(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MIMO RECURSIVE LEAST SQUARES CONTROL ALGORITHM FOR THE AN/FPN-44A LORAN-C TRANSMITTER

by

John D. Wood

September 1993

Thesis Advisor:	Murali Tummala
Second Reader:	Roberto Cristi

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) Dept. of Electrical and Computer Eng.	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 93943-5121		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5121
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 NO.

11. TITLE *(Include Security Classification)*
MIMO RECURSIVE LEAST SQUARES CONTROL ALGORITHM FOR THE AN/FPN-44A LORAN-C TRANSMITTER

12. PERSONAL AUTHOR(S)
Wood, John, D.

13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM TO	14. DATE OF REPORT *(Year, Month, Day)* September 1993	15. PAGE COUNT 108

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 United States Government.

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Multichannel Time Series Analysis, Least Squares Optimization, Recursive Least Squares Modeling and Control

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

A multiple-input, multiple-output (MIMO) recursive least squares (RLS) algorithm is developed to shape and control the Loran-C RF pulse of the AN/FPN-44A tube type transmitter. The control algorithm is incorporated into a transmitter simulation program, where it seeks to produce an optimal transmitter drive waveform (TDW). An optimal TDW produces a near ideal RF pulse.

The control algorithm uses a MIMO reference model of the transmitter; parameters of the model are obtained using recursive least squares multichannel time series techniques. The MIMO reference model has the ability to adapt to the non-LTI characteristics of the simulated transmitter.

The MIMO RLS control algorithm is implemented in both an ideal and a realistic noisy environment. In the ideal environment, when representing the RF pulse with parameters of its half-cycle peak amplitudes and zero-crossings, the MIMO RLS controller is able to shape the RF pulse and control its zero-crossings. Quantization and system noise in the non-ideal environment results in performance deterioration of the control algorithm. The performance of the MIMO RLS algorithm is compared against another method of control, the steepest descent algorithm.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Murali Tummala	22b. TELEPHONE *(Include Area Code)* (408) 656-2645	22c. OFFICE SYMBOL EC/Tu

MIMO Recursive Least Squares Control Algorithm for the AN/FPN-44A Loran-C Transmitter

by
John D. Wood
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1988

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1993

Author: [illegible]
John D. Wood

Approved By: [illegible]
Murali Tummala, Thesis Advisor

[illegible]
Roberto Cristi, Second Reader

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

Abstract

A multiple-input, multiple-output (MIMO) recursive least squares (RLS) algorithm is developed to shape and control the Loran-C RF pulse of the AN/FPN-44A tube type transmitter. The control algorithm is incorporated into a transmitter simulation program, where it seeks to produce an optimal transmitter drive waveform (TDW). An optimal TDW produces a near ideal RF pulse.

The control algorithm uses a MIMO reference model of the transmitter; parameters of the model are obtained using recursive least squares multichannel time series techniques. The MIMO reference model has the ability to adapt to the non-LTI characteristics of the simulated transmitter.

The MIMO RLS control algorithm is implemented in both an ideal and a realistic noisy environment. In the ideal environment, when representing the RF pulse with parameters of its half-cycle peak amplitudes and zero-crossings, the MIMO RLS controller is able to shape the RF pulse and control its zero-crossings. Quantization and system noise in the non-ideal environment results in performance deterioration of the control algorithm. The performance of the MIMO RLS algorithm is compared against another method of control, the steepest descent algorithm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

In 1990 the Coast Guard Electronic Engineering Center initiated a multi-year project entitled the *Electronic Equipment Replacement Project* [Ref. 1], which outlines the need to redesign and upgrade the Loran-C equipment. The redesign of various portions of the Loran-C system is necessary for equipment support structure, the desire to enhance and expand automation, the need to respond to new system requirements, and the desire to remain in step with new technology.

Under plan one, entitled "EPA/PGEN/LORDAC Redesign," the monitor and control methods are to be redesigned to provide automatic Loran-C pulse shaping and improve the monitoring functions. The new control system generates and controls the Loran-C pulse automatically, maintains the pulse within specifications, and records results for an operational database.

In this thesis, a multiple input multiple output (MIMO) recursive least squares (RLS) algorithm is developed to shape and control a Loran-C pulse. The Loran-C pulse is successfully controlled with the AN/FPN-44A transmitter, and it meets the tolerances provided in the Coast Guard's *Specification for the Transmitted Loran-C Signal* [Ref. 2]. The Loran-C pulse is monitored at each control iteration, and data are compiled for pulse analysis.

A MATLAB computer program that simulates the AN/FPN-44A transmitter [Ref. 3] is used to test and analyze the MIMO RLS control algorithm. The control algorithm uses an adaptive MIMO reference model. This MIMO model is developed using multichannel time series techniques. The MIMO model has the ability to adapt to time variations and non-linear changes in the transmitter's operating characteristics.

Comparisons are made between the RLS and a steepest descent control algorithm. The steepest descent control algorithm was developed by Peterson [Ref. 4] and implemented by Bruckner [Ref. 3] to control the Loran-C pulse. The advantages and limitations of the RLS algorithm are addressed.

The thesis is organized as follows: Chapter II presents a summary of the Loran-C operation, pulse specifications, and a proposed control system. Chapter III contains the derivation of the RLS algorithm for a MIMO model of the transmitter. The MIMO RLS control algorithm is derived in Chapter IV, and the results and analysis are presented in Chapter V. In the results, the performance of the RLS algorithm is compared with that of an alternate method of control called the steepest descent algorithm. Finally, conclusions on the work reported and suggestions for improvement (future work) are presented in Chapter VI. Appendix A contains the results of the MIMO RLS control algorithm operating with the input and output waveforms at realistic SNR levels. A derivation of the MIMO RLS control algorithm with memory and its results are included in Appendix B. Appendix C contains the derivation of an integral control algorithm, which behaves similarly to the steepest descent method and has similar control characteristics. Appendix D contains the MatLab code of the control algorithms and other support programs.

II. SUMMARY OF LORAN C

A. LORAN C OPERATION

Loran-C radio navigation is based on time differences between a received master station's pulse and several secondary stations' pulses. A Loran receiver can obtain time differences from as many as four secondary stations, labeled as W, X, Y, and Z. A Loran receiver translates these time differences into hyperbolic lines of position. An intersection of two or more hyperbolic lines of position will formulate its location at the point of intersection.

The master and secondary stations for a specific geographical area, referred to as a chain, all transmit a series of pulse groups at a fixed rate called the Group Repetition Interval (GRI). The GRI for various chains vary from 40,000 to 99,900 microseconds. Each secondary station in the chain transmits its pulse group at the same GRI, but with different emission delays with reference to the master's pulse group (see Figure 2.1). [Ref. 2: p. 2-5]

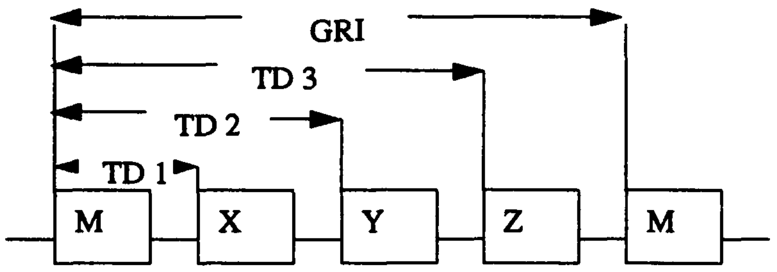

Figure 2.1: Emission delays from the master station.

The master station transmits a group of nine consecutive pulses. Each secondary station then transmits its own group of eight consecutive pulses. The pulses in each group are separated by 1000μs except the master's ninth pulse, which is transmitted 2000μs after the eighth pulse.

The Loran-C receiver has the ability to receive both the Loran ground waves and skywaves. The ground wave is the one used to calculate the time difference. Pulse group phase coding is used to distinguish the ground wave from the skywave. There are two GRI transmission sequences that comprise a phase code interval (PCI). The phases of certain pulses in the transmission sequence are changed by 180^o. The pattern of phase changes of the PCI is illustrated in Ref. 2, pp. 2-6. For this research, the control, testing, and analysis are conducted on pulse one of the eight (or nine in the case of the master station) pulses transmitted by a station in a GRI. Pulse one of the pulse sequence for all stations has a positive phase code which corresponds to a zero phase shift.

B. LORAN-C PULSE SPECIFICATION

1. Description of the Loran-C pulse

The Loran-C signal is made up of individual pulses which must meet specific tolerances for the signal to be acceptable. The carrier frequency of a Loran-C pulse is 100kHz; during the first 65µs, the pulse amplitude is specified by: [Ref. 2: p. 2.1]

$$i(t) = 0; \text{ for } t < \tau$$

$$i(t) = A(t-\tau_e)^2 \exp\left[\frac{-2(t-\tau_e)}{65}\right] \sin(0.2\pi t + \phi_c)$$

$$\text{for } \tau_e \le t \le 65 + \tau_e \qquad (2.1)$$

where

- A is the normalization constant related to the magnitude of the peak antenna current in amperes,
- t is time in microseconds,
- τ_e is the envelope to cycle difference (ECD) in microseconds, and
- ϕ_c is the phase-code parameter (in radians) which is 0 for positive phase code and π for negative phase code.

The first 90μs of the ideal Loran-C pulse is shown in Figure 2.2. The first 65μs of the standard RF pulse, as described by equation 2.1, is called the leading edge. The RF pulse trailing edge is defined as the portion of the pulse following the maximum peak amplitude or the 65 μs point, whichever occurs first. Different transmitters have different decay characteristics of the pulse's trailing edge. In this work, the AN/FPN-44A transmitter is chosen for testing the control algorithm and analysis. The spectrum must be within the bandwidth of 90 to 110kHz. The normalized pulse amplitude for $t > 500\mu s$ is less than or equal to 0.0014. The tolerance specifications for the trailing edge amplitude are established to substantially decay the pulse, so the next transmitted pulse has no interference from the previous one. [Ref. 3: p. 10]

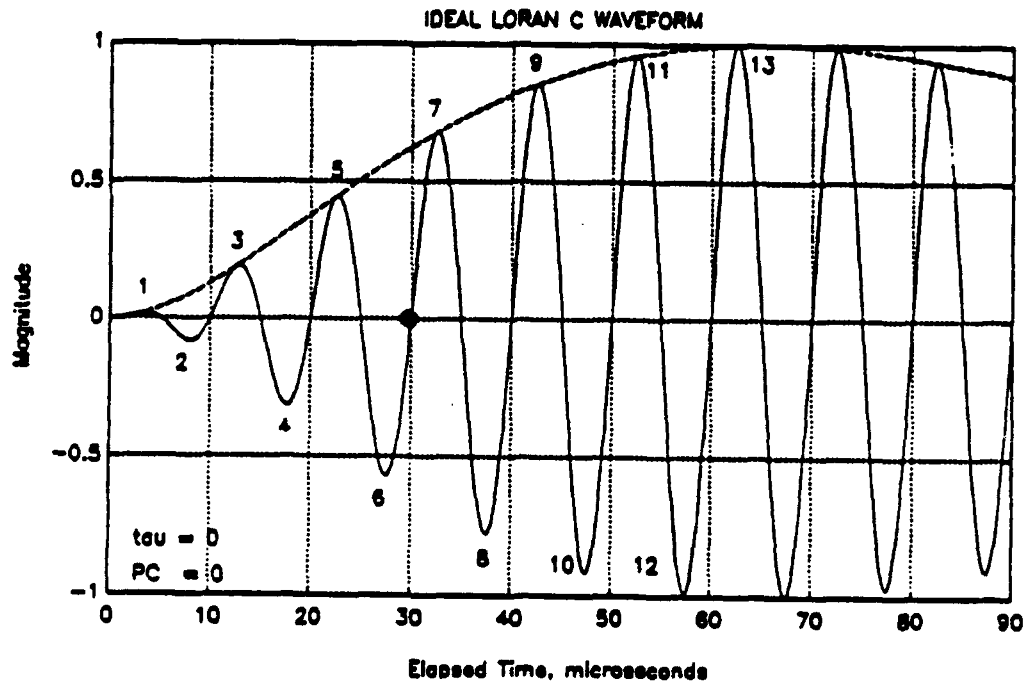

Figure 2.2: The ideal Loran-C pulse [Ref. 3, p. 10].

The third negative to positive zeros-crossing in the Loran-C pulse provides a reference point called the standard zero-crossing (SZC). A Loran receiver locks onto this location by the unique amplitude ratio of the fifth and seventh peaks. The standard zero-crossing occurs approximately 30μs after the beginning of the pulse. The Loran receiver uses the standard zero-crossing as a reference when determining time differences between pulses from the master and secondary stations. [Ref. 2: p. 2.3, Ref. 3: p. 9-11]

2. Pulse Tests

The Coast Guard has established four tests to ensure that the Loran pulse shape resembles the ideal Loran pulse, and the shape is identical from pulse to pulse. The four tests are: envelope-to-cycle difference (ECD), root mean square value of half-cycle-peak amplitudes 1-8, maximum individual error in half-cycle-peak amplitudes 1-8 and 9-13, and the zero-crossing tolerance. [Ref. 2: p. 2.1-2.3, Ref. 3: p. 11] These four tests are fully described in Ref. 2 and are summarized in the following sections.

a. Envelope-to-Cycle Difference (ECD)

The ECD is a time relationship between the position of the pulse envelope relative to the position of the zero-crossings. A positive ECD has an envelope that appears later in time by a factor of τ_e (see equation 2.1), which has the appearance of the envelope shifted to the right along the time axis. The ECD may also be negative, where the envelope is shifted to the left. The calculation of the ECD of a Loran-C pulse is a tedious process, and the details may be found in Ref. 3: p. 137.

The ECD of the Loran pulse can be adjusted at the transmitter to provide the desired RF pulse shape. Once the ECD is computed for the station's transmitted RF pulse, it may be assigned a value in the range of -2.5 to +2.5μs [Ref. 2: p. 2.2]. This is called the local or transmitted ECD value. In computing the RF pulse error, the same ECD value is used for both the actual and the ideal RF pulse. For the half-cycle peak amplitude tests, the local ECD must fall within the allowable range [Ref.2: p. 2.3]. The transmitted pulse cannot have an ECD that exceeds ±0.5μs from the transmitted ECD value [Ref. 2: p. 3.4]. When the RF peak amplitudes and the zero-crossings are within specification, the ECD is automatically within the specified tolerance [Ref. 5]. The control and testing of ECD are not addressed in this thesis. An ECD of zero is used in this thesis when generating the ideal RF pulse, for testing the control algorithm and for the Loran pulse analysis.

b. Ensemble Tolerance of Half-Cycle Peak Amplitudes

The root mean square error between the first eight half-cycle peaks of the ideal and the actual Loran pulse cannot exceed one percent of the peak amplitude of the actual pulse. Let S_p, p =1,2,3,...8, be the first eight half-cycle peak amplitudes of the actual pulse and those for the ideal be I_p, p=1,2,3,...8. When the maximum amplitudes of the actual and ideal pulses are normalized, the ensemble tolerance is expressed as:

$$\sqrt{\frac{\sum_{1}^{8} (I_p - S_p)^2}{8}} \le 0.01$$

c. Individual Tolerance of Half-Cycle Peak Amplitudes

For the first eight RF half-cycle peak amplitudes, the individual pulse peak error between the actual and the ideal must not exceed three percent of the peak amplitude of the pulse. For half-cycle peak amplitudes 9 through 13, the maximum individual error must not exceed ten percent of the peak amplitude. Assuming that the actual and ideal Loran pulses are normalized, these tolerances are expressed as:

$$|I_p - S_p| \le 0.03 \qquad 1 \le p \le 8\,,$$

$$|I_p - S_p| \le 0.10 \qquad 9 \le p \le 13.$$

d. Zero-crossings

Zero-crossing times and their acceptable deviation from the ideal zero-crossings are provided in Table 2.1 for the AN/FPN-44A transmitter. All zero-crossing times are in relation to the standard zero-crossing (SZC). The SZC is the negative to positive zero-crossing at 30 microseconds of a positively phase coded pulse of the antenna-current waveform. There are two categories for zero-crossing tolerances. Category 1 tolerances are for the newer model transmitters, such as the AN/FPN-44A, and category 2 are for the older transmitters, such as the AN/FPN-42. [Ref. 2: p. 2.4]

TABLE 2.1: Zero-crossing Times and Tolerances for AN/FPN-44A Transmitter

Zero-crossing (μs)	Time (μs)	±Tolerances (ns)
5	-25	1000
10	-20	100
15	-15	75
20	-10	50
25	-5	50
30	SZC	standard time reference
35	5	50
40	10	50
45	15	50
50	20	50
55	25	50
60	30	50

C. PRODUCING THE RF SIGNAL

1. The Transmitter Input and Output Waveforms

The input to a LORAN-C transmitter is called the transmitter drive waveform (TDW). The TDW is a cosine waveform with 16 half-cycles of varying peak amplitudes and has a constant carrier frequency of 100kHz. A damped sinusoid is added at the end of the 16 half-cycles to lengthen the decay time of the radio frequency antenna current waveform (RF). This eliminates undesirable high frequency components in the output. A typical TDW is shown in Figure 2.3.

When the shape of the TDW changes, its energy varies which in turn changes the unit sample response of the transmitter. This behavior exemplifies the transmitter as a non-linear system. An assumption is made that the transmitter operates as an LTI system from pulse to pulse with a fixed TDW, over a time duration of a few hours. In this work, an LTI pole-zero model was used to simulate the transmitter at a given operating point, i.e. a fixed TDW [Ref. 3: p.42]. By catenating a number of LTI models that cover a range of operating points based on different TDW, a piece-wise non-linear model is developed. The simulated AN/FPN-44A Loran-C transmitter is modeled using six poles and five zeros. The behavior of the corresponding poles and zeros of the catenated LTI models is characterized by fitting a polynomial curve for each of them. Each pole or zero moves along its own polynomial curve as a function of the TDW's energy. [Ref. 3]

Time variations in the transmitter may occur over several hours, days, or weeks. These time variations are modeled as slight shifts in the position of poles and zeros. Each polynomial curve fitting the trajectory of a pole or zero drifts up and down independently of each other when shifts occur. This allows the pole-zero transmitter model to simulate changes in transmitter characteristics due to time variations. [Ref. 3]

There are two different possible loads on the transmitter: the antenna and the resistive dummy load. The simulation contains an IIR model each for the antenna and the dummy load cases. The transmitter usually starts on the dummy load, where the TDW's half-cycle peak amplitudes converge to form an acceptable RF pulse before switching to the antenna. The TDW can be generated on either loads to form the RF pulse.

Each Loran station has two transmitters. This allows one transmitter to transmit on the antenna while the other remains in a standby mode of operation. The term transmitted pulse refers to the RF pulse measured at the transmitter's ground return, not the RF pulse in the far field. The typical RF pulse transmitted on the antenna is shown in Figure 2.4. The shape of the RF pulse is determined by the amplitudes of the half-cycles of the TDW. The TDW has 16 half-cycles. The half-cycle amplitudes of a TDW can be arranged into a column vector, which makes computing the optimal TDW a 16 dimensional control problem. The vector of parameters used to represent the RF pulse are comprised of the first

16 half-cycle peak amplitudes of the RF pulse, or it can be extended to include the RF pulse samples at the desired zero-crossings which allows for the control of zero-crossings.

In order to solve for the optimal TDW parameters (half-cycle peak values), we need to estimate the mapping function between the TDW and RF parameters spaces. The mapping function must take into account the non-LTI characteristics of the actual transmitter. A multiple input multiple output (MIMO) model is developed to represent the Loran transmitter's behavior on a pulse to pulse basis. The model parameters are estimated using a MIMO recursive least squares (RLS) algorithm. Further discussion of the modeling and control techniques are contained in Chapters III and IV, respectively.

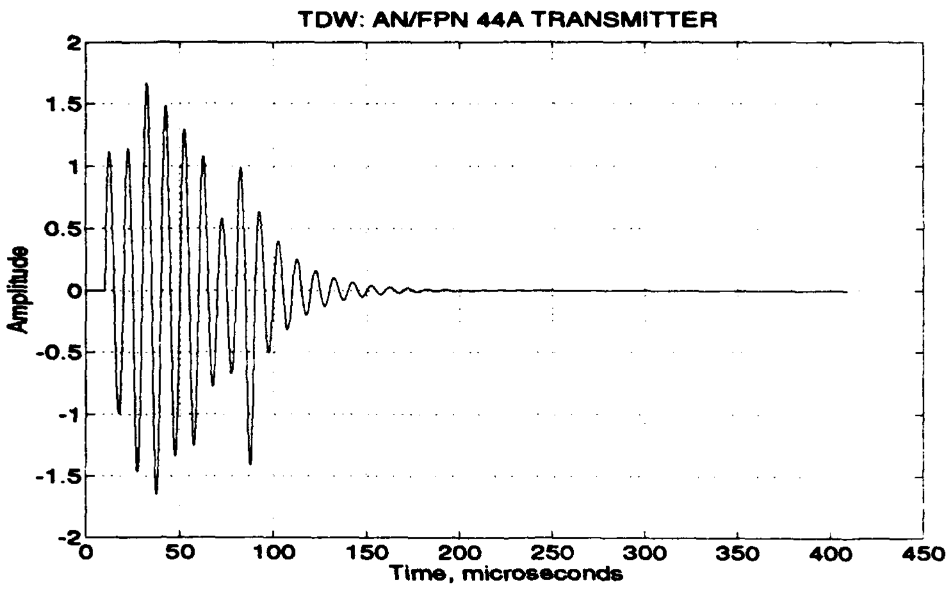

Figure 2.3: A typical TDW for the AN/FPN-44A transmitter.

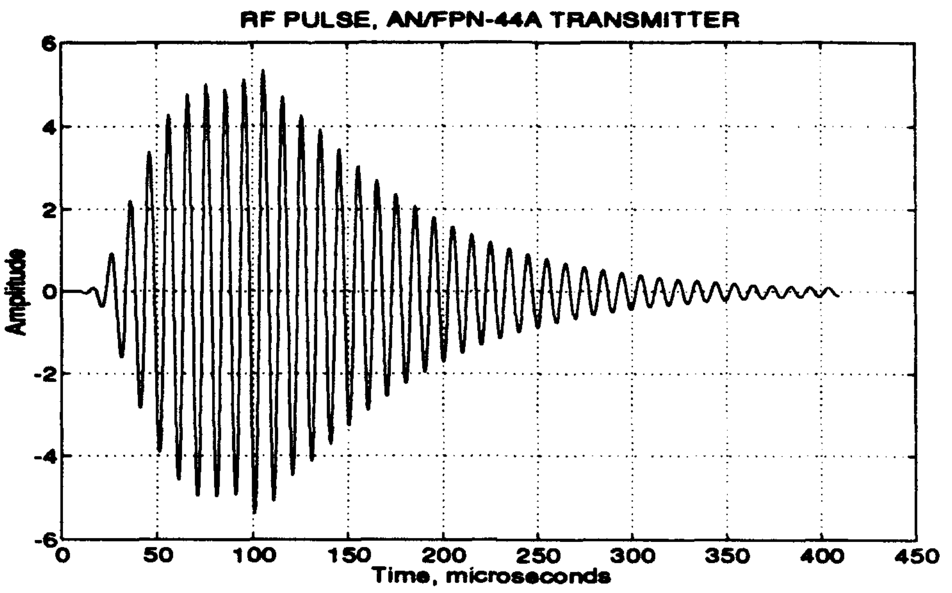

Figure 2.4: A typical RF Pulse from the AN/FPN-44A transmitter.

2. The VXI Based Loran-C Transmitter and Control System

Plan one of the Loran-C Electronic Equipment Replacement Project (EERP) was started in 1990; it is called project W1180: "Timing and Control Equipment (TCE) Redesign." A scheme to implement data acquisition and develop an algorithm to control the Loran-C pulse shape by adaptively generating the necessary TDW waveform were proposed [Ref. 6]. A block schematic of the proposed control system is shown in Figure 2.5.

