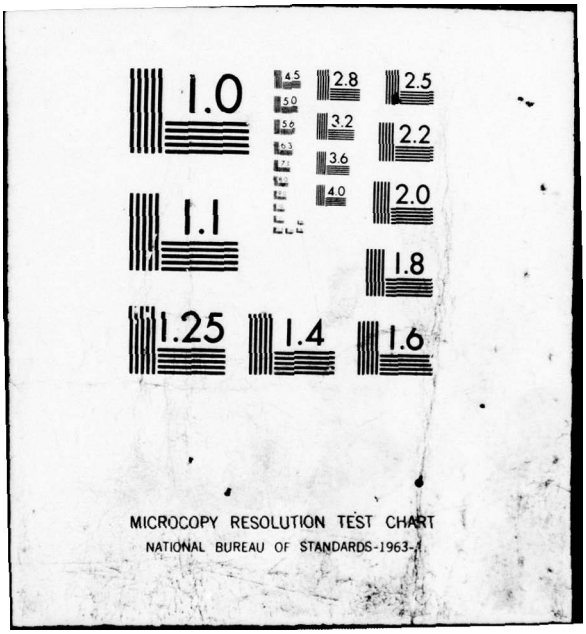
AD-A068 760 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/3
ADAPTIVE LINEAR ESTIMATION ALGORITHMS APPLIED TO SPECTRAL LINE --ETC(U)
AUG 78 S D HUFFMAN N00014-75-C-0191
UNCLASSIFIED TR-15 NL

1 OF 4
AD A068760





1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

TR-15

ADAPTIVE LINEAR ESTIMATION ALGORITHMS APPLIED TO SPECTRAL LINE ENHANCEMENT

by

Stephen D. Huffman
Department of Electrical Engineering

Prepared under:
Office of Naval Research (Code 222)
Contract No. N000 14-75-C-0191

"This document has been approved for public release and sale; its distribution is unlimited."

LEVEL II

(12)

DUKE UNIVERSITY

ADAPTIVE SIGNAL DETECTION LABORATORY

Department of Electrical Engineering
School of Engineering

118230

(9) Technical Report. No. 15

(14) TR-15

(6) ADAPTIVE LINEAR ESTIMATION ALGORITHMS APPLIED TO SPECTRAL LINE ENHANCEMENT

by

(10) Stephen D. Huffman

(12) 336p.

(11) August 1978

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

Approved: L. W. Nolte
L. W. NOLTE
Principal Investigator

Prepared under: Office of Naval Research (Code 222)
Contract No. (15) N00014-75-C-0191

DDC
RECEIVED
MAY 21 1979
D

"This document has been approved for public release and sale; its distribution is unlimited."

118 230
79 05 18 019

This report was also a dissertation submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Electrical Engineering in the Graduate School of Arts and Sciences of Duke University, 1978.

Copyright by
Stephen Doyle Huffman
1978

ABSTRACT

The estimation of a signal in the presence of an additive noise process is a basic problem in communications theory. When the processor chosen to carry out the estimation task is a linear digital filter with a finite unit sample response, the design problem is to determine the unit sample response which will produce the best possible estimate of the signal according to some criterion. If the criterion chosen as a measure of optimality is the mean-square error of estimation and if the statistics of the signal and noise are known, the optimal unit sample response can be found by the solution of a system of linear equations. The resulting processor is, then, an example of the classical Wiener filter.

When the statistics of the signal and noise are not known a priori, the unit sample response of the Wiener filter cannot be determined and some method for approximating the optimal filter is required. One approach to this problem is to use an adaptive processor which attempts to "learn" the characteristics of the signal and the noise and to use this information to obtain an approximation to the optimal filter. The LMS Gradient Algorithm is a widely used method for the solution of the adaptive linear estimation problem which has the advantages of being relatively simple to implement and of being well suited for

real-time processing in many applications. However, since this algorithm is iterative in nature, the convergence properties of the method must be considered in an evaluation of performance.

In this dissertation, two adaptive algorithms based on fixed-point iteration methods and using direct estimates of the required statistics are proposed. These algorithms are slightly more complex than the LMS Gradient Algorithm but have the advantage that the amount of computational effort required during each sample period can be varied. Thus, trade-offs can be made between the number of operations required per sample and the rate of convergence of the algorithm.

The performance characteristics of the LMS Gradient Algorithm, two Adaptive Fixed-Point Iteration Algorithms and of a non-iterative method based on Levinson's Algorithm are considered for the case where an adaptive algorithm is used to determine the unit sample response for a system which attempts to discriminate between the signal and noise processes on the basis of bandwidth. Such a system is often referred to as a Spectral Line Enhancer. Theoretical bounds on the mean-square error as a function of the time index n are derived for each of the three iterative methods. A comparison of these bounds is made for the case where the input data to the Spectral Line Enhancer is composed of a single sinusoid of random phase in the presence of an additive autoregressive noise sequence. The results of extensive computer simulations of the adaptive algorithm considered are used to determine the usefulness of the theoretical bounds and to make comparisons of the performance of

the four methods. The results of the computer simulations indicate that the theoretical bounds on the mean-square error are close to the actual mean-square error produced by the filters determined by the three iterative algorithms for the case considered. In addition, a comparison of the theoretical bounds and the computer simulation results indicates that the Adaptive Fixed-Point Iteration Method using the Successive Over-Relaxation Algorithm can yield significantly better performance than the LMS Gradient Algorithm with an increase in computational complexity which would not be prohibitive in many real-time applications.

ACKNOWLEDGEMENTS

I would like to express my appreciation to Professor Loren W. Nolte, my faculty advisor for his help and encouragement throughout the course of the research described in this dissertation. In addition, I thank Professors Thomas G. Wilson and H.A. Owen, Jr., members of my faculty advisory committee, for their valuable contributions to my graduate education.

I would also like to thank my parents for their support and interest in my education; and Mrs. Nancy Middleton for her typing of this manuscript.

Most importantly, I would like to thank my wife, Ann, not only for her help in typing and proofreading the drafts of this dissertation, but also for her patience, support and encouragement during the course of my research.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS vi

LIST OF FIGURES xi

I. ADAPTIVE LINEAR ESTIMATION AND SPECTRAL LINE ENHANCEMENT 2

1.1 Discrete Time Estimation, 2
1.2 Linear Estimation, 4
1.3 Mean-Square Error and Optimum Estimation, 8
1.4 Solution for Minimum Mean-Square Error, 8
1.5 Application of the Optimal Filter and Adaptive Filtering, 13
1.6 Separation of Signals Based on Bandwidth, 17
1.7 Algorithms for Approximation of the Optimal Solution, 24

II. THE LMS GRADIENT ALGORITHM 29

2.1 Steepest Descent Algorithm, 31
2.2 Convergence of the LMS Gradient Algorithm, 36
2.3 Rate of Convergence of the LMS Gradient Algorithm, 42
2.4 LMS Gradient Algorithm -- Another View, 49
2.5 Conclusion, 56

III. ADAPTIVE ESTIMATION USING FIXED-POINT ITERATION 58

3.1 Estimation of the Autocorrelation Matrix and Vector, 59
3.1.1 Estimation of the Autocorrelation Function -- Time-Varying Case, 68
3.1.2 Recursive Estimation, 71
3.2 Approximate Solution of the Wiener-Hopf Equation by Adaptive Fixed-Point Iteration, 72
3.2.1 Fixed-Point Iteration, 73
3.2.2 Selection of the Matrix C to Insure Convergence, 78
3.2.3 Rate of Convergence of the Mean-Square Error, 79

3.3 The Jacobi Relaxation Method for Adaptive Fixed-Point Iteration, 86
3.3.1 Selection of β to Insure Convergence, 89
3.3.2 Rate of Convergence of the Mean-Square Error, 92
3.3.3 Alternate Form for the Jacobi Relaxation Method, 93
3.4 The Successive Over-Relaxation Method for Adaptive Fixed-Point Iteration, 96
3.4.1 Rate of Convergence of the Mean-Square Error, 100
3.4.2 Alternate Form for the Successive Over-Relaxation Method, 101
3.5 Number of Operations Required by Adaptive Fixed-Point Iteration, 105
3.5.1 Reduction of Number of Operations/Sample by use of Partial Iteration Steps, 108
3.5.2 Implementation of the Jacobi Relaxation Method using the Discrete Fourier Transform, 112
3.5.3 Reduction of Computation Time and Parallel Processing, 124

IV. NON-ITERATIVE SOLUTION -- THE LEVINSON ALGORITHM 129

4.1 The Levinson Algorithm, 131
4.1.1 Proof of Levinson's Algorithm, 133
4.1.2 Number of Operations Required, 138
4.2 Adaptive Estimation Using Levinson's Algorithm, 140
4.3 Conclusion, 143

V. THEORETICAL PERFORMANCE OF THE LMS GRADIENT ALGORITHM AND ADAPTIVE FIXED-POINT ITERATION 145

5.1 Statement of the Problem, 146
5.1.1 Determination of the Autocorrelation Matrix, 148
5.1.2 Initial Conditions and Parameter Values, 153
5.2 Theoretical Performance of the LMS Gradient Algorithm, 154
5.3 Theoretical Performance of the Jacobi Relaxation Method, 156
5.3.1 Theoretical Performance of Jacobi Relaxation for Varying β, 157
5.3.2 Theoretical Performance of Jacobi Relaxation under the Variable Iteration Step Method, 158

5.4 Theoretical Performance of the SOR Method, 162
5.4.1 Theoretical Performance of Successive Over-Relaxation for Varying β, 162
5.4.2 Theoretical Performance of Successive Over-Relaxation under the Variable Iteration Step Method, 163
5.5 Comparison of the Three Algorithms, 166
5.6 Conclusion, 172

VI. INVESTIGATION OF PERFORMANCE THROUGH COMPUTER SIMULATION 174

6.1 The Problem and Simulation Method, 176
6.1.1 The Simulation Method, 179
6.2 Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ, 180
6.3 Comparison of the Theoretical Bound and Simulation Results for the Jacobi Relaxation Method, 186
6.3.1 Performance of the Jacobi Relaxation Method for Various Values of the Relaxation Parameter β, 187
6.3.2 Performance of Jacobi Relaxation using the Variable-Iteration-Step Method, 189
6.4 Comparison of the Theoretical Bound and Simulation Results for the Successive Over-Relaxation Method, 195
6.4.1 Performance of the SOR Method for Various Values of the Relaxation Parameter β, 200
6.4.2 Performance of the SOR Algorithm using the Variable Iteration Step Algorithm, 206
6.5 Comparison of Adaptive Estimation Algorithms Through Computer-Simulation, 212
6.6 Conclusion, 216

VII. SUMMARY AND SUGGESTIONS FOR FUTURE RESEARCH 231

7.1 Summary, 232
7.2 Suggestions for Future Research, 240

APPENDIX A. PROOF OF THEOREM 2.1 242

APPENDIX B. APPROXIMATION OF MSE(n) FOR THE LMS GRADIENT METHOD 246

APPENDIX C. MEAN AND VARIANCE OF THE TERM 250

$$\frac{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)}$$

APPENDIX D. COMPUTER PROGRAMS FOR THE STUDY OF FOUR ADAPTIVE LINEAR ESTIMATION ALGORITHMS 255

REFERENCES 310

LIST OF FIGURES

Figure		Page
1.1	Basic Estimation Problem	3
1.2	Discrete Time Estimation Problem	5
1.3	Transversal Filter for Estimation	7
1.4	Estimator for Narrowband Signal in the Presence of a Wideband Additive Noise Process	20
1.5	Estimator for Wideband Signal in the Presence of a Narrowband Additive Noise Process	22
1.6	Spectral Line Enhancer	25
1.7	Conceptual Flow Diagram for Adaptive Fixed-Point Iteration	27
3.1	Required Number of Operations/Sample for the Estimation of Φ_{XX} and $\underline{R}_{XX}$ by the Moving Average Proposed in Section 3.1.3.	106
3.2	Required Number of Operations/Sample for the Jacobi Relaxation and SOR Methods Computed directly using (3.76) and (3.93) respectively. N=Filter length.	107
3.3	Total required Number of Operations/Sample for the Jacobi Relaxation Method and the SOR Method Computed Directly using (3.76) and (3.93) respectively with Estimation of the Autocorrelation Matrix and Vector by the Moving Average Proposed in Section 3.2.1. N=Filter length, n_1=delay in unit sample response.	108
3.4	Average number of Operations/Sample for Jacobi Relaxation and SOR Methods, with Partial Iteration Steps. N=Filter length, ℓ/k=Partial Iteration Step Parameter.	110
3.5	Number of real Operations required to Implement Jacobi Relaxation using the FFT.	119
4.1	Number of Operations Required for the Solution of a Hermetian Topelitz System by Levinson's Algorithm.	138
4.2	Number of Operations Required for the Solution of a Hermitian Topelitz System by Levinson's Algorithm with Modification due to Zohar.	139

Figure		Page
4.3	Average Number of Operations/Sample Required by Levinson's Algorithm and the Modified Levinson Algorithm when Carried out Over K Sample Periods.	141
4.4	Total Average Number of Operations/Sample Required to Estimate Φ_{XX} and R_{XX} Using (3.25) and to Solve the System (3.26) using Levinson's Algorithm Over K Sample Periods.	142
5.1	Parameter Values in Study of Theoretical Rates of Convergence of the Three Iterative Algorithms Considered.	153
5.2	Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ under Four Different SNR Conditions.	155
5.3	Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β under Four Different SNR Conditions.	159
5.4	Theoretical Performance of the Jacobi Relaxation Algorithm using the Variable Iteration Step Method for Various Values of the Parameter k under Four Different SNR Conditions. $\ell=16$, $\beta=0.14$.	161
5.5	Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β under Four Different SNR Conditions.	164
5.6	Theoretical Performance of the SOR Algorithm using the Variable Iteration Step Method for Various Values of the Parameters ℓ and K under Four Different SNR Conditions. $\mu=1.0$.	167
5.7	Comparison of the Theoretical Performance of the LMS Gradient, Jacobi Relaxation and SOR Algorithms under Four Different SNR Conditions. The Variable Iteration Step Method has been used to Reduce the Number of Operations per Sample Required by the Jacobi and SOR Algorithms to O(N).	171
6.1	Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR=+7.0 db.	182
6.2	Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR=-3.0 db.	183

Figure		Page
6.3	Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR=-13.0 db.	184
6.4	Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of he Parameter μ. SNR=-23.0 db.	185
6.5	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR=+7.0 db.	190
6.6	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR=-3.0 db.	191
6.7	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR=-13.0 db.	192
6.8	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR=-23.0 db.	193
6.9	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm using the Variable-Iteration-Step Method. β=0.14, SNR=+7.0 db.	196
6.10	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm uisng the Variable-Iteration-Step Method. β=0.14, SNR=-3.0 db.	197
6.11	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm using the Variable-Iteration-Step Method. β=0.14, SNR=-13.0 db.	198
6.12	Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm using the Variable-Iteration-Step Method. β=0.14, SNR=-23.0 db.	199
6.13	Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=+7.0 db.	202
6.14	Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=-3.0 db.	203

Figure		Page
6.15	Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=-13.0 db.	204
6.16	Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=-23.0 db.	205
6.17	Comparison of Actual and Theoretical Performance of the SOR Algorithm using the Variable-Iteration-Step Method. β=1.0, SNR=+7.0 db.	208
6.18	Comparison of Actual and Theoretical Performance of the SOR Algorithm using the Variable-Iteration-Step Method. β=1.0, SNR=-3.0 db.	209
6.19	Comparison of Actual and Theoretical Performance of the SOR Algorithm using the Variable-Iteration-Step Method. β=1.0, SNR=-13.0 db.	210
6.20	Comparison of Actual and Theoretical Performance of the SOR Algorithm using the Variable-Iteration-Step Method. β=1.0, SNR=-23 db.	211
6.21	Parameter Values and Average Number of Operations/Sample for the Four Algorithms as Implemented in the Computer Simulation.	218
6.22	Comparison of Actual Performance of the Four Adaptive Algorithms. SNR=+7.0 db.	219
6.23	Comparison of True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates for the Four Adaptive Algorithms. SNR=+7.0 db.	220
6.24	Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=+7.0 db.	221
6.25	Comparison of Actual Performance of the Four Adaptive Algorithms. SNR=-3.0 db.	222
6.26	Comparison of True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=-3.0 db.	223
6.27	Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=-3.0 db.	224

Figure		Page
6.28	Comparison of Actual Performance of the Four Adaptive Algorithms. SNR=-13.0 db.	225
6.29	Comparison of True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=-13.0 db.	226
6.30	Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=-13.0 db.	227
6.31	Comparison of Actual Performance of the Four Adaptive Algorithms. SNR=-23.0 db.	228
6.32	Comparison of True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=-23.0 db.	229
6.33	Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR=-23.0 db.	230
7.1	Summary of Equations Required for the Implementation of the Four Adaptive Algorithms.	236
D.1	Structure of the Simulation Program ESTIM8.	257
D.2	Structure of Program EIGEN -- For the computation of the Eigenvalues of Φ_{XX}.	278
D.3	Structure of Program THEO -- For the Computation of Theoretical Bounds and for Display of Data.	286
D.4	Structure of Program SPECT -- For the Computation and Plotting of the Magnitude of the DFT of the Signal Estimates.	302

ADAPTIVE LINEAR ESTIMATION ALGORITHMS

APPLIED TO SPECTRAL LINE ENHANCEMENT

CHAPTER I

ADAPTIVE LINEAR ESTIMATION AND SPECTRAL LINE ENHANCEMENT

A basic problem in communications theory is the estimation of a signal $\tilde{S}(t)$ in the presence of an additive noise process $\tilde{N}(t)$. The available information about the signal is then in the form of the received data given by the sum:

$$\tilde{X}(t) = \tilde{S}(t) + \tilde{N}(t) \quad . \tag{1.1}$$

The signal processing system which is assigned the task of estimating $\tilde{S}(t)$ must, given the input signal $\tilde{X}(t)$, produce an output which in some sense approximates $\tilde{S}(t)$. Thus, the basic estimation problem is illustrated in Fig. 1.1 where the signal processor is represented by a two-part "black box".

