2009-06

Efficacy of various waveforms to support geolocation

Crnkovich, Joseph G.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

EFFICACY OF VARIOUS WAVEFORMS TO SUPPORT GEOLOCATION

by

Joseph G. Crnkovich, Jr.

June 2009

Thesis Advisor:	Frank Kragh
Co-advisor:	Herschel H. Loomis, Jr.

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2009	Master's Thesis

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Efficacy of Various Waveforms to Support Geolocation	
6. AUTHOR(S) Joseph G. Crnkovich, Jr.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	

13. ABSTRACT (maximum 200 words)

This thesis investigates the impact of various waveform parameters on the ability to estimate accurately the position of the source of a known data-less emission that is visible to multiple simultaneous collectors. It provides an overview of the basic geolocation problem and identifies various parameters affecting geolocation accuracy, showing those that are affected by the waveform and those that are not. Performance estimates are provided for detecting the signal and for estimating the time and frequency of arrival (TOA and FOA) of the signal, which are the key measure of a waveform's ability to support geolocation. Several exemplar waveforms are chosen to illustrate the effects of various waveform parameters, and the performance of these example waveforms is verified through software simulations.

Results show for additive white Gaussian noise (AWGN) interference that accuracy of estimates is predominantly determined by the transmit power (i.e., received SNR), signal bandwidth (for TOA), and signal duration (for FOA). For a given SNR, occupied bandwidth, and total duration, a waveform can be "shaped" in the time and frequency domains to improve performance relative to a reference direct sequence spread spectrum (DSSS) signal. Software simulations confirm theoretical performance estimates.

This thesis summarizes the effects of various waveform parameters on geolocation performance, demonstrates these by modeling exemplar waveforms, and provides software that can be used to simulate performance.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Geolocation, Cross Ambiguity Function, CAF, Matched Filter Detection	185
	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	UU

NSN 7540-01-280-5500

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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

EFFICACY OF VARIOUS WAVEFORMS TO SUPPORT GEOLOCATION

Joseph G. Crnkovich, Jr.
Civilian, Naval Research Laboratory, Washington, D.C.
B.S., Marquette University, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2009

Author: Joseph G. Crnkovich, Jr.

Approved by: Frank Kragh
Thesis Advisor

Herschel H. Loomis, Jr.
Co-advisor

Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis investigates the impact of various waveform parameters on the ability to estimate accurately the position of the source of a known data-less emission that is visible to multiple simultaneous collectors. It provides an overview of the basic geolocation problem and identifies various parameters affecting geolocation accuracy, showing those that are affected by the waveform and those that are not. Performance estimates are provided for detecting the signal and for estimating the time and frequency of arrival (TOA and FOA) of the signal, which are the key measure of a waveform's ability to support geolocation. Several exemplar waveforms are chosen to illustrate the effects of various waveform parameters, and the performance of these example waveforms is verified through software simulations.

Results show for additive white Gaussian noise (AWGN) interference that accuracy of estimates is predominantly determined by the transmit power (i.e., received SNR), signal bandwidth (for TOA), and signal duration (for FOA). For a given SNR, occupied bandwidth, and total duration, a waveform can be "shaped" in the time and frequency domains to improve performance relative to a reference direct sequence spread spectrum (DSSS) signal. Software simulations confirm theoretical performance estimates.

This thesis summarizes the effects of various waveform parameters on geolocation performance, demonstrates these by modeling exemplar waveforms, and provides software that can be used to simulate performance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
A. BACKGROUND 1
B. OBJECTIVE 2
C. RELATED WORK 3
D. THESIS ORGANIZATION 3

II. THE GEOLOCATION PROBLEM 5
A. THE EMITTER 5
B. METHODS OF PASSIVE GEOLOCATION 6
C. KEY DETECTION PARAMETERS 8
D. FIGURES OF MERIT FOR GEOLOCATION ACCURACY 8

III. PARAMETERS BEYOND THE CONTROL OF THE WAVEFORM DEVELOPER THAT AFFECT GEOLOCATION PERFORMANCE 11
A. NON-WAVEFORM PARAMETERS TO CONSIDER 11
B. WAVEFORM CONSTRAINTS 14

IV. PERFORMANCE ESTIMATES 17
A. DETECTION 17
1. Coherent Detection 17
a. Receiver Processing 18
b. Decision Variable Statistics 19
c. Probability of Detection and False Alarm 20
d. An Example 24
2. Noncoherent Detection 25
a. Receiver Processing 25
b. Probability of Detection and False Alarm 26
B. FREQUENCY AND TIME ESTIMATION 27
1. The Complex Ambiguity Function (CAF) 27
2. Theoretical Performance 29

V. PROPOSED WAVEFORMS 33
A. BPSK WAVEFORMS 36
1. Waveform #1 – “Reference Waveform” 38
2. Waveform #2 – “Time Gap” 44
3. Waveform #3 – “Split Spectrum” 46
4. Waveform #4 – “Shortened Pulse” 49
B. FILTERED BPSK WAVEFORMS 52
1. Filtered Waveform #1 – “Reference Waveform” 53
2. Filtered Waveform #2 – “Time Gap” 56
3. Filtered Waveform #3 – “Split Spectrum” 58
4. Filtered Waveform #4 – “Shortened Pulse” 60
C. SHAPED CHIP WAVEFORMS 62

VI. SIMULATION SOFTWARE 71

A. SIMULATION OVERVIEW 71
B. ROUTINES 74
1. main_simulation.m 74
2. generate_waveform.m 78
3. gen_sig.m 78
4. filt_bnn_fft.m 79
5. get_canned_waveform.m 80
6. display_waveform_calc_rmsBW.m 81
7. display_waveform_calc_rmsT.m 81
8. gen_noise_vector.m 81
9. perf_demod_test.m 83
10. CAFv2.m 87
11. display_toa_foa_v_snr_and_prep_data.m 88
12. display_scatter_toa_foa.m 88
C. SCRIPT FILES 88
1. script_top_level_simulate_various_WFs.m 89
2. script_display_toa_foa_v_snr_across_runs_mrkrs.m 90
3. script_plot_WFs.m 90
4. gen_sinc.m 91
5. mls_gen.m 91

VII. RESULTS AND CONCLUSIONS 93
A. SIMULATIONS PERFORMED 93
B. RESULTS OF SIMULATIONS AND COMPARISON 93
1. BPSK-Generated Waveforms 94
2. Shaped-Chip Waveforms 99
3. Bandwidth Constrained Waveforms 101
C. SUMMARY OF FINDINGS 104
D. FUTURE WORK 106

APPENDIX 109
A. MATLAB CODE: SCRIPT_TOP_LEVEL_SIMULATE VARIOUS_WFS.M 109
B. MATLAB CODE: SCRIPT_DISPLAY_TOA_FOA_V_SNR_ACROSS_RUNS_MRKRS.M 113
C. MATLAB CODE: SCRIPT_PLOT_WFS.M 117
D. MATLAB CODE: DISPLAY_TOA_FOA_V_SNR_AND_PREP_DATA.M 118
E. MATLAB CODE: DISPLAY_SCATTER_FOA_TOA.M 120
F. MATLAB CODE: GEN_SINC.M 121
G. MATLAB CODE: MLS_GEN.M 123
H. MATLAB CODE: MAIN_SIMULATION.M 124
I. MATLAB CODE: GENERATE_WAVEFORM.M 130
J. MATLAB CODE: GEN_SIG.M 133
K. MATLAB CODE: FILT_BNN_FFT.M 137
L. MATLAB CODE: GET_CANNED_WAVEFORM.M 138

M. MATLAB CODE: DISPLAY_WAVEFORM_CALC_RMSBW.M 139
N. MATLAB CODE: DISPLAY_WAVEFORM_CALC_RMST.M 141
O. MATLAB CODE: GEN_NOISE_VECTOR.M 143
P. MATLAB CODE: PERF_DEMOD_TEST.M 144
Q. MATLAB CODE: CAFV2.M 147
LIST OF REFERENCES 155
INITIAL DISTRIBUTION LIST 159

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1 Scatter Plot of Waveform Parameters for Select Waveforms. xvii
Figure 2 Temporal and Spectral Plots of Waveform #1F. xviii
Figure 3 Temporal and Spectral Plots of Waveform #3F. xviii
Figure 4 Temporal and Spectral Plots of Waveform #17. xix
Figure 5 Temporal and Spectral Plots of Waveform #2F. xix
Figure 6 Temporal and Spectral Plots of Waveforms #4F. xx
Figure 7 TOA Accuracies – Summary of Alternatives. xxi
Figure 8 FOA Accuracies – Summary of Alternatives. xxii
Figure 9 Eccentricity of Ellipse for CEP is Geometry Dependent (after [1]). 10
Figure 10 Relation Between Angular and Location Errors (from [1]). 11
Figure 11 Example TEC map (from [13]) 13
Figure 12 Basic Radiometer (after [15]). 15
Figure 13 Basic Two-Receiver Correlation Filter (from [15]). 15
Figure 14 Coherent Receiver (after [20]). 18
Figure 15 Coherent Probability Distribution Functions (pdf) (after [22]). 21
Figure 16 Noncoherent Receiver (after [20]). 26
Figure 17 Scatter Plot of Waveform Parameters for All Waveforms. 35
Figure 18 Scatter Plot of Parameters for Waveforms with $B_{nn} = 8$ kHz. 36
Figure 19 Waveform #1 – Power vs. Time. 39
Figure 20 Waveform #1 – Power Spectral Density. 40
Figure 21 Autocorrelation of Waveform #1. 43
Figure 22 Waveform #2 – Power vs. Time. 44
Figure 23 Waveform #2 – Power Spectral Density. 45
Figure 24 Waveform #2 – Autocorrelation. 46
Figure 25 Waveform #3 – Power Spectral Density. 47
Figure 26 Waveform #3 – Power vs. Time. 48
Figure 27 Waveform #3 – Autocorrelation. 49
Figure 28 Waveform #4 – Power vs. Time. 50
Figure 29 Waveform #4 – Power Spectral Density. 51
Figure 30 Waveform #4 – Autocorrelation. 52
Figure 31 Filtered Waveform #1 – Power Spectral Density. 54
Figure 32 Filtered Waveform #1 – Power vs. Time. 55
Figure 33 Filtered Waveform #1 – Autocorrelation. 56
Figure 34 Filtered Waveform #2 – Power Spectral Density. 57
Figure 35 Filtered Waveform #2 – Power vs. Time. 58
Figure 36 Filtered Waveform #3 – Power Spectral Density. 59
Figure 37 Filtered Waveform #3 – Power vs. Time. 60
Figure 38 Filtered Waveform #4 – Power Spectral Density. 61
Figure 39 Filtered Waveform #4 – Power vs. Time. 62
Figure 40 Sinc Function. 64
Figure 41 PSD of Rectangular and Sinc Modulated Signal. 65
Figure 42 Waveform #17 – Power Spectral Density. 66

Figure 43 Waveform #17 – Power vs. Time. 67
Figure 44 Waveform #17 – Autocorrelation. 68
Figure 45 Filtered Waveform #17 – Power Spectral Density. 69
Figure 46 Filtered Waveform #17 – Power vs. Time. 70
Figure 47 MATLAB m-files Created or Modified. 73
Figure 48 Overview of main_simulation.m. 75
Figure 49 Signal Spectrum Before and After Adjusting Noise Equation. 83
Figure 50 Analytic Signal Before and After Mixing Down to Baseband. 85
Figure 51 Signal in I-Channel vs. Q-Channel.in High SNR. 86
Figure 52 The Sampled Decision Variable, Resulting Bits, and Reference Bits. 87
Figure 53 TOA Accuracies – Unfiltered BPSK vs. Filtered. 95
Figure 54 FOA Accuracies – Unfiltered BPSK vs. Filtered. 96
Figure 55 Waveform #4 Example CAF with $SNR = 100$ dB. 97
Figure 56 Waveform #4 Example CAF with $SNR = 0$ dB. 98
Figure 57 TOA Accuracies – Reference Waveform vs. Shaped Chips. 100
Figure 58 FOA Accuracies – Reference Waveform vs Shaped Chips. 101
Figure 59 TOA Accuracies – Summary of Alternatives. 103
Figure 60 FOA Accuracies – Summary of Alternatives. 104

LIST OF TABLES

Table 1 Waveform Summary Table (Bandwidth Constrained)..... xvii
Table 2 GPS Standard Errors (from [10]). 14
Table 3 Waveform Summary Table. 34
Table 4 Summary of main_simulation.m Parameters. 76
Table 5 User Specified Settings in gensig.m. 79
Table 6 Suggested Parameters When Using perf_demod_test.m. 84
Table 7 Waveform Variations Simulated. 90
Table 8 Samples per Shaped Pulse. 91

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis examines the efficacy of a waveform to support geolocation. The research specifically explored how well a waveform could support identifying the location of an emitter based on a single transmission in the presence of additive white Gaussian noise (AWGN) given that the emitter is simultaneously visible to multiple coherent collectors. Various exemplar waveforms are proposed, and MATLAB® simulations modeled the waveforms and processing of the signals for the key parameters, namely the time of arrival (TOA) and frequency of arrival (FOA). These simulations confirm and illustrate the analytical formulae. The simulation code is available to test the performance of other waveforms.

The analysis also assumes that

- the emitter is transmitting isotropically,
- no multipath or atmospheric effects exist,
- the entire channel is linear (including amplifiers),
- the coherent collectors have perfect knowledge of time and their own location,
- the collection geometry is static,
- the transmitted signal is modulated by a completely known chipping sequence,
- the collectors have a copy of the signal being transmitted, and
- no data are being modulated onto the emission.

This thesis identifies the ability of a waveform to support accurate estimation of TOA and FOA as the figures of merit to support geolocation of an emission. The particular metric is the standard deviation σ of these estimates. Any attempt to define the waveform accuracy by using a figure of merit involving physical location requires knowledge of the collectors and collection geometry, which is beyond the scope of this thesis.

The three main parameters affecting σ_{TOA} and σ_{FOA} are the ratio of signal power to noise power E_s/N_0, bandwidth, and signal duration. These parameters are limited not just by physical constraints such as transmit power and the occupied bandwidth, but also by acceptable visibility by an adversary (e.g., low probability of intercept or detection).

Analysis shows that the probability of correctly detecting the signal P_d along with the probability of a false alarm P_{FA} are a function only of the signal power, noise power spectral density, duration of the signal, and detection threshold, but are otherwise independent of the waveform characteristics. Probability of detection P_d, probability of false alarm P_{FA}, and detection threshold are related. For fixed signal power to noise power ratio (SNR), increasing the detection threshold decreases the probability of false alarm. However, for fixed SNR, increasing the detection threshold will also decrease the probability of detection.

On the other hand, the "shape" of the waveform does have an effect on σ_{TOA} and σ_{FOA} as stated by Stein [3]. For a given E_s/N_0, occupied bandwidth and total signal duration, manipulating the PSD and the signal amplitude profile vs. time of the signal cause variations in σ_{TOA} and σ_{FOA}, respectively. "Pushing" the waveform energy from the center to the extremes increases the root mean square rms value of that parameter. For example, generating a waveform that has a higher PSD near the band edges than at the center of the band will provide a higher rms bandwidth signal than one that has a flat PSD, resulting in a smaller value for σ_{TOA} and improved location estimation. Likewise, generating a waveform in which the signal amplitude is greater towards the beginning and end than in the middle of the signal results in an improved (i.e., smaller) σ_{FOA}.

Various bandwidth-constrained waveforms of the same duration and energy are proposed along with a reference waveform at various chip rates. The reference waveform, 1F, and four other waveforms of similar total bandwidth are

listed in Table 1 and shown in Figure 1, which is a scatter plot of the two key parameters, rms radian frequency β and rms duration T_e. In addition to the waveforms shown, the reference waveform is also chipped at higher rates to provide a reference for comparison with the waveform variations.

Table 1 Waveform Summary Table (Bandwidth Constrained)

WF#	Name	rms rad. Freq. (rad/s)	rms duration (s)	Bnn (kHz)
1F	Filtered Reference	8506	0.5577	8
2F	Filtered Time Gap	8434	0.8482	8
3F	Filtered Split Spectrum	14463	0.558	8
4F	Filtered Shortened Pulse	8655	0.1381	8
17	Sinc - 8.3kcps	14968	0.558	8

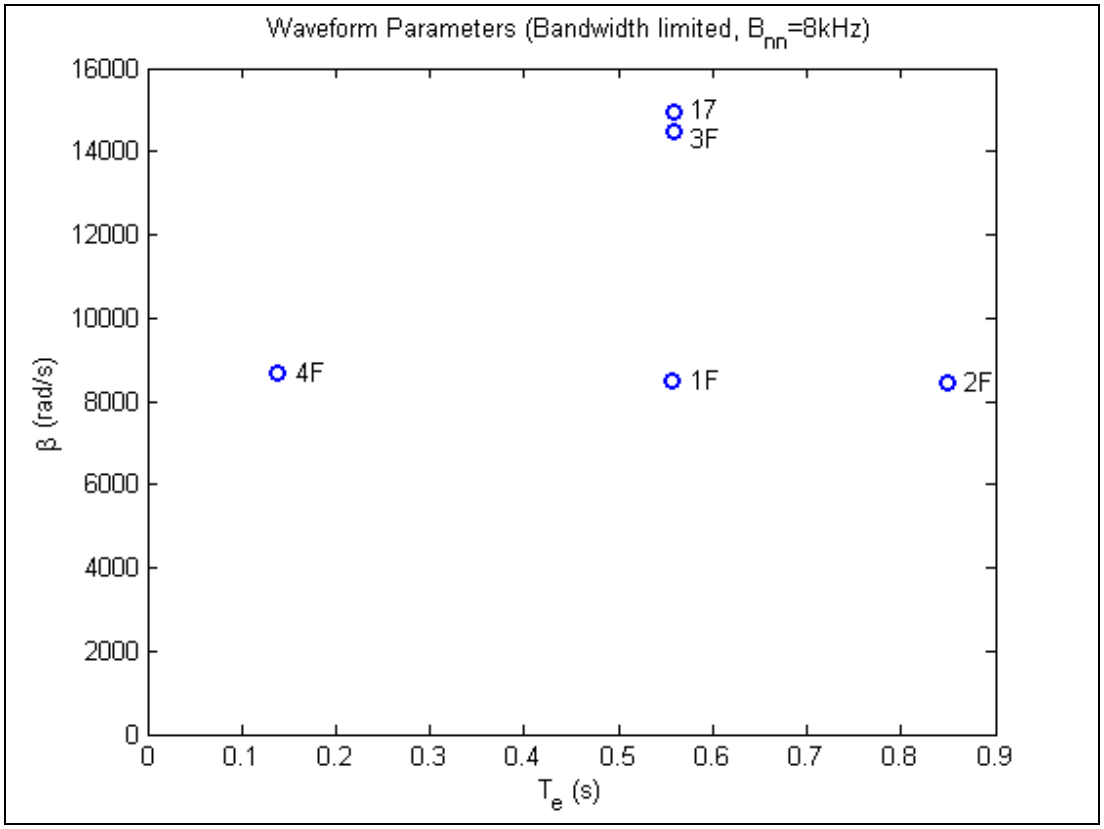

Figure 1 Scatter Plot of Waveform Parameters for Select Waveforms.

The next three sets of plots are of waveforms having the same rms interval T_e.The left and right plots of Figure 2 show, respectively, the temporal

and spectral plots of the waveform #1F, the filtered reference wavefom. Similar types of plots are shown for waveform #3F, Figure 3, and waveform #17, Figure 4. Note that that the power profiles for all three are very similar, although they may have different null depth and ripple. However, the PSD profiles are significantly different for the three, even though they all have the same occupied bandwidth. The unfiltered version of waveform #17 was used because it is not very different from its filtered version, #17F. The shape of the PSD leads to significantly different rms radian frequency β values but does not affect the rms duration as can be seen in Figure 1.

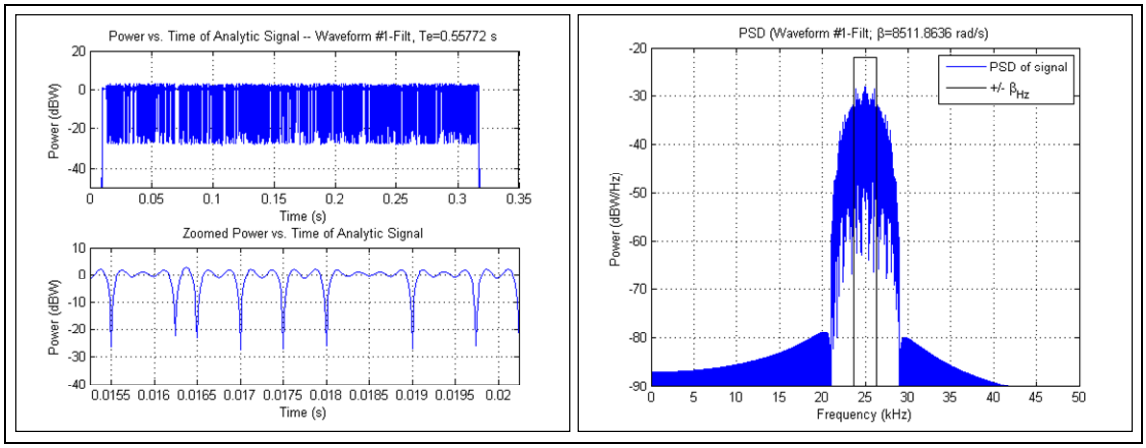

Figure 2 Temporal and Spectral Plots of Waveform #1F.

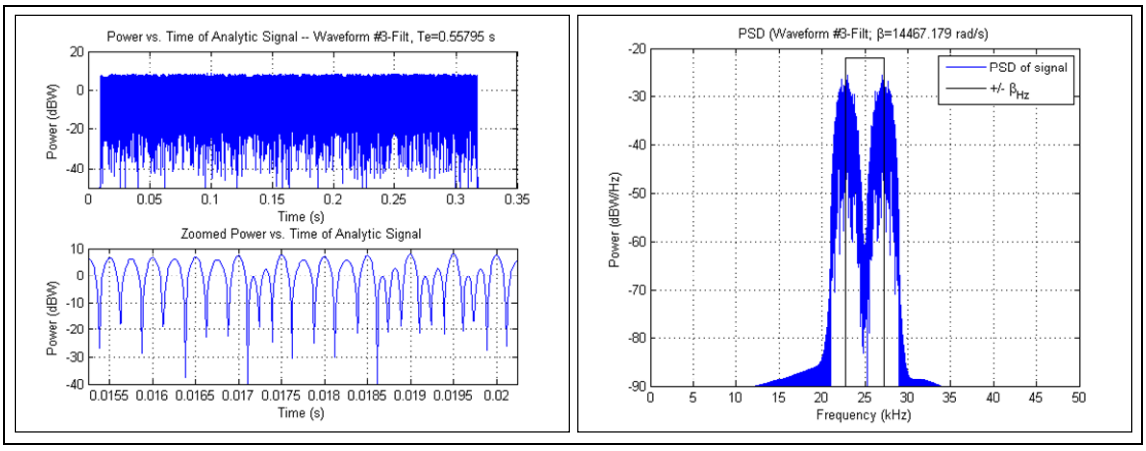

Figure 3 Temporal and Spectral Plots of Waveform #3F.

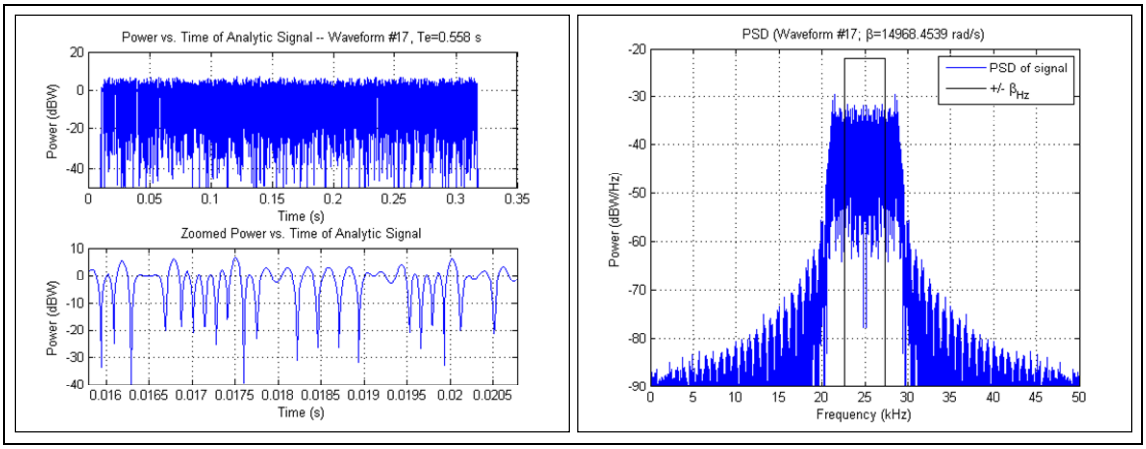

Figure 4 Temporal and Spectral Plots of Waveform #17.

In a similar manner, temporal and spectral plots of the two other waveforms with the same rms radian frequency β as waveform #1F, i.e., waveforms #2F, and #4F, are shown in Figure 5 and Figure 6, respectively. Note that all three have very similar PSD profiles; However, the power profiles differ greatly. Waveform #2F is similar to #1F except the energy in the middle was pushed to the outside. Waveform #3 is the converse of this and has the energy pushed towards the middle of the waveform. These variations in shape lead to significantly different rms duration T_e values while leaving β unchanged as can also be seen in Figure 1.

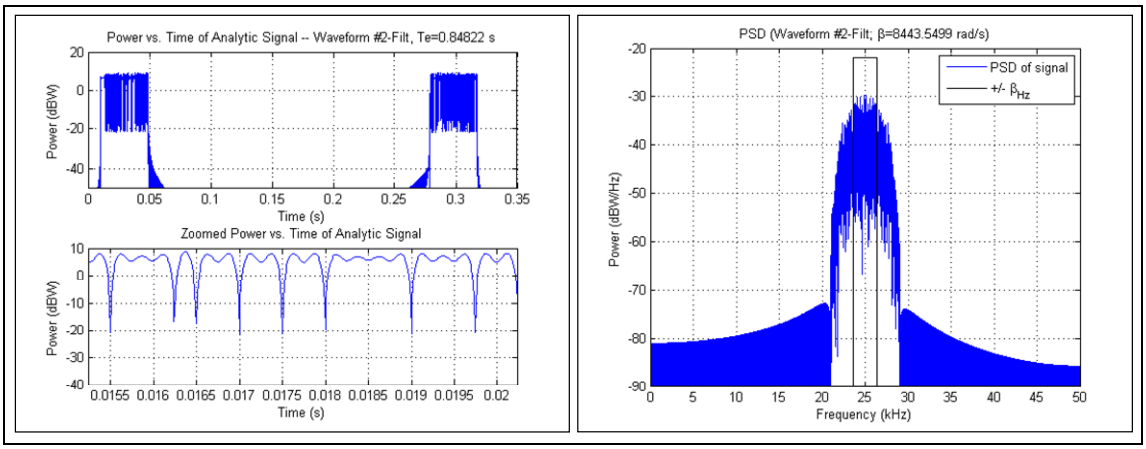

Figure 5 Temporal and Spectral Plots of Waveform #2F.

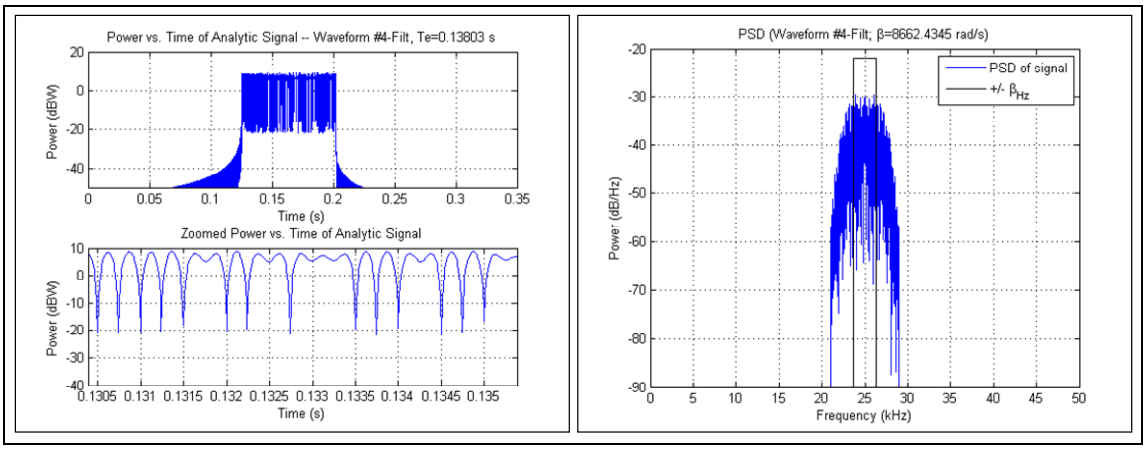

Figure 6 Temporal and Spectral Plots of Waveforms #4F.

These variations in β and T_e lead to significant differences in waveform geolocation performance. Figure 7 shows σ_{TOA} at various values of $SNR = E_s / N_0$ for different waveforms. In the region of high SNR values ($\geq 20dB$), one can see that doubling the chip rate of the reference waveform causes a 50% reduction in σ_{TOA} for a given SNR. Likewise, transmitting a signal with 6 dB more power would also cause a 50% reduction in σ_{TOA} for a given waveform at a given power. However, one could also achieve almost a 50% reduction in σ_{TOA} from the reference waveform, without increased energy or bandwidth, by reshaping it to waveform #3 (filtered) or #17 (unfiltered or filtered). However, this is at a cost of increased peak power.

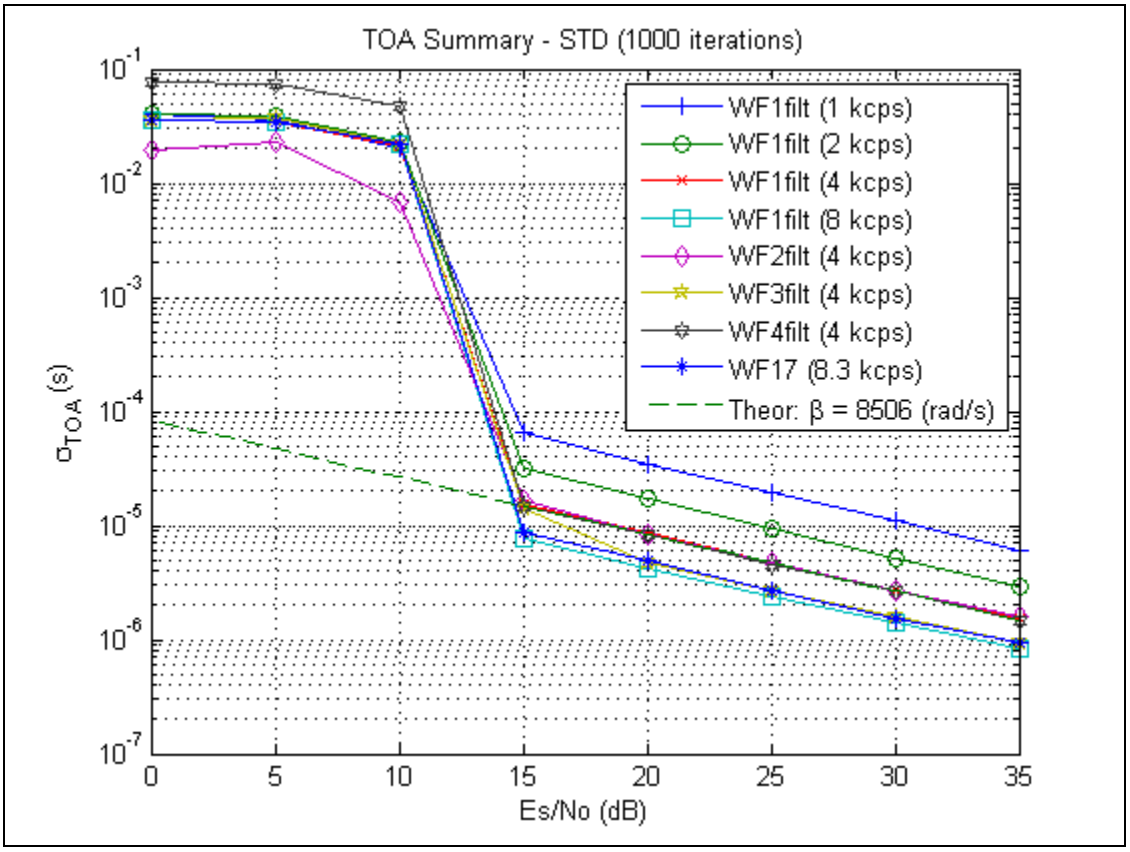

Figure 7 TOA Accuracies – Summary of Alternatives.

Comparing the waveforms for FOA performance (Figure 8) shows that changing the bandwidth has no affect on the resulting standard deviation σ_{FOA}; however, shortening, lengthening, or otherwise changing the power profile over time does affect σ_{FOA}.

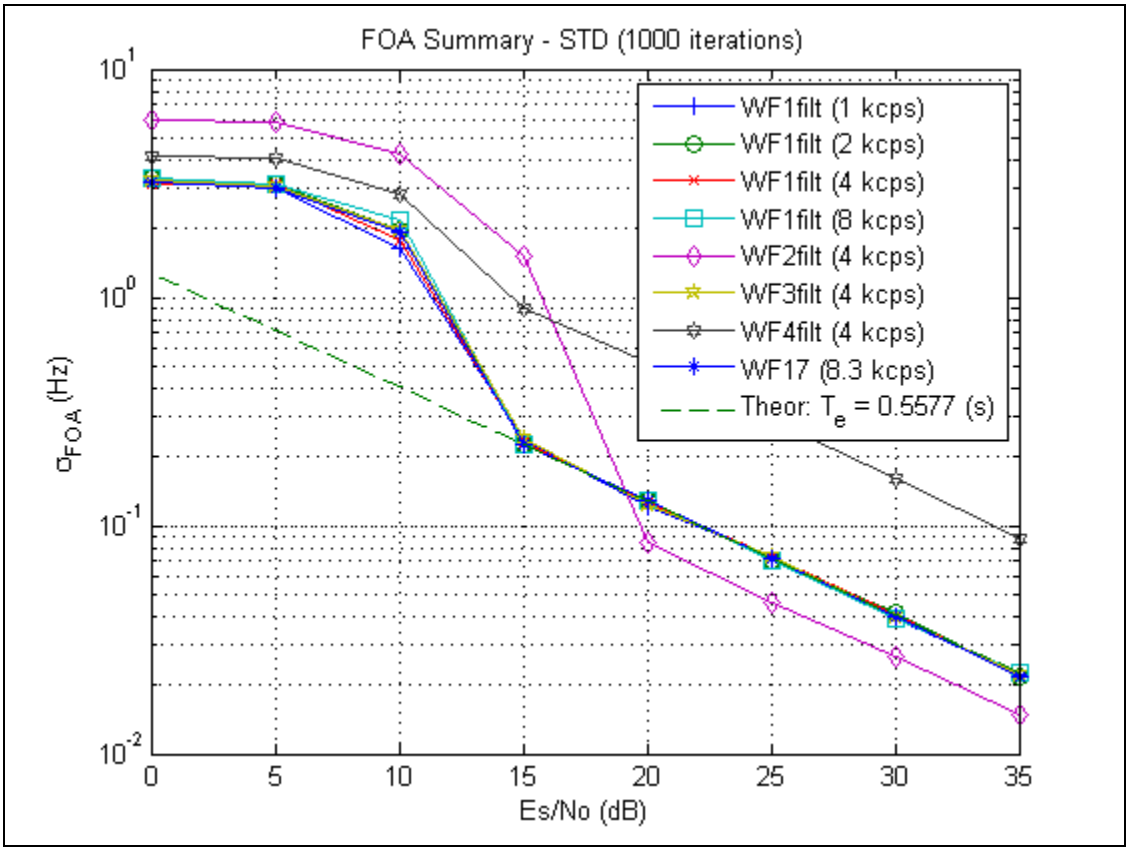

Figure 8 FOA Accuracies – Summary of Alternatives.

This shaping can be performed by filtering (temporal or spectral domain) the signal, synthesizing by adding up component signals of the waveform or otherwise modulating the signal, or by shaping the chipping pulses. One potential cost relative to direct sequence spread spectrum (DSSS) of performing this shaping, however, is potentially greater visibility by an adversary, because shaping the PSD may make the signal more visible at those accentuated frequencies. Another potential cost is forcing the system to deal with a non-constant envelope waveform which can be a challenge in power constrained systems because they typically operate their power amplifiers at or near saturation to improve their power added efficiency (PAE), although techniques are being developed to help alleviate this constraint.

ACKNOWLEDGMENTS

I wish to acknowledge with a huge debt of gratitude my wife, Nela, who ended up bearing so much of the burden in moving, making our temporary house a home and keeping it running, and being so understanding when she could tell my mind was somewhere else. I also thank Zef, Tony, Veronica, Teresa, and Mike for supporting daddy in this endeavor.

I would also like to thank my advisors Frank Kragh and Hersch Loomis for giving me their insights, pointing me in the right directions, and instilling the needed rigor.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The ability to accurately geolocate an object strictly through the use of its radio frequency (RF) emission can support blue force tracking, aid in locating a downed airman, or allow tracking of some object. The numerous techniques which exist to determine the locations of an adversary's signal emitters all involve solving a geometry problem by measuring angles, distances (or differential distances), or otherwise defining relationships in some geometry [1]. While some of these techniques are based solely on measuring the angle of arrival for peak energy detection and are thus waveform independent, others involve measuring the precise time and frequency of arrival of the signals [1], [2]. This thesis examines the effect of waveform parameters on the ability to accurately make these estimates.

When those desiring to geolocate the transmitter also control the design of the transmitter, the waveform should be optimized to support detection and geolocation within the imposed constraints. Examples of systems in which special waveforms are used to support geolocation are navigation systems such as Loran or GPS, which transmit specially designed signals from multiple emitters of known location to allow a receiver to determine its location [2]. This thesis describes the complementary process of geolocating a single emitter using multiple collectors.

A goal of the research was to determine the waveform features one should consider in designing a waveform. These concepts were then applied to develop several example waveforms to demonstrate the effect of each of these parameters and to show how these parameters can be traded off to vary performance.

This analysis makes use of the cross ambiguity function (CAF), which is described later and is a method used to determine the time difference of arrival

(TDOA) and frequency difference of arrival (FDOA) between a signal received by collectors at two locations. [3], [4], [5], [6]

B. OBJECTIVE

The objective of this thesis is to identify the major considerations when designing waveforms to support geolocation and also to develop an understanding of expected geolocation performance where one cannot control the waveform. A waveform should optimize detectability (by the desired collectors) and estimation of the key parameters, time of arrival (TOA) and frequency of arrival (FOA). This optimization must be bounded by real world limitations such as power, bandwidth, and acceptable level of observability by an adversary [15]. Conversely, one could also use the information to minimize the geolocation accuracy of an emission.

Several key assumptions had to be made in this thesis. The first assumption is that multipath does not exist and the only channel impairment is additive white Gaussian noise (AWGN). The second assumption is that the signal is to support geolocation based on a single transmission burst which is received by multiple time-synchronized geographically dispersed collectors all having line of sight visibility to the emitter but no angle of arrival (AoA) capabilities. The final assumption is that the emitter and collectors will undergo very limited relative motion during the burst, and each collector has perfect knowledge of time (i.e., it is coherent with the others) and its own location and velocity.

Chapter V of this thesis proposes several different example waveforms to demonstrate the effects of these features and estimates expected performance of each. Simulations were performed, and the results were compared with theoretical performance estimates.

C. RELATED WORK

This thesis takes advantage of the theoretical work done by Stein [3] on the cross ambiguity function (CAF), which can be used to estimate jointly the time difference of arrival (TDOA) and frequency difference of arrival (FDOA) between signals received by two or more receivers undergoing limited Doppler effects [4]. If sufficient collectors are used, one may be able to use this information to estimate the location of an emitter [2]. Stein presents the CAF and expected accuracy of TDOA and FDOA measurements. This thesis uses [3] to predict the accuracy of time of arrival (TOA) and frequency of arrival (FOA) estimates of a signal that is known a priori by the receivers.

Johnson [5] developed MATLAB® software routines both to implement the CAF and to generate signals as would be received by a pair of independent receivers in a defined collection scenario. The scenario generator allows the user to define the location and velocities of an emitter and two collectors, and the resulting generated signals model the effects of propagation delay, Doppler and noise. This thesis uses the software developed in [5] the simulations performed. The signal generator software is used to synthesize the BPSK waveforms proposed in this thesis, and the CAF algorithms are used to estimate the TOA and FOA of synthesized signals.

D. THESIS ORGANIZATION

This thesis is organized into seven chapters. Chapter II describes the basics of geolocation, identifies the key parameters to be estimated, and discusses figures of merit for geolocation. Chapter III provides a discussion of the factors and constraints that affect geolocation performance but lie outside the control of the waveform developer. Chapter IV quantifies expected performance (e.g., probability of detecting the transmitted burst and standard deviation in the TOA and FOA measurements) and describes the CAF. Chapter V proposes several example waveforms, identifying the rationale for selecting them and the distinguishing features of each. Chapter VI describes the simulation approach

and discusses the MATLAB® code used to perform this processing to assess deviation in TOA and FOA. Finally, Chapter VII presents the simulation results, summarizes the findings of this thesis, and discusses possible follow on efforts.

II. THE GEOLOCATION PROBLEM

A. THE EMITTER

For the sake of bounding the problem, several assumptions are made about the emitter. One basic assumption is that the emitter has no fixed receiver associated with it and it must be able to operate over a large area with neither knowledge of its own location nor that of any of the collectors. This leads to the first assumption: the emitter asynchronously transmits its signal isotropically and any estimate of its location is based strictly on its radio frequency (RF) characteristics.

Second, the emitter should have limited observability to reduce its vulnerability to being detected by an adversary. This topic of low probability of intercept (LPI) or detection (LPD) goes well beyond the scope of this thesis, but the most basic guidelines to be followed are to reduce the power spectral density (PSD) of the signal and to limit the duration and quantity of transmissions. This thesis addresses geolocation based on a single burst of energy.

Third, the collectors know neither the time of this transmission burst nor its exact frequency (although, of course, the frequency must exist within some limited RF band). Each collector does know, however, precise time and its own location. Variability in the frequency can be a result of oscillator drift.

Fourth, the emitter is assumed to be approximately stationary during the transmission burst. Although lack of emitter motion may not always be operationally realistic, this thesis can only briefly discuss the effects of emitter motion.

Finally, although an emitter would likely need to transmit a limited amount of data to identify itself and perhaps some condition or state, this thesis is limited to the case in which the collectors have a priori knowledge of the actual transmission. Examples of such signals include preambles, synchronization patterns, and dataless bursts.

B. METHODS OF PASSIVE GEOLOCATION

The various techniques for geolocating an emitter have existed for many years and all involve solving the geometry between the emitter and the various collectors. Adamy [1] presents five basic approaches; the first of these is *triangulation*, which uses the intersection of lines of bearing from multiple collectors to estimate the emitter's location. The next involves measuring the *angle and distance* from a single site, such as is done with radar. The third approach involves making *multiple distance measurements* (and the variation using time difference of arrival), which involves finding the intersection of arcs of known radii from the various collectors. The fourth approach uses *two angles and known elevation differential*, which finds the intersection of elevation and azimuth angles and a known plane (or terrain map). The fifth approach of using *multiple angle measurements by a single moving collector* against a stationary or slowly moving target is really a variation of the first method. Because the various angle of arrival (AOA) methods are waveform independent, they will not be discussed in this thesis which is addressing waveform issues and will focus on the third of these, multiple distance measurements.

Loomis [2] discusses geolocation of emitters using two collection platforms that make multiple observations of a relatively fixed emitter at various angles from the emitter. Time difference of arrival (TDOA) measurements between the two collectors provide a locus of constant TDOA called an *isochron* ("constant time"), which in 3 dimensions is a hyperboloid of revolution about the axis joining the two collectors. The location of the emitter can be estimated by finding the intersection of the various isochrons, each corresponding to a different observation. This thesis extends the concept to one in which additional geographically distributed collectors can each observe a single transmission. An isochron would then be formed for each pair of collectors, and finding the intersection of these isochrones leads to an estimate of the emitter location.

Likewise, if the collectors have a velocity large enough that the relative Doppler frequency offset is significantly greater than that due to measurement

error or emitter motion, the measurements can provide a locus of constant FDOA, or an *isodop* (short for iso-doppler). Solving for the intersection of all the isochrones and isodops provides an estimate of emitter location. In the presence of measurement error, additional measurements can be made to provide an overconstrained set of equations, which can then be solved to give a minimum-least-squares-error estimate of the position. [2]

All the methods to perform geolocation are attempting to solve a set of simultaneous equations with multiple unknowns. The emitter unknowns are location (x, y, and z), velocity (in the x, y, and z directions), time of emission, and exact frequency of emission. The collectors know their own location (x, y, and z) and velocity (in the x, y, and z directions) and measure the signal's time of arrival (TOA) and frequency of arrival (FOA). If the emitter motion is insignificant, only four unknowns remain, the three position variables and the time of emission. For example, if the emitter is known (or believed) to be on the surface of the earth, only three variables remain to be solved (x position, y position, and time) and all others are known. If the altitude of the emitter is unknown, solving for location in three-dimensional space requires solving for an additional variable. The Global Positioning System (GPS) in fact solves for all four of the variables. [2], [7]

GPS consists of multiple satellites, each broadcasting signals containing precise time and position of the satellites. The time from the various satellites is accurate enough that they can be considered synchronized. If the GPS receiver also had this extrememly accurate time, it would be able to calculate directly the various signal propogation times and thus find its range from each of the satellites. However, the clock on the receiver has an offset, which adds a bias to each of these range calculations. These "pseudorange" estimates are thus the result of the receiver clock error and the time difference of the satellite and receiver clocks. Because the receiver knows the location of each of the satellites from the received signal, it is left with four unknowns consisting of the three position estimates and the receiver clock offset. Receiving the signal from four satellites allows the receiver to calculate these values. [7]

Whether one views the geolocation problem as a version of Loomis' intersection of isochrones or as an inverse GPS approach, relative time and the precise location and velocity of the collectors (or, conversely, the emitters in the case of GPS) must be known.

C. KEY DETECTION PARAMETERS

The previous section indicated that geolocation is dependent on the geometry between the emitter and the collectors, something the waveform cannot control. The waveform, however, does have an effect on the accuracy of estimates of the time difference of arrival (TDOA) and the frequency difference of arrival (FDOA) [3].

Although the collectors do not directly measure TDOA and FDOA, they are assumed to have perfect knowledge of time and can thus make estimates of the absolute time of arrival (TOA) of the received signal. This time of arrival at the nth collector TOA_n is equal to the time the emission is transmitted T_{tx} plus the propagation time $T_{prop,n}$ to that collector, $T_{tx} + T_{prop,n} = TOA_n$. Because the two collectors share a common time reference, TDOA (and FDOA) is simply the difference of the two measurements,

$$\begin{array}{c} T_{tx} + T_{prop,1} = TOA_1 \\ -(T_{tx} + T_{prop,2} = TOA_2) \\ \hline T_{prop1} - T_{prop2} = TOA_1 - TOA_2 = TDOA \end{array}, \tag{2.1}$$

and this value can be used to perform geolocation in the manner indicated by Loomis [2].

Because the focus of this thesis is on the waveform, it identifies those parameters affecting estimation of TOA, primarily, and FOA, secondarily.

D. FIGURES OF MERIT FOR GEOLOCATION ACCURACY

Inherent measurement errors result in a reduction in accuracy in the location estimate [1], [2]. Without discussing the sources of these errors, this section summarizes some of the metrics used to quantify the accuracy of a

geolocation estimate. Because the system accuracies take into account many factors beyond the inherent limitations of the waveform, and thus go beyond the scope of this thesis, this information is provided as reference, and waveform variations are not projected back to geolocation accuracies.

