Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

A MODIFIED LRT-BASED SPREAD-SPECTRUM RECEIVER USING SPATIAL AND TEMPORAL PROCESSING

by
Jeffrey L. Cutcher

The problem of demodulating a direct-sequence (DS) spread-spectrum signal in the presence of single-tone or narrow-band interference and multi-path is discussed. A likelihood-ratio test (LRT) receiver is presented which consists of a whitening filter and a RAKE correlator.

A modified LRT receiver structure is then considered where the whitening filter is replaced by an antenna array with corresponding tap coefficients. The array spatially removes the interference by estimating it's angle-of-arrival. Using the array has an advantage over the original LRT receiver when a narrow-band interference is present. Both receivers are identical in performance under the single-tone interference model.

A third receiver structure is considered in which two LRT receivers are placed in parallel and each receiver is assumed to receive the transmitted signal via independent paths. The correlator outputs are then summed and fed to a common slicer for decision making. The decisions, or estimated bits, are fed back to both receivers.

The recursive least-squares (RLS) algorithm was used to simulate the receivers. Bit error rates (BER) were plotted under the single-tone and narrow-band interference models as well as other parameters.

A MODIFIED LRT-BASED SPREAD-SPECTRUM RECEIVER USING SPATIAL AND TEMPORAL PROCESSING

by
Jeffrey L. Cutcher

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

May 1995

APPROVAL PAGE

A MODIFIED LRT-BASED SPREAD-SPECTRUM RECEIVER USING SPATIAL AND TEMPORAL PROCESSING

Jeffrey L. Cutcher

Dr. Alexander Haimovich, Thesis Advisor — Date
Associate Professor, NJIT

Dr. Y. Bar-Ness, Committee Member — Date
Distinguished Professor, NJIT

Dr. Zoran Siveski, Committee Member — Date
Assistant Professor, NJIT

BIOGRAPHICAL SKETCH

Author: Jeffrey L. Cutcher

Degree: Master of Science in Electrical Engineering

Date: May 1995

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1995
- Bachelor of Science in Electrical Engineering,
  New Jersey Institute of Technology, 1994
- Electronic Technician Diploma,
  DeVRY Technical Institute, Woodbridge, NJ, 1989

Major: Electrical Engineering

This work is dedicated to my parents Robert and Kathleen Cutcher. Without their support this would never have been posssible

ACKNOWLEDGMENT

I wish to thank the following people for their invaluable help:

Jan Punt

Murat Berin

Mehmet Tazebay

Amit Shah

A.(Murali) Arulambalam

Nico van Waes

Chris Peckham

Lisa Fitton

and the many others in the lab.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Wireless communications gives one the advantage of communicating, whether it's voice or data, through an air or underwater channel [4], without the need for some physical connection to a particular network. In wireless communications we can have stationary users or mobile users. The term *mobile* is used to relate to the fact that communications is done between a base station and a moving vehicle or between two vehicles.

Receiving a signal while mobile results in fading. This is due largely to multi-path effects where the receiver sees a superposition of delayed versions of the transmitted signal. In an analog system the user actually hears the effects of multi-path when receiving a voice message. Sometimes the signal will momentarily enter a deep-fade and the user will not be able to comprehend that part of the message. Providing a digital service allows the use of adaptive filter techniques to combat multi-path fading. In a digital system the user would not hear the actual fading but will experience drop-outs in the event of very deep fades.

The second problem to be discussed is the effects of intentional or un-intentional interference. An intentional interference is some high-power narrow-band process generated by an *enemy* source. This is mostly seen in a military scenario. Un-intentional interferences are the result of existing communication services. In the current literature the term *overlay* is used and can be seen in Figure (1.1). This means some users will be on the existing analog system while new users will be using the spread spectrum system. It's therefore beneficial to design a receiver for the spread spectrum system so that it can take into account these narrow-band signals.

To the spread spectrum user these narrow-band signals are considered an interference. To the analog services the spread spectrum signal appears to be noise-like.

Replacing an analog communication system with a digital system gives the designers a new set of tools. An important tool is digital signal processing, which leads to adaptive filter theory. Much has been said about interference rejection and digital whitening techniques [8, 9, 7, 5, 10, 14]. The study of multi-path fading has also been abundant [12]. The use of RAKE receivers, adaptive equalizers, and diversity techniques [11, 16, 12, 13] has been widely studied for combating multi-path. Ronald A. Iltis [6] has proposed a receiver that does both interference rejection and multi-path channel estimation and is the basis for this thesis.

We propose to modify Iltis' receiver by replacing the whitening filter with an antenna array. We refer to LRT to mean the original receiver design with a whitening filter and $ARRAY$ to be our modified receiver design. Using multiple antennae can give us interference rejection by estimating the angle-of-arrival of the interfering signal. Because the interfering signal is considered to be narrow-band, estimation in space or time is possible.

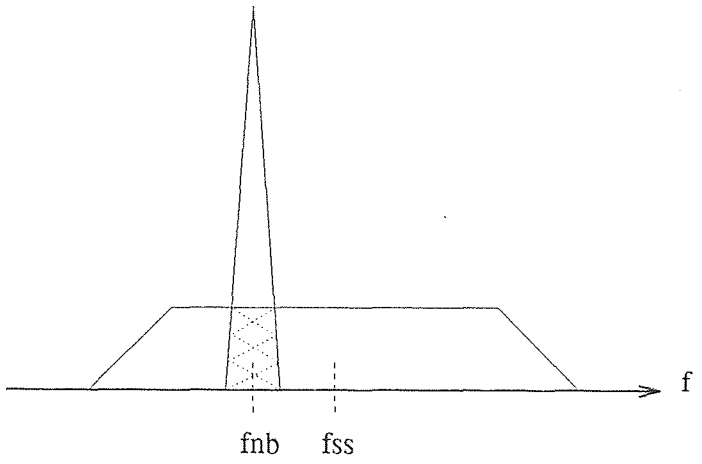

Figure 1.1 Overlay of a Narrow band Signal and a Spread Spectrum Signal

CHAPTER 2

CHANNEL AND SIGNAL MODELS

Below are the mathematical definitions to be used and are represented in the time domain. When necessary, a given signal will be represented in discrete time by letting $t = nT_s$.

2.1 Transmitted Signal

The transmitted signal is modeled as a Direct-Sequence Spread Spectrum (DSSS) signal and is represented in it's baseband form as

$$s(t) = A \sum_{i=0}^{N_b-1} d_i \, c(t - i\,T_b) \tag{2.1}$$

where N_b is the total number of transmitted bits $\{d_i\}$, T_b is the bit duration, and A is the amplitude. The chip waveform is

$$c(t) = \sum_{l=0}^{L_c-1} c_l \, p(t - l\,T_c) \tag{2.2}$$

where L_c is the total number of chips, $\{c_l\}$ is the spreading sequence, and $p(t)$ is the transmitted pulse. Also, Equation (2.2) is defined such that

$$\int_{T_b} c^2(t)\,dt = 1 \tag{2.3}$$

The bandwidth W of $s(t)$ is approximated as $\frac{1}{T_c}$ and the sampling interval is $\frac{1}{2W}$. Thus T_s is equal to $\frac{T_c}{2}$.

2.2 Channel Model

The channel is modeled as a frequency-selective slowly-fading channel [12] with impulse response

$$h_c(t) = \sum_{l=0}^{N_c-1} b_l \, \delta(t - l T_s) \tag{2.4}$$

where N_c is the number of multi-path components and T_s is the sampling interval. The channel is assumed to be wide-sense stationary (WSS) with uncorrelated scattering and can be represented by a tapped-delay line whose coefficients $\{b_l\}$ are Rayleigh distributed, zero-mean and unit variance. The tap spacing is T_s, which equals $\frac{T_c}{2}$ as previously noted.

The Rayleigh distribution [3] is given as

$$f_R(r) = \frac{r}{\sigma^2} exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{2.5}$$

for $r \geq 0$ and 0 elsewhere. We chose $\sigma^2 = 0.2$ which shifts the distribution to the left of unity. This implies that at any given instance the attenuation of a particular path is less than unity. Using $\sigma^2 = 1$ would also suffice but it's mean is roughly unity and thus at a given instance the channel attenuation could be greater than or less than unity. We chose not to allow the channel attenuation to be greater than unity on average. A histogram of the distribution for an ensemble of 1000 coefficients is provided in Figure (5.1) of Chapter 5.

2.3 Channel Assumptions

The reader is urged to refer to [6] and [13] for more information on the slowly fading assumption of the channel. The relevant information will be repeated here.

Given that the Doppler spread f_d is small compared to the information bandwidth $\frac{1}{T_b}$ and that the multi-path spread T_m is less than T_b and much greater than the chip duration T_c, then the channel coefficients are assumed to be constant

over several bit durations. This leads to an adaptive approach to the problem of receiving a DSSS signal effected by multi-path. If these conditions are not met then the receiver could not properly estimate the channel and performance would be at a minimum.

2.4 Received Signal

The received signal is Equation (2.1) convolved with Equation (2.4) plus Additive White Gaussian Noise (AWGN) and interference and is given by

$$r(t) = \sum_{l=0}^{N_c-1} b_l \, s(t - l\frac{T_c}{2}) + j(t) + n(t) \tag{2.6}$$

where the sum $j(t)+n(t)$ is assumed to be exactly modeled as an Nth order circular Gaussian autoregressive (AR) process [6].

2.5 Interference Model

The narrow-band interference is modeled as the superposition of complex sinusoids and is given as

$$j(t) = \sum_{k=1}^{N_j} A_k \, e^{i(\omega_k t + \psi_k)} \tag{2.7}$$

where N_j is the total number of sinusoids, A_k, ω_k, and ψ_k are the kth amplitude, frequency, and phase respectively.

2.6 Array Models

The transmitted signal is modeled exactly the same as Equation (2.1). The channel model is also the same except that each antenna gets its own set of channel coefficients which are denoted by $\{b_{l,m}\}$ for the l th antenna and the m th channel path.

For the antenna array (Figure (4.1)), the first antenna is numbered zero. This serves as a reference to the other array elements. The received signal is modeled as

$$r_l(t) = \sum_{m=0}^{N_c-1} b_{l,m}\, s(t - m\,\frac{T_c}{2}) + j_l(t) + n_l(t) \quad 0 \leq l \leq N_a - 1 \tag{2.8}$$

where

$$j_l(t) = j(t)\, e^{ilU}, \quad U = 2\pi \tfrac{d}{\lambda} sin(\theta_j) \tag{2.9}$$

in which U denotes the phase associated with θ_j, the angle of arrival of the interference, d, the array element spacing in meters and λ, the wavelength of the received interference in meters.

CHAPTER 3

THE LRT RECEIVER

A single-antenna LRT receiver structure can be seen in Figure (3.1). The reader may refer to [15] for more information on the LRT and the GLRT (Generalized). Iltis [6] has designed a GLRT receiver based on Differential Phase Shift Keying (DPSK) and as presented here binary signalling such as Binary Phase Shift Keying (BPSK) is assumed for simplicity. Iltis went to great lengths to derive the GLRT receiver and therefore would be inappropriate to duplicate here. We also do not consider the phase and so we dropped the 'G' in GLRT. Thus, the essentials of the LRT using BPSK will be presented. The adaptive array modification will then follow.

