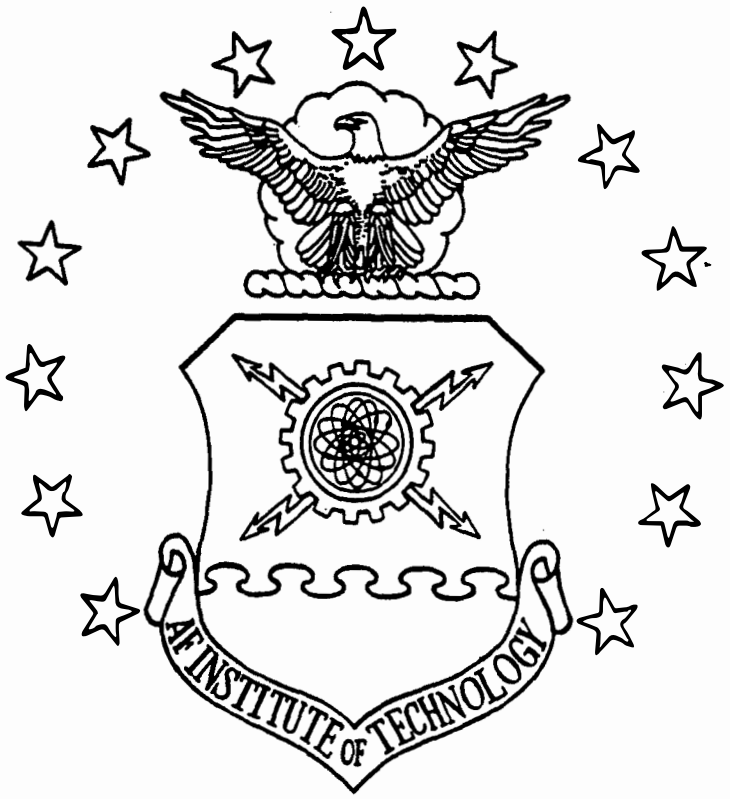
12-1995

Narrowband Interference Suppression in Spread Spectrum Communication Systems

James A. Lascody



Narrowband Interference Supression in Spread Spectrum Communication Systems

THESIS

James A. Lascody
Captain, USAF

AFIT/GE/ENG/95D-12

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/95D-12

Narrowband Interference Supression in Spread Spectrum Communication Systems

THESIS
James A. Lascody
Captain, USAF

AFIT/GE/ENG/95D-12

AFIT/GE/ENG/95D-12

Narrowband Interference Supression in Spread Spectrum Communication Systems

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 Electrical Engineering

James A. Lascody, B.S. Electrical Engineering
Captain, USAF

December 1995

Acknowledgments

I would like to extend my sincerest thanks to my advisor, Captain Joseph Sacchini, and to Major Robert Mills. Their insight and guidance, not to mention their patience, were invaluable during this research effort. They provided support and direction without taking away any freedom to try new ideas. I would also like to thank Dave Doak for his assistance. Without his expertise in troubleshooting C code, parts of this thesis would not be completed. Additionally, I would like to extend my gratitude to Mr. Richard Miller, who worked long and hard on the hardware portion of this thesis. Many thanks to my study and workout partner Captain Jack Pullis, who actually made AFIT fun.

Finally, I would like to thank my wife, Ricki, who provided support and understanding, and put up with my moodiness during the more stressful times. She provided me with the encouragement and motivation to complete this research and to complete the AFIT engineering program.

James A. Lascody

Table of Contents

List of Figures

List of Tables

Abstract

Of significant interest to the United States military is the ability of an enemy to deny or disrupt the operation of the Global Positioning System (GPS). In order to combat this threat the GPS Joint Program Office (JPO) initiated the Tactical GPS Anit-Jam Technology (TGAT) project, which yielded a prototype DEF to remove narrowband interference. A previous thesis effort conducted by Captain Gerry Falen at AFIT validated the performance of the DEF in MATLAB and analyzed the effect of the DEF on the synchronization of the GPS receiver. This research describes the work performed to get the DEF hardware operational and extends the previous research. Comdisco's Signal Processing Worksystem (SPW) was used to correlate this research with Captain Falen's and to examine the effect of the DEF on the probability of bit error. This research uses peak-to-average correlation value, probability of bit error, and percent jammer power removed to examine the performance of the DEF. The peak-to-average correlation value provides insight into the success of acquisition process. Fourteen jamming scenarios are examined using continuous wave, pulsed continuous wave, and broadband noise jammers with the DEF excision threshold set to 6 to 9 dB as recommended by Capt. Falen. In addition, the implementation of the Total Least Squares Prony algorithm is examined as an alternative excision technique.

The behavior of the peak-to-average correlation value in this research was similar to the behavior reported by Capt. Falen (7). An extension of his research was achieved by providing probability of bit error curves. The DEF effectively rejected all of the jammers except the broadband noise jammer. In scenarios other than the broadband noise jammer, the DEF removed over 98% of the jammer power. The best peak-to-average correlation value occurred at a 3 dB threshold, which differed from Capt. Falen's recommended level. The bit error rate curves show that the DEF significantly enhanced the performance of the simulated DS BPSK system in extreme jamming environments. The results presented in this research show that the DEF is a viable, robust option to remove narrowband interference from the GPS signal.

Narrowband Interference Supression in Spread Spectrum Communication Systems

I. Introduction

1.1 Background

Since the advent of the NAVSTAR Global Positioning System (GPS), soldiers, as well as civilians, have come to rely more and more on this system's capability. Countless stories from the Gulf War tell of American soldiers calling stateside seeking GPS receivers. Logsdon (12) cites a story of a group of American soldiers who picked up a new arrival and with blackened headlights in a sand storm drove this soldier back to camp dropping him off a few feet from the flap of his tent. The driver accomplished this remarkable feat by entering the tent's coordinates into the GPS receiver prior to leaving. Stories like this drive home the point that the Global Positioning System is here to stay. Within the near future, engineers will find ways to bring GPS receivers into everyday life. As Logsdon (12) indicates, several Japanese cars already have this technology and American car manufacturers are working on similar systems.

So exactly what is the NAVSTAR Global Positioning System? It is a comprehensive, all-weather, global, three-dimensional satellite navigation system comprising 21 satellites, plus three in-orbit spares (12). These satellites continuously transmit on two frequencies, L1 and L2. L1, 1575.42 MHz, is modulated with two digital codes; coarse acquisition (C/A) and precision (P) code. L2, 1227.6 MHz, is modulated with precision code only. The difference in frequency between the two codes is needed to improve the degree of position accuracy. P code is significantly more accurate than C/A code and is available only to military and other authorized users, whereas the C/A code is free of charge to anyone in the world who owns a GPS receiver. Each satellite has its own unique C/A and P code so that a receiver can tell

which satellite sent the received signal. To obtain lock, the GPS receiver compares an identical time-shifted version of the code against the received signal. At some point, if the received signal is valid, the two signals will match up, or correlate, indicating receiver lock-on.

Of significant interest to the U.S. military is the ability of an enemy to deny or disrupt the GPS signal. In a hostile environment the enemy will attempt to prevent U.S. soldiers from utilizing this system. Since the signal is constructed in such a way as to spread its energy over a large bandwidth, GPS signals have an inherent anti-jam capability. Generally, jammers have a fixed amount of power, and operators must choose the best way to apply this power. The GPS signal is below the natural noise floor because of the spectrum spreading done prior to transmission. The receiver then despreads the signal bringing it up above the noise. With a narrowband jammer present, a large amount of energy is present in a small bandwidth. So, when the signal is despread by the receiver, the energy of the jammer is spread out, significantly reducing its impact on the transmitted signal. Even with this anti-jam capability, Poor and Rusch (16) point out that narrowband interference suppression is necessary to improve error-rate performance and improve acquisition capability.

The GPS Joint Program Office (JPO) awarded Raytheon the Tactical GPS Anti-Jam Technology (TGAT) project to determine the feasibility of reducing the effects of narrowband jamming against the Global Positioning System. Unable to complete the work under the contract, Raytheon delivered the hardware to AFIT by order of the JPO. Initially, this research was intended to complete the construction of Raytheon's low cost narrowband interference suppressor. However, the focus of this research changed after several months of working on the hardware. In June 1995, the JPO was briefed on the status of the hardware. As a result of this briefing, the focus of my research switched to the simulation of Raytheon's suppressor in a software package that allowed for a bit error rate analysis.

1.2 Problem Statement

As stated in the background, narrowband interference of GPS signals is a cause of serious concern to the Department of Defense and to other users. From their work on the

TGAT project, Raytheon developed the Digital Excision Filter (DEF) to improve GPS receiver performance in the presence of these interferers. Since the GPS JPO opted to let the contract expire, Raytheon was unable to finish the prototype development of the DEF. A completed, working DEF will drastically improve narrowband interference suppression in GPS receivers. This thesis will document the work completed on the hardware up to the time of the JPO briefing and analyze the effect of narrowband interference excision on the bit error rate. A previous thesis effort at AFIT validated the DEF performance theoretically by showing that the correlation spike of the receiver was not adversely affected by the DEF (7).

1.3 Assumptions

This research assumes that the received signal is composed of the sum of additive white Gaussian noise, jamming, and signal. It is assumed that the interference suppression will not provide any gain to the received signal, and that the processing time of the suppressor will not affect the clock bias determined by the GPS receiver. Additionally, it is assumed that the size of the frequency bins are not too large or too small to adequately excise the interference. By this, it is meant that of the 256 frequency bins, the jamming signal will be contained in only a small portion of the bins. Finally, synchronization of the direct sequence spread spectrum system is assumed in the simulations through the use of a direct connection and a timing delay between the transmitter and receiver.

1.4 Scope

This research encompassed two distinct projects. The first project dealt with fixing the hardware developed by Raytheon, and the second project focused on conducting a DEF simulation to examine bit error rates. This thesis discusses the DEF simulation results in Chapter 4, and then explains the status of the hardware through June 1995 in Chapter 5. A previous thesis effort, which simulated the DEF in MATLAB, demonstrated the effect the DEF has on the correlation spike of a GPS receiver. In order to perform a bit error rate analysis, Comdisco's Signal Processing Worksystem (SPW) must be used. This thesis compares the

results of the SPW simulations with the results from the MATLAB simulations conducted previously. In addition, probability of bit error values are provided for a variety of jamming scenarios.

1.5 Methodology

To complete the design of the DEF, a deliberate, methodical approach is paramount. Based on discussions with Captain Joe Sacchini and Major Robert Mills, the best approach is to analyze a single section at time. This method is also required because limited documentation existed that described the components and devices used in the test box and the DEF Circuit Card Assembly (CCA). Therefore, the following sections were analyzed and in order to correct detected design flaws:

1. Radio Frequency to Baseband Conversion
2. Analog to Digital Conversion
3. DEF
4. Digital to Analog Conversion
5. Baseband to Radio Frequency Conversion

When the focus of this research changed, SPW was used to implement a DS/BPSK system, with the DEF placed just before the despreading circuitry. Each part of the system was developed modularly. For example, the DEF was constructed by first building a subsystem that vectorized the data and performed an FFT. Then a custom coded block was built to perform the excision, and finally a subsystem was constructed to inverse FFT and serialize the data. Monte Carlo simulations and Semi-Analytical Estimation were used to determine the bit error rate performance of this system under different jamming scenarios.

1.6 Materials and Equipment

All system models and simulations were developed using version 3.0 Signal Processing Worksystem simulation software developed by Comdisco Systems, Inc of Foster City,

California. The simulations ran on Sparc2 and Sparc20 Sun Workstations in the Communications/Radar Laboratory, Room 225, Building 640. MATLAB was used to compare the effect of the DEF on the correlation spike.

1.7 Thesis Organization

Chapter II of this thesis presents a basic introduction into the concept of Direct Sequence Spread Spectrum (DS-SS) communications. It discusses the theory behind the spreading of a signal, the psuedo-noise (PN) spreading codes used, and the interference suppression benefits of DS systems. Chapter III provides an introduction to communication system simulation and discusses the systems and simulations used through-out this research. Chapter IV presents the SPW simulation results. Chapter V chronologically describes the work performed on the DEF hardware through June 1995, and Chapter VI summarizes the thesis and provides recommendations for future research. It should be noted that Sections 2.1 through 2.3 and 3.1 through 3.3 were done in collaboration with Lt Madden (24) since both of our thesis efforts utilized the techniques described in these sections.

II. Spread Spectrum Theory

2.1 Introduction

This chapter provides an overview of spread spectrum communications theory. Specifically, direct sequence spread spectrum systems are discussed including an overview of spreading codes. Then, a review of current literature on interference suppression is presented. Finally, the performance of DS-BPSK system against CW jammers is examined.

2.2 Spread Spectrum Communications

In all communications systems, the modulated waveform occupies a frequency bandwidth that is dependent upon the modulation method used and the data being sent. In a spread spectrum system, the transmitted bandwidth of the signal has been "spread" over a larger bandwidth than the original modulated bandwidth (6). This transforms the power spectral density into a more uniform spectrum much like that of noise (1). To qualify as spread spectrum, a system must satisfy the following three conditions (20):

1. The signal occupies a bandwidth much in excess of the minimum bandwidth required by the information.
2. Spreading is accomplished through the use of a code signal independent of the data itself.
3. At the receiver, despreading is accomplished by the correlation of the received signal with a replica of the spreading code used at the transmitter.

Spread spectrum (SS) techniques, due to their anti-jam and low probability of intercept characteristics, were initially developed for use in military guidance and communications systems. These technologies were eventually integrated into areas such as energy density reduction, high-resolution ranging, and multiple access systems (20). Currently, because of their ability to reject undesired interference and their low power spectral densities, SS techniques are being investigated for integration into commercial wireless communications systems and cellular telephony.

The three general spread spectrum signaling methods are (6):

1. Modulation of a carrier by a digital code sequence whose bit rate is much higher than the information bandwidth. This is known as direct sequence (DS).
2. Carrier frequency shifting in discrete increments dictated by a code sequence. This is called frequency hopping (FH).
3. Pulsed-FM modulation in which a carrier is swept over a wide band during a given interval.

There are also hybrid SS techniques which incorporate a mixture of the above techniques. In this research, only direct sequence will be addressed.

2.2.1 Direct Sequence Spread Spectrum. In direct sequence systems, a psuedo-random code is used to modify the carrier phase prior to transmission of the signal. The code sequence has a much higher rate than the original digital data which greatly expands the bandwidth beyond that of the original information bandwidth (1). After reception of the transmitted signal, the spread spectrum receiver must perform two functions beyond that of a conventional receiver. First, the receiver must learn or reproduce the code used by the modulator. This process is known as code acquisition. The acquired code must then be synchronized with the transmitter code. This process is known collectively as synchronization. Second, the receiver must remove the spreading of the carrier bandwidth to obtain the original message bandwidth (15). This process is known as carrier despreading and is accomplished through multiplication of the received signal with a correlated replica of the original spreading code. This reduces the message signal back to its original bandwidth, while spreading any interference that might be present.

In a typical direct sequence system as shown in Figure 1, a data signal $d(t)$ is used to modulate a carrier signal $x(t)$ represented by

$$x(t) = \sqrt{2P}\cos(\omega_o t) \tag{1}$$

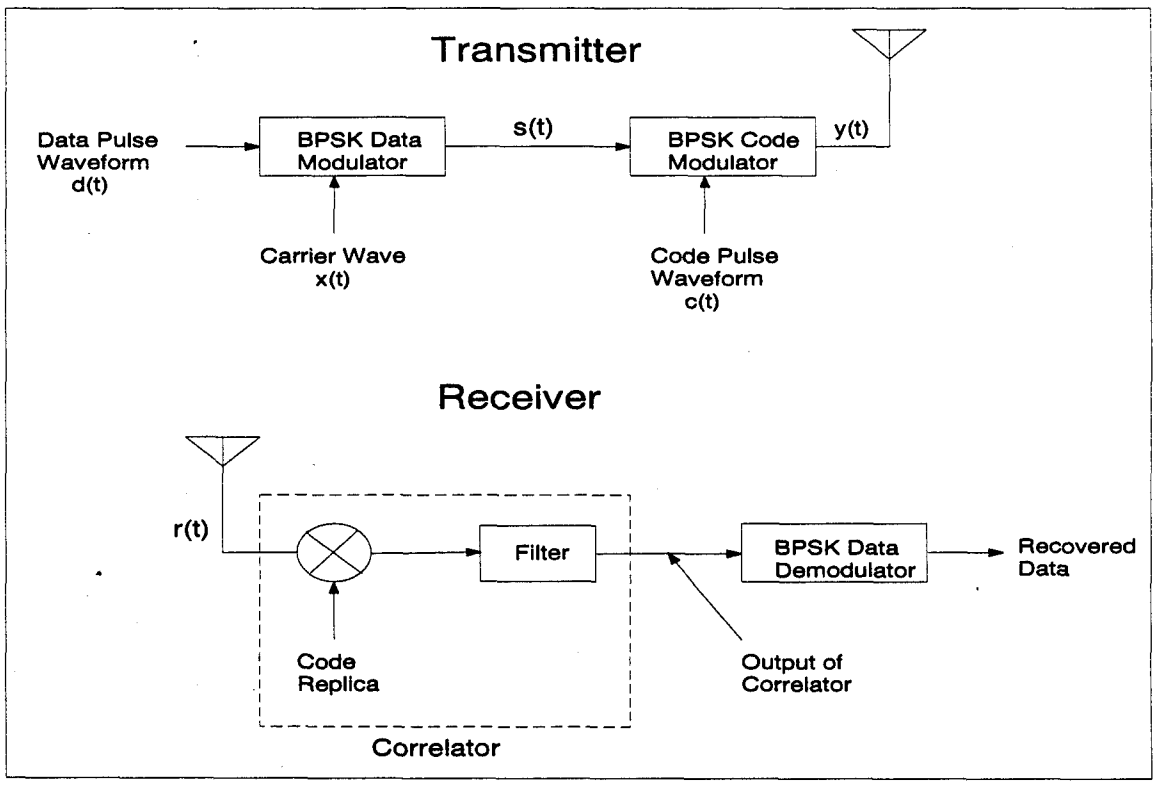


Figure 1. Direct Sequence Spread Spectrum Transmitter and Receiver

where ω_o is the radian frequency and P is the carrier power. The modulated waveform is then given by

$$s(t) = \sqrt{2P}\cos\left(\omega_o t + \theta_x(t)\right) \tag{2}$$

where θ_x is the data phase modulation. This signal is then modulated by a spreading code $c(t)$. The resulting waveform is

$$y(t) = \sqrt{2P}\cos\left(\omega_o t + \theta_x(t) + \theta_c(t)\right) \tag{3}$$

where $\theta_c(t)$ is the phase modulation due to the spreading code. For BPSK modulation with antipodal data and spreading signals the expression for $y(t)$ can be reduced to

$$y(t) = \sqrt{2P}\, x(t)\, c(t)\cos\left(\omega_o t\right). \tag{4}$$

At the receiver, the received signal $r(t)$ is represented by

$$\begin{aligned} r(t) &= y(t - T_d) && (5)\\ &= A\sqrt{2P}\, d(t - T_d)\, c(t - T_d)\, \cos\left(\omega_o(t - T_d) + \phi\right) && (6) \end{aligned}$$

where A is the system gain parameter, ϕ is a random phase in the range $(0, 2\pi)$, and T_d is the propagation delay which is dependent upon the distance between the transmitter and receiver. This signal is multiplied by the original spreading code that has been delayed by $\hat{T}_d$ which is the receiver estimate of the propagation delay T_d. When the code at the receiver is correlated with the code at the transmitter, $T_d = \hat{T}_d$ and the output of the correlator will be the unspread signal $s(t)$ with a phase difference and a delay. This despread signal is then input to a conventional BPSK demodulator (20).

There are two main criteria for measuring the effectiveness of this process. The first criterion, the process gain, compares the input and output signal-to-noise ratios (SNR) and is expressed by:

$$G_{p,dB} = SNR_{out,dB} - SNR_{in,dB} \tag{7}$$

where SNR_{out} is the output signal power to noise power ratio of a given receiver with an input signal to noise ratio SNR_{in} with both quantities in dB's. For example, if SNR_{in} for an SS receiver is $-10\ dB$ and the SNR_{out} is 15 db, the processing gain would be 25 dB. The processing gain can also be expressed by:

$$G_p = \frac{BW_{RF}}{R_{info}} \tag{8}$$

where BW_{RF} is the bandwidth of the transmitted spread spectrum signal, and R_{info} is the data rate of the baseband (unmodulated) data signal (6). For example, if a $1.544\ Mbps$ signal is spread to a bandwidth of 20 MHz, the process gain is 12.95 or 11 dB (1).

The second criterion, the jamming margin, measures the system's performance capability in a hostile environment and is expressed by:

$$M_{j,dB} = G_{p,dB} - [L_{sys,dB} + SNR_{out,dB}] \tag{9}$$

where L_{sys} is the system implementation loss and SNR_{out} is the minimum output signal to noise ratio required to maintain a minimum bit error rate with all quantities in dB's (6). The jamming margin is essentially the process gain with an allowance for implementation losses and a minimum acceptable output SNR (1). The jamming margin is the maximum additional interference power which a system can encounter and still be expected to operate.

2.2.2 Spreading Codes. There are two methods that can be employed by a spread spectrum system to obtain the despreading code at the receiver. These are the transmitted reference and the stored reference. In the first of these, the spreading code is simultaneously transmitted with the data. This method is impractical for secure applications and is not commonly used in military applications. The second and more commonly encountered approach is the stored reference. In this method, the code must either be stored or generated by the receiver. When using the transmitted reference, it is possible to use a completely random sequence; however, for the stored reference, a deterministic sequence must be used.

The codes used for this purpose are known as pseudo-random or psuedo-noise (PN) codes. These codes are a class of linear code sequences that despite being deterministic have statistical properties much like that of uniform noise. These codes must satisfy the following properties (20):

1. Balance - Good balance requires that in each period of the sequence, the number of binary ones differs from the number of binary zeros by at most one digit.
2. Run - A run is defined as a sequence of a single type of binary digit(s). The appearance of the alternate digit in a sequence starts a new run. The length of the run is the number of the digits in the run. It is desirable that approximately 2^{-n} of the runs be of length n.
3. Correlation - If the sequence is compared term by term with a cyclic shift of itself, the number of agreements should differ from the number of disagreements by not more than one count.

Codes satisfying these properties will allow for optimum spreading to occur while minimizing the interference between users. The most common method of generating these codes is to use shift registers with feedback taps going to a modulo two adder as shown in Figure 2. The length of the code, as well as its ability to meet the above criteria, is determined by the number of shift registers and the feedback taps used (6, 20). The longest code that can be generated by a shift register of length N is $2^N - 1$.

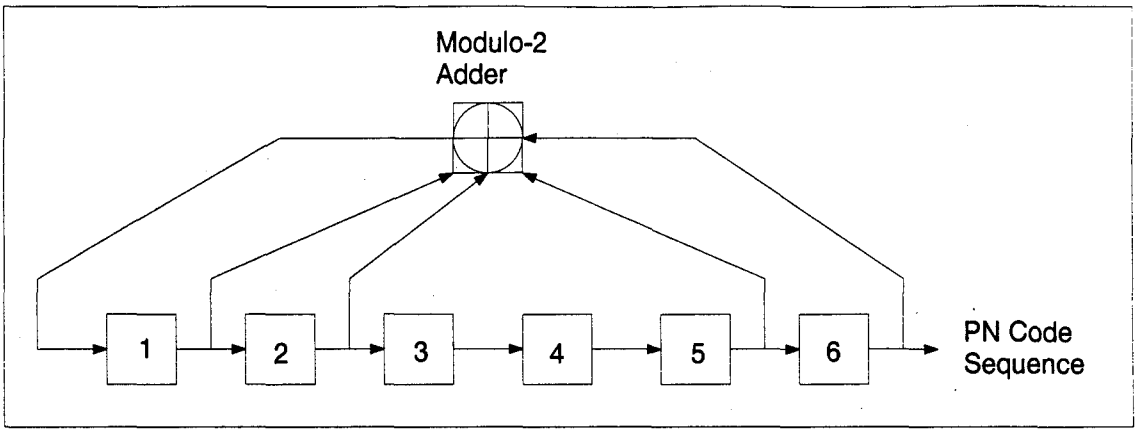


Figure 2. Linear Shift Register PN Code Sequence Generator

"Good" PN codes have high auto-correlation peaks at zero shift. These peaks help to minimize false synchronization (1). A typical PN auto-correlation function is shown in Figure 3. The auto-correlation function for a code of length $p = 2^{N-1}$ is defined as (20)

$$R_x(\tau) = \frac{1}{p}\left[\begin{array}{c} number\ of\ agreements\ less\ number\ of \\ disagreements\ in\ a\ comparison\ of\ one\ full \\ period\ of\ the\ sequence\ with\ a\ \tau\ position \\ cyclic\ shift\ of\ the\ sequence \end{array}\right]. \tag{10}$$

In addition, Gold Codes were developed which have low cross-correlation with other codes. This low cross-correlation minimizes interference between users and allows for effective Code Division Multiple Access (CDMA) communication. For the simulations in this research a maximal length PN code was used for the BER estimation. This was because SPW did not allow for more than one sequence for a given register length. Therefore, choosing preferred pairs of PN sequences was not an option. For the peak to average correlation value, which was used to correlate with Capt. Falen's research (7), 102,400 bits of the p-code obtained by using his MATLAB function, *zplatinum*, were used.

