2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

FAST ENVELOPE CORRELATION
FOR
PASSIVE RANGING

by

Frank J. Mika

September, 1991

Thesis Advisor: Ralph Hippenstiel

Approved for public release; distribution is unlimited.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS	
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			
4 PERFORMING ORGANIZATION REPORT NUMBER(S)		5 MONITORING ORGANIZATION REPORT NUMBER(S)	
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL *(If applicable)* EC	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000		7b ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000	
8a NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c ADDRESS *(City, State, and ZIP Code)*		10 SOURCE OF FUNDING NUMBERS	

Program Element No	Project No	Task No	Work Unit Accession Number

11 TITLE *(Include Security Classification)*
FAST ENVELOPE CORRELATION FOR PASSIVE RANGING

12 PERSONAL AUTHOR(S) Mika, Frank Jude

13a TYPE OF REPORT	13b. TIME COVERED	14 DATE OF REPORT *(year, month, day)*	15 PAGE COUNT
Master's Thesis	From To	1991, September	113

16 SUPPLEMENTARY NOTATION
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.

17 COSATI CODES

FIELD	GROUP	SUBGROUP

18. SUBJECT TERMS *(continue on reverse if necessary and identify by block number)*
fourier transform; correlation; frequency analysis; signal processing

19 ABSTRACT *(continue on reverse if necessary and identify by block number)*

Application of classic triangulation methods will allow the location of a radar to be determined by passive sensors. Through the use of modern digital signal processing techniques this estimate can be made in a simpler fashion using a conventional receiver.

In this thesis a technique is developed for time difference of arrival (TDOA) estimation using a frequency domain based correlation detector driven by an envelope detector. Time lag boundaries are defined on the output of the correlator. A fixed detection threshold is calculated to permit constant false alarm rate (CFAR) detection. The performance of the correlation detector is plotted as a receiver operating characteristic (ROC) curve as a function of signal to noise ratio (SNR). An interactive MATLAB software program is provided to perform either spectral domain or time domain based correlation.

Spectral domain based correlation uses the Fast Fourier Transform (FFT). Implicit with the use of the FFT are finite arithmetic internal processing errors which are modeled as independent uncorrelated noise sources. A method is presented to account for SNR degradation at the output of the FFT.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS REPORT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Hippenstiel, R.D.	22b TELEPHONE *(Include Area code)* 408-646-2633	22c OFFICE SYMBOL EC/Hi

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Fast Envelope Correlation for Passive Ranging

by

Frank J. Mika
Civilian, United States Air Force
B.S.E.E., University of Miami

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: Frank J. Mika
Frank J. Mika

Approved by: Ralph D. Hippenstiel
Ralph D. Hippenstiel, Thesis Advisor

Harold A. Titus
Harold A. Titus, Second Reader

Michael A. Morgan
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Application of classic triangulation methods will allow the location of a radar to be determined by passive sensors. Through the use of modern digital signal processing techniques this estimate can be made in a simpler fashion using a conventional receiver.

In this thesis a technique is developed for time difference of arrival (TDOA) estimation using a frequency domain based correlation detector driven by an envelope detector. Time lag boundaries are defined on the output of the correlator. A fixed detection threshold is calculated to permit constant false alarm rate (CFAR) detection. The performance of the correlation detector is plotted as a receiver operating characteristic (ROC) curve as a function of signal to noise ratio (SNR). An interactive MATLAB software program is provided to perform either spectral domain or time domain based correlation.

Spectral domain based correlation uses the Fast Fourier Transform (FFT). Implicit with the use of the FFT are finite arithmetic internal processing errors which are modeled as independent uncorrelated noise sources. A method is presented to account for SNR degradation at the output of the FFT.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I want to thank Professor Hippenstiel for his support and guidance during the preparation of this thesis. He gave many extra hours of his time teaching me communication theory.

I wish to thank my parents, Walter and Irene Mika who always support me. My father's natural curiosity in electronics started me in engineering.

I. INTRODUCTION

A. BACKGROUND

Interception of radar transmissions by passive sensors is one of the responsibilities of electronic intelligence. Analysis of the signal of a radar can provide an estimate of the distance and direction of an emitter relative to an intercepting receiver. Knowing the location is an important consideration in the evaluation of strategic and tactical capabilities of the radar.

Triangulation is a traditional technique where the angle of arrival of a radar signal at two spatially separated receivers provides an estimate of the transmitters location. With the advent of the computer, digital signal processing algorithms can be implemented that provide emitter range information and hence allow localization in a sequential sense. One of these processing algorithms is called the time difference of arrival (TDOA) method.

B. TIME DIFFERENCE OF ARRIVAL

The TDOA algorithm relies on two bistatic passive sensors attempting to receive the transmission of an active radar. Reception by the two sensors is preferentially done in the main lobe of the transmitting radar, but due to the nonsynchronous nature of the task, reception through one of the sidelobes of the transmitting radar is expected with a resulting decrease in received signal power.

Given a transmitted radar pulse f(t), two physically separated passive sensors will most likely receive this pulse at times t_1 and $t_1+\tau$. The time difference τ is proportional to their differential distance relative to the emitter. This time difference of arrival can be used in the estimation of the differential range of the emitter to the receivers. The reception of the first pulse in a pulse burst by the sensors is paramount in performing the TDOA estimate.

The concept of measuring the TDOA of a radar pulse between two separated receivers can be modeled by a hyperbola. The two sensors whose positions are known are located at the two focal points of the hyperbola. A transmitting radar is located in the area surrounding the two sensors. Figure 1 shows this model.

A hyperbola is the locus of points in which the distance from any one point on the hyperbola to one focus differs by a constant amount from the distance to the other focus. This differential distance is proportional to a difference in arrival time τ, of a radar pulse as detected between the two sensors. Once τ is measured the hyperbola can be drawn. If the sensors (moving aircraft or nongeostationary satellites) repeat the differential arrival time measurements at several time intervals, a series of hyperbolas will be generated, fixing the location of the transmitting radar. Alternately, the angle of arrival at either sensor can be used to define the position of the transmitter on the hyperbola. A good introduction into the use of the hyperbola in navigation is found in [Ref. 1:p. 27].

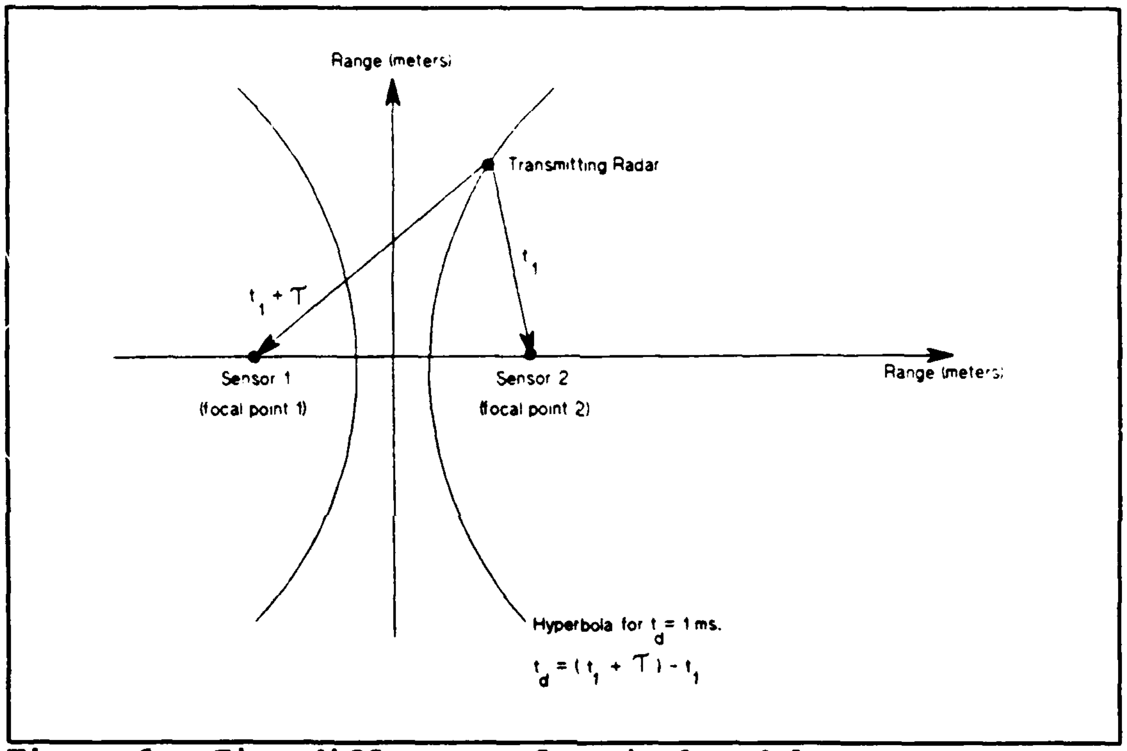


Figure 1. Time difference of arrival model.

This discussion assumes that the radar signal received by the two sensors is induced by the direct wave from the transmitter. The effects of refraction and reflection on propagating radar waves are not covered here.

The purpose of this thesis is to develop and test an algorithm to estimate the differential arrival time τ, of a pulsed radar signal collected by two passive sensors. If the pulse burst is collected in the time domain, time correlation can be used to generate a TDOA estimate. The Fast Fourier Transform (FFT) can also be used to perform the correlation of the two received signals (i.e., fast correlation).

Additive noise n(t), superimposed onto a transmitted radar signal f(t) by the environment and the radar receiver, creates a composite signal s(t) at the receiver

$$s(t) = f(t) + n(t) \ . \qquad (1)$$

The use of the Fourier Transform for TDOA estimation is attractive because of the processing gain (PG) a radar signal receives in being transformed from the time domain to the frequency domain during the detection process. FFT processing gain for real valued signals can be approximated by

$$PG\ (dB) = (\log_2[\textit{number points in data record}] - 1)\cdot 3 \ . \qquad (2)$$

See for example [Ref. 2:p. 33]. The signal to noise ratio (SNR) at the output of a FFT (SNR_{OUT}) is a function of the PG and the SNR at the input to the FFT (SNR_{IN}). This relation is given by

$$SNR_{OUT} = SNR_{IN} + PG \ . \tag{3}$$

The corruptive effects of additive noise are reduced by the transformation, allowing the frequency domain detection of the signal.

C. RADAR TYPES

The TDOA algorithm is primarily applied to pulsed radars. The continuous wave (CW) radar provides target radial velocity through the Doppler shift of its return. The CW radar does not use a pulsed transmission. Therefore, the TDOA algorithm cannot not be directly applied to the reception of CW signals.

Pulsed radars can be divided into two broad categories, ordinary low pulse repetition frequency (PRF) pulsed radars and the pulse Doppler radars.

Low PRF radars can yield unambiguous range information, while their radar returns do not give target velocity information directly. Generally because of the long duty cycle of the low PRF radar, the probability of receiving all radar pulses in a pulse burst by two passive sensors is great. The TDOA algorithm can then estimate the differential distance to the emitter using the low PRF burst.

The pulse Doppler radar uses coherent transmission and reception with a moderately high PRF [Ref. 3:p. 17.1]. It gives both range and radial velocity information about a target, though not with the same accuracy of the ordinary pulsed radar or CW radar respectively. In the pulse Doppler radar, as in all non-bistatic radars, the receiver must be turned off while the transmitter is on. Because of this, a range ambiguity exists. Echoes having delay times equal to an integral multiple of the PRF will be undetected. Target echoes that remain within this blanked time interval will remain undetected. A second source of range ambiguity exists. For a target located at a distance r_1 from the tracking radar, all other targets in the main beam of the radar a distance

$$\begin{gathered} r_1,\ 2r_1,\ 3r_1,\ 4r_1,\ \ldots,\ kr_1 \\ \textit{where } k = \textit{integer} \ , \end{gathered} \tag{4}$$

will appear to be approximately the same distance from the radar.

Pulse Doppler radars can be divided into a low, medium and high PRF class. Both the low and medium PRF pulse Doppler radar transmissions can be passively intercepted and a relatively simple TDOA estimate can be formed.

Reception of each pulse in a pulse burst from a high PRF pulse Doppler radar by two independent passive sensors can be difficult. If either sensor misses the first pulse, the TDOA estimate will be in error. This is true for any radar, but has a higher likelihood of occurring in the high PRF radars due to the shorter pulse widths used. Should the pulse burst be coded so that a nonuniform time pulse sequence is generated (i.e., staggered PRF), then missed pulses by either sensor will not seriously degrade the validity of the TDOA estimate.

D. FAST FOURIER TRANSFORM OUTPUT SIGNAL TO NOISE RATIO

The Fourier transform is a computationally intensive mathematical operation. When implemented on an FFT processor using fixed point arithmetic, internal processing errors are generated. These errors are a function of the transform size, and the number of bits used internally in the FFT processor.

The primary building block of the FFT is the butterfly algorithm. There are three dominant processing errors that occur within the butterfly. They are scaling, truncation and trigonometric errors. These can be modeled as independent, uncorrelated noise sources. The cumulative effect of these noise sources is to reduce the potential SNR at the output of the butterfly, and therefore, the FFT processor.

This thesis will also examine the effects of noise caused by FFT processing, and will present a method to account for the degradation of the output SNR due to finite arithmetic. Certain FFT implementations will keep the signal energy larger with respect to the three noise power sources than others.

II. TIME DOMAIN BASED CORRELATION

A. BACKGROUND

The correlation function R(τ) measures the degree of similarity between two signals s(t) and g(t) and is described by

$$R(\tau) = limit_{T \to \infty} \frac{1}{2T}\int_{-T}^{+T} s(t)g(\tau + t)dt \qquad (5)$$

where $g(\tau + t) = g(t)$ displaced by the shift τ .

Correlation can be performed in a radar receiver between a target return corrupted by additive noise, and a replica of the transmitted signal maintained by the radar. This replica is designed into the frequency response of the matched filter of the radar receiver. [Ref. 4:p. 373].

Correlating two signals produces an output that does not resemble either of the two input signals. The shape of the received waveform is destroyed during the correlation process. The useful item of correlation is an amplitude peak, where the location along the time axis of this peak indicates the amount of time that one signal lags the other.

If the correlation process occurs between a signal and a delayed replica of itself, the process is called autocorrelation. If two dissimilar signals are correlated, the process is called crosscorrelation.

B. AUTOCORRELATION

Autocorrelation is the process by which two identical signals, one a delayed replica of the other, are correlated. The autocorrelation function of a discrete time sequence x(i) is defined by [Ref. 5:p. 556]

$$r_{xx}(l) = \sum_{i=0}^{N-1-|l|} x(i)x(i+l) \tag{6}$$

$$\begin{aligned} \textit{where } l &= \textit{shift operator} \\ N &= \textit{number of data points in } x(n) \quad . \end{aligned}$$

This function is not scaled with respect to the shift operator nor the number of data points. Figure 2(a) shows the effect of autocorrelation on a burst of 13 pulses of unit amplitude. Normalized autocorrelation produces a triangular envelope waveform of height one at a time lag of zero. The location of this peak indicates that the time shift for maximum overlap between the pulse burst and a shifted replica of itself is zero.

For illustration, a Barker coded sequence of pulses of length 13 is also autocorrelated in Figure 2(b). The Barker code is given by +++++--++-+-+, where + and - could represent 0 and π radians carrier phase shift, respectively. Barker coding of a pulse sequence constructively modifies the correlation output to magnify the time lag peak. The zero lag point has the greatest amplitude. Adjacent peaks are significantly reduced in amplitude. Because of this reduction,

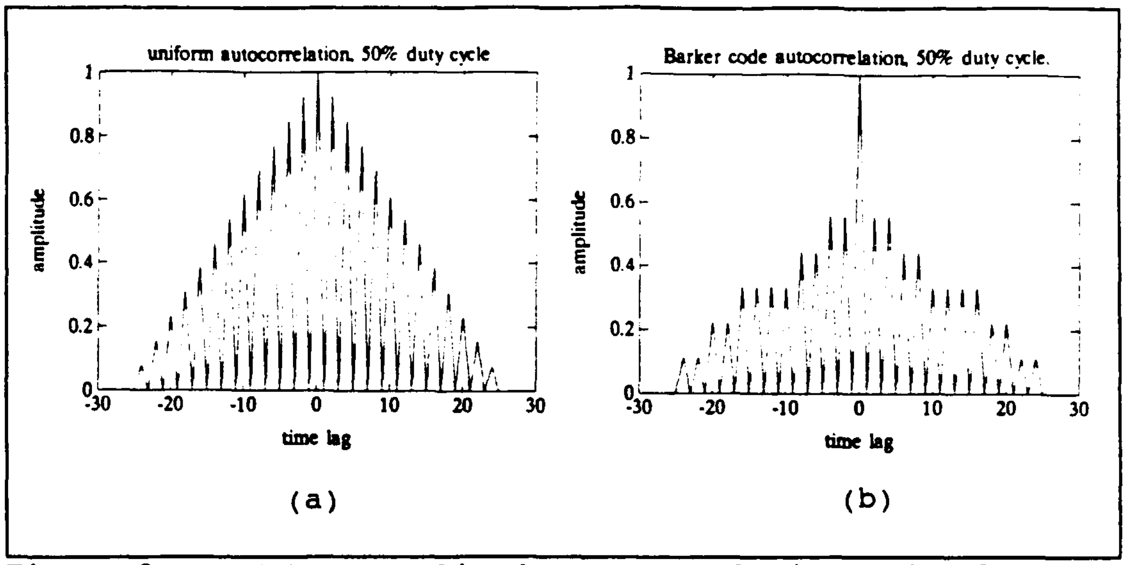


Figure 2. (a) Normalized autocorrelation of 13 equal amplitude pulses. (b) Normalized autocorrelation of 13 Barker coded pulses.

if noise is added to the received signal, a Barker coded radar pulse burst will have a correlation peak that is not as easily confused with adjacent peaks.

Now consider the autocorrelation of a random process. An independent, identically distributed zero mean Gaussian sequence with variance of one is shown in Figure 3. The autocorrelation of this sequence produces a peak at zero time lag. There is no strong correlation of the Gaussian noise except at zero time lag. The autocorrelation of Rayleigh distributed noise is shown in Figure 4.

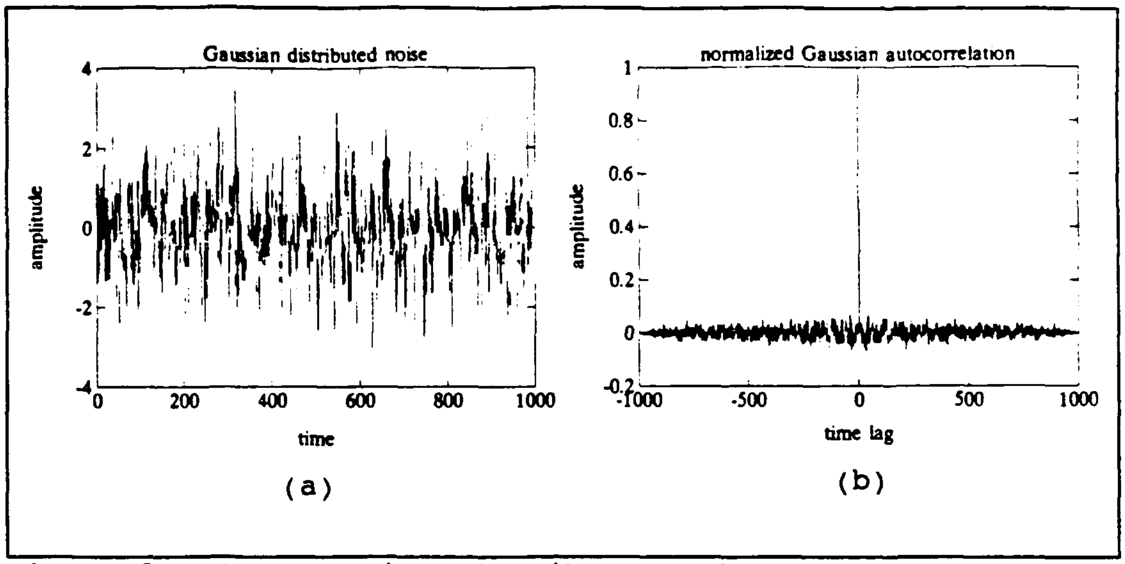


Figure 3. (a) Gaussian distributed noise sequence. (b) Normalized autocorrelation of Gaussian noise.

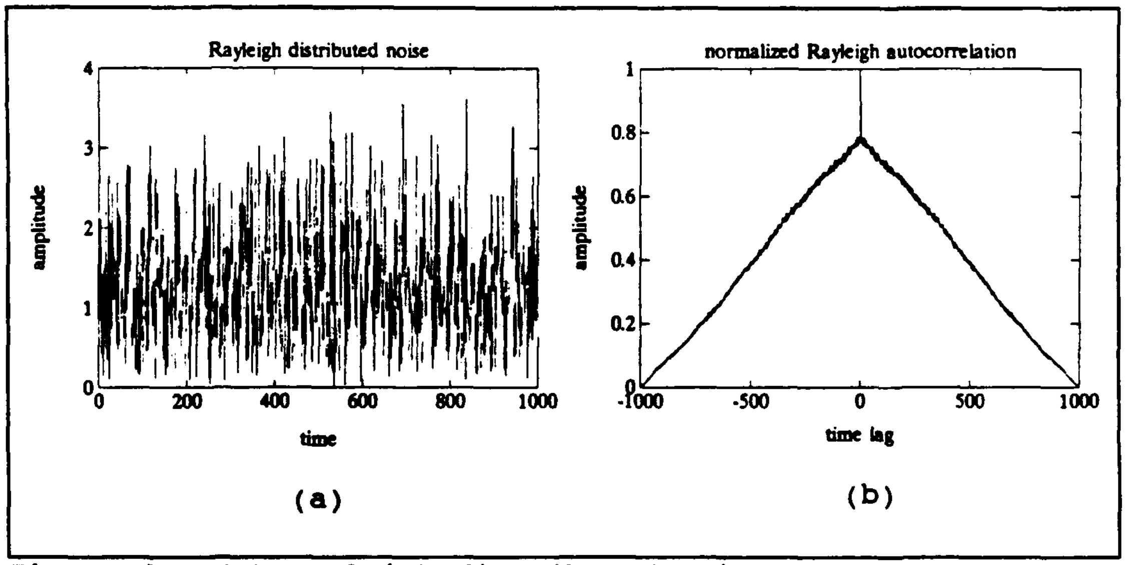


Figure 4. (a) Rayleigh distributed noise sequence. (b) Normalized autocorrelation of Rayleigh noise.