3.1 LRT Receiver Derivation

The sampled versions of Equations (2.1) and (2.6) are

$$s(k) = A \sum_{i=0}^{N_b-1} d_i c(k - i2L_c) \tag{3.1}$$

and

$$r(k) = \sum_{l=0}^{N_c-1} b_l s(k-l) + j(k) + n(k) \tag{3.2}$$

where $T_b = L_c T_c$ and $T_s = \frac{T_c}{2}$. The cost function for our adaptive filter is

$$J = \sum_{k=0}^{N_s-1} \left| \sum_{n=0}^{N_\alpha-1} \alpha_n r(k-n) - \sum_{n=0}^{N_\beta-1} \beta_n \hat{s}(k-n) \right|^2 \tag{3.3}$$

where N_s is the total number of samples, N_α is the size of the whitening filter with Maximum Likelihood (ML) estimates $\{\alpha_n\}$, N_β is the size of the RAKE-combiner

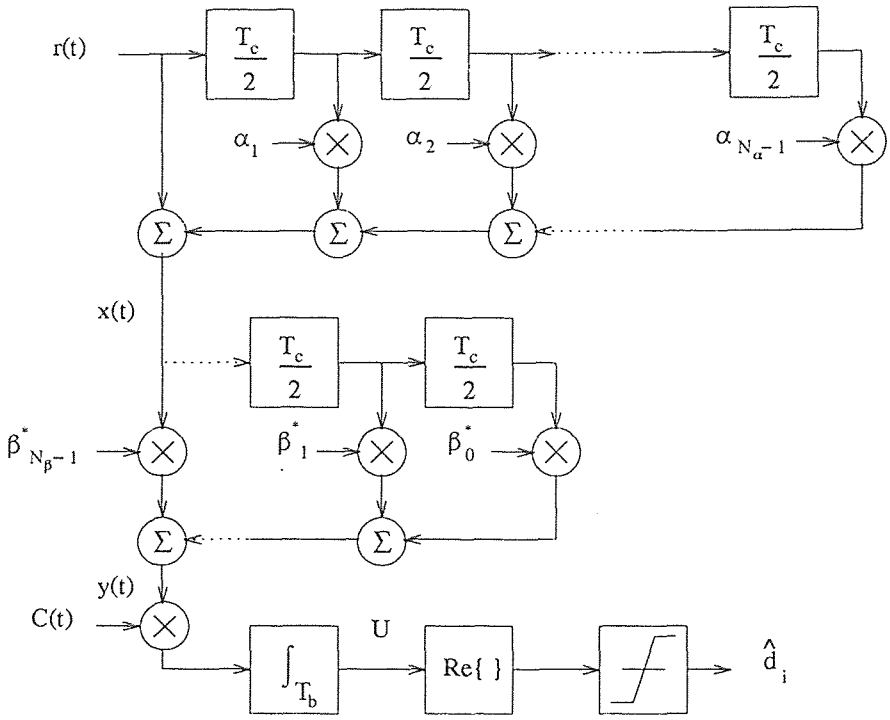

Figure 3.1 LRT Receiver

with ML estimates $\{\beta_n\}$, and $\hat{s}(k)$ is Equation (3.1) with $\{d_i\}$ replaced with the estimated bits $\left\{\hat{d}_i\right\}$ at the output of the slicer.

From the minimization of Equation (3.3) we get

$$\sum_{n=0}^{N_\alpha-1} \alpha_n \sum_{k=0}^{N_s-1} r(k-n)r^*(k-l) = \sum_{n=0}^{N_\beta-1} \beta_n \sum_{k=0}^{N_s-1} r^*(k-l)\hat{s}(k-n) \qquad (3.4)$$

for $l = 1, 2, \cdots, N_\alpha - 1$ and

$$\sum_{n=0}^{N_\alpha-1} \alpha_n \sum_{k=0}^{N_s-1} r(k-n)s^*(k-l) = \sum_{n=0}^{N_\beta-1} \beta_n \sum_{k=0}^{N_s-1} \hat{s}^*(k-l)\hat{s}(k-n) \qquad (3.5)$$

for $l = 0, 1, 2, \cdots, N_\beta - 1$. Note that in Equation (3.4) l starts at 1 because $\alpha_0 = 1$.

With $\{d_i\} \in \{-1, 1\}$, the hypotheses are

$$\mathrm{H}_1 \ : \ d_{N_b} = +1$$

$$\mathrm{H}_0 \ : \ d_{N_b} = -1$$

Iltis [6] shows that the sample correlation functions can be replaced by statistical correlation functions given that $N_b \gg 1$. The above hypotheses yield the same results as Iltis' hypotheses for DPSK, thus our likelihood ratio becomes

$$\Lambda = \frac{exp\left(\frac{1}{2\sigma_e^2}\sum_{k=2L_cN_b-1}^{2L_cN_b-1}\left|\boldsymbol{\alpha}^T\boldsymbol{r}_k - \boldsymbol{\beta}^T\boldsymbol{s}_{1k}\right|^2\right)}{exp\left(\frac{1}{2\sigma_e^2}\sum_{k=2L_c(N_b-1)}^{(2L_cN_b-1)}\left|\boldsymbol{\alpha}^T\boldsymbol{r}_k - \boldsymbol{\beta}^T\boldsymbol{s}_{0k}\right|^2\right)} \tag{3.6}$$

where the summations are over the samples of the last bit and σ_e is the variance of $e(t)$, the whitened interference plus noise given as

$$e(t) = \sum_{m=0}^{N_\alpha-1}\alpha_m\left[j\left(t - m\frac{T_c}{2}\right) + n\left(t - m\frac{T_c}{2}\right)\right] \tag{3.7}$$

and the vectors $\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T, \boldsymbol{r}_k$ and $\boldsymbol{s}_{1,0k}$ are defined as

$$\begin{aligned}
\boldsymbol{\alpha}^T &= [1, \alpha_1, \cdots, \alpha_{N_\alpha-1}] \\
\boldsymbol{\beta}^T &= \left[\beta_0, \beta_1, \cdots, \beta_{N_\beta-1}\right] \\
\boldsymbol{r}_k &= [r(k), r(k-1), \cdots, r(k-N_\alpha+1)]^T \\
\boldsymbol{s}_{1k} &= [s_1(k), s_1(k-1), \cdots, s_1(k-N_\beta+1)]^T \\
\boldsymbol{s}_{0k} &= [s_0(k), s_0(k-1), \cdots, s_0(k-N_\beta+1)]^T
\end{aligned}$$

where the subscript 1,0 of the vector $\boldsymbol{s}$ denotes hypotheses 1 and 0 respectively. From Equation (3.6) we notice that

$$\left|\boldsymbol{\beta}^T\boldsymbol{s}_{1,k}\right|^2 = \left|\boldsymbol{\beta}^T\boldsymbol{s}_{0,k}\right|^2. \tag{3.8}$$

The threshold for Λ is 1 and after combining terms and taking the natural logarithm of both sides we get at bit i

$$U_i = Re\left\{\sum_{m=0}^{N_\beta-1}\beta_m^*\int_{T_b}\sum_{n=0}^{N_\alpha-1}\alpha_n r\left(t-n\frac{T_c}{2}\right)c\left(t-iT_b-m\frac{T_c}{2}\right)\right\}dt. \tag{3.9}$$

If $U_i > 0$ we choose $\hat{d}_i = +1$ and if $U_i < 0$ we choose $\hat{d}_i = -1$.

3.2 Adaptive Algorithm

In order to simulate the LRT receiver we need to use some kind of algorithm to calculate the $\{\alpha\}$ and $\{\beta\}$ coefficients. The Recursive Least Squares (RLS) algorithm [1, 2] is easy to implement using vector and matrix notation and therefore allows the receiver to be simulated using MATLAB©. Figure (3.2) shows a block diagram of the simulation.

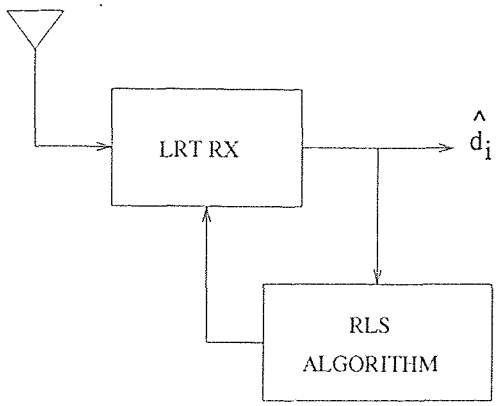

Figure 3.2 Simulation Block Diagram

Two operations are fulfilled in a given simulation and are performed in parallel. The first operation is the actual reception and detection of the transmitted signal. The second operation is the adaptive updating of the $\{\alpha\}$ and $\{\beta\}$ coefficients. This algorithm is now defined.

First we'll define the coefficient and data vectors as

$$\begin{aligned} \boldsymbol{w}^H(n) &= [-\alpha_1(n), -\alpha_2(n), \cdots, -\alpha_{N_\alpha-1}(n), \\ &\qquad \beta_0(n), \beta_1(n), \cdots, \beta_{N_\beta-1}(n)\big] \end{aligned} \tag{3.10}$$

$$\begin{aligned} \boldsymbol{u}(n) &= [r(n-1), r(n-2), \cdots, r(n-N_\alpha+1), \\ &\qquad \hat{s}(n), \hat{s}(n-1), \cdots, \hat{s}(n-N_\beta+1)]^T . \end{aligned} \tag{3.11}$$

The cost function [2, 1]

$$J_{RLS} = \sum_{n=0}^{N_s-1} \lambda^{N_s-1-n} |e(n)|^2 \tag{3.12}$$

is minimized by using the following equations [2]:

$$\begin{aligned} \boldsymbol{k}(n) &= \frac{\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{1+\lambda^{-1}\boldsymbol{u}^H(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)} \\ e(n) &= r(n) - \boldsymbol{w}^H(n-1)\boldsymbol{u}(n) \\ \boldsymbol{w}(n) &= \boldsymbol{w}(n-1) + \boldsymbol{k}(n)e^*(n) \\ \boldsymbol{P}(n) &= \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{u}^H(n)\boldsymbol{P}(n-1). \end{aligned} \tag{3.13}$$

In most adaptive receivers a training sequence is required in order for the receiver to start off in the right direction of the estimation process. Therefore the LRT receiver is fed this known data sequence to calculate the estimated transmitted signal $\hat{s}(n)$. After training the algorithm relies on the previous estimated data bits and the previous received signal $r(n-2L_c)$. Thus Equation (3.11) and (3.13) become:

$$\begin{aligned} \boldsymbol{u}(n) &= [r(n-\Gamma-1), r(n-\Gamma-2), \cdots, r(n-\Gamma-N_\alpha+1), \\ &\qquad \hat{s}(n-\Gamma), \hat{s}(n-\Gamma-1), \cdots, \hat{s}(n-\Gamma-N_\beta+1)]^T \\ e(n) &= r(n-\Gamma) - \boldsymbol{w}^H(n-1)\boldsymbol{u}(n) \end{aligned}$$

where $\Gamma = 2L_c$, which is the number of samples in one bit period.

CHAPTER 4

LRT WITH SPATIAL AND TEMPORAL PROCESSING

Different forms of diversity techniques exist which include frequency, time and space [12] diversity. These techniques use the fact that if we can receive a signal from several different paths of a channel then the probability that all of these signals will be affected in the same manner is unlikely. The LRT receiver in Figure (3.1) has a RAKE correlator as it's second filter. This RAKE correlator [12, 13] exhibits frequency diversity by the fact that we are receiving a wide-band signal.