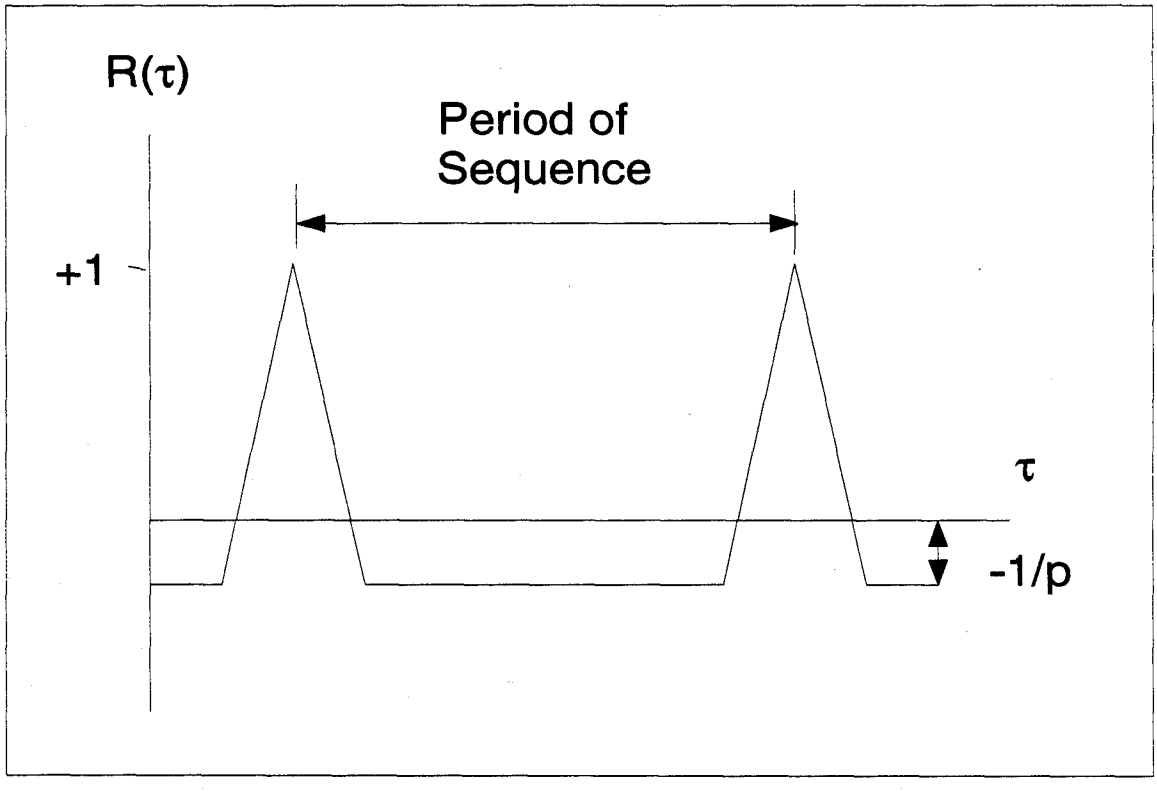


Figure 3. Typical PN Auto-Correlation Function

2.3 *Interference Suppression in Spread Spectrum System*

2.3.1 Literature Review. Since the early eighties, a significant body of work has been concerned with the development and implementation of narrowband interference suppression systems (10). The two main categories this research has developed are time-domain adaptive filter suppressors and transform domain excisers (18). Either of these units can be placed before the despreading circuitry of the spread spectrum receiver in order to remove some of the interferer's energy. While these devices will acceptably remove a significant portion of the interference, they will introduce a small amount of distortion in the received signal. As Saulnier (18) indicates, it is necessary to employ one of these techniques in order to improve the performance of the direct sequence spread spectrum (DS SS) communication system without increasing the system bandwidth. Until a few years ago, linear signal processing techniques represented the largest part of narrowband interference suppression research; however, recent developments in digital signal processing have enabled engineers to concentrate on transform domain techniques as well. The following two subsections discuss these two techniques.

2.3.2 Time-Domain Adaptive Filter Suppressors. There are numerous ways to implement adaptive filters to accomplish narrowband interference suppression. One of the most common ways is to use an estimator/subtracter method (16). The basic operation of this type of implementation involves subtracting a replica of the interference from the received signal. More simply stated, the received signal undergoes a whitening process—the wideband components of the signal are enhanced (16). A significant problem encountered during the implementation of these systems is the formation of the replica. This can be overcome, as Cooper and McGillem (5) point out, by exploiting the predictability of the interference and the spread spectrum signal. We know that the spread spectrum signal will be almost flat. This makes the signal unpredictable unless there is knowledge of the psuedonoise sequence used to

spread the original signal. Since the interferer is narrowband, a large portion of its energy will be concentrated in a small bandwidth. Thus, the interferer can be accurately predicted on the time scale of the received signal. Poor and Rusch point out that by subtracting these values of the received signal from the predicted values formed this way, the effect of the interferer can be significantly reduced (16).

2.3.3 Transform Domain Excisers. The main approach in transform based suppression is to take the Fourier transform of the received signal, apply some type of signal processing algorithm, and then inverse Fourier transform the signal back to the time domain. The resulting signal is then passed to the receiver for correlation with the spreading signal. The area where implementations differ is in the type of signal processing algorithm used to remove or mask the interference. One method used is to excise those frequency components whose energy exceeds a given threshold (14). Another way is to apply a whitening filter derived from applying a nonparametric spectrum estimator to the received signal (16). The significant difficulty encountered in these techniques involves achieving the desired speed so as to minimize the distortion of the spread spectrum signal. The transform domain techniques use similar methods, such as adaptive filtering, as the time domain methods. The difference is simply that the excision is done in the frequency domain.

Although these two techniques accomplish the same result, the performance of each is drastically different. The time-domain techniques work extremely well if the interference bandwidth is small, but when the interference bandwidth exceeds ten percent of the spreading bandwidth, the system becomes useless (10). The transform domain technique works effectively at all interference bandwidths, but degrades to useless when the jammer power to signal power ratio is large (10). Transform domain techniques are also more robust since they can handle multiple narrowband jammers simultaneously. Current research, typified by the work

of Tazebay and Akansu (23), utilizes the benefits of both techniques by analyzing the type of interference and applying the technique which will work the best.

2.3.4 DS BPSK Performance vs CW Jammers. The objective of a jammer is to use a signal source to block the intended receiver from obtaining the transmitted data. This can be done through the following types of jamming signals:

- Continuous Wave
- Swept Tone
- Broadband
- Narrowband
- Pulsed Tone

When a jammer is used against a narrow-band signal, the most common jamming method is to produce a high power tone centered at the carrier frequency of the narrow-band signal. When a jammer encounters a spread spectrum signal, it must decide whether to concentrate a high power signal over a relatively narrow portion of the signal, as shown in Figure 4, or to spread its power over a larger bandwidth. Regardless of the technique chosen, the effect of the interference will be decreased during the despreading process. When the received signal is multiplied by the spreading code, the jamming signal will be spread in the same manner as the data signal was spread before transmission. Thus, the jammer signal will have a lower power density over a given bandwidth and will be less effective.

In order to obtain analytical insight into the effect of spread spectrum upon jamming signals (13), consider a single tone jammer represented by

$$j(t) = \sqrt{2P_j}\cos(\omega_c t + \theta) \tag{11}$$

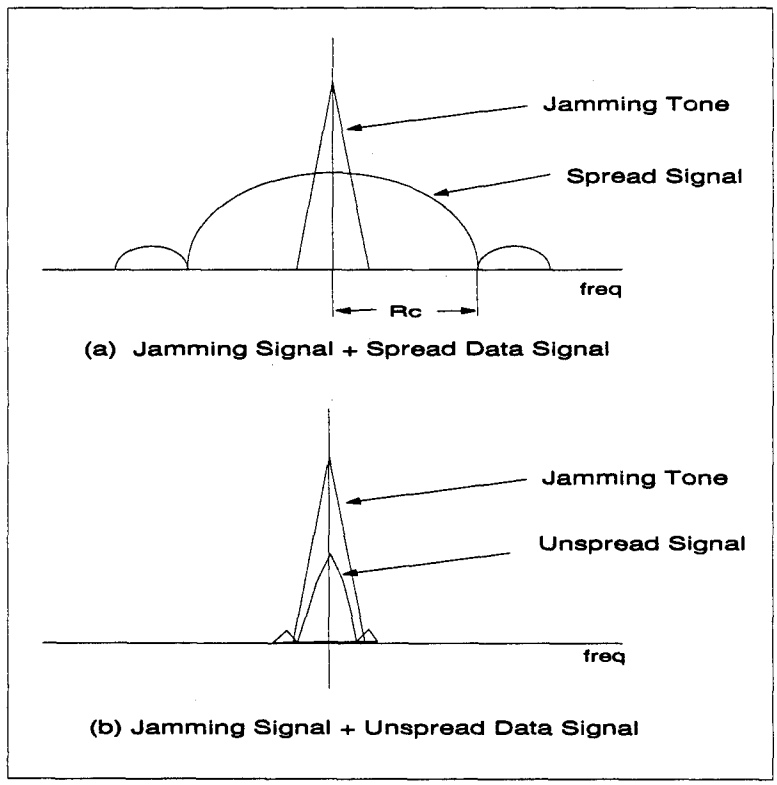


Figure 4. Example of Jammer Effects upon Spread and Unspread Signals

where P_j is the jammer power, and we are assuming that the jammer power is much greater than the noise power such that $P_J + P_N \approx P_J$. The received signal can then be written as

$$r(t) = \sqrt{2P_s}\, d(t)\, c(t)\, \cos(\omega_c t) + \sqrt{2P_j}\, \cos(\omega_c t + \theta) \tag{12}$$

where P_s is the signal power. After $r(t)$ is multiplied by the replica of the code sequence and is carrier demodulated, the resultant signal can be written as

$$\begin{aligned} r(t) &= \sqrt{P_s}\, d(t)\, [1 + \cos(2\omega_c t)] + \sqrt{P_j}\, c(t)\, [1 + \cos(2\omega_c t)] \cos(\theta) \\ &\quad - \sqrt{P_s P_j}\, \sin(2\omega_c t) \sin(\theta). \end{aligned} \tag{13}$$

The signal is then filtered to remove double frequency terms and the output of the correlator in Figure 1 is

$$r_o(t) = P_s\, d(t) + \sqrt{P_s P_j}\, c(t)\, \cos(\theta). \tag{14}$$

From (22) we can then write the PSD of the interference as

$$S_{jj} = \frac{P_j \overline{\cos^2\theta}}{2R_c}\, sinc^2\left(\frac{f}{R_c}\right). \tag{15}$$

After this is integrated over the period T_b, which has an effect equivalent to a lowpass filter with a cutoff frequency of f_b, we get approximately

$$S_{jj} = \frac{P_j \overline{\cos^2\theta}}{2f_c}. \tag{16}$$

Then, letting $N_o \cong S_{jj}$ in the equation for P_e for coherent BPSK, we get

$$P_e = Q\left[\sqrt{\frac{2E_b}{N_o}}\right] \tag{17}$$

$$= Q\left[\sqrt{\frac{2f_cE_b}{P_j\overline{\cos^2\theta}}}\right]. \tag{18}$$

Next, making use of the expressions $E_b = P_sT_b$ and $f_c = \frac{1}{T_c}$ along with the fact that $\overline{\cos^2\theta} = \frac{1}{2}$ for the range $[0, 2\pi]$ we can write

$$P_e = Q\left[\sqrt{\frac{4P_sG_p}{P_j}}\right] \tag{19}$$

where $G_p = \frac{T_b}{T_c}$.

Through the same type of analysis, it can be shown that a worst case approximation for multiple jammers is given by

$$P_e = Q\left[\sqrt{\frac{4P_sG_p}{\sum_{i=1}^{n}P_n}}\right]. \tag{20}$$

This approximation assumes that the jammers are uncorrelated, and that the difference between the carrier frequency and the jammer frequency is small compared to the carrier frequency itself. For the simulations in this thesis, the latter assumption was met by keeping the carrier frequency much smaller than the simulation bandwidth.

2.4 Summary

This chapter provided an overview of direct sequence spread spectrum communications. Additionally, current literature was reviewed that discussed interference suppression methods. Finally, the performance of DS-BPSK systems against CW jammers was examined.

III. Simulation of Communication Systems and SPW Blocks

3.1 Introduction

Over the past decade, the role of simulation has changed drastically. In the past, simulation was used to simply check system functionality and evaluate performance characteristics. Now, simulation is fully integrated into the design process, making it easier to move from a top level simulation toward the implementation of the system (19). Engineers typically have access to extremely powerful computers, which has helped to drive the interest in the simulation area.

Whether one is performing a mathematical calculation or running a computer simulation, the basis for each starts with a system model. This model typically takes the form of a block diagram that shows the subsystem connections (19). All blocks in the system are defined by a signal processing algorithm which describes the relationship between the input and the output. Although calculations and simulation strive to determine performance parameters of the system, simulations differ from calculations in that the simulations estimate the results whereas the other is a direct calculation. Thus, the result for a calculation is usually a number, but the simulation result is a random variable. Hence, the simulations must be repeated for a large number of trials in order to obtain confidence in the results. Shanmugan (19) is quick to point out that simulation should never be a complete substitute for mathematical analysis. Instead, simulation and analysis should be used together in the development of the system. Also, it is important to note that a theoretical analysis is sometimes impossible or extremely difficult.

This chapter provides an overview of simulation theory and signal representation in the simulation environment. Next, it discusses two techniques for evaluating communication

system performance. Then, the system models used within SPW are presented and discussed in detail.

3.2 Signal Representation

Two major problems are present when simulation is used. First, analog waveforms of the actual system must be represented using discrete-time samples. So, the waveforms must be sampled in a way that not only reduces the effects of aliasing, but also does not oversample the waveform. Oversampling the waveform will cause the simulation to have a large runtime. Second, the analog filters of the actual system must be mapped correctly to digital filters. For a given communication system, lowpass and bandpass signals are present. The lowpass signals are the information bearing waveforms and the bandpass signals are the modulated waveforms, such as the transmitter output and receiver input. As stated earlier, these waveforms must be sampled to obtain the discrete time sequences. The sampling theorem states:

> A bandlimited signal $x(t)$ with $X(f) = 0$ for $|f| \geq f_M$ is uniquely determined by its sample values $x(nT_s)$ at a sequence of equidistant points $t = nT_s$ if $f_s > 2f_M$, where $f_s = 1/T_s$. The sampling frequency $f_s = 2f_M$ is known as the Nyquist rate (11).

A number of other factors influence the choice of sampling frequency and are as follows:

1. Aliasing errors
2. Frequency warping in digital filters
3. Nonlinearities
4. Computational constraints.

The computational constraints can be the most demanding factor in the choice of sampling frequency. This is even more true for direct sequence spread spectrum systems with an extremely fast chipping sequence. Thus, the goal is to minimize the sampling rate as much

as possible. To do this, we wish to use signals with low pass spectra (19). Low pass signals are then sampled directly with the appropriate sampling frequency. Bandpass signals also can be directly sampled but the number of samples can be greatly reduced. By using the complex envelope, bandpass signals and modulated signals can be represented as complex low pass signals. A modulated signal with carrier frequency f_c can be written as

$$x(t) = R(t)cos[2\pi f_c t + \phi(t)] \tag{21}$$

where $R(t)$ represents the real envelope of $x(t)$ and $\phi(t)$ represents the phase deviation (19). Using phasors, this equation can be written as

$$x(t) = Re\{R(t)e^{j2\pi f_c t}e^{j\phi(t)}\}. \tag{22}$$

The complex envelope of this signal is then given by

$$\tilde{x}(t) = R(t)e^{j\phi(t)}. \tag{23}$$

The complex envelope of the real signal $x(t)$ is slowly varying with respect to the carrier frequency. However, by using the complex envelope, the sampling frequency can be significantly lower since the bandwidth of the bandpass signal is usually much smaller than the carrier frequency. This yields a smaller number of samples for a specific time period of $x(t)$. The complex envelope can then be expressed in rectangular form as

$$\tilde{x}(t) = x_i(t) + jx_q(t) \tag{24}$$

where $x_i(t)$ and $x_q(t)$ are the in-phase (real) and quadrature (imaginary) components, respectively (19).

3.3 Evaluation Techniques

As stated previously, the goal of the simulation is to estimate the performance characteristics of the communication system. The single most important feature of the digital communication system is the bit error rate (BER) (11, 19). For our simulations the BER describes the fractional number of errors in a transmitted sequence . Jeruchim and others (11) describe and examine five simulation-based approaches for estimating BER.

1. Monte Carlo (MC) simulation
2. Importance Sampling
3. Extreme-value theory
4. Tail extrapolation
5. Quasi-Analytical

The simulations in this research used the first and fifth techniques, which will be described in the next section.

3.3.1 Monte Carlo Simulation. Monte Carlo simulation is the brute force method for estimating BER. The simulation processes a certain number of symbols and the BER is calculated by dividing the number of errors by the total number of symbols. Generally, this will give a sample BER that is consistent and unbiased (19). One of the nuances associated with this technique involves synchronization. To compare the original data to the demodulated data, the processing delay must be known so that the two data streams are aligned in time. Otherwise, excessive errors will result not because of noise, but because the data sequences are not aligned. Advantages of this technique include the fact that it works for any kind of system and that the signals generated by the simulation can be almost identical to the actual signals in the system. The biggest detractor, especially when examining low bit rate DS SS systems, is the computational time. As the run-time approaches infinity, the BER estimate

becomes more consistent and unbiased (19). A rule of thumb for determining the number of iterations is to set the number of iterations so that 50 to 100 errors are counted. For low data rate, high processing gain DS SS systems, this correlates to a very long runtime and a very bored student. For instance if the required probability of bit error was 10^{-3}, the sampling frequency was 10 kHz, and a data rate of 1 bps, then the number of iterations required would be 500×10^6.

3.3.2 Quasi-Analytical Estimation. Quasi-Analytical, or Semi-Analytical (SA), estimation can reduce the simulation time drastically, but places more burden on the analyst (19). This technique combines simulation and analysis. To invoke this method, a noiseless waveform is generated at the transmitter and goes through the channel and receiver front end, where it is demodulated. Then, noise is added to determine how the decision process is affected. It is obvious that the time reduction is there since we do not have to wait for errors to occur. For low bit rate, high processing gain DS SS systems, this technique allows the user to conduct many studies in comparison to the MC method. Jeruchim and others (11) point out that the semi-analytic technique should be implemented in any simulation for the following reasons:

1. In linear channels, it can provide the correct answer rapidly.
2. It can be used as a check for the MC simulations under the same conditions.
3. The technique is easy to implement.

The only "catch" or disadvantage to this technique is that the BER must be calculable conditioned on the transmitted data pattern (19). Shanmugan also clarifies this by stating that to be able to use the semi-analytic estimation technique, the noise must be stationary, additive and Gaussian; the noise and data must be uncorrelated; and there must be no nonlinearities after the insertion of the noise (19). For the simulations in this research, these conditions were met.

In several test cases, the semi-analytic technique yielded BERs extremely close to those of the MC technique. Again a major advantage with this technique is the reduction in the number of iterations. If 100 bits are used for the SA estimator, then the number of iterations required is reduced to 1×10^6. So, under the same conditions used to calculate the number of iterations for the MC technique, the SA technique reduces the number of iterations by a factor of 500, which translates to a tremendous time savings.

3.4 SPW Blocks

3.4.1 Introduction. Comdisco's Signal Processing Worksystem (SPW) is software package for developing, simulating, debugging, and evaluating digital signal processing and communication system algorithms (4). Using the blocks provided in the libraries, and blocks generated by the user, any digital signal processing (DSP) or communication system can be built, and simulated. In addition, SPW allows the designer to generate a hardware description of the system. The following sections provide the reader with insight as to how SPW functions and is not intended as a tutorial. For additional information, the SPW manuals should be referenced.

The Block Diagram Editor (BDE) is the heart of the SPW system. The BDE is where the blocks of the system are laid out and interconnected. A system is built as a hierarchy of block diagrams that specify the signal flow through the simulation. By using the hierarchical design, a single block at one level is a complete system on its own at a lower level. The Signal Calculator is the tool that allows you to create, display, edit, and analyze any type of waveform. The Signal Calculator allows different data types, such as real and complex, scalar and vector to name a few. With this tool many processing options are available such as FFTs, auto- and cross- correlations, histograms, eye diagrams, etc.

3.4.2 Custom Coded Blocks. Although SPW comes with a large number of blocks, it is sometimes necessary to come up with a unique block that is not provided and that cannot be built easily from the provided blocks. In this research, a function block was needed that performed the excision process of the DEF. SPW allows the user to develop custom-coded blocks (CCBs) and the remainder of this section describes the process to create a CCB. The process can be divided into eight subprocesses, which have been taken from the SPW User's Guide.

Create and Save Symbol. Using the BDE, a symbol is created by drawing and labeling the appropriate inputs and outputs or by modifying an existing symbol and saving it under a new name. The symbol can take any shape by using boxes, lines, ports, etc. available in the BDE. The ports are necessary to pass data in and out of the block. The names of the ports must match the names in the source code for the block. When the symbol is completed, the File - Save As command is selected and a file name of the form library/function.symbol, such as lascody/excisor.symbol, is then entered.

Create and Save Parameter File. The easiest way to create a parameter screen is to open an existing one and save it under a new name. However, it is not difficult to create one from scratch. Using lines, boxes, text input, and the add parameter command, all parameters are input. Examples of parameters for the excisor are FFT size, threshold, and default vector length. This file should then be saved using the File - Save As command and entering a filename in the same format except with the params extension, such as lascody/excisor.params.

Link Symbol and Parameter Files. In order for the simulator to know the types and values of the parameters, the symbol file and the parameter screen must be linked. Execute the File - Link command in the BDE. Another window will appear and ask for the *FROM*

file and the *TO* file. In general, you should link from the .symbol file to the .params file using the same names used in the previous steps.

Generate Source Code and Template Files. In the BDE window execute the Tools - Simulator -> Prepare Block command. In the new Prepare Block window, execute the Actions - Template command. This will generate a ".c" and ".h" file for your function. The newly created ".c" and ".h" files are ready to be edited by clicking on the BLOCK.C or BLOCK.H button in the command palette.

Insert Code. The BLOCK.H file contains all the C code structure definitions that the simulator needs. This file should never need editing unless a need arises to include additional files, add to the header, or add to the State Structure. The BLOCK.C file contains all the procedure call declarations and will be edited many times. For the excisor function used in SPW, only the Run Output function of the code needed to be edited. However, the initialize, Run Input, Termination, Link Options, Include Dirs can be edited as needed.

Compile the Code After editing the code, it must be compiled. Execute the Actions - Compile command in the palette to create the object file. If errors occur, make the appropriate changes in the ".c" and ".h" files and recompile until successful. Routine syntax errors will be listed in the Prepare Block Window message area. Other errors will be listed in a file that can be accessed by executing the File - Open - Errors command.

Add New Block to Blocklist. Before the simulator can use the new block, it must be added to its pool of available blocks. The normal blocklist filename is spw_all and can be found in the /comdisco/spwsys/blocklists/ directory. This file can be edited with any text editor. Simply enter the new block information exactly like the other blocks in the file: lascody excisor symbol. Save the new blocklist. This file can be reduced thereby reducing the amount of time to link the new block, which is recommended as long as knowledge exists of which blocks are currently used and which blocks will be used in

later simulations. Each time the source code is edited the block must be recompiled and linked so that the changes will take effect.

Add New Block to Simulation Kernel. Execute the Actions - Link command in the palette. The new CCB is now ready for use in the simulation.

3.5 System Models

3.5.1 Transmitter. As Figure 5 shows, the transmitter takes in random data and inputs it to a BPSK Modulator. The modulator allows you to scale the data to a high and low number, such as 0.0316 and -0.0316. The resulting signal is then multiplied by a PN sequence that has a bit rate 5000 times the data rate, yielding a processing gain of 5000. The values listed in the figure were used during testing of the transmitter and are not the values used in the actual simulation. The actual parameters for the simulation were $f_s = 10000$, and $PG = 5000$. The complex multiplier could also be used to scale the transmitted output, if desired.

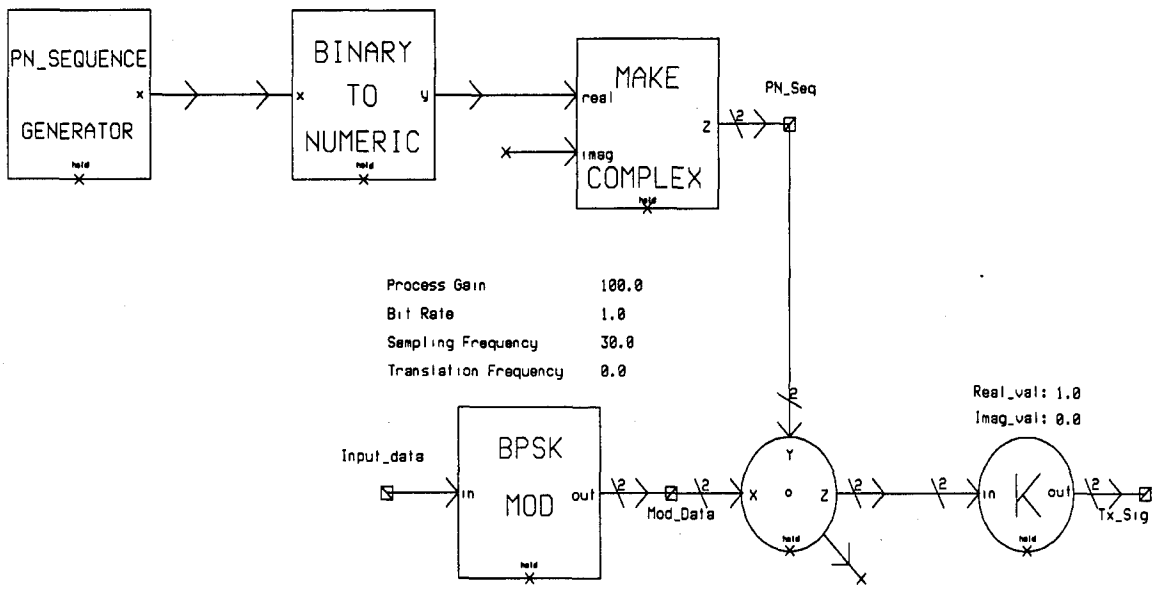


Figure 5. Direct Sequence Spread Spectrum Transmitter

3.5.2 Receiver. Figure 6 illustrates the design of the receiver portion of the system. The PN_Seq input to the despreader, shown in Figure 7, is delayed and conjugated to account for the DEF processing time and the fact that the signal is complex. This signal is then multiplied by the received signal, which is made up of AWGN, narrowband interference, and the transmitted signal. The despreader completely reverses the spreading accomplished in the transmitter. The despread signal is passed to the Matched Filter Demodulator, which detects the bits. For Monte Carlo simulations the PSK error counter simply performed a bit by bit comparison between the delayed original data stream and the output of the detector. The delay is necessary to account for the timing delay caused by the DEF and the integration time of the detector.