Rayleigh distributed noise has a mean (dc) value of

$$mean\ value = \sigma\sqrt{\frac{\pi}{2}} \qquad (7)$$

where σ = standard deviation .

The autocorrelation function of Rayleigh noise has twc components. A triangular envelope caused by the dc component of the Rayleigh noise, and a weighted delta function at the apex of the triangle which indicates the time shift of maximum overlap of the two sequences. Clearly maximum overlap occurs at a time shift of zero for any autocorrelation function.

C. CROSSCORRELATION

Crosscorrelation is the process wherein two different signals are correlated. The crosscorrelation function of a discrete time sequence is defined by

$$r_{xy}(l) = \sum_{i=0}^{N-1-|l|} x(i)y(i+l) \qquad (8)$$

where $x(i)$ and $y(i)$ are two different sequences.

This function is not scaled with respect to the shift operator nor the number of data points. Figure 5 shows the crosscorrelation of two uncorrelated Gaussian noise sequences and two uncorrelated Rayleigh noise sequences. Clearly, no dominant amplitude peak is visible in either crosscorrelation plot. Lack of a dominant peak indicates there is no strong correlation between the two sets of noise sequences.

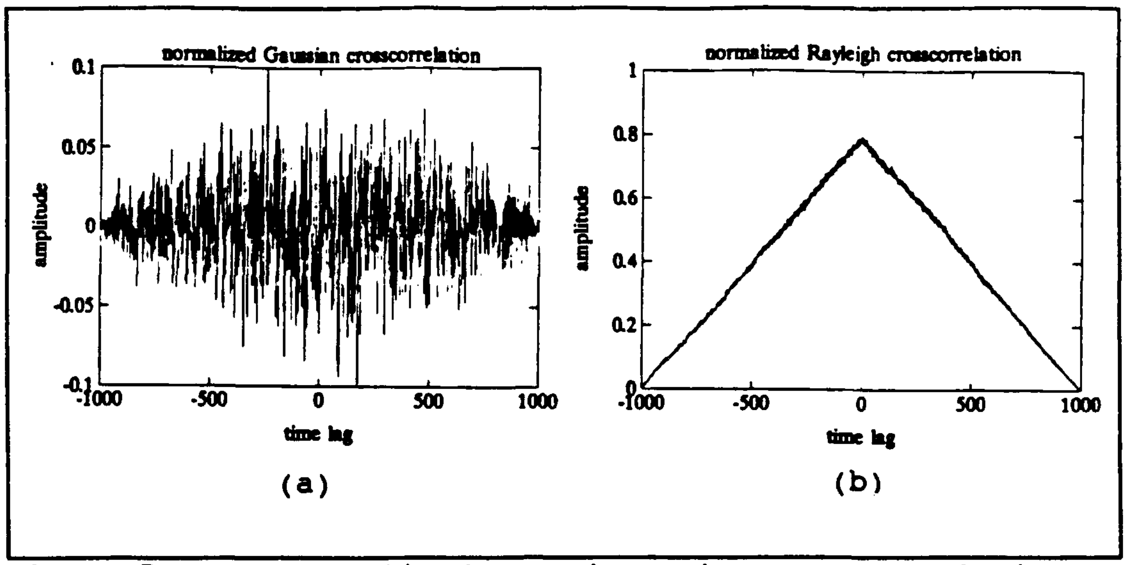


Figure 5. (a) Normalized Gaussian noise crosscorrelation. (b) Normalized Rayleigh noise crosscorrelation.

D. CORRELATION COEFFICIENT

A measure of the linear dependence between any two zero mean stationary time functions x(t) and y(t) is the correlation coefficient function and is defined [Ref. 6:p. 48] as

$$\rho_{xy}(\tau) = \frac{r_{xy}(\tau)}{\sqrt{r_{xx}(0)}\sqrt{r_{yy}(0)}} \tag{9}$$

$$\textit{where } |\rho_{xy}(\tau)| \le 1 \textit{ for all } \tau.$$

This function is the crosscorrelation function normalized by the square root of the maximum values of the individual autocorrelation functions. The normalizing factor is not a lag dependent quantity.

Given two time series x(t) and y(t) where a is any positive number

$$\begin{aligned} \rho_{xy}(\tau) &= 1 \ \textit{if} \ x(t) = a \cdot y(t) \\ \rho_{xy}(\tau) &= -1 \ \textit{if} \ x(t) = -a \cdot y(t) \\ \rho_{xy}(\tau) &= 0 \ \textit{if} \ x(t) \ \textit{and} \ y(t) \ \textit{are uncorrelated.} \end{aligned} \tag{10}$$

Another way of defining a normalized correlation coefficient for a discrete time series of finite duration is given by

$$\rho_{xy}(l) = \frac{\sum_{i=0}^{N-1-|l|} x(i)\, y(i+l)}{\sqrt{\sum_{i=0}^{N-1} x(i)^2}\sqrt{\sum_{i=0}^{N-1-|l|} y(i+l)^2}} \ . \tag{11}$$

E. SQUARE-LAW DETECTION AND CORRELATION

If x(t) is a real valued function of time, the Hilbert transform of x(t) is defined by [Ref. 6:p. 484] as

$$\begin{aligned} \tilde{x}(t) &= H\,[x(t)] \\ &= \int_{-\infty}^{\infty} \frac{x(u)}{\pi(t-u)}\, du \\ &= x(t) * \left(\frac{1}{\pi t}\right) \end{aligned} \tag{12}$$

$$\textit{where} * = \textit{convolution operation} \ .$$

The output of a square-law envelope detector u(t), can be described by

$$u(t) = x^2(t) + \tilde{x}^2(t)$$
$$\text{where } x(t) = \text{real valued time function input to square-law detector} \quad . \tag{13}$$

The correlation coefficient function of two time functions x(t) and y(t) is defined in Equation 9. The Hilbert transform of the correlation function is defined as

$$\tilde{\rho}_{xy}(\tau) = H\,[\rho_{xy}(\tau)] = \frac{\tilde{R}_{xy}(\tau)}{\sigma_x \sigma_y} \tag{14}$$

where σ_x and σ_y = standard deviation of x and y respectively .

The correlation coefficient for two square-law envelope detected signals, $\rho_{uv}(\tau)$, is defined by [Ref. 6:p. 512] as

$$\rho_{uv}(\tau) = \rho^2{}_{xy}(\tau) + \tilde{\rho}^2{}_{xy}(\tau) \tag{15}$$

where $u(t)$ = square-law detected signal of $x(t)$
$v(t)$ = square-law detected signal of $y(t)$.

The quantity $\rho_{uv}(\tau)$ produces a sharper correlation peak than $\rho_{xy}(\tau)$. This sharpening of the correlation function peak output can aid in locating the TDOA correlation peak. An example of both $\rho_{xy}(\tau)$ and $\rho_{uv}(\tau)$ is plotted in Figure 6.

F. ENVELOPE CORRELATION AND RELATED STATISTICS

In this thesis, time domain based correlation will be performed on the output of an envelope detector. An envelope detector can be easily implemented in hardware using a diode.

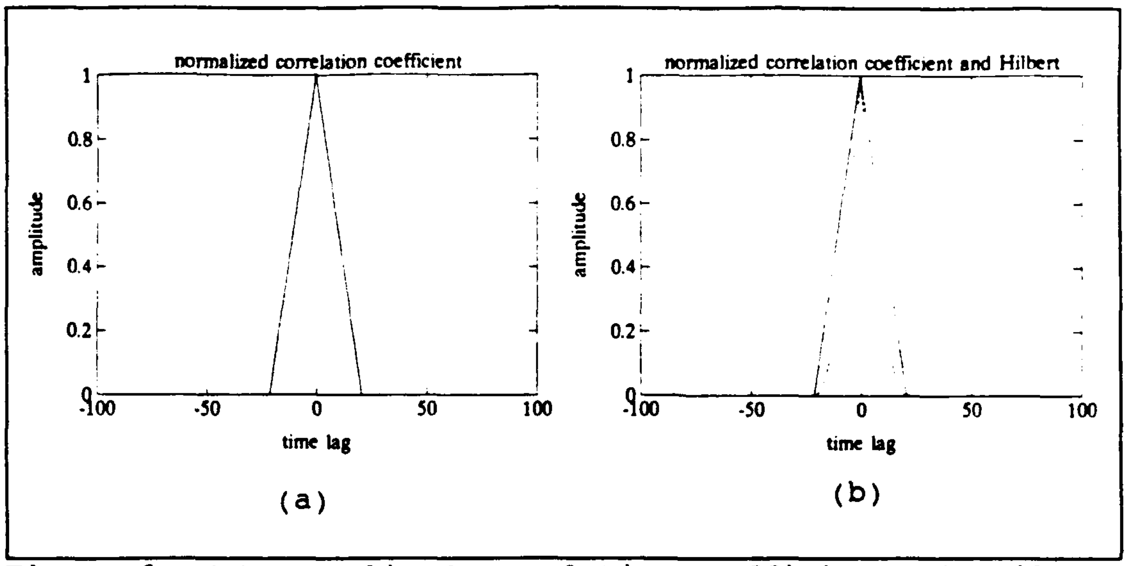


Figure 6. (a) Normalized correlation coefficient. (b) Hilbert transform of the normalized correlation coefficient (dotted line).

To derive the expected value and variance of the crosscorrelation function, two cases must be considered. These cases are noise only present, and signal plus noise present in both channels of the correlator. Under the noise only case, the statistics for the correlator output are easily derived.

We consider two real, independent identically distributed, zero mean noise series at the input to the correlator. Each series has zero mean and variance σ^2 . It is shown in Appendix A, that the expected value of the output of the crosscorrelation function $r_{xy}(\ell)$ is

$$E[r_{xy}(l)] = 0 \quad . \tag{16}$$

The variance of the output of the crosscorrelation function $r_{xy}(\ell)$ is

$$\sigma^2_{r_{xy(l)}} = (N-|l|)\,\sigma^4 \qquad (17)$$
$$\textit{where } N = \textit{number of data points} \; .$$

Since equal variances are assumed for both time series, σ^4 is the square of the variance of either noise series.

G. BLOCK CORRELATION

Two time sequences can be correlated by any of three different methods. These will be discussed below.

The first method blocks the length of each input sequence. Correlation of these blocks will provide an output with a variance given by Equation 17. The linear dependence of the correlation variance on $(N-|\ell|)$ is shown in Figure 5 for zero mean Gaussian noise. Figure 5 shows the correlated output at $\ell=0$ (zero time lag) is scaled by $(N-|0|)=N$. Correlation output at a time lag of $\pm N$ are scaled by $(N-|N|)=0$. The linear dependance of the correlation output variance on the time lag does not allow a constant detection threshold, which is needed to automatically determine the location of the correlation peak.

A second method of correlation blocks a fixed length sequence x(n) and correlates it with an arbitrarily long sequence y(n). The correlation output for this algorithm has a variance of $N\sigma^4$. Clearly, no scaling of the correlation output variance as a function of time lag would occur.

A third method is scaling applied to the output of the first technique. The correlator output is multiplied by a weighting function, $1/(N-|\ell|)^{1/2}$ to correct for the lag dependency of the variance. The selection of a constant detection threshold will be addressed in Chapter V.

III. FREQUENCY DOMAIN BASED CORRELATION

A. FOURIER TRANSFORM

Frequency domain techniques can be used to perform fast correlation rather than the time domain techniques previously discussed. The Fourier transform translates the discrete time sequence x(n) into the frequency domain .

For a finite duration time series x(n), the discrete Fourier transform (DFT) is defined by the pair of equations [Ref 5:p. 100]

$$
\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n)\; e^{-j(\frac{2\pi}{N})nk},\ 0 \le k \le N-1 \\
x(n) &= \frac{1}{N}\sum_{k=0}^{N-1} X(k)\; e^{j(\frac{2\pi}{N})nk},0 \le n \le N-1\ .
\end{aligned}
\tag{18}
$$

Equation 18 is used to transform the N discrete sample values into N frequency domain coefficients.

The following discussion is based on [Ref. 5:p. 286]. The DFT is a computationally intensive algorithm if implemented directly. The direct evaluation of the DFT requires N^2 complex multiplications. The time required for the evaluation of a DFT is proportional to N^2.

Algorithms have been designed that take advantage of the symmetry and periodicity of the DFT, reducing the amount of computation required to solve the DFT. Any of these algorithms are referred to as the FFT.

Generally, the FFT requires $(N/2)\log_2 N$ complex multiplications. This smaller computational workload becomes significant as N, the number of data points in the time series, becomes larger.

B. NORMALIZED FREQUENCY DOMAIN CORRELATION

In some radar receivers the processing of pulse sequences occurs in the frequency domain. Therefore, a frequency representation of the time domain pulses may already exist. The frequency coefficients do not necessarily need to be transformed back into the time domain to perform the crosscorrelation.

Frequency domain data can be crosscorrelated by multiplying one set of coefficients by the complex conjugate of the second set of coefficients. The product is translated back to the time domain through the use of the inverse FFT. Frequency domain based normalized crosscorrelation is described by

$$\rho_{xy}(l) = \frac{1}{N}\sum_{k=0}^{N-1} \frac{(X(k)\cdot Y(k)^*)\; e^{j(\frac{2\pi}{N})lk}}{\sqrt{\sum_{k=0}^{N-1}|X(k)|^2}\;\sqrt{\sum_{k=0}^{N-1}|Y(k)|^2}},\quad 0 \le l \le N-1$$

$$= 0 \;\; otherwise \qquad (19)$$

where $X(k)$ = FFT of the series $x(n)$
$Y(k)$ = FFT of the series $y(n)$
$*$ = conjugation
$\rho_{xy}(l)$ = normalized correlation function .

To avoid circular correlation, both time domain pulse sequences are zero padded to N_z data points

$$N_z \ge N_x + N_y - 1 \qquad (20)$$

where N_x = number of data points in $x(n)$
N_y = number of data points in $y(n)$.

To take advantage of the processing speed of FFT algorithms, N_z is made a power of two by increasing its length with additional zero padding

$$N_z = 2^m \ge N_x + N_y - 1 \qquad (21)$$

where m is an integer .

Prior to any zero padding, the dc component is removed to create zero mean pulse sequences. Both sequences, each of length N_z are then transformed to the frequency domain through the FFT and processed using the frequency domain correlation technique. A MATLAB implementation of frequency domain crosscorrelation is given in Appendix B (FreqCorr7.m).

C. TIME DIFFERENCE OF ARRIVAL (FOURIER DOMAIN)

The normalized crosscorrelation function implemented in the frequency domain (Equation 19) is used to estimate the TDOA. The crosscorrelation of two similar signals will produce a dominant peak in the output of the correlator. This peak is used as an estimate of the TDOA between the two signals. For two signals that have not been corrupted by noise, the location in time of this peak is easily determined. Pulsed signals that have been distorted by noise produce a noisy correlated output. The output noise can mask the TDOA peak in low SNR conditions. Consequently, the probability of selecting a noise sample instead of the true TDOA peak increases with decreasing SNR.

Figures 7,8 and 9 show the frequency domain based crosscorrelation of two pulse sequences as a function of decreasing SNR. A reception is simulated with a pulse burst demodulated by one receiver fed into one channel, and a pulse burst demodulated by a second receiver fed into the second channel of the correlator. Pulse burst one has a transmission delay of zero. Pulse burst two has a delay of 50 time units. Therefore, the total TDOA is 50 time units. The TDOA peak is clearly seen at the high SNR of +7dB and +3dB. At 0dB SNR, noise peaks exceed in amplitude the true TDOA peak causing an incorrect estimate if one was selected.

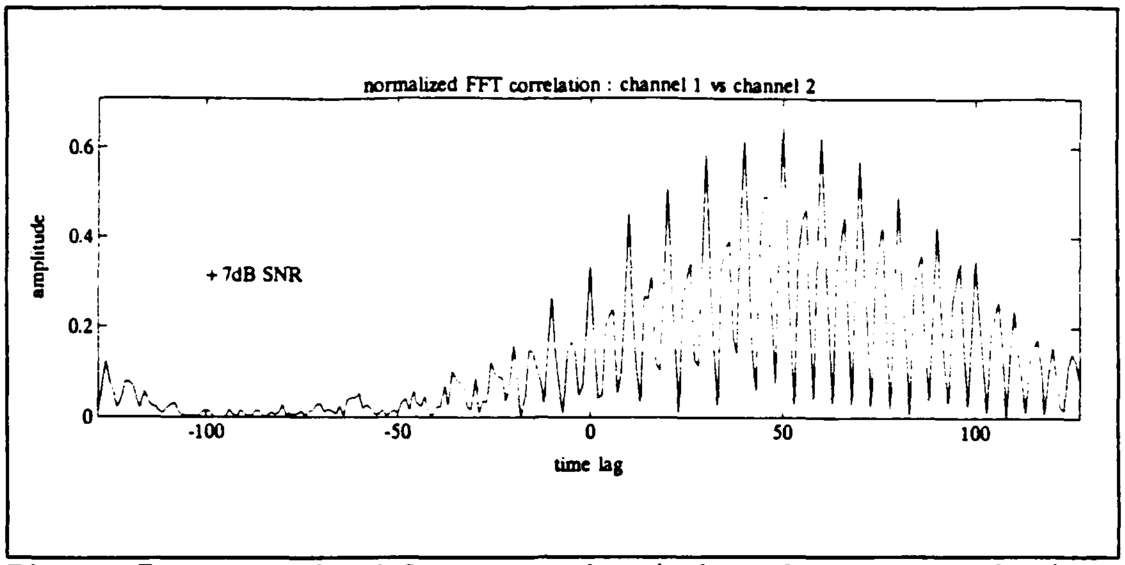


Figure 7. Normalized frequency domain based crosscorrelation. SNR +7dB.

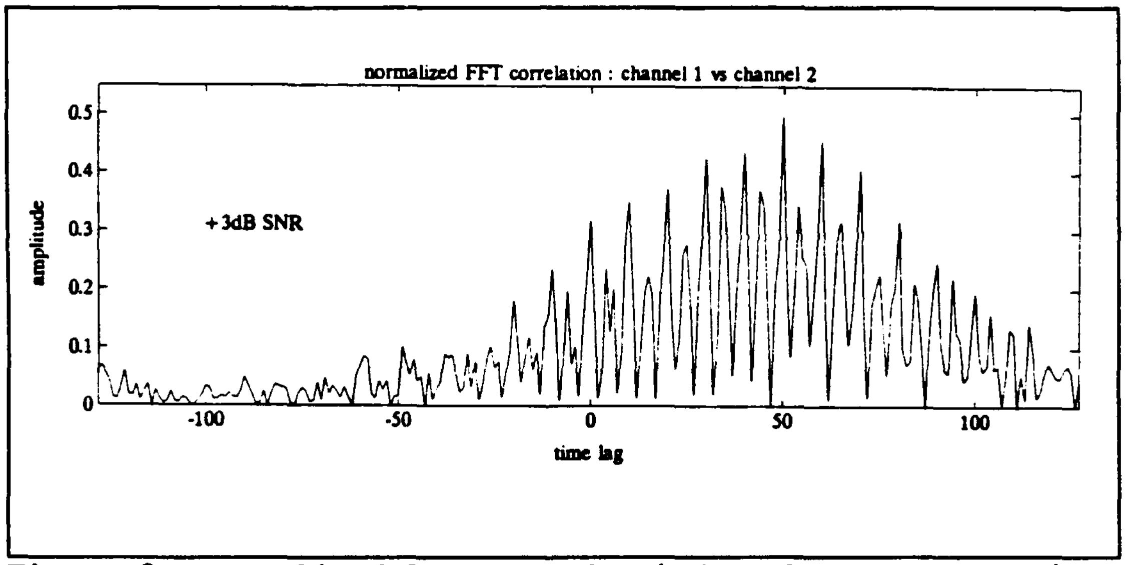


Figure 8. Normalized frequency domain based crosscorrelation. SNR +3dB.

Visually estimating the location of the peak with the maximum amplitude in the output of the correlator is a easy task. The decision rule states that the point with the maximum

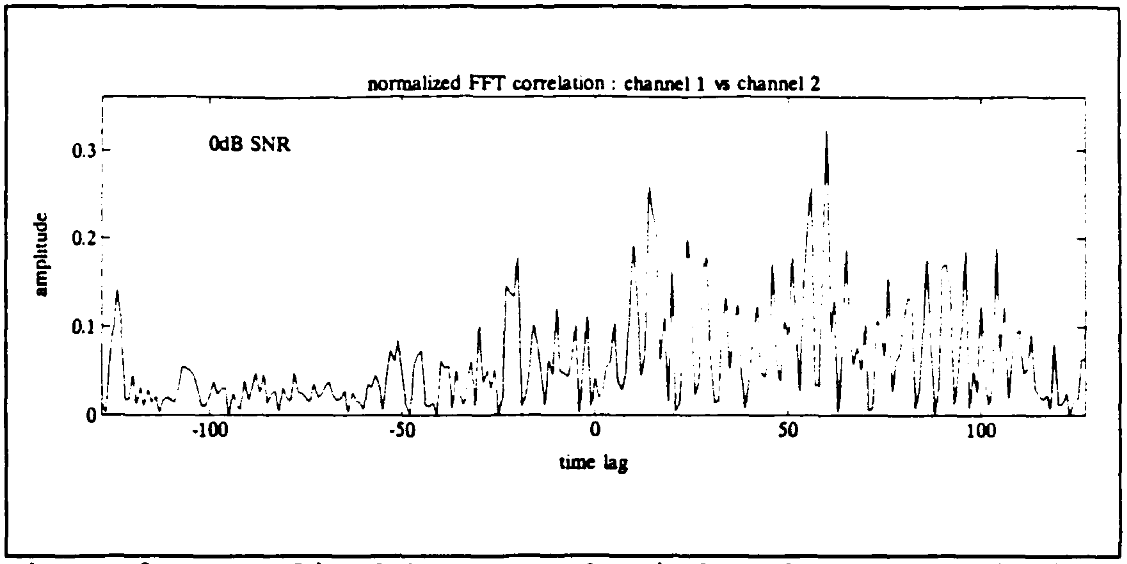


Figure 9. Normalized frequency domain based crosscorrelation. SNR 0dB.

amplitude is selected as the TDOA estimate, provided the amplitude exceeds a predetermined threshold.

When many blocks of data must be processed, visual inspection of the correlation output is not possible. Chapter V discusses the design of a threshold for automated data processing.

IV. NOISE AND SIGNALS IN RECEIVER

A. NOISE AND SIGNALS IN THE RADAR DETECTOR

The purpose of a radar receiver is to process and detect the reflected energy from a target being tracked by the radar. An I/Q demodulator is assumed in the radar receiver, located after the last intermediate frequency (IF) amplifier and prior to the video signal display. Data to be processed for TDOA can be obtained from three locations on the I/Q demodulator. These locations include the envelope output, the envelope squared output and the I/Q channel data.

