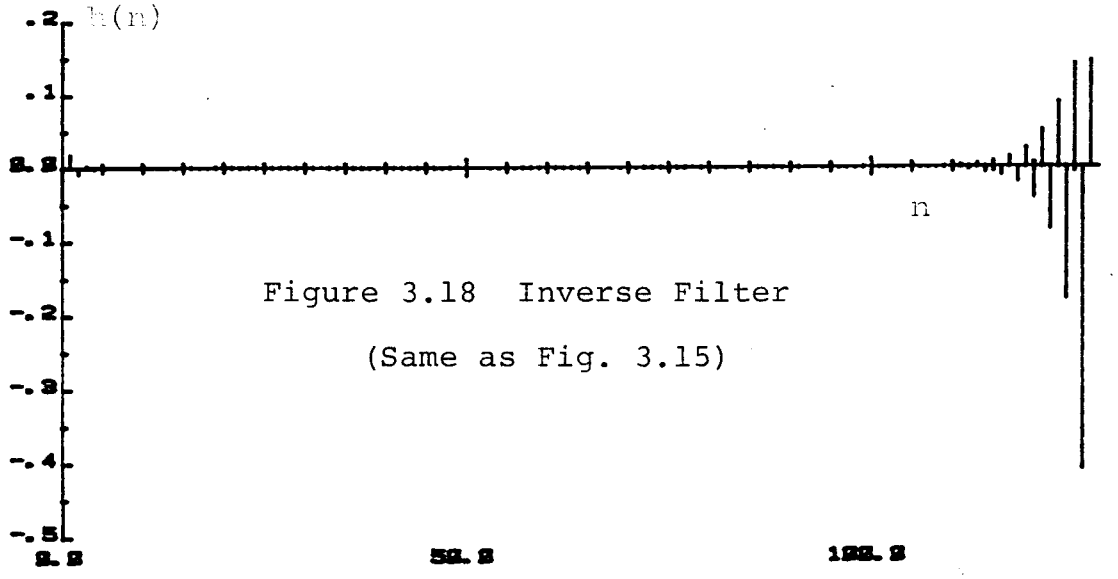


Figure 3.18 Inverse Filter

(Same as Fig. 3.15)

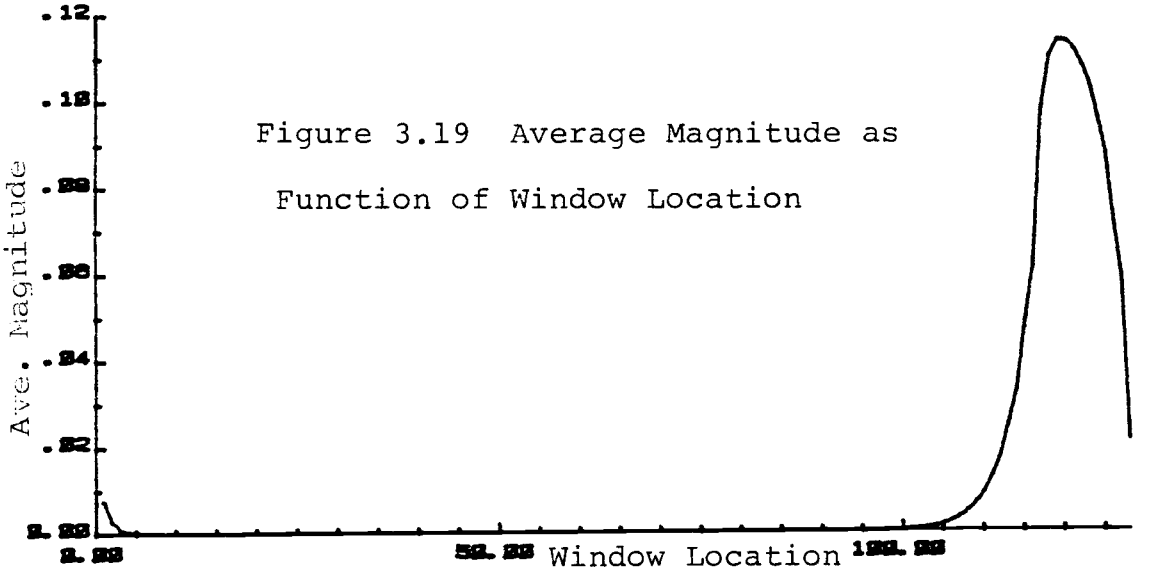


Figure 3.19 Average Magnitude as Function of Window Location

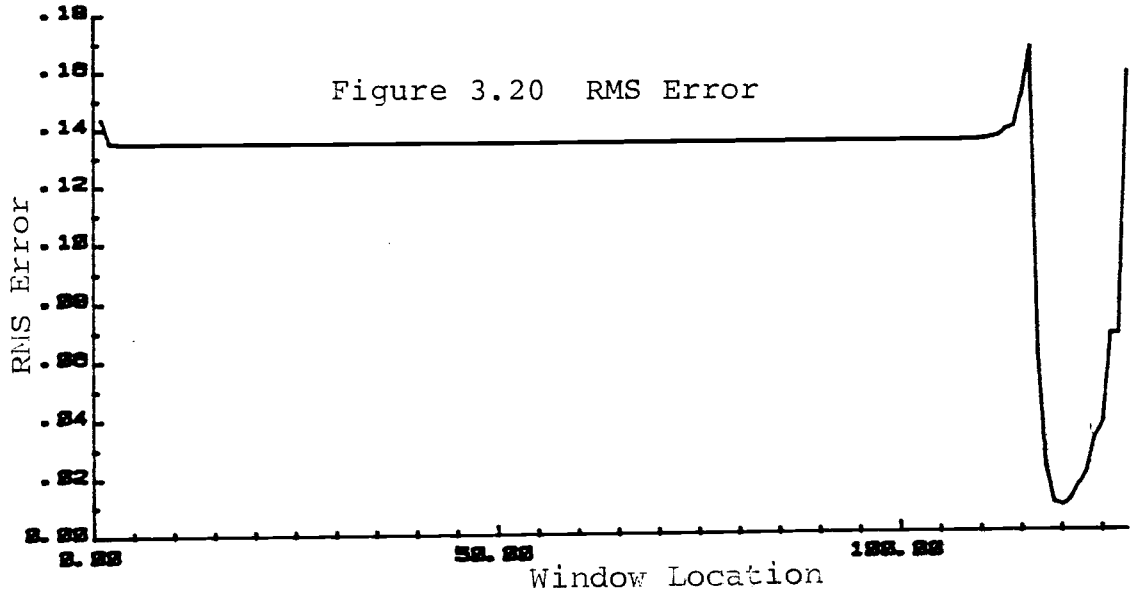


Figure 3.20 RMS Error

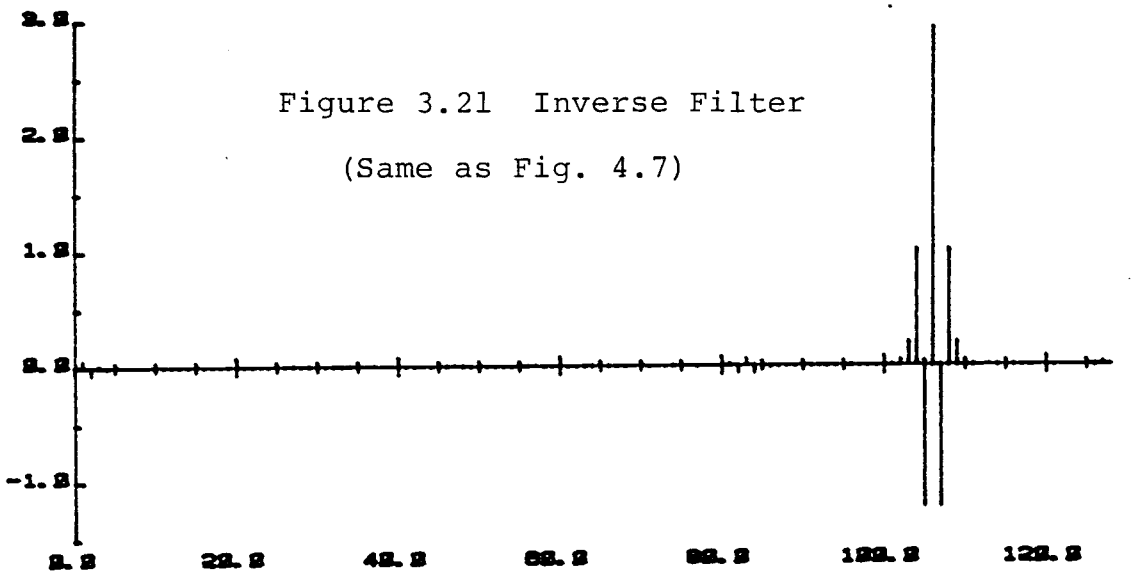


Figure 3.21 Inverse Filter

(Same as Fig. 4.7)

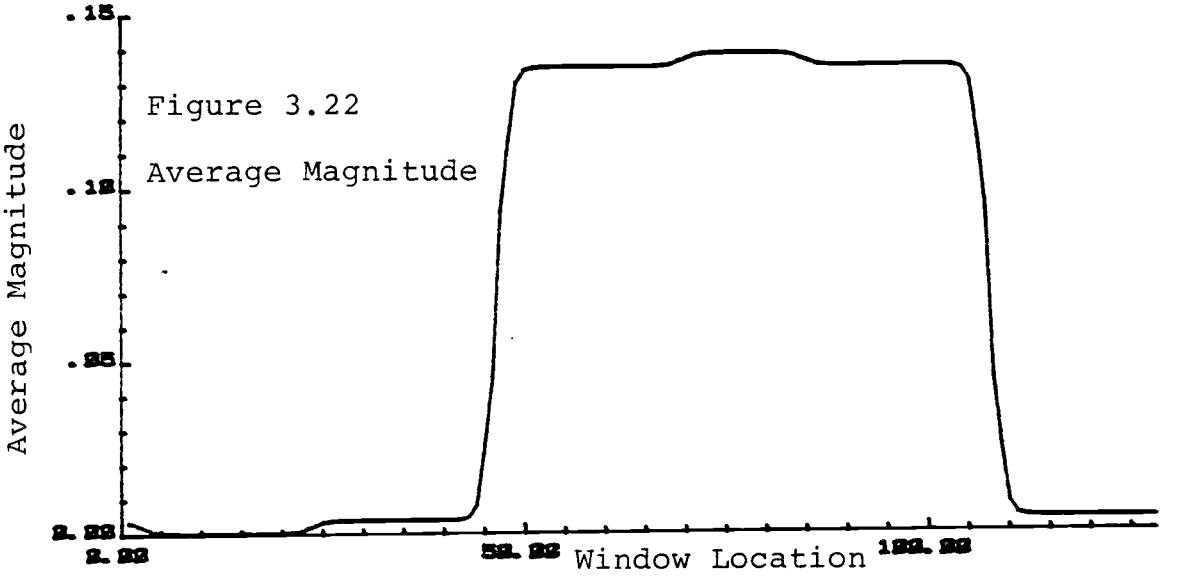


Figure 3.22

Average Magnitude

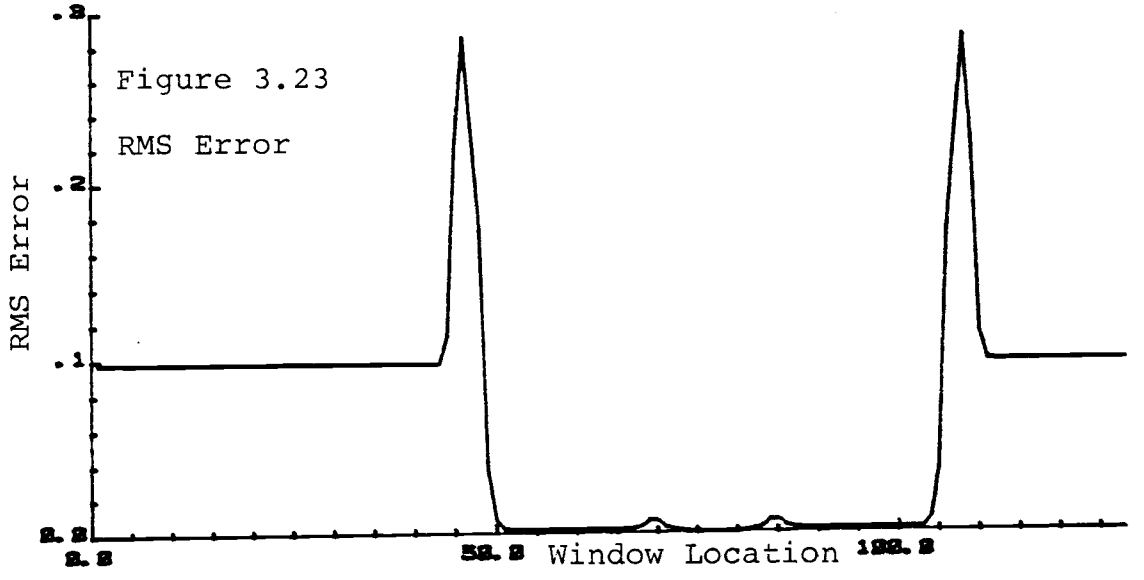


Figure 3.23

RMS Error

3.6.3 Determining the Optimum Window Position

The inverse filter found via the DFT method is a long sequence and has to be approximated by a shorter finite length filter. This is achieved by rectangular windowing the long sequence. The decision of where to place the window is critical as it will determine the resulting RMS error of filtering. For minimum RMS error, it was intuitively suggested (in Section 3.6.1) to place the window over the region where the average magnitude of the filter coefficients is maximum. This was tested and it was found that it is indeed a reliable method of choosing the optimum window position. Several simulations were performed and Figure 3.18 to Figure 3.20 show one set of typical results. The filter tested was the same as that of Figure 3.15. Ths window length was 11 and the window location is defined by using the left edge of the window as the reference point. Figure 3.16 shows the average magnitude of filter coefficents as a function of window location. As the window was shifted to the right, the average magnitude rises because of the large coefficients on the right end. Figure 3.20 shows the RMS error as a result of linearly convolving the windowed sequence with the basic wavelet. The most interesting result was found by comparing the average magnitude with the RMS error. The minimum error and the maximum average

magnitude occurs at exactly the same window position.

A second example is shown in Figures 3.21 to 3.23. The filter used is that of Figure 4.7, and the window length was 61. Similar to the previous example, it was found that minimum RMS error and maximum average magnitude occurs at the same window position.

Further investigation shows the result to be true to within one sample point in the difference of the window positions.

Once the optimum window position is determined, the output delay as a result of implementing the windowed filter with linear convolution can be easily calculated. It is equal to the number right-shifts required to place the window at the left end of the sequence.

3.7 Quasistable Inverse Filters

The case of quasistable inverse filters occurs when at least one zero of the wavelet falls directly on the unit circle. The corresponding inverse filter has a pole at the same location on the unit circle which causes infinite aliasing. The z-transform of the inverse filter can be separated into partial fractions corresponding to each pole in the denominator. If the partial fraction factor for that pole on the unit circle is very small and close to zero compared to the other factors, then its effect may be

negligible. In general, this may not be assumed and the pole on the unit circle will result in a non-decaying infinite length filter that cannot be approximated by a finite sequence.

Example 1:

Figure 3.24 shows a three-point sequence that has zeros on the unit circle.

$$w(n) = \delta(n) + a\,\delta(n-1) + \delta(n-2)$$

where $a = 1.931851653$

The zeros occur as a conjugate pair at

$$z = \exp(\pm\theta) \qquad \text{where } \theta = 165°$$

The resulting inverse filter is unstable and aliased. It cannot be approximated by a finite length sequence. Hence a filter to deconvolve it into a spike cannot be found. However, it will be shown in section 4.22, figure 4.26 that a deconvolution filter for a Gaussian reflector series can be found.

Example 2:

The truncated sinusoid is a classic example of a wavelet with zeros located on the unit circle. For a sequence of length L, it has multiple poles of degree L-1 at the origin, and L-1 zeros evenly spaced on the unit circle. Clearly, one would intuitively expect the inverse filter to be unstable. Figures 3.27 and 3.28

show the wavelet and the corresponding inverse filter.

3.8 Circular Convolution vs. Linear Convolution

Circular convolution involves treating finite length sequences as periodic. If the trailing edge of each sequence has sufficient consecutive zero values, then circular convolution is equivalent to linear convolution of the sequences. Filters that must be implemented with circular convolution have several major disadvantages. First, convolution must be performed with sequences of the same length and the filter may not be extended by appending zeros. Second, such filters cannot be used on sequences that are longer than the filter length. Methods of filtering long sequences such as overlapp-add cannot be applied. It therefore implies that a circular convolution filter of length N can only be applied to data sequences that are appended by zeros to a maximum length of N.

3.9 Circular Inverse Filters

In chapter I, the deconvolution model was presented and it was shown that to obtain a spike reflector series, the filter co-efficients have to be such that Equation 1.2 holds

$$w(n) * h(n) = \delta(n) \qquad (1.2)$$

where w(n) and h(n) are the basic wavelet and the inverse filter respectively.

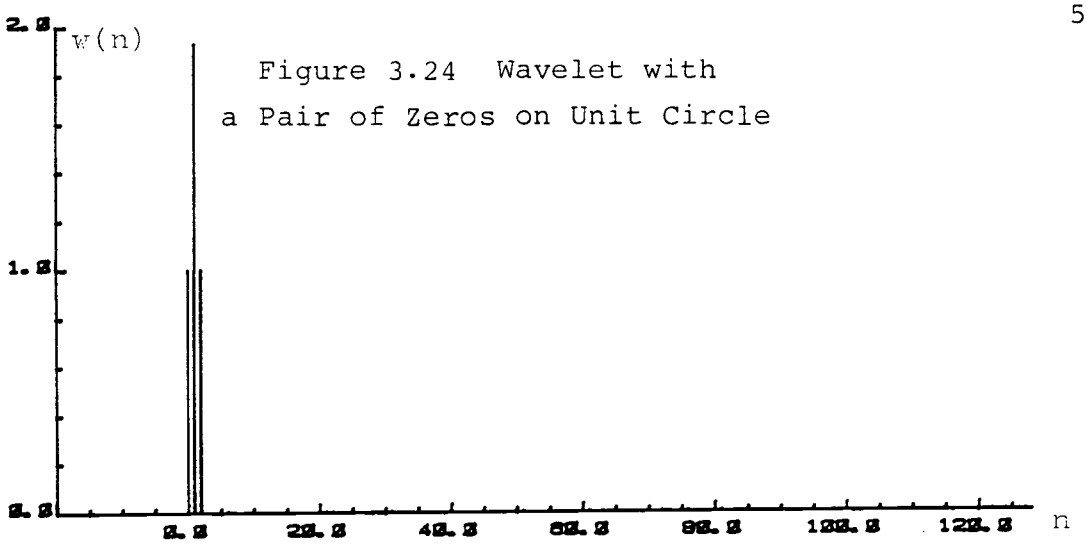


Figure 3.24 Wavelet with a Pair of Zeros on Unit Circle

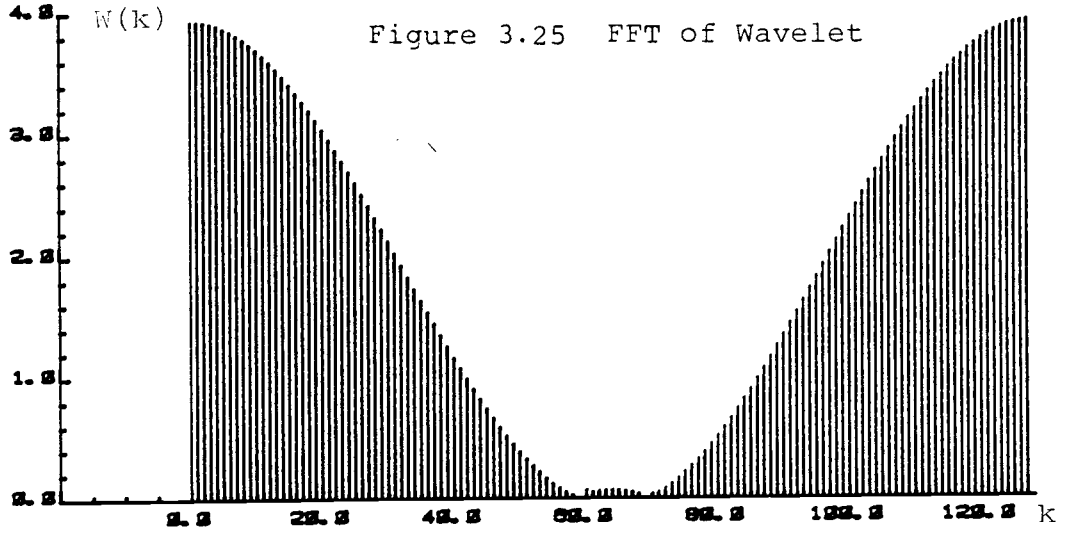


Figure 3.25 FFT of Wavelet

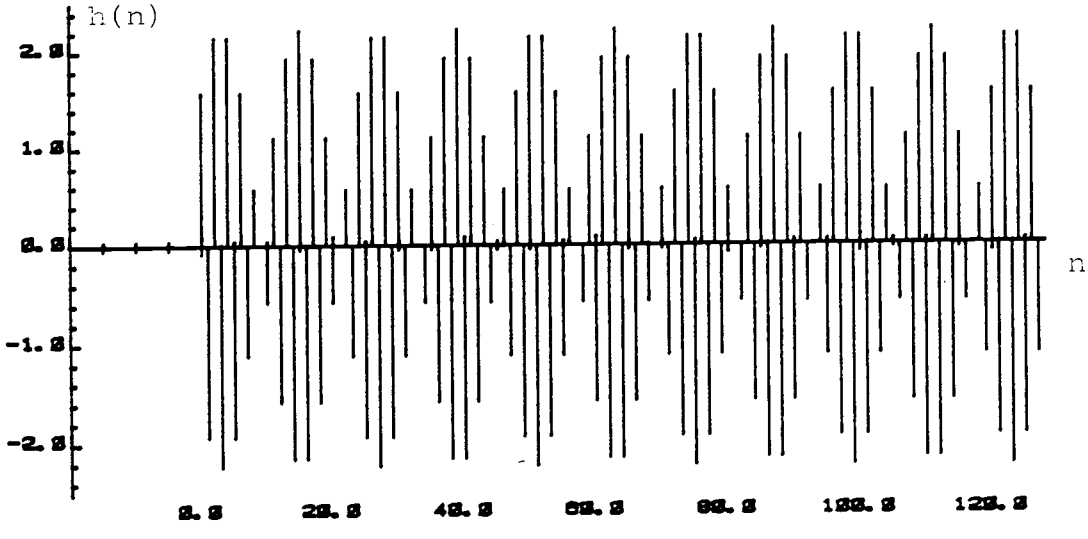


Figure 3.26 Inverse Filter for Spike Output

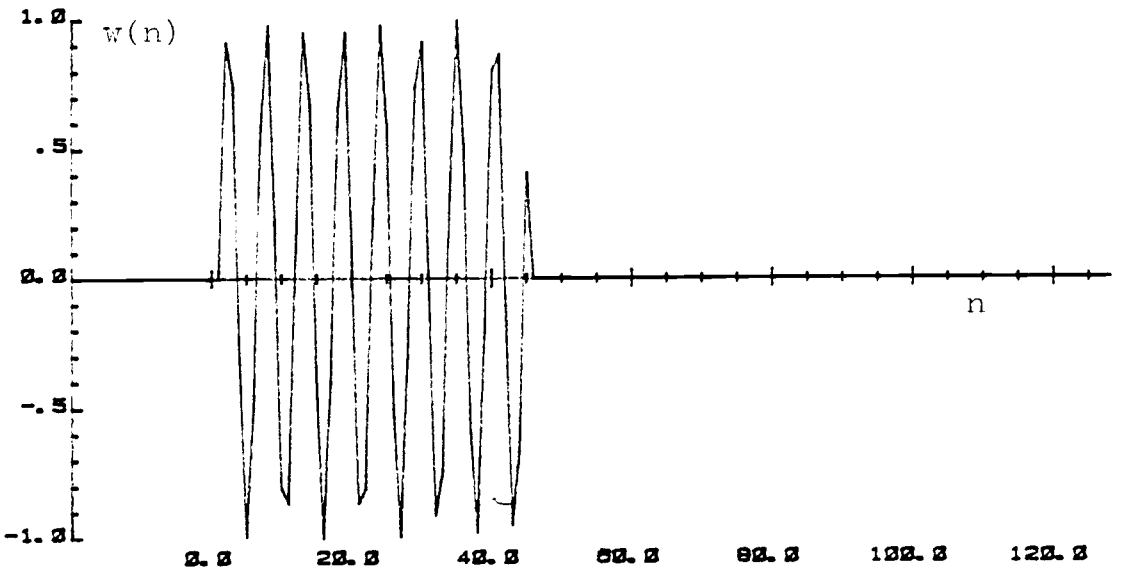


Figure 3.27 Sinusoid Wavelet, Frequency = 4.4 H_z
Sampling Frequency = 30 H_z
Data Length = 45