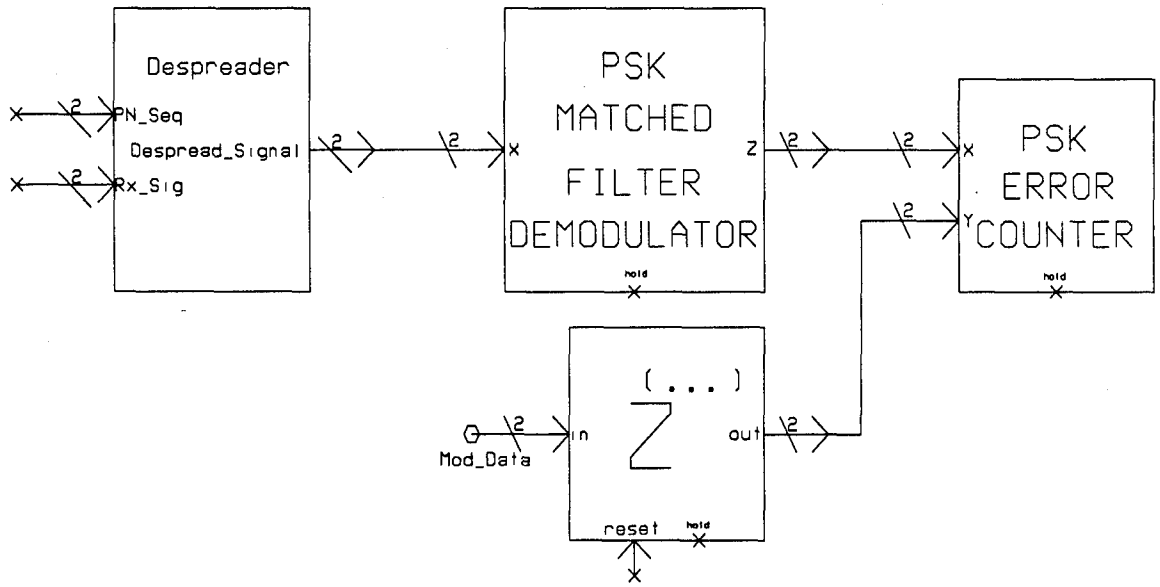


Figure 6. Direct Sequence Spread Spectrum Receiver

3.5.3 Digital Excision Filter. As stated previously, the DEF is composed of three subsystems. After splitting the signal into its real and imaginary parts, the first subsystem, shown in Figure 8, converts the incoming signal into 256 point vectors using circular buffers and performs a 256 point FFT on the data. Within the FFT block, there is an option to

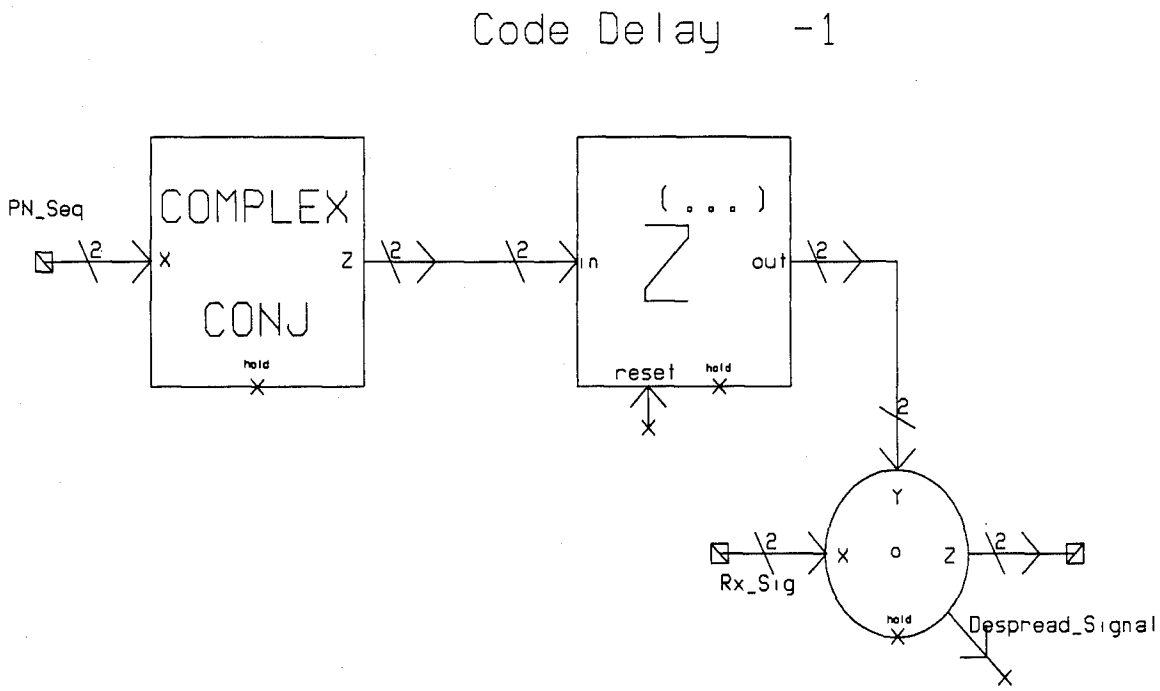


Figure 7. Despreading Block

window the sequence with either a rectangular, Hamming, Bartlett, Blackman, or Hanning window. For the baseline BER estimation with jammers present and the peak to average correlation (PAR) calculations, a Hanning window was implemented because it provided the best frequency localization of the jammer's spectrum due to the sidelobe suppression. For the baseline BER (no jamming), a rectangular window was used. The modulo N impulse train provides timing pulses to the buffers and the FFT so that vectors are output at the correct instant and that the FFT only processes when a valid vector has been output.

The second subsystem, shown in Figure 9, is a custom coded block that performs the excision process on the transformed real and imaginary vectors. The excisor compares the magnitude of the FFT with the threshold and excises every frequency bin that exceeds the threshold. In addition the threshold value for each vector is written to a results file. This value represents the average of the transformed signal plus the threshold offset. The research by Captain Falen showed that this value should be set to 6 to 9 dB (7). For example, if the

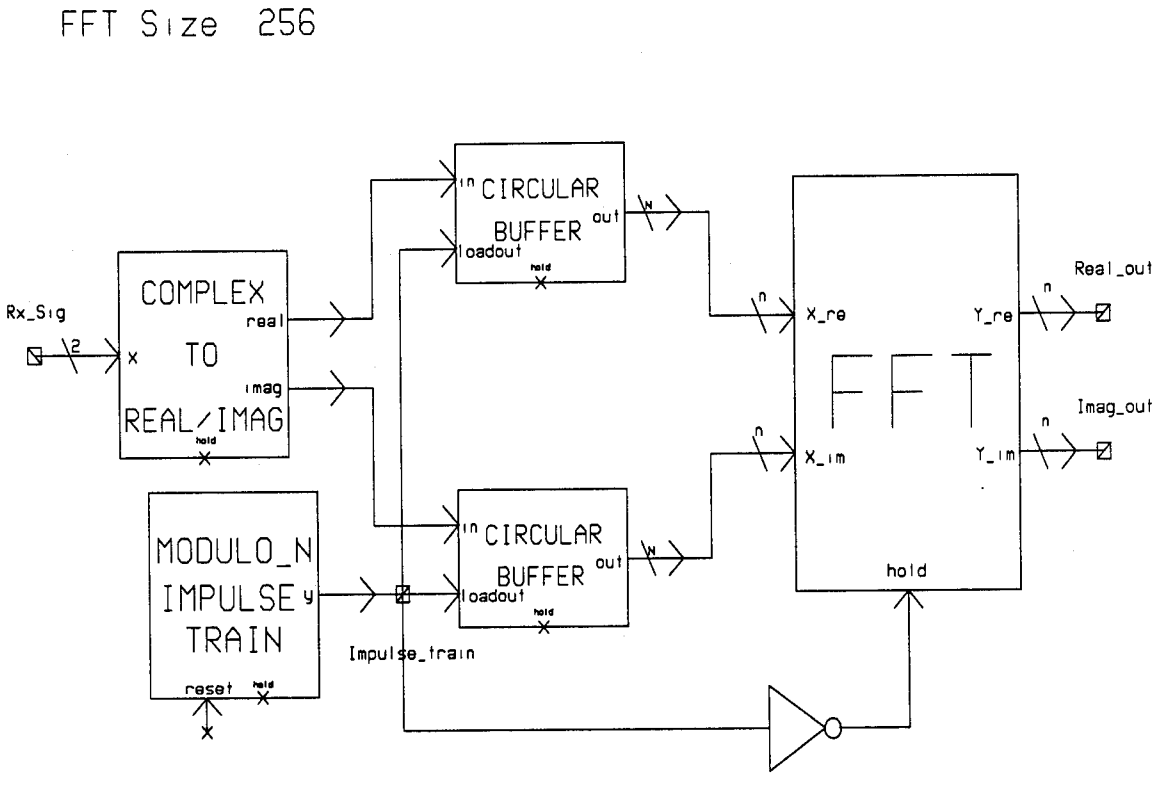


Figure 8. First DEF Subsystem

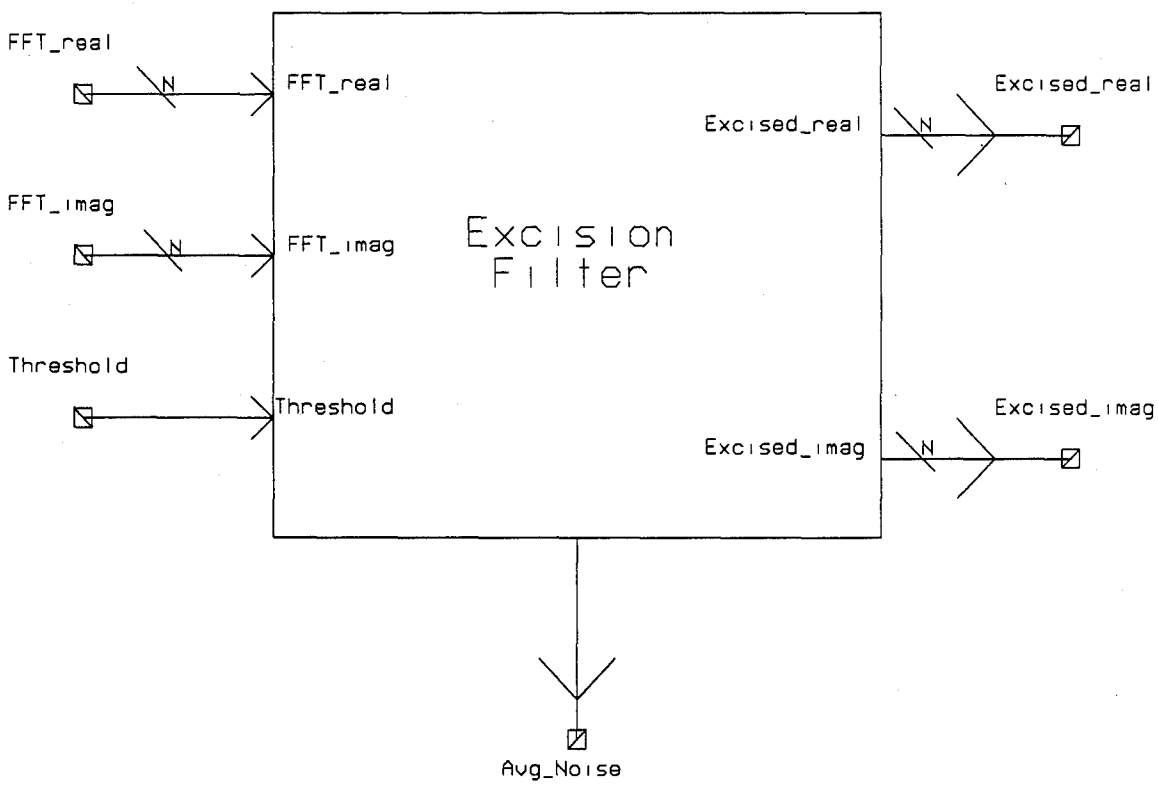


Figure 9. Second DEF Subsystem

desired threshold is $6dB$, then the output value will be 3.98 times the average of the 256 point frequency vector. The code used in programming the excisor custom coded block can be found in Appendix A. The final subsystem of the DEF is shown in Figure 10. The excised data is inverse transformed and serialized. Again, the timing pulses are required to ensure that the IFFT and the vector-to-serial operation are performed on a valid vector. Figure 11 depicts the DEF with all its subsystems interconnected.

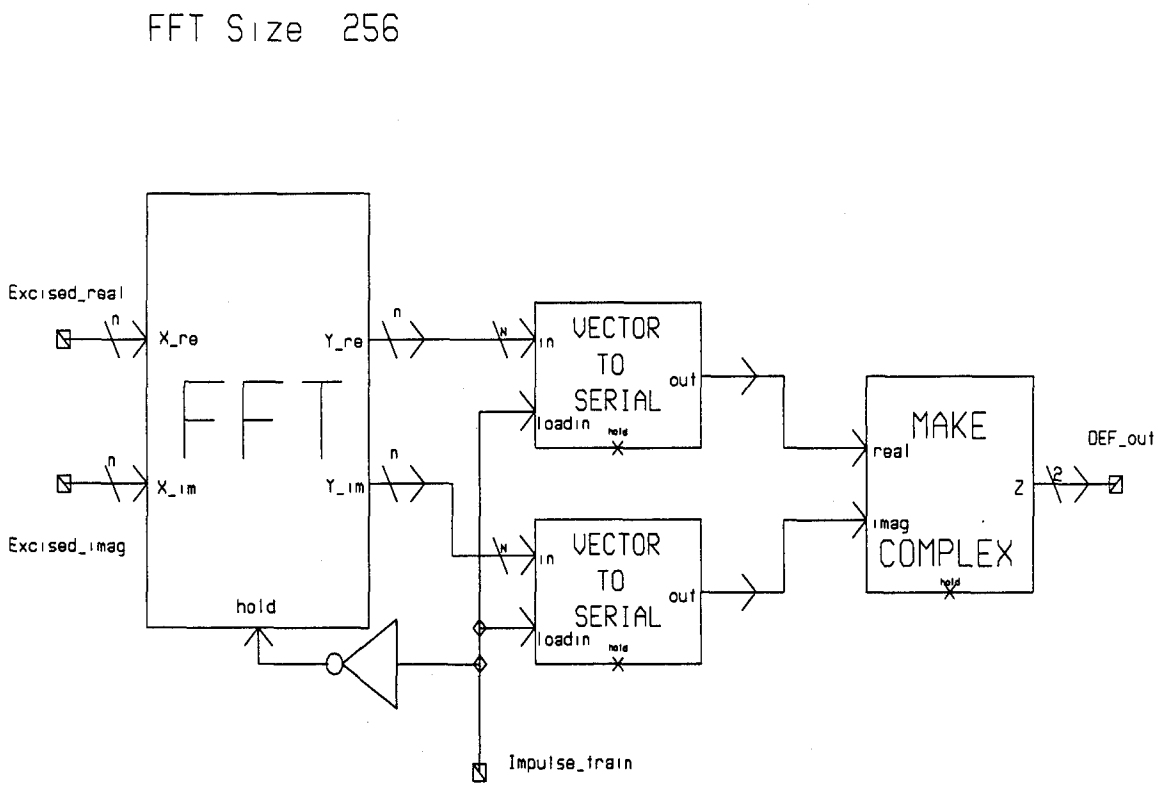


Figure 10. Third DEF Subsystem

3.5.4 Channel and Jammers. The final model needed for completeness is the channel itself and the jammers. For the Monte Carlo Simulations, the channel consisted of AWGN, the jamming signal, and the transmitted spread spectrum signal. Figure 12 shows the block diagram of the channel. The channel adds the transmitted spread spectrum signal

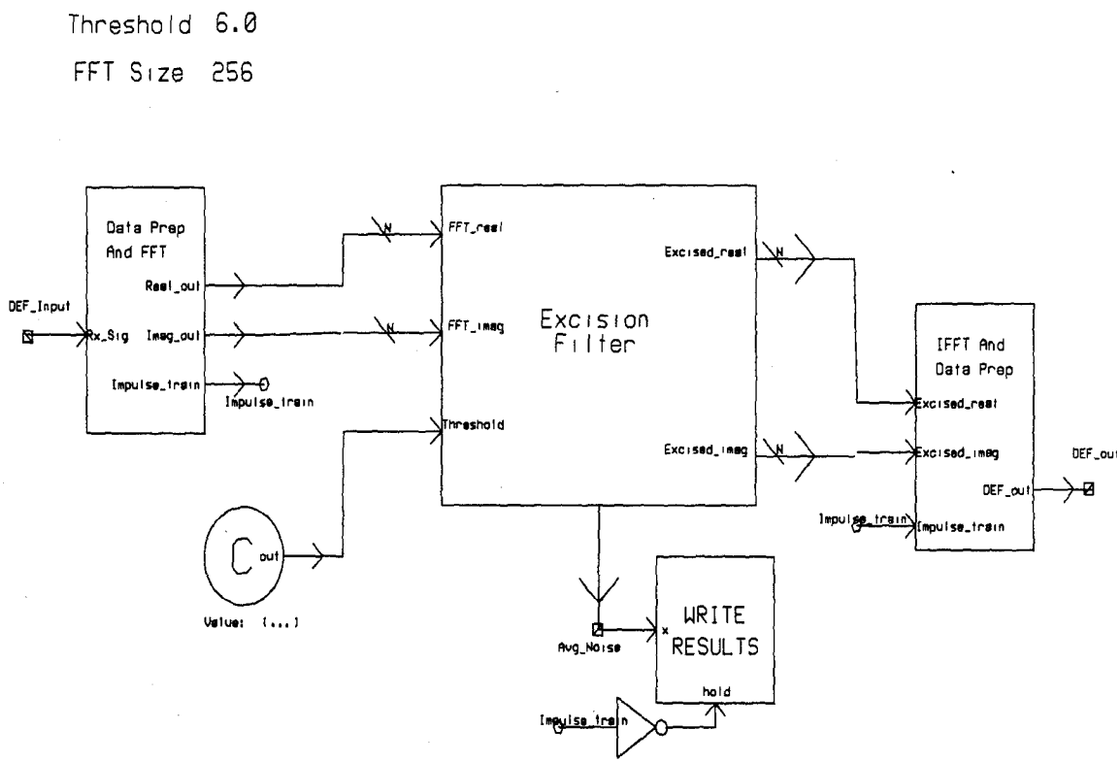


Figure 11. Digital Excision Filter System

to complex white noise and then adds the jamming signal. For the SA simulations, the only difference was the removal of the complex white noise block.

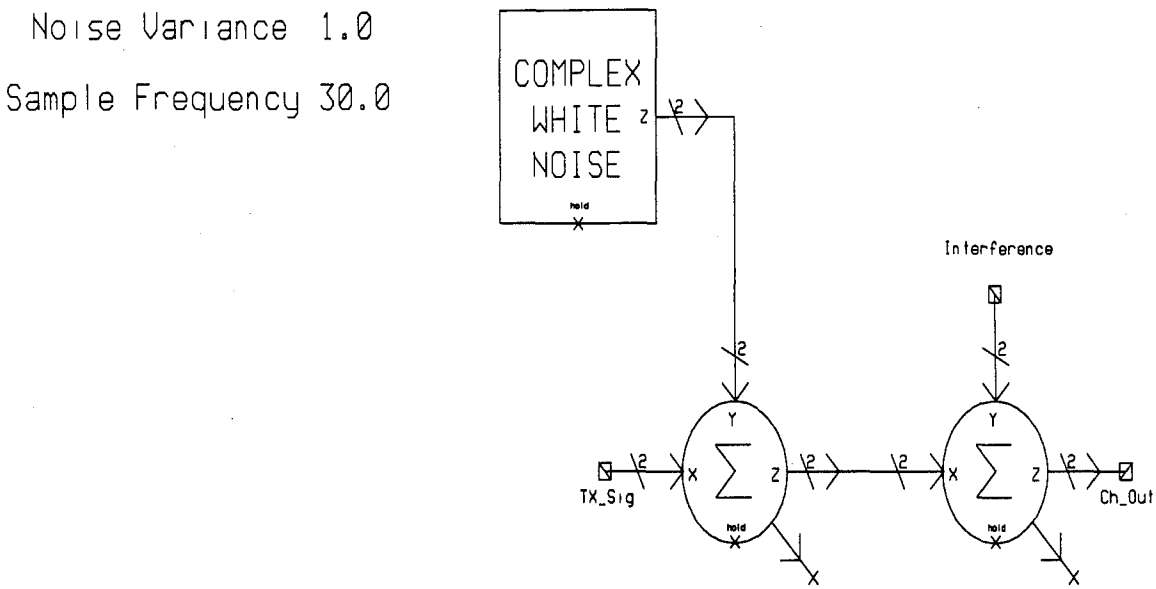


Figure 12. Channel Model

Figure 13 illustrates the construction of the jamming signal, which is based off of a complex tone. The jammers block allows up to four jammers, which can be any combination of continuous wave or pulsed jamming. Only a single section of the jammers block is presented since all four parts are identical and are simply added together to construct the jamming signal. A square wave is used to control the duty cycle of the waveform and a sinusoidal generator is used for the sine wave. The power in the waveform is controlled by changing the values in the parameter screens. For CW jammers the pulse amplitude is set to a desired value and the duty cycle is set to 100.0. For pulsed jamming, the pulse frequency and duty cycle must be adjusted to obtain the desired result. By changing the frequency of the tone, the jammer is moved around in the spectrum. A tone placed at 0 Hz corresponds to a jammer at the center frequency.

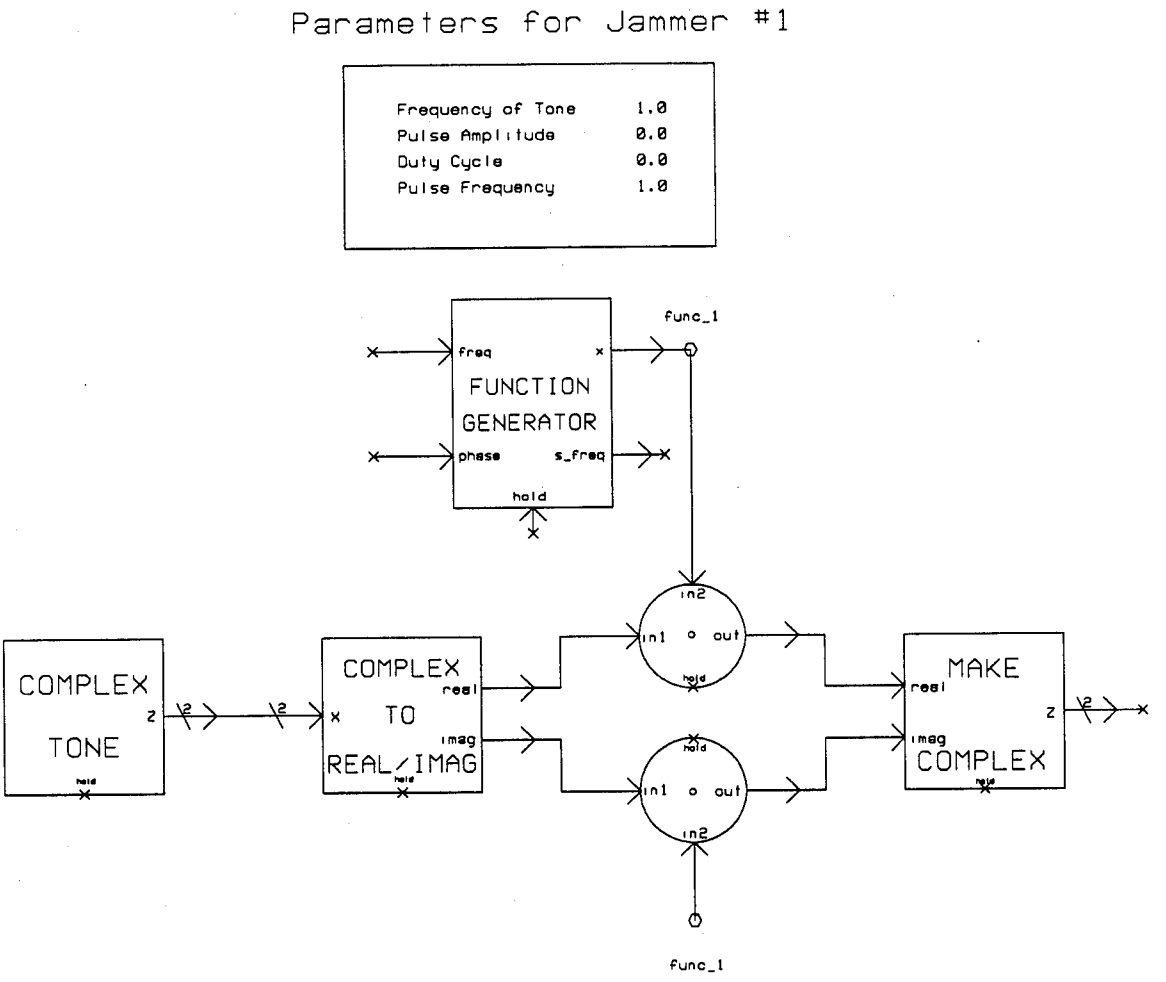


Figure 13. Jammer Model

3.5.5 DS BPSK System with DEF. With all the pieces available, each subsystem was interconnected to construct the complete system. Figure 14 shows the entire system complete with jammers and the DEF. The simulations are controlled through the parameter screen and by changing the required parameters for each jamming scenario. For all simulations, the input data bit rate was held to 1 bps. Processing gains of 33 dB and 37 dB are used for Monte Carlo Simulations and Semi-Analytic simulations, respectively. The 33 dB processing gain was only used during test cases for the Monte Carlo Simulations. All Semi-Analytic simulations used a processing gain of 37 dB. The lower processing gain for the MC simulations was needed to reduce the computer run-time. Recall, for MC simulations 500×10^6 iterations were required for a valid BER estimation, and for SA simulations only 1×10^6 iterations were needed.