A basic figure of merit for position accuracy is the *confidence ellipse*, an ellipse that outlines the area, e.g., on the surface of the earth, containing the emitter with a probability of $1-P_e$ and can be computed from an over-constrained matrix of measurements [2]. Thus one can speak of a “90% confidence ellipse”, i.e., 10% probability the emitter is really outside this ellipse, or a “50% confidence ellipse” by defining the center of the ellipse along with the major axis and minor axis. Thus, a smaller ellipse indicates a greater certainty of emitter position, i.e., increased accuracy.

Among other metrics of location accuracy are *2 drms*, Circular Error Probable (CEP), and Spherical Error Probable (SEP). Reference [8] typically designates accuracy in terms of *2 drms*, which is defined as $2\sqrt{\sigma_N{}^2+\sigma_E{}^2}$ when referring to horizontal positioning where $\sigma_N{}^2$ and $\sigma_E{}^2$ are the variances of the north and east position estimates respectively. It further states that in actuality, the percentage of horizontal positions, e.g., on the surface of the Earth, contained within the area specified by the 2 drms value varies between approximately 95.5 and 98.2 percent depending on the eccentricity of the ellipse of the error distribution [8].

The CEP, which specifies the area defined by a scaled ellipse $0.589\left(\sigma_N+\sigma_E\right)$, where σ_N and σ_E are the rms errors in the estimated user position coordinates along the nort and east axis, and is the same as the confidence ellipse with $P_e=0.5$ [8]. Although called ‘circular’, CEP is really elliptical unless the various variances are the same and the angles from the emitter to the various collectors are all 90º apart from each other as indicated in

Figure 9 [1]. It is sometimes called elliptical error probable (EEP) [1]. If the positioning errors have a circular normal distribution, then 2 drms = 2.4 CEP [8].

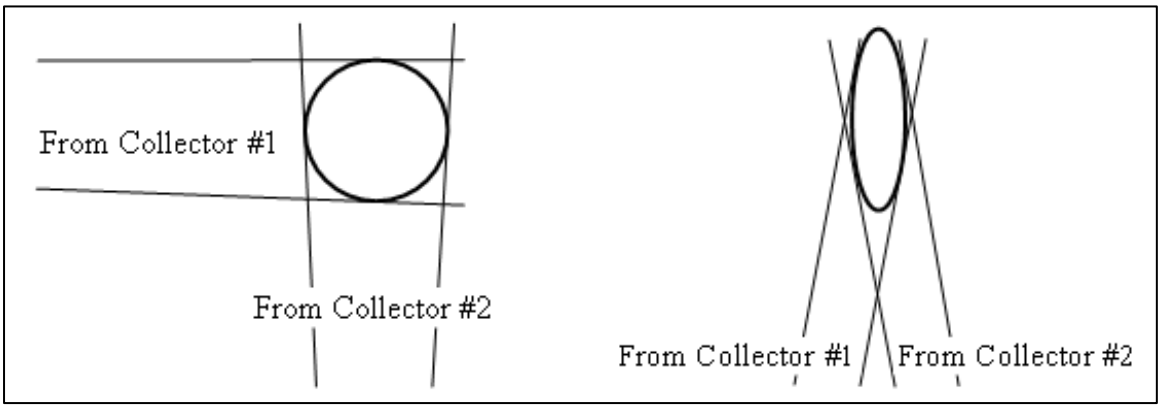

Figure 9 Eccentricity of Ellipse for CEP is Geometry Dependent (after [1]).

SEP defines a volume containing the emitter with a probability of 0.5. As opposed to the previous measures which define an area on a plane, the SEP requires the addition of a vertical element and is defined to be $0.513\left(\sigma_N + \sigma_E + \sigma_h\right)$ [8] where σ_h is the square root of the variance of the height. SEP is truly 'spherical' only when $\sigma_N = \sigma_E = \sigma_h$.

This chapter defined the geolocation problem by identifying assumptions about the transmission, presenting passive geolocation techniques, identifying the key detection parameters of TOA and FOA, and listing figures of merit for geolocation. The next chapter identifies and discusses parameters that can affect geolocation performance but are not waveform-related.

III. PARAMETERS BEYOND THE CONTROL OF THE WAVEFORM DEVELOPER THAT AFFECT GEOLOCATION PERFORMANCE

A. NON-WAVEFORM PARAMETERS TO CONSIDER

The waveform parameters are only a subset of the factors affecting the accuracy of the geolocation estimate. Among other factors are the collection geometry (i.e., the geometric relationship between the location of the emitter and the locations of the various collectors), variations in the propagation delay, clock errors, and collector location errors.

The position error is highly dependent on the position of the collectors relative to the emitter. For example, if the distance between an emitter and collector is large, even a small error in angular estimate can result in a significant location error as illustrated in Figure 10. As illustrated in Figure 9, the size and orientation of the confidence ellipse depends on the relative angle between the emitter and the various collectors.

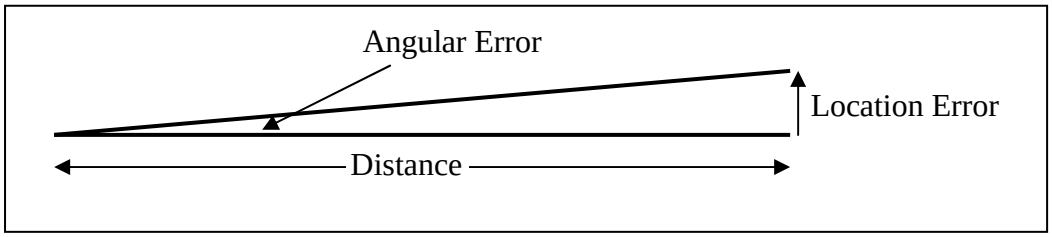

Figure 10 Relation Between Angular and Location Errors (from [1]).

Analysis of the degradation of geolocation precision due to geometry has been well developed for the GPS system [9], [10], which is a complement to our geolocation problem (i.e., multiple emitters received by a single collector vs. a single emission received by multiple collectors). Spilker [9] shows that geometric dilution of precision (GDOP) for a three-dimensional position with four satellites can be minimized by maximizing the volume of a tetrahedron formed by the unit vectors in the direction of each of the satellites.

Because the scope of this thesis is to perform geolocation primarily on TOA, the error sources would be expected to be similar to the ranging errors for GPS. Parkinson [9] identified six classes of these errors:

- Error in knowledge of collector locations and velocities,
- Error in knowledge of the time of emission,
- Ionospheric propagation effects,
- Tropospheric propagation effects,
- Multipath, and
- Receiver sources of error.

These error sources are beyond control of the waveform but would need to be considered at the system level.

The first two items in the bulleted list above would correspond to errors in knowledge of the positions and velocities of the collectors and any time reference errors they may have. As an example of accuracies achievable with the GPS system, root mean square (rms) ranging errors for GPS (in 1984) attributable to ephemeris error, the difference between actual satellite location and reported location, was 2.1 m for satellite ephemeris data up to 24 hours old. Likewise, the resulting positional error due to clock errors (also in 1984) was 4.1 m for 24-hour predictions and 1-2 m is expected for 12-hour updates of the GPS clock. [10]

The next two items, ionospheric and tropospheric propagation effects cause error in the range estimate because of variations in the velocity of light as the radio signal passes through them, caused by varying number of free electrons in the ionosphere and variations in temperature, pressure, and humidity in the troposphere. Ionospheric group delay can be approximated to the first order by $\Delta t_{ion}(f) = \frac{40.3}{f^2}\mathrm{TEC}$ where TEC is the time and spatially varying total electron count (sometimes called total electron content) and f is the carrier frequency [11]. TEC is the total number of electrons in a 1-m^2 cross-sectional tubing along the path of transmission through the ionosphere [11], with units of electrons per square meter, where 10^{16} electrons/m^2 = 1 TEC unit (TECU) [12].

World-wide TEC values can be viewed in near real-time from the Internet [13]. Figure 11 shows an example of one of these TEC maps. Effective accuracies with simple modeling are about 2-5 m for the ionosphere and about 1 meter for the troposphere [10].

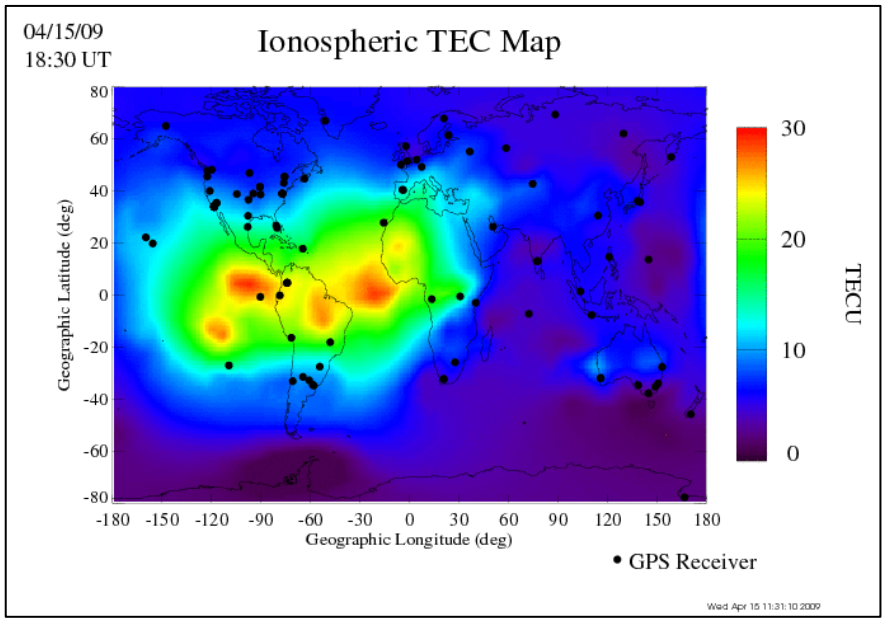

Figure 11 Example TEC map (from [13])

The magnitudes of the final two sources of error are largely a function of the receiver design. Although the receiver cannot prevent multipath, the processing approach can reduce its impact if the signal can be tracked, not something supported by a burst transmission. As a reference, GPS error is typically less than 1 m under most circumstances for the multipath and less than 0.5 m for the receiver error. [10]

Parkinson [10] summarizes all these error sources for GPS in Table 2. Note that he breaks out horizontal & vertical accuracies separately and these include values for dilution of precision (DOP), which are metrics defining the degradation from ideal due to geometry and need to be stated to indicate the assumptions for under which the errors are determined. The two variations of DOP used in the figure are vertical dilution of precision, VDOP, equal to 2.5 and

horizontal dilution of precision, HDOP, equal to 2.0. Parkinson breaks out each source of error into components referred to as bias, which is non-zero mean over a limited time or geographical area, and random which is zero mean. The table is useful for showing the relative contribution of the various error sources as well as the absolute values of an example system that estimates location. To give some context on timing accuracy required, an error of 1 m corresponds to a timing error of approximately 33 ns using $t = \frac{d}{c} = \frac{1\text{ m}}{3\cdot 10^8\text{ m/s}} = 33\text{ ns}$.

Table 2 GPS Standard Errors (from [10]).

	Standard Deviation, m		
Error Source	Bias	Random	Total
Ephemeris data	2.1	0.0	2.1
Satellite Clock	2.0	0.7	2.1
Ionosphere	4.0	0.5	4.0
Troposphere	0.5	0.5	0.7
Multipath	1.0	1.0	1.4
Receiver Measurement	0.5	0.2	0.5
User equivalent range error (UERE), rms	5.1	1.4	5.3
Filtered UERE, rms	5.1	0.4	5.1
Vertical one-sigma errors – VDOP=2.5			**12.8**
Horizontal one-sigma errors – HDOP=2.0			**10.2**

B. WAVEFORM CONSTRAINTS

The waveform is subject to design constraints that limit the features it may have and will limit the performance possible. Key limitations include observability by an adversary, required detection and false alarm rates, and operational physical considerations.

Observability refers to the ability of an adversary to detect or intercept the transmitted signal. For signals of low power spectral density, unless an adversary has knowledge of the signal structure, he cannot do significantly better than using an energy detector (Figure 12), a power detector followed by an integrator. In use, the energy detector would be preceded by a bandpass filter and followed by a thresholder. Another type of energy detector is the two-receiver correlation radiometer, in which two inputs are multiplied together and the product is smoothed with a low-pass filter. [15]

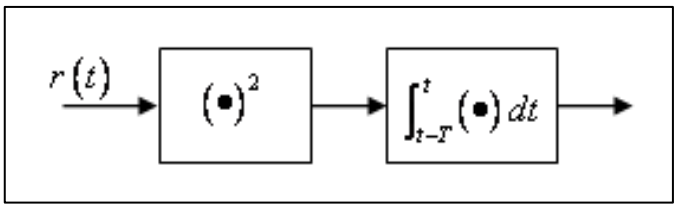

Figure 12 Basic Radiometer (after [15]).

The two-receiver correlation radiometer, Figure 13, is similar to the CAF processing approach (discussed in Chapter IV) in that both have separate antennas and receiver front ends to allow noise to be independent, and the two signals are multiplied by each other and undergo low-pass filtering. The CAF processor, however, allows the time between the signals to be offset and can compensate for the frequency offset between the two receivers.

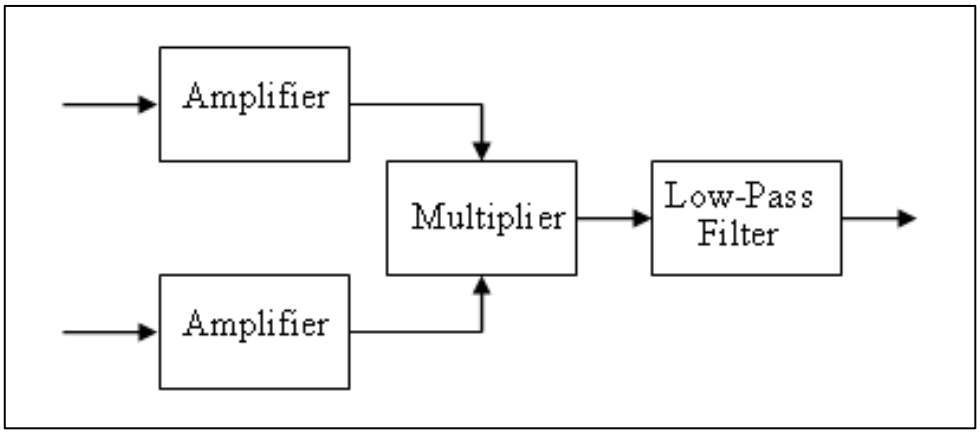

Figure 13 Basic Two-Receiver Correlation Filter (from [15]).

The ability of an interceptor to detect a signal depends on not only the format and strength of the signal relative to the background noise, but also on how much knowledge he has of the signal and how dedicated he is to detecting it. Among the knowledge that generally helps detection are carrier frequency, bandwidth, time, and any fundamental components of the waveform such as PN code or data bits and timing. Additional features that can make a signal harder to detect in the presence of noise are time-hopping, frequency-hopping, and frequency spreading (DSSS or frequency sweep). [15]

Certain features of a waveform may be exploited to increase its detectability. One technique useful against BPSK modulated waveforms (including DSSS) of sufficient signal-to-noise is to square the signal and look for the second harmonic of the modulated carrier. Other techniques exploit the statistical properties of man-made signals known as cyclostationarity, which show themselves as periodic components in the mean and autocorrelation functions in signals of sufficient signal power to noise power ratio (SNR) [16].

In addition to managing the observability of a signal, the waveform developer must work within the limitations specified for probability of detection P_d (by the desired receiver) within the context of a maximum probability of false alarm P_{fa}, which is covered in more detail in Chapter IV.

Finally, the waveform must operate within the operational limitations such as power (e.g., battery life) and spectrum allocation. For example, systems often use constant envelope waveforms because they can be transmitted using with high power-added efficiency (PAE) amplifiers operating near saturation (e.g., traveling wave tube or class “C” devices) [17], but new techniques in non-linear amplifiers may allow waveform freedom without sacrificing power efficiency [18].

Although many constraints and limitations (both requirements and “desirements”) are placed upon the waveform, others need to be defined. The next chapter discusses the effects of waveform parameters on the resulting performance.

IV. PERFORMANCE ESTIMATES

The previous chapter identified various factors affecting geolocation that are beyond the control of the waveform (e.g., collection geometry) and constraints placed upon the waveform (e.g., bandwidth). This chapter identifies expected performance of a waveform including detection by the intended receiver and the ability to support accurate estimation of time and frequency of the received signal.

Determining the TOA and FOA of a signal is a two-step process, first detecting the signal (i.e., detection) and then estimating the TOA and FOA values of the detected signal. This chapter develops performance estimates for detection and false alarm and TOA and FOA estimation.

A. DETECTION

This section develops performance estimates for probability of detection and probability of false alarm using both a coherent receiver and a non-coherent receiver. For each type of receiver, the processing is mathematically described for a BPSK modulated direct sequence spread spectrum (DSSS) signal, the output statistics are derived, and the performance for detection and false alarm probabilities are developed in the presence of additive white Gaussian noise (AWGN).

1. Coherent Detection

Coherent detection is the process of attempting to detect a signal that is frequency and phase synchronized with the carrier of the receiver [19]. Any loss of synchronization may degrade performance. Although perfect synchronization may be an unrealistic real-world situation, it allows the derivation of the optimal performance.

a. Receiver Processing

Figure 14 shows a coherent receiver where $r(t)$ is the received signal, $s(t)$ is the reference signal, and X is the resulting decision variable at the end of each integration time. The received signal $r(t)$ is composed of the sum of the desired signal $s(t)$ and noise $n(t)$ such that $r(t) = s(t) + n(t)$. The product of the received and reference signals is integrated over the period of interest and then sampled to produce the decision variable X.

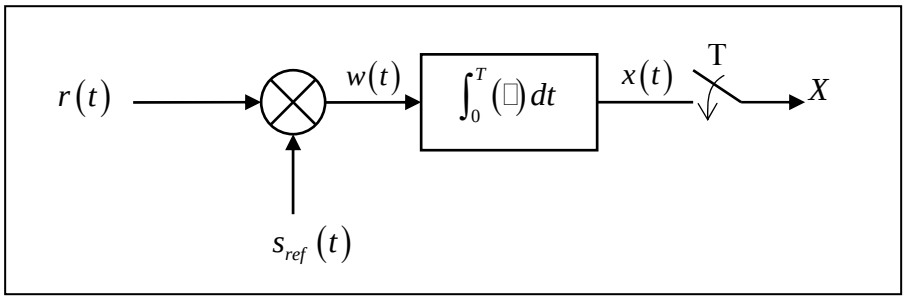

Figure 14 Coherent Receiver (after [20]).

Let the reference signal $s_i(t)$ be a BPSK modulated direct sequence spread spectrum (DSSS) signal,

$$s_{ref}(t) = 2c(t)\cos(2\pi f_c t), \tag{4.1}$$

where $c(t) \in \{-1,1\}$ is the chip sequence used to modulate the carrier and f_c is the carrier frequency. If the received signal is at the same frequency and in phase with s_{ref}, then

$$r(t) = A_c c(t)\cos(2\pi f_c t) + n(t), \tag{4.2}$$

in which $n(t)$ is additive white Gaussian noise (AWGN) with power spectral density (PSD) equal to $N_0/2$ and A_c is the magnitude of the signal carrier. The resulting decision variable X is

$$X = \int_0^T r(t)\, s_{ref}(t)\, dt$$
$$= A_c T + A_c \frac{\sin 4\pi f_c T}{4\pi f_c} + \int_0^T 2c(t)n(t)\cos 2\pi f_c t \; dt \tag{4.3}$$

which simplifies to

$$X = A_c T + \int_0^T 2c(t)n(t)\cos 2\pi f_c t \; dt \tag{4.4}$$

if $f_c = m/2T$ or $f_c \gg 1/T$. [21]

b. Decision Variable Statistics

The mean value of the output decision variable X shown in (4.4) is

$$\overline{X} = E\left[A_c T + \int_0^T 2c(t)n(t)\cos 2\pi f_c t \; dt \right] = \overline{X_s} + \overline{X_n} \tag{4.5}$$

where $\overline{X_s}$ is the contribution to the mean from the signal input $s(t)$ and $\overline{X_n}$ is the contribution from the noise input $n(t)$.

$$\overline{X_s} = E\left[A_c T\right] = A_c T \tag{4.6}$$

because the signal is deterministic, and

$$\overline{X_n} = E\left[\int_0^T 2c(t)n(t)\cos 2\pi f_c t \; dt \right] = 2\int_0^T E\left[c(t)n(t)\right]\cos 2\pi f_c t \; dt = 0 \tag{4.7}$$

because the chip sequence is independent of the noise, which has zero mean.

Thus the mean of X is

$$\overline{X} = \overline{X_s} + \overline{X_n} = A_c T \text{ . [21]} \tag{4.8}$$

The variance of X is

$$\sigma^2 = E\left[\left(X - \overline{X} \right)^2 \right] = E\left[\int_0^T 2c(t)n(t)\cos 2\pi f_c t \; dt \int_0^T 2c(\tau)n(\tau)\cos 2\pi f_c \tau \; d\tau \right]$$
$$= E\left[\int_0^T \int_0^T 4c(t)c(\tau)n(t)n(\tau)\cos 2\pi f_c t \cos 2\pi f_c \tau \; dt d\tau \right]$$
$$= \frac{N_0}{2} \int_0^T 4c(t)^2 \cos^2 2\pi f_c t \; dt \tag{4.9}$$
$$= N_0 \int_0^T \left(1 + \cos 4\pi f_c t\right) dt$$
$$= N_0 T + \frac{\sin 4\pi f_c T}{4\pi f_c} \text{ ,}$$

using the property of AWGN that $E\left[n(t)n(\tau)\right]=\frac{N_0}{2}\delta(t-\tau)$ [21].

This further reduces to

$$\sigma^2 = N_0 T \tag{4.10}$$

if $f_c = m/2T$, where m is an integer, or $f_c \gg 1/T$.

The probability distribution of the output X is Gaussian because $n(t)$ is Gaussian and the integration is a linear process, and thus it has the probability density function

$$f_X(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left[\frac{-\left(x-\overline{X}\right)^2}{2\sigma^2}\right] \text{ [20].} \tag{4.11}$$

The probability density function (pdf) of the detection variable depends on whether the signal was transmitted. The variance σ^2 is independent of whether the signal is present (4.9); however, the mean when no signal exists (4.7) is different from that when the signal is present. Thus, $f_{X|0}$ and m_0 represent the pdf and mean when no signal is present, and $f_{X|1}$ and m_1 represent the pdf and mean when the signal is present, where $m_0 = 0$ and $m_1 = \overline{X_s}$. The area under each of the curves is unity, and they have same width. [22]

c. Probability of Detection and False Alarm

The decision variable statistics allow one to determine the various detection probabilities. A detection is declared if the decision variable X coming out of the receiver (Figure 14) exceeds a threshold V_T . The probability of declaring a detection is thus

$$\Pr\left(X > V_T\right)=\int_{V_T}^{\infty} f_X(x)\,dx \text{ [22], [23],} \tag{4.12}$$

where V_T is the threshold value.

Figure 15 shows the probability density function (pdf) of X when no signal is present $f_{X|0}$ and the pdf when the signal is present $f_{X|1}$. A detection is declared both when a signal is present and detected, called a "detection", and also when no signal is present but the noise causes X to exceed the threshold V_T, called a "false alarm." The probability of a false alarm P_{FA} corresponds to the area to the right of V_T under the first curve and shown in gray, and the probability of a detection P_d corresponds to the area to the right of V_T under the second curve, i.e., all the area under the second curve except that in black. The area in black is $1-P_d$ and is referred to as the probability of a miss. Thus, increasing the threshold, i.e., moving V_T to the right, reduces the probability of a false alarm P_{FA}, but it also reduces the probability of detecting a valid signal P_d for a fixed SNR. Conversely, decreasing the threshold increases the probability of a false alarm P_{FA} but also increases the probability of detecting a valid signal P_d for a fixed SNR. [22], [23], [24]

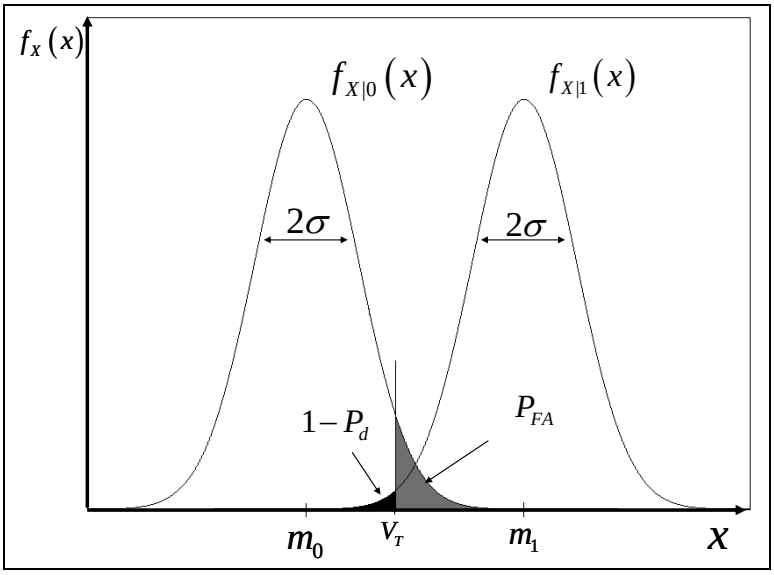

Figure 15 Coherent Probability Distribution Functions (pdf) (after [22]).

Ideally, one wants to detect all signals (i.e., $P_d = 1$) and have no false alarms (i.e., $P_{FA} = 0$), but this is not possible because $f_X(x)$ is never equal to zero. To reduce this range of ambiguity, either $f_X(x)$ must be narrower by making σ smaller by reducing the noise, or the difference between them must be made larger, i.e., increasing the difference between m_0 and m_1, by increasing signal energy [22], [24].

The probability of a false alarm is mathematically defined as

$$P_{FA} = \Pr(X > V_T \mid 0) = \int_{V_T}^{\infty} f_{X|0}(x)\,dx \text{ [22], [23]} \tag{4.13}$$

where

$$f_{X|0}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-\left(x - \overline{X_n}\right)^2}{2\sigma^2}\right] \text{ [22]} \tag{4.14}$$

in which $\overline{X_n} = 0$ as shown in (4.7). Thus

$$P_{FA} = \int_{V_T}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-x^2}{2\sigma^2}\right] dx \tag{4.15}$$

for which no closed form expression exists [23]. However, applying the variable substitution $\lambda = x/\sigma$ gives

$$P_{FA} = \frac{1}{\sqrt{2\pi}} \int_{V_T/\sigma}^{\infty} \exp\left[\frac{-\lambda^2}{2}\right] d\lambda\,, \tag{4.16}$$

which is now the form of the Q-function, which is defined as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-\xi^2/2} d\xi\,, \tag{4.17}$$

for which equations to approximate this and lookup tables have been created, although these approximations and tables assume $x \geq 0$ [23]. Combining (4.16) and (4.17) and applying (4.10) leads to the final expression for P_{FA} in terms of the Q-function as

$$P_{FA} = Q\left(V_T/\sigma\right) = Q\left(V_T/\sqrt{N_0 T}\right)$$
$$= Q\left(\sqrt{\frac{V_T^{\,2}}{N_0 T}}\right), \tag{4.18}$$

which mathematically confirms the conclusion reached earlier that the probability of a false alarm P_{FA} will reduce as the threshold V_T increases or as the product of noise PSD and integration time, $N_0 T$, decreases.

In a similar manner, the probability of valid detection P_d is the probability of declaring a detection when the signal is indeed present,

$$P_d = \Pr\left(X > V_T \,|\, 1\right) = \int_{V_T}^{\infty} f_{X|1}\left(x\right) dx \tag{4.19}$$

where

$$f_{X|1}\left(x\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-\left(x - \overline{X_n}\right)^2}{2\sigma^2}\right]. \tag{4.20}$$

Using (4.8) and the substitution variable $\lambda = \left(x - A_c T\right)/\sigma$,

$$P_d = \int_{V_T}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{-\left(x - A_c T\right)^2}{2\sigma^2}\right] dx$$
$$= \int_{\frac{V_T - A_c T}{\sigma}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[\frac{-\lambda^2}{2}\right] d\lambda$$
$$= Q\left(\frac{V_T - A_c T}{\sigma}\right) \tag{4.21}$$
$$= 1 - Q\left(\frac{A_c T - V_T}{\sigma}\right).$$

.

Finally, substituting (4.10) into (4.21) gives

$$P_d = 1 - Q\left(\frac{A_c T - V_T}{\sqrt{N_0 T}}\right), \tag{4.22}$$

where V_T is the detection threshold. Note that the probability of detect P_d is

waveform independent and is a function of only the signal amplitude A_c, the noise power spectral density $\frac{N_0}{2}$, the integration time T, and the detection threshold V_T.

d. An Example

Suppose one wanted to establish the threshold for a P_{FA} of once per day for a system with a 1 MHz sampling frequency in which the received signal will be at the same frequency and in phase with the reference signal. The detector makes a threshold decisions for each sample. The resulting P_{FA} per sample is

$$P_{FA} = \frac{1}{\text{day}} \frac{\text{day}}{24\ \text{hr}} \frac{\text{hr}}{3600\,\text{sec}} \frac{\text{sec}}{10^6\ \text{S}} = 1.157 \times 10^{-11} \text{ per Sample.} \tag{4.23}$$

One Can solve for V_T using (4.18) and a Q-table to find

$$\begin{aligned} & P_{FA} = Q\left(V_T/\sigma\right) = Q\left(V_T/\sqrt{N_0 T}\right) = 1.157 \times 10^{-11} \\ & \Rightarrow V_T/\sqrt{N_0 T} = 6.685 \\ & \Rightarrow V_T = 6.685\sqrt{N_0 T}\ . \end{aligned} \tag{4.24}$$

Now supposing the requirement is to detect 99% of the emissions, then one can in a similar manner solve for V_T using (4.22), such that

$$\begin{aligned} & P_d = 1 - Q\left(\frac{A_c T - V_T}{\sqrt{N_0 T}}\right) = 0.99 \\ & \Rightarrow Q\left(\frac{A_c T - V_T}{\sqrt{N_0 T}}\right) = 0.01 \\ & \Rightarrow \frac{A_c T - V_T}{\sqrt{N_0 T}} = 2.325 \\ & \Rightarrow V_T = A_c T - 2.325\sqrt{N_0 T} \end{aligned} \quad . \tag{4.25}$$

Equating the two expressions for V_T and solving to find the required ratio of signal power to noise power SNR using $A_c^2 T / N_0 = 2E / N_0 = 2SNR$ [24], where E is the energy in the pulse, gives

$$\begin{aligned} & A_c T - 2.325\sqrt{N_0 T} = 6.685\sqrt{N_0 T} \\ & \Rightarrow A_c T = 9.01\sqrt{N_0 T} \\ & \Rightarrow 9.01 = \frac{A_c T}{\sqrt{N_0 T}} = \sqrt{\frac{A_c^2 T}{N_0}} = \sqrt{\frac{2E}{N_0}} = \sqrt{2SNR} \\ & \Rightarrow SNR = 9.01^2 / 2 = 40.6 = 16.1 \text{ dB}. \end{aligned} \tag{4.26}$$

2. Noncoherent Detection

The previous section described coherent processing; however, in the real world, even if one knew the exact frequency of the received signal he would not know the phase of the signal. This section addresses a non-coherent strategy to detect a signal of unknown phase.

a. Receiver Processing

The noncoherent receiver shown in Figure 16 consists of two receiver arms in which the squared outputs are summed and where the reference signals $s_1(t) = 2c(t)\cos(2\pi f_c t)$ and $s_2(t) = -2c(t)\sin(2\pi f_c t)$ are orthogonal [24]. This summed signal is then sampled to provide the decision variable, or the receiver may take the square root of the summed signal as shown in Figure 16. The resulting distribution of the decision variable with signal present is non-central Chi-squared with two degrees of freedom for the case of the sum of the squares or Ricean in the case where the square root is taken [20], [23]. The resulting distribution for the case with no signal, i.e., noise only, is central Chi-squared with two degrees of freedom or Rayleigh for the case in which the decision variable is the sum of the squares or the square root of this sum, respectively [20], [25].

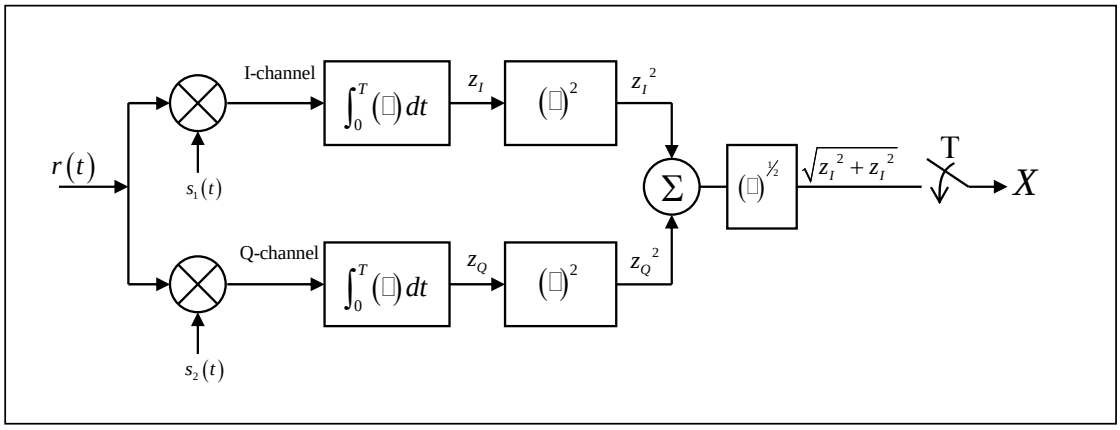

Figure 16 Noncoherent Receiver (after [20]).

b. Probability of Detection and False Alarm

For the detector that is basing its decision on the magnitude of the signal, i.e., $\sqrt{z_I^2 + z_I^2}$, it can be shown that

$$P_d = Q\left[\frac{A_c}{\sigma^2}, \sqrt{2\ln\left(\frac{1}{P_{fa}}\right)}\right], \tag{4.27}$$

where

$$Q(\alpha,\beta) = \int_\beta^\infty \xi I_0(\alpha\xi) e^{-\left(\xi^2+\alpha^2\right)/2} d\xi \tag{4.28}$$

is called Marcum's Q-function [23].

When the probability of false alarm P_{FA} is small and probability of detection P_d is relatively large, (4.27) can be approximated as

$$P_d \approx F\left[\frac{A_0}{\sigma} - \sqrt{2\ln\left(\frac{1}{P_{fa}}\right)}\right] \tag{4.29}$$

where

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\xi^2/2} d\xi \quad \text{[23].} \tag{4.30}$$
$$= 1 - Q(x)$$

Applying (4.29) for $P_{FA} \approx 1\times10^{-11}$ and $P_d = 0.99$ as in the coherent example and using an $F(\square)$ table, such as Table B-1 in [23], gives

$$\begin{aligned} P_d &\approx F\left[\frac{A_c}{\sigma} - \sqrt{2\ln\left(\frac{1}{1.2\times10^{-11}}\right)}\right] = 0.99 \\ \Rightarrow \frac{A_c}{\sigma} &- \sqrt{2\ln\left(\frac{1}{1.2\times10^{-11}}\right)} = 2.33 \quad . \\ \Rightarrow \frac{A_c}{\sigma} &= 2.33 + \sqrt{2\ln\left(\frac{1}{1.2\times10^{-11}}\right)} = 9.4 \end{aligned} \tag{4.31}$$

Finally, the required SNR can be found to be

$$\begin{aligned} SNR &= \frac{A_c^2}{2\sigma^2} = \frac{(9.4)^2}{2} = 44.2 \text{ [25].} \\ &\approx 16.5 \text{ dB} \end{aligned} \tag{4.32}$$

B. FREQUENCY AND TIME ESTIMATION

The previously developed estimates of P_{FA} and P_d assumed that the received signal was at the same frequency, but the signal is likely to have a frequency offset because of Doppler shifts[1] or oscillator drift. This section addresses the joint detection of time and frequency offset between a received and a desired signal.

1. The Complex Ambiguity Function (CAF)

The coherent receiver shown in Figure 14 is a matched filter or correlator receiver and can be mathematically described as

$$X(\tau) = \int_0^{\tau} r(t)s(T-\tau+t)dt \text{ [24],} \tag{4.33}$$

where T is the integration time and τ is the time offset between signals, provides the maximum SNR at the filter output when $\tau = T$ in AWGN [24]. Setting $t = T$ in (4.33) and generalizing for complex variables results in

[1] Doppler shift is really an approximation for a "narrowband" signal in which relative motion exists between the transmitter and receiver. In reality, the Doppler frequency shift varies across the bandwidth and the modulating signal experiences compression or dilation.

$$X(T) = \int_0^T r(t)s^*(t)dt \text{ [19].} \tag{4.34}$$

Finding the resulting value of τ when searching for a peak magnitude (i.e., $\left|X(\tau)\right|_{\max}$) is a reasonable method to find the best approximation of time of arrival for the signal.

The ambiguity function, sometimes referred to as the complex ambiguity function [3] or the cross ambiguity function [5], [6], as presented by Stein [3] is very similar to (4.34), but with the addition of a complex exponential factor is

$$A(\tau, f) = \int_0^T s_1(t)\, s_2^*(t+\tau) e^{-j2\pi ft} dt \,, \tag{4.35}$$

where $s_1(t)$ and $s_2(t)$ are the two received signals in *analytic* form containing a common component, while τ and f are arbitrary time lag and frequency offsets.

The similarity of the cross ambiguity function (CAF) (4.35) to the correlation receiver (4.34) can be shown as follows. Let $s_1(t) = s_L(t)e^{j2\pi f_1 t} + n_1(t)$ and $s_2(t) = s_L(t+\tau)e^{j2\pi f_2 t} + n_2(t)$ where $s_L(t)$ is the complex modulating signal, τ is the difference in propagation times, f_1 and f_2 are the respective apparent carrier frequencies, and $n_i(t)$ is the noise received by the i^{th} collector. Putting this all together results in

$$\begin{aligned}
A(\tau, f) &= \int_0^T s_1(t)\, s_2^*(t+\tau) e^{-j2\pi ft} dt \\
&= \int_0^T \left[s_L(t)e^{j2\pi f_1 t} + n_1(t) \right] \left[s_L(t+\tau)e^{j2\pi f_2 t} + n_2(t) \right]^* e^{-j2\pi ft} dt \\
&= \int_0^T \left[s_L(t)e^{j2\pi f_1 t} + n_1(t) \right] \left[s_L^*(t+\tau)e^{-j2\pi f_2 t} + n_2^*(t) \right] e^{-j2\pi ft} dt \\
&= \int_0^T \left[\begin{array}{l} s_L(t)\, s_L^*(t+\tau) e^{j2\pi(f_1 - f_2)t} + n_1(t)\, s_L^*(t+\tau) e^{-j2\pi f_2 t} \\ \quad + n_2^*(t)\, s_L(t) e^{j2\pi f_1 t} + n_1(t)\, n_2^*(t) \end{array} \right] e^{-j2\pi ft} dt
\end{aligned} \,, \tag{4.36}$$

which simplifies to

$$A(\tau) = \int_0^T s_L(t)\, s_L^*(t+\tau)\, dt = R_{S_L}(\tau) \tag{4.37}$$

in the presence of no noise and when $f = f_1 - f_2$. The peak amplitude of ambiguity function occurs when $A(\tau, f) = \int_0^T s_L(t) s_L^*(t+\tau)\, dt = R_{S_L}(0)$ [23]. Thus, one can search for the values of τ and f which cause $|A(\tau, f)|$ to peak to find the TOA and FOA of a received signal.

2. Theoretical Performance

Stein [3] presents the expected accuracy of the time difference of arrival (TDOA) and frequency difference of arrival (FOA) estimates between two signals in terms of the standard deviation for each. Because, the time of the reference signal inside the receiver is known and can be declared to be zero, his equations can take the forms:

$$\sigma_{TOA} = \frac{1}{\beta} \frac{1}{\sqrt{BT\gamma}} \tag{4.38}$$

and

$$\sigma_{FOA} = \frac{1}{T_e} \frac{1}{\sqrt{BT\gamma}} \tag{4.39}$$

where

B is the noise bandwidth at the receiver input,

T is the integration time of the signal,

β is the “rms radian frequency” of the signal spectrum, detailed below,

T_e is the “rms integration time” of the signal, detailed below, and

γ is the effective input signal to noise ratio.

Each of these is further defined in [3] as follows. The input signal to noise ratio γ is calculated using

$$\frac{1}{\gamma} = \frac{1}{2}\left[\frac{1}{\gamma_1} + \frac{1}{\gamma_2} + \frac{1}{\gamma_1\gamma_2}\right] \tag{4.40}$$

where γ_1 and γ_2 are the signal-to-noise ratio for each of the respective received signals. By definition, the rms radian frequency β is

$$\beta = 2\pi \left[\frac{\int_{-\infty}^{\infty} f^2 W_s(f)\, df}{\int_{-\infty}^{\infty} W_s(f)\, df} \right]^{1/2} \tag{4.41}$$

where $W_s(f)$ is the signal power spectral density, as shaped by the receiver and centered about zero. And the rms integration time T_e is

$$T_e = 2\pi \left[\frac{\int_{-\infty}^{\infty} t^2 \left|u(t)\right|^2 dt}{\int_{-\infty}^{\infty} \left|u(t)\right|^2 dt} \right]^{1/2} \tag{4.42}$$

where $\left|u(t)\right|$ is centered about zero.

The rms radian frequency β is similar to the what is referred to as rms bandwidth B_{rms}, defined to be "the square root of the second moment of a properly normalized form of the squared amplitude spectrum of the signal about a suitably chosen point," which is often used because it facilitates mathematical evaluation better than other definitions of bandwidth [19]. Thus β is $\beta = 2\pi B_{rms}$. Likewise, the definition for rms integration time T_e has a form similar to what is sometimes referred to in literature as the rms duration T_{rms} [19], where the relationship between the two is $T_e = 2\pi T_{rms}$. This thesis uses the terms laid out by Stein, β and T_e.

For example, if the signal has a flat PSD of amplitude 1 over the spectrum from $-B_s/2$ to $+B_s/2$, where B_s is the signal bandwidth,

$$\begin{aligned} \beta = 2\pi \left[\frac{\int_{-\infty}^{\infty} f^2 W_s(f)\, df}{\int_{-\infty}^{\infty} W_s(f)\, df} \right]^{1/2} &= 2\pi \left[\frac{\int_{-B_s/2}^{B_s/2} f^2 df}{\int_{-B_s/2}^{B_s/2} df} \right]^{1/2} . \\ &= \frac{\pi}{\sqrt{3}} B_s \approx 1.8 B_s \end{aligned} \tag{4.43}$$

This leads to

$$\sigma_{TOA} \approx \frac{1}{1.8B_s}\frac{1}{\sqrt{BT\gamma}} = \frac{0.55}{B_s}\frac{1}{\sqrt{BT\gamma}}. \quad (4.44)$$

Likewise, if the signal is constant amplitude over the time interval from $-T/2$ to $+T/2$, T_e can be shown to be

$$T_e = \frac{\pi}{\sqrt{3}}T \approx 1.8T, \quad (4.45)$$

where T is the integration time. This leads to

$$\sigma_{DFO} \approx \frac{1}{1.8T}\frac{1}{\sqrt{BT\gamma}} = \frac{0.55}{T}\frac{1}{\sqrt{BT\gamma}}. \quad (4.46)$$

Stein points out that the quantity $BT\gamma$ can be viewed as the effective output SNR, with γ improved by the BT product of the processing. Because SNR is defined as E_s / N_0 and not E_c / N_0, this improvement is already taken into account and thus $BT = 1$. Also, because the SNR of the reference has no noise, γ equals twice the SNR of the received signal.

In summary, the accuracy of the estimates of TOA and FOA generally improve with increased SNR, bandwidth, and integration time. Because σ_{TOA} is dependent on the rms radian frequency, which is different but related to bandwidth, shaping a waveform may improve the accuracy of TOA estimates without requiring more signal energy or bandwidth. The next chapter applies these equations and concepts to propose waveform variations with improved geolocation performance.

THIS PAGE INTENTIONALLY LEFT BLANK

V. PROPOSED WAVEFORMS

In the previous chapter, equations (4.38) and (4.39) indicated the accuracies for the estimates of TOA and FOA are a function of the Time-Bandwidth-SNR product $TB\gamma$ along with the rms radian frequency β or rms integration time T_e for the TOA or FOA, respectively. This chapter proposes several waveforms of the same signal energy E_s but shaped in the time and frequency dimensions to improve (or degrade) these last two parameters.

The waveforms presented are direct sequence spread spectrum (DSSS) to reduce the power spectral density for reduced observability, to provide interference rejection, and to increase the bandwidth to improve the TOA estimation. DSSS is typically a BPSK-modulated chip sequence and is the basis for the reference waveform. Variations of this signal are proposed giving three classes of waveforms: BPSK-modulated waveforms, filtered BPSK-based waveforms, and spectrally constrained waveforms based on sinc-shaped chips. The performance of the various waveforms is to be compared against the reference BPSK waveform of constant amplitude and duration, waveform #1, unless otherwise indicated. The amplitude of the various waveforms are normalized so the total energy of a signal is the same as the reference. Bandwidth is referenced to the null-to-null bandwidth of the signal B_{nn}.

Table 3 summarizes the key parameters of the waveforms proposed in this chapter. The 'WF#' column lists the designator for the waveform and the next column lists the respective name. The first four waveforms are the BPSK-modulated waveforms, and the following set use the respective designator followed by an 'F' to designate filtered version of the waveform. An additional letter such as 'a', 'b', or 'c' may also be appended to designate variations that use different chipping rates. The next two columns present the rms radian frequency β and the rms duration T_e. These values were determined from waveforms generated using $N = 30720$ samples (S), $f_s = 100$ kS/s, carrier

frequency $f_c = 25$ kHz, and the chip rate R_c specified in the last column of the table. The fifth column presents the approximate null-to-null bandwidth B_{nn} of the signal. The asterisk associated with the first four is a reminder that the bandwidth of these signals is really infinite. Finally, the sixth column identifies whether the signal can be generated using the *gen_sig* MATLAB code which can simulate the effects of a dynamic collection geometry. All of the waveforms produced have the same duration, 0.31 s, and energy. Except for the first four waveforms, the energy is mainly constrained to B_{nn} listed in the 5th column of Table 3. The simulations to estimate the TOA and FOA were performed under various levels of SNR.

Table 3 Waveform Summary Table.

WF#	Name	rms rad. Freq. (rad/s)	rms duration (s)	Bnn (kHz)	gen_sig	comments
1	Reference	23622	0.5572	8*	Yes	Rc=4kcps
2	Time Gap	23226	0.8473	8*	Yes	Rc=4kcps
3	Split Spectrum	26310	0.5572	8*	Yes	Rc=4kcps
4	Shortened Pulse	24101	0.1393	8*	Yes	Rc=4kcps
1F	Filtered Reference	8506	0.5577	8	Yes	Rc=4kcps
1Fa	Filtered Reference	2031	0.5606	1	Yes	Rc=1kcps
1Fb	Filtered Reference	4261	0.5581	2	Yes	Rc=2kcps
1Fc	Filtered Reference	17039	0.5576	16	Yes	Rc=8kcps
2F	Filtered Time Gap	8434	0.8482	8	Yes	Rc=4kcps
3F	Filtered Split Spectrum	14463	0.558	8	Yes	Rc=4kcps
4F	Filtered Shortened Pulse	8655	0.1381	8	Yes	Rc=4kcps
11	Sinc WB	45569	0.5567	25	No	Rc=25kcps
12	Sinc MB	22616	0.5571	13	No	Rc=12.5kcps
13	Sinc NB	11221	0.5581	6	No	Rc=6.25kcps
14	Sinc VNB	5578	0.555	3	No	Rc=3.1kcps
15	Sinc UNB	2753	0.5511	2	No	Rc=1.5kcps
16	Sinc ENB	1460	0.5526	1	No	Rc=0.75kcps
17	Sinc - 8.3kcps	14968	0.558	8	No	Rc=8.3kcps

Figure 17 shows all the proposed waveforms and how each of the different waveforms compare with each other regarding the two main parameters affecting geolocation accuracy. The circle is at the location determined by these values and the waveform designator is placed beside the respective circle. Improved geolocation accuracy is supported for waveforms in the upper right corner of the plot and reduced performance in the lower left. More specifically, increased values of β lead to improved estimates of TOA and increased values of T_e lead to improved estimates of FOA.

