NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SIGNAL ENHANCEMENT USING TIME-FREQUENCY BASED DENOISING

by

John B. Hughes

March 2003

Thesis Advisor:	Monique P. Fargues
Second Reader:	Charles W. Therrien

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY	**2. REPORT DATE** March 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Signal Enhancement Using Time-Frequency Based Denoising			**5. FUNDING NUMBERS**
6. AUTHOR(S) John B. Hughes			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

This thesis investigates and compares time and wavelet-domain denoising techniques where received signals contain broadband noise. We consider how time and wavelet-domain denoising schemes and their combinations compare in the mean squared error sense. This work applies Wiener prediction and Median filtering as they do not require any prior signal knowledge. In the wavelet-domain we use soft or hard thresholding on the detail coefficients. In addition, we explore the effect of these wavelet-domain thresholding techniques on the coefficients associated with cycle-spinning and the newly proposed recursive cycle-spinning scheme. Finally, we note that thresholding does not make an attempt to de-noise coefficients that remain after thresholding; therefore we apply time domain techniques to the remaining detail coefficients from the first level of decomposition in an attempt to de-noise them further prior to reconstruction. This thesis applies and compares these techniques using a mean squared error criterion to identify the best performing in a robust test signal environment. We find that soft thresholding with Stein's Unbiased Risk Estimate (SURE) thresholding produces the best mean squared error results in each test case and that the addition of Wiener prediction to the first level of decomposition coefficients leads to a slightly enhanced performance. Finally, we illustrate the effects of denoising algorithms on longer data segments.

14. SUBJECT TERMS Discrete Wavelet Transform, Soft and Hard Thresholding, Cycle-spinning, Recursive Cycle-spinning, Wiener Filtering, Median Filtering, Denoising			**15. NUMBER OF PAGES** 125
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SIGNAL ENHANCEMENT USING TIME-FREQUENCY BASED DENOISING

John B. Hughes
Lieutenant, United States Navy
B.S., Gonzaga University, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2003

Author: John B. Hughes

Approved by: Monique P. Fargues
Thesis Advisor

Charles W. Therrien
Second Reader

John P. Powers
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis investigates and compares time and wavelet-domain denoising techniques where received signals contain broadband noise. We consider how time and wavelet-domain denoising schemes and their combinations compare in the mean squared error sense. This work applies Wiener prediction and Median filtering as they do not require any prior signal knowledge. In the wavelet-domain we use soft or hard threshold on the detail coefficients. In addition, we explore the effect of these wavelet-domain thresholding techniques on the coefficients associated with cycle-spinning and the newly proposed recursive cycle-spinning scheme. Finally, we note that thresholding does not make an attempt to de-noise coefficients that remain after thresholding; therefore we apply time domain techniques to the remaining detail coefficients from the first level of decomposition in an attempt to de-noise them further prior to reconstruction. This thesis applies and compares these techniques using a mean squared error criterion to identify the best performing in a robust test signal environment. We find that soft thresholding with Stein's Unbiased Risk Estimate (SURE) thresholding produces the best mean squared error results in each test case and that the addition of Wiener prediction to the first level of decomposition coefficients leads to a slightly enhanced performance. Finally, we illustrate the effects of denoising algorithms on longer data segments.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I want to thank my Country and the people of the United States of America and my service of choice the US Navy for giving me the opportunity to succeed in finishing a once in a lifetime milestone. I sincerely appreciate the support of the faculty and technical support personnel for giving me the tools necessary to finish this work. I especially thank Professor Monique P. Fargues for her guidance and mentorship during the thesis research and completion process. I also would like to acknowledge Professor Charles W. Therrien for his insight and support with this research.

My teachers through the years have shown me the importance of the human mind and human potential. I want to recognize my high school physics teacher LtCol. Larry Elsom, U.S.M.C (Ret.) for inspiring me to achieve this goal.

I also would like to thank my family and friends for believing in my abilities. My wife Brenda selflessly gave her support and motivation. This work is dedicated to my beloved daughter Taylor, sweet son Matthew, and my tiny daughter Luna who has just recently entered the world and is just now experiencing all the new things life has to offer.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis investigates time and wavelet-domain denoising algorithms in a robust signal environment, and performs a comparative analysis between the various denoising schemes. Wavelet-based denoising has military applications in acoustic signal processing, surveillance and reconnaissance, and various communication applications. This work reviews the basic theory of wavelet denoising leading to the discrete orthogonal wavelet transform. In addition, we present and apply cycle-spinning denoising, which serves to remove the negative effects of translation variance found in the orthogonal wavelet transform. Finally, we investigate recursive cycle-spinning and a combined time-wavelet denoising technique using a windowed Wiener predictor.

Results show that soft thresholding of the translation-invariant wavelet transform coefficients with the Stein's Unbiased Risk Estimate (SURE) threshold outperforms the other schemes considered in Mean Squared Error (MSE) performance for the shorter signal lengths for the signals investigated. For the longer data length experimental signals considered the translation-invariant visual threshold showed the best denoising ability. In addition, no difference could be seen in the performance between orthogonal wavelet and translation-invariant schemes for the long data sets considered. Results further show recursive cycle-spinning using the SURE threshold for smaller data sets and the visual threshold for longer data sets, though not yet optimized, have excellent potential as a denoising alternative. In addition, we found through experimentation that the window size of the Wiener predictor requires *a priori* information for optimal performance.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. THESIS OBJECTIVE

This study considers a frequency-based approach to denoising. The noisy signal is transformed into the frequency domain using wavelet transforms where denoising is conducted. Next the denoised signal is obtained by back-transforming the processed information to the time domain. Further developments in this area will lead to advances in range and sensitivity with regard to acoustic signal processing, surveillance and reconnaissance, and various communications applications.

Donaho and Johnstone [1,2] introduced several methods where information in the wavelet transform domain may be used to de-emphasize the noise-contaminated frequency contributions prior to performing the inverse transform, thereby producing a cleaner system output. This thesis explores these methods and compares resulting performances. The goal of this thesis is to determine which decomposition type and combination of signal processing schemes achieves the cleanest signal output.

Figure 1.1 presents the schematic overview of the concepts considered in this work. It methodically outlines and organizes this work into six chapters from the theory behind the work to the results and conclusions. Chapter II introduces the wavelet transform and illustrates its advantages over Fourier analysis. Chapter III introduces signal and noise elements and the associated assumptions that went into producing robust test signals. Chapter IV first presents various wavelet thresholding techniques. Next, it introduces translation invariance concepts and shows how they are obtained via processes called cycle-spinning and recursive cycle-spinning. Finally, it introduces a windowed Wiener prediction technique and includes a low computational cost scheme called median filtering. Chapter V applies these techniques to test data. Chapter VI presents conclusions and recommendations.

B. THESIS ORGANIZATION

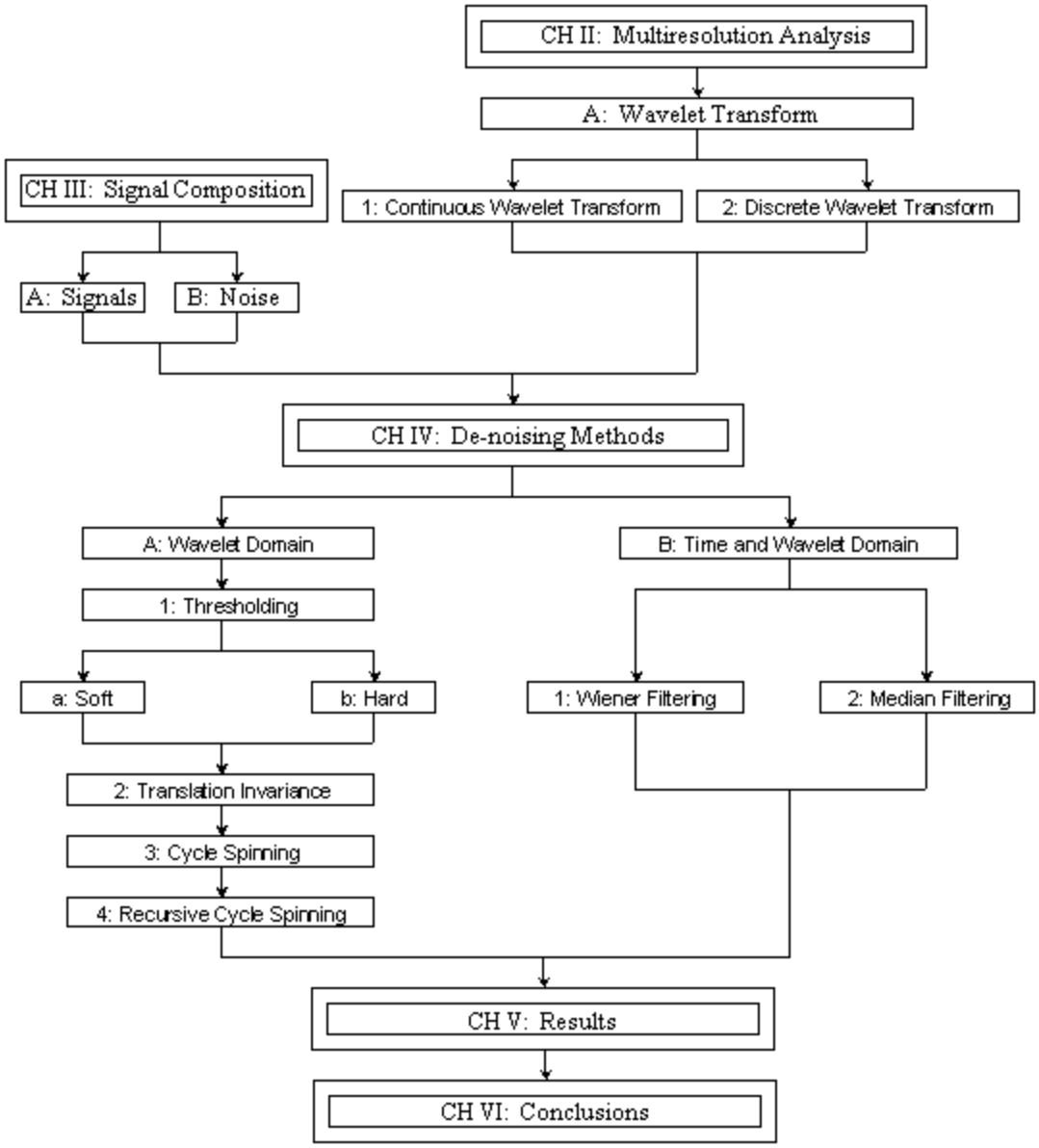


Figure 1.1: Thesis outline flow structure.

II. MULTIRESOLUTION ANALYSIS

A. INTRODUCTION

This chapter presents an overview of Multiresolution Analysis (MRA) as it leads to an important family of signal processing tools. The signal processing applications rooted in MRA have the distinct ability to span both time and frequency, as multiresolution analysis may be viewed as an extension of the Short Time Fourier Transform (STFT).

1. Short Time Fourier Transform

The Fourier Transform is a widely used transformation and is defined for a continuous signal $f(t)$ as:

$$F(w) = \int_{-\infty}^{+\infty} f(t) e^{-jwt} dt. \tag{2.1}$$

The complex basis function, also referred to as a kernel, e^{-jwt} in Equation (2.1) operates on $f(t)$ over all time, creating a one-dimensional representation of the signal in the frequency domain. Note that it is often desirable to localize the specific time at which specific signal characteristics occurred; however, there is no allotted time dimension in the Fourier Transform. Time dependency capability is introduced in the Short Time Fourier Transform (STFT) with a predetermined fixed duration sliding window $g(t-t)$ centered at t. The resulting two-dimensional STFT of the signal $f(t)$ is defined as:

$$F(w,t) = \int_{-\infty}^{+\infty} f(t) g(t-t) e^{-jwt} dt. \tag{2.2}$$

The signal $f(t)$ and the Fourier Transform $F(w)$ are contained entirely in one-dimensional space, whereas the STFT transform $F(w,t)$ is represented in a two-dimensional space. The magnitude squared of the STFT is called the Spectrogram as shown in Figure 2.1, which provides a time-localized frequency content of the transformed signal

$f(t)$. Figure 2.1 illustrates signal presence in red and a lack of signal presence in blue, obtained for the following constant amplitude linear chirp:

$$f(t) = \sin\left(\frac{2p}{F_S} f_o(t)\right), \tag{2.3}$$

where F_S is set to 8000 Hz, and the frequency $f_o(t)$ varies linearly from 1 to 2000 Hz.

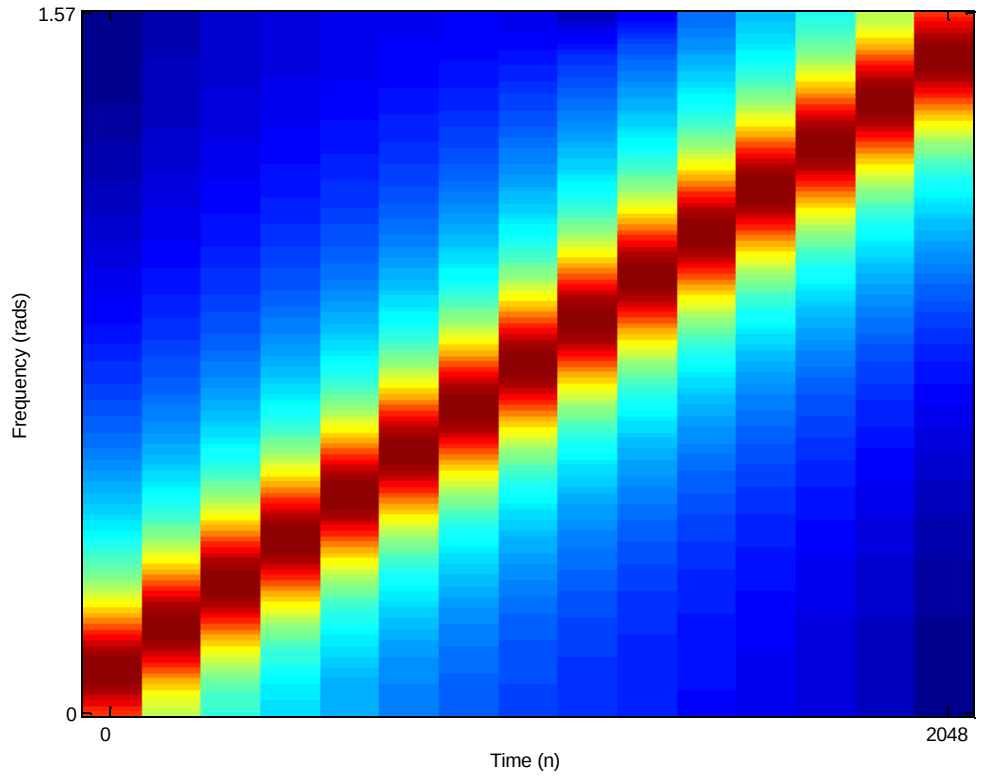


Figure 2.1: Linear chirp spectrogram.

The frequency resolution with a fixed duration sliding window $g(t-t)$ is fixed. In order to get a perfect picture of the signal space from the Spectrogram, it would be desirable to have infinite time and frequency resolution. Unfortunately, the STFT time and frequency resolutions are limited by the Heisenberg Uncertainty Principle [3]:

$$s_t^2 s_f^2 \geq \frac{1}{4}, \tag{2.4}$$

where s_t^2 and s_f^2 represent the temporal variance and frequency variance, respectively. The best case, with equality in Equation (2.4), is found when using a Gaussian window $g(t-t)$ in the STFT. This results in the Gabor Transform [4,5]. For discrete time signals a shortened window leads to fewer samples, and with fewer samples the resultant frequency space will have fewer bins. Hence, intuitively, frequency resolution is lost as time becomes more localized.

Note that time-frequency characteristics may change with time when a signal is transmitted through various channels, or the signal or channel is time-varying. As a result, a small time window is necessary and frequency resolution is degraded when the frequency content of a signal or the channel characteristics change rapidly with time. A larger time window may be sufficient (resulting in better frequency resolution) when the signal frequency characteristics change slowly. For unknown signal characteristics an array of window sizes is required in order to find the ideal time window and frequency resolution combination. Therefore, the fixed window size of the STFT limits its ability to span both time and frequency of unknown signals with resolution well matched to the signal characteristics.

From the above discussion, a transform method that does not utilize a fixed window appears attractive, and Multiresolution Analysis (MRA) techniques do not suffer from such limitations. Next, we discuss MRA techniques, which do not have a fixed window constraint.

2. Multiresolution Theory

In much the same way as the STFT, MRA requires an operator to project the signal $f(t)$ into another domain or vector space. One of the advantages is that the MRA operator does not use a fixed window size.

The Venn Diagram in Figure 2.2 is meant to show that vector space $\mathbf{V_j}$ is a lower resolution approximation of $\mathbf{V_{j+1}}$. Let $\mathbf{W_j}$ represent the loss of information due to the approximation, then the vector sum of $\mathbf{W_j}$ and $\mathbf{V_j}$ fully constitutes $\mathbf{V_{j+1}}$. Further, there is no overlap or repeat of information if $\mathbf{W_j}$ and $\mathbf{V_j}$ are orthogonal. Consider the entire

frequency domain from 0 to p as the vector space $\mathbf{V_{j+1}}$, as shown in Figure 2.3. Then, part of the frequency domain is spanned by $\mathbf{W_j}$ and the other part is spanned by $\mathbf{V_j}$. A natural extension of the above MRA principles is shown as follows [6]:

$$\mathbf{V_{j+1}} = \underbrace{\mathbf{V_k}}_{\text{Appox}} \oplus \underbrace{\mathbf{W_k} \oplus \mathbf{W_{k+1}} \oplus \cdots \oplus \mathbf{W_j}}_{\text{Detail}} \tag{2.5}$$

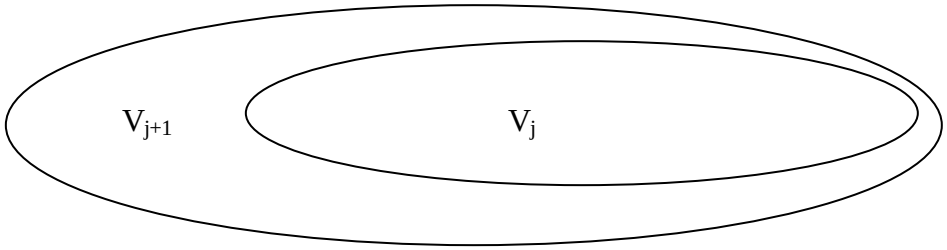


Figure 2.2: Venn diagram illustrating MRA basic principle of successive approximations. After Ref [7].

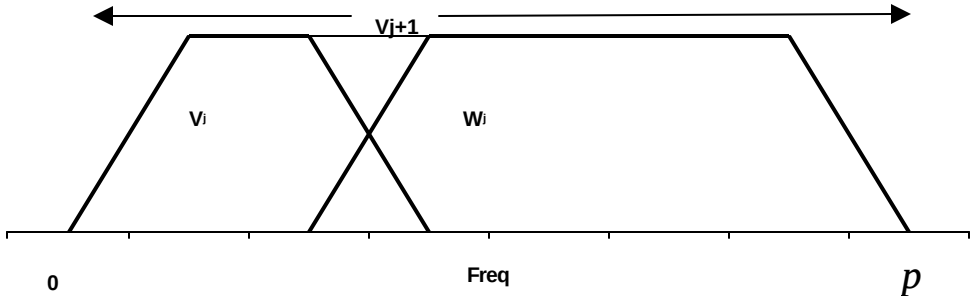


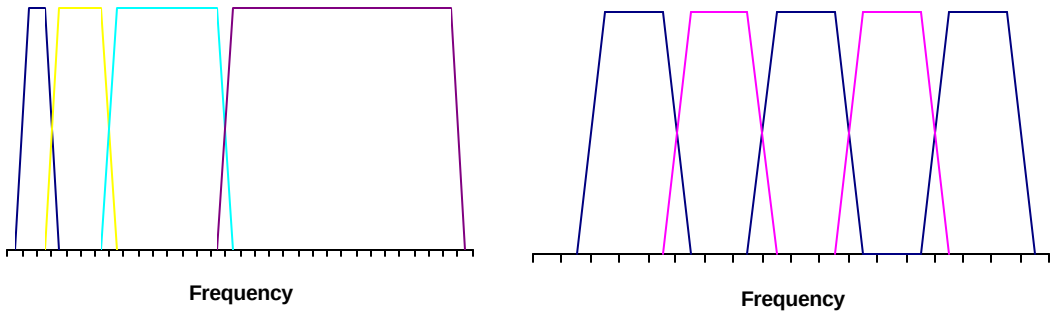
Figure 2.3: Frequency spectrum divided into multi-resolution bands. After Ref [7].

Mallat [3,5] found that MRA could be implemented with minimal redundancy by employing orthonormal filters consisting of a lowpass filter and multiple bandpass filters. The lowpass filter provides the approximation coefficients and the bandpass filters provide the detail coefficients.

Figure 2.4 shows the main difference between MRA and the STFT is in the filter bandwidths. Another promising attribute of MRA is that many different conventional filters can be used to implement it. An additional advantage is that the operator itself can be real, and thus the coefficients will be real resulting in a real-valued transform. Simple

operations such as the application of a Finite-duration impulse response (FIR) filter can be applied to obtain real-valued transform coefficients if needed.

The filters chosen to form the detail and approximation coefficients are called wavelet and scaling functions respectively. These functions will be explored in continuous and discrete time in the following section.



(a) (b)

Figure 2.4: Frequency spectrum partitioning for: (a) MRA; (b) STFT.

B. WAVELET TRANSFORM

Rather than developing a different bandpass filter for each scale, MRA theory allows for a scaled and translated "mother" basis function or wavelet. Wavelet basis functions act in much the same way as the kernel in the Fourier Transform and are orthogonal, normalized to unity, have finite time duration or compact support, and their area sums to zero.

1. Continuous Wavelet Transform

The continuous wavelet transform (CWT) is similar to the Fourier Transform introduced in Equation (2.1). The wavelet is the basis function or operator for the wavelet transform, as e^{-jwt} was the operator for the Fourier Transform.

The equation for the wavelet basis function at scale or resolution a and translation b of the mother wavelet $y(t)$ is defined as [3,8,9]:

$$y_{a,b}(t) = \frac{1}{\sqrt{a}} y\left(\frac{t-b}{a}\right). \tag{2.6}$$

Figure 2.5 (a) shows how the resolution factor a is used to normalize and scale the mother wavelet. Note that the upper, middle, and lower figures in Figure 2.5 (a) illustrate the effect on the wavelet when a is 0, 1, and 2, respectively. This shows how the wavelet is "squeezed" in time as a is increased. Figure 2.5 (b) demonstrates the effect of the translation factor b, where the upper, middle, and lower figures demonstrate the effect on the wavelet when b is 0, 1, and 2, respectively. Hence, the wavelet is translated to the right in time as b is increased. Note above that with the CWT the resolution factor a can be chosen to be any positive real number, and the translation b can be any real number. The CWT coefficients are given by:

$$d_{a,b} = \int_{-\infty}^{\infty} f(t) y^{*}_{a,b}(t) dt. \tag{2.7}$$

Parseval's Theorem states that the energy contained in a signal in the time domain is the same as in another domain if the coefficients are determined by applying a set of orthonormal basis functions $v(t)$ [6], leading to:

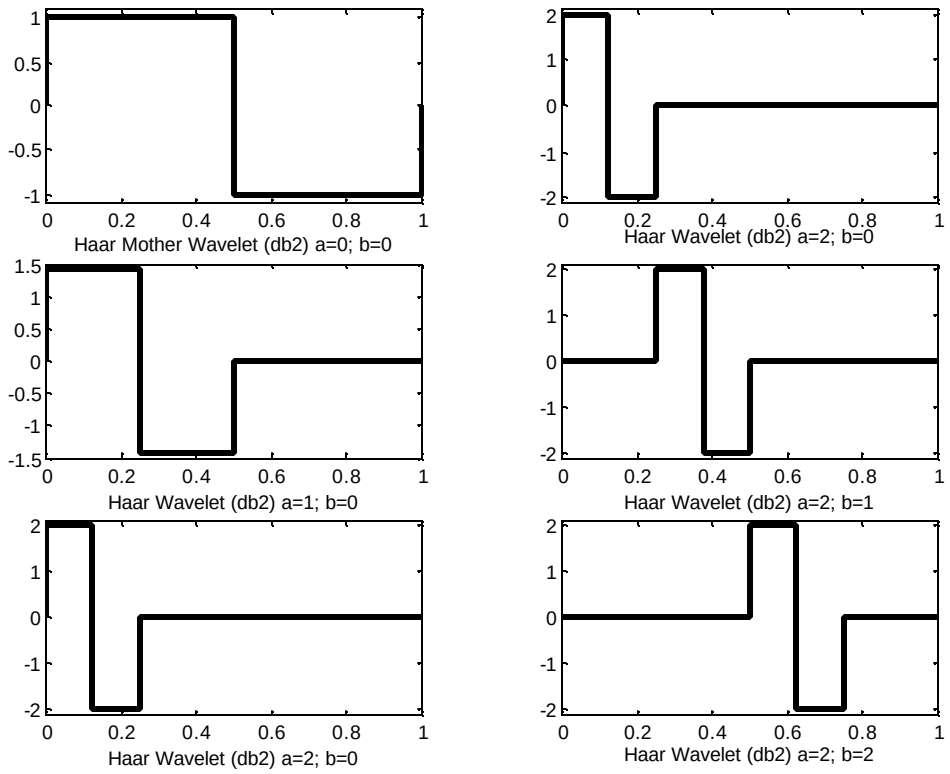
$$\int_{-\infty}^{\infty} f(t) v(t) dt = \frac{1}{2p} \int_{-\infty}^{\infty} F(w) V(w) dw. \tag{2.8}$$

Note that, Parseval's theorem applies both to the orthogonal wavelet and Fourier Transform operations since they both use orthogonal basis sets. Thus, the CWT coefficients may be expressed in terms of the frequency information using Parseval's relationship, which leads to:

$$d_{a,b} = \frac{1}{2p} \int_{-\infty}^{\infty} F(w) \Psi^{*}_{a,b}(w) dw. \tag{2.9}$$

Parseval's relation helps demonstrate that the CWT coefficients are representative of signal energy and larger coefficients correspond to greater signal energy. The wavelet is useful in data compression as well because relatively few coefficients hold

most of the signal energy [1,2,3,5,8]. This attribute is also very valuable to the art of denoising.



(a) (b)

Figure 2.5: Haar wavelet: (a) effect of scaling; (b) effect of translation.

Recall that the STFT has fixed time and frequency resolution as the time window length is fixed, while the CWT does not have such constraints. The CWT time window size gets continuously smaller as the scale increases, while the frequency resolution gets larger.

Wavelets come in different shapes and sizes and can be chosen to meet situational requirements. Figure 2.6 shows examples of popular multipurpose wavelet basis functions. Notice that wavelets have sharper transient characteristics than the sinusoidal based Fourier Transform basis function. Thus, the wavelet is better suited for detecting transient signals, whereas the STFT is ideal for sinusoidal signals.

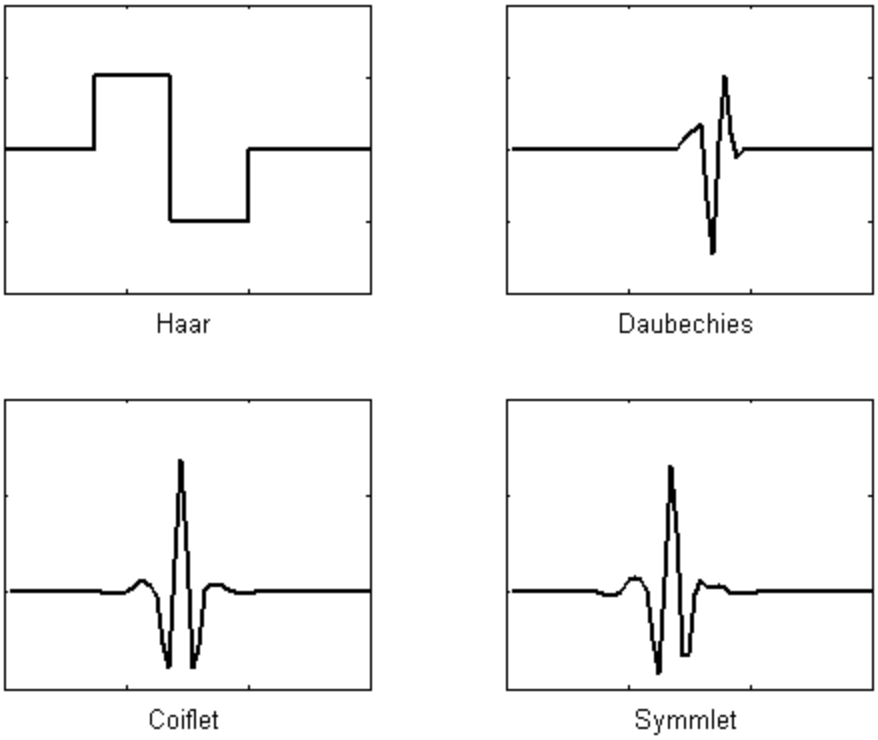


Figure 2.6: Assortment of wavelets. After [10].

The CWT is theoretically significant; however the Discrete Wavelet Transform (DWT) is amenable to fast digital signal processing applications. The DWT takes discrete time input samples and discrete translation and scale factors, and eliminates high processing costs associated with redundancy found in the CWT. The DWT is discussed further in the following section.