1.1 Discrete Time Estimation

The advances in the development of high-speed digital processors have led to their widespread use in the area of signal processing. Within the discrete-time framework of a digital system, the basic estimation problem of Fig. 1.1 may be reformulated as the estimation of a sample sequence $S(n)$ in the presence of

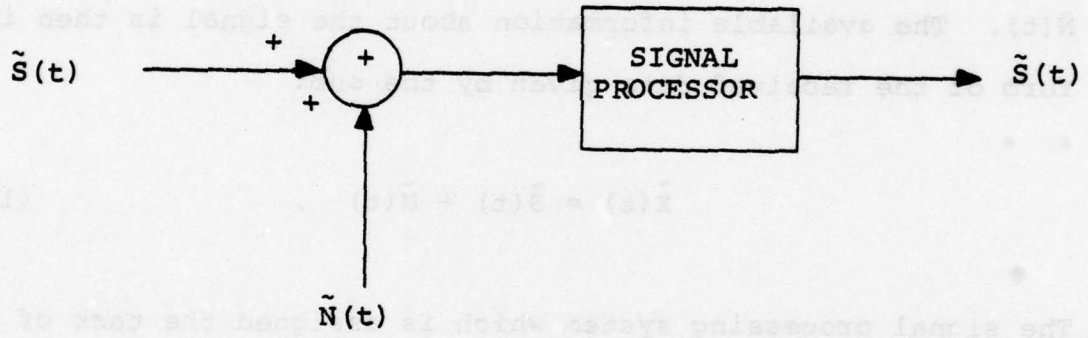


Figure 1.1 Basic Estimation Problem.

an additive noise sequence N(n). The received information is then in the form of the data sequence:

$$X(n) = S(n) + N(n) \tag{1.2}$$

where it is implied that if T is the sampling period:

$$\begin{aligned} S(n) &= \tilde{S}(nT) \\ N(n) &= \tilde{N}(nT) \\ X(n) &= \tilde{X}(nT) \quad . \end{aligned} \tag{1.3}$$

This thesis, then, is concerned with the discrete-time estimation problem illustrated in Figure 1.2 where the Digital Signal Processor block receives the input sequence X(n) and produces an output sequence $\hat{S}(n)$ which in some sense approximates the sequence S(n).

1.2 Linear Estimation

If the Digital Signal Processor in Fig. 1.2 produces the output $\hat{S}(n)$ by applying a linear transformation to the data sequence X(n) then the estimation is said to be linear. In the case considered here, it is assumed that the processor is in the form of a causal, linear digital filter with a finite unit sample response h(n). That is, h(n) may take on non-zero values only over the finite interval:

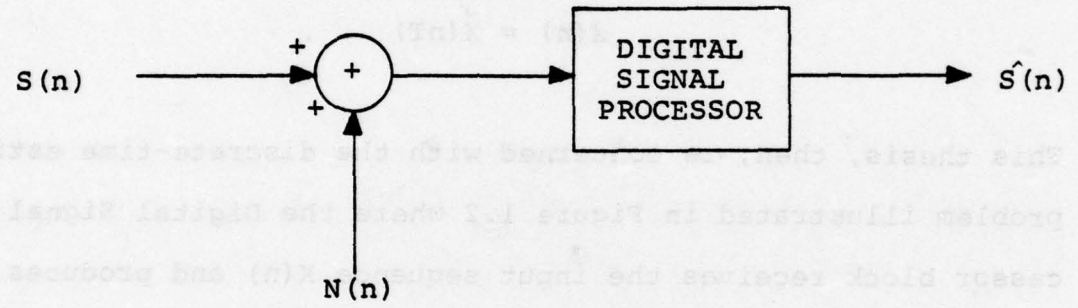


Figure 1.2 Discrete Time Estimation Problem

$$0 \leq n_1 \leq n \leq n_2 < \infty$$

and the signal estimate sequence may be expressed as the linear convolution of the data X(n) with the unit sample response h(n). That is:

$$\hat{S}(n) = \sum_{i=n_1}^{n_2} h(i)\, X(n-i) \quad . \tag{1.4}$$

The estimate $\hat{S}(n)$ can be represented in a more compact form by introducing the following vector notation:

$$\underline{h} = [h(n_1),\ h(n_1+1) \ldots h(n_2)]^T \tag{1.5}$$

$$\underline{X}(n-n_1) = [X(n-n_1), X(n-n_1-1), \ldots X(n-n_2)]^T \quad . \tag{1.6}$$

With this notation, $\hat{S}(n)$ may be expressed as the scalar product:

$$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h} = \underline{h}^T\, \underline{X}(n-n_1) \quad . \tag{1.7}$$

It is further assumed here that the filter is implemented in such a fashion so as to allow h(n) to be specified for each value of n independently. Figure 1.3 gives a conceptual block diagram of the digital estimation process considered in this thesis. Such a system is often referred to as a transversal filter or a weighted tap-delay line.

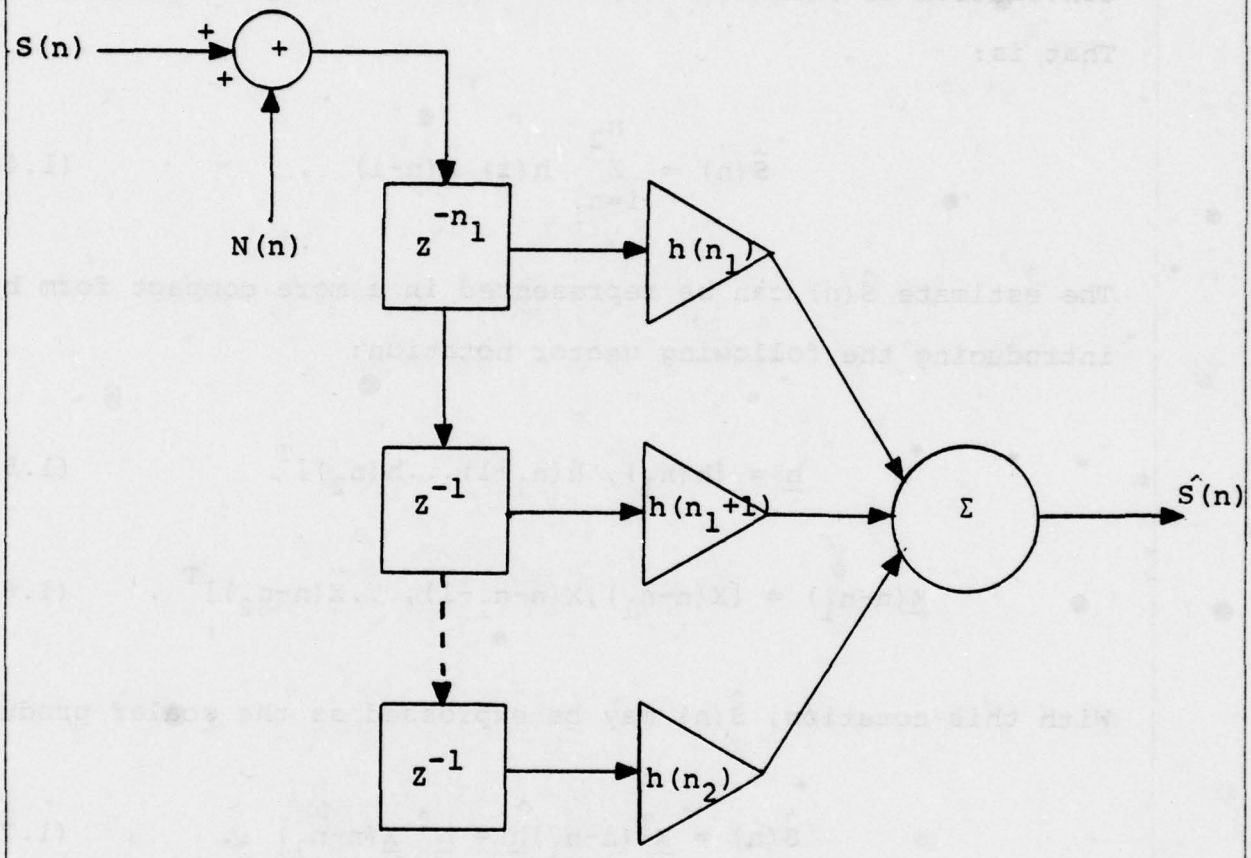


Figure 1.3 Transversal Filter for Estimation.

1.3 Mean-Square Error and Optimum Estimation

In order to determine how closely the signal estimate $\hat{S}(n)$ approximates the true signal sequence $S(n)$, some measure of the accuracy of the estimate is needed. The measure most commonly used is the mean-square error (MSE) given by:

$$E[e^2(n)] = E[(S(n)-\hat{S}(n))^2] \qquad (1.8)$$

where the notation $E[\cdot]$ denotes the expected value operator. Within a class of estimators, an estimate $\hat{S}(n)$ is said to be optimum in the sense of minimum MSE in $E[e^2(n)]$ for the estimate is less than or equal to $E[e^2(n)]$ for any other estimates within the class.

1.4 Solution for Minimum Mean-Square Error

If the class of estimators considered is that given by (1.4) or (1.7), then the problem to be solved is to find the unit sample response $h(n)$ of the filter so that (1.8) is minimized. While the solution of this problem is well known and results in the discrete form of the Wiener Filter [1], the development in this section is included for completeness and will serve as a starting point for the discussion of real-time adaptive linear estimation algorithms.

For each value of n, the signal sequence sample S(n) is to be estimated by a linear combination of $N = (n_2-n_1+1)$ samples of the data sequence X(n) as seen in (1.4). The N sample-values of the unit sample response h(n) are to be chosen so as to minimize the resulting mean-square error given by (1.8). This optimum solution may be obtained by applying the orthogonality principle which states that the constants h(n) which minimize (1.8) must be such that the estimation error is orthogonal to the data [2]. That is the vector identity:

$$E[(S(n)-\hat{S}(n))\ \underline{X}(n-n_1)] = \underline{0} \qquad (1.9)$$

must be satisfied. Substituting (1.7) into (1.9) yields:

$$E[(S(n)-\underline{X}^T(n-n_1)\underline{h})\underline{X}(n-n_1)] = \underline{0}$$

or:

$$E[S(n)\underline{X}(n-n_1) - \underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}] = \underline{0} \ .$$

By the linearity of the expected value operator [3]:

$$E[S(n)\underline{X}(n-n_1)] - E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h} = 0$$

or:

$$E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h} = E[S(n)\underline{X}(n-n_1)] \quad . \qquad (1.10)$$

With the following definitions, (1.10) can be stated in the familiar matrix form:

$\Phi_{XX}(n-n_1,n-n_1)=E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]$ = Input Autocorrelation Matrix (NxN)

$\underline{R}_{SX}(n,n-n_1)=E[S(n)\underline{X}(n-n_1)]$ = Cross-correlation vector between signal and data sequences (Nx1) .

For convenience, the time index will be dropped in the above definitions and unless specifically stated otherwise it will be implied that:

$$\Phi_{XX} = \Phi_{XX}(n-n_1,n-n_1) \qquad (1.11a)$$

$$\underline{R}_{SX} = \underline{R}_{SX}(n,n-n_1) \quad . \qquad (1.12a)$$

With the above definitions, (1.10) becomes:

$$\Phi_{XX}\underline{h} = \underline{R}_{SX} \quad . \qquad (1.13)$$

If the above system has a unique solution then Φ_{XX} is non-singular and thus has an inverse. Then the optimum unit sample response h*(n) is given by the familiar formula:

$$\underline{h}^* = \Phi_{XX}^{-1}\underline{R}_{SX} \quad . \qquad (1.14)$$

It remains to be shown that Φ_{XX} is invertable, or equivalently that the mean-square error given by (1.8) has a unique minimum point. If (1.8) is expanded using the linearity property of the expected value operator, the following equalities are obtained:

$$E[e^2(n)] = E[S^2(n)-2S(n)\hat{S}(n)+\hat{S}^2(n)]$$

$$E[e^2(n)] = E[S^2(n)]-2E[S(n)\hat{S}(n)]+E[\hat{S}^2(n)] \quad . \qquad (1.15)$$

Substitution of (1.7) into (1.15) yields:

$$E[e^2(n)] = E[S^2(n)]-2\underline{h}^T E[S(n)\underline{X}(n-n_1)]+\underline{h}^T E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h}$$

and by applying the definitions (1.11a) and (1.12a) the following expression for the mean-square error is obtained:

$$E[e^2(n)] = E[S^2(n)]-2\underline{h}^T\underline{R}_{SX}+\underline{h}^T\Phi_{XX}\underline{h} \quad . \qquad (1.16)$$

The above expression for the mean-square error is quadratic in the terms of the unit sample response h(n) and can be shown to have a unique minimum point if the following necessary and sufficient conditions are met [4]:

1. The gradient of $E[e^2(n)]$ with respect to $\underline{h}$ is equal to zero.
2. The Hessian matrix of $E[e^2(n)]$ with respect to $\underline{h}$ is positive definite.

Applying these conditions to (1.16) yields the following requirements:

1. $2(\Phi_{XX}\underline{h}-\underline{R}_{SX}) = \underline{0}$.
2. $\underline{Y}^T\Phi_{XX}\underline{Y} > 0 \; \forall \; \underline{Y} \in R^N$ (i.e. Φ_{XX} is positive definite).

If Φ_{XX} is positive definite, it is invertable and thus $\underline{h}$ given by (1.14) will satisfy condition (1). Therefore it remains to be shown only that Φ_{XX} is positive definite.

Theorem 1.1. The matrix $\Phi_{XX} = E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]$ is positive definite provided that $P[\underline{X}(n-n_1)]$ is not a degenerate distribution [5].

Proof: Let $\underline{Y}$ be an arbitrary N-vector and define the scalar random variable:

$$a = \underline{Y}^T\underline{X}(n-n_1) = \underline{X}^T(n-n_1)\underline{Y} \qquad (1.17)$$

consider:

$$\text{VAR}[a] = E[a^2]-(E[a])^2$$

$$\text{VAR}[a] = E[\underline{Y}^T\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{Y}]-(E[a])^2$$

$$\text{VAR}[a] = \underline{Y}^TE[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{Y}-(E[a])^2$$

$$\text{VAR}[a] = \underline{Y}^T\Phi_{XX}\underline{Y}-(E[a])^2 \quad .$$

Now:

$$\text{VAR}[a] \geq 0 \quad .$$

VAR[a] = 0 if and only if $a=a_1$, a constant, with $P[a=a_1]=1$. If this is true, then from (1.17) $P[\underline{X}(n-n_1)]$ must equal zero everywhere except on the linear variety given by:

$$a_1 = \underline{Y}^T\underline{X}(n-n_1)$$

which is an (N-1) dimensional subspace of R^N. Thus, $P[\underline{X}(n-n_1)]$ is degenerate on N-dimensional space. Therefore, if $P[\underline{X}(n-n_1)]$ is not degenerate, and since $(E[a])^2 \geq 0$:

$$\underline{Y}^T\Phi_{XX}\underline{Y}>0 \quad \forall \quad \underline{Y}\varepsilon R^N$$

and thus, it has been shown that Φ_{XX} is positive definite.

Since the necessary and sufficient conditions are satisfied, as has been shown, the mean-square error (1.8) has a unique minimum point $\underline{h}^*$. Therefore Φ_{XX} is invertable and the optimum unit sample response is given by (1.14).

1.5 Application of the Optimal Filter and Adaptive Filtering

It has been shown that, the optimal filter characterized by the unit sample response h*(n) given by (1.14) will yield the minimum mean-square error estimate of S(n) for the class of filters given by (1.4) when applied to the data sequence X(n). However, the compact closed-form solution of (1.14) is misleading when the difficulty of implementing such a filter is considered. Specifically, (1.14) implicitly assumes that the auto-

correlation matrix Φ_{XX} and the cross-correlation vector $\underline{R}_{SX}$ are known a priori so that the optimum unit sample response can be determined and the optimal filter implemented. However, in practice, the autocorrelation matrix and the cross-correlation vector may not be known exactly. In this case, a filter cannot be determined by the use of (1.14) and another approach to the problem must be chosen. One possible method which could be applied in this situation would be to determine a FIR digital filter which would in some sense approximate the optimal filter.

One approach to obtaining a filter is to estimate the input autocorrelation matrix Φ_{XX} and the cross-correlation vector $\underline{R}_{SX}$ from the input data sequence X(n) and to use these estimates to find a unit sample response:

$$\underline{h} = \hat{\Phi}_{XX}^{-1} \hat{\underline{R}}_{SX} \tag{1.18}$$

which can be considered as an example of an adaptive filter. That is, the unit sample response of the filter is determined from estimates of the input statistics based on the data which is to be filtered. If the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{SX}$ and the unit sample response $\underline{h}$ are updated continually by some algorithm based on the old estimates and the newly received data, then the adaptation may be considered to be recursive. The remainder of this thesis is concerned with algorithms which attempt to approximate the minimum mean-square error criterion by the recursive adaptation of the filter unit sample response based on estimates of the input statistics when the autocorrelation matrix and cross-correlation vector are not known a priori.

Examination of (1.18) reveals that estimates of the auto-correlation matrix of the input data and the cross-correlation vector of the signal with the input data are needed for the adaptive processing. Since no a priori knowledge of the signal is assumed, a form of (1.13) in which only correlations of the data occur is desirable. Substituting (1.2) into (1.13) yields:

$$\Phi_{XX}\underline{h} = E[(X(n)-N(n))\underline{X}(n-n_1)] \quad .$$

By the linearity of the expected value operator:

$$\Phi_{XX}\underline{h} = E[X(n)\underline{X}(n-n_1)]-E[N(n)\underline{X}(n-n_1)] \quad . \qquad (1.19)$$

Now, since $\underline{X}(n-n_1)=\underline{S}(n-n_1)+\underline{N}(n-n_1)$, (1.19) becomes:

$$\Phi_{XX}\underline{h} = E[X(n)\underline{X}(n-n_1)]-E[N(n)\underline{S}(n-n_1)]-E[N(n)\underline{N}(n-n_1)]. \quad (1.20)$$

At this point, two simplifying assumptions will be made. First, it is assumed that the signal sequence S(n) and the noise process N(n) are orthogonal. Secondly, the assumption is made that the delay n_1 in the unit sample response is such that noise samples which are separated by n_1 or more samples in the sequence are also orthogonal. These two assumptions imply respectively that:

$$1.\quad E[N(n)\underline{S}(n-n_1)] = \underline{0} \tag{1.21}$$

$$2.\quad E[N(n)\underline{N}(n-n_1)] = \underline{0} \tag{1.22}$$

Thus, (1.20) becomes:

$$\Phi_{XX}\underline{h} = E[X(n)\underline{X}(n-n_1)] \quad . \tag{1.23}$$

With the following definition, the alternate form of the optimal solution given assumptions (1.21) and (1.22) is obtained.

$$\underline{R}_{XX}(n,n-n_1) = E[X(n)\underline{X}(n-n_1)] = \text{Autocorrelation vector of the input data (NX1).} \tag{1.24}$$

Again, the time index will be dropped for convenience and it will be implied, unless specifically stated otherwise that:

$$\underline{R}_{XX} = \underline{R}_{XX}(n,n-1) \quad . \tag{1.24a}$$

With the above definition, (1.23) becomes:

$$\Phi_{XX}\underline{h} = \underline{R}_{XX} \quad . \tag{1.25}$$

And the optimal unit sample response is given by:

$$\underline{h}^* = \Phi_{XX}^{-1}\,\underline{R}_{XX} \quad . \tag{1.26}$$

The above form for the optimal solution involves only correlations of the input data. Thus, applying assumptions (1.21) and (1.22) has eliminated the need for knowledge of the cross-correlation between the signal and the data. The remaining terms can be more easily estimated from the data.