The optimal conditions for the LRT receiver is when the size of the RAKE (number of taps) is $N_\beta = N_\alpha + N_c - 1$. This is due to the fact that the RAKE coefficients $\{\beta\}$ are equal to the convolution of the channel with the whitening filter. Iltis [6] has shown that any other combination of N_β and N_α yields poorer results. Ideally the RAKE tap size is equal to the tap size of the channel model ($N_\beta = N_c$) but because of the whitening filter we do not have this. By replacing this whitening filter with an antenna array we effectively remove this convolution and the RAKE tap size becomes exactly equal to the channel tap size.

The use of this array now gives us spatial processing versus the temporal processing of the whitening filter. One major advantage to the use of this array is that it will reduce the interference by estimating it's angle of arrival and thus subtract the interference from the reference signal at antenna zero. The simulations show that for a narrow-band signal (not single-tone) the array with only two antennae outperforms the original temporal design. With a single-tone interference the array receiver shows roughly 2dB better performance over the LRT receiver. This however us due to the fact that we chose a single frequency for the interference and we did not

average over all possible frequencies ($\omega_j \in \{0, 2\pi\}$). Thus, averaging over all possible frequencies yields identical performance between the LRT and the ARRAY.

Suppose we had an interference that consisted of M sinusoids. The LRT with the whitening filter, since it is a prediction error filter, will need to have at least $M+1$ taps if we want to null out each frequency. Clearly if those frequencies are close enough, they can be attenuated by one null and thus we may use less than $M+1$ taps. The array, however, will only need it's two antennae and subsequently the two taps. If however we imposed upon the receivers more than one source of interference then clearly the array would have to be expanded in the same manner as the whitening filter. However, the performance between the two receivers suppressing a single narrow-band interferer is being considered.

The signal-to-noise ratio (SNR) at the output of the receivers in Figures (3.1) and (4.1) have an identical form equal to

$$\gamma = \frac{A^2 \boldsymbol{\beta}^H \boldsymbol{R} \boldsymbol{\beta}}{\sigma_e^2} \tag{4.1}$$

where σ_e^2 is the variance of the error portion of the output of the whitening filter or antenna array, $\boldsymbol{\beta}$ is a vector containing the RAKE coefficients, and $\boldsymbol{R}$ is the correlation matrix of $c(t)$. The derivation of Equation (4.1) for both receivers can be found in Appendix (A).

Another way of utilizing diversity is depicted in Figure (4.2). The outputs of the independent receivers are combined and used to estimate the transmitted data. The estimated bits $\left\{\hat{d}_i\right\}$ are fed back to each receiver's adaptive algorithm. For the case of two receivers we get

$$\gamma = \frac{A^2\zeta_1^2 + A^2\zeta_2^2 + 2A^2\zeta_1\zeta_2}{\sigma_{1e}^2\zeta_1 + \sigma_{2e}^2\zeta_2}. \tag{4.2}$$

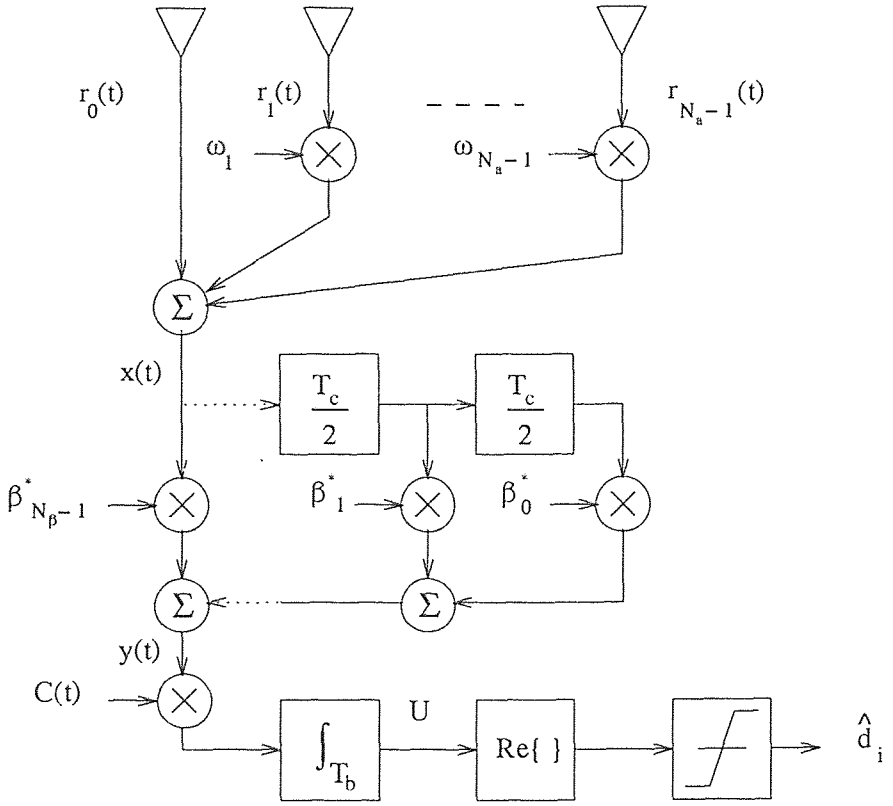

Figure 4.1 Modified LRT Receiver

where $\zeta_1 = \boldsymbol{\beta}_1^H \boldsymbol{R}_1 \boldsymbol{\beta}_1, \zeta_2 = \boldsymbol{\beta}_2^H \boldsymbol{R}_2 \boldsymbol{\beta}_2$, and σ_{1e}^2 and σ_{2e}^2 are defined the same as σ_e^2. For this derivation the reader may consult Appendix (A). If we assume an ideal case where both receivers have on average identical coefficients then we can calculate an upper bound on the SNR as

$$\gamma_o = \frac{2A^2 \boldsymbol{\beta}^H \boldsymbol{R} \boldsymbol{\beta}}{\sigma_e^2} \tag{4.3}$$

which gives us a 3dB performance gain compared to Equation (4.1). This form of reception, however, isn't practical since in this case we need two separate receivers which doubles the cost and complexity.

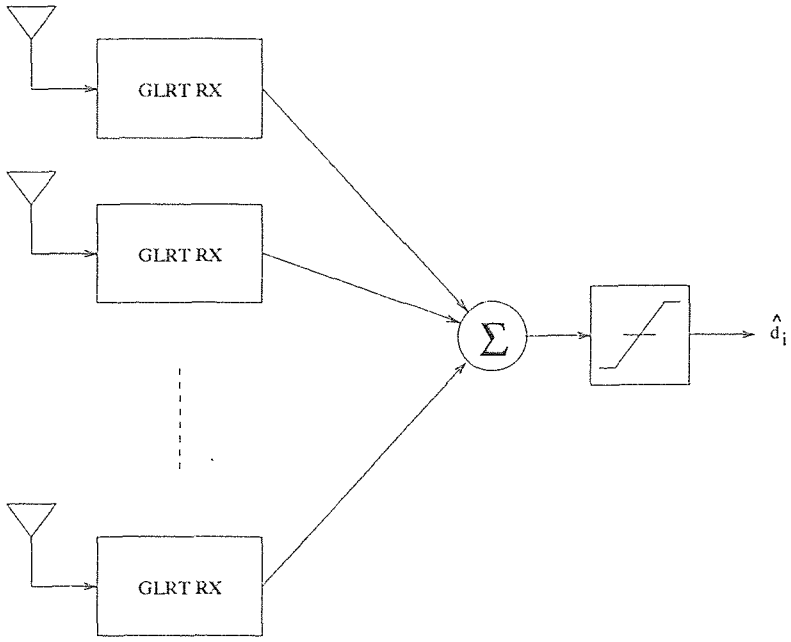

Figure 4.2 Linear Combination of Parallel Receivers

4.1 LRT Derivation for the ARRAY Receiver

Because of the fact that the array is equivalent to the whitening filter, the LRT derivation is similar to that shown in Section 3.1. Equation (3.6) remains as

$$\Lambda = \frac{exp\left(\frac{1}{2\sigma_e^2}\sum_{k=2L_c(N_b-1)}^{2L_cN_b-1}\left|\boldsymbol{\alpha}^T\boldsymbol{r}_k - \boldsymbol{\beta}^T\boldsymbol{s}_{1k}\right|^2\right)}{exp\left(\frac{1}{2\sigma_e^2}\sum_{k=2L_c(N_b-1)}^{(2L_cN_b-1)}\left|\boldsymbol{\alpha}^T\boldsymbol{r}_k - \boldsymbol{\beta}^T\boldsymbol{s}_{0k}\right|^2\right)} \tag{4.4}$$

where the vectors are now defined as

$$\begin{aligned}
\boldsymbol{\alpha}^T &= [1, \omega_1, \cdots, \omega_{N_a-1}] \\
\boldsymbol{\beta}^T &= \left[\beta_0, \beta_1, \cdots, \beta_{N_\beta-1}\right] \\
\boldsymbol{r}_k &= [r_0(k), r_1(k), \cdots, r_{N_a-1}(k)]^T \\
\boldsymbol{s}_{1k} &= [s_1(k), s_1(k-1), \cdots, s_1(k-N_\beta+1)]^T
\end{aligned}$$

$$\boldsymbol{s}_{0k} = [s_0(k), s_0(k-1), \cdots, s_0(k-N_\beta+1)]^T.$$

The differences between the above vectors and those defined in Section 3.1 are that the $\{\alpha\}$'s are replaced with the array parameters $\{\omega\}$ and the time series of the received signal become a spatial series.

Finally, the decision variable for the array receiver is

$$U_i = Re\left\{\sum_{m=0}^{N_\beta-1}\beta_m^*\int_{T_b}\sum_{n=0}^{N_a-1}\omega_n r_n(t)\,c\left(t-iT_b-m\frac{T_c}{2}\right)\right\}dt. \tag{4.5}$$

4.2 Adaptive Algorithm

The RLS algorithm operates in the same fashion as stated in the previous chapter but with some minor changes. Therefore the modified algorithm is as follows:

$$\begin{aligned}
\boldsymbol{w}^H(n) &= [-\omega_1(n), -\omega_2(n), \cdots, -\omega_{N_a-1}(n), \\
&\qquad \beta_0(n), \beta_1(n), \cdots, \beta_{N_\beta-1}(n)\big] \\
\boldsymbol{u}(n) &= [r_1(n), r_2(n), \cdots, r_{N_a-1}(n), \\
&\qquad \hat{s}(n), \hat{s}(n-1), \cdots, \hat{s}(n-N_\beta+1)]^T \\
\boldsymbol{k}(n) &= \frac{\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{1+\lambda^{-1}\boldsymbol{u}^H(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)} \\
e(n) &= r_0(n) - \boldsymbol{w}^H(n-1)\boldsymbol{u}(n) \\
\boldsymbol{w}(n) &= \boldsymbol{w}(n-1) + \boldsymbol{k}(n)e^*(n) \\
\boldsymbol{P}(n) &= \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{u}^H(n)\boldsymbol{P}(n-1)
\end{aligned}$$

where N_a is the number of antennae.

