Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations

Student Graduate Works

3-2005

Noise Estimation in the Presence of BPSK Digital Burst Transmissions

Susan E. Bettison

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Applied Mathematics Commons, and the Signal Processing Commons

Recommended Citation

Bettison, Susan E., "Noise Estimation in the Presence of BPSK Digital Burst Transmissions" (2005). *Theses and Dissertations*. 3715.
https://scholar.afit.edu/etd/3715

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.

NOISE ESTIMATION IN THE PRESENCE OF BPSK DIGITAL BURST TRANSMISSIONS

THESIS

Susan Elizabeth Bettison, Second Lieutenant, USAF

AFIT/GAM/ENC/05-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

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 United States Government.

AFIT/GAM/ENC/05-03

Noise Estimation in the Presence of BPSK Digital Burst Transmissions

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Applied Mathematics

Susan Elizabeth Bettison, B.S.

Second Lieutenant, USAF

March 2005

Approved for public release; distribution unlimited

AFIT/GAM/ENC/05-03

Noise Estimation in the Presence of BPSK Digital Burst Transmissions

Susan Elizabeth Bettison, B.S.

Second Lieutenant, USAF

Approved:

Lawrence K. Chilton (Chairman) Date

Jay E. Apisa (Member) Date

Richard K. Martin (Member) Date

AFIT/GAM/ENC/05-03

Abstract

This research explores noise estimation techniques in an attempt to improve upon a previously developed digital burst transmission Binary Phase Shift Keyed (BPSK) demodulator. The demodulator success is dependant on the accuracy of the estimate of Power Spectral Density (PSD) of the unknown noise. Given a discrete time signal transformed into the frequency domain, the research seeks to determine if it is possible to effectively estimate the PSD of the unknown noise. The demodulator was developed using a new signal model for digital burst transmissions based on linear spectral subspace theory. Using this model and the redundancy properties of BPSK digital burst transmissions, five noise estimation techniques will be presented and tested. The success of the methods will be reported in two ways, first, the effect the new noise PSD estimates have on the success of the demodulator and second, a comparison to the actual PSD of the noise.

Acknowledgements

I would sincerly like to thank my advisor, Dr. Chilton and my committee members Dr. Martin and Mr. Apisa.

Susan Elizabeth Bettison

Table of Contents

Page

Page

List of Figures

Figure Page

Noise Estimation in the Presence of BPSK Digital Burst Transmissions

I. Introduction

Digital burst transmission signals are all around us, everyday. An example everyone has seen is a digital cell phone. As communication techniques become increasingly secure, exploitation difficulties increase. Not only do problems exist when trying to exploit non friendly communications, but in cooperative communications. If successful, the application of this research will be helpful in both scenarios.

1.1 Problem Definition

A fundamental problem in all signal processing has always been the estimation of the noise received with any transmitted signal. The more that is known about the noise, the more effective any signal processing algorithm will be. Using linear subspace theory, a new demodulation algorithm was developed in (11). Existing signal processing techniques did not fully apply to the digital burst transmission signal processing problem (11) thus a new approach was needed. The linear subspace theory research resulted in a new demodulation algorithm which outperformed all other existing demodulation techniques, for digital burst transmissions, including Binary Phase Shift Keyed (BPSK) modulated signals (11). The drawback of the technique is that it performs best when an a priori estimation of the noise environment is known, and a colored noise environment increases the success even further (11). This was the initial motivation for this research. However, to jump directly into the colored noise problem is not feasible since colored noise environments are generally unpredictable. Thus, the white noise estimation problem is a logical starting point. Although the white noise estimation problem is unlikely to immediately improve the performance of the demodulation technique, it provides a starting point for the colored noise estimation problem or even adaptive noise estimation techniques.

The new demodulation algorithm provided a solution to a problem, but also presented additional problems to be solved. The problem of note for this thesis is the estimation of the Power Spectral Density (PSD) of the noise since the demodulation technique presented

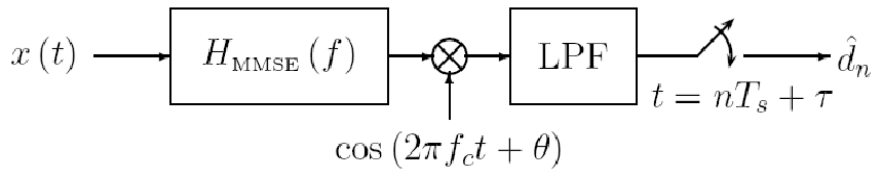


Figure 1.1 Diagram of BPSK Signal Model Construction

in (11) uses this form of the noise. Comprehension of the problem begins with the signal model and the demodulation technique or filter developed in (11). The signal model for BPSK signals used in this thesis is:

$$x(t) = \underbrace{A\,\mathrm{Re}(\sum_{n=0}^{N_s-1} d_n \psi(t - nT_s - \tau)e^{i(2\pi f_c t + \theta)})}_{s(t)\text{ the signal}} + \underset{\text{the noise and interference}}{n(t)} \quad . \tag{1.1}$$

Here, A is the signal gain; N_s is the number of symbols transmitted; d_n are the transmitted data symbols; $\psi(t)$ is the pulse shape; T_s is the symbol duration; τ is the time the first symbol arrives or delay; θ is the phase of the carrier frequency; and f_c is the carrier frequency in Hertz.

This signal model is the same model used in the development of the new demodulation technique based on linear subspace theory (11). Since the original goal was to improve this technique, this is the signal model that will be used throughout the thesis. A received signal has two parts, the originally transmitted signal and the noise. Obviously, the more that is known about a received signal, the easier the separation between what was originally transmitted and the noise becomes. If all of the parameters of a received signal are known the only unknown portions are the transmitted symbols, d_n, and the noise, $n(t)$. Unfortunately, since noise environments can be austere, accurately estimating these two parameters can be difficult, if not impossible. Gisselquist presents a technique for estimating the d_n in (11), but $n(t)$ remains unknown. Without a good estimate of $n(t)$ the estimation of the data symbols will most likely be inaccurate. Using the filter presented in (11) by Gisselquist, and the properties of the signal model also presented in (11), five estimation techniques for $n(t)$ are presented here. These techniques are unique because they are implemented using techniques based on Gisselquist's linear subspace approach. Prior to Gisselquist's research, signal processing techniques for BPSK and other Pulse Amplitude Modulated (PAM) digital signals was based on stationary or cyclostationary signal processing (11). Provided the techniques presented here are sufficiently robust, they will be an excellent starting point for the larger colored noise estimation problem and for the adaptive noise estimation problem. Even if not successful, they should provide direction for future noise estimation techniques.

1.2 Organization

This thesis begins with an in depth explanation of the signal model, the properties used in estimating the PSD of the noise, a description of relevant existing statistical signal processing models and finally an overview of the demodulation technique presented by Gisselquist. Chapter III presents the methods used to estimate the PSD of the noise based off the explanations in Chapter II, and Chapter IV explains the results when tested in simulations. The last chapter, Chapter V draws some conclusions regarding the success of the noise estimation techniques.

II. Background

This chapter presents an explanation of digital burst transmission signals, a summary of relevant statistical signal processing models, an optimal filtering technique for the relevant statistical signal processing models, and finally presents the minimum mean square error filter that suggested the need for noise estimation in the presence of digital burst transmission signals.

The working signal model is a signal with a general modulation type of Pulse Amplitude Modulation (PAM), but more specifically, Binary Phase Shift Keyed (BPSK) modulation. The received signal in time may be represented by equation 1.1. This signal model has many properties that existing statistical signal processing models did not adequately exploit (11). Because of this, Gisselquist established a statistical signal processing model for burst transmission signals and then developed a Minimum Mean Square Error (MMSE) filter which exploits more of the characteristics than previous filters. This filter produced the best symbol recovery results to date, given the noise estimate used in the construction of the filter was accurate in shape and scale (11). The relevant statistical signal processing models will be presented, then the chapter will finish with the derivation of Gisselquist's MMSE filter, presenting some results obtained using his filter on simulated BPSK signals.

2.1 Burst Transmission Signals

Burst transmission signals have many unique properties that distinguish them from other types of signals. The first and most important, is that they are not infinite in time. In fact they are very time limited, and more uniquely, the times when the signal is active may or may not be of all the same duration. This section presents a basic PAM signal model, an explanation of the subspace that describes it, the specifics of a BPSK signal model that exists in this subspace, and finally the distribution of a burst transmission signal is derived.

2.1.1 Pulse Amplitude Modulated Signal Model. The received signal in time may be represented by (11),

$$x(t) = \underbrace{A\,\mathrm{Re}(\sum_{n=0}^{N_s-1} d_n \psi(t - nT_s - \tau)e^{i(2\pi f_c t+\theta)})}_{s(t) \text{ the transmitted signal}} + \underset{\text{the noise and interference}}{n(t)} \quad . \tag{2.1}$$

The parameters in this signal model are defined in Section 1.1.

In this thesis, the signal is analyzed in the frequency domain for two reasons. First, stationary signals which are correlated in time become uncorrelated in frequency for sufficiently long observation times (11, 18). Second, the unknown terms in τ and θ become constant multipliers in the frequency domain, making them easier to work with. When it comes to signal processing in the frequency domain, there are two statistics of interest that have been studied extensively by others; these statistics are the first and second moments (11). The first moment is the expected value, or the mean, and is assumed to be zero for both the signal and noise terms. The second moment, or the variance, is more interesting and is where many of the differentiating characteristics of signals are found. The variance of a signal depends on how the signal is defined (11). If the signal is time-limited then the infinite time Fourier transform $X(f)$ of the signal $x(t)$,

$$X(f) \triangleq \int_{-\infty}^{\infty} x(t)e^{-i2\pi f t}dt, \tag{2.2}$$

converges. With a time limited signal, both the signal and noise are assumed to have zero mean and the variance of $X(f)$ is[1] $\mathbb{E}\{|X(f)|^2\}$. If the signal is of infinite length, the Fourier transform will not always converge (17, 11). Because of this, an alternate method must be used to describe the variance of an infinite time signal. This alternate method is the Power Spectral Density (PSD), $S_x(f)$, and is defined as (17):

$$S_x(f) \triangleq \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\{|X_T(f)|^2\} \tag{2.3}$$

where $X_T(f)$ is the time limited Fourier transform of $x(t)$,

[1]The notation $\mathbb{E}\{\cdot\}$ is used throughout to refer to the expected value of its argument.

$$X_T(f) \triangleq \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-i2\pi ft}dt \tag{2.4}$$

Thus the PSD can be used to describe the distribution of the power in a signal, as a function of frequency (11). Additionally, a nice property of uncorrelated signals is that the PSD of the received signal $S_x(f)$ is the sum of the PSD of the transmitted signal, $S_s(f)$, and the PSD of the noise, $S_n(f)$(11, 23):

$$S_x(f) = S_s(f) + S_n(f). \tag{2.5}$$

This is very useful in both Gisselquist's algorithm development and the methods used in this thesis to estimate the PSD of the noise. The PSD of a burst transmission signal or more generally a pulse amplitude modulated (PAM) signal is[2] (9):

$$S_s(f) = \frac{A^2}{4T_s}|\Psi(f-f_c)|^2\mathbb{E}\{|d_n|^2\} \tag{2.6}$$

Here, $\Psi(f)$ is the Fourier transform of the pulse function $\psi(t)$. T_s is the symbol duration. f_c is the carrier frequency in Hertz, and d_n are the transmitted data symbols. A received PAM transmission signal with a finite number of symbols, N_s, is described by equation 2.1.

The noise and interference term represents a random process that is assumed to be stationary, independent of the signal, and to have a PSD defined as $S_n(f)$(11). This thesis focuses on developing an adequate method of estimating this PSD. Each of the terms in equation 2.1 can be understood in the context of how the signal is created. As an example, consider the signal example in Figure 2.1, an example of three transmissions created with the model in equation 2.1 with the noise term set to zero. The bursts are created from a sum of pulses, $\psi(t - nT_s - \tau)$. The figure is a sum of three bursts all of different lengths. The pulses are separated by T_s, and scaled by a system gain, $A = 1$, from their original amplitude. A time delay τ, has shifted the signal to the right, and the phase, θ, and the

[2]The assumption that the transmitted data symbols are uncorrelated or that $\mathbb{E}\{d_n d_m^*\} = 0$, for $n \neq m$ is made throughout this thesis.

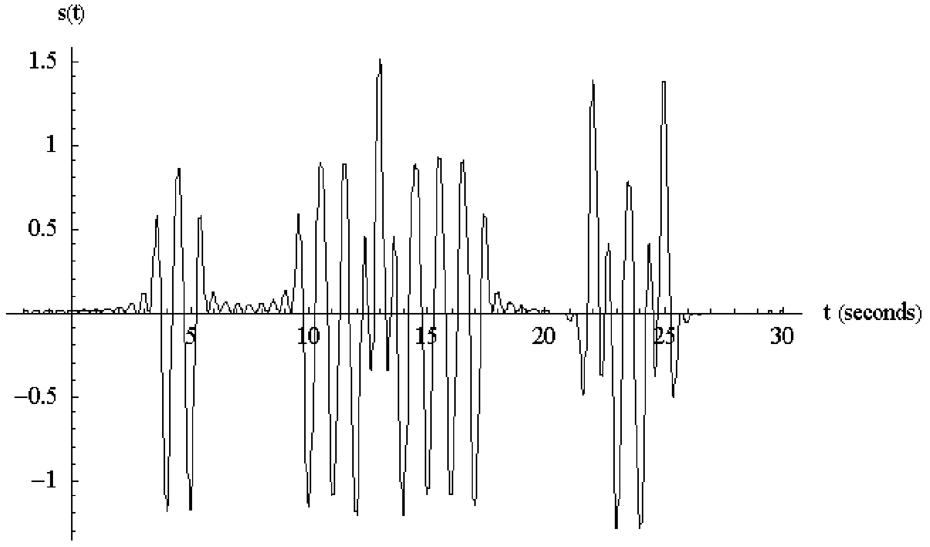


Figure 2.1 A Sum of Three BPSK Burst Transmissions

carrier frequency f_c, are both zero. Each burst is itself a sum of pulses, one pulse for each data symbol transmitted. The first burst has two data symbols, the second has eight and the third, four. Each burst is separated by a time of four seconds. This is obviously a very elementary example but illustrates what burst transmission signals might look like. Although this figure has three bursts, within the scope of this thesis, we will only be concerned with estimating the noise in one burst ignoring the time before and after, when the signal is inactive.

2.1.2 Burst Transmission Subspace and BPSK Signal Model Specifics. Gisselquist took the principles of other statistical signal processing techniques and extended them so they would be applicable to time-limited burst transmission signals. Next we will describe in detail Gisselquist's development of the signal model and the associated subspace, as it plays a major role in the development of the processing technique. Other statistical signal processing techniques assume infinite time length signals as in the case of stationary signals, or that the cyclic spectral density function, also known as the Spectral Correlation Function, defined in equation 2.32 is known, as in cyclostationary signals. Neither of these apply to burst transmission signals. Burst transmissions do have correlation in the frequency domain, however it is not usually known. Further explanation of the differences between burst transmission signals and these other statistical signal processing models will be explained in Section 2.2.

In deriving a new statistical model for burst transmissions, Gisselquist first had to establish the subspace that burst transmission signals exist in. Consider the set of all functions defined over a finite time containing the signal. This can be seen because the signal is a linear combination of evenly spaced pulses weighted by the d_n values. The pulses then form the basis for the subspace where the signal resides. These two sets of vectors span the subspace that contains PAM signals (11):

$$\begin{aligned} \nu_{ki} &= \psi(t - kT_s - \tau)\cos(2\pi f_c t + \theta),\ k \in I_s \\ \nu_{kq} &= \psi(t - kT_s - \tau)\sin(2\pi f_c t + \theta),\ k \in I_s \end{aligned} \tag{2.7}$$

where I_s is some finite set of consecutive integers. The problem with this subspace is that these basis vectors depend on the unknown delay, τ, and the carrier phase, θ, making them difficult to use in practice (11). To continue with the development, the signal must be transformed into the frequency domain with the Fourier transform.

Burst transmission signals are time limited and thus fall within a finite observation time, $t \in (-\frac{T}{2}, \frac{T}{2})$. Although here the time limited Fourier transform is identical to the infinite time Fourier transform, the time limited transform is required since the noise may or may not be time-limited (11). The time-limited Fourier transform of the received signal is then (11):

$$X_T(f) \triangleq \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-i2\pi ft}dt \tag{2.8}$$

Define $N_T(f)$ to be the time-limited Fourier transform of the noise and interference thus the Fourier transform of

$$x(t) = A\,\mathrm{Re}(\sum_{n=0}^{N_s-1} d_n\psi(t - nT_s - \tau)e^{i(2\pi f_c t+\theta)}) + n(t) \tag{2.9}$$

becomes (11):

$$X_T(f) = N_T\,(f) + \tfrac{A}{2}\sum_{n=0}^{N_s-1}\mathrm{Re}(d_n)\int_{-\infty}^{\infty}\psi(t-nT_s-\tau)[e^{i2\pi f_c t+i\theta-i2\pi ft}+e^{-i2\pi f_c-i\theta-i2\pi ft}]dt$$
$$+i\tfrac{A}{2}\sum_{n=0}^{N_s-1}\mathrm{Im}(d_n)\int_{-\infty}^{\infty}\psi(t-nT_s-\tau)[e^{i2\pi f_c t+i\theta-i2\pi ft}+e^{-i2\pi f_c-i\theta-i2\pi ft}]dt \tag{2.10}$$

Since the scope of our problem is BPSK signals only, we can disregard the imaginary part since the signal is purely real in the time domain. Now letting $u = t - T_s - \tau$, and $du = dt$ we have (11):

$$X_T(f) = N_T\,(f) + \frac{A}{2}e^{i\theta}\sum_{n=0}^{N_s-1}\cos(2\pi(f-f_c)\tau)[d_n\cos(2\pi(f-f_c)nT_s)]\int_{-\infty}^{\infty}\psi(u)\cos(2\pi(f-f_c)u)du. \tag{2.11}$$

Continuing with substitutions, if we let the Fourier transform of $\psi(t)$ be
$\Psi(f) \triangleq \int_{-\infty}^{\infty}\psi(t)e^{-i2\pi ft}dt$ and the z-transform of the data symbols be

$$D(z) \triangleq \sum_{n=0}^{N_s-1} d_n z^{-n} \tag{2.12}$$

the function becomes (11):

$$X_T(f) = N_T(f) + \frac{A}{2}e^{i\theta}e^{-i2\pi(f-f_c)\tau}\Psi(f-f_c)D(e^{i2\pi(f-f_c)T_s}) \tag{2.13}$$

Two redundancies can be seen here. It is important to note that these are not redundancies in the sense that they are exactly the same, but redundant because they contain the same information as other locations. Since the information contained is the same in each redundancy, knowing how these locations differ is what allows the signal processor to exploit the redundancies. The differences between these first two locations come from the fact that $D(e^{i2\pi(f-f_c)T_s})$ is periodic (11) and the second from the fact that it is conjugate symmetric.

To see the redundancies, let $X_s(f) \triangleq X_T(f) - N_T(f)$ and ignore the noise term for now. The first redundancy is found between f and $f + \frac{k}{T_s}$ where k is any integer. For two such frequencies (11),

$$\begin{bmatrix} X_s(f) \\ X_s(f + \frac{k}{T_s}) \end{bmatrix} = \frac{A}{2} e^{i\theta} e^{-i2\pi(f-f_c)\tau} \begin{bmatrix} \Psi(f - f_c) \\ e^{-i2\pi\frac{k\tau}{T_s}} \Psi(f + \frac{k}{T_s} - f_c) \end{bmatrix} D(e^{i2\pi(f-f_c)T_s}) \quad (2.14)$$

and the Fourier Transform of the data within the signal is completely redundant. This is caused by the fact that the signal was created by linearly modulating an impulse stream and thus $D(e^{i2\pi(f-f_c)T_s})$ is a periodic function of f (11). This was first noticed by Nyquist (16) and later exploited by Berger and Tufts in their filter development (11, 1, 26). The implication, presented by Nyquist, is that any bandwidth larger than $\frac{1}{T_s}$ contains redundancies (11, 1). This is commonly referred to as the Nyquist Minimum Bandwidth of a complex signal.

The second redundancy can be seen about the carrier frequency since the BPSK signal is purely real, so the Z-Transform of a real sequence and the Fourier transform of a real signal are both conjugate symmetric about zero (25). From this we have (11)

$$\begin{bmatrix} X_s(f) \\ X_s^*(2f_c - f) \end{bmatrix} = \frac{A}{2} e^{i\theta} e^{-i2\pi(f-f_c)\tau} \begin{bmatrix} \Psi(f - f_c) \\ e^{-i2\theta} \Psi^*(f - f_c) \end{bmatrix} D(e^{i2\pi(f-f_c)T_s}) \quad (2.15)$$

This redundancy was first noticed by Nyquist yet not exploited by Tufts (11). When combined with equation 2.14, this equation implies that the minimum bandwidth of an underlying communications signal having real symbols is $\frac{1}{2T_s}$(11, 16) since a smaller bandwidth would not completely contain these redundancies. From these equations 2.15 ,2.14, a subspace can be described in frequency in which the underlying signal must lie (11). This subspace includes a periodic redundancy and, for real-valued signals, a redundancy about the carrier as well (11). Consider a signal constrained to lie within a baseband of $|f_c| \leq \frac{1}{T_s}$, and having symbols in the BPSK

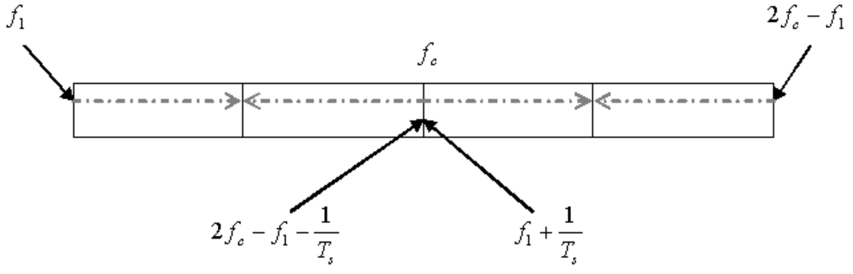


Figure 2.2 Positioning of Redundancies in the Frequency Domain Vector

symbol set, $D = \{\pm 1\}$. In this case the spectral redundancies associated with the frequency f, provided $-\frac{1}{T_s} \leq f \leq \frac{1}{T_s}$ can be written as (11)

$$\begin{bmatrix} X_s(f) \\ X_s(f+\frac{1}{T_s}) \\ X_s^*(2f_c - f) \\ X_s^*(2f_c - f - \frac{1}{T_s}) \end{bmatrix} = \frac{Ae^{i\theta}e^{-i2\pi(f-f_c)\tau}}{2} \begin{bmatrix} \Psi(f-f_c) \\ e^{-i2\pi\frac{\tau}{T_s}}\Psi(f+\frac{1}{T_s}-f_c) \\ e^{-i2\theta}\Psi^*(f-f_c) \\ e^{-i2\theta}e^{-i2\pi\frac{\tau}{T_s}}\Psi^*(f+\frac{1}{T_s}-f_c) \end{bmatrix} D(e^{i2\pi(f-f_c)T_s}) \tag{2.16}$$

It is easier to see with $f_c = 0$, $\tau = 0$, and $\theta = 0$:

$$\begin{bmatrix} X_s(f) \\ X_s(f+\frac{1}{T_s}) \\ X_s^*(-f) \\ X_s^*(-f - \frac{1}{T_s}) \end{bmatrix} = \frac{A}{2} \begin{bmatrix} \Psi(f) \\ \Psi(f+\frac{1}{T_s}) \\ \Psi^*(f) \\ \Psi^*(f+\frac{1}{T_s}) \end{bmatrix} D(e^{i2\pi f T_s}) \tag{2.17}$$

It is important to remember that the $X_s(f)$ refers to the portion of the received signal due to signal alone (11). The actual received signal includes corresponding noise terms and we will use these redundancies of the received signal to estimate the received noise.

2.1.2.1 Compact Representation. While these equations describe redundancies associated with one particular frequency, many frequency components are needed to describe a signal of interest. Gisselquist states that it is readily proved that a complete basis for the subspace of the signal requires a minimum of N_s elements for a real signal and not just the single vectors describing an arbitrary frequency f presented above (11). The transition between a set of frequencies, instead of just one, is examined here. The transition to vector notation is also explained.

In order to represent the whole frequency bandwidth containing the signal, we need to sample the Fourier Transform of the signal across its bandwidth (11). If we let N_f be the minimum number of samples required to span the Nyquist minimum bandwidth of the signal, then for a real signal, i.e. $D(e^{i2\pi(f-f_c)T_s})$ when the d_n are real, then $N_f = \frac{N_s}{2}$ samples are required. From the redundancy equations presented in the previous section, every value of $D(e^{i2\pi(f-f_c)T_s})$ directly determines the signal component of m frequencies, where m is determined by the bandwidth of the signal and the redundancy equations (11). For the BPSK redundancy given in equation 2.16, $m = 4$, frequency samples are required for each $D(e^{i2\pi(f-f_c)T_s})$ sample (11). This means that $4N_f$ complex frequency samples are sufficient to describe the received signal across its entire frequency band (11). This is equivalent to describing four copies of the signal's Nyquist minimum bandwidth (11).