Figure 18 shows a subset of the waveforms that have their energy constrained to $B_{nn} = 8$ kHz. These were selected to better illustrate how waveform shaping can affect the key parameters under the constraint of signal power, transmission duration, and occupied bandwidth. Based on this figure, an ideal waveform (from a geolocation accuracy viewpoint) would have features of filtered waveforms #2 and either #3 or #17.

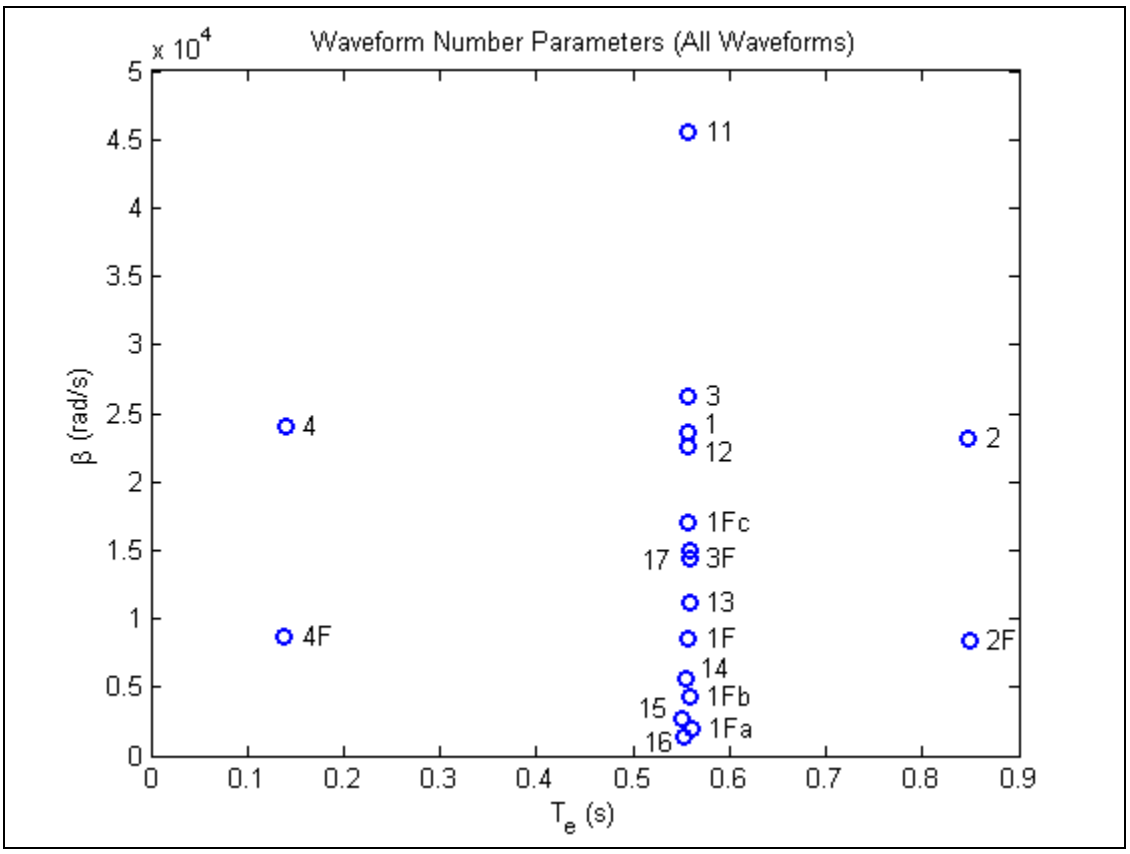

Figure 17 Scatter Plot of Waveform Parameters for All Waveforms.

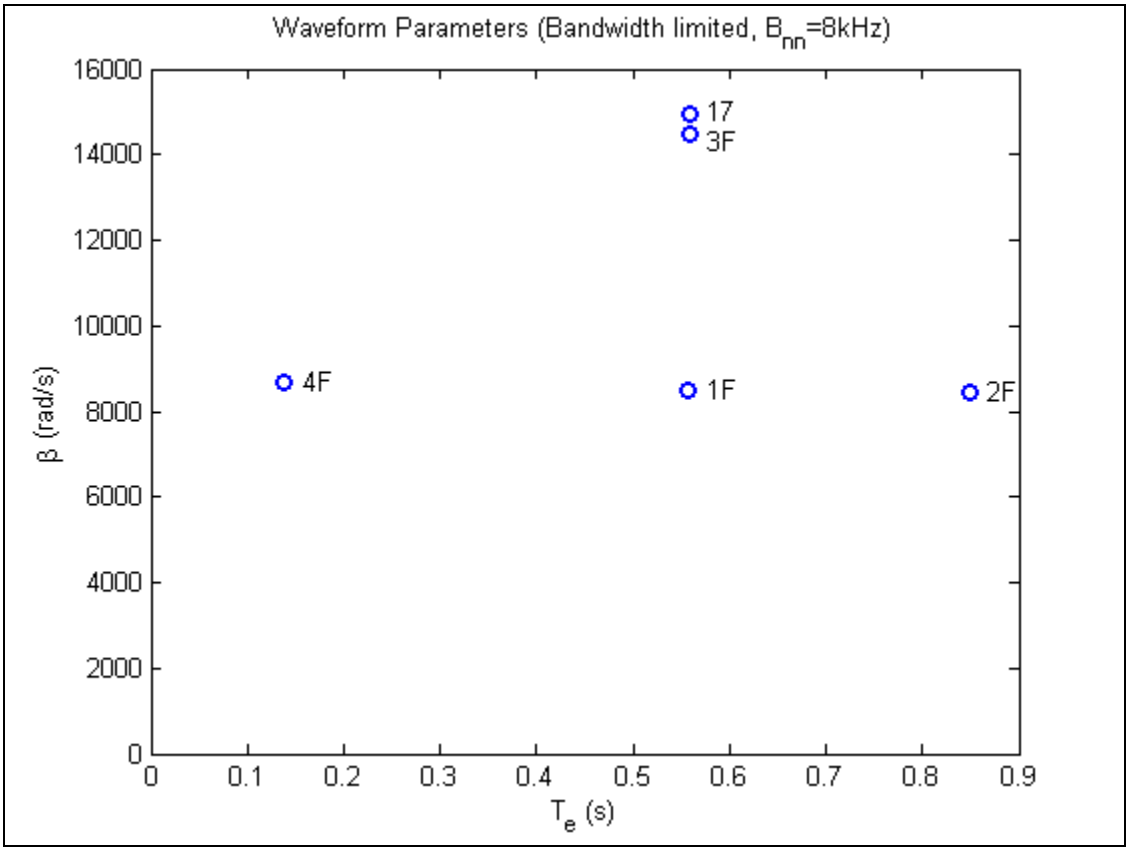

Figure 18 Scatter Plot of Parameters for Waveforms with $B_{nn} = 8$ kHz.

The remainder of this chapter presents details on the waveforms to be processed using the simulations described in the next chapter. Chapter VII presents the results of these simulations.

A. BPSK WAVEFORMS

The first four waveforms (waveforms #1-4) are BPSK modulated and are of the same duration (from beginning to end of the waveform). They consist of

- a constant amplitude waveform (#1),

- amplitude modulated versions of the baseline that disable transmission either during the middle of a pulse (waveform #2) or at the beginning and end of the pulse (waveform #4)[2], and
- a waveform made up of two narrower band BPSK signals spaced in frequency so the composite waveform has the same null-to-null bandwidth as waveform #1.

Waveforms #1 through #4 have the same energy E_s, null-to-null bandwidth B_{nn}, and time duration T. The difference between the waveforms is that they are shaped to improve (or degrade) the rms integration time and/or the rms radian frequency.

The BPSK waveforms are produced by MATLAB code based on sig_gen.m developed by Johnson [5]. The main feature of this code is that it generates a BPSK signal from a randomly generated bit sequence and projects it out to two collectors based on a defined geometry (emitter and collector locations) and velocities, thus properly modeling Doppler effects. The BPSK modulator parameters include:

- carrier frequency f_0,
- sampling frequency f_s,
- total number of samples N, and
- symbol rate R_s, i.e., bit rate for BPSK, which is really the chip rate in this application.

The BPSK waveforms used the following parameters: $N = 30720$ Samples (S), $f_s = 100$ kS/s, and R_s was nominally 4 kchips/s but was varied for some runs. In addition, $f_0 = 20$ kHz was used for the simulations but was set to 25 kHz for the plotting of the waveforms and calculations of T_e and to better compare with the final set of waveforms in which $f_0 = f_s/4$. The duration of the waveform can be found using these values to be

[2] Although this latter waveform could be considered a pulse of different duration from the others, it can also be considered one of the same duration in which the amplitude is zero at the beginning and ends.

$$T = \frac{N}{f_s} = \frac{30720 \text{ S}}{100 \text{ kS/s}} = 0.3072 \text{ s} \tag{5.1}$$

The number of chips transmitted during this period is

$$no\_chips = R_s T = \frac{4 \text{ kchips}}{\text{s}} 0.3072 \text{ s} = 1228 \text{ chips} \tag{5.2}$$

The plots shown for the different waveforms are from the analytic signal as implemented in the simulations. The analytic signal is generated by taking the Hilbert transform of the real signal [5], which results in a complex waveform with no negative frequencies. This feature is needed by the CAF process but is also is useful for presenting the single-sided power spectral density (PSD).

Measurements of the rms bandwidth of a representative signal corresponding to each of the waveforms shows β to be on the order of 25,000 radians/second for all four waveforms. The waveform variations do, however, affect the rms duration T_e which ranges in value from 0.14 to 0.85 seconds.

1. Waveform #1 – "Reference Waveform"

Waveform #1, the reference waveform, is a BPSK modulated DSSS signal of unity amplitude. Figure 19 plots the instantaneous power of the signal as a function time over the entire waveform in the upper plot and for a shorter time segment in the lower plot. Recognizing that the signal is complex, the signal power is the square of the signal magnitude

$$P_s = |s|^2 . \tag{5.3}$$

Except for the small glitches, which can be better seen in the lower plot, the signal power has unity magnitude. These glitches occur at the chip transitions and are caused by the limited bandwidth inherent in the digital signal. This bandwidth limitation removes the higher order frequencies that make up the rectangular modulation pulses [19].

The rms integration time T_e can be calculated for this constant amplitude signal using (4.45) for the signal of duration $T = 0.3072$ s, from equation (5.1), giving

$$T_e = 1.8T = 0.55s\,. \tag{5.4}$$

This compares favorably with the value displayed at the top of Figure 19, "$T_e = 0.5572s$", which was calculated from the digitized waveform using equation (4.42) by summing the waveform power weighted by time from the central time, and dividing this by the sum of the unweighted waveform power, or signal energy E_s.

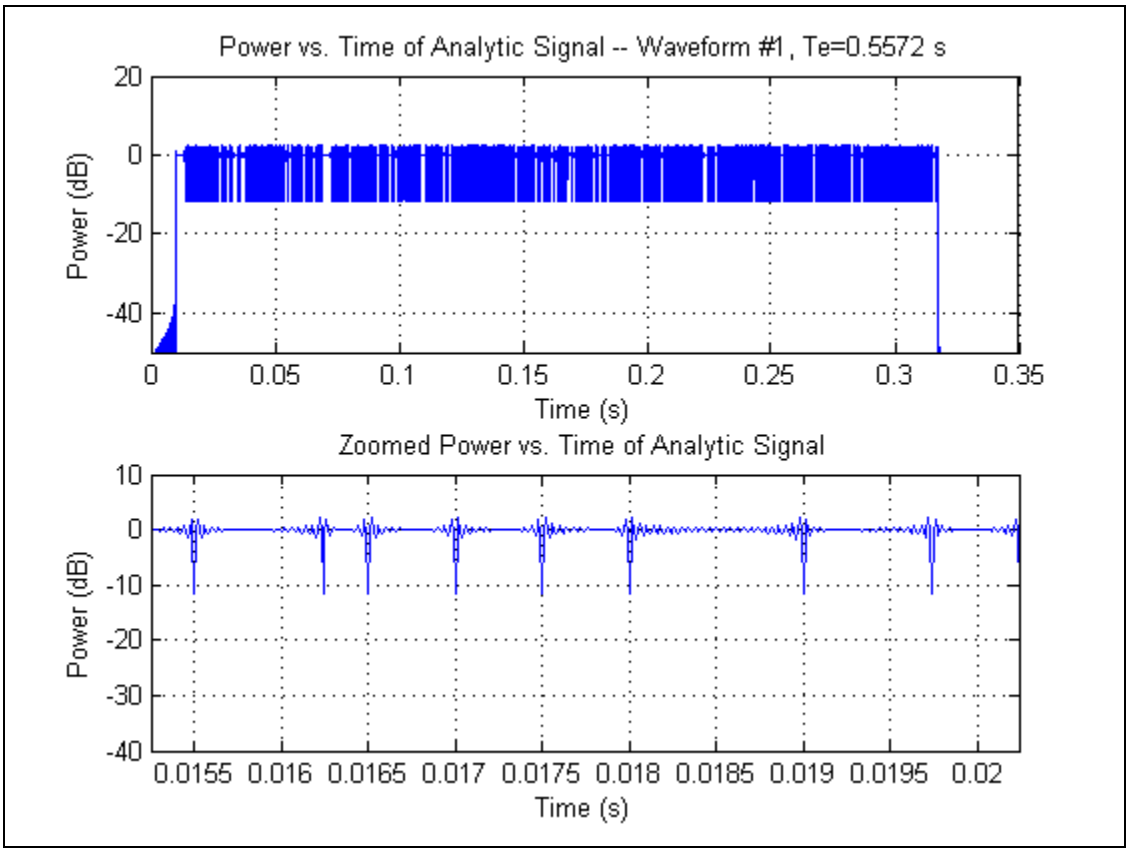

Figure 19 Waveform #1 – Power vs. Time.

In the frequency domain, modulation is equivalent to the Fourier transform of the modulating signal shifted by the frequency of the carrier which for rectangular pulses is represented as

$$\text{rect}\left(\frac{t}{T}\right)\cos\left(2\pi f_c t\right) \quad \frac{T}{2}\left\{\text{sinc}\left[T\left(f-f_c\right)\right]+\text{sinc}\left[T\left(f+f_c\right)\right]\right\} \tag{5.5}$$

where $\text{rect}\left(\frac{t}{T}\right)$ is a pulse of unit amplitude and width T centered about $t=0$, $\cos\left(2\pi f_c t\right)$ is the carrier with center frequency f_c, and t and f are time and frequency, respectively [19]. Because instantaneous power is the square of the signal, the PSD takes on a shape of the form $\text{sinc}^2\left(f-f_c\right)$ which can be seen in Figure 20, which is a plot of the PSD generated by squaring the magnitude of the signal's fast Fourier transform (FFT) and normalizing by the number of samples and sampling frequency [27].

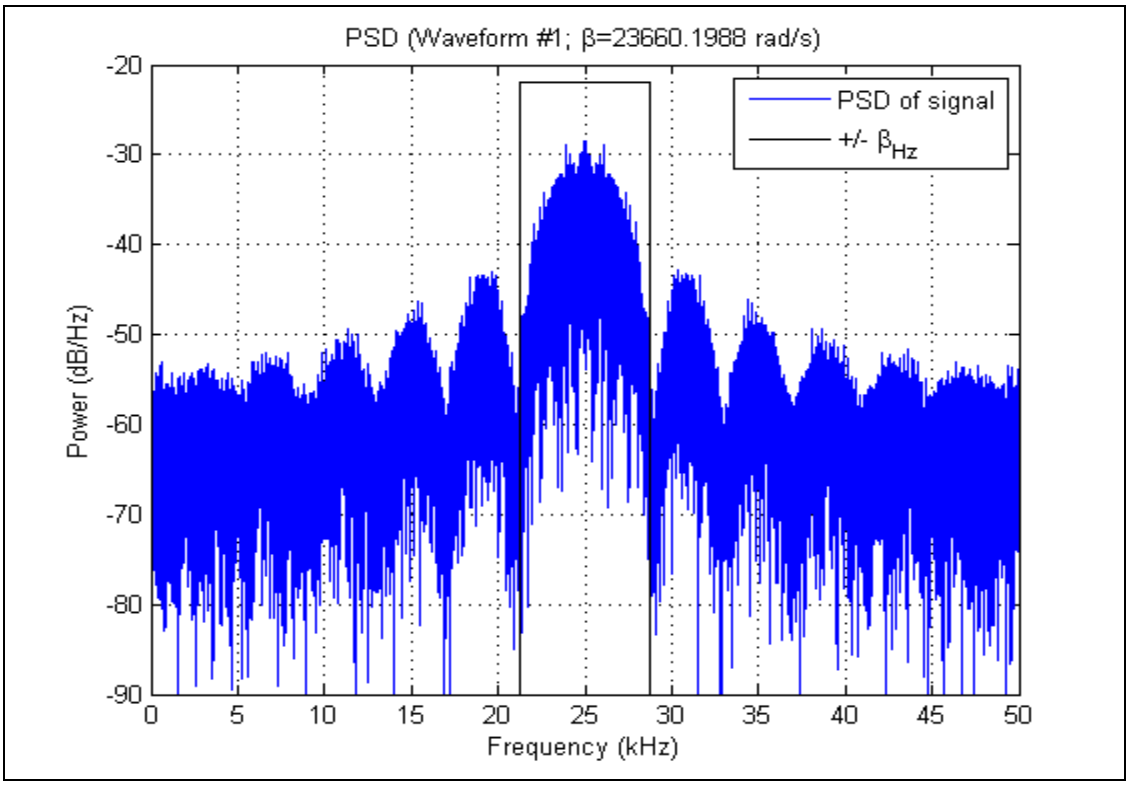

Figure 20 Waveform #1 – Power Spectral Density.

Note that the signal shows some distortion from a sinc-type function which has infinite bandwidth. Because this was generated digitally, those frequency components greater than $f_s/2$ form aliases which are mapped back into this

range [26]. This aliasing is evident in the distortion of the lobes at the edges of the spectrum as the higher frequency lobes fold back upon the lower frequency components filling in some of the nulls.

This PSD, which is basically of the form

$$W_s(f) = \mathrm{sinc}^2(f) = \left(\frac{\sin(\pi f)}{\pi f}\right)^2, \tag{5.6}$$

when inserted into (4.41) gives it the form

$$\begin{aligned}
\beta &= 2\pi\left[\frac{\int_{-\infty}^{\infty} f^2 W_s(f)\,df}{\int_{-\infty}^{\infty} W_s(f)\,df}\right]^{1/2} \\
&= 2\pi\left[\int_{-\infty}^{\infty} f^2 \frac{\sin^2(\pi f)}{(\pi f)^2 k^2}\,df\right]^{1/2} \\
&= 2\pi\left[\int_{-\infty}^{\infty} \frac{\sin^2(\pi f)}{\pi^2 k^2}\,df\right]^{1/2} \\
&= \frac{2}{k}\left[\int_{-\infty}^{\infty} \sin^2(\pi f)\,df\right]^{1/2}
\end{aligned} \tag{5.7}$$

where $k^2 = \int_{-\infty}^{\infty} W_s(f)\,df$ is a normalizing factor. Using the equality $\int \sin^2 u\,du = \frac{1}{2}u - \frac{1}{4}\sin 2u + C$ [28], results in

$$\beta = \frac{2}{k\pi}\left\{\left[\frac{1}{2}u - \frac{1}{4}\sin 2u\right]_{-\infty}^{\infty}\right\}^{1/2} = \infty, \text{ where } k \neq 0. \tag{5.8}$$

Thus, if one had an infinite bandwidth collector, any pure BPSK modulated signal, exhibiting a sinc-squared power spectral density, would yield no TOA error.

In the real world, however, a collector has limited bandwidth, and thus a non-zero value for β exists. A wider bandwidth collector will cause β to be larger resulting in better TOA accuracy using (4.38).

Finally, one last feature to notice in Figure 20 is the rectangular box showing $+/-\beta_{Hz}$, which is the rms radian frequency β converted from radians to

Hz. β was numerically calculated from the generated waveform and displayed in the title. The amplitude of this rectangle has no meaning and is included only to allow the bandwidth to be better visualized. For this waveform, β_{Hz} coincidently falls at approximately the frequency of the first null.

Another feature of a waveform is its autocorrelation, which indicates significance in both the shape of the peak and in the height of minor correlations relative to the peak. Figure 21 shows the autocorrelation R of waveform #1, with sufficient lags to cover the entire waveform in the left plot and with fewer lags in the right plot to see the shape of the correlation near the peak. The MATLAB *xcorr* function, which was used to compute these values, by default computes raw correlations with no normalization using

$$\hat{R}_{xy}(m) = \begin{cases} \sum_{n=0}^{N-m-1} x_{n+m} y_n^* & m \geq 0 \\ \hat{R}_{yx}^*(-m) & m < 0 \end{cases} \tag{5.9}$$

but the 'coeff' option was applied to normalize such that the autocorrelations at zero lag are "identically 1.0" [29]. Because the correlation R is a power, the value was converted to decibel scale using

$$R_{dB} = 10 \log_{10} R \,. \tag{5.10}$$

If the signal were truly noise-like (AWGN), the its autocorrelation would approach that of white noise $N(t)$ which is zero everywhere except at lag equal zero

$$R_{NN}(\tau) = (N_0/2)\delta(\tau) \quad \text{[23].} \tag{5.11}$$

Values of τ where $R_{NN}(\tau)$ is not equal to zero represent hidden periodicities in the signal. DSSS signals can be made noise-like by using m-sequences of length l in which the correlation is constant near zero except at lag zero were the correlation value is l or by using other codes which, while not as good, approximate the noise-like property of equation (5.11) [21].

As can be seen in Figure 21, minor correlation peaks are only 12-13 dB below the peak correlation because the signal did not transmit the full length of the reference m-sequence[3] [30]. These artifacts create a type of noise floor that reduces the margin of discrimination. The correlation performance of this waveform should be able to be improved significantly. This waveform used the first 1228 chips (per equation (5.2)) from a 65,535 bit m-sequence. Matching the number of bits transmitted to the number of bits in an m-sequence [30] should give optimal performance [21]. A 1023 –bit m-sequence should have better than 30dB between the peak and the floor with no minor correlation peaks.

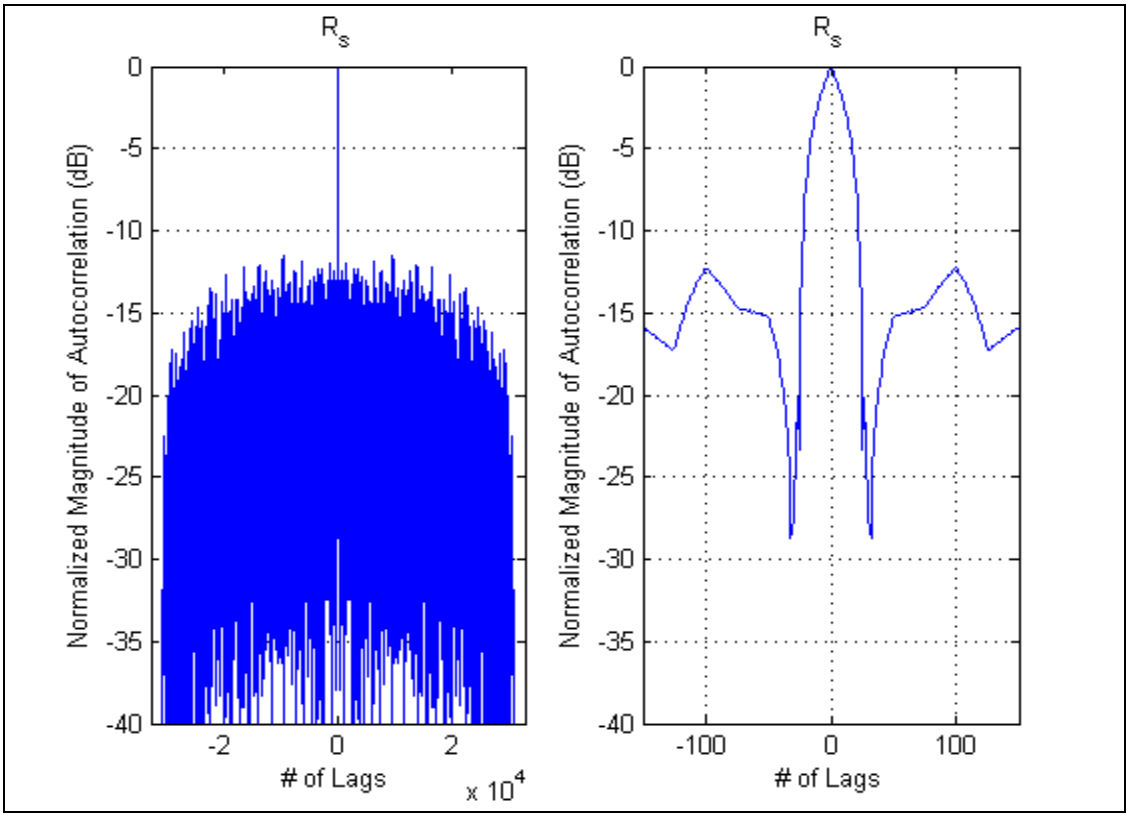

Figure 21 Autocorrelation of Waveform #1.

[3] No attempt was made to match the length of the m-sequence to the no. of chips transmitted.

2. Waveform #2 – "Time Gap"

The second waveform is designed to improve the rms integration time T_e of waveform #1. Equation (4.42) shows that shaping the pulse by moving the power from the middle of the waveform to its beginning and end should increase T_e. Waveform # 2 does this by inhibiting transmission of the signal during the central ¾ of the waveform and transmitting this power during the remaining ¼ of the time, as shown in the upper plot of Figure 22. The lower plot in this figure shows that the signal is still constant power (while transmitting) but is 6 dB higher (i.e., 4 times stronger) to have the same signal energy as waveform #1. The rms integration time T_e is almost 0.85, as seen in the title of the upper plot, an increase of almost 50% over the reference waveform without an increase in actual transmit time which has not changed.

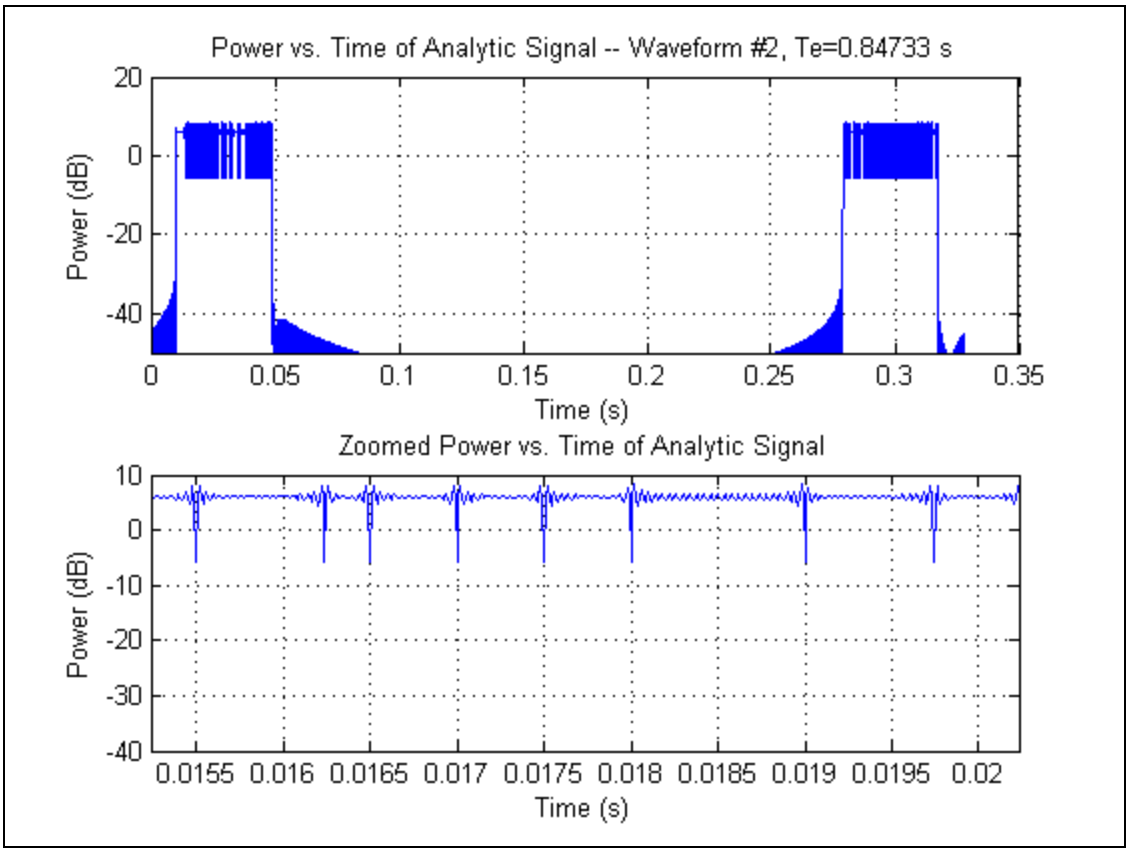

Figure 22 Waveform #2 – Power vs. Time.

Because the chip rate is identical, no significant difference should be expected in bandwidth. Indeed, Figure 25 shows the resulting PSD and rms bandwidths are basically the same as seen for waveform #1.

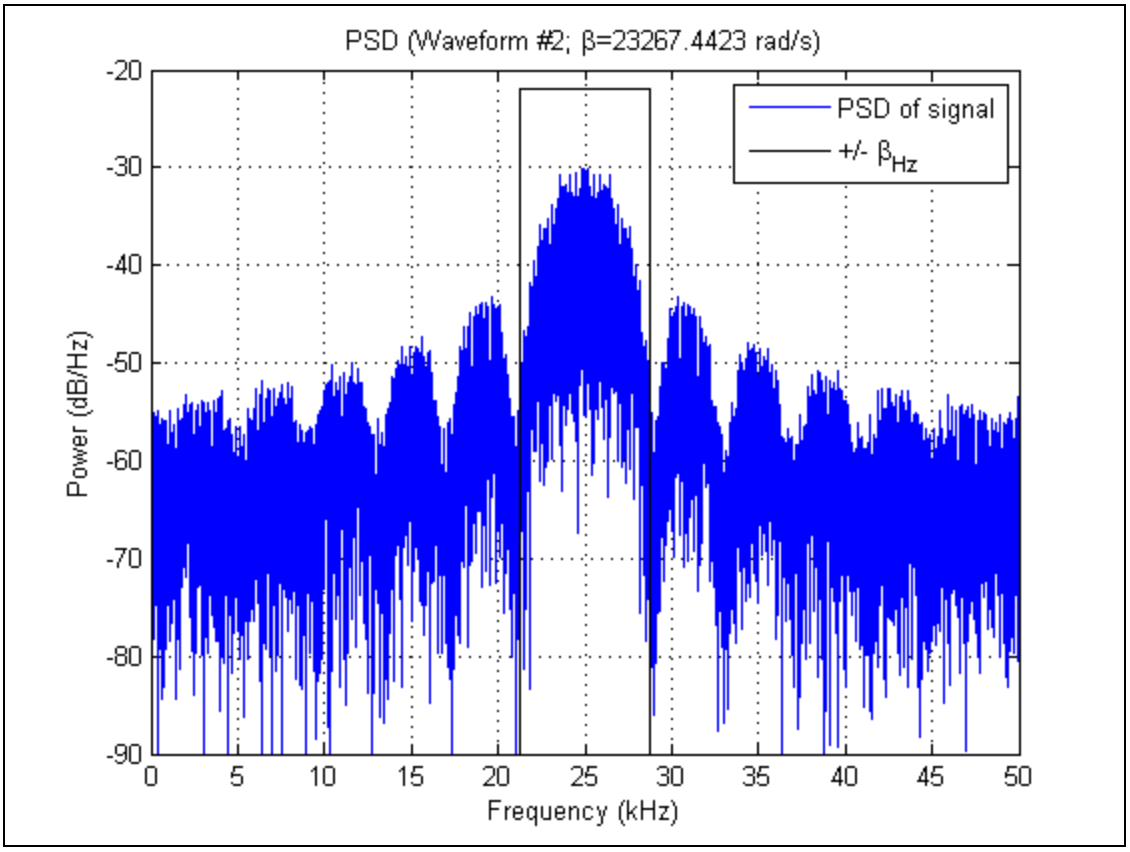

Figure 23 Waveform #2 – Power Spectral Density.

The width of the peak autocorrelation for waveform #2 (right plot of Figure 24) is similar to that for waveform #1, but the minor correlation peaks are now within 10 dB of the peak, consistent with the fact that fewer chips are being transmitted. Note that the correlation peaks near the edges of the waveform (i.e., at larger lags) are approximately four to 5 dB higher that seen in Figure 21 at similar lags.

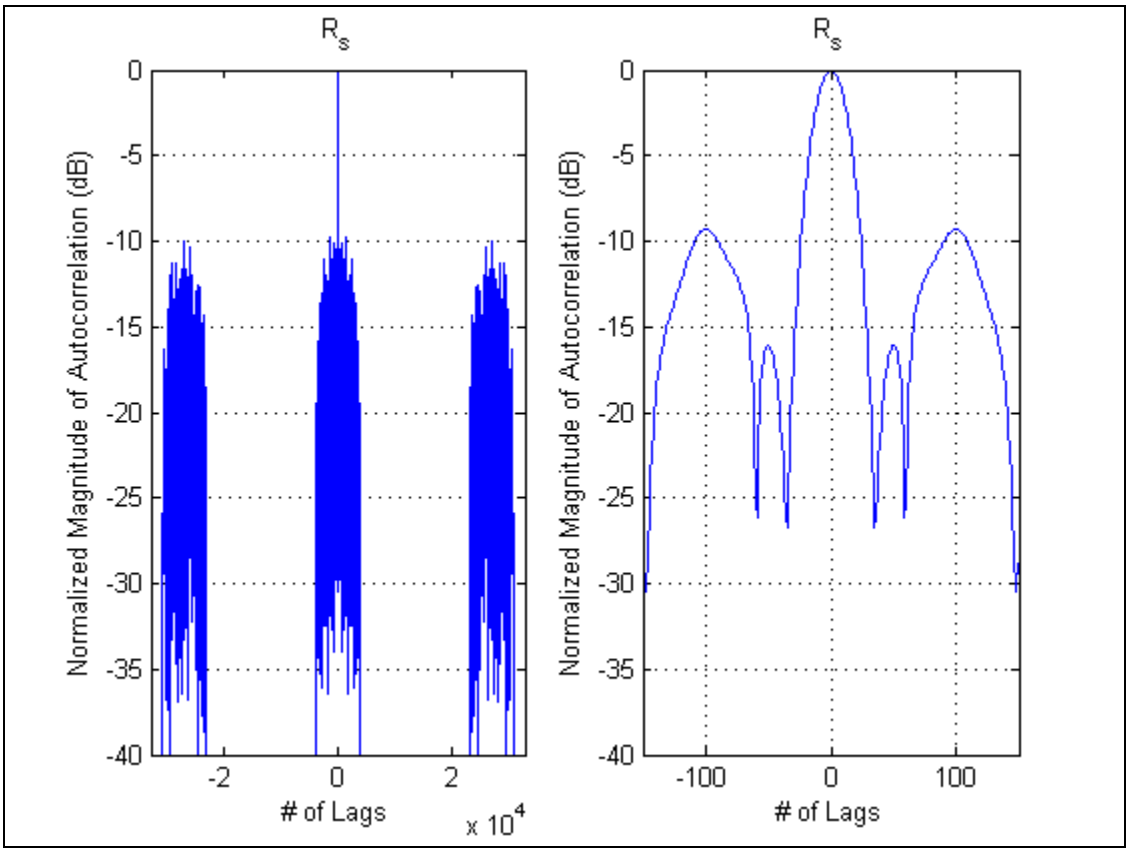

Figure 24 Waveform #2 – Autocorrelation.

3. Waveform #3 – "Split Spectrum"

Just as waveform #2 increased T_e without actually increasing the total transmit duration, waveform #3 attempts to increase the rms radian frequency β without actually increasing the null-to-null bandwidth B_{nn}. Examining (4.41), one notices that moving the energy from the middle of the spectrum to the outer edges (but still within B_{nn}) should increase β without consuming additional bandwidth.

Waveform #3 is created from the addition of two BPSK waveforms, each chipped at half the specified chip rate and offset from the nominal center frequency by half the chip rate, as seen in PSD (Figure 25), and it is described by

$$s(t) = k\left\{c_{i,1}\cos\left[2\pi\left(f_c + \frac{R_c}{2}\right)t\right] + c_{i,2}\cos\left[2\pi\left(f_c - \frac{R_c}{2}\right)t\right]\right\}, \qquad (5.12)$$

where $c_{i,1}$ and $c_{i,2}$ are the is the chip sequence modulated onto each of the two offset subcarriers, f_c is the carrier frequency, R_c is the chip rate, and k is a normalizing factor to set signal power.

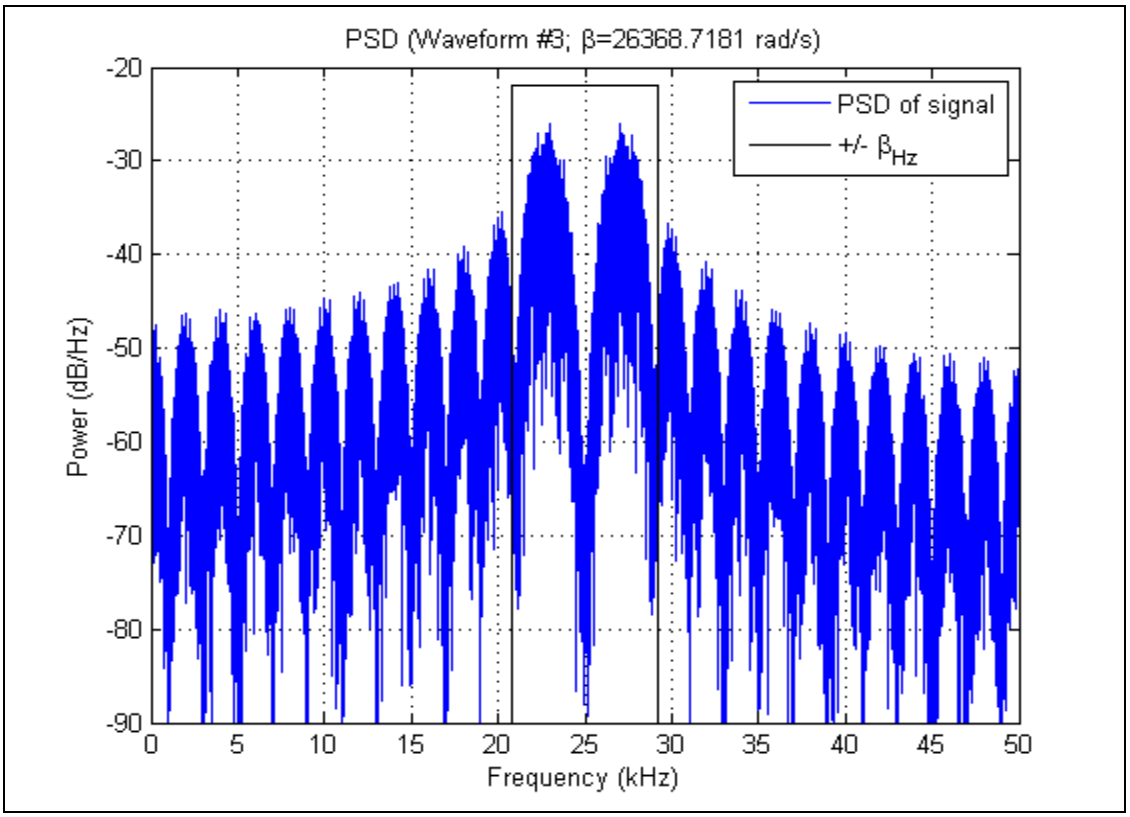

Figure 25 Waveform #3 – Power Spectral Density.

Using the trigonometric identity

$$\cos x \cos y = \frac{1}{2}\left[\cos(x+y) + \cos(x-y)\right] \quad [31] \qquad (5.13)$$

and letting $c_i = c_{i,1} = c_{i,2}$, equation (5.12) can be rewritten as

$$s(t) = 2kc_i \cos(2\pi f t_c)\cos\left(2\pi\frac{R_c}{2}t\right), \qquad (5.14)$$

which is identical to the BPSK signal modulated by a tone at half the chip rate. This modulation of the BPSK is evident in the time domain, Figure 26, where it

can be seen in the bottom plot that amplitude of the signal is no longer constant. The glitches, corresponding to the chip transitions, occur as expected at one millisecond apart, which is equal to the inverse of half the chip rate used, 2000 chips per second. Also note that the peak power is four times that of waveform #1. No attempt was made to form this waveform such that the chip transitions occur at a null, nor to assess whether doing this would reduce spectral artifacts.

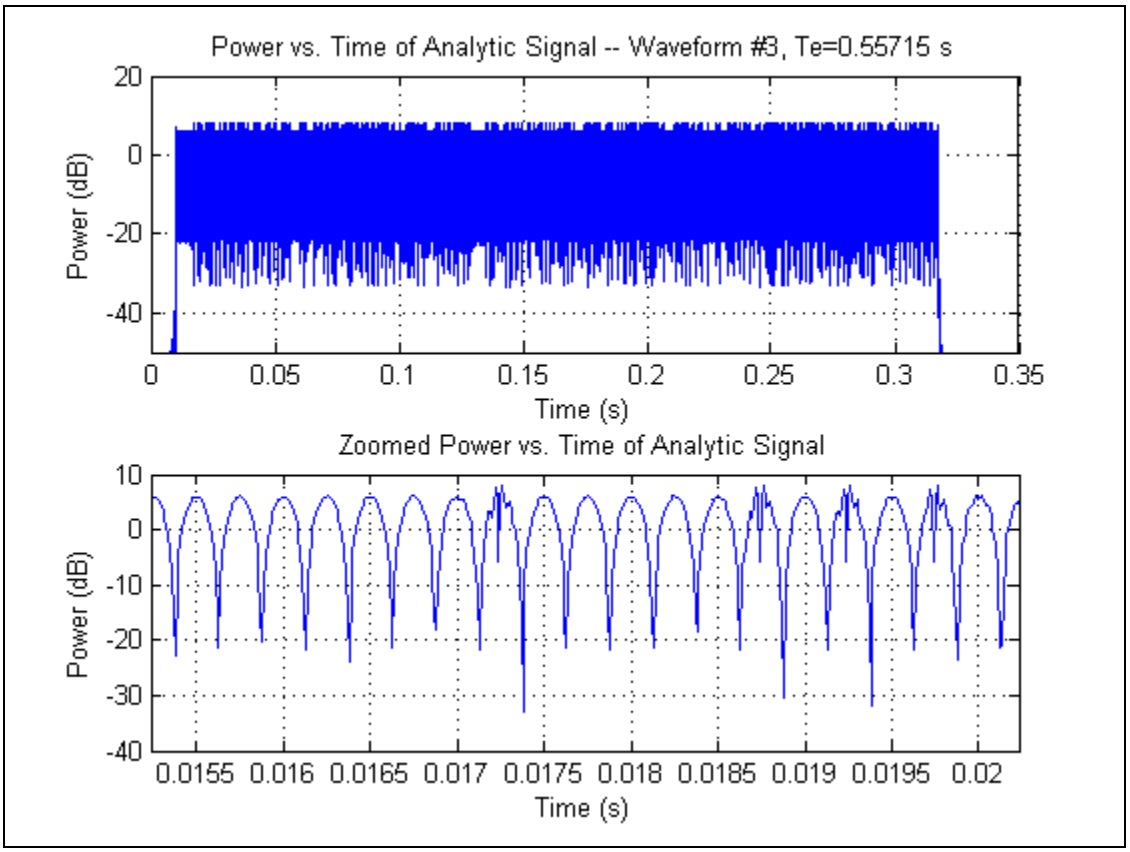

Figure 26 Waveform #3 – Power vs. Time.

The autocorrelation of waveform #3, Figure 27, shows the magnitude of the minor sidelobes are higher than waveform #1 but lower than waveform #2. This is consistent with the fact that the number of chips contained within a transmission is half that of waveform #1 and twice that of waveform #2, because both subcarriers are modulated by the same sequence. The structure around the peak correlation shows the main lobe to be quite narrow, but it has several strong

sidelobes around it. Whether this is because the two constituent signal used the same chip sequence has not been investigated.

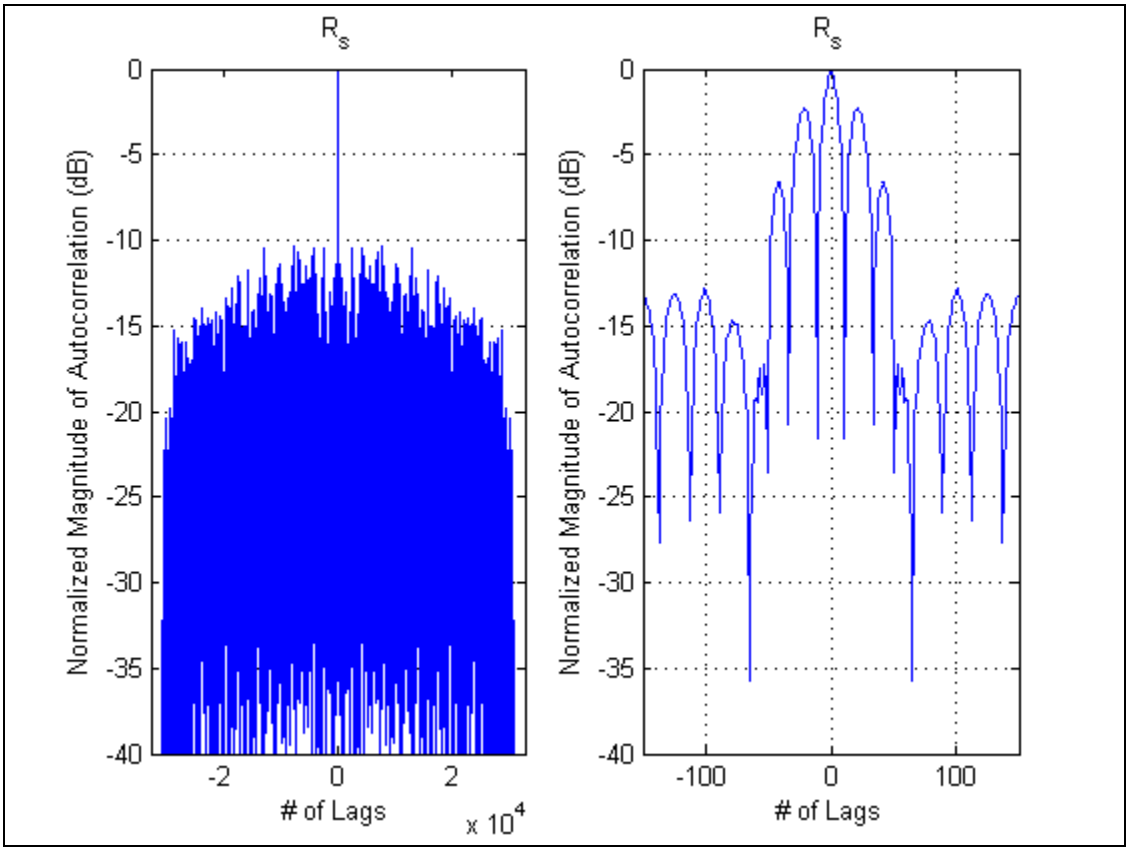

Figure 27 Waveform #3 – Autocorrelation.