White Gaussian distributed noise is assumed to be added to radar pulses during their transmission through the atmosphere and reception by the receiver. This noise contributes to the degradation of the SNR and is caused by fluctuations in the target radar cross-section and the loss of signal energy due to the propagation distance between the target and the receiver.

At typical radar frequencies the front end amplifier of the radar receiver contributes to this noise power. The noise power for a thermally noise limited receiver is defined as N_0,

$$N_0 = k\,T_{sys}\,B$$

$$\text{where } k = \text{Boltzman's constant}$$
$$B = \text{IF bandwidth}$$

$$T_{sys} = T_{ar} + T_e \tag{22}$$

$$T_{ar} = \text{receive antenna temperature}$$
$$T_e = \text{effective noise temperature of the receiver} \ .$$

The total noise is modeled as zero mean Gaussian distributed noise with a probability density function (pdf) of

$$p_X(x) = \frac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}}\,e^{\frac{-x^2}{2\sigma^2}} \ , \tag{23}$$

where σ^2 is the noise variance due to all noise sources (i.e., receiver front end, transmission noise).

For an operational radar, either a pulsed signal plus Gaussian noise or Gaussian distributed noise alone is assumed at the input of the I/Q demodulator. The probability distribution of the demodulated signal is dependant on which of the three outputs of the I/Q demodulator is used.

1. Coherent detection

If the exact frequency and phase of the received pulse burst is known, and matches the frequency and phase of the local oscillator in the receiver, then the signal can be coherently detected. Coherent detection simplifies the probability description of the signal and noise at the output of the I and Q channels. The I and Q sequences have an independent Gaussian distribution, and can be described as

$$I,Q \sim N(\{m_I(n),m_Q(n)\},\sigma^2) \tag{24}$$

where $m_I(n)$, $m_Q(n)$ = *I and Q mean values respectively*
σ^2 = *noise variance due to all noise sources .*

Both the I or Q channel can be processed using the correlation algorithm followed by threshold detection.

Due to the passive nature of signal reception and demodulation, the exact frequency and phase of the received pulse burst are not available at the receiver. Therefore, coherent detection is not feasible for TDOA estimation.

2. Envelope Squared Detection

For a zero mean Gaussian distributed noise input to the I/Q demodulator, I^2 and Q^2 each have a chi-squared distribution with one degree of freedom. Their pdf's can be described by

$$\begin{aligned} p_Y(y) &= \frac{1}{\sigma\sqrt{2\pi y}}e^{-y/2\sigma^2} && \textit{for } y \geq 0 \\ &= 0 && \textit{for } y < 0 \end{aligned} \tag{25}$$

where σ^2 = *variance of I and Q*
$y = I^2$ *or* Q^2 .

The envelope squared output is the sum of I^2 and Q^2 and is therefore chi-squared with two degrees of freedom. This pdf can be described by

$$p_Z(z) = \frac{e^{-z/2\sigma^2}}{2\sigma^2} \quad for\ z \geq 0 \tag{26}$$

$$where\ z = I^2 + Q^2 \ .$$

Envelope squared detection is attractive because the Hilbert transform can be used to enhance the TDOA peak in the correlator's output. Unfortunately in a low SNR environment the noise power will also increase as the noise contribution is squared.

3. Envelope Detection

A common form of receiver demodulation at IF frequencies is the envelope detector. For all simulations in this thesis, envelope detection will be assumed. The output of the envelope detector is the modulation envelope of the received pulse burst. For Gaussian distributed noise at the input to the I/Q demodulator, the noise at the envelope output is Rayleigh distributed with pdf given by

$$\begin{aligned} p_U(u) &= \frac{u}{\sigma^2} e^{-\frac{u^2}{2\sigma^2}}, \quad u > 0 \\ &= 0, \qquad\qquad otherwise \end{aligned} \tag{27}$$

$$\begin{aligned} where\ \sigma^2 &= variance\ of\ Gaussian\ noise \\ u &= \sqrt{z} = \sqrt{I^2 + Q^2} \ . \end{aligned}$$

For a pulsed sinusoid plus Gaussian noise at the input to the I/Q demodulator, the pdf of the envelope at the output is Rician distributed. The Rician pdf is given by

$$p_R(r) = \frac{r}{\sigma^2} e^{\frac{-(r^2+A^2)}{2\sigma^2}} I_o\left(\frac{rA}{\sigma^2}\right), \quad r \geq 0$$
$$= 0, \qquad otherwise \tag{28}$$

where A = *amplitude of received sinusoid*
σ^2 = *variance of Gaussian noise*

$$where\ I_o(z) \triangleq \frac{1}{2\pi} \int_0^{2\pi} e^{z \cos(\theta)}\, d\theta \tag{29}$$

and $I_o(z)$ = *modified Bessel function of the first kind of zero order* .

Figure 10 shows a block diagram of the I/Q demodulator and the various pdfs in the radar receiver.

B. ENVELOPE CORRELATION OF RAYLEIGH NOISE

Rayleigh distributed noise is produced at the output of the envelope detector under noise only conditions. This noise is then processed by the correlator during the TDOA estimate. In all follow on discussions and simulations, the envelope processing scheme is used.

The autocorrelation of Rayleigh noise as discussed earlier produces a large triangular bias in the output which decreases as the time lag increases. This bias is produced by the dc component of the Rayleigh distributed noise.

This triangular function contains no information relative to the TDOA estimate. The large numerical values in the bias can potentially limit the dynamic range in the FFT based correlation. A constant, detection threshold cannot be

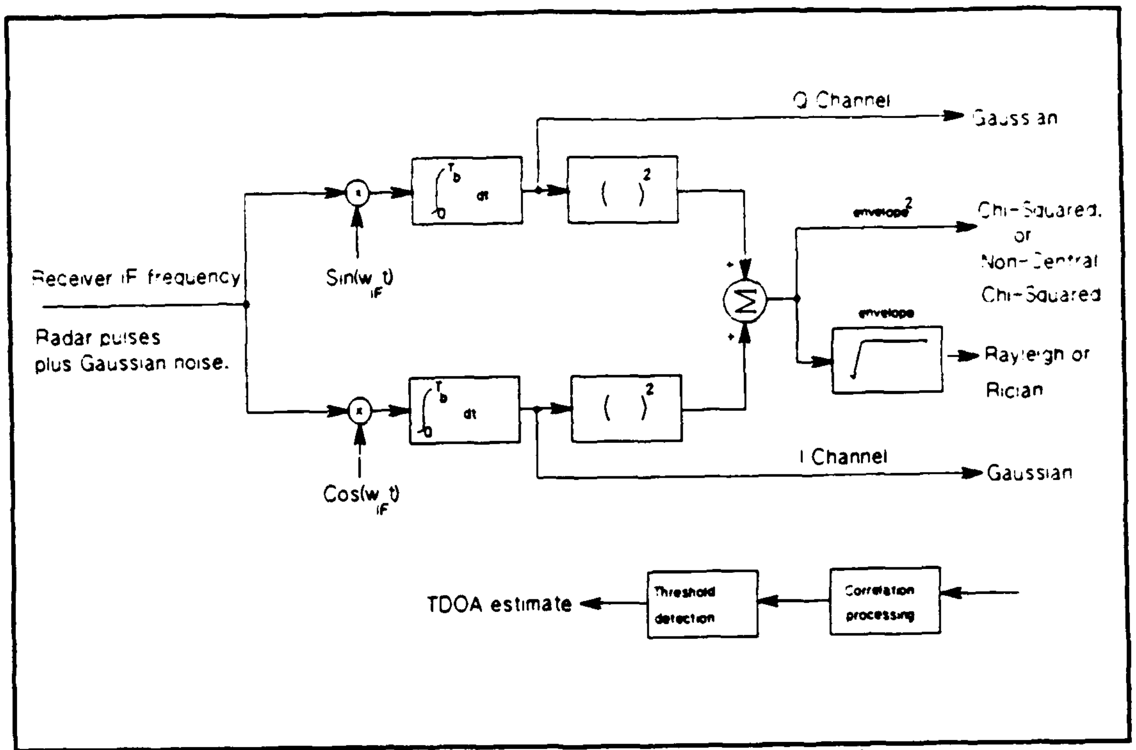


Figure 10. I/Q demodulator in the radar receiver.

implemented at the output of the correlator with this triangular function present. For these three reasons, it is advantageous to force the Rayleigh or Rician distribution to have a zero mean. This will remove the triangular bias formed by the correlation process. This modification of the received pulse burst does not degrade the TDOA estimate. For all realizations (i.e., blocks of data), the dc component of the Rayleigh noise will be removed.

1. Shifted Rayleigh Noise

Forming a Rayleigh distributed time series x(n) and subtracting its expected value produces a shifted Rayleigh sequence. This algorithm is described by

$$x_1(n) = x(n) - \sigma\sqrt{\frac{\pi}{2}} \tag{30}$$

where $x_1(n)$ = *shifted Rayleigh noise sequence*
σ = *Gaussian noise standard deviation* .

The shifted Rayleigh time series has a zero mean. The new time series and the corresponding time domain based autocorrelation function are shown in Figure 11.

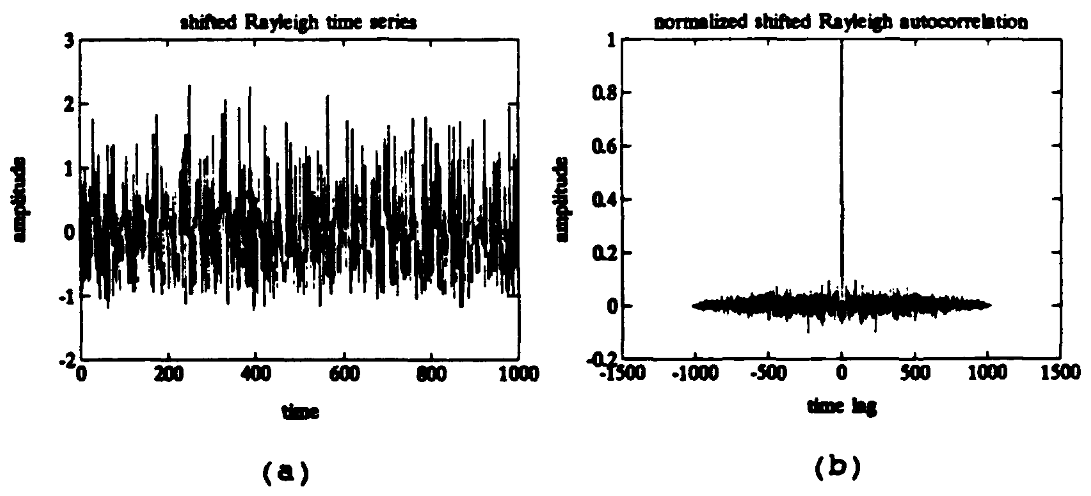


Figure 11. (a) Shifted Rayleigh time series. (b) Normalized shifted Rayleigh time domain based autocorrelation.

Histograms of a Rayleigh pdf and a shifted Rayleigh pdf are shown in Figure 12. Clearly, the effect of removing the dc term from the Rayleigh distributed time series creates

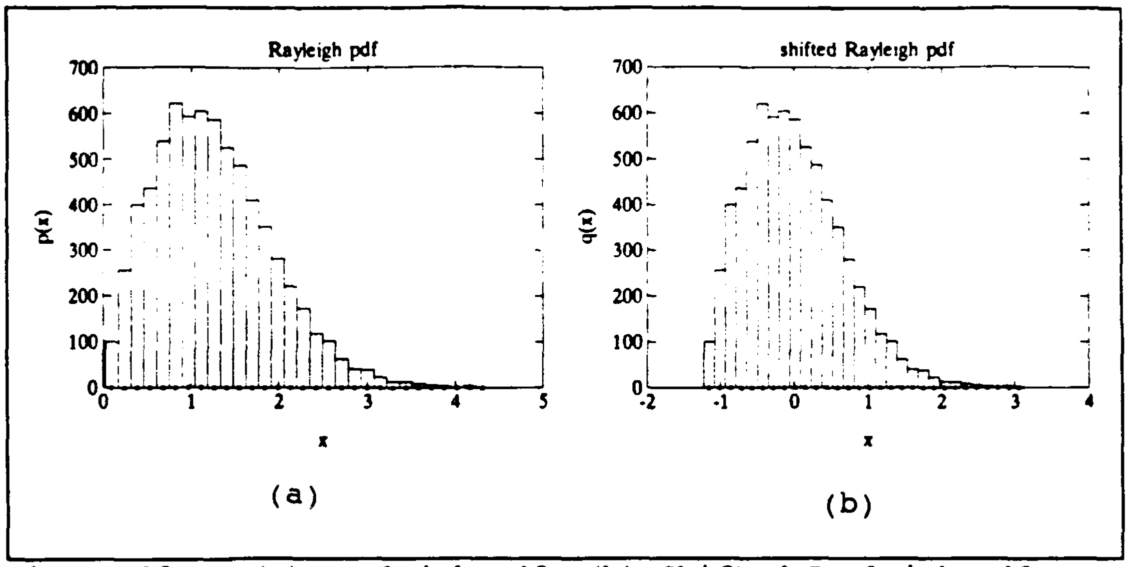


Figure 12. (a) Rayleigh pdf. (b) Shifted Rayleigh pdf.

an autocorrelated output that does not have the limiting triangular bias. The impulse at zero time lag dominates the correlation output. A constant detection threshold, at least over a small number of range bins, could be implemented with this type of correlation output.

2. Frequency Domain Based Correlation of Shifted Rayleigh Noise

The frequency domain based autocorrelation of $x_1(n)$ is now examined. Sequence $x_1(n)$ is zero padded as described in Equation 21. A frequency domain series is created using the FFT as described by Equation 18. The autocorrelation estimate is derived using Equation 19 and is plotted in Figure 13. The results of frequency domain based autocorrelation are very similar to the results of time domain based autocorrelation using shifted Rayleigh noise.

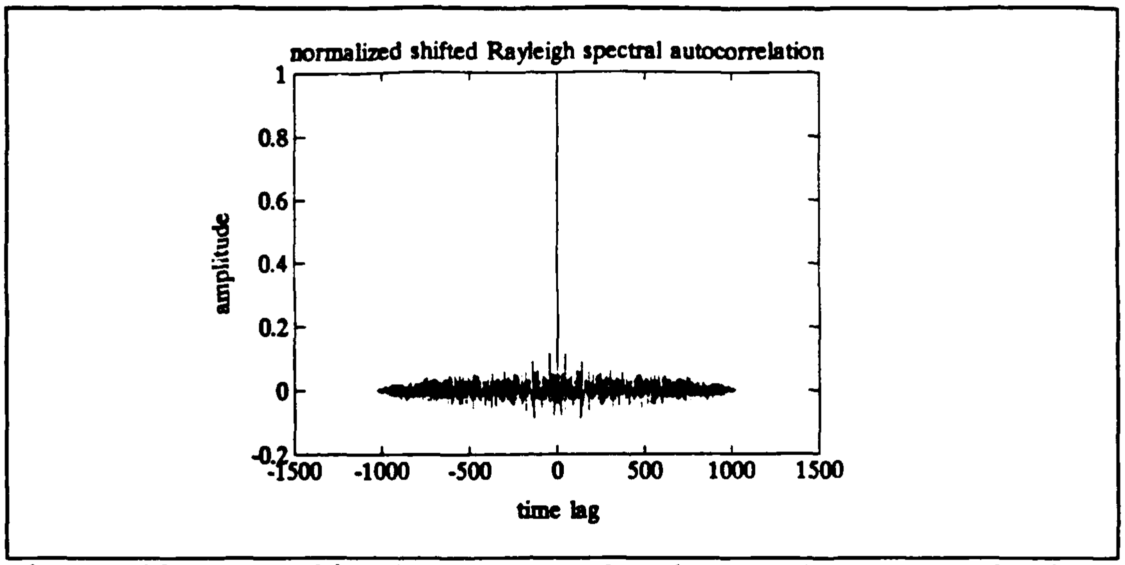


Figure 13. Normalized frequency domain based autocorrelation.

3. Spectral Interpolation

With the incorporation of the FFT in modern radar receivers, it is possible that the received pulse burst has been transformed into frequency data as a byproduct of detection. Processing time would be lost in converting this data back into a time series to be correlated. Of course, there is no guarantee that the dc component has been removed. In this section an algorithm is given to perform frequency domain based correlation on a Rayleigh distributed noise sequence while avoiding the triangular correlation bias created by the (possible) dc component.

A Rayleigh distributed noise series x(n), N points long, is transformed into an ordered sequence of N complex coefficients using the FFT algorithm. In general, a sequence

X(k) of coefficients is indexed from zero to N-1 and is described by

$$X(k) = FFT\ \{\ x(n)\ \},\quad 0 \le k \le N-1 \qquad \text{where } k = \text{frequency index}\ . \tag{31}$$

The FFT creates the zero indexed term X(0) by summing all the elements in x(n). X(0) is real and can be viewed as the dc component of x(n). By removing the X(0) coefficient from the frequency sequence we have in effect removed the dc bias from the time series x(n). At this point, the modified Fourier transform X(k) approximates the DFT of a shifted Rayleigh series.

Time domain correlation of an N point series produces 2N data points. Frequency domain based correlation of an N point sequence produces N data points. To force frequency domain based correlation to equal time domain based correlation (i.e., avoid circular correlation), the number of terms in the frequency series must be doubled. The frequency series is expanded by storing each pair of complex frequency coefficients at a location twice the value of the original index.

As a first approximation, the data points between adjacent coefficients are linearly interpolated in the expanded array. Figure 14 illustrates the effect of FFT interpolation and the zeroing of the dc terms. Correlation results as described by Equation 19 are shown in Figure 15.

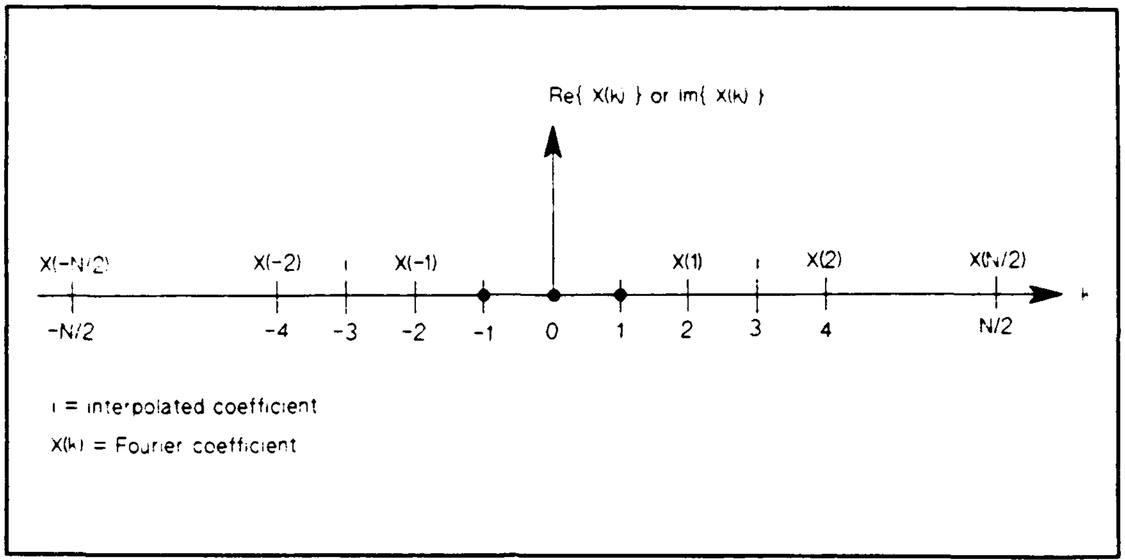


Figure 14. FFT interpolation, and zeroing of the dc coefficient.

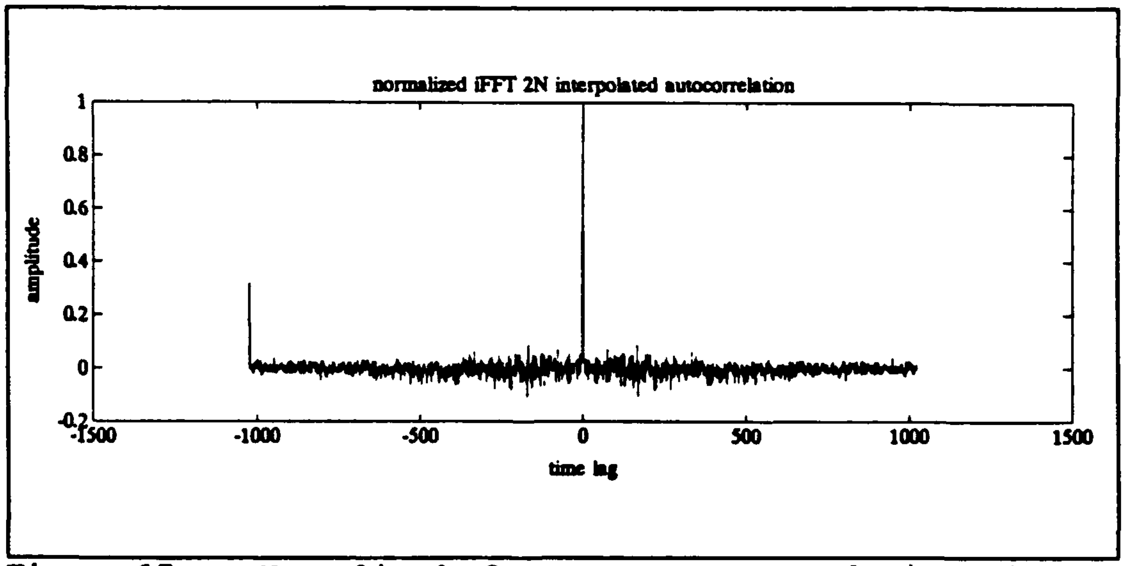


Figure 15. Normalized frequency autocorrelation through interpolation.

Clearly, the triangular bias in the correlation output has been removed. A small impulse at a time lag of -N is observed, which can be disregarded.

For block processing, typically only correlation products within ±10% of the zero time lag position are considered to be statistically reliable. Therefore, the TDOA estimate is formed within this correlation band. The impulse at -N falls outside of the TDOA evaluation area and will not cause a false detection.

C. CORRELATION OF PULSES IN ADDITIVE RAYLEIGH NOISE

Time domain based correlation of a pulse train in additive Rayleigh noise should produce the same result as frequency domain based interpolation followed by an equivalent correlation. A comparison was performed using a 50% duty cycle pulse train in additive shifted Rayleigh noise.