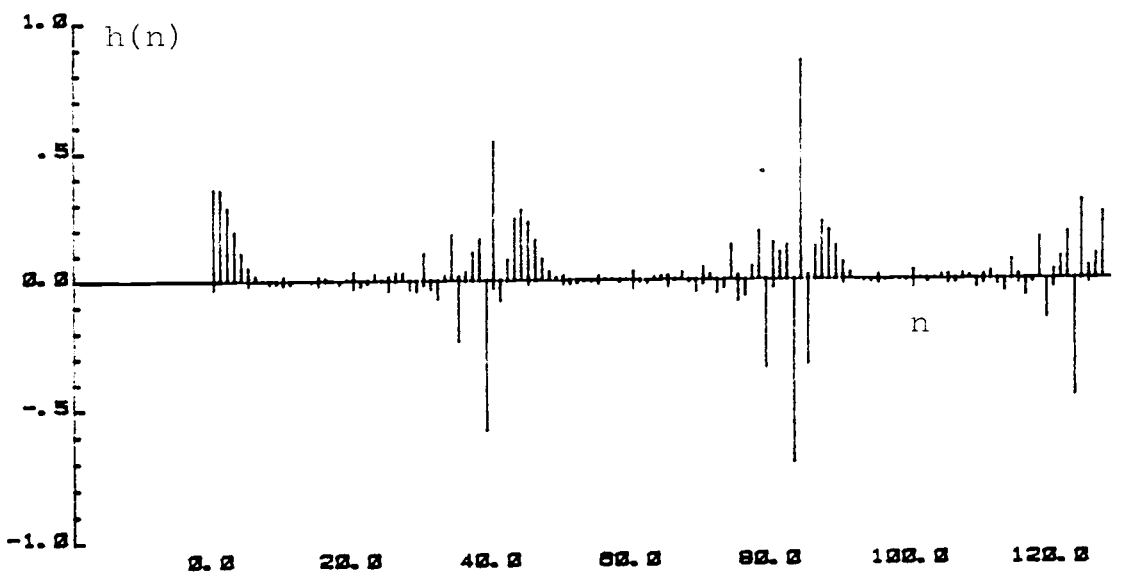


Figure 3.28 Inverse Filter for Spike Output

The impulse response of the inverse filter can be determined in the frequency domain using the FFT. However, a very important and necessary condition was that the linear convolution operator " * " had to be replaced by the circular convolution operator " ⊛ ". Only then does the FFT relationship of equations (1.7) and (1.8) apply.

It is always possible to find a finite length inverse filter that is implemented with circular convolution. Sometimes the case occurs when the FFT of the wavelet is exactly zero for some discrete point or points. In practice this is extremely rare when dealing with real data. However, should it occur, it can be remedied by adding a very small amount of noise to the system. This is known as dithering. It will shift the zeros incrementally and displace any from the unit circle. Therefore, stable inverse filters that are to be implemented with circular convolution can always be found. Such filters will be termed circular inverse filters.

The major disadvantage of circular filters is in the implementation. As previously mentioned, the filter length may not be extended by adding zeros or shortened by truncation. They can only be applied (circularly convolved) with sequences that are no longer than the filter length. However, the filter length may be

made arbitarily long to deal with long sequences. On many machines, FFT routines can compute the DFT of sequences up to a length of 2^{15} data points. Thus, sequences of up to 32k data points can be circular convolved.

The major advantage of using circular inverse filtering is that the filtering error goes to zero and the accuracy obtained is to the maximum precision of the machine. The FFT relationships of Equations (1.7), (1.8) and (1.9) are exact and true, if and only if the circular convolution operators are applied.

Circular inverse filters are the aliased version of the linear inverse filters. Deconvolution of short finite length sequences should be done with circular filters for maximum accuracy. A second advantage of circular filters is that no output delay is required in implementing circular convolution. An example of circular convolution is shown in section 4.7.

3.10 Summary

Frequency domain design of inverse filters using the FFT results in an aliased version of the true inverse filter. The degree of aliasing depends on the zero locations of the wavelet sequence. By making the FFT sequence sufficiently large, aliasing can usually be minimzed and a truncated approximation of the inverse

filter can be obtained. Output delay can be determined by searching the filter for the region of minimum magnitude and applying appropriate shifting. For filters that are strongly aliased, inverse filtering can be accomplished by circular convolution.

IV. DESIGN AND IMPLMENTATION OF THE DECONVOLUTION FILTER.

4.1 Introduction

The input-output relationship for a linear time-invariant system for deconvolution has been presented in chapter I. There exist a number of possible choices for the reflector sequence of which the spike and the Gaussian reflector waveforms are of primary interest. The previous chapters have dealt with the mathematical aspects and implications of designing deconvolution filters in the frequency domain via the FFT. This chapter presents the algorithm for finding the impulse response of the filters. Several typical basic wavelets are analyzed and inverse filters for both Gaussian and spike reflector series are shown. Examples of the implementation of the inverse filters are presented.

4.2 Gaussian Reflector Series

It is appropriate at this point to define two kinds of filters. Those that will deconvolve wavelets into spikes will be termed spike inverse filters and those that will deconvolve them into Gaussian waveforms will be termed Gaussian inverse filters. The unique time and frequency domain relationship of the Gaussian waveform gives it promising potential as a good reflector waveform. The nature and mathematical expressions for this waveform

have been thoroughly discussed in chapter II. As seen in the magnitude spectrum of the FFT in figure 2.1, the Gaussian waveform (for σ_t approximately greater than 2.5) has many zeros that are very close to the unit circle on the left side of the z-plane. The exact values of the small numbers of the magnitude is shown in appendix A and B. The zeros close to the unit circle are very useful as they may potentially be used to reduce the length of the inverse filter. Recall that for a spike reflector wavelet, the z-transform of the inverse filter is H(z) where

$$H(z) = \frac{1}{W(z)} \qquad (1.3)$$

and W(z) = z-transform of basic wavelet.

For a Gaussian inverse filter,

$$H(z) = \frac{G(z)}{W(z)} \qquad (4.1)$$

where G(z) = z-transform of Gaussian waveform.

In designing a spike inverse filter, the filter length may become very long if the poles of H(z) are very close to the unit circle. However, in Gaussian inverse filtering, the filter length may be greatly reduced by specifying the zeros of the Gaussian FFT to lie close to those poles of H(z) that are adjacent to the unit circle. Pole/zero cancellation rarely take place in the discrete domain, but the effect of the poles may be

greatly reduced by placing zeros in the same vicinity. It will be shown in the examples of this chapter how Gaussian inverse filters may sometimes result in filter lengths that are several magnitudes shorter than spike inverse filters. The shortening of the filter length depends on the location of the poles of equation (1.3). If the poles close to the unit circle are distributed in the left side of the z-plane, then the Gaussian inverse filter may be effective in reducing the filter length because a large number of zeros of the Gaussian FFT lie extremely close to the unit circle on the left side of the z-plane.

However, if the poles close to the unit-circle exist in any other constrained sector of the z-plane, it is still possible to reduce the filter length by circularly shifting the Gaussian FFT. Circular shifting the spectrum does not affect the magnitude of the time sequence, and hence the time of occurrence and magnitude of the reflector series is preserved. The FFT spectrum can be circular shifted to place the zeros across the region of poles that are close to the unit circle. This however may result in filters that have complex coefficients which is a trade-off against filters that are extremely long. A necessary condition for this method to be useful is for the poles that are close to the unit circle to be constrained to a region no greater than about half of the circle. An example of

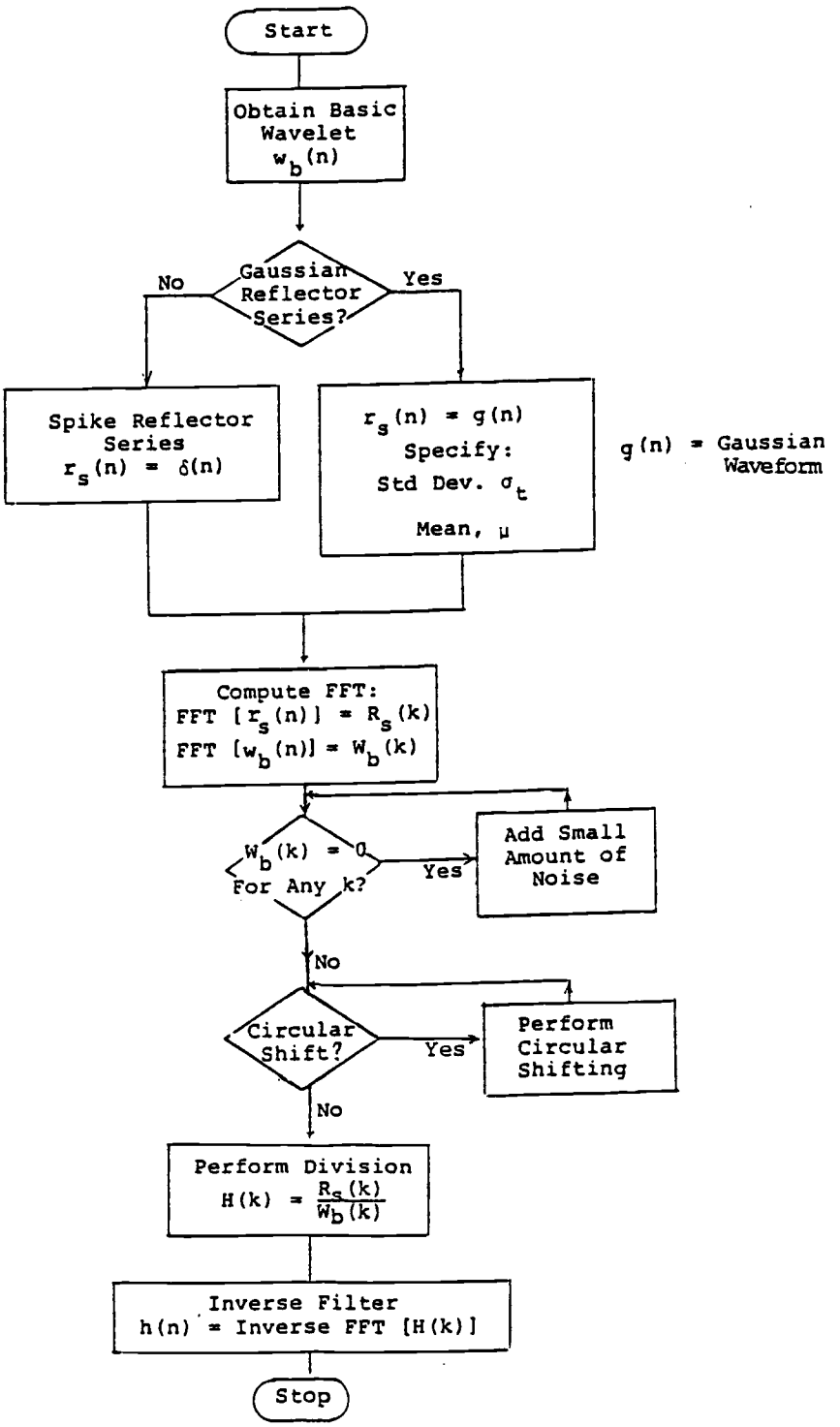
circular shifting the Gaussian FFT is shown in section 4.5.

4.3 Flowchart

It is assumed that the basic wavelet $w_b(n)$ is known and the task is to deconvolve a composite signal comprised of two or more identical wavelets differing only in magnitude and time of arrival. Figure 4.1 shows the flowchart for finding the impulse response of the inverse filter. Appendix D is a listing of the program used. In obtaining the Gaussian reflector series, g(n), two variables, σ_t (standard deviation) and μ (mean) need to be specified. The choice of σ_t is dependent on the required degree of resolution of the reflector series. For σ_t less than 2.0, it is of no real advantage to use the Gaussian reflector series because its frequency spectrum begins to alias and looses its characteristic of having zeros close to the unit circle. To reduce high frequency transients due to truncation, the mean should be chosen to be at the center of the sequence. However, for most practical considerations, it suffices to specify the mean to be about five standard deviations from the extremities of the sequence.

In finding the frequency spectrum by evaluating at discrete points on the z-plane (using the FFT), it may occur (although rarely) that the point of evaluation falls

Figure 4.1 Algorithm for Finding Inverse Filter



extremely close to a zero. The reciprocal of the magnitude will tend towards infinity and a solution cannot be found for the inverse filter. It has been found that in practice this is a trivial problem as a small amount of noise can be added to the system to shift the zero incrementally away from the unit circle. The frequency response of the filter is obtained by dividing the frequency response of the reflector series by the frequency response of the basic wavelet.

In spike inverse filtering, the procedure is to first compute the inverse filter for a zero delay reflector series and then determine from the results how much delay is required. In Gaussian inverse filtering the mean, μ is the delay of the reflector series. The best procedure would be to first specify the mean at the center of the sequence and later make the necessary adjustment to the mean value.

4.4 Examples of Inverse Filters

The theory of frequency domain deconvolution has been presented in the preceding chapters and examples involving very short sequences were used in illustration. In this section, the flowchart algorithm shown in figure 4.1 will be used to find inverse filters of common occurring wavelets. Both spike inverse filters and

Gaussian inverse filters will be found for each of the following sequences, and the relative advantages of each of the two kinds of filters will be mentioned.

Figures 4.2 to 4.13 show examples of spike inverse filters and figures 4.14 to 4.25 show examples of Gaussian inverse filters. In each example, 45 samples of the basic wavelet were taken and the inverse filters were designed with FFT array lengths of 128. The figures are self-explanatory and the length of the required filters can be estimated from observing the results.

Example 1: Pulse Sequence

The basic wavelet (pulse) is

$$w_b(n) = \exp(-a \cdot n) - \exp(-b \cdot n)$$

where $a = 0.2$ and $b = 0.3$

$n = 0, 1, 2 \ldots 45$

Example 2: Damped Cosine Sequence

The basic wavelet:

$$w_b(n) = \cos[2\pi F(n-22)T] \exp[-a(n - 22)T]$$

where $F = 5.5$ Hz

$T = 1/30$

$a = 6.0$

$n = 0, 1, 2 \ldots 45$

Example 3: "Chirp" Sequence

The basic wavelet:

$$w_b(n) = \cos [2\pi(an)n]$$

where $a = 0.002$

$n = 0, 1, 2, \ldots 45$

4.4.1 Spike Inverse Filters

The results of the three examples show that the spike inverse filter for the pulse sequence of figure 4.2 has dominant values for only three coefficients. Hence a filter length of three with an optimum output delay of one is sufficient to implement the filter to a fairly high degree of accuracy.

To a good approximation the damped cosine sequence of figure 4.5 requires a filter length of about 10 (dominant values) and the output spike for this filter length has to be delayed by about 26 samples. A greater degree of accuracy can be achieved in filtering by using a filter long enough to include the small values located somewhat away from the main body of filter coefficients. This would require a filter length of about 52 and optimum delay of about 47.

The "chirp" sequence of figure 4.8 has an inverse filter that is grossly aliased. Clearly 128 data points is insufficient to prevent aliasing and the sequence was extended to 512 data points. Figure 4.13 shows a reduction