1.6 Separation of Signals Based on Bandwidth

Assumptions (1.21) and (1.22) allow the optimum unit sample response to be written in the more tractable form of (1.26). However, these assumptions and the inclusion of the delay n_1 in the unit sample response place restrictions on the conditions under which (1.26) will yield a useful solution. A consideration of these restrictions leads to an important property of the estimator which is not immediately evident in (1.26) but which must be considered if useful results are to be obtained.

If N(n) or S(n) is a zero-mean process, the orthogonality condition of (1.21) is equivalent to the requirement that the signal and the noise be uncorrelated. Similarly, if N(n) has zero mean, (1.22) will be satisfied if noise samples which are separated by n_1 or more samples are uncorrelated. In this case, the minimum value of n_1 needed to satisfy (1.22) is often referred to as the required "decorrelation delay" [6]. Since S(n) and N(n) are orthogonal, (1.26) can be rewritten as:

$$\underline{h}^* = \Phi_{XX}^{-1}(\underline{R}_{SS}+\underline{R}_{NN}) = \Phi_{XX}^{-1}(E[S(n)\underline{S}(n-n_1)]+E[N(n)\underline{N}(n-n_1)]). \quad (1.27)$$

Applying (1.22) to (1.27) yields:

$$\underline{h}^* = \Phi_{XX}^{-1} E[S(n)\underline{S}(n-n_1)] \quad . \tag{1.28}$$

Examination of (1.28) reveals that if n_1 is also sufficient to "decorrelate" the signal, then (1.26) yields only the trivial solution:

$$\underline{h}^* = \underline{0} \quad .$$

Thus, the solution given by (1.26) is only useful if the signal is more correlated, or coherent, than the noise. That is, the signal autocorrelation function must be significantly non-zero for larger delays than is the autocorrelation function of the noise. Since the power spectral density and the autocorrelation function are related by the Fourier Transform, bandwidth and length of the autocorrelation delay are inversely related. Thus, a process with a relatively narrow bandwidth will have an autocorrelation function which is significantly non-zero for larger delays than the autocorrelation function of a wideband signal. Therefore, the solution given by (1.26) is only useful for the case where the process S(n) and N(n) have significantly different bandwidths.

Consideration of assumption (1.22) and expression (1.28) reveals that the signal estimate given by:

$$\hat{S}(n) = \underline{h}^{*T}\underline{X}(n-n_1) \tag{1.29}$$

where $\underline{h}^*$ is given by (1.26) or (1.28), is only valid if a value of n_1 can be found which will decorrelate the noise without decorrelating the signal. Thus, from the preceeding discussion it is apparent that S(n) must have a relatively narrow bandwidth compared to that of N(n). If this is the case, an appropriate value of n_1 can be found and the signal can be estimated using the system shown in Fig. 1.4 where the unit sample response of the causal linear digital filter is given by (1.26).

Conversely, if the signal has a wide bandwidth compared to that of N(n), a value of n_1 which decorrelates the noise and not the signal cannot be found. In this case, one approach which has been suggested is to estimate the noise and to form the signal estimate by a noise-cancelling procedure [7]:

$$\hat{S}(n) = X(n)-\hat{N}(n) \tag{1.30}$$

where the noise estimate is given by:

$$\hat{N}(n) = \underline{h}_N^T \underline{X}(n-n_1) = \underline{X}^T(n-n_1)\underline{h}_N \quad . \tag{1.31}$$

It has been assumed in the above expression that n_1 has been chosen to decorrelate the wideband signal but not the narrow-band noise. That is, it is assumed that:

$$E[S(n)\underline{S}(n-n_1)] = \underline{0} \tag{1.32}$$

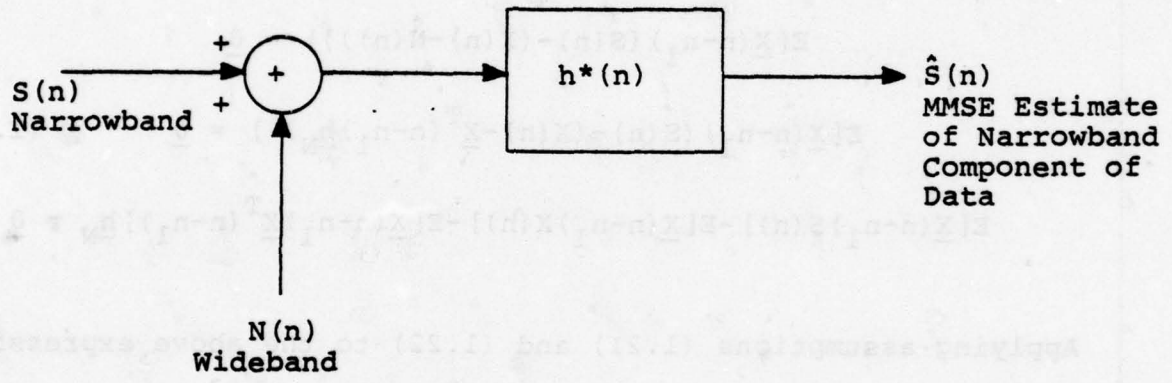


Figure 1.4 Estimator for Narrowband Signal in the Presence of a Wideband Additive Noise Process.

and also that S(n) and N(n) are orthogonal, or that (1.21) holds. Figure 1.5 gives a block diagram of a noise-cancelling system for the estimation of a wideband signal in the presence of narrowband noise.

The unit sample response $h_N(n)$ must be determined so that $\hat{S}(n)$ given by (1.30) is the minimum mean-square estimate of S(n). Applying the orthogonality principle to the estimate in (1.30) yields:

$$E[\underline{X}(n-n_1)(S(n)-(X(n)-\hat{N}(n)))] = \underline{0}$$

$$E[\underline{X}(n-n_1)(S(n)-(X(n)-\underline{X}^T(n-n_1)\underline{h}_N))] = \underline{0} \qquad (1.33)$$

$$E[\underline{X}(n-n_1)S(n)]-E[\underline{X}(n-n_1)X(n)]-E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h}_N = \underline{0} \quad .$$

Applying assumptions (1.21) and (1.22) to the above expression gives:

$$E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h}_N = E[X(n)\underline{X}(n-n_1)] \quad .$$

Finally, applying the definitions (1.6) and (1.24) yields:

$$\Phi_{XX}\underline{h}_N = \underline{R}_{XX} \quad .$$

The optimum unit sample response is then given by:

$$\underline{h}_N^* = \Phi_{XX}^{-1}\underline{R}_{XX} \quad . \qquad (1.34)$$

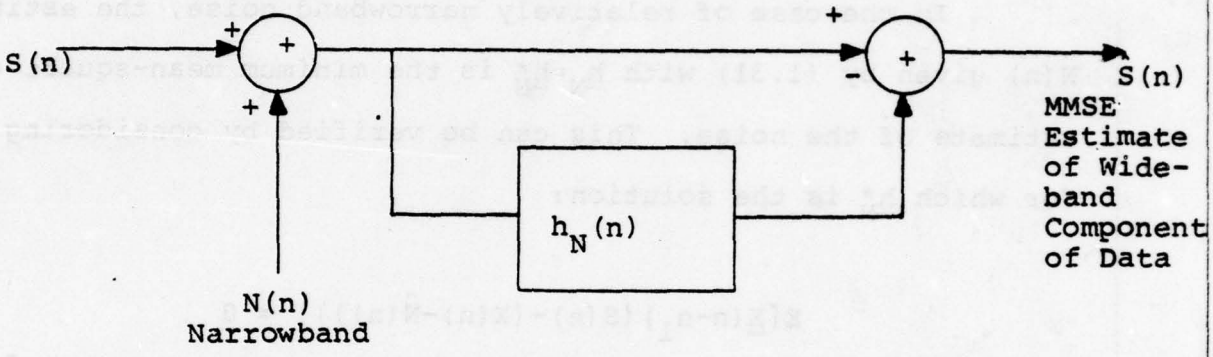


Figure 1.5 Estimator for a Wideband Signal in the Presence of a Narrowband Additive Noise Processor.

Comparison of (1.34) and (1.26) reveals that the same optimum unit sample response is required regardless of whether the signal or the noise is relatively narrowband. That is:

$$\underline{h}_N^* = \underline{h}^* \quad . \tag{1.35}$$

In the case of relatively narrowband noise, the estimate $\hat{N}(n)$ given by (1.31) with $\underline{h}_N = \underline{h}_N^*$ is the minimum mean-square error estimate of the noise. This can be verified by considering (1.33) for which $\underline{h}_N^*$ is the solution:

$$\begin{aligned} E[\underline{X}(n-n_1)(S(n)-(X(n)-\hat{N}(n)))] &= \underline{0} \\ E[\underline{X}(n-n_1)(\hat{N}(n)-N(n))] &= \underline{0} \\ E[\underline{X}(n-n_1)(N(n)-\hat{N}(n))] &= \underline{0} \quad . \end{aligned} \tag{1.36}$$

Now, (1.36) is simply the statement of the orthogonality principle applied to the noise estimate $\hat{N}(n)$. Therefore, $\hat{N}(n)$ is the minimum MSE estimate of $N(n)$ for the class of estimators considered. Thus, it is obvious that when a causal, linear digital filter with a unit-sample response given by (1.26) is applied to a data sequence composed of relatively wideband and relatively narrowband components, the filter output given by:

$$\hat{X}_{NB}(n) = \underline{h}^{*T}\underline{X}(n-n_1) \tag{1.37}$$

is the minimum MSE estimate of the narrowband components of the data while:

$$\hat{X}_{WB}(n) = X(n) - \hat{X}_{NB}(n) \qquad (1.38)$$

is the minimum MSE estimate of the wideband components of the data. Expressions (1.37) and (1.38) hold provided that the decorrelation delay n_1 has been chosen so as to decorrelate the wideband components but not the narrowband components. Fig. 1.6 is a block diagram of a system which combines the estimation of the narrowband and wideband components of the data. This system, which attempts to discriminate between two processes on the basis of bandwidth is an example of a Spectral Line Enhancer [8].

1.7 Algorithms for Approximation of the Optimal Solution

The Spectral Line Enhancer of Fig. 1.6 may be viewed as a generalized estimator for signal and noise processes which differ in bandwidth given that the restrictions discussed above are considered. The remaining task, then, is to obtain an approximation h(n) to h*(n). Historically, there are two basic methods for obtaining such an approximation. The first involves the collection of a block of input data, the estimation of Φ_{XX} and $\underline{R}_{XX}$ from this block of data and the solution of the linear system of equations (1.25) to obtain h(n) which can then be used to filter the block of data. Methods proposed for the solution of (1.25) include Triangularization [9], Gaussian

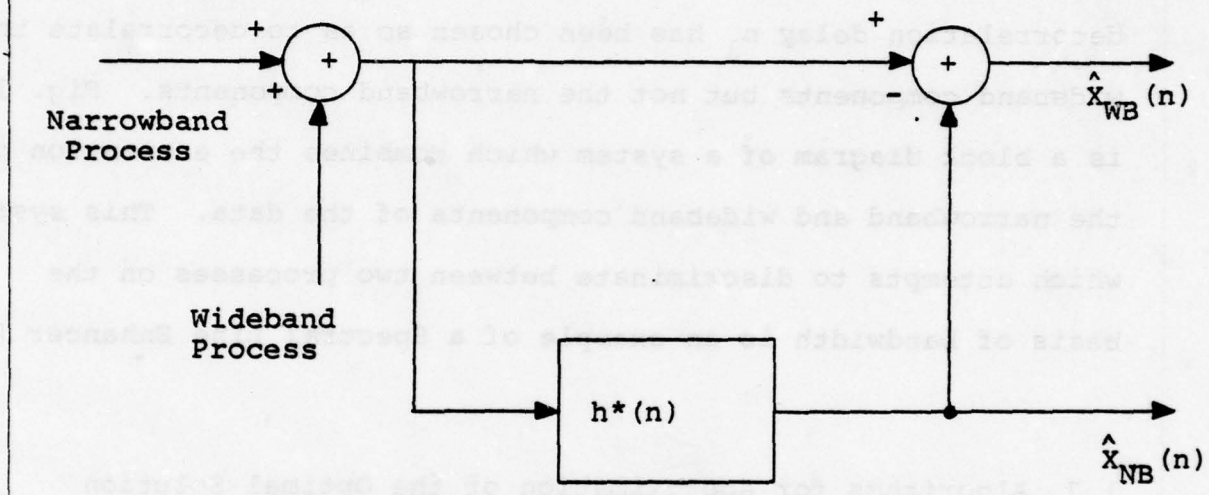


Figure 1.6 Spectral Line Enhancer.

Elimination [10], Steepest Descent [11], Conjugate Gradient [11], and Levinson-Robinson [12]. The second basic method is the use of an adaptive recursive algorithm such as the LMS Gradient Method [13]. Accumulation of data and direct solution of (1.25) may give a better approximation to h(n) than an adaptive recursive algorithm. However, such a method requires considerable computational effort compared to that needed by an algorithm such as the LMS Gradient Method. The approach taken in this thesis lies somewhere between the two basic methods both in terms of computational effort and performance. In this new approach, the received data samples are used to update the estimates of Φ_{XX} and $\underline{R}_{XX}$ as in the first approach. However, rather than solve (1.25) directly, one step of an iterative method such as Jacobi Relaxation or Successive Over-Relaxation is used to update h(n) based on the previous approximation and the updated estimates of Φ_{XX} and $\underline{R}_{XX}$. The conceptual flow diagram of Fig. 1.7 illustrates the algorithm for the adaptation of the unit-sample response. The remainder of this thesis is concerned with the development of Adaptive Fixed-Point Iteration Algorithms based on Jacobi Relaxation and the Successive Over-Relaxation methods and with comparisons between these methods and the LMS Gradient Algorithm. The convergence properties of the three methods are considered and theoretical bounds on their rates of convergence are derived. In addition, a non-iterative approach using Levinson's Algorithm is also considered in which estimates of Φ_{XX} and $\underline{R}_{XX}$ are updated at each sample time n, but the unit-sample response is updated

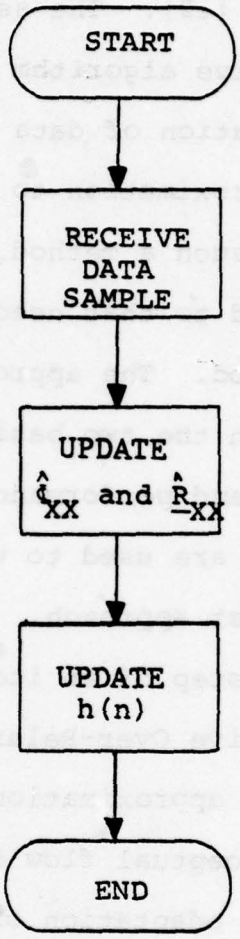


Figure 1.7 Conceptual Flow Diagram for Adaptive Fixed-Point Iteration.

at regular intervals rather than at each sample. Results of extensive computer simulations of the four algorithms are offered in support of the theoretical results.

CHAPTER II

THE LMS GRADIENT ALGORITHM

The LMS Gradient Method proposed by Widrow [13] is a widely used adaptive recursive algorithm for the approximate solution of the discrete Wiener-Hopf equation. The LMS Gradient Algorithm is based on the Steepest Descent Method for the minimization of a non-linear function of several variables. The applicability of the method of Steepest Descent can be seen by considering the mean-square error as a function of the terms of the unit-sample response, h(n). In Chapter I, it was shown that the MSE was a quadratic function of the sample values of the unit-sample response. Specifically, it was shown that:

$$E[e^2(n)] = E[S^2(n)] - 2\underline{h}^T\underline{R}_{SX} + \underline{h}^T\Phi_{XX}\underline{h} \qquad (2.1)$$

where the following definitions apply:

$$\underline{h} = [h(n_1), h(n_1+1) \ldots h(n_2)]^T \qquad (2.2)$$

$$\underline{R}_{SX} = E[S(n)\underline{X}(n-n_1)] \qquad (2.3)$$

$$\Phi_{XX} = E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)] \quad . \qquad (2.4)$$

It was also shown in Chapter I that Φ_{xx} was positive definite and therefore that $E[e^2(n)]$ has a unique minimum point $\underline{h}^*$. Thus, since $E[e^2(n)]$ is quadratic and has a unique minimum it is strictly convex. Therefore, given any two N-vectors $\underline{h}_m$ and $\underline{h}_{m+1}$, if

$$E[e^2(n)|\underline{h}_m] > E[e^2(n)|\underline{h}_{m+1}] \tag{2.5}$$

then:

$$(\underline{h}_m-\underline{h}^*)^T(\underline{h}_m-\underline{h}^*) > (\underline{h}_{m+1}-\underline{h}^*)^T(\underline{h}_{m+1}-\underline{h}^*) \tag{2.6}$$

where the notation $E[e^2(n)|\underline{h}_m]$ is used to indicate the value of the MSE computed using (2.1) with $\underline{h}$ replaced by $\underline{h}_m$. Therefore, if the MSE with $\underline{h}=\underline{h}_{m+1}$ is less than the MSE with $\underline{h}=\underline{h}_m$, then the distance from $\underline{h}_{m+1}$ to the optimal solution $\underline{h}^*$ is less than the distance from $\underline{h}_m$ to $\underline{h}^*$. This implies that if an N-vector $\underline{h}_m$ is given and a direction vector $\underline{d}_m \varepsilon R^N$ is chosen so that an N-vector $\underline{h}_{m+1}$ which is a distance μ_m from $\underline{h}_m$ along $\underline{d}_m$ satisfies the inequality given by (2.5) then $\underline{h}_{m+1}$ is closer to $\underline{h}^*$ than is $\underline{h}_m$. Such a direction vector is referred to as a descent direction at $\underline{h}_m$. Now, the N-vector $\underline{h}_{m+1}$ can be expressed as:

$$\underline{h}_{m+1} = \underline{h}_m+\mu_m\underline{d}_m \quad . \tag{2.7}$$

From the preceeding discussion, it is evident that if Equation (2.7) is applied recursively from an arbitrary starting point, a sequence of vectors $\underline{h}_m$ which converges uniformly to $\underline{h}$ will be

generated provided that μ_m and $\underline{d}_m$ are chosen properly. Hence, (2.7) may be considered as a possible method for the solution of the discrete Wiener-Hopf equation given by (1.13).