Time correlation was performed on the pulse burst. The dc component of the noisy pulse train is removed forming a zero mean signal. The time domain based autocorrelation of this sequence is shown in Figure 16(a).

The pulse burst was also transformed into the frequency domain using the FFT with the dc component intact. N Fourier coefficients are expanded and the X(0), X(±1) coefficients zeroed out. The sequence is correlated after interpolation using Equation 19 and plotted in Figure 16(b).

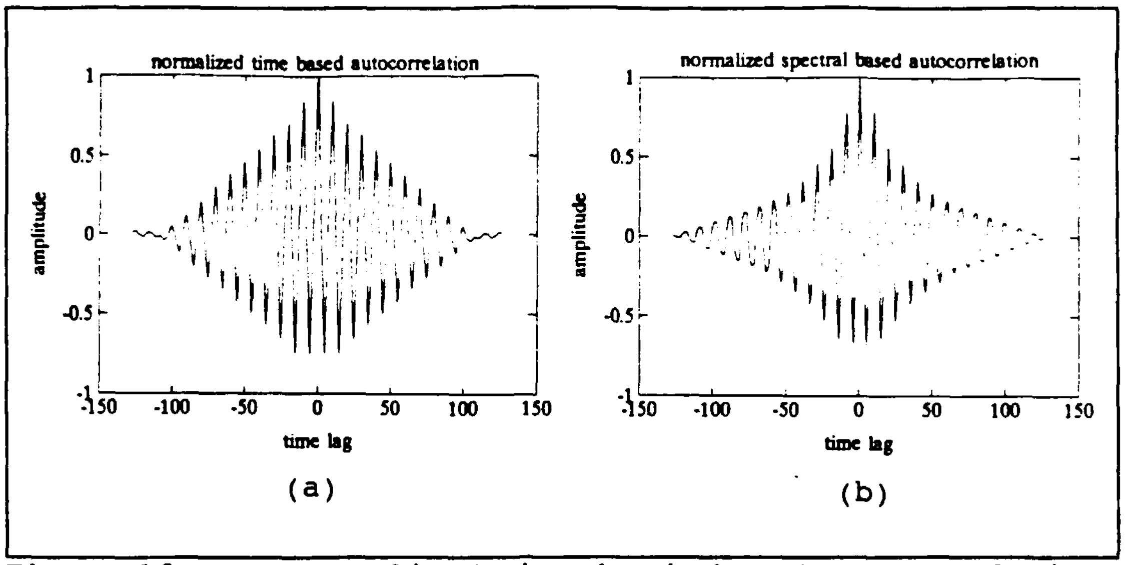


Figure 16. (a) Normalized time domain based autocorrelation. (b) Normalized frequency domain based interpolation and autocorrelation. Zero time lag between each channel.

Clearly, the two correlation algorithms produce similar outputs. Both have a dominant peak in their output for TDOA estimation. Frequency domain based correlation displays a exponential decay of the sidelobes about the main peak. This effect is not observed in time autocorrelation, and is caused by the first order interpolation used in the frequency domain based method. A more robust interpolation algorithm will eliminate the exponential decay, yielding the desired linear decay.

Next, time and frequency domain based crosscorrelation are performed on two pulse sequences. One pulse sequence lags the other by 20 units. Both time and frequency domain based crosscorrelation of the two sequences are plotted in Figure

17. Clearly, both correlation algorithms produce a peak at 20 units. The nonlinear decay of the sidelobes surrounding the dominant peak is again seen in frequency domain correlation.

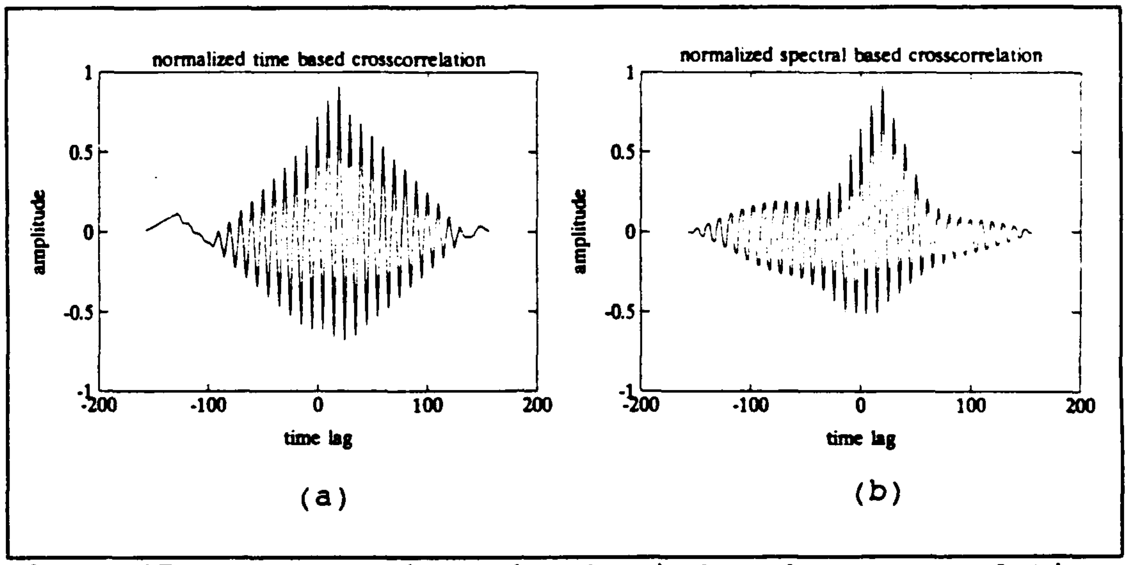


Figure 17. (a) Normalized time domain based crosscorrelation. (b) Normalized frequency domain based interpolation and crosscorrelation. 20 unit time lag between each channel.

In summary, the frequency domain based interpolation and crosscorrelation algorithm produces an output comparable to time domain based crosscorrelation. It offers a method to correlate pulse bursts collected in the frequency domain.

V. THRESHOLD DESIGN FOR ENVELOPE CORRELATION

A. INTRODUCTION

A threshold at the output of a correlator is used to estimate the location of a TDOA peak in a constant false alarm rate (CFAR) receiver. The value of the threshold is a function of the correlation output mean and variance.

B. CORRELATION OUTPUT AND CONSTANT VARIANCE

If the variance of the correlator, when driven by two signals at the input, maintains a fixed value about some mean, a constant threshold above the mean can be used. This allows the design of a constant false alarm rate detector. The largest correlation peak that also crosses the threshold is defined as the TDOA estimate. The constant threshold design is the easiest to implement.

C. CORRELATION OUTPUT AND TRIANGULARLY SHAPED VARIANCE

The crosscorrelation of two zero mean noise sequences with the same number of data points, produces an output whose variance has triangular form. Hence, a constant detection threshold is not easily implemented. Instead, a threshold must be designed that has a slope that matches the slope of the correlation output. The design of a sloping threshold is difficult, particularly if the slope of the correlation peak changes with time (i.e., changing noise environment).

A solution is to mathematically force the triangular correlation variance to be flat. This is done by using a bowtie correction of the form $1/(N-|\ell|)^{1/2}$. A bowtie correction normalizes the correlation output to have a constant (lag independent) variance. This design suffers from time changes in the correlation output forcing the bowtie correction to be recalculated. Note that the amplitude of the TDOA peak will itself be dependant on the type of correction to the correlation output.

D. THRESHOLD BASED ON ZERO LAG

The zero lag threshold method uses the variance of the zero lag correlation output to calculate a constant detection threshold. This is indicated in Figure 18.

The correlation output becomes statistically less reliable the further removed the correlation products are from the zero time lag position. To use correlation intelligently, the correlation output is typically examined ±10% of the distance from the zero time lag position. Consequently, only correlation output ±0.1N from zero time lag will be used in TDOA estimate simulations.

The detection method in this thesis uses a constant threshold against the correlation output. The fixed threshold

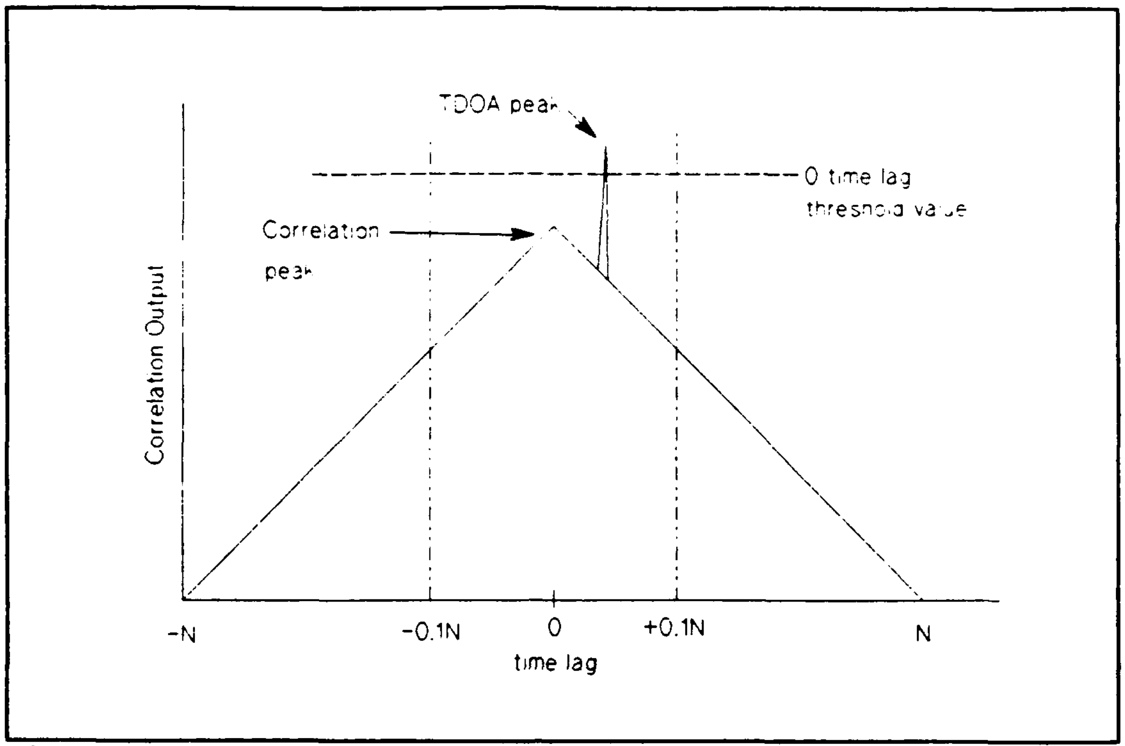


Figure 18. Correlation output with zero lag threshold.

was selected because the correlation output variance is, to a first approximation, nominally flat within the ±0.1N boundary.

The detection scenario that drives this thesis defines a minimum radar pulse width of one microsecond (μsec) and a burst length of four millisecond (msec). A 50% duty cycle radar pulse is assumed. There are 2000 pulses per burst, which when correlated, will generate an output that is 8000μsec (±4000μsec) in length. If ±10% of the correlation output around zero time lag are compared to the threshold value, ±400μsec will be tested. For radar pulses traveling at the speed of light, a maximum distance of ±120km is therefore examined to form a TDOA estimate.

Correlation values at lags -0.1N to 0.1N will be compared in magnitude to the zero time lag based calculated threshold value.

E. THRESHOLD CALCULATION FOR CFAR

The detection threshold is calculated using the output terms of the crosscorrelation algorithm when noise only is injected into the correlator. For convenience, Equation 8 is repeated here,

$$r_{xy}(l) = \sum_{i=0}^{N-|l|-1} x_i \, y_{i+l} \tag{32}$$

x_i and y_{i+l} are zero mean Rayleigh (shifted) distributed noise sequences .

The correlation function at all lags of interest is thought to be Gaussian distributed. According to Equation 32, every estimate is the sum of a fairly large number of products (i.e., central limit theorem). The average value of the correlation output between the ±0.1N endpoints is

$$\tilde{t} = \frac{1}{2a+1} \sum_{l=-a}^{+a} r_{xy}(l) \tag{33}$$

where a = number of data points between the 0 and 0.1N lag positions in the correlation output .

The variance of $r_{xy}(l)$ is calculated from

$$\sigma_t^2 = \frac{1}{2a+1} \sum_{l=-a}^{+a} (r_{xy}(l) - \tilde{t}\,)^2 \quad . \tag{34}$$

For Gaussian distributed noise the probability of false alarm (P_{fa}) is a function of the detection threshold

$$P_{fa} = \frac{1}{\sqrt{2\pi}\sigma_t}\int_T^{\infty} e^{\frac{-x^2}{2\sigma_t^2}}\, dx \tag{35}$$

$$\textit{where } T = \textit{detection threshold} \quad .$$

For a given P_{fa}, the threshold can be calculated from

$$T = erfc^{-1}\,(P_{fa})\,\sigma_t \tag{36}$$

$$\textit{where } erfc = \textit{complementary error function} \quad .$$

Equation 36 relates the threshold to the Gaussian distributed noise variance.

F. VERIFICATION OF CFAR THRESHOLD AND ROC CURVE

Using the above equations, thresholds were calculated for the time correlation detector given in Equation 32 for three different P_{fa}'s. A MATLAB simulation was performed to compare the measured false alarm rate of the time correlation detector to the theoretical false alarm rate.

1. P_{fa} Simulation

Two 100 point sequences of zero mean Rayleigh noise were injected into the correlation detector. The output of the correlation detector between the -0.1N and +0.1N lag points was compared to the calculated threshold to measure the false alarm rate. The number of points in the correlation detector output that exceeded the threshold for a given P_{fa} were recorded. The experimental P_{fa} is given as the ratio of improper threshold crossings to the total number of monitored range cells of the correlator output (see Table 1). For each P_{fa}, 500 realizations were performed. Both theoretical and experimental results are listed.

TABLE 1

THEORETICAL PROBABILITY OF FALSE ALARM

Simulation #	P_{fa}=0.01	P_{fa}=0.001	P_{fa}=0.0001
1	0.0100	0.00076	0
2	0.0099	0.00090	0
3	0.0110	0.00085	0

The data in Table 1 show that a detection threshold can be established for a correlation detector, using the assumption of a Gaussian distributed output for P_{fa}'s of 0.01 and 0.001. For the lower P_{fa} value of 0.0001, the threshold calculation fails, but in a positive sense. The measured P_{fa} of 0 is below the theoretical P_{fa} of 0.0001. Either the correlation variance is smaller than Equation 36 predicts, or more terms must be summed in the correlation output to better approximate the Gaussian pdf as indicated by the central limit theorem.

2. TDOA Simulation

The output of the envelope detector will either be Rayleigh or Rician (non-central Rayleigh) distributed. In those portions of the data where there is no signal energy, the output will be Rayleigh distributed. Where the signal is present, the envelope detector output will be Rician distributed.

The purpose of the time domain based correlation detector is to measure the TDOA of pulsed sequences embedded in Rayleigh noise. Noise in low SNR environments will mask the TDOA peak. A MATLAB simulation was performed to examine the performance of the correlation detector in a decreasing SNR environment.

Two pulse bursts embedded in additive Rayleigh noise were created. The second pulse burst lagged the first by three time units. Each pulse burst is a sequence of ten pulses, each having a period of ten units. Each pulse in the burst had a 50% duty cycle.

The SNR established the amplitude relationship between the noise and signal sequences. First, two sequences of zero mean unit variance Gaussian distributed noise were created. Rayleigh noise was generated from the two Gaussian sequences. The Rayleigh noise had a mean of 1.2533 and a variance of

0.4292. The theoretical second moment of the Rayleigh noise is two and was verified in the MATLAB simulation. The second moment is defined as the average power of the Rayleigh noise. The amplitude of the nonzero elements in each pulse burst was then scaled to meet the desired SNR, using the measured average power of the Rayleigh noise.

A detection threshold was calculated based on shifted Rayleigh noise using Equation 36 where σ^2 was assumed to be known (i.e., simulation information). In practical applications, σ^2 would be estimated from the data. The SNR of the two bursts were varied from 0dB to 18dB in the different simulations. For each SNR, the two sequences were correlated using Equation 32.

Graphs of the correlation output for a SNR of 1dB are given in Appendix C for P_{fa}'s of 0.01, 0.001, 0.0001 and 0.00001. The threshold, which is a function of the P_{fa} and noise variance, is shown to increase with decreasing P_{fa}.

The correlation peak at a time lag of three was the correct TDOA estimate. If the maximum correlation peak in the output was the third lag point, a correct TDOA estimate was obtained. For each SNR, 30 TDOA estimates were made. The correlation function output 100 data (lag) points per TDOA estimate. The TDOA estimate was made on ±10 data points on each side of the zero time lag position in the correlation output. The percent correct TDOA estimate for each SNR was calculated and is presented as a receiver operating characteristic (ROC) curve in Figure 19.

The ROC curve shows that as the SNR increases, the performance of the correlation detector increases. The performance of the correlation detector in Figure 19 can be improved if multiple sequential looks at the same SNR were allowed.

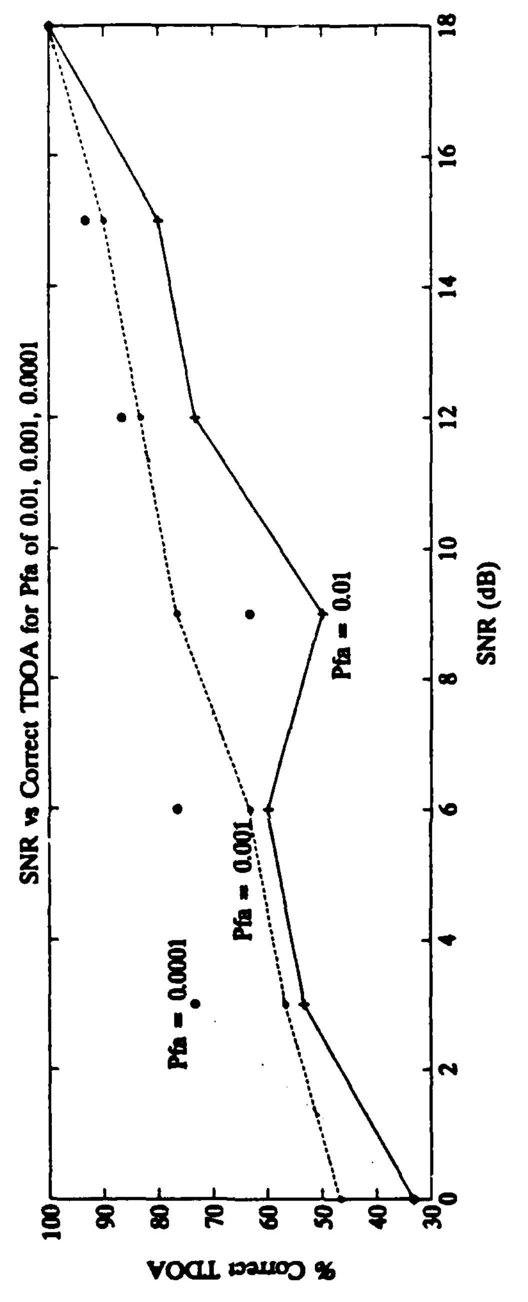


Figure 19. Percentage of estimated TDOA correct versus SNR for time domain based correlation detection.

VI. FFT ERROR ANALYSIS

A. INTRODUCTION

This chapter discusses several sources of processing errors occurring within the FFT, and the effect of these errors on the FFT output signal-to-noise ratio.

The computation of a FFT using fixed point arithmetic, generates internal processing errors. These errors are due to the fixed number of bits in the trigonometric look up table, the truncation of arithmetic results during addition and the scaling of results during the individual butterfly calculations.

The error sources can be modeled as independent additive noise components at the output of the FFT. Their net effect is to reduce the ideal SNR at the output of the FFT.

Each of these processing error sources will be described below. The material in this chapter borrows from [Ref. 7].

B. COSINE/SINE TABLE NOISE

The computation of a butterfly algorithm in the FFT involves multiplication by the complex coefficient (twiddle factor)

$$e^{-j\frac{2\pi}{N}m} = \cos(\frac{2\pi}{N}m) - j\sin(\frac{2\pi}{N}m) \qquad (37)$$
$$= W^{m} \ .$$

A FFT butterfly is shown in Figure 20.

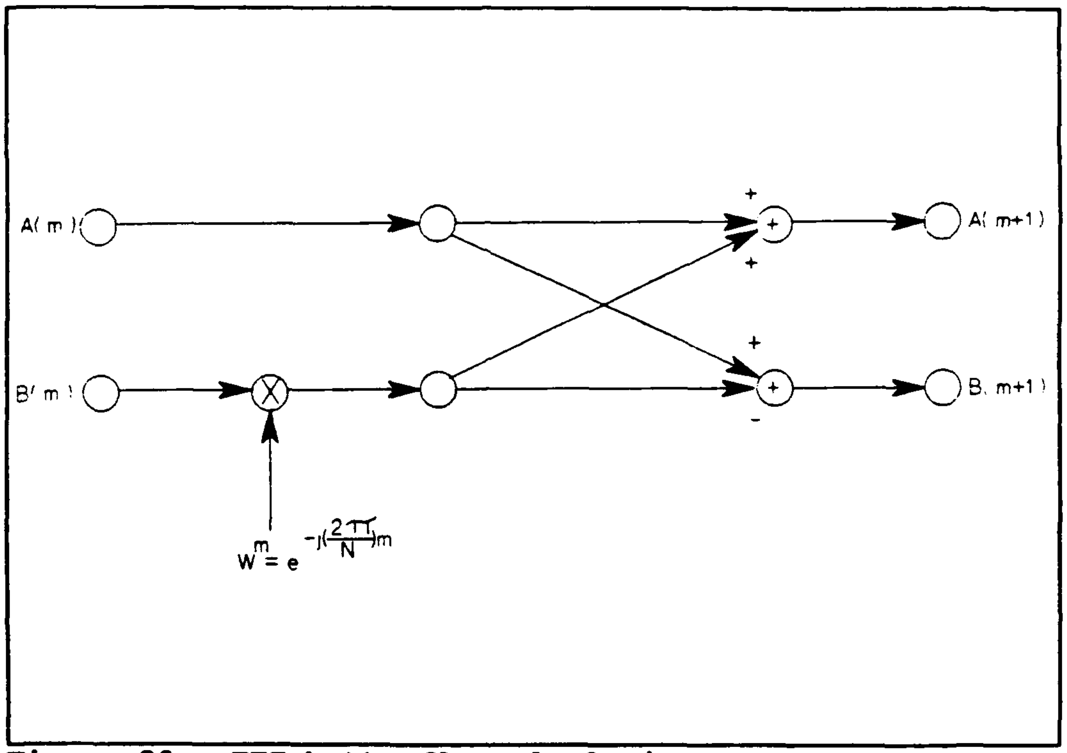


Figure 20. FFT butterfly calculation.