4. Waveform #4 – "Shortened Pulse"

Waveform #4 is the complement to waveform #2; however, instead of pushing the signal energy from the center out toward the front and back, it brings the energy form the two ends back to the middle as can be seen in Figure 28. As in waveform #2, the duty cycle is only one quarter of waveform #1 leading to a commensurate increase in peak power to maintain constant signal energy.

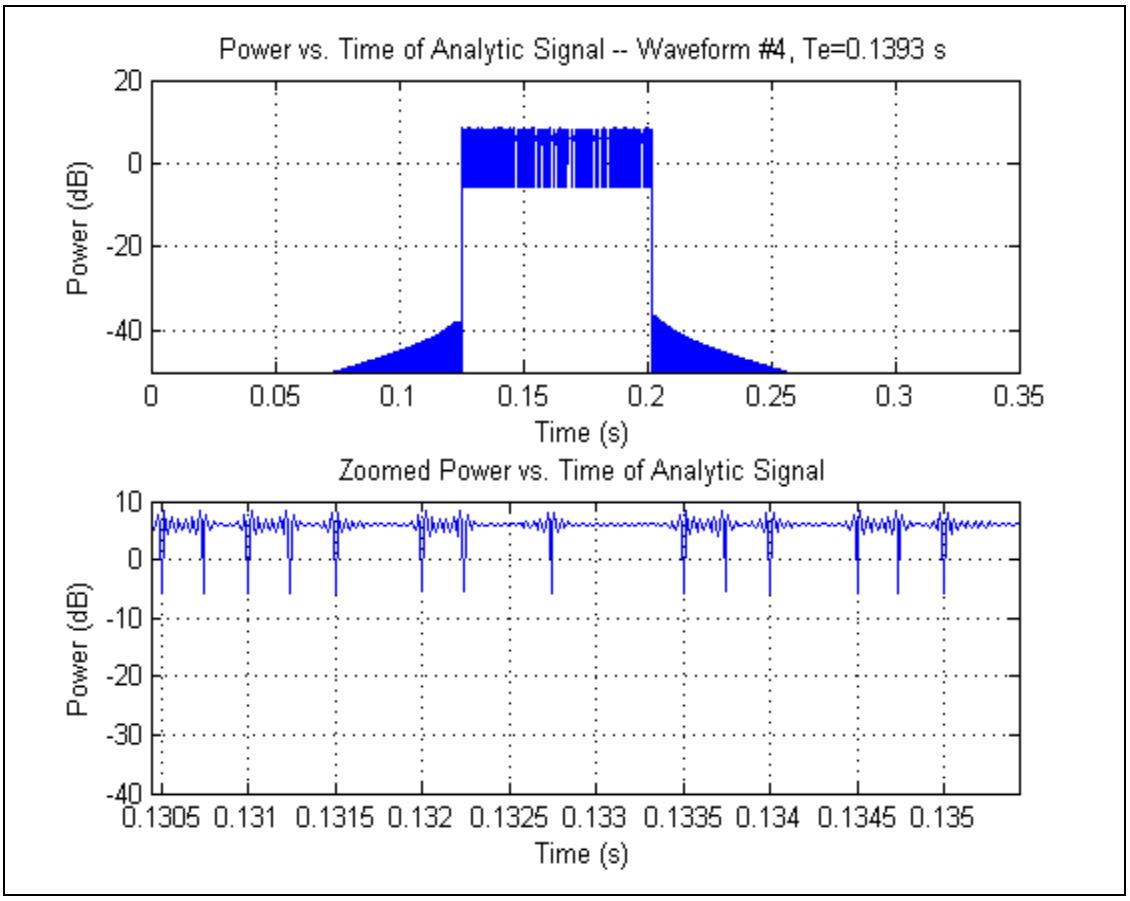

Figure 28 Waveform #4 – Power vs. Time.

The PSD of waveform #4 (Figure 29) is similar to waveform #1 because the signal has the same modulation and chip rate as waveform #1. The PSD should not be expected to be identical because the signal duration is shorter than that of waveform #2 and uses fewer chips; the corresponding short-term statistics causes minor variations between the two waveforms. Likewise, the amplitude of the autocorrelation minor peaks (Figure 30) are similar to those of waveform #2 which has similar duty cycle, although variation do exist because of differences in the chip sequence used.

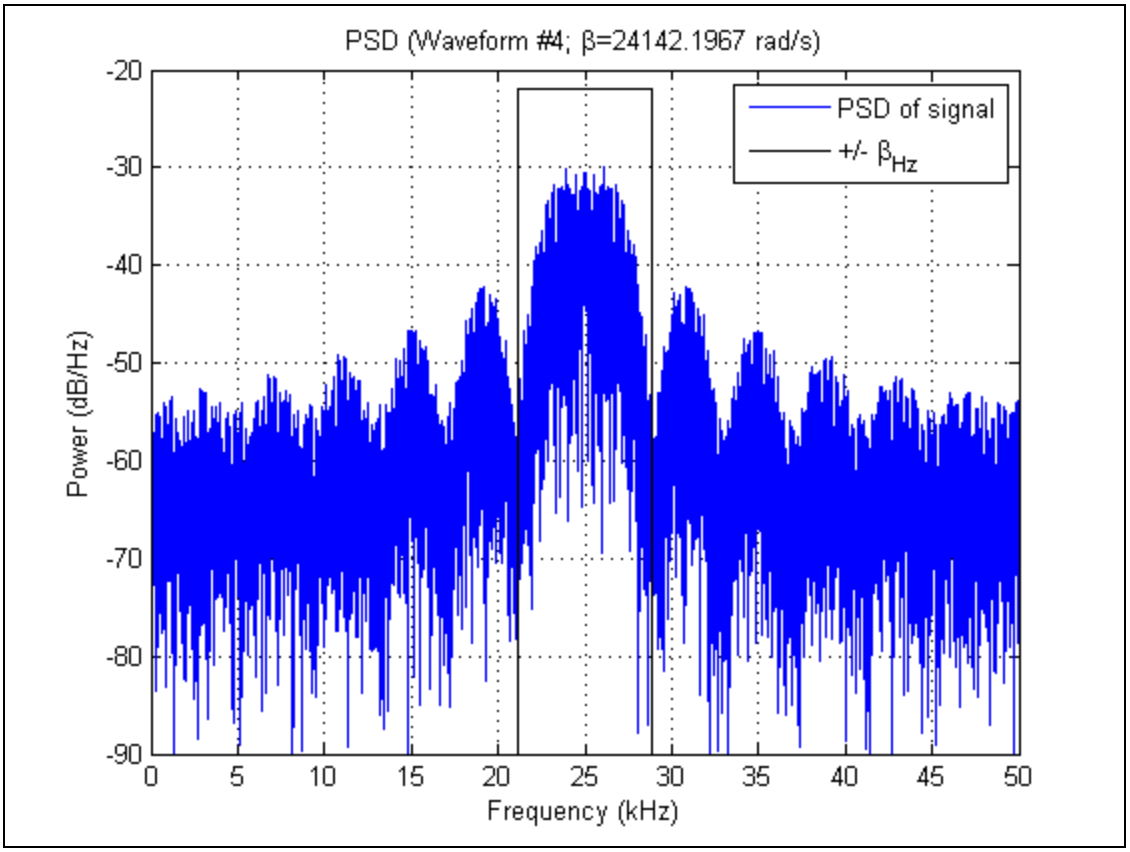

Figure 29 Waveform #4 – Power Spectral Density.

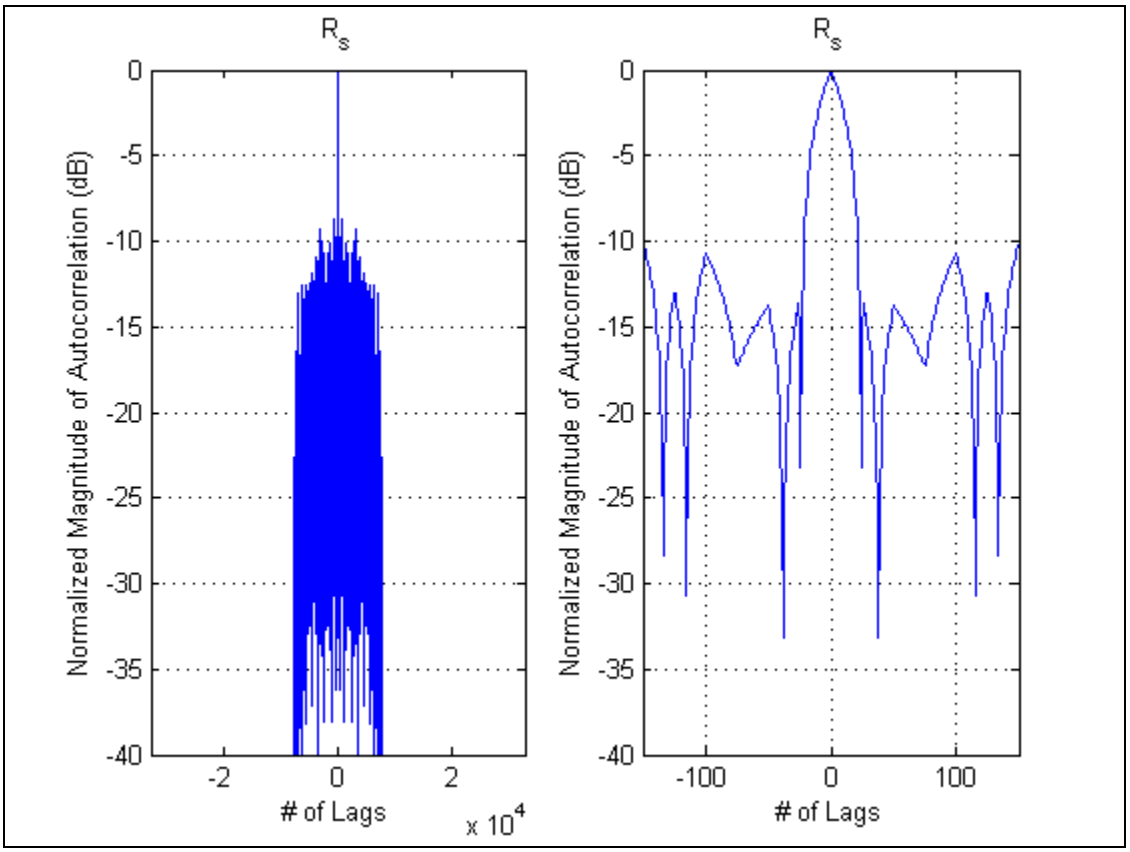

Figure 30 Waveform #4 – Autocorrelation.

B. FILTERED BPSK WAVEFORMS

The measured values for the rms radian frequency β did not vary much between the BPSK waveform types because the relatively slow roll-off of power for the signal sidelobes. Instead these values were limited by the bandwidth of the collector, which in our case was half the sampling frequency, i.e., $f_s/2$. A real-world collector does not have infinite bandwidth but is usually limited by the signal it needs to collect. This section limits the occupied bandwidth of the signal to the null-to-null bandwidth B_{nn} of the signal, because this bandwidth contains most of the signal power and is the most popular measure of bandwidth for digital communications [24]. This bandwidth is

$$\beta_{nn} = \frac{2}{T_c} = 2R_c \,. \tag{5.15}$$

A new set of waveforms, filtered waveforms #1-4, correspond to the original waveforms #1-4 which have been filtered to remove components outside the null-to-null bandwidth B_{nn}. Filtering was performed by taking the FFT of the analytic signal, setting to zero the value of all bins corresponding to being outside B_{nn}, taking the inverse FFT (IFFT) of this, extracting the real component of this signal, and scaling the signal so the total energy is the same as waveform #1.

Filtering the signals did not affect the respective rms duration T_e, which ranges in value from 0.14 to 0.85 seconds for the four waveforms. The rms radian frequency β, however, for the filtered signals range from 8500 to over 14,000 radians/second for a reference waveform at 4000 chips per second, 4 kc/s, as compared with approximately 25,000 for all the unfiltered waveforms..

1. Filtered Waveform #1 – "Reference Waveform"

Figure 31 shows the PSD of filtered waveform #1 chipped at 4 kcps. The rms radian frequency β is approximately 8500 radians per second, about one-third the value for the unfiltered waveform #1 with the sampling frequency f_s of 100 ksamples/second.

Because the higher frequency components of the signal are removed, the amplitude of the signal is no longer constant over time (Figure 32) with deeper and wider nulls at the chip transitions along with additional peak power required to compensate for this loss. The autocorrelation shown in Figure 33 is very similar to the corresponding unfiltered version shown Figure 21, except that the sharper features are rounded. Because the autocorrelation of the filtered waveform is so similar to that of the corresponding unfiltered waveform, the autocorellation plots for the remaining waveforms are not shown.

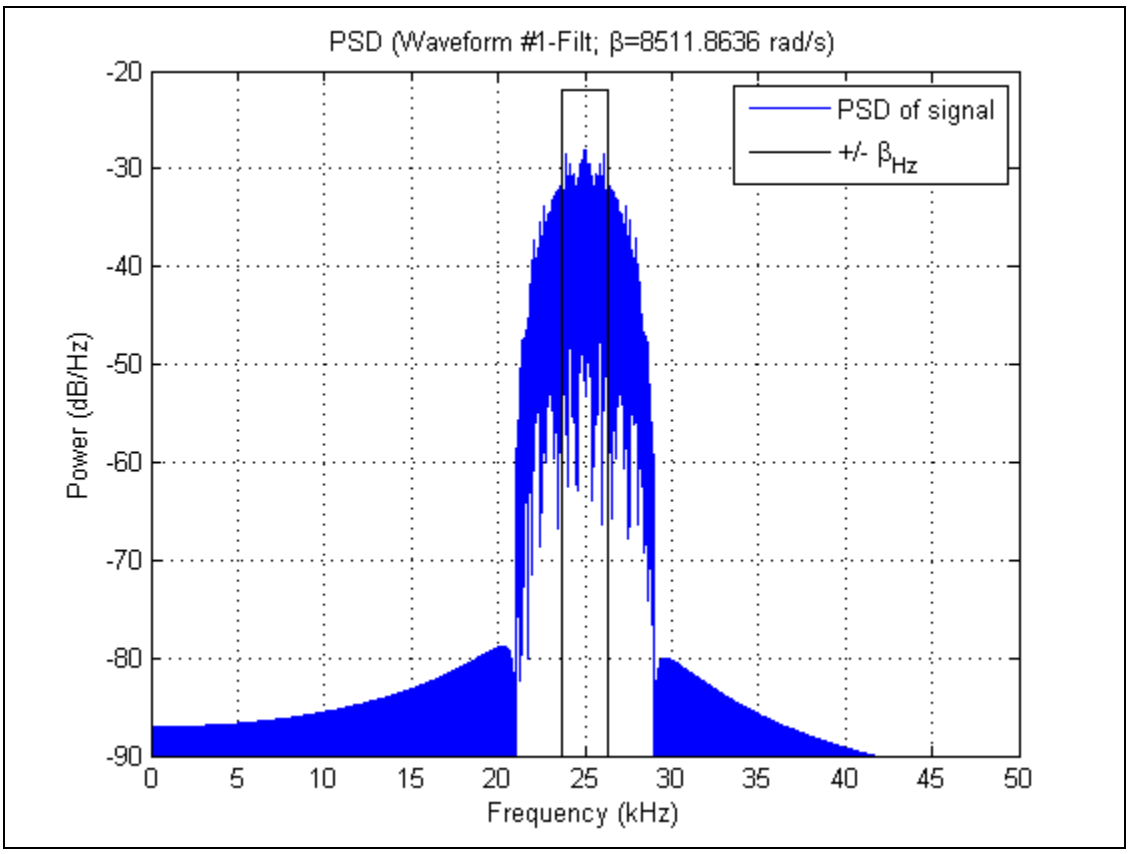

Figure 31 Filtered Waveform #1 – Power Spectral Density.

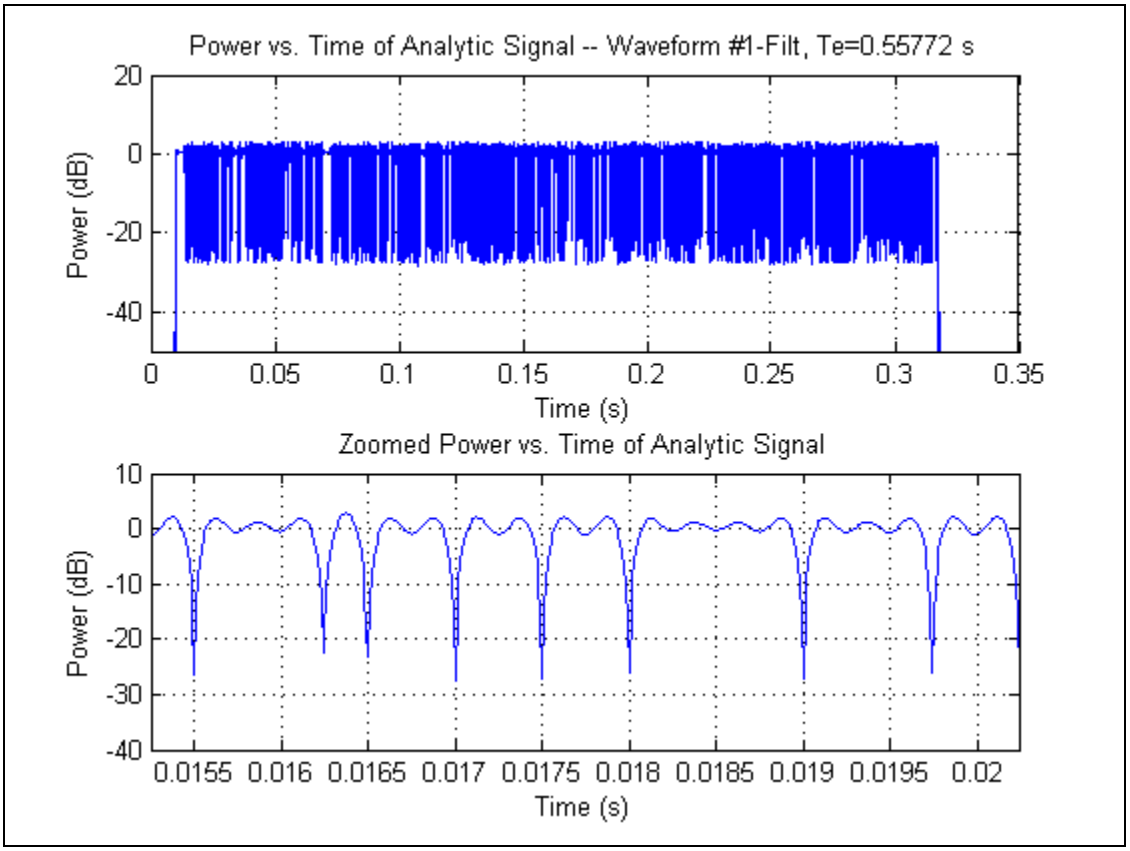

Figure 32 Filtered Waveform #1 – Power vs. Time.

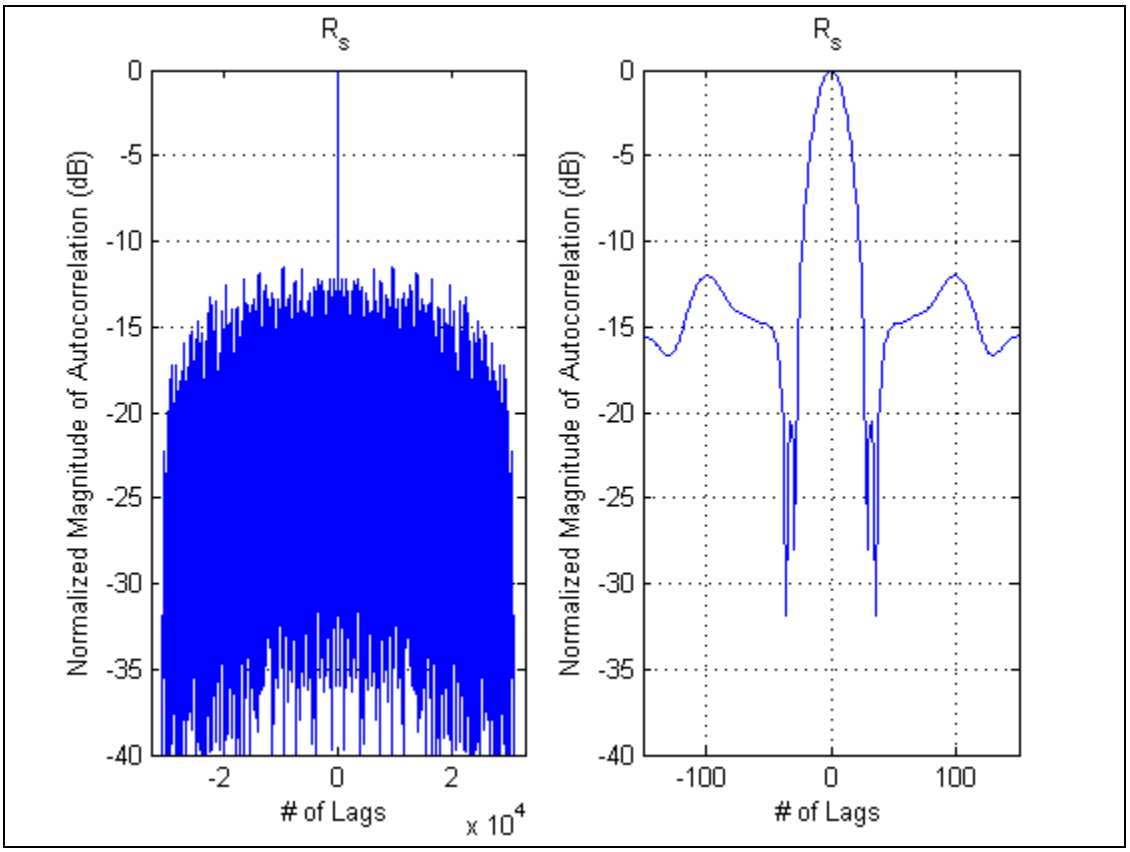

Figure 33 Filtered Waveform #1 – Autocorrelation.

2. Filtered Waveform #2 – "Time Gap"

The PSD of filtered waveform #2 as shown in Figure 34 is very similar to the filtered waveform #1 just discussed, and again β is approximately 8500 radians per second[4]. The waveform also exhibits the deeper and wider nulls at chip transitions along with the additional peak power required to compensate for this loss (Figure 35).

[4] The different values measured for β can be attributed to the fewer chips transmitted.

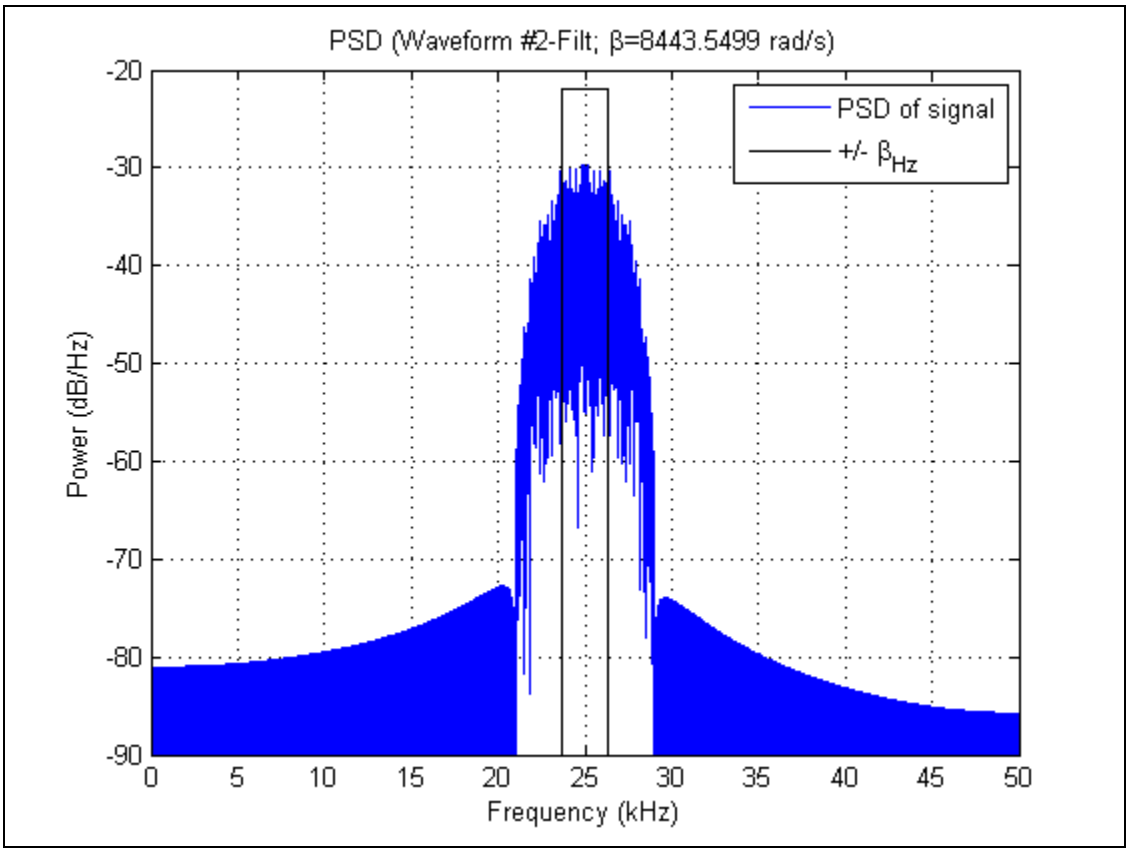

Figure 34 Filtered Waveform #2 – Power Spectral Density.

Figure 35 Filtered Waveform #2 – Power vs. Time.

3. Filtered Waveform #3 – "Split Spectrum"

The PSD of filtered waveform #3 (Figure 36) is similar to the unfiltered waveform #3 (Figure 25) but with the removal of any significant energy outside the B_{nn}. The resulting rms bandwidth ends up occurring at the subcarrier frequencies, as can be expected, because half the energy occurs within this frequency range and half outside as can be readily observed in Figure 36. The rms radian frequency β is approaching 15,000 radians per second, almost 70% higher than waveform #1 without consuming more bandwidth.

The removal of this out of band energy, however, affects the signal in the time domain. The peak power for each of the peaks varies (Figure 37), and the peak amplitude of the signal is higher to compensate for this variability, sometimes approaching close to 10 dB above that required for waveform #1. A

pair of shorter pulses coincides with each chip transitions, which occur at the peak of the signal. Timing the chip transition to occur at the null of the signal such as by modifying (5.14) to instead be

$$s(t) = 2kc_i \cos(2\pi f t_c) \sin\left(2\pi \frac{R_c}{2} t\right) \tag{5.16}$$

might restore the pulses to the same amplitude and duration, regardless of whether a chip transition occurs.

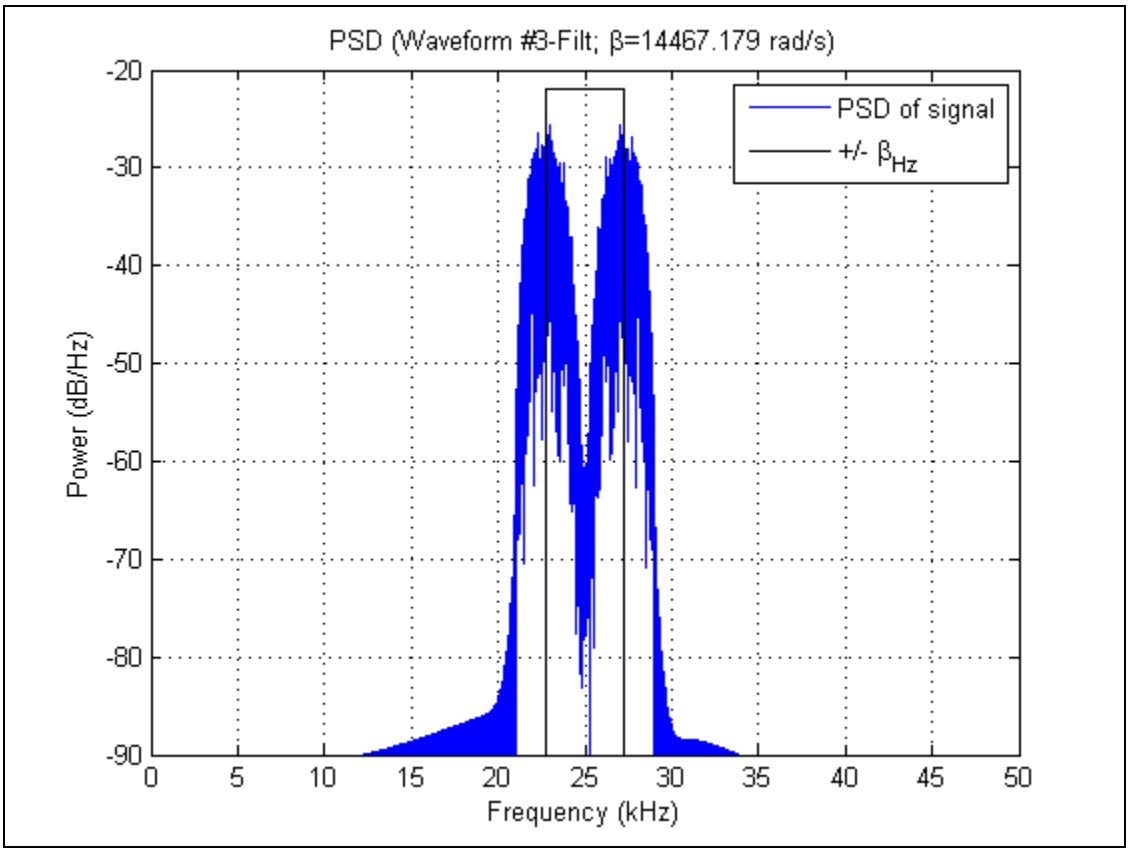

Figure 36 Filtered Waveform #3 – Power Spectral Density.

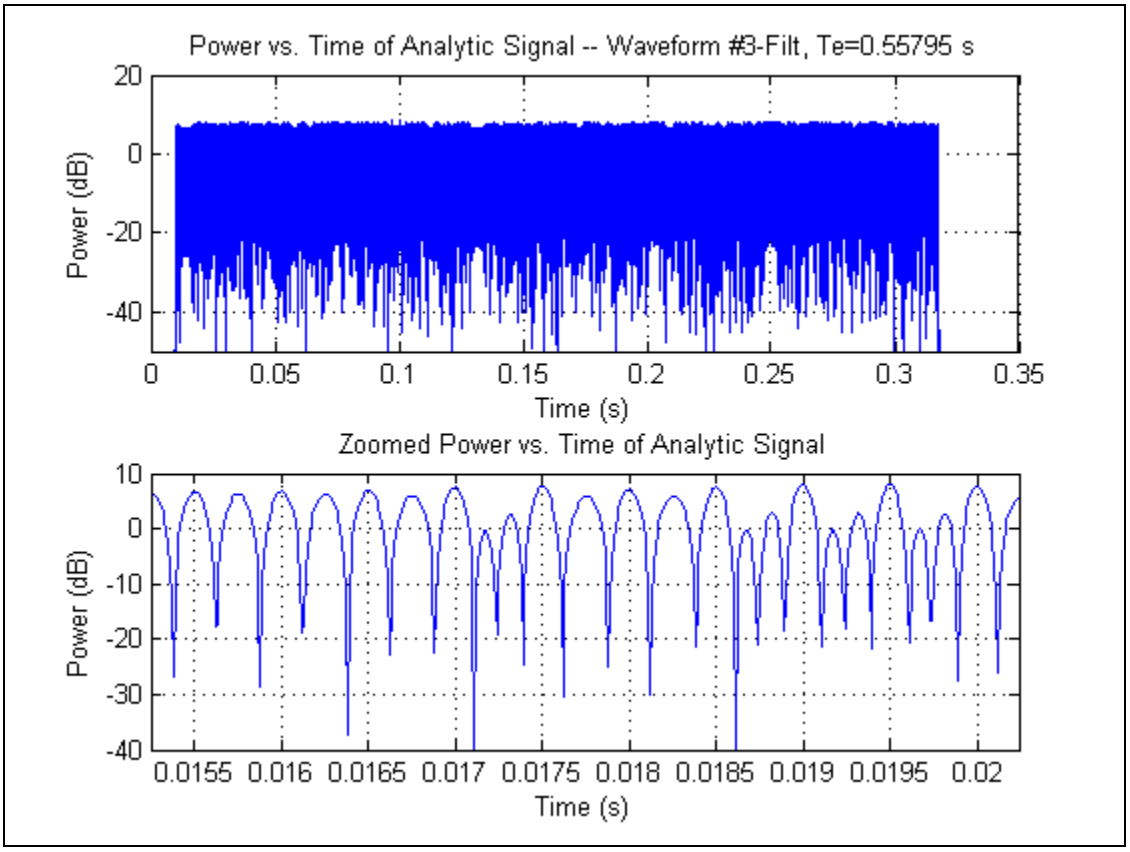

Figure 37 Filtered Waveform #3 – Power vs. Time.

4. Filtered Waveform #4 – "Shortened Pulse"

The PSD of filtered waveform #4 as shown in Figure 38 is very similar to the filtered waveforms #1 and #2, and again β is on the order of 8500 radians per second[5]. This filtered waveform also exhibits nulls that are deeper and wider at chip transitions than for the unfiltered waveform, along with the additional peak power required to compensate for this loss as can be seen in Figure 39.

[5] The different values measured for β may be attributable to the fewer chips transmitted.

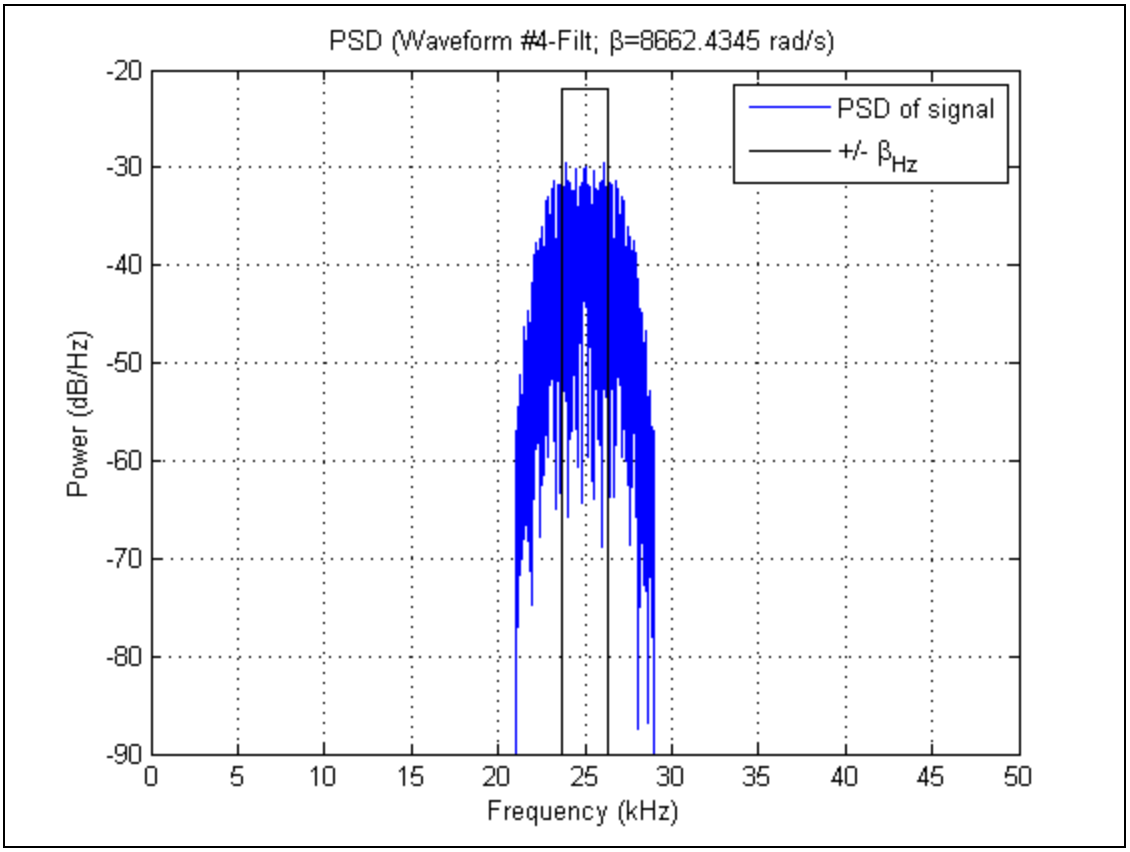

Figure 38 Filtered Waveform #4 – Power Spectral Density.

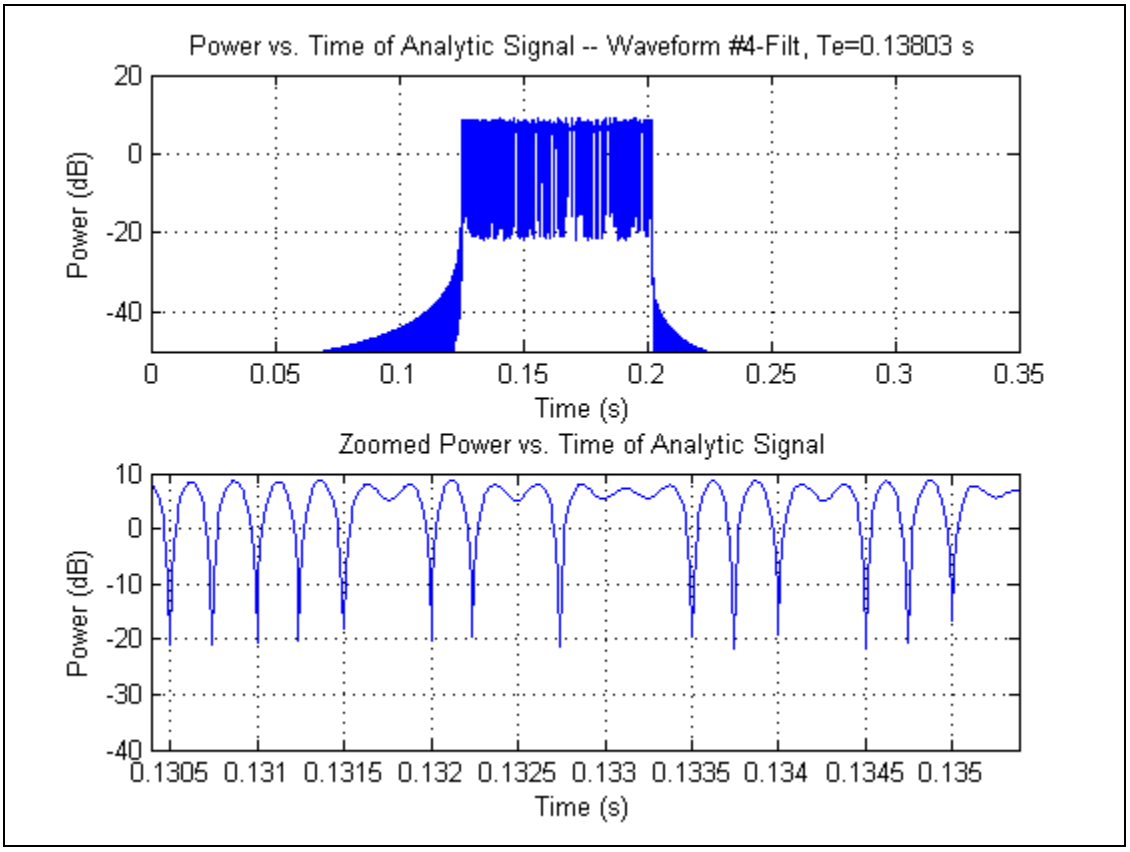

Figure 39 Filtered Waveform #4 – Power vs. Time.

C. SHAPED CHIP WAVEFORMS

A different method from the BPSK signal generator is used to create the waveforms. Recognizing this thesis is assessing the performance of waveforms only in a static collection geometry, arbitrary waveforms can be created and used. Although these waveforms cannot be used in the scenario based generator developed by Johnson [5], they can be effective in assessing performance of different waveforms.

Instead of modulating the carrier with rectangular pulses (chips) as is done for the previous waveforms, this class of waveforms modulates the carrier with sinc shaped pulses to constrain the energy to a limited bandwidth. Applying the Fourier duality and dilation properties to (5.5) gives

$$A\text{sinc}\left[2Wt\right] \leftrightarrow \frac{A}{2W}\text{rect}\left(\frac{f}{2W}\right), \tag{5.17}$$

which shows that because the Fourier transform of a sinc pulse is is zero for $|f| > W$, modulating with a sinc pulse results in a signal that has all its energy constrained within $2W$ [19].

As can be seen in Figure 40, the sinc has its peak at a lag of zero and is zero at lags corresponding to other chip transitions. This particular sinc function has 12 samples per chip and extends out to five chips (it is actually infinitely long, but it is reasonably well approximated over a limited time duration), thus it represents a chip rate of $f_s/12 = 8333$ chips per second. Because the sinc function extends well beyond the particular chip, the transmitted signal is the superposition of all the overlapping sinc functions, which in this case would be ten because that is the length of this particular example. This combined signal is created by passing the impulses corresponding to the chips through a finite impulse response (FIR) filter which has the impulse response shown in Figure 40.

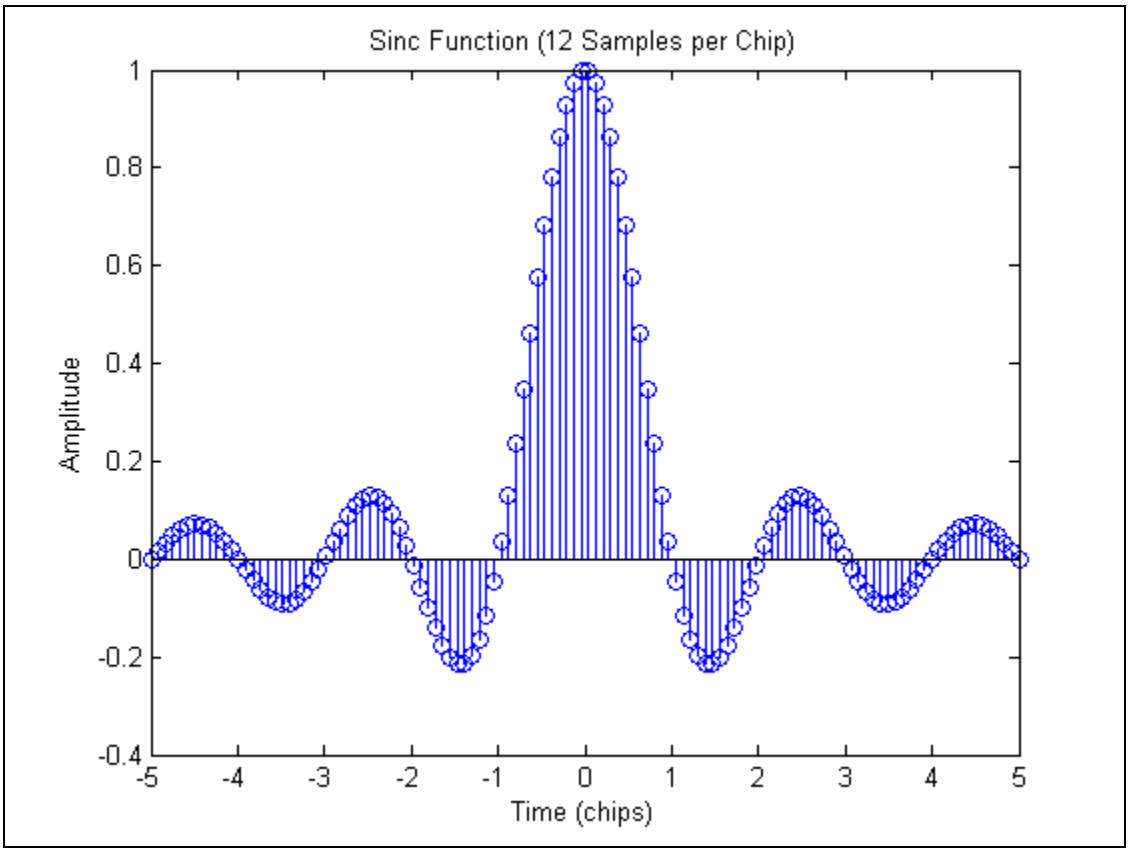

Figure 40 Sinc Function.

Figure 41 shows the PSD of a carrier at $f_s/4$ modulated by the rectangular pulses and sinc pulses. Almost all the energy in the sinc modulated signal is contained in half the null-to-null bandwidth B_{nn} of the BPSK modulated signal, and the sidelobes roll-off much more quickly.

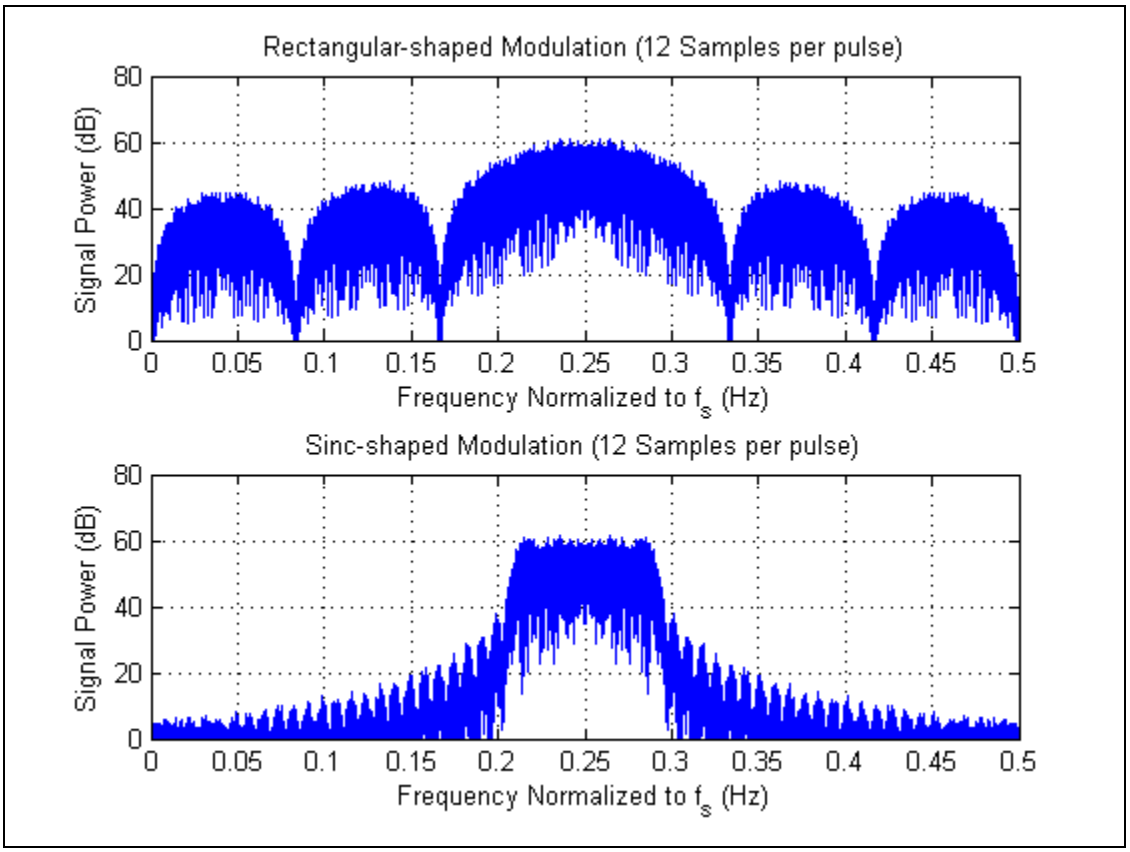

Figure 41 PSD of Rectangular and Sinc Modulated Signal.