2.1 Steepest Descent Algorithm

The recursion equation (2.7) is incomplete as an algorithm unless methods for choosing $\underline{d}_m$ and μ_m are specified. Locally, a function of N variables decreases most rapidly in the direction of the negative gradient. Therefore, the Method of Steepest Descent uses the negative gradient evaluated at $\underline{h}_m$ as the descent direction $\underline{d}_m$. Thus, (2.7) becomes:

$$\underline{h}_{m+1} = \underline{h}_m - \mu_m \nabla E[e^2(n)|\underline{h}_m] \tag{2.8}$$

where:

$$\nabla[e^2(n)|\underline{h}_m] = 2\Phi_{XX}\underline{h}_m - 2\underline{R}_{SX} \quad .$$

Thus, for the quadratic problem considered, (2.8) can be expressed as:

$$\underline{h}_{m+1} = \underline{h}_m - 2\mu_m(\Phi_{XX}\underline{h}_m - \underline{R}_{SX}) \quad . \tag{2.9}$$

The parameter μ_m is chosen so as to minimize the MSE in the direction of the negative gradient at $\underline{h}_m$. That is, μ_m is chosen to minimize:

$$E[e^2(n)|\underline{h}_m]-2\mu_m(\Phi_{XX}\underline{h}_m-\underline{R}_{SX}) \quad . \tag{2.10}$$

With μ_m chosen to minimize the expression in (2.10), Equation (2.9) defines the Steepest Descent Algorithm for this problem.

The Steepest Descent Algorithm, in the form discussed above, is absolutely convergent for the problem considered due to the fact that the MSE function is strictly convex [4]. However, the parameter μ_m must be computed at each iteration step adding to the computational complexity of the algorithm. A modification of the Steepest Descent Algorithm uses a fixed value for the parameter μ_m for all iteration steps. The recursion relation then becomes:

$$\underline{h}_{m+1} = \underline{h}_m-2\mu(\Phi_{XX}\underline{h}_m-\underline{R}_{SX}) \quad . \tag{2.11}$$

This method, sometimes referred to as the Method of Simultaneous Displacements, is the algorithm upon which the LMS Gradient procedure is based. As will be shown later, the Method of Simultaneous Displacements converges for the quadratic problem considered provided that certain restrictions are placed on the value of the parameter μ [14].

The algorithm given by (2.11) defines a method which will converge to the unit-sample response h(n) which minimizes the MSE given by (2.1). However, two problems with this approach are immediately evident. First, as was mentioned previously, the autocorrelation matrix Φ_{XX} and the cross-correlation vector $\underline{R}_{SX}$ are generally unknown in practical problems. Thus, these

statistics must be estimated from the input data in order for (2.11) to be a useful algorithm. A second problem can be seen by noting that at each sample time n a number of iterations of (2.11) denoted by the index m may be required for the Simultaneous Displacement Algorithm to converge to the optimal solution. Also, some method for determining when the algorithm has converged to within a desired distance from the optimal solution must be given to halt the iteration process. The need for a stopping criterion and the number of iterations required may prohibit the use of the algorithm given by (2.11) in real-time processing. This problem may be avoided if it is assumed that x(n) is a stationary process so that Φ_{XX} and $\underline{R}_{SX}$ are not functions of the sample time n. In this case, a possible approach would be to carry out a single step of the iteration defined by (2.11) at each sample time n. The algorithm can then be written as:

$$\underline{h}_{n+1} = \underline{h}_n - 2\mu(\Phi_{XX}\underline{h}_n - \underline{R}_{SX}) \quad . \tag{2.12}$$

The algorithm given by (2.12) converges as the sample time n increases provided that certain restrictions are placed on the value of μ. A stopping criterion is not used since in practice Φ_{XX} and $\underline{R}_{XX}$ must be replaced by estimates for adaptive processing and, hopefully, these estimates will improve as more data is obtained. In addition, the continual updating of h(n) using (2.12) allows the algorithm to adapt to slowly time-varying processes X(n) provided that the time variation of Φ_{XX} and $\underline{R}_{XX}$

can be reflected in their estimates.

The remaining feature needed to make (2.12) into an adaptive recursive algorithm is some method for estimating the statistics of the input process. Specifically, knowledge is required about the gradient of the MSE given by:

$$\nabla E[e^2(n)|\underline{h}_n] = 2(\Phi_{XX}\underline{h}_n - \underline{R}_{SX}) \quad . \tag{2.13}$$

Substituting the definitions of Φ_{XX} and $\underline{R}_{SX}$ given by (1.11) and (1.12) into (2.13) yields:

$$\nabla E[e^2(n)|\underline{h}_n] = 2E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h}_n - 2E[S(n)\underline{X}(n-n_1)] \quad .$$

Now, since $\underline{h}_n$ is known:

$$\nabla E[e^2(n)|\underline{h}_n] = 2E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n] - 2E[S(n)\underline{X}(n-n_1)] \quad .$$

But,

$$S(n) = \underline{X}^T(n-n_1)\underline{h}_n \quad .$$

Therefore:

$$\nabla E[e^2(n)|\underline{h}_n] = 2E[X(n-n_1)\hat{S}(n)] - 2E[S(n)\underline{X}(n-n_1)] \quad . \tag{2.14}$$

Assumptions (1.21) and (1.22) state respectively that the signal sequence S(n) and the noise process N(n) are orthogonal and that noise samples separated by n_1 or more samples are also orthogonal. Noting that the signal S(n) may be written as:

$$S(n) = X(n)-N(n)$$

and by applying assumptions (1.21) and (1.22), Equation (2.14) can be rewritten as:

$$\nabla E[e^2(n)|\underline{h}_n] = 2E[\underline{X}(n-n_1)\hat{S}(n)]-2E[X(n)\underline{X}(n-n_1)]$$

or:

$$\nabla E[e^2(n)|\underline{h}_n] = 2E[(\hat{S}(n)-X(n))\underline{X}(n-n_1)] \quad . \qquad (2.15)$$

Therefore, using (2.15) the iteration algorithm (2.12) can be rewritten as:

$$\underline{h}_{n+1} = \underline{h}_n+2\mu\ E[(X(n)-\hat{S}(n))\underline{X}(n-n_1)] \quad . \qquad (2.16)$$

Thus, the problem of estimating the input statistics for the adaptive algorithm has been reduced to the estimation of the gradient of the MSE given by (2.15). It is obvious from (2.15) that the estimate of the gradient given by:

$$\hat{\nabla}E[e^2(n)|\underline{h}_n] = 2(\hat{S}(n)-X(n))\underline{X}(n-n_1) \qquad (2.17)$$

is an unbiased estimate. That is:

$$E[\hat{\nabla}E[e^2(n)|\underline{h}_n]] = E[2(S(n)-X(n))\underline{X}(n-n_1)] = \nabla E[e^2(n)|\underline{h}_n] \ .$$

Thus, substituting the estimate of the gradient given by (2.17) into (2.16) yields the adaptive recursive algorithm:

$$\underline{h}_{n+1} = \underline{h}_n + 2\mu(X(n)-\hat{S}(n))\underline{X}(n-n_1) \ . \qquad (2.18)$$

Equation (2.18) along with the filter output equation (1.7) defines the LMS Gradient Algorithm developed by Widrow [6], [7], [13].

It is obvious from (2.18) that the LMS Gradient Algorithm is an attractive approach to real-time adaptive estimation due to its simplicity. The number of operations required is on the order of the length N of the unit-sample response h(n). However, the simplicity of the algorithm must be weighed against its performance. Specifically, the convergence properties of the LMS Gradient Algorithm must be considered as well as the number of operations required for its implementation.

2.2 Convergence of the LMS Gradient Algorithm

Convergence properties are very important to consider in the evaluation of iterative algorithms. Specifically, it is desirable to know under what conditions the algorithm will converge to the optimal solution h*(n) and at what rate the

convergence will occur. Ideally, an algorithm should be absolutely convergent. That is, it should converge to the optimal solution under all conditions and with no restrictions. In addition the convergence should be rapid so that the optimal solution is found after as few iterations as possible. As mentioned previously, there are restrictions on the value of the parameter μ in Equation (2.18) which must not be violated if the algorithm is to converge. While already a part of the literature [6], these restrictions are derived in the next section for completeness. In addition, a bound on the rate of convergence of the LMS Gradient Algorithm is developed so that comparisons with the Adaptive Fixed-Point Iteration Algorithms to be introduced later can be made.

It is important to note at this point that in the discussion that follows it is assumed that X(n) is a stationary process. While the convergence properties of the LMS Gradient Algorithm have been explored for non-stationary inputs, closed-form solutions which would allow comparisons with other algorithms are difficult, if not impossible to obtain.

The LMS Gradient Algorithm as developed by Widrow [6], [7], [13], is completely described by (2.18) and the filter output equation (1.7). These expressions are given below in a slightly different form which will be referred to as the LMS Gradient Algorithm in the remainder of this discussion:

$$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n \tag{2.19}$$

$$\underline{h}_{n+1} = \underline{h}_n + 2\mu\varepsilon(n)\underline{X}(n-n_1) \tag{2.20}$$

$$\varepsilon(n) = X(n) - \hat{S}(n) \tag{2.21}$$

Equation (2.19) is identical to the filter output equation (1.7) while equations (2.20) and (2.21) may be combined to form the adaptive recursive algorithm given by (2.18).

To determine the restrictions on the value of μ under which the LMS Gradient Algorithm is convergent, it is necessary to consider the adaptation algorithm given by (2.18) which is repeated below.

$$\underline{h}_{n+1} = \underline{h}_n + 2\mu(X(n) - \hat{S}(n))\underline{X}(n-n_1) \quad . \tag{2.18}$$

Substituting (2.19) into the above expression yields:

$$\underline{h}_{n+1} = \underline{h}_n + 2\mu(X(n) - \underline{X}^T(n-n_1)\underline{h}_n)\underline{X}(n-n_1)$$

or:

$$\underline{h}_{n+1} = \underline{h}_n + 2\mu X(n)\underline{X}(n-n_1) - 2\mu\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n \quad .$$

Collecting terms on the right hand side gives:

$$\underline{h}_{n+1} = [I - 2\mu\underline{X}(n-n_1)\underline{X}^T(n-n_1)]\underline{h}_n + 2\mu X(n)\underline{X}(n-n_1) \quad . \tag{2.22}$$

Obviously, $\underline{h}_{n+1}$ defined by (2.22) is a random vector. Thus, "convergence" of $\underline{h}_{n+1}$ to $\underline{h}^*$ must be interpreted as convergence in the mean. That is, if the LMS Gradient Algorithm "converges", it is in the sense that $\underline{h}_n$ is an asymptotically unbiased estimator of the optimal unit-sample response $\underline{h}^*$. Therefore, taking the expected value of both sides of (2.22) gives the following development:

$$E[\underline{h}_{n+1}] = E[(I-2\mu\underline{X}(n-n_1)\underline{X}^T(n-n_1))\underline{h}_n] + 2\mu E[X(n)\underline{X}(n-n_1)]$$

$$E[\underline{h}_{n+1}] = E[\underline{h}_n] - 2\mu E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n] + 2\mu E[X(n)\underline{X}(n-n_1)] \quad (2.23)$$

$$E[\underline{h}_{n+1}] = (I-2\mu E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)|\underline{h}_n])E[\underline{h}_n] + 2\mu E[X(n)\underline{X}(n-n_1)]$$

where the notation:

$$E[\cdot|\cdot]$$

indicates the conditional expected value. Previous research has shown that if the value of μ is small, then the effect of the covariance between $\underline{X}(n-n_1)\underline{X}^T(n-n_1)$ and $\underline{h}_n$ is negligible [13]. Under this assumption, (2.23) can be rewritten as:

$$E[\underline{h}_{n+1}] = (I-2\mu E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)])E[\underline{h}_n] + 2\mu E[X(n)\underline{X}(n-n_1)] \quad .$$

Substituting definitions (1.6) and (1.24) into the above expression yields:

$$E[\underline{h}_{n+1}] = (I-2\mu\Phi_{XX})E[\underline{h}_n]+2\mu\underline{R}_{XX} \qquad (2.24)$$

Thus, under the assumption that μ is small, the convergence properties of the LMS Gradient Algorithm can be determined from a consideration of (2.24). Equation (2.24) is an example of fixed-point iteration with the iteration matrix:

$$B = (I-2\mu\Phi_{XX})$$

Therefore, the following theorem concerning convergence applies. A proof appears in Appendix A.

Theorem 2.1

A necessary and sufficient condition for the convergence of the fixed point iteration given by:

$$\underline{Y}_{n+1} = B\,\underline{Y}_n+\underline{d}$$

is that the magnitude of the eigenvalues of the iteration matrix B be less than one [15].

Applying Theorem 2.1 to Equation (2.24) it can be seen that the sequence $\underline{h}_n$ produced by the LMS Gradient Method will converge in the mean to $\underline{h}^*$ if and only if all the eigenvalues

of the matrix:

$$B = (I-2\mu\Phi_{XX})$$

are strictly less than one in absolute value. Now, let λ_i be an eigenvalue of Φ_{XX} corresponding to the eigenvector Ψ_i. Then:

$$(I-2\mu\Phi_{XX})\Psi_i = \Psi_i-2\mu\Phi_{XX}\Psi_i$$

$$(I-2\mu\Phi_{XX})\Psi_i = \Psi_i-2\mu\lambda_i\Psi_i \qquad (2.26)$$

$$(I-2\mu\Phi_{XX})\Psi_i = (1-2\mu\lambda_i)\Psi_i \quad .$$

From (2.26) it can be seen that $(1-2\mu\lambda_i)$ is an eigenvalue of $(I-2\mu\Phi_{XX})$ with corresponding eigenvector Ψ_i. Thus, by Theorem 2.1, the LMS Gradient Method will converge in the mean for any value of μ such that the following inequality is satisfied for all λ_i which are eigenvalues of Φ_{XX}:

$$|1-2\mu\lambda_i| < 1 \quad . \qquad (2.27)$$

It was shown in Chapter I that the matrix Φ_{XX} was positive definite. Therefore, all of its eigenvalues are greater than zero. Thus, (2.27) can be written as:

$$0 < 2\mu\lambda_i < 2 \quad . \qquad (2.28)$$

Since (2.28) must hold for all the eigenvalues of Φ_{XX}, μ must satisfy the following inequality if the LMS Gradient Method is to converge in the mean:

$$0 < \mu < \frac{1}{\lambda_{max}} \quad . \tag{2.29}$$

where: λ_{max} is the largest eigenvalue of the input autocorrelation matrix Φ_{XX}.

Equation (2.29) provides a criterion for the selection of the parameter μ to insure convergence in the mean of the LMS Gradient procedure. However, since no knowledge of the input autocorrelation matrix is assumed, (2.29) is of little practical value. In practice, μ is usually chosen to be a "small" positive number to insure that (2.29) is satisfied. However, as will be shown in the next section, the parameter μ directly effects the rate of convergence of the LMS Gradient Algorithm so that the choice of a very small value for μ may have an adverse effect on the rate of convergence.

2.3 Rate of Convergence of the LMS Gradient Algorithm

It has been shown that under certain conditions the sequence $\underline{h}_n$ produced by the LMS Gradient Algorithm converges in the mean to the optimal unit-sample response $\underline{h}^*$. That is, $\underline{h}_n$ is an asymptotically unbiased estimator of $\underline{h}^*$ provided the value of the parameter μ is such that (2.29) is satisfied.

Thus, it would appear obvious to consider the rate at which the convergence takes place; or, how "fast" does $E[\underline{h}_n]$ approach $\underline{h}^*$. However, since the desired result of the signal processing considered here is to produce an approximation to the minimum MSE signal estimate, it is of more interest to consider the rate at which the MSE of the LMS Gradient Algorithm's signal estimate approaches the minimum MSE produced by the optimal filter. The MSE for the LMS Gradient Algorithm is given by:

$$E[(S(n)-\hat{S}(n))^2] = E[S^2(n)]-2E[S(n)\hat{S}(n)] +E[\hat{S}^2(n)] .$$

Substituting for $\hat{S}(n)$ from (1.7) yields:

$$E[(S(n)-\hat{S}(n))^2]=E[S^2(n)]-2E[S(n)\underline{X}^T(n-n_1)\underline{h}_n]+E[\underline{h}_n^T\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n] \quad (2.30)$$

If the covariance between $\underline{X}(n-n_1)$ and $\underline{h}_n$ is neglected as well as the covariance of $h_n(i)$, $h_n(j)$, $X(n-n_1-i)$ and $X(n-n_1-j)$ then (2.30) can be rewritten as:

$$MSE(n)\approx E[S^2(n)]-2E[S(n)\underline{X}^T(n-n_1)]E[\underline{h}_n]+E[\underline{h}_n^T]E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]E[\underline{h}_n] \quad (2.31)$$

Justification for this assumption is offered in Appendix II. Applying definitions (1.11) and (1.12) to (2.31) yields:

$$MSE(n) = E[S^2(n)]-2\underline{R}_{SX}^TE[\underline{h}_n]+E[\underline{h}_n^T]\Phi_{XX}E[\underline{h}_n] \quad . \quad (2.32)$$

Equation (2.32) then gives an approximation to the mean-square error of the LMS Gradient procedure.