The operation of the control system in Figure 2.5 follows these steps [Ref. 3, p. 33]. The computer loads a digitized transmitter drive waveform (TDW) into 'he arbitrary function generator (AFG). The AFG produces an analog TDW signal, which is sent to the transmitter at each timer trigger. A digital storage oscilloscope (DSO) acquires the RF pulse with an eight bit resolution. The digitized RF pulse is stored in the computer's memory which is used by the control algorithm to generate an optimal TDW. The controller computes a new TDW based on the error between the ideal and actual RF pulses. Plan one of the EERP does not address an exact algorithm to accomplish the goal of obtaining an

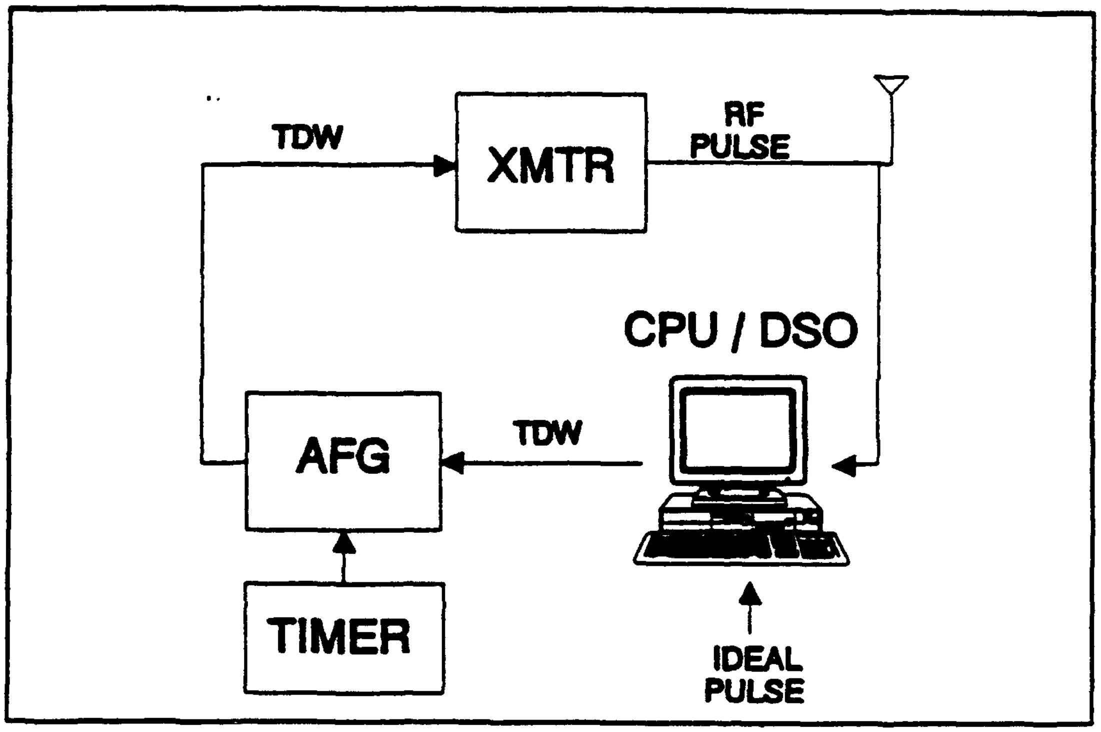

Figure 2.5: The VXIbus based control system [Ref.3: p. 34].

optimal TDW. In this research, we propose and develop a MIMO RLS control algorithm to generate an optimal TDW, where the actual RF pulse is a least squares fit of the ideal RF pulse.

III. MIMO MODELING

In this chapter we present the derivation of a multiple-input, multiple-output model for the Loran-C transmitter based on a least squares data formulation. The half-cycle peaks of the transmitter drive waveform and the RF pulse are considered as multichannel input and output quantities. In order for the algorithm to be able to track the slow time variations in the transmitter operating environment, an adaptive version of the least squares formulation called the recursive least squares algorithm is proposed. Both forward and inverse models are considered which are needed to develop suitable pulse shape control algorithms in the following chapters.

A. MULTICHANNEL SYSTEM APPROACH

Any adaptive algorithm used to control the RF pulse requires a reference model of the transmitter. The reference model is used to map the error in the RF pulse to the TDW input. A reference model is obtained using a multichannel formulation when the input and output parameters are considered to be channels. The output is expressed as:

$$y_t = A_0 x_t + A_1 x_{t-1} + \dots + A_m x_{t-m}, \tag{3.1}$$

where x_t is the input vector (p x 1), y_t is the output vector (q x 1) at discrete time index t, and A_s (s = 0, 1,..., m) are the coefficient matrices (q x p) [Ref. 7: p. 237]. For p = q, the coefficients are square matrices. We consider p = 16, where the elements of x represent the half-cycle peaks of the transmitter drive waveform (i.e., a total of eight cycles). The output vector y can be of size q = 16, or q = 32. For q = 16, vector y is composed of the first 16 half-cycle peaks of the RF pulse. When q = 32, vector y is expanded to include the samples of the first 16 desired zero-crossings. When p does not equal q, rectangular coefficient matrices result. Both cases are considered in the following chapters.

The MIMO reference model is obtained based on a least squares formulation. A least squares estimate of the coefficient matrices is obtained by minimizing the sum of squared errors between the actual output of the transmitter and the computed output of the model.

In this work a memoryless MIMO model is considered. From equation 3.1, the output of the memoryless system is expressed as

$$y_t = A_0 x_t. \tag{3.2}$$

Additional coefficient matrices are used when a model with memory is to be realized. The derivation of the least squares estimate for the MIMO model remains the same whether one or more coefficient matrices are used.

B. THE TRANSMITTER MODEL

1. The Least Squares Method

Considering a memoryless model representation, the output of the multichannel model is expressed as:

$$\hat{y} = Ax,$$

where x is the vector of input parameters, and A is the single coefficient MIMO model. The error vector, at time index i, is expressed as:

$$e_i = y_i - \hat{y}_i, \tag{3.3}$$

where $\hat{y}_i = Ax_i$, and y_i is the vector of RF pulse parameters (actual transmitter output). The weighted, squared error matrix is then given by [Ref. 10: p. C.7.3]:

$$J = \sum_{i=1}^{n} \lambda^{n-i} (e_i e_i^T) W. \tag{3.4}$$

where λ is called the forgetting factor; $(1-\lambda)^{-1}$ is a measure of the memory of the recursive least squares algorithm; a forgetting factor of unity corresponds to infinite memory; and W is a diagonal weighting matrix used to weight the elements of e_i.

Expanding the error, e_i, in equation 3.4 yields the following:

$$J = \sum_{i=1}^{n} \lambda^{n-i} (y_i - Ax_i)\,(y_i - Ax_i)^T W,$$

or

$$J = \sum_{i=1}^{n} \lambda^{n-i} (y_i y_i^T - y_i x_i^T A^T - A x_i y_i^T - A x_i x_i^T A^T) W. \qquad (3.5)$$

Taking the partial derivative of J with respect to A in equation 3.5 and setting it to zero yields:

$$\frac{\partial J}{\partial A} = \sum_{i=1}^{n} \lambda^{n-i} (-2 y_i x_i^T + 2 A x_i x_i^T) W = 0 \ , \qquad (3.6)$$

Rearranging terms in equation 3.6 gives:

$$\sum_{i=1}^{n} \lambda^{n-i} (y_i x_i^T) W = \sum_{i=1}^{n} \lambda^{n-i} (A x_i x_i^T) W. \qquad (3.7)$$

Now define the following matrices: the autocorrelation matrix,

$$\Phi_n = \sum_{i=1}^{n} \lambda^{n-i} (x_i x_i^T) W. \qquad (3.8)$$

and the cross-correlation matrix,

$$\Gamma_n = \sum_{i=1}^{n} \lambda^{n-i} (y_i x_i^T) W. \qquad (3.9)$$

Substituting equation 3.8 and 3.9 into equation 3.7 results in a simplified matrix form:

$$A \, \Phi_n = \Gamma_n; \qquad (3.10)$$

and the least squares memoryless MIMO model is obtained as [Ref. 8: p. 380]:

$$A = \Gamma_n \, \Phi_n^{-1}.$$

2. The Recursive Least Squares Algorithm

Consider that the transmitter being modeled is slowly time-varying; accordingly, the model coefficient is now represented as A_n where n is the time index. To continuously model the transmitter as its characteristics change with time, we propose to develop a recursive least squares solution of A [Ref. 8: p. 477- 485].

From equation 3.8, let

$$P_n^{-1} = \Phi_n = \sum_{i=1}^{n} \lambda^{n-i}(x_i x_i^T) W , \qquad (3.11)$$

and equation 3.10 becomes

$$A_n P_n^{-1} = \Gamma_n. \qquad (3.12)$$

At time index n+1, equation 3.12 can be written as:

$$A_{n+1} P_{n+1}^{-1} = \Gamma_n + y_{n+1} x_{n+1}{}^T W. \qquad (3.13)$$

Substituting equation 3.12 in 3.13 for Γ_n yields

$$A_{n+1} P_{n+1}^{-1} = A_n P_n^{-1} + y_{n+1} x_{n+1}{}^T W. \qquad (3.14)$$

Adding ($A_n x_{n+1} x_{n+1}{}^T W$ - $A_n x_{n+1} x_{n+1}{}^T W$) on the right side of equation 3.14 and realizing that:

$$A_n P_n^{-1} + A_n x_{n+1} x_{n+1}{}^T W = A_n P_{n+1}^{-1}, \qquad (3.15)$$

and

$$- A_n x_{n+1} x_{n+1}{}^T W + y_{n+1} x_{n+1}{}^T W = (y_{n+1} - A_n x_{n+1}) x_{n+1}{}^T W, \qquad (3.16)$$

results in

$$A_{n+1} P_{n+1}^{-1} = A_n P_{n+1}^{-1} + (y_{n+1} - A_n x_{n+1}) x_{n+1}{}^T W, \qquad (3.17)$$

where $(y_{n+1} - A_n x_{n+1}) = e_{n+1}$. The final form of the recursive equation for A_{n+1} is

$$A_{n+1} = A_n + e_{n+1} x_{n+1}{}^T W P_{n+1}. \qquad (3.18)$$

The recursive least squares algorithm in equation 3.18 requires recursive updating of P_n [Ref 8: p.479]. At time index n+1, equation 3.11 becomes

$$P_{n+1}^{-1} = \lambda \Phi_n + x_{n+1} x_{n+1}^T W \cdot \qquad (3.19)$$

Substituting $P_n^{-1} = \Phi_n$ into equation 3.19 yields:

$$P_{n+1}^{-1} = \lambda P_n^{-1} + x_{n+1} x_{n+1}^T W. \quad (3.20)$$

Inverting both sides of equation 3.20:

$$P_{n+1} = (\lambda P_n^{-1} + x_{n+1} x_{n+1}^T W)^{-1}. \quad (3.21)$$

Applying the matrix inversion lemma [Ref. 8: p. 480]

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \quad (3.22)$$

to equation 3.21 provides the required recursion for P_n. Comparing the terms on the right side of equation 3.21 to those on the left side of equation 3.22, we have

$$A = \lambda P_n^{-1},$$

$$B = x_{n+1},$$

$$C = I,$$

$$D = x_{n+1}^T W,$$

which yields the update equation for P_{n+1}:

$$P_{n+1} = \frac{1}{\lambda}P_n - \frac{1}{\lambda} P_n x_{n+1} \left[I + x_{n+1}^T W \frac{1}{\lambda} P_n x_{n+1} \right]^{-1} x_{n+1}^T W \frac{1}{\lambda} P_n. \quad (3.23)$$

The RLS algorithm is one of many possible algorithms for obtaining a reference MIMO model for the transmitter system. The RLS algorithm is chosen because it offers a fast rate of convergence, with negligible noise when λ is unity. The computational complexity of the algorithm is rather demanding, but it is quite simple to implement the equations in MATLAB software. Simplifications, such as the fast RLS algorithm could be considered to overcome the computational complexity of the RLS algorithm presented here [Ref. 8, 10]. The issue of the fast RLS algorithm for reduced complexity is not addressed in this thesis.

3. Estimation of the MIMO Model Parameters

We now present simulation results of the RLS algorithm developed in the previous section. The MIMO model parameters are estimated using equation 3.18. To implement the RLS algorithm, a set of 2000 pairs of input and output data are generated from a Loran-C transmitter simulation model that uses a steepest descent algorithm [Ref.3]. Initially, P is set to $P = 10^5 I$, where I is an identity matrix. The matrix P and the MIMO model are saved after 2000 iterations for later use in the pulse shaping control algorithm.

Figure 3.1 is a plot of the mean square error, MSE:

$$e_{n+1}^T e_{n+1} = (y_{n+1} - A_n x_{n+1})^T (y_{n+1} - A_n x_{n+1}),$$

for 2000 iterations of the algorithm. Figure 3.2 is a plot of a norm, ζ, as defined as

$$\zeta = \sqrt{\sum_i \sum_j (a_{ij}^{(n+1)} - a_{ij}^{(n)})^2},$$

where a_{ij} are the elements of A for 2000 iterations. The MIMO RLS algorithm is effective in producing a model that converges to a reasonable MSE. In Figure 3.1, the MSE decreases to a near steady state value in less than five iterations and generally remains below 1e-7 after that.

In Figure 3.2, ζ reaches a minimum at iteration 125, which indicates the numerical change of the elements of the model from iteration to iteration. The model obtained at the minimum ζ does not differ significantly from the model taken at the end of 2000 iterations when used to initialize the control algorithm.

The following points are noted on the P matrix. The matrix is symmetric, positive semidefinite, and has full rank for all iterations. The condition number of P is very large, on the order of 1e9; ill-conditioned data sequences have correlation matrices with large condition numbers. Based on this empirical observation, the RLS algorithm presented here can be considered robust. An algorithm is said to be robust if it operates satisfactorily with ill-conditioned data [Ref. 8: p. 3].

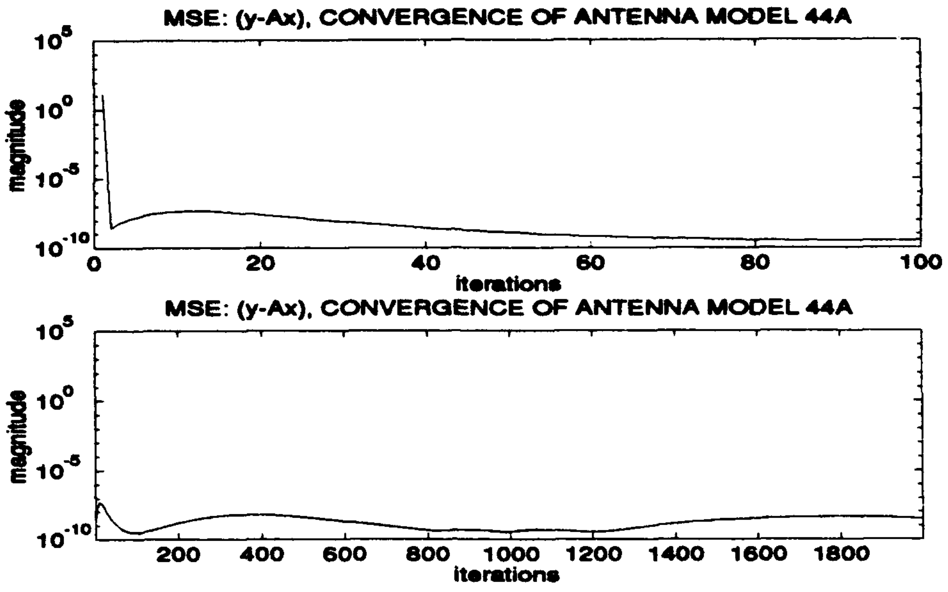

Figure 3.1: Antenna model convergence, MSE: (y - Ax).

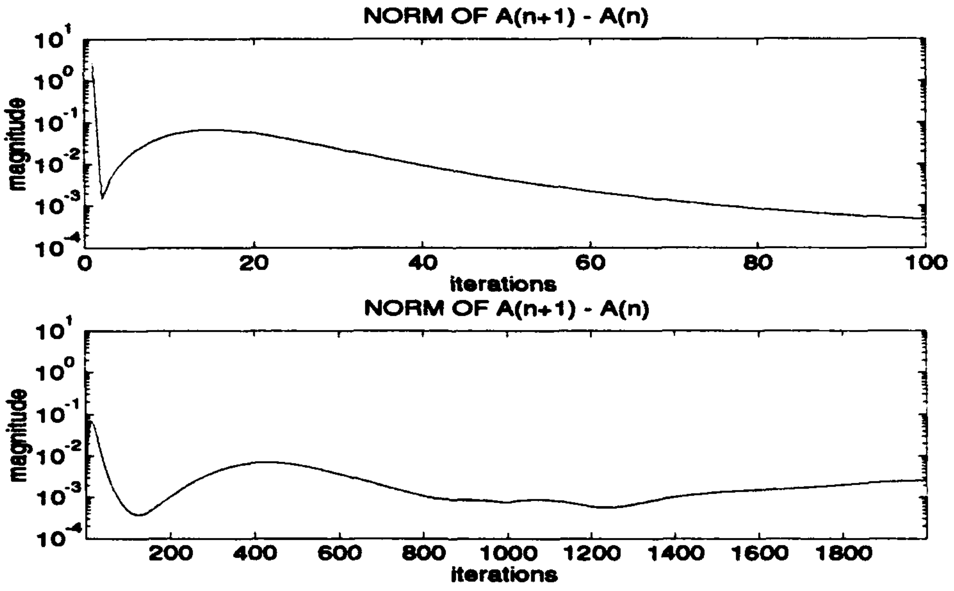

Figure 3.2: Norm of $(A_{n+1} - A_n)$.

4. Operation of the RLS in a Slowly Time-varying Environment

When the RLS algorithm operates in a time-varying environment, the suggested value of λ is usually less than unity. This gives the RLS algorithm a finite memory where slow statistical changes in its environment can be tracked. However, changing the value of λ to less than unity modifies the behavior of the algorithm by introducing misadjustment noise and delay in the formulation of the least squares estimate. [Ref.8: p. 499]

The statistical variations in the environment are considered negligible after several initial iterations or the transient period during which the algorithm converges to steady state parameters. A forgetting factor of unity is used for the RLS algorithm in this thesis; in spite of the presence of slow time variations, $\lambda = 1$ provides the best parameter tracking performance. Several tests were run with different forgetting factors, and the following observations were made: the finite update memory for $\lambda < 1$ increases the misadjustment noise; for $\lambda < 0.98$, the stalling of the algorithm updates was observed; and the forgetting factor being less than one has not improved the convergence speed in general.

In this chapter, a MIMO memoryless model of a Loran-C transmitter is developed with its coefficient matrix estimated using a recursive least squares algorithm. The RLS algorithm provides fast parameter convergence, but it is computationally expensive. The means to reduce the computational complexity is not addressed here. Even though the forgetting factor, λ, is considered an important quantity when the algorithm is operating in a time-varying environment, best performance is achieved for $\lambda = 1$. The estimated MIMO model is used as a reference model in the MIMO RLS control algorithm. An inverse MIMO model can be estimated on the lines of the MIMO model discussed in this chapter; Appendix C contains a brief derivation of the inverse MIMO model.

IV. OPTIMAL TDW ESTIMATION

An algorithm to shape the RF pulse which uses a recursive least squares formulation is proposed. The TDW is updated using the error in the RF pulse parameters; this error is used to correct the TDW parameters to produce an RF pulse which is a close match to the ideal pulse (equation 2.1). A reference model of the transmitter is required to formulate the correction. The proposed control algorithm uses a MIMO RLS estimate of the reference model (see Chapter III). A related algorithm, called the steepest descent, which uses an impulse response matrix as the transmitter reference model, is also briefly discussed.

A. MIMO RLS ALGORITHM FOR UPDATING TDW

A MIMO least squares algorithm is used to formulate the optimal TDW parameter vector, x_{opt}, to produce the desired RF pulse parameter vector, y_{opt}. Obtaining the optimal TDW is an adaptive process because the transmitter's characteristics vary as the energy of the TDW parameters in vector x changes. The transmitter reference model A is continually updated as a MIMO RLS estimate to track the changes in the transmitter. Figure 4.1 shows a block diagram of the proposed scheme to control the RF by continuously updating the TDW. The "control" block generates a correction Δx at each update with reference model parameters, RF error and the previous TDW vector as inputs.

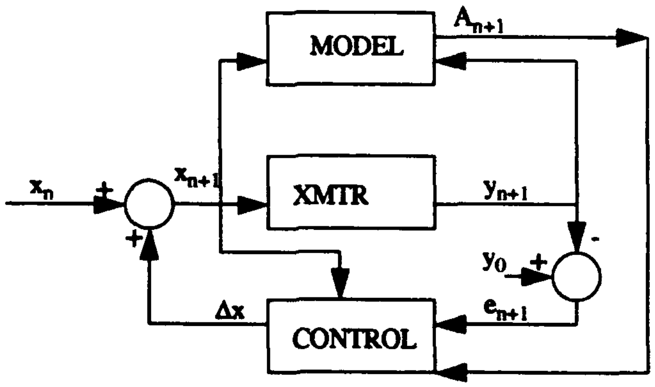

Figure 4.1: A block diagram of the RLS control scheme.

1. The Least Squares Formulation

The error vector, at time index i, in the feedback path of the controller (Figure 4.1) is expressed as

$$e_i = y_{opt} - \hat{y}_i,$$

where y_{opt} is a vector of the ideal RF pulse parameters; vector $\hat{y}_i = A_i x$, where matrix A_i is the adaptive MIMO model, and x is a vector of TDW parameters. In order to develop a least squares formulation, x is considered to be independent of the time index i. The weighted sum of squared errors is expressed as

$$J = \sum_{i=1}^{n} \lambda^{n-i} (e_i^T V e_i), \tag{4.1}$$

where λ is called the forgetting factor, and V is a diagonal weighting matrix used to weigh the elements of e_i. Expanding the error, e_i, in equation 4.1 yields

$$J = \sum_{i=1}^{n} \lambda^{n-i} ((y_{opt} - A_i x)^T V (y_{opt} - A_i x)) \quad , \tag{4.2}$$

or

$$J = \sum_{i=1}^{n} \lambda^{n-i} (y_{opt}^T V y_{opt} - x^T A_i^T V y_{opt} - y_{opt}^T V A_i x$$

$$- x^T A_i^T V A_i x) \; . \tag{4.3}$$

Setting the partial derivative of J with respect to x in equation 4.3 equal to zero

$$\frac{\partial J}{\partial x} = \sum_{i=1}^{n} \lambda^{n-i} (-2A_i^T V y_{opt} + 2A^T{}_i V A_i x) = 0, \tag{4.4}$$

minimizes the cost function J with respect to x. Rearranging the terms in equation 4.4 yields

$$\sum_{i=1}^{n} \lambda^{n-i}(A^T{}_i V y_{opt}) = \sum_{i=1}^{n} \lambda^{n-i}(A^T{}_i V A_i) x . \qquad (4.5)$$

Defining the autocorrelation matrix as

$$\Phi_n = \sum_{i=1}^{n} \lambda^{n-i}(A^T{}_i V A_i), \qquad (4.6)$$

and the cross-correlation vector as

$$\gamma_n = \sum_{i=1}^{n} \lambda^{n-i}(A^T{}_i V y_{opt}), \qquad (4.7)$$

and substituting equations 4.6 and 4.7 into equation 4.5 produces a simplified matrix form

$$\Phi_n x = \gamma_n. \qquad (4.8)$$

From equation 4.8, the optimal TDW is in the form of a least squares solution

$$x = \Phi_n^{-1} \gamma_n. \qquad (4.9)$$

2. The Recursive Least Squares Algorithm

This derivation is somewhat different from the RLS algorithm developed in Chapter III, Section B.2. For the recursive least squares update of x at time index n+1, let $P_n^{-1} = \Phi_n$, and let x be a function of the discrete time index n. Equation 4.8 is now expressed as

$$P_n^{-1} x_n = \gamma_n. \qquad (4.10)$$

At time index n+1, equation 4.10 can be written as

$$P_{n+1}^{-1} x_{n+1} = \gamma_n + A_{n+1}{}^T V y_{opt}, \qquad (4.11)$$

and substituting equation 4.10 in equation 4.11 for γ_n gives

$$P_{n+1}^{-1} x_{n+1} = P_n^{-1} x_n + A_{n+1}{}^T V y_{opt}. \qquad (4.12)$$

When adding $(A_{n+1}{}^T V A_{n+1} x_n - A_{n+1}{}^T V A_{n+1} x_n)$ on the right side of equation 4.12 and realizing that

$$P_n^{-1} x_n + A_{n+1}{}^T V A_{n+1} x_n = P_{n+1}^{-1} x_n, \qquad (4.13)$$

and

$$A_{n+1}{}^T V y_{opt} - A_{n+1}{}^T V A_{n+1} x_n = A_{n+1}{}^T V (y_{opt} - A_{n+1} x_n), \qquad (4.14)$$

equation 4.12 becomes

$$P_{n+1}^{-1} x_{n+1} = P_{n+1}^{-1} x_n + A_{n+1}{}^T V (y_{opt} - A_{n+1} x_n), \qquad (4.15)$$

where $(y_{opt} - A_{n+1} x_n) = e_{n+1/n}$. Multiplying both sides of equation 4.15 by P_{n+1}, and substituting $(y_{opt} - A_{n+1} x_n) = e_{n+1/n}$, yields the final recursive equation

$$x_{n+1} = x_n + P_{n+1} A_{n+1}{}^T V e_{n+1/n}. \qquad (4.16)$$

For the recursive update of P_n, we write equation 4.6 at time index n+1 as

$$P_{n+1}^{-1} = \Phi_{n+1} = \lambda \Phi_n + A^T{}_{n+1} V A_{n+1} \cdot \qquad (4.17)$$

Substituting $P_n^{-1} = \Phi_n$, into equation 4.17 gives

$$P_{n+1}^{-1} = \lambda P_n^{-1} + A^T{}_{n+1} V A_{n+1}, \qquad (4.18)$$

and inverting both sides of equation 4.18 yields

$$P_{n+1} = (\lambda P_n^{-1} + A^T{}_{n+1} V A_{n+1})^{-1}. \qquad (4.19)$$

Apply the matrix inversion lemma

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \qquad (4.20)$$

to equation 4.19 by comparing the right side of equation 4.19 to the left side of equation 4.20, where

$$A = \lambda P_n^{-1},$$

$$B = A^T{}_{n+1},$$

$$C = V,$$

$$D = A_{n+1},$$

the update equation for P_{n+1} becomes the right side of equation 4.20

$$P_{n+1} = \frac{1}{\lambda} P_n - \frac{1}{\lambda} P_n A^T_{n+1} \left[V^{-1} + A_{n+1} \frac{1}{\lambda} P_n A^T_{n+1} \right]^{-1} A_{n+1} \frac{1}{\lambda} P_n.$$

When the MIMO RLS control algorithm is implemented, the elements of the inverse correlation matrix, P_n, become very small values in approximately 150 iterations. This is typically referred to as the stalling phenomenon [Ref. 8: p. 701]. When this occurs, the algorithm stops updating the TDW parameters, and it no longer seeks a lower MSE. A remedy to keep the MIMO RLS control algorithm converging to a lower MSE is to reset P_n every 50 iterations or so.