These trigonometric coefficients are precalculated, quantized to (B) bits (where B is the size of the computer word including sign bit), and stored in memory for future table lookup during the FFT processing. The coefficients $W^{0} = 1$ and $W^{n/4} = -j$ can be obtained in an exact form and therefore do not contribute to the quantization noise. Quantization of the other (twiddle) coefficients stored in the sine and cosine table introduce noise into the output of the FFT. The noise power created by truncation of the sine and cosine coefficients is defined by [Ref. 7] to be

$$\sigma_w^2 = \frac{1}{3} 2^{-2B} \tag{38}$$

where σ_w^2 = *trigonometric noise power*
B = *number bits in computer word* .

C. TRUNCATION NOISE

The DFT of a finite duration sequence {x(n)} is defined in Equation 18 and is repeated here for convenience

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi}{N}n}, \quad 0 \le k \le N-1 \ . \tag{39}$$

The product formed requires four real multiplications because both the exponential and the input data are complex numbers, B bits in length. Each multiplication can produce a result of 2B bits which must be truncated from 2B to B bits, hence there are four quantization errors for each complex valued multiplication. The four quantization errors can be described as four independent noise sources. The truncation noise power is defined by [Ref. 7] as

$$\sigma_T^2 = \{\frac{1}{3}2^{-2B} + \frac{1}{4}2^{-2B}\} \tag{40}$$

where σ_T^2 *= truncation noise power*
B = number of bits in product after truncation .

D. SCALING NOISE

The FFT as an algorithm processes a data vector of length N by successive passes at the vector. During each pass, the algorithm performs N/2 butterfly operations. Each butterfly retrieves two complex numbers from memory, performs the butterfly computation and returns two complex numbers to the same memory address. The complex numbers returned to memory by the butterfly can have a greater magnitude than the two complex numbers initially fetched from memory by the butterfly. In order for the results of the butterfly operation to fit in the fixed word length of the memory of the computer, the results must be truncated (scaled). Scaling is performed by dividing by two the entire data array, and is implemented in software by a right shift and an increment of the arrays exponent register.

During the (m+1) pass of the data, the butterfly algorithm selects two data points A(m) and B(m) and returns to memory A(m+1) and B(m+1). The truncation error results from the addition of two numbers of like sign in the upper part of the butterfly algorithm or the subtraction of two numbers of opposite sign in the lower part of the butterfly algorithm. The input to the butterfly is fixed to have maximum real and imaginary values of ±1. The output of the butterfly is fixed to have a maximum value of ±2, double the input value. To prevent overflow of the butterfly output, either a prescale by 1/2 is required prior to entering the butterfly or an extra bit must be available in memory to accommodate the possible gain of two.

One method, automatic prescaling, truncates the least significant bit of the butterfly output and represents processing noise. In the case for which no scaling is necessary for the pass, this truncation represents significant processing error (noise).

A second method, data dependent scaling, is performed only if a butterfly in the data pass overflows without scaling. If any word in memory has an overflow bit set, all words are shifted right as they are fetched from memory for the next pass of the FFT. The total number of right shifts, p, executed during the FFT process is used at the output of the FFT as a scale factor of 2^p. This scale factor is used at the output of the FFT so that the total processing gain is maintained.

The noise power created by the scaling of the data is defined by [Ref. 7]

$$\sigma_S^2 = \frac{1}{2}\{\frac{1}{3}2^{-2(B-1)} + \frac{1}{4}2^{-2(B-1)}\} \tag{41}$$

where σ_S^2 = *scaling noise power* .

E. OUTPUT SIGNAL TO NOISE RATIO

Figure 21 shows an error model of a butterfly in an FFT.

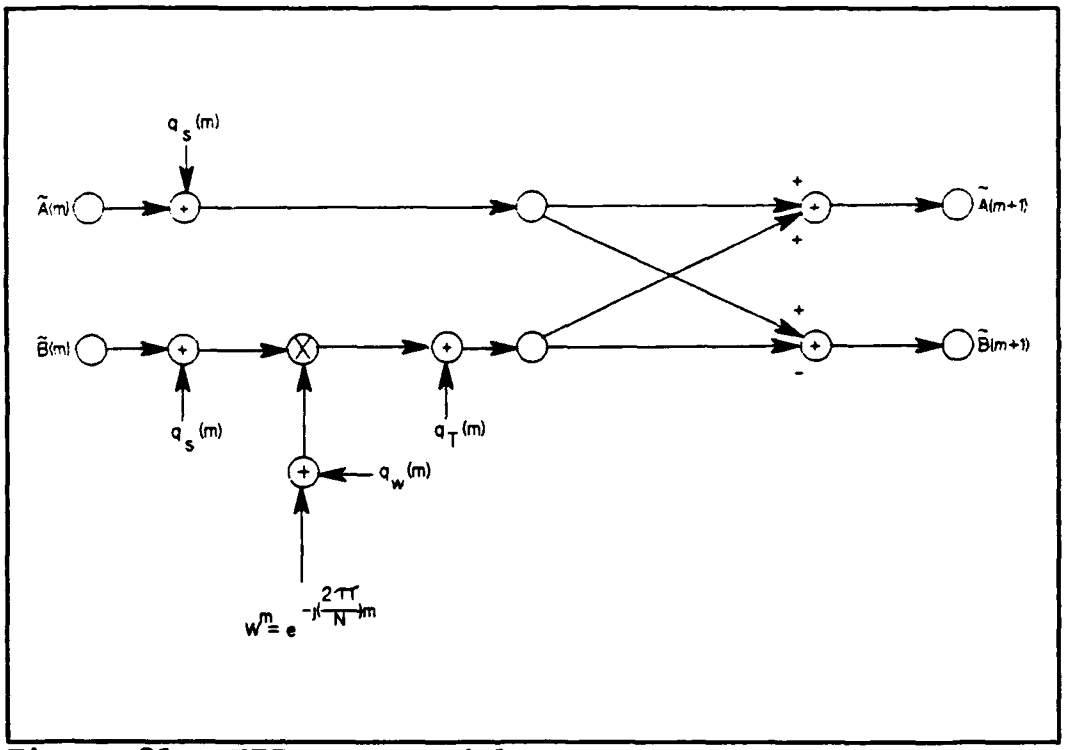


Figure 21. FFT error model.

The input data terms to a butterfly can be described by

$$\begin{aligned}\tilde{A}(m) &= A(m) + N_A(m)\\ \tilde{B}(m) &= B(m) + N_B(m)\\ \text{where } N_A,\ N_B &= \text{additive noise terms}\\ A(m),\ B(m) &= \text{input data terms}\\ \tilde{A}(m),\ \tilde{B}(m) &= \text{noise corrupted data}\quad .\end{aligned} \tag{42}$$

Without the effects of noise the butterfly algorithm produces two outputs described by

$$\begin{aligned}A(m+1) &= A(m) + W(m)B(m)\\ B(m+1) &= A(m) - W(m)B(m)\\ \text{where } W(m) &= \text{twiddle factor}\quad .\end{aligned} \tag{43}$$

The error due to the A component will be calculated. The error due to the B component is statistically equivalent to the error in the A component. Including the effects of noise, the output of the butterfly can be described by

$$\begin{aligned}\tilde{A}(m+1) &= [A(m) + N_A(m) + q_s(m)] + [B(m) + N_B(m) + q_s(m)]\cdot\\ &\quad [W(m) + q_w(m)] + q_T(m)\\ \text{where } q_s(m) &= \text{scale noise}\\ q_w(m) &= \text{sine/cosine table noise}\\ q_T(m) &= \text{truncation noise (in multiply)}\quad .\end{aligned} \tag{44}$$

The total power at the output of the butterfly is then

$$\begin{aligned} E\{\tilde{A}(m+1)\ \tilde{A}^*(m+1)\} = {} & E\{A(m)A^*(m)\} + E\{B(m)B^*(m)\} + \\ & E\{N_A(m)N_A^*(m)\} + E\{N_B(m)N_B^*(m)\} + \\ & E\{q_S(m)q_S^*(m)\} + E\{q_S(m)q_S^*(m)\} + \\ & E\{B(m)B^*(m)\}E\{q_W(m)q_W^*(m)\} + \\ & E\{q_T(m)q_T^*(m)\} \quad . \end{aligned} \tag{45}$$

Redefining the terms of the total power leads to

$$\begin{aligned} E\{\tilde{A}(m+1)\tilde{A}^*(m+1)\} = {} & S^2(m) + S^2(m) + N_A^2(m) + N_B^2(m) + \\ & 2\sigma_s^2(m) + S^2(m)\sigma_W^2(m) + \sigma_T^2(m) \\ \text{where } S^2(m) = {} & \text{input signal power} \\ \sigma_s^2(m) = {} & \text{scaling noise power} \\ \sigma_T^2(m) = {} & \text{truncation noise power} \\ \sigma_w^2(m) = {} & \text{sine/cosine table noise power} \quad . \end{aligned} \tag{46}$$

Combining like terms results in

$$\begin{aligned} E\{\tilde{A}(m+1)\tilde{A}^*(m+1)\} = {} & 2S^2(m) + 2N_A^2(m) + 2\sigma_s^2(m) + \\ & S^2(m)\sigma_W^2(m) + \sigma_T^2(m) \ . \end{aligned} \tag{47}$$

This equation describes the total power as a function of the signal power and the individual independent uncorrelated noise terms. The total power output from the butterfly can also be described as a function of its input signal power and the total additive noise power generated within the butterfly as

$$\begin{aligned} E\{\tilde{A}(m+1)\tilde{A}^*(m+1)\} = {} & E\{A(m+1)A^*(m+1)\} + \\ & E\{N_A(m+1)N_A^*(m+1)\} \quad . \end{aligned} \tag{48}$$

Redefining these terms

$$E\{\tilde{A}(m+1)\tilde{A}^*(m+1)\} = S^2(m+1) + N_A^2(m+1) \ . \tag{49}$$

Equation 47 and 49 are equivalent. The signal and noise power for the (m+1) pass can be formed from those terms in the proceeding pass

$$\begin{aligned} S^2(m+1) + N_A^2(m+1) = 2S^2(m) + 2N_A^2(m) + 2\sigma_s^2(m) + \\ S^2(m)\sigma_W^2(m) + \sigma_T^2(m) . \end{aligned} \tag{50}$$

Equating the signal terms in Equation 50 yields

$$S^2(m+1) = 2S^2(m) . \tag{51}$$

Equating the noise terms in Equation 50 yields

$$N_A^2(m+1) = 2N_A^2(m) + 2\sigma_s^2(m) + S^2(m)\sigma_W^2(m) + \sigma_T^2(m) . \tag{52}$$

Equation 51 can be rewritten to show how the input signal power increases as a function of m as it passes through the mth stage of a butterfly of an FFT

$$\begin{gathered} S^2(m) = 2^m S^2, \\ \textit{here we modeled } S^2(0) = S^2 . \end{gathered} \tag{53}$$

The signal power input to the butterfly is defined as

$$S^2 = \frac{1}{9}2^{-2(B-b)} . \tag{54}$$

Rewriting Equation 52 to include this exponential increase of $S^2(m)$ gives

$$N_A^2(m+1) = 2N_A^2(m) + 2\sigma_S^2(m) + 2^m S^2 \sigma_W^2(m) + \sigma_T^2(m) . \qquad (55)$$

Equation 55 shows the relationship between the noise power of a butterfly stage as a function of the noise power from the proceeding butterfly stage. The noise and signal power can now be calculated using Equations 53 and 55 for m passes through a butterfly. Once calculated, the signal to noise ratio can be formed.

F. RIGHT JUSTIFIED DATA

The length of a word in the computer is defined to be 16 bits (B) and accepts a 12 bit (b) data word. Right justified data places the least significant bit (LSB) of the b bit data word in the LSB of the B bit processor word. The output noise to signal ratio is now calculated for ten passes through a butterfly using right justified data. The following assumptions are used in the calculation;

1. The input signal is incoherent.
2. Scaling is performed every other pass after the most significant bit (MSB) of data reaches the next to MSB of the processor.
3. The first two passes do not use the trigonometric table.

After the tenth pass (i.e., FFT size of 1024), it is shown (Appendix D) that the noise to signal power is

$$\frac{N_A^2}{S^2}(10) = \frac{N_q^2}{S^2} + 4\sigma_W^2 + \frac{1}{4}\frac{\sigma_T^2}{S^2} + \frac{1}{51}\frac{\sigma_s^2}{S^2}. \qquad (56)$$

B and b are previously assumed to be 16 and 12 bits in length respectively. Substituting these values into Equation 56

$$\frac{N_A^2}{S^2}(10) = \frac{\frac{1}{3}2^{-32}}{\frac{1}{9}2^{-8}} + 4\cdot\frac{1}{3}2^{-32} + \frac{1}{4}\frac{\frac{7}{12}2^{-32}}{\frac{1}{9}2^{-8}} + \frac{1}{51}\frac{\frac{7}{24}2^{-30}}{\frac{1}{9}2^{-8}}. \qquad (57)$$

The noise to signal ratio reduces to

$$\begin{aligned}\frac{N_A^2}{S^2}(10) &= -67.5\ db \quad -95.1\ dB \quad -71.1\ dB \quad -79.1\ dB \\ &= \textit{input noise} \quad \textit{trig noise} \quad \textit{truncation noise} \quad \textit{scaling noise}\ .\end{aligned} \qquad (58)$$

Clearly, the dominant degradation noise is the truncation table noise (-71.1dB), followed by the scaling noise to signal ratio (-79.1dB). For right justified data, the signal to noise ratio is calculated to be 65.71dB.

G. LEFT JUSTIFIED DATA

Left justified data places the most significant bit (MSB) of the b bit data word in the MSB of the B bit FFT processor word. The output noise to signal ratio is now calculated for 10 passes through a butterfly for left justified data. The following assumptions are made

1. The signal is incoherent.
2. Data is left justified on input and scaling is performed every other pass starting with the first pass.
3. The first two data passes do not use the trigonometric table.

It can be shown that after the tenth pass through the butterfly that the noise to signal ratio is

$$\frac{N_A^2}{S^2}(10) = \frac{N_q^2}{S^2} + 4\sigma_W^2 + \frac{1}{4}\frac{\sigma_T^2}{S^2} + 2\frac{\sigma_S^2}{S^2}. \tag{59}$$

B and b were previously assumed to be 16 and 12 bits respectively. Substituting these values into Equation 59 we obtain

$$\frac{N_A^2}{S^2}(10) = \frac{\frac{1}{3}2^{-26}}{\frac{1}{9}2^{-2}} + 4\cdot\frac{1}{3}2^{-32} + \frac{\frac{1}{4}\frac{7}{12}2^{-32}}{\frac{1}{9}2^{-2}} + \frac{2\frac{7}{24}2^{-30}}{\frac{1}{9}2^{-2}}. \tag{60}$$

The noise to signal ratio reduces to

$$\begin{aligned}\frac{N_A^2}{S^2}(10) &= -67.5\,dB \quad -95.1\,dB \quad -89.1\,dB \quad -77.1\,dB \\ &= \begin{matrix}\textit{input} & \textit{trig} & \textit{truncation} & \textit{scaling} \\ \textit{noise} & \textit{noise} & \textit{noise} & \textit{noise}\,.\end{matrix}\end{aligned} \tag{61}$$

Clearly, the worst noise contribution to the SNR is the scaling noise (-77.1dB), followed by the truncation noise (-89.1dB). For left justified data, the signal to noise ratio is calculated to be 67.0dB.

Further analysis shows that for large FFT's (i.e., in the range of 2^{15} and above), the trigonometric table noise is the dominant noise source. A simulation was performed using an AMDAHL machine to examine the error generated in performing a FFT and an iFFT. Figure 22 shows the experimental setup.

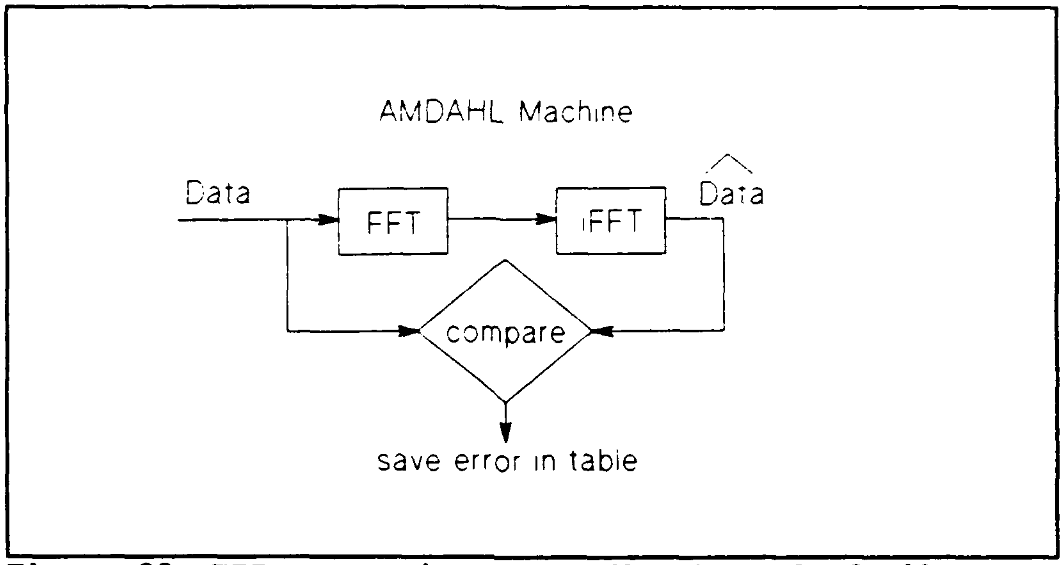


Figure 22. FFT processing error flowchart [Ref. 8].

The error that was generated was tabulated. Figure 23 illustrates the error as a function of FFT length.

H. SUMMARY

For either left or right justified data, the dominant noise at the output of the 1024 point FFT is the scaling or truncation noise. The SNR at the output of the FFT, for left justified data is 67.0dB and exceeds the SNR for right justified data which is 65.71dB. Clearly, left justified data maximizes the computational SNR at the output of the FFT for

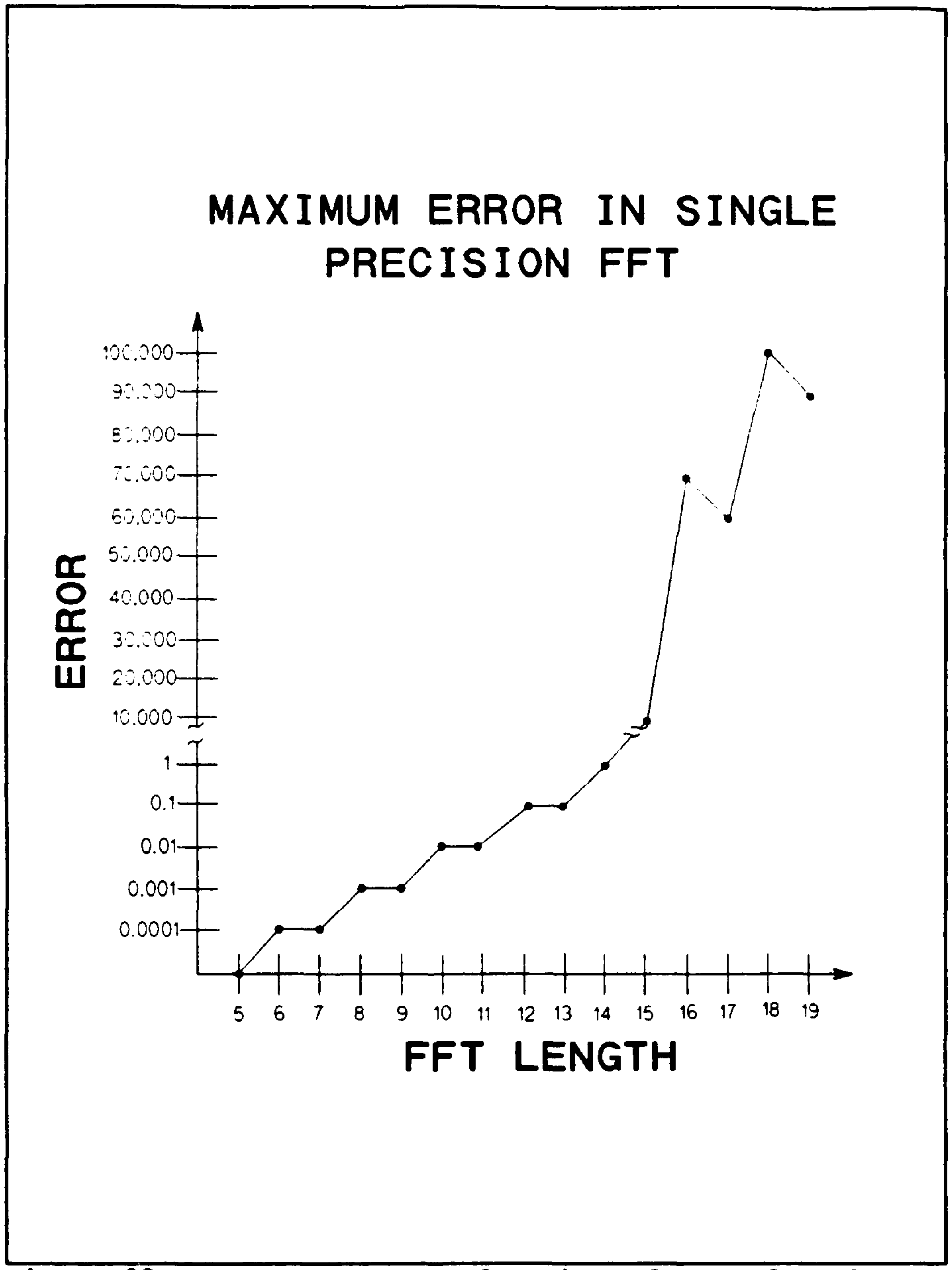


Figure 23. FFT error as a function of transform length [Ref. 8].

the values of b, B and N used in the above examples.