As for which $4N_f$ frequencies need to be chosen, the simplest option is just to uniformly sample all of the frequencies (11). These $4N_f$ frequencies are equally spaced and centered about the carrier frequency f_c. Figure 2.2 shows where the redundancies occur within the vector. The first of the N_f frequencies shall be denoted $f_c - \frac{1}{T_s} < f_1, f_2, f_3, ..., f_{N_f} < f_c - \frac{1}{2T_s}$ and starts at the left in Figure 2.2. The remaining $3N_f$ frequencies are linear combinations of the first N_f frequencies. Let $X_T(f)$ be the Fourier transform of the received signal as in equation 2.13. The vector of these x, would look something like

$$x = \begin{bmatrix} X(f_1) \\ \vdots \\ X(f_{N_f}) \\ X(f_{N_f} + \frac{1}{T_s}) \\ \vdots \\ X(f_1 + \frac{1}{T_s}) \\ X^*(2f_c - f_1) \\ \vdots \\ \vdots \\ X^*(2f_c - f_1 - \frac{1}{T_s}) \end{bmatrix} \tag{2.18}$$

The second half of the signal is conjugated, this is due to the baseband redundancy given in equation 2.15. The noise vector is defined identically, with the exception that it corresponds to the noise and interference terms without the transmitted signal (11). Thus, both the noise and signal vectors are of dimension $4N_f \times 1$ (11).

The data portion of the signal $D(z)$ is organized into a similar vector d, with dimension $N_f \times 1$ as follows (11):

$$d \triangleq \begin{bmatrix} D(e^{i2\pi(f_1 - f_c)T_s}) \\ D(e^{i2\pi(f_2 - f_c)T_s}) \\ \vdots \\ D(e^{i2\pi(f_{N_f} - f_c)T_s}) \end{bmatrix}. \tag{2.19}$$

This data vector is shaped by a pulse weight matrix formed from the $\Psi(f_i - f_c)$ terms in the redundancy equations (11). This shaping matrix shall be referred to as $\boldsymbol{\Psi}$, which is

$4N_f \times N_f$ and is constructed as follows (11):

$$\mathbf{\Psi} = \begin{bmatrix} \Psi(f_1 - f_c) & 0 & \cdots & 0 \\ 0 & \Psi(f_2 - f_c) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Psi(f_{N_f} - f_c) \\ \Psi(f_1 + \frac{1}{T_s} - f_c) & 0 & \cdots & 0 \\ 0 & \Psi(f_2 + \frac{1}{T_s} - f_c) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Psi(f_{N_f} + \frac{1}{T_s} - f_c) \\ \Psi^*(f_c - f_1) & 0 & \cdots & 0 \\ 0 & \Psi^*(f_c - f_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Psi^*(f_c - f_{N_f}) \\ \Psi^*(f_c - f_1 - \frac{1}{T_s}) & 0 & \cdots & 0 \\ 0 & \Psi^*(f_c - f_2 - \frac{1}{T_s}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Psi^*(f_c - f_{N_f} - \frac{1}{T_s}) \end{bmatrix} \tag{2.20}$$

The contributions of τ and θ are grouped into a unitary and diagonal matrix R_ϕ, where ϕ refer to these complex phase values (11). R_ϕ is $4N_f \times 4N_f$ and contains all of the complex exponentials in the subspace equations and is a diagonal matrix as follows (11):

$$\mathbf{R}_\phi = \begin{bmatrix} R_\phi^{11} & 0 & \cdots & 0 \\ 0 & R_\phi^{22} & \cdots & 0 \\ 0 & \cdots & R_\phi^{33} & 0 \\ 0 & \cdots & \cdots & R_\phi^{44} \end{bmatrix} \tag{2.21}$$

where $R_\phi^{11} = \begin{bmatrix} e^{-i2\pi(f_1-f_c)\tau} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{-i2\pi(f_{N_f}-f_c)\tau} \end{bmatrix}$,

$$R_\phi^{22} = \begin{bmatrix} e^{-i2\pi(f_1-f_c)\tau}e^{-i2\pi\frac{\tau}{T_s}} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{-i2\pi(f_{N_f}-f_c)\tau}e^{-i2\pi\frac{\tau}{T_s}} \end{bmatrix},$$

$$R_\phi^{33} = \begin{bmatrix} e^{-i2\theta}e^{-i2\pi(f_1-f_c)\tau} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{-i2\theta}e^{-i2\pi(f_{N_f}-f_c)\tau} \end{bmatrix},$$

and $R_\phi^{44} = \begin{bmatrix} e^{-i2\theta}e^{-i2\pi(f_1-f_c)\tau}e^{-i2\pi\frac{\tau}{T_s}} & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & e^{-i2\theta}e^{-i2\pi(f_{N_f}-f_c)\tau}e^{-i2\pi\frac{\tau}{T_s}} \end{bmatrix}$.

These exponentials depend on τ and θ, values which must be determined in order to specify $\mathbf{R}_\phi$, but the utility of $\mathbf{R}_\phi$ is that even without knowing these values the property that[3] $\mathbf{R}_\phi^\dagger\mathbf{R}_\phi = I$ exists, is quite useful and plays a role in Gisselquist's algorithm (11). Now we are only left with the amplitude scaling factor, $\frac{A}{2}$. Putting all these vectors and matrices together the discrete Fourier transform of the received signal in vector form may be expressed as (11):

$$x=\frac{A}{2}\mathbf{R}_\phi\mathbf{\Psi}d+n \tag{2.22}$$

The dimensions of these terms are:

$x : 4N_f \times 1$,

$n : 4N_f \times 1$,

$d : N_f \times 1$,

$\mathbf{\Psi} : 4N_f \times N_f$,

$\mathbf{R}_\phi : 4N_f \times 4N_f$.

These expressions make it simpler to manipulate the underlying redundancies in the follow-

[3]The notation $x^\dagger$ is used throughout to refer to the conjugate transpose of a matrix or vector.

ing sections (11). It should be stressed that $\mathbf{R}_\phi$ is not fully specified and so any algorithm that uses this model must make assumptions or estimate the values of τ and θ.

2.1.3 The Distribution of a Burst Transmission Signal, $x(t)$. In equation 2.22 there are three unknown quantities, n, ϕ, and d. Proper statistical analysis will depend on knowing whether each of these quantities is random or deterministic and if random, what the distribution is (11). The goal of this research is to determine a way to estimate PSD of the noise term. Gisselquist completed his estimate of x assuming the noise was multi-variate Gaussian with zero mean and approximately diagonal covariance matrix $\mathbf{R}_n$ (11),

$$(\mathbf{R}_n)_{jj} \triangleq \mathbb{E}\left\{\left|\int_{\frac{-T}{2}}^{\frac{T}{2}} n(t)e^{-i2\pi ft}dt\right|^2\right\} \approx TS_n(f). \tag{2.23}$$

The approximation in equation 2.23 introduces a bias into $\mathbf{R}_n$ that comes from the noise and is significant when $S_n(f)$ changes rapidly (11) in frequency. In his statistical model he assumed this matrix was known. He assumed that T was large enough so that the bias was negligible. This is where this research plays a role, in estimating this noise matrix $\mathbf{R}_n$. There is a constraint on the unknown parameter ϕ, it is unknown, but must be a deterministic parameter because if ϕ were to be treated as random, $x(t)$ would become a stationary process and it is not because N_s is finite. The remaining unknown quantity, d, is usually determined by the modulation type. Prior to the z-transform, the probability distribution of $\{d_n\}$ is usually well specified by the modulation type of interest, in this case BPSK (11). In this scheme d_n is chosen from a finite set of elements, precisely $\{\pm 1\}$. These d_ns may be entirely independent, or they may have some known correlation (11) caused by a repeating pattern within each burst transmission. They may be biased or uniformly distributed (11). The correlation in this data sequence can be expressed spectrally as (11):

$$S_d(e^{i2\pi f}) \triangleq \lim_{N_s\to\infty} \frac{1}{N_s}\mathbb{E}\left\{\left|\sum_{n=0}^{N_s-1} d_n e^{-i2\pi T_s fn}\right|^2\right\} \tag{2.24}$$

Many communications systems transmit uncorrelated symbols leaving this value constant (11). Either way, the probability distribution of $\{d_n\}$ is known from the modulation parameters (11). The probability distribution of $D(e^{i2\pi(f-f_c)T_s})$, the z-transform of $\{d_n\}$, is not so obvious (11). Since $D(e^{i2\pi(f-f_c)T_s})$ is constructed from a sum of random numbers, it may actually be Gaussian for a large enough number of symbols, N_s, by the Central Limit Theorem. If $D(e^{i2\pi(f-f_c)T_s})$ is truly Gaussian, all that is required to specify its distribution is its mean, 0, for most modulation types of interest, and its variance, $R_d = N_s I$ for statistically independent values of d_n having unit magnitude (11). If the d_n are not statistically independent, an approximately diagonal matrix with elements

$$(R_d)_{jj} \approx N_s S_d(e^{i2\pi(f_j - f_c)T_s}) \tag{2.25}$$

may be used instead (11). Gisselquist introduces and proves the following theorem to demonstrate that the Central Limit Theorem does apply as $N_s \to \infty$, and identifies when the assumption that d is Gaussian is valid.

Theorem 1 *Consider a message composed of N_s symbols d_n, where d_n is drawn randomly from some finite set of symbols, D, each having finite energy. Assume also that the d_n have zero mean. Then the discrete time Fourier transform of the data, examined from any angle, ϕ, and any radian frequency, $\omega = 2\pi(f - f_c)T_s$,*

$$\mathrm{Re}\{e^{i\phi}D(e^{i\omega})\} \triangleq \mathrm{Re}\{e^{i\phi}\sum_{n=0}^{N_s-1} d_n e^{-i\omega n}\}, \tag{2.26}$$

is asymptotically Gaussian as $N_s \to \infty$ as long as $\mathbb{E}\left\{\mathrm{Re}\left\{d_n e^{i\phi}e^{-i\omega n}\right\}^2\right\} > 0$ for each n (11).

In the case of BPSK transmissions the theorem applies everywhere except when $\phi = \frac{\pi}{2}$. The exception results in a discontinuity in the probability distribution of $D(e^{i2\pi(f-f_c)T_s})$ as $N_s \to \infty$ (11). For this one exception, the probability distribution of $D(e^{i2\pi(f-f_c)T_s})$ will be approximated as Gaussian and the discontinuity will be treated as if it were non-existent (11). The approximation at this exception point does introduce bias, but it is minimal enough not to cause concern, and so the approximation will be applied throughout this thesis.

Putting all of these probability distributions together, and given that we are *approximating* d as Gaussian, with variance R_d and mean 0, the probability density function of a received BPSK signal can be written as[4] (11):

$$\begin{aligned} f(x,d) = &\left(\tfrac{1}{2\pi}\right)^{\frac{4N_f+N_f}{2}} \det|R_n|^{-\frac{1}{2}} \det|R_d|^{-\frac{1}{2}} \\ &\times e^{\{-\frac{1}{2}(x-\frac{A}{2}R_\phi\Psi d)^\dagger R_n^{-1}(x-\frac{A}{2}R_\phi\Psi d)-\frac{1}{2}d^\dagger R_d^{-1}d\}} \end{aligned} \tag{2.27}$$

When x is known or measured, this probability density function is referred to as the *likelihood* function (11, 3). For finite-time signals, all the statistics of interest are contained within the second moment, for any particular observation length T, the second moment is (11):

$$\begin{aligned} \mathbb{E}_{x,d}\{xx^\dagger\} &= \mathbb{E}_d\left\{\left(\tfrac{A}{2}R_\phi\Psi d\right)\left(\tfrac{A}{2}R_\phi\Psi d\right)^\dagger\right\} + \mathbb{E}_n\left\{nn^\dagger\right\} \\ &= \tfrac{A^2}{4}R_\phi\mathbf{\Psi}R_d\mathbf{\Psi}^\dagger R_\phi^\dagger + R_n \end{aligned} \tag{2.28}$$

Normalizing these components by the observation length $T = N_sT_s$, and assuming uncorrelated symbols, $R_d = N_sI$, as explained in Section 2.1.3 (11),

$$\frac{1}{T}\mathbb{E}\{xx^\dagger\} = \frac{A^2}{4T_s}R_\phi\mathbf{\Psi}R_d\mathbf{\Psi}^\dagger R_\phi^\dagger + \frac{1}{T}R_n \tag{2.29}$$

produces the second moment. Since the signal contribution does not increase as time goes to ∞, taking the limit as $T \to \infty$ is a trivial matter (11). Looking at the terms after such a limit, the PSD of the transmitted signal is (11):

$$S_s(f) = \frac{A^2}{4T_s}\left|\Psi(f-f_c)\right|^2. \tag{2.30}$$

This is part of the PSD of the received signal:

$$S_x(f) = S_s(f) + S_n(f) \tag{2.31}$$

The relevance of these subspace formulas is that the subspace in which the signal of interest resides is smaller than the subspace typically used (11, 28, 15). Because of this, for

[4] A complete derivation of this probability density function can be found in Appendix A.

known R_ϕ, Ψ, and R_n a projection operator can be created, $\mathbf{P} = R_\phi \mathbf{\Psi}(\mathbf{\Psi}^\dagger R_n^{-1} \mathbf{\Psi})^{-1} \mathbf{\Psi}^\dagger R_\phi^\dagger R_n^{-1}$, which will project the received waveform onto the signal's subspace (11, 20). Likewise, an alternate projection operator, $\mathbf{I} - \mathbf{P}$, will project the received waveform onto a noise only subspace (11). Gisselquist showed that a filter for symbol estimation similar to this projection operator achieved the minimum mean square error (11), and will be derived in Section 2.5.

The next section will explore the relevant statistical signal processing models that Gisselquist used in the development of his burst transmission signal statistical model, optimal filtering techniques and then goes on to describe the single sensor minimum mean square error filter that Gisselquist developed for symbol estimation.

2.2 Existing Signal Processing Methods

Convention in signal processing is to classify a signal of interest depending on not just the characteristics of the signal but also by which characteristics were to be exploited. When Gisselquist started his research burst transmission signals, he discovered that none of the existing models adequately accounted for all of the characteristics he was interested in exploiting (11). Thus, using the existing methodologies as a springboard, he defined a new working signal model so that he could create a better processing algorithm. This section describes the stationary and cyclostationary approaches to signal processing and describes why neither of these approaches is adequate for burst transmission signals.

2.2.1 Stationary approach. The stationary approach to signal processing assumes that the signal is stationary, meaning that its probability distribution function is independent of absolute time (23). Thus only signals that are infinite in length are truly stationary. Research in the past has been based on signals of finite length using the assumption that it has a multivariate Gaussian probability distribution. So the received signals are treated as Gaussian random vectors with elements independent in the frequency domain (13). In this manner, the first two moments of the distribution are used to completely specify the probability distribution. The first moment is the expected value, assumed to be zero, and the second is specified with the PSD of the signal. Thus, all the descriptive statistics (parameters) of the signal are contained in the power spectral density. If a PAM signal were treated as stationary, the parameters τ and θ in equation 2.1 would not be of concern since the statistics of stationary signals are independent of time and both of these parameters require a time reference (11). While this is a good thing, there still is the fact that the exploitation methods associated with this model do not allow for the redundancies of BPSK signals to be taken advantage of. So although there are benefits to modeling a PAM or BPSK signal as stationary, it is not the best since much is lost when the redundancies are ignored. Exploiting the redundancies and the fact that burst transmissions are time-limited requires a different approach. Because of this, PAM signals cannot be truly be modeled as stationary (22, 9).

2.2.2 Cyclostationary. Another approach is to treat the signal as cyclostationary. A cyclostationary signal is one whose probability distribution is a periodic or polyperiodic function of time (23). Thus a cyclostationary signal is also one which is independent of time and so ideally is of infinite length. Here the time reference is also important since it determines the phase of the period that the observation lies within (11). So the phase parameter, θ is now unknown and must be estimated for any cyclostationary signal processing algorithm (11). In order to use any classical statistical techniques, a probability distribution is needed to describe the signal. To date, an appropriate probability distribution function has not been found for cyclostationary signals (10). Further, it has been observed that digital signals are in general not Gaussian and therefore cannot be treated as such, yet Gisselquist demonstrates that it is possible, and develops an algorithm based on this demonstration, as explained in section 2.1.3. Without a known probability distribution, the cyclostationary signal processor is left to use only the known moments to glean applicable properties (11).

The advantage of cyclostationary signal processing lies in the moments of the signals (11). Cyclostationary signals have the advantageous property that particular pairs of frequencies are correlated while stationary signals exhibit no such correlation (6, 7, 18). This correlation is identified by the cyclic spectral density function or the spectral correlation function (SCF) and is expressed (7),

$$S_x^{\alpha}(f) \triangleq \lim_{t \to \infty} \frac{1}{T} \mathbb{E}\{X_T^*(f - \frac{\alpha}{2}) X_T(f + \frac{\alpha}{2})\} \tag{2.32}$$

where α is the cycle frequency, or the separation in frequency between two correlated pairs. It corresponds to one of the time periods found in the probability distribution function (11). Only man-made signals have the non-zero spectral correlations when $\alpha \neq 0$. Of these man-made signals, only a finite number of values for α yield non-zero spectral correlations (11).

Typical values of α that produce non-zero spectral correlations are zero, integer multiples of the symbol rate $\frac{k}{T_s}$, twice the carrier frequency $2f_c$, and linear combinations of these (11, 9). For PAM signals, the cyclic spectral densities at integer multiples of the baud rate, $\frac{1}{T_s}$, are known to be (11, 9),

$$S_s^{\frac{k}{T_s}}(f) = e^{-i2\pi \frac{k\tau}{T_s}} \frac{A^2}{4T_s} \Psi^*(f - \frac{k}{T_s} - f_c) \Psi(f + \frac{k}{T_s} - f_c) \mathbb{E}\{|d_n|^2\} \tag{2.33}$$

PAM signals with real valued d_n, as in BPSK signals, have a correlation at twice their carrier frequency as was discussed in Section 2.1.2,

$$S_s^{2f_c}(f) = e^{i2\theta} \frac{A^2}{4T_s} |\Psi(f - f_c)|^2 \mathbb{E}\{|d_n|^2\}. \tag{2.34}$$

When using these moments, one assumption must be made: only the signal of interest is correlated at a particular value of α, implying that the chosen α is non-zero (11). Then mathematically, the spectral density of the received signal, $S_x^{\alpha}(f)$, is identical to the spectral density of the signal of interest, $S_s^{\alpha}(f)$ (11). This assumption justifies the philosophy that estimating this value will separate the signal of interest from the received signal, and as such all the other noise and interference in the environment (11). This is how signal selectivity is achieved and is the basis for cyclostationary signal processing. If the estimate for α is non-zero, the signal is present (11, 8).

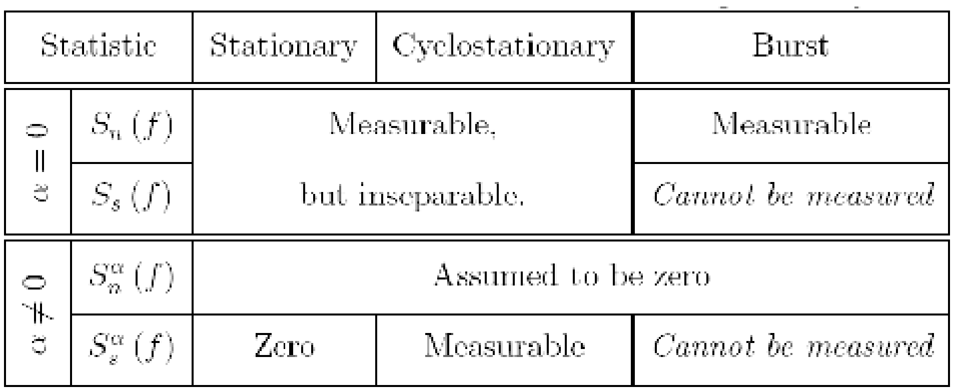
There are two big disadvantages when using this technique. The first is the noise, and the second is that the cyclic correlation function is unknown and must be estimated (11). The noise problem is that it increases the variance in any estimate of the cyclic correlation function, $\hat{S}_x^\alpha(f)$ (11). Thus, despite the assumption that only signal will contribute, noise variations can easily create other apparent contributions (11). The extra variation can be dealt with by averaging the statistic over longer and longer observation periods while requiring some amount of smoothness in the estimated SCF (11, 6). This is appropriate for working with signals of exceptionally long durations buried in noise, but is not applicable for short duration signals of interest, which is the category that burst transmissions fall into (11).

Treating a PAM signal as a cyclostationary signal allows the redundancy property to be taken advantage of, but assumes complete correlation in the frequency domain. This assumption then allows the PSD of the transmitted signal to be estimated, not taking into consideration the bias caused by the noise, since the noise contribution is assumed zero. This is not applicable to burst transmission signals, since the correlation in the frequency domain for burst transmission signals is not completely correlated, especially so when dealing with bursts of different lengths, or when dealing with bursts that do contain some sort of correlation between the transmitted data symbols.

2.3 A Better Signal Model for Burst Transmission Signals

Although each of these previously mentioned techniques has merit, they also have some obvious disadvantages. Since the spectral correlation functions are only defined for infinite length signals, (11) none of the discussed methods apply completely to burst signal transmissions and thus the noise estimation for each is not valid.

The table in Figure 2.3 is helpful in understanding the differences between stationary and cyclostationary approaches of signal processing. The PSD of the signal can be measured if the signal is stationary, but the noise contribution cannot be separated from the signal. When a signal is considered cyclostationary the techniques focus on the places where only the signal contributes, that is where the values of α are non-zero (11). By measuring these properties and averaging them over time an estimate for the signal can be obtained that is theoretically free of noise. The key here is that neither of these techniques works for burst transmission signals since the Spectral Correlation Functions (SCF) are



Statistic		Stationary	Cyclostationary	Burst
$\alpha = 0$	$S_n(f)$	Measurable,		Measurable
	$S_s(f)$	but inseparable.		*Cannot be measured*
$\alpha \neq 0$	$S_n^{\alpha}(f)$	Assumed to be zero		
	$S_s^{\alpha}(f)$	Zero	Measurable	*Cannot be measured*

Figure 2.3 The Spectral Density Problem with Classical Burst Signal Analysis

only defined for infinite length signals, making them inappropriate for time limited signals (11).

The stationary approach is advantageous since some of the unknown parameters disappear in the frequency domain, and since a probability distribution function is known.

Cyclostationary techniques account for more of the true characteristics of the signal, making the methods that use them more accurate. The drawback lies in that cyclostationary techniques have no probability distribution function, making conventional statistical methods difficult to apply. The working signal model for burst transmission signals allowed Gisselquist to apply classical statistical principles such as Maximum Likelihood Estimator (MLE) and likelihood ratios (11) in the development of a demodulator. The next section briefly discusses optimal filtering techniques before presenting the derivation of Gisselquist's Minimum Mean Square Error (MMSE) filter.

2.4 Optimal Filtering

The application of Gisselquist's research was focused on optimal filtering. In the development of Gisselquist's filter, the question of what filter should be applied to get the "best" results is answered (11). That is, he develops a filter that obtains a reliable estimate of the transmitted data symbols in the face of colored or white noise. Here we are concerned with improving his filtering process by obtaining a more reliable estimate of the noise PSD since the PSD of the noise is an important part of the MMSE filter. The

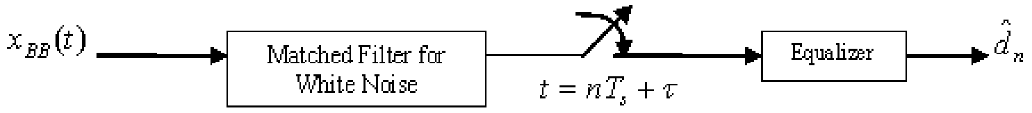


Figure 2.4 The Structure of an Optimal Filter

optimal filter to obtain the "best" results is known to consist of a matched filter for white noise, followed by a Tapped Delay Line (TDL) equalizer (11) as shown in Figure 2.4.

This optimal filter is applicable regardless of the criteria used to define "best" (5). Specific to our purposes, "best" means obtaining the most reliable estimate of the original data symbols. In this figure, $x_{BB}(t)$ represents a signal with zero carrier frequency, $f_c = 0$, t is the time the sample is taken, and $\hat{d}_n$ is the estimated symbol at the output of the filter (11). This section will follow the form of the filter, first discussing matched filtering and then the combination of a matched filter with equalization.