2. Discrete Wavelet Transform

The DWT is a clever discretization of the CWT that enables a digital implementation of the CWT. From Equation (2.5) and its accompanied discussion the approximation coefficients and detail coefficients can be written in terms of the scaling function $f(n-k)$ and the wavelet function $y(2^j n-k)$ respectively, as shown below in Equation (2.10). The wavelet function is the highpass filter, while the scaling function is the lowpass filter. Orthogonality between all the basis functions at various scales and

translations prevent redundancy. The fundamental DWT expansion of the discrete signal $f(n)$, where $a = 2^j$ is given by:

$$f(n) = \sum_k c_{0,k} f\left(n-k\right) + \sum_k \sum_{j=0}^{\infty} d_{j,k} 2^{j/2} y\left(2^j n - k\right). \tag{2.10}$$

Utilizing MRA principles, Mallat [3] proposed that within the scope of MRA a filter-bank of highpass and lowpass filters, $g(n)$ and $h(n)$, to make use of coefficients of higher resolution to synthesize the lower resolution coefficients as shown:

$$c_{j+1,k} = \sum_{n=-\infty}^{\infty} h\left(\frac{n}{2} - k\right) c_{j,k}; \;\; d_{j+1,k} = \sum_{n=-\infty}^{\infty} g\left(\frac{n}{2} - k\right) c_{j,k}. \tag{2.11}$$

The following reconstruction equation follows as:

$$c_{j,k} = \sum_{n=-\infty}^{\infty} h\left(k-2n\right) c_{j+1,k} + \sum_{n=-\infty}^{\infty} g\left(k-2n\right) d_{j+1,k}, \tag{2.12}$$

where $h(n)$ and $g(n)$ are so called quadrature mirror filters, which are finite impulse response (FIR) filters that allow for perfect reconstruction [3] as shown in Figure 2.7. Figure 2.7 illustrates a one-step analysis and synthesis process, where c_{j+1} and d_{j+1} are the approximation coefficients and detail coefficients, respectively. The lowpass and highpass filters are represented by *h* and *g*, respectively, and $\downarrow 2$ represents the decimation by two operation. Mallat's lowpass and highpass filters are simply the scaling basis function and wavelet basis function found in Equation (2.10). This decomposition can be expanded into a multi-scale filter bank structure, as shown in Figure 2.8, where *N* refers to the decomposition level and "WL" indicates the Wavelet analysis operation. Figure 2.8 demonstrates the approximation coefficients from the previous scale are used to form the detail and approximation coefficients of the following scale. Note that the number of scales is not limited to three as shown in Figure 2.8, but is limited by the length of the data segment. The result is that the frequency spectrum is sequentially cut in half as shown in Figure 2.4 (a). The highest resolution coefficients labeled "Detail *N* = 1" in Figure 2.8 represent the highpass filtering signal contribution. Then, the next resolution takes the remaining lower half of the spectrum and splits it in two. By using the factor of 2, Mallat formed what are commonly referred to as *dyadic*

scales. Hence, orthonormal bandpass filters are formed via Mallat's theorem using only lowpass and highpass filtering operations. From Equation (2.11) the approximation coefficients of the previous scale are used to form the approximation and detail coefficients of the next higher scale, and the process continues until the desired scale is reached.

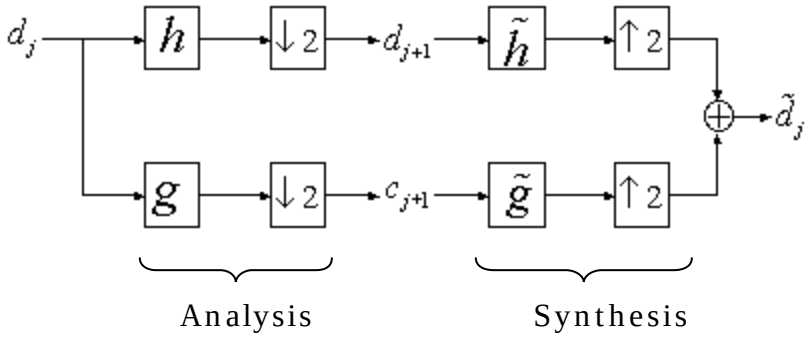


Figure 2.7: Mallat's filter bank wavelet analysis and synthesis. After [3].

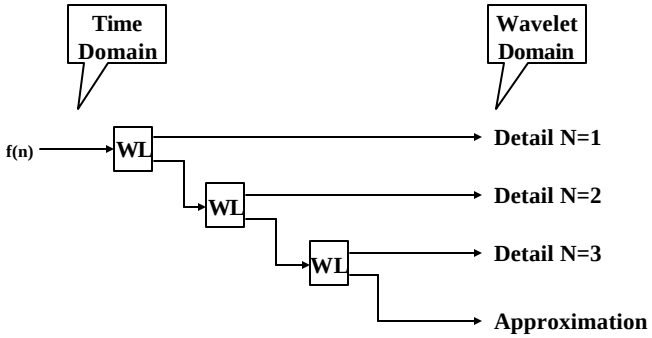


Figure 2.8: Multi-level wavelet decomposition.

Similar to the STFT, the resulting DWT coefficients can be represented in two-dimensional time and frequency. The resulting time-scale visualization is called the scalogram.

Figure 2.9 presents the scalogram of the chirp from Equation (2.3), to be visually compared to Figure 2.1. Decomposition level one shows the presence of high frequency

during the second half of the signal. The resolution of high frequency components was chosen to be low by convention since it is assumed that the majority of the observed signals reside in the lower half of the spectrum. For the case where most of the signal lies in the upper half of the spectrum, a different multiresolution technique can be employed where the higher frequencies have highest resolution. In this work it is assumed that, due to fast processing and sampling speeds, most observed signals will lie in the lower half of the spectrum. Hence, Mallat's multi-level filter decomposition as discussed above is applied as the method of choice throughout this work.

The sinusoidal based signal such as the chirp in Equation (2.3) is more easily interpreted as a constant amplitude linear chirp by analyzing the spectrogram. Hence, the spectrogram and scalogram are often used in concert to correctly identify the characteristics of a broad class of signals.

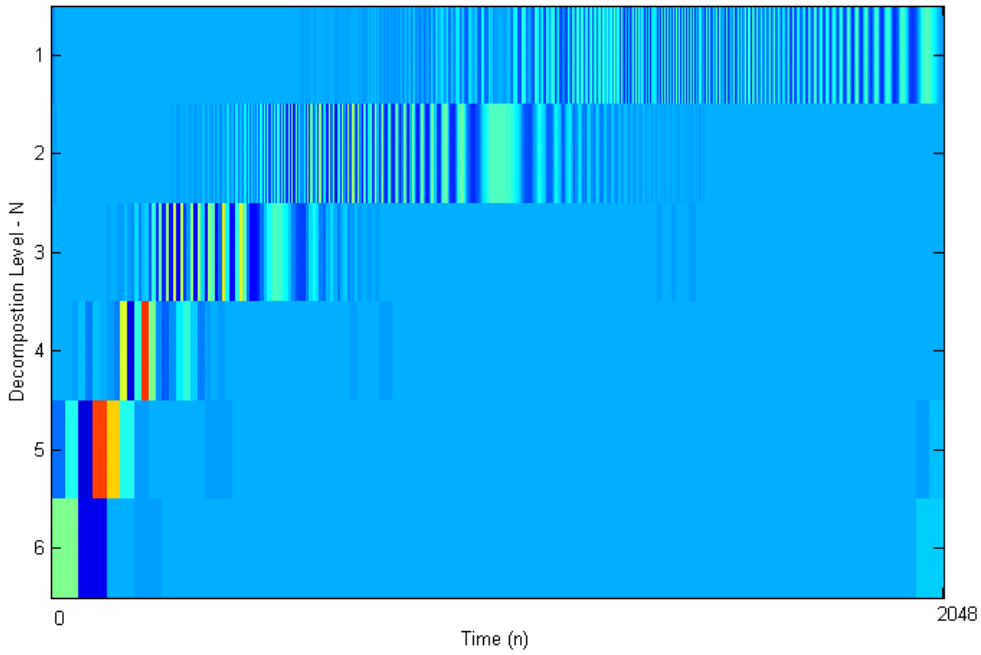


Figure 2.9: Linear chirp Scalogram.

The previous chapter introduced Mallat's theory and the use of MRA as a filter bank structure. The filter bank splits the signal spectrum into *dyadic* resolutions with the highest resolution representing the upper half of the spectrum. When a signal is present,

signal energy is found in one or more of the frequency scales. Noise energy can be present throughout each of the *dyadic* frequency scales. The following chapter introduces signal processing techniques developed to clean up noisy information.

III. SIGNAL COMPOSITION

A variety of modulation formats are used in modern communications, which leads to a large variety of transmitted and received signals. Although signals come in various time-varying shapes, sizes, frequencies, and phases, signal processing techniques have been developed to digitally format signals for lossless reconstruction.

A. SIGNALS

Deterministic functions convey information, which can be transmitted in the form of a signal using the electromagnetic spectrum. Analog or digital signals are transmitted from one point in space to another with the intention of being received and analyzed at the receiver. The formation of a signal with the least redundancy is based on the functions' required bandwidth as defined by the sampling theorem [6,9].

In order to form a discrete signal from a continuous analog signal, the analog signal is sampled at the sampling interval T by multiplying by an impulse train:

$$f_{sampled}(t) = f(t)\sum_{n=-\infty}^{\infty} d(t - nT). \tag{3.1}$$

The sampling interval T is chosen small enough so the sampled data can be used to reconstruct the original signal. Intuitively, the fewer data points available and the higher the signal's frequency the more difficult it will be to estimate the original signal.

Transforming Equation (3.1) into the frequency domain by noting that multiplication in the time domain is equivalent to convolution in the frequency domain leads to the following expression:

$$F_{sampled}(w) = \left[\frac{1}{2p}F(w)\right] \otimes \left[\frac{2p}{T}\sum_{n=-\infty}^{\infty} d\left(w - \frac{2p\,n}{T}\right)\right], \tag{3.2}$$

which further evaluates as follows:

$$F_{sampled}(w) = \frac{1}{T}\sum_{n=-\infty}^{\infty} F\left(w - \frac{2p\,n}{T}\right). \tag{3.3}$$

The sampled spectrum found in Equation (3.3) contains $F(w)$ with periodic identical replicas of itself. The period, which is related to the sampling rate $F_S = 1/T$, is important because if it is too small then overlap and aliasing will occur. Figure 3.1 (a) shows the case where the sampling frequency is large enough, and the original $F(w)$ can be fully recovered with a lowpass filter. Figure 3.1 (b) illustrates an undesirable case where the original signal cannot be recovered due to the aliasing. Notice that the original signal can be recovered as in Figure 3.1 (a) when the sampling frequency $w_{Sampled} = 2p/T$ is twice the signal bandwidth. This special frequency is called the Nyquist frequency and results in the minimum safe sampling rate for real signals [11,12].

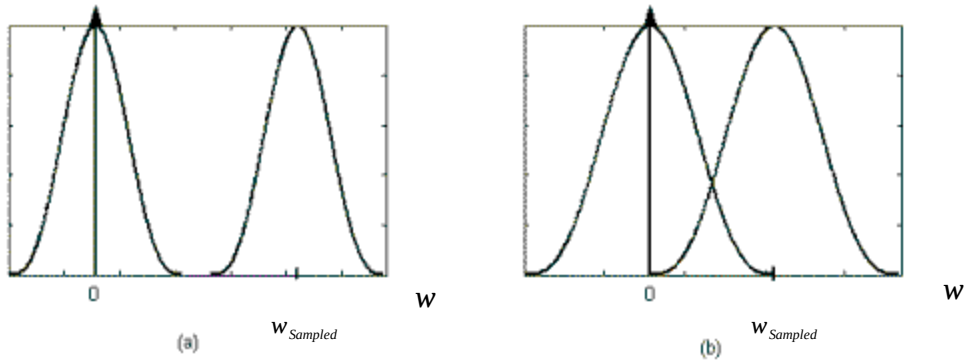


Figure 3.1: Sampled frequency spectrum: (a) Sampling frequency above Nyquist frequency; (b) Sampling frequency below Nyquist frequency.

In this work suitable signals are constructed using the sampling theorem in order to test various denoising algorithms. The following section describes the method used in this work to form a sine wave with frequency set to a uniform random variable between 0 and p radians.

Equation (2.3) is used with frequency $f_o(n)$ as a uniform random variable between 0 and 4000 Hz. Hence, with a sampling frequency set to 8000 Hz we meet the requirements of the Nyquist rate. The input signal for each sinusoidal simulation was therefore a sine wave with frequency set to a value between 0 and p radians for the duration of the simulation. Note that the chirp from Equation (2.3) also met the requirements of the Nyquist Rate; however it only covered frequencies up to $p/2$ radians.

B. NOISE

1. Introduction

Received signals contain noise over the entire received spectrum, while the desired signals may be contained in a small dynamic region of time and frequency. Filters are used to lower the signal-to-noise ratio (SNR) by eliminating noise from the parts of the spectrum not occupied by signal energy. Throughout this thesis it is assumed the signal channel has Additive White Gaussian Noise (AWGN) with a variance equal to one. This is not a severe restriction since pre-whitening filters can be employed when the noise is colored [8,13].

2. Noise Power Estimation

When noise power is not known *a priori*, then it must be determined using noise power estimation techniques. Once the noise power is determined, then the data can be normalized. Two techniques, a time-domain technique, and a wavelet-domain technique are discussed in this section.

The time-domain technique relies on determining a section of the data with relatively signal free characteristics. This section of data, which contains predominately noise energy is used to form an estimate of the noise power for the data. Once an estimate of the noise power is determined it is normalized to unity, by dividing the entire signal by the square root of the noise power. Then the resulting signal can be passed as a whole or in smaller subsections through the proposed denoising algorithm.

The second method used to measure the noise power utilizes knowledge about the wavelet coefficients. Mallat [3: p. 459] shows that any AWGN at the input to the wavelet transform is transformed "at the finest scale" to AWGN of the same variance. If it can be assumed the signal lies predominantly in the lower half of the signal spectrum, then the detail coefficients at the first level of decomposition will be primarily AWGN. Consequently, measuring the variance of these coefficients and normalizing the signal as illustrated above also leads to noise normalized data.

In either case, the result is a signal combined with AWGN with a variance of one. The thesis flow diagram of Figure 1.1 shows that the next step is to run the signal though

various wavelet and time-based denoising processes. The following chapter introduces wavelet-based denoising concepts.

IV. DENOISING METHODS

We now have the tools necessary to delve into several areas of denoising. This work focuses primarily on (1) thresholding as a means of denoising in the wavelet domain due to some important characteristics present in wavelet coefficients; and (2) denoising using the minimum Mean Squared Error (MSE) technique of Wiener Filtering in the time domain. We also introduce a hybrid denoising method. Additionally, translation-invariant denoising is explored using the same techniques as those adopted in the orthogonal wavelet domain.

A. WAVELET DOMAIN

The noisy input signal can be thought of as the sum of the desired signal component (or true signal) and the Additive White Gaussian Noise (AWGN) with variance of one:

$$x(n) = \underbrace{s(n)}_{\text{desired signal}} + \underbrace{n(n)}_{\text{noise component}}. \tag{4.1}$$

It has been shown that when the wavelet basis selected is well matched to the signal characteristics, very few of the wavelet detail coefficients are influenced by the signal, while most of them are influenced by the noise. Therefore, an expression for the wavelet coefficients at each decomposition level can be described by:

$$y_j(i) = \underbrace{w_j(i)}_{\text{desired coefficients}} + \underbrace{n_j(i)}_{\text{noise component}}. \tag{4.2}$$

In addition, the desired signal coefficients are expected to be of larger magnitude when the SNR is not too small. Therefore, denoising is accomplished by thresholding wavelet coefficients, thereby eliminating noise-only coefficients and keeping the desired signal coefficients for reconstruction.

Upon reconstruction, the MSE normalized by the signal power is applied in the following manner for determining how closely our de-noised signal compares to the original noiseless signal:

$$MSE_{normalized} = \frac{E[(s(n) - \hat{s}(n))^2]}{E[s(n)^2]}, \tag{4.3}$$

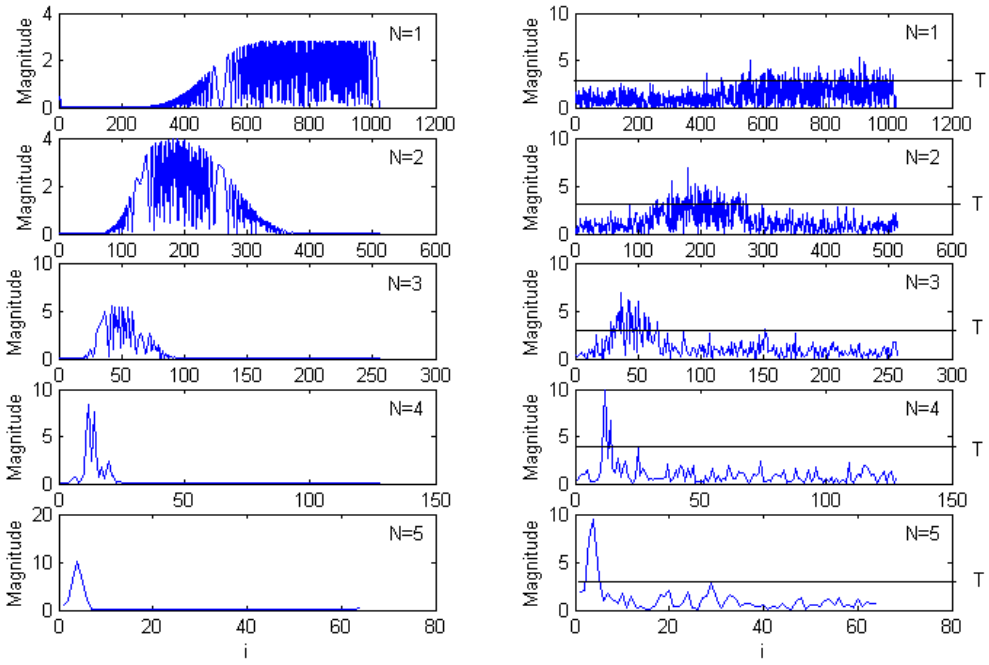
where $E[\bullet]$ is the expectation operator, and $\hat{s}(n)$ is the de-noised signal. A visual comparison can be made between $x(n)$ and $\hat{s}(n)$ when the original signal $s(n)$ is not available. The rest of the chapter defines and illustrates the various wavelet thresholding techniques currently available in the literature.

1. Thresholding

Thresholding is an important concept in denoising and compression, because as previously stated, a few detail coefficients hold the signal information when the wavelet basis selected is well matched to signal characteristics, while the effect of AWGN on the signal is the same over all the coefficients at each scale. Note that the approximation coefficients that do not contain signal energy often do not reside at or near zero, as do their parent detail coefficients. Hence, thresholding schemes will be limited to detail coefficients.

The goal of this section is to define threshold levels that can be used to "kill" detail coefficients at each decomposition level that are likely to contain noise energy [1,2]. To illustrate the above properties and understand their usefulness, it is helpful to compare and contrast the coefficients $w_j(i)$ obtained with no additive noise and the coefficients $y_j(i)$ obtained with noise. Figures 4.1 (a) and (b) provide the detail coefficients from top to bottom starting with the first level of decomposition. Figure 4.1 (a) are the coefficients used to create the scalogram in Figure 2.9. Note the similarities between the two figures. Figure 4.1 (a) provides a plot of the magnitude of the desired detail coefficients $w_j(i)$ for a linear chirp, and Figure 4.1 (b) shows the noisy coefficients $y_j(i)$. Notice that the first level of decomposition coefficients in Figure 4.1 (a) are zero from $n = 0$ to 250, hence the thresholded output of any good thresholding scheme will be close to zero in that region. A horizontal line is drawn as shown in Figure 4.1 (b) and labeled *T* for threshold across the top of the highest coefficient at each level of decomposition produced only by noise. The height of the line would be the maximum necessary threshold to remove all

the noise present in noise-only contributed coefficients. Note that noise is not removed from signal plus noise coefficients. This maximum threshold is called the Universal or Visual threshold and will be described in more detail in the following section.



(a) (b)

Figure 4.1: Magnitude of detail coefficients at level N from chirp in Equation (2.3) with signal to noise ratio of 6 dB. (a) True noiseless coefficients; (b) Noisy coefficients with Threshold = T.

a. Universal Threshold

The visual threshold gets its name from the relatively noiseless feature of $\hat{s}(n)$ because it guarantees the removal of all noise-only coefficients. However, since this thresholding method "kills" the greatest number of coefficients, it also may mistakenly eliminate the most signal. Therefore, this threshold involves the greatest risk of loosing signal-containing coefficients. (The term risk is used synonymously with MSE in some texts.)

The probability that the universal threshold is greater than the magnitude of every noise-only coefficient is equal to one. Recall we have selected the noise to be AWGN with zero-mean and a variance one; thus the wavelet coefficients distribution

function is Gaussian. Note that only a small percentage of coefficients will have large magnitudes. Hence, a small percentage of the noise may be mistaken for signal coefficients, provided the interference is constructive, if the threshold is not sufficiently high. Similarly, a coefficient containing signal energy may be mistaken for noise when destructive interference is encountered.

As the sample size increases, the probability or likelihood of encountering larger magnitude coefficients increases. Hence, a larger threshold is required to ensure the elimination of noise-only coefficients, as n becomes larger so that:

$$\Pr\left(\max\left|n_j(i)\right| \le T\right) = 1, \tag{4.4}$$

where for a Gaussian distribution the threshold T is given by:

$$T = s\sqrt{2\ln(n)}. \tag{4.5}$$

Note that the above threshold value represents the upper bound needed. Since the threshold value does not level off unless large data sizes are used, as shown in Figure 4.2, it is not recommended for data sets involving less than several thousand samples. Finally, note that each successive level of detail coefficients contains half the number of samples found at the previous level. Therefore, while the universal threshold is effective for the first level of decomposition, there may not be enough samples in the next for it to be applied. This property limits the number of decomposition levels when solely applying the universal threshold.

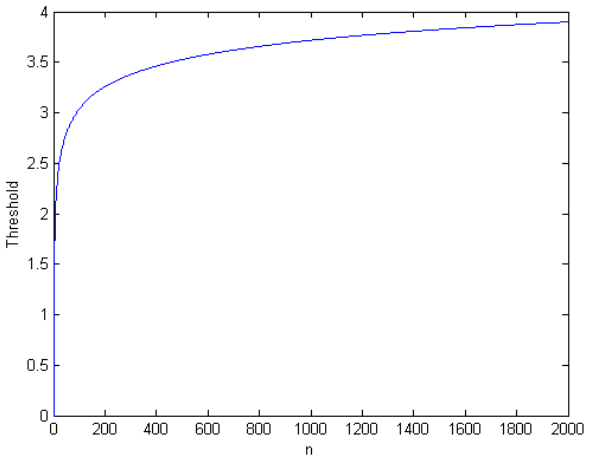


Figure 4.2: Illustration of universal threshold calculation of Threshold T as a function of n.

b. SURE Threshold

The previous threshold can be thought of as the threshold that produces the maximum risk, whereas the Stein's Unbiased Risk Estimate (SURE) threshold results in the minimum risk. These two thresholding techniques can be thought of as the upper and lower bounds in thresholding.

Donoho and Johnstone [1,2] first proposed the SURE algorithm as a result of the work done by Stein on the unbiased risk estimate of multivariate normal estimation problems. They determined that the detail coefficients from each level of decomposition could be modeled using independent multivariate estimation problems. At each scale, through statistical analysis of the coefficients, a threshold is adaptively selected based on the quantity of signal present in the coefficients. By applying SURE [2] over a range of thresholds, the minimum risk is determined.

The SURE algorithm is only effective for SNR's greater than 0 dB because it relies heavily on statistical information obtained from the desired coefficients $w_j(i)$. Hence, another thresholding technique such as the universal threshold should be adopted if a sparse signal condition is found. The Hybrid threshold simply makes a choice between either the universal Threshold or the SURE threshold based on which threshold is smaller.

c. Bayesian Thresholding

Assuming that we can estimate the distribution function of the signal coefficients, a relatively new thresholding technique can be applied. This technique of Bayesian thresholding is not used in this work because no *a priori* signal information is assumed. However, Bayesian thresholding has been shown to outperform both universal and SURE thresholding in image denoising, since natural images often can be modeled with a heavy tailed Gaussian distribution [15,16].

d. Hard and Soft Thresholding

The first step in the denoising process was to obtain the wavelet transform of the signal $x(n)$ using a suitable basis function. Then, a threshold is obtained using one of the above thresholding methods. Once an appropriate threshold is determined we must decide how to apply it. This work discusses the hard and soft thresholding techniques as they pertain to a given threshold.

Hard thresholding zeroes out, or shrinks [1,2], the coefficients that have magnitudes below the threshold, and leaves the rest of the coefficients unchanged [5]:

$$\hat{d}_j(i) = \begin{cases} d_j(i) & \left|d_j(i)\right| > T \\ 0 & \left|d_j(i)\right| \le T. \end{cases} \tag{4.6}$$

Soft thresholding extends hard thresholding by shrinking the magnitude of the remaining coefficients by T, producing a smooth rather than abrupt transition to zero [5,14]:

$$\hat{d}_j(i) = \begin{cases} \text{sign}[d_j(i)](|\, d_j(i)\,| - T) & |\, d_j(i)\,| > T \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

The smooth transition to zero results in noticeably fewer artifacts upon reconstruction, especially when dealing with image denoising [14]. Hence, soft thresholding is generally better for denoising due to its inherent smoothing, whereas hard thresholding is better suited for data compression.

In either case, perfect reconstruction is not possible since some of the signal components are thrown away with the undesired noise. Furthermore, any thresholding technique other than the universal threshold will preserve some of the noise-only coefficients.

A drawback to thresholding is that noise affecting the remaining signal coefficients is not removed. This work explores algorithms that attempt to clean the remaining coefficients prior to reconstruction. Finally, note that the orthogonal wavelet transform is not translation-invariant, i.e., the wavelet coefficients change when the signal is transformed using different time shifts. The next section introduces this important concept as an extension of the orthogonal wavelet transform denoising.

2. Translation Invariance

Translation-invariant denoising is a term coined by Donoho and Coifman [17,18], which illustrates a wavelet-based denoising method that attempts to remove the negative attributes associated with individual translations. There are two important properties associated with orthonormal wavelet coefficients that lead to further developments of wavelet denoising theory. First, wavelet coefficients represent a projection of a signal into its signal subspace. Second, they are translation variant in that, for any right or left shifted input signal $x(n-t)$, one can expect a slightly different projection into signal subspace. It is the alignment or translation combined with the effect of additive noise that prevents the orthonormal wavelet from obtaining the true signal subspace.

Any one projection will contain artifacts or spurious oscillations caused by the alignment of a discontinuity in the signal and the wavelet. This is called the pseudo-Gibbs phenomena [8,17]. This phenomenon depends on the signal and basis function and is localized to a small percentage of wavelet coefficients. An advantage of denoising using the wavelet transform vs. the Fourier transform is that the Gibbs phenomenon is distributed over all the Fourier domain coefficients whereas it is localized to a few coefficients in the wavelet domain [17]. Intuitively, we can expect an average of resultant signal subspace projections to be a closer approximation to the true projection. Additionally, depending on alignment, the discontinuities due to singularities may be

sparse or frequent, hence an average tends to reduce the magnitude of the artifacts overall.

Since each similar projection contains redundant information this could be thought of as a form of diversity. As with diversity, for the same bit rate, the more diversity we have, the slower the information flow. Since it is desired to optimize the process, an efficient systematic process called cycle-spinning was introduced by Donoho and Coifman [17,18].

3. Cycle-Spinning

Denoising with the addition of cycle-spinning utilizes the orthonormal wavelet transform, consequently no new transformation is required. The process of cycle-spinning simply involves right shifting and left shifting by a preset amount prior to the next level of decomposition, and shifting back and averaging prior to the next level of reconstruction. Figure 4.3 (a) illustrates the true output coefficients at each level of decomposition of a constant amplitude linear chirp as a result of cycle-spinning. Notice that each decomposition level has redundant information in the form of redundant signal subspace projections. The first decomposition level consists of the 2048 coefficients from 6145 to 8192 and contains only two projections, each of which consist of 1024 coefficients. Note the first level detail coefficients of the orthonormal wavelet transform consist of 1024 coefficients. The projections are a result of the first step of cycle-spinning, which separately passes right and left shifted versions or translations of the noise free signal through the orthonormal wavelet transform. This process results in two upper projections or detail projections and two lower projections or approximation projections. Through a recursive process, each lower projection results in two upper and lower projections in the next level of decomposition. The decomposition levels proceed from right to left in Figure 4.3 such that the second decomposition level from 4097 to 6144 consists of four detail projections. Note that in Figures 4.3 (b) through (d) the approximation coefficients from 1 to 2048 are not thresholded, as is the case with orthonormal thresholding.

Figure 4.3 (b) shows the result of passing a noisy signal through the translation-invariant wavelet transform. Recall that each projection is a different approximation of

the true signal subspace. Therefore, the denoising process is applied independently to each projection prior to averaging and reconstruction. Figures 4.3 (c) and (d) demonstrate the outcome of soft thresholding with the visual threshold and hybrid threshold respectfully. Note that since the hybrid thresholding algorithm found significant evidence of signal presence it defaulted to the SURE method of thresholding rather than to the visual threshold.