For a right justified data set transform size of N=2^{13} (8192 points), the noise to signal ratio after the 13th pass can be written

$$\frac{N_A^2}{S^2}(13) = \frac{2^{13}N_q^2 + (2^{15} + 2^{13} + 2^{12})S^2\sigma_w^2 + 2047\sigma_T^2 + 170\sigma_s^2}{2^{13}S^2} \quad . \qquad (62)$$

This equation reduces to

$$\frac{N_A^2}{S^2}(13) \cong \frac{N_q^2}{S^2} + 5.5\sigma_w^2 + \frac{1}{4}\frac{\sigma_T^2}{S^2} + \frac{1}{48.2}\frac{\sigma_s^2}{S^2} \quad . \qquad (63)$$

Comparing Equation 56 to Equation 63, we see that by increasing the transform size from 2^{10} to 2^{13} increases the noise to signal ratio. The trigonometric table noise is increasing faster than either the scaling or truncation noise. The signal to noise ratio is reduced, from 65.71dB to 65.68dB. As the transform size is increased to accommodate larger data sets, the trigonometric table noise becomes the dominant error term.

VII. SUMMARY

A. CONCLUSIONS

A frequency domain based algorithm was developed and tested to estimate the differential arrival time of a pulsed radar signal collected by two passive sensors. For convenience, the actual implementation was performed through time domain processing even though frequency domain processing is advocated. The performance of the algorithm is characterized by a ROC curve as a function of SNR. For 3dB SNR and a P_{fa} of 0.01, the probability of making a correct TDOA estimate exceeds that of an incorrect estimate for a single look. At 18dB SNR, the probability of making a correct TDOA estimate is 100 percent for all P_{fa}'s. If multiple looks (i.e., multiple bursts) are allowed, the probability of making a correct TDOA decision at each SNR will increase.

An I/Q demodulator is assumed in the radar receiver. The pdf of the signal driving the correlation detector is determined from where it is obtained in the I/Q receiver. The pdf, depending on selection, will be zero mean Gaussian versus non-zero mean Gaussian, or chi-squared versus non-central chi-squared, or Rayleigh versus Rician (non-central Rayleigh). This thesis assumes an envelope detector at the output of the I/Q demodulator. The envelope detector output has a Rayleigh or Rician pdf and drives the correlation detector. The calculation of a CFAR threshold assumes a Gaussian pdf at the output of the correlation detector. For small time lags, summation of the output of the correlator may produce a probability distribution that deviates from Gaussian. This deviation would produce a biased threshold calculation. To minimize the bias, a sufficient number of terms must be summed at the output of the correlator to allow the Gaussian approximation.

If the received pulses are collected in the frequency domain, a spectral domain based correlation algorithm can be implemented. An algorithm is given in this thesis.

For any digital signal processing algorithm that uses the FFT, processing errors must be considered. For large transform sizes the trigonometric noise power dominates. The length of the trigonometric coefficient word affects the degradation of the SNR at the output of the FFT. The larger this word the smaller the noise power. For a given transform size, left justified data will have a higher output SNR than right justified data.

For a correct TDOA estimate, the correlation algorithm requires the reception of the leading pulses in the radar pulse burst. If the pulses are received in an adequate SNR, the correlation algorithm will produce a well defined peak allowing the TDOA estimate. Should the environment degrade the received signal (i.e., destructive interference due to multipathing, weather or terrain), the correlation algorithm should be discarded in favor of the traditional angle of arrival (AOA) algorithm (i.e., maximize received energy).

If the radar uses a staggered PRF, the reception of the first pulse or several pulses has little impact on the position of the correlation peak. Under this condition, the TDOA algorithm is superior to the AOA algorithm.

B. RECOMMENDATIONS

A ROC curve should be constructed for the correlation algorithm used in this thesis for multiple TDOA looks. An intensity display could be designed that would plot as a function of time (i.e., snapshots), those lag points chosen as TDOA estimates. Patterns displayed on the *intensity plots* would allow the user to visually determine the correct TDOA (i.e., incoherent averaging).

The performance of the spectral domain based correlation algorithm developed in this thesis must be further quantified. Sets of ROC curves should be obtained for different SNR's (i.e., SNR variations between channels).

The simulated FFT error graph should be validated by measuring the error performance of a dedicated properly dimensioned FFT.

APPENDIX A. CORRELATION MEAN AND VARIANCE

Two zero mean, independent time series are the inputs to a correlator. This appendix will derive the expected value and vaıiance of the crosscorrelation function output. In this thesis, the noise is zero mean (i.e., shifted) Rayleigh noise.

The crosscorrelation function $r_{xy}(l)$ is defined in Equation 8 and is repeated here for convenience

$$r_{xy}(l) = \sum_{i=0}^{N-1-|l|} x(i)y(i+l) \quad . \tag{64}$$

A. EXPECTED VALUE

x_i and y_i are assumed to be independent, zero mean sequences

$$E\{x_i\} = E\{y_i\} = 0 . \tag{65}$$

The expected value of $r_{xy}(l)$ is

$$\begin{aligned} E\{r_{xy}(l)\} &= E\{\sum_{i=0}^{N-1-|l|} x_i y_{i+l}\} = \sum_{i=0}^{N-1-|l|} E\{x_i y_{i+l}\} \\ &= \sum_{i=0}^{N-1-|l|} E\{x_i\} E\{y_{i+l}\} \\ &= 0 \quad . \end{aligned} \tag{66}$$

B. VARIANCE

The random variable z has a variance defined to be

$$\sigma_z^2 = E\{(z - E\{z\})^2\}$$
$$\text{because } E\{z\} = 0 \text{ for a zero mean sequence,} \tag{67}$$
$$\sigma_z^2 = E\{z^2\}.$$

The variance of the crosscorrelation function $r_{xy}(l)$ is then

$$\sigma^2_{r_{xy}(l)} = E\{r^2_{xy}\} = E\{(\sum_{i=0}^{N-1-|l|} x_i y_{i+l})^2\} = E\{\sum_{i=0}^{N-1-|l|} x_i y_{i+l} \sum_{i=0}^{N-1-|l|} x_i y_{i+l}\} \tag{68}$$
$$= E\{\sum_{i=0}^{N-1-|l|} \sum_{j=0}^{N-1-|l|} x_i y_{i+l}\, x_j y_{j+l}\} \quad .$$

$x_i x_j$ and $y_{i+l}\ y_{j+l}$ are two groups that are independent of each other. The expectation operation can be applied to each group individually.

$$\sigma^2_{r_{xy}(l)} = \sum_{i=0}^{N-1-|l|} \sum_{j=0}^{N-1-|l|} E\{x_i x_j\}\ E\{y_{i+l} y_{j+l}\} \quad . \tag{69}$$

The two indices i and j define a matrix of values. Two contributions have to be considered.

Contribution 1. When i=j the summation occurs along the diagonal of the matrix. Only one summation is used and essentially the terms are being squared. These squared terms are not independent. To simplify the computation define m equal to i and j

$$C_1 = \sum_{m=0}^{N-1-|l|} E\{x_m^2\} E\{y_{m+l}^2\} \quad . \tag{70}$$

Because x_m and y_m are zero mean sequences,

$$E\{x_m^2\} = \sigma_x^2, \; E\{y_{m+l}^2\} = \sigma_y^2$$

$$\text{and } \sigma_x^2 = \sigma_y^2 = \sigma^2 \quad .$$

Substituting σ^2 into Equation 70

$$\begin{aligned} C_1 &= \sum_{m=0}^{N-1-|l|} \sigma_x^2 \sigma_y^2 = \sigma^4 \sum_{m=0}^{N-1-|l|} 1 \\ &= \sigma^4 [(N-1-|l|)+1] \\ &= \sigma^4 (N-|l|) \quad . \end{aligned} \tag{71}$$

Contribution 2. When i≠j all the terms except those on the diagonal are summed. These terms are independent. For i≠j

$$\begin{aligned} C_2 &= \sum_{i=0}^{N-1-|l|} \sum_{j=0}^{N-1-|l|} E\{x_i\} E\{x_j\} E\{y_{i+l}\} E\{y_{j+l}\} \\ &= 0, \text{ using Equation } 65 \quad . \end{aligned} \tag{72}$$

$$\begin{aligned} \textit{Therefore, Equation 69 becomes } \sigma^2_{\hat{r}_{xy}(l)} &= C_1 + C_2 \\ &= \sigma^4 (N - |l|) \quad . \end{aligned} \tag{73}$$

C. CROSSCORRELATION STATISTICS

Each data point output by the correlation function represents a summation of terms. Using the central limit theorem, the output of the correlation function is assumed to have a Gaussian distribution.

In conclusion $r_{xy}(l) \sim N(0,\ \sigma^4(N-|l|)$

where N *= Normal (Gaussian) distribution* .

APPENDIX B. MATLAB CODE

A. USERS GUIDE

The frequency domain based correlation software given in this appendix is written in Pro-Matlab (version 3.5h). All simulations were run on a Naval Postgraduate School Sun work station using a UNIX operating system.

B. CORRELATION MATLAB CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program Name : FreqCorr7.m                    12 May 91
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clg
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       PULSE CORRELATION
%                          Version 7.0
%
% 1. 1 Channel vs Reference Channel
%             or
%    1 Channel vs Second Channel.
%
% 2. Rician Signal and Rayleigh Noise pdf's .
%    (Low SNR signals have power approximately 2.)
%
% 3. 0 dc Correlation. Subtract dc from both signals.
%
% 4. Normalize correlation output in Fourier domain.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This MatLab program will either :
% 1. Correlate a received pulse sequence against a reference
%   pulse sequence. The reference sequence parameters are
%   specified by the user and are used to construct the
%   sequence.
%                              or
%
```

```
% 2. Correlate two received pulse sequences against each
%    other.
%
%
%  Correlation of the pulse sequences is accomplished by
%  multipling in the frequency domain one spectrum against
%  the conjugate of the other spectrum. An inverse FFT is
%  performed on the result to yield the time difference
%  (TDOA) between the two sequences.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      SELECT CORRELATION OPTION
%
%    Select either 1 or 2 :
%
%  1. Time delay in one receiver channel measured against a
%     reference signal.
%
%  2. Differential time delay between two channels.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
option = input('Select correlation option 1 or 2 ;      ');
clc


echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Option 1 : Time Delay In One Channel
%
%          Reference Pulse Train Parameter Selection.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
if option==1
   wi=input( 'Pulse Width (seconds) :    ');
   pr=input( 'Pulse Period (T seconds) :    ');
   nu=input( 'Number of pulses in pulse train :    ');
   NumberPointsRef=pr*nu;
end
```

```
clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%        Delayed Pulse Train Parameter Selection
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
dly=input('Channel 1 delay tau (sec.) of pulse train :   ');
wi2=input('Pulse Width (seconds) :   ');
pr2=input('Pulse Period (T seconds)  :   ');
nu2=input('Number of pulses in Delayed Pulse Train :   ');
NumberPointsDel=(pr2*nu2)+dly;  %calc. nmbr pts sig + delay

clc
if option==2
 dlyCh2=input('Channel 2 delay tau (sec.) pulse train :   ');
 wiCh2=input('Pulse Width (seconds) :   ');
 prCh2=input('Pulse Period (T seconds)  :   ');
 nuCh2=input('Number of pulses in Delayed Pulse Train :   ');
 NumberPointsDel2=(prCh2*nuCh2)+dlyCh2; %nmbr pts sig + delay
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Zero pad both sequences to N3=N2+N1-1. Must then make N3
%  meet the simple formula N3 = 2^m to allow speedy
%  computation of the FFT.
%  WARNING : IBM 80286 MATLAB WILL NOT ALLOW VECTORS GREATER
%  THAN 4000. So an input pulse train of 128 will exceed the
%  system. Solution is to use the SUN workstation or 386/486.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if option==1
 ZeroPadPoints=NumberPointsRef+NumberPointsDel-1;%N3=N2+N1-1
 for m=2:1:19;                 %2^(19) = 524288 point FFT max
     if 2^m >= ZeroPadPoints,
                ZeroPadPoints=2^m;
                break;     %when find a 2^m power, exit loop
     end
 end
end
```

```
if option==2
ZeroPadPoints=NumberPointsDel2+NumberPointsDel-1;%N3=N2+N1-1
 for m=2:1:19;                   %2^(19) = 524288 point FFT max
    if 2^m >= ZeroPadPoints,
              ZeroPadPoints=2^m;
              break;        %when find a 2^m power, exit loop
    end
 end
end


%%%%%%%%%%%%%% Create Reference Pulse Train  %%%%%%%%%%%%%
if option==1
ReferencePulse=zeros(1:ZeroPadPoints); %zero pad past pr*nu
for PulseCounter=1:pr:NumberPointsRef  %steps of Ref. period
    for CutUpPulse=0:pr;
        if CutUpPulse<=wi
           ReferencePulse(PulseCounter+CutUpPulse)=1.0;
        end
    end
end
Refdc=mean(ReferencePulse(1:NumberPointsRef));
ReferencePulse(1:NumberPointsRef)=ReferencePulse(1:NumberPoi
ntsRef)-Refdc;
end


%%%%%%% Create Channel 1 Delayed Pulse Train  %%%%%%%%%%%%%
DelayedPulseTrain=zeros(1:ZeroPadPoints);%0 pad past pr2*nu2
for PulseCounter=(dly+1):pr2:(pr2*nu2+(dly+1));%pr2*nu2=Nmbr
    for CutUpPulse=0:pr2;        %...pts in Del. Pulse Train
        if CutUpPulse<=wi2
           DelayedPulseTrain(PulseCounter+CutUpPulse)=1.0;
        end
    end
end


if option==2
%%%%%%% Create Channel 2 Delayed Pulse Train  %%%%%%%%%%%%%
DelayedPulseTrain2=zeros(1:ZeroPadPoints);%0 padpast pr2*nu2
for PulseCounter=(dlyCh2+1):prCh2:(prCh2*nuCh2+(dlyCh2+1));
    for CutUpPulse=0:prCh2;
        if CutUpPulse<=wiCh2
           DelayedPulseTrain2(PulseCounter+CutUpPulse)=1.0;
```

```
            end
        end
    end
    end


    clc
    echo on
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %             Signal to Noise Ratio Calculation
    %           Define Rayleigh noise power to equal 2.
    %        Rician pdf created for sinusoid + Gaussian noise.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    echo off
    SNR=input(' Desired SNR :  ');
    NoisePwr=2;                          %rayleigh noise pwr =2

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %      Sum Channel 1 & 2 signals plus their initial delay.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    SigEnergy=0;                 %initialize Channel 1 energy to 0
    for index=1:NumberPointsDel;%sum over Ch. 1 delayed signal
        SigEnergy=SigEnergy+((DelayedPulseTrain(index)).^2);
    end

    if option==2
    SigEnergy2=0;                %initialize Channel 2 energy to 0
    for index=1:NumberPointsDel2;%sum over Ch. 2 delayed signal
        SigEnergy2=SigEnergy2+((DelayedPulseTrain2(index)).^2);
    end
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calc. Ch. 1 signal amplitude to meet required input SNR.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    SNR10=SNR/10;
    SNRLinear=10^SNR10;                %SNR in linear units
    SigAmplitude=sqrt(NoisePwr*SNRLinear*NumberPointsDel/SigEner
    gy);


    if option==2
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calc. Ch.2 signal amplitude to meet required input SNR.
```

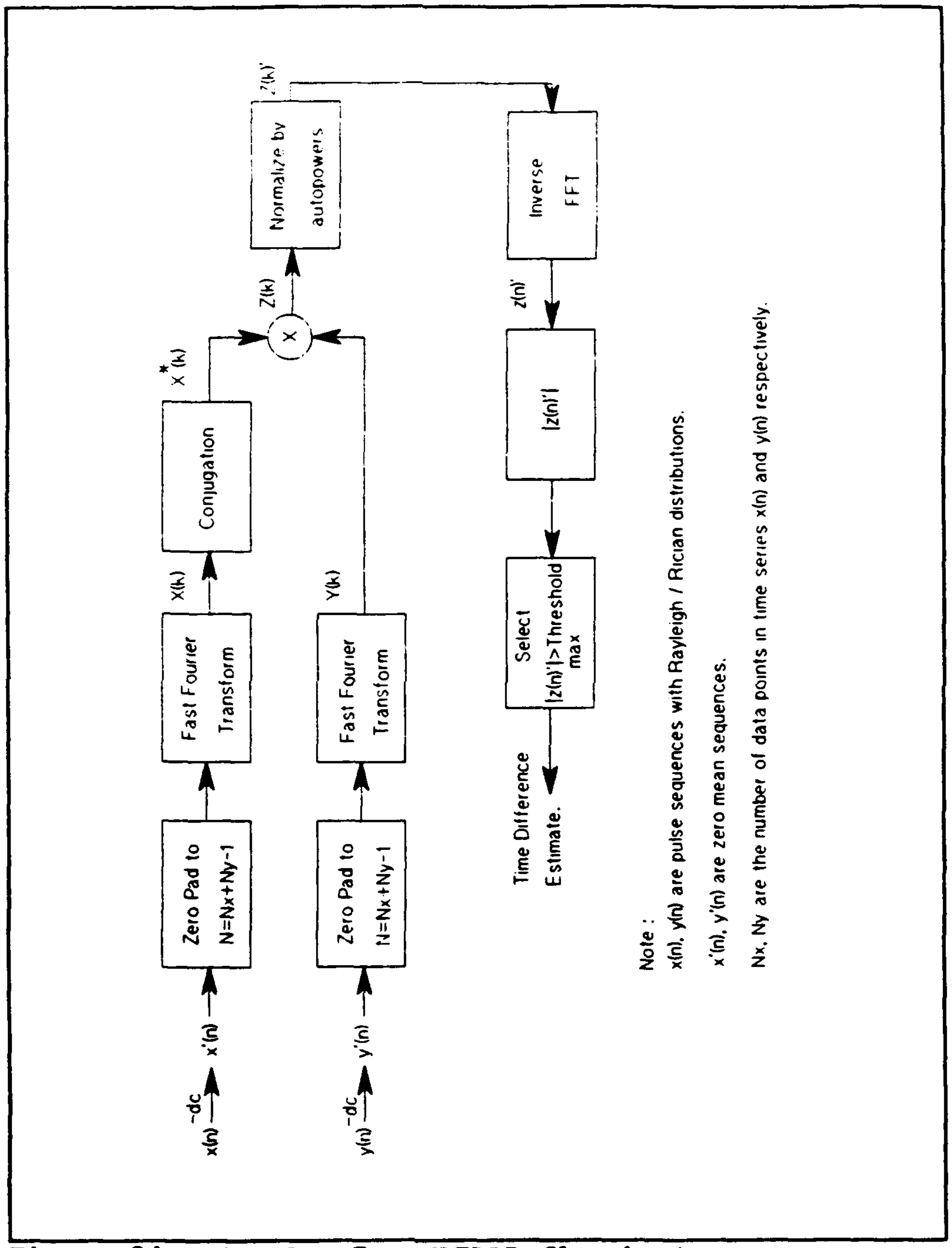
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR10=SNR/10;
SNRLinear=10^SNR10;                 %SNR in linear units
SigAmplitude2=sqrt(NoisePwr*SNRLinear*NumberPointsDel2/SigEn
ergy2);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Create Rayleigh Noise matrix for Channel 1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rand('normal');                    %Gaussian mean=0, variance=1
for index=1:NumberPointsDel,
    GaussNoise1(index)=rand;     %N(0,1)
    GaussNoise2(index)=rand;     %N(0,1)
end
RayleighNoise=sqrt((GaussNoise1).^2+(GaussNoise2).^2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Create Rayleigh Noise matrix for Channel 2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if option==2
rand('normal');                    %Gaussian mean=0, variance=1
for index=1:NumberPointsDel2,
    GaussNoise3(index)=rand;     %N(0,1)
    GaussNoise4(index)=rand;     %N(0,1)
end
RayleighNoise2=sqrt((GaussNoise3).^2+(GaussNoise4).^2);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Make Rician amplitude of Channel 1 delayed pulse train
%  and remove the dc component of entire signal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SE=0;
for index=1:NumberPointsDel,  %1 --> delay + delayed signal
 if DelayedPulseTrain(index)==1        %if one, make Rician


DelayedPulseTrain(index)=DelayedPulseTrain(index).*rician(Si
gAmplitude);
    SE=(DelayedPulseTrain(index)).^2+SE;
 end   %end Rician modification loop
end
```

```
SP=SE/NumberPointsDel;

%%%%%%%  Add noise to Channel 1 Delayed Pulse Train %%%%%%%
for index=1:NumberPointsDel,  %1 --> delay + delayed signal
 if DelayedPulseTrain(index)==0 %no Rayleigh noise to signal

DelayedPulseTrain(index)=DelayedPulseTrain(index)+RayleighNo
ise(index);
 end
end

%%%%% Remove dc component of Channel 1 Pulse Train %%%%
Ch1dc=mean(DelayedPulseTrain(1:NumberPointsDel));
DelayedPulseTrain(1:NumberPointsDel)=DelayedPulseTrain(1:Num
berPointsDel)-Ch1dc;


if option==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make Rician amplitude of Channel 2 delayed pulse train
% and remove dc component of entire signal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SE=0;
for index=1:NumberPointsDel2, %1 --> delay + delayed signal
 if DelayedPulseTrain2(index)==1       %if one, make Rician

DelayedPulseTrain2(index)=DelayedPulseTrain2(index).*rician(
SigAmplitude2);
    SE=(DelayedPulseTrain2(index)).^2+SE;
 end  %end Rician modification loop
end
SP=SE/NumberPointsDel2;


%%%%%%  Add noise to Channel 2 Delayed Pulse Train %%%%%%%%
for index=1:NumberPointsDel2, %1 --> delay + delayed signal
 if DelayedPulseTrain2(index)==0 %no Rayleigh noise to sig

DelayedPulseTrain2(index)=DelayedPulseTrain2(index)+Rayleigh
Noise2(index);
 end
end

%%%%% Remove dc component of Channel 2 Pulse Train %%%%
Ch2dc=mean(DelayedPulseTrain2(1:NumberPointsDel2));
```

```
DelayedPulseTrain2(1:NumberPointsDel2)=DelayedPulseTrain2(1:
NumberPointsDel2)-Ch2dc;
end

subplot(221)                     %two rows, two columns
time=(0:1:ZeroPadPoints-1);      %Start time axis at 0
if option==1
%%%%%%%%%%%% Plot Reference Pulse Train  %%%%%%%%%%%%%%%%
plot(time,ReferencePulse);
title('Reference Pulse Sequence');
xlabel('time');
grid
end


%%%% Plot Channel 1 pulse train + Rayleigh/Rician noise%%%%
topDel=1.1*max(DelayedPulseTrain);
bottomDel=1.1*min(DelayedPulseTrain);
axis([0 NumberPointsDel bottomDel topDel])
plot(time,DelayedPulseTrain);
title('Channel 1 Pulse Sequence + Rician/Rayleigh Noise');
xlabel('time');
grid


if option==2
%%%% Plot Channel 2 pulse train + Rayleigh/Rician noise %%%
topDel2=1.1*max(DelayedPulseTrain2); %10 percent headroom
bottomDel2=1.1*min(DelayedPulseTrain2);
axis([0 NumberPointsDel2 bottomDel2 topDel2])
plot(time,DelayedPulseTrain2);
title('Channel 2 Pulse Sequence + Rician/Rayleigh Noise');
xlabel('time');
grid
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Correlation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if option==1
fR=fft(ReferencePulse);
fD=fft(DelayedPulseTrain);
end
```

```
if option==2
fR=fft(DelayedPulseTrain);
fD=fft(DelayedPulseTrain2);
end

z=fD.*conj(fR);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate normalizing coefficient
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Norm1=0;
Norm2=0;
for k=1:1:length(fD)
    Norm1=(abs(fD(k)).^2)+Norm1;
    Norm2=(abs(fR(k,)).^2)+Norm2;
end

Norm1=sqrt(Norm1);
Norm2=sqrt(Norm2);

NormCoeff=Norm1*Norm2;
z=z./NormCoeff;

z=z*length(fD);

ifftz=ifft(z);
magz=abs(ifftz);


%%%%% Flip vector for appearance %%%%%%%%%%%%%%
Topmagz=(length(magz)/2+1);
Bottommagz=length(magz)/2;
magzLen=length(magz);  %measure length of magz
Transmagz=[magz(Topmagz:magzLen),magz(1:Bottommagz)];

%%%%%% Make -time to +time axis %%%%%%%%%%%%%%%
minustime=-(length(Transmagz)/2);
posittime=(length(Transmagz)/2)-1;
time1=(minustime:1:posittime);

topmagz=1.1*max(magz); %add 10 percent head room
axis([minustime posittime 0 topmagz]) %scale axis

subplot(212)
plot(time1,Transmagz);
title('FFT Correlation : Channel 1 vs. Channel 2');
```

```
xlabel('time lag');
grid
gtext('+7dB SNR')

lj4print            %Spanagel Room 427 Laserjet

axis([1 2 3 4]);  %reset axis scaling
axis;







%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program Name : CorrPfaLoop.m                        19 July 91
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Purpose : To compute the Pfa for Signal / Noise only for
% then correlation function. This program creates zero mean
% RAYLEIGH noise.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clg
subplot(221)
rand('normal')

PFA=1.0;                    %will be divided by ten to start
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for PFACOUNTER=1:1:4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PFA=PFA/10;                 %Pfa = 0.1 0.01, 0.001, 0.001

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for SNR=0:1:18             %Outer loops through all SNR's.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Iterations=30;             %30 data points per SNR point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for loop=1:1:Iterations %inner loop, creates groups of +- a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
loop;
AboveThreshold(loop)=0;
BelowThreshold(loop)=0;
```



Note :

x(n), y(n) are pulse sequences with Rayleigh / Rician distributions.

x'(n), y'(n) are zero mean sequences.