Waveform #17 applies the impulse response shown in Figure 40 to the same chip sequence used in the earlier waveforms to generate a signal occupying about the same null-to-null bandwidth B_{nn} as the other waveforms (at 4000 chips per second).

The corresponding rms radian frequency is about 15,000 radians per second (Figure 42), slightly better than the filtered waveform #3 and without the tell-tale double hump of Figure 36. The time domain plots (Figure 43) show that slightly less peak power is required to send waveform #17 with the same energy as waveform #3 (Figure 37). The autocorrelation of waveform #17 (Figure 44) shows the peak minor correlations are at a lower level than for the filtered waveform #3, probably because more chips are transmitted.

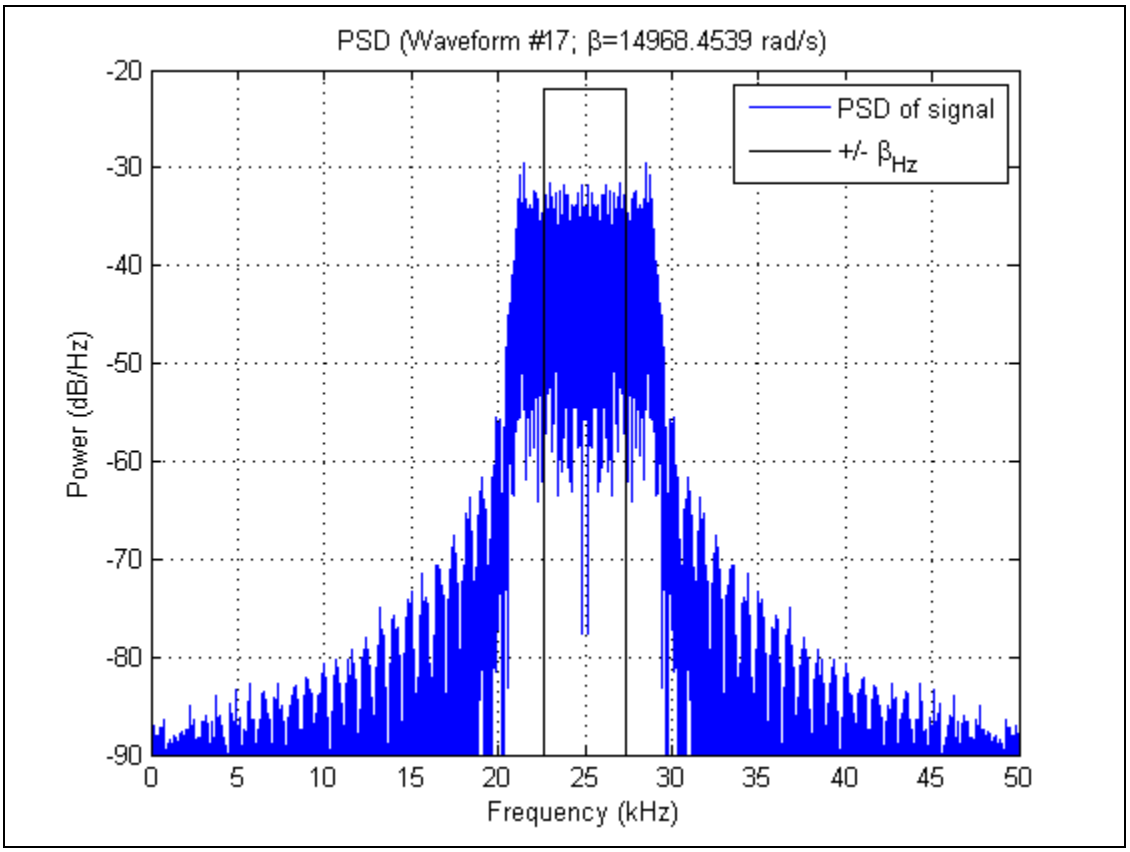

Figure 42 Waveform #17 – Power Spectral Density.

Shaping the chips is very effective in constraining the frequency and can reduce or eliminate the need to filter the signal. The PSD of the filtered waveform #17 (Figure 45) is fairly similar to the filtered signal and the time domain plots (Figure 46) show negligible difference between filtered and unfiltered versions. Because of this, the simulations use only the unfiltered version of waveform #17.

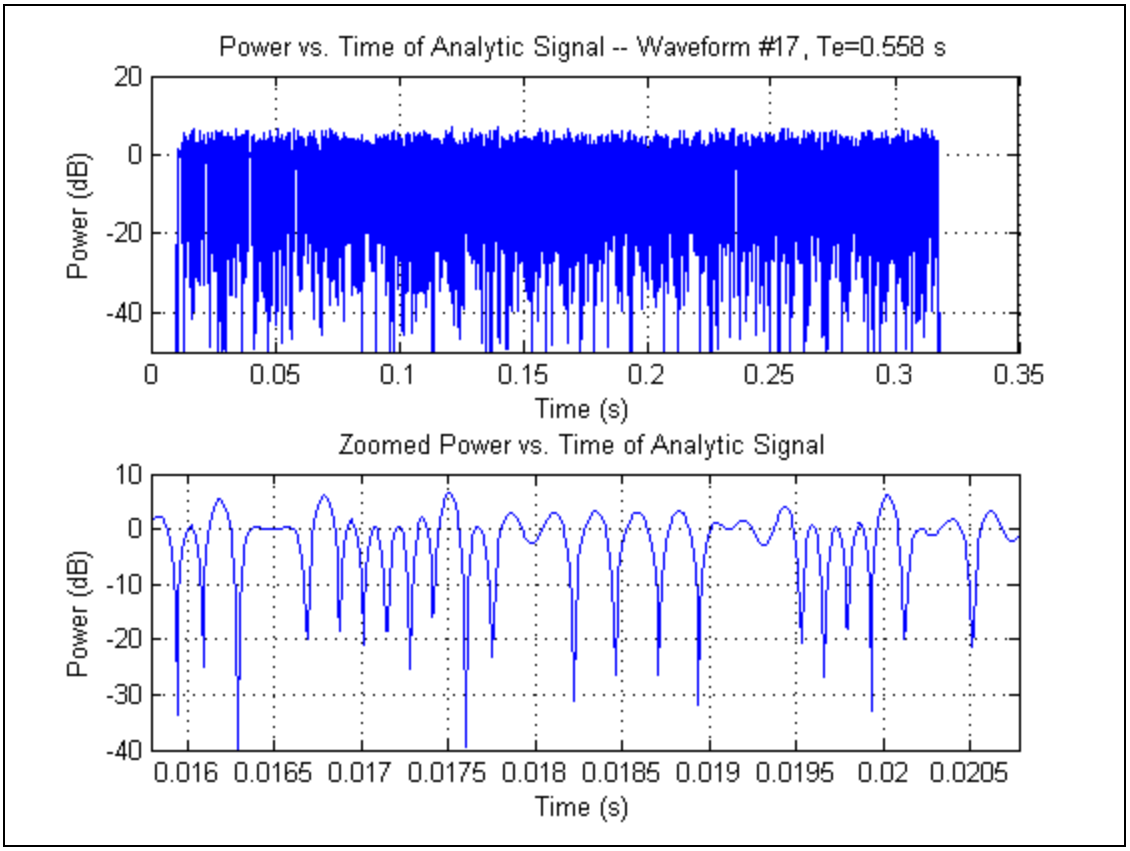

Figure 43 Waveform #17 – Power vs. Time.

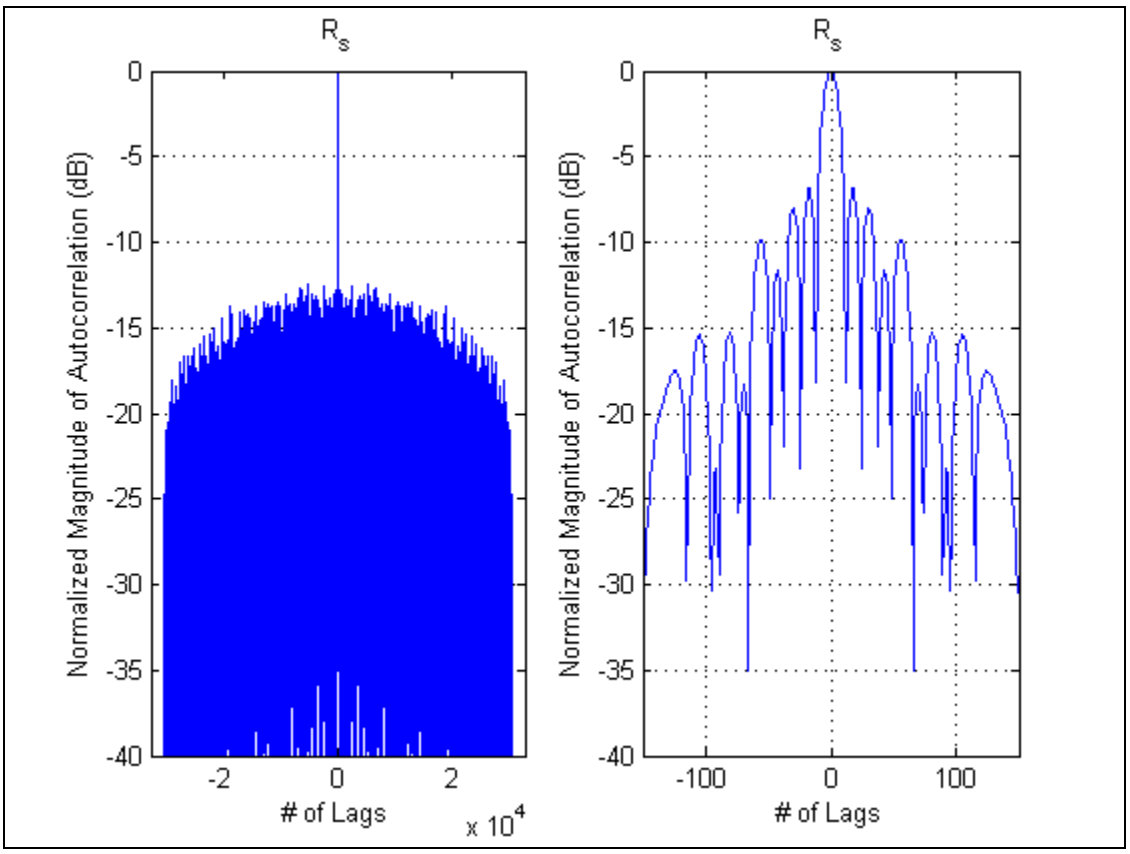

Figure 44 Waveform #17 – Autocorrelation.

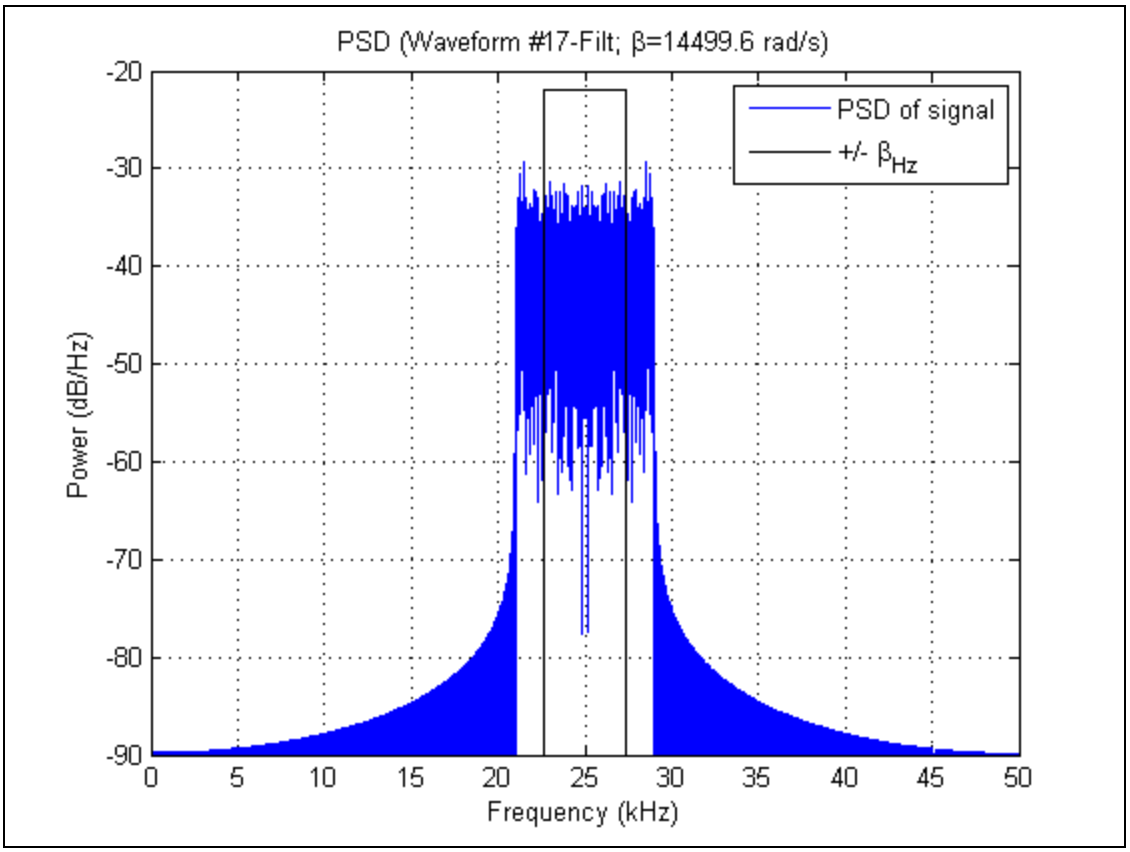

Figure 45 Filtered Waveform #17 – Power Spectral Density.

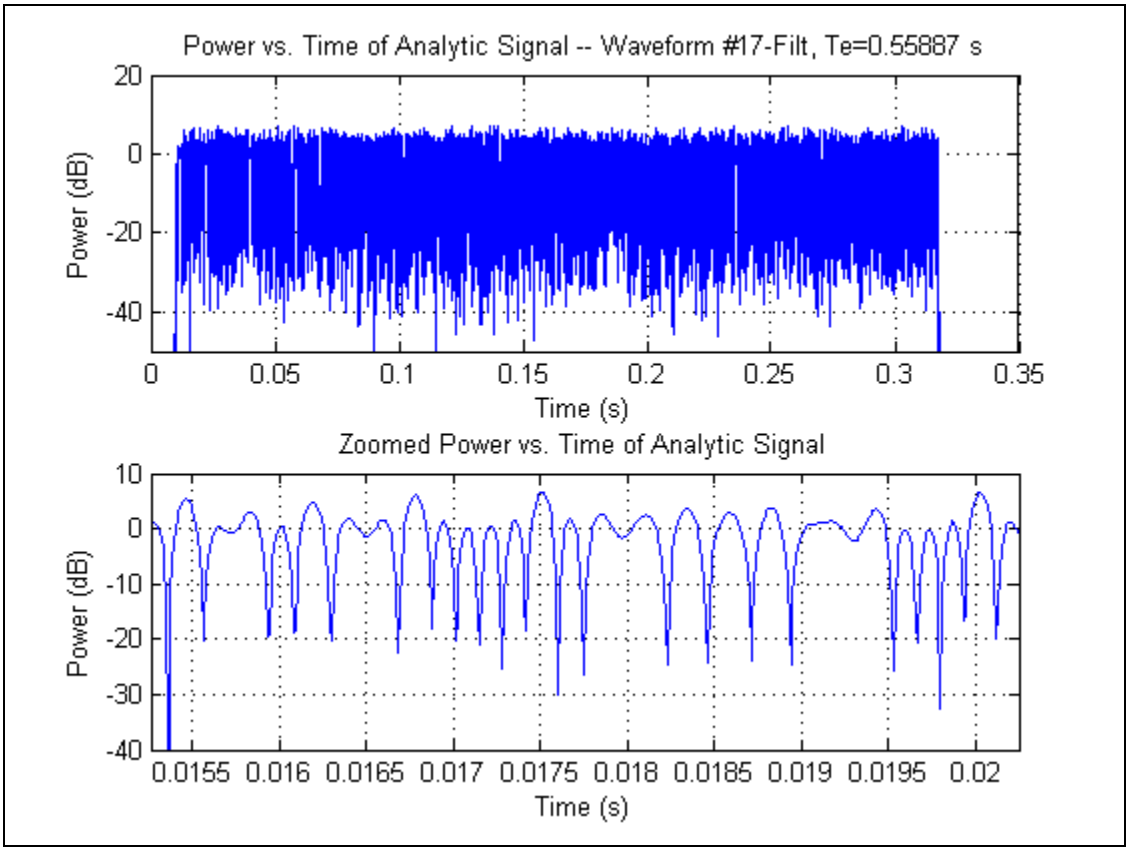

Figure 46 Filtered Waveform #17 – Power vs. Time.

Other waveforms were produced using different numbers of samples per chip to generate different bandwidth signals of the same duration. The relative efficacy of these various waveforms to support accurate geolocation are compared using the results of the simulations discussed in Chapter VI.

VI. SIMULATION SOFTWARE

This chapter presents the overall processing performed by the simulations, describes the MATLAB routines developed or modified, and discusses scripts developed to perform specific simulations. The next Chapter explains the results of the simulations of the various waveforms, and the appendix lists the code from the various MATLAB m-files.

A. SIMULATION OVERVIEW

The purpose of the simulations was to compare the TOA and FOA performance that could be achieved by the different waveforms under various SNR levels, where SNR is E_s/N_0. The figure of merit used to assess performance is the standard deviation of the TOA and FOA estimates for the signal calculated across the different realizations of noise at a given level.

The simulations are run for variations of waveforms to first compare the performance of the filtered vs. the unfiltered BPSK-based waveforms (i.e., unfiltered and filtered waveforms #1-4). Next, the reference waveform and the shaped chip waveforms (i.e., waveform #1 and waveforms #11-16) are compared. Finally, finally the bandwidth constrained waveforms (shown in Figure 18) are compared along with the reference waveform at various chip rates. Unless otherwise specified, the chip rate used is $R_c = 4$ kcps to maintain the same collector bandwidth, which is defined to be the null-to-null bandwidth B_{nn}.

The main reason code from [5] was chosen was to allow the simulations to be performed in dynamic collection scenarios to assess detection performance of a moving target by a single collector. The code generates a BPSK signal and projects this waveform onto two different collectors at specified locations and velocities. This enables one to synthesize signals that have time and frequency offsets as one would have when performing a matched filter detection between a

known reference signal and a distorted received signal. The reference, or basis, waveform s corresponds to the signal received by one of the static collectors, and the received signal r corresponds to the signal received by the other collector. Because the simulations performed in this thesis are static, the two collectors are at the same location and have no velocity and the emitter has no velocity. Thus the generator produces two signals with zero time difference of arrival (TDOA) and zero frequency difference of arrival (FDOA). The simulation would support future analysis involving moving collectors and/or emitter.

The core of the simulation is the MATLAB code main_simulation.m, which loads in various parameters to define the reference and received signals, generates these signals, iterates over a number of noise realizations that are added to the noiseless received signal, and processes each iteration to find the TOA and FOA values that give a peak CAF output. The resulting array of TOA and FOA values can then be processed by the script display_toa_foa_v_snr_and_prep_data.m, which computes and plots the mean and standard deviation for the TOA and FOA at each SNR value for that waveform. These values for each waveform are renamed to a unique variable name (e.g., WFname.stat_summary_array) that is then saved in a MATLAB mat-file of the same name for use by the MATLAB script script_toa_foa_v_snr_across_runs_mrkrs.m, which generates the plots containing multiple waveforms shown in the next chapter.

Figure 47 shows a high-level view of the MATLAB code written or modified for this effort. The m-files, which are shown in the boxes, fall into two basic categories, scripts shown on the left side and routines shown on the right. The script files are custom written for a particular set of simulation runs, and the routines are code that accepts configurations and should not need to be modified to perform different runs. In addition, some of the mat-files are shown along with arrows to indicate source and destination of the data. Note that some of the routines are indented beneath others to indicate what routine calls it. For example, main_simulation.m calls generate_waveform.m, and filt_bnn_fft.m is

called by both generate_waveform.m and get_canned_waveform.m. In addition, some of the mat-files are shown along with arrows to indicate source and destination of the data. For example, mls_gen.m is used to create the file mls65535a.mat, which in turn is used by gen_sinc.m to create the file sinc_XX_mls65535.mat.

Of the MATLAB files shown in Figure 47, only gen_sig.m and CAFv2.m are based on existing code. In addition, the following three files are called by CAFv2.m but have not been modified and thus are not presented here: shiftud.m, tdoa_fdoa.m, and caf_peak.m. All the m-files files shown Figure 47 are listed in the appendix.

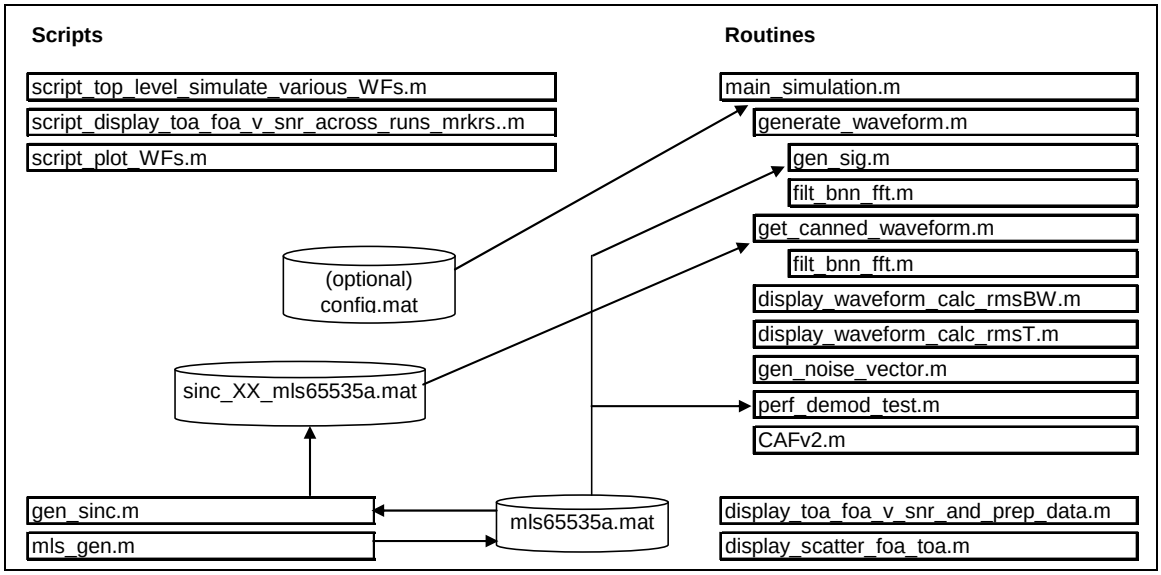

Figure 47 MATLAB m-files Created or Modified.

The following sections of this chapter provide additional detail on each of the various MATLAB routines and scripts used to model the waveforms, simulate TOA and FOA estimation, and process the resulting data. The resulting plots are shown and described in the next chapter.

B. ROUTINES

The routines are MATLAB code m-files that accept parameters and do not need to be edited or modified to perform different simulations of the proposed waveforms. The most significant one of these is main_simulation.m, which reads in a configuration file, if one exists, defining the simulation parameters and in turn calls a number of custom MATLAB functions as shown in Figure 47. Two other m-files that can be used without modification are display_topa_foa_v_snr_and_prep_data.m, which performs the statistical calculations (i.e., finds the mean and standard deviation) on the data generated in the main code, and display_scatter_foa_toa.m, which generates scatter plots of the TOA and FOA data the outputs from the main code to better understand the distribution of the data.

1. main_simulation.m

The core of the simulation is main_simulation.m, which creates an array of TOA and FOA estimates for a desired waveform at multiple SNR values. Most basically, this routine defines operating parameters using configuration data, generates clean versions of the received and reference signals, and then for the desired number of itereations, adds noise to the “clean” received signal and performs the CAF process to determine the combined TOA and FOA values giving the peak correlation magnitude.

```
- get configuration (e.g., WF#, chip rate and filtering, collection geometry)
- LOOP for each SNR value
    - LOOP for offset between reference and received signal
        - generate clean received and reference signals
        - calculate and plot rms radian frequency (if enabled)
        - calculate and plot rms duration (if enabled)
        - LOOP for each noise realization
            - generate noise and and add to received signal
            - compute analytic signal (Hilbert Transform)
            - peform BER test (if enabled)
            - compute crosscorrelation
            - if detection, find TOA & FOA at max CAF amplitude
        - end loop
    - end loop
- end loop
```

Figure 48 Overview of main_simulation.m.

The routine uses the parameters summarized in Table 4 to control processing. The user can either edit the routine to modify the default parameters (allowing him to run the routine directly from the MATLAB interface) or place these values in a file named config.mat to enable running the routine with different parameter values. These parameters include setting the waveform number and whether filtering is on or off, the carrier frequency, the sampling frequency, the chip rate, the length of the waveform in samples, the SNR values to be processed, the number of iterations (noise realizations) at each SNR value, various monitor and debug settings, the collection scenario geometry, and dither variables.

Table 4 Summary of main_simulation.m Parameters.

- Waveform
  - waveform number
  - filtering on/off
- RF carrier frequency (Hz)
- Sampling frequency (Hz)
- Chip rate (Hz) ['Rsym']
- Signal length (Samples)
  - zero_pad length
  - padded vector length
- SNR (Es/No)
  - min value
  - max value
  - step size
- Iterations at each SNR
- Monitor and debug settings
- Collection scenario geometry
  - Position of collector #1
  - Velocity of collector #1
  - Position of collector #2
  - Velocity of collector #2
  - Position of emitter
  - Velocity of emitter
- Dither variables

Several of these parameters define the waveform characteristics. The waveform number and filtering are for the proposed waveforms as defined in Chapter V. The carrier frequency, chip rate[6], and sampling frequency further define the waveform. The carrier frequency affects the location of the signal within the digitized bandwidth and also affects the Doppler frequency offset in a nonstatic collection geometry [5]. Using carrier frequencies greater than the Nyquist frequency work because the signal aliases into a different Nyquist zone [32]. The length N (samples) of the desired signal must also be specified. The routine allows a vector to be specified as the waveform plus padding zeros of length pad_length to support better unnormalized correlation statistics. The

[6] Occasionally this document uses the term symbol rate for chip rate because the legacy BPSK modulator treats each chip as a symbol; however, the entire waveform is only a single symbol, so no confusion should exist.

CAF processing becomes extremely inefficient if the total length of the vector processed is not 2^n, where n is an integer, thus N should be specified as $N = 2^n - pad\_length$.

The SNR values are specified by defining the minimum (starting) SNR value (dB), the step size for the SNR (dB), and the maximum SNR value (dB). Depending on the minimum value and step size, the maximum may not actually be processed. The total number of steps must not be greater than eight if verbose_plot_waveform is not equal to zero, because this will cause an error in trying to plot too many subplots in a figure. The user must also specify the number of monte carlo runs $no\_noise\_iterations$ using different realizations of the noise random vector for each SNR value.

The monitor and debug settings include verbose, verbose_wf_gen, and verbose_plot_wf. The former enables additional outputs (should be set to zero for normal processing) and the latter two enable additional plotting of the waveform and processing. Process_detections allows CAF processing if a signal is detected; setting this to zero allows much faster operation of the code to support simulating detections but not TOA and FOA estimates. Setting enable_BER_test enables the running the BER test function, which was used to verify the noise vector had the correct amplitude. BER testing is discussed later.

The collection geometry settings specify the location and velocity for each of the two collectors and the emitter. The position information is in the form of an array $[x, y, z]$, where x, y, and z are the respective distance in meters from a reference, and the velocity information is in a similar format defined in meters per second. This information is used to generate the BPSK-based waveforms, only, and enables generation of signals that have Doppler effects [5]. Because this thesis is only investigating performance in a static collection geometry (i.e., no Doppler), the velocity values are all set to zero and the position of the two collectors are set to be equal. This information does not have any affect on waveforms #11-17 which are pre-formed.

2. generate_waveform.m

The function main_simulation.m generates the noise-less reference and receive signals using generate_waveform.m for BPSK-based signals (waveforms #1-4, unfiltered and filtered) or get_canned_waveform.m functions for those that are fixed (i.e, waveforms #11-17). The generate waveform manipulates the signals produced by the gen_sig.m function to create waveforms #1-4, and filter them if enabled, to the null-to-null bandwidth B_{nn}. The function can also produce additional plots of the waveform produced, if enabled. It is invoked using

[S1,Sref] = generate_waveform(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N, ...

wf_type, pad_length, filter_outside_bnn, verbose),

where S1 and Sref are the noise-free receive and reference signals, respectively, and the input arguments are from the configuration previously discussed.

Waveform #1 is the signal provided by gen_sig.m. Waveforms #2 and #4 manipulate this signal by removing either the middle or outer three-fourths of the signal and rescaling the amplitude so the total energy of the signal is the same as the original.

Waveform #3, on the other hand, sums the signals generated by calling gen_sig.m twice with a "new" carrier frequency $f_0 = f_{0,orig} \pm R_{sym}/2$ and a new chip rate $R_{sym} = R_{sym,orig}/2$. The amplitude of this new summed signal is then scaled so it has the same energy as waveform #1.

3. gen_sig.m

The function gen_sig.m generates two noiseless BPSK modulated signals as would be received by two collectors receiving an emission in the defined collection scenario. The simulation accurately models the Doppler effects, including frequency offsets as well as time dilation and compression of the modulating signal. BPSK modulation is performed starting with the first bit from the file mls65535a.mat. using the parameters passed to it. The function is invoked using

[S1,S2] = gen_sig(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N),

where S1 and S2 are the noise-free signals received at the two collectors and the input arguments are passed from the simulation parameters.

The function gen_sig uses the core of the MATLAB© code sig_gen.m developed by Johnson [5] but was changed in name because the significance of the variances. The major changes to this code are

- The function does not prompt for user input,
- Noise is not added within this function,
- The function does not convert the signal into the analytic signal, and
- The bit sequence is read from a file (not random).

Instead of prompting for the input parameters, these values need to be passed into the function when called. Table 5 lists the various user specified settings required by the gensig.m.

Table 5 User Specified Settings in gensig.m.

- Position of collector #1
- Velocity of collector #1
- Position of collector #2
- Velocity of collector #2
- Position of emitter
- Velocity of emitter
- RF carrier frequency (Hz)
- Sampling frequency (Hz)
- Symbol rate (Hz)
- No. of samples collected

4. filt_bnn_fft.m

The function filt_bnn_fft.m performs a bandpass function, filtering out signal energy that is outside the null-to-null bandwidth B_{nn} of the signal. The amplitude of the resulting signal is rescaled so the signal has the same energy as the original signal. The function is invoked using

S = filt_bnn_fft(S, Rsym, f0, fs),

where S is the signal, Rsym is the chip rate, f0 is the carrier frequency, and fs is the sampling frequency.

The function first computes the energy of the signal. It then converts the signal to the analytic form (i.e., no negative frequencies) using the Hilbert function and converts the signal to the frequency domain using the FFT function. At this point, all the FFT bins which correspond to frequencies up to $f_0 - R_{sym}$ along with those corresponding to $f_0 + R_{sym}$ and above are set to zero. The signal is then converted back to the time domain using the IFFT function, made real, and amplitude scaled to restore signal power to that of the original signal.

5. get_canned_waveform.m

The function get_canned_waveform.m loads a predefined waveform. It is invoked using

S1 = get_canned_waveform(Es, N, wf_type, pad_length, Rsym, f0, fs, filter_outside_bnn, verbose_wf_gen),

where S1 is the new waveform, and the only input parameters used are the desired signal energy (Es), the length of the waveform in samples (N), the waveform number (wf_type), and the number of leading and trailing pad zeros.

The function first loads in the proper mat-file depending on the waveform number selected, and then uses only the first N samples. The amplitude of this signal is then scaled to get the desired signal energy. Next the signal is filtered to the null-to-null bandwidth, if enabled, using the previously defined routine. Finally the signal is padded at the front and back with the specified number of zeros.

6. display_waveform_calc_rmsBW.m

The function display_waveform_calc_rmsBW.m calculates the rms radian frequency of the waveform and plots the PSD of the waveform along with the rms radian frequency and rms bandwidth. It is invoked using

display_waveform_calc_rmsBW(Sref, f0, fs, wf_type, filter_outside_bnn),

where Sref is the signal to be analyzed, f0 is the carrier frequency, fs is the sampling frequency, wf_type is the waveform number, and filter_outside_bnn is set to zero if filtering is not desired.

The function displays the value of the variable filter_outside _bnn on the PSD plot to document this setting, and it also plots the Welch PSD, which is a particular type of periodogram, and the weighted and unweighted PSD values used to calculate the rms radian frequency β.

7. display_waveform_calc_rmsT.m

The function display_waveform_calc_rmsT.m calculates the rms duration of the waveform T_e and plots the power vs. time of the entire signal along with a zoomed version. It is invoked using

display_waveform_calc_rmsT(Sref, f0, fs, wf_type, filter_outside_bnn),

where Sref is the signal to be analyzed, f0 is the carrier frequency, fs is the sampling frequency, wf_type is the waveform number, and filter_outside_bnn is set to zero if filtering is not desired.

The function calculates the instantaneous power by squaring the input signal and uses this to calculate the rms duration. This value, along with whether the signal was filtered is printed in the title of the plot.

8. gen_noise_vector.m

As stated earlier, one of the differences between the routine gen_sig.m and the original sig_gen.m developed by Johnson [5] is that the random noise is no longer added within that routine. Instead, this task was extracted and placed

as its own function in main_simulation.m to allow simulating the same signal with multiple realizations of the noise. It is invoked using

Noise=gen_noise_vector(N, SNR, Tsym, fs),

where Noise is a vector of length N such that the values in this vector have zero-mean Gaussian distribution and variance σ^2 to give the desired signal-to-noise power ratio SNR. Tsym and fs are the chip period and sampling frequency, respectively. The amplitude of the signal is assumed to be unity. If this is not true, the signal needs to be scaled accordingly.

Testing of the sig_gen.m code in [5] revealed that the code properly modulated the signal but failed to add the proper noise. Johnson correctly states (in his equation 5-9) that

$$\sigma^2 = \frac{P_s T_{sym} B}{E_s / N_0} \tag{5.18}$$

where the σ is the standard deviation and is used as the factor to scale from the MATLAB© generated *randn* normalized (zero mean) Gaussian random variables to the desired noise values based on specified values for signal-to-noise ratio (SNR) E_s/N_0, signal power P_s, symbol period T_{sym}, and bandwidth B. However the bandwidth B should be actual bandwidth of the digitized signal which is $f_s/2$, and not the digital frequency. Thus the noise samples added to the signal were too low by a factor of $\sqrt{f_s/2}$. Figure 49 shows the spectral plots of the signal with 3dB SNR based on original calculations and the correction for $B = f_s/2$.

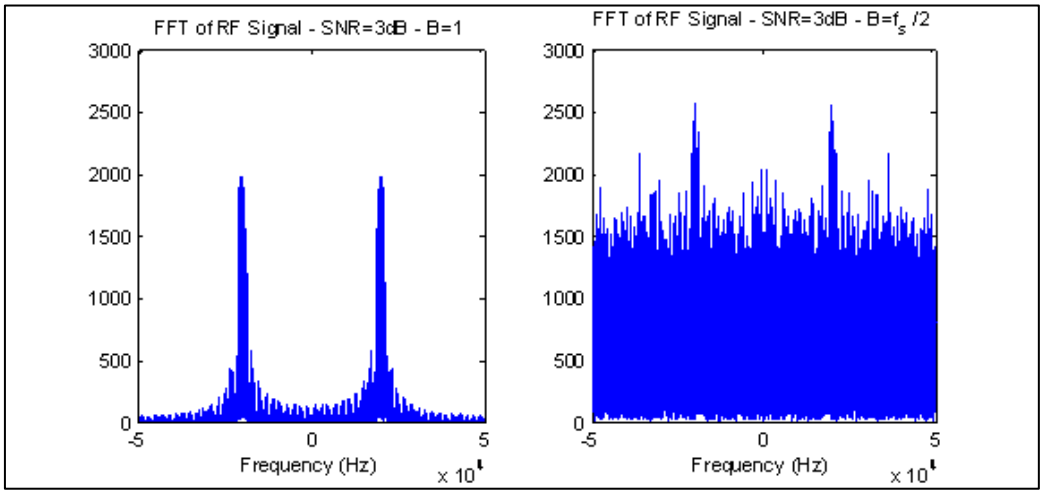

Figure 49 Signal Spectrum Before and After Adjusting Noise Equation.

9. perf_demod_test.m

Before running the simulation, several basic checks were made to assess the accuracy of the outputs of the model, especially in the areas of carrier frequency, symbol rate, and SNR. The function perf_demod_test.m allows these parameters to be verified by attempting to demodulate a reference signal using the modulation parameters. This test is designed to verify the proper operation in the simpler case of a static collection geometry.

The function produces various diagnostic plots and calculates the bit error rate, BER, by comparing demodulated bits to first bits loaded from mls65535a.mat. The user is asked to manually perform phase synchronization by identifying the peak of signal, which assumes no or very low noise (i.e., high SNR). It is invoked using

[BER, no_of_errors, no_of_bits]=perf_demod_test(Sa1, Sa2, fs, f0, Rsym, SNRdB, verbose),

where the returned value BER is the bit error rate calculated by dividing no_of_errors by no_of_bits. Sa1 and Sa2 are the modulated signal with and without added noise, respectively. Other input variables include the carrier

frequency f0, the sampling frequency fs, the waveform number wf_type, and a flag to indicate whether filtering is performed, filter_outside_bnn. To use this function, the parameters listed in Table 6 are suggested when running main_simulation.m.

Table 6 Suggested Parameters When Using perf_demod_test.m.

- verbose=1 - verbose_wf_gen=1 - enable_BER_test=1 - process_detections=0 - wf_type=1 - Es_No_db_min=4.15 (dB) - Es_No_db_max=4.15 (dB) - no_noise_iterations=1 - pad_length=0 - Rsym=2000 or 5000

The demodulation test consisted in generating the signal with an E_c/N_0 of 4.15 dB which should give an average BER on the order of 10^{-2} for BPSK, a carrier frequency f_0 of 20 kHz, a sampling frequency f_s of 100 kHz, a symbol rate R_{sym} of 2 kHz, and 65536 samples. The actual BER, calculated using

$$BER = Q\left(\sqrt{2E_b/N_0}\right) \text{ [20],} \tag{5.19}$$

Shows the BER should be

$$BER = Q\left(\sqrt{2\cdot 10^{4.15/10}}\right) = Q\left(2.28\right) = 1.13\cdot 10^{-2}. \tag{5.20}$$

Running this loop 20 times resulted in 175 errors out of a total of 13,120 bits for an average BER of 0.013 ($1.3x10^{-2}$), indicating accurate modeling.

The demodulation process consists of the following steps:

- mixing the signal back down to baseband using the nominal carrier frequency f_0,
- making sure that the signal is all in the I-channel,
- passing this signal through a matched filter for a pulse,

- downsampling and comparing to a threshold of zero, and
- comparing the resulting bitstream with the modulated bitstream.

Figure 50 shows the result of mixing the signal down to baseband. The plot on the left shows the analytic signal in the frequency domain, and the plot on the right shows the signal after multiplying it by $e^{-j2\pi f_0}$, where f_0 is the carrier frequency, to center the signal back at baseband.

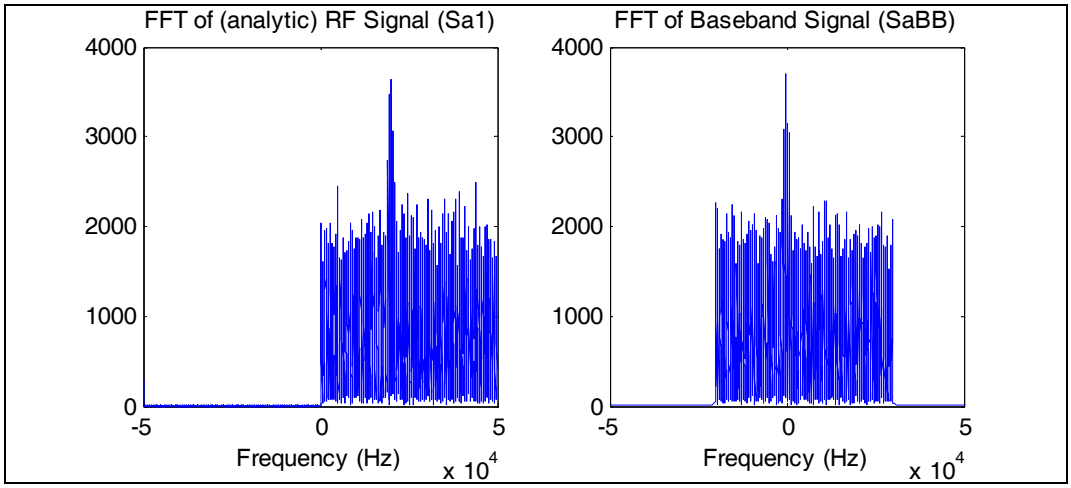

Figure 50 Analytic Signal Before and After Mixing Down to Baseband.

Figure 51 plots the baseband signal in time domain, showing the real component, the imaginary component, and the phase of the signal. Note that almost all the energy, except during bit transitions, is in the I-channel, showing carrier phase synchronization (although ignoring potential phase ambiguity).

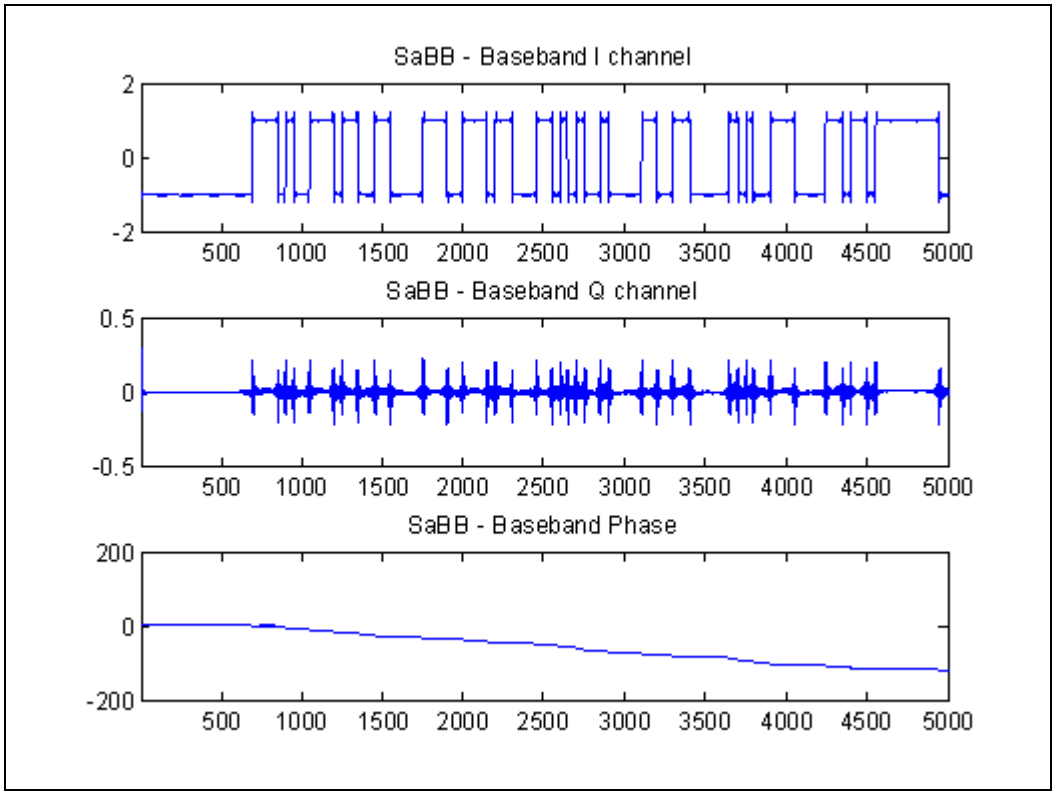

Figure 51 Signal in I-Channel vs. Q-Channel.in High SNR.

Figure 52 shows the output of the matched filter (matched to the pulse) in the top plot. The middle plot shows the output of the comparator at the sample times where the reference is set to zero. In this plot the output is “1” if the sampled decision variable is greater than zero, otherwise the output is “0.” The lines connect the points for improved visibility and are not intended to extrapolate between the points (i.e., the sloped line merely indicates a bit transition). The bottom plot shows the actual data used to modulate the signal. Comparing the two bottom plots, one can see that the bitstream begins with a series of zeros and that the 13th demodulated bit is in error. For BER calculations, the first bit is ignored because it is invalid (i.e., the signal has not yet passed through the matched filter.)

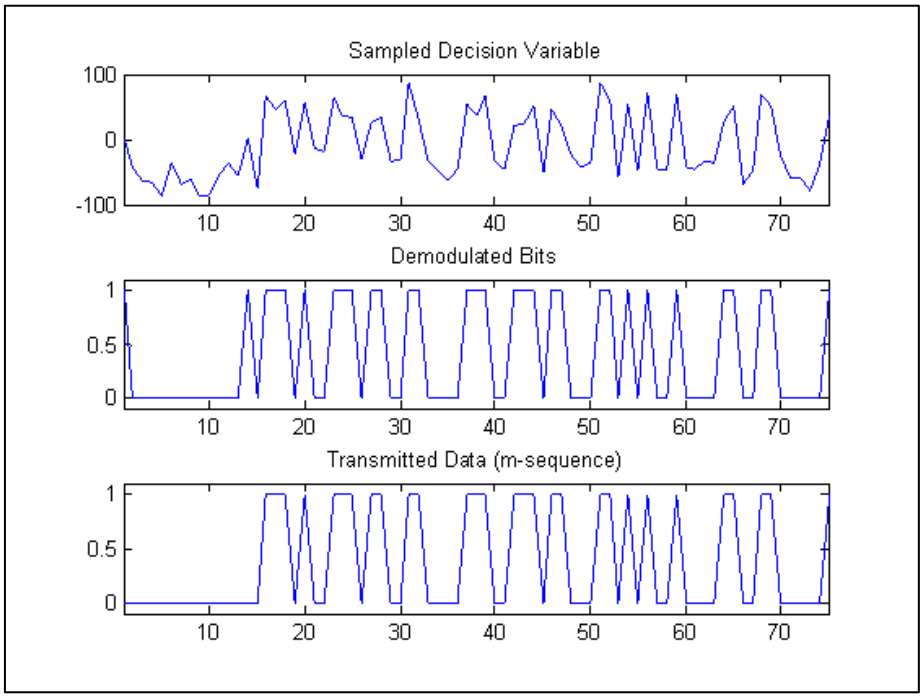

Figure 52 The Sampled Decision Variable, Resulting Bits, and Reference Bits.

10. CAFv2.m

The function CAFv2.m returns the TOA and FOA corresponding to the peak amplitude of the CAF function. It is invoked using

[TDOA, FDOA] = CAFv2(S1, S2, Max_f, fs, Max_t, display_CAF_peak),

where S1 is the analytic form of the noisy receive signal and S2 is the noise-free analytic reference signal. Max_f and Max_t define the maximum expected FOA and TOA values (i.e., they set the CAF search window). Finally, the routine gets input variables specifying the sampling frequency f_s and whether to plot the resulting CAF.

The function CAFv2.m is almost the same as the function CAF.m developed in [5], except the user inputs were removed so that the process keeps iterating until it determines that it has reached its maximum accuracy. The function CAF utilizes Stein's method [3] to initially compute estimates of TDOA

and FDOA between S1 & S2 before switching to using "fine mode" calculations. The speed of the processing is severely degraded if the length of the signal (in samples) is not 2^n, where n is an integer. The routine calls CAF_peak.m, if enabled, to plot the results of the CAF process. [5]

11. display_toa_foa_v_snr_and_prep_data.m

The function display_toa_foa_v_snr_and_prep_data.m uses the arrays of TOA and FOA estimates produced by main_simulation.m, which still reside in the MATLAB workspace,to compute the mean and standard deviation for the TOA and FOA at each of the SNR values simulated and then plots this data as a function of SNR. It is invoked using

display_toa_foa_v_snr_and_prep_data.