2.4.1 Matched Filtering. A matched filter is defined as a unique filter that maximizes the signal-to-noise ratio of a known input (11). Common developments of this are presented in (11, 24, 25, 27). In each development, the signal is assumed to be known, for example $s(t) = A\psi(t)$ with a constant gain factor A (11). The filter that maximizes the signal-to-noise ratio in white noise is (11, 19, 25)

$$h_{MF}(t) = \psi^*(-t) \tag{2.35}$$

in the frequency domain this filter is

$$H_{MF}(f) = \Psi^*(f) \tag{2.36}$$

There are other developments such as (4) which extend this filter to a colored noise environment (11). The matched filter is obviously the optimum filter when only one data symbol is transmitted, but not when multiple pulses are transmitted in succession (11) as in a burst transmission signal. When multiple pulses are transmitted, the pulses may interfere with each other at the output of the filter causing Intersymbol Interference (ISI)(11, 24). Such interference is problematic when $\Psi(f)$ has been designed and fixed in the transmitter, prior to the estimation of $S_n(f)$ (11).

2.4.2 Equalization. Removing the ISI is accomplished through the use of an equalizer (11). Ericson proves, using the spectral redundancies inherent in a modulated signal, that the optimal receiving filter always consists of a matched filter followed by a Tapped Delay Line (TDL) equalizer (11, 5). This filter can be implemented as a discrete time filter operating on the symbols that are sampled after the matched filter (11). Using this structure, there are two ways of applying a filter, fixed and adaptive (11).

A fixed equalizer can be designed by minimizing the mean square error (MSE) between the output and the true symbols that were sent (11). Berger and Tufts present one such Minimum Mean Square Error (MMSE) filter for a PAM communication stream (11, 1). First they modulated the communications signal as a sum of delayed pulses with unknown weights (11),

$$s(t) = \sum_{n=-\infty}^{\infty} d_n \psi(t - nT_s). \tag{2.37}$$

This is similar to the $s(t)$ in equation 2.1 with the exception that burst transmission signals have a limited number of data symbols, N_s. They derive their filter from the PSD of the noise, $S_n(f)$, together with the PSD of the discrete pulse sequence$\{d_n\}_{n=-\infty}^{\infty}$, written as $S_d(e^{i2\pi f T_s})$ (11). They point out that this filter is comprised of both a periodic and non-periodic component,

$$H(f)_{MMSE_{\text{Berger and Tufts}}} \underbrace{\left(\frac{S_d(e^{i2\pi f T_s})}{1 + S_d(e^{i2\pi f T_s}) \frac{A^2}{T_s} \sum_{n=-\infty}^{\infty} \frac{\left|\Psi(f - \frac{n}{T_s})\right|^2}{S_n(f - \frac{n}{T_s})}} \right)}_{\text{TDL Equalizer}} \underbrace{\frac{\Psi^*(f)}{S_n(f)}}_{\text{Matched Filter}} \tag{2.38}$$

where the periodic portion is the TDL equalizer, and the non-periodic portion is the matched filter (11). The periodic portion has periodicity $\frac{1}{T_s}$, making it a TDL equalizer and matching Ericson's prediction (11, 5).

The second type of equalizer is an adaptive equalizer (11). These equalizers were first demonstrated by Lucky (14) in 1966 and later studied by Haykin and many others (11, 12, 2). The problem with these adaptive equalizers lies in their implementation, since no knowledge of the noise PSD nor the true pulse shape is used (11). Without using the noise PSD to filter out narrow band interference, large amounts of interference may enter into the system (11). Without using the true pulse shape, signal energy is arbitrarily lost in the initial filter (11). While the TDL equalizer may be able to reduce any resulting distortion, it cannot fundamentally compensate for poor signal-to-noise conditions in the received signal (11). Thus these systems are less than optimal, despite their practicality.

2.5 A Minimum Mean Square Error Filter for Burst Transmission Signals

The MMSE filter has well known optimization properties, and is well defined when the pulse shape and the noise PSD are defined (11). However, Berger and Tuft's version of the MMSE filter does not exploit the extra spectral redundancies found in a BPSK signal at bandpass frequencies discussed in Section 2.1.2 (11). Gisselquist created a MMSE filter that gives the "best" estimate of the d_n provided the pulse shape is known and a reliable estimate of the noise spectrum is available.

The first step for Gisselquist in the development of his MMSE filter was to find the $\hat{\boldsymbol{d}}$ that satisfies:

$$\hat{\boldsymbol{d}}(x)_{MMSE} \triangleq \arg\min_{\hat{\boldsymbol{d}}(x)} \mathbb{E}\left\{ \left(\hat{\boldsymbol{d}}(x) - d\right)^{\dagger} \left(\hat{\boldsymbol{d}}(x) - d\right) \right\}. \tag{2.39}$$

The statistical solution to this minimization is the expected value of the conditional probability distribution or $\hat{\boldsymbol{d}}(x)_{MMSE} = \mathbb{E}\{d|x\}$ which is known to minimize the mean square error when an a-priori probability distribution for d is known (11, 21). Since d and x are Gaussian, or treated as Gaussian, as explained in Section 2.1.3 the conditional probability distribution of d given x is also Gaussian with mean (11),

$$\mathbb{E}\{d|x\} = \frac{A}{2}\left(\frac{A^2}{2}(\boldsymbol{\Psi}^{\dagger} R_n^{-1} \boldsymbol{\Psi} + R_d^{-1}\right)^{-1} \boldsymbol{\Psi}^{\dagger} R_{\phi}^{\dagger} R_n^{-1} x, \tag{2.40}$$

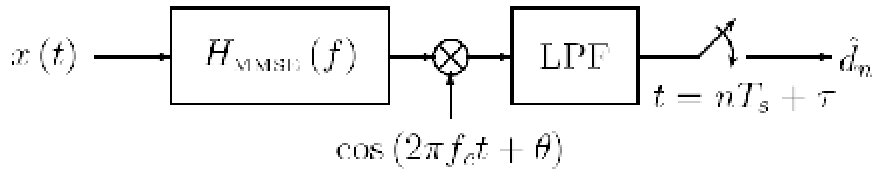


Figure 2.5 Bandpass MMSE (Gisselquist's) System Diagram

and variance (11),

$$\mathbb{E}\{dd^\dagger|x\} = \left(\frac{A^2}{4}(\boldsymbol{\Psi}^\dagger R_n^{-1}\boldsymbol{\Psi} + R_d^{-1}\right)^{-1}. \tag{2.41}$$

So the MMSE estimate of d is (11),

$$\hat{\boldsymbol{d}}(x)_{MMSE} = \left[\frac{A}{2}\left(\frac{A^2}{2}(\boldsymbol{\Psi}^\dagger R_n^{-1}\boldsymbol{\Psi} + R_d^{-1}\right)^{-1}\boldsymbol{\Psi}^\dagger R_\phi^\dagger R_n^{-1}\right]x, \tag{2.42}$$

which is an optimal filter for recovering d. To see this we use the matrices provided in Section 2.1.2.1 to calculate a new form of this filter. This vector of data symbol estimates will have dimension $N_f \times 1$ and after simplification and rearrangement look like (11),

$$\hat{\boldsymbol{D}}(e^{i2\pi(f_j-f_c)T_s}) = \begin{array}{l} e^{-i\theta}e^{i2\pi(f_j-f_c)\tau}H(f_j)X_T(f_j) \\ +e^{-i\theta}e^{i2\pi(f_j-f_c)\tau}e^{i2\pi\frac{\tau}{T_s}}H(f_j+\frac{1}{T_s})X_T(f_j+\frac{1}{T_s}) \\ +e^{i\theta}e^{i2\pi(f_j-f_c)\tau}H^*(2f_c-f_j)X_T^*(2f_c-f_j) \\ +e^{i\theta}e^{i2\pi(f_j-f_c)\tau}e^{i2\pi\frac{\tau}{T_s}}H(2f_c-f_j-\frac{1}{T_s})X_T(2f_c-f_j-\frac{1}{T_s}) \end{array} \tag{2.43}$$

Then, expanding equation 2.42 to solve for $H(f)$, we see that the minimum mean square error filter for BPSK modulation is (11)

$$H_{MMSE}(f) = \frac{\frac{A}{2}\frac{\Psi^*(f-f_c)}{T_sS_n(f)}}{1+\frac{A^2}{4}\frac{|\Psi(f-f_c)|^2}{T_sS_n(f)}+\frac{A^2}{4}\frac{\left|\Psi\left(f+\frac{1}{T_s}-f_c\right)\right|^2}{T_sS_n\left(f+\frac{1}{T_s}\right)}+\frac{A^2}{4}\frac{|\Psi(f_c-f)|^2}{T_sS_n(2f_c-f)}+\frac{A^2}{4}\frac{\left|\Psi\left(f_c-f-\frac{1}{T_s}\right)\right|^2}{T_sS_n\left(2f_c-f-\frac{1}{T_s}\right)}} \tag{2.44}$$

for $f \in (f_c - \frac{1}{T_s}, f_c)$. The implementation of this filter is illustrated in Figure 2.5.

To implement the filter an estimate of the PSD of the noise must be obtained. Since the received signal has both the transmitted signal and the noise intertwined as explained in

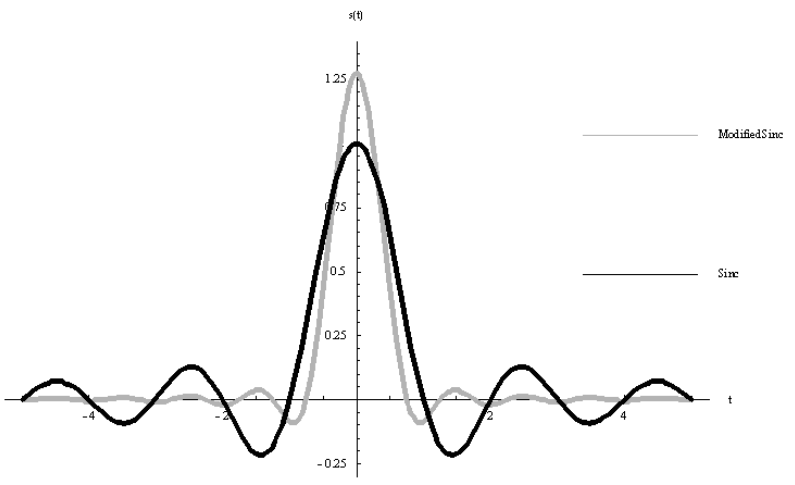


Figure 2.6 A Sinc Function vs. Modified Sinc

Section 2.1.1, methods to estimate the noise must be established. In Chapter III, methods for estimating this noise are discussed and the success evaluated in Chapter IV.

2.6 Preliminaries

Gisselquist's filter is optimal in terms of mean square error when demodulating BPSK burst transmissions (11), assuming a colored noise environment. This thesis is concerned with estimating the PSD of white noise. The results will hopefully lead to effective estimation of colored noise. The first experiment generated a burst transmission signal as in equation 2.1. The pulse shape used was a modified Nyquist[5] pulse shape, specifically a modified sinc:

$$\psi(t) = \frac{1}{T_s}\left[\mathrm{sinc}(\frac{2t}{T_s} - \frac{1}{2}) + \mathrm{sinc}(\frac{2t}{T_s} + \frac{1}{2})\right]. \tag{2.45}$$

Although the sinc pulse is infinite as a function of time, the magnitude decreases as $|t|$ increases, as can be seen in Figure 2.6. The advantage of the modified Nyquist pulse can also be seen in Figure 2.6 because the magnitude decreases even faster than the unmodified sinc. Figure 2.7 shows a burst transmission signal using the following parameters: $A = 1$, $f_c = 1$, $\tau = 0$, $\theta = 0$, $T_s = 1$, $N_s = 64$, and $\psi(t)$ was as in equation 2.45. The equation

[5] For the purposes of this thesis $\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$.

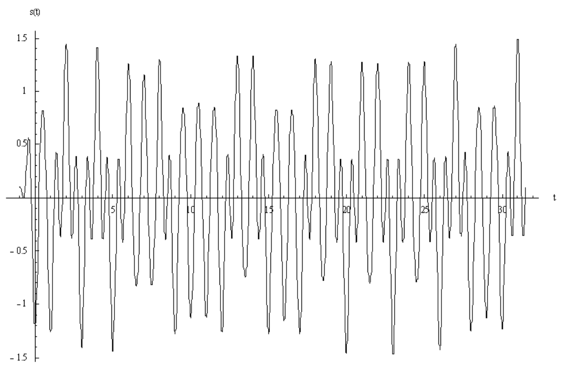


Figure 2.7 BPSK signal with $N_s = 64$

plotted in Figure 2.7 is:

$$s(t) = \mathrm{Re}(\sum_{n=0}^{63} d_n \psi(t-n) e^{i(2\pi t)}). \tag{2.46}$$

This signal is then combined with white noise multiplied by a noise scale as follows:

$$x(t) = \mathrm{Re}(\sum_{n=0}^{63} d_n \psi(t-n) e^{i(2\pi t)}) + (\text{noise scale})\ n(t). \tag{2.47}$$

The noise scale in the simulated received signal, $x(t)$, was allowed to vary incrementally from $10^{-1.5}$ to $10^{1.5}$ while $s(t)$ and $n(t)$ remained the same. Thus the only variable allowed to change was the noise scale. The noise PSD, $S_n(f)$, used in the MMSE filter was simply white noise with a noise scale factor of 1. Thus the shape of the $S_n(f)$ used in the filter differed in both shape and scale from the actual $S_n(f)$ in the simulated received signal. For each noise scale increment, the received signal was processed using equation 2.43 and symbol estimates, $\hat{d}_n$ obtained. These $\hat{d}_n$ were compared to the originally transmitted symbols, d_n, and the bit error rate (BER) recorded. The BER is defined as

$$BER = \frac{\text{Number of Incorrectly Identified Symbols}}{\text{Total Number of Transmitted Symbols}} \tag{2.48}$$

The success of the filter is plotted as a function of the noise scale in Figure 2.8 and BER, and in Figure 2.9 as a function of the Signal to Noise Ratio (SNR) in dB .

These figures show two important results. First, as the noise scale increases, the success of the filter decreases. The second, when the noise scale used in the MMSE filter

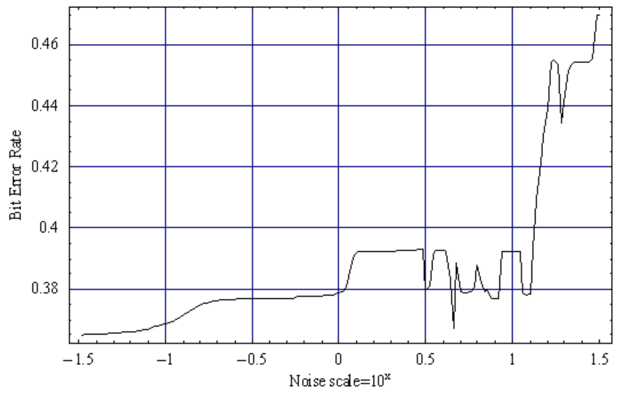


Figure 2.8 BER vs. Noisescale for Unknown Scale and Shape of the Noise PSD

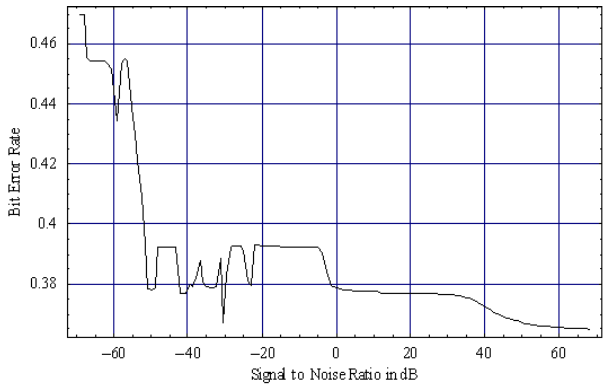


Figure 2.9 BER vs. SNR for Unknown Shape and Scale of the Noise PSD

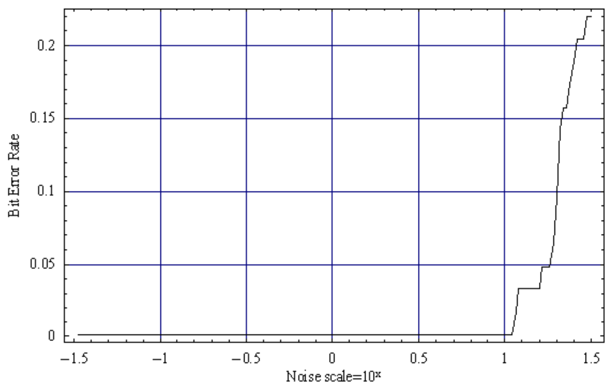


Figure 2.10 BER vs. Noisescale for Known Scale and Unknown Shape of the Noise PSD

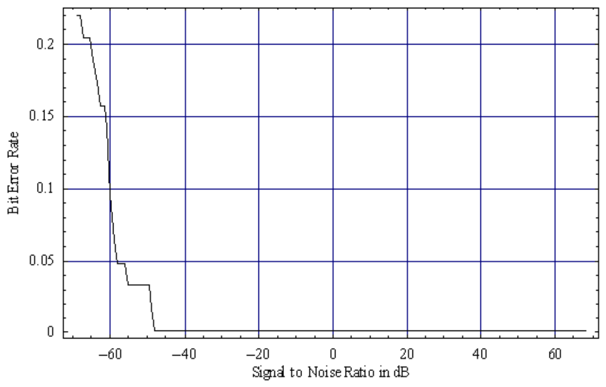


Figure 2.11 BER vs. SNR for Known Scale and Unknown Shape of the Noise PSD

is close to the actual noise scale, the success increases. Likewise, the farther that the noise scale used in the MMSE filter is from the actual noise scale, the poorer the symbol estimation results. This was further examined in a second experiment.

In the second experiment, the received signal was generated exactly as in the first experiment. The difference in the second experiment is that the PSD of the noise, $S_n(f)$ used in the MMSE filter was constant, the constant being the noise scale used in the signal generation. So the MMSE filter has a correct estimate of the scale of $S_n(f)$ but not of the shape of $S_n(f)$. The BER is low for noise scale values < 1, but then sharply increased as the noise scale values increased as shown in Figures 2.10 and 2.11.

The important thing to note here is that the success drops off rapidly when the noise is much larger than the signal, most likely since the shape of the noise PSD, $S_n(f)$ is incorrect. This suggested the need for a third experiment.

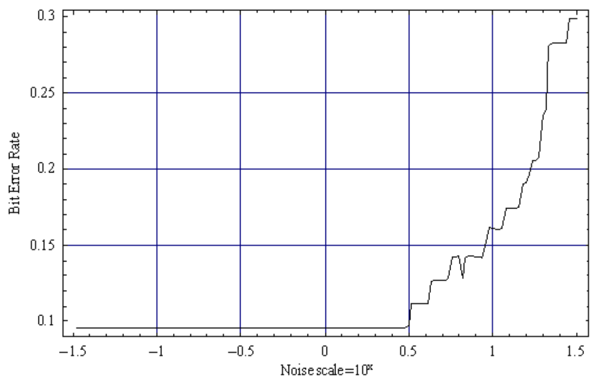


Figure 2.12 BER vs. Noisescale for Known Shape and Unknown Scale of the Noise PSD

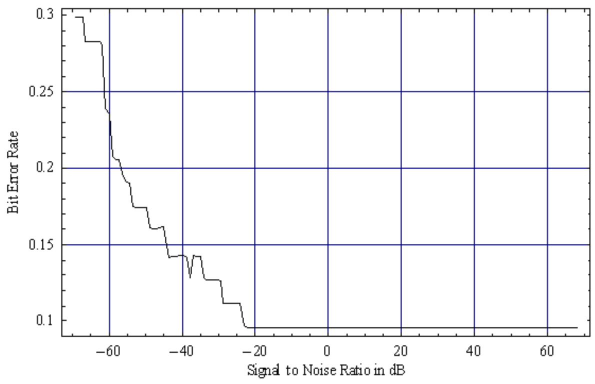


Figure 2.13 BER vs. SNR for Known Shape and Unknown Scale of the Noise PSD

The third experiment was conducted identical to the first with the exception that the PSD of the noise, $S_n(f)$, used in the MMSE was the actual PSD of the noise used in the simulated signal. That is, the $S_n(f)$ used in the filter had a shape identical to the actual $S_n(f)$, but the noise scale was incorrect. The exception being when the noise scale equaled 1. The results of this third experiment were very similar to the first, indicating that although the shape of the PSD of the noise, $S_n(f)$, used in the filter is important, it may not be as important as the scale. These results can be seen in Figures 2.12 and 2.13.

These results show the importance of a good estimate of $S_n(f)$ for use in the filter, thus providing the original motivation for this thesis. Although the improvement of the MMSE filter was the initial motivation for solving this problem, it will be shown in Chapter IV, that even an estimate of $S_n(f)$ that is correct in both shape and scale does not guarantee the absolute success of the MMSE filter. Thus the filter is not a good way to indicate the success of a particular noise estimation technique. Since the MMSE filter is optimized for a colored noise environment it is not expected to be the best measure for evaluating a white noise estimation technique.

III. Theoretical Development

The noise estimation techniques presented in this thesis are based on two properties of PAM signals. The first property is the fact that any received PAM signal can be expressed in time as a sum of the transmitted signal and the noise as follows:

$$x(t) = s(t) + n(t) \tag{3.1}$$

The problem is that the individual quantities of $s(t)$ and $n(t)$ are unknown. Since they are unknown, exploitation becomes a problem that must be solved.

The noise estimation techniques are also based on the redundancies discussed in Section 2.1.2. As stated these redundancies were discovered by Nyquist, exploited first by Berger and Tufts, and then by Gisselquist in the development of his burst transmission subspace theory (11). Gisselquist exploited these redundancies in the development of his Minimum Mean Square Error (MMSE) filter, which produced better results than any previous work (11). It would seems that equation 3.1 would be the obvious choice to use in combination with Gisselquist's filter, however, the results in Chapter 4 will show that this is not the best approach to use, since the success of Gisselquist's filter depends on an accurate a priori estimation of the Power Spectral Density (PSD) of the noise. Additionally, it is not the best solution since Gisselquist's filter works best with colored noise, but gives acceptable results with white noise when the noise scale is known (11). The first section in this chapter explains the theory behind a method based on equation 3.1 and the following sections provide better alternatives.

3.1 Method One: Subtract in Time

Equation 3.1 provides a seemingly simple solution to the noise estimation problem, extract the transmitted signal from the received signal and only the noise will remain. Gisselquist's MMSE provides an estimate of the content of a BPSK signal which is described in equations 2.43 and 2.44. This method proposes using this filter on a received signal to recover an initial estimate for the originally transmitted BPSK symbols. Since an a priori estimate for the noise PSD is required, $S_n(f)$, a small constant value is used, since the expected value of Gaussian white noise over an infinite amount of time is known to be constant. The received signal is processed as in Figure 2.5, providing an estimate for the originally transmitted signals, call it $\hat{\mathbf{d}}_1$. The pulse shape, ψ, gain, A, symbol length, T_s, number of symbols, N_s, time of arrival, τ, and the phase of the carrier, θ, are all assumed to be known or previously recovered. d_n is replaced with $\hat{\mathbf{d}}_1$ in

$$s(t) = A\,\mathrm{Re}(\sum_{n=0}^{N_s-1} d_n\psi(t - nT_s - \tau)e^{i(2\pi f_c t+\theta)}) \tag{3.2}$$

to create a new signal, $\hat{s}_1(t)$. The signal is then sampled to create a discrete form of $\hat{s}_1(t)$. Then $\hat{s}_1(t)$ is subtracted from equation 3.1 giving:

$$x(t) - \hat{s}_1(t) = s(t) + n(t) - \hat{s}_1(t) = \hat{n}_1(t) \tag{3.3}$$

The power spectral density of $\hat{n}_1(t)$ is calculated and used in the MMSE filter to obtain an second estimate of the data symbols, $\hat{\mathbf{d}}_2$. This estimate of the originally transmitted data symbols is then used to construct a new signal estimate in the time, $\hat{s}_1(t)$, assuming the pulse shape is known. Using the property presented in equation 3.1, $\hat{s}_2(t)$ is subtracted from $x(t)$ leaving $\hat{n}_2(t)$. The noise estimate is converted into the frequency domain using the DFT, the PSD calculated, and then $\hat{S}_{n2}(f)$ is used in the MMSE filter to recover a third estimate for the data symbols, $\hat{\mathbf{d}}_3$, the process continuing until the $\hat{\mathbf{d}}_i$ converge. A flow chart for this method is illustrated in Figure 3.1.The method seems to be ideal, but since the success of the original filtering process depends on an accurate estimate of the noise PSD, it will most likely not converge to the actual d_n. Given that the filter depends on an accurate estimate of the noise PSD, it seems intuitive to use the received signal in the noise estimation process. The remaining sections of this chapter do exactly that, and combined with the redundancy property, seem to be better methods in the noise estimation problem.

3.2 Methods Using the Signal Redundancies

Given that there are two properties discussed at length within this thesis, it seems only natural that the ideal estimation techniques presented should take advantage of both.