The MSE produced by the optimal filter $\underline{h}^*$ is given by:

$$MSE^*(n) = E[S^2(n)] - 2R_{SX}^T h^* + h^{*T} \Phi_{XX} \underline{h}^* \quad . \tag{2.33}$$

Thus, the difference between the minimum mean-square error MSE(n) and the estimate of the mean-square error of the LMS Gradient procedure given by (2.32) may be written as:

$$MSE(n) - MSE^*(n) = E[\underline{h}_n^T] \Phi_{XX} E[\underline{h}_n] - 2R_{SX}^T E[\underline{h}_n] - \underline{h}^{*T} \Phi_{XX} \underline{h}^* + 2R_{SX}^T \underline{h}^*$$

$$MSE(n) - MSE^*(n) = (E[\underline{h}_n] - \underline{h}^*)^T \Phi_{XX} (E[\underline{h}_n] + \underline{h}^*) - 2\underline{R}_{SX}^T (E[\underline{h}_n] - \underline{h}^*)$$

or:

$$MSE(n) - MSE^*(n) = (E[\underline{h}_n] - \underline{h}^*)^T (\Phi_{XX} E[\underline{h}_n] + \Phi_{XX} \underline{h}^* - 2\underline{R}_{SX}) \tag{2.34}$$

From (1.14) $\underline{R}_{SX}$ may be written as:

$$\underline{R}_{SX} = \Phi_{XX} \underline{h}^* \quad . \tag{2.35}$$

Thus, by substituting (2.35) into (2.34), (2.34) can be expressed as:

$$MSE(n) - MSE^*(n) = (E[\underline{h}_n] - \underline{h}^*)^T \Phi_{XX} (E[\underline{h}_n] - \underline{h}^*). \tag{2.36}$$

Since the process X(n) is assumed stationary, Φ_{XX} is a constant matrix. Furthermore, since Φ_{XX} is real, symmetric and positive definite it has distinct eigenvalues. Therefore, the iteration matrix given by (2.25) also has distinct eigenvalues. Thus, it is possible to consider the convergence of the LMS Gradient procedure in terms of the uncoupled variables:

$$\tilde{\underline{h}}_n = M^{-1}E[\underline{h}_n] = M^T E[\underline{h}_n] \tag{2.37a}$$

$$\tilde{\underline{h}}^* = M^{-1}\underline{h}^* = M^T\underline{h}^* \tag{2.37b}$$

where M is the modal matrix of the iteration matrix:

$$B = (I-2\mu\Phi_{XX}) \quad .$$

Thus, in terms of the uncoupled variables, the difference between the minimum MSE and the estimate of the MSE of the LMS Gradient Algorithm can be written as:

$$(\tilde{\underline{h}}_n - \tilde{\underline{h}}^*)^T M^T \Phi_{XX} M^T (\tilde{\underline{h}}_n - \tilde{\underline{h}}^*) \quad . \tag{2.38}$$

Recall from (2.26) that the eigenvectors of $(I-2\mu\Phi_{XX})$ are identical to those of the autocorrelation matrix Φ_{XX}. Therefore, M is also the modal matrix of Φ_{XX} and:

$$M^T\Phi_{XX}M = \Lambda \tag{2.41}$$

where Λ is the diagonal matrix containing the eigenvalues of Φ_{XX} along the diagonal. Thus, (2.38) may be expressed as:

$$MSE(n)-MSE^*(n) = (\tilde{\underline{h}}_n-\tilde{\underline{h}}^*)^T\Lambda(\tilde{\underline{h}}_n-\tilde{\underline{h}}^*) \quad . \qquad (2.42)$$

Since Λ is a diagonal matrix, (2.42) can be rewritten as:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i(\tilde{h}_n(i)-\tilde{h}^*(i))^2 \qquad (2.43)$$

where the λ_i are the eigenvalues of Φ_{XX} and $\tilde{h}_n(i)$ and $\tilde{h}^*(i)$ are the expected uncoupled unit-sample response of the LMS Gradient Algorithm and the uncoupled unit sample response of the optimal filter respectively. Now, recall from (2.24) that:

$$E[\underline{h}_{n+1}] = (I-2\mu\Phi_{XX})E[\underline{h}_n]+2\mu\underline{R}_{XX} \quad . \qquad (2.24)$$

Substituting the definition (2.37) into (2.24) yields the following iteration equation for the uncoupled variables:

$$\tilde{\underline{h}}_{n+1} = M^T(I-2\mu\Phi_{XX})M\tilde{\underline{h}}_n+2\mu M^T\underline{R}_{XX}$$

or:

$$\tilde{\underline{h}}_{n+1} = (I-2\mu\Lambda)\tilde{\underline{h}}_n+2\mu M^T\underline{R}_{XX} \quad . \qquad (2.44)$$

The optimal unit sample response vector $\underline{h}^*$ is a fixed point of the iteration equation (2.24) provided condition (2.29) on the value of the parameter μ is met. That is:

$$\underline{h}^* = (I-2\mu\Phi_{XX})\underline{h}^*+2\mu\underline{R}_{XX} \quad . \tag{2.45}$$

Thus, in terms of the uncoupled variables, (2.45) can be rewritten as:

$$\tilde{\underline{h}}^* = (I-2\mu\Lambda)\tilde{\underline{h}}^*+2\mu M^T\underline{R}_{XX} \quad . \tag{2.46}$$

Therefore, the difference between the uncoupled unit sample response of the LMS Gradient procedure and the optimal filter is given by:

$$(\tilde{\underline{h}}_n-\underline{h}^*) = (I-2\mu\Lambda)(\tilde{\underline{h}}_{n-1}\tilde{\underline{h}}^*) \quad . \tag{2.47}$$

The relationship in (2.47) may also be expressed in a non-recursive form as:

$$(\tilde{\underline{h}}_n-\tilde{\underline{h}}^*) = (I-2\mu\Lambda)^n(\tilde{\underline{h}}_0-\tilde{\underline{h}}^*) \tag{2.48}$$

where $\tilde{\underline{h}}_0$ is the initial uncoupled unit-sample response of the LMS Gradient Algorithm. Using (2.43) and (2.48) the difference between the approximate mean square error of the LMS Gradient procedure and the minimum MSE can be expressed as:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i(1-2\mu\lambda_i)^{2n}(\tilde{h}_0(i)-\tilde{h}^*(i))^2. \tag{2.49}$$

The relationship given in (2.49) is the fundamental result of this section. It places a bound on the rate of convergence of the LMS Gradient procedure. Note that this relation

again shows that for convergence of the algorithm (i.e. for convergence of MSE(n) to MSE*(n)):

$$(1-2\mu\lambda_i) < 1 \quad \forall \quad \lambda_i$$

This is consistent with the results derived earlier. Equation (2.49) gives a lower bound on the rate on convergence of the LMS Gradient Procedure in that convergence will proceed according to (2.49) only if precise knowledge of the gradient is used in the algorithm rather than the estimate. While the estimate of the gradient used in the LMS Gradient Algorithm is unbiased as discussed in Section 2.1.1, it does have finite variance. Thus, the LMS Gradient Algorithm will always produce a mean-square error which converges to the minimum at a rate which is slower than that indicated by (2.49). Also, it is emphasized that the results of this section are valid only for cases in which the autocorrelation matrix Φ_{XX} and the autocorrelation vector are independent of time. While the bound on the rate of convergence given by (2.49) is approximate and subject to several constraints, it is still useful for the comparison of the LMS Gradient Algorithm to the Adaptive Fixed-Point Iteration Algorithms discussed in the next Chapter. The relationship between the theoretical and actual convergence properties of the algorithm will be investigated via computer simulations in Chapter VI.

2.4 LMS Gradient Algorithm -- Another View

The LMS Gradient Algorithm developed by Widrow and discussed in the previous sections is based on the use of the Method of Steepest Descent for the minimization of the mean-square error function. Since the gradient of the mean-square error is unknown, an estimate of the gradient is used in the algorithm. The convergence of the LMS Gradient procedure's unit sample response to the optimal implies that the algorithm has somehow adapted to, or "learned" something about, the statistics of the data. However, it is difficult to see from the relationships which make up the algorithm how the estimates of the statistics are obtained. In this section, an algorithm will be proposed based on the orthogonality principle and direct estimates of the statistics. It will be shown that this algorithm is almost identical to the LMS Gradient Algorithm.

The orthogonality principle states that the signal estimate must be such that the estimation error is orthogonal to the data for minimum mean-square error. That is, the vector identity:

$$E[\underline{X}(n-n_1)(S(n)-\hat{S}(n))] = \underline{0} \qquad (1.9)$$

must be satisfied. Expanding this identity gives:

$$E[\underline{X}(n-n_1)S(n)] - E[\underline{X}(n-n_1)\hat{S}(n)] = \underline{0}$$

and, recognizing that $S(n)=X(n)-N(n)$ yields:

$$E[\underline{X}(n-n_1)X(n)]-E[\underline{X}(n-n_1)N(n)]-E[\underline{X}(n-n_1)\hat{S}(n)] = \underline{0}. \quad (2.50)$$

Now, applying the assumptions used in Chapter I that the signal and the noise are orthogonal and that noise samples separated by n_1 or more samples are also orthogonal allows (2.50) to be rewritten as:

$$E[\underline{X}(n-n_1)X(n)]-E[\underline{X}(n-n_1)\hat{S}(n)] = \underline{0} \quad . \quad (2.51)$$

If a priori knowledge of the statistics of the data was available (2.51) could be solved for the unit sample response. Now, suppose that direct estimates are made of the two expected value terms in (2.51) and that these estimates are based on the past data through X(n-1) and the past signal estimates through $\hat{S}(n)$. That is define two estimates $\underline{Y}_1$ and $\underline{Y}_2$ as follows:

$$\hat{\underline{Y}}_1 = \frac{1}{n} \sum_{\ell=0}^{n-1} \underline{X}(\ell-n_1)X(\ell) \quad (2.52)$$

$$\hat{\underline{Y}}_2 = \frac{1}{n+1} \sum_{\ell=0}^{n} \underline{X}(\ell-n_1)\hat{S}(\ell) \quad . \quad (2.53)$$

If the statistics of the data are stationary and if S(n) is the minimum mean-square error estimate then $\hat{\underline{Y}}_1$ and $\hat{\underline{Y}}_2$ are consistent estimates of the two expected value terms in (2.51). Thus, as n becomes large, the estimates (2.52) and (2.53) should come close to satisfying the equality of (2.51). That is:

$$\frac{1}{h}\sum_{\ell=0}^{n-1}\underline{X}(\ell-n_1)X(\ell) - \frac{1}{n+1}\sum_{\ell=0}^{n}\underline{X}(\ell-n_1)\hat{S}(\ell) \approx \underline{0}$$

or:

$$\underline{X}(n-n_1)\hat{S}(n) - \sum_{\ell=0}^{n-1}\underline{X}(\ell-n_1)\left(\frac{n+1}{n}X(\ell)-\hat{S}(\ell)\right) \approx \underline{0}. \quad (2.54)$$

With the past data and signal estimate samples available, (2.54) may be considered as a set of N linear equations in one unknown $\hat{S}(n)$. Thus, there may or may not be an estimate $\hat{S}(n)$ which will exactly satisfy (2.54). However, at least formally, each side of (2.54) may be pre-multiplied by $\underline{X}^T(n-n_1)$ and divided by the scalar $\underline{X}^T(n-n_1)\underline{X}(n-n_1)$. After this manipulation, (2.54) may be written as:

$$\hat{S}(n) = \frac{1}{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}\underline{X}^T(n-n_1)\sum_{\ell=0}^{n-1}\underline{X}(\ell-n_1)\left(\frac{n+1}{n}X(\ell)-\hat{S}(\ell)\right). \quad (2.55)$$

Now, suppose that n is large enough so that:

$$\frac{n+1}{n} \approx 1 \quad .$$

Then, (2.55) can be rewritten as:

$$\hat{S}(n) = \frac{1}{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}\underline{X}^T(n-n_1)\sum_{\ell=0}^{n-1}\underline{X}(\ell-n_1)(X(\ell)-\hat{S}(\ell)). \quad (2,56)$$

Equation (2.56) gives a relationship for a signal estimate based on an attempt to satisfy the orthogonality principle but substituting the direct estimates (2.52) and (2.53) for the expected value terms. However, (2.56) would be more attractive

if it were in a recursive form. The relationship (2.56) may be placed in a recursive form by defining the N-vector $\underline{h}_n$ as:

$$\underline{h}_n = \frac{1}{\underline{X}^T(n-n_1)\underline{X}(n-n_1)} \sum_{\ell=0}^{n-1} \underline{X}(\ell-n_1)(X(\ell)-\hat{S}(\ell)). \qquad (2.57)$$

Thus, (2.56) can be rewritten as:

$$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n = \sum_{i=n_1}^{n_2} h(i)X(n-i). \qquad (2.58)$$

Equation (2.58) is in the form of a finite impulse response digital filter as is used for estimation in the LMS Gradient procedure. The unit-sample response can be updated recursively by considering equation (2.57) and noting that:

$$\underline{h}_{n+1} = \frac{1}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)} \sum_{\ell=0}^{n} \underline{X}(\ell-n_1)(X(\ell)-\hat{S}(\ell))$$

or:

$$\underline{h}_{n+1} = \frac{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)}\underline{h}_n + \frac{1}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)}(X(n)-\hat{S}(n))\underline{X}(n-n_1) \qquad (2.59)$$

Note from (2.50) that the equation for updating the unit sample response in this algorithm is very similar to that used in the LMS Gradient Method (2.18) except that data dependent coefficients are used in the recursion relation. Now, assume that X(n) is a stationary process and consider the term:

$$\underline{X}^T(n-n_1)\underline{X}(n-n_1) = \sum_{i=n_1}^{n_2} x^2(n-i).$$

The terms $\underline{X}^T(n-n_1)\underline{X}(n-n_1)$ and $\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)$ are the squares of the norms of two N-vectors which differ in magnitude due to only one component. That is:

$$\underline{X}^T(n-n_1)\underline{X}(n-n_1) = \underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)+x^2(n+1-n_1)-\underline{x}^2(n+1-n_1)$$

or:

$$\frac{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)} = 1 + \frac{x^2(n+1-N-n_1)-x^2(n+1-n_1)}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)} . \quad (2.61)$$

Now, if the probability density function of $\underline{X}(n-n_1)$ is relatively concentrated near the center of gravity E $[\underline{X}(n-n_1)]$ of the distribution, then it can be shown that to a second order approximation:

$$E\left[\frac{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}{\underline{X}^T(n+1-n_1)\underline{X}^T(n+1-n_1)}\right] = 1 \quad (2.62)$$

and that the variance of this quotient is small if N is relatively large. Appendix C gives a proof of these statements. Since the mean of the term multiplying $\underline{h}_n$ in equation (2.59) is one and its variance is small for relatively large values of N, substituting unity for this term yields a good approximation. Thus, (2.59) can be written:

$$\underline{h}_{n+1} = \underline{h}_n + \frac{1}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)} (X(n)-\hat{S}(n))\underline{X}(n-n_1). \quad (2.63)$$

It can be seen from (2.63) that under the approximation discussed above the updating of the unit-sample response under this algorithm is identical to that under the LMS Gradient Algorithm except that the constant 2μ is replaced by the data dependent term $(\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1))^{-1}$. Now, it has been shown previously that the LMS Gradient Algorithm will converge for any value of μ such that (2.29) is satisfied. That is, if λ_{max} is the largest eigenvalue of the input autocorrelation matrix, then for convergence the following inequality must hold:

$$0 < \mu < \frac{1}{\lambda_{max}} \quad .$$

The trace of a matrix is defined as the sum of its eigenvalues and may also be computed as the sum of its diagonal elements. Thus, since all the eigenvalues of Φ_{XX} are greater than zero as shown in Chapter I:

$$\lambda_{max} < \sum_{i=1}^{N} \lambda_i = \sum_{i=0}^{N-1} E[X^2(n-n_1-i)] \quad .$$

Thus, the LMS Gradient method will converge if μ is chosen so that:

$$\frac{1}{\mu} > \sum_{i=1}^{N-1} E[X^2(n-n_1-i)] \quad .$$

Or, since the process is assumed stationary:

$$\frac{1}{\mu} > N\ E\ X^2(n) \quad .$$

Now, consider the term $\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)$ as given by (2.60) and take the expected value of both sides:

$$E\left[\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)\right] = \sum_{i=n_1-1}^{n_2-1} E\left[X^2(n-i)\right] = N\ E\left[X^2(n)\right] .$$

Thus, the term $\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)$ is an unbiased estimate of $N \cdot E\left[X^2(n)\right]$. Therefore the recursion relation (2.63) for the unit sample response is identical to that for the LMS Gradient procedure except that the constant 2μ has been replaced by an estimate:

$$2\hat{\mu} = \frac{1}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)}$$

of a step size along the gradient which would guarantee convergence if its true value were known.

Thus, with the assumptions mentioned above, the algorithm presented here based on an attempt to make direct estimates (2.52) and (2.53) approximately satisfy the orthogonality principle may be stated as follows:

$$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n \qquad (2.64)$$

$$\underline{h}_{n+1} = \underline{h}_n + 2\hat{\mu}(X(n)-\hat{S}(n))\underline{X}(n-n_1) \qquad (2.65)$$

$$2\hat{\mu} = \frac{1}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)} \quad . \qquad (2.66)$$

It is obvious from the above equations that the algorithm suggested here is very similar to the LMS Gradient Method and, in fact, may be considered as a modification of the LMS Gradient Method to include an adaptive step size. Thus, an algorithm almost identical to the LMS Gradient Method has been obtained by direct estimation of expected values and application of the orthogonality principle of the estimates.

2.5 Conclusion

The LMS Gradient Algorithm as developed by Widrow is an adaptive recursive algorithm for the approximate solution of the discrete Weiner-Hopf equation for a finite impulse response digital filter. The LMS Gradient Method is based on the Method of Steepest Descent for minimization of a function and can be shown to converge in the mean provided that the step size along the gradient lies within a certain bound. A bound on the rate of convergence of the LMS Gradient has been derived. This bound relates the eigenvalues of the input autocorrelation matrix Φ_{XX} to the rate at which the mean-square error of estimation produced by the LMS Gradient Algorithm approaches the minimum mean-square error produced by the optimal filter. The LMS Gradient Algorithm will converge at a rate less than or equal to this bound.

The LMS Gradient Method was developed based on the Method of Steepest Descent for the minimization of the mean-square error function. An estimate of the gradient was used in

developing the Steepest Descent recursion relation. In this chapter, it was proposed that an algorithm for estimation be based on direct estimates of expected values. It was shown that, given certain simplifying assumptions, the resulting algorithm closely resembled the LMS Gradient Method except that a data-dependent step size was used rather than the fixed parameter μ of the LMS Gradient Algorithm. However, this algorithm was based only on direct estimates of the statistics and application of the orthogonality principle. No numerical analysis methods, such as Steepest Descent were used. This development offers another view of the LMS Gradient Algorithm.

CHAPTER III

ADAPTIVE ESTIMATION USING FIXED POINT ITERATION

The LMS Gradient Algorithm offers an alternative to the direct inversion of an estimated autocorrelation matrix in the approximate solution of the discrete finite impulse response Wiener-Hopf equation (1.13). However, as has been previously mentioned, the choice of the parameter μ has a significant effect on the convergence properties of the algorithm. The choice of a "small" value for μ to insure convergence has a determental effect on the rate of convergence as was shown in Chapter II. In this chapter, other algorithms based on fixed-point iteration methods are studied. These methods prove to be superior to the LMS Gradient Method in terms of rate of convergence. The fixed-point iteration algorithms considered here may be divided into two parts. One part is the estimation of the autocorrelation matrix and autocorrelation vector or the "adaptation" to the statistics of the input process. The second part is the recursive algorithm which uses the estimates of Φ_{XX} and $\underline{R}_{XX}$ to determine the unit-sample response h(n). In this chapter, both the estimation of Φ_{XX} and $\underline{R}_{XX}$ and the approximate solution of the resulting linear system by fixed point iteration are considered. The consistency of the chosen estimates of Φ_{XX}

and $\underline{R}_{XX}$ is considered in detail for the case of a stationary input process X(n) and the case of time varying statistics is also discussed. The convergence properties of fixed-point iteration algorithms is explored in the general case in this chapter and two particular algorithms are considered in detail. These two algorithms are based on the familiar Jacobi Relaxation and Successive Over-Relaxation Methods respectively. The methods based on these fixed-point iteration algorithms offer higher rates of convergence than the LMS Gradient Algorithm at the expense of the number of operations required to implement the procedure. Methods for reducing the number of operations required by Jacobi Relaxation and Successive Over-Relaxation are considered in the final section of this chapter.