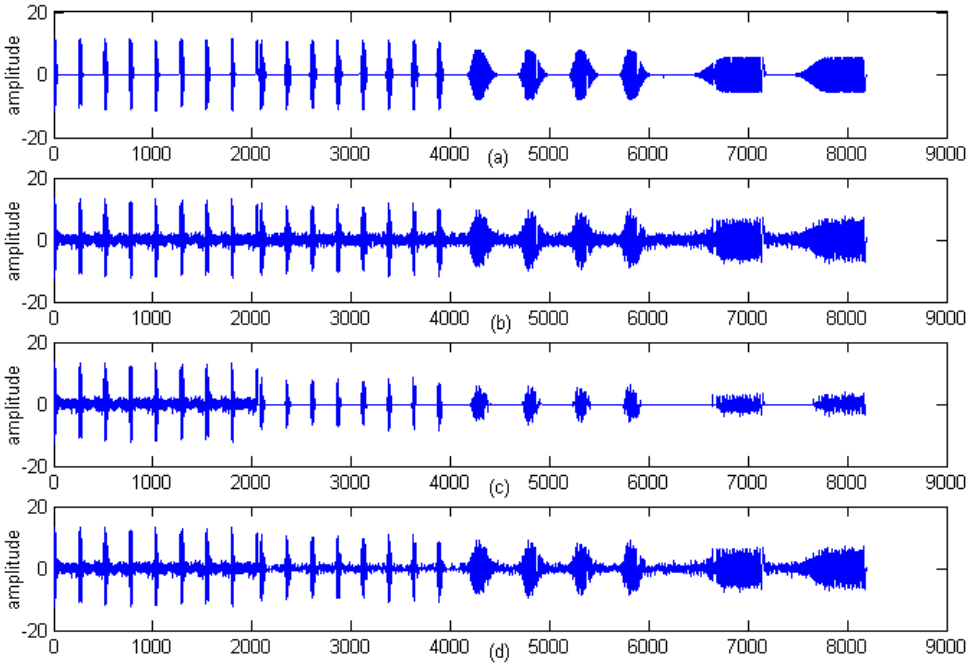


Figure 4.3: Translation-invariant denoising of constant amplitude linear chirp with SNR = 9 dB and $N = 3$. (a) True wavelet coefficients; (b) wavelet coefficients in AWGN; (c) wavelet coefficients de-noised using soft visual thresholding; (d) wavelet Coefficients de-noised using soft SURE thresholding.

In order to enhance the denoising process further, we rely on the convergence property of subspace projections [18]. This property suggests that the projection into a subspace becomes closer or converges toward the true projection each time it is recursively passed through the cycle-spinning process. This concept is explored further in the following section on recursive cycle-spinning [18].

4. Recursive Cycle-Spinning

Figure 4.4 illustrates the process of passing the output of the translation-invariant denoising algorithm through the identical denoising process i times. This process can be performed recursively, hence the term recursive cycle-spinning

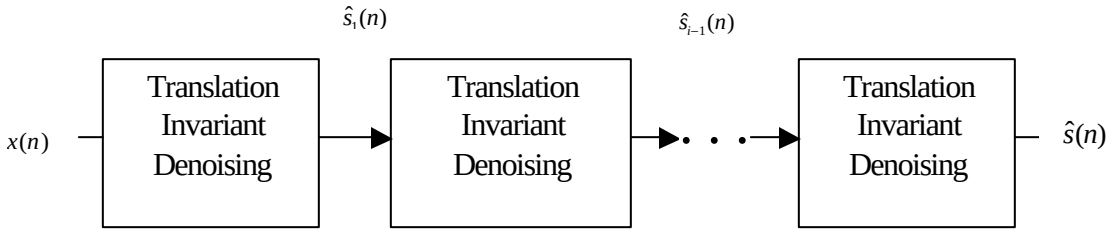


Figure 4.4: Illustration of denoising using multiple passes.

The results of recursive cycle-spinning demonstrate that multiple passes result in a projection, which converges toward the true projection [18]. The threshold and thresholding method will play a large part in the ability of the algorithm to converge to the true projection, however the number of passes at which the recursive cycle-spinning converges to the true subspace depends on the level of the threshold because of inherent losses the signal undergoes in the thresholding process. Absolute convergence to the true subspace is not possible because of the loss of smaller signal components through the course of thresholding. For a properly chosen threshold, each recursive iteration retains relatively large signal energy; however a large portion of noise is removed. The projections converge toward the signal subspace, and at the same time away from the noise subspace at the end of each recursive iteration. A point will be reached where the removal of noise at each level of decomposition is no longer the dominating factor. At this point we are close in the mean squared sense to the true projection and any additional iterations may prove counterproductive. Hence, making the threshold smaller serves to increase the number of productive recursive iterations. The "optimum" threshold will remove the most noise with the lowest information loss. Note that this work does not attempt to find the optimum threshold or the number of iterations necessary to reach the minimum MSE. We take an intuitive approach to picking a suitable threshold and sufficient number of iterations to illustrate the usefulness of the technique.

Although hard thresholding preserves the most signal energy remaining in the coefficients prior to reconstruction, it provides less than a smooth final result. Therefore, we choose to apply ten recursive iterations using hard thresholding followed by an additional iteration using soft thresholding. This method keeps more of the signal energy from iteration to iteration, while providing a smoother result.

Intuitively, the threshold that requires the least fewest number of iterations is the universal threshold since it is the largest threshold. Further, since the universal threshold is guaranteed to remove all the noise on the first iteration, any further iterations may prove counterproductive due to the added risk of signal loss from the higher threshold.

Using the same argument as above, the threshold that requires the largest number of iterations is the SURE threshold because it is the smallest threshold and removes the least noise and signal alike. Recall that upon reaching a certain point, any additional iteration will cause signal degradation as a result of the additional loss of signal energy. This effect worsens with higher thresholds, since they impose a higher risk. Therefore, to prevent overshooting convergence, SURE thresholding is adopted as a method of choice for this work.

The convergence ability of the algorithm is pointed out with Figures 4.5 (a) through (d) by comparing two possible recursive cycle-spinning arrangements to the noiseless and noisy projection. One would expect that as long as there is no overshoot, the coefficients in the case where the most iterations are performed will more closely represent the noiseless coefficients seen in Figure 4.5 (a). Figure 4.5 (c), which represents denoising using two iterations of hard thresholding followed by one iteration of soft thresholding with the SURE threshold appears to be further from the true projection than our recursive results in Figure 4.5 (d). Notice that the lowest decomposition level detail coefficients of Figure 4.5 (d) are most similar to their counter parts of Figure 4.5 (a).

The convergence of the upper level coefficients therefore appears to lag the lower coefficients in the convergence process. Hence convergence overshoot may occur first at the lowest level of decomposition and produce signal degradation of low frequency signal components. We leave this determination for future study.

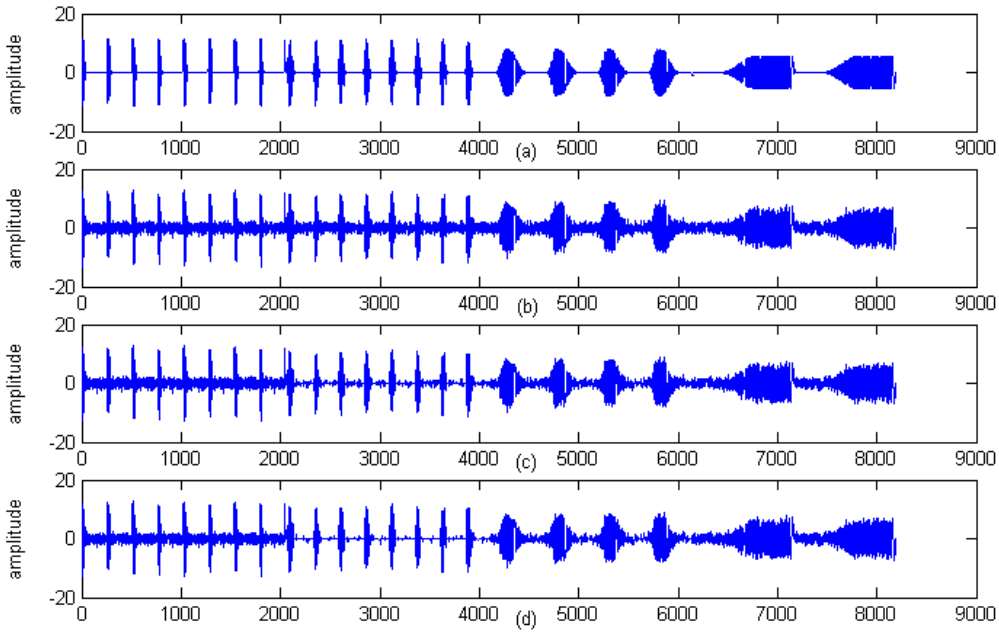


Figure 4.5: Constant amplitude linear chirp in translation-invariant wavelet domain prior to last reconstruction: (a) True noiseless coefficients; (b) Coefficients in AWGN with SNR = 9 dB; (c) Recursive cycle-spinning with two hard iterations followed by one soft iteration using SURE threshold; (d) Recursive cycle-spinning with ten hard iterations followed by one soft iteration using SURE threshold.

This method will be compared to other denoising schemes using the MSE. Additional clean up of the remaining coefficients will also be attempted using the time domain techniques discussed in the next section.

B. TIME AND WAVELET DOMAIN

Time-domain filtering techniques such as Wiener Filtering have been adopted to minimize the mean squared error of the filtered signal. These proven time-domain techniques will be compared to wavelet-domain thresholding schemes, described previously in this chapter. In addition to direct comparison, we consider a combination of both wavelet and time domain techniques to further improve the denoising process.

The following section describes Wiener filtering principles. The Wiener filter itself is not used since we would require *a priori* knowledge of the signal or an accurate calculation of AWGN variance. Although this thesis makes the assumption that the noise

statistics are known or can be determined, we select Wiener prediction as a useful signal processing alternative.

1. Wiener Filtering

Wiener filtering is well-known in the literature [13]. When the observation is of the form:

$$x(n) = s(n) + h(n), \tag{4.8}$$

where the signal $s(n)$ and noise $h(n)$ are independent, the equations for the optimal filter are of the form:

$$\begin{bmatrix} R_x(0) & R_x(1) & \cdots & R_x(P) \\ R_x(1) & R_x(0) & \cdots & R_x(P-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(P) & R_x(P-1) & \cdots & R_x(0) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(P) \end{bmatrix} = \begin{bmatrix} R_s(0) \\ R_s(1) \\ \vdots \\ R_s(P) \end{bmatrix}, \tag{4.9}$$

where $R_x(l) = R_s(l) + R_h(l)$, and $R_s(l)$ and $R_h(l)$ are the autocorrelation function for the signal and noise, respectively. These are the Wiener-Hopf equations.

In order to predict the signal $\left(\text{i.e., estimate } s(n+1) \text{ instead of } s(n)\right)$, the Wiener-Hopf equations change slightly:

$$\begin{bmatrix} R_x(0) & R_x(1) & \cdots & R_x(P) \\ R_x(1) & R_x(0) & \cdots & R_x(P-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(P) & R_x(P-1) & \cdots & R_x(0) \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(P) \end{bmatrix} = \begin{bmatrix} R_s(1) \\ R_s(2) \\ \vdots \\ R_s(P+1) \end{bmatrix}. \tag{4.10}$$

a. Elimination of Predictor Edge Distortion

The adopted method consists of flipping the data from front to back, convolving with the same prediction filter from above, and then flipping back again. Hence, two representations of the data signal have been determined where one is forward predicted and the other reverse predicted. The last two accurate samples from the reverse prediction are put in their respective place in the forward predicted data as shown in Figure 4.6 to eliminate the edge effect cause by the forward predictor. Additionally, the

last two samples of the forward predicted data are kept, as shown in Figure 4.6. Note that **h** is the same in both cases; however the predicted values are different. Figure 4.6 illustrates the process demonstrating that the remaining samples not affected by the edge distortion are then averaged together.

$$\begin{array}{ll} \text{Forward Predicted data} & \left[f_1 \quad f_2 \quad f_3 \quad f_4 \quad f_5 \quad f_6 \quad f_7 \quad f_8 \right] \\ \text{Reverse Predicted data} & \left[r_8 \quad r_7 \quad r_6 \quad r_5 \quad r_4 \quad r_3 \quad r_2 \quad r_1 \right] \\ \hline \text{Resulting Predicted data} & \left[r_1 \quad r_2 \quad \frac{r_3 + f_3}{2} \quad \frac{r_4 + f_4}{2} \quad \frac{r_5 + f_5}{2} \quad \frac{r_6 + f_6}{2} \quad f_7 \quad f_8 \right] \end{array}$$

Figure 4.6: Resulting predicted data using forward and reverse prediction.

There are two distinct advantages found when using the above method. First, the size of the output data will be the same as that of the input data. Next, the edges will be smooth when the segments are put back together because the end points are fully predicted with each of the filter coefficients. These properties are advantageous when a large stream of data is to be analyzed in small incremental segments. Thus, a long data set can be split up into smaller segments and when reassembled will not have noticeable edges.

For a non-stationary signal such as the constant amplitude linear chirp, a windowed version of Wiener Prediction as described above is implemented on individual segments of the data stream. If the segment consists of zeroes, then the predictor returns all zeroes to prevent the formation of a singular matrix. Note that the wavelet-domain coefficients contain AWGN as previously stated; hence Wiener filtering will prove useful in denoising the remaining coefficients.

2. Wavelet- and Time-Domain Techniques

For this work the above windowed predictor was used, and the most suitable data input size to the predictor was determined through experimentation. For this work the prediction window is limited to sixteen samples for first level decomposition wavelet denoising and thirty-two samples for time-based denoising. For the signals considered, we have found through experimentation that the smaller sample interval is necessary in

the wavelet domain for best results when applied only to the first level detail coefficients. Figures 4.7 (a) and (b) show the first 700 points of the noiseless and noisy constant amplitude linear chirp, and Figure 4.7 (c) demonstrates the denoising capability achieved by implementing the predictor on the constant amplitude linear chirp. Notice there is no added distortion at the edges of the waveform segments.

The following section implements a low computational cost filtering method for comparison purposes. The Median Filter performs well for non-stationary, but low-frequency signals because of its inherent lowpass filtering characteristics.

3. Median Filtering

A one-dimensional median filter is chosen because of its simplicity and low computational cost. However, it requires the user to have some *a priori* knowledge of the signal characteristics because it does not perform well if the signal has high frequency components. The constant amplitude linear chirp and random sinusoid generated in this work occupy the high frequency spectrum; so denoising results are expected to be poor.

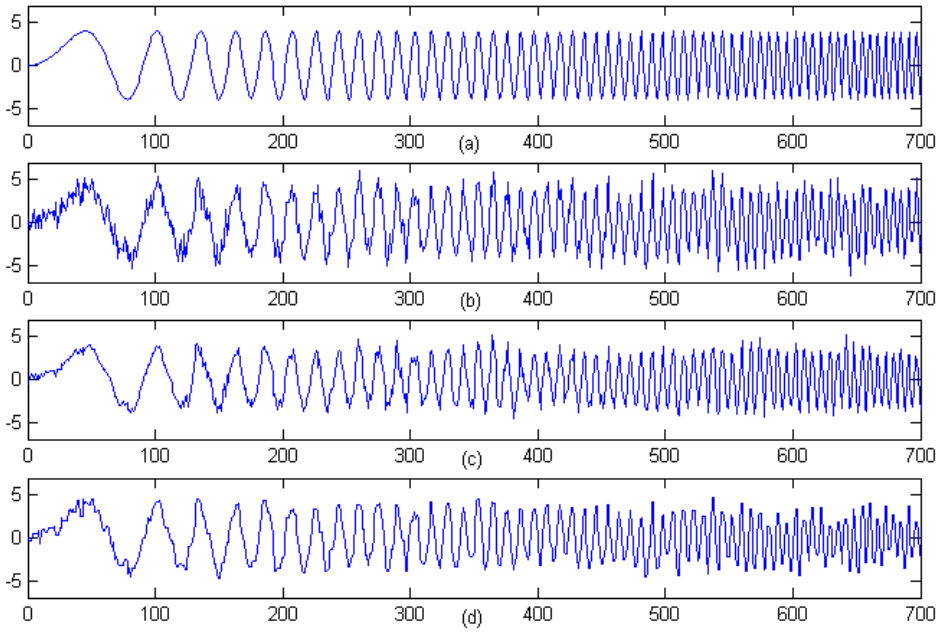


Figure 4.7: Constant amplitude linear chirp in time domain (First 700 samples): (a) True noiseless signal; (b) Signal in AWGN with SNR = 9 dB; (c) Time domain Wiener prediction denoising with window size = 128; (d) 3rd order Median filter denoising.

The median filter or medfilt1 in Matlab [20] estimates $\hat{s}(n)$ by looking at $s(n-1)$, $s(n)$, and $s(n+1)$, and chooses as output $\hat{s}(n)$ the median value of those 3 values. For segments this method also returns the same number of samples it receives. Figure 4.7 (d) demonstrates the median filtering denoising ability on a constant amplitude linear chirp. From visual inspection it appears to do fairly well at denoising the low frequency portion of the signal.

This chapter identified several wavelet- and time-domain denoising schemes. The following chapter compares the preceding denoising scheme. First, orthonormal wavelet thresholding techniques are compared to the time-domain techniques, which are followed by a separate comparison of translation-invariant thresholding techniques. Next, we compare the best wavelet thresholding schemes from the orthonormal wavelet and translation invariant wavelet transforms. Finally, we compare results obtained with the Wiener prediction and Median filter schemes.

V. RESULTS

A. INTRODUCTION

The previous chapters discussed some important principles and theory which form the background for time-frequency based denoising. The following section compares performances obtained for each scheme considered in this work. The signals used are robust in that they encompass a broad range of the spectrum and no *a priori* signal information is exploited.

Throughout the following, the result of Wiener prediction on the data in the time domain is used as a benchmark. Hence, the other denoising techniques will be compared against the benchmark set by time-domain Wiener prediction. In all the following simulations the results from one hundred distinct simulations are averaged in order to produce a more precise statistical comparison.

1. Orthonormal Wavelet Denoising

The first simulations compare and contrast orthonormal wavelet domain techniques. Figures 5.1 through 5.3 illustrate MSE results between wavelet-based methods and time-based methods for the constant amplitude linear chirp of Equation (2.3), and demonstrate the effect of adding additional levels of decomposition N where $N = 2$, 6, and 9, respectively. Based on MSE performance, the time-domain Wiener prediction clearly produced the best results for this signal since its average MSE is predominately less than each of the other methods. Notice that for $N = 6$ and 9 the MSE for the Orthonormal soft SURE thresholding performs better than when $N = 2$. This result is to be expected since more frequency ranges are being cleaned. Note also that nine levels of decomposition did not provide noticeably different results than six levels of decomposition. Hence, a point of diminishing returns is reached where the added computations (for implementing further levels of decomposition) do not provide added benefit.

Recall that visual thresholding is less desirable for smaller data segments because of its lower performance due to the large threshold size relative to the number of data samples. However, the benefits of Visual thresholding may outweigh those of SURE thresholding for longer data samples.

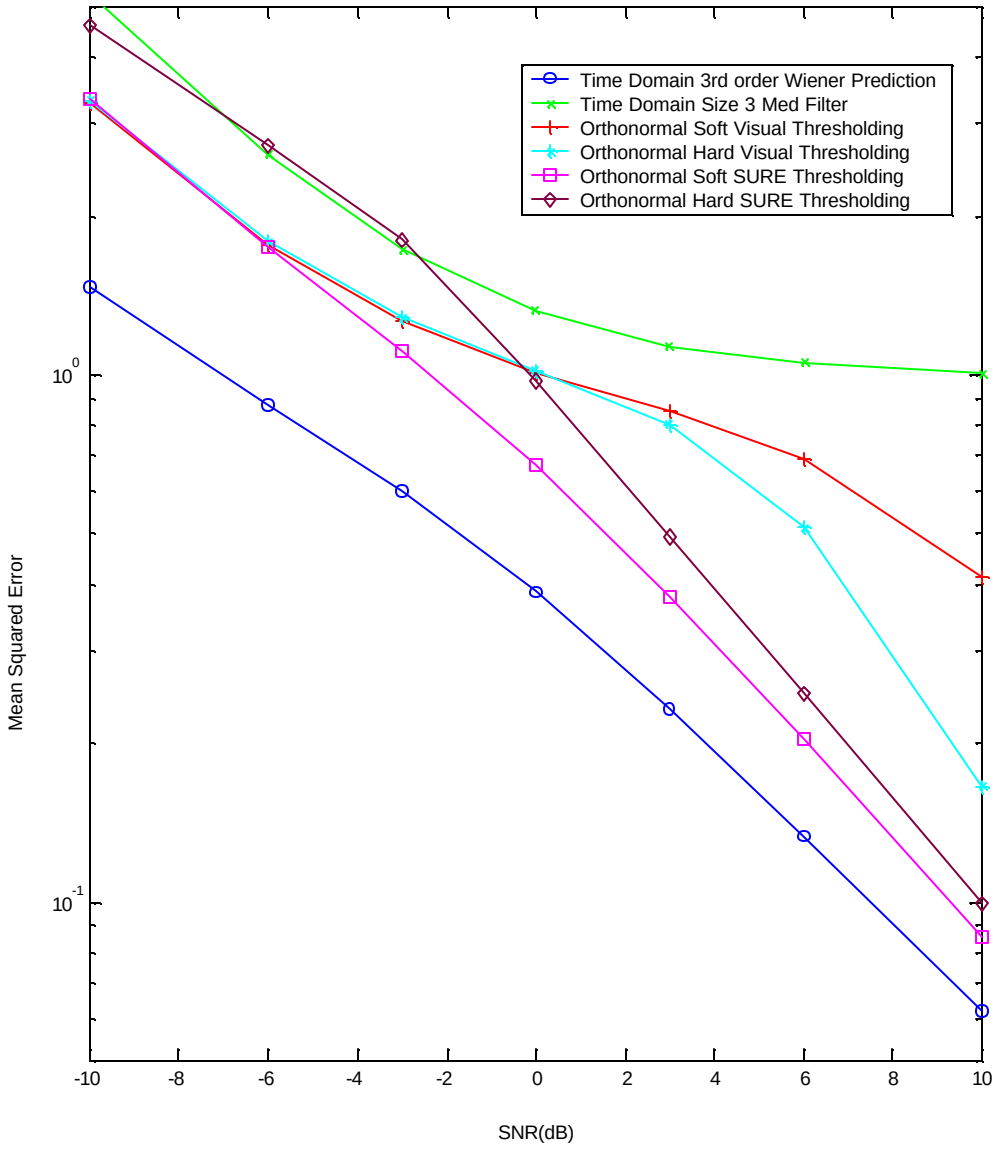


Figure 5.1: MSE vs. SNR for orthogonal wavelet-based denoising of constant amplitude linear Chirp from Equation (2.3) with $N = 2$ (Average of 100 simulations).

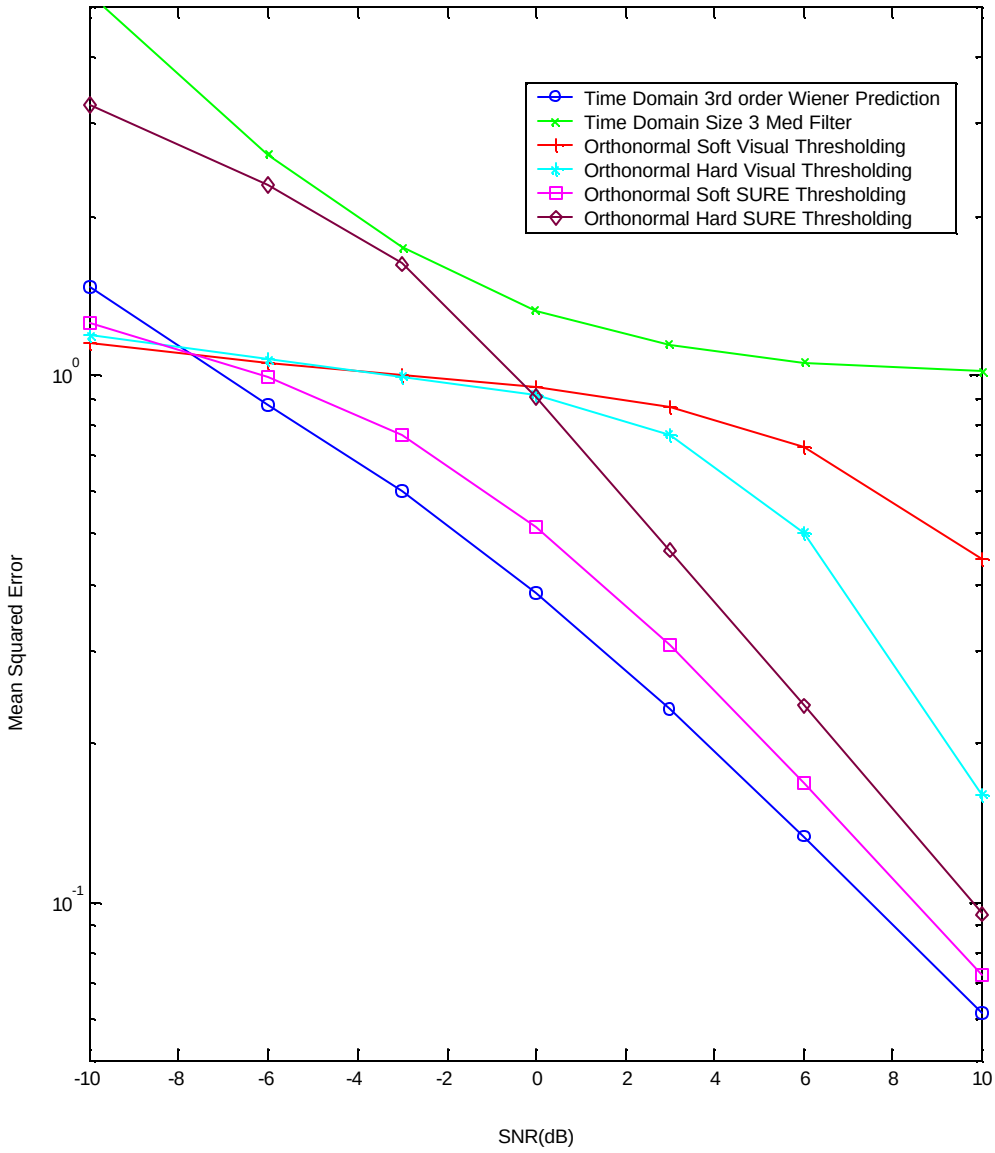


Figure 5.2: MSE vs. SNR for orthogonal wavelet-based denoising of constant amplitude linear Chirp from Equation (2.3) with $N = 6$ (Average of 100 simulations).

In addition, results show that Median filtering produces the worst MSE results, as expected, since the Median filter acts as a lowpass filter. Hence, Median filtering does not achieve acceptable results for the nonstationary multi-frequency scale signals used. Median filtering may produce better results in a higher sample rate environment where changes in signal statistics are relatively insignificant over the length of the filter.

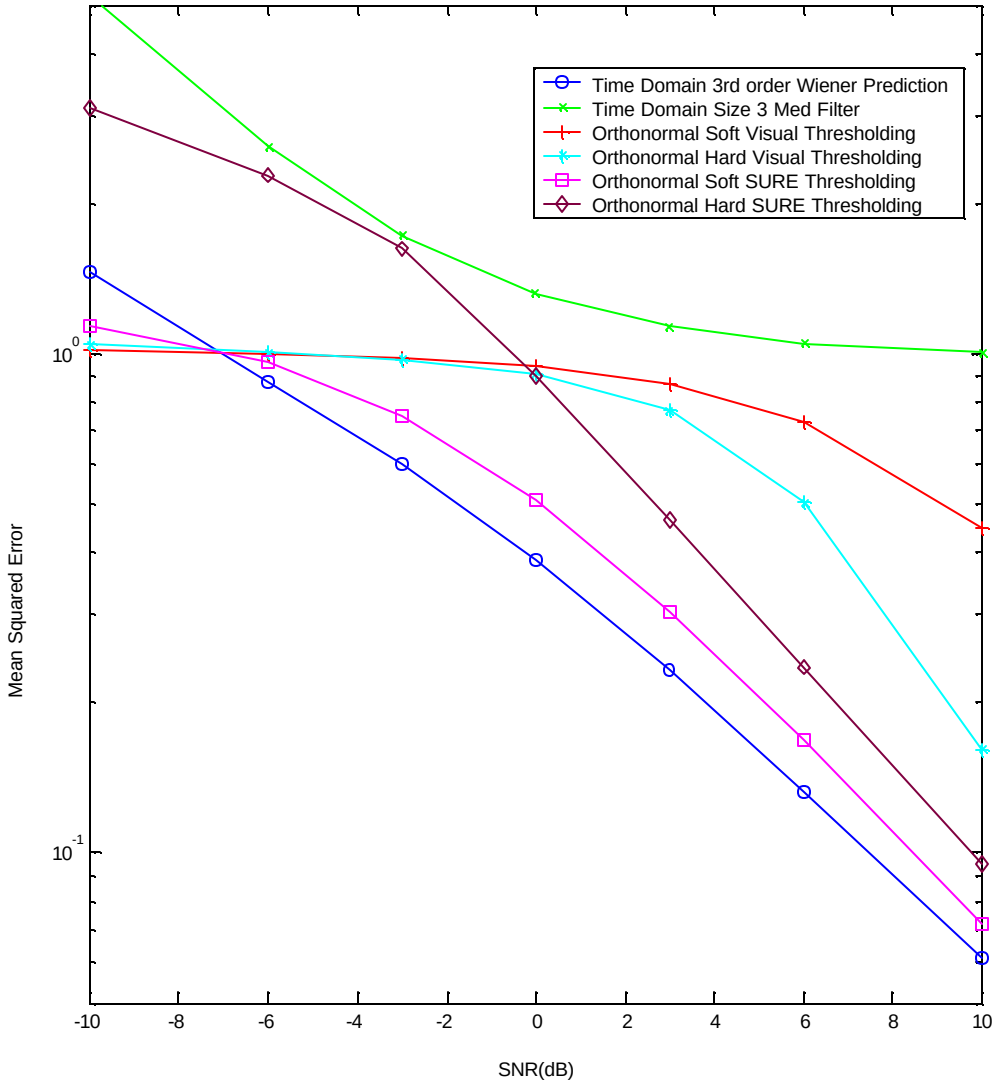


Figure 5.3: MSE vs. SNR for orthogonal wavelet-based denoising of constant amplitude linear Chirp from Equation (2.3) with $N = 9$ (Average of 100 simulations).

Figure 5.4 illustrates one trial of the de-noised signal $\hat{s}(n)$ and the original signal $s(n)$ to demonstrate the visual differences found in the various denoising schemes for $N = 9$. These results are consistent with those found in Figure 5.3 and illustrate the difference between hard and soft thresholding. Results show better MSE performances are obtained with the SURE soft threshold than with the SURE hard threshold for this signal.