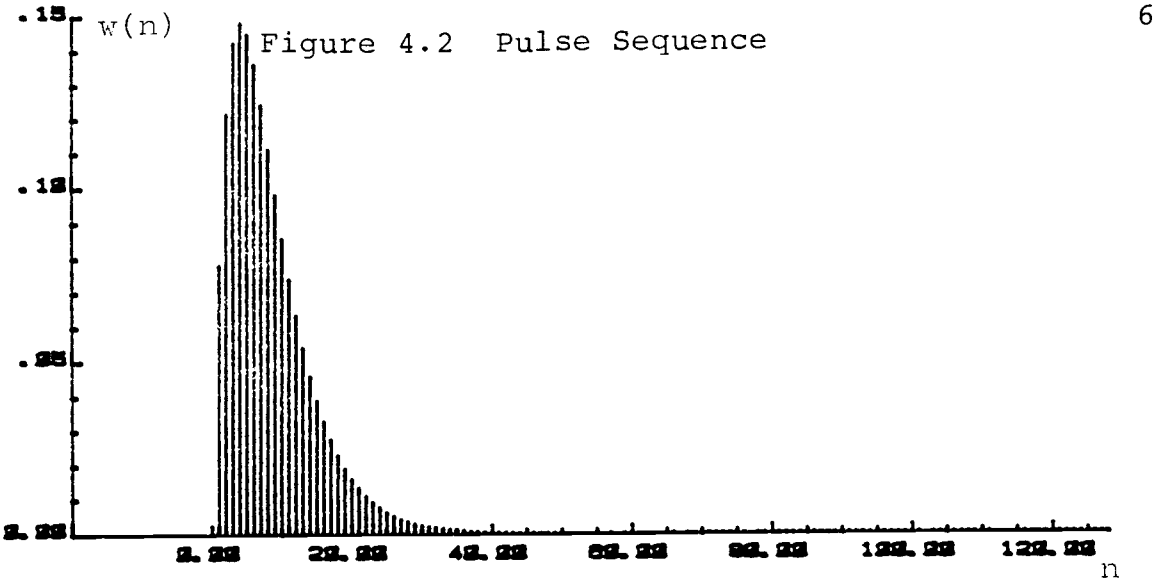


Figure 4.2 Pulse Sequence

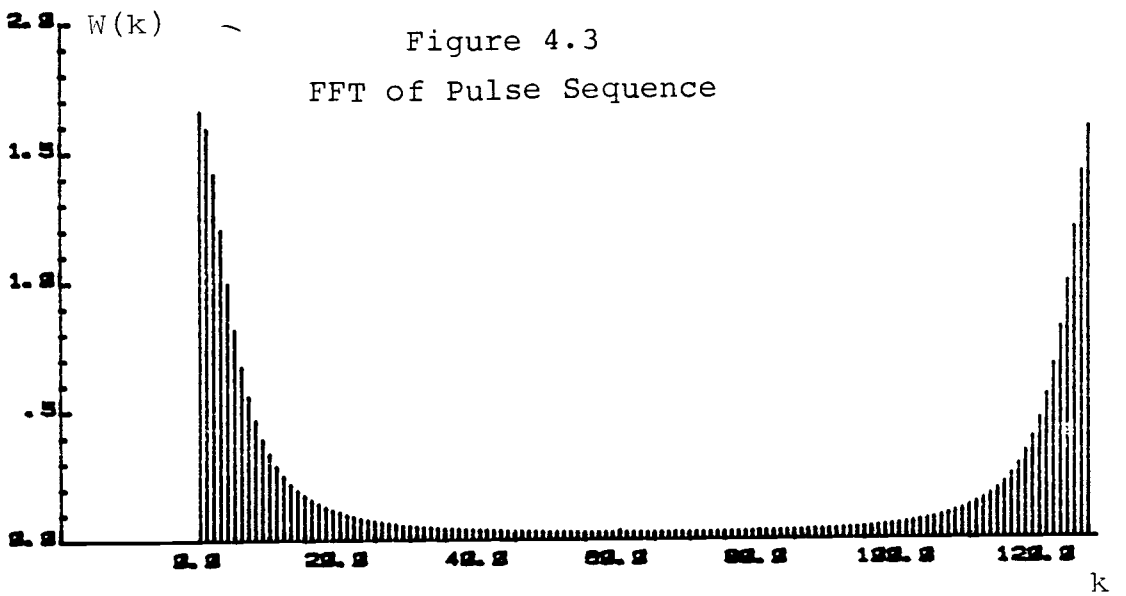


Figure 4.3
FFT of Pulse Sequence

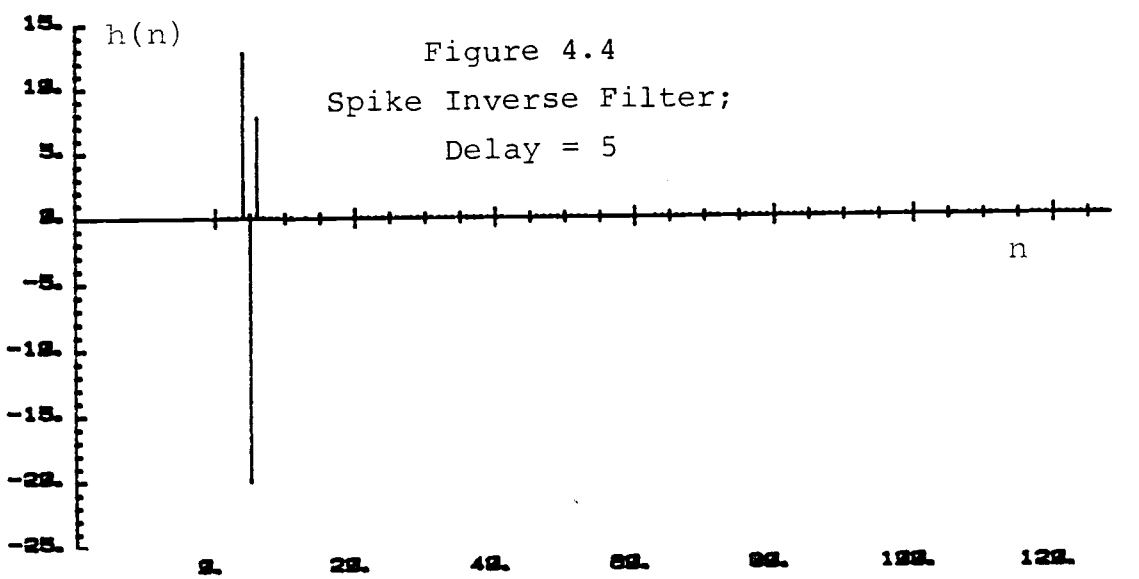


Figure 4.4
Spike Inverse Filter;
Delay = 5

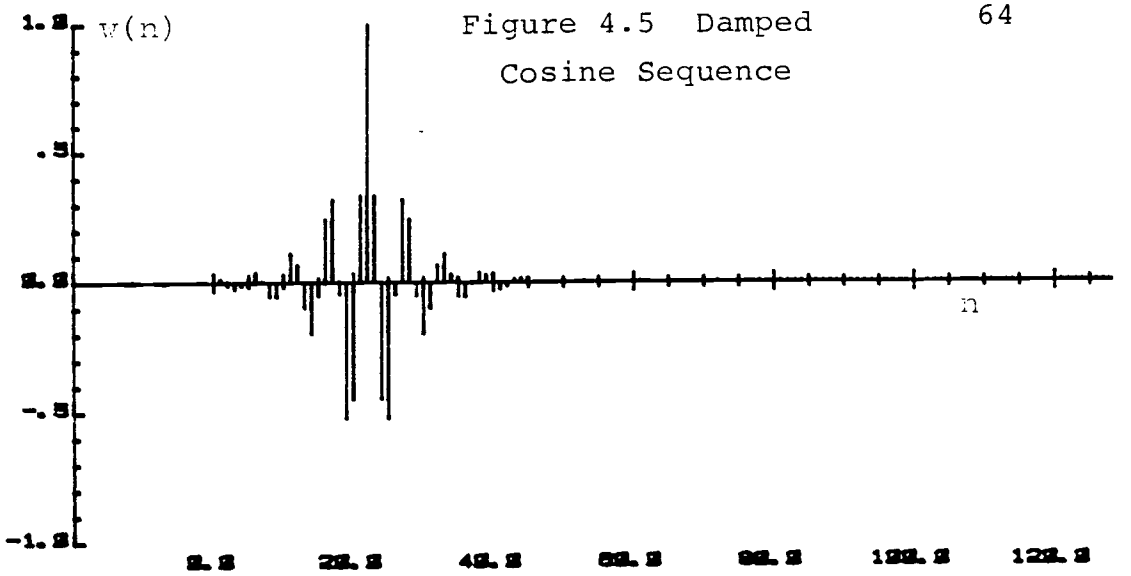


Figure 4.5 Damped Cosine Sequence

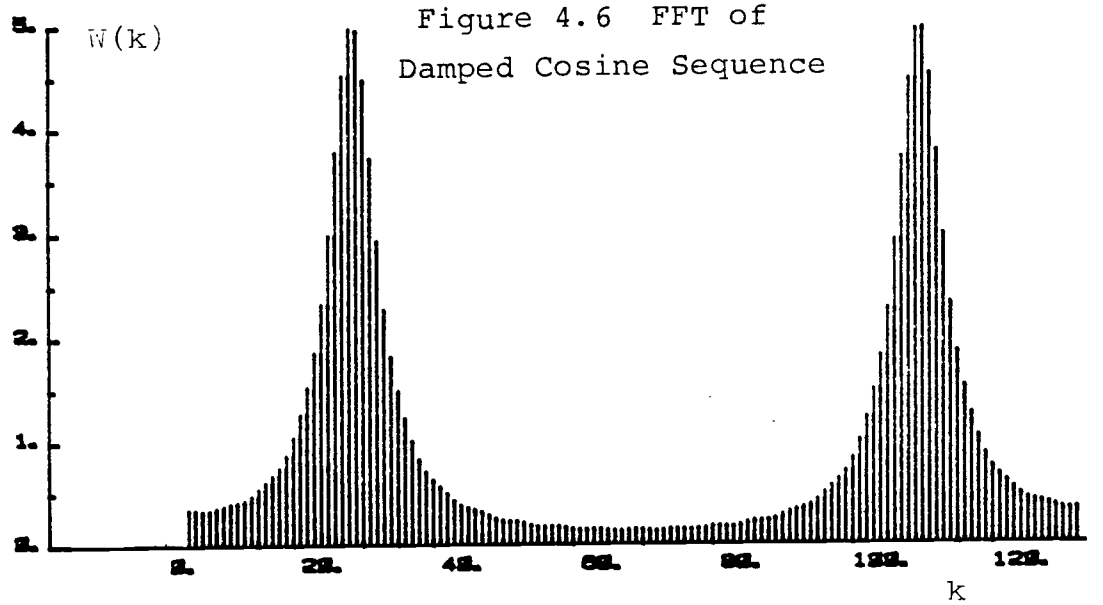


Figure 4.6 FFT of Damped Cosine Sequence

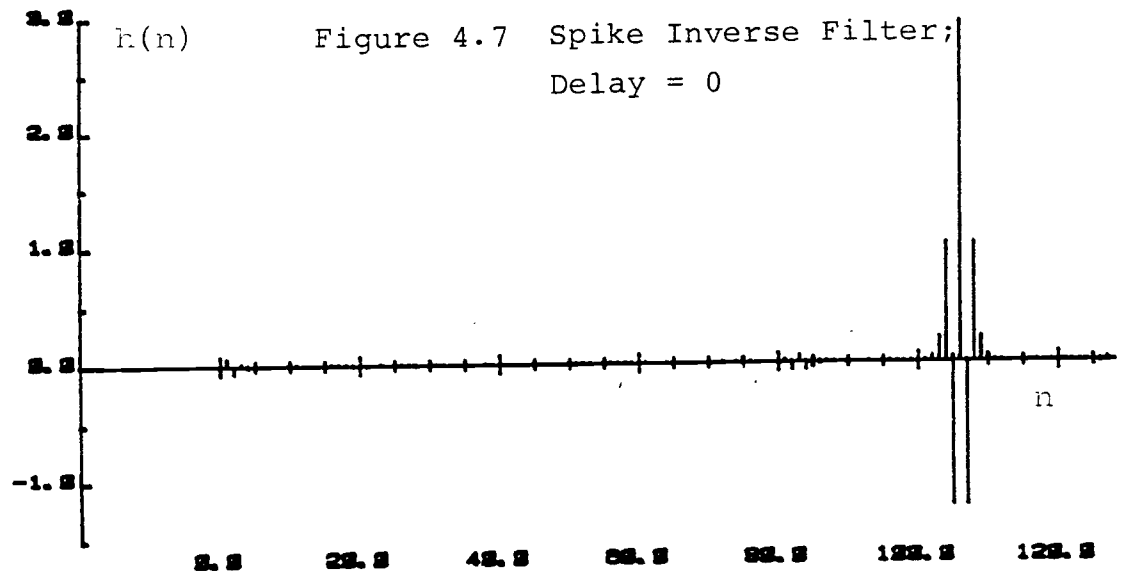


Figure 4.7 Spike Inverse Filter; Delay = 0

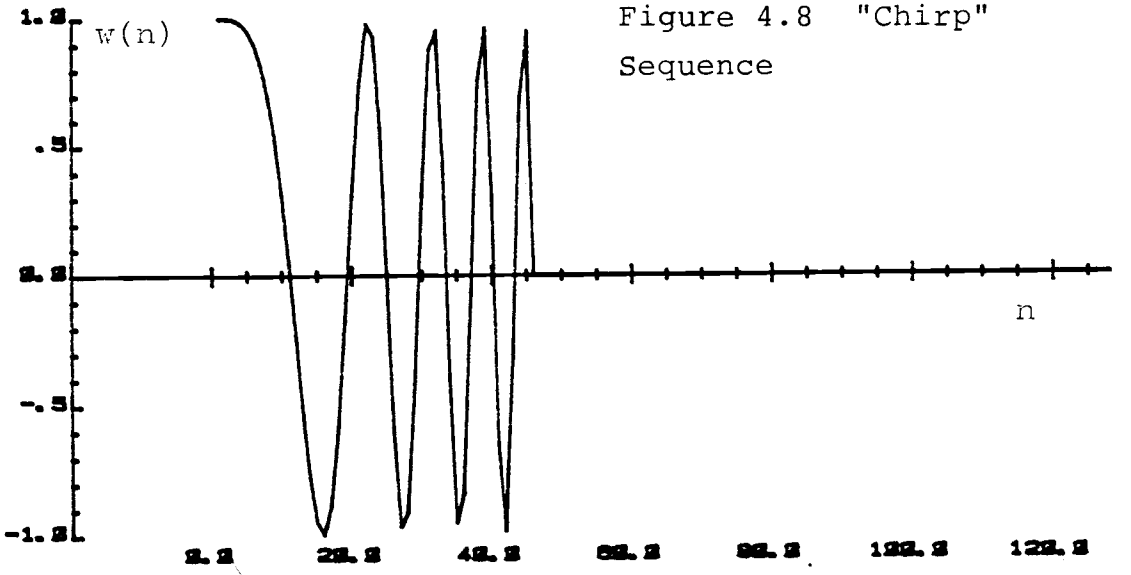


Figure 4.8 "Chirp" Sequence

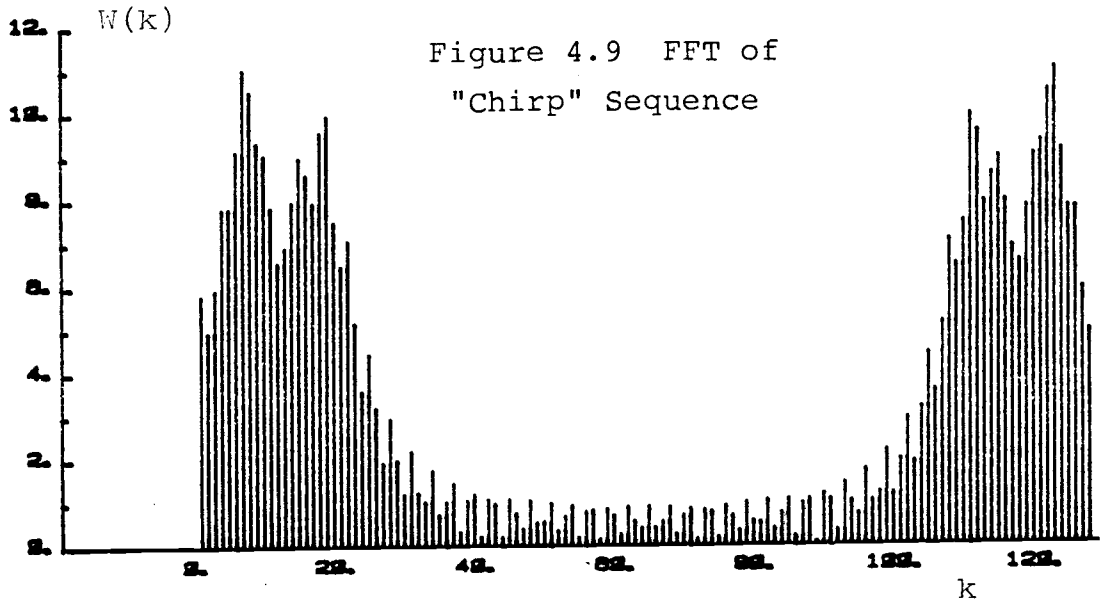


Figure 4.9 FFT of "Chirp" Sequence

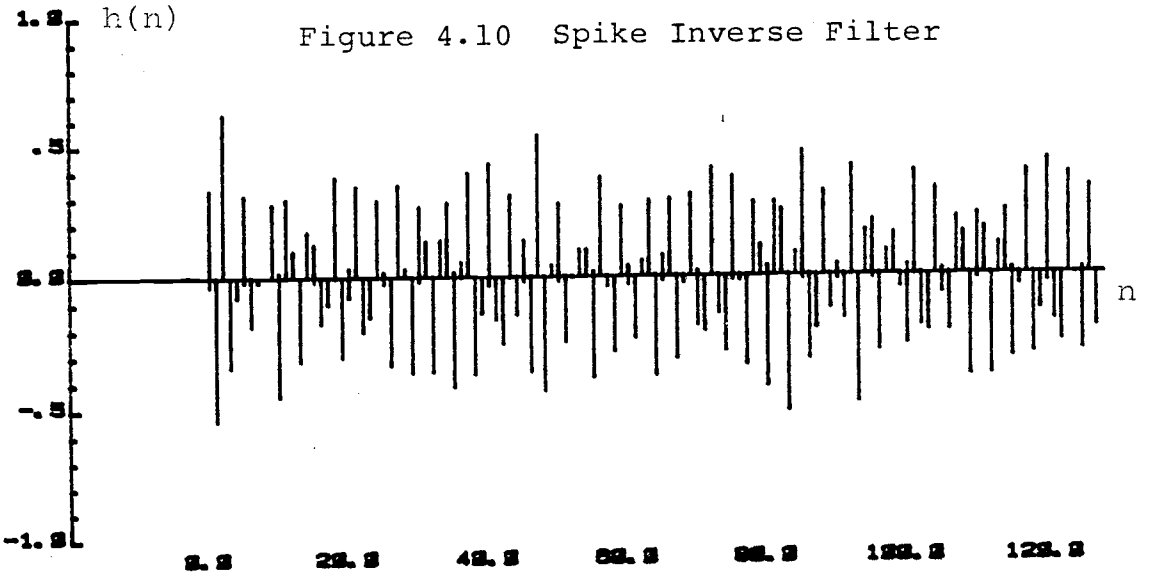


Figure 4.10 Spike Inverse Filter

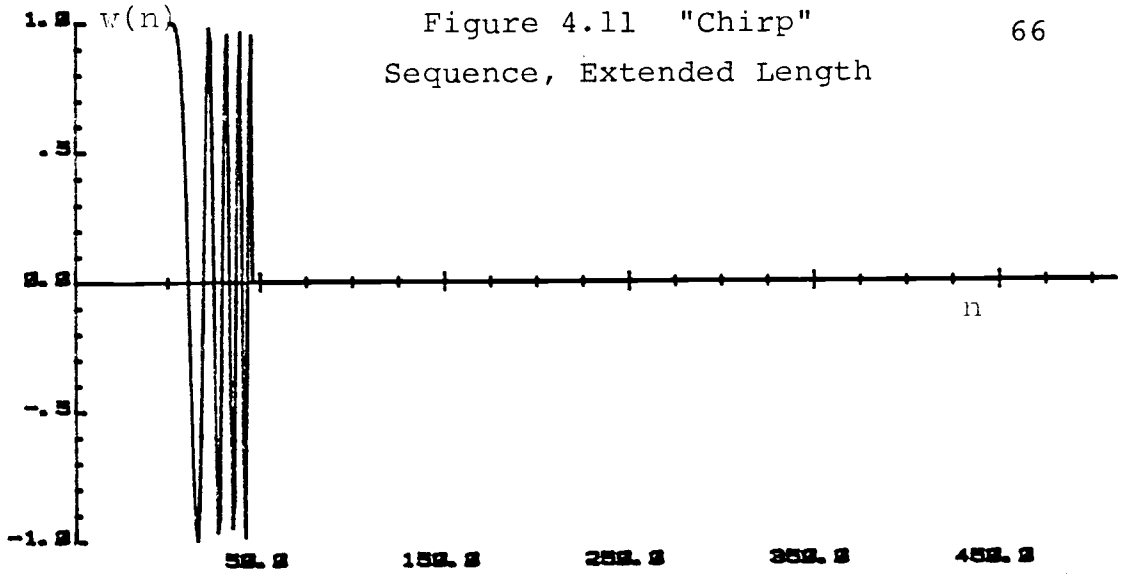


Figure 4.11 "Chirp" Sequence, Extended Length

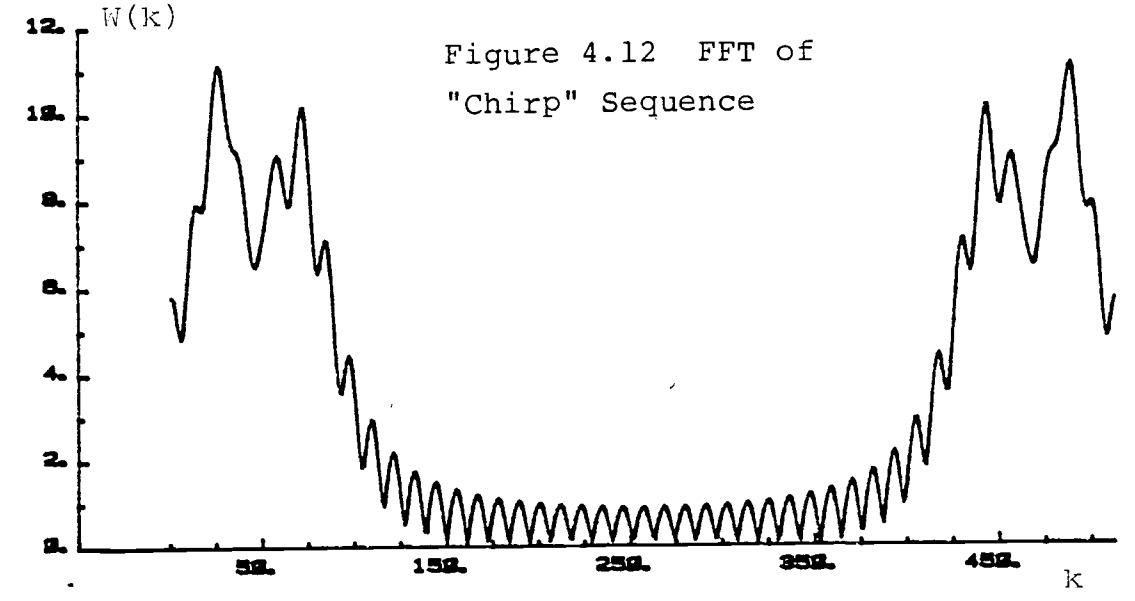


Figure 4.12 FFT of "Chirp" Sequence

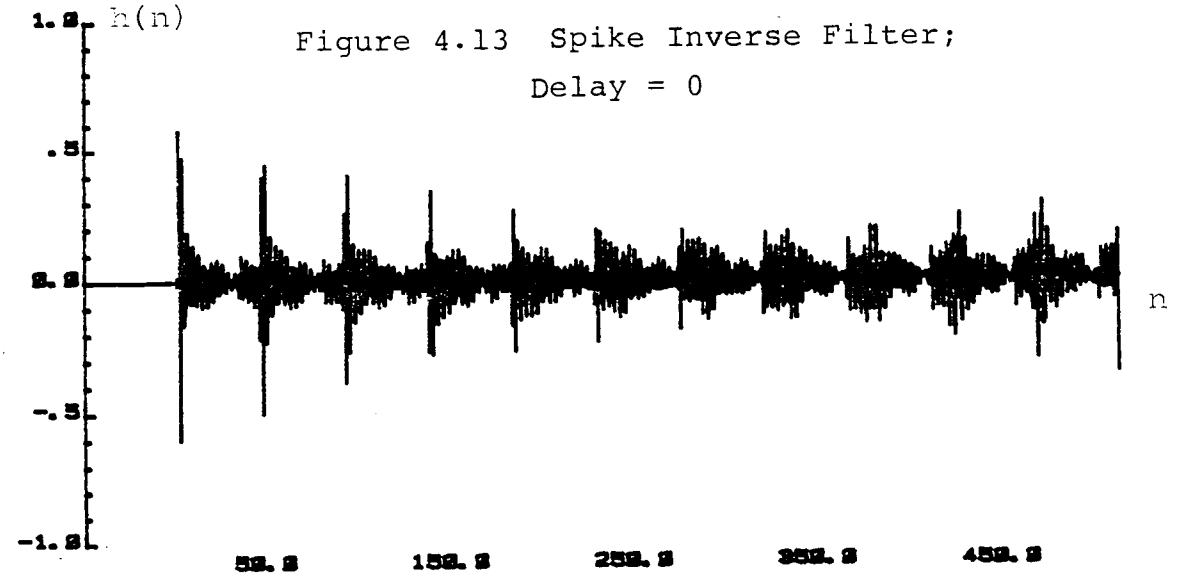


Figure 4.13 Spike Inverse Filter; Delay = 0

in aliasing but 512 data points was still insufficiently long to prevent aliasing.

It can be seen from figures 4.9 and 4.12 that the aliasing of the "chirp" spike inverse filter is due to the many zeros of the wavelet occurring near the unit circle.

4.4.2 Gaussian Inverse Filters

All of the Gaussian inverse filters were designed for a reflector wavelet with standard deviation σ_t of 2.5 and mean $\mu = 64$. The most interesting result is shown in figure 4.21 and figure 4.24. From figure 4.24 it can be estimated that the Gaussian inverse filter for the "chirp" sequence would require a filter length of about 175 coefficients and an optimum output delay of about 90 samples. This filter is far shorter than the corresponding spike filter of figure 4.13 because the cluster of zeros of the Gaussian wavelet are very close to the unit circle and they dominate over the effect of the poles of the spike filter that are adjacent to the unit circle. It should be noted in the FFT of the Gaussian inverse filters that a large region in the center is approximately zero. These values when approximated to zero will save computation time when implementing convolution via the FFT.