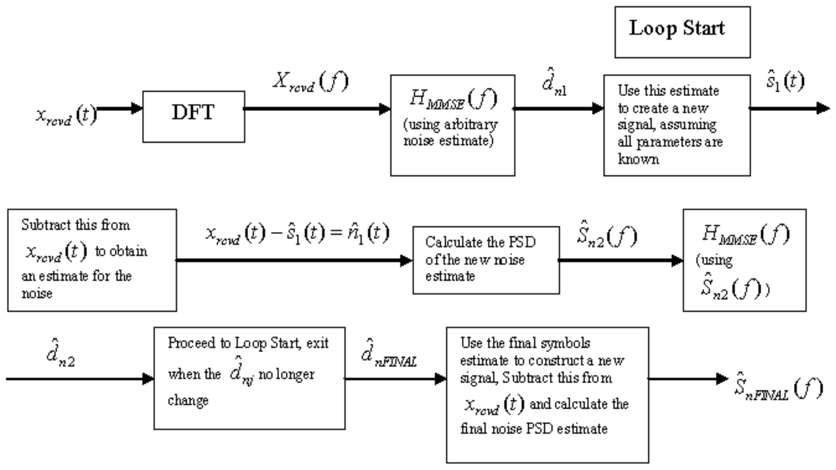

Loop Start
$x_{rcvd}(t)$
DFT
$X_{rcvd}(f)$
$H_{MMSE}(f)$ (using arbitrary noise estimate)
$\hat{d}_{n1}$
Use this estimate to create a new signal, assuming all parameters are known
$\hat{s}_1(t)$
Subtract this from $x_{rcvd}(t)$ to obtain an estimate for the noise
$x_{rcvd}(t)-\hat{s}_1(t)=\hat{n}_1(t)$
Calculate the PSD of the new noise estimate
$\hat{S}_{n2}(f)$
$H_{MMSE}(f)$ (using $\hat{S}_{n2}(f)$)
$\hat{d}_{n2}$
Proceed to Loop Start, exit when the $\hat{d}_{nj}$ no longer change
$\hat{d}_{nFINAL}$
Use the final symbols estimate to construct a new signal, Subtract this from $x_{rcvd}(t)$ and calculate the final noise PSD estimate
$\hat{S}_{nFINAL}(f)$

Figure 3.1 Flow Diagram for Method One: Subtract in Time

In this section, four different methods will be presented. All exploit the redundancy properties, and all will hopefully provide promising results for use in the MMSE filter and in the larger non Gaussian noise estimation problem. First, the development of the redundancies as they apply to the PSD will be presented, and then an in depth explanation of each method will follow.

Since both the Fourier transform and the PSD are linear when dealing with burst transmission signals, the sum property presented in Section 2.1.1 can be easily extended into the frequency domain. It is known that the Fourier transform of equation 3.1 is:

$$X_T(f) = N_T(f) + \frac{A}{2}e^{i\theta}\sum_{n=0}^{N_s-1} e^{-i2\pi(f-f_c)\tau}[d_n e^{-i2\pi(f-f_c)nT_s}]\int_{-\infty}^{\infty}\psi(u)e^{-i2\pi(f-f_c)u}du. \quad (3.4)$$

As presented in Subsection 2.1.3 the PSD of the transmitted signal, $S_s(f)$, is of the form: $\frac{A^2}{4T_s}|\Psi(f-f_c)|^2\mathbb{E}\{|d_n|^2\}$. To simplify this equation, let $\mathbb{E}\{|d_n|^2\} = S_d(f)$ and $k = \frac{A^2}{4T_s}$, resulting in:

$$S_s(f) = k|\Psi(f-f_c)|^2 S_d(f). \quad (3.5)$$

Removing the carrier frequency by moving it to $f_c = 0$ results in:

$$S_s(f) = k\,|\Psi(f)|^2\,S_d(f) \quad (3.6)$$

This is summed with the PSD of the noise $S_n(f)$ to give (11, 23):

$$S_x(f) = S_s(f) + S_n(f). \tag{3.7}$$

$S_x(f)$ has the same redundancy properties presented in Section 2.1.2 and its vector is of the form:

$$\mathbf{S}_x(f) = \begin{bmatrix} S_x(f_1) \\ \vdots \\ S_x(f_{N_f}) \\ S_x(f_1 + \frac{1}{T_s}) \\ \vdots \\ S_x(f_{N_f} + \frac{1}{T_s}) \\ S_x^*(2f_c - f_1) \\ \vdots \\ \vdots \\ S_x^*(2f_c - f_{N_f} + \frac{1}{T_s}) \end{bmatrix} \tag{3.8}$$

The redundancies for a given frequency, $f_i \in (f_c - \frac{1}{T_s}, f_c)$, occur at $S_x(f_i)$, $S_x(f_i + \frac{1}{T_s})$, $S_x^*(2f_c - f_i)$, and $S_x^*(2f_c - f_i + \frac{1}{T_s})$. It is important to note that at these locations, the information is the same, not the actual values. The PSD of the transmitted signal can be further broken down into the PSD of the transmitted symbols and the magnitude of the Fourier Transform of the pulse shape. Both are scaled by a constant, $k = \frac{A^2}{4T_s}$, where A is the gain of the received signal and T_s is the symbol duration. In this section the constant value, k, and the pulse shape are assumed to be known. The first step in all the methods will divide by the magnitude of the pulse shape and the constant, k, to give:

$$\frac{\mathbf{S}_x(f)}{k\,|\Psi(f)|^2} = \mathbf{S}_d(f) + \frac{\mathbf{S}_n(f)}{k\,|\Psi(f)|^2} \tag{3.9}$$

That division occurs for each redundancy at a corresponding location in the pulse shape. Also, keep in mind that each piece has the same data, thus the $S_d(f)$ are all assumed to be the same regardless of the frequency. The explanation will continue using the following notation,

$$\mathbf{F}(f) = \frac{S_x(f)}{k\,|\Psi(f)|^2} = \mathbf{S}_d(f) + \frac{\mathbf{S}_n(f)}{k\,|\Psi(f)|^2} \tag{3.10}$$

When referring to the redundancies, they will be called $F(f^j)$, where $j = 1, 2, 3, 4$. Specifically f^1 refers to the values having the form $S_x(f_i)$, f^2 refers to the $S_x(f_i + \frac{1}{T_s})$ values, f^3 the values at $S_x^*(2f_c - f_i)$, and f^4 the values at $S_x^*(2f_c - f_i + \frac{1}{T_s})$, $i \in [1, N_f]$.

The redundancies and the sum in equation 3.1 are the two properties exploited in this section when obtaining an estimate for $S_n(f)$. The estimation techniques are similar, but have important differences that will hopefully produce distinct results. Thus, although repetitive, each method will be explained starting from equation 3.10. Which method to use should be based on the known and unknown parameters of the received signal and what the estimate will be used for. The explanation of each method will begin at this point.

3.2.1 Subtraction of the Smallest Redundancy.

There are two methods that use the smallest redundancy value to extract an estimate for $S_n(f)$. Starting with equation 3.10, $F(f)$ is evaluated at f^1, f^2, f^3, and f^4. The smallest $F(f^j)$ in absolute value is selected. The smallest of the four values will have the smallest value for $S_n(f)$ since the $S_d(f)$ are the same at each location. Call this new vector of smallest values: $\hat{\mathbf{S}}_{d_{small}}(f)$. This vector of smallest values is used in the following two methods.

3.2.1.1 Subtraction Before Pulse Shape Multiplication.

If we subtract this vector, $\hat{S}_d(f)$, from the values in equation 3.9 we will get:

$$\frac{S_x(f)}{k\,|\Psi(f)|^2} - \hat{S}_{d_{small}}(f) = S_d(f) - \hat{S}_{d_{small}}(f) + \frac{S_n(f)}{k\,|\Psi(f)|^2} \tag{3.11}$$

At this point $(S_d(f) - \hat{S}_d(f))$ is assumed to be small, so the remaining values are the estimates for the PSD of the noise, divided by $k\,|\Psi(f)|^2$. To obtain $\hat{S}_n(f)$, equation 3.11 is multiplied by $k\,|\Psi(f)|^2$ resulting in:

$$S_x(f) - \hat{S}_{d_{small}}(f)k\,|\Psi(f)|^2 = \hat{S}_n(f) \tag{3.12}$$

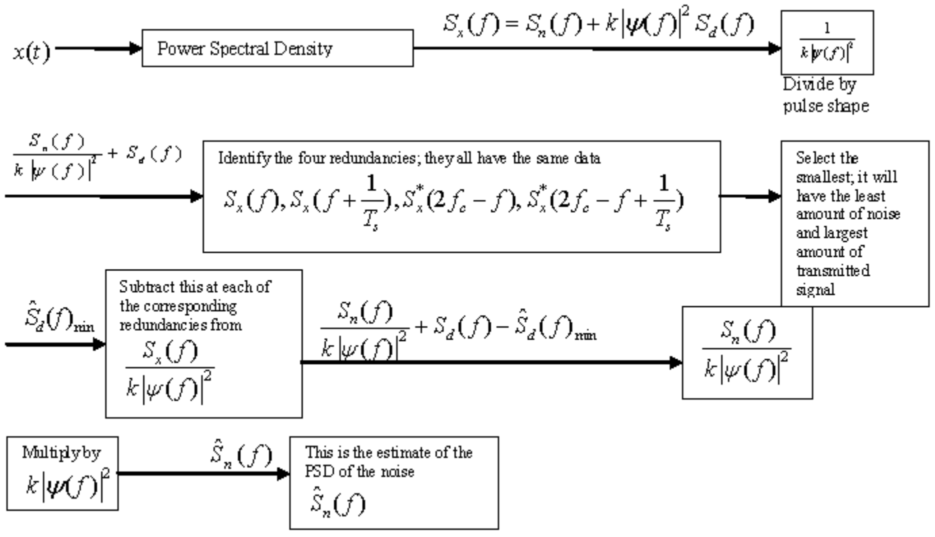

$x(t)$
Power Spectral Density
$S_x(f) = S_n(f) + k|\psi(f)|^2 S_d(f)$
$\frac{1}{k|\psi(f)|^2}$
Divide by pulse shape
$\frac{S_n(f)}{k|\psi(f)|^2} + S_d(f)$
Identify the four redundancies; they all have the same data
$S_x(f), S_x(f+\frac{1}{T_s}), S_x^*(2f_c - f), S_x^*(2f_c - f + \frac{1}{T_s})$
Select the smallest; it will have the least amount of noise and largest amount of transmitted signal
$\hat{S}_d(f)_{\min}$
Subtract this at each of the corresponding redundancies from $\frac{S_x(f)}{k|\psi(f)|^2}$
$\frac{S_n(f)}{k|\psi(f)|^2} + S_d(f) - \hat{S}_d(f)_{\min}$
$\frac{S_n(f)}{k|\psi(f)|^2}$
Multiply by $k|\psi(f)|^2$
$\hat{S}_n(f)$
This is the estimate of the PSD of the noise $\hat{S}_n(f)$

Figure 3.2 Diagram of Subtraction of Minimum Redundancy before Pulse Shape Multiplication

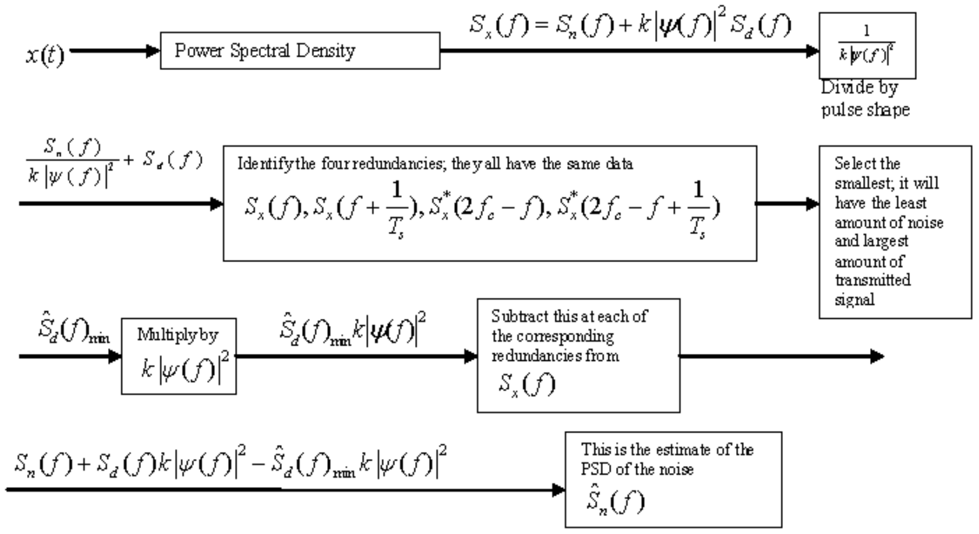
The degree of correctness of this estimation will be demonstrated and measured in Chapter IV.

3.2.1.2 Subtraction After Pulse Shape Multiplication. If we first multiply this vector, $\hat{S}_d(f)$, by $k\,|\Psi(f)|^2$, and then subtract this value from equation 3.7 we will get:

$$S_x(f) - k\,|\Psi(f)|^2\,\hat{S}_{d_{small}}(f) = \hat{S}_n(f) \tag{3.13}$$

The method differs in where the subtraction occurs. Instead of subtracting in the manner of equation 3.11, the vector of smallest values, $\hat{S}_{d_{small}}(f)$, is multiplied by the magnitude of the pulse shape, the scalar k, and then subtracted from equation 3.7. Again $S_d(f) - \hat{S}_{d_{small}}(f)$ is assumed to be small, so the remaining value, is $\hat{S}_n(f)$ is the estimate for the PSD of the noise. Thus, subtracting before or after multiplication by the pulse shape produces the same result, $\hat{S}_n(f)$.

3.2.2 Subtraction of the Average Redundancy. For the methods that subtract the average redundancies, we start with equation 3.9. These methods take the four redundancies: $\mathbf{F}(f) = \frac{S_x(f)}{k|\Psi(f)|^2} = \mathbf{S}_d(f) + \frac{\mathbf{S}_n(f)}{k|\Psi(f)|^2}$, called $F(f^j)$, where $j = 1, 2, 3, 4$ and



$x(t)$
Power Spectral Density
$S_x(f) = S_n(f) + k|\psi(f)|^2 S_d(f)$
$\frac{1}{k|\psi(f)|^2}$
Divide by pulse shape
$\frac{S_n(f)}{k|\psi(f)|^2} + S_d(f)$
Identify the four redundancies; they all have the same data
$S_x(f), S_x(f+\frac{1}{T_s}), S_x^*(2f_c - f), S_x^*(2f_c - f + \frac{1}{T_s})$
Select the smallest; it will have the least amount of noise and largest amount of transmitted signal
$\hat{S}_d(f)_{min}$
Multiply by $k|\psi(f)|^2$
$\hat{S}_d(f)_{min} k|\psi(f)|^2$
Subtract this at each of the corresponding redundancies from $S_x(f)$
$S_n(f) + S_d(f)k|\psi(f)|^2 - \hat{S}_d(f)_{min} k|\psi(f)|^2$
This is the estimate of the PSD of the noise $\hat{S}_n(f)$

Figure 3.3 Diagram of Subtraction of Minimum Redundancy after Pulse Shape Multiplication

averages them to obtain:

$$\mathbf{F}_{average}(f) = \frac{1}{4}\left(\mathbf{S}_d(f^1) + \frac{\mathbf{S}_n(f^1)}{k|\Psi(f^1)|^2} + \mathbf{S}_d(f^2) + \frac{\mathbf{S}_n(f^2)}{k|\Psi(f^2)|^2} + \mathbf{S}_d(f^3) + \frac{\mathbf{S}_n^*(f^3)}{k|\Psi^*(f^3)|^2} + \mathbf{S}_d(f^4) + \frac{\mathbf{S}_n^*(f^4)}{k|\Psi^*(f^4)|^2}\right) \tag{3.14}$$

The $\mathbf{S}_d(f)$ are the same so we can simplify this to:

$$\mathbf{F}_{average}(f) = \frac{1}{4}\left(4\mathbf{S}_d(f) + \frac{\mathbf{S}_n(f^1)}{k\,|\Psi(f^1)|^2} + \frac{\mathbf{S}_n(f^2)}{k\,|\Psi(f^2)|^2} + \frac{\mathbf{S}_n^*(f^3)}{k\,|\Psi^*(f^3)|^2} + \frac{\mathbf{S}_n^*(f^4)}{k\,|\Psi^*(f^4)|^2}\right) \tag{3.15}$$

$$\mathbf{F}_{average}(f) = \mathbf{S}_d(f) + \frac{1}{4}\left(\frac{\mathbf{S}_n(f^1)}{k\,|\Psi(f^1)|^2} + \frac{\mathbf{S}_n(f^2)}{k\,|\Psi(f^2)|^2} + \frac{\mathbf{S}_n(f^3)}{k\,|\Psi^*(f^3)|^2} + \frac{\mathbf{S}_n(f^4)}{k\,|\Psi^*(f^4)|^2}\right) \tag{3.16}$$

$$\mathbf{F}_{average}(f) = \mathbf{S}_{d_{average}}(f) + \frac{1}{4}\sum_{k=1}^{4}\frac{\mathbf{S}_n(f^k)}{k\,|\Psi(f^k)|^2} \tag{3.17}$$

From this point the average methods start.

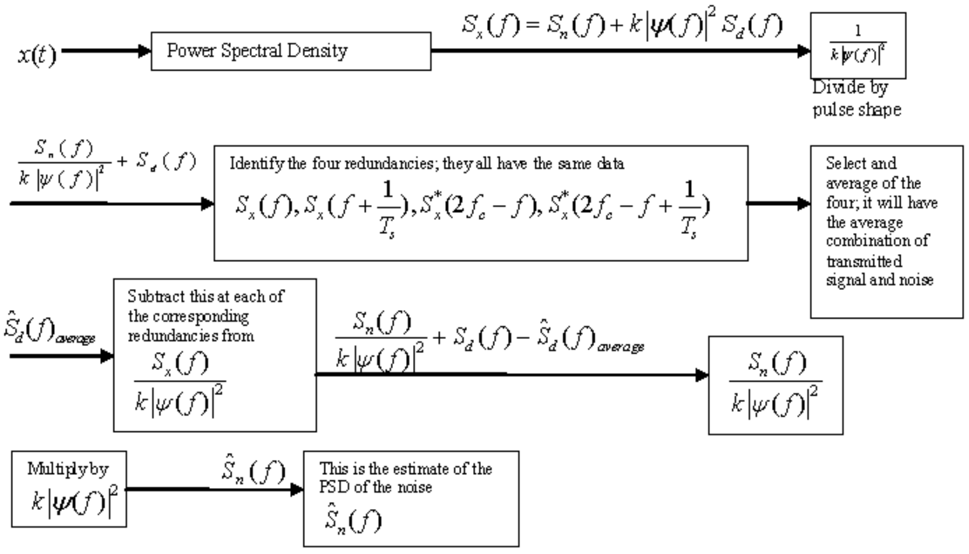

$x(t)$
Power Spectral Density
$S_x(f) = S_n(f) + k|\psi(f)|^2 S_d(f)$
$\frac{1}{k|\psi(f)|^2}$
Divide by pulse shape
$\frac{S_n(f)}{k|\psi(f)|^2} + S_d(f)$
Identify the four redundancies; they all have the same data
$S_x(f), S_x(f+\frac{1}{T_s}), S_x^*(2f_c - f), S_x^*(2f_c - f + \frac{1}{T_s})$
Select and average of the four; it will have the average combination of transmitted signal and noise
$\hat{S}_d(f)_{average}$
Subtract this at each of the corresponding redundancies from $\frac{S_x(f)}{k|\psi(f)|^2}$
$\frac{S_n(f)}{k|\psi(f)|^2} + S_d(f) - \hat{S}_d(f)_{average}$
$\frac{S_n(f)}{k|\psi(f)|^2}$
Multiply by $k|\psi(f)|^2$
$\hat{S}_n(f)$
This is the estimate of the PSD of the noise $\hat{S}_n(f)$

Figure 3.4 Diagram of Subtraction of the Redundancy Average before Pulse Shape Multiplication

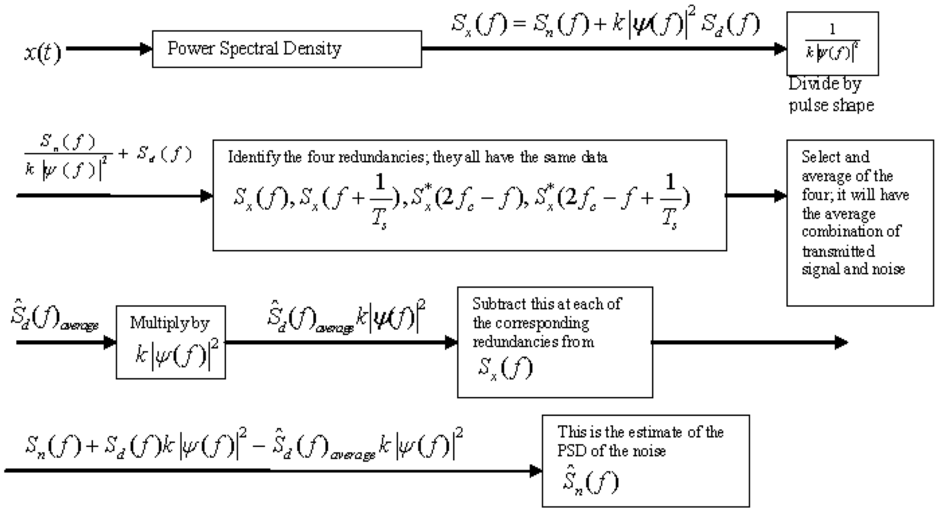
3.2.2.1 Subtraction Before Pulse Shape Multiplication. Subtracting $\mathbf{F}_{average}(f)$ from $\mathbf{F}(f)$ leaves:

$$\begin{gathered}\mathbf{F}(f) - \mathbf{F}_{average}(f) = \\ \mathbf{S}_d(f) + \frac{\mathbf{S}_n(f)}{k|\Psi(f)|^2} - \left(\mathbf{S}_{d_{average}}(f) + \frac{1}{4}\sum_{k=1}^{4} \frac{\mathbf{S}_{n_{average}}(f^k)}{k\left|\Psi(f^k)\right|^2}\right) = \\ \frac{3}{4}\frac{\mathbf{S}_n(f^j)}{k|\Psi(f^j)|^2} - \frac{1}{4}\sum_{k\neq j} \frac{\mathbf{S}_{n_{average}}(f^k)}{k\left|\Psi(f^k)\right|^2}\end{gathered} \tag{3.18}$$

As in previous methods following multiplication by $k\left|\Psi(f^j)\right|^2$ yields:

$$\hat{S}_n(f) = k\left|\Psi(f)\right|^2 \mathbf{F}(f) - \mathbf{F}_{average}(f) = \frac{3}{4}\mathbf{S}_n(f^j) - \frac{1}{4}\sum_{k\neq j} \frac{\mathbf{S}_{n_{average}}(f^k)}{\left|\Psi(f^k)\right|^2}\left|\Psi(f^j)\right|^2 \tag{3.19}$$

This is the estimate of the PSD of the noise, $S_n(f)$, based on the average of the redundancies. This method completely eliminates the constant k, which seems very promising. A flow chart of this method can be seen in Figure 3.4. The degree of correctness of this estimation will be demonstrated and measured in Chapter IV.



$x(t)$
Power Spectral Density
$S_x(f)=S_n(f)+k|\psi(f)|^2 S_d(f)$
$\frac{1}{k|\psi(f)|^2}$
Divide by pulse shape
$\frac{S_n(f)}{k|\psi(f)|^2}+S_d(f)$
Identify the four redundancies; they all have the same data
$S_x(f), S_x(f+\frac{1}{T_s}), S_x^*(2f_c-f), S_x^*(2f_c-f+\frac{1}{T_s})$
Select and average of the four; it will have the average combination of transmitted signal and noise
$\hat{S}_d(f)_{average}$
Multiply by $k|\psi(f)|^2$
$\hat{S}_d(f)_{average}k|\psi(f)|^2$
Subtract this at each of the corresponding redundancies from $S_x(f)$
$S_n(f)+S_d(f)k|\psi(f)|^2-\hat{S}_d(f)_{average}k|\psi(f)|^2$
This is the estimate of the PSD of the noise $\hat{S}_n(f)$

Figure 3.5 Diagram of Subtraction of the Redundancy Average after Pulse Shape Multiplication

3.2.2.2 Subtraction After Pulse Shape Multiplication.

$$\mathbf{F}_{average}(f) = \mathbf{S}_d(f) + \frac{1}{4}\sum_{k=1}^{4}\frac{\mathbf{S}_n(f^k)}{k\left|\Psi(f^k)\right|^2} \tag{3.20}$$

Starting again with the above equation, we first multiply by the pulse shape and constant and then subtract $\mathbf{F}_{average}(f)k\left|\Psi(f^k)\right|^2$ from the original $S_x(f)$ to obtain the estimate for $\mathbf{S}_n(f)$.

$$\mathbf{F}_{average}(f)k\left|\Psi(f^k)\right|^2 = \left(\mathbf{S}_d(f) + \frac{1}{4}\sum_{k=1}^{4}\frac{\mathbf{S}_n(f^k)}{k\left|\Psi(f^k)\right|^2}\right)k\left|\Psi(f^k)\right|^2 \tag{3.21}$$

Finally, we subtract this not from equation 3.10 but from equation 3.7 to obtain:

$$S_x(f) - \mathbf{F}_{average}(f)k\left|\Psi(f^k)\right|^2 = \hat{S}_n(f) \tag{3.22}$$

Again, $S_d(f) - \hat{S}_{d_{average}}(f)$ is assumed to be small, so the remaining value, $\hat{S}_n(f)$, is the estimate for the PSD of the noise. A flow chart for this method is presented in Figure 3.5. Thus, subtracting before or after pulse shape multiplication produces the same result, $\hat{S}_n(f)$.