B. STEEPEST DESCENT CONTROL ALGORITHM

A linear feedback control scheme using a steepest descent algorithm is discussed in this section. This approach seeks the optimal TDW parameters by minimizing the quadratic error surface of $e^T_y W e_y$, where W is a weight matrix and $e_y = (y_{opt} - y)$ [Ref. 3, 4].

The transmitter output is

$$y = Hx,$$

where the transmitter is modeled as impulse response matrix H, vector x is the TDW half-cycle peak parameters, and vector y is the RF half-cycle peak parameters. The simplest form of the steepest descent control algorithm has x and y parameter with 16 half-cycle peak amplitudes. H is formed by the half-cycle peak amplitudes of the impulse response of the transmitter; H is a 16 x 16 matrix.[Ref. 3: p. 82]

We assume that the system matrix H is an accurate model, and the optimal TDW half-cycle peak amplitudes in vector x_{opt}, the ideal pulse peaks are given as

$$y_{opt} = H\, x_{opt}.$$

The vector of errors in the TDW half-cycle peak amplitudes is

$$e_x = x_{opt} - x.$$

Let the correction to the TDW parameter vector x be expressed as Δx; which updates x at each iteration in the direction of steepest descent on the quadratic error surface. We express Δx as

$$\Delta x = (-\frac{\mu}{2})\Delta_{e_x}\left[e_y^T W e_y\right], \tag{4.21}$$

where Δ_{e_x} is the gradient with respect to e_x and μ is a small positive constant. Equation 4.21 is rewritten in terms of e_x

$$\Delta x = (-\frac{\mu}{2})\nabla e_x\left[e^T{}_x H^T W H^T e_x\right]. \tag{4.22}$$

Taking the partial derivative of Δx with respect to e_x, equation 4.22 becomes

$$\Delta x = (-\mu) H^T W H e_x . \tag{4.23}$$

Substituting $e_y = He_x$ in equation 4.23, the correction term

$$\Delta x = (-\mu) H^T W e_y ,$$

where the adaptation constant μ is bounded as [Ref. 3, 4]

$$\mu < \text{largest eigenvalue of } \frac{2}{\left[H^T W H\right]} .$$

The final form of the steepest descent control algorithm is

$$x_{n+1} = x_n + (-\mu) H^T W e_y .$$

The MIMO RLS control algorithm and steepest descent control algorithm are implemented in a simulated control system with a simulated transmitter which allows testing, analysis, and comparison of the control algorithms in Chapter V. With a system matrix, the RF error and the past TDW parameter vector, both control algorithms are able to produce a correction to update the TDW parameters. As the RF pulse parameters approach the ideal values, the TDW parameters converge to optimal values.

V. RESULTS AND COMPARISONS OF TDW CONTROL ALGORITHMS

A. INTRODUCTION

The results of the MIMO RLS control algorithm are presented in this chapter. Comparisons are made between the MIMO RLS and steepest descent control algorithms. The RLS control algorithm is compared against the steepest descent control algorithm under ideal conditions of machine precision, as well as under low noise conditions with eight bits of resolution for data acquisition; the digital storage oscilloscope used to sample the RF pulse has an eight-bit resolution. The signal to noise ratio (SNR) used in each trial for both the TDW and RF pulse is provided in each case:

$$\mathrm{SNR} = 10 \log_{10} \frac{P_s}{P_n},$$

where P_s is the peak signal power and P_n is the average noise power. [Ref. 3: p. 40]

The reference model for each control algorithm is derived differently, resulting in a majcr difference in their control techniques. An adaptive MIMO RLS model is used in the MIMO RLS control algorithm. The steepest descent control algorithm uses an impulse response matrix for its reference model [Ref. 3: p 81].

The results will consist of examining the root mean of squared RF peak errors 1-8 (ensemble error), the maximum RF peak error in half-cycles 1-8 and 9-13 (individual errors), the mean square error (MSE) of the RF pulse peaks 1-16, and the zero-crossing location error in excess of the allowable tolerance. The simulated digital sampling oscilloscope operates at its highest sampling frequency of 10MHz which improves the accuracy in sampling the peaks and the zero-crossings.

For testing purposes, only the first pulse of the PCI is controlled for 400 iterations starting on the dummy load and then switched to the antenna after the errors in the pulse peaks meet a tolerance threshold. The initial TDW peak values for the first iteration of control can be arbitrary; however, constant values of ±1.70 volts were chosen.

B. PEAK SAMPLING

In presenting the results, we consider two cases here. In the first, the TDW and RF waveform parameter vectors are of size 16x1 each; these are the first 16 peak amplitude values of the respective waveforms. In the second case (discussed in section C), the RF parameter vector contains both peak amplitudes and zero-crossing values making its size 32x1 while the TDW parameter vector remains unchanged. Simulation results are also presented for noiseless and noisy environments.

1. Results of RLS Control under Ideal Conditions

Under ideal (noise free) conditions, the TDW and RF pulse are sampled to the computer's machine precision, and the signals are free of any system noise usually present in the actual transmitter. There are no deliberate variations in the simulated transmitter's poles and zeros; changing pole-zero locations of the model simulate time variations in the transmitter. The simulated transmitter's characteristics can be changed by varying the TDW's energy and the power supply voltage [Ref. 2, 3]. The pole-zero locations stated above, fluctuate with variations in the TDW's energy and the transmitter's power supply droops within the GRI as RF pulses are transmitted every 1000μs.

The TDW parameters are updated, so the RF pulse half-cycle peak amplitudes converge on the dummy load, until the three measures of RF pulse peak errors are below a tolerance threshold. The tolerance threshold for each of these errors is different than those described in Chapter II for the error specifications; the tolerance threshold is chosen arbitrarily so the RF pulse peaks are in tolerance within an iteration or two during the swap from the dummy load to the antenna. The transmitter swap from dummy load to antenna can be observed in Figures 5.1-5.4, where the symbol â appears on the graph.

The various terms used in Figures 5.1-5.4 are explained in the following. In Figures 5.1 - 5.4: The "F Factor" is the forgetting factor used in the RLS control algorithm when shaping the RF pulse first on the dummy load and then on the antenna. The term "Drift" is explained in the next section; it is not used in these simulations. The expression "tau" refers to the ECD, which is set to zero for all simulation trials in this thesis. The "noise" is the variance of the additive noise. The expression "bits" indicates the number of bits used to sample the signals; a zero indicates machine precision.

Figures 5.1 and 5.2 illustrate the performance of the RLS control algorithm. In Figure 5.1, the convergence of three measures of RF pulse peak amplitude error (ensemble error, maximum of individual errors 1-8, and maximum of individual errors 9-13) are shown. Figure 5.2 illustrates the mean square error converging toward its minimum value on the error performance surface. Figures 5.3 and 5.4 illustrate the same error convergence for the steepest descent control algorithm. The RF pulse peak errors meet the specification tolerances for all iterations when using a signal to noise ratio (SNR) between 81dB and 89dB for the TDW and 95dB and 97dB for the RF pulse.

In Figure 5.1, the RLS control algorithm reduces the RF peak errors below the threshold in less than five iterations at which time the load is swapped. This is a vast improvement over the steepest descent's convergence on the dummy load in Figure 5.3. The steepest descent controller required over 50 iterations to reach the threshold (μ = 0.8 of μmax, where μmax is a predetermined value of 0.083).

After 400 iterations of pulse shape control, the final converged peak errors for the RLS and steepest descent control algorithms are recorded in Table 5.1. The MSE of the peak values (1-16), the ensemble error, the maximum error of peaks 1-8, and the maximum error of peaks (9-13) are an order of magnitude lower or less using the RLS controller.

The zero-crossing times are calculated with reference to the standard zero-crossing (SZC). The time difference from the maximum allowable tolerance is indicated in Table 5.2 for both the RLS the steepest descent algorithms. The zero-crossings are not in tolerance for either one of them.

Both control algorithms are able to shape the Loran-C pulse. The RF pulse peaks converge to a close approximation of the ideal RF pulse peak values. This is shown in Table 5.3.a for the RLS control algorithm and Table 5.3.b for the steepest descent control algorithm. In these tables, the actual and the ideal RF pulse parameters are normalized. The term "average" is the time average of the RF pulse parameters over each iteration of control. The term "ideal" refers to the first 16 half-cycle peak amplitudes of the normalized ideal pulse. The term "diff" is the difference between the column values specified. This gives the error in the RF peak values.

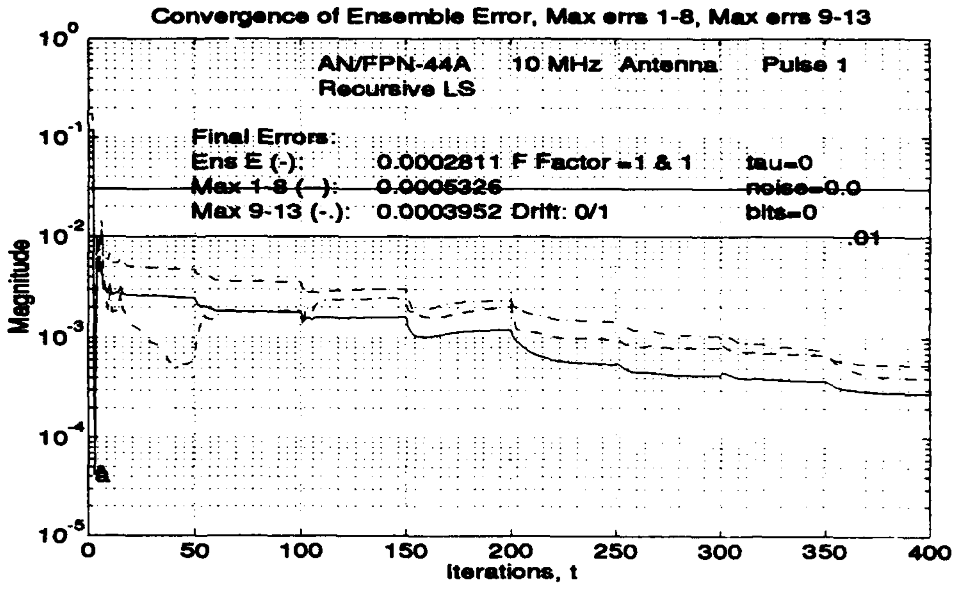

Figure 5.1: Convergence of peak amplitude tolerances of Loran-C error using the RLS control algorithm (machine precision, no noise) with '44A' transmitter.

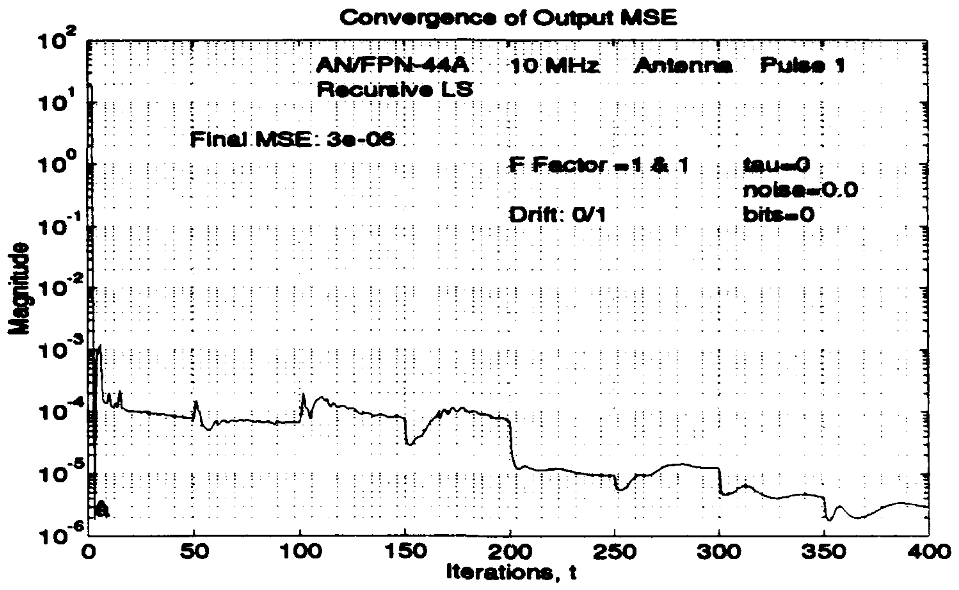

Figure 5.2: Convergence of MSE of RF peak amplitudes 1-16 using the RLS control algorithm (machine precision, no noise) with '44A' transmitter.

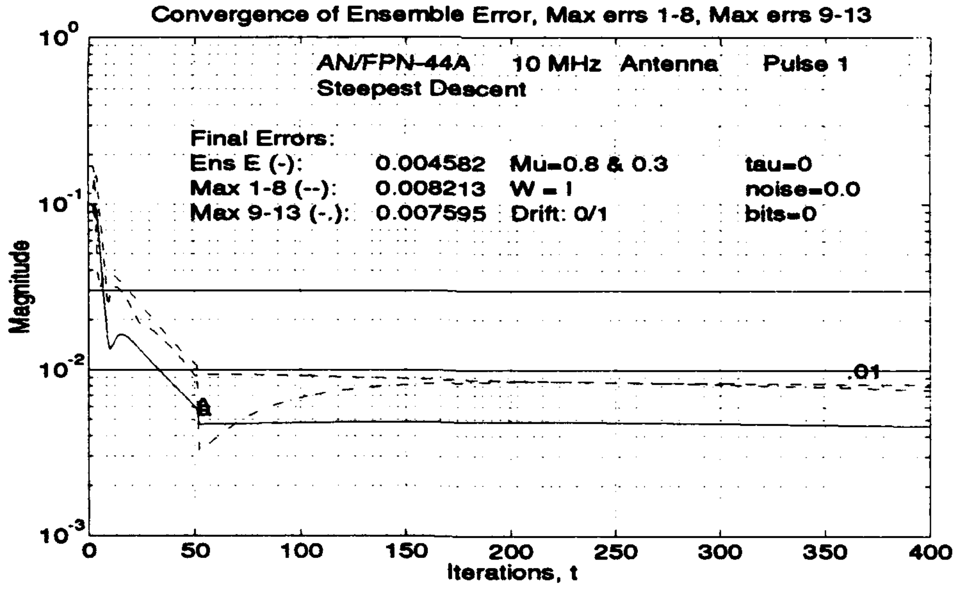

Figure 5.3: Convergence of peak amplitude tolerances of Loran-C error using the steepest descent control algorithm (machine precision, no noise) with '44A' transmitter.

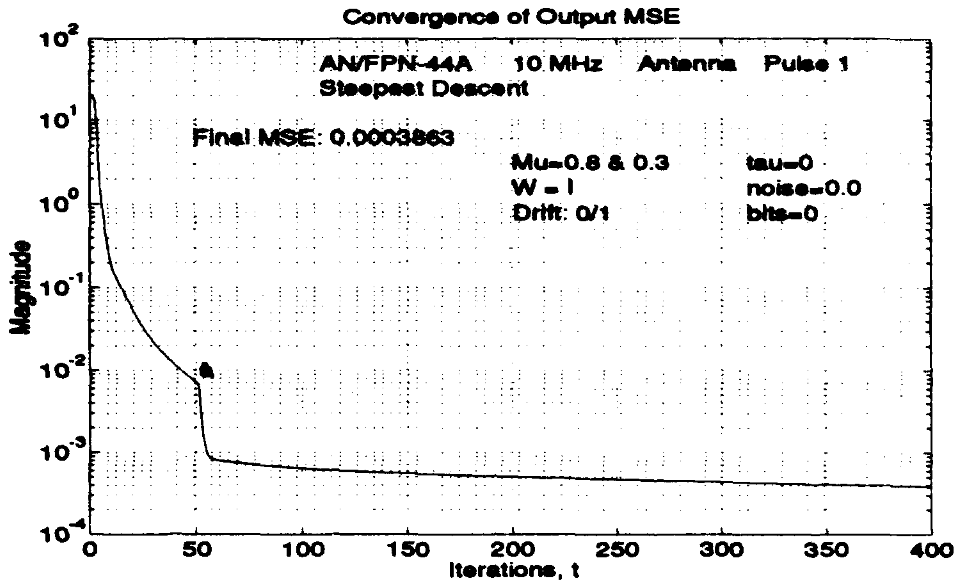

Figure 5.4: Convergence of MSE of RF peak amplitudes 1-16 using the steepest descent control algorithm (machine precision, no noise) with '44A' transmitter.

TABLE 5.1: RF Pulse Peak Tolerances After 400 Iterations (Machine Precision, No Noise)

Control Alg.	MSE out	Ens err	MaxE 1-8	MaxE 9-13
RLS	3.0e-6	2.8e-4	5.3e-4	4.0e-4
Steepest Descent.	4.0e-4	4.6e-3	8.2e-3	7.6e-3

TABLE 5.2: Zero-crossing Errors (ns) After 400 Iterations (Machine Precision, No Noise)

Cont. Alg.	1	2	3	4	5	6	7	8	9	10	11	12
RLS	0	-70.4	-21.8	0	0	0	0	0	0	0	0	0
St. Desc.	0	-118.0	-9.2	0	0	0	0	0	0	0	0	0

TABLE 5.3.a: Normalized RF Peak Values Obtained with RLS Control Algorithm (Machine Precision, No Noise)

Peak	Ideal	Average	After 400 iter	Diff (3-2)	Diff (4-2)
1	0.0157	0.0168	0.0157	0.0011	0.0001
2	-0.0833	-0.0819	-0.0835	0.0015	-0.0002
3	0.1901	0.1905	0.1898	0.0004	-0.0004
4	-0.3158	-0.3170	-0.3163	-0.0012	-0.0005
5	0.4454	0.4450	0.4453	-0.0004	-0.0002
6	-0.5696	-0.5702	-0.5697	-0.0007	-0.0001
7	0.6813	0.6823	0.6817	0.0010	0.0004
8	-0.7771	-0.7771	-0.7771	-0.0000	0.0001
9	0.8556	0.8568	0.8559	0.0012	0.0003
10	-0.9164	-0.9169	-0.9166	-0.0006	-0.0002
11	0.9598	0.9608	0.9599	0.0009	0.0001
12	-0.9872	-0.9884	-0.9876	-0.0013	-0.0004
13	1.0000	1.0000	1.0000	0.0000	0.0000
14	-1.0001	-1.0014	-1.0003	-0.0013	-0.0003
15	0.9892	0.9897	0.9894	0.0005	0.0002
16	-0.9692	-0.9701	-0.9694	-0.0010	-0.0002

TABLE 5.3.b: Normalized RF Peak Values Obtained with Steepest Descent Control Algorithm (Machine Precision, No Noise)

Peak	Ideal	Average	After 400 iter	Diff (3-2)	Diff (4-2)
1	0.0157	0.0202	0.0202	0.0046	0.0046
2	-0.0833	-0.0747	-0.0751	0.0086	0.0082
3	0.1901	0.1867	0.1873	-0.0034	-0.0028
4	-0.3158	-0.3190	-0.3191	-0.0032	-0.0033
5	0.4454	0.4453	0.4453	-0.0001	-0.0001
6	-0.5696	-0.5637	-0.5639	0.0059	0.0057
7	0.6813	0.6762	0.6763	-0.0052	-0.0051
8	-0.7771	-0.7759	-0.7758	0.0013	0.0013
9	0.8556	0.8542	0.8547	-0.0015	-0.0010
10	-0.9164	-0.9102	-0.9112	0.0062	0.0052
11	0.9598	0.9519	0.9522	-0.0079	-0.0076
12	-0.9872	-0.9836	-0.9829	0.0035	0.0042
13	1.0000	1.0000	1.0000	0.0000	0.0000
14	-1.0001	-0.9941	-0.9959	0.0060	0.0042
15	0.9892	0.9761	0.9778	-0.0131	-0.0114
16	-0.9692	-0.9716	-0.9698	-0.0024	-0.0006

2. Performance of the Control Algorithm under Non-ideal Conditions

The tracking performance of the control algorithms are tested by introducing a noise burst into the TDW parameters once the peak amplitudes of the RF pulse have converged for 200 iterations. White gaussian noise is added to the TDW parameters from iteration 200 to 210. The RF pulse is driven out of tolerance and then allowed to reconverge. Observing the RLS algorithm's tracking performance in Figures 5.5 and 5.6 after iteration 210, less than 25 control iterations were needed to bring the RF pulse's peak amplitudes back in tolerance. The RLS control algorithm required a noise variance of 0.04 to drive the RF peak amplitudes out of tolerance. The steepest descent algorithm used a noise with variance of 0.01 (one fourth of the noise variance used in the RLS) to drive the RF peaks out of tolerance. The RF pulse peaks do not reconverge in tolerance until after iteration 400, as observed in Figures 5.7 and 5.8. Using a noise burst with a variance of 0.04 in the steepest descent control drives the algorithm unstable, and it never regains control.