Figure 4.26 is another example of a filter length that

is greatly reduced by the use of the Gaussian reflector series. In chapter III, figure 3.26 shows that the spike inverse filter is infinitely long because of a pair of poles on the unit circle. The Gaussian inverse filter of the same wavelet (figure 4.26) is only about 25 coefficients long.

4.5 Circular Shift of Gaussian FFT

In the later part of section 4.2, it was explained that in designing the inverse filter, the FFT of the Gaussian sequence may be circularly shifted without affecting the magnitude of the time response at the output of the filter. If only the magnitude is of importance, then for a given basic wavelet, an infinite number of filters (of the same filter length and delay) can be found. Most of the filter coefficients, however, will be complex. This is possible in frequency domain design because phase information is included in the design algorithm. It is in stark contrast to inverse Weiner filtering because phase information is generally discarded in time domain design methods.

Of primary interest is to consider how the FFT of the Gaussian distribution can be circularly shifted to reduce the length of a filter. It has been shown (by comparing figure 3.26 and figure 4.26) that when the

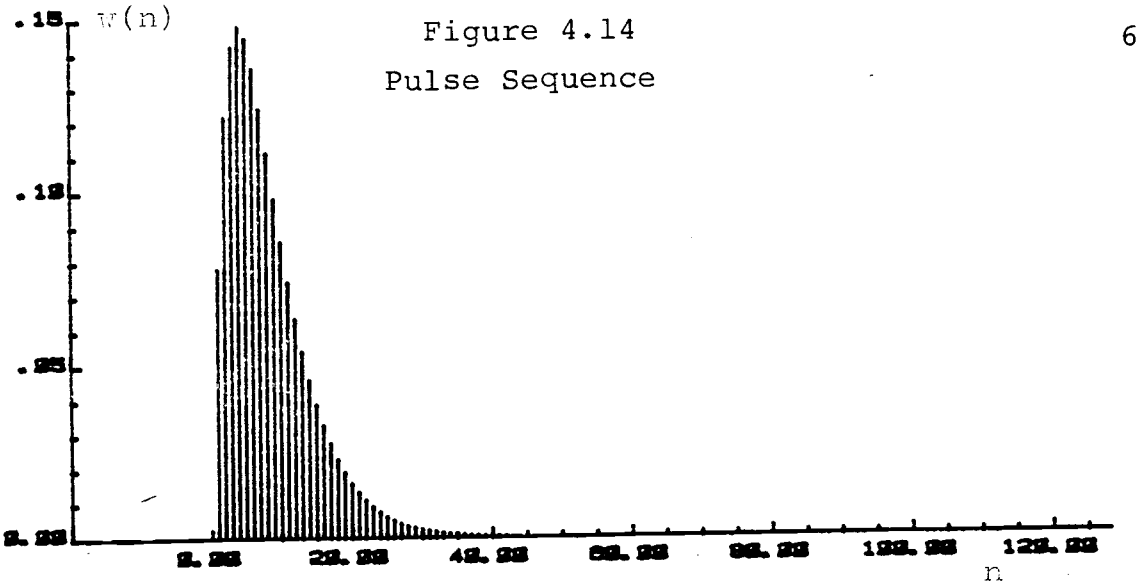


Figure 4.14
Pulse Sequence

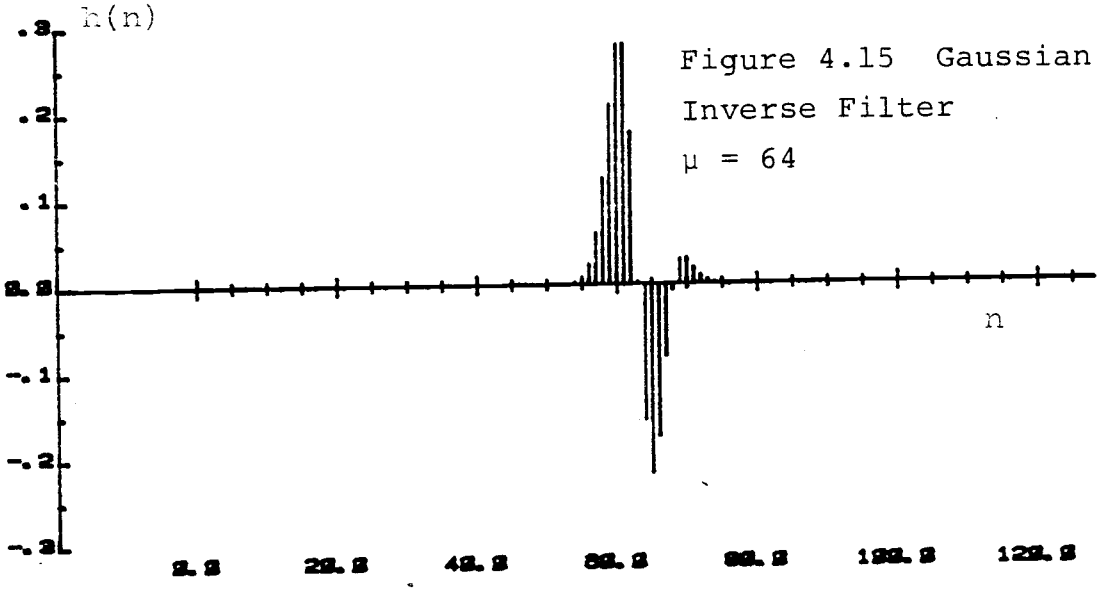


Figure 4.15 Gaussian
Inverse Filter
$\mu = 64$

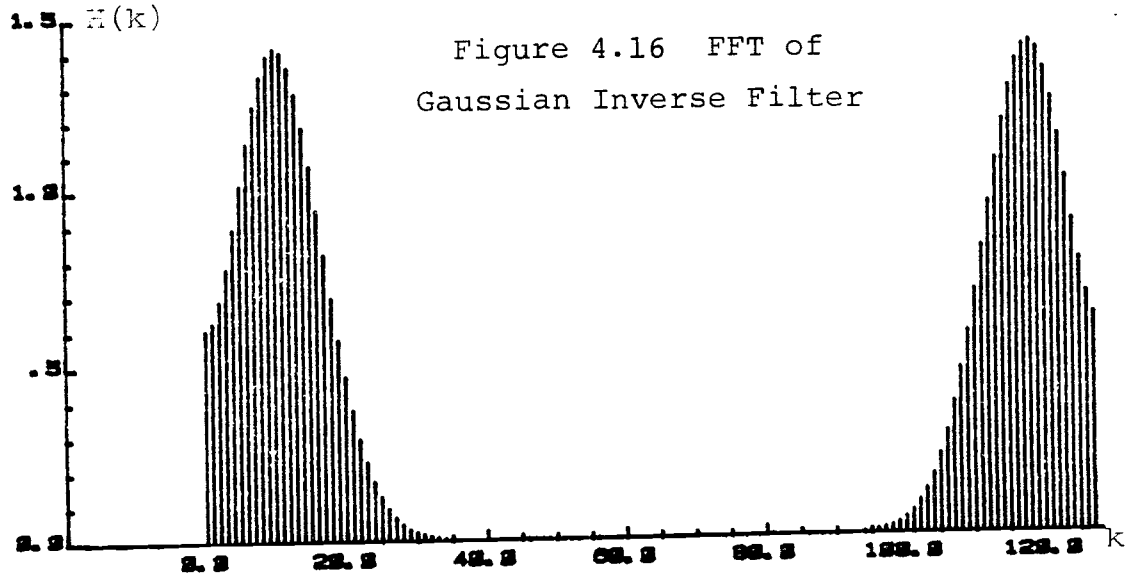


Figure 4.16 FFT of
Gaussian Inverse Filter

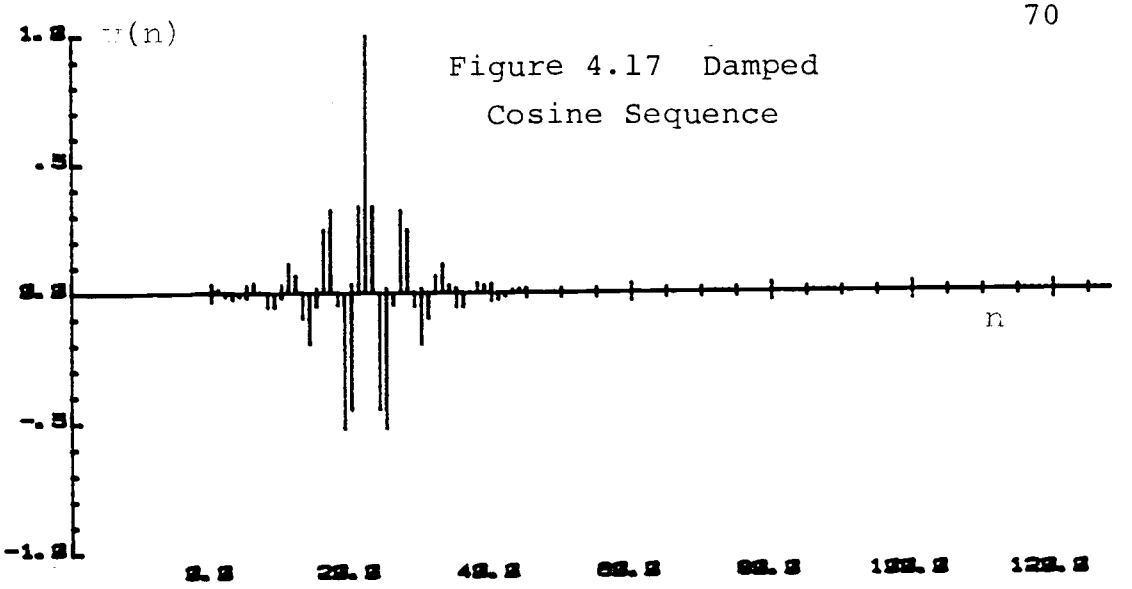


Figure 4.17 Damped Cosine Sequence

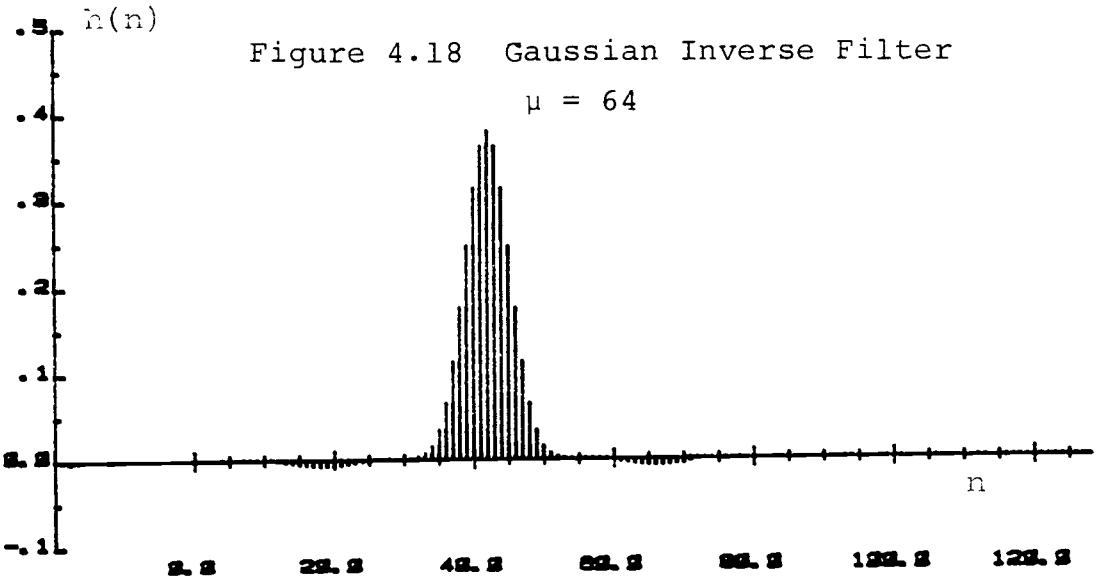


Figure 4.18 Gaussian Inverse Filter
$\mu = 64$

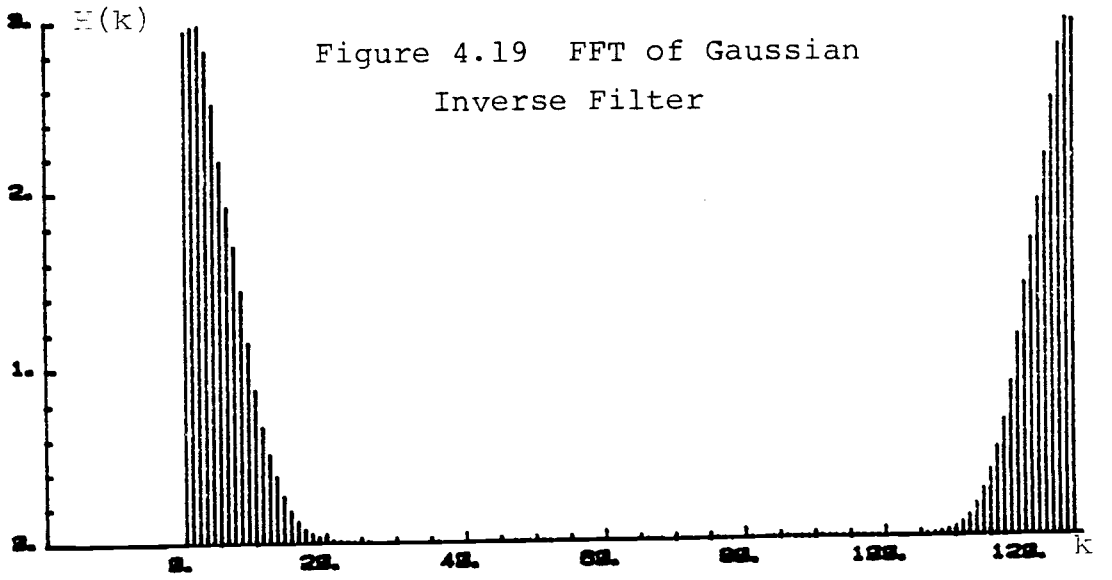


Figure 4.19 FFT of Gaussian Inverse Filter

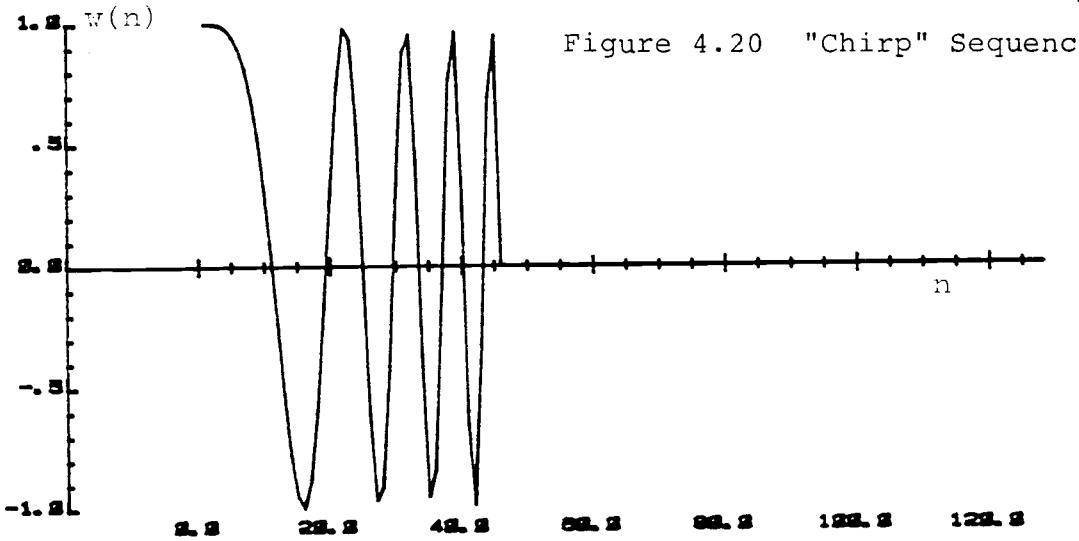


Figure 4.20 "Chirp" Sequence

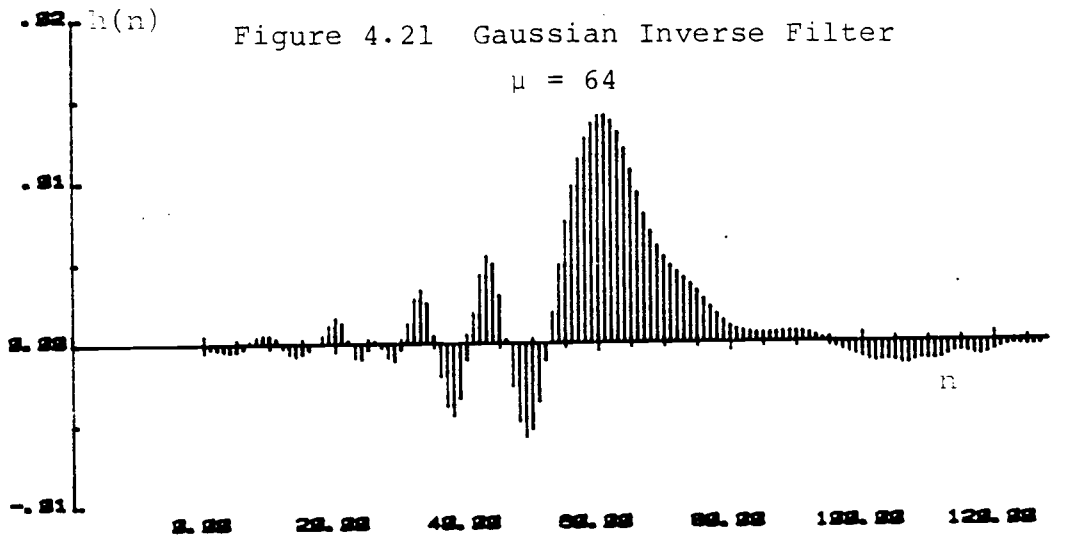


Figure 4.21 Gaussian Inverse Filter
$\mu = 64$

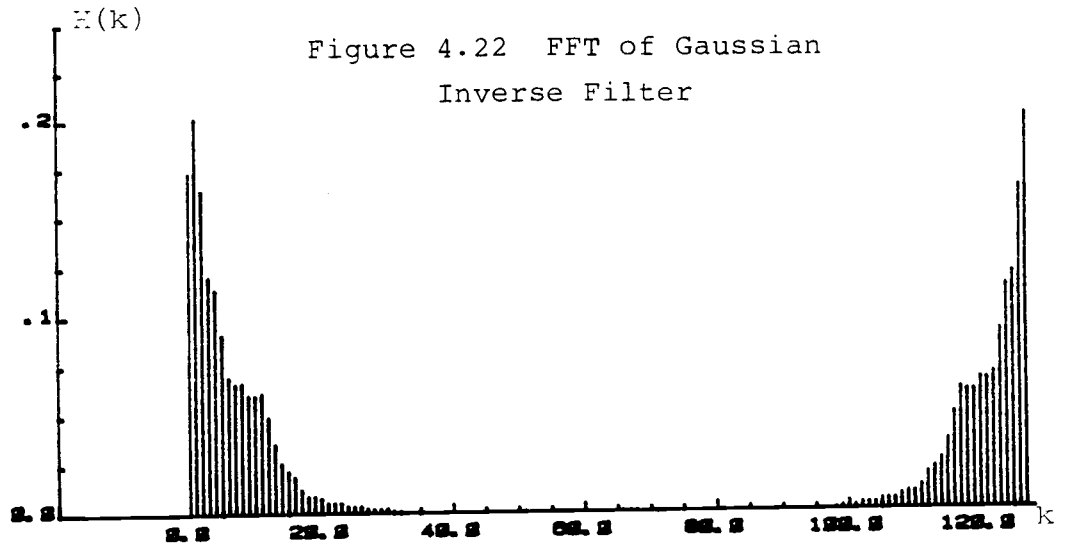


Figure 4.22 FFT of Gaussian
Inverse Filter

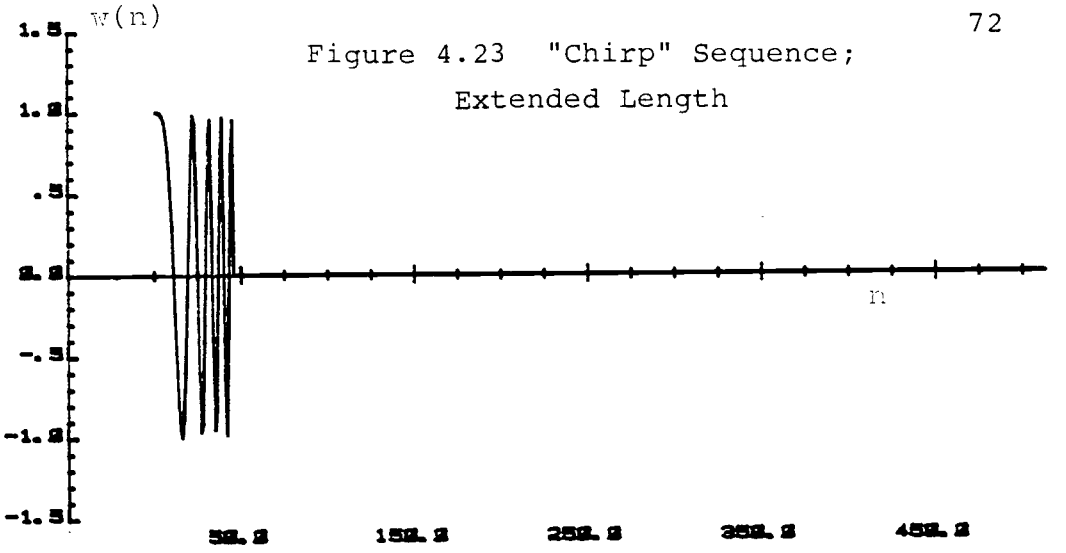


Figure 4.23 "Chirp" Sequence; Extended Length

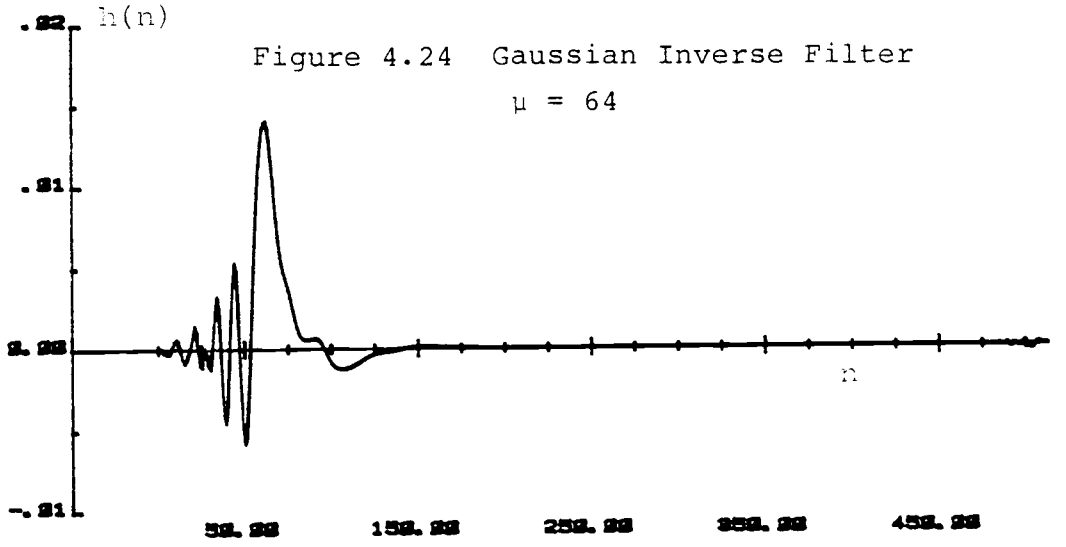


Figure 4.24 Gaussian Inverse Filter
$\mu = 64$

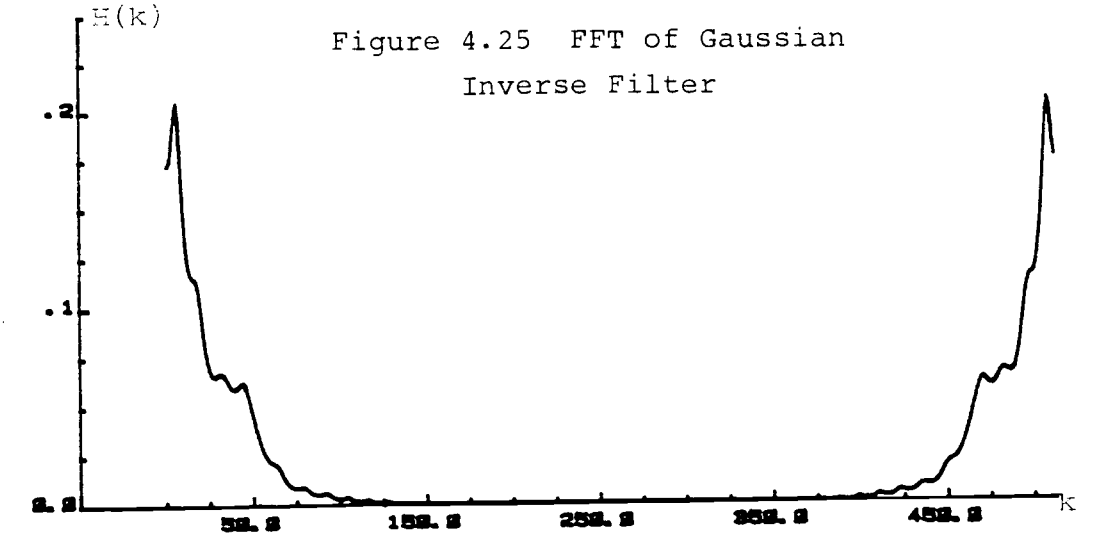


Figure 4.25 FFT of Gaussian Inverse Filter

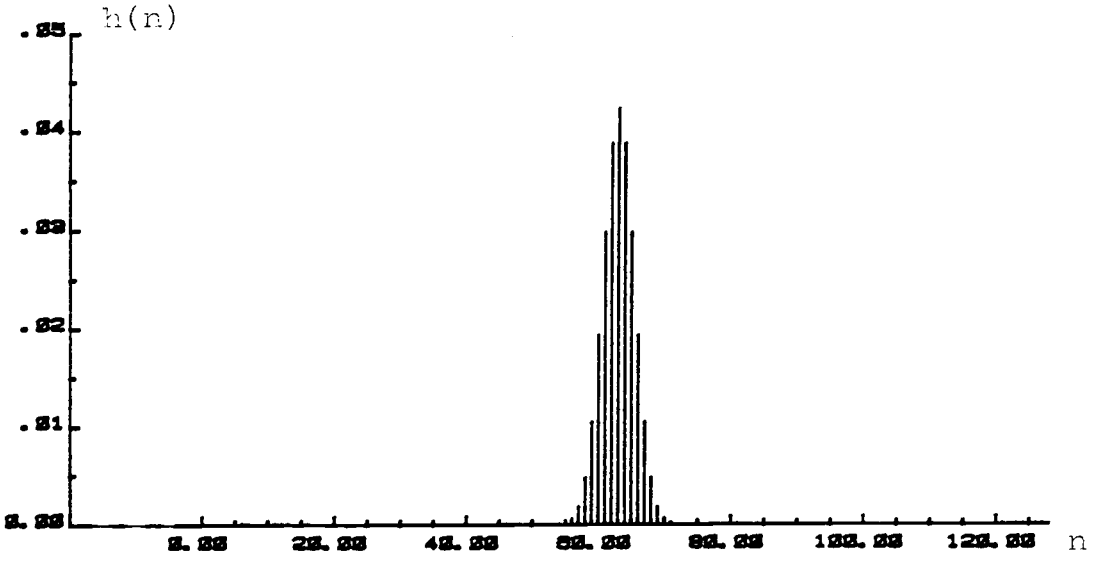


Figure 4.26 Gaussian Inverse Filter for Wavelet of Figure 3.24

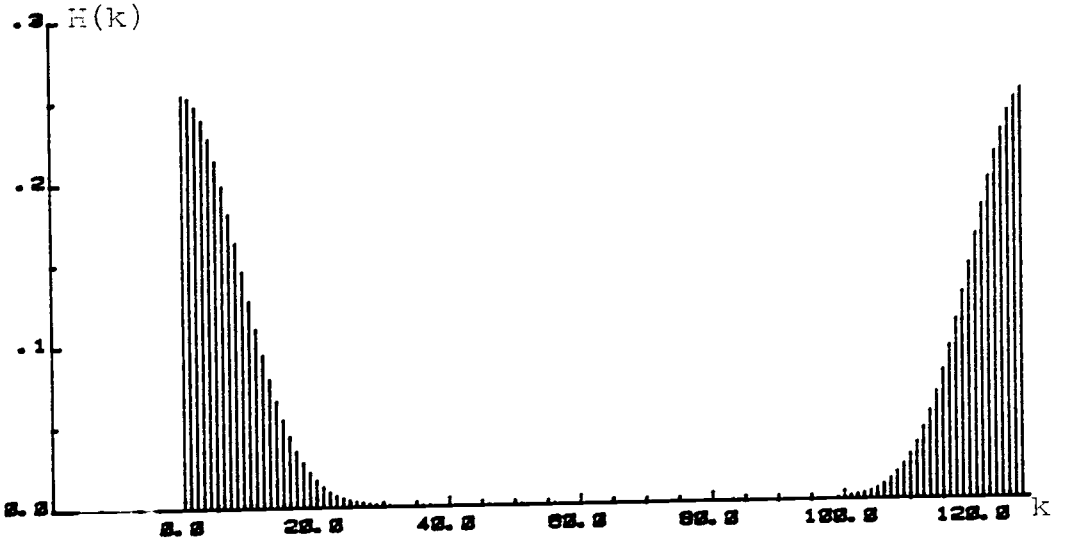


Figure 4.27 FFT of Gaussian Inverse Filter

$\mu = 64$

function $\frac{1}{W(z)}$ has poles that exist close to the unit circle on the left side of the z-plane, the use of the Gaussian reflector wavelet can greatly shorten the required length of the filter. This is possible because the zeros of the Gaussian wavelet dominate the effect of the poles on the left half of the z-plane.