3.1 Estimation of the Autocorrelation Matrix and Vector

As mentioned previously, one task of an adaptive processor is to make some estimate of the statistics of the input process. In the case of the LMS Gradient Algorithm, the gradient of the mean square error was estimated by (2.17) which proved to be an unbiased estimate. In the fixed-point iteration algorithms considered in this chapter, the problem of adaptation is approached by the use of direct estimates of the input autocorrelation matrix and the input autocorrelation vector. The estimates used are shown to be consistent for processes for which a Gaussian approximation is reasonable. In addition, a bound on the variance of these estimates is given. Throughout

the following discussion, the input process X(n) is assumed to be stationary. The non-stationary case is discussed at the end of this section.

The estimates of the autocorrelation matrix and the autocorrelation vector of the input process X(n) considered here are the direct estimates given by:

$$\hat{\phi}_{ij} = \frac{1}{M} \sum_{n=0}^{M-1} X(n-i)X(n-j) \qquad (3.1)$$

$$\hat{R}_i = \frac{1}{M} \sum_{n=0}^{M-1} X(n)X(n-n_1-i+1) \qquad (3.2)$$

where $\hat{\phi}_{ij}$ and $\hat{R}_i$ are the $(i,j)^{th}$ element of the autocorrelation matrix estimate and the i^{th} component of the autocorrelation vector estimate respectively. Equations (3.1) and (3.2) are estimates of the $(i,j)^{th}$ element of Φ_{XX} and the i^{th} component of $\underline{R}_{XX}$ respectively whose true values are given by:

$$\phi_{ij} = E[X(n-i)X(n-j)] \qquad (3.3)$$

$$R_i = E[X(n)X(n-n_1-i+1)] \quad . \qquad (3.4)$$

Since the input process has been assumed to be stationary, the autocorrelation matrix estimate $\hat{\Phi}_{XX}$ and the autocorrelation vector estimate $\hat{R}_{XX}$ are made up of samples of the sample autocorrelation function $\hat{R}_{XX}(k)_M$ which is given by:

$$\hat{R}_{XX}(k)_M = \frac{1}{M} \sum_{n=0}^{M-1} X(n)X(n-|k|) \qquad (3.5)$$

where it is assumed that data is available for all $-(n_1+N-1) \leq n \leq m-1$. Note that the symmetry of the sample autocorrelation function has been indicated by the use of $|k|$ on the right-hand side. Since the estimates (3.1) and (3.2) can be obtained from (3.5) as:

$$\hat{\phi}_{ij} = \hat{R}_{XX}(i-j)_M \qquad (3.6)$$

$$\hat{R}_i = \hat{R}_{XX}(n_1+i-1)_M \qquad (3.7)$$

the properties of the estimate given by (3.5) will be considered here. Specifically, the bias and variance of (3.5) is of interest. If the estimate (3.5) is consistent (i.e., if it is unbiased and its variance approaches zero for large M) then $R_{XX}(|k|)$ converges to the true autocorrelation function with probability one. That is for all $\varepsilon > 0$:

$$\lim_{M\to\infty} P(|\hat{R}_{XX}(k)_M - R_{XX}(k)| < \varepsilon) \to 1 \quad .$$

Taking the expected value of both sides of (3.5) yields:

$$E[\hat{R}_{XX}(k)_M] = E[\frac{1}{M} \sum_{n=0}^{M-1} X(n)X(n-|k|)] \quad . \qquad (3.8)$$

By the linearity of the expected value operator, (3.8) can be written as:

$$E[\hat{R}_{XX}(k)_M] = \frac{1}{M} \sum_{n=0}^{M-1} E[X(n)X(n-|k|)]$$

or, since X(n) is stationary:

$$E[\hat{R}_{XX}(k)_M] = E[X(n)X(n-|k|)] = R_{XX}(k) \quad . \qquad (3.9)$$

As can be seen from (3.9) $E[R_{XX}(k)_M]$ is equal to the true value of the autocorrelation function $R_{XX}(k)$. Thus, by definition, the estimate given by (3.5) is unbiased.

The bias of $R_{XX}(k)_M$ was determined above with relatively little effort. However, the variance of $R_{XX}(k)_M$ is difficult to determine in the general case unless the fourth-order statistics of the process X(n) are known. Thus, it will be assumed that the sample values of the process X(n) are jointly Gaussian. However, it can be shown that in most cases the use of the Gaussian assumption yields a good approximation [16], [17]. The variance of $R_{XX}(k)$ may be written as follows:

$$VAR[\hat{R}_{XX}(k)_M] = E[(\hat{R}_{XX}(k)_M - E[\hat{R}_{XX}(k)_M])^2] \qquad (3.10)$$

or:

$$VAR[\hat{R}_{XX}(k)_M] = E[\hat{R}^2_{XX}(k)_M] - 2E[\hat{R}_{XX}(k)_M]^2 + E[\hat{R}_{XX}(k)_M]^2 \quad .$$

Thus, combining terms in the above equation and substituting from (3.9) yields:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] = E[\hat{R}^2_{XX}(k)_M] - R^2_{XX}(k) \quad . \tag{3.11}$$

Now, substituting the definition of $R_{XX}(k)_M$ from (3.5) into (3.11) the following expression is obtained:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] = E[(\frac{1}{M}\sum_{j=0}^{M-1} X(j)X(j-|k|))(\frac{1}{M}\sum_{i=0}^{M-1} X(i)X(i-|k|))] - R^2_{XX}(k)$$

or:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] = E[\frac{1}{M^2}\sum_{j=0}^{M-1}\sum_{i=0}^{M-1} X(j)X(j-|k|)X(i)X(i-|k|)] - R^2_{XX}(k).$$

Applying the linearity of the expected value operator yields:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] = \frac{1}{M^2}\sum_{j=0}^{M-1}\sum_{i=0}^{M-1} E[X(j)X(j-|k|)X(i)X(i-|k|)] - R^2_{XX}(k). \tag{3.12}$$

Note from (3.12) that the variance of the autocorrelation estimate depends on the fourth order statistical expression $E[X(j)X(j-|k|)X(i)X(i-|k|)]$. At this point, it will be assumed that the processes X(i) and X(j) are jointly Gaussian. In this case the fourth-order statistical expression in (3.12) is given by:

$$E[X(j)X(j-|k|)X(i)X(i-|k|)] = R^2_{XX}(k) + R^2_{XX}(i-j) + R_{XX}(i-j-|k|) + R_{XX}(i-j+|k|) \quad . \tag{3.13}$$

Thus, by substituting (3.13) into (3.12), the variance of $R_{XX}(k)$ may be written:

$$\text{VAR}[\hat{R}_{XX}(k)_M] = \frac{1}{M^2}\sum_{j=0}^{M-1}\sum_{k=0}^{M-1}(R_{XX}^2(i-j)+R_{XX}(i-j-|k|)R_{XX}(i-j+|k|)). \quad (3.14)$$

The Gaussian assumption is not extremely restrictive in that it can be shown that (3.14) gives a good approximation to the true variance of the estimate provided that certain constraints are met. It can also be shown that most reasonable processes meet the required constraints [17].

Equation (3.14) may be simplified further by introducing a new variable n = (i-j) which gives:

$$\text{VAR}[\hat{R}_{XX}(k)_M] = \frac{1}{M^2}\sum_{j=0}^{M-1}\sum_{n=-j}^{M-1-j}(R_{XX}^2(n)+R_{XX}(n-|k|)R_{XX}(n+|k|))$$

or:

$$\text{VAR}[\hat{R}_{XX}(k)_M] = \frac{1}{M^2}\sum_{j=0}^{M-1}\sum_{n=1-M}^{M-1}(R_{XX}^2(n)+R_{XX}(n-|k|)R_{XX}(n+|k|))$$

$$-\sum_{n=1-M}^{-(j+1)}(R_{XX}^2(n)+R_{XX}(n-|k|)R_{XX}(n+|k|))$$

$$-\sum_{n=j+1}^{m-1}(R_{XX}^2(n)+R_{XX}(n-|k|)R_{XX}(n+1|k|))) \quad . \quad (3.15)$$

Now, since $R_{XX}(n)=R_{XX}(-n)$, (3.15) can be rewritten as:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] = \frac{1}{M^2} \sum_{j=0}^{M-1} \Big[\sum_{n=1-M}^{M-1} (R_{XX}^2(n) + R_{XX}(n-|k|)R_{XX}(n+|k|)$$

$$- 2 \sum_{n=j+1}^{M-1} (R_{XX}^2(n) + R_{XX}(n-|k|)R_{XX}(n+|k|)) \Big] \quad .$$

Interchanging the order of summation and summing over j gives:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] = \frac{1}{M} \sum_{n=1-M}^{M-1} \left(1 - \frac{|n|}{M}\right) (R_{XX}^2(n) + R_{XX}(n-|k|)R_{XX}(n+|k|)) . \tag{3.16}$$

Equation (3.16) is an expression for the variance of $\hat{R}_{XX}(k)_M$ in terms of the autocorrelation function $R_{XX}(k)$ of the input process X(n). As was stated earlier, (3.16) is exact for zero mean Gaussian processes and approximate for most other processes of interest.

If $\hat{R}_{XX}(k)_M$ is a consistent estimate of $R_{XX}(k)$, then its variance given by (3.16) must approach zero as M becomes large. That is:

$$\lim_{M \to \infty} \mathrm{VAR}[\hat{R}_{XX}(k)_M] \to 0 \quad . \tag{3.17}$$

To see that (3.17) does hold, consider the following limit:

$$\lim_{M \to \infty} M \, \mathrm{VAR}[\hat{R}_{XX}(k)_M] = \sum_{n=-\infty}^{\infty} (R_{XX}^2(n) + R_{XX}(n+|k|)R_{XX}(n-|k|)) . \tag{3.18}$$

Now, if the sequences R_{XX}^2 and $R_{XX}(n-|k|)R_{XX}(n+|k|)$ are absolutely summable then:

$$\lim_{M\to\infty} M\ \mathrm{VAR}[\hat{R}_{XX}(k)_M] \to C < \infty \quad .$$

Thus, since the limit in (3.18) approaches a finite constant as M approaches ∞, the variance of the estimate must approach zero as M becomes large.

It has been shown that the estimate of the autocorrelation function given by (3.5) is unbiased and its variance approaches zero as M approaches ∞. Thus, $\hat{R}_{XX}(k)_M$ is a consistent estimate of $R_{XX}(k)$.

The rate at which the variance of $\hat{R}_{XX}(k)_M$ approaches zero can be determined by considering the expression for the variance given by (3.16). From (3.16):

$$\mathrm{VAR}[\hat{R}_{XX}(k)_{M+1}] = \frac{1}{M+1} \sum_{n=-M}^{M} (1 - \frac{|n|}{M+1}) (R_{XX}^2(n) + R_{XX}(n-|k|) R_{XX}(n+|k|))$$

or, in terms of $\mathrm{VAR}[\hat{R}_{XX}(k)_M]$:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_{M+1}] = \frac{M}{M+1} \mathrm{VAR}[\hat{R}_{XX}(k)_M] + \frac{2}{M+1}(1 - \frac{M}{M+1}) (R_{XX}^2(M) + R_{XX}$$

$$(M-|k|) R_{XX}(M+|k|)) \quad . \qquad (3.19)$$

Subtracting both sides of (3.19) from $\mathrm{VAR}[\hat{R}_{XX}(k)_M]$ yields:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] - \mathrm{VAR}[\hat{R}_{XX}(k)_{M+1}] = (1 - \frac{1}{M+1}) [\mathrm{VAR}[\hat{R}_{XX}(k)_M] -$$

$$- \frac{2}{M+1} (R_{XX}^2(M) + R_{XX}(M|k|) R_{XX}(M+|k|)]$$

or:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_M] - \mathrm{VAR}[\hat{R}_{XX}(k)_{M+1}] = \frac{1}{M+1}\left[\mathrm{VAR}[\hat{R}_{XX}(k)_M] - \frac{2}{M+1}\left(R_{XX}^2(M) + R_{XX}(M-|k|)R_{XX}(M+|k|)\right)\right] . \quad (3.20)$$

From (3.20) it can be seen that if $(R_{XX}^2(M)+R_{XX}(M-|k|)R_{XX}(M+|k|))$ is small for large values of M (which is a necessary condition for the sequence to be absolutely summable), then the variance of $\hat{R}_{XX}(k)_M$ decreases as 1/M for large values of M.

A bound of the variance of $\hat{R}_{XX}(k)_M$ can be determined by considering equation (3.19). From (3.19):

$$\mathrm{VAR}[\hat{R}_{XX}(k)_{M+1}] = \frac{M}{M+1}\,\mathrm{VAR}[\hat{R}_{XX}(k)_M] + \frac{2}{(M+1)^2}\left(R_{XX}^2(M) + R_{XX}(M-|k|)R_{XX}(M+|k|)\right) .$$

Now, since the autocorrelation has a maximum at k=0, the following inequality holds for all k:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_{M+1}] \leq \frac{M}{M+1}\,\mathrm{VAR}[\hat{R}_{XX}(k)_M] + \frac{4}{(M+1)^2}\,R_{XX}^2(0). \quad (3.21)$$

For M=1, the variance of $\hat{R}_{XX}(k)_1$ is found from (3.16) to be:

$$\mathrm{VAR}[\hat{R}_{XX}(k)_1] = R_{XX}^2(0) + R_{XX}^2(k) \leq 2R_{XX}^2(0). \quad (3.22)$$

Thus, using (3.22) and (3.21) a bound for VAR[$\hat{R}_{XX}(k)_M$] can be generated sequentially. This bound can be used to determine the value of M which will insure that the variance of $\hat{R}_{XX}(k)_M$ is less than a specified level. Determination of an appropriate value of M is discussed in the next section.

3.1.1 Estimation of the Autocorrelation Function -- Time-Varying Case

In the preceeding section, it was shown that, for the case of a stationary input process X(n), $\hat{R}_{XX}(k)_M$ given by (3.5) is a consistent estimate of the autocorrelation function. In the case of a process with time-varying statistics, estimation of those statistics from the data is difficult. However, if the variation with time of the statistics is slow compared to the number of samples used in making the estimate and to the length of the filter, then one possible approach is to use a sliding window of length $(M+n_1+N)$ and compute a moving average. That is at time $n > M$, the estimate $\hat{R}_{XX}(k)_M$ would be given by:

$$\hat{R}_{XX}(k)_M = \frac{1}{M} \sum_{\ell=n-M+1}^{n} X(\ell)X(\ell-|k|). \qquad (3.23)$$

Obviously, if $\hat{R}_{XX}(k)_M$ is to be a good approximation to the time-varying statistic $R_{XX}(k)$, then $(M+n_1+N)$ should be small as possible while still meeting any specified requirements regarding the variance of the estimate. Ideally, the variance of

$\hat{R}_{XX}(k)_M$ should be very small while keeping the value of $(M+n_1+N)$ small with respect to the rate at which the statistic $R_{XX}(k)$ is varying. Since the variance cannot be reduced without increasing M, some engineering decision must be made. The bound on the variance of $\hat{R}_{XX}(k)_M$ which can be computed using (3.21) and (3.22) can be used as an aid in this decision. Chebychev's inequality states that:

$$P[|\hat{R}_{XX}(k)_M - R_{XX}(k)| \geq \varepsilon] \leq \frac{VAR[\hat{R}_{XX}(k)_M]}{\varepsilon^2} \quad .$$

Thus, if $B_k(M)$ denotes the bound on the variance of $\hat{R}_{XX}(k)_M$ computed using (3.21) and (3.22) then the following inequality holds:

$$P[|\hat{R}_{XX}(k)_M - R_{XX}(k)| \geq \varepsilon] \leq \frac{B_k(M)}{\varepsilon^2}$$

or:

$$P[|\hat{R}_{XX}(k)_M - R_{XX}(k)| \leq \varepsilon] \geq 1 - \frac{B_k(M)}{\varepsilon^2} \quad . \qquad (3.24)$$

Equation (3.24) can be used to determine a bound on the probability that the random variable $\hat{R}_{XX}(k)_M$ lies within a specified ε-region of the true value $R_{XX}(k)$ or to determine the value of M required to insure that $P[|R_{XX}(k)_M - R_{XX}(k)| \leq \varepsilon]$ meets or exceeds a certain value for a specified value of ε.

Thus, the constraints on the selection of an appropriate value of M are that the variation of the statistic $R_{XX}(k)$ be "small" over $(M+n_1+N)$ units of time and that the value of the

right-hand side of (3.24) be close to 1 for some "small" value of ε. The definition of "small" is an engineering decision and will vary over a range of problems. Obviously, in all cases, the constraints on the value of M cannot be met. Thus, trade-offs must be made in some situations.

It should be noted that the selection of a value of M as described in this section requires more knowledge about the statistics of the data process X(n) than was originally assumed to be given. Specifically, some information about the rate at which the statistics vary with time and some knowledge of $R_{XX}(0)$ is required to select M. However, the restrictions imposed by requiring this a priori knowledge are not great in a large number of practical problems.

It has been proposed that a moving average given by (3.23) be used to estimate the statistic $R_{XX}(k)$ in the non-stationary case under the constraint that $R_{XX}(k)$ be approximately stationary over $(M+n_1+N)$ units of time. If the assumption of local time invariance is valid, then (3.23) is a consistent estimate of the locally stationary statistic $R_{XX}(k)$ with variance given by (3.16). The bound which can be computed using (3.21) and (3.22) can be used along with (3.24) to aid in the selection of the length M of the sliding window for the moving average provided that some a priori knowledge of the time variation of $R_{XX}(k)$ and $R_{XX}(0)$ is available.

3.1.2 Recursive Estimation

The moving average given by (3.23) was proposed in the preceeding section for the estimation of the autocorrelation of a non-stationary process whose statistics vary slowly with respect to the length of the data window used in the moving average. In this section, a method for computing the estimate given by (3.23) will be presented which is computationally more efficient than direct evaluation of the sum. If $\hat{R}_{XX,n}(k)_M$ is defined as the estimate of the autocorrelation function at sample time n as given by (3.23), then at sample time n+1 the estimate is given by:

$$\hat{R}_{XX,n+1}(k)_M = \frac{1}{M} \sum_{\ell=n-M+2}^{n+1} X(\ell)X(\ell-|k|)$$

or:

$$\hat{R}_{XX,n+1}(k)_M = \frac{1}{M} \sum_{\ell=n-M+1}^{n} X(\ell)X(\ell-|k|) + \frac{1}{M} X(n+1)X(n+1-|k|)$$

$$- \frac{1}{M} X(n-m+1)X(n-M+1-|k|) \ .$$

Thus, $\hat{R}_{XX,n}(k)_M$ can be written in recursive form as:

$$\hat{R}_{XX,n+1}(k)_M = \hat{R}_{XX,n}(k)_M + \frac{1}{M} [X(n+1)X(n+1-|k|)$$

$$- X(n-M+1)X(n-M+1-|k|)] \ . \qquad (3.25)$$

The recursive form (3.25) may be used to compute $\hat{R}_{XX,n}(k)_M$ and provides substantial savings in computation time over the direct evaluation of the sum in (3.23). Thus, the form of (3.25) will be used in the Adaptive Fixed-Point Iteration Algorithms.