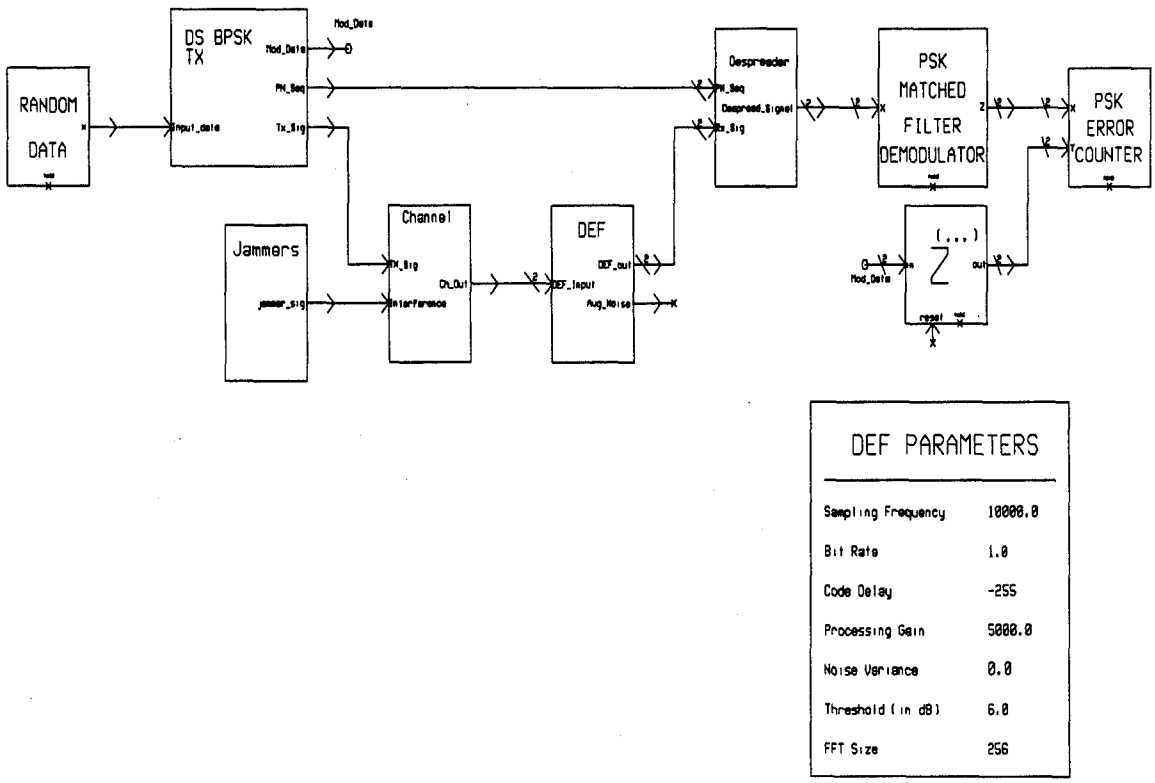


Figure 14. DS BPSK System Model

With the entire system interconnected, it is important to show that the system functions correctly. Figure 15 contains spectrum plots of the original data, the spread data, and the despread data all uncorrupted by noise. It is clearly seen that the data is spread over 2000 Hz, and then despread to its original bandwidth, which proves the system is properly synchronized. All transmitted bits matched the received bits. The data information provided to the right of the plots are for a data cursor which cannot be seen. The cursor was a vertical line that resembled a tone jammer, or spike and was moved off the plot to avoid confusion.

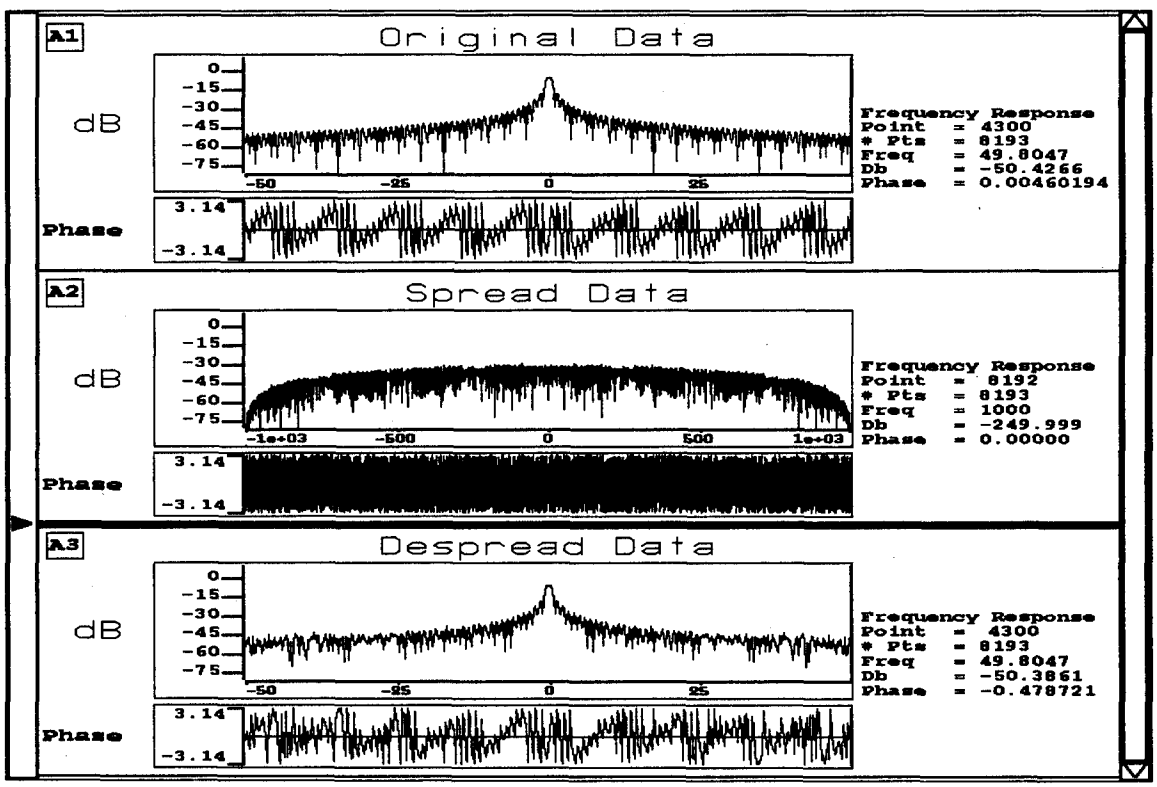


Figure 15. Spectra of Data

Another important point in the system is the output of the DEF. Figure 16 contains the spectrum of the DEF input and output. As shown Figure 16(a), there is one continuous wave jammer present at the center frequency prior to the DEF. After processing through the DEF, a large portion of the jammer spectrum has been excised. In other words, the frequency bin containing the jammer was set to zero since its magnitude exceeded the threshold level.

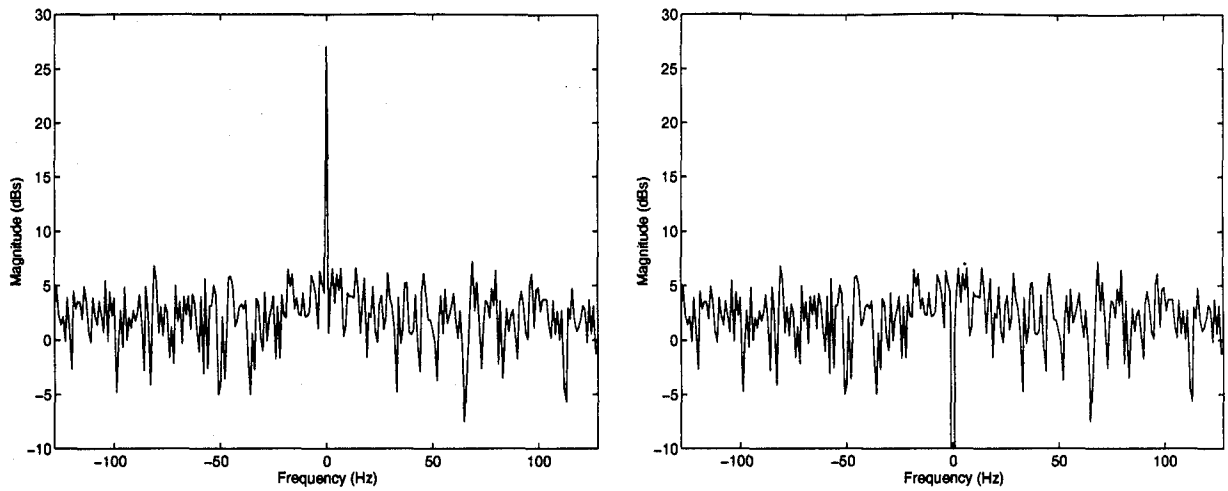


Figure 16. Spectra Plots for 1 Jammer: (a) Channel Output (b) DEF Output

An important characteristic of the DEF is to not alter the signal if no jammer is present, as demonstrated in Figure 17. The spectra before and after DEF processing are identical, which indicates the DEF did not perturb the unjammed signal. The results shown in these figures clearly proves the DEF to be functioning properly.

3.5.6 System for Measuring PAR Value. A major objective of this research was to correlate the results obtained in SPW with those achieved in MATLAB by Capt. Falen (7). His research focused upon the effects of jamming upon the peak to average ratio (PAR). The PAR is used as a measure of how well the autocorrelation receiver will be able to discern the correlation spike of the received signal and is defined as the correlation spike divided by the average correlation value after the peak spike has been removed (7). A major problem encountered while developing the model in SPW was the fact that SPW imposed limitations on the PN sequences generated by their PN sequence generator. For a given length register, only one set of taps could be used, which results in using only one sequence. Therefore, a Gold Code could not be constructed from the modulo-2 addition of preferred pairs of PN sequences.

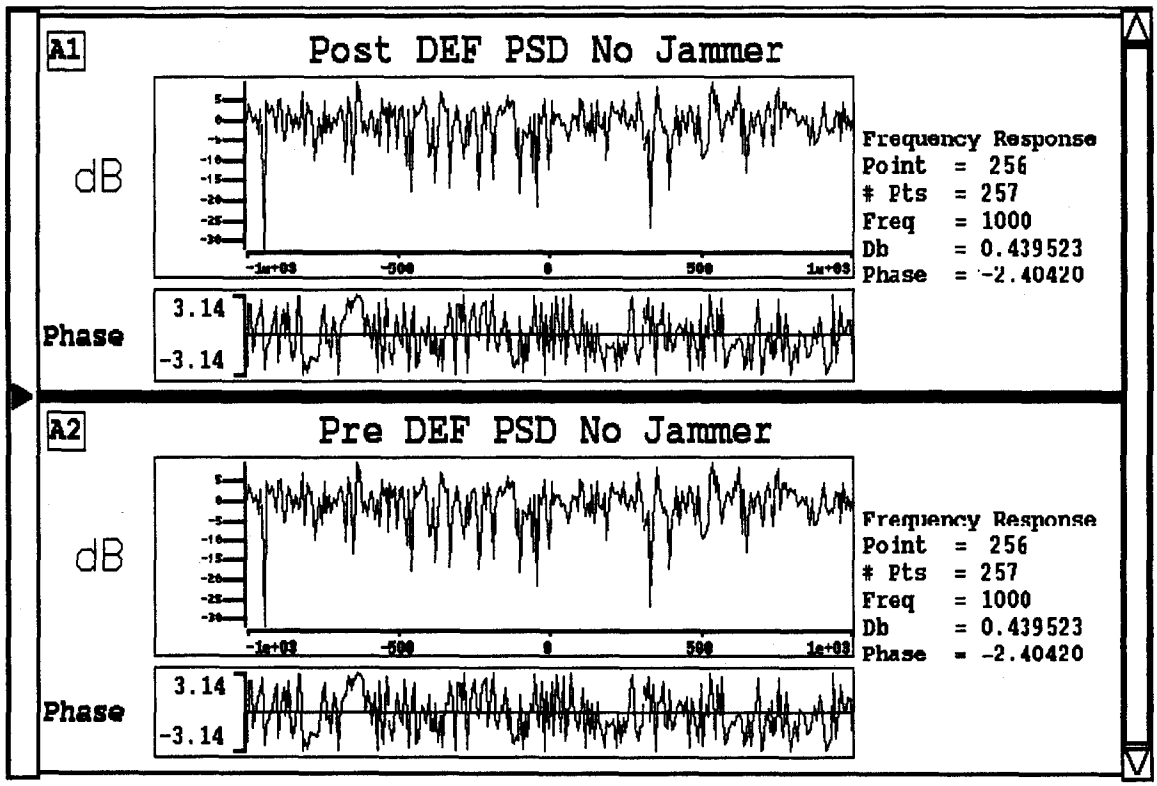


Figure 17. Spectra Plots of DEF and Channel Output With No Jammer Present

As a result, the P code was generated with Capt. Falen's zplatinum MATLAB function. This code was imported into SPW for use as the source signal. Figure 18 shows the system used to obtain the waveforms necessary to calculate the PAR. As can be seen, AWGN and jamming are added to the code. The noise was configured to achieve an SNR of $-30dB$ for the baseline PAR value, and added jammers as shown in Table 1 for subsequent PAR calculations.

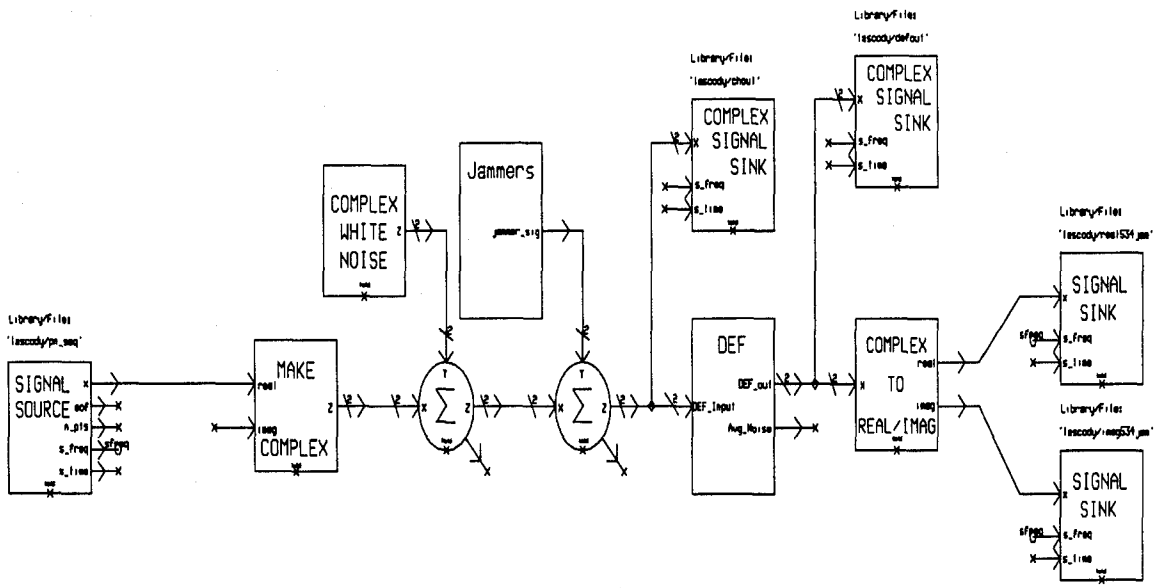


Figure 18. System Model for Measuring PAR

At the completion of the simulation, the real and imaginary portions of the excised signal were exported to MATLAB. The following MATLAB code was used to calculate the PAR value.

```
function y=par(a,b)
  A=zcor2(a,b);
  A1=A;
  [m,n]=max(abs(A1(102100:102190)));
  A1(n)=0;
  avg=mean(abs(A1));
  y=10*log10(m/avg);
```

The peak spike search was limited to 45 points on either side of the expected location of the true correlation spike. This restriction reduces the risk of calculating an invalid PAR value

when the correlation spike is not in the right location. Figure 19 illustrates the correlation spike in the receiver. The large spike in the center with an amplitude of 1024 represents the true

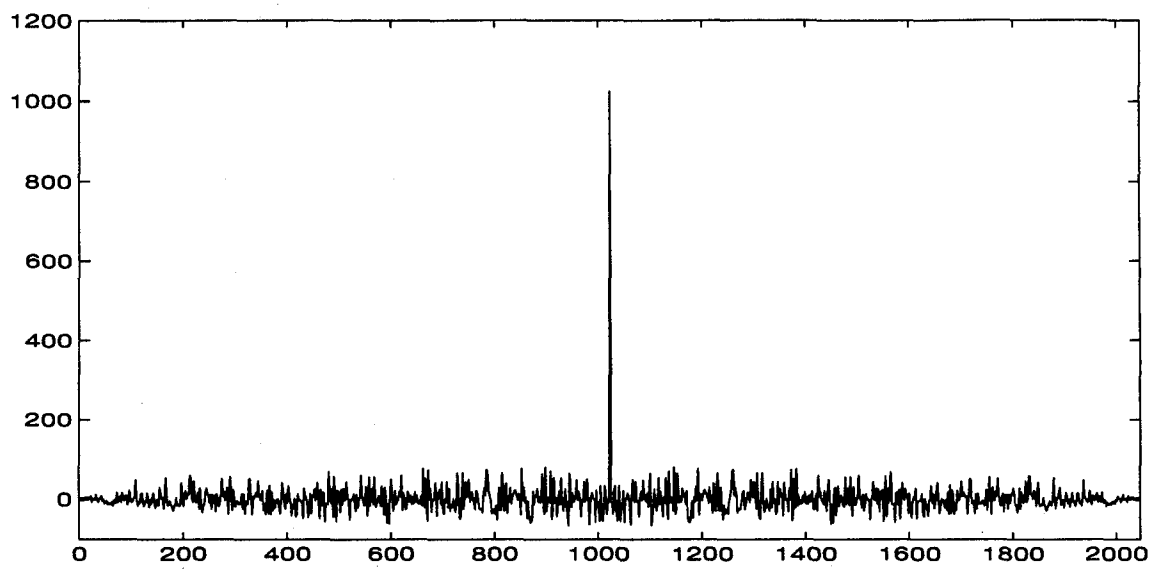


Figure 19. Sample Correlation Sequence Used to Calculate PAR

correlation spike. Eliminating this spike and taking the average yields and average correlation sidelobe of approximately 12. The PAR is then the ratio of the peak and the average sidelobe, $19.3dB$.

3.5.7 Total Least Squares (TLS) Prony. Another outcome of the June 1995 JPO briefing was to look into another excision technique. After deliberation, the second method selected to implement was TLS Prony, since it quickly and effectively estimates sinusoids in noise. It does not use an iterative approach like some other estimation routines, such as Iterative Quadrature Maximum Likelihood, which is highly desirable in consideration of the run-times involved for these simulations. Once the interference is estimated from the input data, the estimated interference data is subtracted from the input signal, which provides the

interference excision. Prony's Model, which models the interference, is given by (17)

$$d(n) = \sum_{k=1}^{K} a_k p_k^n, \quad n = 0, 1, \ldots, N-1, \tag{25}$$

where

$$\begin{aligned} p_k &= k\text{th pole, complex number} \\ a_k &= k\text{th amplitude coefficient, complex number} \\ K &= \text{number of modes (or number of interferers).} \end{aligned}$$

Prony's Model models the time domain data. Parameter estimation is required to estimate the poles and amplitude coefficients from the given data. It is assumed that the data is noise corrupted, i.e. $d'(n) = d(n) + w(n)$, where $w(n)$ is an additive white noise process. The noise, $w(n)$, is the quantity of interest since that is where the spread spectrum signal is located. The interferers, modeled as sinusoids $(d(n))$, are undesirable, and are subtracted away from $d'(n)$. A backward linear prediction approach which utilizes a Singular Value Decomposition (SVD) based noise cleaning (which is described in (17)) technique is used to determine the pole estimates. The backward linear prediction equations are

$$\begin{bmatrix} d'(0) & d'(1) & d'(2) & \cdots & d'(Q) \\ d'(1) & d'(2) & d'(3) & \cdots & d'(Q+1) \\ \vdots & \vdots & \vdots & & \vdots \\ d'(N-Q-1) & d'(N-Q) & d'(N-Q+1) & \cdots & d'(N-1) \end{bmatrix} \begin{bmatrix} 1 \\ b_1 \\ b_2 \\ \vdots \\ b_Q \end{bmatrix} \approx 0. \tag{26}$$

or

$$S \begin{bmatrix} 1 \\ b \end{bmatrix} \approx 0 \tag{27}$$

where Q is the order of prediction, and b is the coefficient vector of the polynomial $B(z)$ given by

$$B(z) = 1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_Q z^{-Q}. \tag{28}$$

Ideally, Q can be any integer greater than or equal to the model order K; in practice, choosing $Q > K$ results in more accurate parameter estimates (3),(21).

Equation 26 is now solved for b by performing an SVD of the matrix S and truncating all but the first K singular values to arrive at a noise cleaned estimate $\widehat{S}$ (17). Next, $\widehat{S}$ can be written as $[\widehat{s}_1 \ \widehat{S}_1]$, where $\widehat{s}_1$ is a column vector consisting of the first column of $\widehat{S}$ and where $\widehat{S}_1$ is a matrix consisting of the remaining columns of $\widehat{S}$. Next, the estimate of b, $\widehat{b}$, is found using a least squares solution, which can be written as

$$\widehat{b} = -(\widehat{S}_1^H \widehat{S}_1)^{-1} \widehat{S}_1^H \widehat{s}_1, \tag{29}$$

although numerically more robust solutions (using, *i.e.*, the QR decomposition (9)) are preferred to direct computation of Equation 29.

Finally, the estimated poles are found by

$$\widehat{p}_q = \frac{1}{\text{zero}_q\left(\widehat{B}(z)\right)}, \qquad q = 1, 2, \ldots, Q. \tag{30}$$

Because only K singular values of $\widehat{S}$ are nonzero, there are at most K pole estimates which can correspond to true modes. Therefore, only the K poles which have the largest energy will be retained. To determine the energy, the amplitude coefficients must first be estimated.

Consider Equation 25 in matrix form,

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ p_1 & p_2 & \cdots & p_Q \\ p_1^2 & p_2^2 & \cdots & p_Q^2 \\ \vdots & \vdots & & \vdots \\ p_1^{N-1} & p_2^{N-1} & \cdots & p_Q^{N-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_Q \end{bmatrix} = \begin{bmatrix} d'(0) \\ d'(1) \\ \vdots \\ d'(N-1) \end{bmatrix}, \tag{31}$$

or

$$PA = \mathcal{D}'. \tag{32}$$

The amplitude coefficients are found from a least squares solution to Equation 32 using the pole estimates; this can be written

$$\hat{A} = \left(\hat{P}^H \hat{P}\right)^{-1} \hat{P}^H \mathcal{D}' \tag{33}$$

although numerically more robust solutions (using, *i.e.*, the QR decomposition (9)) are preferred to direct computation of Equation 33.

Next, the Q mode energies are calculated using

$$E_q = |\hat{a}_q|^2 \sum_{n=0}^{N-1} |\hat{p}_q|^{2n} \qquad q = 1, 2, \ldots, Q. \tag{34}$$

The K poles whose corresponding energies are highest are retained. The final set of K amplitude coefficients are re-estimated in a manner analogous to that outlined in Equations 31 to 33. The only differences are that now there are only K pole estimates to be utilized and only K amplitude coefficients to be estimated. Now that the K poles and K amplitude coefficients are estimated, $d(n)$ must be subtracted from $d'(n)$, leaving only $w(n)$, which is the spread spectrum signal and real noise.

3.5.7.1 Implementation in SPW. MATLAB files were provided by Capt. Sacchini that performed the TLS Prony parameter estimation. To get the routine working in SPW, the MATLAB computational engine must be started within SPW. To accomplish this, a test function, provided in the MATLAB External Interface Guide on page 1-33, was coded in C. This function successfully placed a matrix into the MATLAB computational engine and returned the second eigenvalue. With that success, a simulation was run in SPW to just start the MATLAB engine, which was successful. Then, the previously mentioned C-function was converted from straight C-code into a custom-coded block for use in SPW. This function successfully placed a matrix from SPW into the MATLAB computational engine and returned the correct eigenvalue from MATLAB to the SPW simulation. However, it was noticed that the MATLAB computational engine was started and stopped every iteration resulting in even longer run times. At this time, a method to start the computational engine once and leave it running until the simulation is over has not been found.

Finally, the TLS Prony files were implemented within SPW. Figure 20 shows the function symbol identifying the inputs and outputs and the SPW C-code can be found in Appendix 2. Figure 21 illustrates the subsystems that make up the TLS Prony Excisor. To do the actual excision, the estimates from TLS Prony were subtracted from the input data. Thus, only the estimated jammer(s) are removed, in lieu of excising the entire portion of the spectrum.