When the poles lie close to the unit circle on the right side of the z-plane, a circular shift is required to place the zeros of the Gaussian waveform across the region of poles. For real input sequences, $\frac{1}{W(z)}$ always has poles occuring in conjugate pairs that are symmetric along the real axis. The Gaussian waveform has zeros that are located symmetrically on the left half of the z-plane.

Therefore, for purposes of reducing pole effects on the right half of the z-plane, the number of circular shifts of the Gaussian FFT is always optimally half the sequence length. This is an extremely important consequence because it implies that the output is phase shifted by a factor of exp $[(\pi)^n]$, which is essentially a negative one factor for odd of values of n. Thus, the output remains as real numbers, which implies that filter coefficients are also real numbers.

The following example and illustration will show the effectiveness of circular shifting. Figure 4.29 shows a sequence with the following expression

$$W(z) = z^{\circ} + az^{-1} + bz^{-2} + cz^{-3} + dz^{-4}$$

where

$$a = -2.638958434$$

$$b = 2.616025404$$

$$c = -1.190069695$$

$$d = 0.25$$

The function $\frac{1}{W(z)}$ has poles at the following locations.

$$z_1 = 0.5\ e^{j45^{\circ}}$$

$$z_2 = 0.5\ e^{-j45^{\circ}}$$

$$z_3 = 1.0\ e^{j15^{\circ}}$$

$$z_4 = 1.0\ 3^{-j15^{\circ}}$$

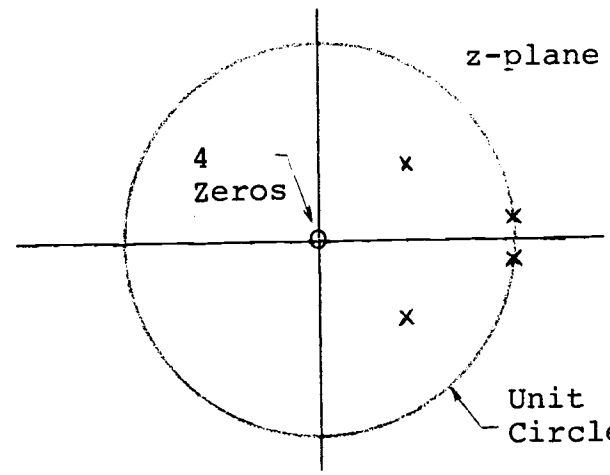


Figure 4.28

Pole/Zero Location

Figure 4.31 shows the corresponding spike inverse filter, and figure 4.32 shows the Gaussian inverse filter. Note that both filters are infinitely long. The Gaussian inverse filter remains infinitely long because the zeros of the Gaussian waveform are on the left region of the unit circle on the z-plane and do not dominate over poles on the right half.

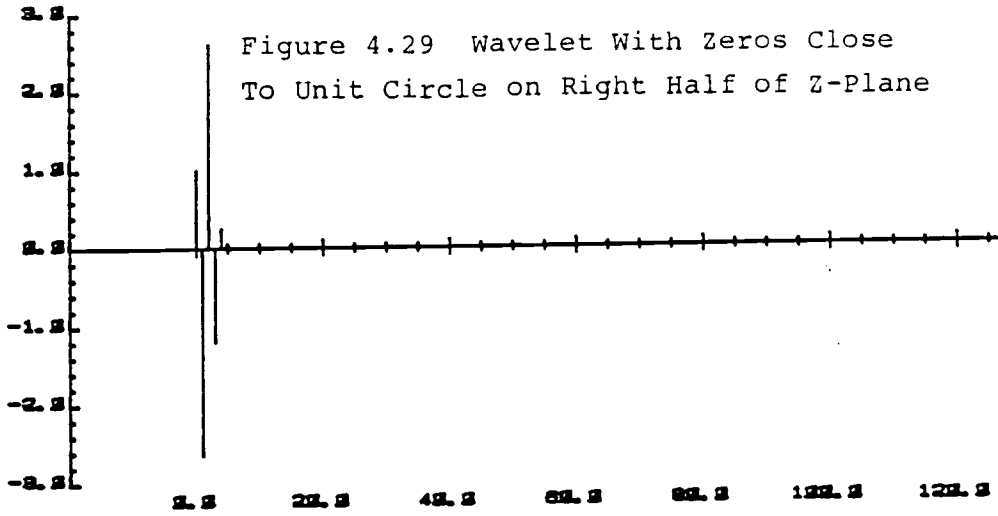


Figure 4.29 Wavelet With Zeros Close To Unit Circle on Right Half of Z-Plane

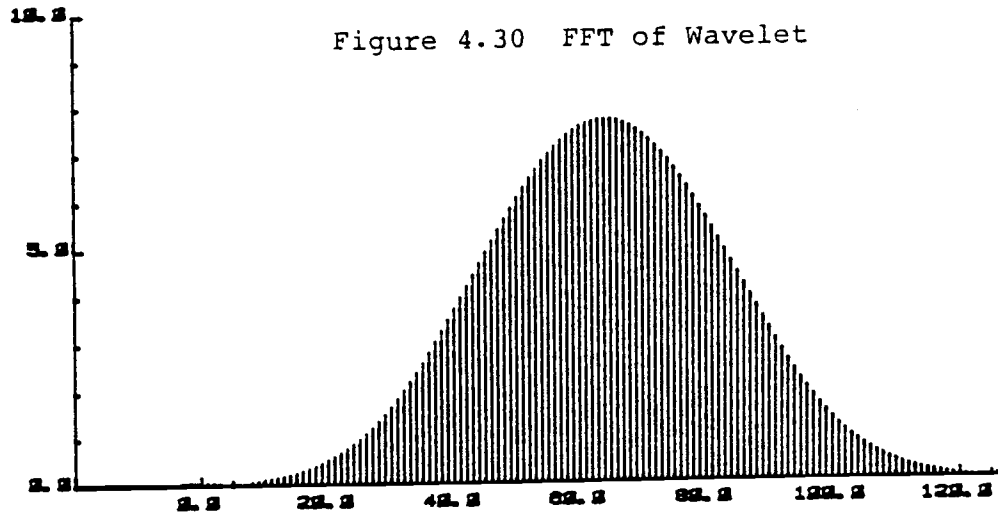


Figure 4.30 FFT of Wavelet

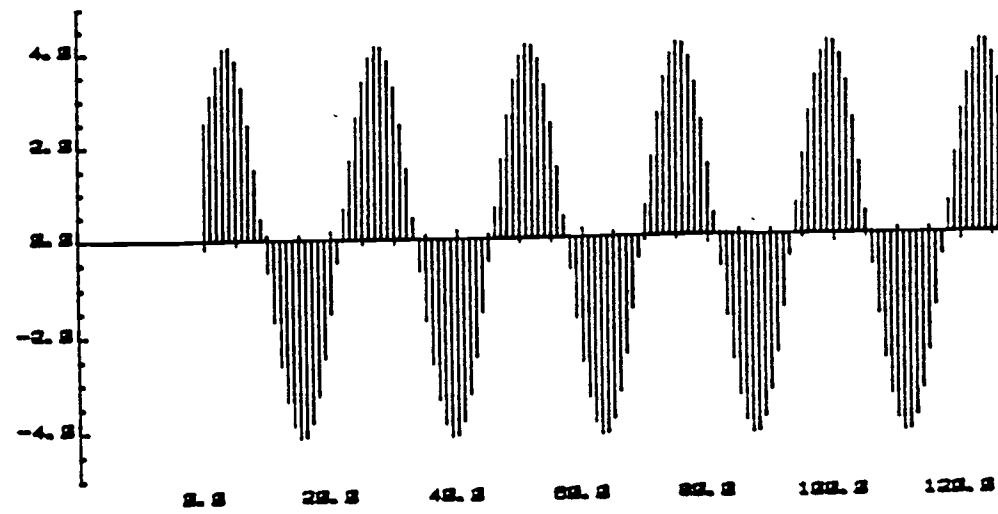


Figure 4.31 Spike Inverse Filter

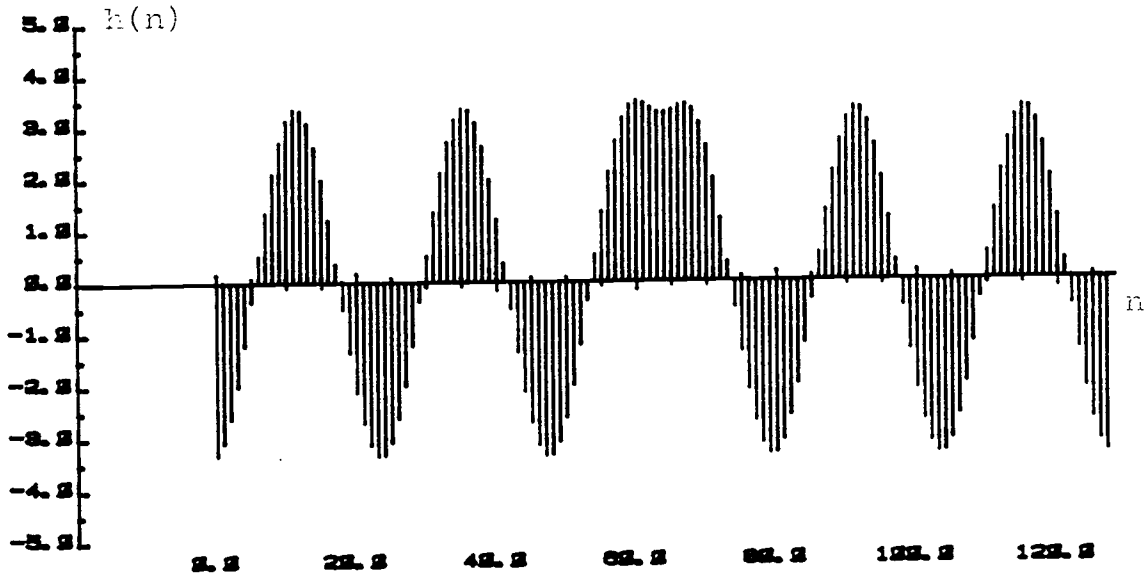


Figure 4.32 Gaussian Inverse Filter

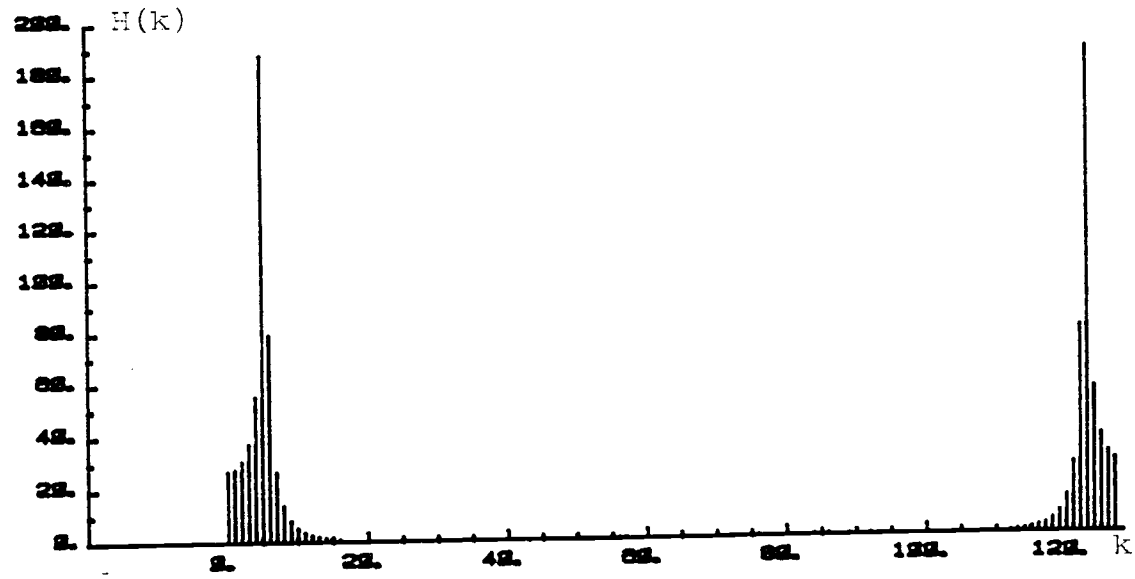


Figure 4.33 FFT of Gaussian Inverse Filter

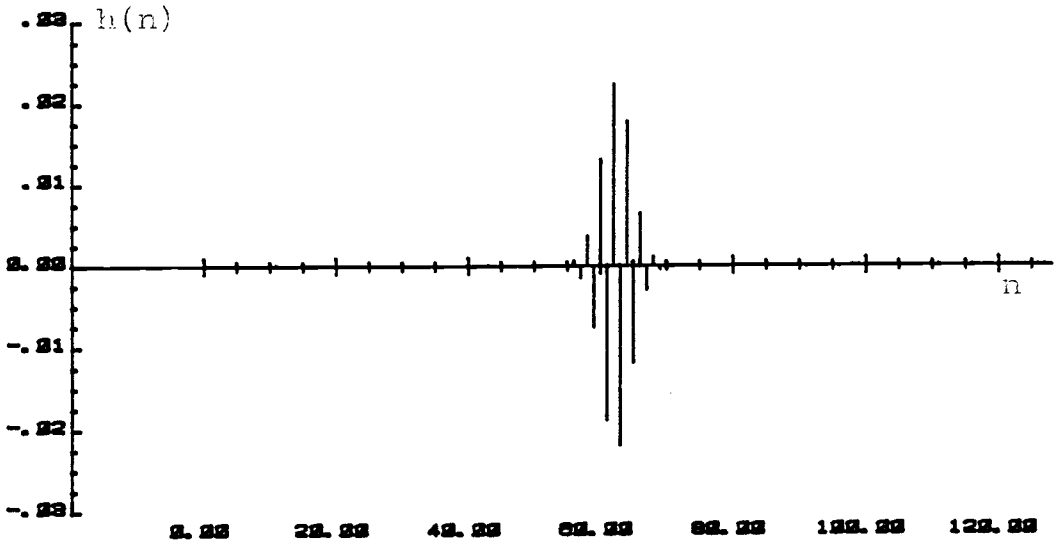


Figure 4.34 Gaussian Inverse Filter
Obtained after Circular Shifting Gaussian FFT

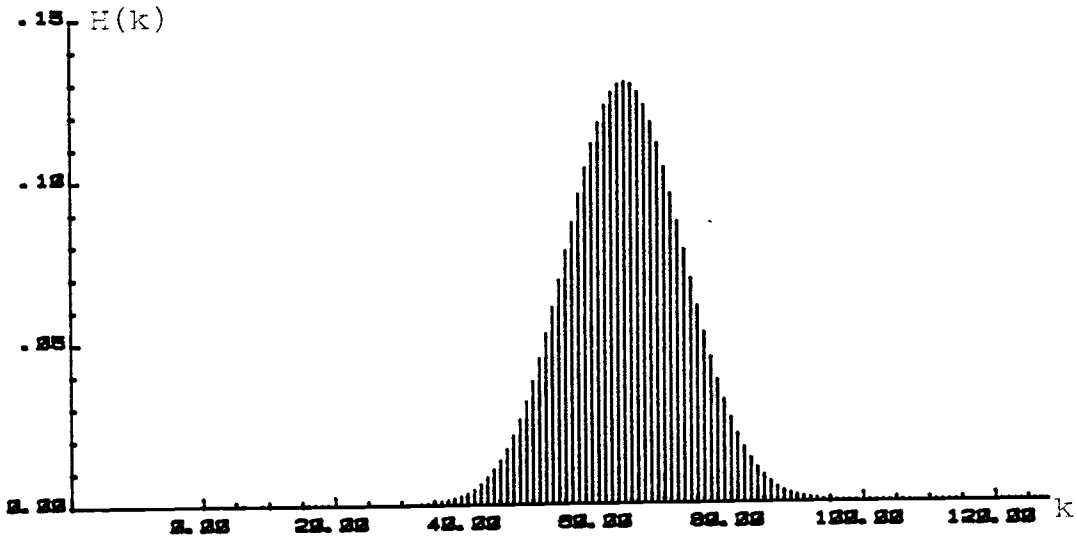


Figure 4.35 FFT of Gaussian Inverse Filter

Figure 4.34 shows the Gaussian inverse filter obtained after circular shifting the FFT of the Gaussian waveform by exactly half the sequence length (i.e. 64). Notice the short filter obtained.