Training applies to this receiver as well and so after training we have

$$\begin{aligned} \boldsymbol{u}(n) &= [r_1(n-\Gamma), r_2(n-\Gamma), \cdots, r_{N_a-1}(n-\Gamma), \\ &\qquad \hat{s}(n-\Gamma), \hat{s}(n-\Gamma-1), \cdots, \hat{s}(n-\Gamma-N_\beta+1)]^T \\ e(n) &= r_0(n-\Gamma) - \boldsymbol{w}^H(n-1)\boldsymbol{u}(n) \end{aligned}$$

where $\Gamma = 2L_c$ which is the number of samples in one bit period.

CHAPTER 5

NUMERICAL RESULTS

Three receiver configurations were simulated, the original LRT receiver, the modified LRT receiver (ARRAY), and the combination of two LRT receivers. Attention was put mostly on the LRT and ARRAY receivers for comparison. Unless specified, all simulations calculated the average probability of error with each experiment choosing a new set data bits, channel coefficients, interference phase and noise. To insure the best possible results for the simulations each experiment set the random generator seed to a scaled value of the real-time clock.

The channel was modeled as Rayleigh Fading (see Figure (5.1)) with four ($N_c = 4$) independent paths. In the case of the array receiver, each antenna had it's own set of channel coefficients and the antenna spacing was set to 10λ. However, because the receiver does not achieve additional diversity the antenna spacing does not have to be constrained to 10λ. If we choose $\frac{\lambda}{2}$, for example, the independent path assumption does not hold. This does not cause poorer performance and was verified under separate simulations.

The value of N_c was chosen based on assuming a signal bandwidth of $1.25MHz$. With another assumption that the coherence bandwidth of the channel is roughly $300kHz$ we get

$$N_c = \frac{1.25\text{x}10^6}{300\text{x}10^3} \approx 4. \tag{5.1}$$

The interference was modeled as a single-tone sinusoid for one set of simulations and a multi-tone signal for the other. This multi-tone signal comprised of the sum of five sinusoids close together in frequency so as to represent a narrow-band signal.

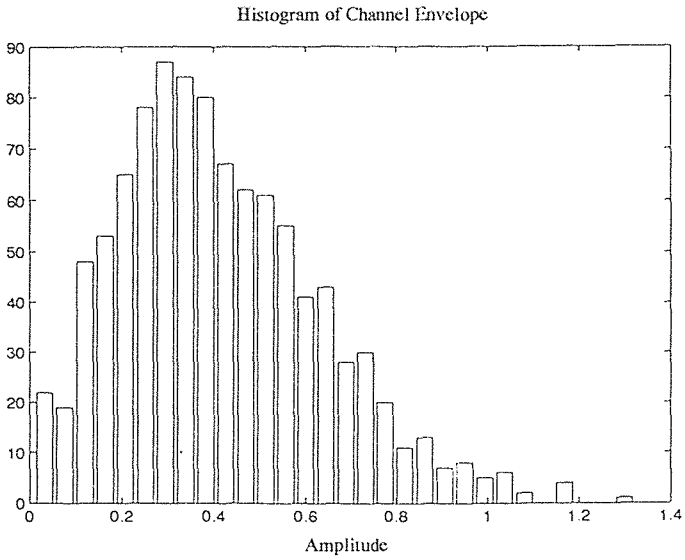

Figure 5.1 Histogram of Channel Attenuation Distribution

Both interferences had random phase and a power level of 20dB above the signal power. For the array receiver the interference had a random angle-of-arrival. The normalized frequency for the single-tone sinusoid was set to 1 rads/sec where 2π rads/sec is our transmitted signal bandwidth. In the case of the narrow-band interference the five frequencies were chosen to achieve 20 percent of the spread spectrum signal bandwidth, hence 1.25 rads/sec. This narrow-band signal can be seen in Figure (5.2).

The noise was modeled as a complex random variable with a normal distribution. The variance of the noise was set to unity and the transmitted signal power was varied.

Mention must be made in reference to how the actual filtering took place. That is, how was the output of the receivers defined? Equation (3.9) is an expression showing the output of the LRT receiver at the ith bit. To simplify this in the simulations we correlate the output of the RAKE with a delayed version of Equation (2.2) instead of correlating each tap individually. Letting the output of

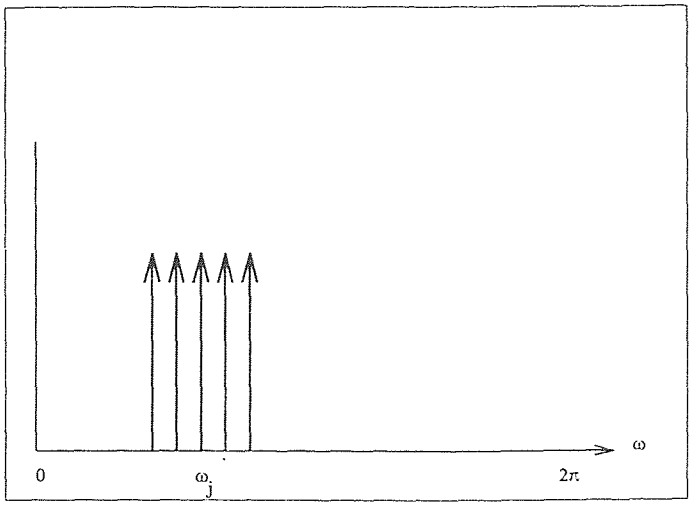

Figure 5.2 Narrow-band Interference Spectrum

the RAKE equal $y(t)$ we have

$$U_i = \int_{T_b} y(t) c \left(t - iT_b - (N_\beta - 1) \frac{T_c}{2} \right) dt. \tag{5.2}$$

This also holds for the ARRAY simulation.

The simulations measured the average SNR at the input of the slicer. The BER was calculated using those average SNR measurements with the error function. This was done under the assumptions that the output of the whitening filter is white with a Gaussian distribution. We may define the output SNR to be

$$SNR = \frac{E\left[Re\left\{U_i\right\}\right]^2}{VAR\left[Re\left\{U_i\right\}\right]}. \tag{5.3}$$

To easily calculate this we passed three sets of data through the receiver. The first set was our original received signal which is defined as Equation (2.6) for the LRT and Equation (2.8) for the ARRAY. The next set was the signal portion of the received signal only, and the final set was the interference plus noise only. From Appendix A we see that the numerator is the signal portion and the denominator is noise portion thus allowing us to easily calculate the SNR at the output of the RAKE correlator

given the two sets of data. Thus the SNR per bit is

$$\gamma_b = \frac{Re\{U_s\}^2}{Re\{U_n\}^2}. \tag{5.4}$$

These values are then averaged over all of the transmitted bits which gives us

$$\bar{\gamma} = \frac{Re\{U_s\}^2}{VAR[Re\{U_n\}]}. \tag{5.5}$$

The average probability of error (BER) becomes

$$BER = \frac{1}{2} erfc\left(\sqrt{\bar{\gamma}}\right). \tag{5.6}$$

These calculations are done for different transmitted SNRs. This whole process is then repeated several times with each experiment having a new set of transmitted bits, noise, phases, etc.

5.1 Simulation Plots

Table (5.1) provides some notes to the simulations that were carried out.

Table 5.1 Simulation Notes

FIG.	RECEIVER	NOTES
5.3	LRT	$N_\alpha = 3, N_\beta = 6, N_c = 4, J/S = 20dB, w = 1\, rads/sec$
5.4	ARRAY	$N_\beta = 4, N_c = 4, J/S = 20dB, w = 1\, rads/sec$
5.5	LRT/ARRAY	Comparison of above results
5.6	ARRAY	Single-Tone vs. NB ($N_a = 2$)
5.7	LRT/ARRAY	Single-Tone, Fixed Channel, Average over all frequencies
5.8	LRT/ARRAY	NB Int., LRT - $N_\alpha = 6, 11, N_\beta = 9, 14$
		NB Int., ARRAY - $N_a = 2, N_\beta = 4$

In Figure (5.3) simulations of the LRT-based receiver for the single case and dual case are shown. We see that when a single-tone interference is present and the LRT receiver is reduced to a RAKE correlator (no whitening filter), the receiver is

rendered useless with a BER of around 0.5. Under the conditions shown in Table (5.1) for the LRT we see fairly good performance. Using the combination of two receivers improves the performance by about 2-3dB.

For the ARRAY receiver we see from Figure (5.4) that with one antenna the receiver is not useable. Expanding the array to two antennae gives us very good performance. Figure (5.5) shows the LRT performance versus the ARRAY. For the case of a narrow-band interference the ARRAY performance roughly remains unchanged. This can be seen Figure (5.6).

In comparing the performance of the LRT with the ARRAY for the narrow-band interference we fix the channel and average over the interference phase and angle. The ARRAY has two antennae so we also chose another fixed channel for the second antenna. The results for single-tone and narrow-band interference can be seen in Figures (5.7) and (5.8). Notice that the LRT's whitening filter had to be expanded to 11 taps to combat the narrow-band interference, yet the ARRAY receiver was unchanged and performs better than the LRT.