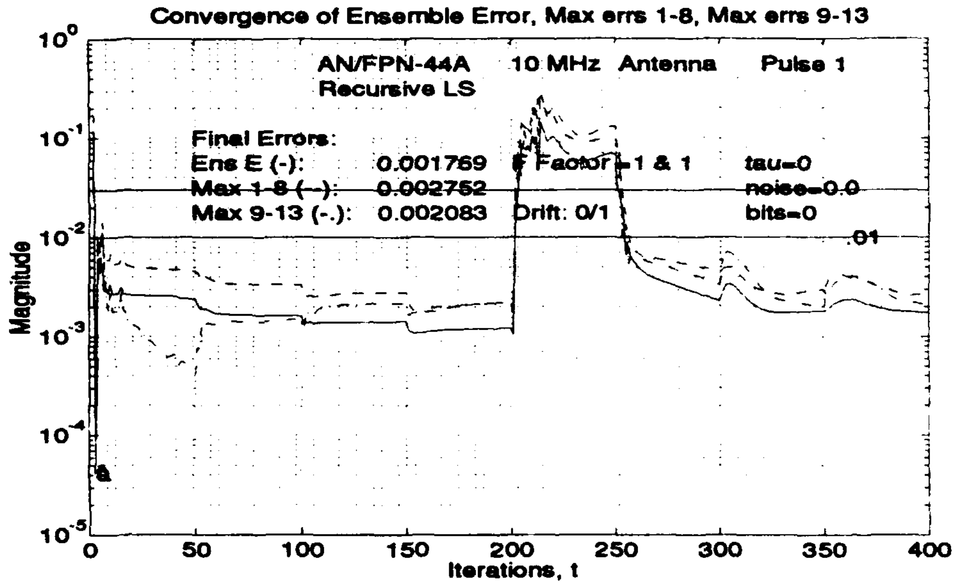

Figure 5.5: Peak amplitude tolerances using the RLS control algorithm (machine precision, no noise); tracking performance with noise burst (variance 0.04) during iterations 200-210.

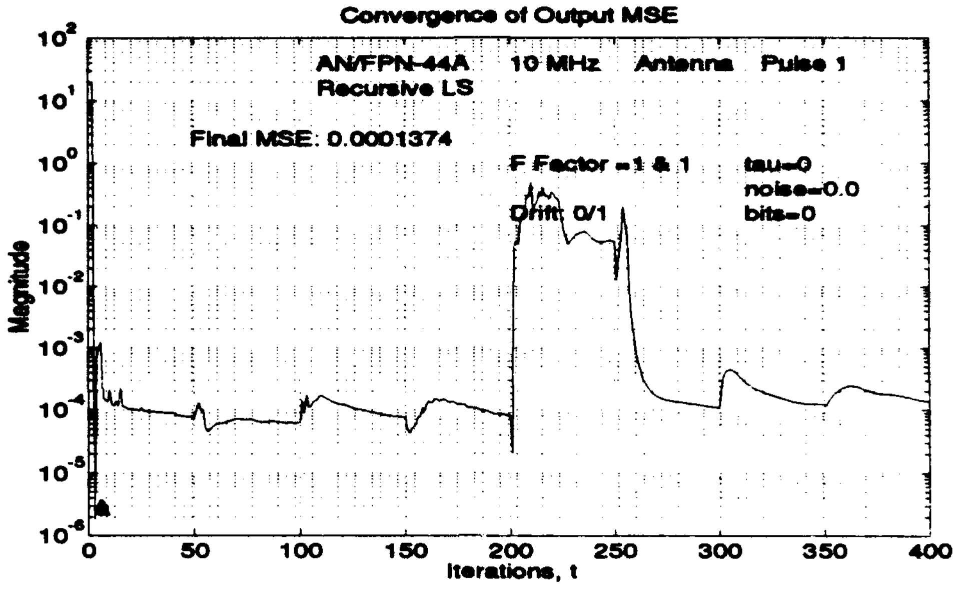

Figure 5.6: MSE of RF peaks 1-16 using the RLS control algorithm (machine precision, no noise); tracking performance with noise burst (variance 0.04) during iterations 200-210.

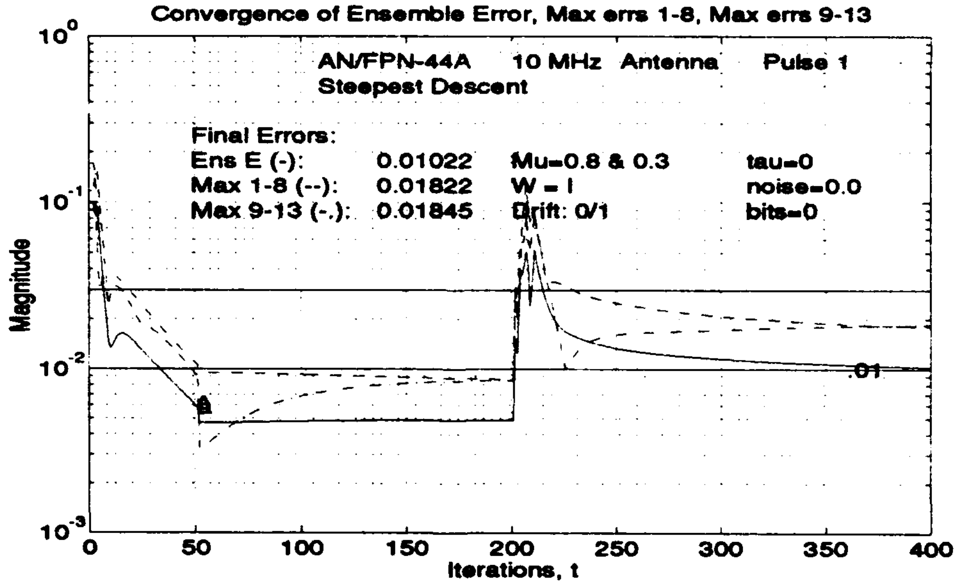

Figure 5.7: Peak amplitude tolerances using the steepest descent control algorithm (machine precision, no noise); tracking performance with noise burst (variance 0.01) during iterations 200-210.

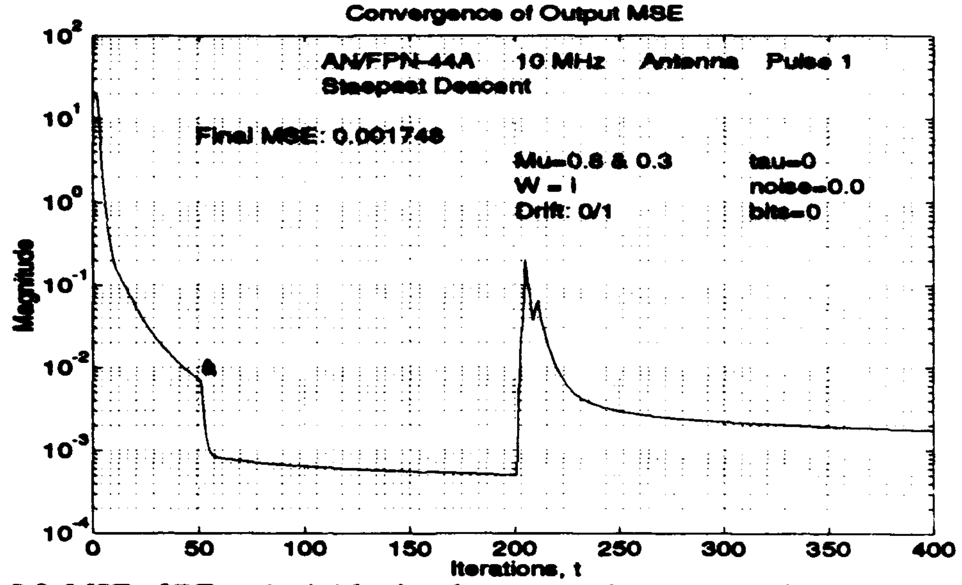

Figure 5.8: MSE of RF peaks 1-16 using the steepest descent control algorithm (machine precision, no noise); tracking performance with noise burst (variance 0.01) during iterations 200-210.

3. Finite Bit Resolution, Additive Noise, and Parameter Drift

The control algorithms are implemented in machine precision; however, the RF pulse and TDW are digitized at eight-bit resolution. The SNR of the TDW and RF pulse are lowered by adding white gaussian white noise to the signals; the white noise has a variance of 1.166 x 10^{-4}. The SNR of the TDW and RF pulse for these simulation results are 64dB and 73dB.

The tracking performance of the control algorithms are tested while time variations occur in the simulated transmitter. The time variations introduced are “drift” of the transmitter and transmitter “switches.” The poles and zeros that make up the transmitter’s IIR model are allowed to change slightly or “drift” to a new location within the predetermined bounds in order to simulate the transmitter’s time varying characteristics over days and weeks. These new poles and zeros correspond to “switching” to a new transmitter when the change is abrupt. The amount of variation occurring to the poles and zeros with drift and transmitter switching is shown in Figure 5.9 for the RLS control algorithm and Figure 5.10 for the steepest descent control algorithm.

In Figures 5.11-5.18, drift occurs every fourth iteration (see Figures 5.9 and 5.10), which is displayed as "Drift 4/1" in these graphs. The movement of poles and zeros for 400 iterations corresponds to a 25 hour period of slow transmitter time variations [Ref. 3: p. 86]. Transmitter switches occur at iteration 150 and 300, as indicated by the symbol $\hat{x}$. It was observed that the percentage of pulse peaks in tolerance for both control algorithms is very similar under these conditions.

The amount of fluctuation in the simulated transmitter due to drift and switching is random; therefore, direct comparison between pulse peak errors using different control algorithms is not beneficial. However, overall observations of how the algorithms respond to time variations can be made. Both algorithms are able to compensate for slow time variations in the simulated transmitter.

Figures 5.11-16 show the convergence performance of the RLS and steepest descent algorithms under non-ideal conditions. For both the RLS and steepest descent algorithms, as seen in Figures 5.11 and 5.14, the ensemble error occasionally exceeds the tolerance level while the individual peak errors (see Figures 5.12, 5.13, 5.15, and 5.16) remain in tolerance.

Figures 5.17 and 5.18 show the MSE convergence performance of the two algorithms when slow time variations are introduced in the simulated transmitter every fourth iteration. Both algorithms converge to a similar MSE of approximately 1×10^{-3} from iteration 75 to 150. At iteration 150, a transmitter switch is simulated. It is arbitrary whether the MSE will rise or lower. If the transmitter switch decreases the error between the simulated transmitter and the given reference model, the MSE decreases. If the switch provides a worse estimate of the reference model, the MSE rises. The RLS and steepest descent algorithms have difficulty converging to a lower MSE once a switch is made. The quantization noise and additive white noise degrade the tracking performance of the algorithms [Ref. 9: p. 67-81]. The RLS control algorithm, as with the steepest descent control algorithm, is sensitive to noise and large shifts in the poles and zeros of the transmitter model.

The error in the zero-crossing times for these trails are provided in Table 5.4 for both control algorithms. The second and third zero-crossing that were out of tolerance in the ideal environment are also out of tolerance in this environment, but to a greater extent.

After 400 iterations, errors in the RF pulse peaks for both control algorithms are provided in Table 5.5 and Tables 5.6.a and 5.6.b. Table 5.5 lists the MSE, ensemble error, and maximum individual peak error in half-cycles 1-8 and 9-13 with eight bit sampling and a noise of variance 1.2e-4. These errors were all within tolerance specifications. Table 5.6.a for the RLS algorithm, and Table 5.6.b for the steepest descent algorithm list individual RF pulse peaks taken after iteration 400.

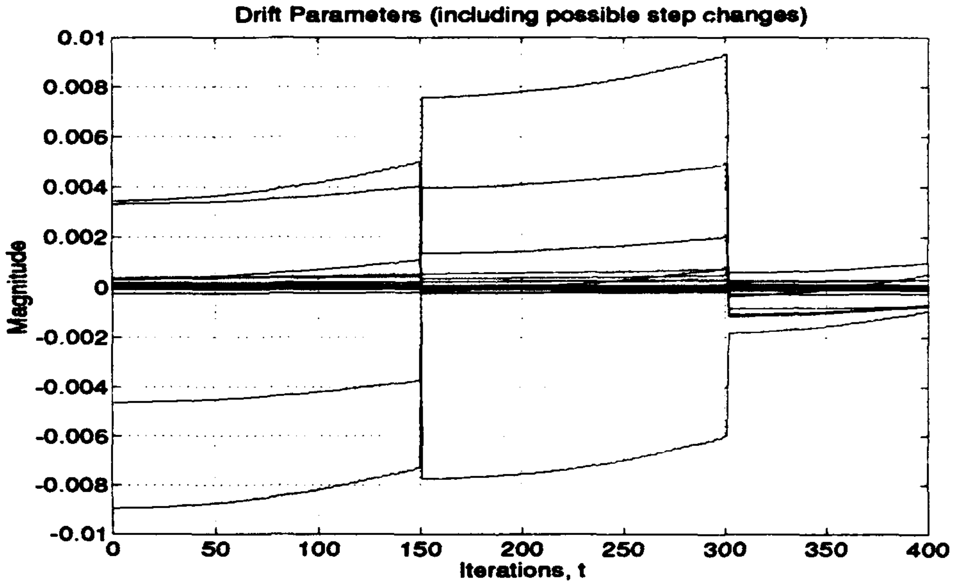

Figure 5.9: Drifting of the pole and zero magnitudes including step changes for the RLS control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

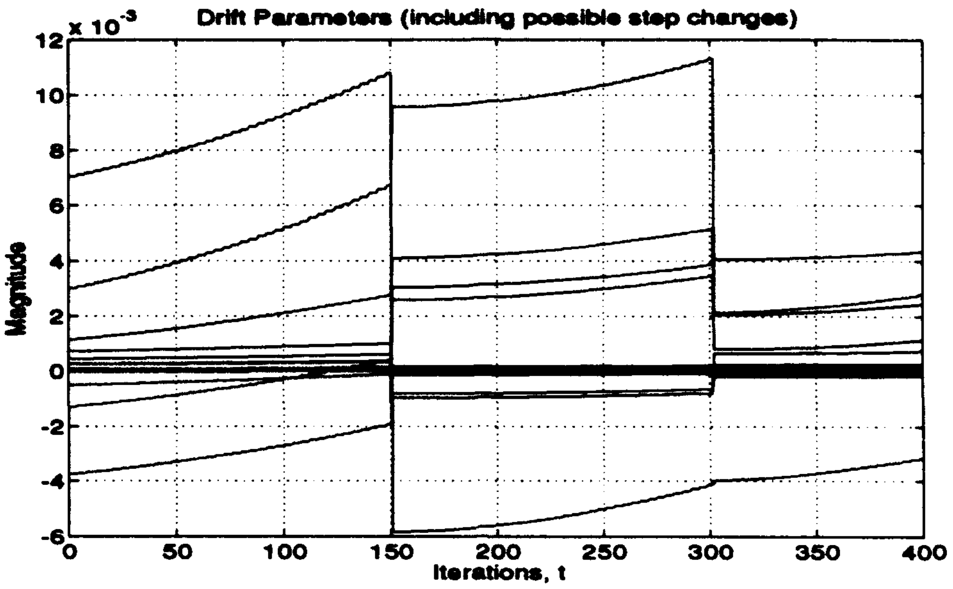

Figure 5.10: Drifting of the pole and zero magnitudes including step changes for the steepest descent control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

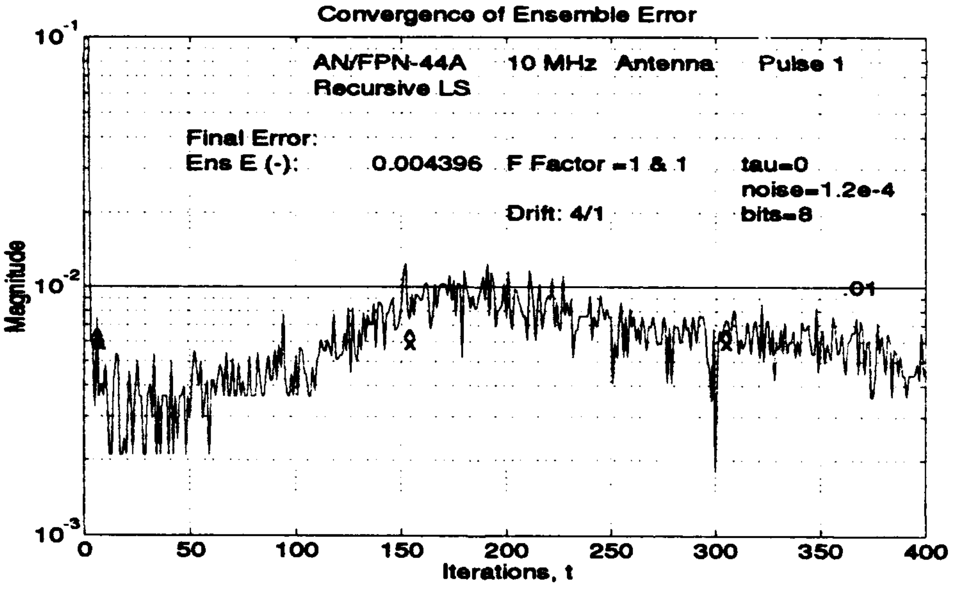

Figure 5.11: Ensemble error using the RLS control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

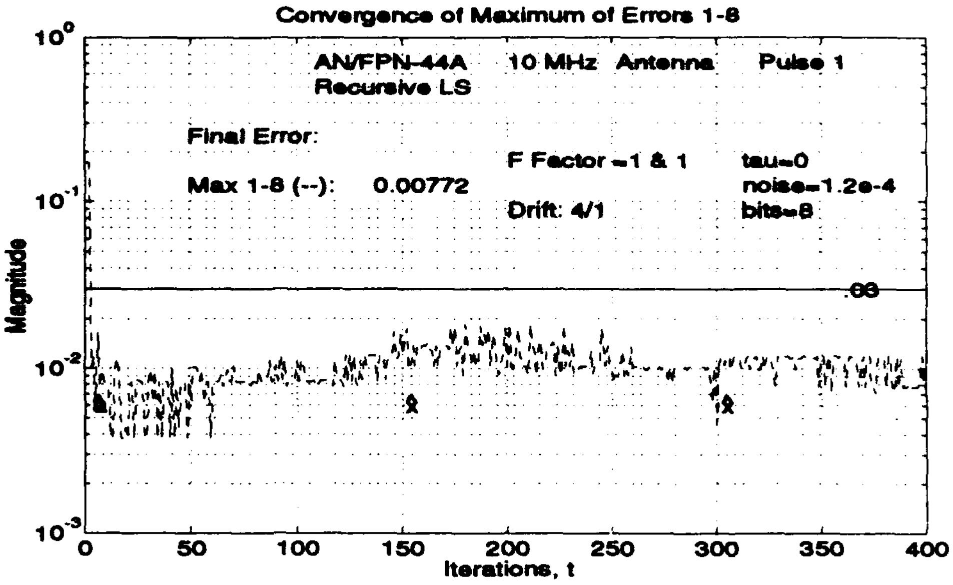

Figure 5.12: Maximum error of peaks 1-8 using the RLS control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

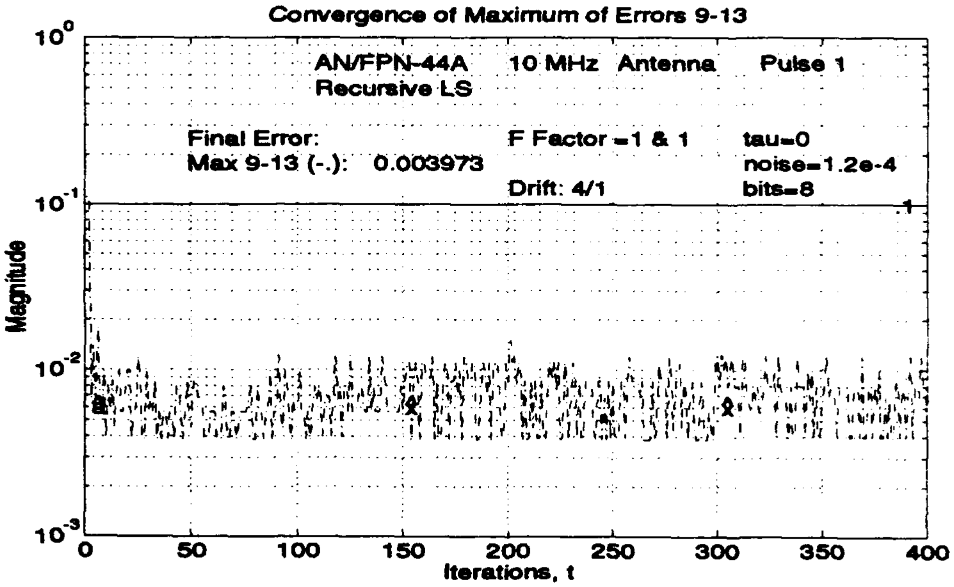

Figure 5.13: Maximum error of peaks 9-13 using the RLS control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

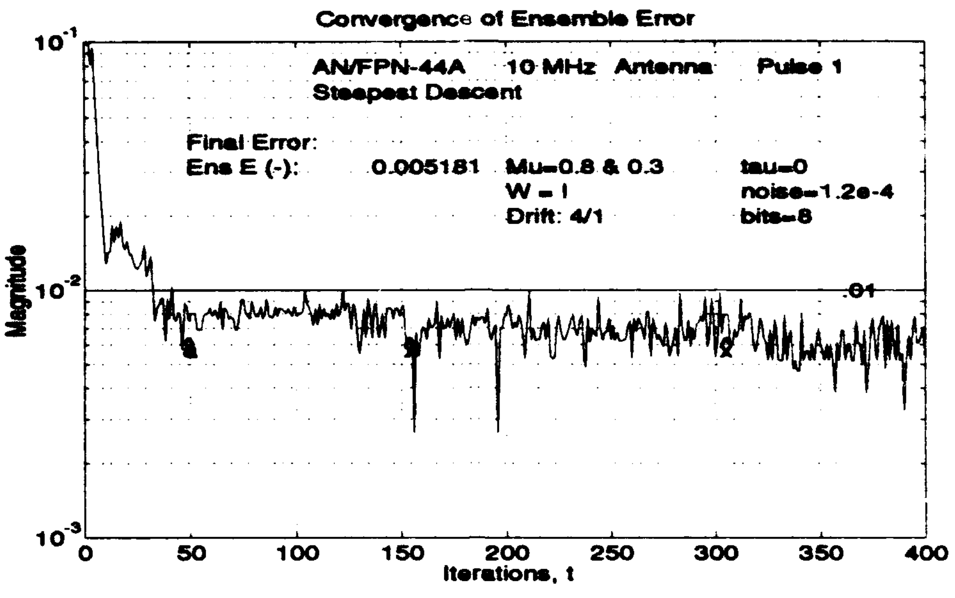

Figure 5.14: Ensemble error using the steepest descent control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

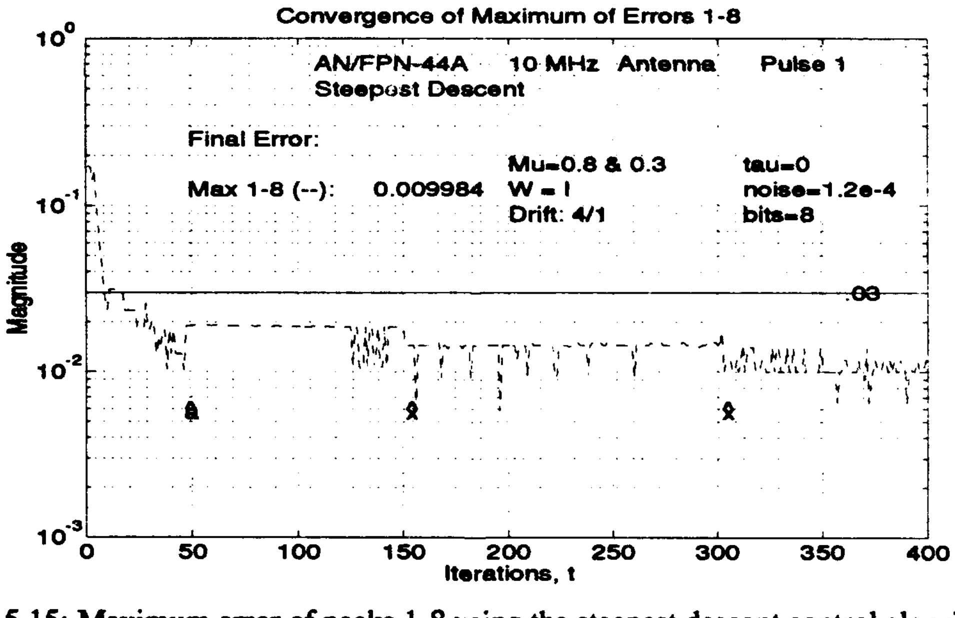

Figure 5.15: Maximum error of peaks 1-8 using the steepest descent control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