3.2 Approximate Solution of the Wiener-Hopf Equation By Adaptive Fixed-Point Iteration

Historically, there are two widely used methods for the approximate solution of the discrete FIR form of the Wiener-Hopf equation. One method is an adaptive algorithm such as the LMS Gradient procedure discussed in Chapter II. Another approach is to receive a block of input data, make some estimate of the autocorrelation matrix and autocorrelation vector based on this block of data and then to solve the system of equations:

$$\hat{\Phi}_{XX}\,\underline{h} = \hat{\underline{R}}_{XX} \quad . \tag{3.26}$$

The solution of (3.26) is then used to filter the block of data to obtain an estimate of the signal during the time interval covered by the data record used. Such an approach is usually an off-line process due to the number of computations required and the processing delay.

The approach considered in this section lies somewhere between the LMS Gradient Algorithm and the second approach discussed above. The Adaptive Fixed-Point Iteration approach is an attempt to solve an equation of the form of (3.26). However, in contrast to the second method discussed above, both

the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ and the unit sample response vector $\underline{h}_n$ are updated recursively. The estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ are updated by the addition of the new data point to the data window being used while $\underline{h}_n$ is updated by carrying out one step of a fixed-point iteration algorithm. Thus, the system (3.26) is not solved at any one sample time. Rather, the solution is obtained iteratively over a number of samples with the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ being updated concurrently. The Adaptive Fixed-Point Iteration discussed here is similar to the LMS Gradient Method in that one step of an iterative algorithm for finding $\underline{h}^*$ is carried out at each sample point in time. However, it differs from the LMS Gradient Algorithm in regards to the handling of the past data. In Adaptive Fixed-Point Iteration, the history of the past data is summarized in the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ as well as in the unit sample response $\underline{h}_n$. In constrast, $\underline{h}_n$ contains all the information about the past data available in the LMS Gradient Algorithm. Therefore, Adaptive Fixed-Point Iteration lies somewhere between the two historical methods discussed in terms of computational effort and data utilized in the estimation of the statistics.

3.2.1 Fixed-Point Iteration

In this section, the general case of fixed-point iterative solutions of systems of the form (3.26) will be discussed. This will serve as a framework for the consideration of the particular algorithms discussed in later sections. Specifically, results derived for the general case can be directly applied to

particular algorithms. Initially, it will be assumed that the matrix $\hat{\Phi}_{XX}$ and the vector $\hat{\underline{R}}_{XX}$ are constant. The results derived for this case will be extended to the case of Adaptive Fixed-Point Iteration where $\hat{\Phi}_{XX}$ is a matrix random process and $\hat{\underline{R}}_{XX}$ is a vector random process.

The principle behind the fixed-point iterative solution of (3.26) is to find the solution $\underline{h}^*$ as a limit of a vector sequence $\underline{h}_0$, $\underline{h}_1$, $\underline{h}_2$,... whose terms are computed recursively as:

$$\underline{h}_{n+1} = F(\underline{h}_n) \qquad n=0,1,2,... \tag{3.27}$$

where $\underline{h}_0$ is an initial guess and the operator F is chosen in such a way that $\underline{h}^*$ is a fixed point of F. That is:

$$\underline{h}^* = F(\underline{h}^*) \quad . \tag{3.28}$$

For the linear system (3.26) it is desired to find a matrix B and a vector $\underline{d}$ so that the vector equation:

$$\underline{h} = B\,\underline{h} + \underline{d} \tag{3.29}$$

is equivalent to the system (3.26). That is, the solution vector of (3.26) should also be a solution of (3.29) and a fixed-point of the iteration:

$$\underline{h}_{n+1} = B\,\underline{h}_n + \underline{d} \quad . \tag{3.30}$$

The matrix B and the vector $\underline{d}$ may be found by considering the vector equation:

$$\underline{h} = \underline{h} + C^{-1}(\hat{\underline{R}}_{XX} - \hat{\Phi}_{XX}\,\underline{h}) \tag{3.31}$$

where C is any invertable NXN matrix. Obviously, (3.31) has a solution if and only if:

$$C^{-1}(\hat{\underline{R}}_{XX} - \hat{\Phi}_{XX}\underline{h}) = \underline{0} \tag{3.32}$$

and, since C was chosen to be invertable, (3.32) holds if and only if:

$$\hat{\Phi}_{XX}\underline{h} = \hat{\underline{R}}_{XX} \quad . \tag{3.26}$$

Thus, the solution of (3.31) is also the solution of the linear system (3.26). Therefore, a fixed-point iteration of the form:

$$\underline{h}_{n+1} = (I - C^{-1}\hat{\Phi}_{XX})\underline{h}_n + C^{-1}\hat{\underline{R}}_{XX} \tag{3.33}$$

may be used to find a solution of the linear system (3.26) provided the sequence $\underline{h}_0$, $\underline{h}_1$, $\underline{h}_2$,... produced by (3.33) converges to the fixed-point $\underline{h}^*$.

To determine whether or not the sequence $\underline{h}_n$ generated by (3.33) will converge for all initial guesses $\underline{h}_0$, Theorem 2.1 from Chapter 2 can be applied. Theorem 2.1 states that a fixed-point iteration of the form (3.30) converges regardless of the

initial guess $\underline{h}_0$ if and only if the eigenvalues of the iteration matrix B are strictly less than one in absolute value. Thus, the iteration given by (3.33) converges if and only if the eigenvalues of the NxN matrix:

$$(I-C^{-1}\hat{\Phi}_{XX})$$

are strictly less than one in absolute value. Therefore, the matrix C must be chosen in such a way as to meet this criterion.

In addition to requiring that the matrix C be such that the eigenvalues of $(I-C^{-1}\hat{\Phi}_{XX})$ are less than one in absolute value, the restriction must also be made that C be easily invertable. Obviously, if C is simply a general NxN matrix, the iteration would be pointless since with the same amount of work the matrix $\hat{\Phi}_{XX}$ could be inverted and the solution found directly. The choice of the matrix C is what distinguishes particular algorithms within the class of fixed-point iterative methods. Two particular algorithms will be considered in the sections that follow. These methods use a diagonal matrix and a lower triangular matrix respectively for the matrix C. Such forms are easily invertable and can be chosen so as to guarantee convergence for the problem considered.

The convergence of fixed-point iteration algorithms can be guaranteed for a constant matrix $\hat{\Phi}_{XX}$ and a constant vector $\underline{\hat{R}}_{XX}$ if the matrix C is chosen so that the requirements of Theorem 2.1 are satisfied. However, in Adaptive Fixed-Point

Iteration, $\hat{\Phi}_{XX}$ and $\underline{\hat{R}}_{XX}$ are a random matrix and a random vector respectively. Thus, the sequence $\underline{h}_n$ produced by the adaptive fixed-point iteration is actually a sequence of random vectors and "convergence" of the algorithm must be defined as convergence in the mean. Thus, the convergence of the iteration given by:

$$E[C_n \underline{h}_{n+1}] = E[(C_n - \hat{\Phi}_{XXn})\underline{h}_n] + E[\underline{\hat{R}}_{XXn}]$$

must be considered where $\hat{\Phi}_{XX,n}$, $\underline{\hat{R}}_{XX,n}$ and C_n are the random matrix $\hat{\Phi}_{XX}$, the random vector $\underline{\hat{R}}_{XX}$ and the random matrix C at sample time n. The matrix C_n is considered to be random since in the general case it may be derived from the random matrix $\hat{\Phi}_{XX,n}$. If the length M of the data window is sufficiently large, then cross-correlation between terms can be neglected and (3.34) can be rewritten as:

$$E[\underline{h}_{n+1}] = (I - E[C_n]^{-1} E[\hat{\Phi}_{XXn}]) E[\underline{h}_n] + E[C_n]^{-1} E[\underline{\hat{R}}_{XXn}]. \quad (3.35)$$

Now, the matrix $(I - E[C_n]^{-1} E[\hat{\Phi}_{XX,n}])$ and the vector $E[C_n^{-1}] E[\underline{\hat{R}}_{XX,n}]$ are constant provided that the process X(n) is stationary. In the case of non-stationary processes, it is assumed that the statistics of X(n) vary slowly with respect to the rate of convergence of the fixed-point iteration so that the matrix $E[C_n]^{-1} E[\hat{\Phi}_{XX,n}]$ and the vector $E[C_n]^{-1} E[\underline{\hat{R}}_{XX,n}]$ remain constant until the algorithm has converged. Thus, Adaptive Fixed-Point Iteration will converge in the mean provided that the eigenvalues of the matrix:

$$(I-E[C_n]^{-1}\Phi_{XX}) \qquad (3.36)$$

are strictly less than one in absolute value.

3.2.2 Selection of the Matrix C to Insure Convergence

In practice, the determination of the eigenvalues of the matrix in (3.36) may be difficult since little or no a priori knowledge of the statistics of the process X(n) is assumed. Thus, it is useful to have some bound on the eigenvalues of the matrix in (3.36). If $\tilde{\lambda}$ is an eigenvalue of the matrix in (3.36), then the following equation is satisfied:

$$DET[\tilde{\lambda}I-(I-E[C_n]^{-1}\Phi_{XX})] = 0$$

or:

$$DET[E[C_n]^{-1}((\tilde{\lambda}-1)E[C_n]+\Phi_{XX})] = 0 \quad . \qquad (3.37)$$

The determinant of a product is the product of the determinants. Thus, (3.37) can be rewritten as:

$$DET[E[C_n]^{-1}]DET[(\tilde{\lambda}-1)E[C_n]+\Phi_{XX}] = 0 \quad .$$

But, by hypotheses, $E[C_n]^{-1}$ is invertable and thus $DET[E[C_n]^{-1}] \neq 0$. Therefore,

$$DET[(\tilde{\lambda}-1)E[C_n]+\Phi_{XX}] = 0 \quad . \qquad (3.38)$$

From (3.38), it follows that the matrix:

$$(\tilde{\lambda}-1)E[C_n]+\Phi_{XX}$$

is singular. Thus, by Hadammard's Theorem [19]:

$$|(\tilde{\lambda}-1)E[C_{ii}]+\phi_{ii}| \leq \sum_{\substack{j=1 \\ j\neq i}}^{N} |(\tilde{\lambda}-1)C_{ij}+\phi_{ij}| \qquad (3.39)$$

where ϕ_{ij} and C_{ij} are the i,j^{th} elements of the matrices Φ_{XX} and C_n respectively.

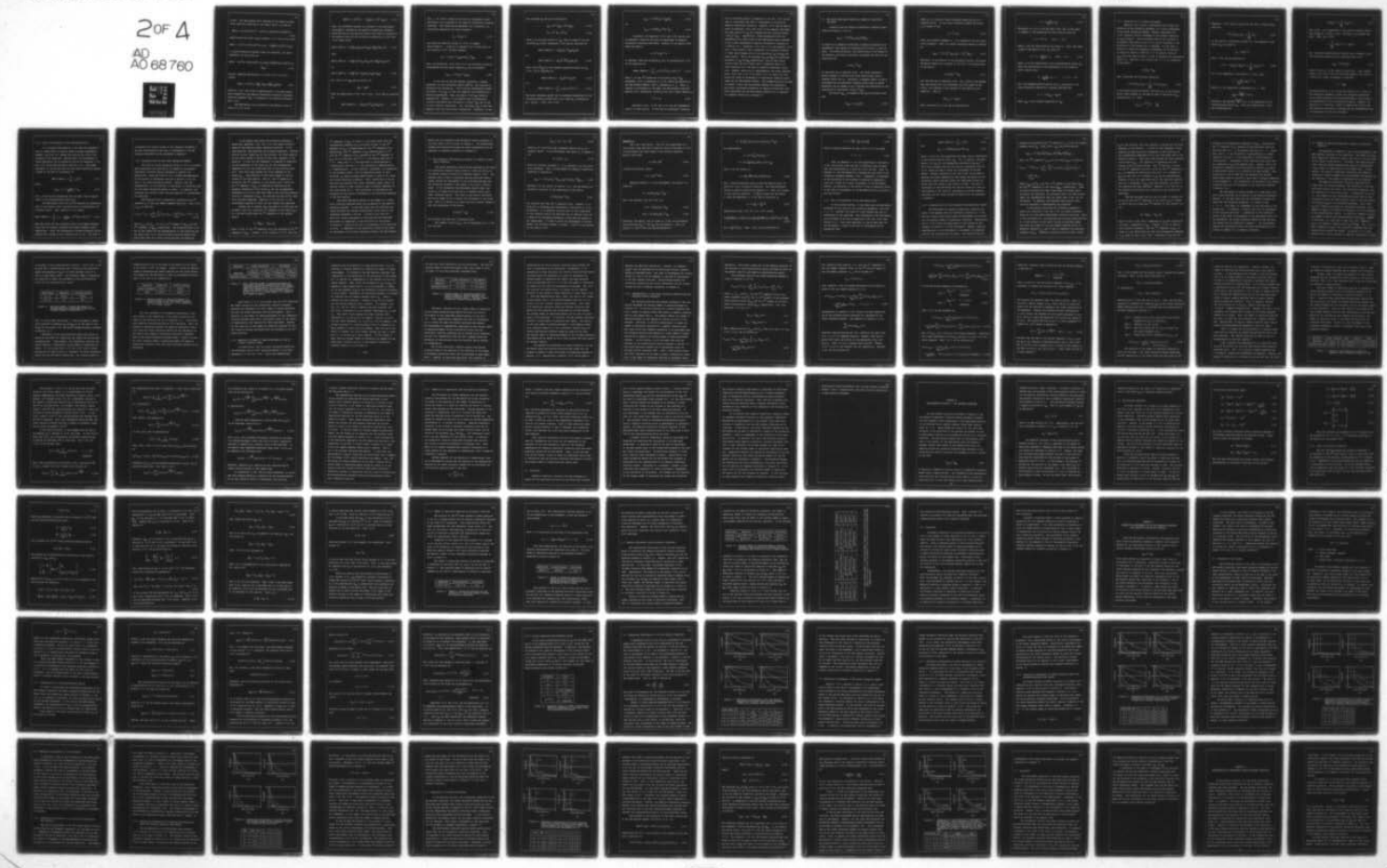
Equation (3.39) does not present a simple expression for a bound on the values of the eigenvalues of the matrix in (3.36). However, as will be seen in later sections, (3.39) will provide a useful bound for the particular algorithms considered in this dissertation and will allow the selection of the matrix C_n so that convergence in the mean is guaranteed.

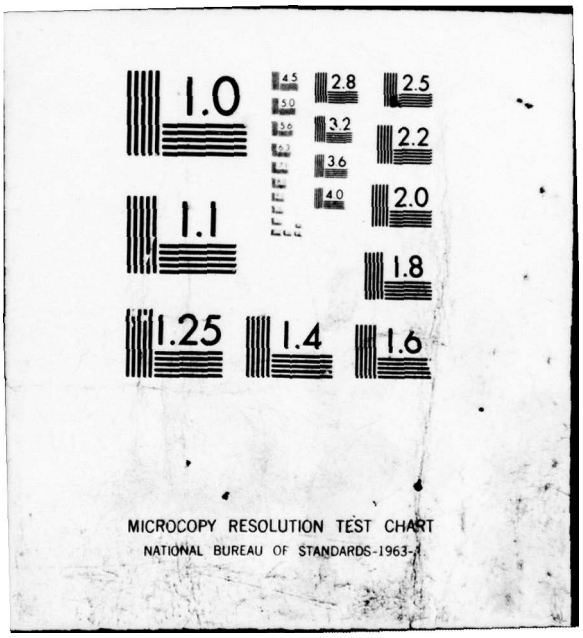
3.2.3 Rate of Convergence of the Mean Square Error

Once it has been determined by the use of Theorem 2.1 that the vector sequence $E[\underline{h}_0]$, $E[\underline{h}_1]$,... produced by Adaptive Fixed-Point Iteration converges to the desired solution $\underline{h}^*$, it is then of interest to consider the rate of convergence of the sequence. As in the case of the LMS Gradient Algorithm, it is of the most interest to consider the convergence of the mean-square error produced by the Adaptive Fixed-Point Iteration to the minimum mean-square error produced by the optimal

AD-A068 760 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/3
ADAPTIVE LINEAR ESTIMATION ALGORITHMS APPLIED TO SPECTRAL LINE --ETC(U)
AUG 78 S D HUFFMAN N00014-75-C-0191
UNCLASSIFIED TR-15 NL

2 OF 4
AD
AO 68760




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

filter. The mean-square error produced by the Adaptive Fixed-Point Iteration Algorithm at any sample time n is given by:

$$MSE(n) = E[(S(n)-\hat{S}(n))^2] = E[S^2(n)]-2E[S(n)\hat{S}(n)]+E[\hat{S}^2(n)]. \tag{3.40}$$

Substituting the filter output equation (1.7) into (3.40) gives:

$$MSE(n) = E[S^2(n)]-2E[S(n)\underline{X}^T(n-n_1)\underline{h}_n] + E[\underline{h}_n^T\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n] .$$

If cross-correlations between terms are neglected, the above expression can be written as:

$$MSE(n) = E[S^2(n)]-2E[S(n)\underline{X}^T(n-n_1)]E[\underline{h}_n]+E[\underline{h}_n^T]E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]$$
$$E[\underline{h}_n] . \tag{3.41}$$

Finally, applying definitions (1.11) and (1.12) to (3.41) yields:

$$MSE(n) = E[S^2(n)]-2\underline{R}_{SX}^T E[\underline{h}_n]+E[\underline{h}_n^T]\Phi_{XX}E[\underline{h}_n]. \tag{3.42}$$

Equation (3.38) then gives an approximation to the mean-square error produced by the Adaptive Fixed-Point Iteration algorithm where the sequence $E[\underline{h}_n]$ is produced by the recursive relationship (3.34).