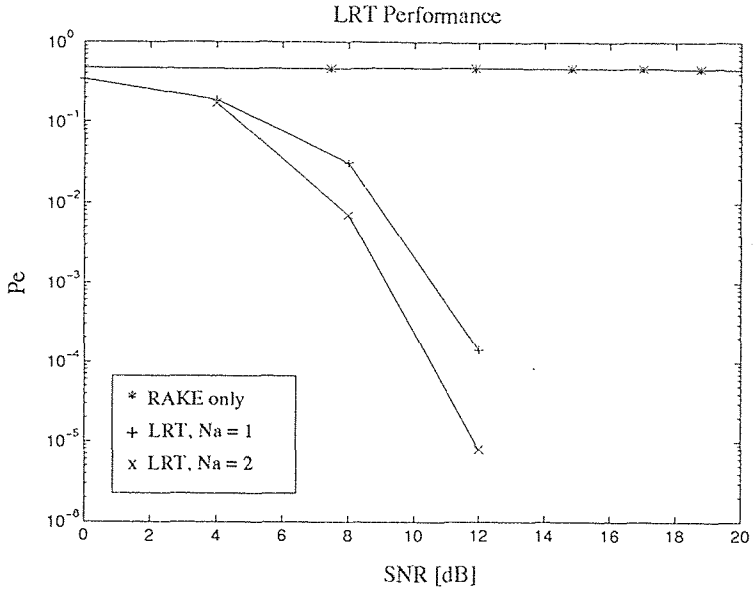

Figure 5.3 BER For LRT Receiver

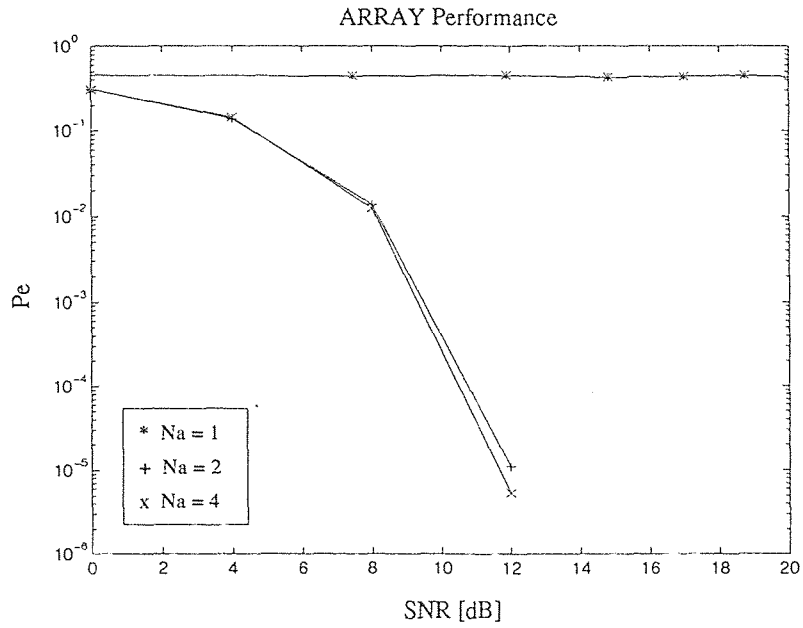

Figure 5.4 BER For ARRAY Receiver

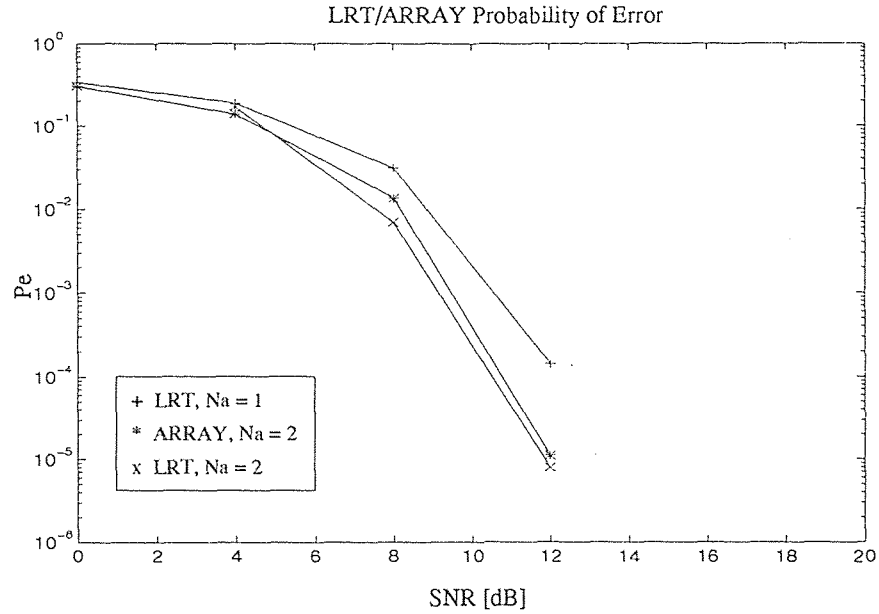

Figure 5.5 BER Comparison Between LRT and ARRAY

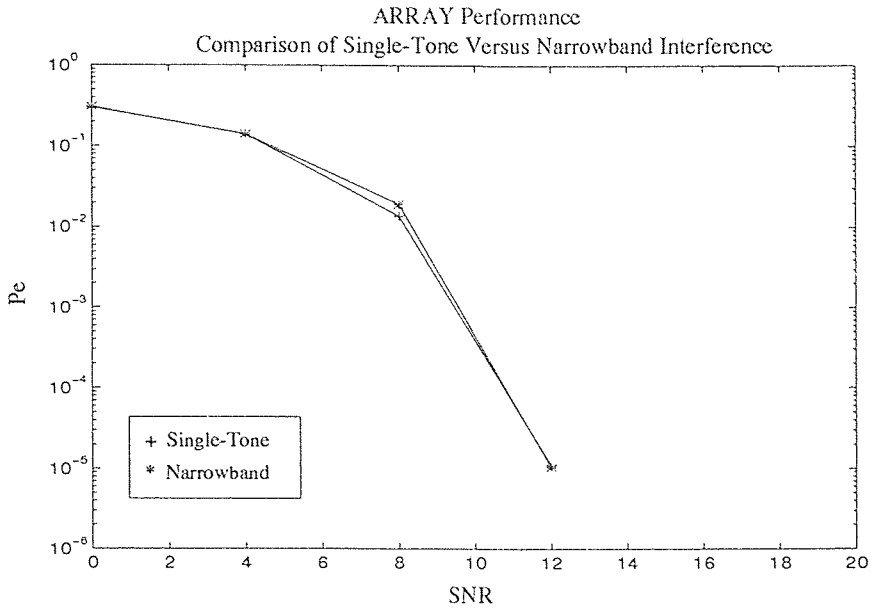

Figure 5.6 ARRAY ST vs NB Interference

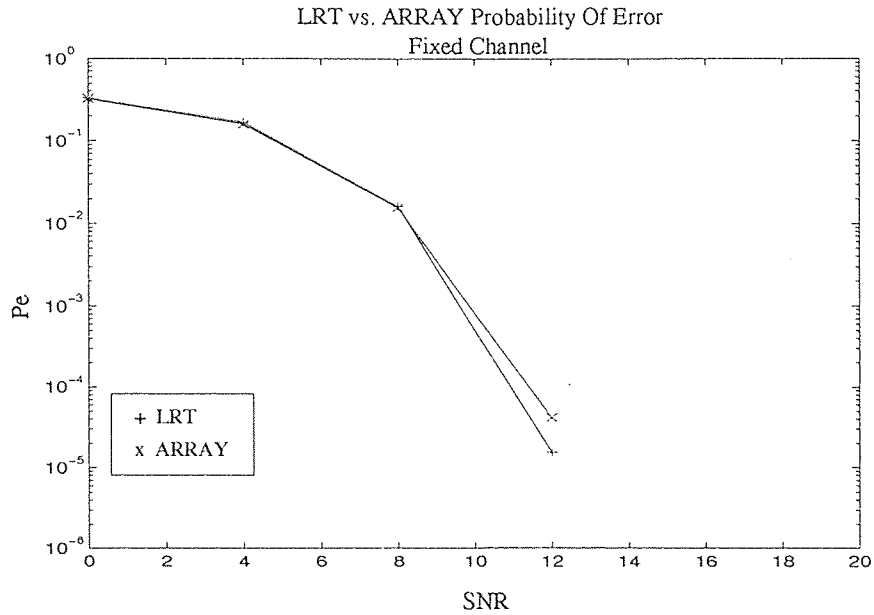

Figure 5.7 LRT/ARRAY ST Interference With Fixed Channel

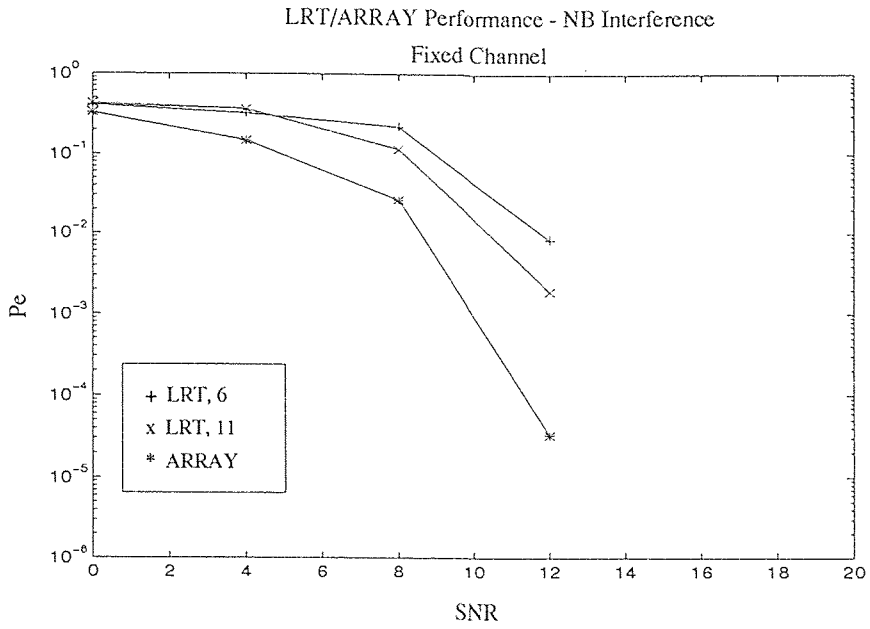

Figure 5.8 LRT/ARRAY NB Interference With Fixed Channel

CHAPTER 6

CONCLUSION

It has been shown [6] that the LRT receiver is a suboptimal solution, however, assuming proper estimates we should achieve the results shown. The ARRAY receiver and LRT receiver are basically equivalent for a single-tone interference averaged over all possible frequencies. The first two plots show the ARRAY achieving better performance only because we fixed the single-tone frequency to 1 $rads/sec$. From the simulations we see that for the narrow-band case, the array receiver's performance remained unchanged. The LRT, on the other hand, did not perform as well as it did for the single-tone case. Even with the whitening filter expanded to 11 taps the LRT did not perform as well as the ARRAY. This is due to basically two effects. One, the whitening filter cannot totally null out the narrow-band interference because of the use of a finite number of taps, and two, as the RAKE portion of the receiver expands we begin introducing more cross-correlations which result in performance degradation.

We believe that our modified LRT receiver using two antennae shows acceptable performance and could be utilized in current or future wireless spread spectrum systems. Increasing the array to more than two antennae doesn't gain much in performance. However, for multiple interferences (ie: at different locations) we'll need to expand to array appropriately. Increasing the whitening filter taps of the original LRT receiver improves it's performance for the narrow-band case but we also gain complexity which is undesirable. Even for the single-tone case the ARRAY receiver is less complex, that is, in the RLS algorithm the correlation matrix is 5x5. For the narrow-band case this is also true but for the LRT the correlation matrix

becomes 14x14. Clearly the algorithm for that case will take much longer to execute then the 5x5 case thus giving the ARRAY receiver another advantage.

Future work would include the use of mixed temporal and spatial processing on each antenna element. This would possibly combat multiple interferers in both frequency and space. Another issue is that the RAKE coefficients are not optimal in the sense that they do not consider the cross-correlations. So, another algorithm could be designed such that these coefficients do take care of the cross-correlations thus providing an increase in performance.

APPENDIX A

SNR CALCULATIONS

The signal-to-noise ratio (SNR) calculations for the LRT and ARRAY receivers will be shown to be

$$\gamma = \frac{A^2 \beta^H R \beta}{\sigma_e^2} \tag{A.1}$$

and for the case of the combination of two LRT receivers the SNR will be shown to double.