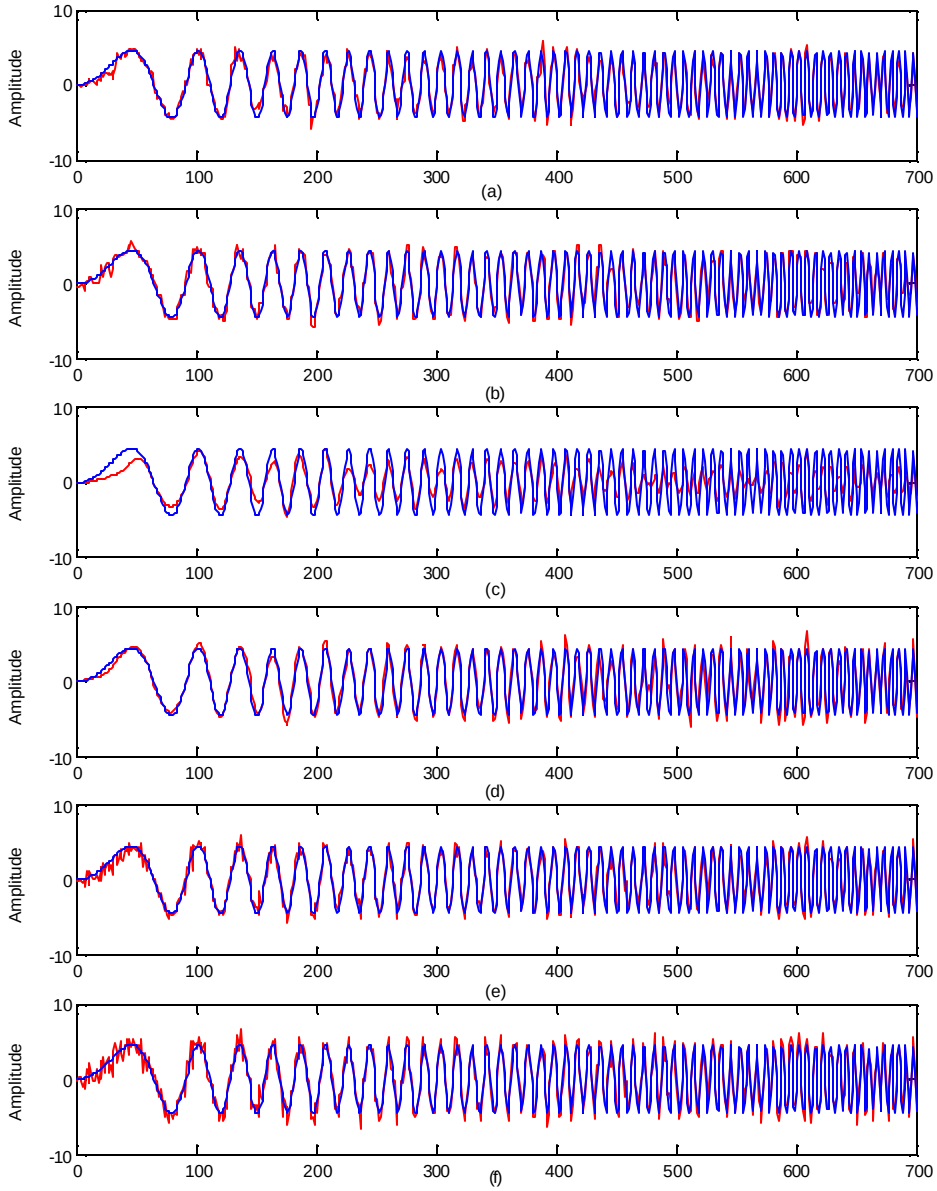


Figure 5.4: First 700 samples of constant amplitude linear Chirp from Equation (2.3) using orthogonal wavelet-based denoising with $N = 9$ and SNR = 10 dB. (a) Time-domain 3rd order Wiener Prediction; (b) Time-domain Size 3 Median Filter; (c) Soft visual thresholding; (d) Hard visual thresholding; (e) Soft SURE thresholding; (f) Hard SURE thresholding (Original noiseless signal is shown in blue).

Figure 5.5 demonstrates the average effectiveness of soft SURE wavelet denoising over time domain Wiener prediction on sinusoidal signals. Note that for this signal, a decrease in the MSE is illustrated for both hard and soft SURE thresholding compared to the Wiener prediction benchmark. The SURE soft threshold MSE was slightly lower than the time domain Wiener prediction in this case, whereas for the constant amplitude linear chirp it was slightly higher. Thus, the overall MSE performance with regard to the performance of wavelet and Wiener prediction on both signal types is almost identical. The advantage of wavelet thresholding in this case is not in the MSE performance. Recall that the best-suited size of the prediction window was dependent on *a priori* experimentation whereas wavelet thresholding did not require any *a priori* information other than segment size to determine the number of levels of decomposition. The next Section explores the effect of cycle-spinning using the same comparisons as above.

2. Translation-Invariant Wavelet Denoising

In this Section cycle-spinning results are reported and compared using the wavelet thresholding techniques discussed. For ease of comparison, nine decomposition levels are used in the comparison. Figures 5.6 and 5.7 average the results for the various denoising schemes on the constant amplitude chirp and random sinusoid, respectively. Results for the constant amplitude chirp are similar to those obtained with orthonormal wavelet denoising; however a few differences are noted. First, it appears that translation invariance as applied produces slightly enhanced denoising capability for each of the thresholding schemes. In addition, hard thresholding with the visual threshold shows a marked improvement, considering the small sample space involved.

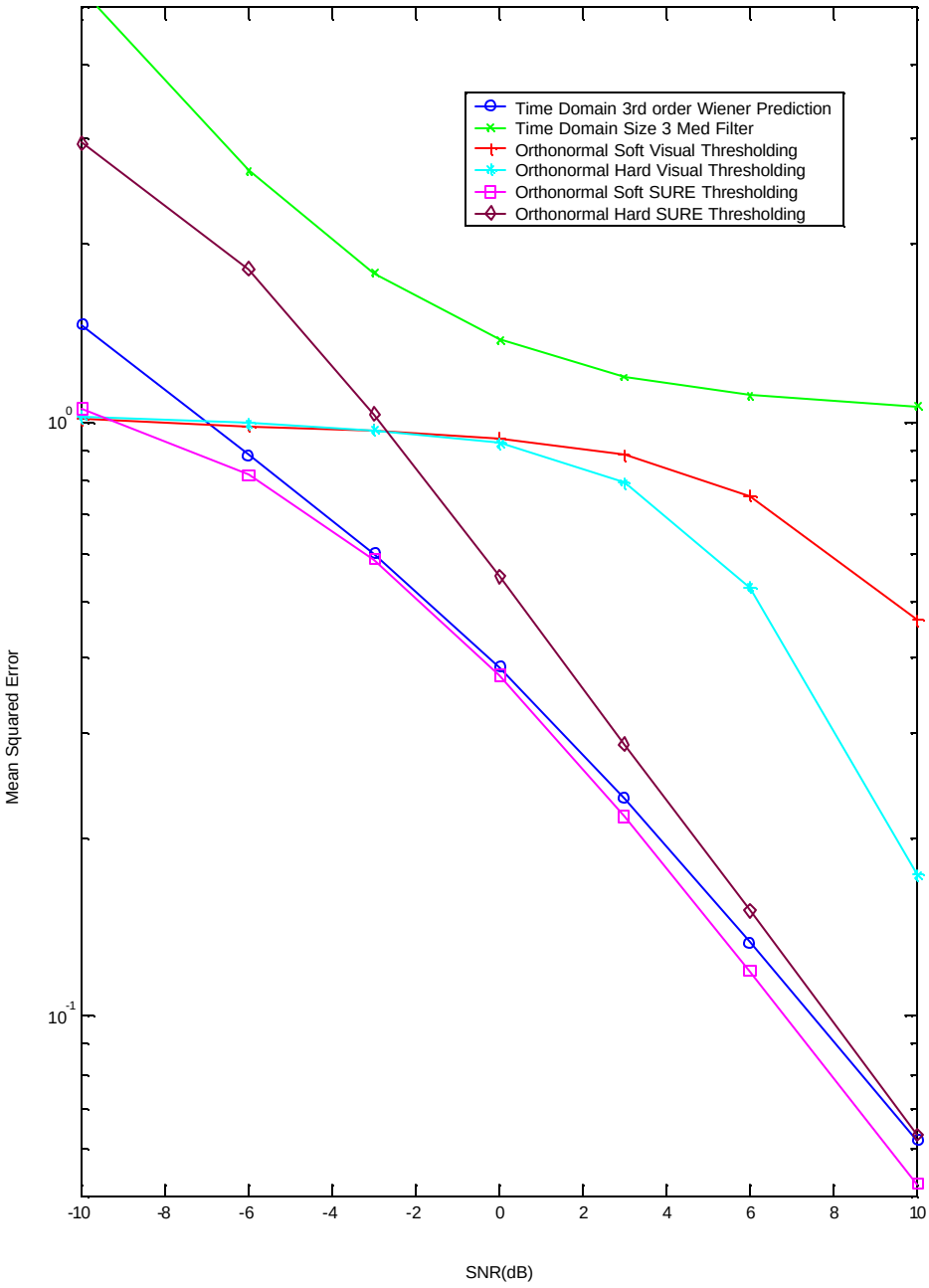


Figure 5.5: MSE vs. SNR for orthogonal wavelet-based denoising of sine wave with frequency randomly set to a value between 0 and half sampling frequency with $N = 9$ (Average of 100 simulations).

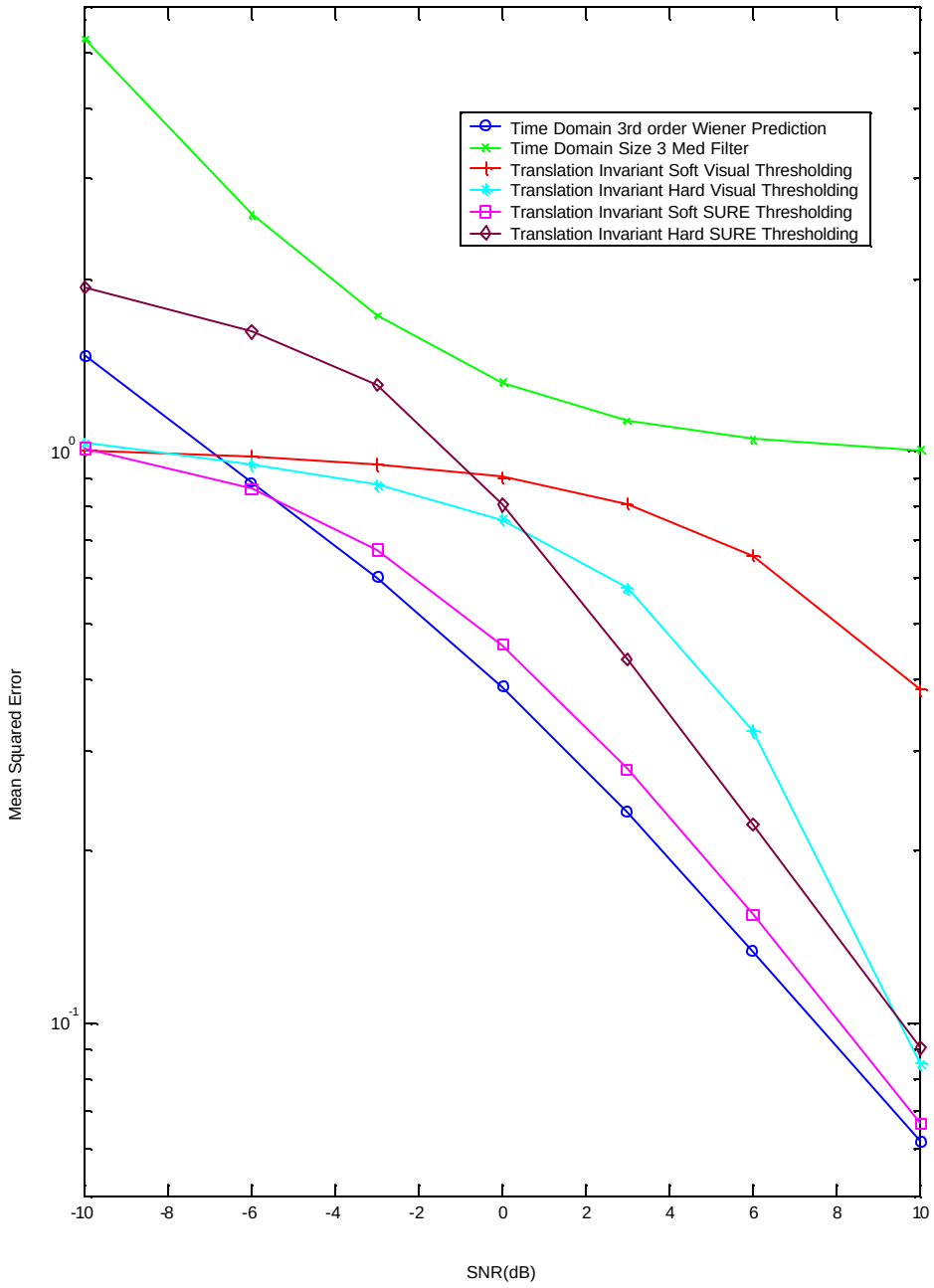


Figure 5.6: MSE vs. SNR for translation-invariant wavelet-based denoising of constant amplitude linear Chirp from Equation (2.3) with $N = 9$ (Average of 100 simulations).

Figure 5.7 that shows the best MSE performances are obtained with the soft SURE threshold. Next, we compare the soft SURE threshold results to those obtained with cycle-spinning. In addition, we compare combined scheme performances.

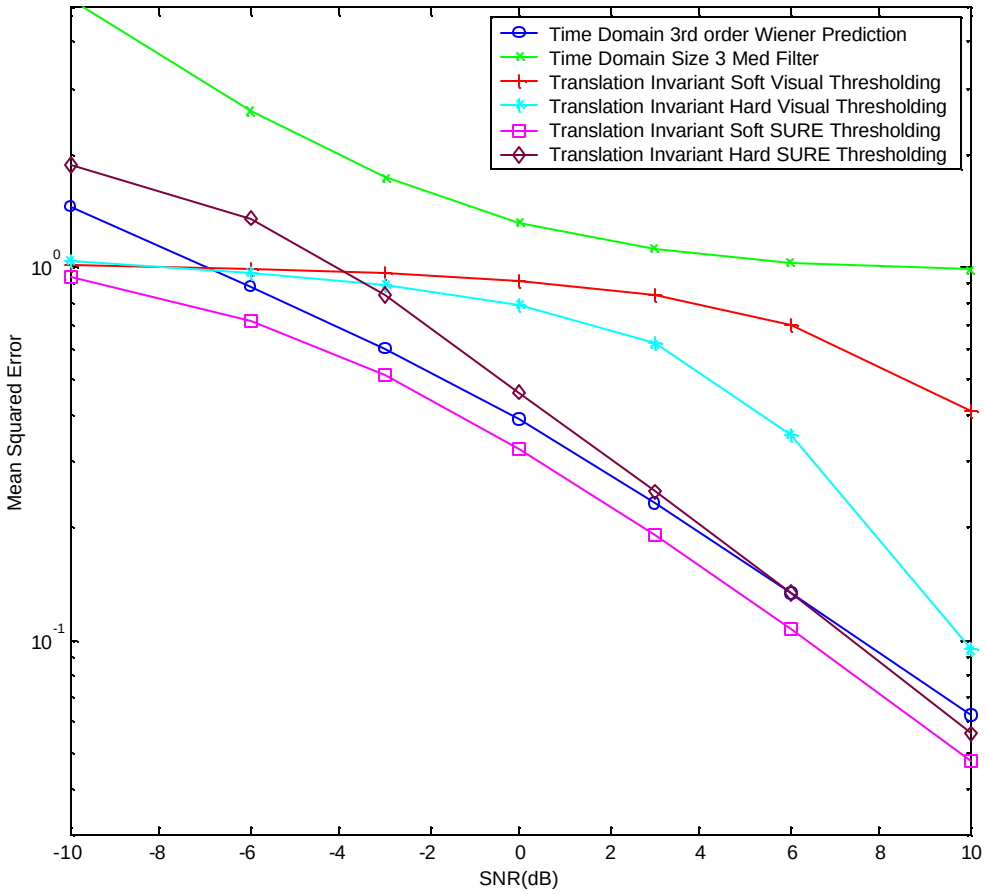


Figure 5.7: MSE vs. SNR for translation-invariant wavelet-based denoising of sine wave with frequency randomly set to a value between 0 and half sampling frequency with $N = 9$ (Average of 100 simulations).

3. Combined Time and Wavelet Based Denoising

This section compares the best orthonormal and translation-invariant thresholding schemes, and provides results for the combined schemes. Recall that in a combined scheme either Wiener prediction or median filtering is applied to the wavelet coefficients after thresholding and prior to reconstruction. This technique addresses the fact that no attempt is made in thresholding to eliminate noise in the remaining coefficients. Thresholding continues to be vital in the denoising process because it is able to completely zero

out a portion of the noise-only coefficients whereas no such guarantee exists with Wiener prediction and median filtering. Recall, however, that with Wiener prediction one must make certain the autocorrelation matrix is not singular, whereby ensuring any string of coefficients thresholded to zero remain zero. Verifying whether the matrix is ill-conditioned will be left to future work; however in such a case a small quantity of noise may be added to the data segment prior to taking its autocorrelation. Additionally, recall that for this work the time-based techniques are restricted to the detail coefficients from the first level of decomposition.

Figures 5.8 and 5.9 show that applying Wiener prediction in the prescribed manner does provide a slight advantage over denoising utilizing only thresholding for the sine wave signal. This result is encouraging because it illustrates that additional denoising can be accomplished in conjunction with thresholding. Additionally, it shows that translation-invariant denoising provides added denoising capability over that of orthonormal wavelet denoising.

Figure 5.10 provides a visual representation of the first 700 samples of the de-noised constant amplitude chirp $\hat{s}(n)$ and the original signal $s(n)$. The de-noised signal in Figure 5.10 (e) demonstrates the effectiveness of the combined algorithm; we notice the relative noise-free nature of the result. Figure 5.10 (f) shows the (huge) degradation introduced by applying the median filter to the coefficients prior to reconstruction.

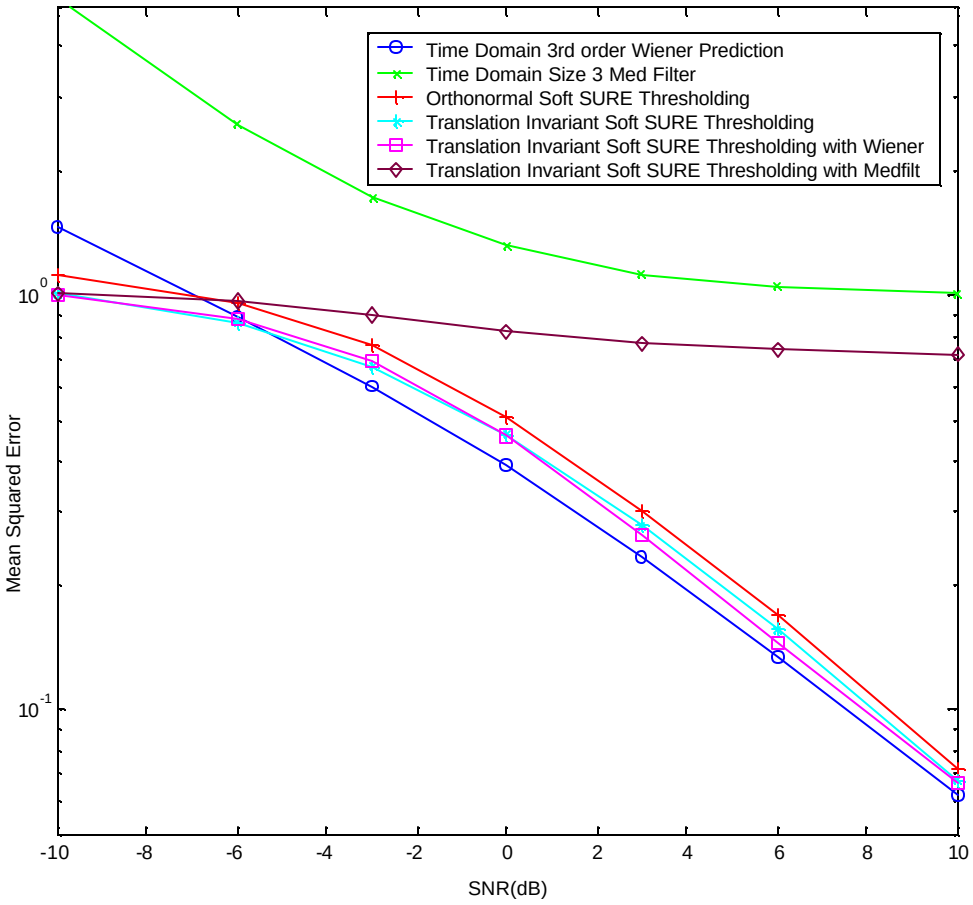


Figure 5.8: MSE vs. SNR for combined wavelet and time denoising of constant amplitude linear Chirp from Equation (2.3) with $N = 9$ (Average of 100 simulations).

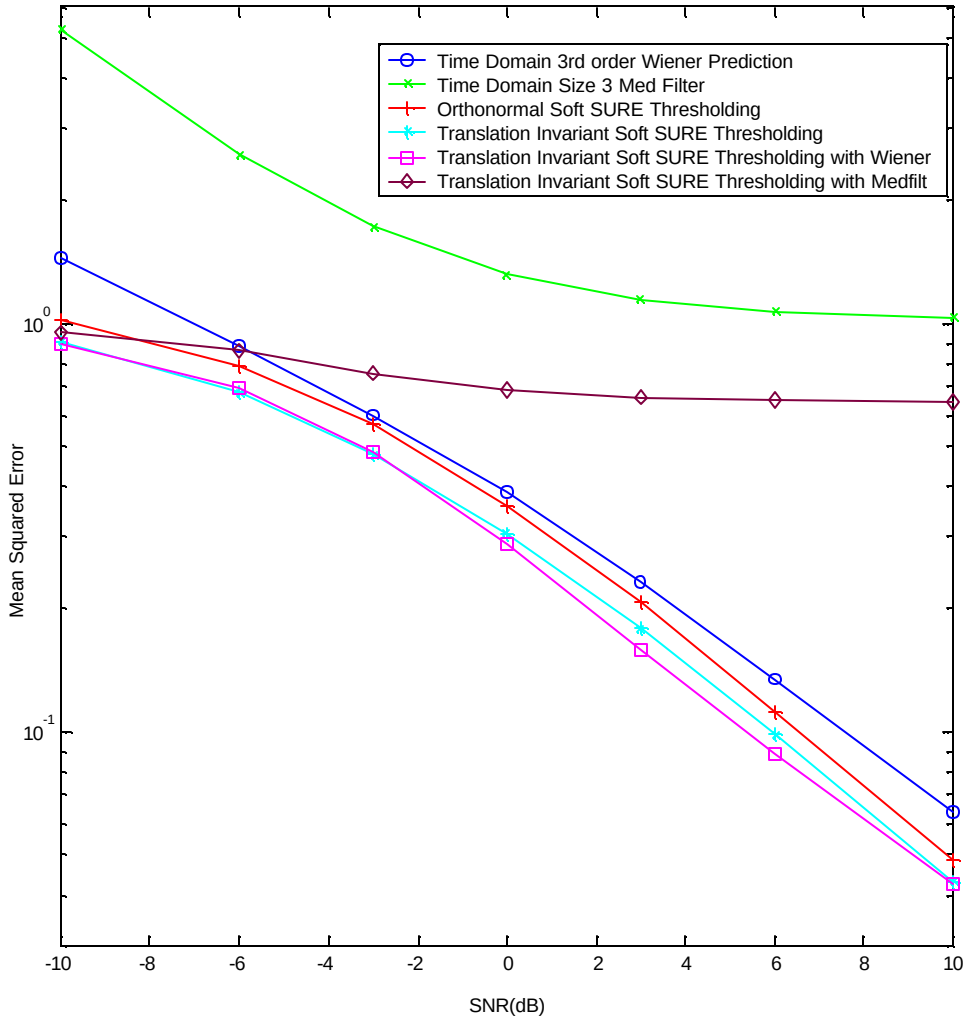


Figure 5.9: MSE vs. SNR for combined wavelet and time denoising of sine wave with frequency randomly set to a value between 0 and half sampling frequency with $N = 9$ (Average of 100 simulations).

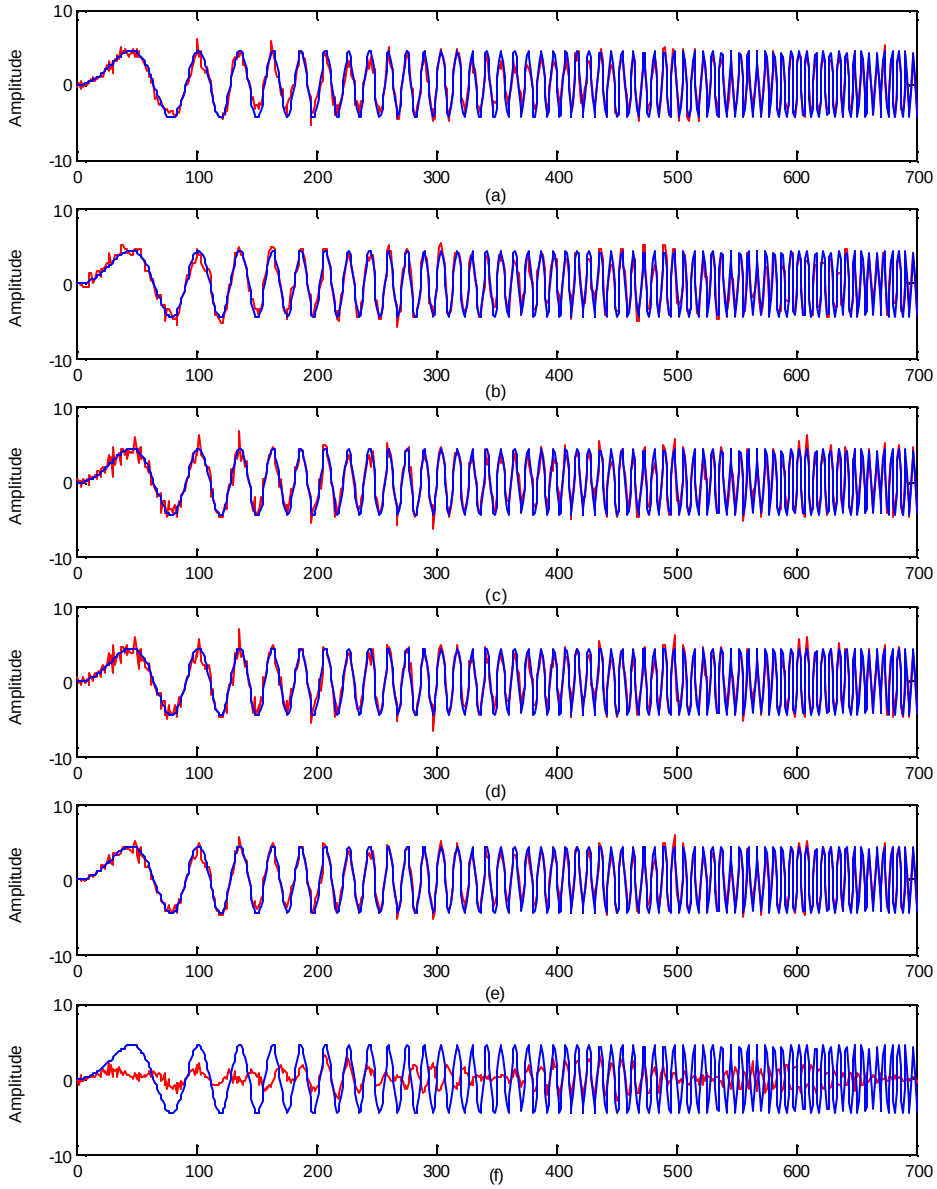


Figure 5.10: First 700 samples of constant amplitude linear Chirp from Equation (2.3) using combined wavelet and time-based denoising with $N = 9$ and SNR = 10 dB. (a) Time domain 3rd order Wiener Prediction; (b) Time domain Size 3 Median Filter; (c) Orthonormal soft SURE thresholding; (d) Translation-invariant soft SURE thresholding; (e) Translation-invariant soft SURE thresholding with Wiener Prediction prior to reconstruction; (f) Translation-invariant soft SURE thresholding with Median filtering prior to reconstruction.

The following Section provides the results for recursive cycle-spinning as defined in this work and includes a variant of recursive cycle-spinning that includes Wiener prediction just prior to the last reconstruction step.

4. Recursive Translation-Invariant Wavelet Denoising

This section combines the best results from the above sections and compares them to recursive cycle-spinning and a combined version of recursive cycle-spinning and Wiener prediction.

Figures 5.11 and 5.12 show that at higher SNR levels the combined recursive method with Wiener prediction performed prior to the last reconstruction has the best MSE performance. Notice that in the case of the constant amplitude linear chirp illustrated in Figure 5.11, the combined scheme produces better results at lower SNR's than for the random sine wave illustrated in Figure 5.12. Hence, *a priori* information about the signal is necessary in the implementation of this combined scheme; therefore it should not be used in an unknown signal environment.

The following section performs the denoising algorithms on experimental data sets. No *a priori* information was available for these data sets. Due to the large sample rate during data collection these segments contain a large number of samples.