After filtering. the odd values of the output have to be negated to yield the Gaussian reflector series. Negation is necessary to correct for the linear phase introduced by circularly shifting the Gaussian FFT.

4.6 Examples of Linear Filtering

Figures 4.36 to 4.44 show examples of linear filtering. The filters are obtained from figures 4.7 and 4.18. Both spike and Gaussian inverse filtering are performed. The mathematical expression of the basic wavelets can be found in section 4.4. Visually it was estimated that a good filter length to use would be 61. For purposes of comparison, both filter lengths were set equal. From the results of figure 4.17 and 4.18, the optimum output delay was visually estimated at 50 for the spike filter and 55 for the Gaussian inverse filter. The exact values of the filter coefficients are given in Appendix C. The results following are self-explanatory from the figures viewed in sequence. In the results of spike filtering in figure 4.40, the RMS error was about 1.88×10^{-4} and error at the spike peaks of less than 1×10^{-6}. For the

Gaussian inverse filtering of figure 4.41, the RMS error was about 6.53×10^{-4} and error at the Gaussian peaks of about 80×10^{-6}.

Figures 4.42 to 4.44 shows the deconvolution of closely occurring wavelets. It can be seen in figure 4.44 that Gaussian inverse filtering is unsuitable for wavelets that are closely separated. However, spike inverse filtering shows no difficulty in dealing with closely overlapping wavelets.

4.7 Example of Circular Inverse Filtering

In section 3.9 it was shown that inverse filtering can always be performed by circular convolution. For finite length sequences that are not too large, deconvolution can be performed without finding the non-aliased version of the inverse filter. Furthermore, no output delay is necessary. Figure 4.45 shows a superpositioned "chirp" wavelet of figure 4.8. The circular filter of figure 4.46 is the same as that of figure 4.10. On inverse filtering, the results show extremely accurate spikes at the times of occurrence of each wavelet. The RMS error was in the order of 1×10^{-16}, which is the accuracy of the machine.