A.1 SNR Calculation for the LRT Receiver

We'll begin by defining the output of the whitening filter as

$$x(t) = \sum_{m=0}^{N_\alpha - 1} \alpha_m r\left(t - m\frac{T_c}{2}\right) \tag{A.2}$$

where $r(t)$ is defined in Equation(2.6). This can be divided into two parts, the signal part $v(t)$ and the interference part $e(t)$. Thus,

$$x(t) = v(t) + e(t) \tag{A.3}$$

where

$$v(t) = \sum_{k=0}^{N_\beta - 1} \beta_k s\left(t - k\frac{T_c}{2}\right) \tag{A.4}$$

and

$$e(t) = \sum_{m=0}^{N_\alpha - 1} \alpha_m \left[j\left(t - m\frac{T_c}{2}\right) + n\left(t - m\frac{T_c}{2}\right)\right]. \tag{A.5}$$

Equation (A.4) can be written that way because of the fact that the $\{\beta\}$ coefficients are equal to the convolution of the channel with the whitening filter [6]. The output

of the RAKE becomes

$$y(t) = \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{k=0}^{N_\beta-1} \beta_k s\left(t - k\frac{T_c}{2} - n\frac{T_c}{2}\right) + \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} e\left(t - n\frac{T_c}{2}\right). \quad \text{(A.6)}$$

The next step is to calculate the decision variable which is Equation (5.2) and is shown here again

$$U_i = Re\left\{\int_{T_b} y(t)c\left(t - iT_b - (N_\beta - 1)\frac{T_c}{2}\right)dt\right\}. \quad \text{(A.7)}$$

Now let $U_i = U_v + U_e$ where

$$\begin{aligned} U_v &= Ad_i \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{k=0}^{N_\beta-1} \beta_k \int_{T_b} c\left(t - iT_b - k\frac{T_c}{2} - nk\frac{T_c}{2}\right) \\ &\quad \cdot c\left(t - iT_b - (N_\beta - 1)\frac{T_c}{2}\right)dt \end{aligned} \quad \text{(A.8)}$$

and

$$U_e = \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \int_{T_b} e\left(t - n\frac{T_c}{2}\right) c\left(t - iT_b - (N_\beta - 1)\frac{T_c}{2}\right)dt. \quad \text{(A.9)}$$

To simplify these expressions we'll define the correlation function of $c(t)$ as

$$r_c(k) = \int_{T_b} c(t)\, c\left(t - k\frac{T_c}{2}\right)dt. \quad \text{(A.10)}$$

Thus, Equation (A.8) becomes

$$U_v = Ad_i \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{k=0}^{N_\beta-1} \beta_k r_c\left(N_\beta - 1 - k - n\right). \quad \text{(A.11)}$$

We may now define the signal-to-noise ratio as

$$\gamma = \frac{E\left[U_i\right]^2}{VAR\left[U_i\right]} \quad \text{(A.12)}$$

where

$$\begin{aligned} E[U_i] &= E[U_v] + E[U_e] \\ E[U_v] &= U_v \\ E[U_e] &= 0. \end{aligned} \tag{A.13}$$

The above is based on the assumptions that the received signal is independent of the noise and the white process $e(t)$ is assumed to be zero-mean Gaussian noise. Now we define the variance to be

$$\begin{aligned} VAR[U_i] &= E\left[U_i^2\right] - E[U_i]^2 \\ &= U_v^2 + E\left[U_e^2\right] + 2E[U_v U_e] - U_v^2 \\ &= E\left[U_e^2\right] \end{aligned} \tag{A.14}$$

where all that is needed is $E[U_e^2]$. Therefore

$$\begin{aligned} E\left[U_e^2\right] = & \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{k=0}^{N_\beta-1} \beta_{N_\beta-k-1} \int\int_{T_b} E\left[e\left(t - n\frac{T_c}{2}\right) e^*\left(\tau - k\frac{T_c}{2}\right)\right] \\ & \cdot c\left(t - iT_b - (N_\beta - 1)\frac{T_c}{2}\right) \\ & \cdot c\left(\tau - iT_b - (N_\beta - 1)\frac{T_c}{2}\right) d\tau dt \end{aligned} \tag{A.15}$$

and

$$E\left[e\left(t - n\frac{T_c}{2}\right) e^*\left(\tau - k\frac{T_c}{2}\right)\right] = \sigma_e^2 \delta\left(t - \tau - n\frac{T_c}{2} + k\frac{T_c}{2}\right). \tag{A.16}$$

After carrying out the integration we get

$$VAR[U_i] = \sigma_e^2 \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{k=0}^{N_\beta-1} \beta_{N_\beta-k-1} r_c(n-k). \tag{A.17}$$

Since the correlation function $r_c(k)$ is real, $r_c(-k) = r_c(k)$, we can re-arrange the index on the $\{\beta\}$ coefficients. Thus we have

$$\begin{aligned} VAR[U_i] &= \sigma_e^2 \sum_{n=0}^{N_\beta-1} \beta_n^* \sum_{k=0}^{N_\beta-1} \beta_k r_c(n-k) \\ &= \sigma_e^2 \beta^H \boldsymbol{R} \beta. \end{aligned} \tag{A.18}$$

Similarly we can re-arrange U_v so that

$$\begin{aligned} U_v &= A d_i \sum_{l=0}^{N_\beta-1} \beta_l^* \sum_{k=0}^{N_\beta-1} \beta_k r_c(l-k) \\ &= A d_i \beta^H \boldsymbol{R} \beta. \end{aligned} \tag{A.19}$$

And so the SNR becomes

$$\begin{aligned} \gamma &= \frac{A^2 d_i^2 \left(\beta^H \boldsymbol{R} \beta\right)^2}{\sigma_e^2 \beta^H \boldsymbol{R} \beta} \\ &= \frac{A^2 \beta^H \boldsymbol{R} \beta}{\sigma_e^2}. \end{aligned} \tag{A.20}$$

A.2 SNR Calculation for the ARRAY Receiver

Lets begin by calculating the output of the array as

$$x(t) = \sum_{l=0}^{N_a-1} \omega_l r_l(t) \tag{A.21}$$

where $r_l(t)$ is defined in Equation (2.8). We can break this up into two parts as

$$x(t) = v(t) + e(t) \tag{A.22}$$

where

$$v(t) = \sum_{l=0}^{N_a-1} \omega_l \sum_{m=0}^{N_c-1} b_{l,m} s\left(t - m\frac{T_c}{2}\right) \tag{A.23}$$

and

$$e(t) = \sum_{l=0}^{N_a-1} \omega_l \left(j_l(t) + n_l(t) \right). \tag{A.24}$$

The output of the RAKE becomes

$$y(t) = \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} x\left(t - n\frac{T_c}{2}\right) \tag{A.25}$$

$$= \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{l=0}^{N_a-1} \omega_l \sum_{m=0}^{N_c-1} b_{l,m} s\left(t - m\frac{T_c}{2} - n\frac{T_c}{2}\right) + \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} e\left(t - n\frac{T_c}{2}\right). \tag{A.26}$$

In the LRT derivation we used the fact that the $\{\beta\}$ coefficients were the convolution of the whitening filter and the RAKE. For the array we do not have convolution but we have

$$\beta_m = \sum_{k=0}^{N_a-1} \omega_k b_{k,m}. \tag{A.27}$$

Using this fact we may express $y(t)$ as

$$y(t) = \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} \sum_{k=0}^{N_\beta-1} \beta_k s\left(t - k\frac{T_c}{2} - n\frac{T_c}{2}\right) + \sum_{n=0}^{N_\beta-1} \beta^*_{N_\beta-n-1} e\left(t - n\frac{T_c}{2}\right). \tag{A.28}$$

We can see that Equation (A.28) is identical to Equation (A.6) and therefore the rest of this derivation is identical to Equations (A.7 - A.20).

A.3 Combination of Two LRT Receivers

From Figure (4.2) we see that the decision variable U will be the sum of each decision variable. For two receivers we have $U = U_1 + U_2$ so let

$$U_1 = U_{1s} + U_{1n} \tag{A.29}$$

and

$$U_2 = U_{2s} + U_{2n}. \tag{A.30}$$

Taking the expected value of U we get

$$\begin{aligned} E[U] &= E[U_1 + U_2] \\ &= E[U_1] + E[U_2]. \end{aligned} \tag{A.31}$$

Using Equation (A.13) we get

$$\begin{aligned} E[U_1] &= U_{1s} \\ E[U_2] &= U_{2s}. \end{aligned} \tag{A.32}$$

Thus

$$E[U]^2 = U_{1s}^2 + U_{2s}^2 + 2U_{1s}U_{2s}. \tag{A.33}$$

We now calculate the variance of U as

$$\begin{aligned} VAR[U] &= E[U^2] - E[U]^2 \\ &= E[(U_{1s} + U_{1n} + U_{2s} + U_{2n})^2] - E[U]^2. \end{aligned} \tag{A.34}$$

Based on the assumption that the noise components are zero-mean and statistically independent and that the noise components are independent of the signal

components, the variance becomes

$$\begin{aligned} VAR[U] &= E[U]^2 + E[U_{1n}^2] + E[U_{2n}^2] - E[U]^2 \\ &= \sigma_{1e}^2 + \sigma_{2e}^2. \end{aligned} \tag{A.35}$$

The signal-to-noise ratio (SNR) can be expressed as

$$\gamma = \frac{U_{1s}^2 + U_{2s}^2 + 2U_{1s}U_{2s}}{\sigma_{1e}^2 + \sigma_{2e}^2}. \tag{A.36}$$

To calculate an upper bound on the SNR we can assume that the noise statistics are identical and that the receiver coefficients are identical. Thus letting $U_{1s} = U_{2s} \rightarrow U_s$ and $\sigma_{1e}^2 = \sigma_{2e}^2 \rightarrow \sigma_e^2$ and using Equations (A.18 - A.19) we can express this (SNR) as

$$\begin{aligned} \gamma_o &= \frac{4A^2 d_i^2 \left(\boldsymbol{\beta}^H \boldsymbol{R} \boldsymbol{\beta}\right)^2}{2\sigma_e^2 \boldsymbol{\beta}^H \boldsymbol{R} \boldsymbol{\beta}} \\ &= \frac{2A^2 \boldsymbol{\beta}^H \boldsymbol{R} \boldsymbol{\beta}}{\sigma_e^2}. \end{aligned} \tag{A.37}$$

APPENDIX B

PROGRAM LISTINGS

The following files were run using MATLAB ©version 4.2.