After running this routine, the array named *stat_summary_array*, containing these data, resides in the MATLAB workspace.

12. display_scatter_toa_foa.m

The function display_scatter_foa_toa.m also reads in the arrays of TOA and FOA estimates produced by main_simulation.m, which still reside in the MATLAB workspace. It generates a scatter of the FOA vs. TOA for each of the SNR levels. It is invoked using

display_toa_foa_v_snr_and_prep_data.

C. SCRIPT FILES

The script files are MATLAB code m-files that are customized for a given set of simulations performed. Unlike the routines which are controlled through parameters but don't need to be edited, these files need to be customized with the various parameters embedded into them.

Three scripts are listed here. The first script, script_top_level_simulate_various_WFs.m, is used to create summary data files for simulations of the various waveforms. The second,

script_display_toa_foa_v_snr_across runs_mrkrs.m, reads in these various files and creates the performance plots shown in the next chapter. Finally, the third script, script_plot_WFs.m, is used to generate plots and data about each of the waveforms. The scripts use the routines previously defined as well as operate directly on some of the variables in the MATLAB workspace. In addition, two other m-files are presented, gen_sinc.m, which is used to create the fixed waveforms #11-17, and mls_gen.m, which creates m-sequences, maximal length PN sequences as defined in [30].

1. script_top_level_simulate_various_WFs.m

The script script_top_level_simulate_various_WFs.m is used to call main_simulation.m and save the resulting TOA and FOA statistics into appropriately named data mat-files. For each of the waveform parameters configurations shown in Table 7, the script

- clears the workspace,
- runs a short m-file that has the configuration information,
- saves the workspace as config_file.m.,
- runs main_simulation.m to simulate using these data,
- creates the statistical summary data by calling display_toa_foa_v_snr_and_prep_data.m,
- renames the variable to match the name shown inTable 7, and
- saves this as a mat-file of the same name.

When execution is completed, data for each waveform run is saved in its own file.

Table 7 Waveform Variations Simulated.

Name	WF#	Filtered	Iterations	Chip Rate
WF1It1000Rs4000	1	No	1000	4 kcps
WF2It1000Rs4000	2	No	1000	4 kcps
WF3It1000Rs4000	3	No	1000	4 kcps
WF4It1000Rs4000	4	No	1000	4 kcps
WF1filtIt1000Rs4000	1	Yes	1000	4 kcps
WF2filtIt1000Rs4000	2	Yes	1000	4 kcps
WF3filtIt1000Rs4000	3	Yes	1000	4 kcps
WF4filtIt1000Rs4000	4	Yes	1000	4 kcps
WF1filtIt1000Rs1000	1	Yes	1000	1 kcps
WF1filtIt1000Rs2000	1	Yes	1000	2 kcps
WF1filtIt1000Rs8000	1	Yes	1000	8 kcps
WF1It1000Rs1000	1	No	1000	1 kcps
WF1It1000Rs2000	1	No	1000	2 kcps
WF1It1000Rs8000	1	No	1000	8 kcps
WF11It1000	11	No	1000	25 kcpc
WF12It1000	12	No	1000	12.5 kcps
WF13It1000	13	No	1000	6.25 kcps
WF14It1000	14	No	1000	3.13 kcps
WF15It1000	15	No	1000	1.66 kcps
WF16It1000	16	No	1000	0.83 kcps
WF17It1000	17	No	1000	8.3 kcps
WFfilt17It1000	17	Yes	1000	8.3 kcps

2. script_display_toa_foa_v_snr_across_runs_mrkrs.m

The script script_display_toa_foa_v_snr_across_runs_mrkrs.m is used to read in the data files saved in the last section and plot the data. These plots are shown in the next chapter.

3. script_plot_WFs.m

The script script_plot_WFs.m is used to generate plots and data, β and T_e, for a given waveform. The script sets the variables to be used by main_simulation.m, saves these in the mat-file config_file.mat, and calls the main simulation routine.

4. gen_sinc.m

The script gen_sinc.m was used to create waveforms #11-17. It loads the file mls65535a.mat, converts data bits into bipolar pulses, and shapes these into sinc pulses. For a given waveform, the code generates a sinc pattern signal +/- five chips wide with the specified number of samples per pulse to create a FIR filter impulse response. The bipolar pulses are also upsampled by the number of samples per bit, where the new "samples" are equal to zero. This new data is run through the FIR filter and used to modulate a carrier at $f_c/4$. Table 8 shows by waveform number the number of samples used to make each sinc pulse. The resulting chip rate R_c shown in Table 7 is the sampling frequency f_s divided by the number of samples per pulse. The resulting vector, *modulation*, is saved into the respective mat-file as shown in Table 8.

Table 8 Samples per Shaped Pulse.

Waveform#	Samples per Pulse	mat-file name
11	4	sinc_wb_mls65535a
12	8	sinc_mb_mls65535a
13	16	sinc_nb_mls65535a
14	32	sinc_vnb_mls65535a
15	64	sinc_unb_mls65535a
16	128	sinc_xnb_mls65535a
17	12	sinc_12Spc_mls65535a

5. mls_gen.m

The script mls_gen.m is used to create a maximal sequence, m-sequence, using the linear feedback shift register (LFSR) parameters selected. The particular configuration shown in the appendix is for a 65536 bit long m-sequence using a 16-bit LFSR with feedback from the taps 1, 3, 12, and 16. The two vectors it creates are *mls_code*, in which each element $\in \{0,1\}$, and *signal_sent*, in which each element $\in \{-1,1\}$. The script also plots the autocorrelation of the generated sequence to allow a user to assess the autocorrelation properties.

This chapter presented the MATLAB files used to perform the simulations. The code itself is included in the appendix. The resulting plots are shown and described in the next chapter.

VII. RESULTS AND CONCLUSIONS

This chapter discusses the specific simulations performed and explains the results of the simulations of the various waveforms.

A. SIMULATIONS PERFORMED

Each of the waveforms presented in Chapter V was generated and processed over 1000 realizations of noise for each E_s/N_0 value ranging from 0 to 35 dB in steps of 5 dB. All of the waveforms use the sampling frequency $f_s = 100$ kS/s and the number of samples of the waveform (not including zero padding) is $N = 30720$ Samples. The chipping rates R_s are the same as defined in Table 3, and the carrier frequency f_0 is 20 kHz for waveforms #1-4 (both unfiltered and filtered) and 25 kHz, which is $f_s/4$, for waveforms #11-17. The resulting values for β and T_e are also shown in Table 3.

The resulting statistics (*summary_array*) from each of these runs provides mean and standard deviation for TOA and FOA at each E_s/N_0. The MATLAB script *script_top_level_simulate_various_WFs* configured the settings for each waveform, called the main MATLAB routines to run the simulations, *main_simulation*, and to generate the summary statistics for the waveforms, *display_toa_foa_v_snr_and_prep_data*, and saved the resulting summary statistics in the appropriately name MATLAB .mat data file.

The MATLAB script *script_display_toa_foa_v_snr_across_runs* reads in all these saved data files and plots the standard deviations for TOA and FOA on a logarithmic scale. The mean of these values is not plotted because they were all about zero, as expected.

B. RESULTS OF SIMULATIONS AND COMPARISON

The results of simulation showed the standard deviations of the TOA and FOA estimates found in simulation matched the expected values determined by

(4.38) and (4.39). For waveforms of similar SNR, the standard deviation of the TOA, σ_{TOA}, was inversely related to β as defined in (4.41). Likewise, for waveforms of similar SNR, the standard deviation of the FOA, σ_{FOA}, was inversely related to T_e as defined in (4.42). Because SNR is defined to be E_s/N_0, as opposed to E_c/N_0, γ will not undergo improvement due to processing gain and therefore the quantity BT in (4.38) and (4.39) equals unity.

Three sets of comparisons are presented. Unless otherwise specified, the chip rate is $R_c = 4$ kcps. First, the BPSK-generated waveforms (i.e., waveforms #1-4 filtered and unfiltered) are compared. Next, the reference waveform and various shaped-chip waveforms (i.e., waveform #1 and waveforms #11-16) are compared. Finally, the bandwidth constrained waveforms (shown in Figure 18) along with the reference waveform at various chip rates are all compared. For each of these, TOA and FOA data are plotted for the waveforms under consideration along with the theoretical performance expected for the filtered waveform #1 derived using (4.38) and (4.39).

1. BPSK-Generated Waveforms

The first comparison made is between the filtered and unfiltered BPSK generated waveforms all at the same chip rate, and hence the same null-to-null bandwidth B_{nn} as shown in Table 3. Filtering of a BPSK signal will reduce the rms radian frequency β, which would be infinite for an infinite bandwidth receiver, but the rms duration T_e would remain unchanged. Thus, one would expect to see a larger standard deviation σ of TOA but no change for σ of FOA going from a particular waveform (i.e., waveform #1-4) to its corresponding filtered version.

Waveforms #1-4, both filtered and unfiltered, were simulated with 1000 noise realizations for each waveform at each SNR. The standard deviation σ of the TOA and FOA values determined from these simulations are plotted in Figure

53 and Figure 54, respectively. The data points, which are at SNR values in 5 dB steps and indicated by the symbols, are connected by straight line interpolations. These lines are not meant to imply the actual values between the data points but to aid visualizing the data points and observe trends.

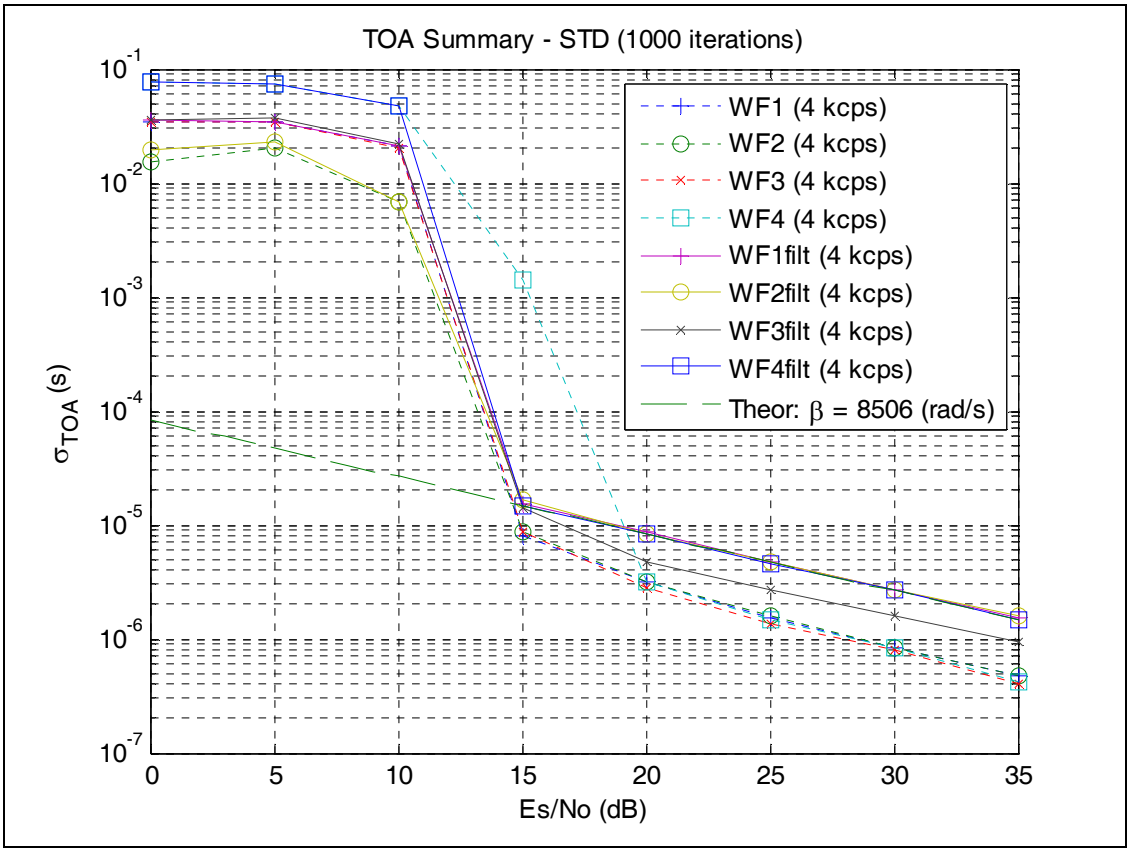

Figure 53 TOA Accuracies – Unfiltered BPSK vs. Filtered.

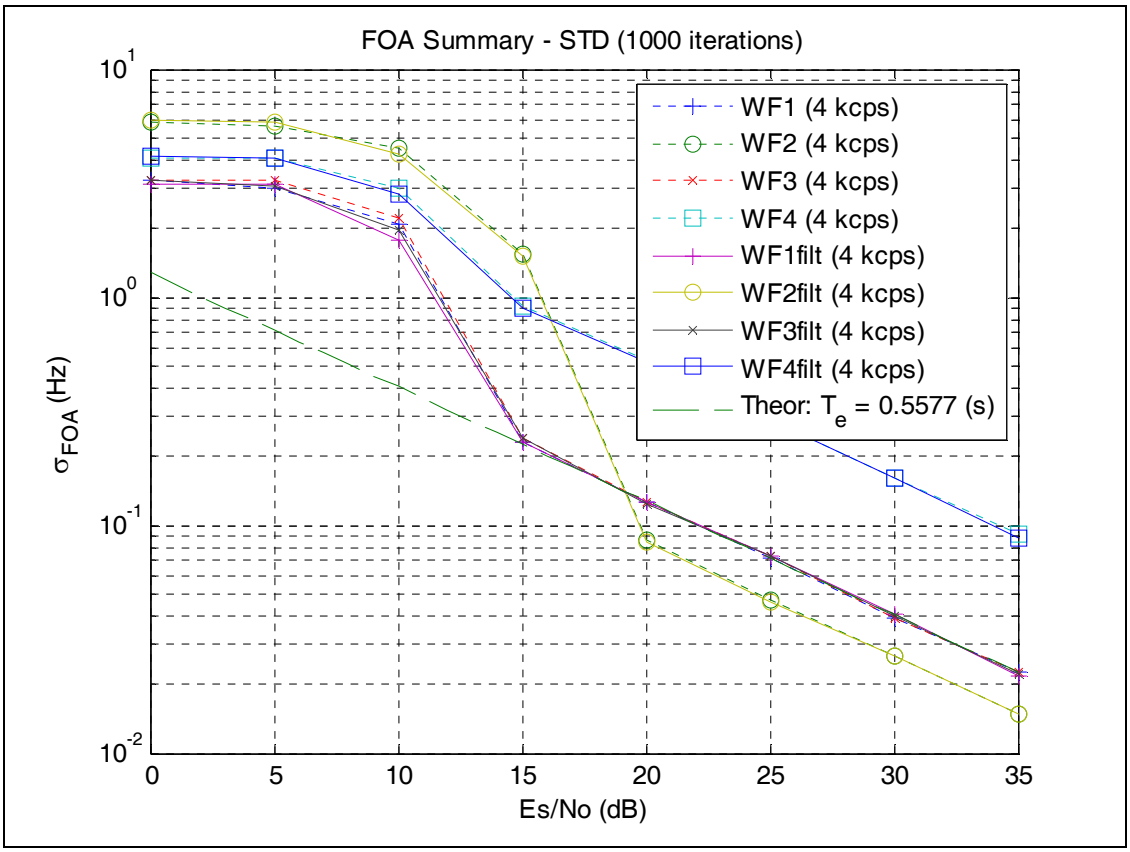

Figure 54 FOA Accuracies – Unfiltered BPSK vs. Filtered.

In addition, the theoretical TOA and FOA values are also calculated for the filtered waveform #1 for various SNR and plotted on the respective plot. These values are calculated using (4.38) and (4.39), where β and T_e are extracted from Table 3, the waveform duration T is from (5.1), the signal bandwidth out of the receiver is $1/T$, and γ is twice the SNR defined as E_s/N_0.

As can be seen on the right side of these plots, at high SNR values ($\geq 20dB$), all the curves either match the theoretical curve or are parallel with it, and at low SNR values ($\leq 10dB$), the curves flatten out with a very poor σ indicating the spurious detections throughout the CAF space (i.e., the region being searched over TOA and FOA). This is consistent with Stein [3] who comments, “In order for the desired lobe peak to be uniquely identified (very low probability of spurious noise lobes exceeding a detection threshold), the SNR in

the output has to exceed about 10 dB." Viewing the resulting CAF at high and low SNR helps to illustrate this. Figure 55 shows an example CAF output (magnitude only) of waveform #4 in a basically noiseless environment (100 dB SNR). Note how the peak of the mainlobe in the center of the plot is easily discernable. Figure 56, on the other hand, shows an example CAF of the same waveform with 0 dB SNR (i.e., the noise power is equal to signal power). Note in this case how the peaks can be seen distributed throughout the CAF space. Because the CAF space is limited, it will set a limit on how poor σ can become, thus causing the flattening of the curves.

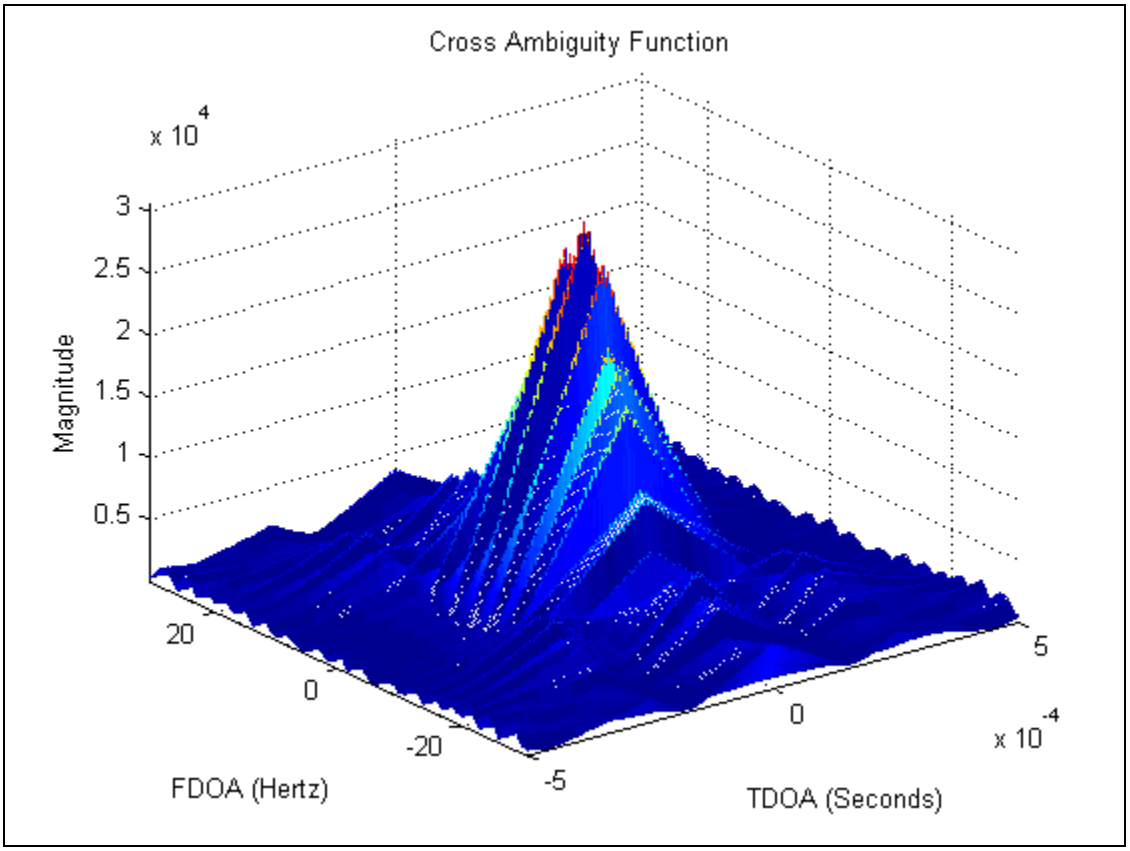

Figure 55 Waveform #4 Example CAF with $SNR = 100$ dB.

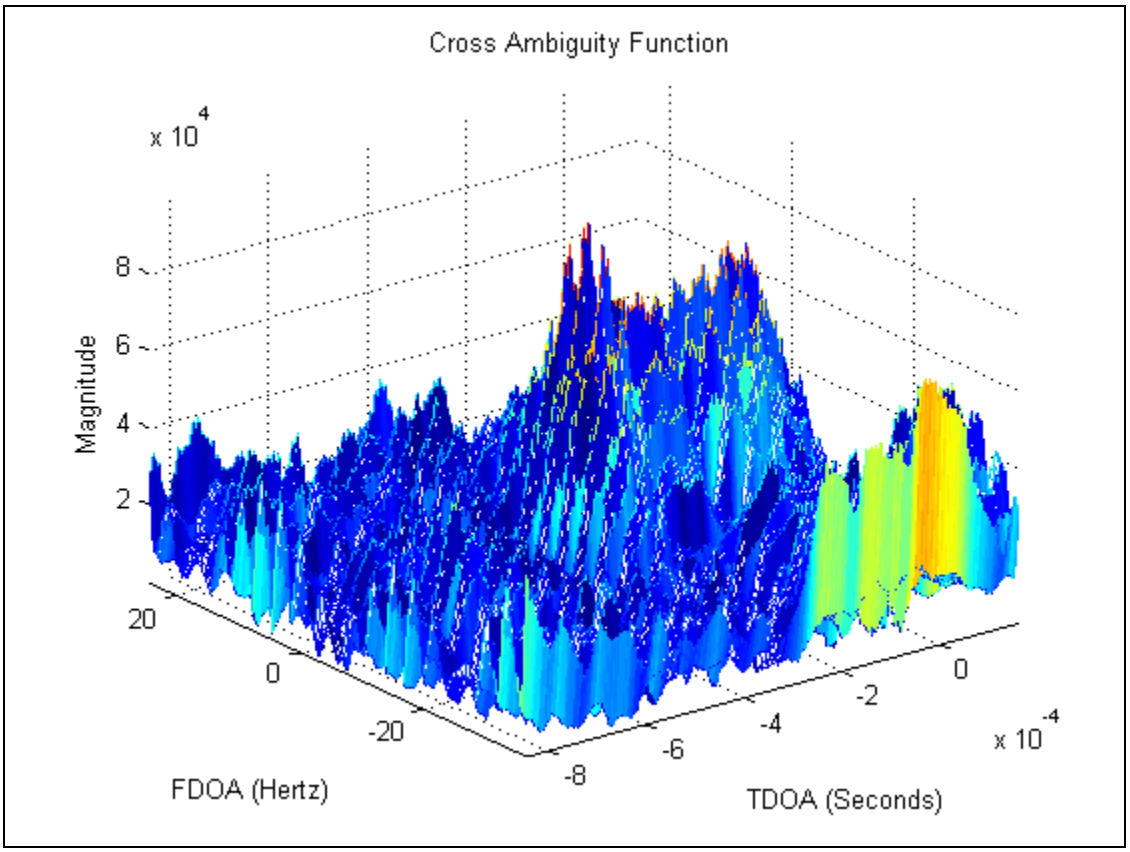

Figure 56 Waveform #4 Example CAF with $SNR = 0$ dB.

Inspecting the curves at high SNR in Figure 53 in more detail, one can note three things. First, the simulation results for the filtered waveform #1 fall directly on top of the lines for expected of theoretical performance and filtered waveforms #2 and #4, matching theory. Second, filtered waveform #3, which has a higher rms radian frequency β than the other three filtered waveforms, also has a smaller standard deviation for TOA. Finally, all four of the unfiltered waveforms have about the same standard deviation because β for these waveforms is really limited by the collector bandwidth.

Likewise, examination of the curves at high SNR in Figure 54 in more detail shows, first, that the standard deviations for FOA σ_{FOA} for both the filtered and unfiltered versions of waveform #1 fall directly on the curve for theoretical performance. Second, filtering has no effect on σ_{FOA} for a given waveform (i.e.,

the curve of the performance for a filtered waveform falls directly on top of the respective unfiltered waveform). Third, it shows that shaping the time domain profile of the waveform does indeed have a significant impact on the standard deviation.

2. Shaped-Chip Waveforms

The next comparison is between the filtered waveform #1, the reference signal, and sinc-shaped chipping of the carrier at various chip rates and corresponding bandwidth. As the chip rate increases, the bandwidth and rms radian frequency should also increase, thus improving TOA accuracy (i.e., reducing σ_{TOA}).

The standard deviation σ of the TOA and FOA values, respectively, determined from simulations for filtered waveform #1 and unfiltered waveforms #11-16 are plotted in Figure 57 and Figure 58. As can be seen on the right side of these plots once again, at high SNR values ($\geq 20dB$), all the curves either lie on top of the theoretical curve or are parallel with it, and at low SNR values ($\leq 10dB$), the curves flatten out with a very poor σ indicating the spurious detections throughout the CAF space. In addition, all the curves in Figure 58 lie on top of each other, as expected, because the waveforms are all of the same duration and have basically constant power over this duration,ignoring the ripples and nulls. Finally, the theoretical value of σ_{TOA} for each of the waveforms #11-16 at 25 dB SNR are shown with the dark bullseyes. These were computed using the values of β from the third column in Table 3.

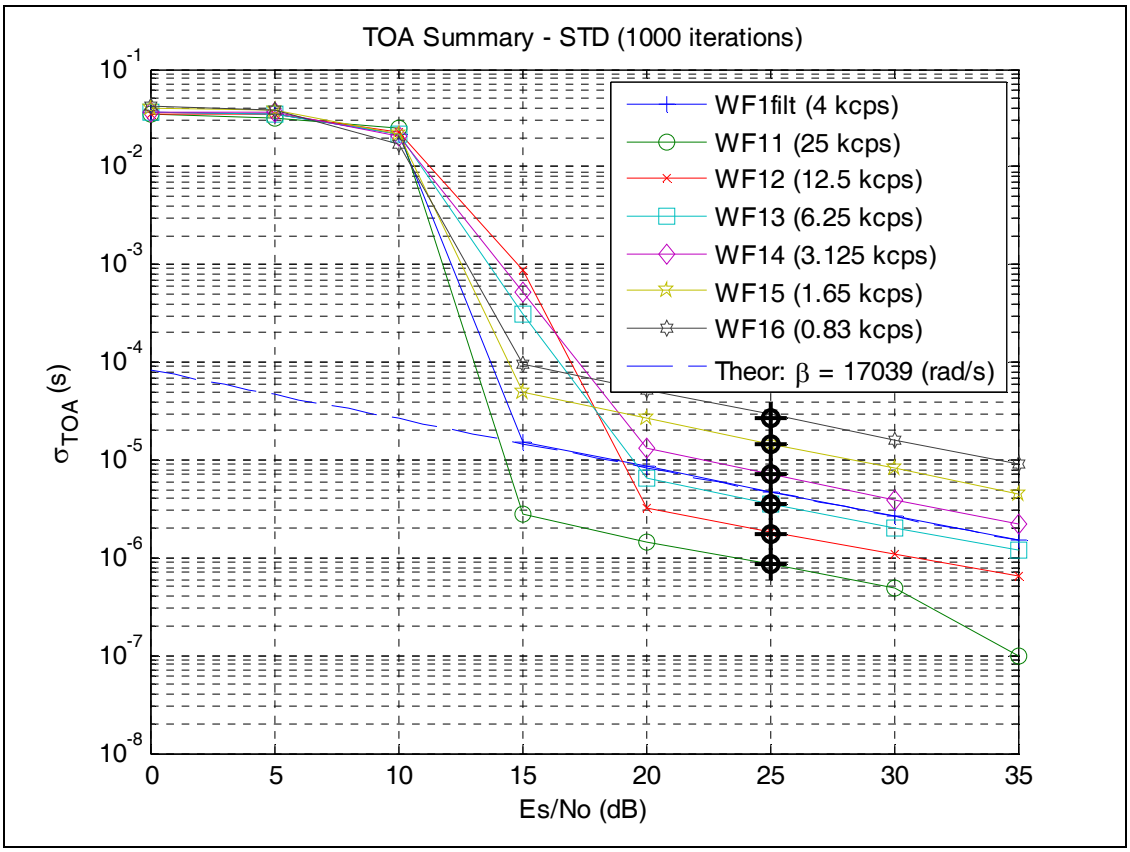

Figure 57 TOA Accuracies – Reference Waveform vs. Shaped Chips.

Examining the curves at high SNR in Figure 57, one can note that each doubling of the chip rate (i.e., bandwidth) causes a 50% reduction in σ_{TOA} for the same SNR (i.e., one can double the accuracy of the measurements by doubling the bandwidth without increasing the transmit power). Conversely, one can also note from these plots that for a given bandwidth, one can double the accuracy by increasing transmit power by 6 dB. These two observations are consistent with (4.38).

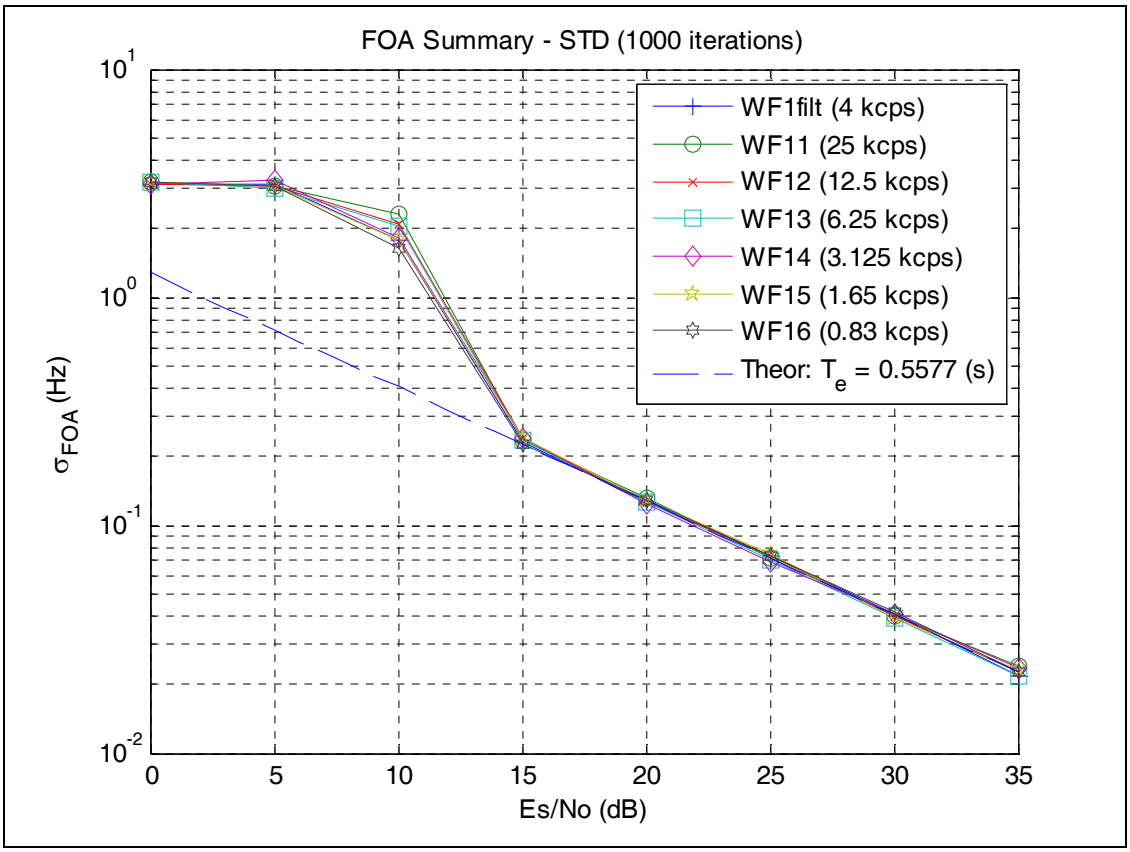

Figure 58 FOA Accuracies – Reference Waveform vs Shaped Chips.

3. Bandwidth Constrained Waveforms

The final comparison is between the various bandwidth constrained waveforms of $B_{nn} = 8$ kHz and with the reference waveform (filtered waveform #1) at various chip rates. The bandwidth constrained waveforms, which are shown in Figure 18, consist of filtered waveforms #1-4 at the chip rate $R_c = 4$ kcps and waveform #17, which is modulated with sinc-shaped pulses, is chipped at $R_c = 8.3$ kcps to give a similar null-to-null bandwidth B_{nn}. The reference waveform chipped at higher rates provides a reference by which to compare the various waveforms.

The standard deviation σ of the TOA and FOA values, respectively, determined from simulations for these waveforms are plotted in Figure 59 and Figure 60. As can be seen on the right side of these plots once again, at high

SNR values ($\geq 20dB$), all the curves either lie on top of the theoretical curve for the reference waveform or are parallel with it. In addition, the theoretical values of σ_{TOA} for the reference waveform, #1F, at each of the chipping rates and at 25 dB SNR are shown with the dark bullseyes. Note how well they match the results of the simulation for the various bandwidths.

In the region of high SNR values ($\geq 20dB$), Figure 59 makes evident once again that doubling the chip rate of the reference waveform causes a 50% reduction in σ_{TOA} for a given SNR. Likewise, transmitting a signal with 6 dB more power would also cause a 50% reduction in σ_{TOA} for a given waveform at a given power. However, one could also achieve almost a 50% reduction in σ_{TOA} from the reference waveform, without increased energy or bandwidth, by reshaping it to waveform #3 (filtered) or #17 (unfiltered or filtered). However, this is at a cost of increased peak power.

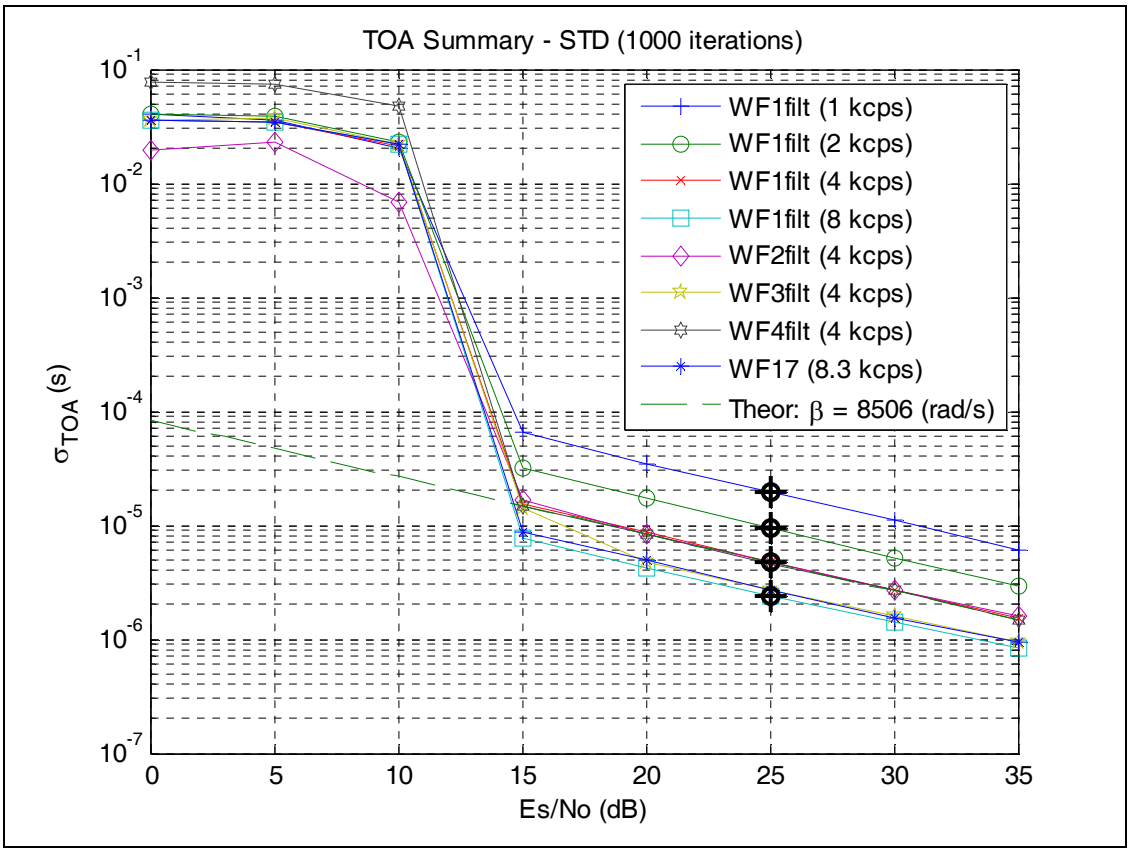

Figure 59 TOA Accuracies – Summary of Alternatives.

Comparing the waveforms for FOA performance (Figure 60) shows that changing the bandwidth has no effect on the resulting standard deviation σ_{FOA}; however, shortening, lengthening, or otherwise changing the power profile over time does affect σ_{FOA}.

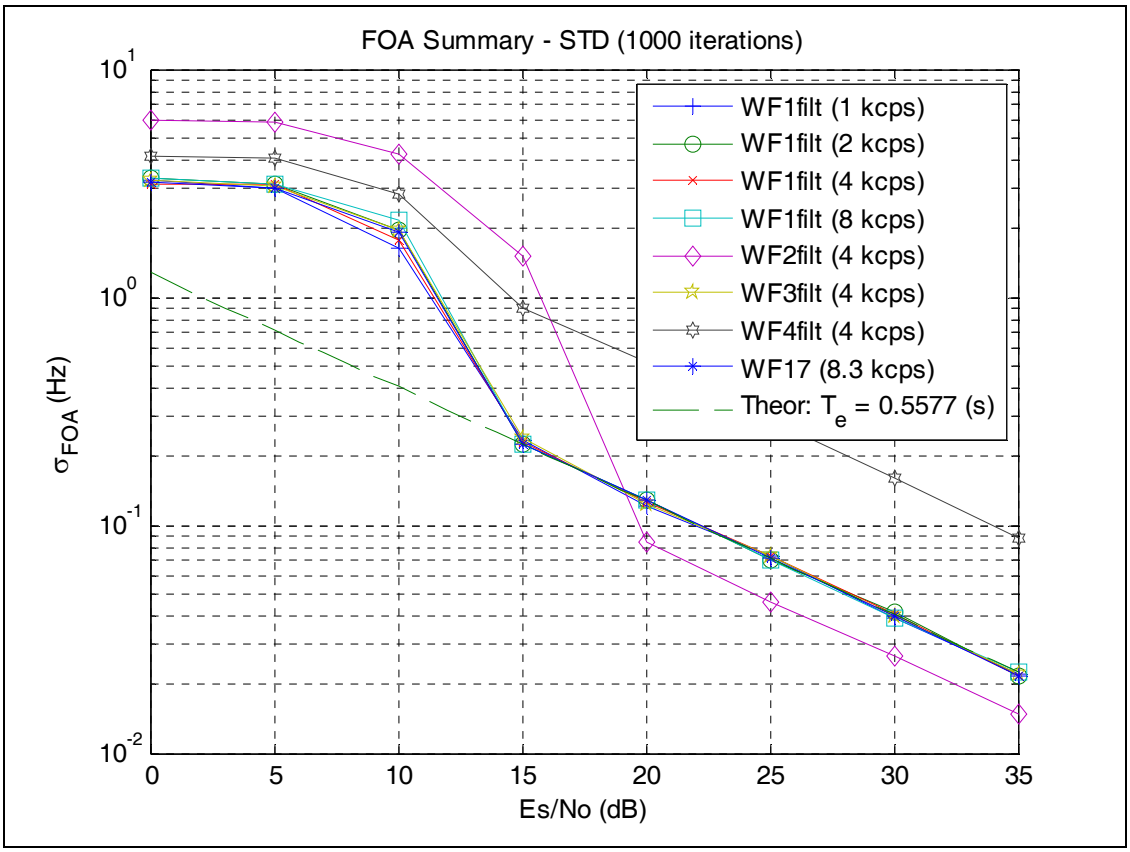

Figure 60 FOA Accuracies – Summary of Alternatives.

C. SUMMARY OF FINDINGS

This effort examined the efficacy of a waveform to support geolocation. It specifically investigated how well a waveform could support identifying the location of a single emission in the presence of AWGN given that the emitter is simultaneously visible to multiple coherent collectors. The analysis also assumes that

- the emitter is transmitting isotropically,
- no multipath or atmospheric effects exist,
- the entire channel is linear (including amplifiers),
- the coherent collectors have perfect knowledge of time and their own location,
- the collection geometry is static,

- the transmitted signal is modulated by a completely known chipping sequence,
- the collectors have a copy of the signal being transmitted, and
- no data are being modulated onto the emission.

This thesis identified the ability of a waveform to support accurate estimation of TOA and FOA as the figures of merit to support geolocation of an emission. The particular metric is the standard deviation σ of these estimates. Any attempt to define the waveform accuracy by using a figure of merit involving physical location requires knowledge of the collectors and collection geometry. The three main parameters affecting σ_{TOA} and σ_{FOA} are E_s/N_0, bandwidth, and signal duration. These parameters are limited not just by physical constraints on transmit power and the occupied bandwidth, but also by acceptable visibility by an adversary (e.g., low probability of intercept or detection).

Equations show that the probability of correctly detecting the signal P_d along with the probability of a false alarm P_{FA} ("detecting" a signal that is not really there) are a function only of the signal power, noise power spectral density, duration of the signal, and detection threshold, but are otherwise independent of the waveform characteristics. Probability of detection P_d, probability of false alarm P_{FA}, and detection threshold are related. For fixed signal power to noise power ratio (SNR), increasing the detection threshold decreases the probability of false alarm. However, for fixed SNR, increasing the detection threshold will also decrease the probability of detection.

On the other hand, the "shape" of the waveform does have an effect on σ as stated by Stein [3]. For a given E_s/N_0, occupied bandwidth (e.g., null-to-null bandwidth B_{nn}), and total signal duration, manipulating the PSD and the amplitude profile (vs. time) of the signal affect σ_{TOA} and σ_{FOA}, respectively. This shaping can be performed by filtering (temporal or spectral domain) the signal, synthesizing by adding up component signals of the waveform or otherwise

modulating the signal, or by shaping the chipping pulses. However this shaping is accomplished, "pushing" the waveform energy from the center to the extremes increases the rms value of that parameter. For example, generating a waveform that has a higher PSD near the band edges than at the center of the band will provide a higher rms bandwidth signal than one, which has flat PSD, resulting in a smaller value for σ_{TOA} and improved location estimation. Likewise, generating a waveform in which the signal amplitude is greater towards the beginning and end than in the middle of the signal results in an improved (i.e., smaller) σ_{FOA}. One potential cost relative to DSSS[7] of performing this shaping, however, is potentially greater visibility by an adversary (e.g., shaping the PSD implies that the signal may be more visible at those accentuated frequencies). Another potential cost is forcing the system to deal with a non-constant envelope waveform which can be a challenge in power constrained systems because they typically operate their power amplifiers at or near saturation to improve their power added efficiency (PAE), although techniques are being developed to help alleviate this constraint [18].

D. FUTURE WORK

Future work is needed to better define the real-world performance one might expect to see in a fielded system. The first of these would be to run simulations in which the length of the chip sequence matches the m-sequence to find optimal performance. Because these were not matched, the chips appear random, but they do not experience the noise-like property of having a autocorrelation peak only when the two signals have no lag (i.e., no minor correlation peaks). The existing model and routines support this; however new mat-files need to be created for chip sequence (i.e, mls65535a.mat) using mls_gen.m and the corresponding shaped chip waveforms using gen_sinc.m.

[7] Typically DSSS is PSK modulated.

A second area is to extend the analysis and simulation into a dynamic collection geometry with at least velocity, but also limited acceleration, which would affect waveform length (or at least coherent processing length).

A third area of investigation is to identify the effect from non-AWGN interference (both colored noise and other emitters). This analysis should be supplemented by simulation.

A fourth area is to model propagation effects. These effects include multipath fading and atmospheric distortion, but they may also include the effects of nonlinear channels (e.g., nonlinear amplifiers). Although the former would be scenario dependent, the latter would not and could be useful in system design to better understand and specify linearity tolerances.

A fifth area is to evaluate different waveforms balanced by the constraint of hardware complexity. For example, an infinitely wide bandwidth signal would give optimal σ_{TOA} performance, but it is also not realizable. Tradeoffs should be evaluated to identify features and limitations in a waveform that greatly simplify processing without significantly degrading performance.

A sixth area is to investigate vulnerability of specific waveforms. This, however, becomes very scenario specific.

Finally, another area would be investigating the feasibility of using shaped noise waveforms. Instead of shaping BPSK waveforms as was done for waveforms #1-4, spectrally and temporally shaping a noise burst, although not deterministic, may lead to some interesting concepts and conclusion.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

This Appendix contains all the MATLAB® functions and scripts developed for this thesis. MATLAB® Version 2008a and 2008b were used in this thesis.