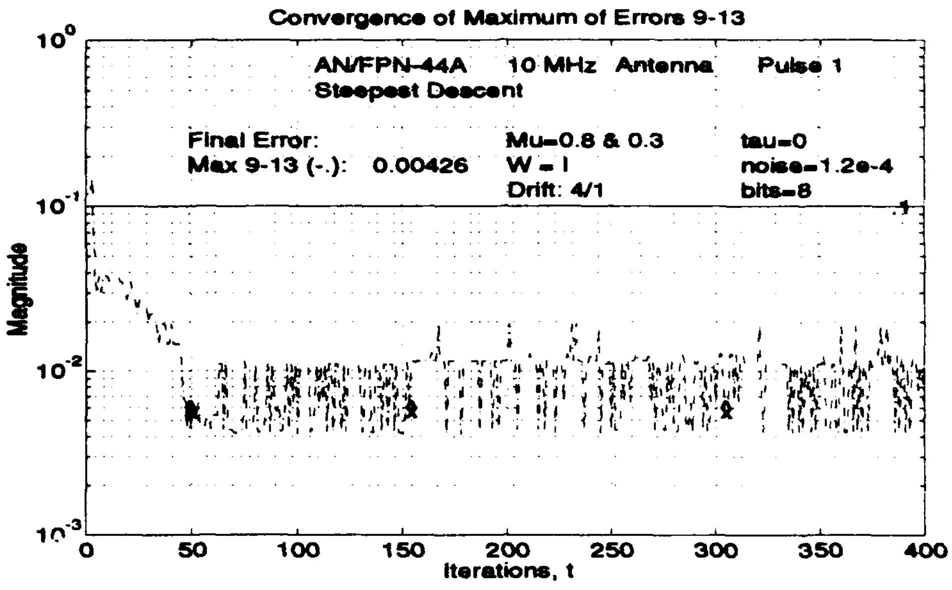

Figure 5.16: Maximum error of peaks 9-13 using the steepest descent control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

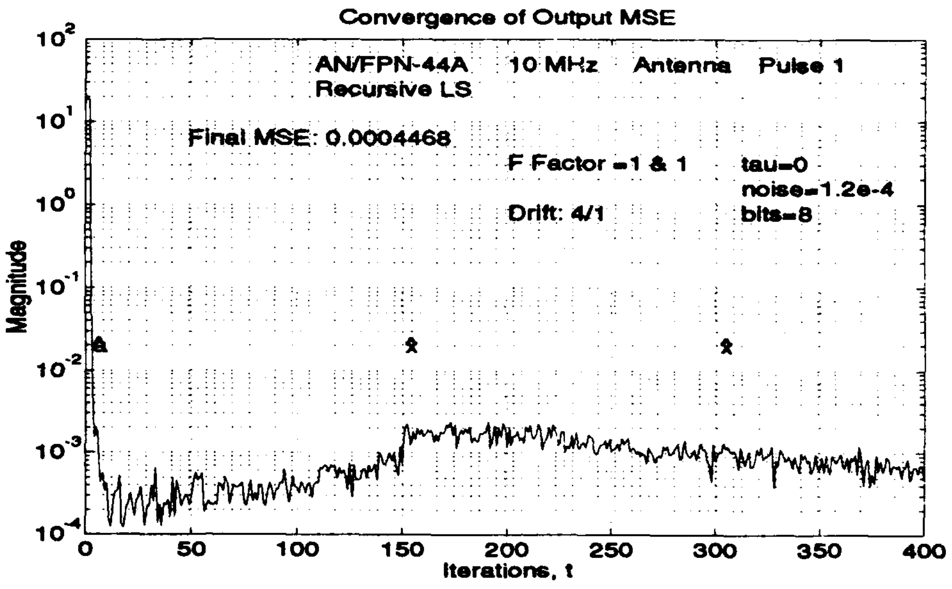

Figure 5.17: MSE of peaks 1-16 using the RLS control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

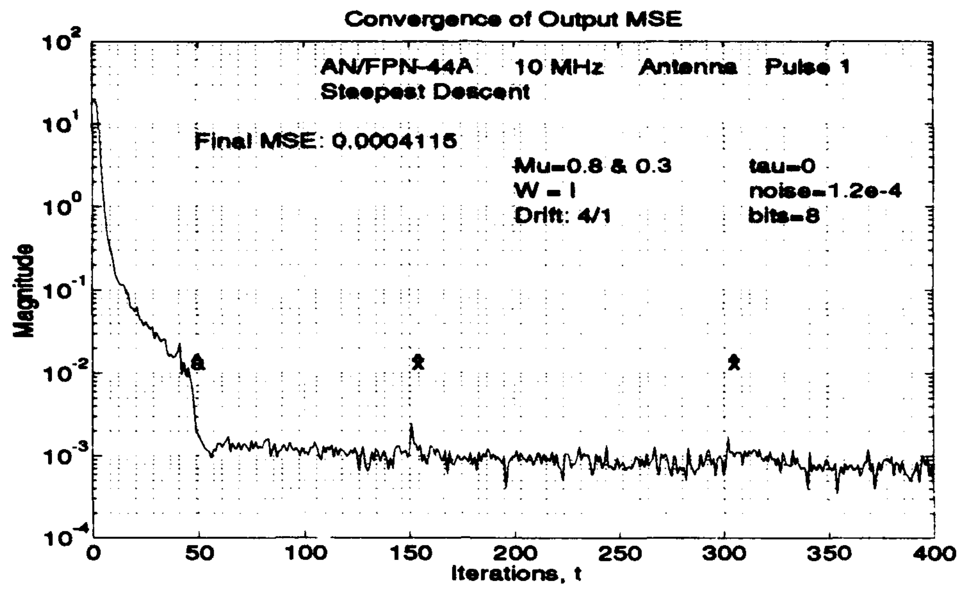

Figure 5.18: MSE of peaks 1-16 using the steepest descent control algorithm (8 bits, noise variance of 1.2e-4). Transmitter switches at iteration 150 & 300 and drift occurs every 4 iterations.

TABLE 5.4: Zero-crossings Errors (ns) After 400 Iterations (8 Bits, Noise Variance of 1.2e-4)

Cont. Alg.	1	2	3	4	54	6	7	8	9	10	11	12
RLS	0	-306.8	-48.8	0	0	0	0	0	0	0	0	0
St Des	0	-110.5	-49.9	0	0	0	0	0	0	0	0	0

TABLE 5.5: RF Pulse Peak Tolerances After 400 Iterations (8 Bits, Noise Variance of 1.2e-4)

Control Alg.	MSE out	Ens err	MaxE 1-8	MaxE 9-13
RLS	0.0004	0.0044	0.0077	0.0040
Steepest Descent	0.0004	0.0052	0.0100	0.0043

TABLE 5.6.a: Normalized RF Peak Values with RLS Control Algorithm (8 Bits, Noise Variance of 1.2e-4)

Peak	Ideal	Average	Last iter	Diff (3-2)	Diff (4-2)
1	0.0157	0.0240	0.0252	0.0083	0.0096
2	-0.0833	-0.0809	-0.0840	0.0025	-0.0007
3	0.1901	0.1952	0.2017	0.0050	0.0116
4	-0.3158	-0.3212	-0.3277	-0.0054	-0.0119
5	0.4454	0.4458	0.4454	0.0004	-0.0001
6	-0.5696	-0.5700	-0.5714	-0.0004	-0.0018
7	0.6813	0.6840	0.6891	0.0027	0.0077
8	-0.7771	-0.7776	-0.7815	-0.0005	-0.0044
9	0.8556	0.8554	0.8571	-0.0003	0.0015
10	-0.9164	-0.9184	-0.9160	-0.0021	0.0004
11	0.9598	0.9607	0.9580	0.0009	-0.0018
12	-0.9872	-0.9852	-0.9832	0.0019	0.0040
13	1.0000	1.0000	1.0000	0	0
14	-1.0001	-1.0019	-1.0000	-0.0018	0.0001
15	0.9892	0.9875	0.9832	-0.0017	-0.0060
16	-0.9692	-0.9705	-0.9748	-0.0014	-0.0056

TABLE 5.6.b: Normalized RF Peak Values with Steepest Descent Control Alg. (8 Bits, Noise Variance of 1.2e-4)

Peak	Ideal	Average	After 400 iter	Diff (3-2)	Diff (4-2)
1	0.0157	0.0135	0.0169	-0.0021	0.0013
2	-0.0833	-0.0699	-0.0763	0.0134	0.0071
3	0.1901	0.1870	0.1864	-0.0031	-0.0037
4	-0.3158	-0.3205	-0.3136	-0.0047	0.0022
5	0.4454	0.4464	0.4492	0.0010	0.0037
6	-0.5696	-0.5628	-0.5678	0.0068	0.0018
7	0.6813	0.6752	0.6780	-0.0061	-0.0034
8	-0.7771	-0.7761	-0.7712	0.0011	0.0059
9	0.8556	0.8548	0.8475	-0.0008	-0.0082
10	-0.9164	-0.9104	-0.9068	0.0059	0.0096
11	0.9598	0.9517	0.9492	-0.0082	-0.0107
12	-0.9872	-0.9832	-0.9746	0.0040	0.0126
13	1.0000	1.0000	1.0000	0	0
14	-1.0001	-0.9950	-0.9831	0.0051	0.0170
15	0.9892	0.9763	0.9746	-0.0129	-0.0146
16	-0.9692	-0.9706	-0.9576	-0.0015	0.0115

Additional results on the MIMO RLS control algorithm are included in Appendix A. In obtaining these results, the RLS algorithm is tested with the TDW and RF sampled with eight bit resolution and additive white noise of variance 9×10^{-4}. This is the same operating environment that is used to test the steepest descent control algorithm in Ref. 3: p. 121; the results obtained in Appendix A can be directly compared to those in Ref. 3.

In Appendix B, the MIMO RLS control algorithm is derived using a MIMO model with memory that uses past TDW parameter vectors to update the present TDW vector. The control algorithm with memory does not perform as well as the MIMO RLS control algorithm without memory. The algorithm with memory is tested under ideal conditions only; results are found in Appendix B.

C. PEAK AND ZERO SAMPLING

The control algorithm using only the RF peak amplitudes of the first 16 half-cycles does not take into account the error in the desired locations of the zero-crossings. Letting the MIMO RLS controller, under ideal conditions described in section B.1 of Chapter V, run for 1000 iterations results in the RF pulse peaks converging to the ideal peaks with an error of approximately $\pm 1 \times 10^{-4}$ in each peak. Even with the best case, the second and third zero-crossings are still out of tolerance. By expanding the vector parameters that represent the RF pulse to include peak amplitudes and the sample values at the desired zero-crossing locations, control over the zero-crossings is obtained.

The x vector containing the TDW parameters remains unchanged with the 16 half-cycle peak amplitudes. The y vector is expanded to 32 parameters. The odd values (1, 3,...,31) are the RF pulse peaks, and the even values (2, 4,..., 32) are the desired zero crossings. The desired zero crossing samples of the RF pulse are obtained by first locking onto the standard zero-crossing (SZC) and then sampling at multiples of 5μs for a total of 25μs before and 50μs after the SZC. The sampling frequency is 10MHz which corresponds to a sampling interval of 100ns.

The desired zero-crossing tolerances are less than ±100ns after the third zero-crossing of the pulse; therefore, interpolation is performed to increase the accuracy in sampling the desired zero-crossing locations. The following procedure is used for interpolation. Samples are found on either side of the SZC and the SZC location is estimated by linear interpolation between these points. The increase in accuracy of the SZC location is used to better approximate the desired locations of the remaining zero-crossings. Linear interpolation is performed based on the assumption that the RF pulse has a linear slope at crossing points.

A diagonal weighting matrix, W, is used in the TDW update (equation 4.16) to weigh the RF parameter errors as desired. Particularly elements 4 and 6 of the RF error vector required a higher weighting than the others; their weights were chosen to be 15 and 10, respectively. Positions 4 and 6 of the RF error vector are the second and third zero-crossings.

The inverse correlation matrix P_n in the control algorithm is re-initialized every 50 iterations. When sampling both peaks and zero-crossings, P_n is re-initialized as

$$P_n = P_{n-1} + \varphi I,$$

where φ is in the range of 30 to 50, which is smaller than φ in the case of peak sampling only.

The most impressive results are obtained when shaping the RF pulse with samples of the RF pulse peaks and zero-crossings. Under ideal conditions, all zero-crossings are in tolerance after 110 iterations of control, and the RF half-cycle peak amplitudes converge to below tolerance values after 15 iterations. The convergence of the three measures of pulse peak error is shown in Figure 5.19. Table 5.8 lists the RF pulse peak tolerances and Table 5.9 lists the individual RF pulse peak amplitude errors after 400 iterations.

Figure 5.20 illustrates the convergence of the MSE of the RF pulse peaks 1-16. The convergence of the MSE of the RF pulse peaks 1-16, after 400 iterations, is observed to be higher compared with the peak sampling case. This should be expected when increasing the number of parameters required to form a least squares fit.

The second zero-crossing is most frequently out of tolerance. Figure 5.21 illustrates the second zero-crossing initially deviating 225ns from the desired location. After approximately 110 iterations of control, the second zero-crossing falls within the acceptable tolerance of 100ns.

Figure 5.22 and 5.23 show plots of the TDW and RF pulse respectively, obtained after 400 iterations. The second half-cycle of the TDW in Figure 5.22 has a peak amplitude close to zero. This is unusual; however, the resulting RF pulse (synthetic RF pulse) in Figure 5.23 is very similar to the ideal pulse.

The performance of the RLS algorithm has degraded when tested under non-ideal conditions of additive noise and eight bit digitization of the RF waveform. The discussion and results reported here for the MIMO RLS control algorithm using pulse peaks and zero-crossings do not encompass all possible scenarios of the transmitter operation; these results may be considered preliminary. Further research is necessary to explore the strengths and weaknesses of this approach.

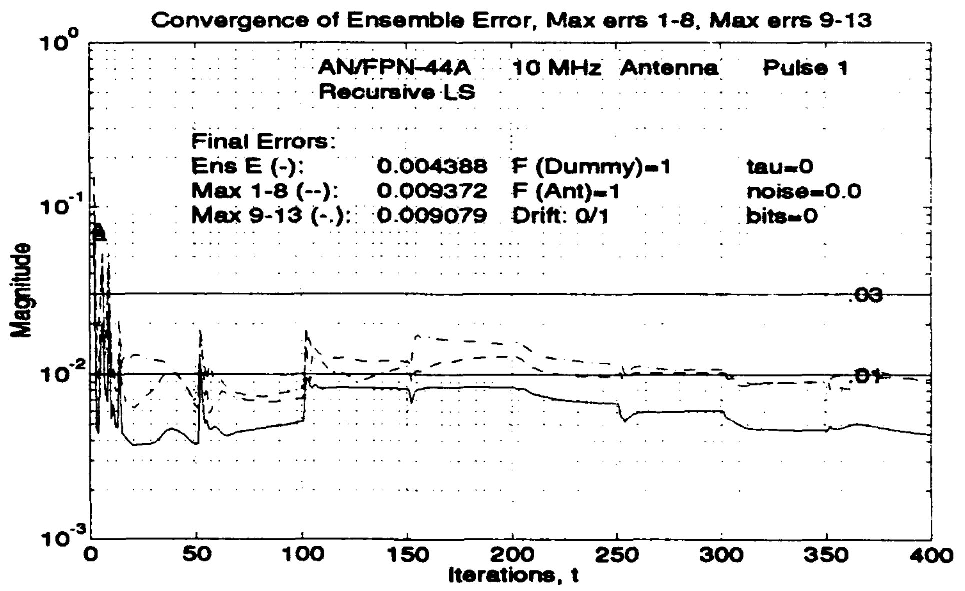

Figure 5.19: Peak amplitude tolerances using the RLS control algorithm (machine precision, no noise) when sampling peaks and zero-crossings.

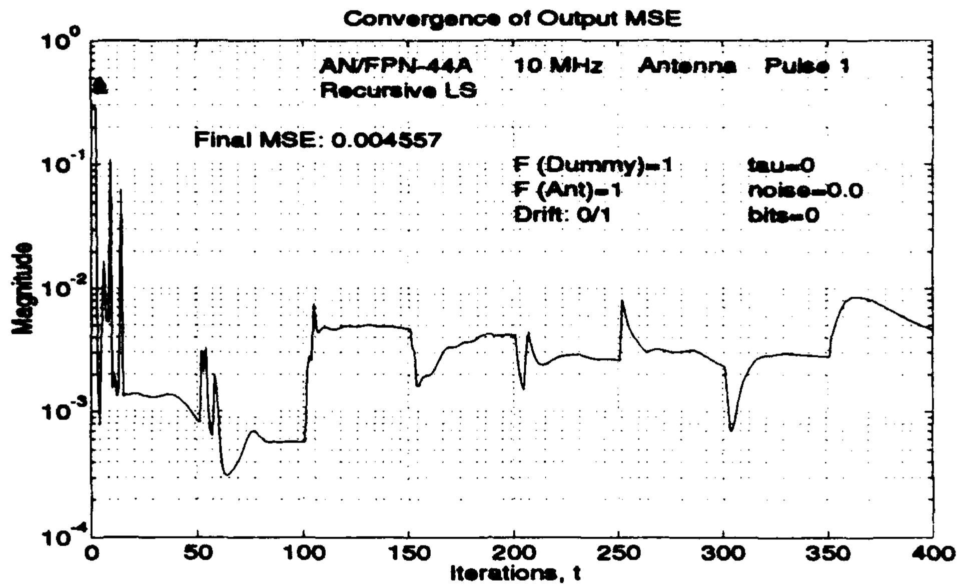

Figure 5.20: MSE of RF peaks 1-16 using the RLS control algorithm (machine precision, no noise) when sampling peaks and zero-crossings.

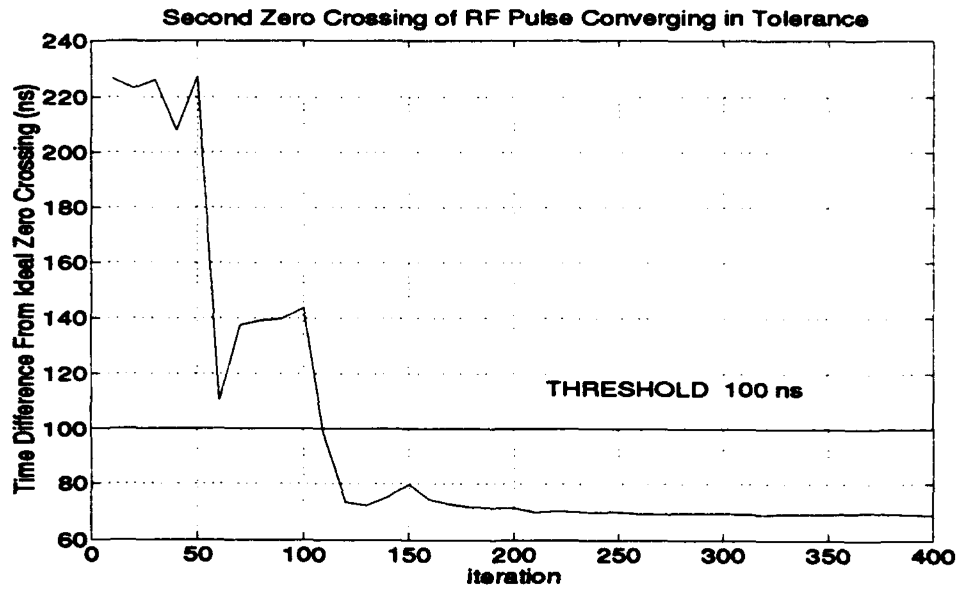

Figure 5.21: Second zero-crossing error (ns) using the RLS control algorithm (machine precision, no noise) when sampling peaks and zero-crossings.

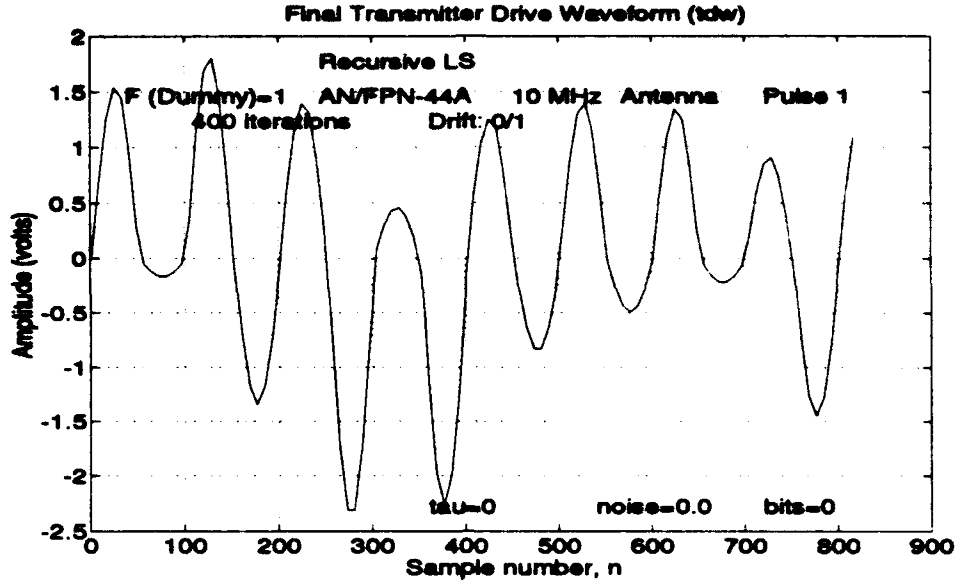

Figure 5.22: TDW produced after 400 iterations using the RLS control algorithm (machine precision, no noise) when sampling peaks and zero-crossings.

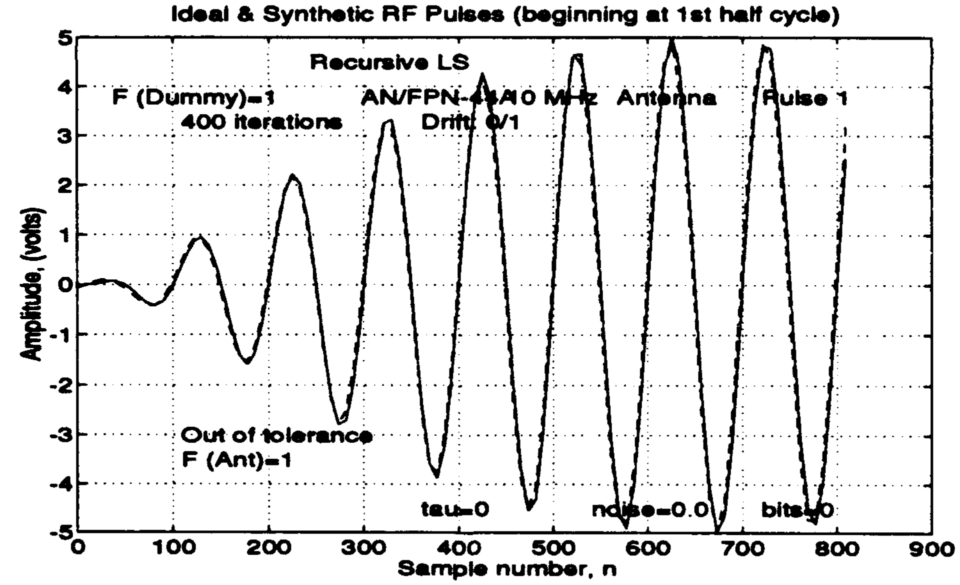

Figure 5.23: Converged synthetic (dash) RF pulse with ideal (solid) after 400 iterations using the RLS control algorithm (machine precision, no noise) when sampling peaks and zero-crossings.