Nx, Ny are the number of data points in time series x(n) and y(n) respectively.

Figure 24. FreqCorr7.m MATLAB flowchart.

```
Threshold(loop)=0;
MaxRxy(loop)=0;
RxyThreelag(loop)=0;

%%%%%%%%%%%% Set Parameters %%%%%%%%%%%%%%%%%%%%%%%%
N=100;  %100 noise points, therefore 200 correlation points
a=10;   %+- 10 % on either side of 0 time lag
Pfa=PFA;    %Reestablish because Pfa destroyed each loop
PfaDefined=Pfa;

%%%%%%%%%%%%% Make N points Noise %%%%%%%%%%%%%%%%%%
snrnoiseloop %call signal plus noise generation at a SNR


%%%%% Correlation of zero mean noise only for Pfa calcul. %%
MeanRay1=mean(RayleighNoise);   %noise created in snrnoise.m
MeanRay2=mean(RayleighNoise2);
RayleighNoise=RayleighNoise-MeanRay1;
RayleighNoise2=RayleighNoise2-MeanRay2;

Rxy=xcorr(RayleighNoise,RayleighNoise2); %xcorr signal+noise
later

%%%%%%%%%%%%%%%%%% Compute t~ %%%%%%%%%%%%%%%%%%%%%%%
t=0;
for i=(N-a):1:(N+a)
    t=t+Rxy(i);
end
t=t/(2*a+1);

%%%%%%%%%%%%%%% Compute variance t %%%%%%%%%%%%%%
Vart=0;
for i=(N-a):1:(N+a)
    Vart=Vart+(Rxy(i)-t).^2;
end
Vart=Vart/(2*a+1);

%%%%%%%%%%%%%%%%%% Define Pfa %%%%%%%%%%%%%%%%%%%%
Pfa=Pfa*2;                          %multiply by 2
Pfa=1-Pfa;                          %subtract one
Threshold(loop)=inverf(Pfa);
Threshold(loop)=Threshold(loop)*sqrt(2)*sqrt(Vart);  %mult by
the sqrt(2)

%%%%% Xcorr of signal + zero mean noise %%%%%%%%%%
Rxy=xcorr(DelayedPulseTrain,DelayedPulseTrain2);
```

```
%%%%%%% Compare +a to -a to Threshold %%%%%%%%%%%
for i=(N-a):1:(N+a)
    if Rxy(i)>=Threshold(loop)
       AboveThreshold(loop)=AboveThreshold(loop)+1;
    else
       BelowThreshold(loop)=BelowThreshold(loop)+1;
    end

    if Rxy(i)>MaxRxy(loop)
       MaxRxy(loop)=Rxy(i);
       SaveThreshold(loop)=Threshold(loop);
       %SaveVart(loop)=sqrt(Vart);
    end
end

RxyThreelag(loop)=Rxy(97);

end                              %end # Iterations loop

%%%%%%% Count correct threshold crossings %%%%%%%
CorrectTDOA=0;
for count=1:1:Iterations
    if RxyThreelag(count)==MaxRxy(count)
       CorrectTDOA=CorrectTDOA+1;
    end
end
PercentCorrectTDOA=(CorrectTDOA*100)/Iterations;

diary flag4
PercentCorrectTDOA
BelowThreshold
AboveThreshold
PfaDefined
SNR
SaveThreshold                       %compare threshold to zeroth
lag below
RxyThreelag                         %Channel 2 lags Channel 1 by
three pulses.
MaxRxy
diary off

end                               %end # test points loop

end                               %end Pfa loop
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program Name : snrnoiseloop.m                  19 JULY 91
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dly=0;    %Channel 1 delay
wi2=5;    %Channel pulse width
pr2=10;   %Channel 1 pulse period
nu2=10;   %Channel 1 number of pulses
NumberPointsDel=(pr2*nu2)+dly;  %calc. nmbr pts sig + delay

dlyCh2=3;   %Channel 2 delay
wiCh2=5;    %Channel 2 pulse width
prCh2=10;   %Channel 2 pulse period
nuCh2=10;   %Channel 2 number of pulses
NumberPointsDel2=(prCh2*nuCh2)+dlyCh2; %nmbr pts sig + delay

ZeroPadPoints=NumberPointsDel2+NumberPointsDel-1;%N3=N2+N1-1
 for m=2:1:19;              %2^(19) = 524288 point FFT max
    if 2^m >= ZeroPadPoints,
              ZeroPadPoints=2^m;
              break;        %when find a 2^m power, exit loop
    end
 end

%%%% Create Channel 1 Delayed Pulse Train  %%%%%%%%%%
DelayedPulseTrain=zeros(1:ZeroPadPoints);%0 pad past pr2*nu2
for PulseCounter=(dly+1):pr2:(pr2*nu2+(dly+1));%pr2*nu2=Nmbr
    for CutUpPulse=0:pr2;        %...pts in Del. Pulse Train
        if CutUpPulse<=wi2
           DelayedPulseTrain(PulseCounter+CutUpPulse)=1.0;
        end
    end
end

%%%%% Create Channel 2 Delayed Pulse Train  %%%%%%%%%
DelayedPulseTrain2=zeros(1:ZeroPadPoints);%0 padpast pr2*nu2
for PulseCounter=(dlyCh2+1):prCh2:(prCh2*nuCh2+(dlyCh2+1));
    for CutUpPulse=0:prCh2;
        if CutUpPulse<=wiCh2
           DelayedPulseTrain2(PulseCounter+CutUpPulse)=1.0;
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Signal to Noise Ratio Calculation
%           Define Rayleigh noise power to equal 2.
%       Rician pdf created for sinusoid + Gaussian noise.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
NoisePwr=2;                               %rayleigh noise pwr =2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Sum Channel 1 & 2 signals plus their initial delay.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SigEnergy=0;                    %initialize Channel 1 energy to 0
for index=1:NumberPointsDel;%sum over Ch. 1 delayed signal
    SigEnergy=SigEnergy+((DelayedPulseTrain(index)).^2);
end

SigEnergy2=0;                   %initialize Channel 2 energy to 0
for index=1:NumberPointsDel2;%sum over Ch. 2 delayed signal
    SigEnergy2=SigEnergy2+((DelayedPulseTrain2(index)).^2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calc. Ch. 1 signal amplitude to meet required input SNR.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR10=SNR/10;
SNRLinear=10^SNR10;                  %SNR in linear units
SigAmplitude=sqrt(NoisePwr*SNRLinear*NumberPointsDel/SigEner
gy);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calc. Ch.2 signal amplitude to meet required input SNR.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR10=SNR/10;
SNRLinear=10^SNR10;                  %SNR in linear units
SigAmplitude2=sqrt(NoisePwr*SNRLinear*NumberPointsDel2/SigEn
ergy2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Create Rayleigh Noise matrix for Channel 1.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for index=1:NumberPointsDel,
    GaussNoise1(index)=rand;     %N(0,1)
    GaussNoise2(index)=rand;     %N(0,1)
end
RayleighNoise=sqrt((GaussNoise1).^2+(GaussNoise2).^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Create Rayleigh Noise matrix for Channel 2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for index=1:NumberPointsDel2,
    GaussNoise3(index)=rand;     %N(0,1)
    GaussNoise4(index)=rand;     %N(0,1)
end
RayleighNoise2=sqrt((GaussNoise3).^2+(GaussNoise4).^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Make Rician amplitude of Channel 1 delayed pulse train
%  and remove the dc component of entire signal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SE=0;
for index=1:NumberPointsDel,  %1 --> delay + delayed signal
 if DelayedPulseTrain(index)==1  %if one, make Rician

DelayedPulseTrain(index)=DelayedPulseTrain(index).*rician(Si
gAmplitude);
    SE=(DelayedPulseTrain(index)).^2+SE;
 end      %end Rician modification loop
end
SP=SE/NumberPointsDel;

%%%%%%% Add noise to Channel 1 Delayed Pulse Train %%%%%%%
for index=1:NumberPointsDel,  %1 --> delay + delayed signal
 if DelayedPulseTrain(index)==0 %no Rayleigh noise to signal

DelayedPulseTrain(index)=DelayedPulseTrain(index)+RayleighNo
ise(index);
 end
end

%%%%% Remove dc component of Channel 1 Pulse Train %%%%
Ch1dc=mean(DelayedPulseTrain(1:NumberPointsDel));
DelayedPulseTrain(1:NumberPointsDel)=DelayedPulseTrain(1:Num
berPointsDel)-Ch1dc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make Rician amplitude of Channel 2 delayed pulse train
% and remove dc component of entire signal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SE=0;
for index=1:NumberPointsDel2, %1 --> delay + delayed signal
 if DelayedPulseTrain2(index)==1       %if one, make Rician

DelayedPulseTrain2(index)=DelayedPulseTrain2(index).*rician(
SigAmplitude2);
    SE=(DelayedPulseTrain2(index)).^2+SE;
 end  %end Rician modification loop
end
SP=SE/NumberPointsDel2;

%%%%%%  Add nc se to Channel 2 Delayed Pulse Train %%%%%%%%
for index=1:NumberPointsDel2, %1 --> delay + delayed signal
 if DelayedPulseTrain2(index)==0 %no Rayleigh noise to sig

DelayedPulseTrain2(index)=DelayedPulseTrain2(index)+Rayleigh
Noise2(index);
 end
end

%%%%% Remove dc component of Channel 2 Pulse Train %%%%
Ch2dc=mean(DelayedPulseTrain2(1:NumberPointsDel2));
DelayedPulseTrain2(1:NumberPointsDel2)=DelayedPulseTrain2(1:
NumberPointsDel2)-Ch2dc;

DelayedPulseTrain=DelayedPulseTrain(1:100);
DelayedPulseTrain2=DelayedPulseTrain2(1:100);

RayleighNoise=RayleighNoise(1:100);
RayleighNoise2=RayleighNoise2(1:100);
```

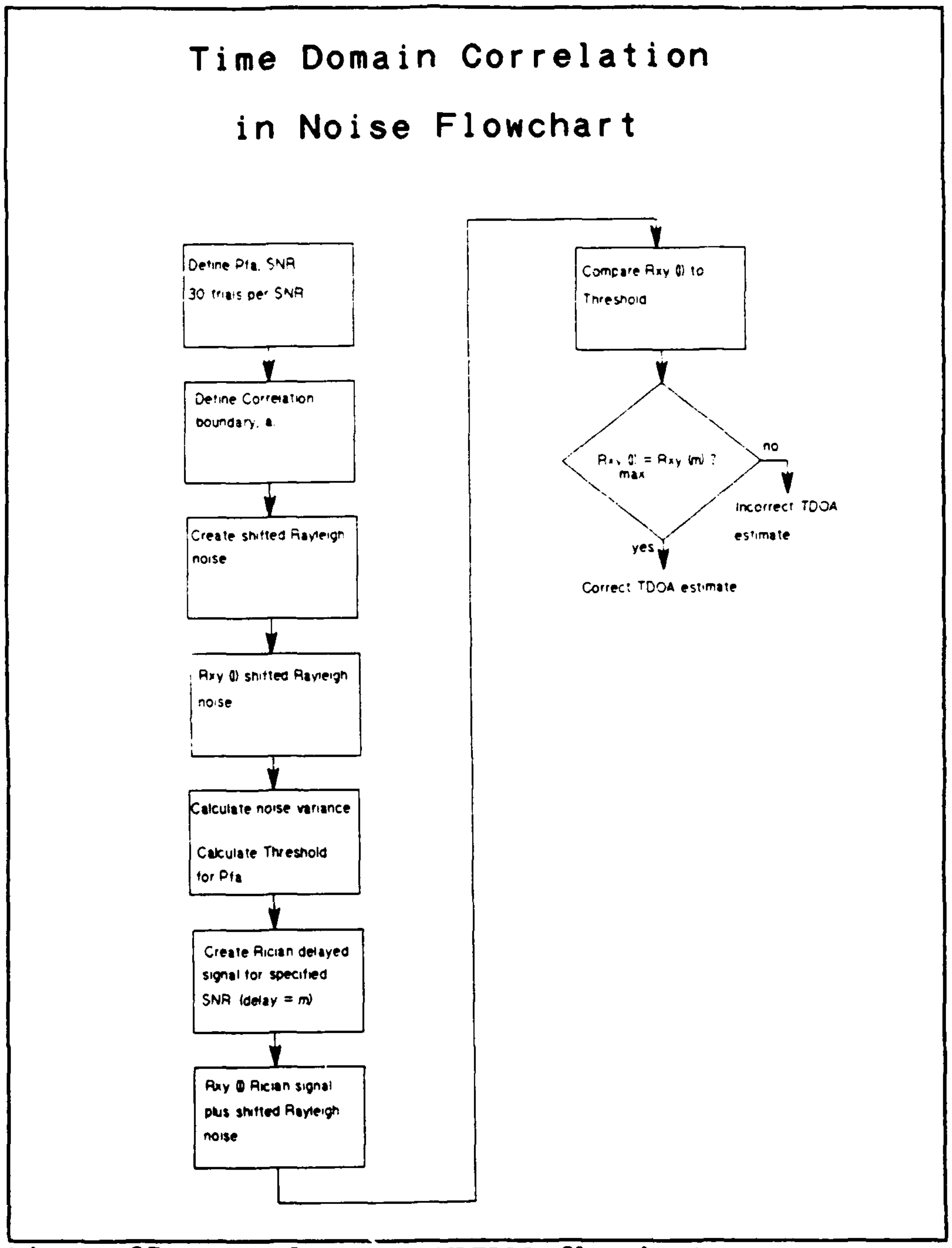

Time Domain Correlation
in Noise Flowchart
Define Pfa, SNR
30 trials per SNR
Define Correlation boundary, a.
Create shifted Rayleigh noise
Rxy (0) shifted Rayleigh noise
Calculate noise variance
Calculate Threshold for Pfa
Create Rician delayed signal for specified SNR (delay = m)
Rxy (0) Rician signal plus shifted Rayleigh noise
Compare Rxy (0) to Threshold
Rxy (0) = Rxy (m) ?
max
no
Incorrect TDOA estimate
yes
Correct TDOA estimate

Figure 25. CorrPfaLoop.m MATLAB flowchart.

APPENDIX C. TDOA CONSTANT THRESHOLD SIMULATION

A. INTRODUCTION

Four graphs of the output of a time domain based correlation detector using a constant threshold are given. Each figure has four simulations using a SNR of 1dB. The P_{fa} is varied from 0.01 to 0.00001 from Figure 26 to Figure 29.

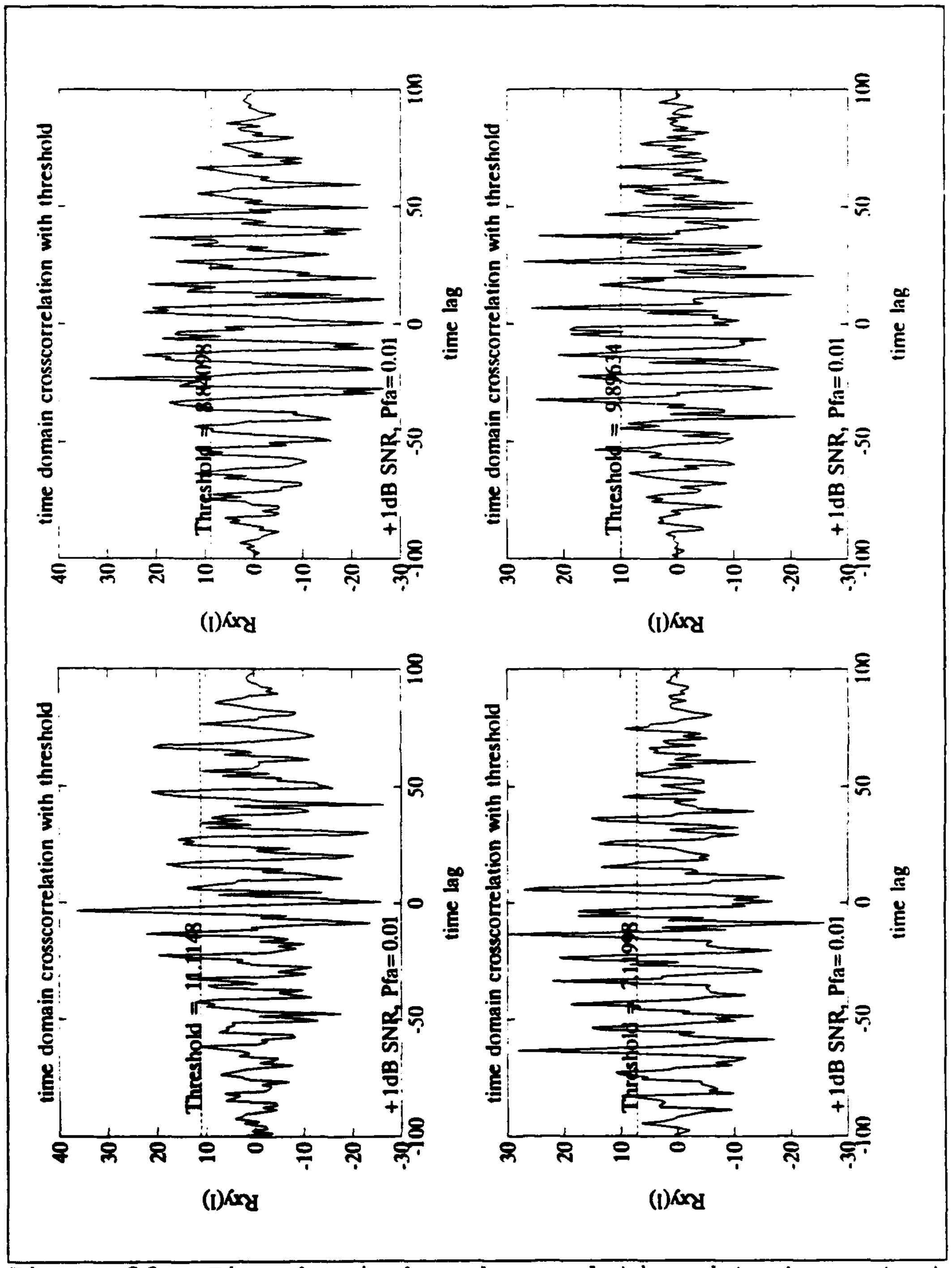


Figure 26. Time domain based correlation detector output for a SNR of 1dB and a P_{fa} of 0.01.