IV. Analysis by Simulation

As observed at the end of Chapter II, the success of the filter when no characteristics of the noise are known decreases dramatically when the noise scale exceeds the scale of the transmitted signal. Thus it seems logical that one method of testing should be to test the success of the demodulator at various noise levels. The results in this chapter are presented in two sections. The first section presents the effect the noise PSD estimates have on the demodulation success of the MMSE filter, reviewed in Section 2.5. As noise levels increase, the ability to accurately recover the originally transmitted data symbols is measured. The second section presents a comparison between the actual and estimated PSDs of the noise. As explained, the noise estimation problem has two important goals. First, can these methods accurately measure the scale of the noise? Second, can they obtain an accurate estimate of the noise shape? It is important to note that the noise estimates do not give actual noise content as it exists in the time domain, but rather estimates of the PSD of the noise, since this is the form of the noise that the filter presented in Chapter II and other signal processing methods use.

4.1 Impact on Demodulation

The demodulation technique, Gisselquist's Minimum Mean Square Error filter, can serve not only for measuring noise estimation success, but as part of the algorithm to estimate the noise. This section is organized into two subsections, the first presents the results of Method One: Subtract in Time, and the second exploits the signal redundancies in the estimation algorithms. The measure of success in this section is the Bit Error Rate (BER) as the noise scale is incrementally increased or in the first method compared to iterations.

4.1.1 Method One: Subtract in Time.

This method seemed to be the most obvious, and possibly the most flawed. There may be other ways to estimate the noise in the time domain, but only Subtract in Time is evaluated here. This approach was not effective as Figure 4.1 shows. The BER in this experiment did not converge, even after fifty iterations. The initial estimate of the data symbols obtained using an arbitrary value

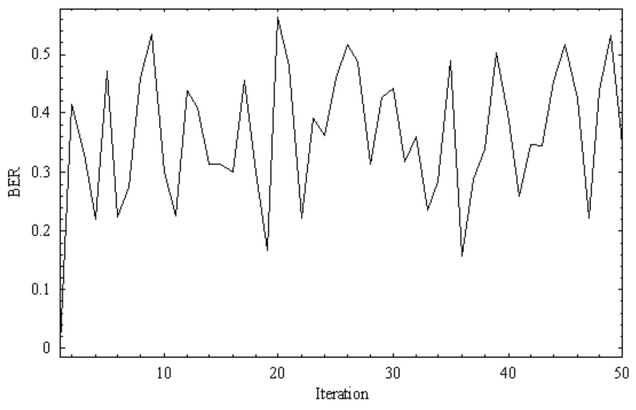


Figure 4.1 Method One: Subtract in Time BER vs. Iteration Number

for the PSD of the noise, $S_n(f)$, was the best estimate. The estimates seemed to fluctuate wildly as the noise level increased. This is an indication that the method is not an adequate solution to the PSD of the noise estimation problem. The formula used to calculate the BER in this section and all following sections is:

$$BER = \frac{\text{Number of Incorrectly Identified Symbols}}{\text{Total Number of Transmitted Symbols}} \tag{4.1}$$

There are probably several reasons why this method is not successful. One reason may be that the demodulation technique that the noise estimation is trying to improve is part of the noise estimation algorithm. The next subsection implements algorithms that rely only on information gleaned from the received signal. The next subsection also takes advantage of more properties of the received signal and the processing occurs in the frequency domain, for these reasons they hold more promise in accurately estimating the noise.

4.1.2 Methods Using the Signal Redundancies. The previous method did not exploit the properties of the received signal to estimate the PSD of the noise and it is likely that because of this, was not very successful. The results presented here use the redundancies of the signal to estimate the noise PSD. An improvement in these techniques is that they do not use the demodulation technique that they are trying to improve as part of the noise estimation algorithms.

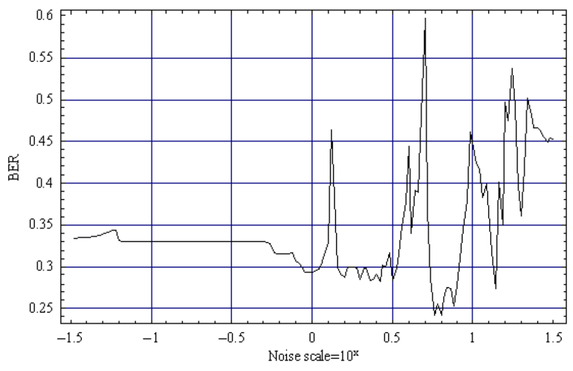


Figure 4.2 BER vs. Noisescale, Subtraction of the Smallest Redundancy before Pulse Shape Multiplication

4.1.2.1 Subtraction of the Smallest Redundancy. As explained in Section 2.1.2, BPSK signals have redundancy properties that can be exploited. Unfortunately the results using the smallest redundancy were not as successful as expected in the improvement of data symbol recovery. Section 3.2.1 explains that subtraction of the smallest redundancy can occur at two points in the estimation algorithm. The results of each are presented here.

Subtraction of the Smallest Redundancy Before Pulse Shape Multiplication. This method improved the data symbol estimation algorithm, as can be seen if we compare the results in Chapter II, Figures 2.8 and 2.9 with Figures 4.2 and 4.3. These results are slightly better than using an arbitrary value for the PSD of the noise in the demodulation algorithm. As the noise scale increased, the BER increased to a level no better than random. Here the results are plotted in two ways, against the noise scale and against the SNR. Future results will only have the BER versus the SNR.

Subtraction of the Smallest Redundancy After Pulse Shape Multiplication. This variation of the previous method produced identical results which can be seen in Figure 4.4. Hopefully it will be more successful in predicting the noise scale, which is explored in the following section.

Although the methods that subtract the smallest redundancy didn't have the positive impact on the demodulation technique as hoped, it should not be taken as an indication that the methods are invalid. Since the demodulation technique, the MMSE filter, uses

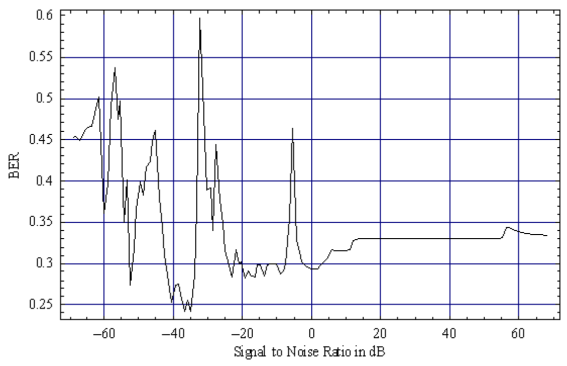


Figure 4.3 BER vs. SNR in dB, Subtraction of the Smallest Redundancy before Pulse Shape Multiplication

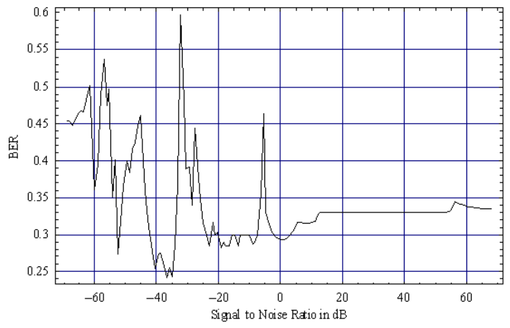


Figure 4.4 BER vs. SNR, Subtraction of the Smallest Redundancy after Pulse shape Multiplication

only a small portion of the observed noise spectrum, it is possible that the methods are not applicable in those particular portions of the observed noise spectrum.

4.1.2.2 Subtraction of the Average Redundancy. Subtracting the minimum redundancy in the previous two methods, did not improve the demodulator performance as much as hoped. Instead of subtracting the minimum, the next two methods subtract an average of the four redundancies, since they all contain the same information. Hopefully these methods will provide a more positive impact on the demodulation success of the MMSE filter. As when subtracting the minimum, the subtraction of the average can occur at two different places in the algorithm. Both variations are presented here.

Subtraction of the Average Redundancy Before Pulse Shape Multiplication. The results when subtracting the average of the redundancies before multiplying

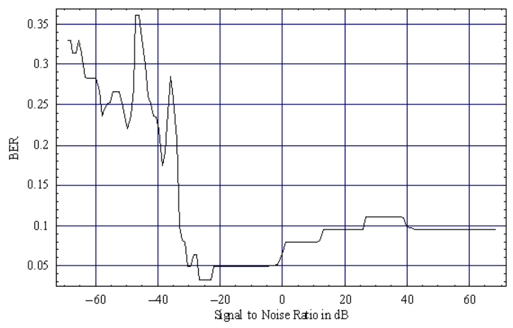


Figure 4.5 BER vs. SNR, Subtraction of the Average Redundancy before Pulse Shape Multiplication

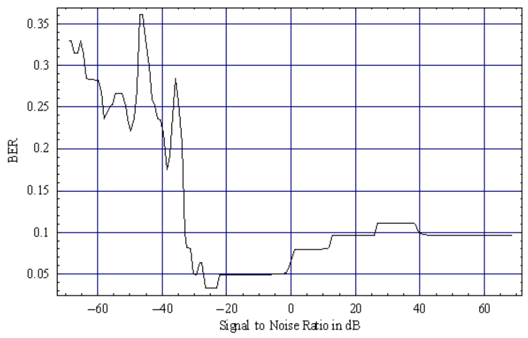


Figure 4.6 BER vs. SNR, Subtraction of the Average Redundancy after Pulse Shape Multiplication

by the pulse shape is shown in Figure 4.5. This produced an improvement over the method that subtracted the minimum before multiplying by the pulse shape. Additionally these results were better than using an unknown noise scale and unknown noise shape as can be seen in Figure 2.9 in Chapter II.

This method improves the success of the demodulator, however the BER continues to be high and fluctuate wildly when the SNR is high. The acceptable BER is system dependent and so the measure of how well the methods work is dependent on the system.

Subtraction of the Average Redundancy After Pulse Shape Multiplication. As seen in Figure 4.6 this method like the subtraction of the smallest redundancy methods provided identical results as subtracting before multiplication by the pulse shape.

The methods that subtract the average of the redundancies improved the success of the demodulator which can be seen in Figure 4.7. The results using the methods that

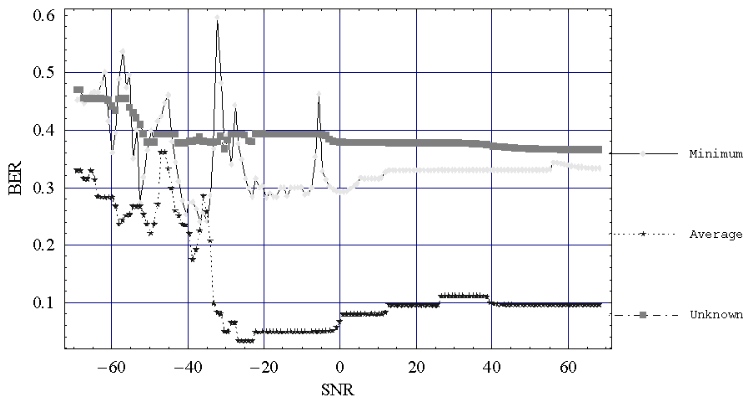


Figure 4.7 BER vs. SNR

subtract the average redundancies improved the success of the demodulator even when the SNR exceeded -30 dB. This is promising since SNRs rarely exceed -40 dB. In Figure 4.7 it can be seen that the subtraction of the average redundancies is the most successful as compared to subtracting the minimum redundancies and using an arbitrary noise SNR value in the demodulator, which are the results labeled "Unknown".

The next section measures the success of the methods over the entire observed noise spectrum, rather than just in the one small portion that the filter uses for demodulation, using the Mean Square Error (MSE) between the actual noise PSD and the estimated noise PSD.

4.2 Comparison to Actual Power Spectral Density of the Noise

Improvement of Gisselquist's MMSE filter was the initial motivation for this research, and it yielded positive results when subtracting the average of the redundancies. This section examines the whole observed noise spectrum in an attempt to measure the success of the previously explained methods. The two important characteristics of the PSD of the noise, scale and shape, are examined and then a measure of the mean square error between the actual PSD of the noise and the estimated PSD of the noise is presented. Each technique captured a part of the characteristics we desire to estimate. Although they were not completely successful individually, perhaps a combination of these methods may prove to be an improvement.

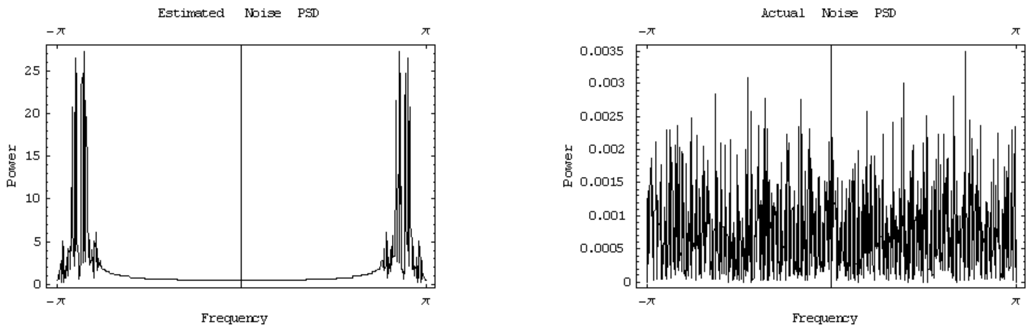


Figure 4.8 Results of Method One: Subtract in Time

4.2.1 Method One: Subtract in Time. Since this method uses the filter in the estimation process, the results are as expected, less than stellar. Not only did the method fail to capture the correct noise scale, but the method did not obtain the correct shape of the PSD of the noise. Figure 4.8 shows the estimated power spectral density of the noise plotted next to the actual power spectral density of the noise. The figure shows that neither the scale nor shape of the noise is accurately estimated.

These results, like the results in the filter improvement, probably stem from the fact that the results we are looking for are in the frequency domain, while the processing technique occurs in the time domain. Additionally, since the filter is used in the estimation algorithm, the estimate accuracy at best is only valid in the portions of the observed noise spectrum that the demodulation algorithm uses. The following results do not rely on the filter, and thus may have more success in estimating the PSD of the noise, both in shape and scale.

4.2.2 Methods Using Signal Redundancies. Although these methods were not all successful in the improvement of the demodulation algorithm, they did show some promise at least in some small portions of the observed noise spectrum. Perhaps this success can be used to implement algorithms with a smaller observation period or algorithms that combine one or more of the methods. Algorithms that use a smaller observation period would use a portion of the burst transmission, instead of using the entire burst as these methods do. These algorithms would account for rapid changes in the noise since the PSD of the noise would be normalized over a smaller amount of time. These methods use the

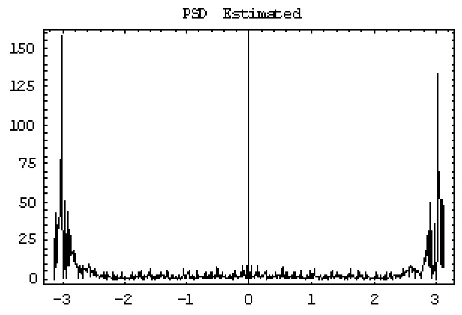


Figure 4.9 Estimated Noise PSD Using Subtraction of the Smallest Redundancy before Mulitplication by Pulse Shape

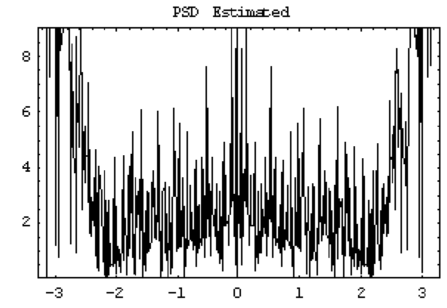


Figure 4.10 Zoomed in Estimated Noise PSD Using Subtraction of the Smallest Redundancy before Mulitplication by Pulse Shape

entire burst to estimate the PSD of the noise, which tends to minimize the contribution of the noise over the period. A smaller observation period would not do this.

4.2.2.1 Subtraction of the Smallest Redundancy. The hope for these methods was that the scale would be estimated accurately, which is in fact reflected in the results. A suggestion for improvement may be to use a smaller observation window as previously suggested. The results can be seen in Figures 4.9, 4.10, and 4.11. Figure 4.9 shows the entire estimated noise spectrum, while Figure 4.10 shows a zoomed in picture. When comparing Figure 4.10 with Figure 4.11 it can be seen that the methods accurately estimated the shape and scale of the noise PSD in the center of the plot, but when compared with Figure 4.11 examination of the outer edges shows that the method did not accurately capture the scale of the actual noise PSD.

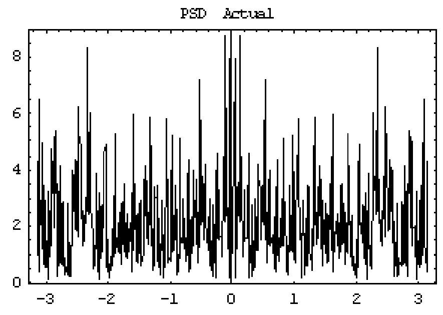


Figure 4.11 Actual Noise PSD

Subtraction of the Smallest Redundancy Before Multiplication by Pulse Shape. This method did not estimate the shape of the actual PSD of the noise with accuracy, but did capture the correct scale at least in a portion of the estimated noise spectrum. Since it was not very successful in filter improvement, it can be assumed that the estimated shape was not correct in the portions of the spectrum that the filter uses.

Figure 4.12 presents the mean square error between the actual power spectral density of the noise and the estimated power spectral density of the noise. This mean square error was calculated in the following manner:

$$MSE \triangleq \frac{1}{N}\sum_{i=1}^{N}(p_i - \hat{p}_i)^2 \tag{4.2}$$

where N is the number of points at which the noise was estimated, which equates to the number of signal samples in both the time and frequency domains. The p_i are the actual values at each point in the noise spectrum and the $\hat{p}_i$ are the estimated noise PSD values. As expected, as the noise scale increases, the mean square error increases greatly. However the MSE was very low when the noise scales were low.

Subtraction of the Smallest Redundancy After Multiplication by Pulse Shape. For estimating the scale of the noise spectrum, this method is more promising than the previous method. These results can be seen in Figures 4.13, 4.14, and 4.15. Although it produced BER results identical to the method that subtracts the smallest redundancy before multiplication, it clearly does a better job of estimating the scale across the entire observed noise spectrum. Again since this method did not improve the demodu-

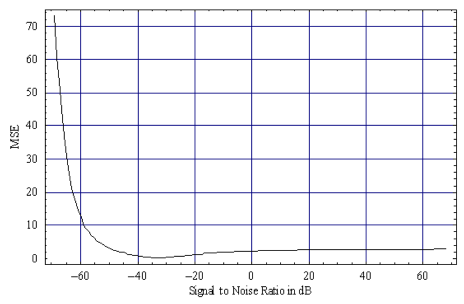


Figure 4.12 MSE vs. SNR, Using Subtraction of the Smallest Redundancy before Mulitplication by Pulse Shape

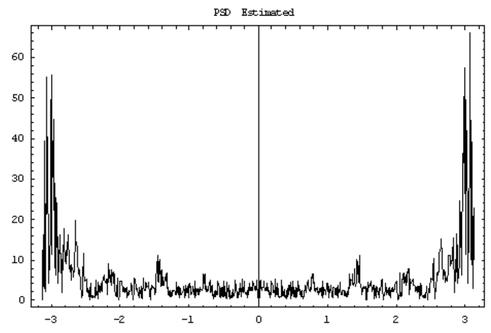


Figure 4.13 Estimated Noise PSD Using Subtraction of the Smallest Redundancy after Mulitplication by Pulse Shape

lation as much as the average methods, it can be assumed that the estimation of the shape was incorrect in the portions that the filter employs. As in the previous method, Figure 4.13 shows the entire spectrum of the estimated noise PSD, Figure 4.14 shows a zoomed in view of the estimated noise PSD and Figure 4.15 shows the actual noise PSD.

Figure 4.16 shows the same results as the previous method, reflecting the increase in the mean square error as the noise scale increased to a level beyond the transmitted signal.

Subtraction of the smallest redundancy does have a positive result. It accurately estimates the shape of the noise in the center of the spectrum, and closely estimates the noise scale on the outer portions of the spectrum. The methods that subtract the smallest redundancies also achieve the smallest MSE as compared to the actual noise PSD.

4.2.2.2 Subtraction of the Average Redundancy. The expectation for this method was that more of the actual noise shape would be captured since the subtraction portion is a combination of the redundancies. They were not very successful in estimating

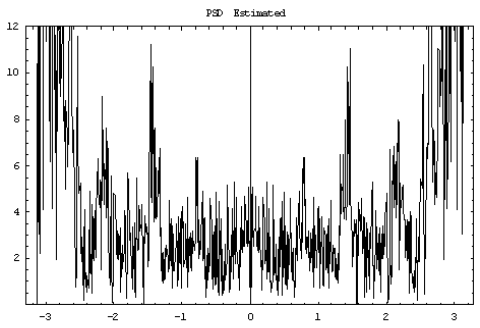


Figure 4.14 Zoomed in Estimated Noise PSD Using Subtraction of the Smallest Redundancy after Mulitplication by Pulse Shape

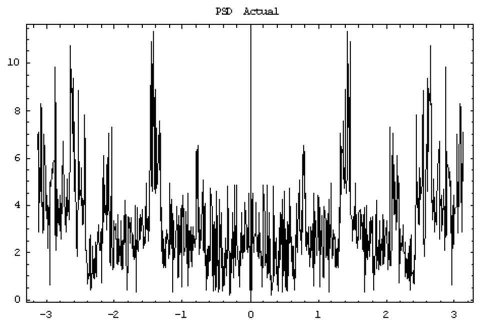


Figure 4.15 Actual Noise PSD

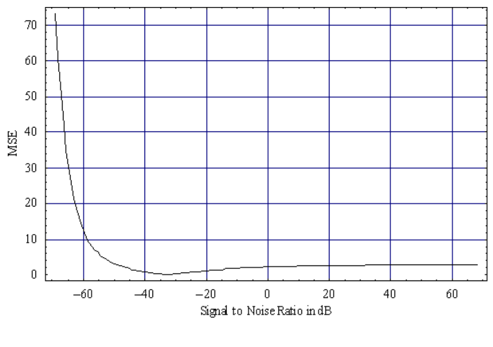


Figure 4.16 MSE vs. SNR Using Subtraction of the Smallest Redundancy after Pulse Shape Multiplication

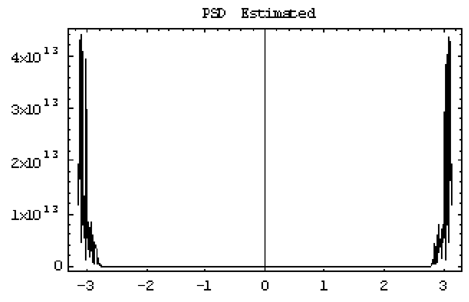


Figure 4.17 Estimated Noise PSD Using Subtraction of the Average Redundancy before Pulse Shape Multiplication

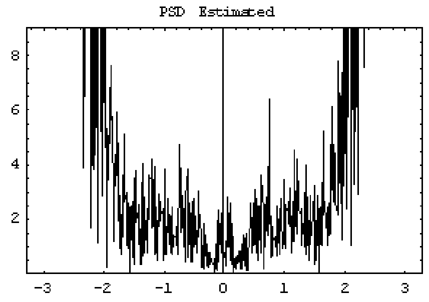


Figure 4.18 Zoomed in Estimated Noise PSD Using Subtraction of the Average Redundancy before Pulse Shape Multiplication

the noise scale over the whole noise spectrum, but the shape resembles at least a portion of the actual noise PSD.

Subtraction of the Average Redundancy Before Multiplication by Pulse Shape. This method almost captured the scale and the shape in the center of the observed noise spectrum. This portion can be seen best in Figure 4.18. Figure 4.17 shows that the scale estimate on the outer portion, which the filter uses, is completely wrong. However incorrect the scale is in these portions, the shape must be accurately captured since the demodulation results were so successful. Figure 4.20 shows the mean square error between the estimated and actual PSDs of the noise. It is calculated as in equation 4.2.

As expected because of the horrible scale error in the higher frequencies, the mean square error increased dramatically as the noise scale increased.