TABLE 5.7: RF Pulse Peak Tolerances After 400 Iterations with RLS Control Algorithm (Machine Precision, No Noise, Peaks and Zero-crossings)

MSE out	Ens err	MaxE 1-8	MaxE 9-13
0.0011	0.0071	0.0111	0.0117

TABLE 5.8: Normalized RF Peak Values with RLS Control Algorithm (Machine Precision, No Noise, Peaks and Zero-crossings)

Peak	Ideal	Average	After 400 iter.	Diff (1-3)	Diff (2-3)
1	0.0157	0.0210	0.0203	0.0054	0.0046
2	-0.0833	-0.0901	-0.0921	-0.0067	-0.0088
3	0.1901	0.1973	0.1976	0.0072	0.0075
4	-0.3158	-0.3122	-0.3097	0.0036	0.0061
5	0.4454	0.4390	0.4379	-0.0064	-0.0076
6	-0.5696	-0.5606	-0.5585	0.0090	0.0111
7	0.6813	0.6844	0.6843	0.0030	0.0030
8	-0.7771	-0.7803	-0.7822	-0.0032	-0.0050
9	0.8556	0.8486	0.8487	-0.0071	-0.0069
10	-0.9164	-0.9195	-0.9209	-0.0031	-0.0046
11	0.9598	0.9695	0.9715	0.0097	0.0117
12	-0.9872	-0.9900	-0.9915	-0.0028	-0.0043
13	1.0000	1.0000	1.0000	0	0
14	-1.0001	-1.0034	-1.0015	-0.0033	-0.0014
15	0.9892	0.9958	0.9936	0.0066	0.0044
16	-0.9692	-0.9787	-0.9735	-0.0096	-0.0043

D. RLS CONTROL ADVANTAGES AND LIMITATIONS

The MIMO RLS controller is considered to operate in a slow time-varying environment at a given operating point. The RLS algorithm in this environment is typically an order of magnitude faster in convergence than the steepest descent algorithm [Ref. 8: p 491-501]. This was observed in the ideal simulation environment.

The greatest advantage with the MIMO RLS control algorithm is the use of a time-varying reference model. The MIMO RLS control algorithm uses a MIMO RLS model which allows the reference model to adapt to changes and time variations in the transmitter. On the other hand, the steepest descent algorithm uses a fixed impulse response matrix when modeling the transmitter.

The MIMO RLS control algorithm is subject to the stalling phenomenon. The algorithm's inverse correlation matrix, P_n, rapidly decreases numerically until the algorithm stalls. The inverse correlation matrix is continually reset every 50 iterations to overcome this problem

Another limitation of the MIMO RLS control algorithm is its computational complexity. It requires MIMO RLS estimate of the reference model and optimal TDW half-cycle peak amplitudes at each iteration. To decrease the number of computations, the recursive least squares estimate of the MIMO model could be carried out every few iterations rather than at every iteration once the algorithm reaches steady state TDW parameters.

VI. CONCLUSIONS

A. CONCLUSIONS

As part of the US Coast Guard's effort to redesign and modernize the AN/FPN-44A tube type transmitter with automatic Loran pulse shaping and control, this thesis developed a MIMO RLS algorithm. This control algorithm shapes the transmitted RF pulse to match an ideal pulse by producing a near optimal TDW. The MIMO RLS control algorithm uses a MIMO RLS estimate of the transmitter as a reference model. The RLS algorithm is implemented on a simulated transmitter, and the RF pulse peak tolerances and zero-crossing errors are studied. The performance of the RLS algorithm is compared to another method of control, the steepest descent algorithm.

When using the first 16 RF pulse peaks to represent the transmitter output, the MIMO RLS control algorithm was able to shape the pulse, but it was unable to bring the zero-crossings into tolerance. Using these 16 output parameters, the MIMO RLS control algorithm proved to have faster convergence and better ability to match the RF pulse to the ideal over the method of steepest descent. The use of a MIMO RLS model of the transmitter in the RLS controller was a significant advantage. The MIMO model adapts to the non-LTI characteristics in the transmitter.

By representing the RF pulse with parameters of the first 16 pulse peaks and samples of the first 16 desired zero-crossing locations, the MIMO RLS control algorithm was able to decrease the half-cycle peak amplitude errors below their tolerance levels and bring the zero-crossings to within specification. When the RF pulse parameters are expanded from 16 to 32, the least squares formulation introduces more error into the RF pulse peak amplitudes; however, all pulse specifications are still met.

In the non-ideal simulation, where the simulated transmitter's input and output are corrupted with white noise and digitized with eight bits, the performance of the control algorithms degraded. The RF pulse peaks are not in tolerance for all control iterations and the second and third zero-crossings do not meet specifications.

B. RECOMMENDATIONS FOR FURTHER STUDY

To improve the controller's performance in shaping the RF pulse, noise reduction is necessary. The control algorithms can not decrease the error in the RF pulse parameters below the level of the noise. Filtering the noise prior to control might improve the performance of the control algorithms.

Studying the characteristics of the RF pulse's quantization noise has lead to the conclusion that sampling the RF pulse with eight bits or more produces noise that is essentially white. By averaging several successively digitized RF pulses, of the same phase code, the quantization noise can be reduced. A study of the system noise will also be necessary to gain any advantage from waveform averaging.

If it is not possible to reduce the system noise to meet the specifications, a new set of acceptable performance figures could be suggested. This may included redeveloping the error tolerances for the RF peaks and zero-crossings.

Further research on controlling the RF pulse with peak and zero-crossing samples needs to be conducted. The adjustment (i.e., shifting the zero-crossings) accomplished by the RLS control algorithm requires further testing and analysis. Better control may be established by an expanded RF pulse parameter vector to include samples at the quadrature locations.

When expanding the parameters that represent the signals or using memory to match the RF pulse to the ideal, the time required to update the TDW increases; the computational expense of the RLS algorithm may be undesirable. To increase the speed of the RLS calculations, the fast RLS algorithm may be an acceptable substitute [Ref. 8].

APPENDIX A

MIMO RLS CONTROL IN A NON-IDEAL SIMULATION

In this appendix, results obtained using the MIMO RLS control algorithm under non-ideal conditions are included. In a non-ideal simulation, the RF pulse and TDW are subject to quantization and additive white noise. The signal to noise ratio (SNR) of the RF pulse and TDW is approximately 67.48dB and 56.29dB. To obtain these SNR levels, the signals are sampled with an eight-bit resolution, and white noise with a variance of $9.26\text{x}10^{-4}$ is added.

These results were produced to compare the performance of the MIMO RLS algorithm with that of the steepest descent algorithm as reported in Ref. 3: p. 120. The steepest descent algorithm was implemented under the same non-ideal conditions as those considered here.

The following is a summary of the operating conditions for the results shown in Figures A.1-6. The sampling frequency was 10MHz. The forgetting factor in the RLS algorithm was unity. The ECD (τ_e) was set to zero. The additive noise has a variance of $9.26\text{x}10^{-4}$. An eight-bit A/D converter was used to quantize the signals.

Figure A.1 is a plot of the MSE convergence of the RF pulse peaks 1-16 to their ideal values. The steady state MSE, shown in Figure A.1, using the RLS algorithm is a close resemblance of the steady state MSE using the steepest descent method under the same operating conditions [Ref. 3, p. 121, Figure 5.12.a]. The noise power is observed as the floor of the MSE convergence in both sets of results.

The RF peak amplitudes are in tolerance for 92% of the iterations examined. In Figure A.2, the ensemble error is occasionally out of tolerance. The other measures of peak amplitude error are well within tolerance limits, as observed in Figures A.3 and A.4.

The second and third zero-crossings listed in Table A.1 are out of tolerance. Even in the ideal environment, the second and third zero-crossings do not meet tolerance specifications. These zero-crossings are most frequently out of tolerance when controlling the 44A Loran-C pulse, see Chapter V, Section C, for a method of control of the zero-crossings.

After 400 iterations of control, the RF pulse peak errors are listed in Table A.2, and the difference of the pulse peak amplitudes from their ideal values are listed in Table A.3. The converged TDW is plotted in Figure A.5. The corresponding RF pulse and the ideal pulse are plotted in Figure A.6.

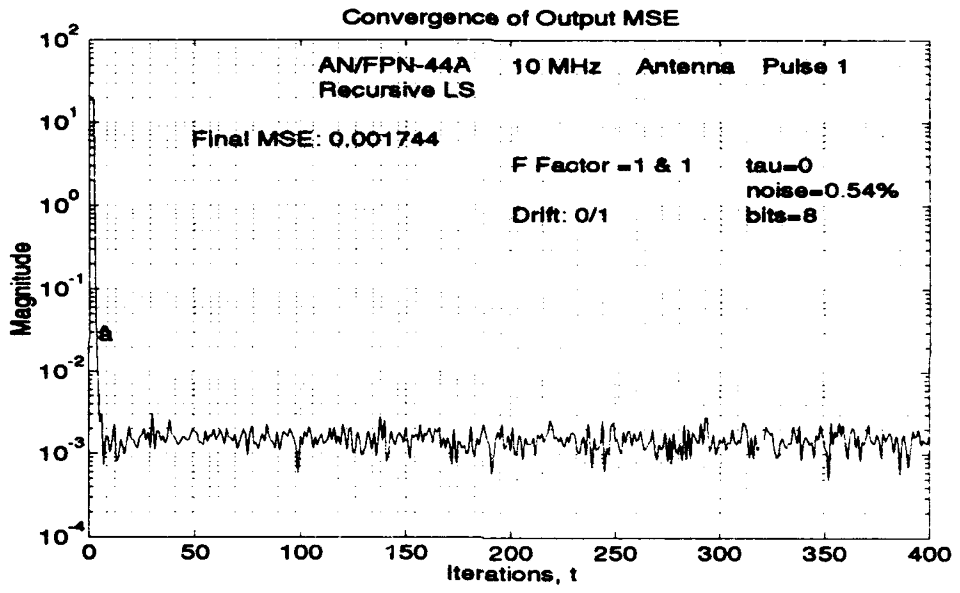

Figure A.1: MSE of peaks 1-16 using the RLS control algorithm (8 bits, noise variance of 9.3e-4).

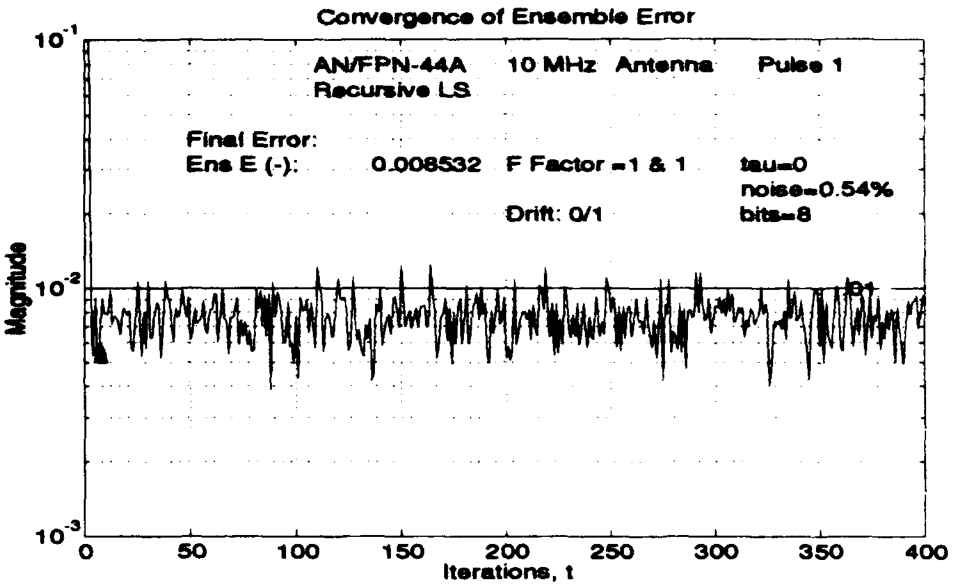

Figure A.2: Ensemble error using the RLS control algorithm (8 bits, noise variance of 9.3e-4).

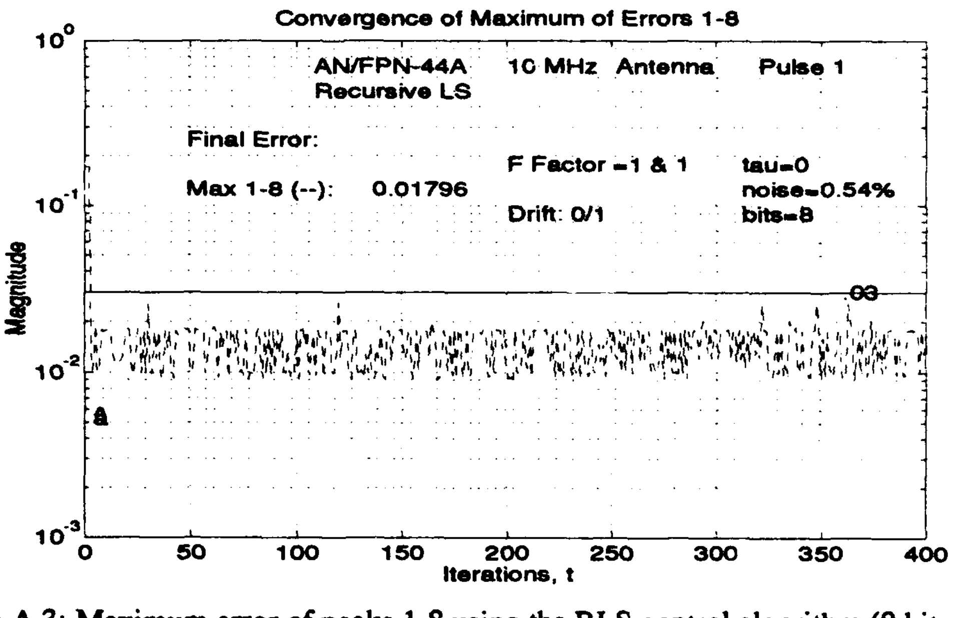

Figure A.3: Maximum error of peaks 1-8 using the RLS control algorithm (8 bits, noise variance of 9.3e-4).

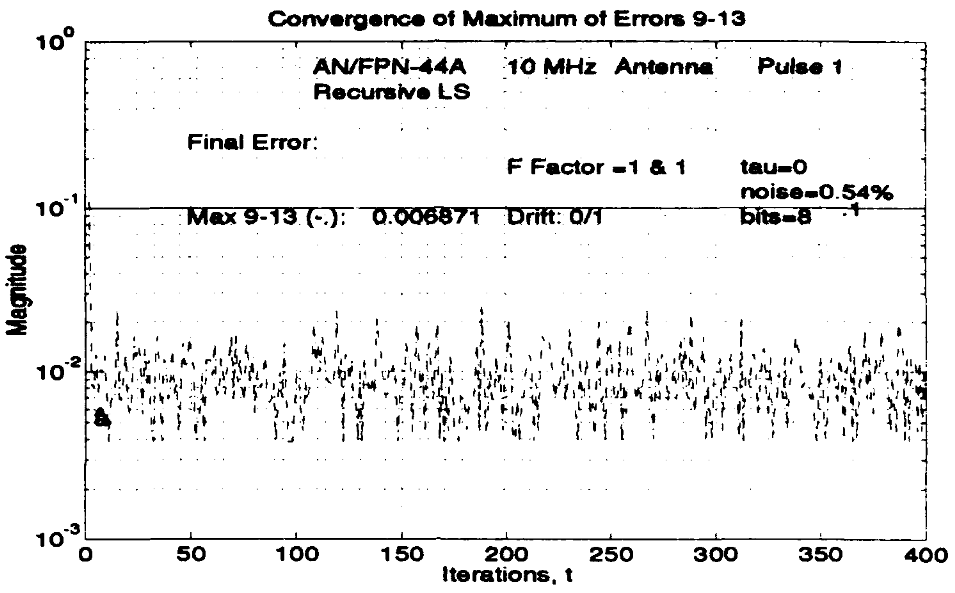

Figure A.4: Maximum error of peaks 9-13 using the RLS control algorithm (8 bits, noise variance of 9.3e-4).

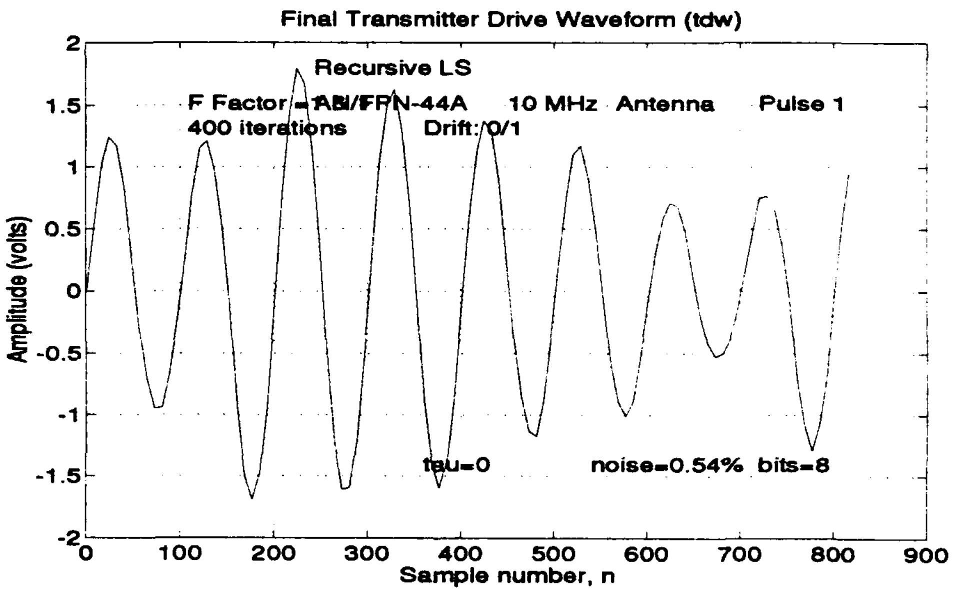

Figure A.5: TDW produced after 400 iterations using the RLS control algorithm (8 bits, noise variance of 9.3e-4).

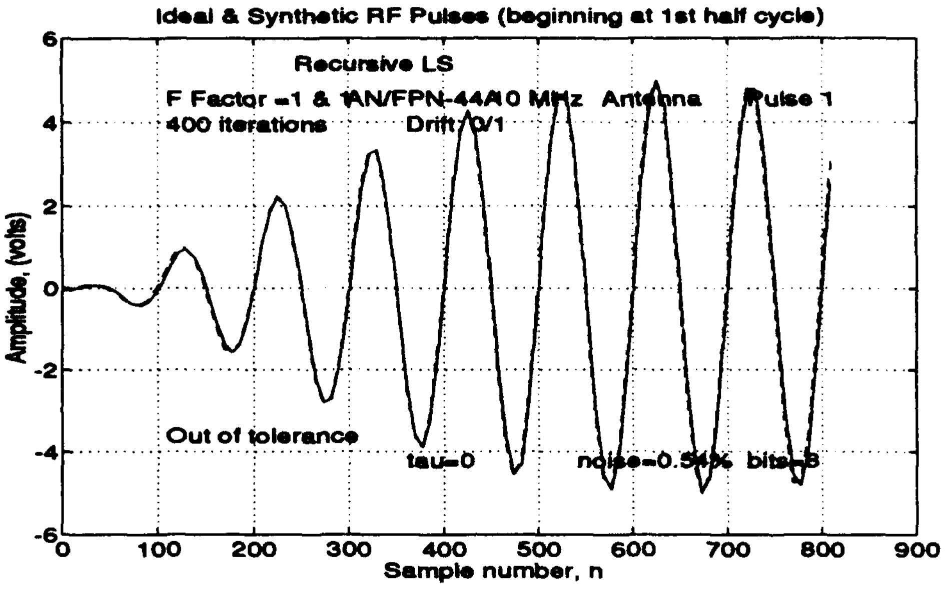

Figure A.6: Converged synthetic (dash) RF pulse with ideal (solid) after 400 iterations using the RLS control algorithm (8 bits, noise variance of 9.3e-4).

TABLE A.1: Zero-crossings Errors (ns) After 400 Iterations with RLS Control Algorithm (8 Bits, Noise Variance of 9.3e-4)

0	-231.3	-6.9	0	0	0	0	0	0	0	0	0

TABLE A.2: RF Pulse Peak Tolerances After 400 Iterations with RLS Control Algorithm (8 Bits, Noise Variance of 9.3e-4)

MSE out	Ens err	MaxE 1-8	MaxE 9-13
0.0017	0.0085	0.0180	0.0069

TABLE A.3: Normalized RF Peak Values with RLS Control Algorithm (8 Bits, Noise Variance of 9.3e-4)

Peak	Ideal	Average	After 400 iter.	Diff (3-2)	Diff (4-2)
1	0.0157	0.0289	0.0336	0.0132	0.0180
2	-0.0833	-0.0853	-0.0840	-0.0020	-0.0007
3	0.1901	0.1969	0.1933	0.0067	0.0032
4	-0.3158	-0.3226	-0.3193	-0.0068	-0.0035
5	0.4454	0.4476	0.4538	0.0022	0.0084
6	-0.5696	-0.5714	-0.5714	-0.0018	-0.0018
7	0.6813	0.6844	0.6807	0.0030	-0.0007
8	-0.7771	-0.7780	-0.7899	-0.0009	-0.0128
9	0.8556	0.8561	0.8487	0.0005	-0.0069
10	-0.9164	-0.9179	-0.9160	-0.0016	0.0004
11	0.9598	0.9603	0.9580	0.0005	-0.0018
12	-0.9872	-0.9850	-0.9832	0.0022	0.0040
13	1.0000	1.0000	1.0000	0	0
14	-1.0001	-1.0018	-1.0000	-0.0017	0.0001
15	0.9892	0.9864	0.9832	-0.0029	-0.0060
16	-0.9692	-0.9701	-0.9748	-0.0010	-0.0056

APPENDIX B

OPTIMAL TDW ESTIMATION WITH MEMORY

In this appendix, we derive the a MIMO RLS control algorithm with memory. This derivation, in section A, presents a recursive least squares procedure similar to the memoryless RLS algorithm in Chapter III, Section A.1 and A.2. In section B of this appendix, results are presented for the MIMO RLS algorithm under ideal conditions.

A. MIMO RLS CONTROL ALGORITHM WITH MEMORY

The MIMO RLS controller uses a reference model A. The reference model A is a MIMO RLS estimate of the transmitter at each control iteration (see Chapter III). The transmitter model output at time i is expressed as

$$\hat{y}_i = A\, x_i$$

where

$$A = \begin{bmatrix} [A_0] & [A_1] & \cdots & [A_N] \end{bmatrix},$$

and

$$x_i = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{n-N} \end{bmatrix}.$$

Column vector x_i is composed of the current TDW parameter vector, x_n, and past TDW parameter vectors, $(x_{n-1}, x_{n-2}, \ldots, x_{n-N})$. Each submatrix of A has a column width equal to the number of parameters in vector x_n (TDW parameter vector), and the number of rows are equal to the length of $\hat{y}_i$ (RF parameter vector). N is the order of the system [Ref. 14].