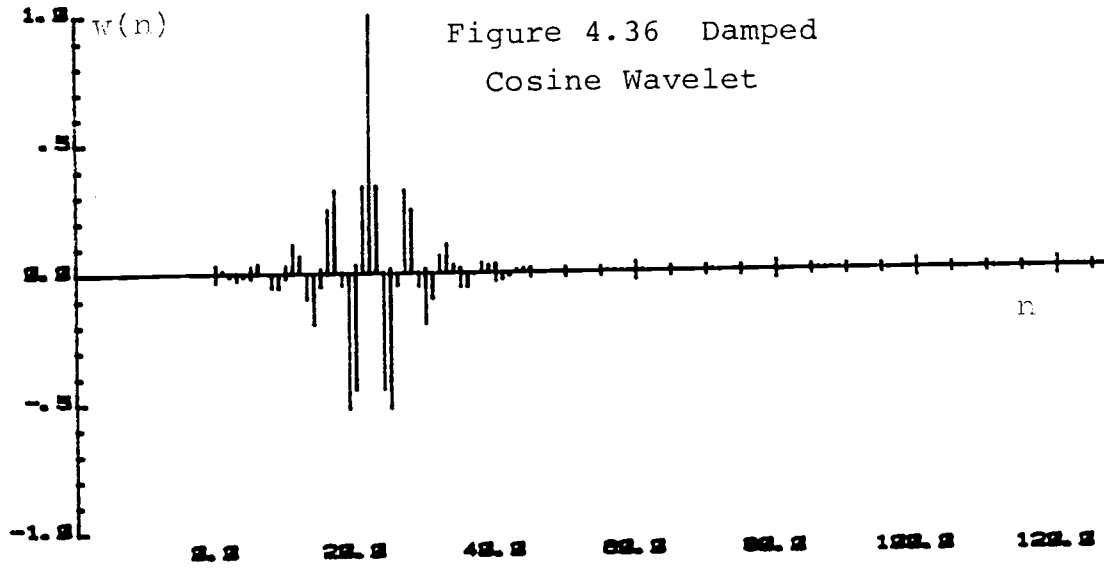


Figure 4.36 Damped Cosine Wavelet

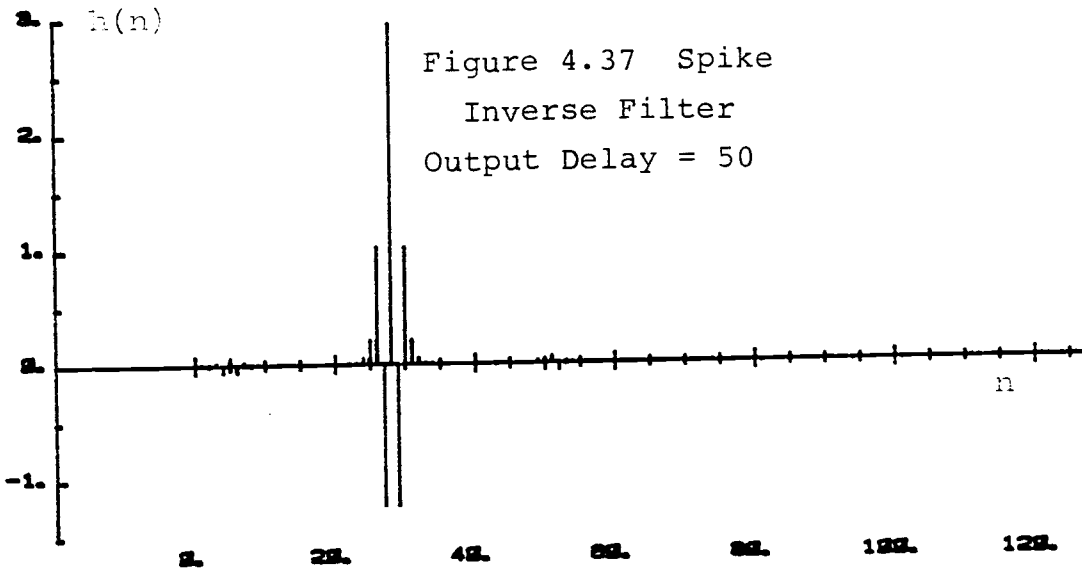


Figure 4.37 Spike Inverse Filter
Output Delay = 50

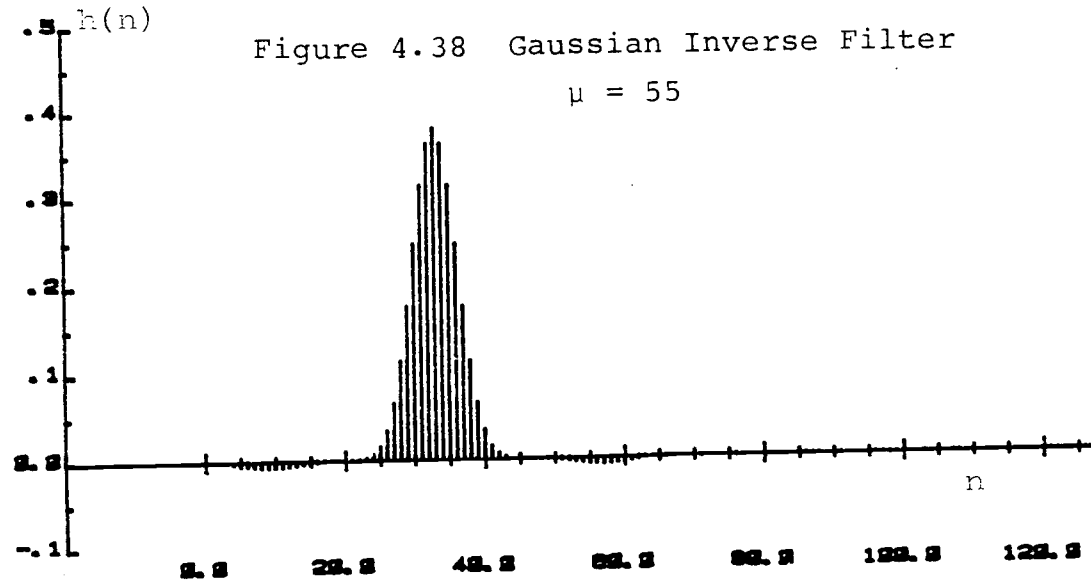


Figure 4.38 Gaussian Inverse Filter
$\mu = 55$

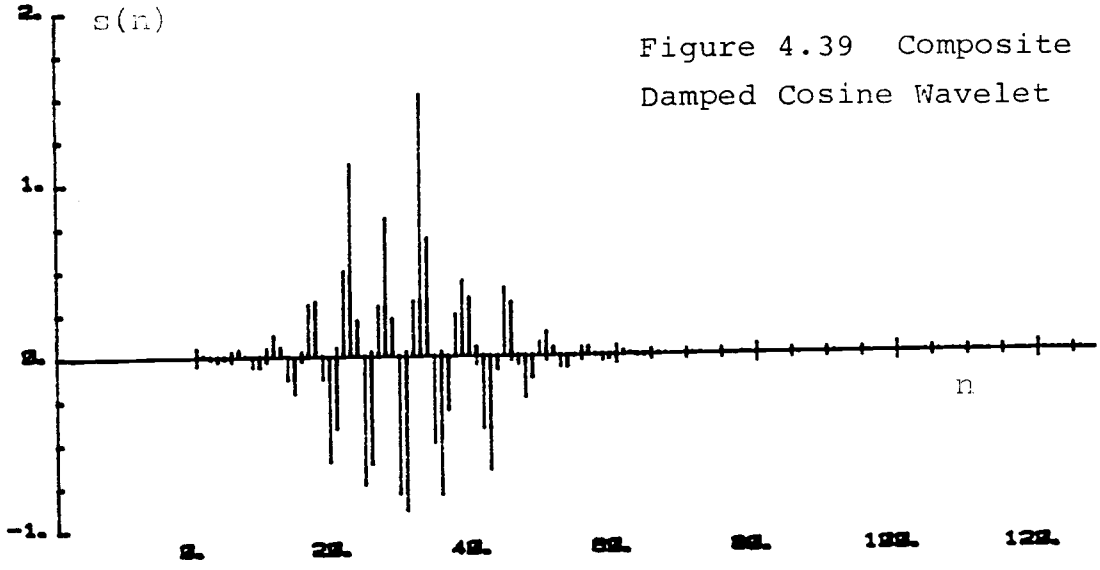


Figure 4.39 Composite Damped Cosine Wavelet

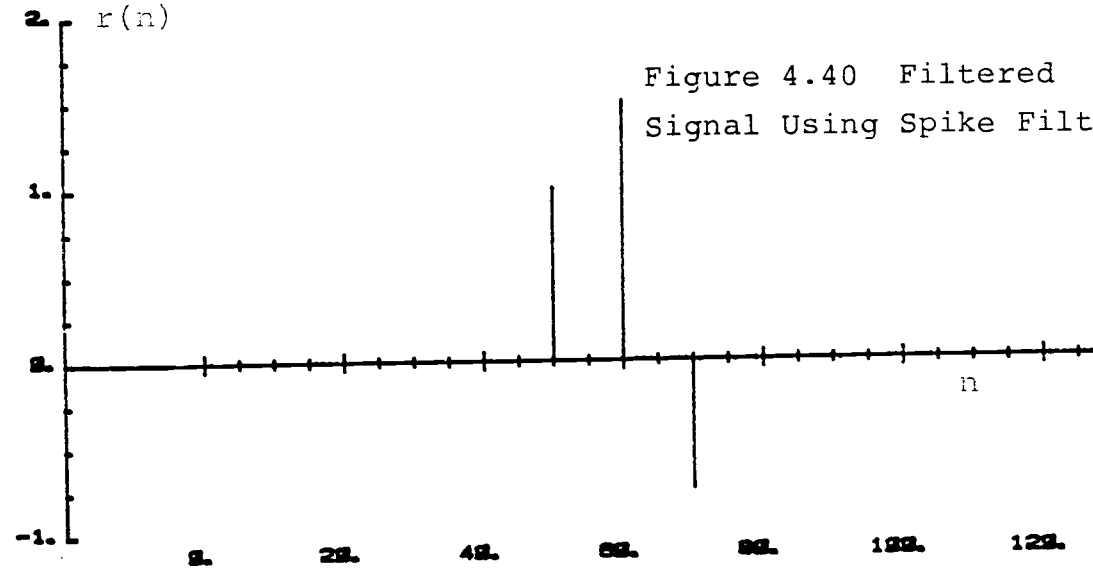


Figure 4.40 Filtered Signal Using Spike Filter

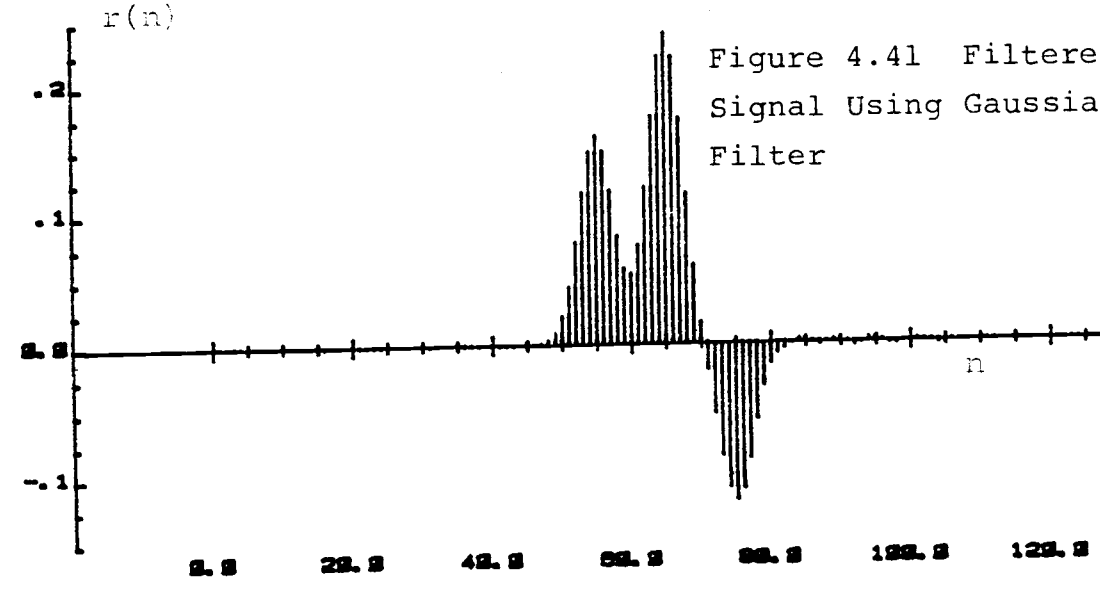


Figure 4.41 Filtered Signal Using Gaussian Filter

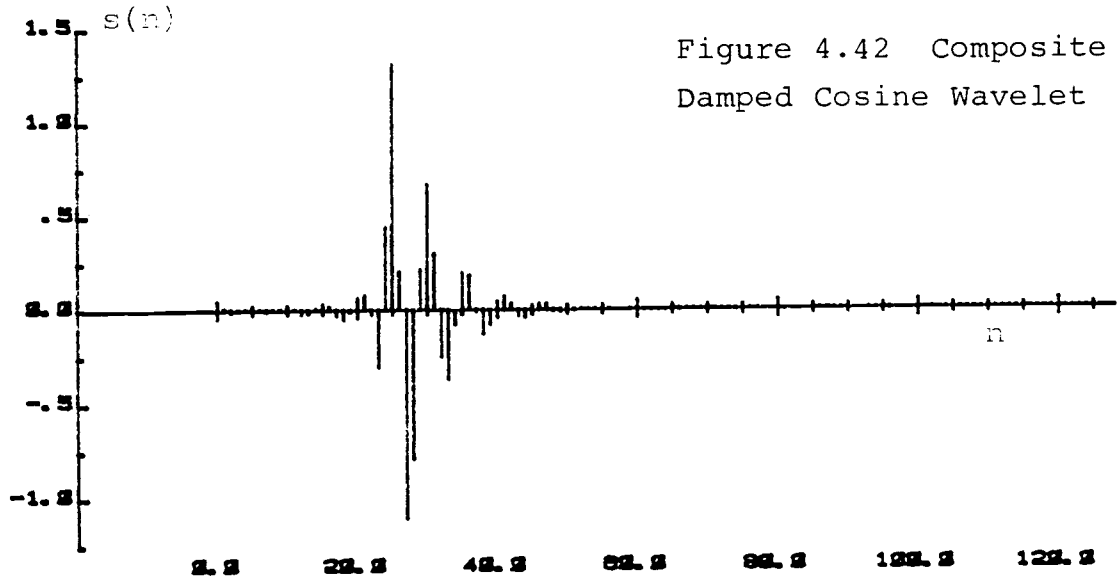


Figure 4.42 Composite Damped Cosine Wavelet

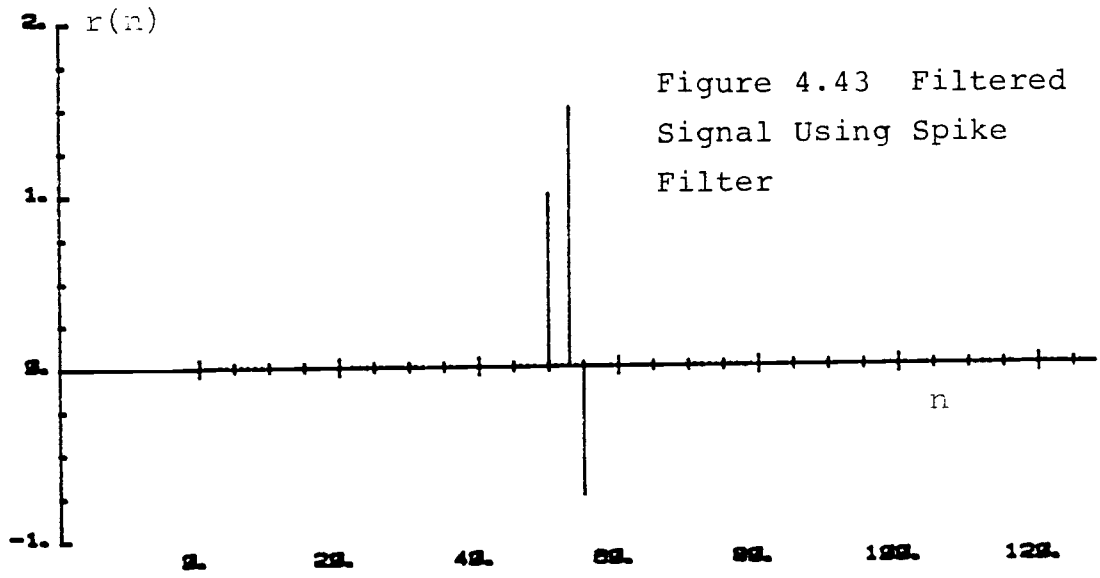


Figure 4.43 Filtered Signal Using Spike Filter

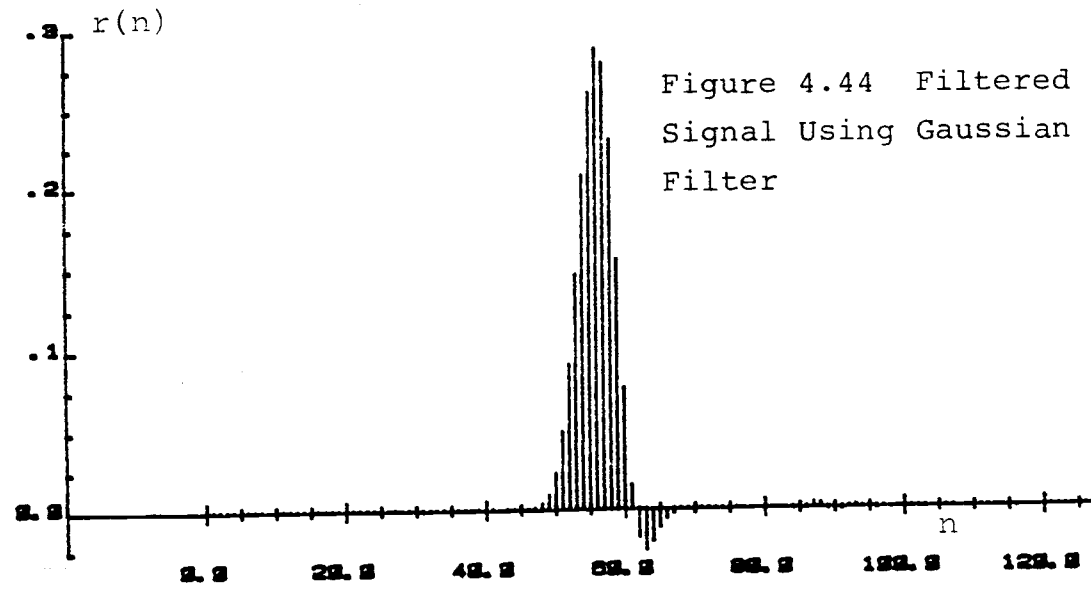


Figure 4.44 Filtered Signal Using Gaussian Filter

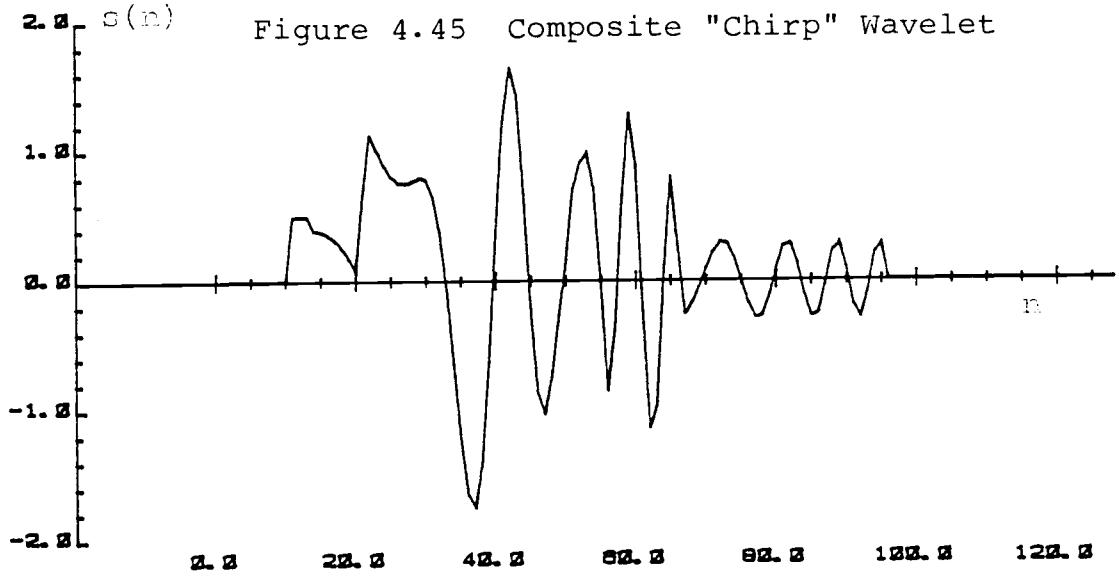


Figure 4.45 Composite "Chirp" Wavelet

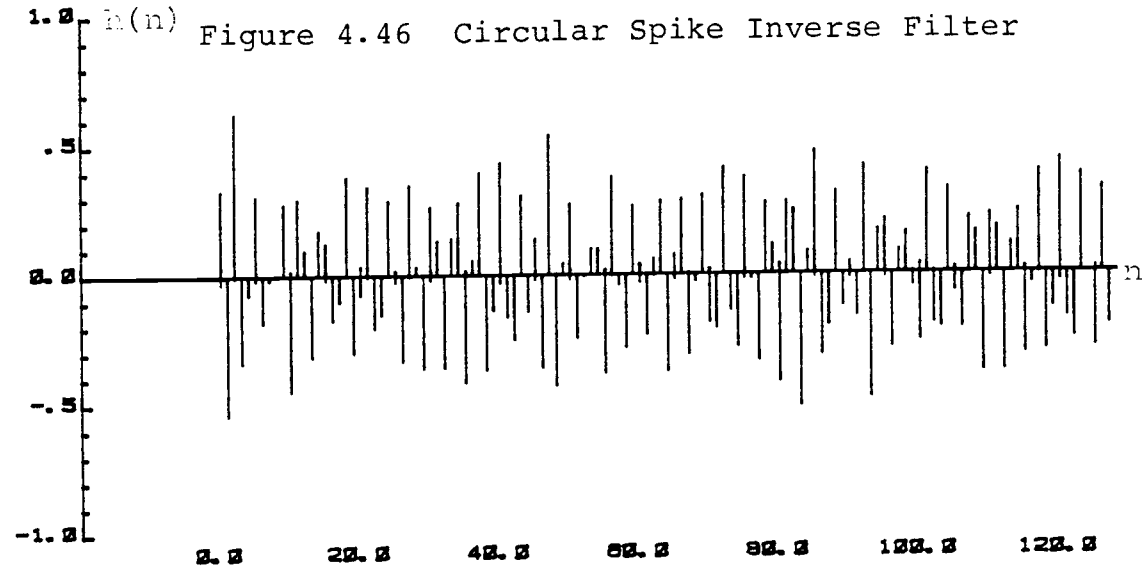


Figure 4.46 Circular Spike Inverse Filter

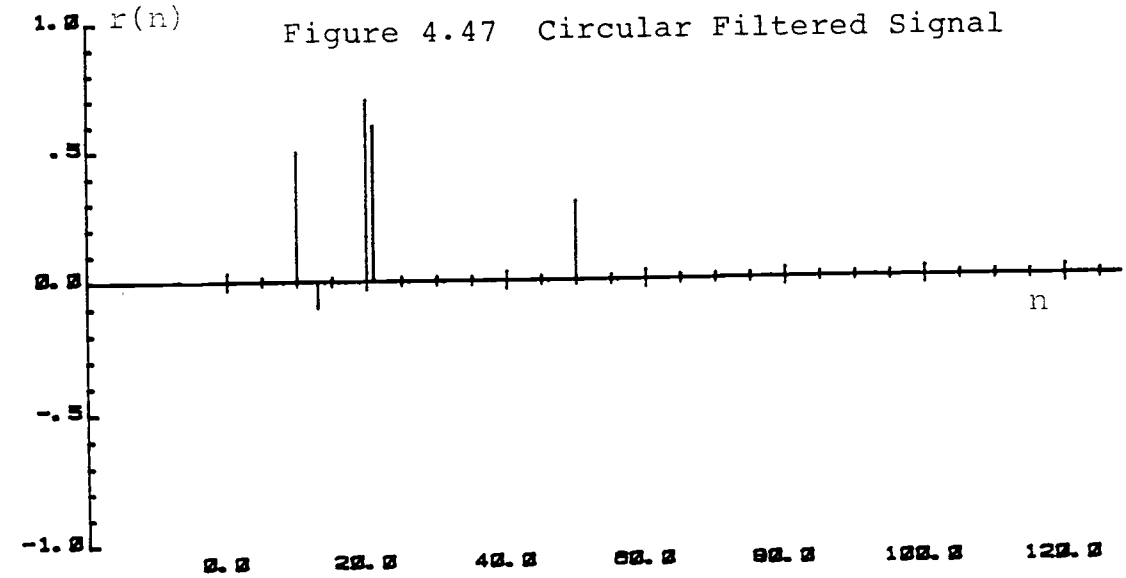


Figure 4.47 Circular Filtered Signal

4.8 Summary

Spike and Gaussian inverse filters can be found using frequency domain methods. The examples of this chapter have shown that frequency domain analysis is effective, especially since phase information is not discarded in the design procedure. Gaussian inverse filters may often be shorter than spike inverse filters. Circular shifting of the Gaussian FFT is an effective means of reducing the filter length without any loss of information at the output of the filter. The main disadvantage of the Gaussian reflector series is its inability to differentiate very closely occurring wavelets. For finite length sequences that are not too large, the most effective method of inverse filtering is by circular convolution.

V. SUMMARY/CONCLUSION

5.1 Summary of Results

The purpose of this thesis was twofold. First it was to investigate the practicability of designing deconvolution filters using the FFT. Second, it was to attempt to design inverse filters that will deconvolve composite signals into Gaussian wavelets.

In Chapter 1, the deconvolution model was presented and it was shown how frequency domain techniques may be used in designing the deconvolution filter. The frequency response of the required filter could be found by a term by term division of the DFT of the reflector wavelet by the DFT of the basic wavelet. The impulse response of the filter was found by taking the inverse FFT of the frequency response.

In Chapter 2, the characteristics of the Gaussian waveform was discussed and it was shown that the DFT of a Gaussian waveform is another Gaussian waveform.

Specifically, there exists an inverse relationship between the standard deviation of the time and that of the frequency waveform. For standard deviations greater than 2.0, the mid portion of the Gaussian FFT sequence has values that are extremely small and for most practical purposes may be approximated by zero with insignificant loss of information. This is an important result as it implies that when implementing the Gaussian inverse filter

using the FFT, a significant amount of computation time can be saved because of the zero values in the FFT sequence.

The main purpose of Chapter 3 was to show simple examples of designing deconvolution filters for short sequences. Frequency domain analysis via the FFT results in an aliased version of the true inverse (deconvolution) filter. The objective was to find the non-aliased version of the filter and approximate it with a finite length sequence.

It was shown that the length of the filter is dependent on the location of the zeros of the basic wavelet. Aliasing can usually be minimized by extending the length of the filter. If the non-aliased version of the filter can be found, then a finite length approximation to the true inverse filter can be determined. Minimum phase wavelets will have zero output delay while maximum phase basis wavelets will have delays equal to the filter length. Mixed phase basic wavelets have output delays that are between these two extremeties. The delay is determined by the position of the window which selects the filter coefficients to be implemented. The optimum window position is at the region where the average magnitude of the filter coefficients is maximum.

In Chapter 4, the practical aspects of finding the inverse filter is discussed and several examples of designing inverse

filters are shown. If the finite length inverse filters can be found, then filtering can be performed with linear convolution. If the inverse filter is quasistable with zeros on the unit circle, then filtering has to be performed with circular convolution.

The Gaussian waveform has many zeros that lie close to the unit circle on the left half of the z-plane. Because of this, the Gaussian inverse filter is sometimes much shorter than the corresponding spike inverse filter. By judiciously rotating the DFT of the Gaussian waveform, it is often possible to cancel out quasistable sequences arising from poles on the unit circle. Several examples of this have been shown in Chapter 4. Inverse filters that are implemented with circular convolution can always be found, but they cannot be used for deconvolving very long sequences.

5.2 Additional Research

A great deal of research involving the use of the FFT in deconvolution is presently in progress. However, the author has been unable to locate any material that deals particularly with Gaussian deconvolution. It is forseeable that more analysis and simulation is appropriate in this area. Of particular interest would be to apply the theory presented in this dissertation to practical real time signals.

Determination of the optimum window position in selecting the filter coefficients is another area that requires further research and simulation.

REFERENCES

1 Oppenheim, Alan V. and Schafer, Ronald W., Digital Signal Processing, Prentice-Hall, Inc. 1975.

2 Childers, Donald G. and Alan Durling, Digital Filtering and Signal Processing, West Publishing Company, 1975.

3 Kemerait, Robert Chester, "Signal Detection and Extraction by Cepstrum Techniques," Ph.D. thesis submitted to the Department of Electrical Engineering at the University of Florida, 1971.

4 Rabiner, Lawrence R. and Gold, Bernard, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc. 1975.

5 Rathja, Roy C., "Tone Burst Deconvolution," Ph.D. thesis submitted to the Department of Electrical and Computer Engineering at Oregon State University on April 22, 1980.

6 Weinstein, C.J., "Roundoff Noise in Floating Point Fast Fourier Transform," IEEE Trans. Audio Electroacoust., Vol. AU-17, Sept. 1969, pp. 209-215.

7 Peacock, K.L. and Treitel, Suen, "Predictive Deconvolution: Theory and Practice," Geophysics, Vol. 34, No. 2, April 1969, pp. 155-169.

A P P E N D I C E S

APPENDIX A: DFT OF GAUSSIAN WAVEFORM

STANDARD DEVIATION = 2.5

1	1.
2	.9924983681690 3
3	.9703294341219 9
4	.9344761302102 2
5	.8864960853298 2
6	.8284094312655 2
7	.7625579441684 5
8	.6914491830104 3
9	.6176000001775 37
10	.5433928405670 7
11	.4709555787672 6
12	.4020739786893 1
13	.3381359910123 2
14	.2801150440049 2
15	.2285814962866 4
16	.1837406766541 5
17	.1454886634865 6
18	.1134782422277 81
19	.7187785770092 0E-2
20	.5986987139901 7E-2
21	.4919495340908 63E-2
22	.6127874394960 2E-2
23	.6135508216864 36E-2
24	.8662365556142 093E-2
25	.1307270910061 71E-2
26	.9.039114027354 18E-3
27	6.156668011490 90E-3
28	4.130714783269 14E-3
29	2.730010296685 40E-3
30	1.777309097628 59E-3
31	1.139780560498 58E-3
32	7.200109976623 00E-4
33	4.480398623234 651E-4
34	2.746333623614 763E-4
35	1.658251504148 21E-4
36	9.862946909523 52E-5
37	5.778559947337 891E-5
38	3.335017327535 21E-5
39	1.895597758439 747E-5
40	1.061763777314 15E-5
41	5.857094274610 64E-6
42	3.182703095204 46E-6
43	1.703607884433 857E-6
44	8.982614440739 00E-7
45	4.665470840726 70E-7
46	2.386974626272 10E-7
47	1.202983660464 34E-7
48	5.972157559978 88E-8
49	2.920534834804 92E-8
50	1.406886727094 497E-8
51	6.675802396993 75E-9
52	3.120422718827 63E-9
53	1.436756123796 17E-9
54	6.516469516778 43E-10
55	2.911398552815 63E-10
56	1.281301176347 20E-10
57	5.554735838838 75E-11
58	2.372144151458 25E-11
59	9.979281387401 67E-12
60	4.136025214419 13E-12
61	1.689649120216 21E-12
62	6.819431584082 84E-13
63	2.758412615005 51E-13
64	1.215745347783 41E-13
65	8.057443601217 07E-14
66	1.201541608441 20E-13
67	2.745304973551 06E-13
68	6.808028046243 47E-13
69	1.688856597045 253E-12
70	4.135136705510 37E-12
71	9.978531900092 33E-12
72	2.372090025430 07E-11
73	5.554680326851 33E-11
74	1.281298400720 33E-10
75	2.911396609908 80E-10
76	6.516470002496 46E-10
77	1.436756144612 75E-9
78	3.120423031077 80E-9
79	6.675802754346 79E-9
80	1.406867330272 51E-8
81	2.920534834804 92E-8
82	5.972157593805 98E-8
83	1.202983663725 62E-7
84	2.386974632378 33E-7
85	4.665470845011 47E-7
86	8.982614448952 92E-7
87	1.703607885224 40E-6
88	3.182703096298 21E-6
89	5.857094275273 74E-6
90	1.061763777444 06E-5
91	1.895597758452 192E-5
92	3.335017327691 94E-5
93	5.778599473503 94E-5
94	9.862946909704 44E-5
95	1.658251504164 98E-4
96	2.746333623616 829E-4
97	4.480398623234 651E-4
98	7.200109976636 16E-4
99	1.139780560499 70E-3
100	1.777309097630 15E-3
101	2.730010296686 39E-3
102	4.130714783270 94E-3
103	6.156668011492 39E-3
104	9.039114027356 26E-3
105	1.307270910061 79E-2
106	1.862365556142 116E-2
107	2.613508216864 55E-2
108	3.612787439496 29E-2
109	4.919495340908 79E-2
110	6.598698713990 46E-2
111	8.718778577009 43E-2
112	.1134782422277 81
113	.1454886634865 6
114	.1837406766541 5
115	.2285814962866 5
116	.2801150440049 2
117	.3381359910123 2
118	.4020739786893 2
119	.4709555787672 26
120	.5433928405670 8
121	.6176000001775 37
122	.6914491830104 3
123	.7625579441684 5
124	.8284094312655 2
125	.8864960853298 2
126	.9344761302102 1
127	.9703294341219 8
128	.9924983681690 3

APPENDIX B: DFT OF GAUSSIAN WAVEFORM

STANDARD DEVIATION = 5.0

1	1.00000000000000
2	.970329434121999
3	.886496085329882
4	.762557944168445
5	.617600001775370
6	.470955787672260
7	.338135991012320
8	.228581496286604
9	.145488663486560
10	8.71877857700919E-2
11	4.91949534090862E-2
12	2.61350821686433E-2
13	1.30727091006168E-2
14	6.15666801149056E-3
15	2.73001029668509E-3
16	1.13978056049819E-3
17	4.48039823234661E-4
18	1.65582515041473E-4
19	5.77855994733700 3E-5
20	1.89559775843784 9E-5
21	5.85709427445742E-6
22	1.70360788412323E-6
23	4.66547083708445E-7
24	1.20298365484160E-7
25	2.92053481394579E-8
26	6.67580193449926E-9
27	1.43675550556295E-9
28	2.91138881938173E-10
29	5.55462823138134E-11
30	9.97777480912949E-12
31	1.68722489630257 8E-12
32	2.68822783753545 6E-13
33	4.02872180060854E-14
34	5.58172009097718 1E-15
35	7.02435880377832E-16
36	8.81705414284898E-17
37	1.57551132745622E-16
38	3.17551196358445 8E-16
39	3.07321943529935E-16
40	4.58025453309626E-16
41	7.01846964599829E-17
42	1.60314107360 13E-16
43	2.21173614469983E-16
44	2.77512839107230E-16
45	2.67408986278357E-16
46	4.29213360658268E-16
47	4.42312041847334E-16
48	3.94479909190932E-16
49	1.81457569421976E-16
50	1.40225691856193 6E-16
51	1.35101980551643E-16
52	2.60005965331681E-16
53	1.48801524557824E-16
54	.901948129941 45E-16
55	.107700944850 88E-16
56	.875098659114 99E-16
57	.996417643213 58E-16
58	.363464563128 64E-16
59	.397329554974 71E-16
60	1.09107066148993E-16
61	1.27048096655989E-16
62	1.65457910080698E-16
63	1.89237262149758E-16
64	2.17450760368678E-16
65	2.77555756156289E-17
66	1.30329320334715E-16
67	1.36109481554223E-16
68	2.50862826372565E-16
69	1.35364019818864E-16
70	2.62123365363411E-16
71	2.60642202265515E-16
72	3.88088977131536E-16
73	7.47794603228970E-17
74	1.91881126735933E-16
75	1.99066723131493E-16
76	3.41687362059199E-16
77	2.49202548502631E-16
78	4.01425836060898E-16
79	4.43858409962743E-16
80	5.76132281126528E-16
81	9.78788114969022E-16
82	1.31160317225591E-16
83	1.30770766735852E-16
84	2.77950890755001E-16
85	1.41422801762491E-16
86	2.95509792251449E-16
87	3.15162373826721E-16
88	4.87096314353648E-16
89	1.56066155878214E-16
90	3.63792334457860E-16
91	4.55501487762164E-16
92	7.09158810464119E-16
93	7.03660904573304E-16
94	1.07456599389861E-15
95	1.84498022949007E-15
96	6.97664276740358E-15
97	4.02872180060854E-14
98	2.68890871395144 8E-13
99	1.68783982021445E-12
100	9.97851753550124E-12
101	5.55467427621704E-11
102	2.91139671374782E-10
103	1.43675613826660E-9
104	6.67580279517259E-9
105	2.92053483339448E-8
106	1.20298366399796E-7
107	4.66547084498282E-7
108	1.70360788524589E-6
109	5.85709427522888E-6
110	1.89559775845191 3E-5
111	5.77855994734971 3E-5
112	1.65582515041645 8E-4
113	4.48039823234661E-4
114	1.13978056049973E-3
115	2.73001029668638E-3
116	6.15666801149241E-3
117	1.30727091006179E-2
118	2.61350821686455E-2
119	4.91949534090878E-2
120	8.71877857700943E-2
121	.145488663486560
122	.228581496286605
123	.338135991012320
124	.470955578767226
125	.617600001775370
126	.762557944168445
127	.886496085329882
128	.970329434121998

SPIKE FILTER

```
h(  0) =    -.001629
h(  1) =    -.004987
h(  2) =    -.010414
h(  3) =     .015646
h(  4) =    -.058186
h(  5) =     .052925
h(  6) =    -.056518
h(  7) =     .020197
h(  8) =    -.009752
h(  9) =    -.001532
h( 10) =    -.000223
h( 11) =    -.000026
h( 12) =    -.000001
h( 13) =     .000002
h( 14) =     .000000
h( 15) =     .000000
h( 16) =     .000000
h( 17) =     .000002
h( 18) =     .000004
h( 19) =     .000017
h( 20) =     .000083
h( 21) =     .000400
h( 22) =     .001921
h( 23) =     .009204
h( 24) =     .044049
h( 25) =     .210490
h( 26) =    1.015800
h( 27) =   -1.233100
h( 28) =   -2.950400
h( 29) =   -1.233100
h( 30) =    1.015800
h( 31) =     .210490
h( 32) =     .044049
h( 33) =     .009204
h( 34) =     .001921
h( 35) =     .000400
h( 36) =     .000083
h( 37) =     .000017
h( 38) =     .000004
h( 39) =     .000002
h( 40) =     .000000
h( 41) =     .000000
h( 42) =     .000000
h( 43) =     .000002
h( 44) =    -.000001
h( 45) =    -.000026
h( 46) =    -.000223
h( 47) =    -.001532
h( 48) =    -.009752
h( 49) =     .020197
h( 50) =    -.056518
h( 51) =     .052925
h( 52) =    -.058186
h( 53) =     .015646
h( 54) =    -.010414
h( 55) =    -.004987
h( 56) =    -.001629
h( 57) =    -.000463
h( 58) =    -.000122
h( 59) =    -.000031
h( 60) =    -.000007
```