Several problems became readily apparent after a few attempts at implementing TLS Prony. One major problem was the use of computer memory. With MATLAB running inside of SPW, memory conflicts arose during many of the trials. Also, many trials failed when SPW tried to pass the data into the MATLAB engine and call the TLS Prony function. It was determined that the problem was related to the first 255 data points. The first 255 points are always zero because it takes that many points until the first vector is formed and if all zeros are

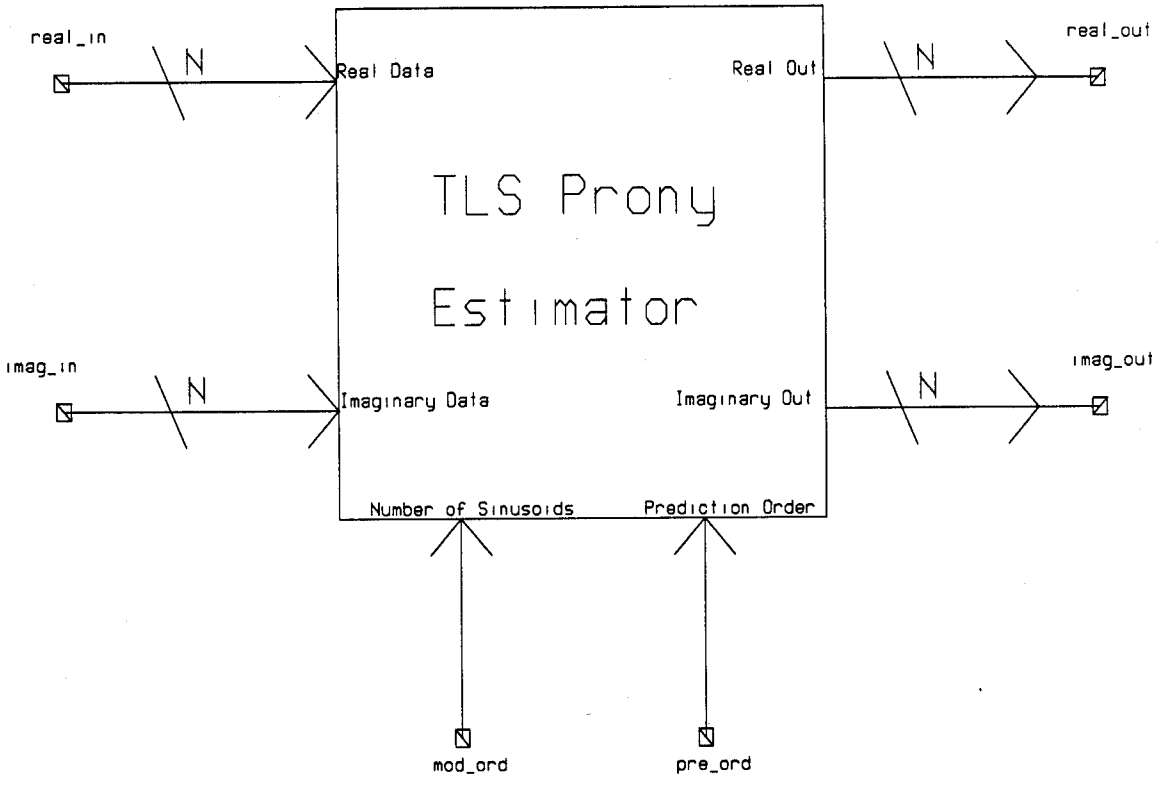


Figure 20. SPW Function Symbol For TLS Prony

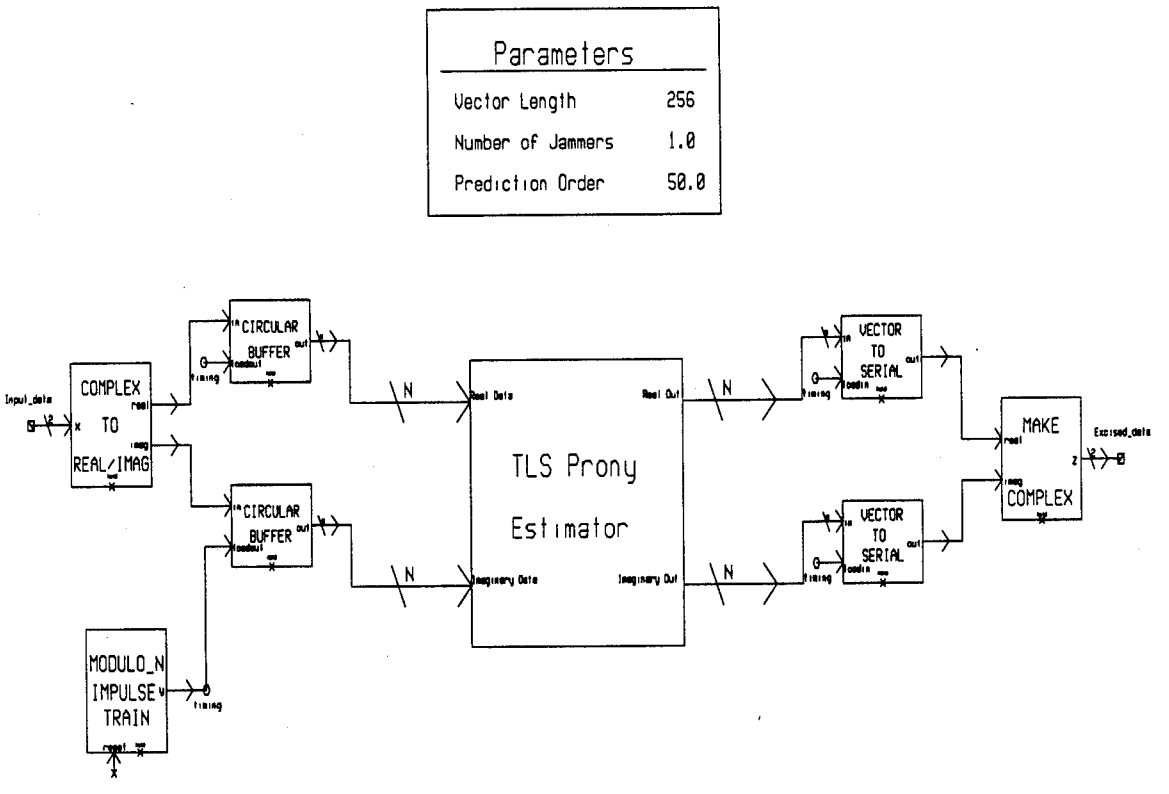


Figure 21. TLS Prony Subsystems

passed to TLS Prony, then it fails returning a null back to SPW. To overcome this, a test was implemented to determine if the data was all zeros and, if so, then the MATLAB engine would not be called. The output was assigned to the input so no nulls should have occurred. Many different ways of implementing this check were tried, such as doing the check in MATLAB, but none solved the problem. Due to time limitations, this objective could not be pursued, but ideas that may prove successful are provided in the following section.

3.5.7.2 Recommendations For Implementing TLS Prony in SPW. The following methods are proposed to get the TLS Prony excisor functioning. The current method relies on activating the MATLAB computational engine for every simulation iteration, which is extremely time consuming. One approach would be to alter the simulation to record the real and imaginary portions of the signal coming out of the channel, which consists of noise, jamming, and the spread data signal. After eliminating the first 255 zeros caused by converting the serial data into a vector signal, these files could be ported over to MATLAB and recombined into a complex signal. The complex signal would then serve as the input to the TLS Prony MATLAB function. The result obtained from subtracting the input signal from the output of the MATLAB function could then be converted for use in SPW. Then, bit error rates and PAR values could be calculated in another SPW simulation. However, drawbacks to this approach are that the simulation is not continuous and that the simulation could not be converted into VHDL.

An alternate approach, and the suggested one, would be to convert the MATLAB files that implement TLS Prony into C-code, which could be used in the construction of a custom coded block in SPW. This method would bypass the issue of starting and stopping the MATLAB computational engine during each iteration. The remaining problem with this method deals with eliminating the preliminary zero vectors produced by the circular buffers. A transmitted data signal could be formed in a separate simulation and then used as inputs for

the new simulation containing the TLS Prony custom coded block and receiver blocks. This data file of a transmitted DS BPSK waveform and jamming would, in effect, take the place of the transmitter. This method would be easier to manage since only one platform, SPW, would be used.

3.5.7.3 Tips For Linking MATLAB and SPW. In order to make the MATLAB computational engine available to SPW, several sections of the C-code must be edited,using the same procedures outlined previously. The following lines, also in Appendix B, must be added exactly as shown to the C-code:

1. In the LINK_ OPTIONS INFORMATION section, add LINK_ OPTIONS = {"-L/apps/matlab/4.2c/extern/lib/sun4","-lmat"};.
2. In the INCLUDE_DIRS INFORMATION section, add INCLUDE_ DIRS = {"//apps/matlab/4.2c/extern/include"};.

3.5.8 System for Measuring Percent Jammer Power Removed. Another metric that is important when considering jammer excision is the amount of jammer power the excisor removes from the incoming signal. Figure 22 depicts the simulation used to determine the percent jammer power removed from the spread spectrum signal. One output of the DEF block discussed in Chapter 3.5.3 is the excision threshold level, which is based upon the input threshold and the average magnitude of the input vector. This level is used as the threshold level for the DEF FOR JAMMER SIGNAL ONLY block, whose custom code is provided in Appendix 3. The Jammers block is configured exactly as it was for the regular simulation, except that no noise or data is present at the DEF input. The power in the jamming waveform is calculated by summing up the squared magnitudes of the transformed data. In other words,

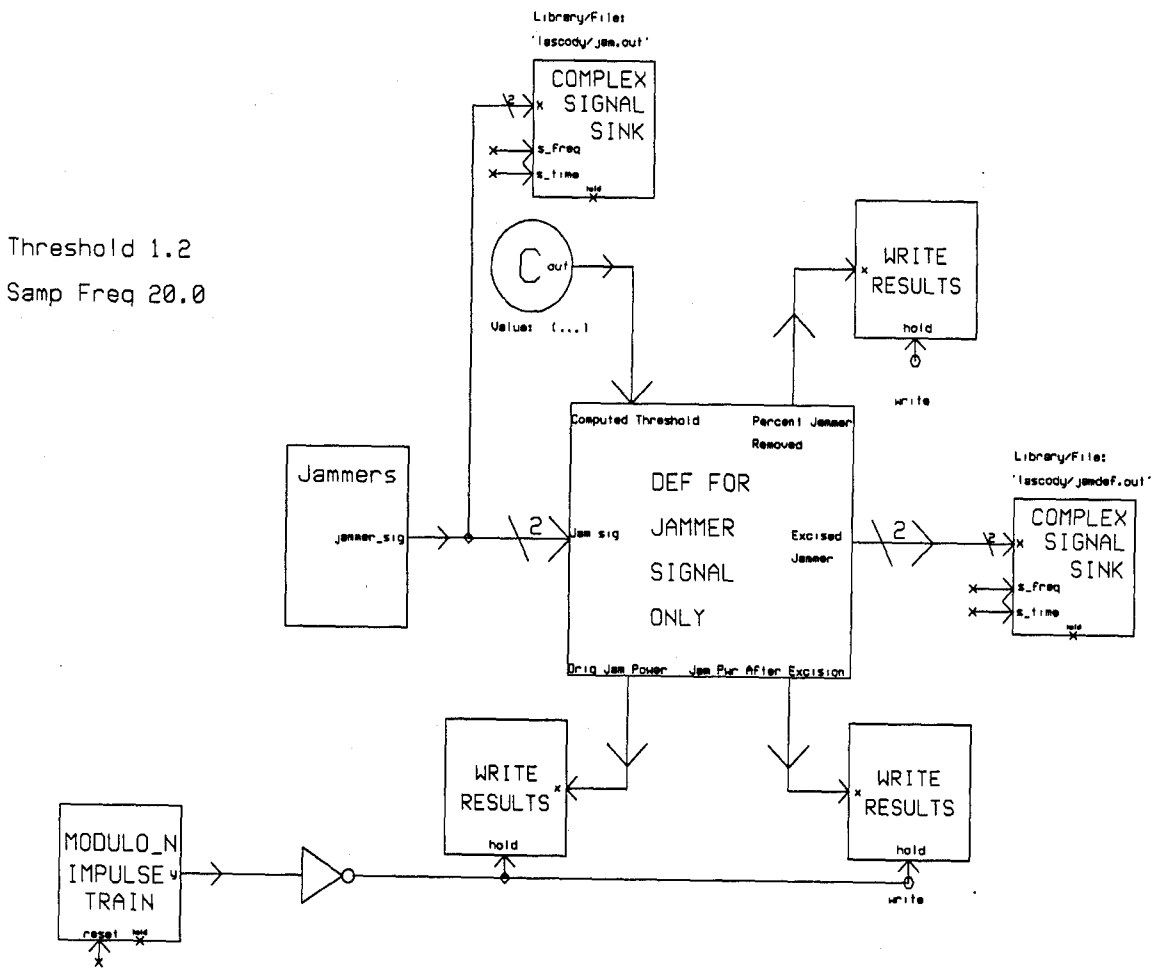


Figure 22. SPW Model for Determining Jammer Power

Parseval's relationship for the DFT was used in the following manner:

$$\sum_{n=0}^{N-1} \mid x[n] \mid^2 = \frac{1}{N} \sum_{k=0}^{N-1} \mid X[k] \mid^2 . \tag{35}$$

Then, the signal is excised and the power is recomputed using the same method. Using these values, the percentage of jammer power removed is given by

$$PJR = 100(1.0 - \frac{P_{excised}}{P_{jammer}}), \tag{36}$$

where $P_{excised}$ is the power contained in the signal after excision. It is important to note that the threshold value taken from the original DEF simulation be from a MC simulation because the MC simulations use the AWGN source to setup the proper SNR. If it is not taken from these simulations, the threshold value will not be correct.

3.6 Summary

In this Chapter, an introduction to simulation theory was presented along with the proper representation of communication signals. Of the five BER evaluation techniques mentioned, Monte Carlo simulations and Quasi-analytical Estimation techniques and how they applied to the simulations in this research were discussed. Finally, the system models that were developed in SPW were presented and explained in detail.

IV. Simulation Results

4.1 Introduction

This chapter presents the results of the simulations for a number of different jamming scenarios. Peak-to-average correlation values and BER curves are shown to illustrate the performance improvements the DEF brings to the DS-BPSK system. The sections are broken into single and multiple CW jamming; single and multiple pulsed CW jamming; and broadband noise jamming. Results are also provided to show the amount of jamming power the DEF removes from the incoming signal.

4.2 Scenarios

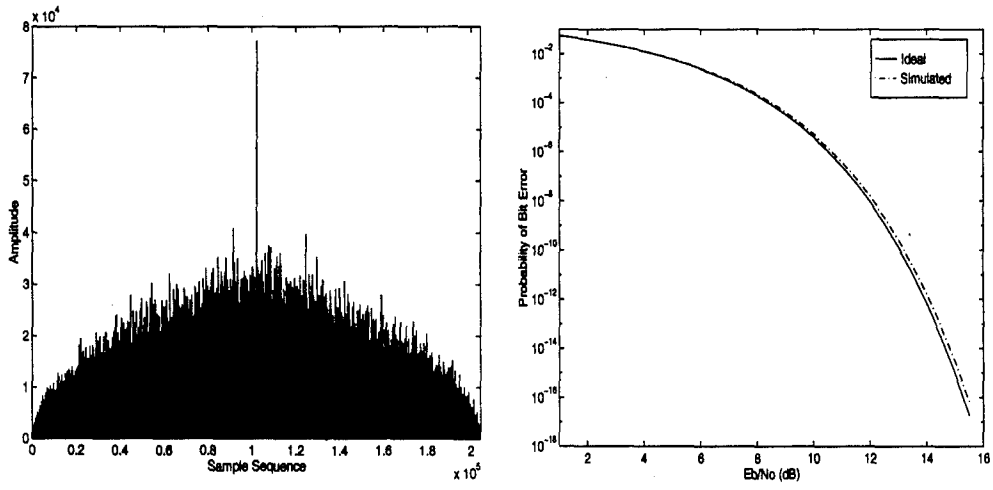
In Chapter 2, a number of different jamming signals are listed that could be used against a communications system. For this research, three types of jammers are considered: CW, pulsed CW, and Broadband Noise. It has been documented that the tone jammer is the most effective when it is centered on the operational frequency of its target (13). Table 1 outlines the different jamming scenarios examined in this research. Care was taken to correlate these scenarios with the previous work done by Capt. Falen. Specifically, the simulations in SPW were configured to closely model the results Capt. Falen achieved in MATLAB with respect to the PAR value behavior. Single tone jammers were placed exactly on the center frequency (no offset), and multiple jammers were placed at 625, 1250, and 2500 Hz offsets from the center frequency. J/N ratios cited in the tables are for single jammers. Each jammer in the multiple jammer scenario has the stated J/N ratio (i.e. four jammers each having 33 dB J/N).

For the most part the J/N ratios examined were 33, 50, and 53 dB. These values were chosen based on the J/N ratios and processing gain used in Capt. Falen's research. He used a processing gain of 53 dB and J/N ratios of 50 and 70 dB. The J/N ratios used in this research

Table 1. Scenarios

SCENARIO	*NUMBER AND TYPE OF JAMMERS*	*PARAMETERS*
1	NONE	NONE
2	1 CW	Freq Offset (kHz): 0 J/N: $33dB$
3	1 CW	Freq Offset (kHz): 0 J/N: $50dB$
4	4 CW	Freq Offset (kHz): $0, 0.625, 1.25, 2.5$ J/N: $33dB$
5	4 CW	Freq Offset (kHz): $0, 0.625, 1.25, 2.5$ J/N: $50dB$
6	1 Pulsed CW	Freq Offset (kHz): 0 J/N: $33dB$
7	1 Pulsed CW	Freq Offset (kHz): 0 J/N: $36dB$
8	1 Pulsed CW	Freq Offset (kHz): 0 J/N: $50dB$
9	1 Pulsed CW	Freq Offset (kHz): 0 J/N: $53dB$
10	4 Pulsed CW	Freq Offset (kHz): $0, 0.625, 1.25, 2.5$ J/N: $33dB$
11	4 Pulsed CW	Freq Offset (kHz): $0, 0.625, 1.25, 2.5$ J/N: $50dB$
12	4 Pulsed CW	Freq Offset (kHz): $0, 0.625, 1.25, 2.5$ J/N: $53dB$
13	1 Broadband Noise	Bandwidth: Main Lobe J/N: $6dB$
14	1 Broadband Noise	Bandwidth: Main Lobe J/N: $9dB$

were scaled due to the difference in processing gains. Threshold values were kept to either 6 or 9 dB, as dictated by Capt. Falen's research (7), except for a few scenarios where the threshold was varied from 1 to 12 dB to plot the PAR value behavior over an extended range. A PAR value was measured for each of the above scenarios, and bit error rate curves were plotted for scenarios 1-5, 6, 8, 10, and 11. For the baseline system (no jamming, $-30\,dB$ SNR), the PAR value was 11.60 dB, and the PAR value for the p-code correlated with the p-code(no noise, no jamming) was 27.80 dB. The second PAR value is presented as an indicator of how the noise affected the correlation. Figure 23 shows the cross correlation sequence used to calculate the baseline PAR and the BER curve for the baseline conditions. The BER curves presented graph the probability of bit error against the average signal power to average noise power, $\frac{E_b}{N_o}$. This ratio is the standard quality measure used in determining system performance in digital communications systems.



(a) (b)

Figure 23. Results of Scenario #1: (a) PAR Correlation Sequence (b) Baseline BER Curve

The peak value occurs exactly in the middle of the PAR correlation sequence. Other peaks occur that are within 3 dB of the peak, but these peaks were outside the search window discussed in Section 3.5.6. The BER curve illustrates that the system is almost identical to

an ideal BPSK system. Again, it is important to point out that the BER curves were obtained with *precisely* known synchronization and show the effects of excision and the remaining jammer power on the demodulation. The PAR correlations are an indicator of the ability of the system to synchronize. These value were obtained with two separate and distinct simulations. Synchronization was not a part of the BER simulations. Thus, some plots may seem contradictory in the sense that error rates are provided under circumstances where the receiver couldn't synchronize properly. However, the figures are not correlated and should not interpreted as such.

4.3 CW Jammers

4.3.1 One CW Jammer. The CW jammer was placed exactly on the center frequency of the simulation. In SPW, this results in the jammer being represented as a single spike, as expected. Therefore, for these simulations 100% of the jammer power is removed since all the power is contained in a single frequency bin. Table 2 contains the PAR values measured using

Table 2. PAR Values For 1 CW Jammer

Scenario	*PAR Value*	*J/N (dB)*	*Threshold (dB)*
2	11.1821	33	6
2	11.1739	33	9
3	10.5821	50	6
3	10.5821	50	9

the system outlined in Section 3.5.6. For Scenario 2, the PAR value decreased minimally when the threshold was increased from 6 to 9 dB. When the J/N was increased in scenario 3, the PAR value decreased by 0.6 dB. With the increased power, the jammer spectrum occupied a few more frequency bins resulting in a larger portion of the spectrum being excised.

Figure 24 contains the bit error rate curves for three J/N ratios and using a 6dB threshold. The flat solid line in each of the graphs shows the probability of bit error for the system without

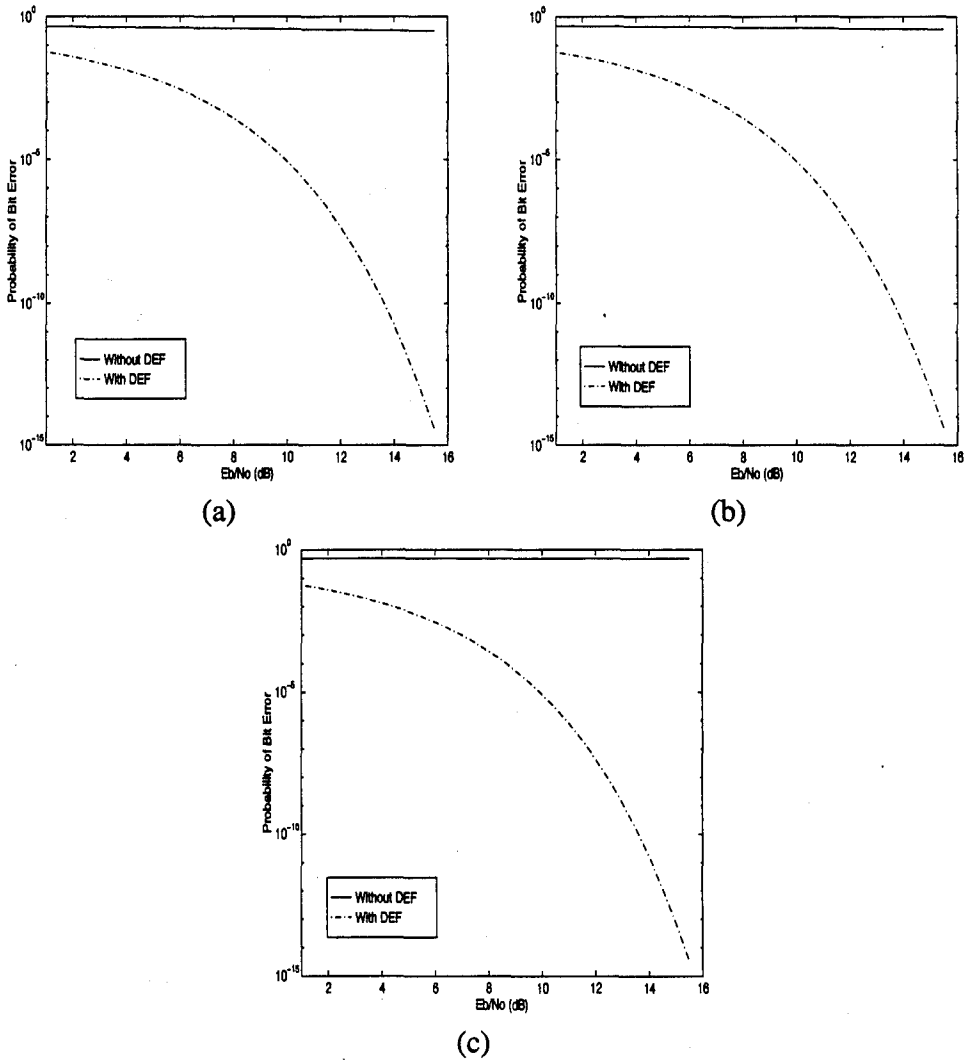


Figure 24. BER Curves For 1 CW Jammer: (a) 27 dB J/N (b) 33 dB J/N (c) 50 dB J/N

using the DEF. The dashed, dotted line graphs the performance of the system using the DEF, resulting in a significant improvement in BER. Comparing these to the simulated baseline BER curve, it can be seen that these curves closely follow the baseline.

4.3.2 Four CW Jammers. As stated previously, one jammer remained on the center frequency and the other three were offset by 625, 1250, and 2500 Hz from the center frequency. Note that the chip rate was 5000 Hz, which places the final jammer just at the edge of the main lobe. Table 3 contains the PAR values measured for Scenarios 4 and 5. PAR behavior over an extended range of thresholds was examined in Scenario 4. Figure 25 shows the PAR value fluctuations in Scenario 4 for the different threshold settings with the circles indicating actual data points. The PAR values for Scenario 5 and the latter part of Scenario 4 indicate that the

Table 3. PAR Values For 4 CW Jammers

Scenario	*PAR Value*	*J/N (dB)*	*Threshold (dB)*
4	10.3208	33	1
4	10.7581	33	3
4	10.4519	33	6
4	8.6227	33	9
4	7.0521	33	12
5	6.1703	50	6
5	6.1170	50	9

autocorrelation receiver would have a difficult time acquiring the signal. PAR values in this range indicate that the cross correlation sequence has no discernible peak. The PAR curve for Scenario 4 closely mimics the results of Capt. Falen's research. The PAR value increases to a peak for the 3 dB threshold and steadily decreases with increasing threshold, as did Capt. Falen's plots. One difference between these plots and Capt. Falen's is the peak PAR value. His best PAR value occurred between 6 and 9 dB of threshold; whereas, the best PAR value for these simulations occurred with a 3 dB threshold. The difference might be attributed to the smaller cross correlations performed in this research, and should be checked using the same

correlation lengths in Captain Falen's research. Figure 26 contains the probability of bit error

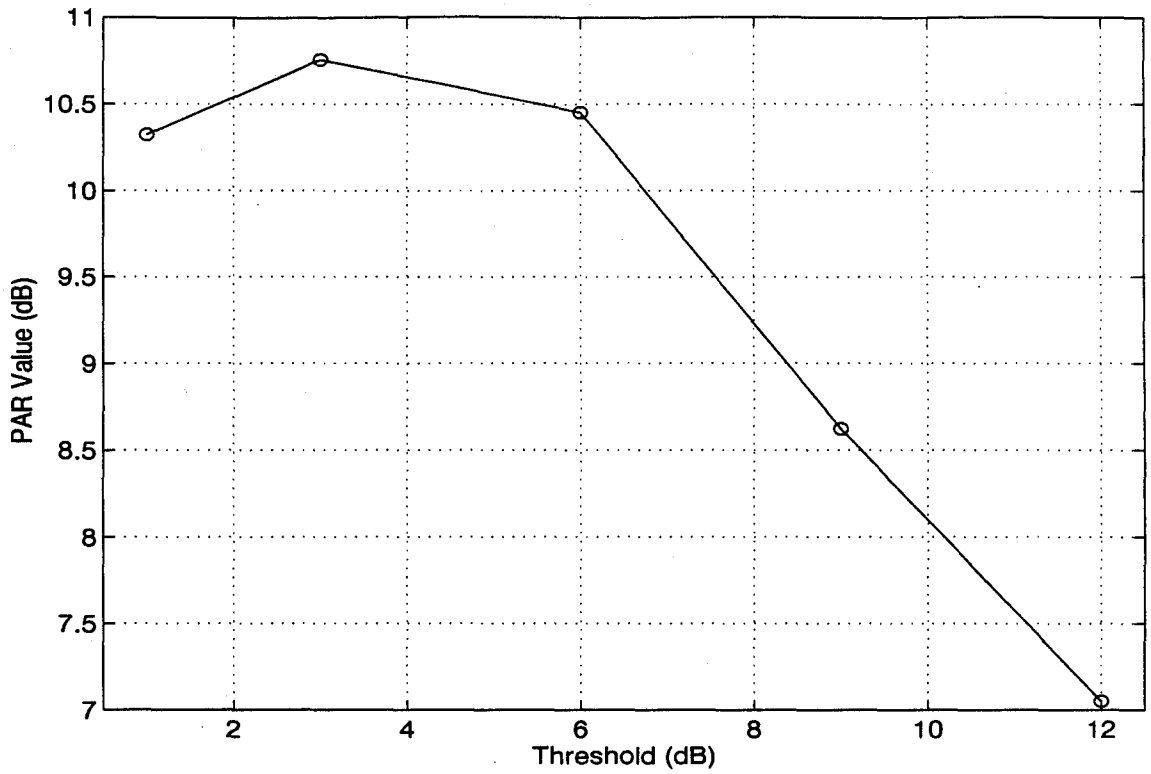


Figure 25. PAR Curve for Scenario #4

rate curves for Scenarios 10 and 11. Without the DEF, the simulated bit error rate was 0.4 on average. With the DEF in place, the BER closely modeled the simulated baseline. With the additional excised spectra, the probability of bit error increased to more appropriate levels. Upon inspection, it can be seen that the BER curves with the DEF in Figure 24 are identical and the BER curves with the DEF in Figure 26 are identical. This is because the jammer spectra in each case are identical. For one CW jammer at the center frequency, the spectrum is a spike with minimal width. As the jammer is offset from the center, the spectrum increases its width, but, for the jammer power used in these simulations, the spectrum for 33 dB and 53 dB J/N are similar in width. Thus, identical portions of the spectrum were excised in each case. The receiver demodulates identical signals to determine the bit error rate.