A. MATLAB CODE: SCRIPT_TOP_LEVEL_SIMULATE VARIOUS_WFS.M

```
% ********************************************************************
% script_top_level_simulate various_WFs.M;
% This code establishes the simulation parameters and calls functions
% to perform the simulation and plot resulting data.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 14 May 2009
%
% ********************************************************************

 % unfiltered waveforms 1-4

clear all
WF1It1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1It1000Rs4000_stat_summary_array = stat_summary_array;
save WF1It1000Rs4000_stat_summary_array WF1It1000Rs4000_stat_summary_array

clear all
WF2It1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF2It1000Rs4000_stat_summary_array = stat_summary_array;
save WF2It1000Rs4000_stat_summary_array WF2It1000Rs4000_stat_summary_array

clear all
WF3It1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF3It1000Rs4000_stat_summary_array = stat_summary_array;
save WF3It1000Rs4000_stat_summary_array WF3It1000Rs4000_stat_summary_array

clear all
WF4It1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF4It1000Rs4000_stat_summary_array = stat_summary_array;
save WF4It1000Rs4000_stat_summary_array WF4It1000Rs4000_stat_summary_array
```

```
% filtered waveforms 1-4

clear all
WF1filtIt1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1filtIt1000Rs4000_stat_summary_array = stat_summary_array;
save WF1filtIt1000Rs4000_stat_summary_array WF1filtIt1000Rs4000_stat_summary_array

clear all
WF2filtIt1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF2filtIt1000Rs4000_stat_summary_array = stat_summary_array;
save WF2filtIt1000Rs4000_stat_summary_array WF2filtIt1000Rs4000_stat_summary_array

clear all
WF3filtIt1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF3filtIt1000Rs4000_stat_summary_array = stat_summary_array;
save WF3filtIt1000Rs4000_stat_summary_array WF3filtIt1000Rs4000_stat_summary_array

clear all
WF4filtIt1000Rs4000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF4filtIt1000Rs4000_stat_summary_array = stat_summary_array;
save WF4filtIt1000Rs4000_stat_summary_array WF4filtIt1000Rs4000_stat_summary_array

% filtered waveform 1 at other Rs values

clear all
WF1filtIt1000Rs1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1filtIt1000Rs1000_stat_summary_array = stat_summary_array;
save WF1filtIt1000Rs1000_stat_summary_array WF1filtIt1000Rs1000_stat_summary_array

clear all
WF1filtIt1000Rs2000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1filtIt1000Rs2000_stat_summary_array = stat_summary_array;
save WF1filtIt1000Rs2000_stat_summary_array WF1filtIt1000Rs2000_stat_summary_array
```

```
clear all
WF1filtIt1000Rs8000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1filtIt1000Rs8000_stat_summary_array = stat_summary_array;
save WF1filtIt1000Rs8000_stat_summary_array WF1filtIt1000Rs8000_stat_summary_array


% unfiltered waveform 1 at other Rs values

clear all
WF1It1000Rs1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1It1000Rs1000_stat_summary_array = stat_summary_array;
save WF1It1000Rs1000_stat_summary_array WF1It1000Rs1000_stat_summary_array

clear all
WF1It1000Rs2000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1It1000Rs2000_stat_summary_array = stat_summary_array;
save WF1It1000Rs2000_stat_summary_array WF1It1000Rs2000_stat_summary_array

clear all
WF1It1000Rs8000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF1It1000Rs8000_stat_summary_array = stat_summary_array;
save WF1It1000Rs8000_stat_summary_array WF1It1000Rs8000_stat_summary_array


% canned waveforms 11-16 --- shaped chips

clear all
WF11It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF11It1000_stat_summary_array = stat_summary_array;
save WF11It1000_stat_summary_array WF11It1000_stat_summary_array

clear all
WF12It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF12It1000_stat_summary_array = stat_summary_array;
save WF12It1000_stat_summary_array WF12It1000_stat_summary_array
```

```
clear all
WF13It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF13It1000_stat_summary_array = stat_summary_array;
save WF13It1000_stat_summary_array WF13It1000_stat_summary_array

clear all
WF14It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF14It1000_stat_summary_array = stat_summary_array;
save WF14It1000_stat_summary_array WF14It1000_stat_summary_array

clear all
WF15It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF15It1000_stat_summary_array = stat_summary_array;
save WF15It1000_stat_summary_array WF15It1000_stat_summary_array

clear all
WF16It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF16It1000_stat_summary_array = stat_summary_array;
save WF16It1000_stat_summary_array WF16It1000_stat_summary_array


% canned waveform 17 (unfiltered and filtered) --- shaped chips

clear all
WF17It1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF17It1000_stat_summary_array = stat_summary_array;
save WF17It1000_stat_summary_array WF17It1000_stat_summary_array

clear all
WF17filtIt1000_config_file
save config_file
main_simulation %run simulation using config parameters from above
display_toa_foa_v_snr_and_prep_data
WF17filtIt1000_stat_summary_array = stat_summary_array;
save WF17filtIt1000_stat_summary_array WF17filtIt1000_stat_summary_array
```

B. MATLAB CODE: SCRIPT_DISPLAY_TOA_FOA_V_SNR_ACROSS_RUNS_MRKRS.M

```
% ********************************************************************
% script_display_toa_foa_v_snr_across_runs_mrkrs.m;
% Calls functions to generate final plots.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 14 May 2009
%
% ********************************************************************

clear all
close all
clc

load WF1It1000Rs4000_stat_summary_array
load WF2It1000Rs4000_stat_summary_array
load WF3It1000Rs4000_stat_summary_array
load WF4It1000Rs4000_stat_summary_array

load WF1filtIt1000Rs4000_stat_summary_array
load WF2filtIt1000Rs4000_stat_summary_array
load WF3filtIt1000Rs4000_stat_summary_array
load WF4filtIt1000Rs4000_stat_summary_array

load WF1filtIt1000Rs1000_stat_summary_array
load WF1filtIt1000Rs2000_stat_summary_array
load WF1filtIt1000Rs8000_stat_summary_array

load WF11It1000_stat_summary_array
load WF12It1000_stat_summary_array
load WF13It1000_stat_summary_array
load WF14It1000_stat_summary_array
load WF15It1000_stat_summary_array
load WF16It1000_stat_summary_array

load WF17It1000_stat_summary_array

[no_rows, no_cols] = size(WF1It1000Rs4000_stat_summary_array)

[no_rows, no_cols] = size(WF1It1000Rs4000_stat_summary_array)

%% Calculate theoretical TDOA and FDOA

SNR_idx = 0;
for SNR_dB = WF1It1000Rs4000_stat_summary_array(1,1):
WF1It1000Rs4000_stat_summary_array(no_rows,1)
   SNR_idx = SNR_idx+1;

   SNR = 10^(SNR_dB/10);
   gamma = 2* SNR;
```

```
    T = 0.3; %sec i.e., signal duration
    B = 3.3; %Hz  i.e., 1/T

    BTg = B*T*gamma
    sqrt_BTg = sqrt(BTg);

    % compute theoretical for Filtered WF#1 from computed values
    Beta = 8506
    Te = 0.5577

    sigma_tdoa = (1/Beta)/sqrt_BTg;
    sigma_fdoa = (1/Te)/sqrt_BTg;

    theor_stat_summary_array(SNR_idx,1) = SNR_dB;
    theor_stat_summary_array(SNR_idx,2) = 0; %mean theoretical tdoa = 0
    theor_stat_summary_array(SNR_idx,3) = sigma_tdoa;
    theor_stat_summary_array(SNR_idx,4) = 0; %mean theoretical fdoa = 0
    theor_stat_summary_array(SNR_idx,5) = sigma_fdoa;
end

theor_stat_summary_array

%% Plot WF1-4 & filtered-WF1-4 at 4kcps, fs=100kSps, fc=20kHz

figure;
%subplot(2,1,1);
semilogy( ...
    WF1It1000Rs4000_stat_summary_array(:,1),      WF1It1000Rs4000_stat_summary_array(:,3),
':+', ...
    WF2It1000Rs4000_stat_summary_array(:,1),      WF2It1000Rs4000_stat_summary_array(:,3),
':o', ...
    WF3It1000Rs4000_stat_summary_array(:,1),      WF3It1000Rs4000_stat_summary_array(:,3),
':x', ...
    WF4It1000Rs4000_stat_summary_array(:,1),      WF4It1000Rs4000_stat_summary_array(:,3),
':s', ...
    WF1filtIt1000Rs4000_stat_summary_array(:,1),
WF1filtIt1000Rs4000_stat_summary_array(:,3), '-+', ...
    WF2filtIt1000Rs4000_stat_summary_array(:,1),
WF2filtIt1000Rs4000_stat_summary_array(:,3), '-o', ...
    WF3filtIt1000Rs4000_stat_summary_array(:,1),
WF3filtIt1000Rs4000_stat_summary_array(:,3), '-x', ...
    WF4filtIt1000Rs4000_stat_summary_array(:,1),
WF4filtIt1000Rs4000_stat_summary_array(:,3), '-s', ...
    theor_stat_summary_array(:,1), theor_stat_summary_array(:,3), '--'    );
title('TOA Summary - STD (1000 iterations)'); xlabel('Es/No (dB)'); ylabel('\sigma_{TOA} (s)');
legend('WF1 (4 kcps)', 'WF2 (4 kcps)', 'WF3 (4 kcps)', 'WF4 (4 kcps)', ...
    'WF1filt (4 kcps)', 'WF2filt (4 kcps)', 'WF3filt (4 kcps)', 'WF4filt (4 kcps)', ...
    ['Theor: \beta = ', num2str(Beta), ' (rad/s)']);
grid on


figure;
%subplot(2,1,1);
```

```
semilogy( ...
    WF1It1000Rs4000_stat_summary_array(:,1),      WF1It1000Rs4000_stat_summary_array(:,5),
':+', ...
    WF2It1000Rs4000_stat_summary_array(:,1),      WF2It1000Rs4000_stat_summary_array(:,5),
':o', ...
    WF3It1000Rs4000_stat_summary_array(:,1),      WF3It1000Rs4000_stat_summary_array(:,5),
':x', ...
    WF4It1000Rs4000_stat_summary_array(:,1),      WF4It1000Rs4000_stat_summary_array(:,5),
':s', ...
    WF1filtIt1000Rs4000_stat_summary_array(:,1),
WF1filtIt1000Rs4000_stat_summary_array(:,5), '-+', ...
    WF2filtIt1000Rs4000_stat_summary_array(:,1),
WF2filtIt1000Rs4000_stat_summary_array(:,5), '-o', ...
    WF3filtIt1000Rs4000_stat_summary_array(:,1),
WF3filtIt1000Rs4000_stat_summary_array(:,5), '-x', ...
    WF4filtIt1000Rs4000_stat_summary_array(:,1),
WF4filtIt1000Rs4000_stat_summary_array(:,5), '-s', ...
    theor_stat_summary_array(:,1), theor_stat_summary_array(:,5), '--'    );
title('FOA Summary - STD (1000 iterations)'); xlabel('Es/No (dB)'); ylabel('\sigma_{FOA} (Hz)');
legend('WF1 (4 kcps)', 'WF2 (4 kcps)', 'WF3 (4 kcps)', 'WF4 (4 kcps)', ...
    'WF1filt (4 kcps)', 'WF2filt (4 kcps)', 'WF3filt (4 kcps)', 'WF4filt (4 kcps)', ...
    ['Theor: T_e = ', num2str(Te), ' (s)']);
grid on

%% Plot WF1 at 1,2,8kcps & filtered-WF1-4 at 4kcps, fs=100kSps, fc=20kHz,
%% also overlay WF17

figure;
%subplot(2,1,1);
semilogy( ...
    WF1filtIt1000Rs1000_stat_summary_array(:,1),
WF1filtIt1000Rs1000_stat_summary_array(:,3), '-+', ...
    WF1filtIt1000Rs2000_stat_summary_array(:,1),
WF1filtIt1000Rs2000_stat_summary_array(:,3), '-o', ...
    WF1filtIt1000Rs4000_stat_summary_array(:,1),
WF1filtIt1000Rs4000_stat_summary_array(:,3), '-x', ...
    WF1filtIt1000Rs8000_stat_summary_array(:,1),
WF1filtIt1000Rs8000_stat_summary_array(:,3), '-s', ...
    WF2filtIt1000Rs4000_stat_summary_array(:,1),
WF2filtIt1000Rs4000_stat_summary_array(:,3), '-d', ...
    WF3filtIt1000Rs4000_stat_summary_array(:,1),
WF3filtIt1000Rs4000_stat_summary_array(:,3), '-p', ...
    WF4filtIt1000Rs4000_stat_summary_array(:,1),
WF4filtIt1000Rs4000_stat_summary_array(:,3), '-h', ...
    WF17It1000_stat_summary_array(:,1), WF17It1000_stat_summary_array(:,3), '-*', ...
    theor_stat_summary_array(:,1), theor_stat_summary_array(:,3), '--'    );
title('TOA Summary - STD (1000 iterations)'); xlabel('Es/No (dB)'); ylabel('\sigma_{TOA} (s)');
legend('WF1filt (1 kcps)', 'WF1filt (2 kcps)', ...
    'WF1filt (4 kcps)', 'WF1filt (8 kcps)', ...
    'WF2filt (4 kcps)', 'WF3filt (4 kcps)', ...
    'WF4filt (4 kcps)', 'WF17 (8.3 kcps)', ...
    ['Theor: \beta = ', num2str(Beta), ' (rad/s)']);
grid on
```

```
figure;
%subplot(2,1,1);
semilogy( ...
   WF1filtIt1000Rs1000_stat_summary_array(:,1),
WF1filtIt1000Rs1000_stat_summary_array(:,5), '-+', ...
   WF1filtIt1000Rs2000_stat_summary_array(:,1),
WF1filtIt1000Rs2000_stat_summary_array(:,5), '-o', ...
   WF1filtIt1000Rs4000_stat_summary_array(:,1),
WF1filtIt1000Rs4000_stat_summary_array(:,5), '-x', ...
   WF1filtIt1000Rs8000_stat_summary_array(:,1),
WF1filtIt1000Rs8000_stat_summary_array(:,5), '-s', ...
   WF2filtIt1000Rs4000_stat_summary_array(:,1),
WF2filtIt1000Rs4000_stat_summary_array(:,5), '-d', ...
   WF3filtIt1000Rs4000_stat_summary_array(:,1),
WF3filtIt1000Rs4000_stat_summary_array(:,5), '-p', ...
   WF4filtIt1000Rs4000_stat_summary_array(:,1),
WF4filtIt1000Rs4000_stat_summary_array(:,5), '-h', ...
   WF17It1000_stat_summary_array(:,1), WF17It1000_stat_summary_array(:,5), '-*', ...
   theor_stat_summary_array(:,1), theor_stat_summary_array(:,5), '--'    );
title('FOA Summary - STD (1000 iterations)'); xlabel('Es/No (dB)'); ylabel('\sigma_{FOA} (Hz)');
legend('WF1filt (1 kcps)', 'WF1filt (2 kcps)', ...
   'WF1filt (4 kcps)', 'WF1filt (8 kcps)', ...
   'WF2filt (4 kcps)', 'WF3filt (4 kcps)', ...
   'WF4filt (4 kcps)', 'WF17 (8.3 kcps)', ...
   ['Theor: T_e = ', num2str(Te), ' (s)']);
grid on


%% Plot WF11-16 & filtered-WF1-4 at 4kcps, fs=100kSps, fc=20kHz

figure;
%subplot(2,1,1);
semilogy( ...
   WF1filtIt1000Rs4000_stat_summary_array(:,1),
WF1filtIt1000Rs4000_stat_summary_array(:,3), '-+', ...
   WF11It1000_stat_summary_array(:,1), WF11It1000_stat_summary_array(:,3), '-o', ...
   WF12It1000_stat_summary_array(:,1), WF12It1000_stat_summary_array(:,3), '-x', ...
   WF13It1000_stat_summary_array(:,1), WF13It1000_stat_summary_array(:,3), '-s', ...
   WF14It1000_stat_summary_array(:,1), WF14It1000_stat_summary_array(:,3), '-d', ...
   WF15It1000_stat_summary_array(:,1), WF15It1000_stat_summary_array(:,3), '-p', ...
   WF16It1000_stat_summary_array(:,1), WF16It1000_stat_summary_array(:,3), '-h', ...
   theor_stat_summary_array(:,1), theor_stat_summary_array(:,3), '--'    );
title('TOA Summary - STD (1000 iterations)'); xlabel('Es/No (dB)'); ylabel('\sigma_{TOA} (s)');
legend('WF1filt (4 kcps)', 'WF11 (25 kcps)', 'WF12 (12.5 kcps)', ...
   'WF13 (6.25 kcps)', 'WF14 (3.125 kcps)', 'WF15 (1.65 kcps)', ...
   'WF16 (0.83 kcps)', ...
   ['Theor: \beta = ', num2str(Beta), ' (rad/s)']);
grid on


figure;
%subplot(2,1,1);
```

```
semilogy( ...
   WF1filtIt1000Rs4000_stat_summary_array(:,1),
WF1filtIt1000Rs4000_stat_summary_array(:,5), '-+', ...
   WF11It1000_stat_summary_array(:,1), WF11It1000_stat_summary_array(:,5), '-o', ...
   WF12It1000_stat_summary_array(:,1), WF12It1000_stat_summary_array(:,5), '-x', ...
   WF13It1000_stat_summary_array(:,1), WF13It1000_stat_summary_array(:,5), '-s', ...
   WF14It1000_stat_summary_array(:,1), WF14It1000_stat_summary_array(:,5), '-d', ...
   WF15It1000_stat_summary_array(:,1), WF15It1000_stat_summary_array(:,5), '-p', ...
   WF16It1000_stat_summary_array(:,1), WF16It1000_stat_summary_array(:,5), '-h', ...
   theor_stat_summary_array(:,1), theor_stat_summary_array(:,5), '--'    );
title('FOA Summary - STD (1000 iterations)'); xlabel('Es/No (dB)'); ylabel('\sigma_{FOA} (Hz)');
legend('WF1filt (4 kcps)', 'WF11 (25 kcps)', 'WF12 (12.5 kcps)', ...
   'WF13 (6.25 kcps)', 'WF14 (3.125 kcps)', 'WF15 (1.65 kcps)', ...
   'WF16 (0.83 kcps)', ...
   ['Theor: T_e = ', num2str(Te), ' (s)']);
grid on
```

C. MATLAB CODE: SCRIPT_PLOT_WFS.M

```
% ************************************************************************
% script_plot_WFs.M;
% This code set the variables and and calls functions
% to plot a particular waveform.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 14 May 2009
%
% ************************************************************************

% signal parameters
wf_type=4; % 1:const env, const psd; 2:gap in time; 3:gap in psd; 4:shortened pulse
filter_outside_bnn=0; %limit signal, if generated (i.s., WF1-4), to within Bnn
process_detections=1; %set to '1' to CAF and get estimates of TOA and FOA

f0 = 20000;  %carrier frequency
%f0 = 25000;  %carrier frequency - f0 of canned WF is fs/4
fs = 100000; %sample frequency
Rsym=4000; %2000 %10000;   %symbol rate

pad_length = 1024; %no. of zeros to add onto each side of S1
N = 32768-2*pad_length  %length(samples) of burst; CAF alg. prefers N=2^k

%--- SNR (Ec_No) of 2.6 (4.15 dB) should give BER .01 for BPSK
Es_No_dB_min = 10
Es_No_dB_step = 500;
Es_No_dB_max = 100

no_noise_iterations = 1000;
no_noise_iterations = 1 %200%0; %500 %40 %250;

%monitor and debug setting
verbose=0; %set to zero to stop sending debug info to MATLAB window
verbose_wf_gen=1; %enable plots and sending debug info to MATLAB window
```

```
verbose_plot_wf=1; %enable plots of waveform and calculate rms BW
enable_BER_test=0; %set to 1 to enable running of BER test function
limit_CAF_to_search_freq_only=0;  % remove ambiguity in CAF (find FDOA only) %Warning: assumes tau =0
limit_CAF_to_search_time_only=0;  % remove ambiguity in CAF (find TDOA only)

%geometry
Pc1 = [0 0 500]; %use only z position (i.e., leave x&y=0)
Vc1 = [0 0 0];
Pc2 = Pc1;
Vc2 = [0 0 0];
Pe = [0 0 0];
Ve = [0 0 0];

tau = 1.25e-7 % time offset step used to dither sampling relative to signal
c = 2.997925e8; % Speed of light in m/s

% dither position to remove clock sync between runs
pos_offset_min = 0; % in meters
pos_offset_step = c*tau; % in meters
pos_offset_max = 0; %9*c*tau; % in meters  %run ten iterations

save config_file
main_simulation
```

D. MATLAB CODE: DISPLAY_TOA_FOA_V_SNR_AND_PREP_DATA.M

```
% ************************************************************************
% display_toa_foa_v_snr_and_prep_data.m;
% This code reads the arrays produced by the simulation code,
% generates the statistics, and plots data from a singlesim run.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 15 May 2009
%
% ************************************************************************

   fprintf('\n *** Statistical Summary *** \n');

   for stat_index=1:Es_No_step_no
      offset=(stat_index-1)*no_noise_iterations*pos_offset_index;
      fprintf('\nEs/No = %f dB (%d samples)\n', toa_est(offset+1,2),
no_noise_iterations*pos_offset_index);
      fprintf('- TOA estimates: mean=%f  std=%f \n', ...
         mean(toa_est(offset+1:offset+no_noise_iterations*pos_offset_index,1)), ...
         std(toa_est(offset+1:offset+no_noise_iterations*pos_offset_index,1)));
      fprintf('- FOA estimates: mean=%f  std=%f \n', ...
         mean(foa_est(offset+1:offset+no_noise_iterations*pos_offset_index,1)), ...
         std(foa_est(offset+1:offset+no_noise_iterations*pos_offset_index,1)));

      stat_summary_array(stat_index,1)=toa_est(offset+1,2); %Es_No
```

```
stat_summary_array(stat_index,2)=mean(toa_est(offset+1:offset+no_noise_iterations*pos_offset
_index,1)); %mean toa

stat_summary_array(stat_index,3)=std(toa_est(offset+1:offset+no_noise_iterations*pos_offset_in
dex,1)); %std toa

stat_summary_array(stat_index,4)=mean(foa_est(offset+1:offset+no_noise_iterations*pos_offset
_index,1)); %mean foa

stat_summary_array(stat_index,5)=std(foa_est(offset+1:offset+no_noise_iterations*pos_offset_in
dex,1)); %std foa
   end

   %%
%    % Template to save data
%    WFxxItxx_stat_summary_array = stat_summary_array
%    save WFxxItxx_stat_summary_array WFxxItxx_stat_summary_array

%  %example:
%   WF3filtIt100Rs4000_stat_summary_array = stat_summary_array
%   save WF3filtIt100Rs4000_stat_summary_array WF3filtIt100Rs4000_stat_summary_array

   %%

   figure;
   subplot(2,1,1);
   plot(stat_summary_array(:,1),          stat_summary_array(:,2),          stat_summary_array(:,1),
stat_summary_array(:,3));
      title('TOA Summary'); xlabel('Es/No (dB)'); ylabel('TDOA (s)'); legend('Mean', 'Standard
Deviation');
   subplot(2,1,2);
   plot(stat_summary_array(:,1),          stat_summary_array(:,4),          stat_summary_array(:,1),
stat_summary_array(:,5));
      title('FOA Summary'); xlabel('Es/No (dB)'); ylabel('FDOA (Hz)'); legend('Mean', 'Standard
Deviation');



   figure;
   subplot(2,1,1);
   plot(stat_summary_array(:,1),          stat_summary_array(:,2),          stat_summary_array(:,1),
stat_summary_array(:,3));
      title('TOA Summary'); xlabel('Es/No (dB)'); ylabel('TDOA (s)'); legend('Mean', 'Standard
Deviation');
      xlim([10,35]);
   subplot(2,1,2);
   plot(stat_summary_array(:,1),          stat_summary_array(:,4),          stat_summary_array(:,1),
stat_summary_array(:,5));
      title('FOA Summary'); xlabel('Es/No (dB)'); ylabel('FDOA (Hz)'); legend('Mean', 'Standard
Deviation');
      xlim([10,35]);
```

```
   figure;
   subplot(2,1,1);
   plot(stat_summary_array(:,1), stat_summary_array(:,2), stat_summary_array(:,1),
stat_summary_array(:,3));
      title('TOA Summary'); xlabel('Es/No (dB)'); ylabel('TDOA (s)'); legend('Mean', 'Standard
Deviation');
      xlim([20,35]);
   subplot(2,1,2);
   plot(stat_summary_array(:,1), stat_summary_array(:,4), stat_summary_array(:,1),
stat_summary_array(:,5));
      title('FOA Summary'); xlabel('Es/No (dB)'); ylabel('FDOA (Hz)'); legend('Mean', 'Standard
Deviation');
      xlim([20,35]);


   figure;
   subplot(2,1,1);
   semilogy(stat_summary_array(:,1), stat_summary_array(:,3),'-x');
      title('TOA Summary'); xlabel('Es/No (dB)'); ylabel('TDOA (s)'); legend('Standard Deviation');
      grid on
   subplot(2,1,2);
   semilogy(stat_summary_array(:,1), stat_summary_array(:,5),'-x');
      title('FOA Summary'); xlabel('Es/No (dB)'); ylabel('FDOA (Hz)'); legend('Standard Deviation');
      grid on
```

E. MATLAB CODE: DISPLAY_SCATTER_FOA_TOA.M

```
% ********************************************************************
% display_scatter_foa_toa.m;
% routine to display scatter plots of FOA v. TOA of detections
% for various levels of noise.
% requires: snr_step_no, no_noise_iterations, toa_est, foa_est
%
% Written by: Joe Crnkovich, NRL
% Last modified: 9 April 2009
% ********************************************************************

 close all

for stat_index=1:Es_No_step_no
   %offset=(stat_index-1)*no_noise_iterations;
   offset=(stat_index-1)*no_noise_iterations*pos_offset_index;

   %toa_series=toa_est(offset+1:offset+no_noise_iterations,1);
   %foa_series=foa_est(offset+1:offset+no_noise_iterations,1);
   toa_series=toa_est(offset+1:offset+no_noise_iterations*pos_offset_index,1);
   foa_series=foa_est(offset+1:offset+no_noise_iterations*pos_offset_index,1);

   title_string=['FOA v. TOA Scatterplot (Ec/No=',num2str(toa_est(offset+1,2)),'dB)'];

   figure;
   subplot(2,1,1);
   plot(toa_series, foa_series, 'x');
```

```
   xlabel('Time of Arrival (TOA) (s)'), ylabel('Frequency of Arrival (FOA) (Hz)');
   title(title_string);

%    figure;
   [n,xout] = hist(toa_series)
%    subplot(2,1,1);
   subplot(4,1,3);
   bar(xout,n)
   xlabel('Time of Arrival (TOA) (s)'), ylabel('Occurences');
%    title_string=['Histogram of TOA results (Ec/No=',num2str(toa_est(offset+1,2)),'dB)'];
   title_string=['---------- Histograms of results ----------'];
   title(title_string);

   %figure;
   [n,xout] = hist(foa_series)
%    subplot(2,1,2);
   subplot(4,1,4);
   bar(xout,n)
   xlabel('Frequency of Arrival (FOA) (Hz)'), ylabel('Occurences');
%    title_string=['Histogram of FOA results (Ec/No=',num2str(toa_est(offset+1,2)),'dB)'];
%    title(title_string);


end
```

F. MATLAB CODE: GEN_SINC.M

```
% ************************************************************************
% gen_sinc.m;
% script used to generate canned waveform using sinc-shaped
% chips.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 17 April 2009
% ************************************************************************

clear
close all
clc

samples_per_pulse = 12;  %no of quantizations per pulse
%samples_per_pulse = 50;  %no of quantizations per pulse
no_repeat=1;            %no of times a given sample is repeated


t = linspace(-5,5,10*samples_per_pulse);
y = sinc(t);
stem(t,y);
xlabel('Time (chips)');ylabel('Amplitude');
title_text = ['Sinc Function (', num2str(samples_per_pulse),' Samples per Chip)']
title(title_text)
```

```
load mls65535a

bit_stream = -1 + 2*mls65535a(1:5000); %first 5,000 chips
% bit_stream = zeros(1,200); % test vector
% bit_stream(50) = 1;

%upsample bitstream by inserting zeros
bit_stream = upsample(bit_stream, samples_per_pulse);

modulation=filter(y,1,bit_stream);  % sinc-shaped modulation
bit_stream=filter(ones(1,samples_per_pulse),1,bit_stream); %rect. mod.

%modulation samples are repeated based on no_repeat
modulation = upsample(modulation, no_repeat);
modulation=filter(ones(1,no_repeat),1,modulation);

bit_stream = upsample(bit_stream, no_repeat);
bit_stream=filter(ones(1,no_repeat),1,bit_stream);



% plot baseband modulation signal
figure
subplot(2,1,1)
plot(10*log10(abs(fft(bit_stream)).^2))
%title_text = ['"Squared" Baseband Signal (Samples per pulse=', num2str(samples_per_pulse),')']
title_text = ['num-repeat=', num2str(no_repeat), ...
    '; "Squared" Baseband Signal (Samples per pulse=', num2str(samples_per_pulse),')'];
title(title_text)
ylim([0,80])
grid on
%figure
subplot(2,1,2)
plot(10*log10(abs(fft(modulation)).^2))
%title_text = ['"Shaped" Baseband Signal (Samples per pulse=', num2str(samples_per_pulse),')']
title_text = ['num-repeat=', num2str(no_repeat), ...
    '; "Shaped" Baseband Signal (Samples per pulse=', num2str(samples_per_pulse),')'];
title(title_text)
ylim([0,80])
grid on

% plot modulated signal @ IF = fs/4
n=0:length(modulation)-1;
signal1 = cos(0.25*2*pi*n);

figure
x_axis = [0:length(modulation)-1]/length(modulation);

subplot(2,1,1)
plot(x_axis, 10*log10(abs(fft(signal1.*bit_stream)).^2))
%title_text = ['num-repeat=', num2str(no_repeat),'; "Squared" Signal (Samples per pulse=',
num2str(samples_per_pulse),')']
```

```
title_text = ['Rectangular-shaped Modulation (', num2str(samples_per_pulse),' Samples per
pulse)']
title(title_text)
xlabel('Frequency Normalized to f_s (Hz)');
ylabel('Signal Power (dB)');
xlim([0,0.5]) %fs/2
ylim([0,80])
grid on

subplot(2,1,2)
plot(x_axis, 10*log10(abs(fft(signal1.*modulation)).^2))
%title_text = ['num-repeat=', num2str(no_repeat),'; "Shaped" Baseband Signal (Samples per
pulse=', num2str(samples_per_pulse),')']
title_text = ['Sinc-shaped Modulation (', num2str(samples_per_pulse),' Samples per pulse)']
title(title_text)
xlabel('Frequency Normalized to f_s (Hz)');
ylabel('Signal Power (dB)');
xlim([0,0.5]) %fs/2
ylim([0,80])
grid on

% modulated signal
modulation = signal1.*modulation;


% save sinc_unb_mls65535a modulation samples_per_pulse
```

G. MATLAB CODE: MLS_GEN.M

```
% ************************************************************************
% mls_gen.m;
% script used to generate m-sequence.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 15 May 2009
% ************************************************************************

clear
close all
clc


reg_len = 16;
taps = [1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1];
out_len =70000;  %length of mls code returned (max val 2^reg_len - 1)
seed = [1 0 0];
% ref Dixin table 3.7 for mls codes and respecdtive taps (not all shown)
% 2: [2,1]
% 3: [3,1]
% 4: [4,1]
% 5: [5,2]
% 6: [6,1]
```

```
% 7: [7,1], [7,3] (127)
% 8: [8,4,3,2], [8,6,5,3] (255)
% 9: [9,4], [9,6,4,3] (511)
%10: [10,3], [10,8,3,2] (1023)
%11: [11,1], [11,8,5,2] (2047)
%12: [12,6,4,1], [12,9,3,2] (4095)
%13: [13,4,3,1], [13,10,9,7,5,4] (8191)
%14: [14,12,2,1], [14,13,4,2] (16383)
%15: [15,13,10,9] (32767)
%16: [16,12,3,1] (65535)

reg = [seed, zeros(1, reg_len-length(seed))];
%reg(1:length(seed)) = seed+ seed;

%fprintf('register: [%1d%1d%1d%1d]\n',reg(1), reg(2), reg(3), reg(4))
fprintf('    register: [');
fprintf('%1d',reg);
fprintf(']\n');

for i=1:out_len
   %feedback = xor(reg(1),reg(4));
   %feedback = xor(and(reg,taps));
   feedback = mod(sum(and(reg,taps)),2);
   reg = [feedback, reg(1:reg_len-1)];
   fprintf('%3d register: [',i);
   fprintf('%1d',reg);
   fprintf(']\n');
   mls_code(i) = reg(reg_len);
end

mls_code

A=1;
signal_sent = -A + 2*A * mls_code;
plot(xcorr(signal_sent,'none'))

% use thge following to verify autocorrelation
mls_len=2^reg_len-1
for i = 1:200
   corrval(i) = signal_sent(1:mls_len)*signal_sent(1+i:mls_len+i)';
end
figure; plot(corrval)
```

H. MATLAB CODE: MAIN_SIMULATION.M

```
% ************************************************************************
% main_simulation.m;
% main_simulation calls functions to generate signals and compute detection and
% TOA & FOA statistics for various levels of noise.
%
% Written by: Joe Crnkovich, NRL
```

```
% Last modified: 15 May 2009
%
% ********************************************************************

%% reset working environment
clear
close all
%clc

%% set operating variables

if exist('config_file.mat','file')  % check if 'config_file.mat' exists
   % use paramters defined in config_file
   source_config_text = ['"config_file.mat" used as parameter source']
   load config_file
else  % use default signal parameters if 'config_file.m' does not exist
   source_config_text = ['DEFAULT values used as parameter source']
   wf_type=3; % 1:const env, const psd; 2:gap in time; 3:gap in psd; 4:shortened pulse
   filter_outside_bnn=1; %limit signal, if generated (i.s., WF1-4), to within Bnn

   f0 = 20000;  %carrier frequency
   fs = 100000; %sample frequency
   Rsym=4000; %2000 %10000;   %symbol rate

   pad_length = 1024; %no. of zeros to add onto each side of S1
   N = 32768-2*pad_length;  %length(samples) of burst; CAF alg. prefers N=2^k

   %--- SNR (Ec_No) of 2.6 (4.15 dB) should give BER .01 for BPSK
   Es_No_dB_min = 0;
   Es_No_dB_step = 5;
   Es_No_dB_max = +35;

   no_noise_iterations = 1000;
   %no_noise_iterations = 100 %200%0; %500 %40 %250;

   %monitor and debug setting
   verbose=0; %set to zero to stop sending debug info to MATLAB window
   verbose_wf_gen=1; %enable plots and sending debug info to MATLAB window
   verbose_plot_wf=1; %enable plots of waveform and calculate rms BW
   enable_BER_test=0; %set to 1 to enable running of BER test function
   process_detections=1; %process detections to get estimates of TOA and FOA
   limit_CAF_to_search_freq_only=0;  % remove ambiguity in CAF (find FDOA only) %Warning:
assumes tau =0
   limit_CAF_to_search_time_only=0;  % remove ambiguity in CAF (find TDOA only)

   %geometry
   Pc1 = [0 0 500]; %use only z position (i.e., leave x&y=0)
   Vc1 = [0 0 0];
   Pc2 = Pc1;
   Vc2 = [0 0 0];
   Pe = [0 0 0];
   Ve = [0 0 0];
```

```
    tau = 1.25e-7; % time offset step used to dither sampling relative to signal
    c = 2.997925e8; % Speed of light in m/s

    % dither position to remove clock sync between runs
    pos_offset_min = 0; % in meters
    pos_offset_step = c*tau; % in meters
    pos_offset_max = 0; %9*c*tau; % in meters  %run ten iterations
end

%no_noise_iterations = 1
wf_type

save init_parameters_dump

%% initialize variables

distr_plot_no = figure(1); %used to plot all the distribution on same figure

Es_No_min = 10^(Es_No_dB_min/10); %convert from dB
Es_No_step = 10^(Es_No_dB_step/10);
Es_No_max = 10^(Es_No_dB_max/10);
Ts = 1/fs;
Tsym = 1/Rsym;
total_bit_errors = 0; total_bits = 0;
no_chips = (Rsym/fs)*N;  %no of PN chips in burst
no_chips_dB = 10*log10(no_chips);

threshold = N/2; %This can be refined, but it allows a quick check

% define CAF search window
Max_f = 1/(N*Ts); % for CAF; 1st null at 1/T = fs/N
Max_t = 2/Rsym;  % for CAF; for a m-sequence, "trainagular peak between +/- 1/Rsym
if limit_CAF_to_search_freq_only
    Max_t=0; end  %Warning: assumes tau =0
if limit_CAF_to_search_time_only
    Max_f=0; end

%clear counters
proc_index=0; %to be incremented for each detection iteration
Es_No_step_no=0;
prev_Es_No_dB = NaN;

%for Es_No=Es_No_min:Es_No_step:Es_No_max
for Es_No_dB=Es_No_dB_min:Es_No_dB_step:Es_No_dB_max
    Es_No_dB; % display SNR (in dB) to MATLAB window
    Es_No = 10^(Es_No_dB/10); %convert from dB
    Es_No_step_no=Es_No_step_no+1;

    Ec_No = Es_No/no_chips;
    Ec_No_dB = 10*log10(Ec_No);

    threshold = 1;  % TBD
```

```
    %clear counters
    pos_offset_index=0;  % keep track of the number of position offsets
    detects_vector_cum(Es_No_step_no)=0; detects_cum(Es_No_step_no)=0;

    for pos_offset=pos_offset_min:pos_offset_step:pos_offset_max

        pos_offset_index=pos_offset_index+1;
        Pc2(3) = Pc1(3) + pos_offset;
        % Pc1(3) = Pc2(3) + pos_offset;  %used to test 3-23-09v2 toa offset

        % First, generate the "received" & "reference" waveforms
        if (wf_type < 10) %  generate_waveform();
            [S1,Sref] = generate_waveform(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N, ...
                wf_type, pad_length, filter_outside_bnn, verbose_wf_gen);
        end

        if (wf_type > 10)  % use previously generated waveform - does not use model
            % generate a waveform to get Es
            wf_type_tmp=1; verbose_wf_gen_tmp=0;
            [S1,Sref] = generate_waveform(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N, ...
                wf_type_tmp, pad_length, verbose_wf_gen_tmp); %  generate_waveform();
            Es = sum(S1.^2);

            %S1 = get_canned_waveform(Es, N, wf_type, pad_length, verbose_wf_gen);
            S1 = get_canned_waveform(Es, N, wf_type, pad_length, Rsym, f0, fs, filter_outside_bnn,
verbose_wf_gen)
            Sref = S1;

            f0 = fs/4;  % the canned waveforms were generated with fc=fs/4
        end

        if (verbose_plot_wf && ~proc_index)
            display_waveform_calc_rmsBW(Sref, f0, fs, wf_type, filter_outside_bnn);
            display_waveform_calc_rmsT(Sref, f0, fs, wf_type, filter_outside_bnn);
        end

        SAref = hilbert(Sref); % Calculates the ANALYTIC SIGNAL of Sref
        clear Sref; %free up memory
        for noise_iteration=1:no_noise_iterations
            proc_index=proc_index+1

            %check if Es/No is new value (and set flag if it is)
            flag_is_new_Es_No = ~(prev_Es_No_dB == Es_No_dB);
            prev_Es_No_dB = Es_No_dB;

            %add noise and take hilbert transform to get I&Q channels
            N_t = gen_noise_vector(length(S1),Ec_No, Tsym, fs);
            S1wnoise=S1+N_t';
            SA1 = hilbert(S1wnoise); % Calculate the ANALYTIC SIGNAL
            clear S1wnoise;  % free up memory

            if enable_BER_test %perform test on gen_sig output with noise
```

```
            %To use, set:
            % - verbose=1; %set to zero to stop sending debug info to MATLAB window
            % - verbose_wf_gen=1; %enable plots and sending debug info to MATLAB window
            % - enable_BER_test=1; %set to 1 to enable running of BER test function
            % - process_detections=0; %process detections to get estimates of TOA and FOA
            % - wf_type=1; % 1:const env, const psd; 2:gap in time; 3:gap in psd; 4:shortened
pulse
            % - Es_No_dB_min & Es_No_dB_max = 4.15 (dB) + no_chips_dB (to give BER .01 for
BPSK)
            % - no_noise_iterations = 1
            % - pad_length = 0; %no. of zeros to add onto each side of S1
            % - Rsym=2000 or 5000;   %symbol rate
            %--- SNR of 2.6 (4.15 dB) should give BER .01 for BPSK
            [BER, no_of_errors, no_of_bits]= perf_demod_test(SA1, ...
                SAref, fs, f0, Rsym, Ec_No_dB, verbose);
            if verbose
                prinf('BER from test demod is %f \n',BER);
            end;
            BER_array(proc_index)=BER;
            total_bit_errors = total_bit_errors + no_of_errors;
            total_bits = total_bits + no_of_bits;
        end

        [rx_out,lags] = xcorr(SA1, SAref, 500);

        if (flag_is_new_Es_No)  % verbose || (flag_is_new_Es_No)
            figure; subplot(1,2,1)
            plot(lags,abs(rx_out));  %'abs' gives envelope, i.e., sqrt(I^2+Q^2)
            %title(['Crosscorrelation Output - Es/No=',num2str(Es_No_dB),'dB']);
            title(['R_{rs} (Es/No=',num2str(Es_No_dB),'dB)']);
            xlabel('# of Lags'); ylabel('Crosscorrelation');
            grid on;

            subplot(1,2,2) %figure;
            plot(lags,10*log10(abs(rx_out)));  %'abs' gives envelope, i.e., sqrt(I^2+Q^2)
            %title(['Crosscorrelation Output - Es/No=',num2str(Es_No_dB),'dB']);
            title(['R_{rs} (Es/No=',num2str(Es_No_dB),'dB)']);
            xlabel('# of Lags'); ylabel('Crosscorrelation (dB)');
            ylim([20, 50]);
            grid on;
        end

        % generate decsion variable at T0 for s+n, s, and noise-only
        xcorr_val(proc_index) = xcorr(SA1, SAref, 0);
        xcorr_val_s(proc_index) = xcorr(hilbert(S1), SAref, 0);
        xcorr_val_n(proc_index) = xcorr(hilbert(N_t),SAref, 0);

        detection = (max(abs(rx_out)) > threshold);
        if detection && process_detections  % if detection occurs (and processing for TOA/FOA
desired
            detects_cum(Es_No_step_no)=detects_cum(Es_No_step_no)+1;

            fprintf('...starting CAF processing...\n');
```

```
            display_CAF_peak = flag_is_new_Es_No; %display CAF plot for 1st iteration

            % Returns TDOA in seconds, FDOA in Hz
            [TDOA, FDOA] = CAFv2(SA1, SAref, Max_f, fs, Max_t,display_CAF_peak);

            if (flag_is_new_Es_No && verbose_plot_wf)  %generate FDOA view of CAF
               [TDOA, FDOA] = CAFv2(SA1, SAref, Max_f, fs, Max_t,display_CAF_peak);
               az = 90; el = 0; view(az, el);
            end

            TDOA=TDOA-tau;  %compensate for position offset

            if(flag_is_new_Es_No)
               title(['Cross Ambiguity Function - Es/No=', num2str(Es_No_dB),'dB']);
            end

            toa_est(proc_index,1)=TDOA;
            toa_est(proc_index,2)=Es_No_dB;
toa_est(proc_index,3)=pos_offset;toa_est(proc_index,4)=noise_iteration;

            foa_est(proc_index,1)=FDOA;
            foa_est(proc_index,2)=Es_No_dB;
foa_est(proc_index,3)=pos_offset;foa_est(proc_index,4)=noise_iteration;
         end

      end
      %no_chips = (Rsym/fs)*N
      %fprintf('Es/No (dB):\n');
      %true_snr(snr_step_no) = 10*log10(no_chips*SNR)

      if (verbose_plot_wf)
         figure; %(1);
         subplot(2,2,1); plot(real(SA1));
         %title(['I-Channel Amplitude vs. Samples - Es/No=', num2str(Es_No_dB),'dB']);
         title('I-Channel Amplitude vs. Samples');
         xlabel('Sample number'); ylabel('Magnitude');

         subplot(2,2,3); plot(abs(SA1));
         %title(['Signal Amplitude vs. Samples - Es/No=', num2str(Es_No_dB),'dB']);
         title('Signal Amplitude vs. Samples');
         xlabel('Sample number'); ylabel('Magnitude');



         no_samples_displayed = 100; % zoom in and display fewer samples
         start_indx = find( (abs(SA1)>0.5), 1, 'first') + 20*fs/Rsym; %4 chips in
         stop_indx = start_indx + no_samples_displayed - 1;

         %subplot(2,2,2); plot([start_indx:stop_indx],real(SA1(start_indx:stop_indx) ));
         subplot(2,2,2); plot(real(SA1));
         xlim([start_indx,stop_indx]);
         title('I-Channel Amplitude vs. Samples');
         xlabel('Sample number'); ylabel('Magnitude');
```

```
            subplot(2,2,4); plot(abs(SA1));
            xlim([start_indx,stop_indx]);
            title('Signal Amplitude vs. Samples');
            xlabel('Sample number'); ylabel('Magnitude');
        end

    end
%     no_chips_dB=10*log10(no_chips)
%     fprintf('Es/No (dB):\n');
%     disp(true_snr);

    if (verbose_plot_wf)  %fills in subplot for each of 8 SNR steps
        figure(distr_plot_no); %(6);
        subplot(4,2,Es_No_step_no);
        %for coherent processing use... (real)
        histfit(real(xcorr_val_n(proc_index-no_noise_iterations+1:proc_index)));
        %otherwise for envelope (magnitude) use... (abs)
        histfit(abs(xcorr_val_n(proc_index-no_noise_iterations+1:proc_index)));
        title(['Correlation Value Distribution - Es/No=', num2str(Es_No_dB),'dB']);

        save interim_all_variables_dump  %save for each iteration of Es/No
    end

end



if enable_BER_test %print out BER results
    fprintf('Cumulative BER is %f (%d of %d)\n', mean(BER_array), ...
        total_bit_errors, total_bits);
end

% save variables to file so they're not lost
save all_variables_dump
save tdoa-fdoa_est_lastrun no_noise_iterations Es_No_step_no toa_est foa_est
```

I. MATLAB CODE: GENERATE_WAVEFORM.M

```
 function [S1,Sref] = generate_waveform(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N, ...
    wf_type, pad_length, filter_outside_bnn, verbose)

% ********************************************************************
% GENERATE_WAVEFORM.m;
% This function generates waveforms 1-4, using gen_sig (a derivative of
% sig_gen developed by Johnson, NPS Thesis Sep '01)which projects a
% BPSK modulated signal onto two collectors as defined by a scenario and
% can accurately introduce doppler compression/expansion onto the signal.
% Waveform #1 (WF1) is the waveform produced by gen_sig (sinc^2 PSD,
% constant amplitude waveform).
% Waveform #2 excises the middle 3/4 of WF1 and increases amplitude so WF2
% has the same energy as WF1.
% Waveform #4 excises the outer 3/4 of WF1 and increases amplitude so WF2
% has the same energy as WF1.
% Waveform #3 has same duration as WF1 but is the sum of two BPSK signals
```

```
% offset from fc but having same Bnn as WF1.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 15 May 2009
%
% ********************************************************************



% if verbose  %open figure for plotting
    wfX_fig = figure;
%    wf3_fig_freq=figure;
% end % end verbose

if wf_type==3 % 1:const env, const psd; 2:gap in time; 3:gap in psd; 4:shortrened pulse

   %generate baseline once to find Es
   [S1,Sref] = gen_sig(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N);
   if verbose
      wfX_fig_freq=figure;
      wf3_fig_freq=figure;
      figure(wfX_fig); subplot(4,1,1); plot(S1);
      title('S1 Amplitude v. Sample Number');
      figure(wf3_fig_freq); subplot(3,1,1); plot(abs(fft(S1)));
      title('S1 FFT');
      xlim([0,N/2]);
   end
   E_s_tmp = sum(S1.^2); % calculate energy in baseline signal

   %generate lower frequency component
   [S1,Sref] = gen_sig(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0-Rsym/2,fs,Rsym/2,N);
   if verbose
      figure(wfX_fig); subplot(4,1,2); plot(S1);
      title('Lower Freq Component');
      figure(wf3_fig_freq); subplot(3,1,2); plot(abs(fft(S1)));
      title('FFT of Lower Freq Component');
      xlim([0,N/2]);
   end
   S1_tmp=S1;  Sref_tmp=Sref;  % make copy of data

   %generate upper frequency component and add to lower
   [S1,Sref] = gen_sig(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0+Rsym/2,fs,Rsym/2,N);
   S1=S1+S1_tmp; Sref=Sref+Sref_tmp;
   if verbose
      figure(wfX_fig); subplot(4,1,3); plot(S1);
      title('New Composite S1');
   end

   % normalize amplitude so same Es
   E_s_reduction = sum(S1.^2)/E_s_tmp
   S1=S1/sqrt(E_s_reduction);  % normalize ampl. so Es same as baseline
   if verbose
      figure(wfX_fig); subplot(4,1,4); plot(S1); title('New Normalized S1');
      figure(wf3_fig_freq); subplot(3,1,3); plot(abs(fft(S1))); title('New Normalized S1');
```

```
        xlim([0,N/2]);
    end

else % WF type is constant PSD (but may have gaps) such as WF1, WF2, WF4, ...

    [S1,Sref] = gen_sig(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N);  % WF1 (baseline)

    if wf_type==2 % (2:gap in time) move power from middle to ends

        E_s_tmp = sum(S1.^2) % calculate energy in baseline signal
        if verbose
            wfX_fig = figure;
            figure(wfX_fig); subplot(3,1,1); plot(S1);
            title('S1 Amplitude v. Sample Number');
        end

        % next zeroize signal for middle 3/4 of baseline waveform
        S1(length(S1)/2-3*length(S1)/8:length(S1)/2+3*length(S1)/8)=0;
        if verbose
            subplot(3,1,2); plot(S1); title('S1 After Excising Middle');
        end

        % normalize amplitude so same Es
        E_s_reduction = sum(S1.^2)/E_s_tmp
        S1=S1/sqrt(E_s_reduction);
        if verbose
            subplot(3,1,3); plot(S1); title('New Normalized S1');
        end
        %new_E_s_tmp = sum(S1.^2);

        %...and do the same for the reference signal
        E_s_tmp = sum(Sref.^2) % calculate energy in baseline signal
        Sref(length(Sref)/2-3*length(Sref)/8:length(Sref)/2+3*length(Sref)/8)=0;
        E_s_reduction = sum(Sref.^2)/E_s_tmp
        Sref=Sref/sqrt(E_s_reduction);  % normalize amplitude so same Es
    end

    if wf_type==4 % (4:shortened pulse) move power from ends to middle

        E_s_tmp = sum(S1.^2) % calculate energy in baseline signal
        if verbose
            wfX_fig_freq=figure;
            figure(wfX_fig); subplot(3,1,1); plot(S1); title('S1 Amplitude v. Sample Number');
        end  %end verbose
        S1(1:length(S1)/2-length(S1)/8)=0;  % remove signal from front
        S1(length(S1)/2+length(S1)/8:length(S1))=0;  %remove signal from back
        if verbose
            subplot(3,1,2); plot(S1); title('S1 After Excising Ends');
        end  %end verbose

        % normalize amplitude so same Es
        E_s_reduction = sum(S1.^2)/E_s_tmp
        S1=S1/sqrt(E_s_reduction);  % normalize amplitude so same Es
```

```
    if verbose
       subplot(3,1,3); plot(S1); title('New Normalized S1');
    end  %end verbose

    %...and do the same for the reference signal
    %new_E_s_tmp = sum(S1.^2)
    E_s_tmp = sum(Sref.^2) % calculate energy in baseline signal
    if verbose
       figure(wfX_fig); subplot(3,1,1); plot(Sref);
       title('Sref Amplitude v. Sample Number');
    end  %end verbose
    Sref(1:length(Sref)/2-length(Sref)/8)=0;  % remove signal from front
    Sref(length(Sref)/2+length(Sref)/8:length(Sref))=0;  %remove signal from back
    if verbose
       subplot(3,1,2); plot(Sref); title('Sref After Excising Middle');
    end  %end verbose
    E_s_reduction = sum(Sref.^2)/E_s_tmp
    Sref=Sref/sqrt(E_s_reduction);  % normalize amplitude so same Es
    if verbose
       subplot(3,1,3); plot(Sref); title('New Normalized Sref');
    end  %end verbose

  end  %end wf_type=4

end

if filter_outside_bnn
   % note: this was meant to be used only for static signal scenario
   S1 = filt_bnn_fft(S1, Rsym, f0, fs);
   Sref = filt_bnn_fft(Sref, Rsym, f0, fs);
end

if pad_length %pad beginning and end of waveform with zeros
   S1 = [zeros(1,pad_length), S1, zeros(1,pad_length)];
   Sref = [zeros(1,pad_length), Sref, zeros(1,pad_length)];
end
```