1. The Least Squares Method

The vector of errors is expressed as

$$e_i = y_{opt} - \hat{y}_i,$$

where y_{opt} is a vector of ideal RF pulse parameters. The sum of squared errors is

$$J = \sum_{i=0}^{n} (y_{opt} - A_i x_i)^T (y_{opt} - A_i x_i),$$

or is expanded to become

$$J = \sum_{i=0}^{n} (y_{opt}^T y_{opt} - x_i^T A_i^T y_{opt} - y_{opt}^T A_i x_i - x_i^T A_i^T A_i x_i). \qquad \text{(B.1)}$$

Substituting $A_i x_i = \sum_{j=0}^{N} A_j^{(i)} x_{n-j}^{(i)} = \sum_{j=0}^{N} A_j x_{n-j}$ in equation B.1, where i is the time index;

we then have

$$J = \sum_{i=0}^{n} y_{opt}^T y_{opt} - \sum_{i=0}^{n} \left\{ \sum_{j=0}^{N} x_{n-j}^T A_j^T y_{opt} \right\}$$

$$- \sum_{i=0}^{n} y_{opt}^T \sum_{k=0}^{N} A_k x_{n-k} - \sum_{i=0}^{n} \left\{ \sum_{j=0}^{N} \left\{ \sum_{k=0}^{N} x_{n-j}^T A_j^T A_k x_{n-k} \right\} \right\}.$$

We assume x_n is independent of time index i. Setting the partial derivative of J with respect to vector x_n equal to zero

$$\frac{\partial J}{\partial x_n} = \sum_{i=0}^{n} \left\{ -2A_0^T y_{opt} + \sum_{k=0}^{N} 2A_0^T A_k x_{n-k} \right\} = 0, \qquad \text{(B.2)}$$

switching the index of summation of i and k and rearranging terms in equation B.2 yields

$$\sum_{i=0}^{n} (A_0^T A_0) x_n = \sum_{i=0}^{n} A_0^T y_{opt} - \sum_{k=1}^{N} \left\{ \sum_{i=0}^{n} A_0^T A_k x_{n-k} \right\}. \qquad \text{(B.3)}$$

Let the autocorrelation matrix be

$$\Phi = \sum_{i=0}^{n} A_0^T A_0, \qquad \text{(B.4)}$$

and the cross-correlation vector be

$$\gamma_n = \sum_{i=0}^{n} A_0^T y_{opt} - \sum_{k=1}^{N} \left\{ \sum_{i=0}^{n} A_0^T A_k x_{n-k} \right\}. \quad \text{(B.5)}$$

Substituting equations B.4 and B.5 into equation B.3, produces a simplified matrix form

$$\Phi_n x_n = \gamma_n, \quad \text{(B.6)}$$

and letting $P_n^{-1} = \Phi_n$ yields

$$P_n^{-1} x_n = \gamma_n. \quad \text{(B.7)}$$

The least squares solution for x_n from equations B.6 and B.7 is

$$x_n = \Phi_n^{-1} \gamma_n = P_n \gamma_n.$$

2. The Recursive Least Squares Method

For the recursive update of x_n, we express equation B.7 at time index (n+1)

$$P_{n+1}^{-1} x_{n+1} = \gamma_n + A_0^{(n+1)^T} y_{opt} - \sum_{k=1}^{N} A_0^{(n+1)^T} A_k^{(n+1)} x_{n-k}. \quad \text{(B.8)}$$

Substituting equation B.7 for γ_n in equation B.8 yields

$$P_{n+1}^{-1} x_{n+1} = P_n^{-1} x_n + A_0^{(n+1)^T} y_{opt} - \sum_{k=1}^{N} (A_0^{(n+1)^T} A_k^{n+1}) x_{n-k}. \quad \text{(B.9)}$$

Adding $(A_0^{(n+1)^T} A_0^{(n+1)} x_n) - (A_0^{(n+1)^T} A_0^{(n+1)} x_n)$ on the right side of equation B.9 and grouping terms such that

$$P_n^{-1} x_n + (A_0^{(n+1)^T} A_0^{(n+1)}) x_n = P_{n+1}^{-1} x_n,$$

and

$$-(A_0^{(n+1)^T} A_0^{(n+1)}) x_n - \sum_{k=1}^{N} (A_0^{(n+1)^T} A_k^{(n+1)}) x_{n-k} = -\sum_{k=0}^{N} (A_0^{(n+1)^T} A_k^{n+1}) x_{n-k},$$

equation B.9 becomes

$$P_{n+1}^{-1}\, x_{n+1} = P_{n+1}^{-1}\, x_n + A_0^{(n+1)^T} y_{opt} - \sum_{k=0}^{N} A_0^{(n+1)^T} A_k^{n+1} x_{n-k}. \qquad (B.10)$$

Multiplying both sides of equation B.10 by P_{n+1}, and expressing

$$\left\{A_0^{(n+1)^T} y_{opt} - \sum_{k=0}^{N} (A_0^{(n+1)^T} A_k^{(n+1)}) x_{n-k}\right\} \text{ as } \left\{A_0^{(n+1)^T}\left(y_{opt} - \sum_{k=0}^{N} A_k^{(n+1)} x_{n-k}\right)\right\},$$

yields the final recursive equation for x_{n+1}

$$x_{n+1} = x_n + P_{n+1}\, A_0^{(n+1)^T}\left(y_{opt} - \sum_{k=0}^{N} A_k^{(n+1)} x_{n-k}\right), \qquad (B.11)$$

where $y_{opt} - \sum_{k=0}^{N} A_k^{(n+1)} x_{n-k} = e_{(n+1/n)}$.

For the recursive solution of P_{n+1}, let

$$P_{n+1}^{-1} = \lambda\, P_n^{-1} + A_0^{(n+1)^T} A_0^{(n+1)}. \qquad (B.12)$$

Following on the lines of equation 4.17 - 4.20 the recursive equation for P_{n+1} becomes

$$P_{n+1} = \frac{1}{\lambda} P_n - \frac{1}{\lambda} P_n\, A_0^{(n+1)^T}\left[I + A_0^{(n+1)} \frac{1}{\lambda} P_n A_0^{(n+1)^T}\right]^{-1} A_0^{(n+1)}\, \frac{1}{\lambda} P_n.$$

B. RESULTS OF MIMO RLS CONTROL ALGORITHM WITH MEMORY

The MIMO RLS control algorithm is tested under ideal conditions, as in Chapter V, Section B.1. The signal to noise ratio (SNR) of the TDW and RF pulse are approximately 81dB and 99dB, respectively. The 16 parameters that represent the TDW and RF pulse are the first 16 half-cycle peak amplitudes. Let N = 2, so x_i becomes a vector of size 48 x 1, and A becomes a matrix of size 16 x 48. A forgetting factor of $\lambda = 1$ is used.

Figure B.1 shows the three measures of RF pulse peak error (ensemble error, maximum of individual peak errors 1-8 and 9-13). The RF pulse peak errors converge to acceptable levels by iteration 50; however, undesirable perturbations occur after iterations 300 and 350 which drive the pulse errors out of tolerance.

Figure B.2 illustrates the convergence of the MSE of RF pulse peak errors 1-16. When using memory in the MIMO RLS control algorithm, there is no observable improvement in the pulse peak errors or the MSE. The pulse peaks are in tolerance for 96.9% of iterations examined.

Table B.1 provides a listing of the zero-crossings after 400 iterations that exceed tolerance specifications. The second zero-crossing in Table B.1 is out of tolerance while all others are acceptable. The RF pulse peak tolerances are listed in Table B.2 and the individual RF pulse peak errors (1-16) are listed and analyzed in Table B.3.; both after 400 iterations. The pulse peak tolerances were met using the MIMO RLS algorithm, but zero-crossings were not.

Comparing the convergence of the MSE using the RLS algorithm without memory, in Figure 5.2, to the MSE convergence with memory, in Figure B.2, the convergence without memory (past TDW information) performs better. The spurious disturbances observed in the MSE for the RLS algorithm with memory are absent for the RLS without memory.

When white noise with a variance of 1.2e-4 is added to the TDW and RF pulse or the signals are sampled with eight bit resolution, the results are unsatisfactory. The lower SNR accentuates the perturbations and degrades the tracking ability of the MIMO RLS algorithm with memory.

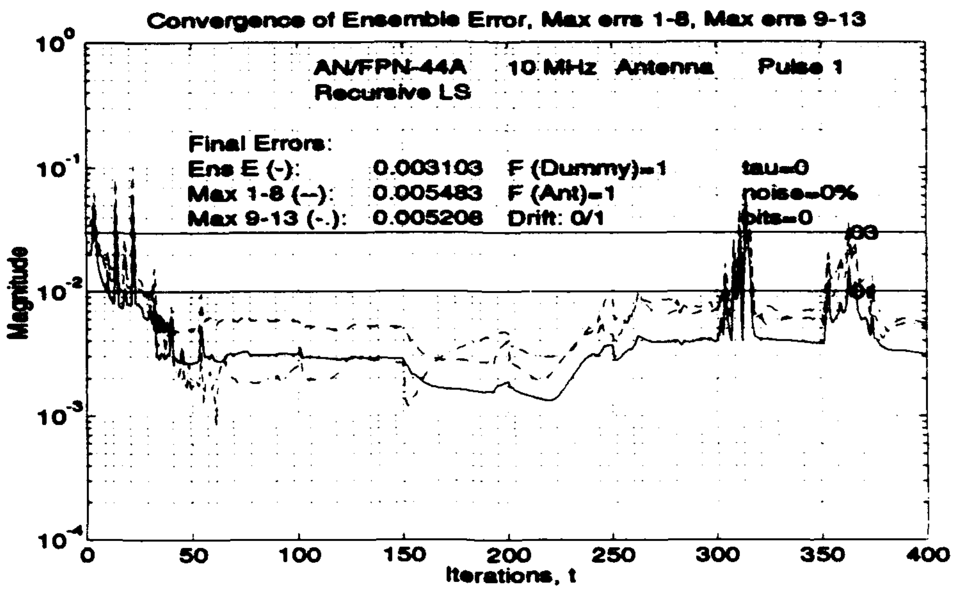

Figure B.1: Peak amplitude tolerances using the RLS control algorithm with N = 2 (machine precision, no noise).

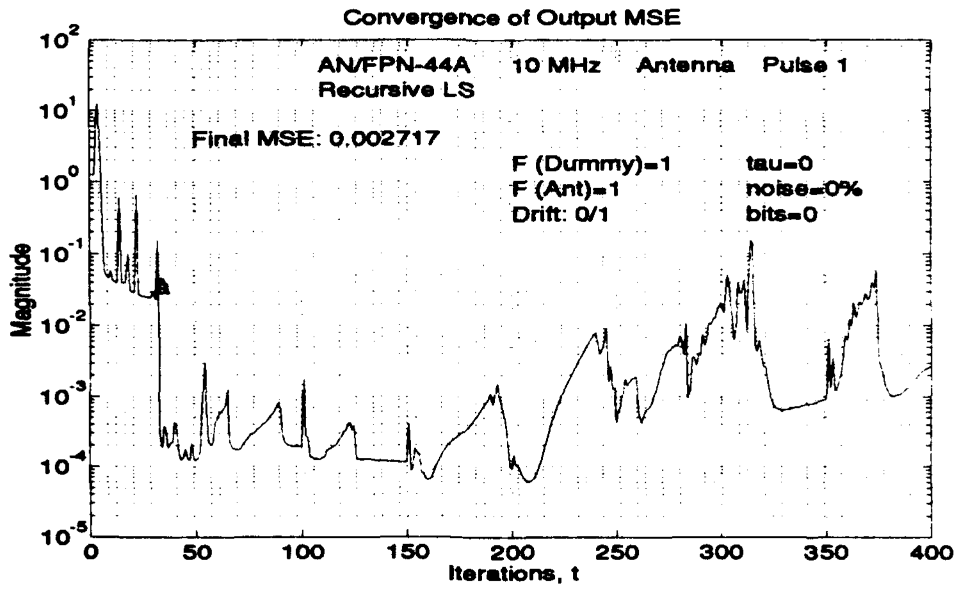

Figure B.2: MSE of RF peaks 1-16 using the RLS control algorithm with N = 2 (machine precision, no noise).

TABLE B.1: Zero-crossings Errors (ns) After 400 Iterations with RLS Control Algorithm where N = 2 (Machine Precision, No Noise)

0	-104.3	0	0	0	0	0	0	0	0	0	0

TABLE B.2: RF Pulse Peak Tolerances After 400 Iterations with RLS Control Algorithm where N = 2 (Machine Precision, No Noise)

MSE out	Ens err	MaxE 1-8	MaxE 9-13
0.0027	0.0031	0.0055	0.0052

TABLE B.3: Normalized RF Peak Values with RLS Control Algorithm where N = 2 (Machine Precision, No Noise)

Peak	Ideal	Average	After 400 iter.	Diff (1-3).	Diff (2-3)
1	0.0157	0.0202	0.0191	0.0045	0.0034
2	-0.0833	-0.0791	-0.0817	0.0043	0.0016
3	0.1901	0.1906	0.1948	0.0005	0.0047
4	-0.3158	-0.3163	-0.3181	-0.0005	-0.0023
5	0.4454	0.4431	0.4438	-0.0024	-0.0016
6	-0.5696	-0.5681	-0.5690	0.0015	0.0006
7	0.6813	0.6830	0.6868	0.0017	0.0055
8	-0.7771	-0.7739	-0.7756	0.0032	0.0015
9	0.8556	0.8549	0.8557	-0.0007	0.0001
10	-0.9164	-0.9179	-0.9179	-0.0015	-0.0015
11	0.9598	0.9579	0.9546	-0.0020	-0.0052
12	-0.9872	-0.9894	-0.9906	-0.0022	-0.0035
13	1.0000	1.0000	1.0000	0	0
14	-1.0001	-0.9996	-0.9986	0.0005	0.0015
15	0.9892	0.9892	0.9934	-0.0000	0.0042
16	-0.9692	-0.9680	-0.9733	0.0011	-0.0041

APPENDIX C

INTEGRAL CONTROL ALGORITHM

In this appendix we derive a MIMO inverse model using RLS estimation techniques used in Chapter III for the forward MIMO model. The inverse model is used in an integral control algorithm, which updates the TDW parameters to drive the RF to the ideal pulse described in equation 2.1

A. MIMO RLS ALGORITHM FOR THE INVERSE MODEL

The memoryless inverse MIMO model is obtained by reversing the roles of x and y vectors in the RLS algorithm presented in the previous section. The inverse model allows direct mapping of the errors in the RF parameters to the TDW parameters. The output of the inverse model is expressed as:

$$\hat{x} = By,$$

where vector y is the RF pulse parameters of the transmitter output, B is the coefficient matrix of the inverse model, and vector $\hat{x}$ is the estimated TDW half-cycle peak amplitudes. The error vector, at time index i, is given by:

$$e_i = x_i - \hat{x}_i, \tag{C.1}$$

where x_i is a vector of the actual TDW parameters. Forming a performance measure as a sum of squared errors and following on the lines of the development in equation 3.4 to equation 3.7 yields

$$\sum_{i=1}^{n} \lambda^{n-i}(x_i y_i^T) W = \sum_{i=1}^{n} \lambda^{n-i} B(y_i y_i^T) W. \tag{C.2}$$

Defining the autocorrelation matrix,

$$\Phi_n = \sum_{i=1}^{n} \lambda^{n-i} (y_i y_i^T) W , \qquad \text{(C.3)}$$

and the cross-correlation matrix,

$$\Gamma_n = \sum_{i=1}^{n} \lambda^{n-i} (x_i y_i^T) W , \qquad \text{(C.4)}$$

provides the solution

$$B = \Gamma_n \Phi_n^{-1}.$$

Derivation for the recursive least squares estimate of the inverse MIMO reference model, B, again follows the same procedure as the forward MIMO reference model, A, developed in the previous section. The recursive update equation for the inverse MIMO model parameters is

$$B_{n+1} = B_n + e_{n+1}\, y_{n+1}^T\, W\, P_{n+1},$$

where $e_{n+1} = (x_{n+1} - B_n\, y_{n+1})$, and the recursive expression for the inverse autocorrelation matrix is

$$P_{n+1} = \frac{1}{\lambda} P_n - \frac{1}{\lambda} P_n\, y_n + \left[I + y_{n+1}^T W \left(\frac{1}{\lambda} P_n \right) y_{n+1} \right]^{-1} y_{n+1}^T W\, \frac{1}{\lambda}\, P_n.$$

B. INTEGRAL CONTROL ALGORITHM

By employing an inverse reference model, the error between the ideal and the actual RF pulse parameters can be used to update the TDW parameters. This relationship is expressed as

$$e_x^{(n)} = B e_y^{(n)}, \qquad \text{(C.5)}$$

where vector $e_y^{(n)} = (y_{opt} - y_n)$, and B is the inverse reference model. The error vector $e_x^{(n)}$ is scaled and then summed with the TDW parameter vector x_n to produce the updated TDW parameters x_{n+1}. The expression of the integral control scheme is

$$x_{n+1} = x_n + \alpha\, e_x^{(n)}, \qquad \text{(C.6)}$$

where $0 < \alpha < 1$.

Figure C.1 shows the block diagram for the integral controller; the initial TDW parameters are set to arbitrary values. The closed loop transfer function [Ref. 11, 12] of Figure C.1 with an integrator in the forward path is

$$\frac{Y(z)}{Y_{opt}(z)} = \frac{B\alpha(\frac{1}{z-1})T}{1+B\alpha(\frac{1}{z-1})T} = \frac{\alpha BT}{z-(1-\alpha(BT))} \quad , \qquad \text{(C.7)}$$

where T represents the actual transmitter, and B is the estimated inverse model of the transmitter.

If B is an accurate inverse model where BT = 1, then the controller will have a pole at $(1 - \alpha)$. A small value for α puts the pole close to the unit circle. The controller is marginally stable for a very small α.

If B is not an exact inverse model so that $BT = 1 + \varepsilon$, then the controller will have a pole at $(1 - \alpha\,(1 + \varepsilon))$. For guaranteed stability, $0 < \alpha < 1$. Small values of α between 0.1 and 0.001 were used in the actual control algorithm, which produced the best TDW update performance.

The integral control algorithm closely resembles the steepest descent control algorithm, discussed in Chapter IV. The factor α, is comparable to the adaptive constant μ of the steepest descent algorithm. The overall performance of the integral control algorithm is similar to that of the steepest descent method discussed in Chapter V; therefore, detailed results of the tests and analysis aie not included in this report.

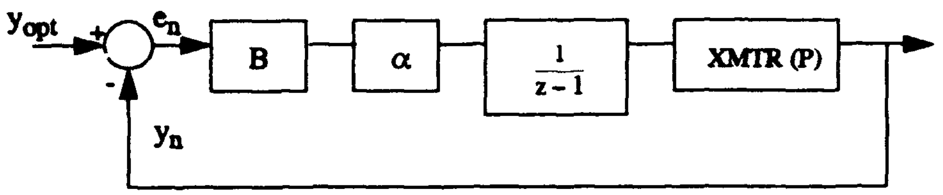

Figure C.1: A block diagram of the integral control scheme.

The most important result was found when testing the algorithm with samples of the RF pulse peaks and the desired zero-crossings. The RF parameter vector is simply expanded to 32 elements; the odd values (1,3,...,31) are the first 16 pulse peaks, and the even values (2,4,...,32) are the first 16 samples of the desired zero-crossings. The integral control algorithm was able to control the RF pulse half-cycle peak amplitudes and force the zero-crossings in tolerance when the simulated transmitter was run under ideal conditions. A weighting matrix was used to emphasize second and third zero-crossings.

Any degradation to the TDW or the RF pulse due to waveform quantization and additive noise limits the integral control algorithm's ability to meet zero-crossing tolerances in Table 2.1. Additive noise also decreases the algorithms performance to match the RF half-cycle peak amplitudes to the ideal amplitudes. Even though the noise is a limiting factor, the integral control algorithm has pulse shaping capabilities.

APPENDIX D

MATLAB PROGRAM LISTING

The MatLab program control algorithms are called by the simulated transmitter's main file SIMZ. The following MatLab M-files include the algorithm to develope an initial MIMO reference model for the control algorithm, and the MIMO RLS control algorithms with and without memory; and the integral control algorithm of Appendix C is also included. [Ref. 15]

Two MatLab programs from the initial development of the tranmsmitter simulator, written by Bruckner [Ref. 3], have been modified to accomodate for the MIMO RLS controller and the integral controller. The revised M-files are SETUP and DISPZ, these two programs are not listed here. The M-file ENVEL was modified to sample the half-cycle peaks and the desired zero-crossing locations; it is named ENVELPZ and listed below.

MIMO RLS Model Estimation

```
% convA.m
% Estimates a model (M,(D+1)*N) from x,y pairs obtained from steepest
% descent algorithm.
% model initialized as zeros(M,(D+1)*N)
% PgainM initialized as 100000*eye((D+1)*N,(D+1)*N), initialize loop = 1, ff = 1
% forgetting factor < 0.98 drives Pgain to zero matrix
% MIMO RLS Model Algorithm by John D. Wood 5/25/93
N = 16;             % # of input parameters
M = 1;              % # of output parameters
D = 2;              % # of delays in Model
 ff=1 ;
```

```
load data44           % load converged RF and TDW parameters for 44A
                      % transmitter both dummy load and antenna
PgainM = 100000*eye((D+1)*N,(D+1)*N);        % initialize P
model = zeros(M,(D+1)*N);                    % initialize MIMO model
x = fliplr(tdw44a(:,1:1+D));                 % create TDW parameter
x = reshape(x,(D+1)*N,1);                    % vector with possible prior
                                             % TDW parameters
y = rf44a(:,D+1);                            % RF paramter vector
for loop = (1):(2000-D-1)                     % 2000 x, y pairs to train
                                             % MIMO model
 if loop/100 == fix(loop/100)
 loop
 end                                         % Compute MIMO RLS Model
Kgain =ff*PgainM * x * inv(1 + x' * ff * PgainM * x);
PgainM =ff*PgainM- Kgain * x' * ff * PgainM;
mse1(loop) = mse(y,(model*x));                % MSE of model tracking
stomodel = model;
 ee = y-model*x;                              % model error
 model = model + ee * x' * PgainM;            % RLS update equation
norm1(loop) = sqrt(sum(sum((stomodel-model).^2)));% identify change in model by
                                             % taking norm of difference
                                             % between old and new models
x = fliplr(tdw44a(:,loop+1:loop+D+1));   % update x with new TDW parameters
x = reshape(x,(D+1)*N,1);
y = rf44a(:,loop+D+1);                   % update y with new RF parameters
end
                                         % display MSE of model convergence
subplot(211), semilogy(mse1(1:100))
```

```
title('MSE: (y-Ax), CO?'VERGENCE OF ANTENNA MODEL 44A')
xlabel('iterations'), ylabel('magnitude')
subplot(212), semilogy(mse1)
title('MSE: (y-Ax), CONVERGENCE OF ANTENNA MODEL 44A')
xlabel('iterations '), ylabel('magnitude')
print convA48a.ps, pause
                                        % display model norm
subplot(211), semilogy(norm1(1:100))
title('NORM OF A(n+1) - A(n)')
xlabel('iterations '), ylabel('magnitude')
subplot(212), semilogy(norm1)
xlabel('iterations '), ylabel('magnitude')
title('NORM OF A(n+1) - A(n)')
print convA48b.ps
```

Initialize RLS Controller/ with or without memory

```
% RLS_INI: Initializes the recursive least squares algorithm
% and provides the function call in a
% text string which is evaluated by the main program.
% RLSMENU changes the values in the simulation.
% See also RLS, RLSMENU, RLSHEAD, RLS_SWP
% All algorithms must at a minimum initialize the following
% variables: y0A, y0D, energy_out, x, tdw, y0, f_call,
% max_volts_A, max_volts_D, for both transmitters, as shown.
% John D. Wood, 4/4/93, rev. 7/7/93
tdw_pci=zeros(len,length(control));          % Input pulse buffer
```

```
rf_pci =zeros(len,length(control));          % Output pulse buffer
eval(['y0',num2str(N),'_scpt'])              % Load ideal half-cycle peak vals
                                             % for dummy load pulse building
                                              % (variable y0D). Note: ECD
                                             % does not apply.

if xmtr_id==2                                % 44A Transmitter
 max_volts_A=5;                              % Set desired output max voltage
                                             % (steady-state)
 x=1.7*sign(-cos(pi*(1:16)'));               % Initial TDW half-cycle values
 tdw=pgen(x,0,ETA);                          % Produce input vector to xmtr
                                             % (positive phase code here--
                                             % phase code is added when
                                             % tdw_pci is filled in SETUP).
 y0A=max_volts_A*envel(ideal(0,tau));        % Obtain ideal half-cycle peak
                                             % vals for local ECD, pos
                                             % phase code.
 boost=1.0015;                               % Used to scale tdws in pulses
                                             % after the one being controlled
                                             % so convergence is faster for
                                             % each one.

 if N==8                                     % N is decimation factor
 max_volts_D=23.7;
 f1 = 1.0; f2 = 1.0;                         % forgetting factor in RLS
 elseif N==4                                 % f1 for Dummy load
 max_volts_D=22.26;                          % f2 for Antenna
 f1 = 1.0; f2 = 1.0;
 elseif N==2
 max_volts_D=22.35;
```

```
f1=1.0; f2=1.0;
elseif N==1
max_volts_D=22.47;
f1=1.0; f2=1.0;
else
error('N must equal 1,2,4 or 8')
end
y0D=max_volts_D*y0D(:,2);                    % Select column & scale
load rls44_ini                               % Initializes Ant & Dum model,
                                             % and P for Ant and Dum Load,
                                             % for 44A Transmitter

else                                         % Initializes for 42 transmitter
 max_volts_A=22.24;                          % Set desired output max voltage
                                             % (steady-state)
 x = .4*sign(-cos(pi*(1:16)'));              % Initial TDW half-cycle values
 tdw=pgen(x,0,ETA);                          % Produce input vector to xmtr
                                             % (positive phase code here--
                                             % phase code is added when
                                             % tdw_pci is filled in SETUP).
 y0A=max_volts_A*envel(ideal(0,tau));        % Obtain ideal half-cycle peak
                                             % vals for local ECD, pos
                                             % phase code.

 boost=1.02;
 if N==8;
 max_volts_D=8.14;
 f1= 1.0; f2 = 1.0;
 elseif N==4
 max_volts_D=8.15;
 f1 = 1.0; f2 = 1.0;
 elseif N==2
 max_volts_D=8.07;
 f1 = 1.0; f2 = 1.0;
 elseif N==1
 max_volts_D=8.27;
 f1 = 1.0; f2 = 1.0;
 else
 error('N must be 1,2,4 or 8')
 end
 y0D=max_volts_D*y0D(:,1);                  % Select column & scale
 load rls42_ini                             % Initializes Ant and Dum model,
                                            % P for Ant and Dum Load
                                            % 42 xmtr
end
F1=num2str(f1);                             % Initial f factor (dummy load)
F2=num2str(f2);                             % f factor after switch from
                                            % dummy load to antenna
% *** No delays, Model is 16x16, PgainM and PgainC are 16x16 ******
if xmtr_load=='Antenna '
ff=f2; model = Amod; PgainM = PgainA;       % load initial parameters
PgainC = 100*eye(16,16);
y0=y0A;
else                                        % Dummy load
ff=f1; model = Dmod; PgainM = PgainD;
PgainC = 100*eye(16,16);
y0 = y0D;
end
```

```
% ********** Model had 2 delays, Model = (16x48) matrix ******************
%if xmtr_load=='Antenna '
% initializes model, PgainM, & x with delays for antenna
%load model48a;
% load PgainM48a;
%PgainC = 100*eye(16,16);
%else
%initializes model, PgainM, & x with delays for dummy
%load model48d;
%load PgainM48d;
%PgainC = 100*eye(16,16);
%end
% ****************************************************************
disp('Initializing RLS')                        % control algorithm
f_call = '[tdw,x,y,model,PgainC,PgainM,loop] =
rls(rf,tdw,PC,ETA,y0,x,ff,model,PgainC,PgainM,loop);';
boost1 = boost.^[0:len_p/2-1 0:len_p/2-1];      % boost for droop , p = pulse #
for p = 1:length(control)                       % Load simulated buffer for AFG
 tdw_pci(:,p) = boost1(control(p))*tdw*(-1)^phasecode(control(p));
end
clear boost1
```