B.1 GLRT

```
% FILE          : glrtx.m
% AUTHOR        : Jeffrey L. Cutcher
% VERSION       : 2.1
% DATE          : 12APR95
% GLRT Receiver with parallel option

clear;
NPaths = 4;                     % Channel Paths (Includes direct path)
NAlpha = 3;                     % Size of Alpha Filter (Taps)
NBeta = NPaths+NAlpha-1;        % Size of Beta Filter (Taps)
Nant = 1;                       % Number of Receivers
DataSize = 101;                 % Number of Data bits
TrainSize = 5;                  % Number of Training Bits
w = 0.95;                       % RLS weighting factor
AV = 1000;                      % Number of Averaging Iterations

load code                       % Load in PN_
EPN = 2*PN_*PN_';               % Energy of code x 2

A = [0.1270, 0.2013, 0.3190, 0.5056];
GAIN=sqrt(EPN)*A;
IGAIN = 10*GAIN;
TrainSeq = ones(1,TrainSize);
GAMMA1 = [];
GAMMA2 = [];
SNR1 =[];
SNR2 = [];
PN = [];
RD = [];
PNLength = length(PN_);
Td = 1/2;
T = PNLength / Td;
```

```
PNSize = T;
Tx_Seq_Len = DataSize * PNSize;
Rx_Seq_Len = Tx_Seq_Len + NPaths - 1; % Because of convolution
Train_Len = TrainSize * PNSize;

% Construct double samples of the PN sequence
PN = signat(PN_,PNSize,PNSize) / sqrt(EPN);

for NUM=1:AV    % LOOP For Averaging
GAMMA1 = [];
SNR1 = [];

% Pick a new seed
rand('seed',100*sum(clock));
randn('seed',100*sum(clock));

for G=1:length(GAIN)
   [NUM G]

   % Inital conditions and definitions
   TxData = zeros(1, Tx_Seq_Len);
   RxData =zeros(Nant, Rx_Seq_Len);
   ChData = zeros(Nant, Rx_Seq_Len);
   IData = zeros(Nant, Rx_Seq_Len);
   NIData = zeros(Nant, Rx_Seq_Len);
   OutPutData = zeros(1,DataSize);

   % Randomly construct a data set
   DATA = [TrainSeq, sign(randn(1,DataSize - TrainSize))];

   % Create transmitted data (Two samples per chip)
    for k=1:DataSize
       indx = (k-1)*PNSize+1:k*PNSize;
       TxData(:,indx) = GAIN(G)*diag(DATA(:,k))*PN;
    end

   % Pass data through channel (Rayleigh channel)
   C = channel(Nant, NPaths);
   for k=1:Nant
      ChData(k,:) = conv(C(k,:),TxData);
   end
   VarC = diag(cov(ChData.'));

   % Generate White Gaussian noise
```

```
NAmp = 1.0 / sqrt(2);
Noise = NAmp*(randn(Nant,Rx_Seq_Len)+i*randn(Nant,Rx_Seq_Len));
VarN = cov(Noise(1,:)); % Use Antenna #0

% Generate Sine Wave interfernce;
for k=1:Nant
   theta = 2*pi*rand(1,5);

   % Single-Tone Model
   IData(k,:) = IGAIN(G)*exp(i*(1.0*(1:Rx_Seq_Len) + theta(1)));

   % Narrowband Model
  % I1 = exp(i*(0.3750*(1:Rx_Seq_Len) + theta(1)));
  % I2 = exp(i*(0.6875*(1:Rx_Seq_Len) + theta(2)));
  % I3 = exp(i*(1.0000*(1:Rx_Seq_Len) + theta(3)));
  % I4 = exp(i*(1.3125*(1:Rx_Seq_Len) + theta(4)));
  % I5 = exp(i*(1.6250*(1:Rx_Seq_Len) + theta(5)));
  % IData(k,:) = IGAIN(G)*(I1 + I2 + I3 + I4 + I5);
  % clear I1 I2 I3 I4 I5;
end

% Add Interferer and Noise
RxData = ChData + IData + Noise;
NIData = IData + Noise;

% Start receiving
% Define Data Vector for Alpha-Filter,
%   RXA = [r(k), r(k-1), ... , r(k-NAlpha+1)]
RXA = zeros(Nant, NAlpha);
RXANI = zeros(Nant, NAlpha);
RXAS = zeros(Nant, NAlpha);

% Define Data Vector for Beta-Filter,
%   RXB = [r'(k), r'(k-1), ... , r'(k-NBeta+1)]
RXB = zeros(Nant, NBeta);
RXBNI = zeros(Nant, NBeta);
RXBS = zeros(Nant, NBeta);

% Define Vector for Alpha,
%   Alpha = [1, A(1), ... , A(k-NAlpha+1)]
Alpha = [ones(Nant,1) zeros(Nant, NAlpha-1)];

% Define Vector for Beta,
%   Beta = [b0, b1, ... , b(k-NBeta+1)
```

```
Beta = zeros(Nant, NBeta);

% Define Vector for Estimated Signal
EstData = zeros(Nant, NBeta);

% Initialize Algorithm
PKInit = 0.0001 * cov(RxData(1,:).'); % Use Antenna #0
Pk_1 = [];
PO = NAlpha + NBeta - 1;
for k=0:Nant-1
   i1 = k*PO + 1;
   i2 = PO*(k+1);
   Pk_1(:,i1:i2) = (1 / PKInit) * eye(NAlpha + NBeta - 1);
end
Whk_1 = zeros(Nant,(NAlpha+NBeta-1));
Sum = zeros(Nant,1);
SumNI = zeros(Nant,1);
SumS = zeros(Nant,1);
error = zeros(Nant,1);
Kalman = zeros(PO, Nant);
E_PNCount = 1;
R_PNCount = 1;
EstBIT = 1;
U = zeros(Nant,DataSize);
Us = zeros(Nant,DataSize);
Un = zeros(Nant,DataSize);

for SampleCount=1:Rx_Seq_Len
   % Shift Data through Alpha Taps
   for k=1:Nant
      RXA(k,2:NAlpha) = RXA(k,1:NAlpha-1);
      RXA(k,1) = RxData(k,SampleCount);

      RXANI(k,2:NAlpha) = RXANI(k,1:NAlpha-1);
      RXANI(k,1) = NIData(k,SampleCount);

      RXAS(k,2:NAlpha) = RXAS(k,1:NAlpha-1);
      RXAS(k,1) = ChData(k,SampleCount);
   end

   % Calculate Estimated TxData
   EstData(:,2:NBeta) = EstData(:,1:NBeta-1);

   if (SampleCount < (Train_Len + NBeta))
```

```
   % Training
   EstData(:,1) = TxData(SampleCount) * ones(Nant,1);
   Xk = [RXA(:,2:NAlpha) EstData].';
   for k=1:Nant
      error(k) = RXA(k,1) - Whk_1(k,:)*Xk(:,k);
   end
else
   code = PN(E_PNCount);
   EstData(:,1) = EstBIT*GAIN(G)*code*ones(Nant,1);
   RD = [];
   for k=1:NAlpha-1
      RD = [RD, RxData(:,SampleCount-PNSize-k)];
   end
   Xk = [RD, EstData].';
   for k=1:Nant
      error(k) = RxData(k,SampleCount-PNSize) - ...
                 Whk_1(k,:)*Xk(:,k);
   end
end

% Recursive Algorithm
for k=0:Nant-1
   i1 = k*PO + 1;
   i2 = PO*(k+1);
   Kalman(:,k+1) = (Pk_1(:,i1:i2) * Xk(:,k+1)) / ...
                   (w + Xk(:,k+1)'*Pk_1(:,i1:i2)*Xk(:,k+1));
end

% Calculate new coefficients
for k=0:Nant-1
   i1 = k*PO + 1;
   i2 = PO*(k+1);
   Wk = Whk_1(k+1,:)' + (Kalman(:,k+1) * conj(error(k+1)));
   Whk_1(k+1,:) = Wk';
   Pk = (Pk_1(:,i1:i2) - Kalman(:,k+1)*Xk(:,k+1)' ...
        *Pk_1(:,i1:i2)) / w;
   Pk_1(:,i1:i2) = Pk;
end

% Update coefficients in filter
Alpha(:,2:NAlpha) = -Whk_1(:,1:NAlpha-1);
Beta = fliplr(Whk_1(:,NAlpha:NBeta+NAlpha-1));

% Calculate Output of Alpha Filter
```

```
AlphaOut = diag(RXA * Alpha.');
AlphaOutNI = diag(RXANI * Alpha.');
AlphaOutS = diag(RXAS * Alpha.');

% Shift Data through Beta Taps
for k=1:Nant
   RXB(k,2:NBeta) = RXB(k,1:NBeta-1);
   RXB(k,1) = AlphaOut(k);

   RXBNI(k,2:NBeta) = RXBNI(k,1:NBeta-1);
   RXBNI(k,1) = AlphaOutNI(k);

   RXBS(k,2:NBeta) = RXBS(k,1:NBeta-1);
   RXBS(k,1) = AlphaOutS(k);
end

code = PN(R_PNCount);

% Calculate Output of RAKE
BetaOut = code*diag(RXB * Beta');
BetaOutNI = code*diag(RXBNI * Beta');
BetaOutS = code*diag(RXBS * Beta');

% Sufficient Statistic Summation
if (SampleCount > (NBeta - 1))
   Sum = Sum + BetaOut;
   SumNI = SumNI + BetaOutNI;
   SumS = SumS + BetaOutS;
end

E_PNCount = E_PNCount + 1;
if (E_PNCount > PNSize)
   E_PNCount = 1;
end
if (SampleCount > (NBeta - 1))
   R_PNCount = R_PNCount + 1;
   if (R_PNCount > PNSize)
      R_PNCount = 1;
      % Now we make decision
      if (real(sum(Sum)) > 0)
         EstBIT = 1;
      else
         EstBIT = -1;
      end
```

```
            BitNum = (SampleCount - NPaths + 1) / PNSize;

            % Store Us and Un
            U(:,BitNum) = Sum;
            Us(:,BitNum) = SumS;
            Un(:,BitNum) = SumNI;

            Sum = zeros(Nant,1);
            SumNI = zeros(Nant,1);
            SumS = zeros(Nant,1);
         end
      end
   end % SampleCount Loop

   % Calculate GAMMA and SNR
   if (Nant > 1)
      SigmaS = cov(sum(real(Us(:,TrainSize:DataSize-1))));
      SigmaN = cov(sum(Un(:,TrainSize:DataSize-1)));
   else
      SigmaS = cov(real(Us(TrainSize:DataSize-1)));
      SigmaN = cov(Un(TrainSize:DataSize-1));
   end
   GAMMA1(G) =  SigmaS / SigmaN;
   SNR1(G) = mean(VarC) / VarN;
end % for GAIN

GAMMA2(NUM,:) = GAMMA1;
SNR2(NUM,:) = SNR1;

end % for NUM

% Calculate Average Pe and SNR
AveGAMMA = mean(GAMMA2);
AveSNR = mean(SNR2);
APe = 0.5*erfc(sqrt(AveGAMMA));

% Plot Pe
figure(1)
semilogy(20*log10(GAIN), APe);
title('Pe for GLRT');
xlabel('SNR [dB]');
ylabel('Pe');
grid on;
```