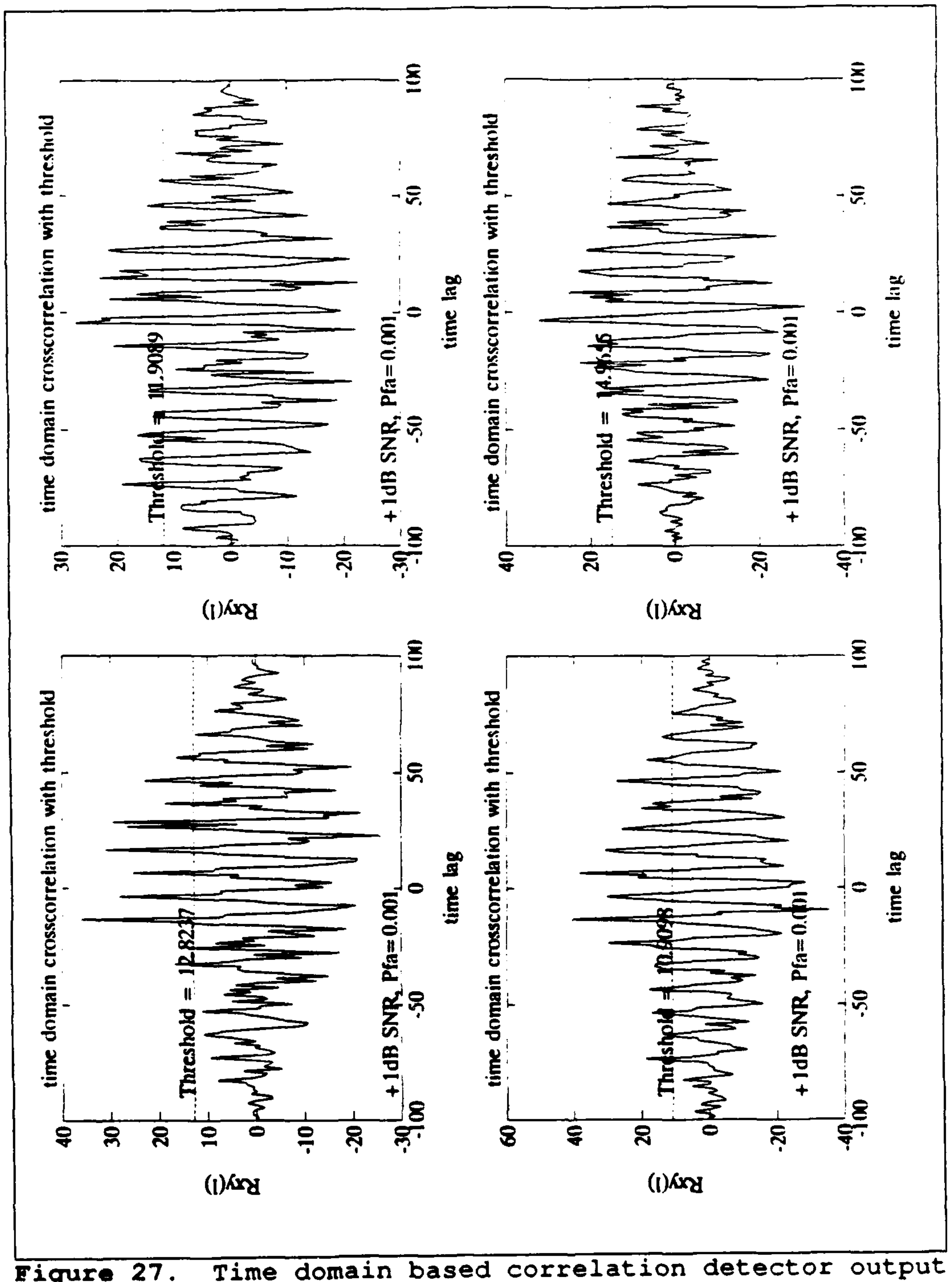


Figure 27. Time domain based correlation detector output for a SNR of 1dB and a P_{fa} of 0.001.

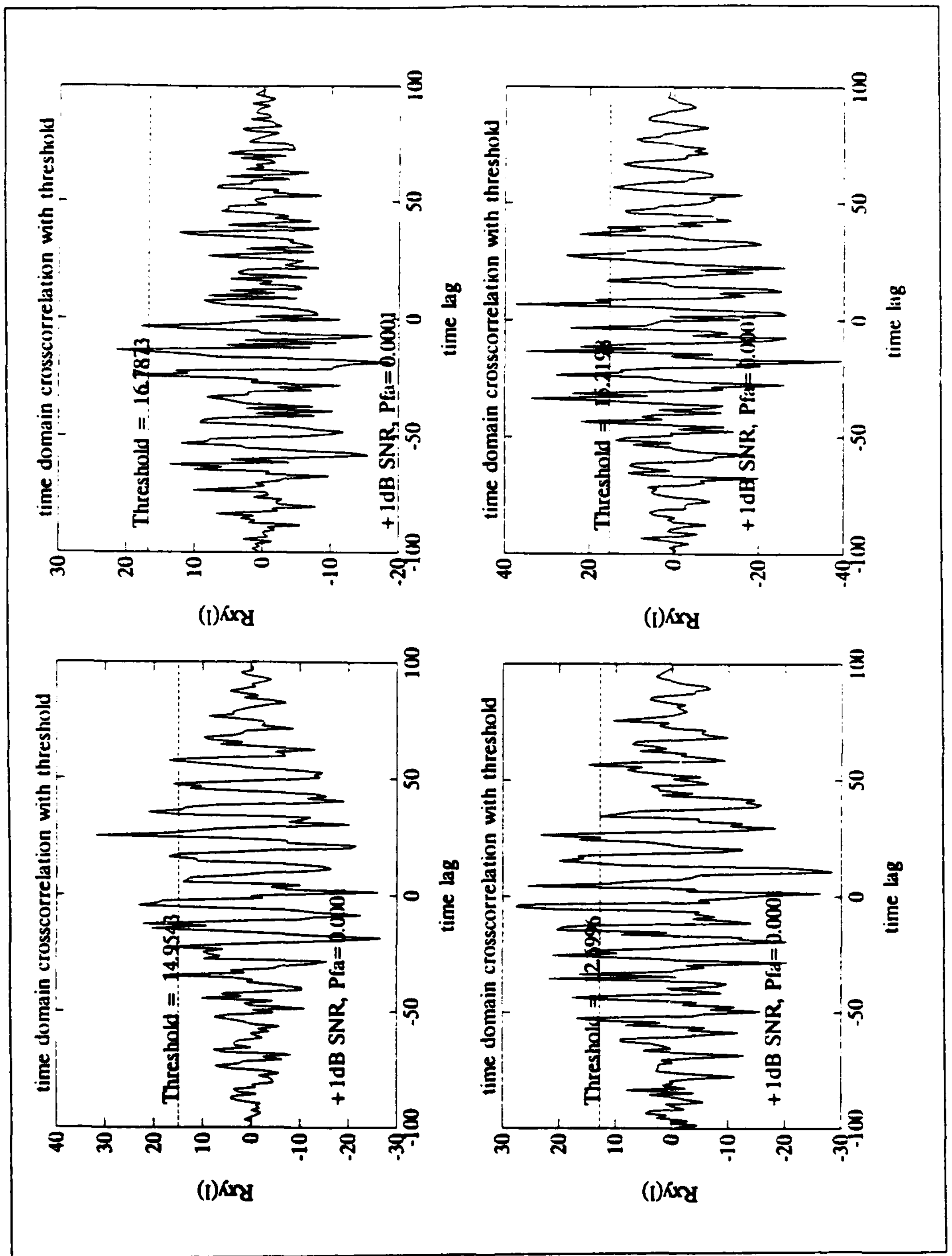


Figure 28. Time domain based correlation detector output for a SNR of 1dB and a P_{fa} of 0.0001.

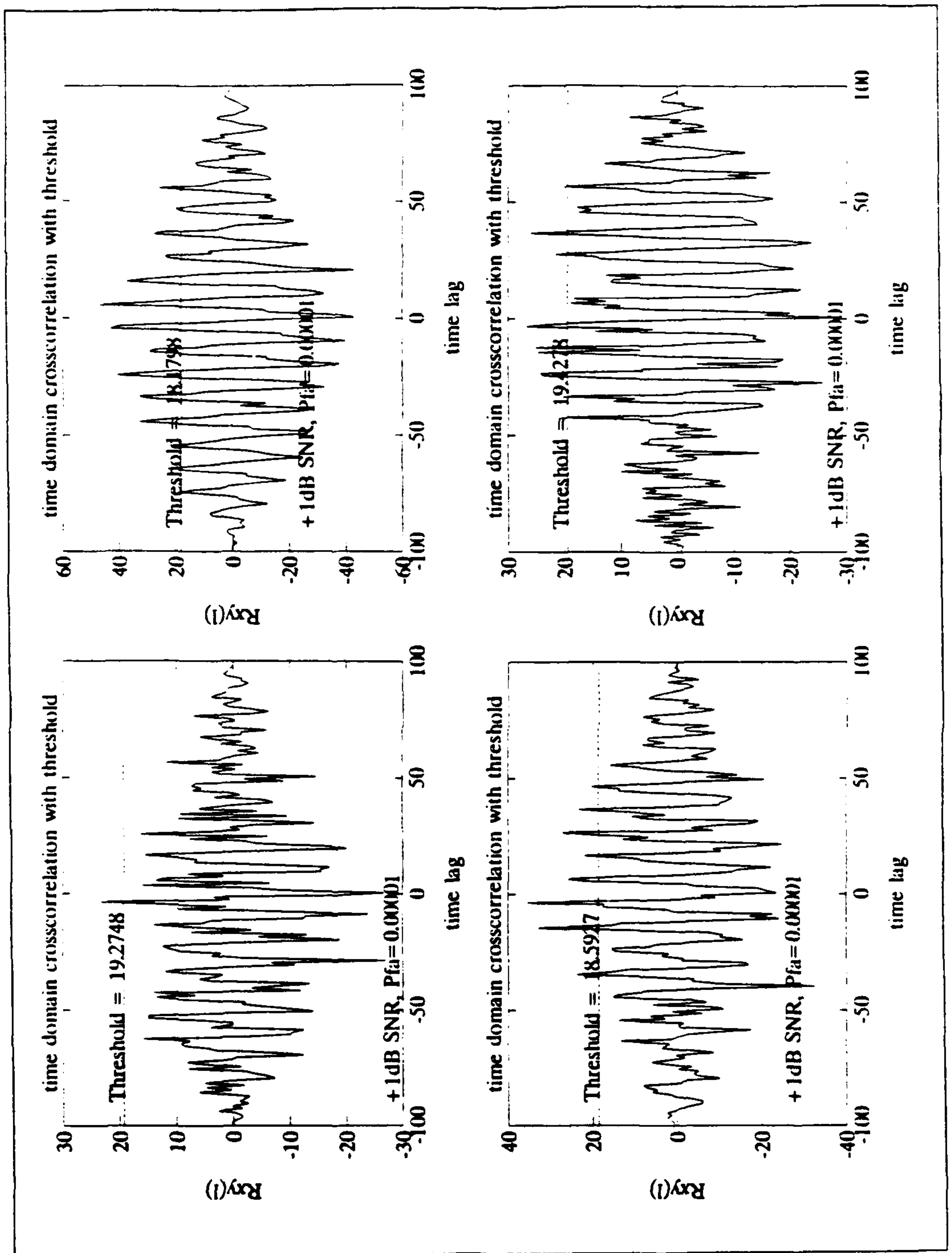


Figure 29. Time domain based correlation detector output for a SNR of 1dB and a P_{fa} of 0.00001.

APPENDIX D. FFT OUTPUT NOISE TO SIGNAL RATIO

A. INTRODUCTION

The noise to signal ratio for thirteen passes through a FFT butterfly are calculated below. The calculation assumes the word length of the arithmetic logic unit (ALU) of the computer is 16 bits long. The length of the input data points are 12 bits long. The data points are right justified when placed into the ALU. For a 16 bit ALU and a 12 bit data input, scaling occurs at the seventh pass and every other pass after that.

Pass 1

The noise power at the output of the butterfly after the first pass $N_A^2(1)$ is a function of

$$N_A^2(1) = 2N_A^2(0) + 2\sigma_s^2(0) + S^2\sigma_w^2(0) + \sigma_T^2(0)$$

$$\begin{aligned} \textit{where } S^2 &= \textit{signal power input to butterfly} \\ \sigma_T^2 &= \textit{truncation noise power} \\ \sigma_S^2 &= \textit{scaling noise power} \\ \sigma_w^2 &= \textit{sine/cosine table noise power} \\ N_A^2(0) &= \textit{noise power input to FFT} \ . \end{aligned} \tag{74}$$

During pass one there is no scaling, no truncation, and the trig table is not used. These error terms are zero. Equation 74 reduces to

$$\begin{aligned} N_A^2(1) &= 2N_Q^2 \\ \textit{where } N_Q^2 &= \frac{1}{3}2^{-2B} \\ &= \textit{noise input to butterfly} \quad . \end{aligned} \tag{75}$$

Pass 2

The noise power at the output of the butterfly after the second pass $N_A^2(2)$ is a function of

$$N_A^2(2) = 2N_A^2(1) + 2\sigma_S^2(1) + 2S^2\sigma_w^2(1) + \sigma_T^2(1) \quad . \tag{76}$$

During pass two scaling and truncation are not performed, and the trig table is not used. These noise terms are zero. Substituting Equation 75 into 76

$$\begin{aligned} N_A^2(2) &= 2(2N_Q^2) \\ &= 2^2N_Q^2 \quad . \end{aligned} \tag{77}$$

Pass 3

The noise power at the output of the butterfly after the third pass $N_A^2(3)$ is a function of

$$N_A^2(3) = 2N_A^2(2) + 2\sigma_S^2(2) + 2^2S^2\sigma_w^2(2) + \sigma_T^2(2) \quad . \tag{78}$$

Scaling noise is zero because it is not performed during pass three. Substituting Equation 77 into 78

$$\begin{aligned} N_A^2(3) &= 2(2^2N_Q^2) + 2^2S^2\sigma_w^2 + \sigma_T^2 \\ &= S^3N_Q^2 + 2^2S^2\sigma_w^2 + \sigma_T^2 \ . \end{aligned} \tag{79}$$

Pass 4

The noise power at the output of the butterfly after the fourth pass $N_A^2(4)$ is a function of

$$N_A^2(4) = 2N_A^2(3) + 2\sigma_S^2(3) + 2^3S^2\sigma_w^2(3) + \sigma_T^2(3) \ . \tag{80}$$

Scaling noise is zero because it is not performed during pass four. Substituting Equation 79 into 80

$$N_A^2(4) = 2^4N_Q^2 + 2^4S^2\sigma_w^2 + 3\sigma_T^2 \ . \tag{81}$$

Pass 5

The noise power at the output of the butterfly after the fifth pass $N_A^2(5)$ is a function of

$$N_A^2(5) = 2N_A^2(4) + 2\sigma_S^2(4) + 2^4S^2\sigma_w^2(4) + \sigma_T^2(4) \ . \tag{82}$$

Scaling noise is zero because it is not performed during pass five. Substituting Equation 81 into 82

$$N_A^2(5) = 2^5N_Q^2 + (2^5 + 2^4)S^2\sigma_w^2 + 7\sigma_T^2 \ . \tag{83}$$

Pass 6

The noise power at the output of the butterfly after the sixth pass $N_A^2(6)$ is a function of

$$N_A^2(6) = 2N_A^2(5) + 2\sigma_s^2(5) + 2^5 S^2\sigma_w^2(5) + \sigma_T^2(5) \quad . \tag{84}$$

Scaling noise is zero because it is not performed during pass six. Substituting Equation 83 into 84

$$N_A^2(6) = 2^6 N_q^2 + 2^7 S^2\sigma_w^2 + 15\sigma_T^2 \quad . \tag{85}$$

Pass 7

The noise power at the output of the butterfly after the seventh pass $N_A^2(7)$ is a function of

$$N_A^2(7) = 2N_A^2(6) + 2\sigma_s^2(6) + 2^6 S^2\sigma_w^2(6) + \sigma_T^2(6) \quad . \tag{86}$$

Scaling is performed for the first time during pass seven. Substituting Equation 85 into 86

$$N_A^2(7) = 2^7 N_q^2 + (2^8 + 2^6) S^2\sigma_w^2 + 31\sigma_T^2 + 2\sigma_s^2 \quad . \tag{87}$$

Pass 8

The noise power at the output of the butterfly after the eighth pass $N_A^2(8)$ is a function of

$$N_A^2(8) = 2N_A^2(7) + 2\sigma_s^2(7) + 2^7 S^2\sigma_w^2(7) + \sigma_T^2(7) \quad . \tag{88}$$

Scaling is not performed during this pass. Only the scaling noise from the previous pass is added. Substituting Equation 87 into 88

$$N_A^2(8) = 2^8 N_q^2 + (2^9 + 2^8) S^2\sigma_w^2 + 63\sigma_T^2 + 4\sigma_s^2 \quad . \tag{89}$$

Pass 9

The noise power at the output of the butterfly after the ninth pass $N_A^2(9)$ is a function of

$$N_A^2(9) = 2N_A^2(8) + 2\sigma_s^2(8) + 2^8 S^2\sigma_w^2(8) + \sigma_T^2(8) \quad . \tag{90}$$

Substituting Equation 89 into 90

$$N_A^2(9) = 2^9 N_q^2 + (2^{10} + 2^9 + 2^8) S^2\sigma_w^2 + 127\sigma_T^2 + 10\sigma_s^2 \quad . \tag{91}$$

Pass 10

The noise power at the output of the butterfly after the tenth pass $N_A^2(10)$ is a function of

$$N_A^2(10) = 2N_A^2(9) + 2\sigma_s^2(9) + 2^9 S^2\sigma_w^2(9) + \sigma_T^2(9) \quad . \tag{92}$$

Scaling is not performed this pass. Only the scaling noise from the previous pass is added. Substituting Equation 91 into 92

$$N_A^2(10) = 2^{10} N_q^2 + 2^{12} S^2\sigma_w^2 + 255\sigma_T^2 + 20\sigma_s^2 \quad . \tag{93}$$

Equation 93 describes the noise power at the output of a butterfly after the tenth data pass. The noise to signal ratio after the tenth pass through the butterfly is expressed as the ratio

$$\frac{N_A^2}{S^2}(10) = \frac{2^{10}N_q^2 + 2^{12}S^2\sigma_w^2 + 255\sigma_T^2 + 20\sigma_s^2}{2^{10}S^2} \quad . \qquad (94)$$

Dividing through by the denominator (signal power) each independent and uncorrelated noise term can be identified

$$\frac{N_A^2}{S^2}(10) = \frac{N_q^2}{S^2} + 4\sigma_w^2 + \frac{1}{4}\frac{\sigma_T^2}{S^2} + \frac{1}{51}\frac{\sigma_s^2}{S^2}$$

$$\begin{aligned} \textit{where } \frac{N_A^2}{S^2}(10) &= \textit{output Noise to Signal ratio} \\ \frac{N_q^2}{S^2} &= \textit{input Noise to Signal ratio} \qquad (95) \\ 4\sigma_w^2 &= \textit{sine/cosine table noise} \\ \frac{1}{4}\frac{\sigma_T^2}{S^2} &= \textit{truncation Noise to Signal ratio} \\ \frac{1}{51}\frac{\sigma_s^2}{S^2} &= \textit{scaling Noise to Signal ratio} \ . \end{aligned}$$

The noise to signal ratio calculation for 13 passes (2^{13} = 8192 points) through the butterfly algorithm continues

Pass 11

The noise power at the output of the butterfly after the 11th pass $N_A^2(11)$ is a function of

$$N_A^2(11) = 2N_A^2(10) + 2\sigma_s^2(10) + 2^{10}S^2\sigma_w^2(10) + \sigma_T^2(10) \quad . \qquad (96)$$

Substituting Equation 93 into 96

$$N_A^2(11) = 2^{11}N_q^2 + (2^{13} + 2^{10})S^2\sigma_w^2 + 511\sigma_T^2 + 42\sigma_s^2 \quad . \qquad (97)$$

Pass 12

The noise power at the output of the butterfly after the 12th pass $N_A^2(12)$ is a function of

$$N_A^2(12) = 2N_A^2(11) + 2\sigma_S^2(11) + 2^{11}S^2\sigma_W^2(11) + \sigma_T^2(11) \quad . \tag{98}$$

Scaling is not performed this pass. Only scaling noise from the previous pass is added. Substituting Equation 97 into 98

$$N_A^2(12) = 2^{12}N_q^2 + (2^{14} + 2^{12})S^2\sigma_W^2 + 1023\sigma_T^2 + 84\sigma_S^2 \quad . \tag{99}$$

Pass 13

The noise power at the output of the butterfly after the 13th pass $N_A^2(13)$ is a function of

$$N_A^2(13) = 2N_A^2(12) + 2\sigma_S^2(12) + 2^{12}S^2\sigma_W^2(12) + \sigma_T^2(12) \quad . \tag{100}$$

Substituting Equation 99 into 100

$$N_A^2(13) = 2^{13}N_q^2 + (2^{15} + 2^{13} + 2^{12})S^2\sigma_W^2 + 2047\sigma_T^2 + 170\sigma_S^2 \quad . \tag{101}$$

Equation 101 describes the noise power at the output of a butterfly after the 13th data pass. The noise to signal ratio after the 13th pass is expressed as the ratio

$$\frac{N_A^2}{S^2}(13) = \frac{2^{13}N_q^2 + (2^{15} + 2^{13} + 2^{12})S^2\sigma_W^2 + 2047\sigma_T^2 + 170\sigma_S^2}{2^{13}S^2} \quad . \tag{102}$$

Dividing through by the denominator, each independent and uncorrelated noise term can be identified

$$\frac{N_A^2}{S^2}(13) \cong \frac{N_q^2}{S^2} + 5.5\sigma_W^2 + \frac{1}{4}\frac{\sigma_T^2}{S^2} + \frac{1}{48.2}\frac{\sigma_S^2}{S^2}$$

where $\frac{N_A^2}{S^2}(13)$ = *output Noise to Signal ratio*

$\frac{N_q^2}{S^2}$ = *input Noise to Signal ratio* (103)

$5.5\sigma_W^2$ = *sine/cosine table noise*

$\frac{1}{4}\frac{\sigma_T^2}{S^2}$ = *truncation Noise to Signal ratio*

$\frac{1}{48.2}\frac{\sigma_S^2}{S^2}$ = *scaling Noise to Signal ratio* .

LIST OF REFERENCES

1. Sonnenberg G.J., Radar and Electronic Navigation, D. Van Nostrand Company, INC., 1951.

2. Adamo R.C., Kalman Filtering in the Spectral Domain, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1991.

3. Skolnik M.I., Radar Handbook, McGraw-Hill, 1990.

4. Skolnik M.I., Introduction to Radar Systems, McGraw-Hill, 1980.

5. Oppenheim A.V., Digital Signal Processing, Prentice-Hall, 1975.

6. Bendat J.S. and Piersol A.G., Random Data Analysis and Measurement Procedures, Wiley-Interscience, 1986.

7. harris f.j., Working papers, 2 May 1991.

8. harris f.j., Working papers, 9 August 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 22304-6145

2. Library, Code 52 — 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Chairman, Code EC — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Ralph Hippenstiel, Code EC/Hi — 3
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor Harold Titus, Code EC/Ti — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Professor Roberto Cristi, Code EC/Cx — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

7. Professor Charles Therrien, Code EC/Th — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

8. Mr. Frank Mika — 2
   3246 Test Wing
   Electronic Warfare Test Division, TZW
   Eglin AFB, FL 32542-5000

9. Mr. George Barrow — 1
   3246 Test Wing
   Range Systems Directorate, TFRR
   Eglin AFB, FL 32542-5000

10.	Naval Ocean Systems Center ATTN: Dr. C.E. Persons, Code 732 San Diego, CA 92152	1