RLS Contoller without Memory / Also Used for Peaks and Zeros

```
function [tdw,x,y,model,PgainC,PgainM,loop]=
rls(rf,tdw,PC,ETA,y0,x,ff,model,PgainC,PgainM,loop)
% Recursive Least Squares Control Algorithm using first 16 RF half-cycle peaks
% This RLS controller can be modified to control Peaks and Zeros by
% substituting envelPZ( ) for envel( ) in the beginning and
% added to the last line y = envel( ).
% A RLS estimate is provided for the transmitter model
% and used to generate a RLS estimate of x optimal.
% John D. Wood 6/1/93 rev(7/6/93)
% PgainC = 100*eye(16,16) initially
% x initial parameters are arbitrary
% model, PgainM, initial parameters are loaded in rls_ini.m
% model and PgainM parameters were converged with convA.m
global mse3 mse4 mse1 xmtr_id zc ztimes
y = envel(rf,tdw,PC);                          % Obtain output half-cycles
                                               % Substitute envelPZ for envel
                                               % when samp peaks & zeros
if loop/50 == fix(loop/50) & xmtr_id == 2      % 44 transmitter
 PgainC = PgainC + 100*eye(16,16);             % Re-initialize PgainC
 loop
end
if loop/50 == fix(loop/50) & xmtr_id == 1      % 42 transmitter
PgainC = PgainC + 10*eye(16,16);               % Re-initialize PgainC
loop
end
if loop/10 == fix(loop/10)                     % store zero-crossing times
 [z_err, zc_ns]=zcross(rf,tdw,PC);             % relative to SZC
 zc(:,loop) = abs(zc_ns - ztimes);
end
if loop/50 == fix(loop/50) & xmtr_id == 2      % 44 transmitter
 disp('model')                                 % every 50 iteratios display
```

```
loop
 (PgainM)
 rank((PgainM))
 cond((PgainM))
 eig((PgainM))'
 disp('controller')
 (PgainC)
 rank((PgainC))
 cond((PgainC))
 eig((PgainC))'
end
% if loop > 200 & loop<211                          % for added noise burst
%   x = x + randn(16,1)*.2;
% end
ee = y - model*x;                                   % error for model update
                                                    % shut off model udate when
                                                    % MSE < 2e-5 to reduce comput
% if mean((y0*(-1)^PC-y).^2) > 2e-5
 KgainM =ff*PgainM * x * inv(1 + x' * ff*PgainM * x); % inverse correlation
                                                    % matrix update
 PgainM =ff*PgainM - KgainM * x' * ff*PgainM;
 model = model + ee * x' * PgainM;                   % RLS estimate of model
%end
KgainC = PgainC*model'*inv(eye(16,16)+model*PgainC*model');
PgainC = PgainC - KgainC*model*PgainC;              % controller inverse
error = y0*(-1)^PC - model * x;                     % correlation matrix update
x = x + PgainC*model'*error;                        % RLS estimate of optimal x
mse4(loop) = mean(error.^2);                        % mse of rls to optimal x
```

```
mse3(loop) = mean((y0*(-1)^PC-y).^2);       % mse to ideal y
mse1(loop) = mean(ee.^2);                   % mse of model tracking
loop = loop +1;
tdw=pgen(x,PC,ETA);                         % Added envel( ) at end when
% y = envel(rf,tdw,PC);                     % sampling RF peaks and zeros
```

RLS Controller with memory

```
function [tdw,x,y,model,Pgain,PgainM,loop] =
rls(rf,tdw,PC,ETA,y0,x,ff,model,Pgain,PgainM,loop)
% Recursive Least Squares Control Algorithm with memory
% two previous TDW parameter vector used in control
% model is 16x48, PgainM = 48x48
% A recursive least squares estimate is provided for the transmitter model
% and used to generate a recursive least squares estimate of x optimal
% John D. Wood 6/15/93 rev(7/6/93)
% Pgain = 100*eye(16,16) initially
% x initial parameters are arbitrary
% model, PgainM, initial parameters are loaded in rls_ini.m
% model and PgainM parameters were converged with convA.m
global mse3 mse4 mse1
y=envel(rf,tdw,PC);                         % Obtain output half-cycles
 if loop/50 == fix(loop/50)                 % every 50 iterations display
 loop                                       % iteration number
 (y0*(-1)^PC)'                              % ideal parameters
 y'                                         % RF pulse parameters
 Pgain = Pgain + 100*eye(16,16);            % Re-initialize Pgain
 end
```

```
ee = y - model*x;                               % Model error
                                                % Shut off model estimation
                                                %if mean((y0-y).^2) > 2e-5
 KgainM =ff*PgainM * x * inv(1 + x' * ff*PgainM * x);
 PgainM =ff*PgainM - KgainM * x' * ff*PgainM;   % Recursive update of P
 model = model + ee * x' * PgainM;              % RLS estimate of model
%end
Kgain= Pgain*model(:,1:16)'*inv(eye(16,16)+model(:,1:16)*Pgain*model(:,1:16)');
Pgain = Pgain - Kgain*model(:,1:16)*Pgain;      % inverse corr matrix update
error = y0*(-1)^PC - model * x;                 % error in RF pulse
xx = x(1:16)+Pgain*model(:,1:16)'*error;        % RLS estimate of x
mse4(loop) = mean(error.^2);                    % mse of rls to optimal x
mse3(loop) = mean((y0*(-1)^PC-y).^2);           % mse to ideal y
mse1(loop) = mean(ee.^2);                       % mse of model tracking
loop = loop +1;
tdw=pgen(xx,PC,ETA);                            % Use updated parameters to
                                                % form TDW
x = [xx; x(1:32)];                               % x using past tdw paramters
```

Initialize the Integral Control Algorithm

```
% INTEG_INI: Initializes the integral control algorithm
% and provides the function call in a
% text string which is evaluated by the main program.
% INTEGMENU changes the values in the simulation.
% See also INTEG, INTEGMENU, INTEGHEAD, INTEG_SWP
% All algorithms must at a minimum initialize the following
% variables: y0A, y0D, energy_out, x, tdw, y0, f_call,
% max_volts_A, max_volts_D, for both transmitters, as shown.
% John D. Wood, 4/4/93, rev. 6/7/93
                                         % vector control = pulses to
                                         % control in PCI
                                         % len = length of pulse
tdw_pci=zeros(len,length(control));      % Input pulse buffer
rf_pci =zeros(len,length(control));      % Output pulse buffer
eval(['y0',num2str(N),'_scpt'])          % Load ideal half-cycle peak vals
                                         % for dummy load pulse building
                                         % (variable y0D). Note: ECD
                                         % does not apply.
if xmtr_id==2                            % xmtr_id = 2 is 44A Transmitter
 max_volts_A=5;                          % Set desired output max voltage
 x=1.7*sign(-cos(pi*(1:16)'));           % Initial TDW half-cycle values
 tdw=pgen(x,0,ETA);                      % Produce input vector to xmtr
load integ44_ini                         % load P and Inverse Model for
                                         %Antenna/Dummy
 y0A=max_volts_A*envel(ideal(0,tau));    % Obtain ideal half-cycle peak
                                         % vals for local ECD, pos
                                         % phase code.
 boost=1.0015;                           % Used to scale tdws in pulses
                                         % after the one being controlled
                                         % so convergence is faster for
                                         % each one.
 if N==8                                 %N is decimation factor
 max_volts_D=23.7;
 f1 =1; f2 =1; alpha = .03;              % f1 forgetting factor dum load
```

```
elseif N==4                                  % f2 forgetting factor antenna
max_volts_D=22.26;                           % alpha is the error gain
f1 =1; f2 =1; alpha = .03;
elseif N==2
max_volts_D=22.35;
f1=1; f2=1; alpha = .03;
elseif N==1
max_volts_D=22.47;
f1=1; f2=1; alpha = .03;
else
error('N must equal 1,2,4 or 8')
end
y0D=max_volts_D*y0D(:,2);                    % Select column & scale
else
max_volts_A=22.24;                           % Set desired output max voltage
load integ42_ini                             % load P and Inverse Model
                                             % for Antenna and Dummy Load
x=.4*sign(-cos(pi*(1:16)'));                 % Initial TDW half-cycle values
tdw=pgen(x,0,ETA);                           % Produce input vector to xmtr
                                             % (positive phase code here--
                                             % phase code is added when
                                             % tdw_pci is filled in SETUP).
y0A=max_volts_A*envel(ideal(0,tau));         % Obtain ideal half-cycle peak
                                             % vals for local ECD, pos
                                             % phase code.
boost=1.012;
if N==8;
max_volts_D=8.14;
```

```
f1=1; f2 = 1.0; alpha = .03;
elseif N==4
max_volts_D=8.15;
f1 = 1; f2 = 1.0; alpha = .03;
elseif N==2
max_volts_D=8.017;
f1 = 1; f2 = 1.0; alpha = .03;
elseif N==1
max_volts_D=8.27;
f1 = 1; f2 = 1.0; alpha = .03;
else
error('N must be 1,2,4 or 8')
end
y0D=max_volts_D*y0D(:,1);                        % Select column & scale
end
if xmtr_load=='Dummy Load'                        % Initialize inverse model, P,
                                                  % forgetting factor, y0
 y0=y0D; InModel=Emod; Pgain=PgainE; ff=f1;       % Dummy load parameters
else
 y0=y0A; InModel=Bmod; Pgain = PgainB; ff=f2;     % Antenna parameters
end
F1=num2str(f1);                                   % Initial f factor (dummy load)
F2=num2str(f2);                                   % f factor for antenna
ALPHA=num2str(alpha);                             % error gain in control
loop = 1;
W = eye(16,16);                                   % create weighting matrices:
 for h = 1:4                                      % W is in controller
 W(h,h)=1; V(h,h)=1;                              % V is in model estimation
```

```
end                                                     % weight error vectors
for h = 5:8
W(h,h)=1; V(h,h)=1;
end
for h = 9:12
W(h,h)=1; V(h,h)=1;
end
for h = 13:16
W(h,h)=1; V(h,h)=1;
end
f_call='[tdw,x,y,InModel,Pgain,loop,alpha] =            % control algorithm
integ(rf,tdw,PC,ETA,y0,x,ff,InModel,Pgain,loop,alpha,W,V);';
boost1=boost.^[0:len_p/2-1 0:len_p/2-1];                % compensate for droop
for p = 1:length(control)                               % Load simulated buffer for AFG
 tdw_pci(:,p)=boost1(control(p))*tdw*(-1)^phasecode(control(p));
end
clear boost1
```

Integral Control Algorithm

```
function [tdw,x,y,InModel,Pgain,loop,alpha] =
integ(rf,tdw,PC,ETA,y0,x,ff,InModel,Pgain,loop,alpha,W,V)
% Variables loaded from INTEG_INI : Pgain, InModel, x, ff, loop, W, V, y0, alpha
% x initialized from arbitrary tdw parameters
% InModel initialized as converged inverse model
% Pgain initialized as converged P matrix
% start loop = 1
% W and V are weighting matrix,
% forgetting factor < 1 drives Pgain to zero
% alpha between .1 and .001 best
% Integral Control Algorithm by John D. Wood 5/1/93
global mse1 mse3
y=envel(rf,tdw,PC);                               % Obtain RF half-cycles peaks
%if loop > 200 & loop<211                          % Added noise burst
% x = x + randn(16,1)*.2;
%end
 Kgain =ff*Pgain * y * inv(1 + y' * ff * V * Pgain * y);
 Pgain =ff*Pgain - Kgain * y' * ff * V * Pgain;   %update inverse corr matrix
ee = x-InModel*y;                                 % error in model estimate
InModel = InModel + ee * y' * V * Pgain;          % RLS estimate of Inverse
                                                  % Model
x = x + alpha * InModel * W * (y0-y);             % Integral Control of x
 mse3(loop) = mse(x,(InModel*y));                 % MSE of model tracking
 mse1(loop) = mse(y0,y);
 loop = loop+1;
 tdw=pgen(x,PC,ETA);
```

Samples the First 16 RF Pulse Half-cycle Peaks and First 16 Desired Zero-crossings

```
function yPZ = envelPZ(rf,tdw,PC,fs_pk)
% Function yPZ =ENVELQ(rf,tdw,PC,fs_pk): Computes the first 16 half-
% cycle peak values of the Loran-C transmitter output
% and samples the values at the desired zero crossings (rf feedback).
% If tdw is included, the function auto-synchronizes rf to find
% correctly the beginning of the first half-cycle. If tdw is absent
% or if tdw==0, then sample 1 of rf is taken to be the beginning of
% the first half cycle.
% Vector rf must be interpolated to a
% higher sample freq (>=10.0 MHz) to find accurate half cycle
% peak values and zero crossing values.
% Calls: INTERP, FLIPUD
% Uses global vars: xmtr_id, xmtr_load, fs
global xmtr_id xmtr_load fs
% usually fs = 10e6;
% usually xmtr_id=2;
% usually xmtr_load = 'Antenna ';
% Variables:
% bin = Half-cycle number (1-16)
% bin_start = First sample in bin (half-cycle)
% bin_width = Number of samples per half cycle
% fs_pk = Higher sampling frequency (used internally only)
% in interpolation process.
% fs = Sampling frequency used in rf, tdw
% PC = Pulse phase code (0=pos, 1=neg)
% rf = Transmitter output (radio frequency feedback)
% tdw = Transmitter drive waveform
% y = Output half-cycle peak values
% Fernando Pires, 1/15/92; rev. by Dean C. Bruckner, 9/2/92,
% rev. by John D. Wood 7/3/93
%********************** Find Standard Zero Crossing of RF *************
if nargin < 2;
 tdw = 1; rf_start = 1;
```

```
else
 for i = 1:length(tdw)                          % Auto-synchronize: find 1st
 if abs(tdw(i)) > .1, break                     % sample of sin pulse at fs
 end
 end
 rf_start = round(20/5e6*fs) + i;              % Normal delay: start of tdw to
                                               %beginning of 1st half cycle
                                               % is 25 samples at 5 Mhz
end
rf =rf-mean(rf);
intp_fact = round(10e6/fs);
if intp_fact ==1
 rf_temp = rf(rf_start: rf_start + 18 * 5e-6 * fs); % interp to 10Mhz
else
 rf_temp = interp(rf(rf_start: rf_start + 18 * 5e-6 *fs),intp_fact);
end
                           % sample rf at freq 10Mhz starting at beginning of bin 1.
                           % Keeping 18 half cycles leaves enough rf
s = sign(rf_temp);
f = find(s(2:650) - s(1:649)~=0);         % Find samples nearst zero crossings
f300 = find(f>275 & f<325);               % find index of SZC sample in rf_temp
SZC = f(f300);                            % finding exact zero crossing of SZC
delta = 1 - 2*abs(rf_temp(SZC+1)/(rf_temp(SZC+1)-rf_temp(SZC-1)));
% ***********************************************************
% Determine Values at Ideal Zero Crossing Locations
% sample every 1.25 usec from start of 1st half-cycle to 16th
% half-cycle going through SZC
yPZ = zeros(32,1);
```

```
if SZC >= 270
 for k = 1:16
 i = SZC - 300 + 50*k;
 yPZ(2*k,1) = ((rf_temp(i+1)-rf_temp(i-1))/2)*delta + rf_temp(i);
 end
end;
if SZC < 270
Disp(' Error in find first zero crossing ')
end
% ****** Determine Peak Values in first 16 Half - Cycles *******
if nargin<4;fs_pk=5e6;end                        % Default: 5 MHz
if nargin<3;PC=0;end                             % Default: pos phase code
if xmtr_id==2                                    % Input to Output delay in samp
 if xmtr_load=='Antenna '
 cut=14;
 else
 cut=-1;
 end
else
 if xmtr_load=='Antenna '
 cut=20;
 else
 cut=18;
 end
end
if nargin<2;                                     % Default externally synchron
 rf_start=1;
elseif tdw==0;
```

```
rf_start=1;
else
 for i=1:length(tdw)                          % Auto-synchronize (find 1st
 if abs(tdw(i)) > 0.1, break                  % sample of sine pulse, at
 end % lower sample freq, fs)
 end
 rf_start=round(cut/5e6*fs) + i;              % Start of 1st output half-cycle
 if rf_start<1;rf_start=1;end
end
intp_fact=round(fs_pk/fs);
if intp_fact<=1
 rf=rf(rf_start: rf_start + 18 * 5e-6 * fs);
 bin_width=5e-6 * fs;                         % Number ot samp per half cycle
else
 rf=interp(rf(rf_start: rf_start + 18 * 5e-6 * fs),intp_fact);
 bin_width=5e-6 * fs_pk;                      % Number of samp per half cycle
end                                           % Samp rf at higher freq (fs_pk),
                                              % starting at beginning of
                                              % bin 1. Keeping 18 half-
                                              % cycles leaves enough
                                              % of rf to work with.
%plot(rf),grid,pause
bin_start=1;
if PC==0
 for bin=1:4:29 % Peak value of each half-cycle
 yPZ(bin)=max(rf(bin_start:bin_start+bin_width));
 % plot(rf(bin_start:bin_start+bin_width)),title(num2str(yPZ(bin))),pause
 bin_start=bin_start+bin_width;
 yPZ(2+bin)=min(rf(bin_start:bin_start+bin_width));
 % plot(rf(bin_start:bin_start+bin_width)),title(num2str(yPZ(2+bin))),pause
 bin_start=bin_start+bin_width;
 end
else
 for bin=1:4:29 % Peak value of each half-cycle
 % plot(rf(bin_start:bin_start+bin_width)),pause
 yPZ(bin)=min(rf(bin_start:bin_start+bin_width));
 bin_start=bin_start+bin_width;
 %% plot(rf(bin_start:bin_start+bin_width)),pause
 yPZ(2+bin)=max(rf(bin_start:bin_start+bin_width));
 bin_start=bin_start+bin_width;
 end
end
% **************** PLOT RF Pulse and Samples ***************
%plot(rf_temp(1:900)) % Plot RF\
%hold on
 %for k = 1:16
 % plot((k*50)+SZC-300,yPZ(2*k,1),'o'); % PLOT Zero Crossings\
 % end
%bin_start=1;
%for bin = 1:8
 % Peak value of each half-cycle\
 % val=max(rf_temp(bin_start:bin_start+bin_width));
 % plot(find(rf_temp(bin_start:bin_start+bin_width)==val)+bin_start-1,val,'o')
 % bin_start=bin_start+bin_width;
 % val=min(rf_temp(bin_start:bin_start+bin_width));
 % plot(find(rf_temp(bin_start:bin_start+bin_width)==val)+bin_start-1,val,'o')
 % bin_start=bin_start+bin_width;
 % end
%hold off
```

REFERENCES

1. Doug Taggart and Ben Stewart, "Electronic Equipment Replacement Project (EERP)," Technical Report, USCG Electronics Engineering Center 1989.

2. Specification of the Transmitted Loran-C Signal, USCG Commandant Instruction M16562.4, July 1981.

3. Dean Bruckner, "Automatic Pulse Shaping with the AN/FPN-42 and AN/FPN-44 LORAN-C Transmitters," M.S. Thesis, Naval Postgraduate School, 1992.

4. Benjamin Peterson, "Closed Loop Control of Loran-C Pulse Shape for Tube Type Transmitters," Derivation, USCG Academy 1988.

5. Dean Bruckner, personal communication, July 1993.

6. Doug Taggart and Jon Turban, "VXIbus Based Loran-C Transmitter Monitor and Control System," Technical Report, USCG Electronics Engineering Center, 1991.

7. Enders A. Robinson, *Multichannel Time Series Analysis with Digital Computer Programs Revised Edition*, Holden-Day, Inc., CA, 1978.

8. Simon Haykin, *Adaptive Filter Theory*, Prentice-Hall, Inc., Englewood Cliffs, NJ, *1991*.

9. Bernard Widrow and Samuel D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985.

10. Torsten Soderstrom and Petre Stoica, *System Identification*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.

11. Richard O. Hill, Jr., *Elementary Linear Algebra with Applications*, Harcourt Brace Jovanovich, Publishers, Orlando, FL, 1991.

12. Karl Johan Astrom and Bjorn Wittenmakr, *Adaptive Control*, Addison-Wesley Publishing Company, 1989.

13. Benjamin C. Kuo, *Automatic Control Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

14. Robert Strum and Donald Kirk, *Discrete Systems and Digital Signal Processing*, Addison-Wesley Publishing Co., Menlo Park, CA, 1989.

15. *MATLAB Reference Guide*, published by Math Works, Inc., 1992.

INITIAL DISTRIBUTION LIST

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

2. Dudley Knox Library, Code 52 2
   Naval Postgraduate School
   Monterey, CA 93943-5101

3. Chairman, Code EC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Prof. Murali Tummala, Code EC/Tu 2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Prof. Roberto Cristi, Code EC/Cx 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. CAPT B. B. Peterson 1
   Chief, Electrical Engineering Section
   Department of Engineering
   U.S. Coast Guard Academy
   New London, CT 06320

7. Commandant (G-NRN-1) 1
   U.S. Coast Guard
   2100 W. 2nd Street S.W.
   Washington, DC 20593-0001
   Attn: CDR Doug Taggart

8. Commandant (G-PIM-2/O) 1
   U.S. Coast Guard
   2100 W. 2nd Street S.W.
   Washington, DC 20362-0001

9.	Commandant (G-TPR-2) U.S. Coast Guard 2100 W. 2nd Street S.W. Washington, DC 20362-0001	1
10.	Commanding Officer (nlx) U.S. Coast Guard Electronics Engineering Center P.O. Box 60 Wildwood, NJ 08260-0060 Attn: LT John D. Wood	1
11.	Commanding Officer (nlx) U.S. Coast Guard Electronics Engineering Center P.O. Box 60 Wildwood, NJ 08260-0060 Attn: LT Dean C. Bruckner	1
12.	Commanding Officer (nlx) U.S. Coast Guard Electronics Engineering Center P.O. Box 60 Wildwood, NJ 08260-0060 Attn: LCDR G. Kmiecik	1