GAUSSIAN FILTER

```
h(  0) =    -.000145
h(  1) =    -.000325
h(  2) =    -.000657
h(  3) =    -.001197
h(  4) =    -.001973
h(  5) =    -.002962
h(  6) =    -.004065
h(  7) =    -.005118
h(  8) =    -.005923
h(  9) =    -.006310
h( 10) =    -.006198
h( 11) =    -.005620
h( 12) =    -.004703
h( 13) =    -.003627
h( 14) =    -.002573
h( 15) =    -.001674
h( 16) =    -.000994
h( 17) =    -.000536
h( 18) =    -.000257
h( 19) =    -.000100
h( 20) =     .000009
h( 21) =     .000106
h( 22) =     .000697
h( 23) =     .002202
h( 24) =     .006163
h( 25) =     .015277
h( 26) =     .033507
h( 27) =     .065176
h( 28) =     .112980
h( 29) =     .175670
h( 30) =     .246510
h( 31) =     .313530
h( 32) =     .362160
h( 33) =     .380000
h( 34) =     .362160
h( 35) =     .313530
h( 36) =     .246510
h( 37) =     .175670
h( 38) =     .112980
h( 39) =     .065176
h( 40) =     .033507
h( 41) =     .015277
h( 42) =     .006163
h( 43) =     .002202
h( 44) =     .000697
h( 45) =     .000106
h( 46) =     .000009
h( 47) =    -.000100
h( 48) =    -.000257
h( 49) =    -.000536
h( 50) =    -.000994
h( 51) =    -.001674
h( 52) =    -.002573
h( 53) =    -.003627
h( 54) =    -.004703
h( 55) =    -.005620
h( 56) =    -.006198
h( 57) =    -.006310
h( 58) =    -.005923
h( 59) =    -.005118
h( 60) =    -.004065
```

FILTER USING THE FLOWCHART OF FIGURE 4.1

```
0001  FTN77,L,Y
0002  $FILES(3,3)
0003
0004        PROGRAM FILTR
0005
0006  C*****************************************************************
0007  C     THIS PROGRAM USES THE FFT TO DESIGN THE INVERSE FILTER FOR
0008  C     A SPIKE OR GAUSSIAN REFLECTOR WAVELET
0009  C*****************************************************************
0010
0011        DOUBLE COMPLEX S(128),G(128),RESF(128),RESC(128)
0012        DOUBLE PRECISION XMAX,XMIN
0013        CHARACTER*15 OF1,OF2,OF3,OFS,OFSFT,HKM,OFP
0014        INTEGER NZ,NLT
0015        DATA S,G/256*(0.0,0.0)/
0016
0017  C     SET ARRAY SIZE
0018        ISIZE=128
0019        NUMFFT=7
0020
0021        WRITE(1,*) 'OUTFILE FOR SIGNAL'
0022        CALL NAME(OFS)
0023        WRITE(1,*) 'OUTFILE FOR SIGNAL FFT'
0024        CALL NAME(OFSFT)
0025        WRITE(1,*) 'OUTFILE FOR H(K) FFT-MAGNITUDE'
0026        CALL NAME(HKM)
0027        WRITE(1,*) 'FILTER IMPULSE RESPONSE : MAGNITUDE'
0028        CALL NAME(OF1)
0029        WRITE(1,*) 'FILTER IMPULSE RESPONSE : PHASE'
0030        CALL NAME(OFP)
0031        WRITE(1,*) 'FILTER IMPULSE RESPONSE : COMPLEX NOS.'
0032        CALL NAME(OF2)
0033        WRITE(1,*) 'TEST CONVOLUTION RESULT'
0034        CALL NAME(OF3)
0035
0036        WRITE(1,*) 'ENTER TYPE OF SIGNAL :  1) SINE    2) SIGNL'
0037        WRITE(1,*) '                        3) SEQ1'
0038        WRITE(1,*) '                        4) SPECIFIED SPIKES'
0039        WRITE(1,*) '                        5) CHIRP'
0040        WRITE(1,*) '                        6) EXPONENTIAL PULSE'
0041        READ(1,*) NTY
0042
0043        IF (NTY.EQ.2) CALL SIGNL(S,ISIZE)
0044        IF (NTY.EQ.1) CALL SINE(S,ISIZE)
0045        IF (NTY.EQ.3) CALL SEQ1(S,ISIZE)
0046        IF (NTY.EQ.4) CALL SPIKE(S,ISIZE)
0047        IF (NTY.EQ.5) CALL CHIRP(S,ISIZE)
0048        IF (NTY.EQ.6) CALL EXPON(S,ISIZE)
0049
0050        WRITE(1,*) 'ADD NOISE TO SIGNAL ?   ENTER "1" FOR YES'
0051        READ(1,*) NNO
0052        IF (NNO.EQ.1) CALL NOISE (S,ISIZE)
0053
0054        CALL PRTC(S,ISIZE,OFS)
0055        CALL FFT1D(S,NUMFFT)
0056        CALL PRTM(S,ISIZE,OFSFT)
0057
0058        CALL SCAN1(S,ISIZE,XMAX,XMIN,NZ,NLT)
0059        WRITE(1,*) 'ARRAY S:'
0060        WRITE(1,5) XMAX,XMIN,NZ,NLT
0061   5    FORMAT('XMAX = ',E25.17,/'XMIN = ',E25.17/'NUMBER OF ZEROS =',I4/
0062       X       'NUMBER OF SMALL VALUES =',I4 //)
0063
0064        WRITE(1,*) 'STOP PROGRAM ?    1) YES      2) NO'
0065        READ(1,*) NSTOP
0066        IF (NSTOP.EQ.1) STOP
0067
0068
0069  C     SPECIFY REFLECTOR REFLECTOR WAVELET:
0070        WRITE(1,*) 'DECON INTO:  1) SPIKE   2) GAUSS'
0071        READ(1,*) N
0072        IF (N.EQ.1) CALL SPIKE(G,ISIZE)
0073        IF (N.EQ.2) CALL GAUSS(G,ISIZE)
0074
0075        CALL FFT1D(G,NUMFFT)
0076        CALL SCAN1(G,ISIZE,XMAX,XMIN,NZ,NLT)
0077        WRITE(1,*)'ARRAY G:'
0078        WRITE(1,5) XMAX,XMIN,NZ,NLT
```

```
0079
0080
0081   C      TRUNCATION OF FFT DATA (IF SPECIFIED);
0082          WRITE(1,*) 'ZERO OUT SMALL VALUES OF G(K) ?  - "1" IS YES'
0083          READ(UNIT=1,FMT=**,ERR=11) NA1
0084          IF (NA1.EQ.1) CALL ZEROX(G,128)
0085
0086
0087   C      CIRCULAR SHIFT OF FFT DATA (IF SPECIFIED)
0088          WRITE(1,*) 'SHIFT G(K) ?   "1" IS YES'
0089          READ(1,*) NSH
0090          IF (NSH.EQ.1) CALL SHIFT(G,ISIZE)
0091
0092    11    CALL DIVIS(S,G,RESF,ISIZE)
0093          CALL SCAN1(RESF,ISIZE,XMAX,XMIN,NZ,NLT)
0094          WRITE(1,*) 'ARRAY RESF'
0095          WRITE(1,5)XMAX,XMIN,NZ,NLT
0096
0097          CALL PRTM(RESF,ISIZE,HKM)
0098          CALL IFFTD(RESF,NUMFFT)
0099          CALL PRTM(RESF,ISIZE,OF1)
0100          CALL PRTP(RESF,ISIZE,OFP)
0101          CALL PRTC(RESF,ISIZE,OF2)
0102
0103          CALL IFFTD(S,NUMFFT)
0104          CALL CCON(S,RESF,RESC,ISIZE)
0105          CALL PRTM(RESC,ISIZE,OF3)
0106          STOP
0107          END
```

```
0001  FTN77,L,Y
0002  $FILES(0,0)
0003        SUBROUTINE FFT1D(X,M)
0004  C     DOUBLE PRECISION FFT
0005
0006        DOUBLE COMPLEX X(1024),U,W,T
0007        DOUBLE PRECISION PI
0008        N=2**M
0009        NV2=N/2
0010        NM1=N-1
0011        J=1
0012        DO 7 I=1,NM1
0013        IF (I.GE.J) GOTO 5
0014        T=X(J)
0015        X(J)=X(I)
0016        X(I)=T
0017   5    K=NV2
0018   6    IF (K.GE.J) GOTO 7
0019        J=J-K
0020        K=K/2
0021        GOTO 6
0022   7    J=J+K
0023        PI=3.1415926535897D0
0024        DO 20 L=1,M
0025        LE=2**L
0026        LE1=LE/2
0027        U=(1.0,0.0)
0028        W=DCMPLX(DCOS(PI/DBLE(LE1)),-DSIN(PI/DBLE(LE1)))
0029        DO 20 J=1,LE1
0030        DO 10 I=J,N,LE
0031        IP=I+LE1
0032        T=X(IP)*U
0033        X(IP)=X(I)-T
0034   10   X(I)=X(I)+T
0035   20   U=U*W
0036        RETURN
0037        END
0038
0039
0040
0041        SUBROUTINE IFFTD(X,M)
0042        DOUBLE COMPLEX X(1024)
0043        DOUBLE PRECISION A
0044        N=2**M
0045        DO 10 I=1,N
0046           X(I)=CONJG(X(I))
0047   10      CONTINUE
0048
0049        CALL FFT1D(X,M)
0050
0051        A=1.0/DBLE(N)
0052        DO 20 I=1,N
0053        X(I)= A*CONJG(X(I))
0054   20   CONTINUE
0055        RETURN
0056        END
0057
```

```
0001  FTN77,L,Y
0002
0003
0004          SUBROUTINE GAUSS(X,ISIZE)
0005  C       THIS SUB CALCULATES A GAUSSIAN WAVELET
0006
0007          DOUBLE COMPLEX X(ISIZE)
0008          DOUBLE PRECISION C1,C2,A,SIGMA,XM,PI2,00
0009
0010          WRITE(1,*) 'ENTER MEAN(U) , SIGMA  '
0011          READ(1,*) XM,SIGMA
0012          PI2=2.0*3.14159265358979D0
0013          C1=1.0/DSQRT(PI2)
0014          00=0.0
0015          DO 10 I= 1,ISIZE
0016            A = DBLE(I)
0017            C2=DEXP( (-0.5*(A-XM)**2.0)/(SIGMA**2.0) )
0018            A= C2*C1/SIGMA
0019            X(I)= DCMPLX(A,00)
0020    10      CONTINUE
0021          RETURN
0022          END
0023
0024
0025
0026
0027          SUBROUTINE SINE(X,ISIZE)
0028  C       THIS SUB SAMPLES A SINUSOID
0029
0030          DOUBLE COMPLEX X(ISIZE)
0031          DOUBLE PRECISION PI2,THETA,A,FREQ,FSAM,00
0032
0033          DO 10 I=1,ISIZE
0034    10      X(I)=(0.0,0.0)
0035
0036          00=0.0
0037          PI2 = 2.0 * 3.14159265358979D0
0038          WRITE(1,*) 'ENTER SINE WAVE FREQUENCY'
0039          READ(1,*) FREQ
0040          WRITE(1,*) 'ENTER SAMPLING FREQUENCY'
0041          READ(1,*) FSAM
0042          WRITE(1,*) 'FREQ = ',FREQ,'       FSAMP = ',FSAM
0043
0044          WRITE(1,*) 'ARRAY SIZE= ',ISIZE
0045          WRITE(1,*) 'ENTER # DATA POINTS'
0046          READ(1,*) N
0047
0048          DO 20 I=1,N
0049            K=I-1
0050            THETA=PI2*FREQ*(1.0/FSAM)*DBLE(K)
0051            A=DSIN(THETA)
0052            X(I)=DCMPLX(A,00)
0053    20      CONTINUE
0054
0055          RETURN
0056          END
0057
0058
0059          SUBROUTINE SPIKE(X,ISIZE)
0060  C       SETS A SERIES OF SPIKES
0061
0062          DOUBLE COMPLEX X(ISIZE)
0063
0064          DO 5 I=1,ISIZE
0065    5     X(I)=(0.0,0.0)
0066          WRITE(1,*) 'ENTER NUMBER OF SPIKES'
0067          READ(1,*) N1
0068          DO 10 I=1,N1
0069            WRITE(1,*) 'ENTER SPIKE # ',I,'  LOCATION,MAGNITUDE'
0070            READ(1,*) N,A
0071            X(N)= A * (1.0,0.0)
0072    10      CONTINUE
0073          RETURN
0074          END
0075
0076
0077
0078          SUBROUTINE PRTM(X,ISIZE,OUTFL)
```

```
0079  C      PRINTS THE MAG OF A CMPLX ARRAY
0080
0081           CHARACTER*15 OUTFL
0082           DOUBLE COMPLEX X(ISIZE)
0083           OPEN(UNIT=15,FILE=OUTFL)
0084           WRITE(1,*) 'FILE OPENED TO WRITE : ',OUTFL
0085           DO 10 I=1,ISIZE
0086             WRITE(15,*) I,'   ',ABS(X(I))
0087    10     CONTINUE
0088           CLOSE(15)
0089           RETURN
0090           END
0091
0092
0093           SUBROUTINE PRTR(X,ISIZE,OUTFL)
0094  C        PRINTS THE REAL PART OF A CMPLX ARRAY
0095
0096           CHARACTER*15 OUTFL
0097           DOUBLE COMPLEX X(ISIZE)
0098           OPEN(UNIT=15,FILE=OUTFL)
0099           WRITE(1,*) 'FILE OPENED TO WRITE : ',OUTFL
0100           DO 10 I=1,ISIZE
0101             WRITE(15,*) I,'   ',DBLE(X(I))
0102    10     CONTINUE
0103           CLOSE(15)
0104           RETURN
0105           END
0106
0107
0108           SUBROUTINE PRTC(X,ISIZE,OUTFL)
0109  C        PRINTS THE CMPLX VALUES OF A CMPLX ARRAY
0110
0111           CHARACTER*15 OUTFL
0112           DOUBLE COMPLEX X(ISIZE)
0113           OPEN(UNIT=15,FILE=OUTFL)
0114           WRITE(1,*) 'FILE OPENED TO WRITE : '.OUTFL
0115           DO 10 I=1,ISIZE
0116             WRITE(15,*) I,'   ',X(I)
0117    10     CONTINUE
0118           CLOSE(15)
0119           RETURN
0120           END
0121
0122
0123           SUBROUTINE PRTP(X,ISIZE,OUTFL)
0124  C        PRINTS THE PHASE OF CMPLX ARRAY
0125
0126           CHARACTER*15 OUTFL
0127           DOUBLE COMPLEX X(ISIZE)
0128           OPEN(UNIT=15,FILE=OUTFL)
0129           WRITE(1,*) 'FILE OPENED TO WRITE : ',OUTFL
0130           DO 10 I=1,ISIZE
0131             A=REAL(IMAG(X(I)))
0132             B=REAL(X(I))
0133             C = ATAN2 ( A,B )
0134    10       WRITE(15,*) I,C
0135           CLOSE(15)
0136           RETURN
0137           END
0138
0139           SUBROUTINE NAME(FLNME)
0140  C        NAMES A FILE
0141
0142           CHARACTER*15 FLNME
0143           WRITE(1,*) 'ENTER NAME OF FILE'
0144           READ(1,*) FLNME
0145           WRITE(1,*) 'FILE NAME IS : ',FLNME
0146           WRITE(1,*) ' '
0147           RETURN
0148           END
0149
0150
0151           SUBROUTINE DIVIS(S,G,RESF,ISIZE)
0152  C        DIVISION ROUTINE : IF DENOMINATOR IS ZERO,THEN
0153  C                           NEUMERATOR IS ALSO SET TO ZERO
0154
0155           DOUBLE COMPLEX S(ISIZE),G(ISIZE),RESF(ISIZE)
0156           DOUBLE PRECISION T
0157
0158           WRITE(1,*) 'DIVISION ROUTINE'
```

```
      DO 10 I=1,ISIZE
        T=ABS( S(I) )
        IF (T.NE.0.0) THEN
            RESF(I) = G(I)/S(I)
          ELSE
            WRITE(1,*) 'ZERO AT I = ',I
            RESF(I) = (0.0,0.0)
          ENDIF
   10   CONTINUE
      RETURN
      END


      SUBROUTINE SCAN1(X,LX,XMAX,XMIN,NZ,NLT)
C     SCANS AN ARRAY FOR LARGE AND SMALL VALUES
      DOUBLE COMPLEX X(LX)
      DOUBLE PRECISION XMAX,XMIN,T,REF
      INTEGER NZ,NLT

      NZ=0
      NLT=0
      XMIN= ABS(X(1))
      XMAX= XMIN
      DO 10 I=2,LX
        T=ABS(X(I))
        IF (T.GT.XMAX) XMAX=T
        IF (T.LT.XMIN) XMIN=T
   10   CONTINUE

      REF= 1.0D-10
      WRITE(1,*) 'REFERENCE MAGNITUDE = ',REF
      DO 20 I=1,LX
        T= ABS(X(I))
        IF (T.LT.REF) NLT=NLT+1
        IF (T.EQ.0.0) NZ=NZ+1
   20   CONTINUE

      RETURN
      END


      SUBROUTINE CCON(X1,X2,Y,LX)
C     CIRCULAR CONVOLUTION  : X1,X2 - INPUTS
C                           : Y     - OUTPUT
C                           : LX    - LENGTH

      DOUBLE COMPLEX X1(LX),X2(LX),Y(LX),T,SUM

      WRITE(1,*) 'CIRCULAR CONVOLUTION ROUTINE'

      DO 10,I=1,INT(LX/2)
        T=X1(I)
        X1(I)=X1(LX-I+1)
   10   X1(LX-I+1)=T

      DO 40 IND=1,LX
        T=X1(LX)
        DO 20 I=LX,2,-1
   20     X1(I)=X1(I-1)
        X1(1)=T

        SUM=0.0
        DO 30 I=1,LX
   30     SUM=SUM + X1(I)*X2(I)
   40   Y(IND)=SUM

      RETURN
      END


      SUBROUTINE ZEROX(X,N)
C     TRUNCATES DATA THAT IS LESS THAN A MINIMUM MAGNITUDE

      DOUBLE COMPLEX X(N)
      DOUBLE PRECISION XMIN,A
      INTEGER N,NZ
```

```
      WRITE(1,*) 'SUBROUTINE ZEROX'

      NZ=0

      WRITE(1,*) 'ENTER MINIMUN MAGNITUDE'
      READ(1,*) XMIN
      DO 10 I=1,N
        A=ABS(X(I))
        IF(A.LT.XMIN) THEN
          X(I)=(0.0,0.0)
          NZ=NZ+1
        ENDIF
        CONTINUE
10    WRITE(1,*) 'NUMBER OF ZEROS : ',NZ
      RETURN
      END


      SUBROUTINE SHIFT(X,N)
C     CIRCULAR SHIFTS A SEQUENCE

      DOUBLE COMPLEX X(N),T(1024)

      WRITE(1,*) 'NUMBER OF SHIFTS TO THE RIGHT ?'
      READ(1,*) NS

      DO 10 I=1,N
        IND=I+NS
        IF (IND.GT.N) IND=IND-N
10      T(IND) = X(I)

      DO 20 I=1,N
20      X(I) = T(I)

      RETURN
      END


      SUBROUTINE NOISE(X,ISIZE)
C     ADDS NOISE TO A SEQUENCE

      DOUBLE COMPLEX X(ISIZE),CM
      DOUBLE PRECISION NMAG,A,ZERO

      WRITE(1,*) 'UNIFORM RANDOM NOISE : ENTER MAX MAGNITUDE'
      READ(1,*) NMAG
      WRITE(1,*) 'ENTER MIN ,MAX OF ARRAY LOCATIONS'
      READ(1,*) N1,N2

      WRITE(1,*) 'ENTER SEED'
      READ(1,*) NSEED
      CALL SSEED(NSEED)
      ZERO=0.0
      DO 10 I=N1,N2
        A=1.0 - 2.0*URAN()
        A=NMAG*A
        CM=DCMPLX(A,ZERO)
10      X(I)=X(I) + CM

      RETURN
      END



      SUBROUTINE SIGNL(X,ISIZE)

C     EXPONENTIALLY DAMPED SINUSOID SIGNAL

      DOUBLE COMPLEX X(ISIZE)
      DOUBLE PRECISION FSAM,FSINE,ALPHA,DT,T,PI2,ZERO

      WRITE(1,*) 'ENTER FSAM,FSINE,ALPHA'
      READ(1,*) FSAM,FSINE,ALPHA

      WRITE(1,*) 'ENTER # OF DATA POINTS,SHIFT'
      READ(1,*) ND,NSHFT

      DT=1.0/FSAM
      PI2=2.0*3.14159265358979D0
```

```
      00=0.0
      DO 10 I=1,ND
        K=I
        T=DCOS( FSINE*PI2*DBLE(K-NSHFT)*DT )
        IA=IABS(K-NSHFT)
        T=DEXP( (-ALPHA)*DBLE(IA)*DT )*T
        X(I)=DCMPLX(T,00)
10      CONTINUE

      RETURN
      END


      SUBROUTINE SEQ1(X,ISIZE)
C     THIS IS A SEQUENCE OF 4 NOS THAT HAS MIXED PHASE ZEROS

      DOUBLE COMPLEX X(ISIZE)

      DO 10 I=5,ISIZE
10      X(I)=(0.0,0.0)
      X(1)=(1.0,0.0)
      X(2)=(-0.25,0.0)
      X(3)=(-3.0,0.0)
      X(4)=(-1.0,0.0)

      RETURN
      END


      SUBROUTINE EXPON(X,ISIZE)
C     EXPONENTIAL PULSE

      DOUBLE COMPLEX X(ISIZE)
      DOUBLE PRECISION A,B,T,00
      WRITE(1,*) 'SUM OF 2 EXPONENTIALS'
      WRITE(1,*) 'ENTER # OF DATA POINTS'
      READ(1,*) N1
      WRITE(1,*) 'ENTER A,B (POSITIVE NOS. A LESS THAN B)'
      READ(1,*) A,B
      A=-A
      B=-B

      00=0.0
      DO 10 I=1,N1
        K=I-1
        T=DEXP(A*K) - DEXP(B*K)
        X(I)= DCMPLX(T,00)
10      CONTINUE
      RETURN
      END


      SUBROUTINE CHIRP(X,ISIZE)
C     CHIRP WAVELET

      DOUBLE COMPLEX X(ISIZE)
      DOUBLE PRECISION A,THETA,T,PI,00
      WRITE(1,*) 'CHIRP DATA'
      WRITE(1,*) 'ENTER # OF DATA POINTS'
      READ(1,*) N1
      WRITE(1,*) 'ENTER PARAMETER A'
      READ(1,*) A

      PI=3.1415926535897900
      00=0.0
      DO 10 I=1,N1
        K=I
        THETA=2.0*PI*A*K*K
        T=DCOS(THETA)
        X(I)=DCMPLX(T,00)
10      CONTINUE
      RETURN
      END
```