Subtraction of the Average Redundancy After Multiplication by Pulse Shape. Like the previous method, this method accurately captured the scale and

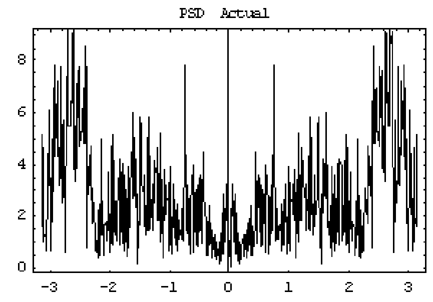


Figure 4.19 Actual PSD Using Subtraction of the Average Redundancy before Pulse Shape Multiplication

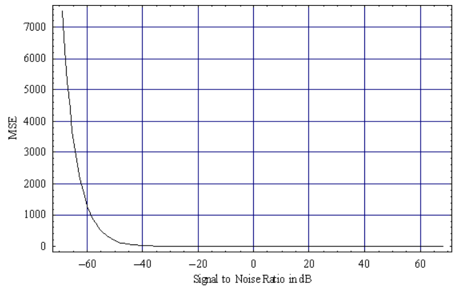


Figure 4.20 MSE vs. SNR Using Subtraction of the Average Redundancy before Pulse Shape Multiplication

the shape in the center of the observed noise spectrum. Figures 4.21, 4.22, and 4.23 show the results of this final method. Figure 4.21 shows that like the previous method, the scale in the higher frequencies is completely of from the actual noise PSD. Figure 4.22 shows that this method nearly captures the shape and scale in the center of the noise spectrum. Like the previous method, subtraction of the average redundancy after multiplication by the pulse shape is not very successful over the entire spectrum, this is reflected in Figure 4.24.

The methods that subtract the average of the redundancies do not accurately capture the scale of the actual noise PSD, but they did have the best success in improvement of the demodulator success. Figure 4.25 shows a comparison of the MSE between all the subtraction methods and the MSE using a noise PSD with unknown shape and scale. As previously shown, the methods that subtract the average redundancies do not do a good job of estimating the noise PSD over the whole noise spectrum. The methods that subtract the minimum redundancies obtain a MSE about equal to using a noise PSD with unknown

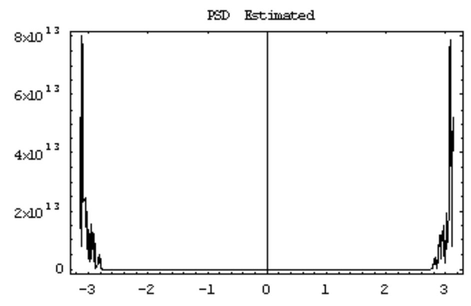


Figure 4.21 Estimated Noise PSD Using Subtraction of the Average Redundancy after Pulse Shape Multiplication

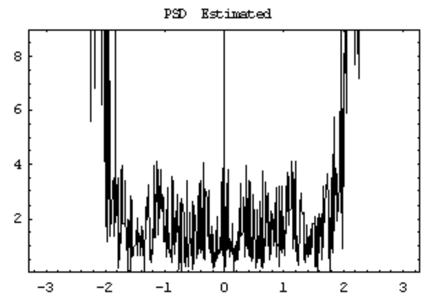


Figure 4.22 Zoomed in Estimated Noise PSD Using Subtraction of the Average Redundancy after Pulse Shape Multiplication

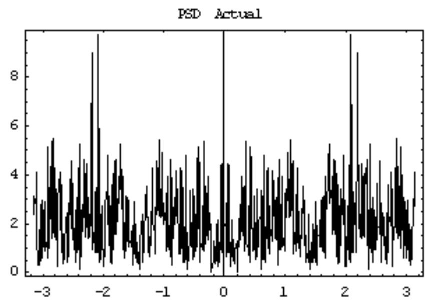


Figure 4.23 Actual Noise PSD

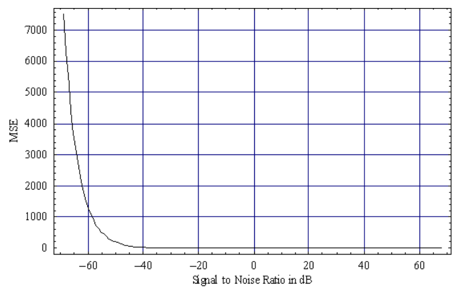


Figure 4.24 MSE vs. SNR Using Subtraction of the Average Redundancy after Pulse Shape Multiplication

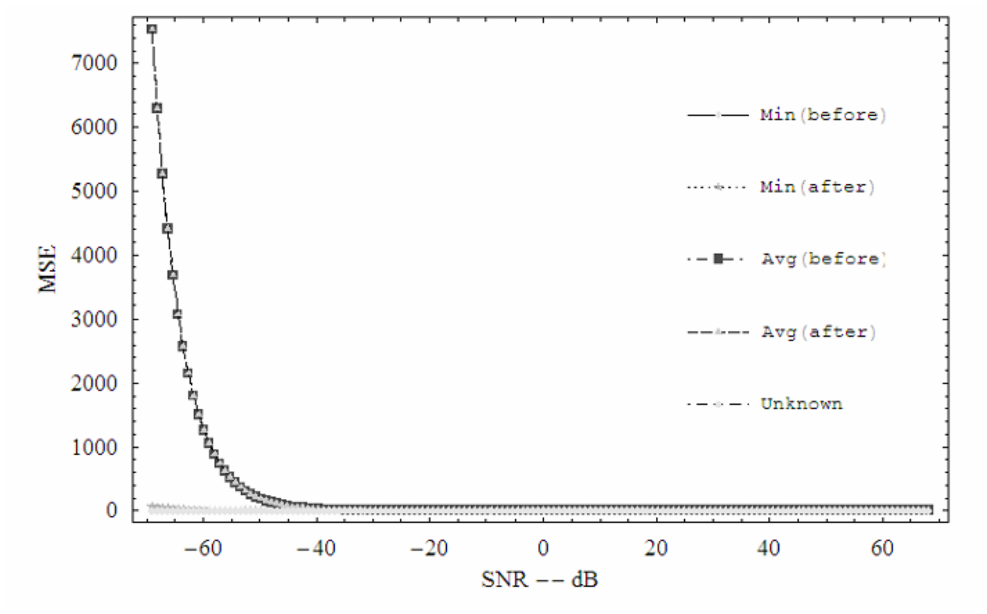


Figure 4.25 MSE vs. Noise Scale Subtraction Methods Compared

shape and scale. Thus the best methods for estimating the noise PSD over the entire spectrum are the methods that subtract the smallest redundancies.

V. *Conclusions/Recommendations*

The initial goal of this research was to determine if an accurate estimate of the noise power spectral density (PSD) could be obtained. The need for a solution to this problem stemmed from research done by Gisselquist in (11). He created a minimum mean square error (MMSE) filter whose success depended on an accurate a priori estimate of the noise PSD. His filter is designed and works best in a colored noise environment (11). This thesis tackled the white noise estimation problem, in the hopes of providing direction for the colored noise problem. Five methods were developed and tested to determine if they improved his filter results. The methods were also measured against the entire noise spectrum as well, since the filter that Gisselquist created only employs a small part of the noise PSD.

The estimation techniques can be improved and suggestions for further research are included in this chapter following a summary of the results obtained. Overall the research was a success as it provided a good starting point for further research.

5.1 Summary of Results

The methods tested included Subtraction in Time, and four other methods which exploited the redundancies of the signal properties. The second group of methods included the Subtraction of the Smallest Redundancy Before and After Pulse Shape Multiplication. Finally the third group included Subtraction of the Average Redundancy Before and After Pulse Shape Multiplication. The results of the Subtract in Time method were totally unsuccessful. In fact, the results obtained using an arbitrary noise PSD in the filter with a random shape and scale provided better results than Subtract in Time. The methods that used the smallest redundancy provided a slight improvement in the demodulator performance, but were most successful in obtaining the most accurate estimate across the entire noise PSD spectrum. The methods that subtracted the average redundancies provided the best results. Not the best when compared to the actual entire noise PSD spectrum, but in improvement of the MMSE filter. This is a success since the most important goal of this thesis was to improve the success of Gisselquist's MMSE filter.

5.2 Future Work

As stated previously, Gisselquist's filter is optimal in a colored noise environment. Because of this it is obvious that a suggestion for further work is to extend these methods to the colored noise environment. Another area for continued research is the exploration of why the subtraction of the average redundancy methods do not capture the scale of the noise PSD in the higher frequencies. Perhaps if a more accurate estimate of the scale can be obtained in this region, the success of the method can be improved further. A third area to explore is combining these methods which only use an active signal with methods that use observations when the signal is not active. When the signal is not active, only noise is present, thus a good estimate should be possible when combined with the estimates of the noise when the signal is present.

Overall, the methods presented in this thesis provide a good ground work for several future estimation methods in both the colored and white noise environments. One last suggestion for future work is to employ a smaller observation window combined with adaptive filtering. This would allow the noise PSD used in the filter to perhaps capture more of the noise characteristics in a smaller region, improving the success incrementally as the observation time moves along.

5.3 Conclusion

Gisselquist created an excellent and new signal model for burst transmission signals. Additionally he created an excellent demodulation technique based on this model. The results presented in this thesis can hopefully be implemented so that the MMSE filter can be improved. The results show that the methods employed are successful in the white noise environment. Hopefully future research can extend the methods and combine them with other methods to further the improvement of Gisselquist's MMSE filter.

VI. Appendix A Mathematica Notebooks

All the figures in the thesis were generated using Mathematica. The notebooks are all presented here. The figures that each notebook produced are listed in the explanation which precedes each notebook.

6.1 Data Generation

This notebook creates the variables needed for all subsequent experiments. It was not used to generate any figures. The first section creates the variables used in the signal generation process. The comments after each variable describe what the variable corresponds to in the thesis.

```
<< Statistics`ContinuousDistributions`
ts = 1 (*symbol length based on original symbol Ts *);
```

$$\texttt{ns} = 2^{6}\texttt{(* message length Ns in the orginal*);}$$

```
d = Table[2 Random[Integer] - 1, {i, 1, ns}]; (* the message *)
tau = 0; (* delay *)
fc = 1(* carrier freq fc in Hertz*);
```

$$\texttt{k} = 2^{4}\texttt{; (*used to establish sample rate*)}$$

$$\texttt{tt} = \frac{1}{\texttt{k fc}}\texttt{; (*sample rate*)}$$

$$\texttt{bign} = \frac{\texttt{ns ts}}{\texttt{tt}}\texttt{; (*this is the number of samples we will take in both the frequency and time domains*}$$

```
theta = 0; (*phase*)
aa = 1; (*amplitude*);
mu = 0; (*used in random noise generation*)
sigma = 1; (*used in random noise generation*)
```

$$\texttt{noisescale} = 10^{-1}\texttt{;}$$

$$\texttt{sinc[t\_] := If}\left[\texttt{t} \neq 0,\ \frac{\texttt{Sin[}\pi\texttt{ t]}}{\pi\texttt{ t}},\ 1\right]\texttt{;}$$

$$\texttt{nyq[t\_] :=} \frac{1}{\sqrt{\texttt{ts}}}\left(\texttt{sinc}\left[2\frac{\texttt{t}}{\sqrt{\texttt{ts}}}-\frac{1}{2}\right]+\texttt{sinc}\left[2\frac{\texttt{t}}{\sqrt{\texttt{ts}}}+\frac{1}{2}\right]\right)\texttt{; (*Gisselquist's modified sinc*)}$$

$$\texttt{s[t\_] := N}\left[\texttt{Re}\left[\texttt{aa}\sum_{n=0}^{ns-1}\texttt{(d[[n + 1]] nyq[t - n ts - tau] Exp[((I 2 }\pi\texttt{ fc t) + theta)]})\right]\right]\texttt{;(*signal equation*)}$$

This portion creates a random vector of white noise and calculates the PSD.

```
white = noisescale * Table[Random[NormalDistribution[mu, sigma]], {i, 1, bign}];
whitefourier = Fourier[white, FourierParameters → {1, -1}];
noisefourier = whitefourier; (*choose coloredfourier or whitefourier*)
noise = white; (*choose colored or white*)
```

$$\texttt{rxnoise = Table}\left[\frac{1}{\texttt{Length[noise] - k}}\sum_{n=0}^{\texttt{Length[noise]-1-k}}\texttt{noise[[n + 1]] * noise[[n + 1 + k]], \{k, 0, Length[noise] - 1\}}\right]\texttt{;}$$

```
PSDnoise = Fourier[rxnoise, FourierParameters → {1, -1}];
```

This portion generates the signal vector, adds the white noise and takes the discrete Fourier Transform of the generated signal and noise combined.

```
Timing[tx = Table[(ns ts)/bign i, {i, 0, bign - 1}];
 st = Table[s[tx[[i]]], {i, 1, Length[tx]}];
 signalnoise = noisescale * Table[Random[NormalDistribution[mu, sigma]], {i, 1, Length[tx]}];
 signal = st + signalnoise;
 signalfourier = Fourier[signal, FourierParameters -> {1, -1}];]
```

This portion creates the Minimum Mean Square Error Filter.

```
somega = Table[(2 π)/bign k, {k, 0, bign - 1}];
high = Length[somega];

(*PSD of the noise and FT of the signal for the MMSE filter*)
psd[w_] := If[0 < Abs[w] < 1/tt, left = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
  PSDnoise[[left]] + (PSDnoise[[left + 1]] - PSDnoise[[left]])/(somega[[left + 1]] - somega[[left]]) *(2 π tt w - somega[[left]]), 0];

XofF[d_] := If[0 < Abs[d] < 1/tt, left = Min[bign - 1, 1 + Floor[bign Abs[d] tt]];
  signalfourier[[left]] + (signalfourier[[left + 1]] - signalfourier[[left]])/(somega[[left + 1]] - somega[[left]]) *(2 π tt d - somega[[left]]), 0];

bigpsi[k_] := If[-1/ts < k < 1/ts, Sqrt[ts] Cos[π (k ts)/2], 0]; (*Fourier Transform of Gisselquist's sinc pulse*)

hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p])) /
  (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) + aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts]) + aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) +
    aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
```

This portion accomplishes the recovery of the originally transmitted data symbols.

```
nf = ns; (*number of samples for recovery, same as the number of transmitted symbols*)
endomega = Sort[N[Table[fc - (i - 1 10^-3)/(nf ts), {i, 1, nf}]]]; (*array variable*)

bigdhat = Table[(Exp[-I theta] Exp[I 2 π (endomega[[i]] - fc) tau] hmmse[endomega[[i]]] XofF[endomega[[i]]]) +
   (Exp[-I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts] hmmse[endomega[[i]] + 1/ts] XofF[endomega[[i]] + 1/ts]) +
   (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Conjugate[hmmse[2 fc - endomega[[i]]]]
     Conjugate[XofF[2 fc - endomega[[i]]]]) +
   (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts] Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
     Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
(*bigdhat is the Z transform of the data symbols and must be undone*)
```

```
z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}]; (*matrix for z transform*)
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]]; (*Z transform matrix*)
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
Chop[LinearSolve[zzz, bigdhat]];
dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]]; (*recovered data symbols*)
MatrixForm[Transpose[{dhat, d}]];
(*BER*)
```

$$N\left[\frac{\frac{\mathrm{Total}[\mathrm{Abs}[\mathrm{dhat}-\mathrm{d}]]}{2}}{\mathrm{ns}}\right]$$

6.2 Known Noise Scale

This first portion of the notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." This notebook was used to create Figures 2.10 and 2.11.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
ns = 2^6;
nf = ns;
fc = 1;
k = 2^4;
tt = 1/(k fc);
bign = (ns ts)/tt;
ts = 1;
tau = 0;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
somega = Table[(2 π)/bign k, {k, 0, bign - 1}];
endomega = Sort[N[Table[fc - (i - 1 10^-3)/(nf ts), {i, 1, nf}]]];
z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}];
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]];
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
bigpsi[k_] := If[-1/ts < k < 1/ts, √ts Cos[π (k ts)/2], 0];
```

This section tests the BER using a known noise scale. Notice that the PSD used in the filter is a constant, equal to the noise scale.

```
results = Table[0, {i, 1, 151}];
rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters → {1, -1}];
  psd[w_] := noisescale;

  hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p]))/
    (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) + aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts]) +
      aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) + aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
  XofF[d_] := If[0 < Abs[d] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[d] tt]];
    signalfourier[[lft]] + (signalfourier[[lft + 1]] - signalfourier[[lft]])/(somega[[lft + 1]] - somega[[lft]])
      * (2 π tt d - somega[[lft]]), 0];

  bigdhat = Table[(Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       hmmse[endomega[[i]]] XofF[endomega[[i]]]) +
     (Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts] hmmse[endomega[[i]] +
         1/ts] XofF[endomega[[i]] + 1/ts]) + (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       Conjugate[hmmse[2 fc - endomega[[i]]]] Conjugate[XofF[2 fc - endomega[[i]]]]) +
     (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
       Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
       Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
  dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]];
  percentcorrect = N[(Total[Abs[dhat-d]]/2)/ns];
  results[[i]] = percentcorrect;]]
```

This section plots the results.

```
results = Take[results, -{Length[results] - 1}];
rng = Take[rng, -{Length[rng] - 1}];

noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa / noisescale[[i]]], {i, 1, Length[noisescale]}];
ch2scalesn = ListPlot[Transpose[{rng, results}], PlotJoined → True,
   PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
   FrameLabel → {"Noise scale=10^x", "Bit Error Rate"}, Frame → True,
   Axes → False, GridLines → Automatic];
ch2scalesndb = ListPlot[Transpose[{SNR, results}], PlotJoined → True,
   PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
   FrameLabel → {"Signal to Noise Ratio in dB", "Bit Error Rate"},
   Frame → True, Axes → False, GridLines → Automatic];
```

6.3 Known Noise Shape

This first portion of the notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." This notebook was used to create Figures 2.12 and 2.13.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
ns = 2^6;
nf = ns;
fc = 1;
k = 2^4;
tt = 1/(k fc);
bign = (ns ts)/tt;
ts = 1;
tau = 0;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
somega = Table[(2 π)/bign k, {k, 0, bign - 1}];
endomega = Sort[N[Table[fc - (i - 1 10^-3)/(nf ts), {i, 1, nf}]]];
z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}];
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]];
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
bigpsi[k_] := If[-1/ts < k < 1/ts, √ts Cos[π (k ts)/2], 0];
filternoise = white;
rxfilternoise = Table[1/(Length[filternoise] - k)
    Sum[filternoise[[n + 1]] * filternoise[[n + 1 + k]], {n, 0, Length[filternoise] - 1 - k}], {k, 0, bign - 1}];
PSDnoise = Fourier[rxfilternoise, FourierParameters → {1, -1}];

psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
   PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) * (2 π tt w - somega[[lft]]), 0];
```

This section tests the recovery rate using a known noise shape. Notice that the PSD used in the filter is identical to the noise used in the signal. The scale is the only thing that varies.

```
results = Table[0, {i, 1, 151}];
rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters → {1, -1}];

  hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p])) /
   (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) + aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts]) +
     aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) + aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
  XofF[d_] := If[0 < Abs[d] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[d] tt]];
    signalfourier[[lft]] + (signalfourier[[lft + 1]] - signalfourier[[lft]])/(somega[[lft + 1]] - somega[[lft]])
     * (2 π tt d - somega[[lft]]), 0];

  bigdhat = Table[(Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
        hmmse[endomega[[i]]] XofF[endomega[[i]]]) +
     (Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
        hmmse[endomega[[i]] + 1/ts] XofF[endomega[[i]] + 1/ts]) +
     (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
        Conjugate[hmmse[2 fc - endomega[[i]]]] Conjugate[XofF[2 fc - endomega[[i]]]]) +
     (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
        Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
        Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
  dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]];
  percentcorrect = N[(Total[Abs[dhat-d]]/2)/ns];
  results[[i]] = percentcorrect;]]
Null
```

This section plots the results.

```
results = Take[results, -(Length[results] - 1)];
rng = Take[rng, -(Length[rng] - 1)];

noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa / noisescale[[i]]], {i, 1, Length[noisescale]}];


ch2shapesn = ListPlot[Transpose[{rng, results}], PlotJoined → True,
   PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
   FrameLabel → {"Noise scale=10^x", "Bit Error Rate"}, Frame → True,
   Axes → False, GridLines → Automatic];
ch2shapesndb = ListPlot[Transpose[{SNR, results}], PlotJoined → True,
   PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
   FrameLabel → {"Signal to Noise Ratio in dB", "Bit Error Rate"},
   Frame → True, Axes → False, GridLines → Automatic];
```

6.4 Unknown Shape and Scale

This first portion of the notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." This notebook was used to create Figures 2.8 and 2.9.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
ns = 2^6;
nf = ns;
fc = 1;
k = 2^4;
        1
tt =  ----- ;
      k fc
       ns ts
bign = ----- ;
        tt
ts = 1 ;
tau = 0;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
                    2 π
somega = Table[ ------ k, {k, 0, bign - 1}] ;
                  bign
                                  i - 1 10^-3
endomega = Sort[N[Table[fc - ------------- , {i, 1, nf}]]] ;
                                    nf ts
z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}];
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]];
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
                     -1          1                     k ts
bigpsi[k_] := If[ ---- < k < ---- , √ts Cos[π ------ ], 0] ;
                     ts          ts                      2
realnoise = white;


                              1
acnoise = Table[ -------------------------
                 Length[realnoise] - k

     Length[realnoise]-1-k
           ∑                realnoise[[n + 1]] *
          n=0
      realnoise[[n + 1 + k]], {k, 0, Length[realnoise] - 1}] ;
actualPSD = Fourier[acnoise, FourierParameters → {1, -1}];

filternoise = Table[Random[NormalDistribution[mu, sigma]], {i, 1, bign}];
                                     1
rxfilternoise = Table[ ---------------------------
                       Length[filternoise] - k

     Length[filternoise]-1-k
           ∑                filternoise[[n + 1]] *
          n=0
       filternoise[[n + 1 + k]], {k, 0, bign - 1}] ;
PSDnoise = Fourier[rxfilternoise, FourierParameters → {1, -1}];


Null
```

```
psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
  PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) *
    (2 π tt w - somega[[lft]]), 0];

hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p]))/
  (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) + aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts])
    + aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) + aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
```

This section tests the recovery rate using an unknown noise shape and scale.

```
results = Table[0, {i, 1, 151}];
MSE = Table[0, {i, 1, 151}];
rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters → {1, -1}];
  XofF[d_] := If[0 < Abs[d] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[d] tt]];
    signalfourier[[lft]] + (signalfourier[[lft + 1]] - signalfourier[[lft]])/(somega[[lft + 1]] - somega[[lft]])
      * (2 π tt d - somega[[lft]]), 0];

  bigdhat = Table[(Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       hmmse[endomega[[i]]] XofF[endomega[[i]]]) +
     (Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       Exp[I 2 π tau/ts] hmmse[endomega[[i]] + 1/ts] XofF[endomega[[i]] + 1/ts]) +
     (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       Conjugate[hmmse[2 fc - endomega[[i]]]]
       Conjugate[XofF[2 fc - endomega[[i]]]]) +
     (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       Exp[I 2 π tau/ts] Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
       Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
  temp = 1/bign Total[(actualPSD - PSDnoise)^2];
  MSE[[i]] = temp;
  dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]];
  percentcorrect = N[(Total[Abs[dhat-d]]/2)/ns];
  results[[i]] = percentcorrect;]]
Null
```

This section plots the results.