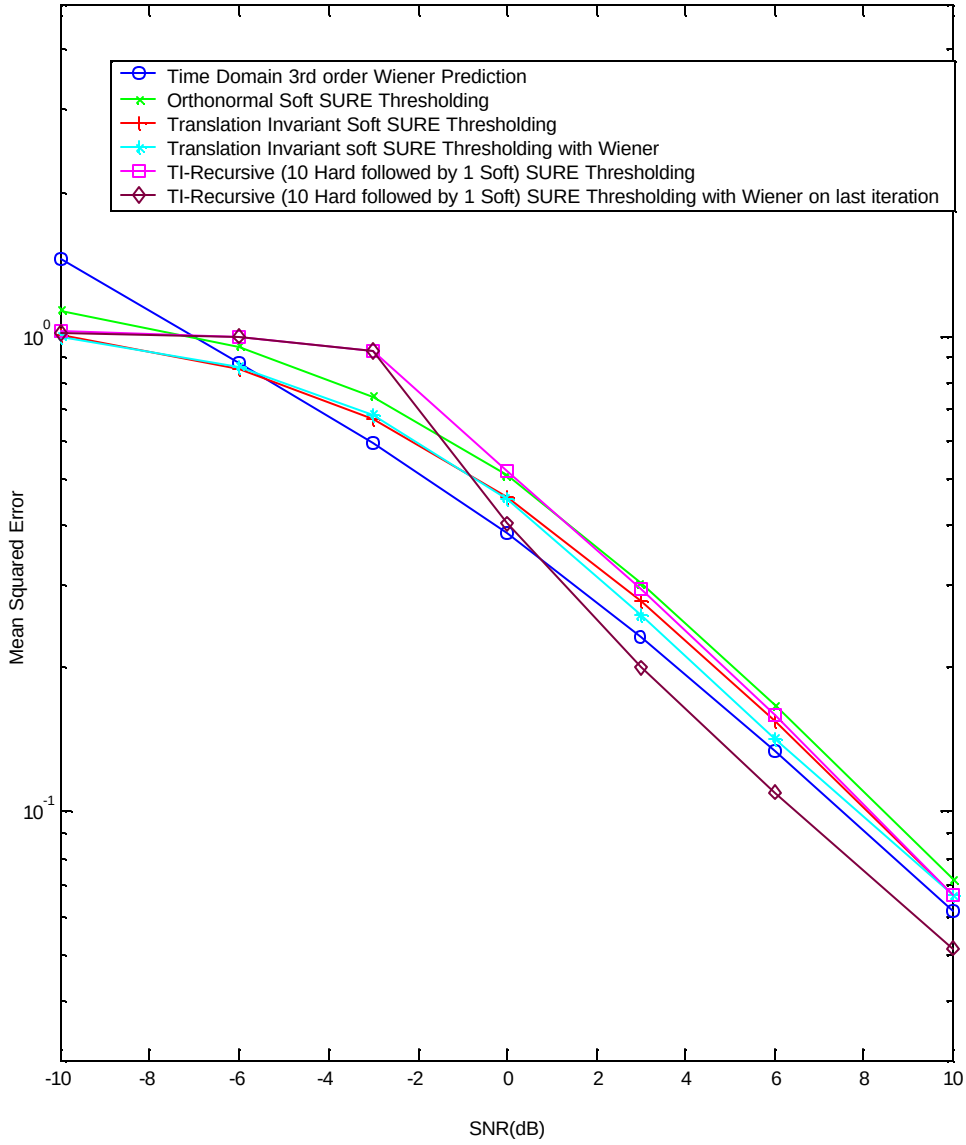


Figure 5.11: MSE vs. SNR for recursive translation-invariant wavelet denoising of constant amplitude linear Chirp from Equation (2.3) with $N = 9$ (Average of 100 simulations).

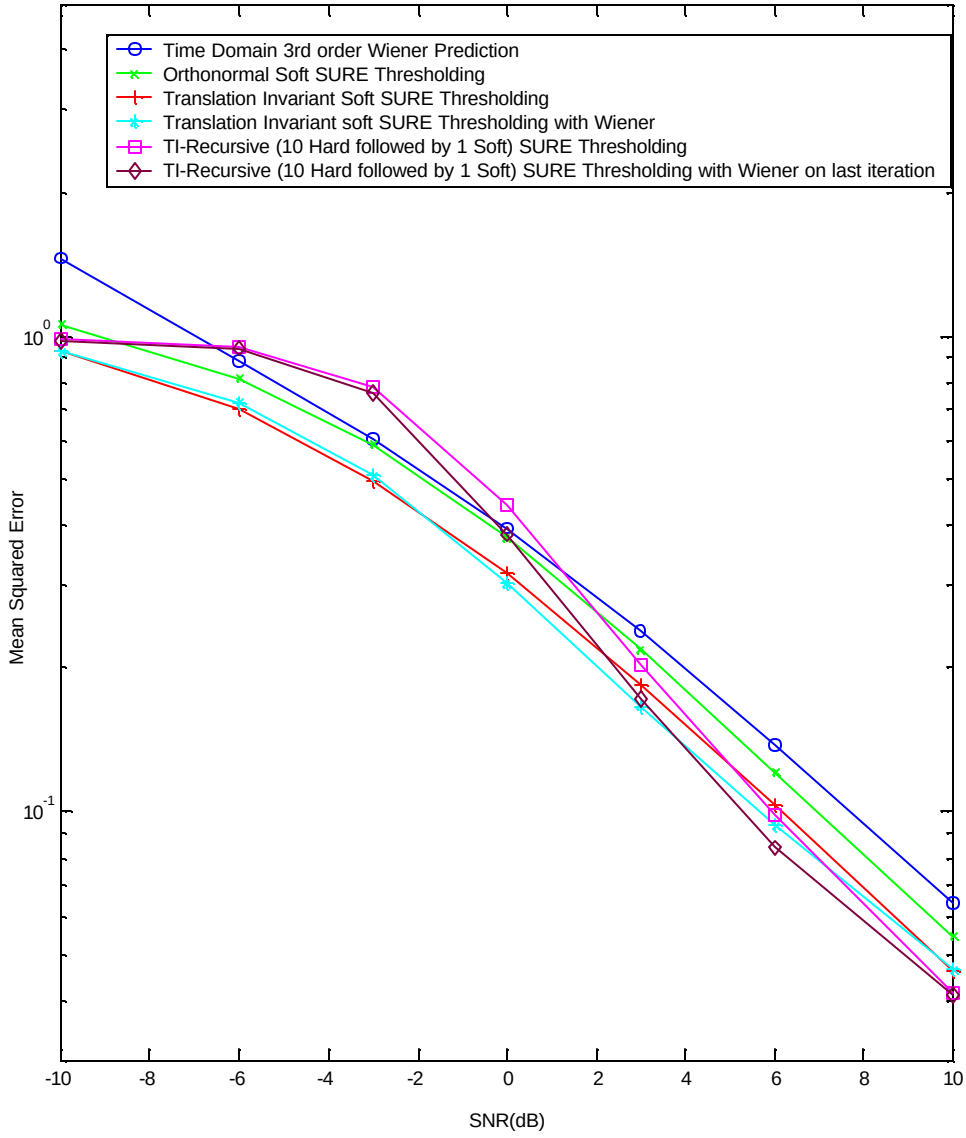


Figure 5.12: MSE vs. SNR for recursive translation-invariant wavelet denoising of sine wave with frequency randomly set to a value between 0 and half sampling frequency with $N = 9$ (Average of 100 simulations).

5. Denoising of Experimental Data Sets

The experimental data we worked with are referred to as data sets A through F. They are de-noised via several of the above methods as shown in Figures 5.13 through 5.26. No *a priori* information has been collected on this data, hence statistical analysis was performed on a relatively signal-free portion of the signal to determine whether it could be modeled white and Gaussian. Whiteness of the noise was evaluated by computing correlation estimates of noise-only data segments, while the Gaussian assumption was tested with quantile-quantile plots. Simulations show that the noise segments exhibit some colored properties and deviate somewhat from the Gaussian assumptions. The deviations were deemed small enough so that we could still attempt to consider the noise distortions as white and Gaussian for the purpose of Wavelet denoising. An additional pre-whitening step could be introduced for potentially more accurate results at a higher computational cost. Figures 5.13 through 19 compare effects of the visual and SURE threshold and the orthogonal transformation to the translation-invariant transformation.

In this case it is not possible to notice any noticeable difference between denoising techniques that included Prediction and those that did not. A possible explanation is that the signal did not change significantly during the prediction window duration due to the large sampling rate. Hence, a *priori* information about the signal and the sampling frequency is necessary to correctly apply Wiener prediction in both time and wavelet domains.

In addition, note that the visual thresholding technique essentially eliminates all noise, as expected. Since the data segment is very large in this case the negative effects experienced with the smaller data sets are not experienced as illustrated in Figures 5.13 through 5.15. Unfortunately, visual thresholding still finds the threshold with the greatest risk of signal loss. For instance, the latter half of the data stream in Figure 5.19 may have small amplitude signal components that the SURE thresholding technique picks up. Visual thresholding places the threshold above those components and eliminates them, however, resulting in greater jeopardy of thresholding desired signal coefficients.

Finally, as seen by comparing Figures 5.13 (e) and (f), there is no apparent difference viewed between the orthogonal transformation and the translation-invariant

transformation for these data sets. Therefore, the added cost of applying the translation-invariant transform does not justify the benefit for these signals.

Figures 5.20 through 26 illustrate the effect of recursive cycle-spinning on the visual and SURE soft thresholding. From top to bottom are placed the original data set, Wiener predicted data with a larger window size of 512, translation-invariant soft visual, recursive cycle-spinning with one iteration of hard visual followed by one iteration of soft visual, translation-invariant soft SURE, and recursive cycle-spinning with ten iterations of hard SURE thresholding followed by one iteration of soft SURE thresholding.

Figures 5.22 (e) and (f) show a slight denoising advantage of recursive cycle-spinning over its translation-invariant soft SURE counterpart. In most cases for these signals, however, the denoising effect of recursive cycle-spinning was unnoticeable as seen in Figures 5.21 (e) and (f). Thus, the cost involved in recursive cycle-spinning is much greater than its benefit.

This chapter systematically identified the best denoising schemes for the signals considered and applied those schemes to experimental data. The following chapter concludes which scheme is the method of choice for the signals and signal lengths considered. In addition, we make recommendations for further study.

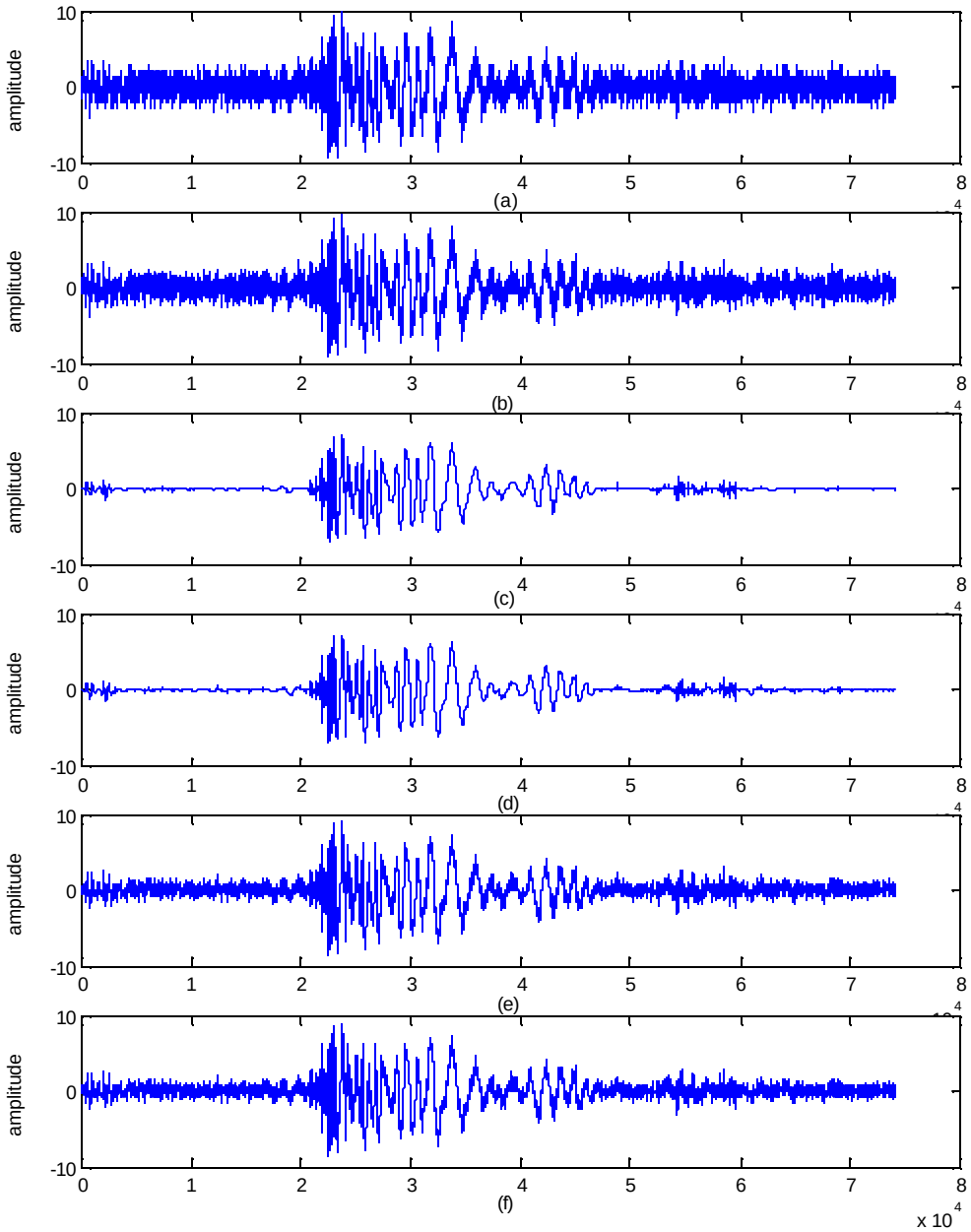


Figure 5.13: Denoising of Data segment A. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

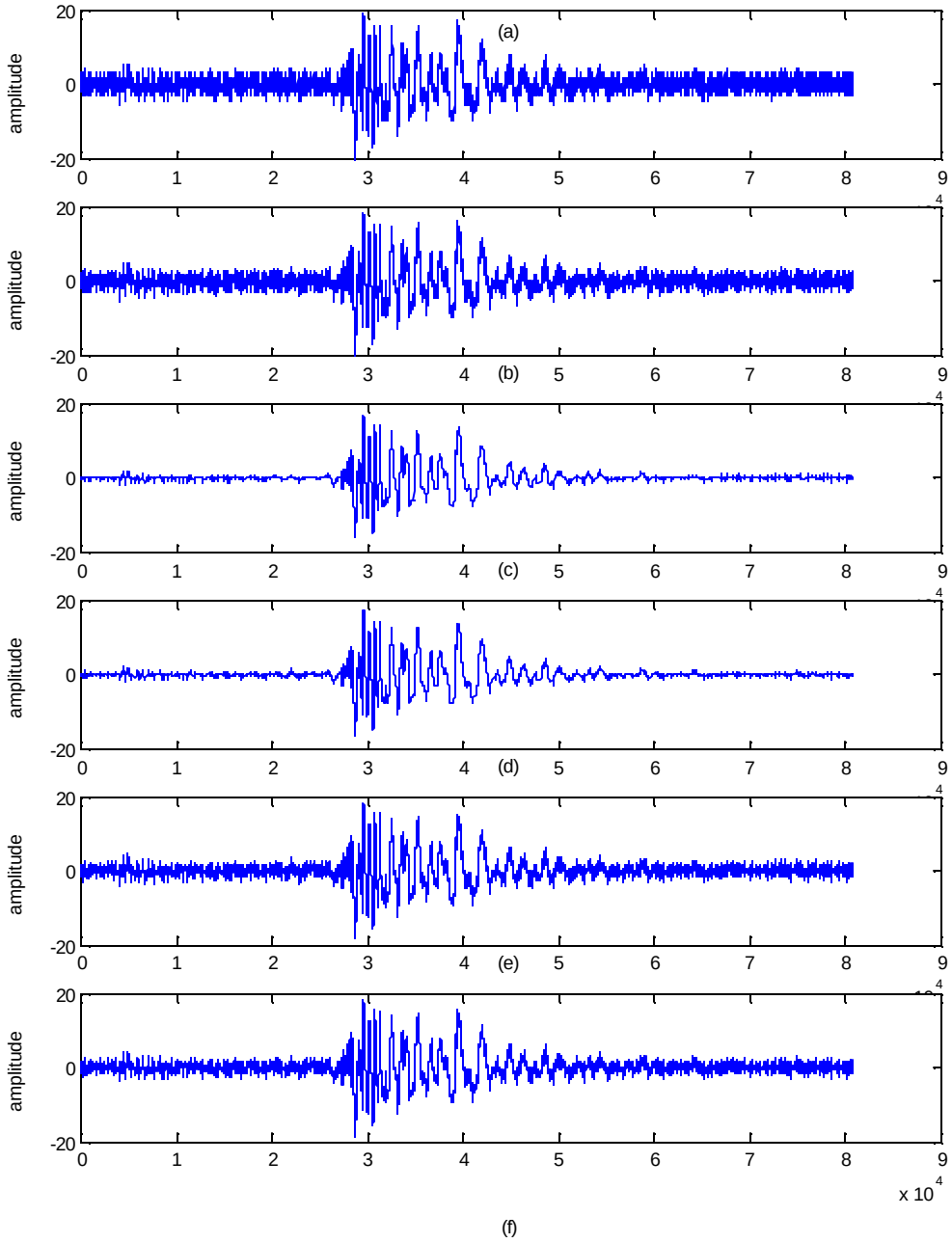


Figure 5.14: Denoising of Data segment B. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

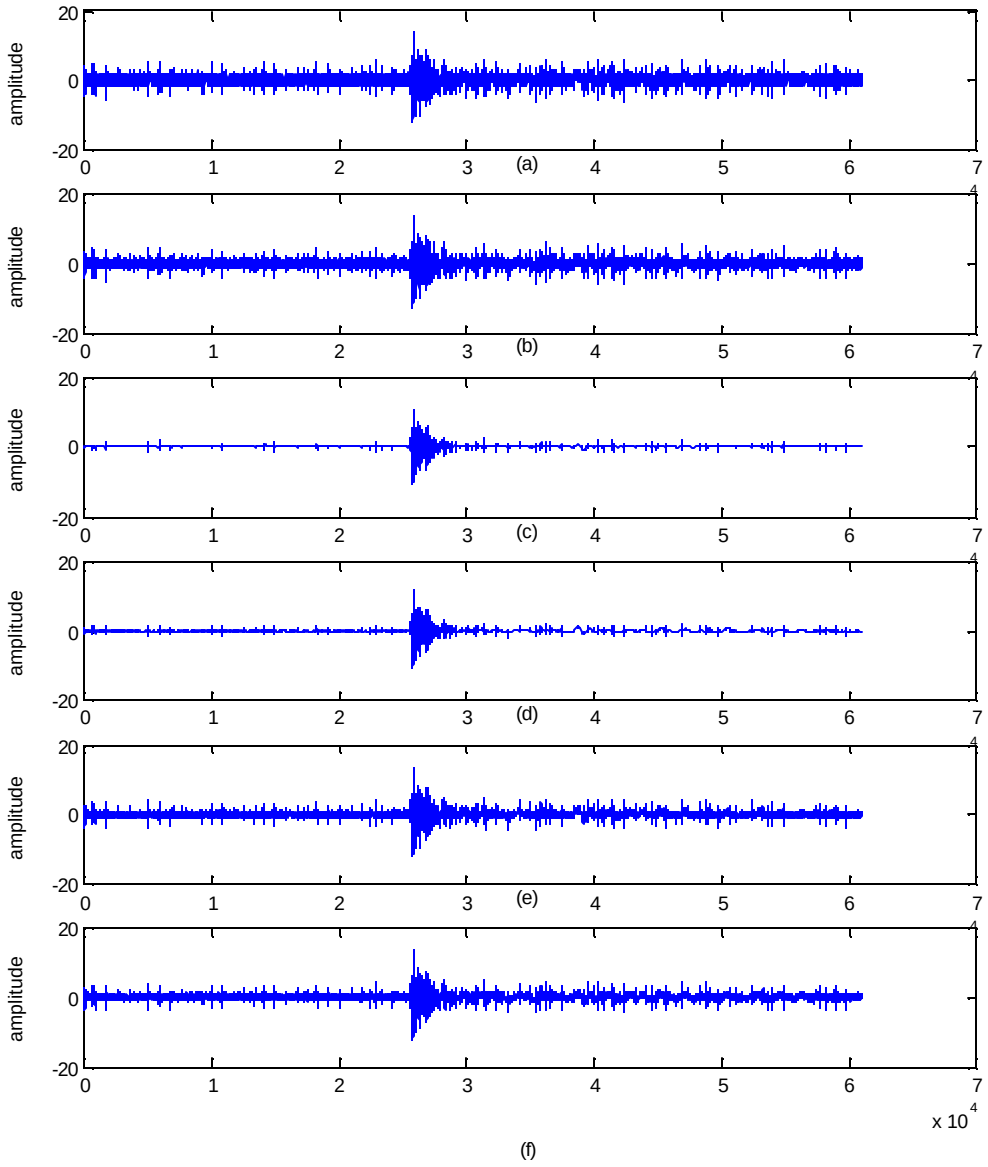


Figure 5.15: Denoising of Data segment C. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

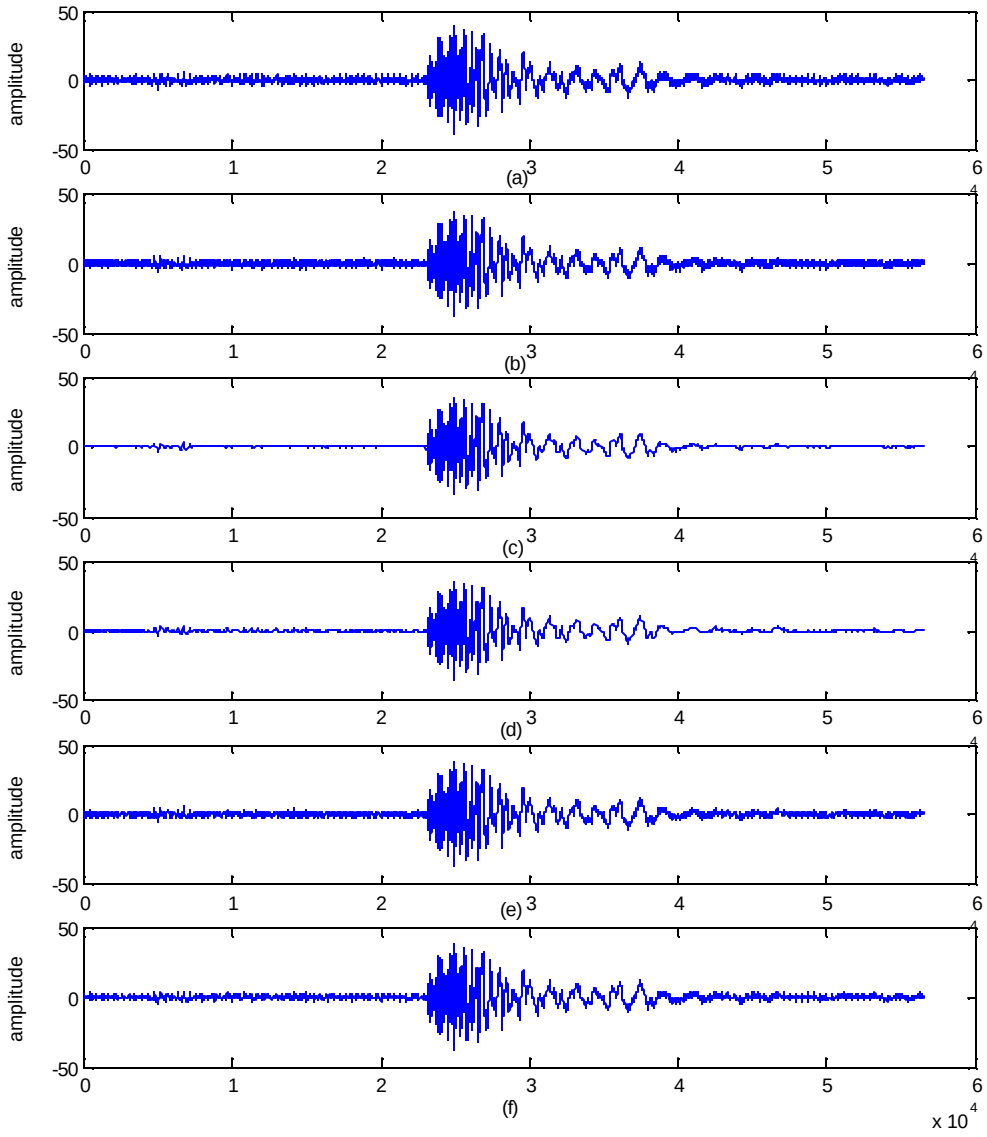


Figure 5.16: Denoising of Data segment D. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

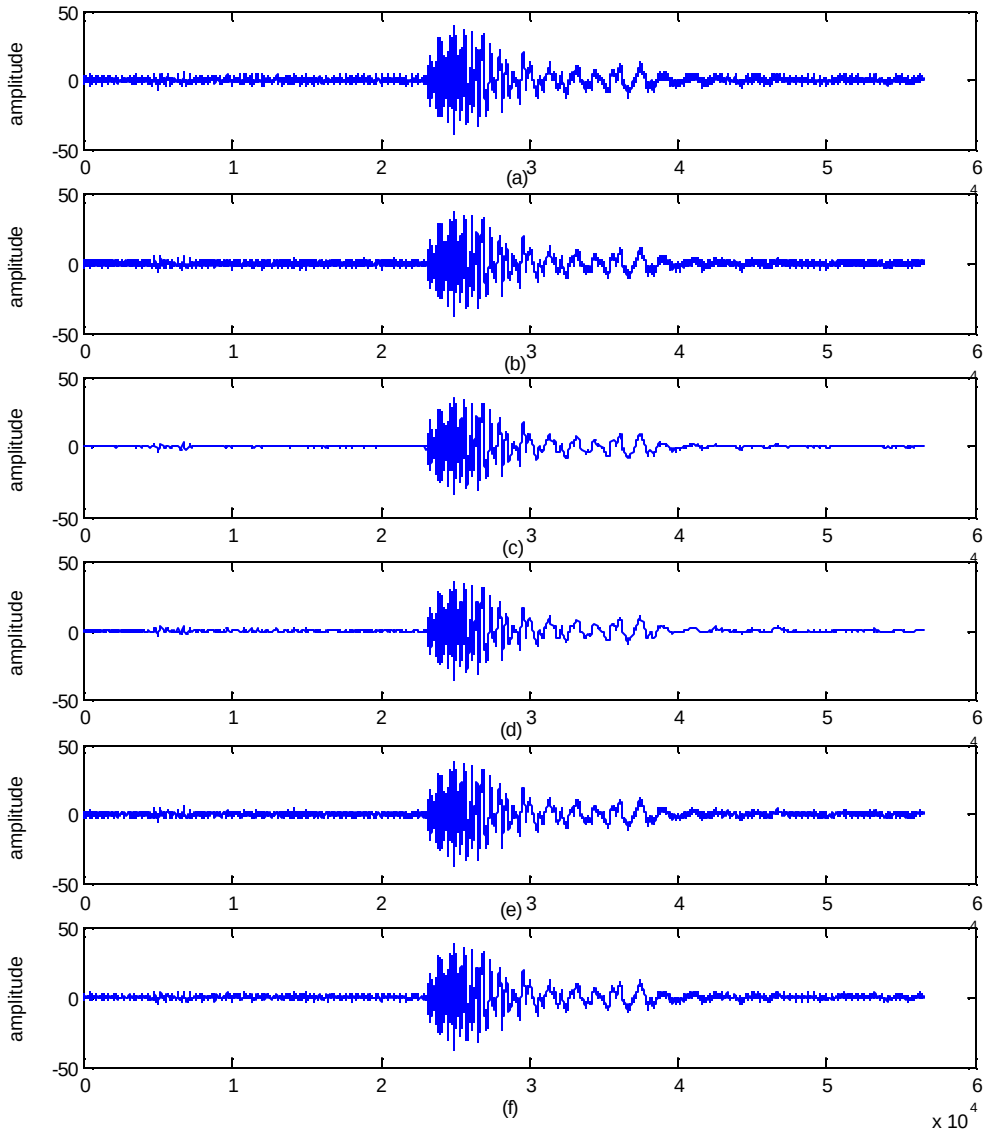


Figure 5.17: Denoising of Data segment E. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