J. MATLAB CODE: GEN_SIG.M

```
 function [S1,S2] = gen_sig(Pc1,Vc1,Pc2,Vc2,Pe,Ve,f0,fs,Rsym,N)
% ************************************************************************
% [S1] = gen_sig;
% GEN_SIG generates BPSK signal pairs based upon user-defined param-
% eters and Cartesian emitter-collector geometries usign the signal . The following
% input arguments are used:
%   Pc1 - initial position of collector1 in meters [x y z] (e.g., [0 0 7500])
%   Vc1 - velocity of collector1 in m/s [x y z]
%   Pc2 - initial position of collector2 in meters [x y z] (e.g., [0 0 7500])
%   Vc2 - velocity of collector2 in m/s [x y z]
%   Pe - initial position of emitter in meters [x y z] (e.g., [0 0 7500])
%   Ve - velocity of emitter in m/s [x y z]
%   f0 - carrier frequency
%   fs - sampling rate
```

```
%   Rsym - symbol rate
%   N - number of samples
%
% The function returns the vector S1 which is the Real representation
% of the received signal.
%
% Extracted from SIG_GEN.m, which was by: LCDR Joe J. Johnson, USN
%
% Modified by J. Crnkovich
% major changes from SIG_GEN.m:
%    1- Does not prompt for input arguments (they must now be passed in)
%    2- Es_No not used because this is processed externally
%    3- Conversion to analytic signal performed externally
%    4- bit sequence is read in from file (currently an m-sequence)
%    5- does not perform Hilbert transform to convert to analytic signal
%
% Last modified: 15 May 2009
%
% ********************************************************************

Ts = 1/fs;
Tsym = 1/Rsym;

Pc1 = [Pc1; zeros(N-1, 3)]; % Initializing all the matrices makes
Pe1 = zeros(N, 3); % later computations much faster.
Pc2 = [Pc2; zeros(N-1, 3)];
Pe2 = zeros(N, 3);
t1 = zeros(1, N);
t2 = zeros(1, N);
S1 = zeros(1, N);
S2 = zeros(1, N);

A = 1; % Amplitude of Signal
c = 2.997925e8; % Speed of light in m/s
Ps = (A^2)/2; % Power of Signal

% % sigma1 = sqrt(Ps*Tsym/Es_No1) % Calculate Noise Amplification fac
% % sigma2= sqrt(Ps*Tsym/Es_No2) % tors using Es/No = Ps*Tsym/sigma^2
% % Corrected formula below - JGC 2/12/09
% % From Johnson paper, sigma^2 = (Ps*Tsym*B/Es_No):  However B is not equal
% % to 1 (as stated in the paper), the digital frequency bandwidth, but is
% % rather the true bandwidth, fs/2 (or 1/2Ts).
% % sigma1 = sqrt(Ps*Tsym/Es_No1) % Calculate Noise Amplification fac
% % sigma2= sqrt(Ps*Tsym/Es_No2) % tors using Es/No = Ps*Tsym/sigma^2
%  sigma1 = sqrt(0.5*Ps*(Tsym/Ts)/Es_No1) % Calculate Noise Amplification fac
%  sigma2 = sqrt(0.5*Ps*(Tsym/Ts)/Es_No2) % tors using Es/No = Ps*Tsym*B/sigma^2
%
% Noise1 = sigma1.*randn(N, 1); % Random Noise values for Signal 1
% Noise2 = sigma2.*randn(N, 1); % Random Noise values for Signal 2


% Builds the position vectors for the two collectors
for index = 2 : N
```

```
    Pc1(index,:) = Pc1(index - 1,:) + Ts*Vc1;
    Pc2(index,:) = Pc2(index - 1,:) + Ts*Vc2;
end

% While loop determines first elements of Pe1 and t1. t1(1) is the
% time AT THE EMITTER that produces the 1st sample received at
% collector 1! Pe1(1,:) is the position of the emitter when it
% produces the 1st sample received by collector 1.

temp = inf; % Ensures while loop executes at least once
t1(1) = 0;
tempPe = Pe(1,:);
while abs(temp - t1(1)) > 1/f0
    temp = t1(1);
    t1(1) = -norm(Pc1(1,:) - tempPe) / c;
    tempPe = Pe(1,:) + t1(1)*Ve;
end
Pe1(1,:) = tempPe;

% While loop determines first elements of Pe2 and t2. t2(1) is the
% time AT THE EMITTER that produces the 1st sample received at
% collector 2! Pe2(1,:) is the position of the emitter when it
% produces the 1st sample received by collector 2.

temp = inf; % Ensures while loop executes at least once
t2(1) = 0;
tempPe = Pe(1,:);
while abs(temp - t2(1)) > 1/f0
    temp = t2(1);
    t2(1) = -norm(Pc2(1,:) - tempPe) / c;
    tempPe = Pe(1,:) + t2(1)*Ve;
end
Pe2(1,:) = tempPe;

% Platform positions at middle of snapshot
Pcc1=(Pc1(N/2,:));
Pcc2=(Pc2(N/2,:));
% Determines the earliest time at the emitter for this pair of signals.
StartPoint = min(t1(1), t2(1));


% Next 2 lines determine offsets needed for signals 1 & 2 to enter the
% phase vector (P). This simply ensures proper line up so that bit
% changes occur at the right times.
SymbolIndex1 = 1 + floor(abs(t1(1) - t2(1))/Tsym) * (t1(1)>t2(1));
SymbolIndex2 = 1 + floor(abs(t1(1) - t2(1))/Tsym) * (t2(1)>t1(1));


for index = 2 : N % Builds the Pe1 and t1 vectors
    temp = inf;
    t1(index) = 0;

    % 1st guess is that emitter will advance exactly Ts seconds.
```

```
    tempPe = Pe1(1,:) + (t1(index -1) + Ts)*Ve;

    % While loop iteratively determines actual time & position for
    % emitter, based on instantaneous geometry.

    while abs(temp - t1(index)) > 1/f0
        temp = t1(index);
        t1(index) = (index - 1)*Ts - norm(Pc1(index,:) - tempPe) / c;

        % Due to negative times, must multiply Ve by ELAPSED time!
        tempPe = Pe1(1,:) + abs(t1(1)-t1(index))*Ve;
    end
    Pe1(index,:) = tempPe;
end


for index = 2 : N %Builds the Pe2 and t2 vectors
    temp = inf;
    t2(index) = 0;

    % 1st guess is that emitter will advance exactly Ts seconds.
    tempPe = Pe2(1,:) + (t2(index -1) + Ts)*Ve;

    % While loop iteratively determines actual time & position for
    % emitter, based on instantaneous geometry.
    while abs(temp - t2(index)) > 1/f0
        temp = t2(index);
        t2(index) = (index - 1)*Ts - norm(Pc2(index,:) - tempPe) / c;
        % Due to negative times, must multiply Ve by ELAPSED time!
        tempPe = Pe2(1,:) + abs(t2(1)-t2(index))*Ve;
    end
    Pe2(index,:) = tempPe;
end


% Could change this seed to whatever you want; or could have user
% define it as an input. This just ensures, for simulation purposes
% that every time the program is run, the BPSK signals created will
% have the same random set of data bits.
rand('seed',5);

% % Create enough random #'s to figure phase shift (data bits)
% r = rand(N,1);
% P = (r > 0.5)*0 + (r <= 0.5)*1; % Since BPSK, random # determines if phase is 0 or pi

%%  Import 65535 length m-sequence to use instead of random numbers
load mls65535a
P = zeros(N,1);
tmp=min(N,65535);
P(1:tmp)=mls65535a(1:tmp);

% Building Xmitted Signal #1 vector... These represent the pieces of
```

```
% the signal that were transmitted by the emitter to arrive at
% Collector 1 at its sample intervals.

S1(1) = A*cos(2*pi*f0*t1(1) + P(SymbolIndex1)*pi) ;%+ Noise1(1);

% The if statement inside the loop changes the data bit if the time
% has advanced into the next symbol period.
for index = 2 : N
   if t1(index) - StartPoint >= (SymbolIndex1) * Tsym
      SymbolIndex1 = SymbolIndex1 + 1;
   end
   S1(index) = A*cos(2*pi*f0*t1(index) + P(SymbolIndex1)*pi) ;%+ ...
%       Noise1(index);
end

% Sa1 = hilbert(S1); % Calculates the ANALYTIC SIGNAL of S1. To
              % compute the COMPLEX ENVELOPE, multiply Sa1
              % by .*exp(-j*2*pi*f0.*t1);


% Building Xmitted Signal #2 vector... These represent the pieces of
% the signal that were transmitted by the emitter to arrive at
% Collector 2 at its sample intervals.

S2(1) = A*cos(2*pi*f0*t2(1) + P(SymbolIndex2)*pi) ;%+ Noise2(1);

% The if statement inside the loop changes the data bit if the time
% has advanced into the next symbol period.
for index = 2 : N
   if t2(index) - StartPoint >= (SymbolIndex2) * Tsym
      SymbolIndex2 = SymbolIndex2 + 1;
   end
   S2(index) = A*cos(2*pi*f0*t2(index) + P(SymbolIndex2)*pi) ;%+ ...
%       Noise2(index);
end
```

K. MATLAB CODE: FILT_BNN_FFT.M

```
 function S = filt_bnn_fft(S, Rsym, f0, fs)
% ***********************************************************************
% filt_bnn_fft.m;
% This function filters the out all signal energy outside the
% null-null-bandwidth (fc +/- Rsym) and returns the real signal S.
% The output signal is rescaled so that it has the same energy as the
% input signal.
%
% Note, a constant envelope signal passing through this "brick-wall"
% filter will no longert be constant envelope.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 15 May 2009
%
% ***********************************************************************
```

```
Es_in = sum(S.^2);  % Energy in original signal

SA=hilbert(S);  % calculate the analytic signal (make "positive spectrum only")

SA_fft = fft(SA); % convert to frequency domain

% filter out any signal outside Bnn (i.e., fc+/-Rsy
SA_fft(1:round((f0-Rsym)*length(S)/fs)) = 0;
SA_fft(round((f0+Rsym)*length(S)/fs:length(S))) = 0;

SA = ifft(SA_fft);  % convert back to time domain

S = real(SA); % make the signal real again

Es_filt = sum(S.^2);  % Energy in filtered signal

S = S*sqrt(Es_in/Es_filt);  % scale signal so it has same energy as input

% figure; plot(abs(fft(S)))
% figure; plot(S)
```

L. MATLAB CODE: GET_CANNED_WAVEFORM.M

```
   function S1 = get_canned_waveform(Es, N, wf_type, pad_length, Rsym, f0, fs,
filter_outside_bnn, verbose_wf_gen)
% ************************************************************************
% get_canned_waveform.m;
% get_canned_waveform retrieves previous saved waveforms as determined
% by wf_type.  The waveform is truncated to N samples plus padded at
% the beginning and end wit zeros each of length 'pad_length'.  It is
% scaled to so total energy is Es.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 4 May 2009
%
% ************************************************************************
if (wf_type==11) % wideband shaped pulses (4 Samples/pulse)
   load sinc_wb_mls65535a
end

if (wf_type==12) % mediumband shaped pulses (8 Samples/pulse)
   load sinc_mb_mls65535a
end

if (wf_type==13) % narrowband shaped pulses (16 Samples/pulse)
   load sinc_nb_mls65535a
end

if wf_type==14 % very-narrow-band shaped pulses (32 Samples/pulse)
   load sinc_vnb_mls65535a
end
```

```
if wf_type==15 % ultra-narrow-band shaped pulses (64 Samples/pulse)
   load sinc_unb_mls65535a
end

if wf_type==16 % extremely-narrow-band shaped pulses (128 Samples/pulse)
   load sinc_xnb_mls65535a
end

if wf_type==17 % 12 Samples/pulse (~Bnn of 4kcps BPSK @fs=100000
   load sinc_12Spc_mls65535a
end


S1 = modulation(1:N);

Es_new = sum(S1.^2);

% normalize Es
S1 = S1*sqrt(Es/Es_new);

if filter_outside_bnn
   % note: this was meant to be used only for static signal scenario
   S1 = filt_bnn_fft(S1, Rsym, f0, fs);
end

if pad_length %pad beginning and end of waveform with zeros
   S1 = [zeros(1,pad_length), S1, zeros(1,pad_length)];
end

%           figure; plot(abs(fft(S1)))
%           figure; plot(10*log10(abs(fft(S1)).^2))
```

M. MATLAB CODE: DISPLAY_WAVEFORM_CALC_RMSBW.M

```
function display_waveform_calc_rmsBW(Sref, f0, fs, wf_type, filter_outside_bnn)
% ************************************************************************
% display_waveform.m;
% display_waveform plots waveform and calculates rms radian frequency.
%
% Written by: Joe Crnkovich (NRL)
% Last modified: 10 June 2009
%
% ************************************************************************

%%  display psd (Welch)

figure;
h = spectrum.welch;              % Create a Welch spectral estimator.
Hpsd = psd(h,Sref,'Fs',fs);          % Calculate the PSD
plot(Hpsd);
```

```
%%  display psd along with bandwidths used

% PSD = (1/N)|fft(x(n)|^2

% Convert to analytic waveform
Sref = hilbert(Sref);

% calculate true rms radian frequency
fc_idx = floor(f0*length(Sref)/fs);
i=1:length(Sref);
Sref_psd = abs(fft(Sref)).^2;  % find (unnormalized) PSD
f_squared = ( abs(i-fc_idx) .* (fs/length(Sref)) ).^2;  % f is weighting

% calculate rms radian frequncy (beta)
beta_Hz = ( sum(f_squared.*Sref_psd)/sum(Sref_psd) ).^0.5
beta = 2*pi*beta_Hz;

x_axis=(i-1)*fs/length(Sref);

b_rms=[f0-beta,f0-beta,f0+beta,f0+beta];
b_Hz = [f0-beta_Hz,f0-beta_Hz,f0+beta_Hz,f0+beta_Hz];

%% Plot PSD overlayed with both \beta and B_{Hz} 'bandwidths'

PSD = (abs(fft(Sref)).^2)/(length(Sref)*fs);  % PSD in linear (non-dB) scale

figure;
plot(x_axis/1000,10*log10(PSD), ...
   b_Hz/1000,[-90,-22,-22,-90], '-k')

if filter_outside_bnn
   title_text=['PSD (Waveform #', ...
      num2str(wf_type),'-Filt; \beta=', num2str(beta),' rad/s)'];
else
   title_text=['PSD (Waveform #', ...
      num2str(wf_type),'; \beta=', num2str(beta),' rad/s)'];
end
title(title_text);
ylabel('Power (dB/Hz)'); xlabel('Frequency (kHz)')
legend('PSD of signal', '+/- \beta_{Hz}')
xlim([0 fs/2000])
ylim([-90,-20])
grid on

%%  Plot 'weighted' vs 'unweighted' PSD in separate subplots

x_axis = ((i-1)*fs/length(Sref))/1000;

figure
subplot(2,1,1)
plot(x_axis, 10*log10(f_squared.*Sref_psd))
title_text = ['Weighted PSD of Analytic Signal -- Waveform #', num2str(wf_type)];
```

```
title(title_text)
xlim([0,fs/2000]); ylim([80, 140])
xlabel('kHz');  ylabel('dB')
grid on

%figure
subplot(2,1,2)
plot(x_axis, 10*log10(Sref_psd))
title_text = ['PSD of Analytic Signal -- Waveform #', num2str(wf_type)];
title(title_text)
xlim([0,fs/2000]); ylim([10, 70])
xlabel('kHz');  ylabel('dB')
grid on
```

N. MATLAB CODE: DISPLAY_WAVEFORM_CALC_RMST.M

```
 function display_waveform_calc_rmsT(Sref, f0, fs, wf_type, filter_outside_bnn)
% ***********************************************************************
% display_waveform_calc_rmsT.m;
% display_waveform plots waveform and calculates rms Time ('Te').
%
% Written by: Joe Crnkovich (NRL)
% Last modified: 15 May 2009
%
% ***********************************************************************

%%  convert to analytic signal

Sref = hilbert(Sref);

%% calculate true rms time

tc_idx = floor(length(Sref)/2)  %index to center of waveform (assumes symmetric)
i=1:length(Sref);

Sref_ut2 = abs(Sref).^2;  %power vs. time

t_squared = ( abs(i-tc_idx)/fs ).^2;
%plot(t_squared)

Te = 2*pi*( sum(t_squared.*Sref_ut2)/sum(Sref_ut2) ).^0.5

%%

%% Plot Power vs. Time and zoomed Power vs. Time in separate subplots
x_axis = [1:length(Sref)]/fs;

%first find where signal power breaks threshold
no_samples_displayed = 500;
offset=500;  %4 chips in @ 4kcps, 100kS/s

start_indx = find( (abs(Sref)>0.5), 1, 'first') + offset; %4 chips in
```

```
stop_indx = start_indx + no_samples_displayed - 1;

start_indx = start_indx/fs
stop_indx = stop_indx/fs

figure
subplot(2,1,1)
plot(x_axis,10*log10(Sref_ut2))

if filter_outside_bnn
   title_text = ['Power vs. Time of Analytic Signal -- Waveform #', ...
      num2str(wf_type), '-Filt, Te=', num2str(Te),' s'];
else
   title_text = ['Power vs. Time of Analytic Signal -- Waveform #', ...
      num2str(wf_type), ', Te=', num2str(Te),' s'];
end
title(title_text)
ylim([-50, 20])
xlabel('Time (s)');  ylabel('Power (dB)')
grid on

%figure
subplot(2,1,2)
plot(x_axis, 10*log10(Sref_ut2))
title_text = ['Zoomed Power vs. Time of Analytic Signal'];
title(title_text)
xlim([start_indx,stop_indx]); ylim([-40,10])
xlabel('Time (s)');  ylabel('Power (dB)')
grid on


%% Plot autocorrelation of reference signal

% find autocorrelation of signal normalized to 1
%[autocorrel,lags] = xcorr( Sref, 1000,'coeff');
[autocorrel,lags] = xcorr( Sref,'coeff');

figure; subplot(1,2,1)
plot(lags,abs(autocorrel),'LineWidth',2);  %'abs' gives envelope, i.e., sqrt(I^2+Q^2)
title('R_s');
xlabel('# of Lags'); ylabel('Magnitude of Autocorrelation');
xlim([lags(1), -lags(1)]);
grid on;

subplot(1,2,2) %figure;
plot(lags,10*log10(abs(autocorrel)),'LineWidth',2);  %'abs' gives envelope, i.e., sqrt(I^2+Q^2)
title('R_s');
xlabel('# of Lags'); ylabel('Magnitude of Autocorrelation (dB)');
xlim([lags(1), -lags(1)]);
ylim([-40, 0]);
grid on;

% and zoomed dB version...
```

```
no_lags_displ = 150;

figure; subplot(1,2,1)
plot(lags,10*log10(abs(autocorrel)));  %'abs' gives envelope, i.e., sqrt(I^2+Q^2)
title('R_s');
xlabel('# of Lags'); ylabel('Normalized Magnitude of Autocorrelation (dB)');
xlim([lags(1), -lags(1)]);
ylim([-40, 0]);
set(gca,'YGrid','on');

subplot(1,2,2) %figure;
plot(lags,10*log10(abs(autocorrel)));  %'abs' gives envelope, i.e., sqrt(I^2+Q^2)
title('R_s');
xlabel('# of Lags'); ylabel('Normalized Magnitude of Autocorrelation (dB)');
xlim([-no_lags_displ, no_lags_displ]);
ylim([-40, 0]);
%set(gca,'YGrid','on');
grid on

figure
%plot(abs(fft(autocorrel)))
plot(10*log10(abs(fft(autocorrel))))
title('|FFT(R_s)|_{dB}');
xlabel('FFT bin number'); ylabel('Magnitude (dB)');
```

O. MATLAB CODE: GEN_NOISE_VECTOR.M

```
 function Noise=gen_noise_vector(N, SNR, Tsym, fs)
% ************************************************************************
% gen_noise_vector.m;
% gen_noise_vector generates vector containing noise samples.
%
% Written by: Joe Crnkovich, NRL
% Last modified: 3 April 2009
%
% ************************************************************************

A = 1; % Amplitude of Signal
% % c = 2.997925e8; % Speed of light in m/s
Ps = (A^2)/2; % Power of Signal

% % sigma1 = sqrt(Ps*Tsym/Es_No1) % Calculate Noise Amplification fac
% % sigma2= sqrt(Ps*Tsym/Es_No2) % tors using Es/No = Ps*Tsym/sigma^2
% % Corrected formula below - JGC 2/12/09
% % From Johnson paper, sigma^2 = (Ps*Tsym*B/Es_No):  However B is not equal
% % to 1 (as stated in the paper), the digital frequency bandwidth, but is
% % rather the true bandwidth, fs/2 (or 1/2Ts).
% % sigma1 = sqrt(Ps*Tsym/Es_No1) % Calculate Noise Amplification fac
% % sigma2= sqrt(Ps*Tsym/Es_No2) % tors using Es/No = Ps*Tsym/sigma^2
% % sigma1 = sqrt(0.5*Ps*(Tsym/Ts)/Es_No1) % Calculate Noise Amplification fac
%  sigma2 = sqrt(0.5*Ps*(Tsym/Ts)/Es_No2) % tors using Es/No = Ps*Tsym*B/sigma^2
```

```
sigma1 = sqrt(Ps*(Tsym*fs/2)/SNR); % Calculate Noise Amplification fac

Noise = sigma1.*randn(N, 1); % Random Noise values for Signal 1
```

P. MATLAB CODE: PERF_DEMOD_TEST.M

```
 function [BER, no_of_errors, no_of_bits]=perf_demod_test(Sa1, Sa2, fs, f0, Rsym, SNRdB,
verbose)

% ************************************************************************
% PERF_DEMOD_TEST.m;
% This function is used to test validity of Sa1 signal by attempting to
% demodulate a BPSK modulated signal.  Various diagnostic plots are
% produced, the user is asked to manually perform phase synchronization
% by identifying peak signal (assume no/low noise (high SNR), and BER is
% calculated by comparing demodulated bits to first bits loaded from
% mls65535a.mat.
%
%To use within main_simulate.m, set the following parameters:
% - verbose=1; %set to zero to stop sending debug info to MATLAB window
% - verbose_wf_gen=1; %enable plots and sending debug info to MATLAB window
% - enable_BER_test=1; %set to 1 to enable running of BER test function
% - process_detections=0; %process detections to get estimates of TOA and FOA
% - wf_type=1; % 1:const env, const psd; 2:gap in time; 3:gap in psd; 4:shortened pulse
% - Es_No_dB_min & Es_No_dB_max = 4.15 (dB) + no_chips_dB (to give BER .01 for BPSK)
% - no_noise_iterations = 1
% - pad_length = 0; %no. of zeros to add onto each side of S1
% - Rsym=2000 or 5000;   %symbol rate
%--- SNR of 2.6 (4.15 dB) should give BER .01 for BPSK

%
% Written by: Joe Crnkovich, NRL
% Last modified:  15 May 2009
%
% ************************************************************************

%%
% use the following to perform crosscorrelation

N=length(Sa1);
window = 1000;
hlfwndw = window/2;
for i = 1:window
   corrval(i) = Sa1(hlfwndw:N-hlfwndw)*Sa2(i:N-window+i)';
end

if verbose
   figure;
   subplot(4,1,1); plot(real(corrval))
   title('real(corrval) - Sa1 & Sa2');
   subplot(4,1,2); plot(imag(corrval))
   title('imag(corrval) - Sa1 & Sa2');
```

```
    subplot(4,1,3); plot(abs(corrval))
    title('abs(corrval) - Sa1 & Sa2');
    subplot(4,1,4); plot(10*log10(abs(corrval)))
    title('abs(corrval) - Sa1 & Sa2');
end

%%
% Plot Sa1 & Sa2 (freq domain) to show old v. new calc of noise signal
mix=[1:length(Sa1)];
if verbose
    figure;
    subplot(1,2,1); plot((mix/length(mix) * fs)-fs/2, fftshift(abs(fft(real(Sa1)))));
    title(['FFT of RF Signal With Noise - SNR=',num2str(SNRdB),' dB']);
    xlabel('Frequency (Hz)');
    ylim([0,2000]);

    subplot(1,2,2); plot((mix/length(mix) * fs)-fs/2, fftshift(abs(fft(real(Sa2)))));
    title('FFT of RF Signal - No noise');
    xlabel('Frequency (Hz)');
    ylim([0,2000]);
end

%% Mix signal back down to baseband
mix=[1:length(Sa1)];

if verbose
    figure; plot(mix/length(mix) * fs, abs(fft(Sa1)));
    %figure; plot((mix/length(mix) * fs)-fs/2, fftshift(abs(fft(Sa1))));
    title('FFT of (analytic) RF Signal (Sa1)');
    xlabel('Frequency (Hz)');
    (1.319/6.554)*fs
end

% show baseband signal
% f0 = 20000;
SaBB = Sa1.*exp(-2*pi*(f0/100000)*1j.*mix);
SaBBref = Sa2.*exp(-2*pi*(f0/100000)*1j.*mix);

if verbose
    figure; plot(mix/length(mix) * fs, abs(fft(SaBB)));
    title('FFT of Baseband Signal (SaBB)');
    xlabel('Frequency (Hz)');

    figure;
    subplot(1,2,1); plot((mix/length(mix) * fs)-fs/2, fftshift(abs(fft(Sa1))));
    title('FFT of (analytic) RF Signal (Sa1)');
    xlabel('Frequency (Hz)');
    subplot(1,2,2); plot((mix/length(mix) * fs)-fs/2, fftshift(abs(fft(SaBB))));
    title('FFT of Baseband Signal (SaBB)');
    xlabel('Frequency (Hz)');
end
```

```
%%
% figure out phase error
mix2=[0:0.01:2*pi]; SaBB2500=SaBBref(2500); %SaBB2500=SaBBref(2500);
SaBB2 = SaBB2500.*exp(1j*mix2);

default_phase_offset = 4.6; %4.83;

if verbose
   figure; plot(mix2,real(SaBB2));
   title('Amplitude (I) vs. Phase Offset of Baseband Sample #2500'); %Sample #2500');
   xlabel('Phase Offset (Radians)');
   fprintf('default phase offset is %d\n',default_phase_offset);
end

% ask user for phase offset (i.e., when is signal peak)
phase_offset = input('Enter Desired Phase Offset (radians) from Plot ("d" for default): ')
if (phase_offset=='d')
   phase_offset = default_phase_offset
end;

%%
% show phase corrected I&Q signals

SaBB = SaBB.*exp(1j*phase_offset);  %add phase offset to bring signal to I channel
%SaBB = SaBBref.*exp(j*phase_offset);  %add phase offset to bring signal to I channel

if verbose
   figure;
   subplot(3,1,1); plot(real(SaBB)); xlim([1,5000]); title('SaBB - Baseband I channel')
   subplot(3,1,2); plot(imag(SaBB)); xlim([1,5000]); title('SaBB - Baseband Q channel')
   subplot(3,1,3); plot(unwrap(angle(SaBB))); xlim([1,5000]); title('SaBB - Baseband Phase')
end

%%
if verbose
   figure;
   subplot(2,2,1); histfit(real(SaBB)); title('real(SaBB)');
   subplot(2,2,3); histfit(imag(SaBB)); title('imag(SaBB)');
   subplot(2,2,2); qqplot(real(SaBB)); title('real(SaBB)');
   subplot(2,2,4); qqplot(imag(SaBB)); title('imag(SaBB)');
   fprintf('mean(real(SaBB)=%f\n', mean(real(SaBB)));
   fprintf('variance(real(SaBB)=%f\n', (std(real(SaBB)))^2);
   fprintf('mean(imag(SaBB)=%f\n', mean(imag(SaBB)));
   fprintf('variance(imag(SaBB)=%f\n\n', (std(imag(SaBB)))^2);
   fprintf('skewness(real(SaBB)=%f\n', skewness(real(SaBB)));
   fprintf('kurtosis(real(SaBB)=%f\n', kurtosis(real(SaBB))-3);
   fprintf('skewness(imag(SaBB)=%f\n', skewness(imag(SaBB)));
   fprintf('kurtosis(imag(SaBB)=%f\n', kurtosis(imag(SaBB))-3);
end

%%
% apply matched filter for pulse of length p_length
p_length = fs/Rsym; %50;
```

```
if verbose p_length; end
mf_pulse = ones(p_length,1); %column vector
mf_out = filter(mf_pulse,1,real(SaBB));
if verbose
   figure; plot(mf_out); xlim([1,5000]);
   title('Output of I-Channel Matched Filter')

   figure;
   subplot(2,2,1); histfit(real(mf_out)); title('real(mf out)');
   subplot(2,2,3); histfit(imag(mf_out)); title('imag(mf out)');
   subplot(2,2,2); qqplot(real(mf_out)); title('real(mf out)');
   subplot(2,2,4); qqplot(imag(mf_out)); title('imag(mf out)');
end

%%
sampled_decision_variable = downsample(mf_out,p_length);
load mls65535a;
ref_data = [0,mls65535a(1:length(sampled_decision_variable)-1)];
demodulated_bits = (sampled_decision_variable > 0);
errors=xor(demodulated_bits, ref_data);
no_of_errors = sum(errors);
no_of_bits = length(errors);
BER = no_of_errors/no_of_bits;
if verbose
   no_of_errors
   no_of_bits
   BER

   figure;
   subplot(3,1,1); plot(sampled_decision_variable); title('Sampled Decision Variable');
   xlim([1,75]); %ylim([-1.1,1.1]);
   subplot(3,1,2); plot(demodulated_bits); title('Demodulated Bits');
   xlim([1,75]); ylim([-0.1,1.1]);
   subplot(3,1,3); plot(ref_data); title('Transmitted Data (m-sequence)');
   xlim([1,75]); ylim([-0.1,1.1]);

   % Squaring the signal  - should produce tone at twice the carrier freq
   figure; plot(mix/length(mix) * fs, abs(fft(Sa1.*Sa1)))
   %plot(mix/length(mix) * fs, 10*log10(abs(fft(Sa1.*Sa1))))
   title('FFT of Sa1^2 (i.e., Sa1 Squared');
   xlabel('Frequency (Hz)');

   figure;plot(mix/length(mix) * fs, 10*log10(abs(fft(Sa1.*Sa1))))
   title('FFT of Sa1^2 (i.e., Sa1 Squared');
   xlabel('Frequency (Hz)');
   ylabel('dB')
end
```

Q. MATLAB CODE: CAFV2.M

```
function [TDOA, FDOA] = CAFv2(S1, S2, Max_f, fs, Max_t, display_CAF_peak);

% ********************************************************************
```

```
% CAF takes as inputs two sampled signal vectors (S1 & S2) in analytic
%       signal format, the maximum expected FDOA in Hertz (Max_f), the
%       sampling frequency used to generate S1 & S2 (fs), and the maximum
%       expected TDOA in seconds (Max_t).  The function then utilizes
%       Stein's method in [1] to compute coarse estimations of TDOA and
%       FDOA between S1 & S2.  Finally, "fine mode" calcualtions are made
%       to compute the final TDOA and FDOA, which are returned to the
%       user via the output arguments.

% Written by:  LCDR Joe J. Johnson, USN
% Last modified:  17 September 2001
%
% Modified by J. Crnkovich, NRL
% Last Modified: 5 March 2009
%
% ********************************************************************

%clc;

%display_CAF_peak=1; %allows program to call CAF_peak.m which displays CAF peak

N = length(S1);
S1 = reshape(S1,N,1);       % Ensures signals are column vectors due to
S2 = reshape(S2,N,1);       % Matlab's better efficiency on columns

S1_orig = S1;               % Want to preserve original input signals
S2_orig = S2;               % for later use; S1 & S2 will be
                            % manipulated in the fine mode below.
TDOAold=NaN;
FDOAold=NaN;



% The following while loop ensures that the sub-block size, N1, is
% large enough to ensure proper resolution.  If Max_f/fs*N1 were
% less than 1, then the Freq calculated at the end would always be
% + or - 1/N1!  2^19 = 524288 is about the limit for efficient
% processing speed.
N1=1024;
while (Max_f/fs*N1 < 2) & (N1 < 2^19)
   N1 = 2*N1;
end

N2=N1/2;

if N1 > N                       % For cases where resolution calls for
   S1 = [S1;zeros(N1-N,1)];     % a sub-block size larger than the
   S2 = [S2;zeros(N1-N,1)];     % signal vectors, pad the vectors with
   N = N1;                      % zeros so that they have a total of
end                             % N1 elements.

% Want magnitude of Max_f, since +&- will be used below
Max_f = abs(Max_f);
Number_of_Blocks = length(S1)/N1;   % Number of sub-blocks to break
```

```
                                % the signal into

Min_v = floor(-Max_f/fs*N1);          % Smallest freq bin to search
Max_v = -Min_v;                       % Largest freq bin to search
v_values = Min_v : Max_v;             % Vector of all bins to search

Max_samples = Max_t * fs;       % Maximum number of samples to search

% Finds max number of block shifts (q) that must occur for each
% R and v below.
if Max_samples > N2
   q_max = min(ceil((Max_samples - N2)/N1),Number_of_Blocks-1);
else q_max = 0;
end

x=0;
divisors = Number_of_Blocks:-1:1;       % Used to scale "temp" below...


% ***********************************************************************
% COARSE MODE computations.
% ***********************************************************************

for v = 1:length(v_values)
  temp(1:N1,1:q_max+1)=0;         % Initializing -- saves time....
  for R = 0:Number_of_Blocks-1

    % temp1 is the FFT of the R'th block of S1, shifted by "v" bins.
    temp1 = fftshift(fft(S1(1+R*N1 : N1*(R+1))));
    temp1 = shiftud(temp1,v_values(v),0);
    for q = 0:q_max
      % R+q cannot exceed the number of sub-blocks
      if R + q > Number_of_Blocks-1 break
      end

      % FFT of the (R+q)'th block of S2
      temp2 = fftshift(fft([S2(1+(R+q)*N1 : N2 + N1*(R+q));...
                  zeros(N2,1)]));

      % Multiplies temp1 & temp2, FFTs the product, then adds to
        % previous values for the same value of q (but different R)
      temp(:,q+1) = temp(:,q+1) + ...
                          abs(fftshift(fft(temp1.*conj(temp2))));
    end
  end

  % Each value of q was used a different # of times, so they must be
  % scaled properly.
  for q_index = 1:q_max+1
    temp(:,q_index) = temp(:,q_index) / divisors(q_index);
  end
```

```
    % If combination of current v and any q provides a greater value
    % than the previous max, then remember m, Q, & V.
   if max(max(temp))>x
      x = max(max(temp));
      [m Q] = find(temp == max(max(temp)));

      % Must do this since q starts at 0, but Matlab doesn't allow for
      % zero indexing.
      Q = Q - 1;
      V = v_values(v);
   end
end

% Coarse estimate of TDOA (in # of samples)
TDOA_Coarse = Q * N1 + (-N2+1 + m);

% Coarse estimate of FDOA (in Freq Bin #)
FDOA_Coarse = V/N1*N;



% The following 3 lines can be used to display the coarse estimates,
% if desired.

%disp(['The coarse TDOA estimate is: ', num2str(TDOA_Coarse),...
%       ' samples.']);
%disp(['The coarse FDOA estimate is: ', num2str(FDOA_Coarse/N),...
%      ' (digital frequency).']);



% ************************************************************************
% FINE MODE computations.
% ************************************************************************

S2 = conj(S2);          % S2 is conjugated in basic CAF definition


% Vector of freq "bins" to use (DON'T have to be integers!!)
k_val = FDOA_Coarse-10 : FDOA_Coarse+10;

% Vectors of TDOAs to use (must be integers)
tau_val = TDOA_Coarse-10 : TDOA_Coarse+10;

done = 0;
multiple = 1;
decimal = 0;
while ~done     % Fine mode iterations continue until user is done.

   % Initialize to make later computations faster
   amb(length(k_val),length(tau_val))=0;
    Ntemp = N * multiple;
    for k = 1:length(k_val)     % Must loop through all values of k
```

```
      % Vector of complex exponentials that will be used
     exponents = exp(-j*2*pi*k_val(k)/Ntemp*(0:Ntemp-1)');

   % Must loop through all potential TDOAs
     for t = 1:length(tau_val)

       % S2 is shifted "tau" samples
       S2temp = shiftud(S2,tau_val(t),0);

       % Definition of CAF summation
       temp = abs(sum(S1.*S2temp.*exponents));

       % Save CAF magnitude for the values of k & t
      amb(k,t)=temp;
    end
  end

  [k, t]=find(amb==max(max(amb)));     % Find the peak of the CAF matrix


  TDOA = tau_val(t);  % TDOA and FDOA associated with the peak of the
 FDOA = k_val(k);     % CAF plane.  These represent the final TDOA
                      % & FDOA estimates.


  % The results are displayed.
  disp(' ');disp(' ');disp(' ');
  disp(['The TDOA is ', num2str(TDOA/multiple), ' samples']);
 disp(['         or ', num2str(TDOA/(multiple*fs)), ' seconds.']);
 disp(' ');
 disp(['The resolution is ', num2str(0.5/...
                                    (multiple*fs)),' seconds.']);
  disp(' ');disp(' ');

  disp(['The FDOA is ', num2str(FDOA/N),...
                                 ' in digital frequency (k/N)']);
  disp(['         or ', num2str(FDOA/N*fs), ' Hz.']); disp(' ');
 disp(['The resolution is ', num2str(0.5*...
                                    (10^decimal)/N*fs), ' Hz.']);
  disp(' ');disp(' ');disp(' ');


 % If the signal length exceeds 524288 elements, max processing
 % capability has been achieved, and the user will not be given
 % the option of refining TDOA any further.
  if Ntemp >= 2^19
    disp('Maximum TDOA processing capability has been achieved.')
    doneT = 1;
 else doneT = 0;
 end

%    % User chooses whether to compute more accurate TDOA &/or
```

```
%    % FDOA, or to stop fine mode computations.
%    disp('Do you desire a solution with finer resolution?');
%    disp('Select one of the following:'); disp(' ');
%
%    if ~doneT
%       disp('1.  Finer resolution for TDOA.');
%    else disp(' ');
%    end
%
%    disp('2.  Finer resolution for FDOA.');
%
%    if ~doneT
%       disp('3.  Finer resolution for both TDOA and FDOA.');
%    else disp(' ');
%    end
%
%    disp('4.  The TDOA and FDOA resolutions are fine enough.');
%    disp(' ');
%    choice = input('What is your selection?  ');

choice= ~(TDOAold==TDOA) + ~(FDOAold==FDOA)*2;

switch choice

  % TDOA is refined by resampling the signals at twice the
  % previous sampling rate.  Increases resolution two-fold.
  case 1
    if ~doneT
      multiple = multiple*2;
       S1 = interp(S1, 2);
     S2 = interp(S2, 2);
      tau_val = TDOA*2 - 1 : TDOA*2 + 1;
    else done = 1;
    end
    %clc;

  % FDOA resolution is improved by a factor of 10.
  case 2
    decimal = decimal - 1;
    k_val = FDOA - 5*10^decimal : 10^decimal : FDOA + 5*10^decimal;
    %clc;

  % Both TDOA and FDOA resolutions are improved.
  case 3
    if ~doneT
      multiple = multiple*2;
       S1 = interp(S1, 2);
     S2 = interp(S2, 2);
      tau_val = TDOA*2 - 1 : TDOA*2 + 1;

      decimal = decimal - 1;
       k_val = FDOA - 5*10^decimal : 10^decimal : FDOA + ...
                                                  5*10^decimal;
```

```
    else done = 1;
    end
    %clc;
  otherwise
    done = 1;
  end

  if done
    disp(' ');disp(' '); disp('TDOA & FDOA estimation complete.');
  end
end


% % If user wants to see the CAF surface graphically, a call to
% % CAF_peak is made.
% disp(' ');%disp(' ');disp(' ');
% choice = input...
%    ('Would you like to see the CAF peak graphically (Y or N)?  ','s');
% choice = upper(choice);
%
% switch choice
% case 'Y'
%    intp=4;
%    caf_peak(S1_orig, S2_orig, floor(TDOA/multiple) - 50, ...
%       floor(TDOA/multiple) + 50, (FDOA-20)/N, (FDOA+20)/N, fs,intp);
% end

if display_CAF_peak  %display CAF surface graphically by calling CAF_peak.m
   intp=4;
  caf_peak(S1_orig, S2_orig, floor(TDOA/multiple) - 50, ...
    floor(TDOA/multiple) + 50, (FDOA-20)/N, (FDOA+20)/N, fs,intp);
end


TDOA = TDOA/(multiple*fs);          % Returns TDOA in seconds.
FDOA = FDOA/N*fs;                    % Returns FDOA in Hertz.
%disp('Program Complete.');
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] David Adamy, *EW101: A First Course in Electronic Warfare*, Artech House, 2001.

[2] Herschel H. Loomis, Jr., "Geolocation of Electromagnetic Emitters," Technical Report No. NPS-EC-00-003, Naval Postgraduate School, Monterey, CA, November 1999 (last revised October 2007).

[3] Seymour Stein, "Algorithms for Ambiguity Function Processing," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-29, no. 3, pp. 588-599, June 1981.

[4] Robert J. Ulman and Evaggelos Geraniotis, "Wideband TDOA/FDOA Processing Using Summation of Short-Time CAF's," IEEE Transactions On Signal Processing, vol. 47, No. 12, December 1999.

[5] Joe J. Johnson, "Implementing The Cross Ambiguity Function And Generating Geometry-Specific Signals," Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2001.

[6] Glenn D. Hartwell, "Improved Geo-Spatial Resolution Using A Modified Approach To The Complex Ambiguity Function (CAF)," Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2005.

[7] B. Hofman-Wellenhof, H. Lichtenegger, and J. Collins, *GPS: Theory and Practice*, 4th revised edition, Springer-Verlag Wien, New York, 1992, 1993, 1994, and 1997.

[8] Aeronautics and Space Engineering Board, National Research Council, L. Adams (Chair), *The Global Positioning System: A Shared National Asset: Recommendations for Technical Improvements and Enhancements*, National Academy Press, Washington, DC, 1995.

[9] J. J. Spilker Jr., "Satellite Constellation and Geometric Dilution of Precision," American Institute of Aeronautics and Astronautics, 1994.) (Reprinted in B. Parkinson and J. Spilker, *Global Positioning System: Theory and Applications, Volume I*, American Institute of Aeronautics and Astronautics, Inc., Washington, 1996.)

[10] B. Parkinson, "GPS Error Analysis," American Institute of Aeronautics and Astronautics, 1994. (Reprinted in B. Parkinson and J. Spilker, *Global Positioning System: Theory and Applications, Volume I*, American Institute of Aeronautics and Astronautics, Inc., Washington, 1996, pp. 469-483).

[11] J. Farrell & M. Barth, *The Global Positioning System & Inertial Navigation*, McGraw-Hill, New York, 1998.

[12] Wikipedia, http://en.wikipedia.org/wiki/Total_electron_content, accessed April 28, 2009.

[13] Jet Propulsion Laboratory, Real-Time and Daily Ionospheric Maps, http://iono.jpl.nasa.gov//latest_rti_global.html, accessed April 28, 2009.

[14] J. J. Spilker Jr., "GPS Signal Structure and Theoretical Performance," American Institute of Aeronautics and Astronautics, 1994. (Reprinted in B. Parkinson and J. Spilker, *Global Positioning System: Theory and Applications, Volume I*, American Institute of Aeronautics and Astronautics, Inc., Washington, 1996, pp. 57-109).

[15] Robin A. Dillard and George M. Dillard, *Detectability of Spread-Spectrum Signals*, Artech House, 1989.

[16] D. Streight, "Maximum Likelihood Estimators for the Time and Frequency Differences of Arrival of Cyclostationary Digital Communications Signals," PhD Dissertation, Naval Postgraduate School, Monterey, CA, June 1999.

[17] Subbarayan Pasupathy, "Minimal Shift Keying: A Spectrally Efficient Modulation," IEEE Communications Magazine, July 1979.

[18] Greg Rawlins, David Sorrells, and Richard Harlan, "Using an IQ Data to RF Power Transmitter to Realize a Highly-Efficient Transmit Chain for Current and Next-Generation Mobile Handsets," Proceedings of the 38th European Microwave Conference, October 2008.

[19] S. Haykin and M. Moher, *Introduction to Analog and Digital Communications, 2nd Edition*, John Wiley & Sons, Inc., 2007.

[20] Naval Postgraduate School EC3510 class notes.

[21] Tri Ha, *Digital Communication, Principles and Practice*, to be published.

[22] Charles W. Therrien and Murali Tummala, *Probability for Electrical and Computer Engineers*, CRC Press, LLC, 2004.

[23] Peyton Z. Peebles, Jr., *Probability, Random Variables and Random Signal Principles*, 4th Ed., McGraw-Hill, 2001.

[24] Bernard Sklar, *Digital Communications*, 2nd Ed., Prentice-Hall, 2001.

[25] Anthony D. Whalen, *Detection of Signals in Noise*, Academic Press, Inc., 1971.

[26] Roberto Cristi, *Modern Digital Signal Processing*, Thomson Brooks/Cole, 2004.

[27] Charles W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice-Hall, 1992.

[28] James Stewart, *Calculus Early Transcendentals, 6e*, Thomson Brooks/Cole, 2008.

[29] MATLAB® Product Help function, *xcorr*, MATLAB version 7.7.0.471 (R2008b).

[30] Robert C. Dixon, *Spread Spectrum Systems*, 2nd Ed., John Wiley & Sons, 1984.

[31] Elle Zimmerman (Ed.), *Mathematics Tables*, Mathematics Department, Naval Postgraduate School, 1998.

[32] Jeffrey H. Reed, *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall PTR, 2002.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Ruth H. Hooker Library
   Naval Research Laboratory
   Washington, DC