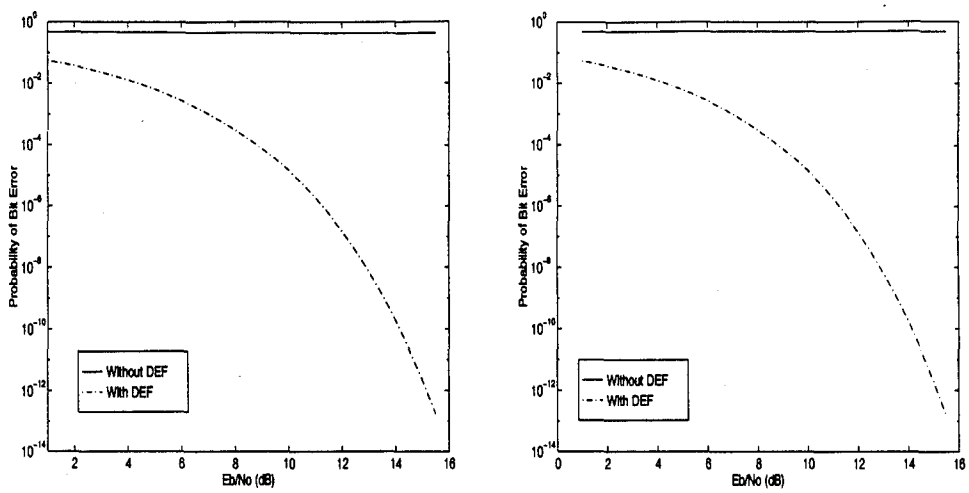


Figure 26. BER Curves For 4 CW Jammers: (a) 33 dB J/N (b) 50 dB J/N

4.4 Pulsed Jammers

4.4.1 One Pulsed Jammer. The pulsed CW jammer was placed directly on the center frequency like the CW jammer. A 50% duty cycle was used since that was reported to be the most damaging setting (7). The pulse repetition frequency (PRF) was set to match the PRF to sampling ratio, 0.00005, used in Capt. Falen's research. However, this setting would have been too low and would have been very close to a CW jammer. So, a ratio of 0.001 was used, which provides a PRF of 20 Hz. Table 4 contains the PAR values for Scenarios 6 through 9. Scenario 8 examined PAR value behavior over an extended range of thresholds. Comparing Scenario 6 to Scenario 2, it can be seen that the PAR value is affected by the pulsed CW jammer more so than the CW jammer. This was expected since the pulsed CW jammer has significant sidelobes that increase with jammer power. Further examination of Table 4 reveals that the PAR value steadily declines with increased jammer power due to the aforementioned reason. Figure 27 illustrates the PAR value behavior for the various threshold settings for Scenario 8. The same trend exists as before. The PAR value peaks at a 3 dB threshold and decreases thereafter. Scenario 9 registered low PAR values indicating the

Table 4. PAR Values For 1 Pulsed CW Jammer

Scenario	*PAR Value*	*J/N (dB)*	*Threshold (dB)*
6	11.0373	33	6
6	10.9758	33	9
7	10.9272	36	6
7	10.8661	36	9
8	9.7200	50	1
8	10.1142	50	3
8	10.0277	50	6
8	9.4615	50	9
8	9.0711	50	12
9	8.2237	53	6
9	7.1962	53	9

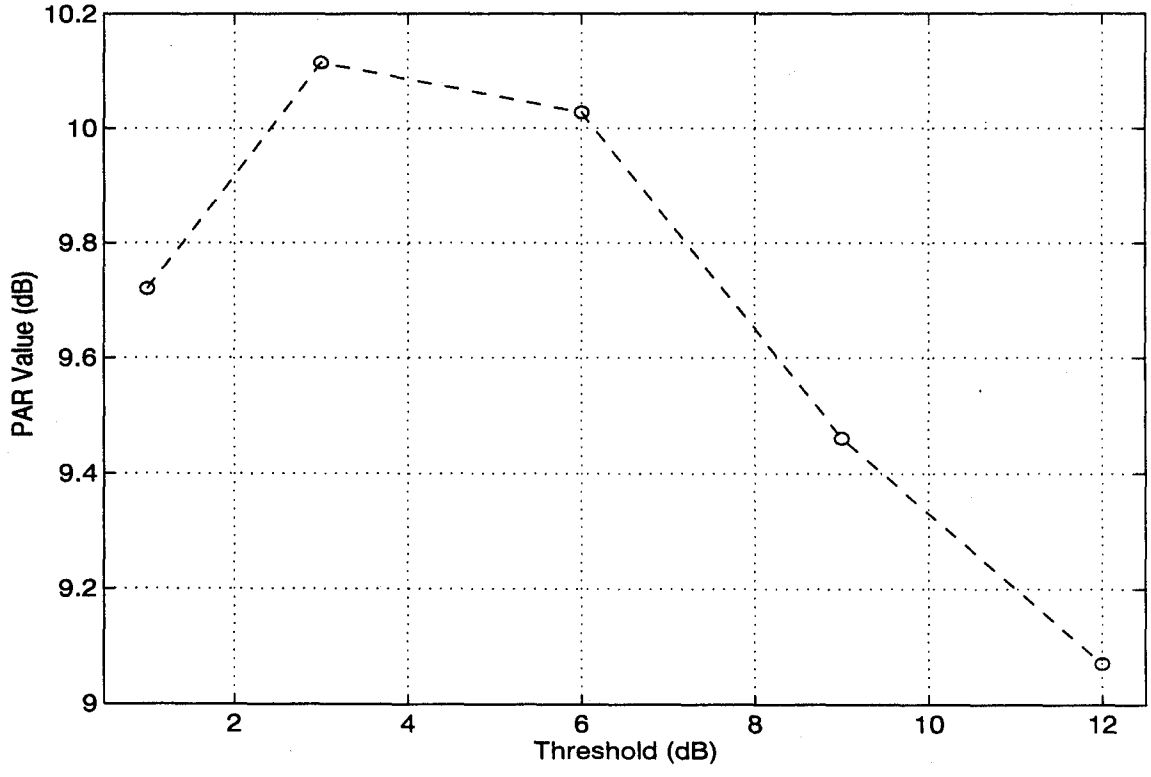
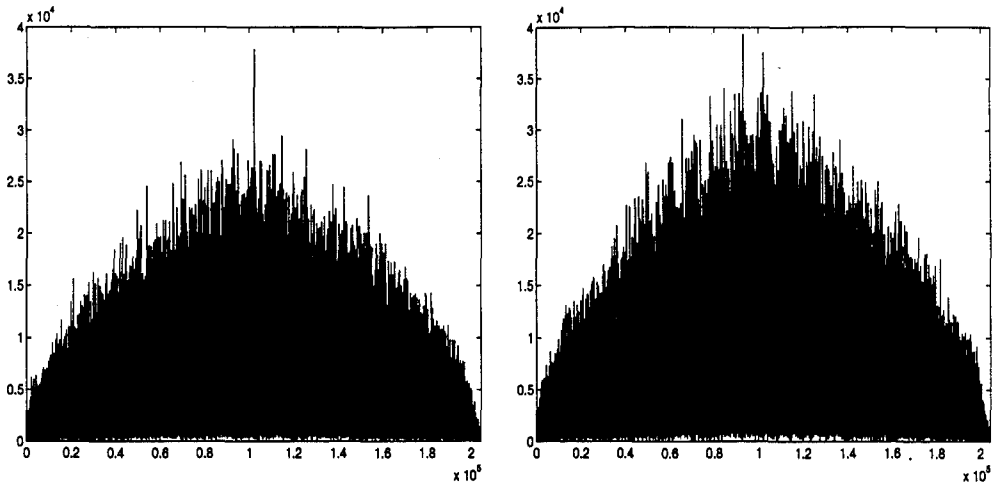


Figure 27. PAR Curve for Scenario #8

autocorrelation receiver would have a difficult time acquiring synchronization of the signal. Figure 28 contains the cross correlation sequences for Scenario 9. The first correlation in



(a) (b)

Figure 28. Cross Correlations For 1 Pulsed CW Jammer: (a) 53 dB J/N with 6 dB Threshold (b) 53 dB J/N with 9 dB Threshold

Figure 28(a) has a distinct peak at the proper location, but other peaks occur within 2 dB of the main spike. These other peaks could pose a problem during acquisition. The second correlation in Figure 28(b) contains two peaks with the main peak not occurring at the proper time. This condition will clearly cause acquisition problems for the autocorrelation receiver. Figure 29 shows the BER curves for one pulsed CW jammer. Recall, the flat lines represent the BER without the DEF and the dashed lines represent the BER with the DEF. For the single pulsed CW jammer at the center frequency, more power translates to an increased BER degradation because of the sidelobe properties of the pulsed jammer. Hence, the BER curves are not identical as was the case with the CW jammer case.

4.4.2 Four Pulsed Jammers. The four pulsed jammers were offset in the same manner as the four CW jammers. Table 5 contains the PAR values for Scenarios 10 through 12. It is clear that the four pulsed jammers overwhelmed the DS BPSK system. The PAR values

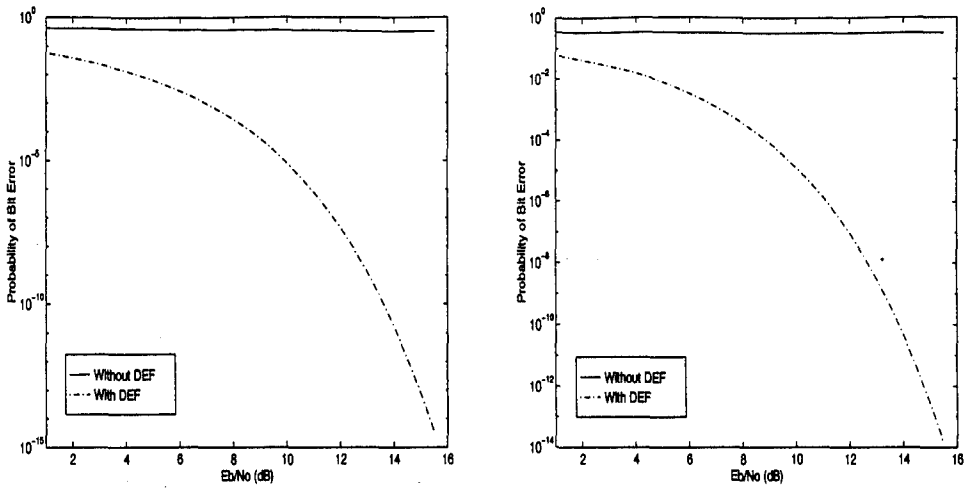


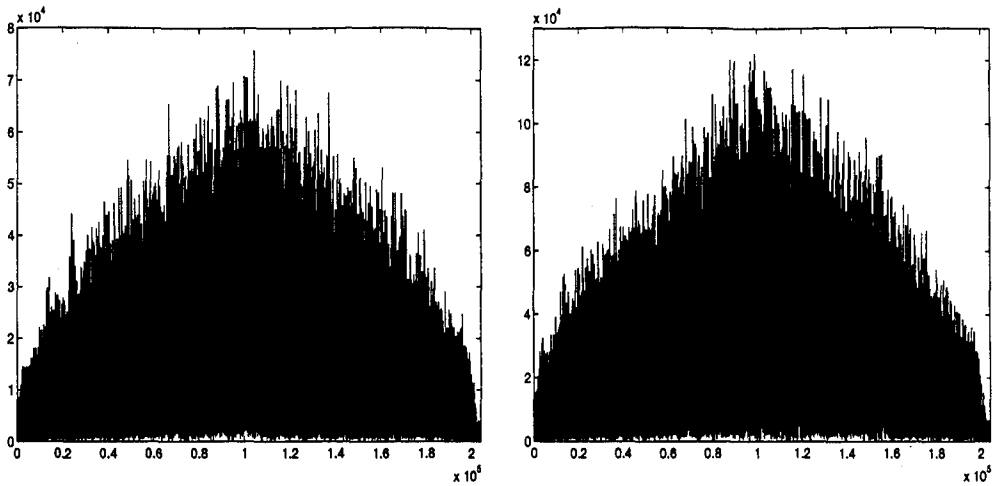
Figure 29. BER Curves For 1 Pulsed CW Jammer: (a) 33 dB J/N (b) 50 dB J/N (Note the scale difference)

decrease significantly after Scenario 10. Figure 30 shows the cross correlation sequences for

Table 5. PAR Values For 4 Pulsed CW Jammers

Scenario	*PAR Value*	*J/N (dB)*	*Threshold (dB)*
10	10.6651	33	6
10	9.5778	33	9
11	7.5039	50	6
11	5.7370	50	9
12	5.3276	53	6
12	4.7867	53	9

Scenario 12. These plots support the notion that the system has been overwhelmed since there is no distinct peak within either plot. Figure 30(a) contains the peak spike at the correct position, but there are numerous other spikes within 1 dB of the main spike which could cause acquisition problems for the receiver. Figure 30(b) has no distinct peak. Numerous peaks occur in the middle of the correlation that would prevent the receiver from acquiring the signal. Figure 31 presents the BER curves for Scenario 10 and 11. Without the DEF the system is



(a) (b)

Figure 30. Cross Correlations For 4 Pulsed CW Jammer: (a) 53 dB J/N with 6 dB Threshold (b) 53 dB J/N with 9 dB Threshold

reduced to a BER of 0.45 on average. With the DEF, the system operates as effectively as it did against 4 CW jammers at a 33 dB J/N. At a 53 dB J/N, the system performance degrades drastically even with the DEF. However, the benefit of using the DEF is clearly displayed in this figure.

4.5 Broadband Noise Jammer

For the broadband noise jammer, a noise source was added to the channel instead of the jammer block. This method resulted in jamming the entire main lobe of the spread spectrum signal with noise. Table 6 contains the PAR values obtained in Scenarios 13 and 14. Scenario 14 examined the PAR value behavior over an extended range of threshold values. The broadband noise jammer caused severe degradation of the PAR value at 6 and 9 dB J/N. The PAR value dropped 3 or more dB from the baseline PAR value presented earlier. Figure 32 plots the PAR values for Scenario 14. Again, the peak PAR occurs at a 3 dB threshold. However, the PAR levels off for increasing threshold. This makes sense since

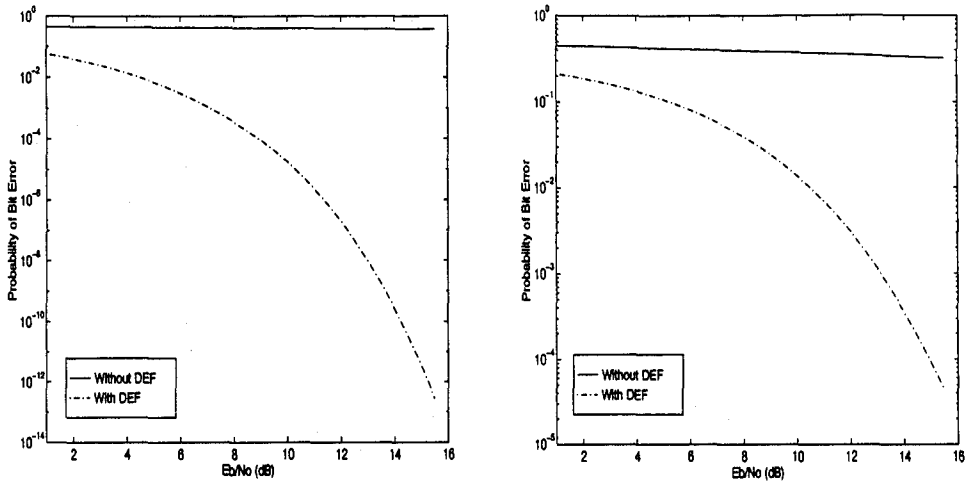


Figure 31. BER Curves For 4 Pulsed CW Jammers: (a) 33 dB J/N (b) 50 dB J/N (Note the scale difference)

Table 6. PAR Values For Broadband Noise Jammer

Scenario	*PAR Value*	*J/N (dB)*	*Threshold (dB)*
13	8.4436	6	6
13	8.4319	6	9
14	6.8781	9	1
14	7.5779	9	3
14	7.5436	9	6
14	7.5457	9	9
14	7.5457	9	12

only noise is added to the spectrum and nothing is excised by the filter. The noise jammer simply increases the average noise value calculated by the DEF. So, when the threshold is set to 9 dB or higher, there is no signal exceeding the threshold, which causes the PAR value to level off. Figure 33 contains plots of the cross correlation sequences for several runs of

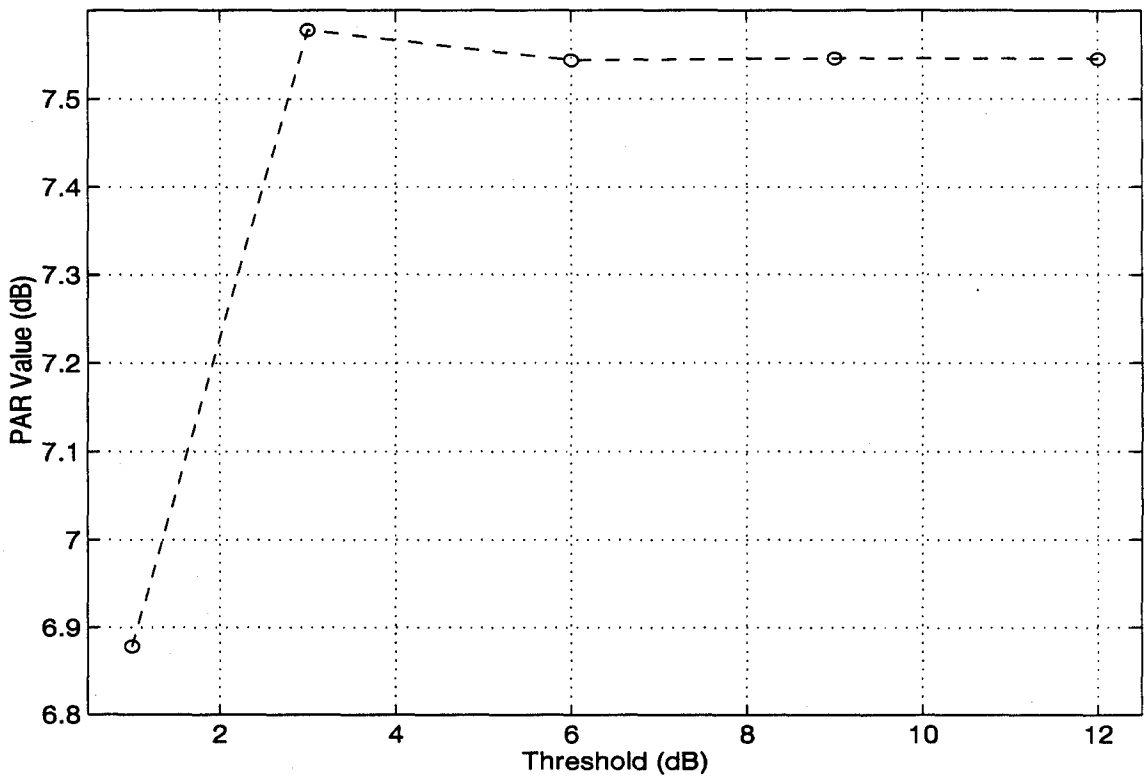


Figure 32. PAR Curve for Scenario #14

Scenarios 13 and 14. Figure 33(a) and (b) show that the peak correlation spike was easily distinguishable for 6 dB J/N. The PAR values for these conditions are nearly identical, with only 0.089 dB difference. Additionally, the plots are identical indicating that these settings resulted in identical processing by the DEF. Figure 33(c) and (d) show that the correlation sequences were affected significantly by the increased J/N ratio. The main correlation peak was surrounded by many extraneous correlation peaks that were within 1 dB or less. Such peaks would prevent the autocorrelation receiver from acquiring the signal. Figure 33(e) and (f) contain the correlation sequences for 9 dB J/N, but with thresholds set to 1 and 3 dB. At a

1 dB threshold, it is clear that the correlation receiver would not acquire the signal. The main peak does not occur at the proper position and its magnitude was nearly halved. At a 3 dB threshold, the main peak occurs at the proper position, but there are other extraneous peaks close enough in magnitude to cause the autocorrelation receiver problems during acquisition. Bit error rate curves were not provided for these scenarios since the effect of the noise jammer would be to decrease the E_b/N_O by increasing the noise power in the simulation. This would simply result in moving the BER curve upwards with respect to the y-axis, probability of bit error.

4.6 Percent Jammer Power Removed

The third metric of interest in determining the performance of the DEF is the percentage of jammer power removed from the incoming signal. For the CW jamming cases, it was expected that the DEF would excise 100% of the jamming signal. The CW jammer occupied a set number of frequency bins, which are excised, resulting in all the jammer power being removed. For the pulsed CW jammer cases, it was expected that the DEF would excise nearly all the power of the jammer since the pulsed jammer has significant sidelobes. However, the effect of the sidelobes depends upon the duty cycle and pulse repetition frequency. Table 7 contains the results of running the simulations to determine the amount of jammer power removed. For the CW jammer cases, all of the jammer power was removed. For the single pulsed CW cases, all the jammer power was removed as well. This was due to the fact that the jammer was placed directly on the center frequency, which limited the effect of the sidelobes. When four jammers were used, the percentage of jammer power removed decreased a little over 1%. The decrease in removed power was expected to be greater. However, the duty cycle and PRF settings made the pulsed jammer resemble a CW jammer in that its spectra was not a wide as it would have been if the duty cycle were decreased and the PRF increased. Thus, the sidelobes were not as significant as they could have been.

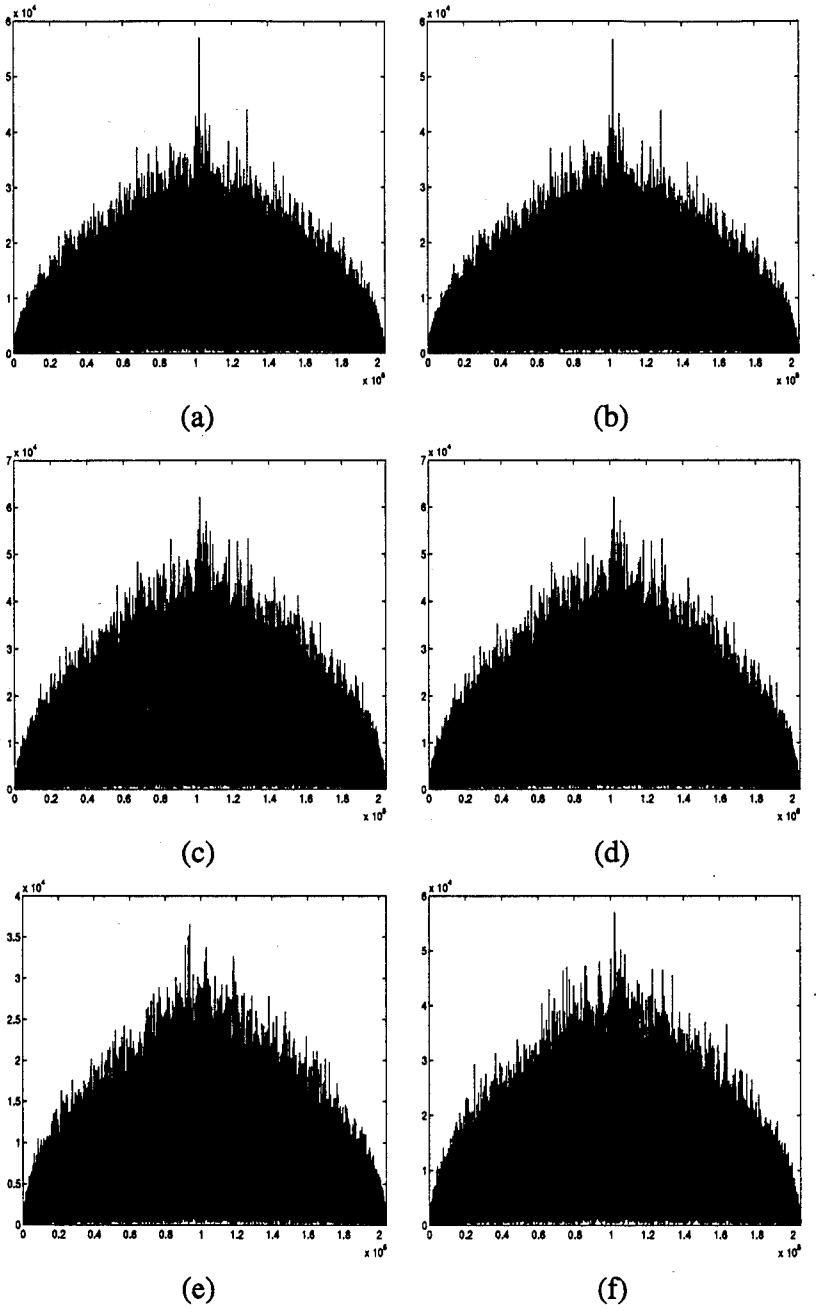


Figure 33. Cross Correlations For Broadband Noise Jammer: (a) & (b) 6 dB J/N with 6 & 9 dB Threshold (c) & (d) 9 dB J/N with 6 & 9 dB Threshold (e) & (f) 9 dB J/N with 1 & 3 dB Threshold

Table 7. Percent Jammer Power Removed

Scenario	*Percent Jammer Power Removed*
2	100
3	100
4	100
5	100
6	100
8	100
10	98.45
11	98.43

4.7 Summary

This chapter provided the results of the SPW simulations. Scenarios were discussed for single and multiple CW jammers; single and multiple pulsed CW jammers; and for a broadband noise jammer. Peak-to-Average correlation values were presented that correlated with the results presented by Capt. Falen. Bit error rate curves were presented to determine how the DEF affects the performance of the DS-BPSK system under the different jamming scenarios. Finally, the percentage of jammer power removed from the incoming signal was presented.