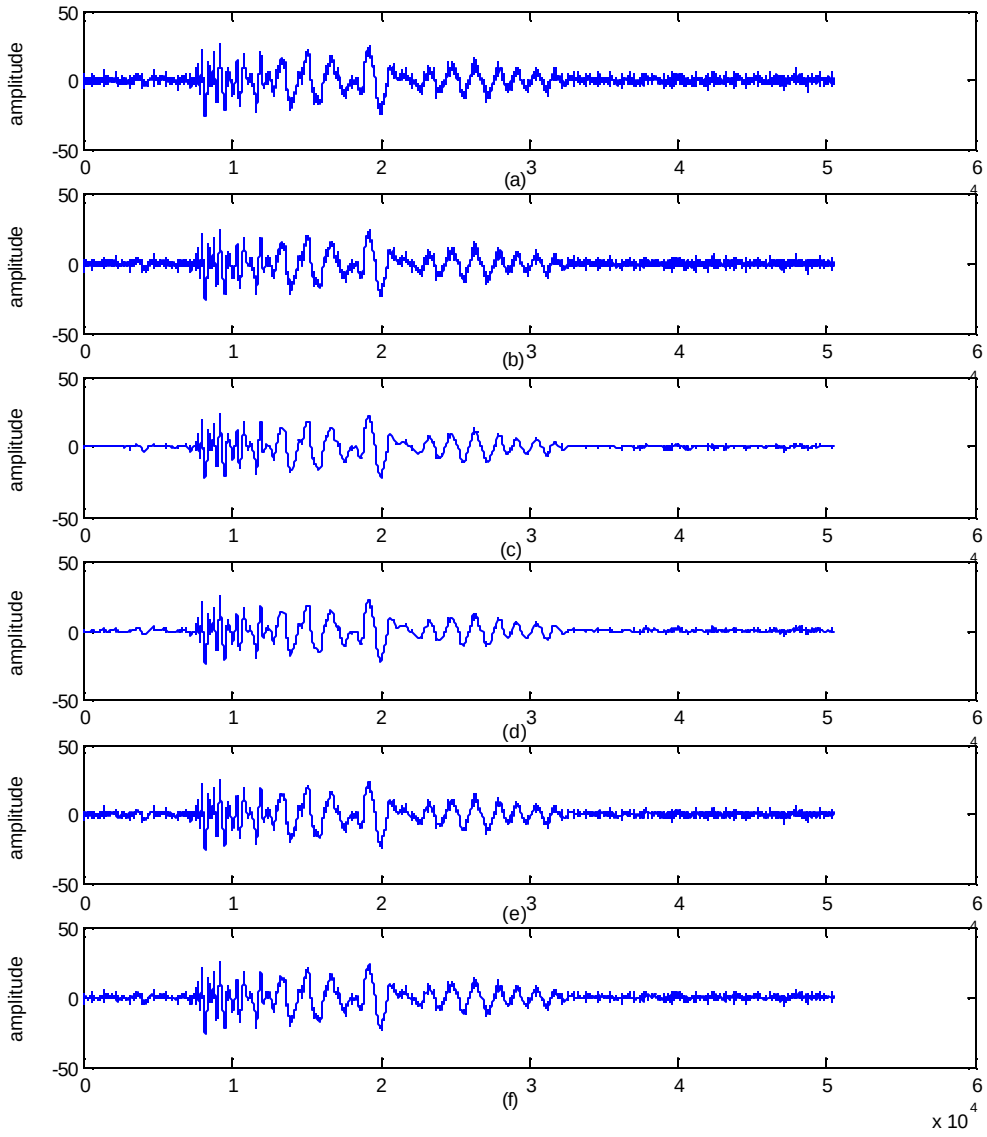


Figure 5.18: Denoising of Data segment F. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

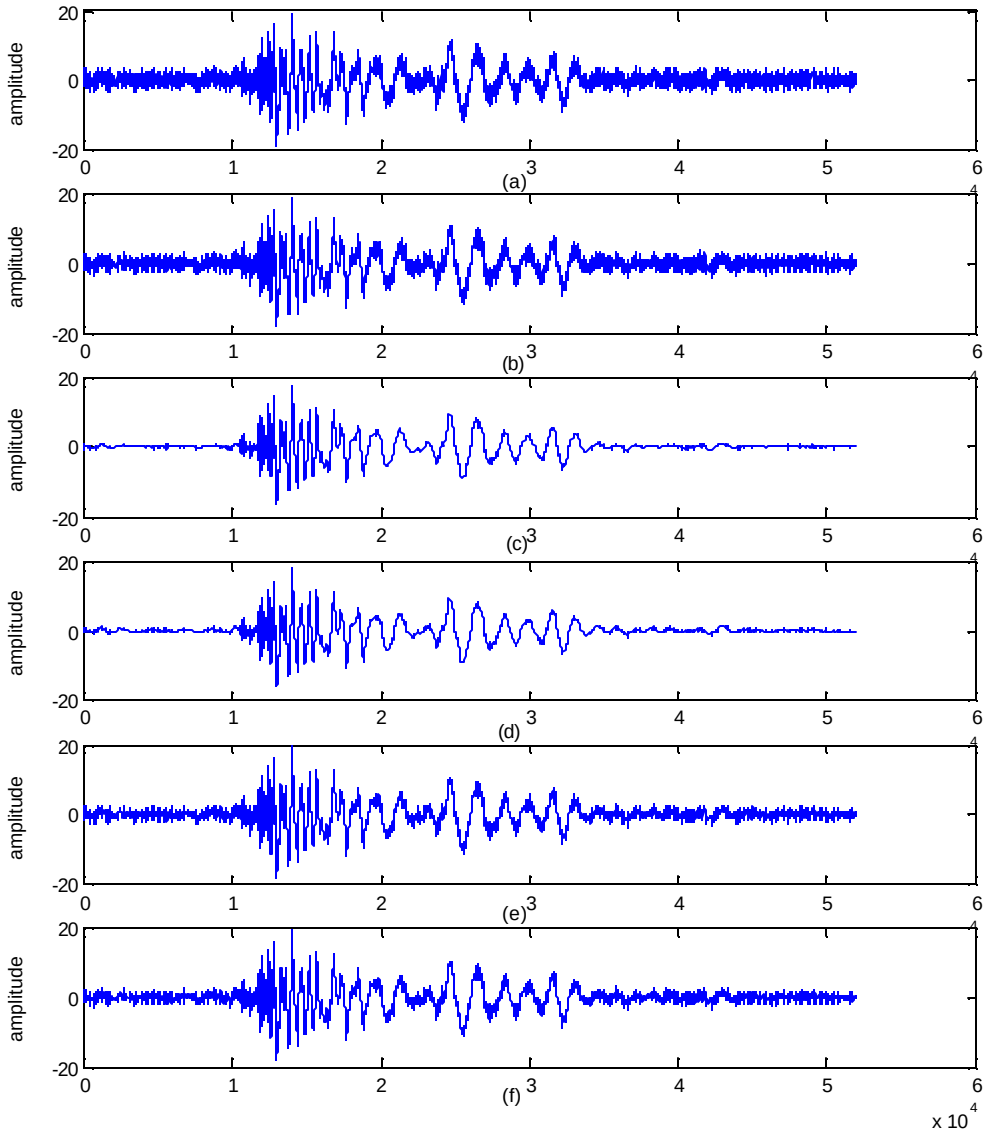


Figure 5.19: Denoising of Data segment G. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 32; (c) Orthonorml soft visual thresholding; (d) Translation-invariant soft visual thresholding; (e) Orthonormal soft SURE thresholding; (f) Translation-invariant soft SURE thresholding.

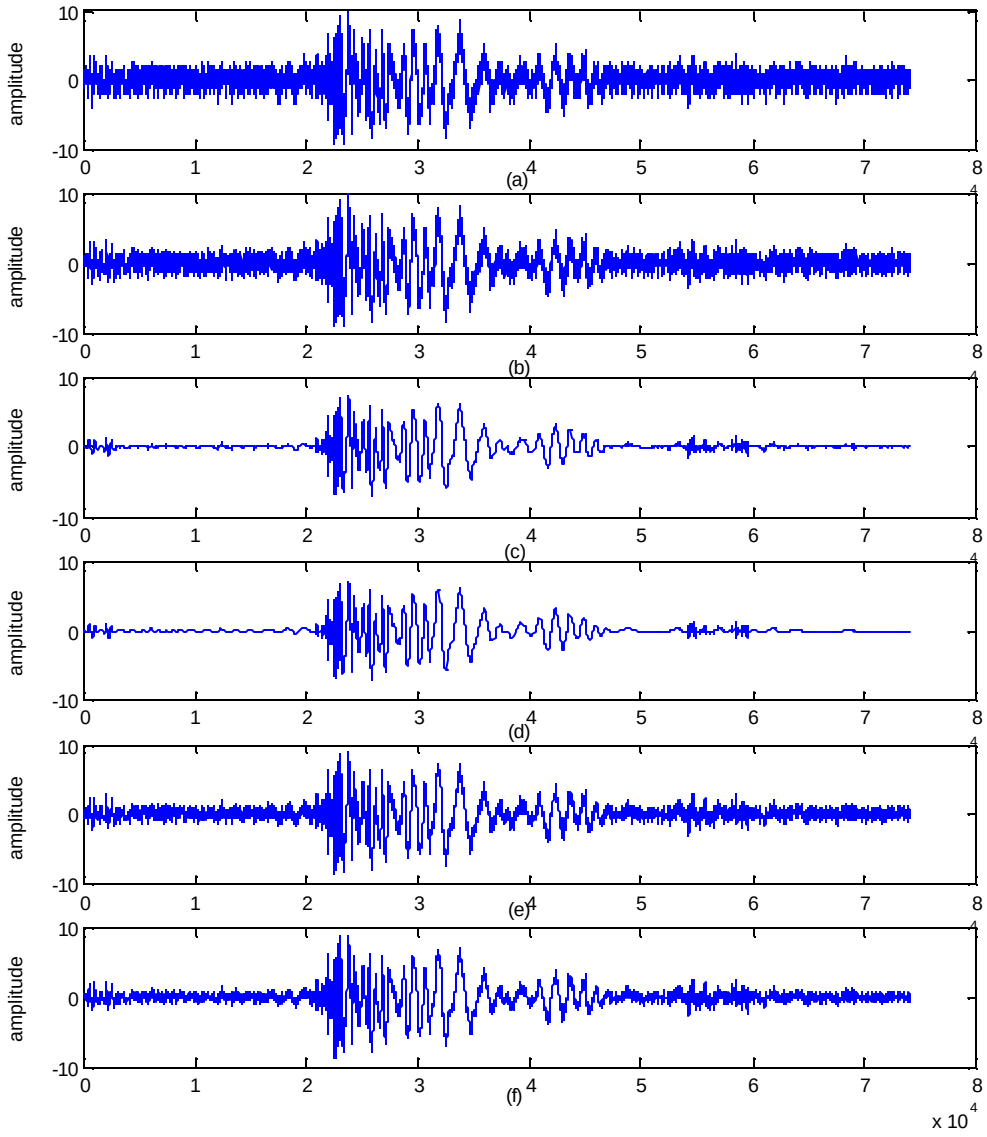


Figure 5.20: Denoising of Data segment A. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

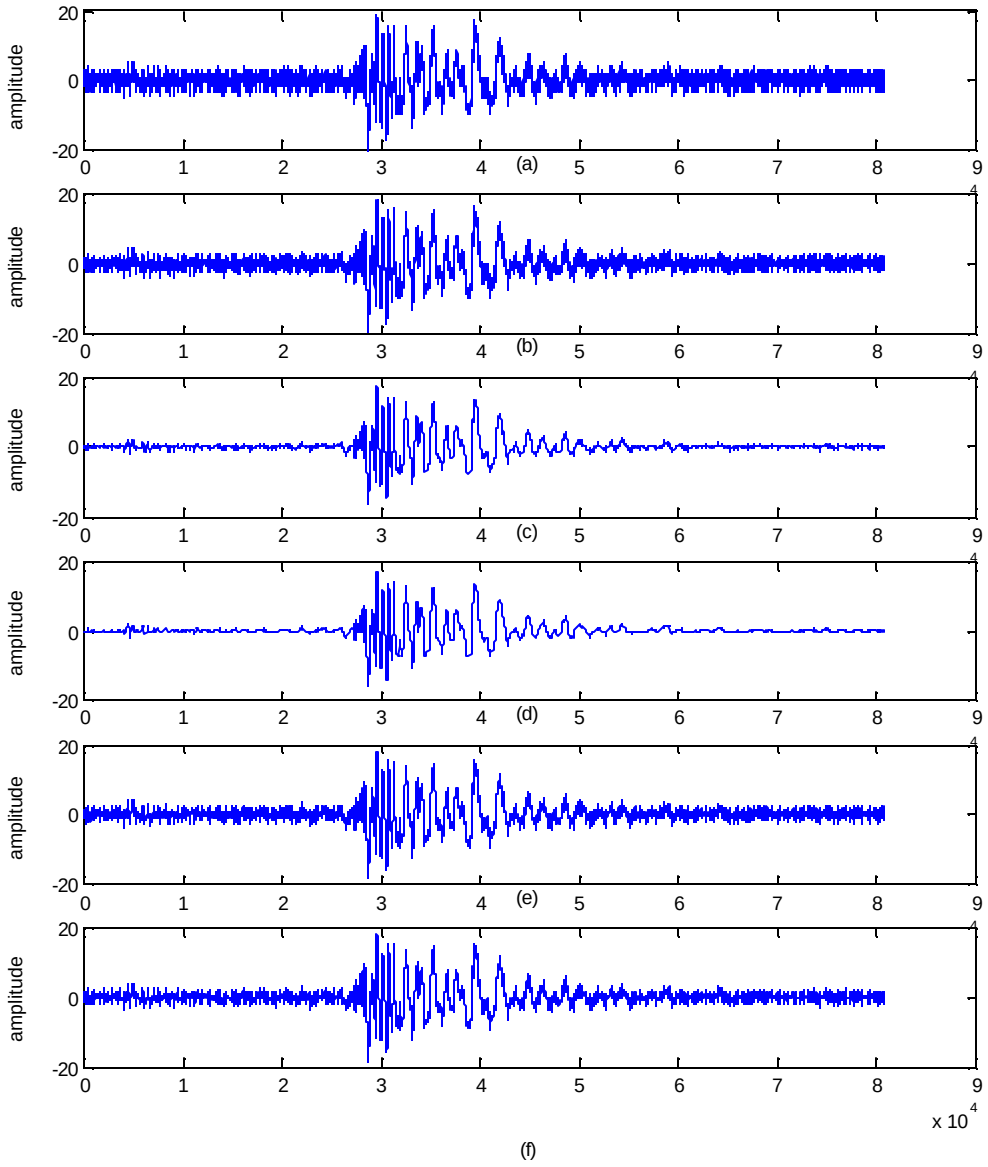


Figure 5.21: Denoising of Data segment B. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

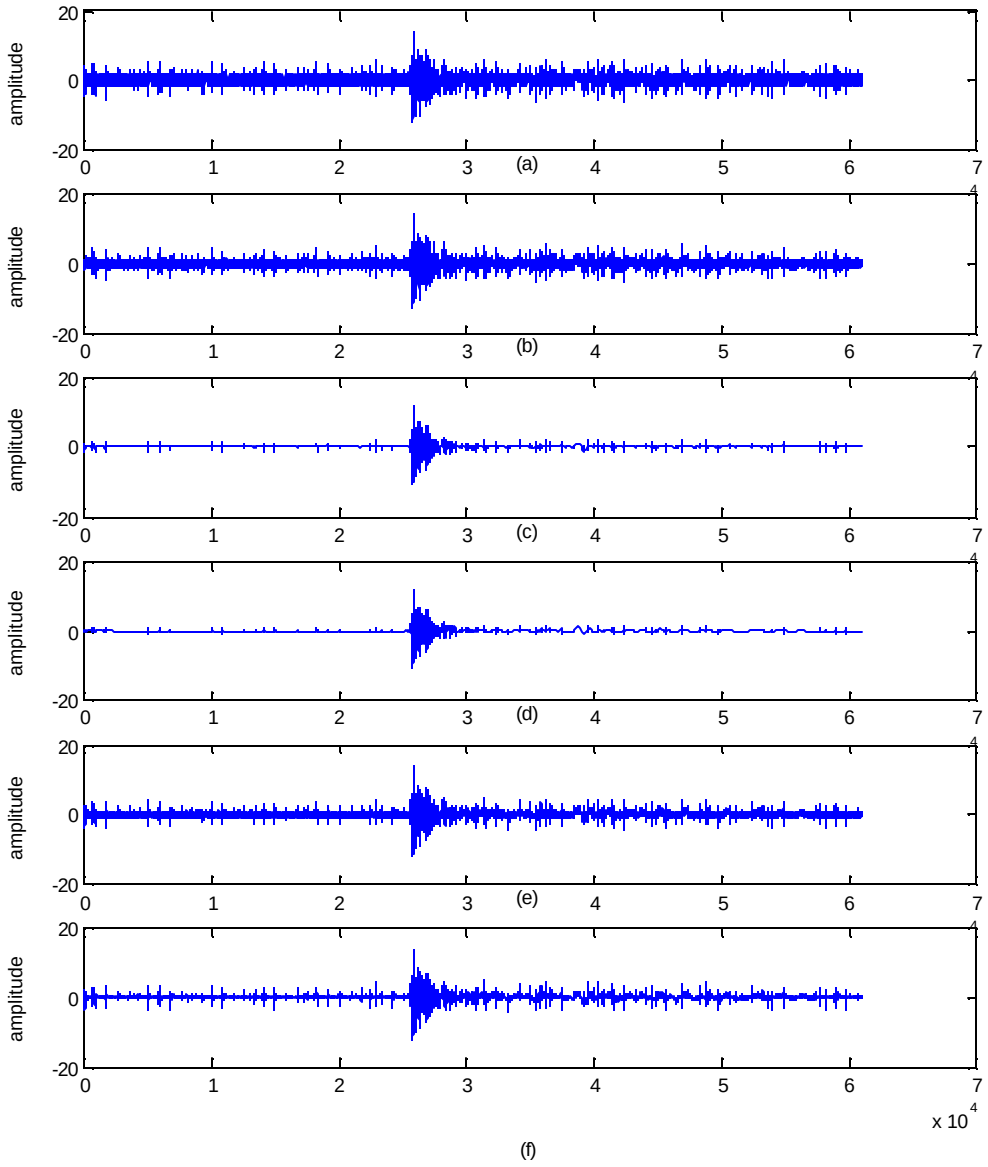


Figure 5.22: Denoising of Data segment C. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

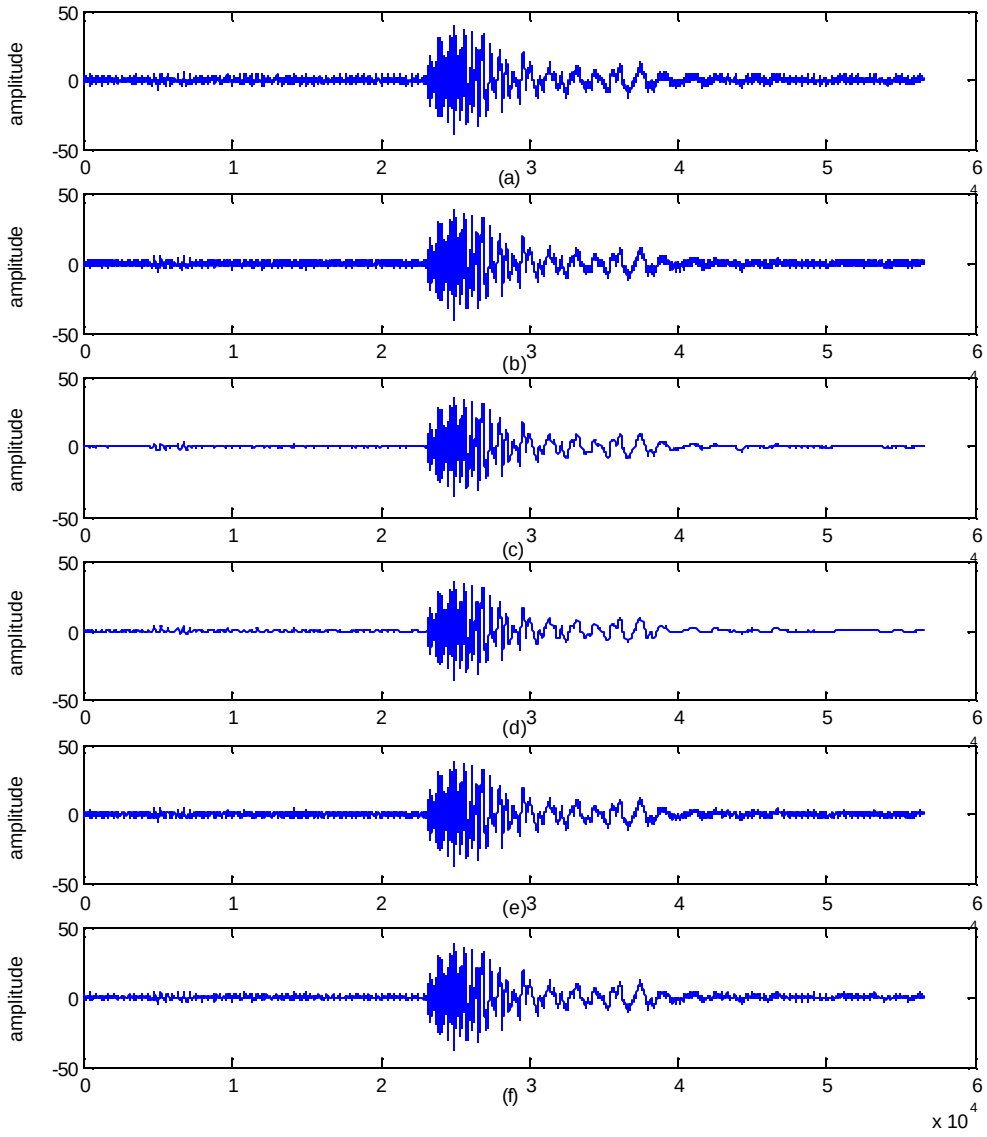


Figure 5.23: Denoising of Data segment D. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

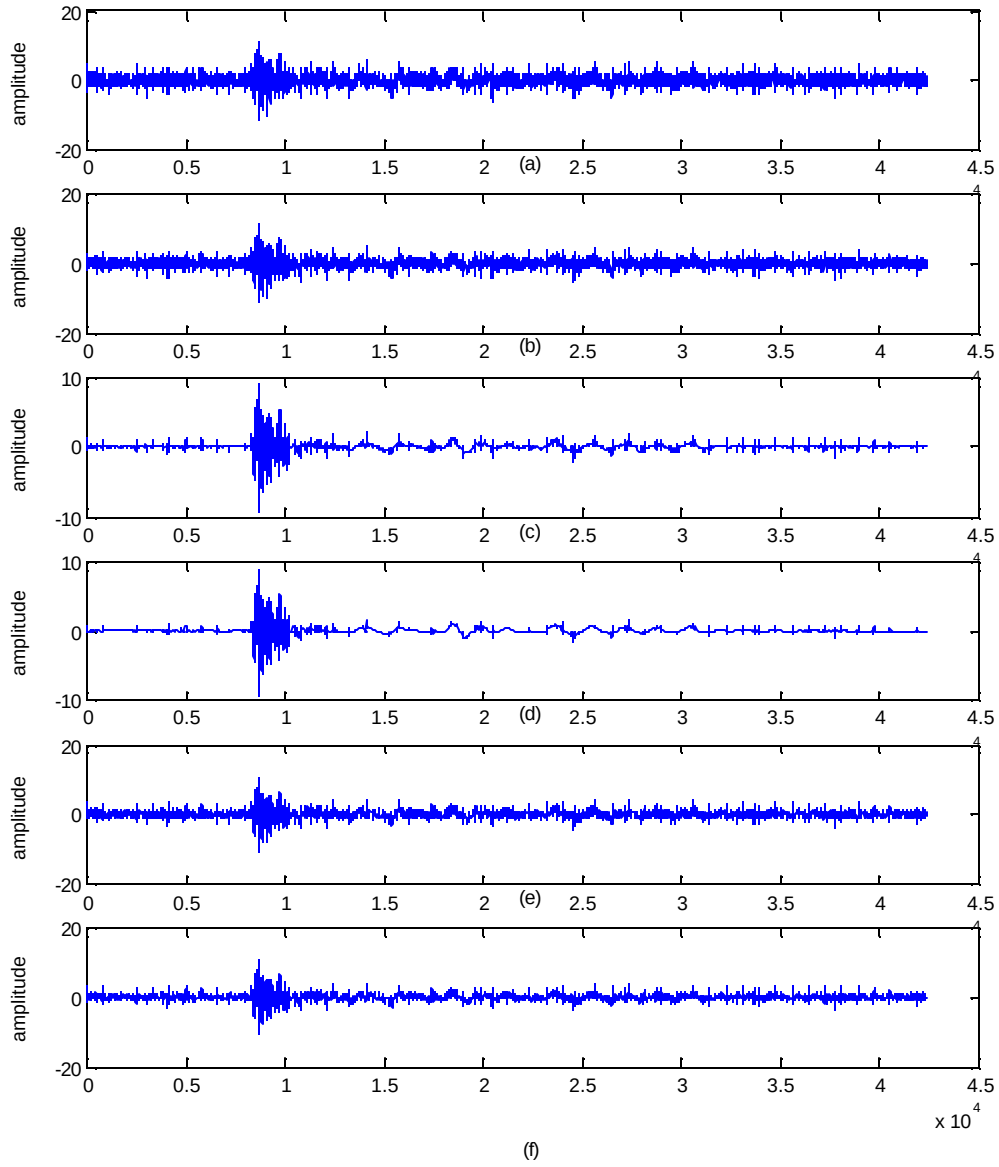


Figure 5.24: Denoising of Data segment E. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

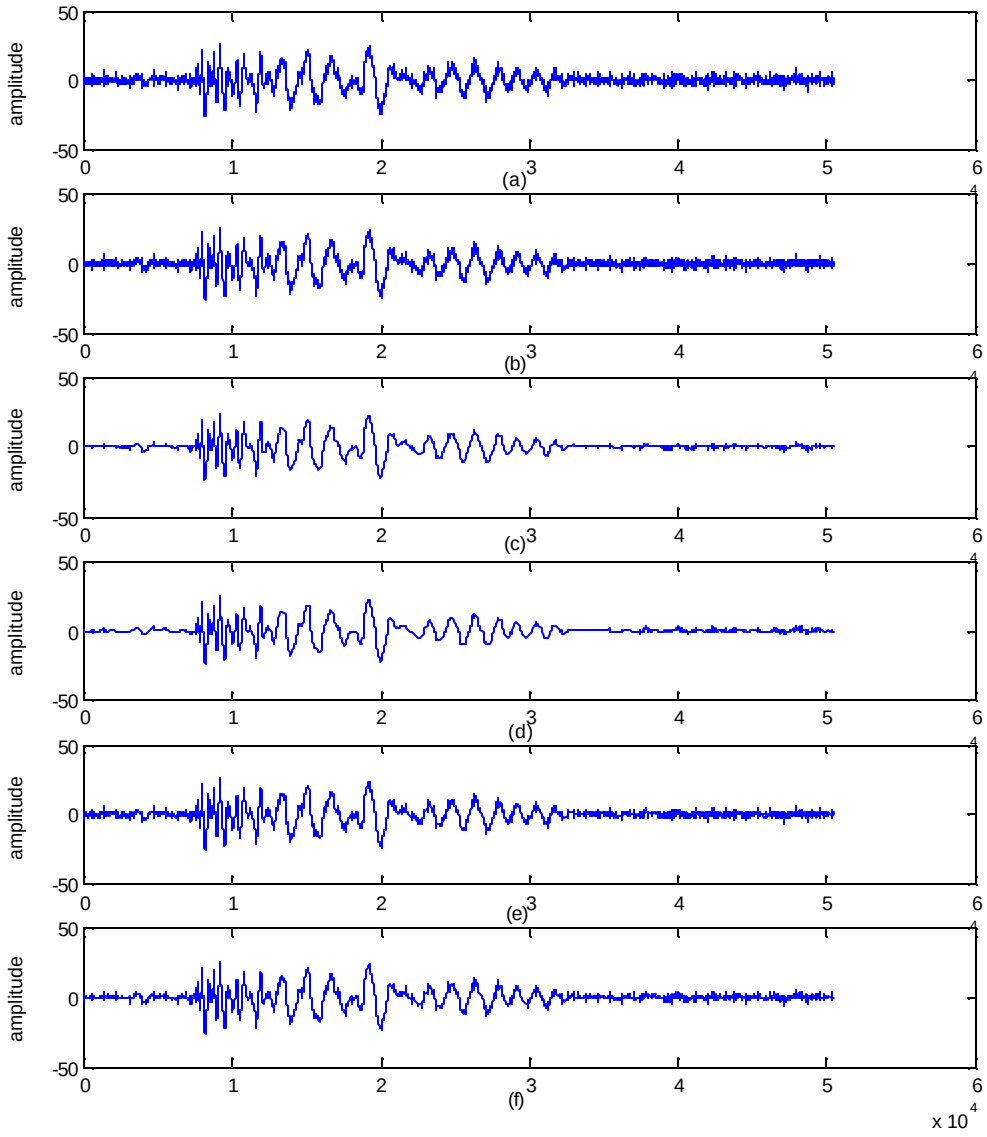


Figure 5.25: Denoising of Data segment F. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

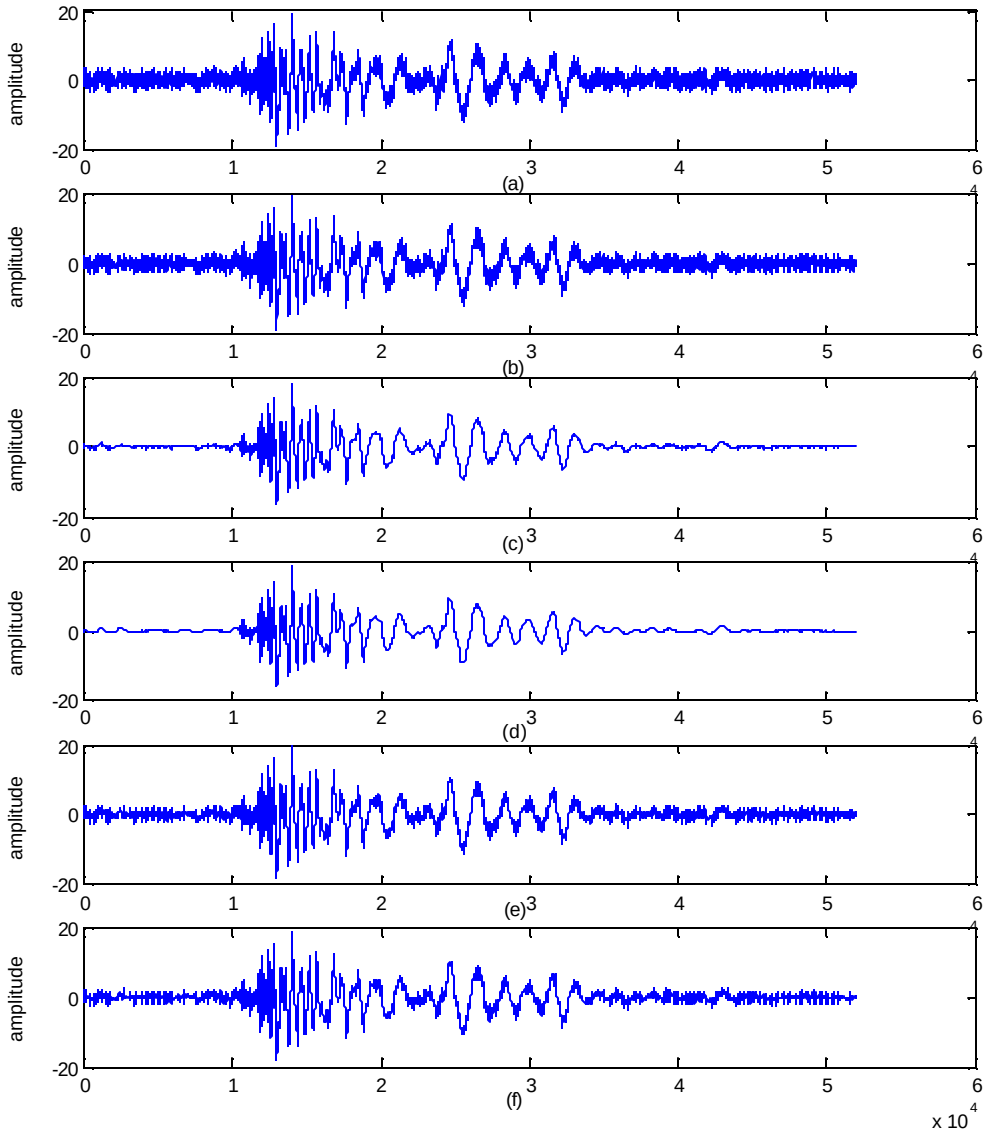


Figure 5.26: Denoising of Data segment G. (a) Data prior to denoising; (b) Time domain 3rd order Wiener Prediction with window size of 512; (c) Translation-invariant soft visual thresholding; (d) Translation-invariant recursive with one hard visual followed by one soft visual ; (e) Translation-invariant soft SURE thresholding; (f) Translation-invariant recursive with ten hard followed by one soft SURE thresholding.