```
results = Take[results, -(Length[results] - 1)];
rng = Take[rng, -(Length[rng] - 1)];
results >> BERUNK;
MSE = Take[MSE, -(Length[MSE] - 1)];
MSE >> MSEUNK;
noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa / noisescale[[i]]], {i, 1, Length[noisescale]}];
ch2unksn = ListPlot[Transpose[{rng, results}], PlotJoined → True,
   PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
   FrameLabel → {"Noise scale=10^x", "Bit Error Rate"}, Frame → True,
   Axes →  False, GridLines → Automatic];
ch2unksndb = ListPlot[Transpose[{SNR, results}], PlotJoined → True,
   PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
   FrameLabel → {"Signal to Noise Ratio in dB", "Bit Error Rate"},
   Frame → True, Axes →  False, GridLines → Automatic];
```

6.5 Subtraction of the Minimum Redundancy

6.5.1 Subtraction of the Minimum Redundancy Before Multiplication by Pulse Shape.

This portion of the Mathematica notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." The vectors' meaning is the same as in the first notebook. This notebook was used to create Figures 4.12 and 4.3.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
shape = << shape;
shapefourier = << shapefourier;
ns = 2^6;
nf = ns;
fc = 1;
k = 2^4;
tt = 1/(k fc);
bign = (ns ts)/tt;
ts = 1;
tau = 0;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
somega = Table[(2 π)/bign k, {k, 0, bign - 1}];
endomega = Sort[N[Table[fc - (i - 1 10^-3)/(nf ts), {i, 1, nf}]]];


z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}];
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]];
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
bigpsi[k_] := If[-1/ts < k < 1/ts, Sqrt[ts] Cos[π (k ts)/2], 0];
XofF[d_] := If[0 < Abs[d] < 1/tt, lft = Min[bign - 1, 1 +
      Floor[bign Abs[d] tt]]; signalfourier[[lft]] +
    (signalfourier[[lft + 1]] - signalfourier[[lft]])/(somega[[lft + 1]] - somega[[lft]]) *
     (2 π tt d - somega[[lft]]), 0];
hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p])) /
  (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) + aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts])
    + aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) + aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
```

This section is the loop that tests each of the various noise scales, recovers data symbols at each noise scale, and compares the actual PSD to the estimated PSD using MSE.

```
results = Table[0, {i, 1, 151}];
MSE = Table[0, {i, 1, 151}];
rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters → {1, -1}];
  rxsignal = Table[1/(Length[signal] - k)
     Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal] - 1 - k}], {k, 0, Length[signal] - 1}];
  rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
  realnoise = noisescale * white;
  acnoise = Table[1/(Length[realnoise] - k)
     Sum[realnoise[[n + 1]] * realnoise[[n + 1 + k]], {n, 0, Length[realnoise] - 1 - k}], {k, 0, Length[realnoise] - 1}];
  actualPSD = Fourier[acnoise, FourierParameters → {1, -1}];
  rn = Table[0, {i, 1, bign}];

kk = aa^2/(4 ts);
For [j = 1, j < bign/4, j++,
 sx1 = rcvdPSD[[j]];
 sx2 = rcvdPSD[[bign/2 - j + 1]];
 sx3 = rcvdPSD[[bign/2 + j]];
 sx4 = rcvdPSD[[bign - j + 1]];
 Φ1 = kk shapefourier[[j]]^2;
 Φ2 = kk shapefourier[[bign/2 - j + 1]]^2;
 Φ3 = kk shapefourier[[bign/2 + j]]^2;
 Φ4 = kk shapefourier[[bign - j + 1]]^2;
 f1 = sx1/Φ1;
 f2 = sx2/Φ2;
 f3 = sx3/Φ3;
 f4 = sx4/Φ4;
 mn = Min[Abs[f1], Abs[f2], Abs[f3], Abs[f4]]; (*the minimum*)
 delta1 = (f1 - mn); (*subtract the minimum*)
 rn[[j]] = rn[[j]] + delta1 Φ1; (*multiply by the pulse shape and constant*)
 delta2 = (f2 - mn);
 rn[[bign/2 - j + 1]] = rn[[bign/2 - j + 1]] + delta2 Φ2;
 delta3 = (f3 - mn);
 rn[[bign/2 + j]] = rn[[bign/2 + j]] + delta3 Φ3;
 delta4 = (f4 - mn);
 rn[[bign - j + 1]] = rn[[bign - j + 1]] + delta4 Φ4;]
PSDnoise = rn;
  Clear[rn];
  temp = 1/bign Total[(actualPSD - PSDnoise)^2];
  MSE[[i]] = temp;
  psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
     PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) *
       (2 π tt w - somega[[lft]]), 0];
  bigdhat = Table[(Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
        hmmse[endomega[[i]]] XofF[endomega[[i]]]) +
      (Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
        hmmse[endomega[[i]] + 1/ts] XofF[endomega[[i]] + 1/ts]) +
      (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
        Conjugate[hmmse[2 fc - endomega[[i]]]]
        Conjugate[XofF[2 fc - endomega[[i]]]]) +
      (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
        Exp[I 2 π tau/ts] Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
        Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
  dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]];
  BER = N[(Total[Abs[dhat-d]]/2)/ns];
  results[[i]] = BER;]]
```

This section plots the results and stores them to be used in a comparison of all the results.

```
aa = 1;
results = Take[results, -(Length[results] - 1)];
rng = Take[rng, -(Length[rng] - 1)];
MSE = Take[MSE, -(Length[MSE] - 1)];
MSE >> MSERNMIN;
results >> BERRNMIN;
noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa/noisescale[[i]]], {i, 1, Length[noisescale]}];
ListPlot[Transpose[{rng, results}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Noise scale=10^x", "BER"}, Frame → True, Axes → False,
 GridLines → Automatic]

ListPlot[Transpose[{SNR, results}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 rameLabel → {"Signal to Noise Ratio in dB", "BER"}, Frame → True,
 Axes → False, GridLines → Automatic]

ListPlot[Transpose[{rng, Abs[MSE]}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Noise scale=10^x", "MSE"}, Frame → True, Axes → False,
 GridLines → Automatic]

ListPlot[Transpose[{SNR, Abs[MSE]}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "MSE"}, Frame → True,
 Axes → False, GridLines → Automatic]
```

This notebook plots the estimated PSD and the actual PSD, the noise scale here is fixed as 1 to get a good comparison of the noise. All of the variables are the same as in the notebook "Data Generation." This notebook was used to create Figures 4.9, 4.10, and 4.11.

```
<< Statistics`ContinuousDistributions`
ts = 1
ns = 2^6
d = Table[2 Random[Integer] - 1, {i, 1, ns}];
tau = 0;
fc = 0
k = 2^4;
tt = 1/k;
bign = (ns ts)/tt;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
noisescale = 1;
sinc[t_] := If[t ≠ 0, Sin[π t]/(π t), 1] ;
nyq[t_] := 1/√ts (sinc[2 t/√ts - 1/2] + sinc[2 t/√ts + 1/2]);
shape[t_] := nyq[t];

s[t_] := N[ Re[aa Sum[(d[[n + 1]] shape[t - (n - ns/2) ts - tau] Exp[((I 2 π fc t) + theta)]), {n, 0, ns - 1}]]];
white = noisescale * Table[Random[NormalDistribution[mu, sigma]], {i, 1, bign}];
whitefourier = Fourier[white, FourierParameters → {1, -1}];
noisefourier = whitefourier;
noise = white;


Timing[tx = Table[(-ts (ns + 1))/2 + (ns ts)/(bign - 1) i, {i, 0, bign - 1}];
 ω = Range[-π, π, (2 π)/(bign - 1)];
 st = Table[s[tx[[i]]], {i, 1, Length[tx]}];
 signal = st + white;
 stf = Fourier[st, FourierParameters → {1, -1}];
 signalfourier = Fourier[signal, FourierParameters → {1, -1}];

 shape = Table[shape[tx[[i]]], {i, 1, Length[tx]}];
 shapefourier = Fourier[shape, FourierParameters → {1, -1}];
 rxsignal = Table[1/(Length[signal] - k)
       Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal] - 1 - k}], {k, 0, Length[signal] - 1}];
 rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
 rxnoise = Table[1/(Length[noise] - k)
       Sum[noise[[n + 1]] * noise[[n + 1 + k]], {n, 0, Length[noise] - 1 - k}], {k, 0, Length[noise] - 1}];
 ActualPSDnoise = Fourier[rxnoise, FourierParameters → {1, -1}];]
```

```
rn = Table[0, {i, 1, bign}];
kk = aa^2/(4 ts); (*scale factor*)
For [i = 1, i < bign/4, i++,
  sx1 = rcvdPSD[[i]];
  sx2 = rcvdPSD[[bign/2 - i + 1]];
  sx3 = rcvdPSD[[bign/2 + i]];
  sx4 = rcvdPSD[[bign - i + 1]];
  Φ1 = kk shapefourier[[i]]^2;
  Φ2 = kk shapefourier[[bign/2 - i + 1]]^2;
  Φ3 = kk shapefourier[[bign/2 + i]]^2;
  Φ4 = kk shapefourier[[bign - i + 1]]^2;
  f1 = sx1/Φ1;
  f2 = sx2/Φ2;
  f3 = sx3/Φ3;
  f4 = sx4/Φ4;
  mn = Min[Abs[f1], Abs[f2], Abs[f3], Abs[f4]]; (*minimum redundancy*)
  delta1 = (f1 - mn); (*subtract the minimum*)
  rn[[i]] = rn[[i]] + delta1 Φ1; (*multiply by the pulse shape*)
  delta2 = (f2 - mn);
  rn[[bign/2 - i + 1]] = rn[[bign/2 - i + 1]] + delta2 Φ2;
  delta3 = (f3 - mn);
  rn[[bign/2 + i]] = rn[[bign/2 + i]] + delta3 Φ3;
  delta4 = (f4 - mn);
  rn[[bign - i + 1]] = rn[[bign - i + 1]] + delta4 Φ4;]
```

This section plots the results.

```
PSDnoise = rn;
p = ActualPSDnoise;
phat = PSDnoise;
diff = p - phat;
ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True,
  Frame → True, PlotLabel → "PSD Estimated", PlotRange → All]
ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True,
  Frame → True, PlotLabel → "PSD Estimated", PlotRange → {Automatic, {0, 9}}]
ListPlot[Transpose[{ω, Abs[ActualPSDnoise]}], PlotJoined →
   True, Frame → True, PlotLabel → "PSD Actual", PlotRange → All]
```

6.5.2 *Subtraction of the Minimum Redundancy After Multiplication by Pulse Shape.*

This portion of the Mathematica notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." The vectors' meaning is the same as in the first notebook. This notebook was used to create Figures 4.4 and 4.16.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
shape = << shape;
shapefourier = << shapefourier;
```
$\texttt{ns} = 2^6;$

$\texttt{nf} = \texttt{ns};$

$\texttt{fc} = 1;$

$\texttt{k} = 2^4;$

$\texttt{tt} = \frac{1}{\texttt{k fc}};$

$\texttt{bign} = \frac{\texttt{ns ts}}{\texttt{tt}};$

$\texttt{ts} = 1;$

$\texttt{tau} = 0;$

$\texttt{theta} = 0;$

$\texttt{aa} = 1;$

$\texttt{mu} = 0;$

$\texttt{sigma} = 1;$

$\texttt{somega} = \texttt{Table}\left[\frac{2\pi}{\texttt{bign}}\texttt{k},\ \{\texttt{k},\ 0,\ \texttt{bign}-1\}\right];$

$\texttt{endomega} = \texttt{Sort}\left[\texttt{N}\left[\texttt{Table}\left[\texttt{fc} - \frac{\texttt{i} - 1\, 10^{-3}}{\texttt{nf ts}},\ \{\texttt{i},\ 1,\ \texttt{nf}\}\right]\right]\right];$

$\texttt{z} = \texttt{Table[Exp[}2\pi\,\texttt{I (endomega[[k]] - fc) ts]},\ \{\texttt{k},\ 1,\ \texttt{nf}\}];$

$\texttt{zzz} = \texttt{N}\left[\texttt{Transpose}\left[\texttt{Table}\left[\texttt{z}^{-\texttt{k}},\ \{\texttt{k},\ 0,\ \texttt{nf}-1\}\right]\right]\right];$

$\texttt{MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];}$

$\texttt{bigpsi[k\_]} := \texttt{If}\left[\frac{-1}{\texttt{ts}} < \texttt{k} < \frac{1}{\texttt{ts}},\ \sqrt{\texttt{ts}}\,\texttt{Cos}\left[\pi\,\frac{\texttt{k ts}}{2}\right],\ 0\right];$

$\texttt{XofF[d\_]} := \texttt{If}\Big[0 < \texttt{Abs[d]} < \frac{1}{\texttt{tt}},\ \texttt{lft} = \texttt{Min[bign - 1, 1 + Floor[bign Abs[d] tt]]};$
$\quad \texttt{signalfourier[[lft]]} + \frac{\texttt{signalfourier[[lft + 1]] - signalfourier[[lft]]}}{\texttt{somega[[lft + 1]] - somega[[lft]]}}$
$\quad * (2\pi\,\texttt{tt d} - \texttt{somega[[lft]]}),\ 0\Big];$

$$\texttt{hmmse[p\_]} := \left(\frac{\texttt{aa}}{2}\ \frac{\texttt{Conjugate[bigpsi[p - fc]]}}{\texttt{ts psd[p]}}\right) \Big/ \left(1 + \frac{\texttt{aa\^{}2}}{4}\ \frac{\texttt{Abs[bigpsi[p - fc]]\^{}2}}{\texttt{ts psd[p]}} + \frac{\texttt{aa\^{}2}}{4}\ \frac{\texttt{Abs}\left[\texttt{bigpsi}\left[\texttt{p} + \frac{1}{\texttt{ts}} - \texttt{fc}\right]\right]\texttt{\^{}2}}{\texttt{ts psd}\left[\texttt{p} + \frac{1}{\texttt{ts}}\right]} + \frac{\texttt{aa\^{}2}}{4}\ \frac{\texttt{Abs[bigpsi[fc - p]]\^{}2}}{\texttt{ts psd[2 fc - p]}} + \frac{\texttt{aa\^{}2}}{4}\ \frac{\texttt{Abs}\left[\texttt{bigpsi}\left[\texttt{fc - p} - \frac{1}{\texttt{ts}}\right]\right]\texttt{\^{}2}}{\texttt{ts psd}\left[\texttt{2 fc - p} - \frac{1}{\texttt{ts}}\right]}\right);$$

This section is the loop that tests each of the various noise scales, recovers data symbols at each noise scale, and compares the actual PSD to the estimated PSD using MSE.

```
results = Table[0, {i, 1, 151}];
MSE = Table[0, {i, 1, 151}];
rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters → {1, -1}];
  rxsignal = Table[1/(Length[signal] - k)
       Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal]-1-k}], {k, 0, Length[signal] - 1}];
  rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
  realnoise = noisescale * white;
  acnoise = Table[1/(Length[realnoise] - k)
       Sum[realnoise[[n + 1]] * realnoise[[n + 1 + k]], {n, 0, Length[realnoise]-1-k}], {k, 0, Length[realnoise] - 1}];
  actualPSD = Fourier[acnoise, FourierParameters → {1, -1}];
  rn = Table[0, {i, 1, bign}];
```

```
kk = aa^2/(4 ts);
For[j = 1, j < bign/4, j++,
 sx1 = rcvdPSD[[j]];
 sx2 = rcvdPSD[[bign/2 - j + 1]];
  sx3 = rcvdPSD[[bign/2 + j]];
  sx4 = rcvdPSD[[bign - j + 1]];
 Ψ1 = kk shapefourier[[j]]^2;
 Ψ2 = kk shapefourier[[bign/2 - j + 1]]^2;
 Ψ3 = kk shapefourier[[bign/2 + j]]^2;
 Ψ4 = kk shapefourier[[bign - j + 1]]^2;
 f1 = sx1/Ψ1;
 f2 = sx2/Ψ2;
 f3 = sx3/Ψ3;
 f4 = sx4/Ψ4;
 mn = Min[Abs[f1], Abs[f2], Abs[f3], Abs[f4]]; (*the minimum*)
 delta1 = (sx1 - mn Ψ1);
 (*subtract the minimum after multiplication by the pulse shape and constant*)
 rn[[j]] = rn[[j]] + delta1;
 delta2 = (sx2 - mn Ψ2);
 rn[[bign/2 - j + 1]] = rn[[bign/2 - j + 1]] + delta2;
 delta3 = (sx3 - mn Ψ3);
 rn[[bign/2 + j]] = rn[[bign/2 + j]] + delta3;
 delta4 = (sx4 - mn Ψ4);
 rn[[bign - j + 1]] = rn[[bign - j + 1]] + delta4;]
```

```
PSDnoise = rn;
temp = 1/bign Total[(actualPSD - PSDnoise)^2];
MSE[[i]] = temp;
Clear[rn];
psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
   PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) *
     (2 π tt w - somega[[lft]]), 0];

bigdhat = Table[(Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       hmmse[endomega[[i]]] XofF[endomega[[i]]]) +
    (Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
       hmmse[endomega[[i]] + 1/ts] XofF[endomega[[i]] + 1/ts]) +
    (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
       Conjugate[hmmse[2 fc - endomega[[i]]]]
       Conjugate[XofF[2 fc - endomega[[i]]]]) +
    (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
       Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
       Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]];
BER = N[(Total[Abs[dhat-d]]/2)/ns];
results[[i]] = BER;]]
```

This section plots the results and stores them to be used in a comparison of all the results.

```
aa = 1;
results = Take[results, -(Length[results] - 1)];
rng = Take[rng, -(Length[rng] - 1)];
MSE = Take[MSE, -(Length[MSE] - 1)];
MSE >> MSERNEST;
results >> BERRNEST;
noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa/noisescale[[i]]], {i, 1, Length[noisescale]}];
ListPlot[Transpose[{rng, results}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Noise scale=10^x", "BER"}, Frame → True, Axes → False,
 GridLines → Automatic]

ListPlot[Transpose[{SNR, results}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "BER"}, Frame → True,
 Axes → False, GridLines → Automatic]

ListPlot[Transpose[{rng, Abs[MSE]}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Noise scale=10^x", "MSE"}, Frame → True, Axes → False,
 GridLines → Automatic]

ListPlot[Transpose[{SNR, Abs[MSE]}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "MSE"}, Frame → True,
 Axes → False, GridLines → Automatic]
```

This Mathematica notebook plots the estimated PSD and the actual PSD, the noise scale here is fixed as 1. All of the variables are the same as in the notebook "Data Generation."

It was used to create Figures 4.13, 4.14, and 4.15.

```
<< Statistics`ContinuousDistributions`
ts = 1
ns = 2^6
d = Table[2 Random[Integer] - 1, {i, 1, ns}];
tau = 0;
fc = 0
k = 2^4;
tt = 1/k;
bign = (ns ts)/tt;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
noisescale = 1;
sinc[t_] := If[t ≠ 0, Sin[π t]/(π t), 1] ;
nyq[t_] := 1/Sqrt[ts] (sinc[2 t/Sqrt[ts] - 1/2] + sinc[2 t/Sqrt[ts] + 1/2]);
shape[t_] := nyq[t];

s[t_] := N[ Re[aa Sum[(d[[n + 1]] shape[t - (n - ns/2) ts - tau] Exp[((I 2 π fc t) + theta)]), {n, 0, ns - 1}]]];
white = noisescale * Table[Random[NormalDistribution[mu, sigma]], {i, 1, bign}];
whitefourier = Fourier[white, FourierParameters → {1, -1}];
noisefourier = whitefourier;
noise = white;


Timing[tx = Table[(-ts (ns + 1))/2 + (ns ts)/(bign - 1) i, {i, 0, bign - 1}];
 ω = Range[-π, π, (2 π)/(bign - 1)];
 st = Table[s[tx[[i]]], {i, 1, Length[tx]}];
 signal = st + white;
 stf = Fourier[st, FourierParameters → {1, -1}];
 signalfourier = Fourier[signal, FourierParameters → {1, -1}];

 shape = Table[shape[tx[[i]]], {i, 1, Length[tx]}];
 shapefourier = Fourier[shape, FourierParameters → {1, -1}];
 rxsignal = Table[1/(Length[signal] - k)
      Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal] - 1 - k}], {k, 0, Length[signal] - 1}];
 rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
 rxnoise = Table[1/(Length[noise] - k)
      Sum[noise[[n + 1]] * noise[[n + 1 + k]], {n, 0, Length[noise] - 1 - k}], {k, 0, Length[noise] - 1}];
 ActualPSDnoise = Fourier[rxnoise, FourierParameters → {1, -1}];]
```

This section creates the estimated noise PSD, plots the estimated noise PSD zoomed out and zoomed in, and plots the actual noise PSD.

```
rn = Table[0, {i, 1, bign}];
kk = aa^2/(4 ts); (*scale factor*)
For[i = 1, i < bign/4, i++,
 sx1 = rcvdPSD[[i]];
 sx2 = rcvdPSD[[bign/2 - i]];
  sx3 = rcvdPSD[[bign/2 + i]];
  sx4 = rcvdPSD[[bign - i]];

 Φ1 = kk shapefourier[[i]]^2;
 Φ2 = kk shapefourier[[bign/2 - i]]^2;
 Φ3 = kk shapefourier[[bign/2 + i]]^2;
 Φ4 = kk shapefourier[[bign - i]]^2;

 f1 = sx1/Φ1;
 f2 = sx2/Φ2;
 f3 = sx3/Φ3;
 f4 = sx4/Φ4;
 mn = Min[Abs[f1], Abs[f2], Abs[f3], Abs[f4]]; (*minimum redundancy*)
  delta1 = (sx1 - mn Φ1); (*subtract the minimum after multiplication by pulse shape*)
  rn[[i]] = rn[[i]] + delta1;
  delta2 = (sx2 - mn Φ2);
  rn[[bign/2 - i]] = rn[[bign/2 - i]] + delta2;
  delta3 = (sx3 - mn Φ3);
  rn[[bign/2 + i]] = rn[[bign/2 + i]] + delta3;
  delta4 = (sx4 - mn Φ4);
  rn[[bign - i]] = rn[[bign - i]] + delta4;]

 PSDnoise = rn;
 p = ActualPSDnoise;
 phat = PSDnoise;
 diff = p - phat;
 ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True, Frame → True,
  PlotLabel → "PSD Estimated", PlotRange → All]
 ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True, Frame → True,
  PlotLabel → "PSD Estimated", PlotRange → {Automatic, {0, 9}}]
 ListPlot[Transpose[{ω, Abs[ActualPSDnoise]}], PlotJoined → True, Frame → True,
  PlotLabel → "PSD Actual", PlotRange → All]
```

6.6 *Subtraction of the Average Redundancy*

6.6.1 *Subtraction of the Average Redundancy Before Multiplication by Pulse Shape.*

This portion of the Mathematica notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." Each of these experiments uses the same vectors. This notebook was used to create Figures 4.5 and 4.20.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
shape = << shape;
shapefourier = << shapefourier;
ns = 2^6;
nf = ns;
fc = 1;
k = 2^4;
tt = 1/(k fc);
bign = (ns ts)/tt;
ts = 1;
tau = 0;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
somega = Table[(2 π)/bign k, {k, 0, bign - 1}];


endomega = Sort[N[Table[fc - (i - 1 10^-3)/(nf ts), {i, 1, nf}]]];
z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}];
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]];
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
bigpsi[k_] := If[-1/ts < k < 1/ts, Sqrt[ts] Cos[π (k ts)/2], 0];
XofF[d_] := If[0 < Abs[d] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[d]  tt]];
   signalfourier[[lft]] + (signalfourier[[lft + 1]] - signalfourier[[lft]])/(somega[[lft + 1]] - somega[[lft]]) * (2 π tt d - somega[[lft]]), 0];

hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p])) /
  (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) + aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts]) + aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) +
   aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
```

This section is the loop that tests each of the various noise scales, recovers data symbols at each noise scale, and compares the actual PSD to the estimated PSD using MSE.

```
results = Table[0, {i, 1, 151}];
MSE = Table[0, {i, 1, 151}];

rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters -> {1, -1}];
  rxsignal = Table[1/(Length[signal] - k) Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal] - 1 - k}], {k, 0, Length[signal] - 1}];
  rcvdPSD = Fourier[rxsignal, FourierParameters -> {1, -1}];
  realnoise = noisescale * white;
  acnoise = Table[1/(Length[realnoise] - k) Sum[realnoise[[n + 1]] * realnoise[[n + 1 + k]], {n, 0, Length[realnoise] - 1 - k}], {k, 0, Length[realnoise] - 1}];
  actualPSD = Fourier[acnoise, FourierParameters -> {1, -1}];
  rn = Table[0, {i, 1, bign}];
  kk = aa^2/(4 ts);

For [j = 1, j < bign/4, j++,
 sx1 = rcvdPSD[[j]];
 sx2 = rcvdPSD[[bign/2 - j + 1]];
 sx3 = rcvdPSD[[bign/2 + j]];
 sx4 = rcvdPSD[[bign - j + 1]];
 Ψ1 = kk shapefourier[[j]]^2;
 Ψ2 = kk shapefourier[[bign/2 - j + 1]]^2;
 Ψ3 = kk shapefourier[[bign/2 + j]]^2;
 Ψ4 = kk shapefourier[[bign - j + 1]]^2;
 f1 = sx1/Ψ1;
 f2 = sx2/Ψ2;
 f3 = sx3/Ψ3;
 f4 = sx4/Ψ4;
 sdhat = 1/4 (f1 + f2 + f3 + f4); (*calculate the average*)

delta1 = (f1 - sdhat); (*subtract the average*)
 rn[[j]] = rn[[j]] + delta1 Ψ1; (*multiply by the pulse shape and constant*)
 delta2 = (f2 - sdhat);
 rn[[bign/2 - j + 1]] = rn[[bign/2 - j + 1]] + delta2 Ψ2;
 delta3 = (f3 - sdhat);
 rn[[bign/2 + j]] = rn[[bign/2 + j]] + delta3 Ψ3;
 delta4 = (f4 - sdhat);
 rn[[bign - j + 1]] = rn[[bign - j + 1]] + delta4 Ψ4 ;]
PSDnoise = rn;
temp = 1/bign Total[(actualPSD - PSDnoise)^2];
MSE[[i]] = temp;
Clear[rn];
psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
   PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) * (2 π tt w - somega[[lft]]), 0];
```

```
    bigdhat = Table[(Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau] hmmse[endomega[[i]]]
        XofF[endomega[[i]]]) + (Exp[- I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Exp[I 2 π tau/ts]
        hmmse[endomega[[i]] + 1/ts] XofF[endomega[[i]] + 1/ts]) +
      (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau] Conjugate[hmmse[2 fc - endomega[[i]]]]
        Conjugate[XofF[2 fc - endomega[[i]]]]) + (Exp[ I theta] Exp[I 2 π (endomega[[i]] - fc) tau]
        Exp[I 2 π tau/ts]
        Conjugate[hmmse[2 fc - endomega[[i]] - 1/ts]]
        Conjugate[XofF[2 fc - endomega[[i]] - 1/ts]]), {i, 1, Length[endomega]}];
  dhat = Sign[Chop[LinearSolve[zzz, bigdhat]]];
  BER = N[(Total[Abs[dhat-d]]/2)/ns] ;
  results[[i]] = BER;]]
```