B.2 ARRAY

```
% FILE          : glrtx.m
% AUTHOR        : Jeffrey L. Cutcher
% VERSION       : 3.0
% DATE          : 12APR95
%

clear;
NPaths = 4;                     % Channel Paths (Includes direct path)
NAlpha = Nant;                  % Size of Alpha Filter (Taps)
NBeta = NPaths;                 % Size of Beta Filter
Nant = 1;                       % Antenna Array Size
DataSize = 101;                 % Number of Data bits
TrainSize = 5;                  % Number of Training Bits
w = 0.95;                       % RLS weighting factor
AV = 1000;                      % Number of Averaging Iterations

load code                       % Load in PN_
EPN = 2*PN_*PN_';               % Energy of code x 2

A = [0.1270, 0.2013, 0.3190, 0.5056];
GAIN=sqrt(EPN)*A;
IGAIN = 10*GAIN;
TrainSeq = ones(1,TrainSize);
GAMMA1 = [];
GAMMA2 = [];
SNR1 =[];
SNR2 = [];
PN = [];
RD = [];
PNLength = length(PN_);
Td = 1/2;
T = PNLength / Td;
PNSize = T;
Tx_Seq_Len = DataSize * PNSize;
Rx_Seq_Len = Tx_Seq_Len + NPaths - 1; % Because of convolution
Train_Len = TrainSize * PNSize;

% Construct double samples of the PN sequence
PN = signat(PN_,PNSize,PNSize) / sqrt(EPN);

for NUM=1:AV    % LOOP For Averaging
GAMMA1 = [];
```

```
SNR1 = [];

% Pick a new seed
rand('seed',100*sum(clock));
randn('seed',100*sum(clock));

for G=1:length(GAIN)
   [NUM G]
   % Inital conditions and definitions
   TxData = zeros(1, Tx_Seq_Len);
   RxData =zeros(Nant, Rx_Seq_Len);
   ChData = zeros(Nant, Rx_Seq_Len);
   IData = zeros(Nant, Rx_Seq_Len);
   NIData = zeros(Nant, Rx_Seq_Len);
   OutPutData = zeros(1,DataSize);

   % Randomly construct a data set
   DATA = [TrainSeq, sign(randn(1,DataSize - TrainSize))];

   % Create transmitted data (Two samples per chip)
    for k=1:DataSize
       indx = (k-1)*PNSize+1:k*PNSize;
       TxData(:,indx) = GAIN(G)*diag(DATA(:,k))*PN;
    end

   % Pass data through channel (Rayleigh channel)
   C = channel(Nant, NPaths);
   for k=1:Nant
      ChData(k,:) = conv(C(k,:),TxData);
   end
   VarC = diag(cov(ChData.'));

   % Generate White Gaussian noise
   NAmp = 1.0 / sqrt(2);
   Noise = NAmp*(randn(Nant,Rx_Seq_Len)+i*randn(Nant,Rx_Seq_Len));
   VarN = cov(Noise(1,:)); % Use Antenna #0

   % Generate Sine Wave interfernce;
   theta = 2*pi*rand(1,5);              % phase
   thetaJ = 2*pi*rand(1);               % Angle of Arrival of Jammer
   d = 10;                              % distance between elements
   lambda = 1;                          % normalized to one
   phiJ = 2*pi*(d/lambda)*sin(thetaJ); % electrical angle
   SJ = exp(i*(0:Nant-1)*phiJ);
```

```
for k=1:Nant
   % Single-Tone Model
   IData(k,:) = IGAIN(G)*exp(i*(0.3*(1:Rx_Seq_Len) + ...
                theta(1)))*SJ(k);
   % Narrowband Model
  % I1 = exp(i*(0.3750*(1:Rx_Seq_Len) + theta(1)));
  % I2 = exp(i*(0.6875*(1:Rx_Seq_Len) + theta(2)));
  % I3 = exp(i*(1.0000*(1:Rx_Seq_Len) + theta(3)));
  % I4 = exp(i*(1.3125*(1:Rx_Seq_Len) + theta(4)));
  % I5 = exp(i*(1.6250*(1:Rx_Seq_Len) + theta(5)));
  % IData(k,:) = IGAIN(G)*(I1 + I2 + I3 + I4 + I5)*SJ(k);
  % clear I1 I2 I3 I4 I5;
end

% Add Interferer and Noise
RxData = ChData + IData + Noise;
NIData = IData + Noise;

% Start receiving
% Define Data Vector for Alpha-Filter,
%   RXA = [r(k), r(k-1), ... , r(k-NAlpha+1)]
RXA = zeros(Nant,1);
RXANI = zeros(Nant,1);
RXAS = zeros(Nant,1);

% Define Data Vector for Beta-Filter,
%   RXB = [r'(k), r'(k-1), ... , r'(k-NBeta+1)]
RXB = zeros(1, NBeta);
RXBNI = zeros(1, NBeta);
RXBS = zeros(1, NBeta);

% Define Vector for ALpha,
%   Alpha = [1, omega(1), ... , omega(k-Nant+1)]
Alpha = [1 zeros(1, Nant-1)];

% Define Vector for Beta,
%   Beta = [b0, b1, ... , b(k-NBeta+1)
Beta = zeros(1, NBeta);

% Define Vector for Estimated Signal
EstData = zeros(1, NBeta);

% Initialize Algorithm
PKInit = 0.0001 * cov(RxData(1,:).'); % Use Antenna #1
```

```
PO = Nant + NBeta - 1;
Pk_1 = (1 / PKInit) * eye(PO);
Whk_1 = zeros(1,PO);
Sum = 0;
SumNI = 0;
SumS = 0;
error = 0;
Kalman = zeros(PO,1);
E_PNCount = 1;
R_PNCount = 1;
EstBIT = 1;
U = zeros(1,DataSize);
Us = zeros(1,DataSize);
Un = zeros(1,DataSize);

for SampleCount=1:Rx_Seq_Len
   % Bring in next SPACE sample
   RXA = RxData(:,SampleCount);

   RXANI = NIData(:,SampleCount);

   RXAS = ChData(:,SampleCount);

   % Calculate Estimated TxData
   EstData(2:NBeta) = EstData(1:NBeta-1);

   if (SampleCount < (Train_Len + NBeta))
      % Training
      EstData(1) = TxData(SampleCount);
      Xk = [RXA(2:Nant).' EstData].';
      error = RXA(1) - Whk_1*Xk;
   else
      code = PN(E_PNCount);
      EstData(1) = EstBIT*GAIN(G)*code;
      Xk = [RxData(2:Nant,SampleCount-PNSize).' EstData].';
      error = RxData(1,SampleCount-PNSize) - Whk_1*Xk;
   end

   % Recursive Algorithm
   Kalman = (Pk_1 * Xk) / (w + Xk'*Pk_1*Xk);

   % Calculate new coefficients
   Wk = Whk_1' + (Kalman * conj(error));
   Whk_1 = Wk';
```

```
Pk = (Pk_1 - Kalman*Xk'*Pk_1) / w;
Pk_1 = Pk;

% Update coefficients in filter
Alpha(2:Nant) = -Whk_1(1:Nant-1);
Beta = fliplr(Whk_1(Nant:PO));

% Calculate Output of Array
AlphaOut = Alpha * RXA;
AlphaOutNI = Alpha * RXANI;
AlphaOutS = Alpha * RXAS;

% Shift Data through Beta Taps
RXB(2:NBeta) = RXB(1:NBeta-1);
RXB(1) = AlphaOut;

RXBNI(2:NBeta) = RXBNI(1:NBeta-1)
RXBNI(1) = AlphaOutNI;

RXBS(2:NBeta) = RXBS(1:NBeta-1);
RXBS(1) = AlphaOutS;

code = PN(R_PNCount);

% Calculate Output of RAKE
BetaOut = code * RXB * Beta';
BetaOutNI = code * RXBNI * Beta';
BetaOutS = code * RXBS * Beta';

% Sufficient Statistic Summation
if (SampleCount > (NBeta - 1))
   Sum = Sum + BetaOut;
   SumNI = SumNI + BetaOutNI;
   SumS = SumS + BetaOutS;
end

E_PNCount = E_PNCount + 1;
if (E_PNCount > PNSize)
   E_PNCount = 1;
end
if (SampleCount > (NBeta - 1))
   R_PNCount = R_PNCount + 1;
   if (R_PNCount > PNSize)
      R_PNCount = 1;
```

```
            % Now we make decision
            if (real(Sum) > 0)
               EstBIT = 1;
            else
               EstBIT = -1;
            end
            BitNum = (SampleCount - NPaths + 1) / PNSize;

            % Store U, Us and Un
            U(BitNum) = Sum;
            Us(BitNum) = SumS;
            Un(BitNum) = SumNI;

            Sum = 0;
            SumNI = 0;
            SumS = 0;
         end
      end
   end % SampleCount Loop

   % Calculate GAMMA and SNR
   SigmaS = cov(real(Us(TrainSize:DataSize-1)));
   SigmaN = cov(Un(TrainSize:DataSize-1));
   GAMMA1(G) =  SigmaS / SigmaN;
   SNR1(G) = mean(VarC) / VarN;
end % for GAIN

GAMMA2(NUM,:) = GAMMA1;
SNR2(NUM,:) = SNR1;

end % for NUM

% Calculate Average Pe and SNR
AveGAMMA = mean(GAMMA2);
AveSNR = mean(SNR2);
APe = 0.5*erfc(sqrt(AveGAMMA));

% Plot Pe
figure(1)
semilogy(20*log10(GAIN), APe);
title('Pe for GLRT');
xlabel('SNR [dB]');
ylabel('Pe');
grid on;
```

B.3 MISCELLANEOUS

```
% channel : Jan 23, 1995
% Modeling frequncy selective Fading Channel by tap delay line.

% C Channel coeficients
% N  number of antenna
% M number of resolvable multipath

function C = channel(N, M )
C = (randn(N,M) + j * randn(N,M)) / 3;
% end channel.m
-----------------------------------------------------------------
% signat: Jan 20, 1995

% Calculate the signature waveform of every mobile

% code   spreading code
% D   spreading gain, also equal to code length
% num   total number of mobiles = number of interferences + 1
% l   at time l
% L       length of sequence
% u    signature waveform

function u = signat(code, L, T)
[num, D]= size(code);
Td=D/T;

c  = reshape(code.', 1, num*D);
c1 = ones(1/Td, 1)*c;
c2 = reshape(c1, 1, num*T);
c3 = reshape(c2, T, num).';

for l=1:ceil(L/T)
   u1(:,(l-1)*T+1: l*T) = c3;
end

u=u1(:,1:L);
% end signat.m
-----------------------------------------------------------------
% Gold Code
PN_ = [+1 +1 +1 -1 +1 -1 +1 +1 -1 -1 ...
       -1 +1 -1 +1 -1 -1 +1 +1 -1 -1 ...
       +1 +1 -1 +1 -1 +1 -1 -1 +1 -1 +1];
```

REFERENCES

1. A. A. Giordano and F. M. Hsu, *Least Square Estimation with Applications to Digital Signal Processing*, John Wiley and Sons, New York, 1985.

2. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, New Jersey, 2nd ed., 1991.

3. S. Haykin, *Communication Systems*, John Wiley and Sons, New York, 3rd ed., 1994.

4. L. Hoff, "Signaling techniques for underwater acoustic communications," *Tech. Rep. NOSC TR-649*, Oct. 1980.

5. F. M. Hsu and A. A. Giordano, "Digital whitening techniques for improving spread spectrum communications performance in the presence of narrow-band jamming and interference," *IEEE Transactions on Communications*, vol. COM-26, no. 2, pp. 209–216, February 1978.

6. R. A. Iltis, "A glrt-based spread-spectrum receiver for joint channel and interference suppression," *IEEE Transactions on Communications*, vol. 37, no. 3, pp. 277–288, March 1989.

7. J. W. Ketchum and J. G. Proakis, "Adaptive algorithms for estimating and suppressing narrow-band interference in pn spread spectrum," *IEEE Transactions on Communications*, vol. COM-30, no. 5, pp. 913–924, May 1982.

8. L.-M. Li and L. B. Milstein, "Rejection of narrow-band interference in pn spread spectrum systems using transversal filters," *IEEE Transactions on Communications*, vol. COM-30, no. 5, pp. 925–928, May 1982.

9. L. B. Milstein and R. A. Iltis, "Signal processing for interference rejection in spread spectrum communications," *IEEE ASSP Magazine*, pp. 18–31, April 1986.

10. L. B. Miltstein, "Interference rejection techniques in spread spectrum communications," *PROCEEDINGS OF THE IEEE*, vol. 76, no. 6, pp. 657–671, June 1988.

11. R. Price and P. Green, "A communications technique for multi-path channels," *Proc. IRE*, vol. 46, pp. 555–570, Mar. 1958.

12. J. Proakis, *Digital Communications*, McGraw-Hill, New York, 2nd ed., 1989.

13. J. Proakis and M. Salehi, *Communication Systems Engineering*, Prentice Hall, New Jersey, 1994.

14. L. A. Rusch and H. V. Poor, "Improved algorithms for narrowband interference suppression in direct sequence spread spectrum," *Proceedings of the Third International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 623–628, Oct. 1992.

15. H. V. Trees, *Detection, Estimation and Modulation Theory: Part I*, John Wiley and Sons, New York, 1968.

16. G. Turin, "Introduction to spread-spectrum antimulti-path techniques and their application to urban radio," *Proc. IEEE*, vol. 68, no. 3, pp. 328–352, Mar. 1980.