VI. CONCLUSIONS

A. METHOD OF CHOICE

This work considered and compared the application of time-based and wavelet-based denoising schemes in a robust signaling environment. A time-based predictor was compared against time-based median filtering, and various wavelet-based techniques were considered along with a combination of time- and wavelet-based techniques. A systematic process was used to establish which would be the method of choice for the signals considered.

First, we compared time-based techniques with the orthonormal wavelet thresholding techniques. Results show that the soft SURE threshold scheme has best performance for the signal types and length considered. Results also show that overall the Wiener prediction scheme produced the best overall results for the two signal types considered. Note, however that Wiener predictor performance is window size dependent, and Wiener prediction performance became similar to those obtained with the other schemes for an appropriately chosen window size. In addition, the results show that the visual thresholding technique was inferior to the SURE thresholding schemes for small data sets. However, results also indicate that the visual threshold performs well for long data sets on the data considered in this study.

Second, we compared translation invariance soft SURE thresholding schemes. Results show that the soft SURE thresholding scheme produced the best results for the test signals considered. Results also show that better performances are obtained for random sinusoids than for the constant amplitude linear chirp.

Next, we compared the best thresholding schemes obtained for the orthonormal and translation-invariant wavelet domain against the combined schemes. We found that the translation-invariant soft SURE thresholding outperformed its orthonormal counterpart consistently over the range of SNR levels for both signal sets. In addition, Wiener prediction applied to the thresholded first level of decomposition detail coefficients just prior to reconstruction added slightly to the denoising ability for SNR (levels from two to

eight dB). However, results show no noticeable difference for our test data for the window size considered, which may be due to the larger data sample size.

Finally, we compared recursive wavelet-based techniques to the best performing orthonormal and translation-invariant schemes. We found that the recursive scheme combined with Wiener prediction outperformed other schemes for SNR levels between six and eight dB. Thus, results show that the signal power must be known *a priori* in order to implement such a technique. In addition, the Wiener prediction window size must be optimized for the sampling environment.

Our results suggest that the translation-invariant soft SURE thresholding schemes should be employed with a relatively small data segment and no *a priori* signal information. However, for larger data segments soft visual thresholding gives the greatest noise-free result, yet at the same time presents the greatest risk of loosing small amplitude signal components. In either case wavelet thresholding was more robust than time-based methods. Therefore, thresholding is indeed robust and highly capable denoising instrument. Soft SURE thresholding requires the least quantity of *a priori* signal information and is especially robust since techniques are available to account for colored noise environments.

B. RECOMMENDATION FOR FURTHER STUDY

Further study is needed in the area of recursive cycle-spinning and Wiener predictive denoising of coefficients. Recall that Wiener prediction was only performed on the detail coefficients at the first level of decomposition. It may be possible to implement an adaptive Wiener prediction scheme that chooses its window size based on statistics determined from a range of window sizes for each level of decomposition. In addition, it may also be possible to de-noise approximation coefficients. They were left untouched in our study.

The visual threshold provides the largest necessary threshold and thereby employs the greatest risk of signal loss, whereas the SURE threshold provides the minimum risk of signal loss. Recall that the Hybrid scheme chooses the smaller of the two thresholds

where the SURE threshold is based on evidence of signal presence and the visual threshold is based on the size of the segment. We leave for further work another version of the Hybrid scheme where the visual threshold is chosen if it is less than the SURE threshold, and an average of the two thresholds is used as the actual threshold when the SURE threshold is less than the visual threshold. Hence an average of these thresholds may provide a good compromise between the two schemes.

Recursive cycle-spinning was not optimized in this work, however the results combined with Wiener prediction showed promise. Therefore, optimization of the recursive cycle-spinning technique for each level of decomposition may be a viable area of study.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A MATLAB/WAVELAB SOURCE CODE

The *Matlab* [20] and *Wavelab* [10] source code developed and/or modified respectively for the purposes of this study are presented in this appendix.

A. MAIN PROGRAM (AVERAGES 100 ITERATIONS)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This Matlab Code performs averaging of 100 separate denoising
%   iterations for N wavelet decomposition levels .
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
%The following lines of code are used to create the constant
%Linear Chirp from Equation (2.3).  Commented out if Random Sine
%is performed.
%
        n=1:2048;
        F=1:.9765:2000;
        Fs=8000;
        %
         %Create Constant amplitude linear chirp from 0 to one fourth sampling
frequency
         %
        sig1=sin(2*pi.*F./Fs.*n);
        sig2=sig1/sqrt(mean(sig1.^2));
%
%Set median filter size
%
        medsize=3;
%
%Set Wiener Filter Order
%
        order=3;
%
%Set number of decomposition levels to be performed in wavelet routines
%
        %
        for N=[2 6 9];
        %
                for nnn=1:100
```

```
        %
        %Create random sinusoid to be used for this iteration nnn
        %
            %Random number between 0 and 1
            F=rand;
            n=1:2048;
            Fs=2;
        %
        %Create Random Sinusoid from 0 to one half sampling frequency
        %
            sig1=sin(2*pi*(F/Fs)*n);
            sig2=sig1/sqrt(mean(sig1.^2));
        %
        %Store snr and sigma from each run in Matrix form
        %
            [L,snr(nnn,:),sigma1(nnn,:),sigma2(nnn,:),sigma3(nnn,:),...
            sigma4(nnn,:),sigma5(nnn,:),sigma6(nnn,:),orthohybridwcoef_soft] =...
            OrthoDenoising(sig2,order,medsize,N);
        %
       end
 %
figure(N);
%
subplot(1,1,1),semilogy (mean(snr),mean(sigma1),'bo-')
          ,hold on, semilogy (mean(snr),mean(sigma2),'gx-')
          ,hold on,semilogy (mean(snr),mean(sigma3),'r+-')
          ,hold on,semilogy (mean(snr),mean(sigma4),'c*-')
          ,hold on,semilogy (mean(snr),mean(sigma5),'ms-')
          ,hold on,semilogy (mean(snr),mean(sigma6),'yd-')
          ,hold on,hold off;
          %
 legend(sprintf('Time Domain %srd order Wiener Prediction',num2str(order) ),...
sprintf('Time Domain Size %s Med Filter',num2str(medsize)),...
'Orthonormal Soft Visual Thresholding','Orthonormal Hard Visual
Thresholding',...
'Orthonormal Soft SURE Thresholding','Orthonormal Hard SURE Thresholding');

%
subplot(1,1,1),xlabel('SNR'),ylabel('Mean Squared Error');
%
end
```

B. DENOISING FUNCTION (ORTHONORMAL)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This Matlab Function calls wavelab and modified wavelab denoising
% functions [10]. Data is one row of normalized 1 dimensional data
% of dyadic length 2048=2^11.
% Function passes the signal through various
% Denoising Schemes and Passes back MSE vs SNR for each of those schemes
%
% Function Syntax as follows:
%
%     [noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6]
%         =OrthoDenoising(sig2,order,medsize,N)
%
% where:
%         "sig2" = True signal passed into function
%         "order" = Wiener Predictor Order
%         "medsize" = median filter size
%         "N" = Number of levels of decomposition for wavelet transform
%         "sigma1" = mean squared error of Time Domain Prediction
%         "sigma2" = mean squared error of Time Domain Size %s Med Filter
%         "sigma3" = mean squared error of Orthonormal Soft Visual thresholding
%         "sigma4" = mean squared error of Orthonormal Hard Visual thresholding
%         "sigma5" = mean squared error of Orthonormal Soft SURE thresholding
%         "sigma6" = mean squared error of Orthonormal Hard SURE thresholding
%         "noisearray" = SNR levels [-10 -6 -3 0 3 6 10] dB
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [noisearray, sigma1, sigma2, sigma3, sigma4, sigma5, sigma6]...
     =OrthoDenoising(sig2,order,medsize,N)
%
%Make a copy of signal
%
        sig1=sig2;
%
%Form array of chosen SNR's
%
        noisearray=[-10 -6 -3 0 3 6 10];
%
%create noise kernel
%
        noise = randn(1,length(sig1));
%
%find noise power
%
```

```
        noisepower=mean(noise.^2);
%
%normalize noise with noisepower
%
        noise = noise/sqrt(noisepower);%sets noise power to 1
%
%For loop for each SNR; Noise seed stays the same for each SNR,
%and signal power changed to meet requirements of noisearray.
%
for count=1:length(noisearray)
%
%Change the signal power assuming unit variance noise for desired SNR
%
        sig1forsnr=sqrt(10^(.1*noisearray(count)))*sig1;
%
%Add normalized noise with var=1, thereby generating SNR dictated
%
        s1 = noise + sig1forsnr ;
%
%Perform Time Domain Prediction Denoising
%
        [FilteredPredict] = predict_window_time(s1,order);
%
%Perform Time Domain Median filtering
%
        [FilteredMed] = medfilt1(s1,medsize);
%
%Perform Orthonormal Wavelet Denoising
%
        %
        %Create quadrature mirror filter for Translation-invariant Transform
        %
                QMF = MakeONFilter('Coiflet',5);
        %
        %Determine length of signal and number of dyads in Data using Wavelab routine
        %dyadlength [10]
        %
                [lengthofs1,Jdyads] = dyadlength(s1);
        %
        % Using number of dyads and number of levels of decomposition
         % find degree of coarsest scale.  ie. if N=6 and Jdyads=5; 11-6=5
        %
                L=Jdyads-N;
        %
        %Perform Orthonormal Transformations with Shrinkage using modified
        %Wavelab function WaveShrink [10].
```

```
        %
                [orthovisu_soft, orthovisuwcoef_soft]  = WaveShrink_soft(s1, 'Visu', L,
                QMF);
                [orthovisu_hard, orthovisuwcoef_hard]  = WaveShrink_hard(s1, 'Visu', L,
                QMF);
                [orthosure_soft, orthosurewcoef_soft]  = WaveShrink_soft(s1, 'SURE', L,
                QMF);
                [orthosure_hard, orthosurewcoef_hard]  = WaveShrink_hard(s1, 'SURE',
                L, QMF);
    %
    %Plot results of the various filter methods over the top of the noise free signal.
    %
        figure(N-1);
                subplot(6,1,1),plot(FilteredPredict(1:700),'r'),hold on, plot
                (sig1forsnr(1:700), 'b'), hold off;
                subplot(6,1,1),ylabel('Amplitude');
                subplot(6,1,2),plot(FilteredMed(1:700),'r'),hold on, plot
                (sig1forsnr(1:700), 'b'), hold off;
                subplot(6,1,2),ylabel('Amplitude'),title('(a)');
                subplot(6,1,3),plot(orthovisu_soft(1:700),'r'),hold on, plot
                (sig1forsnr(1:700),'b'),hold off;
                subplot(6,1,3),ylabel('Amplitude'),title('(b)');
                subplot(6,1,4),plot(orthovisu_hard(1:700),'r'),hold on, plot
                (sig1forsnr(1:700), 'b'),hold off;
                subplot(6,1,4),ylabel('Amplitude'),title('(c)');
                subplot(6,1,5),plot(orthosure_soft(1:700),'r'),hold on, plot
                (sig1forsnr(1:700), 'b'),hold off;
                subplot(6,1,5),ylabel('Amplitude') ,title('(d)');
                subplot(6,1,6),plot(orthosure_hard(1:700),'r'),hold on, plot
                (sig1forsnr(1:700),'b'),hold off;
                subplot(6,1,6),ylabel('Amplitude'), xlabel('(f)'),title('(e)');
%
% Perform Mean Squared Error Calculation.
%
        %Time Domain Prediction
        [sigma1(count)] = msemed(FilteredPredict,sig1forsnr);
        %Median filtered
        [sigma2(count)] = msemed(FilteredMed,sig1forsnr);
         %Orthonormal Wavelet Domain Visual Soft
        [sigma3(count)] = msemed(orthovisu_soft,sig1forsnr);
        % Orthonormal Wavelet Domain Visual Hard
        [sigma4(count)] = msemed(orthovisu_hard,sig1forsnr);
        % Orthonormal Wavelet Domain SURE Soft
        [sigma5(count)] = msemed(orthosure_soft,sig1forsnr);
        % Orthonormal Wavelet Domain SURE Hard
        [sigma6(count)] = msemed(orthosure_hard,sig1forsnr);
```

```
        %
end
```

C. DENOISING (TRANSLATION-INVARIANT)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This Matlab Function calls wavelab and modified wavelab denoising
%   functions [10]. Data is one row of normalized 1 dimensional data
%   of dyadic length 2048=2^11.
%   Function passes the signal through various
%   Denoising Schemes and Passes back MSE vs SNR for each of those schemes
%
%   Function Syntax as follows:
%
%      [noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6]
%          =TransDenoising(sig2,order,medsize,N)
%
%   where:
%          "sig2" = True signal passed into function
%          "order" = Wiener Predictor Order
%          "medsize" = median filter size
%          "N" = Number of levels of decomposition for wavelet transform
%          "noisearray" = SNR levels [-10 -6 -3 0 3 6 10] dB
%          "sigma1" = mean squared error of Time Domain Prediction
%          "sigma2" = mean squared error of Time Domain Med Filter
%          "sigma3" = mean squared error of Translation-invariant Soft Visual thresholding
%          "sigma4" = mean squared error of Translation-invariant Hard Visual
%                         thresholding
%          "sigma5" = mean squared error of Translation-invariant Soft SURE thresholding
%          "sigma6" = mean squared error of Translation-invariant Hard SURE
%                         thresholding
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6] =...
          TransDenoising(sig2,order,medsize,N)
%
%Make a copy of signal
%
        sig1=sig2;
%
%Form array of chosen SNR's
%
        noisearray=[-10 -6 -3 0 3 6 10];
%
```

```
%create noise kernel
%
        noise = randn(1,length(sig1));
%
%find noise power
%
        noisepower=mean(noise.^2);
%
%normalize noise with noisepower
%
        noise = noise/sqrt(noisepower);%sets noise power to 1
%
%For loop for each SNR; Noise seed stays the same for each SNR,
%and signal power changed to meet requirements of noisearray.
%
for count=1:length(noisearray)
%
%Change the signal power assuming unit variance noise for desired SNR
%
        sig1forsnr=sqrt(10^(.1*noisearray(count)))*sig1;
%
%Add normalized noise with var=1, thereby generating SNR dictated
%
        s1 = noise + sig1forsnr ;
%
%Perform Time Domain Prediction Denoising
%
        [FilteredPredict] = predict_window_time(s1,order);
%
%Perform Time Domain Median filtering
%
        [FilteredMed] = medfilt1(s1,medsize);
%
%Perform Translation Wavelet Denoising
%
        %
        %Create quadrature mirror filter for Translation-invariant Transform
        %
                QMF = MakeONFilter('Coiflet',5);
        %
        %Determine length of signal and number of dyads in Data using Wavelab routine
        %dyadlength [10]
        %
                [lengthofs1,Jdyads] = dyadlength(s1);
        %
        % Using number of dyads and number of levels of decomposition
```

```
        % find degree of coarsest scale.  ie. if N=6 and Jdyads=5; 11-6=5
        %
                L=Jdyads-N;
        %
        %Perform Translation-invariant Transformations with Shrinkage using modified
        %Wavelab function FWT_TI [10].
        %
                [transvisuwcoef_soft]=FWT_TI_visuthreshSoft(s1,L,QMF);
                [transvisuwcoef_hard]=FWT_TI_visuthreshHard(s1,L,QMF);
                [transsurewcoef_soft]=FWT_TI_SurethreshSoft(s1,L,QMF);
                [transsurewcoef_hard]=FWT_TI_SurethreshHard(s1,L,QMF);
        %
        %Perform Inverse Translation-invariant Transform with Wavelab function
        %IWT_TI [10]
        %
                transvisu_soft=IWT_TI(transvisuwcoef_soft,QMF);
                transvisu_hard=IWT_TI(transvisuwcoef_hard,QMF);
                transsure_soft=IWT_TI(transsurewcoef_soft,QMF);
                transsure_hard=IWT_TI(transsurewcoef_hard,QMF);
%
%Plot results of various filter methods over the top of the noise free signal.
%
        figure(N-1);
        subplot(6,1,1),plot(FilteredPredict(1:700),'r'),hold on, plot (sig1forsnr(1:700),
        'b'),hold off;
        subplot(6,1,1),ylabel('Amplitude');
        subplot(6,1,2),plot(FilteredMed(1:700),'r'),hold on, plot (sig1forsnr(1:700),
        'b'),hold off;
        subplot(6,1,2),ylabel('Amplitude'),title('(a)');
        subplot(6,1,3),plot(transvisu_soft(1:700),'r'),hold on, plot (sig1forsnr(1:700),
        'b'),hold off;
        subplot(6,1,3),ylabel('Amplitude'),title('(b)');
        subplot(6,1,4),plot(transvisu_hard(1:700),'r'),hold on, plot
        (sig1forsnr(1:700),'b'),hold off;
        subplot(6,1,4),ylabel('Amplitude'),title('(c)');
        subplot(6,1,5),plot(transsure_soft(1:700),'r'),hold on, plot (sig1forsnr(1:700),
        'b'),hold off;
        subplot(6,1,5),ylabel('Amplitude') ,title('(d)');
        subplot(6,1,6),plot(transsure_hard(1:700),'r'),hold on, plot (sig1forsnr(1:700),
        'b'),hold off;
        subplot(6,1,6),ylabel('Amplitude'), xlabel('(f)'),title('(e)');
%
% Perform Mean Squared Error Calculation.
%
        %Time Domain Prediction
        [sigma1(count)] = msemed(FilteredPredict,sig1forsnr);
```

```
        %Translation-invariant Wavelet Domain sure
        [sigma2(count)] = msemed(FilteredMed,sig1forsnr);
        %Translation-invariant Wavelet Domain sure
        [sigma3(count)] = msemed(transvisu_soft,sig1forsnr);
        %Translation-invariant Wavelet Domain Universal
        [sigma4(count)] = msemed(transvisu_hard,sig1forsnr);
        %Translation-invariant Wavelet Domain Universal
        [sigma5(count)] = msemed(transsure_soft,sig1forsnr);
        %Time domain Med Filtered
        [sigma6(count)] = msemed(transsure_hard,sig1forsnr);
%
end
```

D. DENOISING (COMBINED TRANSLATION-INVARIANT AND WIENER PREDICTION)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This Matlab Function calls wavelab and modified wavelab denoising
%   functions [10]. Data is one row of normalized 1 dimensional data
%   of dyadic length 2048=2^11.
%   Function passes the signal through various
%   Denoising Schemes and Passes back MSE vs SNR for each of those schemes
%
%   Function Syntax as follows:
%
%      [noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6]
%         =CombinedDenoising(sig2,order,medsize,N)
%
%   where:
%         "sig2" = True signal passed into function
%         "order" = Wiener Predictor Order
%         "medsize" = median filter size
%         "N" = Number of levels of decomposition for wavelet transform
%                        "noisearray" = SNR levels [-10 -6 -3 0 3 6 10] dB
%         "sigma1" = mean squared error of Time Domain Prediction
%         "sigma2" = mean squared error of Time Domain Med Filter
%         "sigma3" = mean squared error of Orthonormal Wavelet Domain sure
%         "sigma4" = mean squared error of Translation-invariant Soft SURE thresholding
%         "sigma5" = mean squared error of Translation-invariant Soft SURE thresholding
%                with Wiener Prediction on 1st level decomposition detail coeffs
%         "sigma6" = mean squared error of Translation-invariant Hard SURE
thresholding
%                with Median Filtering on 1st level decomposition detail coeffs
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
function    [noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6] =...
             CombinedDenoising(sig2,order,medsize,N)
%
%Make a copy of signal
%
        sig1=sig2;
%
%Form array of chosen SNR's
%
         noisearray=[-10 -6 -3 0 3 6 10];
%
%create noise kernel
%
        noise = randn(1,length(sig1));
%
%find noise power
%
        noisepower=mean(noise.^2);
%
%normalize noise with noisepower
%
        noise = noise/sqrt(noisepower);%sets noise power to 1
%
%For loop for each SNR; Noise seed stays the same for each SNR,
%and signal power changed to meet requirements of noisearray.
%
for count=1:length(noisearray)
%
%Change the signal power assuming unit variance noise for desired SNR
%
        sig1forsnr=sqrt(10^(.1*noisearray(count)))*sig1;
%
%Add normalized noise with var=1, thereby generating SNR dictated
%
        s1 = noise + sig1forsnr ;
%
%Perform Time Domain Prediction Denoising
%
        [FilteredPredict] = predict_window_time(s1,order);
%
%Perform Time Domain Median filtering
%
        [FilteredMed] = medfilt1(s1,medsize);
%
%Perform Translation Wavelet Denoising
%
```

```
%
%Create quadrature mirror filter for Translation-invariant Transform
%
        QMF = MakeONFilter('Coiflet',5);
%
%Determine length of signal and number of dyads in Data using Wavelab routine
%dyadlength [10]
        [lengthofs1,Jdyads] = dyadlength(s1);
%
% Using number of dyads and number of levels of decomposition
% find degree of coarsest scale.  ie. if N=6 and Jdyads=5; 11-6=5
%
        L=Jdyads-N;
%
%Perform Orthonormal,Translation-invariant, and combined Transformations
%with Shrinkage using modifification of Wavelab function FWT_TI [10],
%and our prediction
%function.
%
%
%Perform Orthonormal Transformations with SURE Shrinkage using
%modifification of
%Wavelab function WaveShrink [10].
%
[orthosure_soft, orthosurewcoef_soft]  = WaveShrink_soft(s1,'SURE',L,QMF);
%
%Perform Translation-invariant Transformations with Shrinkage using modified
%Wavelab function FWT_TI [10].
%
[transsurewcoef_soft]=FWT_TI_SurethreshSoft(s1,L,QMF);
[transsurewcoef_soft_w]=FWT_TI_SurethreshSoft_weiner(s1,L,QMF,order);
[transsurewcoef_soft_m]=FWT_TI_SurethreshSoft_medfilt(s1,L,QMF);
%
%Perform Inverse Translation-invariant Transform with Wavelab function
%IWT_TI [10]
%
transsure_soft=IWT_TI(transsurewcoef_soft,QMF);
transsure_soft_w=IWT_TI(transsurewcoef_soft_w,QMF);
transsure_soft_m=IWT_TI(transsurewcoef_soft_m,QMF);
%
%Plot results of various filter methods over the top of the noise free signal.
%
figure(N-1);
subplot(6,1,1),plot(FilteredPredict(1:700),'r'),hold on, plot (sig1forsnr(1:700),...
'b'),hold off;
subplot(6,1,1),ylabel('Amplitude');
```

```
        subplot(6,1,2),plot(FilteredMed(1:700),'r'),hold on, plot (sig1forsnr(1:700),...
        'b'),hold off;
        subplot(6,1,2),ylabel('Amplitude'),title('(a)');
        subplot(6,1,3),plot(orthosure_soft(1:700),'r'),hold on, plot (sig1forsnr(1:700),...
        'b'),hold off;
        subplot(6,1,3),ylabel('Amplitude'),title('(b)');
        subplot(6,1,4),plot(transsure_soft(1:700),'r'),hold on, plot (sig1forsnr(1:700),...
        'b'),hold off;
        subplot(6,1,4),ylabel('Amplitude'),title('(c)');
        subplot(6,1,5),plot(transsure_soft_w(1:700),'r'),hold on, plot (sig1forsnr(1:700),...
        'b'),hold off;
        subplot(6,1,5),ylabel('Amplitude') ,title('(d)');
        subplot(6,1,6),plot(transsure_soft_m(1:700),'r'),hold on, plot (sig1forsnr(1:700),...
        'b'),hold off;
        subplot(6,1,6),ylabel('Amplitude'), xlabel('(f)'),title('(e)');
%
% Perform Mean Squared Error Calculation.
%
        %Time Domain Prediction
        [sigma1(count)] = msemed(FilteredPredict,sig1forsnr);
        %Time domain Med Filtered
        [sigma2(count)] = msemed(FilteredMed,sig1forsnr);
        %Orthonormal Wavelet Domain sure
        [sigma3(count)] = msemed(orthosure_soft,sig1forsnr);
        %Translation-invariant Soft SURE thresholding
        [sigma4(count)] = msemed(transsure_soft,sig1forsnr);
        %Translation-invariant Soft SURE thresholding with Wiener Prediction
        [sigma5(count)] = msemed(transsure_soft_w,sig1forsnr);
        %Translation-invariant Soft SURE thresholding with Medfilter
        [sigma6(count)] = msemed(transsure_soft_m,sig1forsnr);
%
end
```

E. DENOISING (RECURSIVE, SURE SOFT TRANSLATION-INVARIANT, AND COMBINED)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This Matlab Function calls wavelab and modified wavelab denoising
%   functions [10]. Data is one row of normalized 1 dimensional data
%   of dyadic length 2048=2^11.
%   Function passes the signal through various
%   Denoising Schemes and Passes back MSE vs SNR for each of those schemes
%
%   Function Syntax as follows:
%
```

```
%       [noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6]
%           =Recursive(sig2,order,medsize,N)
%
%   where:
%           "sig2" = True signal passed into function
%           "order" = Wiener Predictor Order
%           "medsize" = median filter size
%           "N" = Number of levels of decomposition for wavelet transform
%                          "noisearray" = SNR levels [-10 -6 -3 0 3 6 10] dB
%           "sigma1" = mean squared error of Time Domain Prediction
%           "sigma2" = mean squared error of Orthonormal Wavelet Domain SURE Soft
%           "sigma3" = mean squared error of Translation-invariant Soft SURE thresholding
%           "sigma4" = mean squared error of Translation-invariant Soft SURE thresholding
%                   with Wiener Prediction
%           "sigma5" = mean squared error of Recursive cycle-spinning 10 hard SURE
%                   Translation-invariant followed by 1 soft SURE Translation-invariant
%           "sigma6"= mean squared error of Recursive cycle-spinning 10 hard SURE
%                   Translation-invariant followed by 1 soft SURE Translation-invariant
%                   with Wiener Prediction on 1st level decomposition detail coeffs
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [L,noisearray,sigma1,sigma2,sigma3,sigma4,sigma5,sigma6] =...
             Recursive(sig2,order,medsize,N)
%
%Make a copy of signal
%
   sig1=sig2;
%
%Form array of chosen SNR's
%
   noisearray=[-10 -6 -3 0 3 6 10];
%
%create noise kernel
%
   noise = randn(1,length(sig1));
%
%find noise power
%
   noisepower=mean(noise.^2);
%
%normalize noise with noisepower
%
         noise = noise/sqrt(noisepower);%sets noise power to 1
%
%For loop for each SNR; Noise seed stays the same for each SNR,
```

```
%and signal power changed to meet requirements of noisearray.
%
for count=1:length(noisearray)
%
%Change the signal power assuming unit variance noise for desired SNR
%
        sig1forsnr=sqrt(10^(.1*noisearray(count)))*sig1;
%
%Add normalized noise with var=1, thereby generating SNR dictated
%
        s1 = noise + sig1forsnr ;
%
%Perform Time Domain Prediction Denoising
%
        [FilteredPredict] = predict_window_time(s1,order);
%
%Perform Time Domain Median filtering
%
        [FilteredMed] = medfilt1(s1,medsize);
%
%Perform Translation Wavelet Denoising
%
        %
        %Create quadrature mirror filter for Translation-invariant Transform
        %
                QMF = MakeONFilter('Coiflet',5);
        %
        %Determine length of signal and number of dyads in Data using Wavelab routine
        %dyadlength [10]
                [lengthofs1,Jdyads] = dyadlength(s1);
        %
        % Using number of dyads and number of levels of decomposition
        % find degree of coarsest scale.  ie. if N=6 and Jdyads=5; 11-6=5
        %
                L=Jdyads-N;
        %
        %Perform Orthonormal,Translation-invariant, and combined Transformations
        %with Shrinkage using modifification of Wavelab function FWT_TI [10], and
our prediction
        %function.
        %
        %
        %Perform Orthonormal Transformations with Shrinkage using modified
        %Wavelab function WaveShrink [10].
        %
```