This section plots the results and stores them to be used in a comparison of all the results.

```
aa = 1;
results = Take[results, -(Length[results] - 1)];
rng = Take[rng, -(Length[rng] - 1)];
MSE = Take[MSE, -(Length[MSE] - 1)];
MSE >> MSEAverage;
results >> BERAverage;
noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa/noisescale[[i]]], {i, 1, Length[noisescale]}];
ListPlot[Transpose[{SNR, results}], PlotJoined→ True, PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "BER" }, Frame → True, Axes→ False, GridLines → Automatic]
ListPlot[Transpose[{SNR, Abs[MSE]}], PlotJoined → True, PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "MSE" }, Frame → True, Axes→ False, GridLines → Automatic]
```

This Mathematica notebook plots the estimated PSD and the actual PSD, the noise scale here is fixed as 1. All of the variables are the same as in the notebook "Data Generation." This notebook creates Figures 4.17, 4.18, and 4.19.

```
<< Statistics`ContinuousDistributions`
ts = 1
ns = 2^6
d = Table[2 Random[Integer] - 1, {i, 1, ns}];
tau = 0;
fc = 0
k = 2^4;
tt = 1/k;
bign = (ns ts)/tt;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
noisescale = 1;
sinc[t_] := If[t ≠ 0, Sin[π t]/(π t), 1] ;
nyq[t_] := 1/√ts (sinc[2 t/√ts - 1/2] + sinc[2 t/√ts + 1/2]);
shape[t_] := nyq[t];

s[t_] := N[ Re[aa Sum[(d[[n+ 1]] shape[t - (n- ns/2) ts - tau] Exp[(( I 2 π fc t) + theta)]), {n, 0, ns-1}]]];
white = noisescale * Table[Random[NormalDistribution[mu, sigma]], {i, 1, bign}];
whitefourier = Fourier[white, FourierParameters → {1, -1}];
noisefourier = whitefourier;
noise = white;

Timing[tx = Table[(-ts (ns + 1))/2 + (ns ts)/(bign - 1) i, {i, 0, bign - 1}];
 (*ω=Table[(2 π)/bign k, {k, 0, bign-1}];*)
 ω = Range[-π, π, (2 π)/(bign - 1)];
 st = Table[s[tx[[i]]], {i, 1, Length[tx]}];
 signal = st + white;
 stf = Fourier[st, FourierParameters → {1, -1}];
 signalfourier = Fourier[signal, FourierParameters → {1, -1}];
 shape = Table[shape[tx[[i]]], {i, 1, Length[tx]}];
 shapefourier = Fourier[shape, FourierParameters → {1, -1}];
 rxsignal = Table[1/(Length[signal] - k)
      Sum[signal[[n+ 1]] * signal[[n+ 1+ k]], {n, 0, Length[signal]-1-k}], {k, 0, Length[signal] - 1}];
 rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
 rxnoise = Table[1/(Length[noise] - k)
      Sum[noise[[n+ 1]] * noise[[n+ 1+ k]], {n, 0, Length[noise]-1-k}], {k, 0, Length[noise] - 1}];
 ActualPSDnoise = Fourier[rxnoise, FourierParameters → {1, -1}];]
```

This section creates the estimated noise PSD, plots the estimated noise PSD zoomed out and zoomed in, and plots the actual noise PSD.

```
rn = Table[0, {i, 1, bign}];
kk = aa^2/(4 ts); (*scale factor*)
For[i = 1, i < bign/4, i++,
 sx1 = rcvdPSD[[i]];
 sx2 = rcvdPSD[[bign/2 - i + 1]];
 sx3 = rcvdPSD[[bign/2 + i]];
 sx4 = rcvdPSD[[bign - i + 1]];

 Ψ1 = kk shapefourier[[i]]^2;
 Ψ2 = kk shapefourier[[bign/2 - i + 1]]^2;
 Ψ3 = kk shapefourier[[bign/2 + i]]^2;
 Ψ4 = kk shapefourier[[bign - i + 1]]^2;

 f1 = sx1/Ψ1;
 f2 = sx2/Ψ2;
 f3 = sx3/Ψ3;
 f4 = sx4/Ψ4;
 sdhat = 1/4 (f1 + f2 + f3 + f4); (*calculate the average redundancy*)


 delta1 = (f1 - sdhat); (*subtract the average*)
  rn[[i]] = rn[[i]] + delta1 Ψ1; (*multiply by the scale factor and pulse shape*)
  delta2 = (f2 - sdhat);
  rn[[bign/2 - i + 1]] = rn[[bign/2 - i + 1]] + delta2 Ψ2;
  delta3 = (f3 - sdhat);
  rn[[bign/2 + i]] = rn[[bign/2 + i]] + delta3 Ψ3;
  delta4 = (f4 - sdhat);
  rn[[bign - i + 1]] = rn[[bign - i + 1]] + delta4 Ψ4;]

PSDnoise = rn;
p = ActualPSDnoise;
phat = PSDnoise;
diff = p - phat;
ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True, Frame → True,
 PlotRange → {Automatic, {0, 9}}, PlotLabel → "PSD Estimated"]
ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True, Frame → True,
 PlotRange → All, PlotLabel → "PSD Estimated"]
ListPlot[Transpose[{ω, Abs[ActualPSDnoise]}], PlotJoined → True, Frame → True,
 PlotLabel → "PSD Actual"]
```

6.6.2 Subtraction of the Average Redundancy After Multiplication by Pulse Shape.

This portion of the notebook creates the variables used throughout. Some are recovered from data vectors created in the notebook "Data Generation." Each of these experiments uses the same vectors. This notebook was used to create Figures 4.24 and 4.6.

```
<< Statistics`ContinuousDistributions`
white = << white;
st = << st;
d = << d;
shape = << shape;
shapefourier = << shapefourier;
aa = 1;
ns = 2^6;
nf = ns;
fc = 1;
k = 2^4;
tt = 1/(k fc);
bign = (ns ts)/tt;
ts = 1;
tau = 0;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
somega = Table[(2 π)/bign k, {k, 0, bign - 1}];

endomega = Sort[N[Table[fc - (i - 1 10^-3)/(nf ts), {i, 1, nf}]]];
z = Table[Exp[2 π I (endomega[[k]] - fc) ts] , {k, 1, nf}];
zzz = N[Transpose[Table[z^-k, {k, 0, nf - 1}]]];
MatrixForm[Chop[Conjugate[Transpose[zzz]].zzz]];
bigpsi[k_] := If[-1/ts < k < 1/ts, √ts Cos[π (k ts)/2], 0];
hmmse[p_] := (aa/2 Conjugate[bigpsi[p - fc]]/(ts psd[p])) /
    (1 + aa^2/4 Abs[bigpsi[p - fc]]^2/(ts psd[p]) +
      aa^2/4 Abs[bigpsi[p + 1/ts - fc]]^2/(ts psd[p + 1/ts]) + aa^2/4 Abs[bigpsi[fc - p]]^2/(ts psd[2 fc - p]) +
      aa^2/4 Abs[bigpsi[fc - p - 1/ts]]^2/(ts psd[2 fc - p - 1/ts]));
psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
   PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) * (2 π tt w - somega[[lft]]), 0];
XofF[d_] := If[0 < Abs[d] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[d] tt]];
   signalfourier[[lft]] + (signalfourier[[lft + 1]] - signalfourier[[lft]])/(somega[[lft + 1]] - somega[[lft]])
     * (2 π tt d - somega[[lft]]), 0];
Null
```

This section is the loop that tests each of the various noise scales, recovers data symbols at each noise scale, and compares the actual PSD to the estimated PSD using MSE.

```
results = Table[0, {i, 1, 151}];
MSE = Table[0, {i, 1, 151}];
rng = Range[-1.5, 1.5, .02];
Timing[For[i = 1, i < 151, i++
  noisescale = 10^rng[[i]];
  signal = (noisescale white) + st;
  signalfourier = Fourier[signal, FourierParameters → {1, -1}];
  rxsignal = Table[1/(Length[signal] - k)
      Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal] - 1 - k}], {k, 0, Length[signal] - 1}];
  rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
  realnoise = noisescale * white;
  acnoise = Table[1/(Length[realnoise] - k)
      Sum[realnoise[[n + 1]] * realnoise[[n + 1 + k]], {n, 0, Length[realnoise] - 1 - k}], {k, 0, Length[realnoise] - 1}];
  actualPSD = Fourier[acnoise, FourierParameters → {1, -1}];
  rn = Table[0, {i, 1, bign}];
  kk = aa^2/(4 ts);

For [j = 1, j < bign/4, j++,
 sx1 = rcvdPSD[[j]];
 sx2 = rcvdPSD[[bign/2 - j + 1]];
  sx3 = rcvdPSD[[bign/2 + j]];
  sx4 = rcvdPSD[[bign - j + 1]];
 Ψ1 = kk shapefourier[[j]]^2;
 Ψ2 = kk shapefourier[[bign/2 - j + 1]]^2;
 Ψ3 = kk shapefourier[[bign/2 + j]]^2;
 Ψ4 = kk shapefourier[[bign - j + 1]]^2;
 f1 = sx1/Ψ1;
 f2 = sx2/Ψ2;
 f3 = sx3/Ψ3;
 f4 = sx4/Ψ4;
sdhat = 1/4 (f1 + f2 + f3 + f4); (*calculate the average*)
 delta1 = (sx1 - sdhat Φ1);
 (*subtract the average multiplied by pulse shape and constant*)
 rn[[j]] = rn[[j]] + delta1;
 delta2 = (sx2 - sdhat Φ2);
 rn[[bign/2 - j + 1]] = rn[[bign/2 - j + 1]] + delta2 ;
 delta3 = (sx3 - sdhat Φ3 );
 rn[[bign/2 + j]] = rn[[bign/2 + j]] + delta3 ;
 delta4 = (sx4 - sdhat Φ4);
 rn[[bign - j + 1]] = rn[[bign - j + 1]] + delta4 ;]
PSDnoise = rn;
temp = 1/bign Total[(actualPSD - PSDnoise)^2];
MSE[[i]] = temp;
Clear[rn];
psd[w_] := If[0 < Abs[w] < 1/tt, lft = Min[bign - 1, 1 + Floor[bign Abs[w] tt]];
   PSDnoise[[lft]] + (PSDnoise[[lft + 1]] - PSDnoise[[lft]])/(somega[[lft + 1]] - somega[[lft]]) * (2 π tt w - somega[[lft]]), 0];
```

This section plots the results and stores them to be used in a comparison of all the results.

```
aa = 1;
results = Take[results, -(Length[results] - 1)];
rng = Take[rng, -(Length[rng] - 1)];
MSE = Take[MSE, -(Length[MSE] - 1)];

MSE >> MSEAverageNew;
results >> BERAverageNew;
noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
SNR = Table[20 Log[aa/noisescale[[i]]], {i, 1, Length[noisescale]}];
ListPlot[Transpose[{SNR, results}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "BER"},
 Frame → True, Axes → False, GridLines → Automatic]
ListPlot[Transpose[{SNR, Abs[MSE]}], PlotJoined → True, PlotRange → All,
 TextStyle → {FontFamily → "Times", FontSize → 12},
 FrameLabel → {"Signal to Noise Ratio in dB", "MSE"}, Frame → True,
 Axes → False, GridLines → Automatic]
```

This Mathematica notebook plots the estimated PSD and the actual PSD, the noise scale here is fixed as 1. All of the variables are the same as in the notebook "Data Generation." This notebook creates Figures 4.21, 4.22, and 4.23.

```
<< Statistics`ContinuousDistributions`
ts = 1
ns = 2^6
d = Table[2 Random[Integer] - 1, {i, 1, ns}];
tau = 0;
fc = 0
k = 2^4;
tt = 1/k;
bign = (ns ts)/tt;
theta = 0;
aa = 1;
mu = 0;
sigma = 1;
noisescale = 1;
sinc[t_] := If[t ≠ 0, Sin[π t]/(π t), 1] ;
nyq[t_] := 1/√ts (sinc[2 t/√ts - 1/2] + sinc[2 t/√ts + 1/2]);
shape[t_] := nyq[t];

s[t_] := N[Re[aa Sum[(d[[n + 1]] shape[t - (n - ns/2) ts - tau] Exp[(( I 2 π fc t) + theta)]), {n, 0, ns - 1}]]];
white = noisescale * Table[Random[NormalDistribution[mu, sigma]], {i, 1, bign}];
whitefourier = Fourier[white, FourierParameters → {1, -1}];
noisefourier = whitefourier;
noise = white;


Timing[tx = Table[(-ts (ns + 1))/2 + (ns ts)/(bign - 1) i, {i, 0, bign - 1}];
 (*ω=Table[(2 π)/bign k, {k, 0, bign-1}];*)
 ω = Range[-π, π, (2 π)/(bign - 1)];
 st = Table[s[tx[[i]]], {i, 1, Length[tx]}];
 signal = st + white;
 stf = Fourier[st, FourierParameters → {1, -1}];
 signalfourier = Fourier[signal, FourierParameters → {1, -1}];
 shape = Table[shape[tx[[i]]], {i, 1, Length[tx]}];
 shapefourier = Fourier[shape, FourierParameters → {1, -1}];
 rxsignal = Table[1/(Length[signal] - k)
     Sum[signal[[n + 1]] * signal[[n + 1 + k]], {n, 0, Length[signal] - 1 - k}], {k, 0, Length[signal] - 1}];
 rcvdPSD = Fourier[rxsignal, FourierParameters → {1, -1}];
 rxnoise = Table[1/(Length[noise] - k)
     Sum[noise[[n + 1]] * noise[[n + 1 + k]], {n, 0, Length[noise] - 1 - k}], {k, 0, Length[noise] - 1}];
 ActualPSDnoise = Fourier[rxnoise, FourierParameters → {1, -1}];]
```

This section creates the estimated noise PSD, plots the estimated noise PSD zoomed out and zoomed in, and plots the actual noise PSD.

```
rn = Table[0, {i, 1, bign}];
kk = aa^2/(4 ts); (*scale factor*)
For [i = 1, i < bign/4, i++,
 sx1 = rcvdPSD[[i]];
 sx2 = rcvdPSD[[bign/2 - i + 1]];
  sx3 = rcvdPSD[[bign/2 + i]];
  sx4 = rcvdPSD[[bign - i + 1]];

 Ψ1 = kk shapefourier[[i]]^2;
 Ψ2 = kk shapefourier[[bign/2 - i + 1]]^2;
 Ψ3 = kk shapefourier[[bign/2 + i]]^2;
 Ψ4 = kk shapefourier[[bign - i + 1]]^2;
 f1 = sx1/Ψ1;
 f2 = sx2/Ψ2;
 f3 = sx3/Ψ3;
 f4 = sx4/Ψ4;
sd = {f1, f2, f3, f4};
 sdhat = Mean[sd]; (*calculate the average redundancy*)
 delta1 = (sx1 - sdhat Ψ1); (*subtract the average after multiplying by the scale factor and pulse shape*)
 rn[[i]] = rn[[i]] + delta1;
 delta2 = (sx2 - sdhat Ψ2);
 rn[[bign/2 - i + 1]] = rn[[bign/2 - i + 1]] + delta2 ;
 delta3 = (sx3 - sdhat Ψ3);
 rn[[bign/2 + i]] = rn[[bign/2 + i]] + delta3 ;
 delta4 = (sx4 - sdhat Ψ4);
 rn[[bign - i + 1]] = rn[[bign - i + 1]] + delta4;]
PSDnoise = rn;
p = ActualPSDnoise;
phat = PSDnoise;
diff = p - phat;
ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True,
 Frame → True, PlotRange → {Automatic, {0, 9}}, PlotLabel → "PSD Estimated"]
ListPlot[Transpose[{ω, Abs[PSDnoise]}], PlotJoined → True,
 Frame → True, PlotRange → All, PlotLabel → "PSD Estimated"]
ListPlot[Transpose[{ω, Abs[ActualPSDnoise]}], PlotJoined → True,
  Frame → True, PlotRange → All, PlotLabel → "PSD Actual"]
```

6.7 Comparison

The final notebook combines the results from all the previous notebooks and plots them together. This notebook was used to create Figures 4.7 and 4.25.

```
<< Graphics`MultipleListPlot`
rng = Range[-1.5, 1.5, .02];
rng = Take[rng, -(Length[rng] - 1)];
noisescale = Table[10^rng[[i]], {i, 1, Length[rng]}];
aa = 1;
SNR = Table[20 Log[aa / noisescale[[i]]], {i, 1, Length[noisescale]}];
BERrnmin = << BERRNMIN;
MSErnmin = << MSERNMIN;
BERrnest = << BERRNEST;
MSErnest = << MSERNEST;
BERavg = << BERAverage;
MSEavg = << MSEAverage;
BERavgNEW = << BERAverageNew;
MSEavgNEW = << MSEAverageNew;
BERunk = << BERUNK;
MSEunk = << MSEUNK;
one = Transpose[{SNR, BERrnmin}];
two = Transpose[{SNR, BERrnest}];
three = Transpose[{SNR, BERavg}];
four = Transpose[{SNR, BERavgNEW}];
five = Transpose[{SNR, BERunk}];

one1 = Transpose[{SNR, Abs[MSErnmin]}];
two2 = Transpose[{SNR, Abs[MSErnest]}];
three3 = Transpose[{SNR, Abs[MSEavg]}];
four4 = Transpose[{SNR, Abs[MSEavgNEW]}];
five5 = Transpose[{SNR, Abs[MSEunk]}];
SNR = Table[20 Log[aa / noisescale[[i]]], {i, 1, Length[noisescale]}];
MultipleListPlot[one, three, five, PlotJoined → True,
 PlotRange → All, TextStyle → {FontFamily → "Times", FontSize → 12},
 SymbolStyle → {{GrayLevel[.9], Thickness[.01]},
   {GrayLevel[.1], Thickness[.075]}, {GrayLevel[.5],
    Thickness[.05]}}, FrameLabel → {"SNR", "BER"},
 Frame → True, Axes → False, PlotLegend → {"Minimum", "Average", "Unknown"},
 LegendPosition → {.9, -.5}, LegendShadow → None,
 GridLines → Automatic]

MultipleListPlot[one1, two2, three3, four4, five5,
 PlotJoined → True, PlotRange → All, TextStyle → {FontFamily → "Times",
   FontSize → 12}, SymbolStyle → {{GrayLevel[.9], Thickness[.01]},
   {GrayLevel[.2], Thickness[.075], GrayLevel[.7], Thickness[.01]},
   {GrayLevel[.3], Thickness[.075]}, {GrayLevel[.8], Thickness[.05]}},
 FrameLabel → {"SNR -- dB", "MSE"}, Frame → True, Axes → False,
 PlotLegend → {"Min(before)", "Min(after)", "Avg(before)", "Avg(after)", "Unk"},
 LegendPosition → {.4, -.3}, LegendShadow → None]
```

Bibliography

1. Berger, Toby and Donald W. Tufts. "Optimal Pulse Amplitude Modulation Part I: Transmitter Receiver Design and Bounds From Information Theory," *IEEE Transactions on Information Theory*, 196–208 (April 1967).
2. Bernard Widrow, et Al. "Adaptive Noise Cancelling: Principles and Applications," *Proceedings of the IEEE*, 1692–1716 (December 1975).
3. Casella, George and Roger L. Berger. *Statistical Inference (Second Edition)*. Pacific Grove CA: Duxbury, 2002.
4. Dwork, B. M. "Detection of a Pulse Superimposed on Fluctuation Noise," *Proceedings of the IRE*, *38*:771–774 (July 1950).
5. Ericson, Thomas. "Structure of Optimum Receiving Filters in Data Transmission Systems," *IEEE Transactions on Information Theory*, 352–353 (May 1971).
6. Gardner, William A. *Statistical Spectral Analysis: A Nonprobabilistic Theory*. Englewood Cliffs NJ: Prentice Hall, 1988.
7. Gardner, William A. "Exploitation of Spectral Redundancy in Cyclostationary Signals," *IEEE SP Magazine*, 14–36 (April 1991).
8. Gardner, William A. "Signal Interception : A Unifying Theoretical Framework for Feature Detection," *IEEE Transactions on Communications*, 36:897–906 (August 1988).
9. Gardner, William A. "Spectral Correlation of Modulated Signals: Part II-Digital Modulation," *IEEE Transactions on Communications*, COM–35(6):595–601 (June 1987).
10. Gardner, William A. and Chih Kang Chen. "Signal-Selective Time-Difference-of-Arrival Estimation for Passive Location of Man-Made Signal Sources in Highly Corruptive Environments, Part I: Theory and Method," *IEEE Transactions on Signal Processing*, 40(5):1168–1184 (May 1992).
11. Gisselquist, Daniel E. *A Linear Subspace Approach to Bust Communication Signal Processing*. AFIT/DS/ENG/04-02, Graduate School of Management and Engineering, Air Force Institute of Technology (AU), Wight Patterson AFB, OH, 2002.
12. Haykin, Simon. *Adaptive Filter Theory*. Upper Saddle River NJ: Prentice Hall, 1996.
13. Knapp, Charles H. and G. Clifford Carter. "The Generalized Correlation Method for Estimation of Time Delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP–24(4): 320–327 (August 1976).
14. Lucky, R. W. "Techniques for Adaptive Equalization of Digital Communication Systems," *The Bell System Technical Journal*, 45 (April 1966).
15. Mills, R. F. and G. E. Prescott. "A Comparison of Various Radiometer Detection Models," *IEEE Transactions on Aerospace and Electronic Systems*, *32*:467–473 (January 1996).

16. Nyquist, H. "Certain Topics in Telegraph Transmission Theory," *Transactions of the AIEE(Communications and Electronics)*, *47*:617–644 (April, 1928).

17. Oppenheim, Alan V. and Ronald W. Schafer. *Discrete-Time Signal Processing (Second Edition)*. Englewood Cliffs NJ: Prentice Hall, 1999.

18. Priestley, M. B. *Spectral Analysis and Time Series*. San Diego CA: Academic Press, 1981.

19. Roberts, Richard A. and Clifford T. Mullis. *Digital Signal Processing*. Reading MA: Addison-Wesley, 1987.

20. Scharf, Loius L. and Benjamin Friedlander. "Matched Subspace Detectors," *IEEE Transactions on Signal Processing*, 42(8):2146–2157 (August 1994).

21. Scharf, Louis L. *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Reading MA: Addison–Wesley Publishing Company, 1991.

22. Shanmugan. "Spectral Correlation of Modulated Signals: Part I-Analog Modulation," *IEEE Transactions on Communications*, COM–35(6):584–594 (June 1987).

23. Shanmugan, K. Sam and A.M. Breipohl. *Random Signals: Detection, Estimation, and Data Analysis*. New York NY: John Wiley and Sons, 1988.

24. Sklar, Bernard. *Digital Communications: Fundamentals and Applications (2nd Edition)*. Upper Saddle River NJ: Prentice Hall, 2001.

25. Stremler, Ferrel G. *Introduction to Communication Systems (3rd Edition)*. Reading MA: Addison-Wesley Publishing, 1992.

26. Tufts, Donald W. "Nyquist's Problem–The Joint Optimization of Transmitter and Reciever in Pulse Amplitude Modulation," *Proceedings of the IEEE*, 248–259 (March 1965).

27. Turin, George L. "An Introduction to Matched Filters," *IRE Transactions on Information Theory*, 311–329 (June 1960).

28. Urkowitz, Harry. "Energy Detection of Unknown Deterministic Signals," *Proceedings of the IEEE*, *55*(4):523–531 (April 1967).

REPORT DOCUMENTATION PAGE		*Form Approved* *OMB No. 074-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, 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 the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* 21-03-2005	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* August 2003 – March 2005

4. TITLE AND SUBTITLE Noise Estimation in the Presence of BPSK Digital Burst Transmissions	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Bettison, Susan E., Second Lieutenant, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GAM/ENC/05-03
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research explores noise estimation techniques in an attempt to improve upon a previously developed digital burst transmission Binary Phase Shift Keyed (BPSK) demodulator. The demodulator success is dependant on the accuracy of the estimate of Power Spectral Density (PSD) of the unknown noise. Given a discrete time signal transformed into the frequency domain, the research seeks to determine if it is possible to effectively estimate the PSD of the unknown noise. The demodulator was developed using a new signal model for digital burst transmissions based on linear spectral subspace theory. Using this model and the redundancy properties of BPSK digital burst transmissions, five noise estimation techniques will be presented and tested. The success of the methods will be reported in two ways, first, the effect the new noise PSD estimates have on the success of the demodulator and second, a comparison to the actual PSD of the noise.

15. SUBJECT TERMS

signal processing, digital communications, white noise, filters

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lawrence K. Chilton, Ph.D. (AFIT/ENC)
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	109	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4523 (lawrence.chilton@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18