V. Work Completed On DEF Hardware

5.1 Introduction

This chapter provides a chronological description of the work completed on the DEF hardware until June 1995 when the JPO was briefed. By direction of the JPO, Raytheon mailed all materials to AFIT in 1994. AFIT received the hardware in August 1994 and performed an inventory. AFIT received two circuit card assemblies (CCAs), a wire-wrapped test box, bench stock, and wiring diagrams for the CCAs. Notably missing from the material were specification sheets for the chips used in the design, and documentation outlining the tests that had been completed by Raytheon. It became painfully clear that an uphill battle would be fought. Mr. Dick Miller and I gathered as much information as we could from various local vendors. However, the unenviable task of troubleshooting the DEF hardware with minimal supporting documentation was started. The following list summarizes the equipment used to generate and measure signals and voltages, and to remove and reconnect ICs.

1. Hewlett Packard 8566B Spectrum Analyzer
2. Tektronix High Impedance Oscilloscope
3. Hewlett Packard 1631D Logic Analyzer
4. Hewlett Packard 3478A Multimeter
5. Hewlett Packard 8643A Synthesized Signal Generator
6. New Wave Instruments LRS100 Spread Spectrum Generator
7. DC Power Supplies (± 5 V, ± 15 V, -5.2 V)
8. Soldering Iron
9. Circuit Card Holder

5.2 DEF Hardware

As stated in Chapter 1, a deliberate, methodical approach was used to troubleshoot the DEF. Basically, the signal path was stepped through one device at a time with the signal input to the test box as the starting point. The test box is an aluminum alloy base structure made up of the following subsystems:

1. Radio Frequency to Baseband Conversion
2. Analog to Digital Conversion
3. DEF
4. Digital to Analog Conversion
5. Baseband to Radio Frequency Conversion

Prior to performing any work on the DEF, the first two subsystems had to be operating correctly. Troubleshooting the hardware progressed as follows: RF to baseband conversion, the A/D conversion, the baseband to RF conversion, and, finally, the DEF. The D/A conversion was never tested since the DEF CCA never functioned. The remaining subsections describe the work completed on the individual subsystems.

5.2.1 RF to Baseband Conversion. The incoming GPS signal is first passed through a broadband filter, $10 - 2000\ MHz$, and then passed through another filter centered at $1575\ MHz$. An onboard crystal oven provides the local oscillator with $10.23\ MHz$. The local oscillator generates $20.46\ MHz$, $1401\ MHz$, and $173.91\ Hz$ signals for use in various portions of the test box. The filtered GPS signal is then mixed with the $1401\ MHz$ signal to convert it down to baseband. The signal is then split into in-phase and quadrature components, which are then conditioned prior to being passed to the A/D converter.

After acquiring the necessary power supplies, the signal path was traced through the first subsystem. The system was tested by injecting a sinusoid at $1575\ MHz$. All signal

measurements were as expected until the output of the signal conditioning cards was measured. These cards filtered the real and imaginary parts of the incoming signal for use in the next subsystem. As a result of examining these cards, it was determined that a couple of cold solder joints were responsible for the poor signals. These connections were resoldered, which fixed the problem. The RF to baseband conversion did not have many problems in operation. The difficulty lied in figuring out what the designer had in mind and how that idea was implemented, which can be said of the entire troubleshooting process. Another time consuming process encountered, prior to troubleshooting anything, dealt with obtaining all the necessary equipment. Mr. Dick Miller's ability to scavenge equipment from the far corners of AFIT was greatly appreciated in this end.

5.2.2 A/D Conversion. For the A/D conversion, a Comlinear CLC935/936 EVAL board is used. The output from the signal conditioning cards is buffered by a Motorola 10114 clock buffer, before entering the A/D chip, which is the Comlinear CLC936AC. The local oscillator provides the 20.46 MHz sampling frequency to the A/D. The CLC936 is a high-speed 12-bit analog-to-digital converter with the quantizer, track-and-hold, and references all contained on the chip. The only signals required for operation are the analog signal, power, and the clock signal. The digital output is then passed to the DEF CCA.

Upon first examination of the A/D card, it was noticed that a large piece of semi-rigid coaxial cable was missing. A new cable was constructed and attached to the board. Initial testing also indicated a bad connection from a signal conditioning card to the A/D. After a new connector was installed, the input signal was measured on the A/D board. It was also noticed that the CLC936 ran at a very hot temperature, which was higher than expected. The output of the card was examined on the HP Logic Analyzer, which showed that no digital data came out of the board. After probing the CLC936 and the input buffering circuit, it was determined that the input buffer was not functioning. Since the buffer was surface mounted,

the chip had to be either removed and a new one resoldered or a new A/D card had to be built up from the stock. The first option was chosen because building up an extra board had a few more problems associated with it that could not be handled with the present equipment. After re-attaching a new 10114, the card output valid digital data as evidenced on the HP Logic Analyzer.

This digital data is then supposed to pass to the DEF CCA. After careful scrutiny, it was determined that the data lines were wire wrapped to incorrect ports, and that no common ground was passed to the DEF. At this point, a decision was made to bypass the wiring problems and move to the baseband to RF conversion.

5.2.3 Baseband To RF Conversion. This subsystem recombines the real and imaginary data output from the D/A card. The signal is then passed through a seven section attenuator, which is programmed to ensure that the system provided unity gain. Then, the signal is modulated to 1575 MHz and bandpass filtered. This signal is then passed on to the GPS receiver.

To test this section, the RF to baseband was hardwired to the baseband to RF conversion and a sinusoid was injected. The output signal was measured using the HP 8566, which showed that the output was an identical attenuated sinusoid. The A/D and D/A sections were not tested because the correct input information to the D/A was not available. In summary, the test box correctly functioned for the RF to baseband, A/D, and the baseband to RF conversions. It was intended to examine the D/A after troubleshooting the DEF.

5.2.4 Digital Excision Filter Circuit Card Assembly. At the time of delivery the DEF CCAs were stated to be 90% and 80% complete. It was obvious from the condition of the test box that the boards were never tested or powered up. The primary CCA seemed to have all of its chips, resistors, etc. on board, but the second CCA only had some of the surface

mounted chips attached. The condition of the test box and the state of the boards indicated that many problems with the DEF had not yet been identified.

As with the other subsystems, troubleshooting the DEF CCA followed the signal path through the card. Figure 34 illustrates the data signal flow in a block diagram. The output

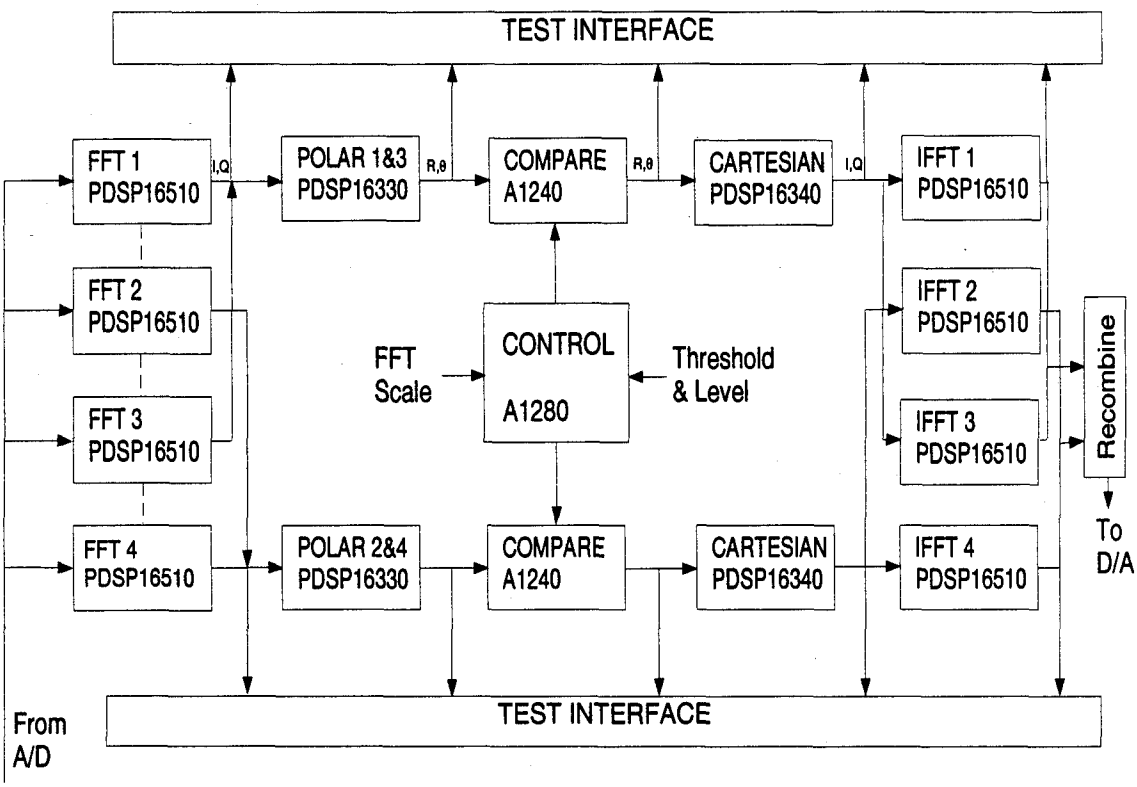


Figure 34. Data Flow Through DEF Circuit Card Assembly

from the A/D converter card enters the DEF CCA, where it undergoes an emitter-coupled logic (ECL) to transistor-transistor logic (TTL) conversion. The A/D subsystem implemented ECL, whereas the remaining devices in the DEF used TTL. The conversion is accomplished using a Motorola 10125 ECL to TTL translator. The converted TTL signal is used to drive clock doubling circuitry that supplies the 10 and 20 MHz signal to the rest of the devices on the CCA. The digital data is passed to Plessey PDSP16510A FFT chips. These are stand alone chips that perform forward or inverse fast Fourier transforms on real or complex data

containing up to 1024 points (8). Additionally, the chip can be programmed for 0, 50, or 75% overlapping data blocks and for applying either a Hamming or Blackman-Harris window to the input. It is necessary to provide the FFTs with a control word, which is accomplished with National Semiconductor ABT16244 buffer/line drivers. The initial state is set up by selecting settings on dual-in line pin (DIP) switches. From the FFT chips, the data is passed to a Plessey PDSP16330 Pythagoras Processor. The 16330 is a high-speed digital CMOS IC that converts real and imaginary data to magnitude and phase at rates up to 20 MHz (8). The polar data is passed to the threshold comparison and excision section, which is implemented with field programmable gated arrays (FPGAs). For this design, ACTEL 1280 and 1240 FPGAs are used, which have 8000 and 4000 devices respectively. The 1280 uses the scale of the FFT and threshold to control the 1240, which performs the compare and select operation to excise the proper frequency bins. After the excision takes place, the polar data is passed to a Plessey PDSP16340 Polar to Cartesian converter, which operates at 20 MHz (8). The real and imaginary data is then inverse transformed by the Plessey PDSP16510A. Once the data is recombined, it is passed out of the DEF CCA to the D/A converter.

When power was first applied to the DEF CCA, a major problem was immediately recognized as the meter on the power supply increased by 2.5 amps. The problem area was readily identified because of the intense heat radiating off of the National Semiconductor ABT16244 16-bit buffer/line driver. Several days were spent probing the chip and surrounding areas for short and open circuits, but no clear indications of either were measured. Finally, the chip was removed with the assistance of Wright Labs AAW. The power consumption problem was solved, but it was still unclear as to whether the 16244 was the cause, or if it was caused by another chip down line, or a wiring problem within the board itself. At the present time another 16244 has not been re-attached to the board. To accomplish this, a surface mount soldering station would be required, or assistance from another office/lab that has the capability to perform this task.

Continuing along the signal path brought us to the ECL-to-TTL clock conversion circuitry. As the pins of the 10125 were probed, it was determined that no output came from the chip. When viewed on the oscilloscope, the output resembled noise instead of a clock signal. The input signals, power, and ground were all correctly input to the chip. Since all inputs were being received and no signal was output, the chip was removed from the CCA. Prior to re-attaching a surface mounted chip, larger holes were drilled into the board to fit a socket. The TTL clock is used to drive the clock doubling circuitry, which supplied the necessary clock signals to other chips. The ECL-to-TTL conversion together with the clock doubling was successfully recreated on a daughter board. At that point, the GPS JPO asked for a status briefing, which resulted in stopping work on the hardware and switching to the simulation. Work on the DEF CCA did not progress very far. In fact, these problems were encountered within the first section of the card. None of the larger chips, such as the FFTs and FPGAs, were tested. At this time, it is not known whether the FPGAs were ever programmed. No fuse maps or code for the FPGAs were provided in the delivered documentation. So, the DEF CCA was not completed and, by direction of the GPS JPO, all work on the hardware was stopped.

5.3 Summary

This chapter summarized the work completed on the hardware portion of this thesis. The hardware task was sectioned off into specific subsystems that were analyzed, and in most cases, fixed. The sheer number of problems and unavailability of information concerning the design of the hardware prompted the shift in focus of this research from the hardware to the simulation effort.

VI. Conclusions and Recommendations

6.1 Summary

This thesis analyzed the effects of narrowband interference rejection on the bit error rate using the DEF. The fourteen scenarios used in this research were chosen from the larger set of scenarios performed by Capt. Falen. These jamming scenarios typically represented the worst case results from his research. Most scenarios examined used Capt. Falen's recommended threshold level, 6 to 9 dB above the noise floor (7). The remaining simulations examined the PAR value behavior with the threshold varying from 1 to 12 dB. This research then extended Capt. Falen's work by utilizing SPW to provide probability of bit error curves for selected scenarios. Monte Carlo simulations for BER are not recommended because of the long computer run-times involved. Typically, run-times exceeded four days when actual system parameters were used, providing only one point on the BER curve. The Semi-Analytic Error estimate should be used to determine BER because the run-time is reduced to a few hours, the error estimates match those achieved with the MC simulations, and the necessary conditions to use the SA method are met. The BER and percentage of jammer power removed results achieved in this research clearly illustrate the performance improvement provided by using the DEF.

6.2 Conclusions

Although Capt. Falen's best threshold was 6 to 9 dB, the best threshold level for these simulations was 3 dB. As discussed in Chapter 4, this discrepancy was due primarily to the fact that these simulations used a smaller section of P-code, and, hence, used a smaller correlation period. Additionally, these simulations differed with respect to processing gain, sampling frequency, and data rate. These differences led to slightly different results for PAR

values. In SPW, it was not possible to use system parameters that exactly matched those of Capt. Falen because of disk space, memory, and run-time limitations.

The percentage of jammer power removed results indicate that the DEF is highly successful dealing with single and multiple CW jammers in the environment. Single and multiple pulsed CW jammers with different frequency offsets, duty cycles, and PRFs, would most likely decrease the amount of jammer power removed more significantly than the ones used in this research. However, under these conditions, the DEF showed that over 98% of the jammer power was removed from pulsed CW jammers.

Examining the BER curves leads to the conclusion that the DEF provides a dramatic performance improvement in a hostile environment. Without the DEF in place, the system was rendered useless as evidenced by the BERs of 0.30 to 0.50. Implementation of the DEF resulted in BER curves that approximated the baseline BER curve in the CW and single pulsed CW jamming scenarios, and in Scenario 10. For the remaining multiple pulsed CW jamming scenarios, the BER was reduced, especially in Scenario 11. However, this reduction is offset by the fact that system still provided acceptable probability of bit errors with four jammers with a $50\,dB$ J/N ratio, which significantly exceeds the capability of the original system. Thus, the DEF has been shown as a viable capability to remove narrowband interference even when subjected to the worst case scenarios in this research.

Finally, the Signal Processing Worksystem allowed this research to extend the previous work accomplished by Captain Falen. Although many problems occurred when using SPW, it has proven to be an excellent simulation tool that will allow the design of the Digital Excision Filter to be constructed on a single chip in the near future. Additionally, SPW should be incorporated into more of the AFIT Communications curriculum.

6.3 Recommendations

Further research should be conducted using the same jammers with different parameters, and with additional jammers, such as a swept frequency, pulsed noise, spot noise, and frequency hopping jammers. These jammers should be set to simulate real world threats expected to act against the Global Positioning System. Once the final design of the DEF is set, it should be converted by a Digital Thesis student to implement in the Very High Design Language (VHDL) translator. This process would lead to the DEF being written in VHDL so that the DEF can be realized in hardware. It is also recommended to investigate the effect of increasing the FFT size to 1024 points. This would allow for better frequency localization of the jammers, and may affect the bit error rate.

The reason for using MATLAB and SPW to obtain PAR values in this research is because of SPW's inability to provide more than one PN sequence from the PN generator. By building a Gold code generator, or a P-code generator, SPW could be used to calculate everything from bit error rate to PAR values.

Appendix A. C-Code for Excisor Block

A.1 Excisor.c

```
#include "spw_platform.h"
#ifdef UNIX
#include "caedata/lascody/excisor/blockcode/excisoru.c"
#else
#ifdef VAX_VMS
#include "[caedata.lascody.excisor.blockcode]excisoru.c"
#endif VAX_VMS
#endif UNIX
static char *REVISION = "2.50";

/*
 *
 * Block Function: excisor
 * Library: lascody
 * Date: Mon Jul 31 16:42:40 1995
 *
 */


/************************************************************************/
/*                                                                      */
/*    FEED_THROUGH_LIST INFORMATION:                                    */
/*                                                                      */
/*    --> FEED_THROUGH_TYPE IS NOT EDITABLE. The BDE parameter          */
/*        screen associated with the block must be edited to change     */
/*        the block's FEED_THROUGH_TYPE.                                */
/*                                                                      */
/*        FEED_THROUGH_TYPE = ALL_FEED_THROUGH.                         */
/*                                                                      */
/************************************************************************/

/************************************************************************/
/*                                                                      */
/*    LINK_OPTIONS INFORMATION:                                         */
/*                                                                      */
/*    --> The LINK_OPTIONS list is editable.  It contains all the       */
/*        libraries which the code must be linked to.  Each item in     */
/*        the list must be surrounded by double quotes and              */
/*        separated by commas.  The math library is automatically       */
/*        linked, and does not need to be specified.  The paths         */
/*        may be specified as full paths or as paths relative to        */
/*        the host.                                                     */
/*        A link option can also be specified in the form "-lx"         */
/*        (where x is defined in the UNIX manual on "ld"                */
/*                                                                      */
/*        IMPORTANT:  The entire LINK_OPTIONS list must be deleted      */
/*        if it doesn't contain any elements.                           */
/*                                                                      */
/*        Sample LINK_OPTIONS list:                                     */
/*        (Actual list should be placed below this comment block)       */
/*                                                                      */
/*        LINK_OPTIONS = { "-lm",                                       */
/*                         "//host/code/lib/sample.a" };                */
/*                                                                      */
/*                                                                      */
/************************************************************************/
```

```
/*********************************************************************/
/*                                                                   */
/*    INCLUDE_DIRS INFORMATION:                                      */
/*                                                                   */
/*    --> The INCLUDE_DIRS list is editable.  The list should        */
/*        contain all directory search paths needed to locate all    */
/*        the include files used by this block.  It has the same     */
/*        format as the LINK_OPTIONS list.                           */
/*                                                                   */
/*        IMPORTANT:  The entire INCLUDE_DIRS list must be deleted   */
/*        if it doesn't contain any elements.                        */
/*                                                                   */
/*        Sample INCLUDE_DIRS list:                                  */
/*        (Actual list should be placed below this comment block)    */
/*                                                                   */
/*        INCLUDE_DIRS = { "//host/u/code/include",                  */
/*                         "//host/lib/dir" };                       */
/*                                                                   */
/*                                                                   */
/*********************************************************************/

/*********************************************************************/
/*                                                                   */
/*      EDITABLE FUNCTIONS                                           */
/*                --> In_excisor_lascody ()                          */
/*                --> Ro_excisor_lascody ()                          */
/*                --> Te_excisor_lascody ()                          */
/*                                                                   */
/*      Structure use:                                               */
/*        Typical input value reference                              */
/*          local_var = *(spb_input->var_name);                      */
/* **OR**    local_var = I_var_name;     */
/*        Typical output value update                                */
/*          spb_output->var_name = local_var;                        */
/* **OR**    O_var_name = local_var;     */
/*        Typical parameter reference                                */
/*          local_var = spb_parm->var_name;                          */
/* **OR**    local_var = P_var_name;     */
/*                                                                   */
/*      (See reference manual for further information)               */
/*                                                                   */
/*********************************************************************/

/*
 *      Initialize Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to initialize the state structure
 *           and constant outputs of the block.  It is called once
 *           for each block instance during simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *           or SYS_FATAL by using the return() function.
 *           User may modify the line containing "return(SYS_OK);".
 */

In_excisor_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_excisor_lascody *spb_parm;
STRUCT It_excisor_lascody *spb_input;
STRUCT Ot_excisor_lascody *spb_output;
STRUCT St_excisor_lascody *spb_state;
{
```

```
return (SYS_OK);
}

/*
 *      Run Output Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to update the outputs and/or state
 *          of the block.  It is called each iteration, for each
 *          block instance during simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *          or SYS_FATAL by using the return() function.
 *          User may modify the line containing "return(SYS_OK);".
 */

Ro_excisor_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_excisor_lascody *spb_parm;
STRUCT It_excisor_lascody *spb_input;
STRUCT Ot_excisor_lascody *spb_output;
STRUCT St_excisor_lascody *spb_state;
{
 int i,n;
  double mag, avg_mag, Threshold;

/* This section calculates the average noise floor of the real and
 * imaginary FFT data.  Then the new threshold is set by adding the
 * desired excision threshold to the average noise level.
 */

   mag = 0.0;
   for (n=0; n< I_FFT_real_iovec_len; n++) {
    mag+=sqrt((I_FFT_real[n]*I_FFT_real[n])+(I_FFT_imag[n]*I_FFT_imag[n]));
   }
   avg_mag = mag/(double)I_FFT_real_iovec_len;
   Threshold = avg_mag*pow((double)10.0,(double)(I_Threshold*0.1));
   O_Avg_Noise = Threshold;

/* This section compares the magnitude of the FFT to the threshold
 * and sets it to zero if the magnitude is above the threshold.
 */

 for (i=0; i < I_FFT_real_iovec_len; i++)
  {if (sqrt(I_FFT_real[i]*I_FFT_real[i]+I_FFT_imag[i]*I_FFT_imag[i])
>= Threshold)
      {  O_Excised_real[i]=0.000000000000000001;
         O_Excised_imag[i]=0.000000000000000001; }
       else {
         O_Excised_real[i]=I_FFT_real[i];
         O_Excised_imag[i]=I_FFT_imag[i]; }}


return (SYS_OK);
}

/*
 *      Termination Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to dump the final state of the
 *          block.  It is called once for each block instance
 *          during the simulation.
 *
```

```
 *      Function must always return either SYS_OK, SYS_TERM,
 *          or SYS_FATAL by using the return() function.
 *          User may modify the line containing "return(SYS_OK);".
 */

Te_excisor_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_excisor_lascody *spb_parm;
STRUCT It_excisor_lascody *spb_input;
STRUCT Ot_excisor_lascody *spb_output;
STRUCT St_excisor_lascody *spb_state;
{

return (SYS_OK);
}

/*************************************************************************/
/*                                                                       */
/*      Add any additional functions you need here.                      */
/*                                                                       */
/*************************************************************************/
```

A.2 Excisor.h

```
#include "FBCDEFS.h"

/*
 *
 * Block Function: excisor
 * Library: lascody
 * Date: Mon Jul 31 16:42:40 1995
 *
 */

/*************************************************************************/
/*                                                                       */
/*      EDITABLE USER DEFINED STATE STRUCTURE                            */
/*                --> STRUCT St_excisor_lascody                          */
/*                                                                       */
/*************************************************************************/

/*
 * State Structure (User Defined, editable)
 */
STRUCT St_excisor_lascody {
int instance;
};

/*************************************************************************/
/*                                                                       */
/*      UNEDITABLE SIMULATOR DEFINED STRUCTURES                          */
/*                --> STRUCT Pt_excisor_lascody                          */
/*                --> STRUCT It_excisor_lascody                          */
/*                --> STRUCT Ot_excisor_lascody                          */
/*                                                                       */
/*************************************************************************/
/*
 * Parameter Structure, Simulator Defined, uneditable
 */
STRUCT Pt_excisor_lascody {
double Threshold;
double initial_value;
```

```
};

/*
 * Input Structure, Simulator Defined, uneditable
 */
STRUCT It_excisor_lascody {
long FFT_imag_iovec_len;
double *FFT_imag;
long FFT_real_iovec_len;
double *FFT_real;
double *Threshold;
};

/*
 * Output Structure, Simulator Defined, uneditable
 */
STRUCT Ot_excisor_lascody {
double Avg_Noise;
long Excised_imag_iovec_len;
double *Excised_imag;
long Excised_real_iovec_len;
double *Excised_real;
};

/*********************************************************************/
/*                                                                   */
/*      The following #defines may be used to shorten                */
/*      references to members of the above structures.               */
/*                                                                   */
/*********************************************************************/
#define P_Threshold (spb_parm->Threshold)
#define P_initial_value (spb_parm->initial_value)
#define I_FFT_imag_iovec_len (spb_input->FFT_imag_iovec_len)
#define I_FFT_imag (spb_input->FFT_imag)
#define I_FFT_real_iovec_len (spb_input->FFT_real_iovec_len)
#define I_FFT_real (spb_input->FFT_real)
#define I_Threshold (*spb_input->Threshold)
#define O_Avg_Noise (spb_output->Avg_Noise)
#define O_Excised_imag_iovec_len (spb_output->Excised_imag_iovec_len)
#define O_Excised_imag (spb_output->Excised_imag)
#define O_Excised_real_iovec_len (spb_output->Excised_real_iovec_len)
#define O_Excised_real (spb_output->Excised_real)
```