```
        [orthosure_soft, orthosurewcoef_soft]  =
WaveShrink_soft(s1,'SURE',L,QMF);
    %
    %Perform Translation-invariant Transformations with Shrinkage using modified
    %Wavelab function FWT_TI [10].
    %
        [transsurewcoef_soft]=FWT_TI_SurethreshSoft(s1,L,QMF);

    [transsurewcoef_soft_w]=FWT_TI_SurethreshSoft_weiner(s1,L,QMF,order);
    %
    %Perform Inverse Translation-invariant Transform with Wavelab function
IWT_TI [10]
    %
        transsure_soft=IWT_TI(transsurewcoef_soft,QMF);
        transsure_soft_w=IWT_TI(transsurewcoef_soft_w,QMF);
    %
    %Perform Recursive Cycle-spinning with Wavelab function IWT_TI [10]
    %
    transsure_hard_r=s1;
    for n=1:10;
        %
        %Perform Translation-invariant Transformations with Shrinkage using
modified
        %Wavelab function FWT_TI [10].
        %
            [transsurewcoef]=FWT_TI_Surethresh(transsure_hard_r,L,QMF);
        %
        %Perform Inverse Translation-invariant Transform with Wavelab function
IWT_TI [10]
        %
        transsure_hard_r=IWT_TI(transsurewcoef,QMF);
        %
  end
    %
    %
    %Perform Translation-invariant Transformations with Shrinkage using modified
    %Wavelab function FWT_TI [10].
    %
        [transsurewcoef]=FWT_TI_SurethreshSoft(transsure_hard_r,L,QMF);

    [transsurewcoef_wiener]=FWT_TI_SurethreshSoft_weiner(transsure_hard_r,L,Q
MF,order);
    %
    %Perform Inverse Translation-invariant Transform with Wavelab function
IWT_TI [10]
    %
```

```
        transsure_hard_r=IWT_TI(transsurewcoef,QMF);
        transsure_hard_wr=IWT_TI(transsurewcoef_wiener,QMF);
    %
    %Plot results of various filter methods over the top of the noise free signal.
    %
            figure(N-1);
            subplot(6,1,1),plot(FilteredPredict(1:700),'r'),hold on, plot
(sig1forsnr(1:700),'b'),hold off;
            subplot(6,1,1),ylabel('Amplitude');
            subplot(6,1,2),plot(orthosure_soft(1:700),'r'),hold on, plot
(sig1forsnr(1:700),'b'),hold off;
            subplot(6,1,2),ylabel('Amplitude'),title('(a)');
            subplot(6,1,3),plot(transsure_soft(1:700),'r'),hold on, plot
(sig1forsnr(1:700),'b'),hold off;
            subplot(6,1,3),ylabel('Amplitude'),title('(b)');
            subplot(6,1,4),plot(transsure_soft_w(1:700),'r'),hold on, plot
(sig1forsnr(1:700),'b'),hold off;
            subplot(6,1,4),ylabel('Amplitude'),title('(c)');
            subplot(6,1,5),plot(transsure_hard_r(1:700),'r'),hold on, plot
(sig1forsnr(1:700),'b'),hold off;
            subplot(6,1,5),ylabel('Amplitude') ,title('(d)');
            subplot(6,1,6),plot(transsure_hard_wr(1:700),'r'),hold on, plot
(sig1forsnr(1:700),'b'),hold off;
            subplot(6,1,6),ylabel('Amplitude'), xlabel('(f)'),title('(e)');

            %
            % Perform Mean Squared Error Calculation.
            %
                    %Time Domain Prediction
                    [sigma1(count)] = msemed(FilteredPredict,sig1forsnr);
                    %Orthonormal Wavelet Domain SURE Soft
                    [sigma2(count)] = msemed(orthosure_soft,sig1forsnr);
                    %Translation-invariant Soft SURE thresholding
                    [sigma3(count)] = msemed(transsure_soft,sig1forsnr);
                    %Translation-invariant Soft SURE thresholding with Wiener Prediction
                    [sigma4(count)] = msemed(transsure_soft_w,sig1forsnr);
        %Recursive cycle-spinning 10 hard followed by 1 soft
        [sigma5(count)] = msemed(transsure_hard_r,sig1forsnr);
                    %Recursive cycle-spinning with Wiener Prediction
                    [sigma6(count)] = msemed(transsure_hard_wr,sig1forsnr);
%
end
```

F. MODIFIED WAVELAB FUNCTIONS

1. Orthonormal

a. Modified Waveshrink; Waveshrink_hard.[10]

```
%Modified version of Waveshrink from Wavelab [10].
function [xh, wcoef]  = WaveShrink_hard(y,type,L,qmf)
% WaveShrink -- Soft Threshold Shrinkage Applied to Wavelet
%Coefficients
%  Usage
%    [xh,xwh] = WaveShrink(y,type,L,qmf)
%  Inputs
%    y      1-d signal. length(y)= 2^J
%             Normalized to noise level 1! (See NoiseNorm)
%    type   string. Type of shrinkage applied:
%             'Visu','SURE','Hybrid','MinMax','MAD'
%             Optional; default == 'Visu'
%    L      Low-Frequency cutoff for shrinkage (e.g. L=4)
%             Should have L << J!
%    qmf    Quadrature Mirror Filter for Wavelet Transform
%             Optional, Default = Symmlet 8.
%  Outputs
%    xh     estimate, obtained by applying soft thresholding on
%           wavelet coefficients
%    xwh    Wavelet Transform of estimate
%
%  Description
%    WaveShrink smooths noisy data presumed to have noise level 1
%    by transforming it into the wavelet domain, applying soft
%    thresholding to the wavelet coefficients and inverse transforming.
%
%    The theory underlying these methods is described in a variety of
%    papers by D.L. Donoho and I.M. Johnstone.
%
%    The different methods of selecting thresholds are detailed
%    in their articles.
%
%  See Also
%       FWT_PO, IWT_PO, MakeONFilter, NoiseNorm, RigorShrink
%
  if nargin < 2,
     type = 'Visu';
  end
  if nargin < 3,
     L = 3;
  end
  if nargin < 4,
     qmf = MakeONFilter('Symmlet',8);
  end
%
```

```
 [n,J] = dyadlength(y) ;
 wcoef = FWT_PO(y,L,qmf) ;
%
 if     strcmp(type,'Visu'),
        wcoef((2^(L)+1):n) = VisuThresh(wcoef((2^(L)+1):n),'hard') ;
 elseif strcmp(type,'SURE'),
        wcoef = MultiSURE(wcoef,L);
 elseif strcmp(type,'Hybrid'),
        wcoef = MultiHybridhard(wcoef,L);
 elseif strcmp(type,'MinMax'),
        wcoef((2^(L)+1):n) = MinMaxThresh(wcoef((2^(L)+1):n)) ;
 elseif strcmp(type,'MAD'),
        wcoef = MultiMAD(wcoef,L);
 end
%
 xh    = IWT_PO(wcoef,L,qmf);

%
% Copyright (c) 1993-5.  Jonathan Buckheit, David Donoho and Iain
Johnstone
%



%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

b. Modified Waveshrink; Waveshrink_soft[10].

```
function [xh, wcoef]  = WaveShrink_soft(y,type,L,qmf)
% WaveShrink -- Soft Threshold Shrinkage Applied to Wavelet
%Coefficients
%  Usage
%    [xh,xwh] = WaveShrink(y,type,L,qmf)
%  Inputs
%    y      1-d signal. length(y)= 2^J
%             Normalized to noise level 1! (See NoiseNorm)
%    type   string. Type of shrinkage applied:
%             'Visu','SURE','Hybrid','MinMax','MAD'
%             Optional; default == 'Visu'
%    L      Low-Frequency cutoff for shrinkage (e.g. L=4)
%             Should have L << J!
%    qmf    Quadrature Mirror Filter for Wavelet Transform
```

```
%               Optional, Default = Symmlet 8.
%  Outputs
%    xh     estimate, obtained by applying soft thresholding on
%           wavelet coefficients
%    xwh    Wavelet Transform of estimate
%
%  Description
%    WaveShrink smooths noisy data presumed to have noise level 1
%    by transforming it into the wavelet domain, applying soft
%    thresholding to the wavelet coefficients and inverse transforming.
%
%    The theory underlying these methods is described in a variety of
%    papers by D.L. Donoho and I.M. Johnstone.
%
%    The different methods of selecting thresholds are detailed
%    in their articles.
%
%  See Also
%        FWT_PO, IWT_PO, MakeONFilter, NoiseNorm, RigorShrink
%
  if nargin < 2,
     type = 'Visu';
  end
  if nargin < 3,
     L = 3;
  end
  if nargin < 4,
     qmf = MakeONFilter('Symmlet',8);
  end
%
  [n,J] = dyadlength(y) ;
  wcoef = FWT_PO(y,L,qmf) ;
%
  if     strcmp(type,'Visu'),
         wcoef((2^(L)+1):n) = VisuThresh(wcoef((2^(L)+1):n)) ;
  elseif strcmp(type,'SURE'),
         wcoef = MultiSUREsoft(wcoef,L);
  elseif strcmp(type,'Hybrid'),
         wcoef = MultiHybrid(wcoef,L);
  elseif strcmp(type,'MinMax'),
         wcoef((2^(L)+1):n) = MinMaxThresh(wcoef((2^(L)+1):n)) ;
  elseif strcmp(type,'MAD'),
         wcoef = MultiMAD(wcoef,L);
  end
%
  xh    = IWT_PO(wcoef,L,qmf);
```

```
%
% Copyright (c) 1993-5.  Jonathan Buckheit, David Donoho and Iain
%Johnstone
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

c. Modified MultSURE; MultiSUREsoft [10].

```
function ws = MultiSUREsoft(wc,L)
% MultiSURE -- Apply Shrinkage to Wavelet Coefficients
%  Usage
%    ws = MultiSURE(wc,L)
%  Inputs
%    wc    Wavelet Transform of noisy sequence with N(0,1) noise
%    L     low-frequency cutoff for Wavelet Transform
%  Outputs
%    ws    result of applying SUREThresh to each dyadic block
%
        [n,J] = dyadlength(wc);
        for j=(J-1):-1:L
                wc(dyad(j)) = SUREThreshsoft(wc(dyad(j))) ;
        end
        ws = wc;

%
% Copyright (c) 1993-5.  Jonathan Buckheit, David Donoho and Iain
%Johnstone
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

d. Modified SUREThresh; SUREThreshsoft [10].

```
function [x,thresh] = SUREThreshsoft(y)
% SUREThresh -- Adaptive Threshold Selection Using Principle of SURE
%  Usage
%    thresh = SUREThresh(y)
%  Inputs
%    y       Noisy Data with Std. Deviation = 1
%  Outputs
%    x       Estimate of mean vector
%    thresh   Threshold used
%
%  Description
%    SURE referes to Stein's Unbiased Risk Estimate.
%
        thresh = ValSUREThresh(y);
        x      = SoftThresh(y,thresh);

%
% Copyright (c) 1993-5.  Jonathan Buckheit, David Donoho and Iain
%Johnstone
%



%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

2. Translation-Invariant

a. Modified FWT_TI; FWT_TI_visuthreshHard [10].

```
function wp = FWT_TI_visuthresh(x,L,qmf)
% FWT_TI -- translation-invariant forward wavelet transform
%  Usage
%    TIWT = FWT_TI(x,L,qmf)
%  Inputs
%    x       array of dyadic length n=2^J
%    L       degree of coarsest scale
%    qmf     orthonormal quadrature mirror filter
%  Outputs
%    TIWT    stationary wavelet transform table
%            formally same data structure as packet table
%
```

```
%  See Also
%    IWT_TI
%

        [n,J] = dyadlength(x);
        D = J-L;
        wp = zeros(n,D+1);
        x = ShapeAsRow(x);
%
        wp(:,1) = x';
        for d=0:(D-1),
                for b=0:(2^d-1),
                  s = wp(packet(d,b,n),1)';
                  hsr = DownDyadHi(s,qmf);
                  hsl = DownDyadHi(rshift(s),qmf);
                  lsr = DownDyadLo(s,qmf);
                  lsl = DownDyadLo(rshift(s),qmf);
                  wp(packet(d+1,2*b  ,n),d+2) = VisuThresh(hsr','Hard');
                  wp(packet(d+1,2*b+1,n),d+2) = VisuThresh(hsl','Hard');
                  wp(packet(d+1,2*b  ,n),1  ) = (lsr');
                  wp(packet(d+1,2*b+1,n),1  ) = (lsl');
                end
        end

%
% Copyright (c) 1994. David L. Donoho
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

b. Modified FWT_TI; FWT_TI_visuthreshSoft [10].

```
function wp = FWT_TI_visuthresh(x,L,qmf)
% FWT_TI -- translation-invariant forward wavelet transform
%  Usage
%    TIWT = FWT_TI(x,L,qmf)
%  Inputs
%    x       array of dyadic length n=2^J
%    L       degree of coarsest scale
%    qmf     orthonormal quadrature mirror filter
```

```
%  Outputs
%    TIWT     stationary wavelet transform table
%             formally same data structure as packet table
%
%  See Also
%    IWT_TI
%

          [n,J] = dyadlength(x);
          D = J-L;
          wp = zeros(n,D+1);
          x = ShapeAsRow(x);
%
          wp(:,1) = x';
          for d=0:(D-1),
                    for b=0:(2^d-1),
                      s = wp(packet(d,b,n),1)';
                      hsr = DownDyadHi(s,qmf);
                      hsl = DownDyadHi(rshift(s),qmf);
                      lsr = DownDyadLo(s,qmf);
                      lsl = DownDyadLo(rshift(s),qmf);
                      wp(packet(d+1,2*b  ,n),d+2) = VisuThresh(hsr','Soft');
                      wp(packet(d+1,2*b+1,n),d+2) = Visuthresh(hsl','Soft');
                      wp(packet(d+1,2*b  ,n),1  ) = (lsr');
                      wp(packet(d+1,2*b+1,n),1  ) = (lsl');
                    end
          end

%
% Copyright (c) 1994. David L. Donoho
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

c. Modified FWT_TI; FWT_TI_SUREthreshSoft [10].

```
function wp = FWT_TI_SurethreshSoft(x,L,qmf)
% FWT_TI -- translation-invariant forward wavelet transform
%  Usage
%    TIWT = FWT_TI(x,L,qmf)
```

```
%  Inputs
%    x        array of dyadic length n=2^J
%    L        degree of coarsest scale
%    qmf      orthonormal quadrature mirror filter
%  Outputs
%    TIWT     stationary wavelet transform table
%             formally same data structure as packet table
%
%  See Also
%    IWT_TI
%

          [n,J] = dyadlength(x);
          D = J-L;
          wp = zeros(n,D+1);
          x = ShapeAsRow(x);
%
          wp(:,1) = x';
          for d=0:(D-1),
                    for b=0:(2^d-1),
                      s = wp(packet(d,b,n),1)';
                      hsr = DownDyadHi(s,qmf);
                      hsl = DownDyadHi(rshift(s),qmf);
                      lsr = DownDyadLo(s,qmf);
                      lsl = DownDyadLo(rshift(s),qmf);
                      wp(packet(d+1,2*b  ,n),d+2) =
                      SUREThreshTransSoft(hsr');
                      wp(packet(d+1,2*b+1,n),d+2) =
                      SUREThreshTransSoft(hsl');
                      wp(packet(d+1,2*b  ,n),1  ) = lsr';
                      wp(packet(d+1,2*b+1,n),1  ) = lsl';
                    end
          end

%
% Copyright (c) 1994. David L. Donoho
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

d. Modified FWT_TI; FWT_TI_SUREthreshHard [10].

```
function wp = FWT_TI_SurethreshHard(x,L,qmf)
% FWT_TI -- translation-invariant forward wavelet transform
%  Usage
%    TIWT = FWT_TI(x,L,qmf)
%  Inputs
%    x        array of dyadic length n=2^J
%    L        degree of coarsest scale
%    qmf      orthonormal quadrature mirror filter
%  Outputs
%    TIWT     stationary wavelet transform table
%             formally same data structure as packet table
%
%  See Also
%    IWT_TI
%

        [n,J] = dyadlength(x);
        D = J-L;
        wp = zeros(n,D+1);
        x = ShapeAsRow(x);
%
        wp(:,1) = x';
        for d=0:(D-1),
                for b=0:(2^d-1),
                  s = wp(packet(d,b,n),1)';
                  hsr = DownDyadHi(s,qmf);
                  hsl = DownDyadHi(rshift(s),qmf);
                  lsr = DownDyadLo(s,qmf);
                  lsl = DownDyadLo(rshift(s),qmf);
                  wp(packet(d+1,2*b  ,n),d+2) = SUREThreshTrans(hsr');
                  wp(packet(d+1,2*b+1,n),d+2) = SUREThreshTrans(hsl');
                  wp(packet(d+1,2*b  ,n),1  ) = lsr';
                  wp(packet(d+1,2*b+1,n),1  ) = lsl';
                 end
        end

%
% Copyright (c) 1994. David L. Donoho
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
```

```
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

e. Modified FWT_TI; SUREThreshTrans [10].

```
function [x,thresh] = SUREThreshTrans(y)
% SUREThresh -- Adaptive Threshold Selection Using Principle of SURE
%  Usage
%    thresh = SUREThresh(y)
%  Inputs
%    y        Noisy Data with Std. Deviation = 1
%  Outputs
%    x        Estimate of mean vector
%    thresh   Threshold used
%
%  Description
%    SURE referes to Stein's Unbiased Risk Estimate.
%
        thresh = ValSUREThresh(transpose(y));
        x      = HardThresh(y,thresh);

%
% Copyright (c) 1993-5.  Jonathan Buckheit, David Donoho and Iain
%Johnstone
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

f. Modified FWT_TI; SUREThreshTransSoft [10].

```
function [x,thresh] = SUREThreshTransSoft(y)
% SUREThresh -- Adaptive Threshold Selection Using Principle of SURE
%  Usage
%    thresh = SUREThresh(y)
%  Inputs
```

```
%    y        Noisy Data with Std. Deviation = 1
%  Outputs
%    x        Estimate of mean vector
%    thresh   Threshold used
%
%  Description
%    SURE referes to Stein's Unbiased Risk Estimate.
%
          thresh = ValSUREThresh(transpose(y));
          x      = SoftThresh(y,thresh);

%
% Copyright (c) 1993-5.  Jonathan Buckheit, David Donoho and Iain
%Johnstone
%



%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

3. Combined

a. Modified FWT_TI; FWT_TI_SurethreshSoft_wiener [10].

```
function wp = FWT_TI_SurethreshSoft_wiener(x,L,qmf,order)
% FWT_TI -- translation-invariant forward wavelet transform
%  Usage
%    TIWT = FWT_TI(x,L,qmf)
%  Inputs
%    x        array of dyadic length n=2^J
%    L        degree of coarsest scale
%    qmf      orthonormal quadrature mirror filter
%  Outputs
%    TIWT     stationary wavelet transform table
%             formally same data structure as packet table
%
%  See Also
%    IWT_TI
%

          [n,J] = dyadlength(x);
          D = J-L;
          wp = zeros(n,D+1);
```

```
                x = ShapeAsRow(x);
        %
                wp(:,1) = x';
                for d=0:(D-1),
                        for b=0:(2^d-1),
                          s = wp(packet(d,b,n),1)';
                          hsr = DownDyadHi(s,qmf);
                          hsl = DownDyadHi(rshift(s),qmf);
                          lsr = DownDyadLo(s,qmf);
                          lsl = DownDyadLo(rshift(s),qmf);
                    size(SUREThreshTransSoft(hsr'));
                    if length(hsr')>length(x)/4
                          wp(packet(d+1,2*b  ,n),d+2) =
predict_window(SUREThreshTransSoft(hsr'),order,16);
                    end
                    if length(hsr')<=length(x)/4
                          wp(packet(d+1,2*b  ,n),d+2) =
SUREThreshTransSoft(hsr');
                    end
                    if length(hsl')>length(x)/4
                          wp(packet(d+1,2*b+1,n),d+2) =
predict_window(SUREThreshTransSoft(hsl'),order,16);
                    end
                    if length(hsl')<=length(x)/4
                          wp(packet(d+1,2*b+1,n),d+2) =
SUREThreshTransSoft(hsl');
                    end

                          wp(packet(d+1,2*b  ,n),1  ) = lsr';
                          wp(packet(d+1,2*b+1,n),1  ) = lsl';
                        end
                 end

        %
        % Copyright (c) 1994. David L. Donoho
        %


        %
        % Part of WaveLab Version 802
        % Built Sunday, October 3, 1999 8:52:27 AM
        % This is Copyrighted Material
        % For Copying permissions see COPYING.m
        % Comments? e-mail wavelab@stat.stanford.edu
        %
```

b. Modified FWT_TI; FWT_TI_SurethreshSoft_medfilt [10].

```
function wp = FWT_TI_SurethreshSoft_medfilt(x,L,qmf,order)
% FWT_TI -- translation-invariant forward wavelet transform
%  Usage
%    TIWT = FWT_TI(x,L,qmf)
%  Inputs
%    x        array of dyadic length n=2^J
%    L        degree of coarsest scale
%    qmf      orthonormal quadrature mirror filter
%  Outputs
%    TIWT     stationary wavelet transform table
%             formally same data structure as packet table
%
%  See Also
%    IWT_TI
%

          [n,J] = dyadlength(x);
          D = J-L;
          wp = zeros(n,D+1);
          x = ShapeAsRow(x);
%
          wp(:,1) = x';
          for d=0:(D-1),
                    for b=0:(2^d-1),
                      s = wp(packet(d,b,n),1)';
                      hsr = DownDyadHi(s,qmf);
                      hsl = DownDyadHi(rshift(s),qmf);
                      lsr = DownDyadLo(s,qmf);
                      lsl = DownDyadLo(rshift(s),qmf);
                size(SUREThreshTransSoft(hsr'))
                      wp(packet(d+1,2*b  ,n),d+2) =
medfilt1(SUREThreshTransSoft(hsr'),3);
                size(wp(packet(d+1,2*b  ,n),d+2))
                      wp(packet(d+1,2*b+1,n),d+2) =
medfilt1(SUREThreshTransSoft(hsl'),3);
                size(wp(packet(d+1,2*b+1,n),d+2))
                      wp(packet(d+1,2*b  ,n),1  ) = lsr';
                      wp(packet(d+1,2*b+1,n),1  ) = lsl';
                    end
           end

%
% Copyright (c) 1994. David L. Donoho
```

```
%


%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%
```

G. WINDOWED PREDICTOR

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   This Matlab Function performs Windowed Wiener prediction
%   to the given "order" on sequencial segments of Data.
%   Input must be 2^Integer, and greater than W in length.
%   Wiener Prediction conducted forward in time and reverse in time
%   The Data is forward predicted, then reverse
%   predicted.  The first (order-1) data bits not fully covered by the forward filter
%   are replaced by the data bits found using the reverse filter and vice versa
%   The estimate for predicting one sample ahead is s(n+1).
%
%   Function Syntax as follows:
%
%      [Filtered] = predict_window(Data,order,W)
%
%   where:
%          "Data"     = True signal passed into function length 2^Integer
%          "order"    = Wiener Predictor Order or predictor size
%          "W"        = Window Size
%          "Filtered" = Predictor output
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
function [Filtered] =predict_window(Data,order,W)
%
% Put Data in proper [1 length(Data)] format
          Predictsize = size(Data);
          if Predictsize(1)>1
                    Data=transpose(Data);
          end
%
% Split Data into segments and perform denoising on each segment
```

```
%
for g=0:(length(Data)/W)-1
        %
        %Form Rx of Wiener Hopf Equation
        %
                DataW=Data(1+g*W:W+g*W);
                Datasize=size(Data);
                DataWsize=size(DataW);
                AutoCorrX=xcorr(DataW);
        %
        %Form Rx[0]..Rx[order-1]
        %
                rx=AutoCorrX(W:W+order-1)';%r(0),...r(order-1)
        %
        %Form Rs[1]..Rs[order]
        %
                 rs=AutoCorrX((W+1):W+order)';
        %
        %Find toeplitz Rx
        %
                R=toeplitz(rx);
        %
        %Check to ensure matrix not singular
        %
                flag=0;
                for b=1:length(DataW)
                        if DataW(b) ~= 0
                                flag =1;
                        end
                end
                if flag ==1
                        Rinv=inv(R);
                end
         %
        %If all values are one then singular
        %Future work should also correct for
        %ill-conditioned
         %
                if flag == 0
                        Rinv=R;
                end
        %
        %Form prediction filter
        %
                h=Rinv*rs;
        %
```

```
%Convolve S(n+1) = h[0]x[n] + h[1]x[n-1] + ... + h[p]x[n-p]
%
        for n=1:(W-(order))
                FF=DataW(n:(n+order-1)).*fliplr(transpose(h));
                Filtered(n+g*W)=sum(FF);
        end
%
%Ensure data still in correct format
%
        Filteredsize=size(Filtered);
        if Filteredsize(1)>1
                Filtered=transpose(Filtered);
        end
%
%Keep the filtered values that touched entire filter
%Hence Keep all but first (order-1) values
%
        Filtered1(1+g*W:W+g*W)=[DataW(1:order), Filtered(1+g*W:W+ g*W-
        order )];
        Filtered1size=size(Filtered1);
%
%Ensure data still in correct format
%
        if Filtered1size(1)>1
                Filtered1=transpose(Filtered1);
        end
%
%reverse Data
%
        DataW1=fliplr(flipud(DataW));
%
%Filter in reverse direction
%
        for n=1:(W-(order))
                FF1=DataW1(n:(n+order-1)).*fliplr(transpose(h));
                Filtered3(n+g*W)=sum(FF1);
        end
%
%Ensure data still in correct format
%
        Filtered3size=size(Filtered3);
        if Filtered3size(1)>1
                Filtered3=transpose(Filtered3);
        end
%
%Keep the filtered values that touched entire filter
```

```
        %
                Filtered2(1+g*W:W+g*W)=[DataW1(1:order),…
                Filtered3(1+g*W:W+g*W-order)];
                Filtered2size=size(Filtered2);
        %
        %Ensure data still in correct format
        %
        if Filtered2size(1)>1
                Filtered2=transpose(Filtered2);
        end
        %
        %Flip reversed data back, such that it compares to forward data point
        %by point
        %
                Filtered2(1+g*W:W+g*W)=fliplr(flipud(Filtered2(1+g*W:W+g*W)));
        %
        %The first (order-1) data bits not fully covered by the forward filter
        %are replaced by the data bits found using the reverse filter and vice versa
        %
                Filtered1(1+g*W:1+g*W+order)=Filtered2(1+g*W:1+g*W+order);
                Filtered2(W-(order-1):W)=Filtered1(W-(order-1):W);
                Filtered(1+g*W:W+g*W)= .5*(Filtered2(1+g*W:W+ g*W)+
                Filtered1(1+g*W:W+g*W));
        %
        %Ensure data still in correct format
        %
                if Predictsize(1)>1
                        Filtered=transpose(Filtered);
                end
        %
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] D. Donoho and I. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometerika*, Vol. 81, No. 3, pp. 425-455, 1994.

[2] D. Donoho and I. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, Vol. 90, pp. 1200-1224, December 1995.

[3] S. Mallat, A *Wavelet Tour of Signal Processing*, Academic Press, San Diego, California, 1998.

[4] G. Bachman, L. Narici, and E. Beckenstein, *Fourier and Wavelet Analysis*, Springer-Verlag, New York, 2000.

[5] C. Cebeci, "Denoising of acoustic signals using wavelet/wiener based techniques", MSEE Thesis, Naval Postgraduate School, Monterey, California, June 1998.

[6] K. Sayood, *Introduction to Data Compression*, Academic Press, San Diego, California, 2000.

[7] N. Hamlett, "Comparison of multiresolution techniques for digital signal processing", MSEE Thesis, Naval Postgraduate School, Monterey, California, March 1993.

[8] R. J. Barsanti, Jr., "Denoising of ocean acoustic signals using wavelet-based techniques", MSEE Thesis, Naval Postgraduate School, Monterey, California, December 1996.

[9] O. Duzenli, "Classification of underwater signals using wavelet-based decompositions", MSEE Thesis, Naval Postgraduate School, Monterey, California, June 1998.

[10] J. Buchheit, S. Chen, D. Donoho, and J. Scargle, "Wavelab .802," http://www.wavelab/playfair.stanford.edu, 1996, last accessed December 2002.

[11] A. V. Oppenheim, and A. S. Willsky with S. H. Nawab, *Signals and Systems*, Prentice-Hall, Inc., New Jersey, 1997.

[12] J. G. Proakis, and D. G. Manolakis, *Digital Signal Processing*, Prentice-Hall, Inc., New Jersey 1996.

[13] C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice-Hall, Inc., New Jersey 1992.

[14] D. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, Vol. 41, pp. 613-627, May 1995.

[15] L. Sendur and I. Selesnick, "Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency," *IEEE Transactions on Signal Processing*, Vol. 50, No. 11, pp. 2744-2756, 2002.

[16] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, Vol. 9, No. 9, pp. 1532-1546, 2000.

[17] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," Internal Report, Department of Statistics, Stanford University, 1995.

[18] A. Fletcher and K. Ramchandran, "Wavelet denoising by recursive cycle spinning," *IEEE Proceedings International Conference on Image Processing* , Vol. 2, pp. 873-876, 2002.

[19] The Mathworks, Inc., *Matlab Reference Guide*, Massachusetts 1992.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Monique P. Fargues
   ECE Dept, Code EC/FA
   Naval Postgraduate School
   Monterey, California

4. Dr. Charles W. Therrien
   ECE Dept, Code EC/TI
   Naval Postgraduate School
   Monterey, California

5. Dr. John P. Powers
   Chairman ECE Dept, Code EC/PO
   Naval Postgraduate School
   Monterey, California

6. LT John B. Hughes
   Spokane, Washington