Appendix B. C-Code For TLS Prony Block

B.1 Tlsprony.c

```
#include "spw_platform.h"
#ifdef UNIX
#include "caedata/lascody/tlsprony2a/blockcode/tlsprony2au.c"
#else
#ifdef VAX_VMS
#include "[caedata.lascody.tlsprony2a.blockcode]tlsprony2au.c"
#endif VAX_VMS
#endif UNIX
static char *REVISION = "2.50";

/*
 *
 * Block Function: tlsprony2a
 * Library: lascody
 * Date: Tue Aug 29 10:08:44 1995
 *
 */


/************************************************************************/
/*                                                                      */
/*    FEED_THROUGH_LIST INFORMATION:                                    */
/*                                                                      */
/*    --> FEED_THROUGH_TYPE IS NOT EDITABLE. The BDE parameter          */
/*        screen associated with the block must be edited to change     */
/*        the block's FEED_THROUGH_TYPE.                                */
/*                                                                      */
/*        FEED_THROUGH_TYPE = ALL_FEED_THROUGH.                         */
/*                                                                      */
/************************************************************************/

/************************************************************************/
/*                                                                      */
/*    LINK_OPTIONS INFORMATION:                                         */
/*                                                                      */
/*    --> The LINK_OPTIONS list is editable.  It contains all the       */
/*        libraries which the code must be linked to.  Each item in     */
/*        the list must be surrounded by double quotes and              */
/*        separated by commas.  The math library is automatically       */
/*        linked, and does not need to be specified.  The paths         */
/*        may be specified as full paths or as paths relative to        */
/*        the host.                                                     */
/*        A link option can also be specified in the form "-lx"         */
/*        (where x is defined in the UNIX manual on "ld"                */
/*                                                                      */
/*        IMPORTANT:  The entire LINK_OPTIONS list must be deleted      */
/*        if it doesn't contain any elements.                           */
/*                                                                      */
/*        Sample LINK_OPTIONS list:                                     */
/*        (Actual list should be placed below this comment block)       */
/*                                                                      */
/*        LINK_OPTIONS = { "-lm",                                       */
/*                         "//host/code/lib/sample.a" };                */
/*                                                                      */
/*                                                                      */
/************************************************************************/
LINK_OPTIONS = { "-L/apps/matlab/4.2c/extern/lib/sun4","-lmat"};
```

```
/*********************************************************************/
/*                                                                   */
/*    INCLUDE_DIRS INFORMATION:                                      */
/*                                                                   */
/*    --> The INCLUDE_DIRS list is editable.  The list should        */
/*        contain all directory search paths needed to locate all    */
/*        the include files used by this block.  It has the same     */
/*        format as the LINK_OPTIONS list.                           */
/*                                                                   */
/*        IMPORTANT:  The entire INCLUDE_DIRS list must be deleted   */
/*        if it doesn't contain any elements.                        */
/*                                                                   */
/*        Sample INCLUDE_DIRS list:                                  */
/*        (Actual list should be placed below this comment block)    */
/*                                                                   */
/*        INCLUDE_DIRS = { "//host/u/code/include",                  */
/*                         "//host/lib/dir" };                       */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
INCLUDE_DIRS = {"//apps/matlab/4.2c/extern/include"};

/*********************************************************************/
/*                                                                   */
/*      EDITABLE FUNCTIONS                                           */
/*                --> In_tlsprony2a_lascody ()                       */
/*                --> Ro_tlsprony2a_lascody ()                       */
/*                --> Te_tlsprony2a_lascody ()                       */
/*                                                                   */
/*      Structure use:                                               */
/*        Typical input value reference                              */
/*          local_var = *(spb_input->var_name);                      */
/* **OR**    local_var = I_var_name;     */
/*        Typical output value update                                */
/*          spb_output->var_name = local_var;                        */
/* **OR**    O_var_name = local_var;     */
/*        Typical parameter reference                                */
/*          local_var = spb_parm->var_name;                          */
/* **OR**    local_var = P_var_name;     */
/*                                                                   */
/*      (See reference manual for further information)               */
/*                                                                   */
/*********************************************************************/

/*
 *      Initialize Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to initialize the state structure
 *           and constant outputs of the block.  It is called once
 *           for each block instance during simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *           or SYS_FATAL by using the return() function.
 *           User may modify the line containing "return(SYS_OK);".
 */

In_tlsprony2a_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_tlsprony2a_lascody *spb_parm;
STRUCT It_tlsprony2a_lascody *spb_input;
STRUCT Ot_tlsprony2a_lascody *spb_output;
STRUCT St_tlsprony2a_lascody *spb_state;
{
return (SYS_OK);
```

```
}

/*
 *      Run Output Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to update the outputs and/or state
 *          of the block.  It is called each iteration, for each
 *          block instance during simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *          or SYS_FATAL by using the return() function.
 *          User may modify the line containing "return(SYS_OK);".
 */

Ro_tlsprony2a_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_tlsprony2a_lascody *spb_parm;
STRUCT It_tlsprony2a_lascody *spb_input;
STRUCT Ot_tlsprony2a_lascody *spb_output;
STRUCT St_tlsprony2a_lascody *spb_state;
{
#include "/apps/matlab/4.2c/extern/include/engine.h/"
#include "/apps/matlab/4.2c/extern/include/mat.h/"

/* Variables for running MATLAB engine */
Engine *ep;   /* pointer to interface with MATLAB engine */
Matrix *data, *p, *sh, *amp;
double *Nreal, *Nimag, tmp;
int n;

tmp=0.0;
for (n=0;n=spb_input->real_in_iovec_len;n++) {
   tmp+=sqrt(spb_input->real_in[n]*spb_input->real_in[n]);
}

if (tmp < 0.00001) {
    spb_output->real_out=spb_input->real_in;
    spb_output->imag_out=spb_input->imag_in;
    return (SYS_OK);
}


data = mxCreateFull(256,1,COMPLEX);
memcpy(mxGetPr(data),spb_input->real_in,256*sizeof(double));
memcpy(mxGetPi(data),spb_input->imag_in,256*sizeof(double));
mxSetName(data, "D");


/* Start MATLAB engine */

if (!(ep = engOpen("matlab "))) {
  fprintf(stderr,"/nCan't start MATLAB engine");
  exit(-1);
}

/* MATLAB engine */

engPutMatrix(ep, data);
engEvalString(ep,"sh=D");
sh=engGetMatrix(ep, "sh");


/* Close MATLAB engine */
```

```
engClose(ep);

/* MATLAB engine */

Nreal=mxGetPr(sh);
Nimag=mxGetPi(sh);

spb_output->real_out=Nreal;
spb_output->imag_out=Nimag;

mxFreeMatrix(data);
mxFreeMatrix(sh);



return (SYS_OK);
}

/*
 *      Termination Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to dump the final state of the
 *          block.  It is called once for each block instance
 *          during the simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *          or SYS_FATAL by using the return() function.
 *          User may modify the line containing "return(SYS_OK);".
 */

Te_tlsprony2a_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_tlsprony2a_lascody *spb_parm;
STRUCT It_tlsprony2a_lascody *spb_input;
STRUCT Ot_tlsprony2a_lascody *spb_output;
STRUCT St_tlsprony2a_lascody *spb_state;
{

return (SYS_OK);
}

/************************************************************************/
/*                                                                      */
/*     Add any additional functions you need here.                      */
/*                                                                      */
/************************************************************************/
```

B.2 Tlsprony.h

```
#include "FBCDEFS.h"

/*
 *
 * Block Function: tlsprony2a
 * Library: lascody
 * Date: Tue Aug 29 10:08:44 1995
 *
 */
```

```
/*************************************************************************/
/*                                                                       */
/*      EDITABLE USER DEFINED STATE STRUCTURE                            */
/*                --> STRUCT St_tlsprony2a_lascody                       */
/*                                                                       */
/*************************************************************************/

/*
 * State Structure (User Defined, editable)
 */
STRUCT St_tlsprony2a_lascody {
int instance;
};

/*************************************************************************/
/*                                                                       */
/*      UNEDITABLE SIMULATOR DEFINED STRUCTURES                          */
/*                --> STRUCT Pt_tlsprony2a_lascody                       */
/*                --> STRUCT It_tlsprony2a_lascody                       */
/*                --> STRUCT Ot_tlsprony2a_lascody                       */
/*                                                                       */
/*************************************************************************/
/*
 * Parameter Structure, Simulator Defined, uneditable
 */
STRUCT Pt_tlsprony2a_lascody {
double initial_value;
};

/*
 * Input Structure, Simulator Defined, uneditable
 */
STRUCT It_tlsprony2a_lascody {
long imag_in_iovec_len;
double *imag_in;
long real_in_iovec_len;
double *real_in;
};

/*
 * Output Structure, Simulator Defined, uneditable
 */
STRUCT Ot_tlsprony2a_lascody {
long imag_out_iovec_len;
double *imag_out;
long real_out_iovec_len;
double *real_out;
};

/*************************************************************************/
/*                                                                       */
/*      The following #defines may be used to shorten                    */
/*      references to members of the above structures.                   */
/*                                                                       */
/*************************************************************************/
#define P_initial_value (spb_parm->initial_value)
#define I_imag_in_iovec_len (spb_input->imag_in_iovec_len)
#define I_imag_in (spb_input->imag_in)
#define I_real_in_iovec_len (spb_input->real_in_iovec_len)
#define I_real_in (spb_input->real_in)
#define O_imag_out_iovec_len (spb_output->imag_out_iovec_len)
#define O_imag_out (spb_output->imag_out)
#define O_real_out_iovec_len (spb_output->real_out_iovec_len)
#define O_real_out (spb_output->real_out)
```

Appendix C. C-Code for DEF For Jammers Only Block

C.1 Excis_jam_only.c

```
#include "spw_platform.h"
#ifdef UNIX
#include "caedata/lascody/excis_jam_only/blockcode/excis_jam_onlyu.c"
#else
#ifdef VAX_VMS
#include "[caedata.lascody.excis_jam_only.blockcode]excis_jam_onlyu.c"
#endif VAX_VMS
#endif UNIX
static char *REVISION = "2.50";

/*
 *
 * Block Function: excis_jam_only
 * Library: lascody
 * Date: Tue Aug  8 09:06:23 1995
 *
 */


/************************************************************************/
/*                                                                      */
/*    FEED_THROUGH_LIST INFORMATION:                                    */
/*                                                                      */
/*    --> FEED_THROUGH_TYPE IS NOT EDITABLE. The BDE parameter          */
/*        screen associated with the block must be edited to change     */
/*        the block's FEED_THROUGH_TYPE.                                */
/*                                                                      */
/*        FEED_THROUGH_TYPE = ALL_FEED_THROUGH.                         */
/*                                                                      */
/************************************************************************/

/************************************************************************/
/*                                                                      */
/*    LINK_OPTIONS INFORMATION:                                         */
/*                                                                      */
/*    --> The LINK_OPTIONS list is editable.  It contains all the       */
/*        libraries which the code must be linked to.  Each item in     */
/*        the list must be surrounded by double quotes and              */
/*        separated by commas.  The math library is automatically       */
/*        linked, and does not need to be specified.  The paths         */
/*        may be specified as full paths or as paths relative to        */
/*        the host.                                                     */
/*        A link option can also be specified in the form "-lx"         */
/*        (where x is defined in the UNIX manual on "ld"                */
/*                                                                      */
/*        IMPORTANT:  The entire LINK_OPTIONS list must be deleted      */
/*        if it doesn't contain any elements.                           */
/*                                                                      */
/*        Sample LINK_OPTIONS list:                                     */
/*        (Actual list should be placed below this comment block)       */
/*                                                                      */
/*        LINK_OPTIONS = { "-lm",                                       */
/*                         "//host/code/lib/sample.a" };                */
/*                                                                      */
/*                                                                      */
/************************************************************************/
```

```
/*********************************************************************/
/*                                                                   */
/*    INCLUDE_DIRS INFORMATION:                                      */
/*                                                                   */
/*    --> The INCLUDE_DIRS list is editable.  The list should        */
/*        contain all directory search paths needed to locate all    */
/*        the include files used by this block.  It has the same     */
/*        format as the LINK_OPTIONS list.                           */
/*                                                                   */
/*        IMPORTANT:  The entire INCLUDE_DIRS list must be deleted   */
/*        if it doesn't contain any elements.                        */
/*                                                                   */
/*        Sample INCLUDE_DIRS list:                                  */
/*        (Actual list should be placed below this comment block)    */
/*                                                                   */
/*        INCLUDE_DIRS = { "//host/u/code/include",                  */
/*                         "//host/lib/dir" };                       */
/*                                                                   */
/*                                                                   */
/*********************************************************************/

/*********************************************************************/
/*                                                                   */
/*       EDITABLE FUNCTIONS                                          */
/*                 --> In_excis_jam_only_lascody ()                  */
/*                 --> Ro_excis_jam_only_lascody ()                  */
/*                 --> Te_excis_jam_only_lascody ()                  */
/*                                                                   */
/*       Structure use:                                              */
/*         Typical input value reference                             */
/*           local_var = *(spb_input->var_name);                     */
/* **OR**     local_var = I_var_name;     */
/*         Typical output value update                               */
/*           spb_output->var_name = local_var;                       */
/* **OR**     O_var_name = local_var;     */
/*         Typical parameter reference                               */
/*           local_var = spb_parm->var_name;                         */
/* **OR**     local_var = P_var_name;     */
/*                                                                   */
/*       (See reference manual for further information)              */
/*                                                                   */
/*********************************************************************/

/*
 *       Initialize Function (must be present)
 *       --> If editing, modify only the lines within the
 *           function's opening and closing brackets.
 *
 *       This function is used to initialize the state structure
 *            and constant outputs of the block.  It is called once
 *            for each block instance during simulation.
 *
 *       Function must always return either SYS_OK, SYS_TERM,
 *            or SYS_FATAL by using the return() function.
 *            User may modify the line containing "return(SYS_OK);".
 */

In_excis_jam_only_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_excis_jam_only_lascody *spb_parm;
STRUCT It_excis_jam_only_lascody *spb_input;
STRUCT Ot_excis_jam_only_lascody *spb_output;
STRUCT St_excis_jam_only_lascody *spb_state;
{
  return (SYS_OK);
```

```
]

/*
 *      Run Output Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to update the outputs and/or state
 *          of the block.  It is called each iteration, for each
 *          block instance during simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *          or SYS_FATAL by using the return() function.
 *          User may modify the line containing "return(SYS_OK);".
 */

Ro_excis_jam_only_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_excis_jam_only_lascody *spb_parm;
STRUCT It_excis_jam_only_lascody *spb_input;
STRUCT Ot_excis_jam_only_lascody *spb_output;
STRUCT St_excis_jam_only_lascody *spb_state;
{
int n;
double mag_sqrd, emag_sqrd;


/* This loop calculates the power of the original jamming signal
 * using Parseval's relationship for the DFT.
 */

mag_sqrd = 0.0;
for (n=0; n< I_Jam_real_iovec_len; n++) {
 mag_sqrd+=((I_Jam_real[n]*I_Jam_real[n])+(I_Jam_imag[n]*I_Jam_imag[n]));
}
 O_Orig_Jam_Power = mag_sqrd;

/* This loop performs the excision of the jammer based on the threshold
 * calculated from the simulation using signal plus jammer plus noise.
 */

for (n=0; n < I_Jam_real_iovec_len; n++)
   {  if (sqrt((I_Jam_real[n]*I_Jam_real[n])+(I_Jam_imag[n]*I_Jam_imag[n]))
> I_Thresh)
   { O_Real_out[n]=0.000000000000000001;
     O_Imag_out[n]=0.000000000000000001; }
     else {
     O_Real_out[n]=I_Jam_real[n];
     O_Imag_out[n]=I_Jam_imag[n]; }}

/* This loop calculates the power of the jammer after excision.
 */

emag_sqrd = 0.0;
for (n=0; n < I_Jam_real_iovec_len; n++)
  {emag_sqrd+=((O_Real_out[n]*O_Real_out[n])+(O_Imag_out[n]*O_Imag_out[n]));
  }
   O_Power_removed = emag_sqrd;
if (mag_sqrd > 0.0000001) {
     O_pjr = 100.0*(1.0 - (emag_sqrd/mag_sqrd));}

return (SYS_OK);
}

/*
```

```
 *      Termination Function (must be present)
 *      --> If editing, modify only the lines within the
 *          function's opening and closing brackets.
 *
 *      This function is used to dump the final state of the
 *           block.  It is called once for each block instance
 *           during the simulation.
 *
 *      Function must always return either SYS_OK, SYS_TERM,
 *           or SYS_FATAL by using the return() function.
 *           User may modify the line containing "return(SYS_OK);".
 */

Te_excis_jam_only_lascody (spb_parm, spb_input, spb_output, spb_state)
STRUCT Pt_excis_jam_only_lascody *spb_parm;
STRUCT It_excis_jam_only_lascody *spb_input;
STRUCT Ot_excis_jam_only_lascody *spb_output;
STRUCT St_excis_jam_only_lascody *spb_state;
{

return (SYS_OK);
}

/************************************************************************/
/*                                                                      */
/*      Add any additional functions you need here.                     */
/*                                                                      */
/************************************************************************/
```

C.2 *Excis_jam_only.h*

```
#include "FBCDEFS.h"

/*
 *
 * Block Function: excis_jam_only
 * Library: lascody
 * Date: Tue Aug  8 09:06:23 1995
 *
 */

/************************************************************************/
/*                                                                      */
/*      EDITABLE USER DEFINED STATE STRUCTURE                           */
/*                --> STRUCT St_excis_jam_only_lascody                  */
/*                                                                      */
/************************************************************************/

/*
 * State Structure (User Defined, editable)
 */
STRUCT St_excis_jam_only_lascody {
int instance;
};

/************************************************************************/
/*                                                                      */
/*      UNEDITABLE SIMULATOR DEFINED STRUCTURES                         */
/*                --> STRUCT Pt_excis_jam_only_lascody                  */
/*                --> STRUCT It_excis_jam_only_lascody                  */
```

```
/*              --> STRUCT Ot_excis_jam_only_lascody
/*                                                                    */
/**********************************************************************/
/*
 * Parameter Structure, Simulator Defined, uneditable
 */
STRUCT Pt_excis_jam_only_lascody {
double Threshold;
double initial_value;
};

/*
 * Input Structure, Simulator Defined, uneditable
 */
STRUCT It_excis_jam_only_lascody {
long Jam_imag_iovec_len;
double *Jam_imag;
long Jam_real_iovec_len;
double *Jam_real;
double *Thresh;
};

/*
 * Output Structure, Simulator Defined, uneditable
 */
STRUCT Ot_excis_jam_only_lascody {
long Imag_out_iovec_len;
double *Imag_out;
double Orig_Jam_Power;
double Power_removed;
long Real_out_iovec_len;
double *Real_out;
double pjr;
};

/**********************************************************************/
/*                                                                    */
/*      The following #defines may be used to shorten             */
/*      references to members of the above structures.             */
/*                                                                    */
/**********************************************************************/
#define P_Threshold (spb_parm->Threshold)
#define P_initial_value (spb_parm->initial_value)
#define I_Jam_imag_iovec_len (spb_input->Jam_imag_iovec_len)
#define I_Jam_imag (spb_input->Jam_imag)
#define I_Jam_real_iovec_len (spb_input->Jam_real_iovec_len)
#define I_Jam_real (spb_input->Jam_real)
#define I_Thresh (*spb_input->Thresh)
#define O_Imag_out_iovec_len (spb_output->Imag_out_iovec_len)
#define O_Imag_out (spb_output->Imag_out)
#define O_Orig_Jam_Power (spb_output->Orig_Jam_Power)
#define O_Power_removed (spb_output->Power_removed)
#define O_Real_out_iovec_len (spb_output->Real_out_iovec_len)
#define O_Real_out (spb_output->Real_out)
#define O_pjr (spb_output->pjr)
```

Bibliography

1. Barnes, G.R. "Spread Spectrum Wireless Links," *Fourth European Conference on Radio Relay Systems*, 39–44 (October 1993).
2. Cadzow, J. A. "Spectrun estimation: An overdetermined rational model equation approach." *Proc. IEEE70*. 907–939. September 1982.
3. Chan, Y. T. and R. P. Langford. "Spectral Estimation via the High–Order Yule–Walker Equations," *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP–30*(5):689–698 (October 1982).
4. Comdisco Systems, Inc. *SPW User's Manual* (Document version 3.0 Edition), September 1992.
5. Cooper, George R. and Clare D. McGillem. *Modern Communications and Spread Spectrum*. New York: McGraw-Hill Book Company, 1986.
6. Dixon, Robert C. *Spread Spectrum Systems*. John Wiley and Sons, 1984.
7. Falen, Gerald L. *Analysis and Simulation of Narrowband GPS Jamming Using Digital Temporal Filtering*. MS thesis, AFIT/GE/ENG/94D-09, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1994.
8. GEC Plessey Semiconductors, Scotts Valley, CA. *Digital Video and Digital Signal Processing IC Handbook*, December 1993.
9. Golub, G. H. and C. F. VanLoan. *Matrix Computations* (2 Edition). Baltimore, MD: Johns Hopkins, 1989.
10. Gottesman, Lynn D. and Laurence B. Milstein. "The Effect of a Narrowband Interference Rejection Filter on Coarse Acquisition in Direct Sequence Spread Spectrum," *Proceedings GLOBECOM '90, 1*:256–260 (1990).
11. Jeruchim, Michel C. and others. *Simulation of Communication Systems*. New York: Plenum Press, 1992.
12. Logsdon, Tom. *The NAVSTAR Global Positioning System*. New York: Van Nostrand Reinhold, 1992.
13. Mills, Michael S. *Evaluation of an Acoustic Charge Transport (ACT) Device For Adaptive Interference Suppression in Spread Spectrum Communication Systems*. MS thesis, AFIT/GE/ENG/93D-27, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1993.
14. Milstein, L. B. and P. K. Das. "An Analysis of a Real-time Transform Domain Filtering Digital Communication System - Part I: Narrowband Interference Rejection," *IEEE Trans. Comm., COMM-28*:816–824 (June 1980).
15. Nicholson, David L. *Spread Spectrum Signal Design LPE and AJ Systems*. Computer Science Press, 1988.

16. Poor, H. Vincent. "Narrowband Interference Suppression in Spread Spectrum CDMA," *IEEE Personal Communications*, 14–27 (Third Quarter 1994).

17. Rahman, M. A. and K. B. Yu. "Total least squares approach for frequency estimation using linear prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-35*:1440–1454 (October 1987).

18. Saulnier, Gary J. "Suppression of Narrowband Jammers in a Spread Spectrum Receiver Using Transform-Domain Adaptive Filtering," *IEEE Journal on Selected Areas of Communications, 10*:742–749 (May 1992).

19. Shanmugan, K. Sam. "Simulation and Implementation of Communication and Signal Processing Systems," *IEEE Communications Magazine*, 25–53 (July 1994).

20. Sklar, Bernard. *Digital Communications Fundamentals and Applications.* Prentice Hall, 1988.

21. Stoica, P., et al. "Asymptotic Properties of the High-Order Yule-Walker Estimates of Sinusoidal Frequencies," *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-37*(11):1721–1734 (November 1989).

22. Taub, Herbert and Donald L. Schilling. *Principles of Communication Systems* (Second Edition). McGraw-Hill, 1986.

23. Tazebay, Mehmet V. and Ali N. Akansu. "A Smart Time-Frequency Exciser For Spread Spectrum Communications," *1995 International Conference on Acoustics, Speech, and Signal Processing* (May 1995).

24. Madden, Christopher B. *Effects of Jamming and Excision Filtering Upon Error Rates and Detectability of Spread Spectrum Communications System.* MS thesis, AFIT/GE/ENG/95D-13, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1995.

Vita

Captain James A. Lascody [redacted] He graduated from Potomac Senior High School in 1987 and entered undergraduate studies at the United States Air Force Academy in Colorado Springs, Colorado. In May 1991, he graduated from USAFA with a Bachelor of Science Degree in Electrical Engineering and a regular commission as a 2nd Lieutenant. He first assignment was at the Air Force Information Warfare Center, Kelly AFB, TX as an Electronic Warfare Test Support Engineer. In May 1994, he entered the School of Engineering, Air Force Institute of Technology.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1995	Master's Thesis

4. TITLE AND SUBTITLE
Narrowband Interference Suppression In Spread Spectrum Communication Systems

5. FUNDING NUMBERS

6. AUTHOR(S)
James A. Lascody
Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/95D-12

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Alan Oester
GPS JPO/CZJ
2435 Vela Way
Los Angeles AFB, CA 90245-5500

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Of significant interest to the United States military is the ability of an enemy to deny or disrupt the operation of the Global Positioning System. To combat this threat the GPS JPO initiated the Tactical GPS Anit-Jam Technology project, which yielded a prototype Digital Excision Filter (DEF) to remove narrowband jammers. This research describes the work performed to get the DEF hardware operational and extends the previous research performed in this area. Comdisco's Signal Processing Worksystem was used to examine the effect of the DEF on the probability of bit error. This research uses peak-to-average correlation value, probability of bit error, and percent jammer power removed to examine the performance of the DEF. Fourteen jamming scenarios are examined using CW, pulsed CW, and broadband noise jammers. The DEF effectively rejected all of the jammers except the broadband noise jammer. In scenarios other than the broadband noise jammer, the DEF removed over 98% of the jammer power. The bit error rate curves show that the DEF significantly enhanced the performance of the system in extreme jamming environments. The results presented in this research show that the DEF is a viable, robust option to remove narrowband interference.

14. SUBJECT TERMS
GPS, Digital Excision Filtering, Interference Suppression, Spread Spectrum, Simulation

15. NUMBER OF PAGES
109

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.