The mean-square error produced by the optimal filter is given by (2.33) to be:

$$MSE^*(n) = E[S^2(n)] - 2\,\underline{R}_{SX}^T\,\underline{h}^* + \underline{h}^{*T}\,\Phi_{XX}\,\underline{h}^*. \quad (2.33)$$

Thus, the difference between the estimate of the mean-square error MSE(n) produced by the Adaptive Fixed-Point Iteration Algorithm and the minimum mean-square error MSE*(n) produced by the optimal filter $\underline{h}^*$ can be found by subtracting (2.33) from (3.42) which gives:

$$MSE(n)-MSE^*(n) = E[\underline{h}_n^T]\Phi_{XX}E[\underline{h}_n]h^*T\Phi_{XX}\underline{h}^*-2\underline{R}_{SX}^T[E[\underline{h}_n]-\underline{h}^*] \; . \quad (3.43)$$

Thus,

$$MSE(n)-MSE^*(n) = (E[\underline{h}_n^T]-\underline{h}^{*T})\,\Phi_{XX}(E[\underline{h}_n]+\underline{h}^*)-2\underline{R}_{SX}^T[E(\underline{h}_n]-\underline{h}^*]$$

or:

$$MSE(n)-MSE^*(n) = (E[\underline{h}_n^T]-\underline{h}^*)\,[\Phi_{XX}E[\underline{h}_n]+\Phi_{XX}\underline{h}^*-2\underline{R}_{SX}] \; . \quad (3.44)$$

But, from (1.14) $\underline{R}_{SX}$ may be written as:

$$\underline{R}_{SX} = \Phi_{XX}\underline{h}^* \; . \quad (3.45)$$

Thus, by substituting (3.45) into (3.44), (3.44) may be rewritten as:

$$MSE(n)-MSE^*(n) = (E[\underline{h}_n]-\underline{h}^*)^T\Phi_{XX}(E[\underline{h}_n]-\underline{h}^*). \quad (3.46)$$

Thus, (3.46) gives a bound on the rate of convergence of the mean-square error produced by the adaptive fixed-point iteration algorithm. This bound may be generated for increasing n by recursively generating the vector sequence:

$$\underline{Y}_n = E[\underline{h}_n] - h^* \tag{3.47}$$

and using the resulting values in (3.46) to compute (MSE(n)-MSE*(n)). Since $\underline{h}^*$ is assumed to be a fixed point of the iteration (3.35), $\underline{h}^*$ must satisfy:

$$\underline{h}^* = (I - E[C_n]^{-1}\Phi_{XX})\underline{h}^* + E[C_n]^{-1}\underline{R}_{XX} \quad . \tag{3.48}$$

Thus, by subtracting (3.48) from (3.35) the following recursive relationship for the vector sequence $\underline{Y}_n$ may be obtained:

$$\underline{Y}_{n+1} = (I - E[C_n]^{-1}\Phi_{XX})\underline{Y}_n \quad . \tag{3.49}$$

In the case of the LMS Gradient Algorithm, a simpler expression for the difference (MSE(n)-MSE*(n)) was obtained by uncoupling the variable $\underline{h}_n$. This could be accomplished because the matrix $(I-2\mu\ \Phi_{XX})$ is real and symmetric and therefore has distinct eigenvalues. Thus, the matrix $(I-2\mu\ \Phi_{XX})$ could be diagonalized. However, in the general case of fixed-point iteration considered here the matrix $(I-E[C_n]^{-1}\Phi_{XX})$ may or may not have distinct eigenvalues. But, the matrix Φ_{XX} is real and symmetric and thus has distinct eigenvalues. Therefore, if the

new variables $\underline{h}_n$ and $\underline{h}^*$ are defined as:

$$\tilde{\underline{h}}_n = M^{-1} E[\underline{h}_n] = M^T E[\underline{h}_n] \tag{3.50}$$

$$\tilde{\underline{h}}^* = M^{-1} \underline{h}^* = M^T \underline{h}^* \tag{3.51}$$

where M is the modal matrix of Φ_{XX}, then in terms of the new variables $\tilde{\underline{h}}_n$ and $\tilde{\underline{h}}^*$, expression (3.46) may be rewritten as:

$$MSE(n)-MSE^*(n) = (M(\tilde{\underline{h}}_n-\tilde{\underline{h}}^*))^T \Phi_{XX} M(\tilde{\underline{h}}_n-\tilde{\underline{h}}^*)$$

or:

$$MSE(n)-MSE^*(n) = (\tilde{\underline{h}}_n-\tilde{\underline{h}}^*)^T M^T \Phi_{XX} M(\tilde{\underline{h}}_n-\tilde{\underline{h}}^*) \; . \tag{3.52}$$

Thus, since $M^T \Phi_{XX} M$ is the diagonal eigenvalue matrix of Φ_{XX}, (3.52) can be expressed as:

$$MSE(n)-MSE^*(n) = (\tilde{\underline{h}}_n-\tilde{\underline{h}}^*)^T \Lambda (\tilde{\underline{h}}_n-\tilde{\underline{h}}^*) \tag{3.53}$$

or:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i (\tilde{h}_n(i)-\tilde{h}^*(i))^2 \; . \tag{3.54}$$

The vector sequence $(\tilde{\underline{h}}_n-\tilde{\underline{h}}^*)$ may be generated recursively by the following formula derived from (3.49) where $\tilde{\underline{Y}}_n$ is defined as $\tilde{\underline{Y}}_n = (\tilde{\underline{h}}_n-\tilde{\underline{h}}^*)$. Thus, from (3.49):

$$\tilde{\underline{Y}}_{n+1} = (I-M^T E[C_n]^{-1}\Phi_{XX}M)\tilde{\underline{Y}}_n \qquad (3.55)$$

or:

$$\tilde{\underline{Y}}_{n+1} = (I-M^T E[C_n]^{-1}M\Lambda)\ \tilde{\underline{Y}}_n \ . \qquad (3.56)$$

In general, the equations (3.53) and (3.56) can be used to determine a bound on the rate of convergence of Adaptive Fixed-Point Iteration Algorithms. However, in the special case where the matrix:

$$D = M^T\ E[C_n]^{-1}\ M$$

is diagonal, then the variables $\tilde{\underline{Y}}_n$ will be uncoupled and (3.54) can be rewritten as:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i(1-\tilde{\lambda}_i)^{2n}(\tilde{h}_0(i)-\tilde{h}^*(i))^2 \qquad (3.57)$$

where $\tilde{\lambda}_i$ is the i^{th} eigenvalue of the matrix $E[C_n]^{-1}\Phi_{XX}$. Expression (3.57) is identical to (2.49) (which was derived for the LMS Gradient Method) with $\tilde{\lambda}_i$ replaced by $2\mu\ \lambda_i$. Thus, in regards to convergence in the mean, the LMS Gradient Algorithm behaves like a fixed-point iteration with the C matrix chosen as:

$$C^{-1} = 2\mu I \ . \qquad (3.58)$$

Equations (3.54), (3.55) and (3.57) are the fundamental results of this section. In the case of fixed-point iteration

with an arbitrary matrix C_n equations (3.54) and (3.55) can be used to investigate the rate of convergence in the mean of Adaptive Fixed-Point Iteration. Equation (3.57) may be used in place of equations (3.54) and (3.55) in the special case where the modal matrix of Φ_{XX} will diagonalize the iteration matrix $(I-E[C_n]^{-1}\Phi_{XX})$. Specifically, these expressions allow the computation of a lower bound on the mean-square error produced by the filter determined by Adaptive Fixed-Point Iteration as a function of n. Equations (3.54) and (3.55) and equation (3.57) give a lower bound in that convergence will proceed according to these relationships only if the statistics Φ_{XX} and $\underline{R}_{XX}$ are known exactly. While the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ are consistent estimates of Φ_{XX} and $\underline{R}_{XX}$ respectively, they do have finite variance when the data window used in computing them is of finite duration as in the case in Adaptive Fixed-Point Iteration. However, while only an approximation, the bound computed using (3.54) and (3.55) or by using (3.57) is useful for comparing the rates of convergence of fixed-point iteration algorithms and the LMS Gradient Method. Such comparisons will be made in Chapter V while the relationship between the theoretical and actual convergence properties of Adaptive Fixed-Point Iteration Algorithms and the LMS Gradient Method will be explored via computer simulation in Chapter VI.

3.3 The Jacobi Relaxation Method for Adaptive Fixed-Point Iteration

A general form for Adaptive Fixed-Point Iteration Algorithms was given by (3.33) as:

$$\underline{h}_{n+1} = (I-C_n^{-1}\hat{\Phi}_{XX,n})\underline{h}_n+C_n^{-1}\hat{\underline{R}}_{XX,n} \quad .$$

In order for an Adaptive Fixed-Point Iteration Algorithm to be implemented, some method for determining the matrix C_n must be given. As stated previously, the restrictions on the choice of the matrix C_n are that C_n be easily invertable and that all the eigenvalues of:

$$(I-E[C_n]^{-1}\Phi_{XX})$$

be less than one in absolute value. The Jacobi Relaxation Method attempts to satisfy both these restrictions using a diagonal matrix for C_n. Obviously, a diagonal matrix is easily invertable and the particular diagonal matrix used in Jacobi Relaxation can be chosen so as to satisfy the constraints on the eigenvalues of the matrix $(I-E[C_n]^{-1}\Phi_{XX})$.

The matrix $\hat{\Phi}_{XX,n}$ is symmetric and may be written as the sum:

$$\hat{\Phi}_{XX,n} = L_n+D_n+L_n^T \qquad (3.59)$$

where L_n is a strictly lower triangular matrix and D_n is a diagonal matrix. In the Jacobi Relaxation Method the matrix C_n is chosen to be:

$$C_n = 1/\beta \; D_n \tag{3.60}$$

where the constant parameter $\beta > 0$ is referred to as the relaxation parameter. Thus, the Jacobi Relaxation Method is defined as:

$$\underline{h}_{n+1} = (I-\beta \; D_n^{-1} \; \hat{\Phi}_{XX,n})\underline{h}_n + \beta \; D_n^{-1} \; \hat{\underline{R}}_{XX,n} \quad . \tag{3.61}$$

Therefore, by the results of the preceeding section, the Jacobi Relaxation Method will converge if and only if the eigenvalues of the matrix:

$$(I-\beta \; E[D_n]^{-1} \; \Phi_{XX}) \tag{3.62}$$

are less than one in absolute value. But, since it was assumed that the process X(n) was stationary over the length of the filter, the elements on the diagonal of the matrix D_n are identical. That is:

$$E[d_{ii,n}] = R_{XX}(0) \quad . \tag{3.63}$$

Thus, the matrix in (3.62) may be rewritten as:

$$(I - \beta \frac{1}{R_{XX}(0)} \Phi_{XX}) \quad . \tag{3.64}$$

The matrix in (3.54) is of the form (I-kA) and, as was shown in Chapter 2, the eigenvalues of this form are given by:

$$\tilde{\lambda}_i = (1-K\,\lambda_i) \qquad i=0,1,\ldots,N-1$$

where λ_i are the eigenvalues of the matrix A. Thus, the eigenvalues of the matrix (3.64) are given by:

$$\tilde{\lambda}_i = (1 - \frac{\beta}{R_{XX}(0)} \lambda_i) \qquad i=0,1,\ldots,N-1 \tag{3.65}$$

where λ_i are the eigenvalues of the autocorrelation matrix Φ_{XX}. Since the restrictions on the eigenvalues of the iteration matrix require that:

$$|1 - \frac{\beta}{R_{XX}(0)} \lambda_i| < 1 \qquad i = 0,1,\ldots,N-1$$

and β was assumed positive and all the λ_i are positive, the Jacobi Relaxation Method will converge provided that:

$$0 < \beta < 2/\lambda_{max} \cdot R_{XX}(0) \tag{3.66}$$

where λ_{max} is the largest eigenvalue of Φ_{XX}.

3.3.1 Selection of β to Insure Convergence

Equation (3.66) gives a relationship which allows the selection of the relaxation parameter β to insure convergence of the Jacobi Relaxation Method. However, application of (3.66) requires a priori knowledge of the largest eigenvalue of Φ_{XX} and $R_{XX}(0)$. This information may not be available in the general case where little or no a priori knowledge of the statistics of the process X(n) is assumed. In this case, a value for β which will guarantee convergence can be obtained by using the results of Section 3.3.2 to obtain a bound on λ_{max} which can then be used in (3.66) to determine an appropriate value of β. Equation (3.39) states that if $\tilde{\lambda}$ is an eigenvalue of the matrix:

$$(I-E[C_n]^{-1}\Phi_{XX})$$

then $\tilde{\lambda}$ satisfies the following inequality:

$$|(\tilde{\lambda}-1)E[C_{ii}]+\phi_{ii}| \leq \sum_{\substack{j=1 \\ j\neq 1}}^{N} |(\tilde{\lambda}-1)C_{ji}+\phi_{ij}| \quad . \qquad (3.39)$$

In the Jacobi Relaxation Method, the matrix C_n is the diagonal matrix whose diagonal is the same as that of the matrix $\hat{\Phi}_{XX,n}$ multiplied by 1/β. Thus,

$$E[C_{ij}] = \begin{cases} 1/\beta\ \phi_{ij} & i=1 \\ 0 & i\neq j \end{cases} \quad .$$

Therefore, (3.39) can be written for the case of Jacobi Relaxation as:

$$|(\tilde{\lambda}-1)\ \frac{1}{\beta}\ \phi_{ii}+\phi_{ii}| \leq \sum_{\substack{j=1 \\ j\neq 1}}^{N} |\phi_{ij}| \quad . \tag{3.67}$$

In the case of a stationary process X(n), the elements of the matrix Φ_{XX} are given by:

$$\phi_{ij} = R_{XX}(i-j) \quad .$$

Thus, (3.67) can be rewritten as:

$$|(\tilde{\lambda}-1)\ \cdot\ \frac{1}{\beta}\ R_{XX}(0)+R_{XX}(0)| \leq \sum_{\substack{j=1 \\ j\neq 1}}^{N} |R_{XX}(i-j)| \quad . \tag{3.68}$$

If $\tilde{\lambda}$ is an eigenvalue of the matrix in (3.64), then:

$$(I - \frac{\beta}{R_{XX}(0)}\ \Phi_{XX})\underline{\Psi} = \tilde{\lambda}\ \underline{\Psi}$$

where $\underline{\Psi}$ is the eigenvector corresponding to $\tilde{\lambda}$. Thus:

$$\Phi_{XX}\underline{\Psi} = \frac{R_{XX}(0)}{\beta}\ (\tilde{\lambda}-1)\underline{\Psi} \quad .$$

Therefore, the constant $\frac{R_{XX}(0)}{\beta}(\tilde{\lambda}-1)$ is an eigenvalue of the autocorrelation matrix Φ_{XX}. With this substitution, (3.68) can be expressed as:

$$|\lambda + R_{XX}(0)| \leq \sum_{\substack{j=1 \\ j \neq 1}}^{N} |R_{XX}(i-j)| \quad . \tag{3.69}$$

Now, since λ is an eigenvalue of the positive definite matrix Φ_{XX} and $\underline{R}_{XX}(0) = E[X^2_{(n)}]$, both λ and $R_{XX}(0)$ are greater than zero. Thus, from (3.69):

$$\lambda \leq \sum_{\substack{j=1 \\ j \neq 1}}^{N} |R_{XX}(i-j)| - R_{XX}(0) \quad . \tag{3.70}$$

But, since the autocorrelation function has a maximum at zero, the following inequality can be obtained from (3.70):

$$\lambda_{max} < (N-2) R_{XX}(0) \tag{3.71}$$

where $N=(n_2-n_1+1)$ is the length of the filter. Thus, substituting (3.71) into (3.66) the following bound on the value of β which will insure convergence is obtained:

$$0 < \beta < \frac{2}{N-2} \quad . \tag{3.72}$$

The bound given by (3.72) allows the selection of a value for the relaxation parameter β which will guarantee convergence of the Jacobi Relaxation Algorithm regardless of the values of the eigenvalues of the autocorrelation matrix. However, the selection of a value for β which is greater than (2/N-2) may also cause the algorithm to converge and the corresponding rate of convergence will be more favorable.

3.3.2 Rate of Convergence of the Mean-Square Error

It is obvious from equation (3.66) that the parameter β must be chosen within a certain range to insure convergence. However, the value of β chosen also effects the rate of convergence of the algorithm. Specifically, the convergence of Jacobi Relaxation can be studied by the use of equations (3.54) and (3.55) which were derived in Section 3.2.3. From these equations, it can be seen that for the Jacobi Relaxation Method, a bound on the rate of convergence is:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i \tilde{y}_n^2(i) \qquad (3.73)$$

where:

$$\underline{\tilde{Y}}_{n+1} = (I - \frac{\beta}{R_{XX}(0)} \Lambda)\ \underline{\tilde{Y}}_n \qquad (3.74)$$

and λ_i and M are the eigenvalues and the modal matrix respectively of the autocorrelation matrix Φ_{XX}.

The approximate bound can also be obtained by recognizing that in this special case, the modal matrix M diagonalizes the iteration matrix. Thus, equation (3.57) yields:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i (1 - \frac{\beta}{R_{XX}(0)} \lambda_i)^{2n} (\tilde{h}_0(i)-\tilde{h}^*(i))^2. \qquad (3.75)$$

where $\underline{\tilde{h}}_0$ and $\underline{\tilde{h}}^*$ are the uncoupled initial unit-sample response vector and the optimal uncoupled unit-sample response vector, respectively. Thus, the convergence of Jacobi Relaxation may be studied more easily by the alternate form (3.75). The rate of

convergence for various values of the relaxation parameter β and the relationship of the rate of convergence of the LMS Gradient Algorithm will be considered in Chapter V.

3.3.3 Alternate Form for the Jacobi Relaxation Method

The form for Jacobi Relaxation given in (3.6.1) is useful when considering the algorithm as an example of the general fixed-point iteration method considered in Section 3.2. Specifically, using the form of (3.61) allowed the application of the results on convergence derived in Section 3.2 to the case of the Jacobi Relaxation Method. However, when the implementation of the algorithm is considered, an alternate form for the Jacobi Relaxation Method is useful. In addition, the alternate form allows a more intuitive approach to understanding the algorithm.

The alternate form is obtained by considering the i^{th} component h_i of the unit sample response vector $\underline{h}_n$. From (3.61), $h_{i,n+1}$ is given by:

$$h_{i,n+1} = h_{i,n} - \frac{\beta}{\hat{\phi}_{ii,n}} \sum_{j=1}^{N} \hat{\phi}_{ij,n} h_{j,n} + \frac{\beta}{\hat{\phi}_{ii,n}} \hat{R}_{i,n} \quad i=1,2,\ldots N. \tag{3.76}$$

where $\hat{\phi}_{ij,n}$ and $\hat{R}_{i,n}$ are the $(i,j)^{th}$ element of $\hat{\Phi}_{XX,n}$ and the i^{th} component of $\hat{\underline{R}}_{XX,n}$ respectively. The alternate form (3.76) may be more useful for the implementation of the algorithm than the vector form (3.61) since the matrix multiplications have been broken down into scalar multiplications and additions.

In the special case where the relaxation parameter β equals one, equations (3.61) and (3.76) are simply alternate forms for the familiar Jacobi Iteration Algorithm. In the case where $\beta=1$, the algorithm can be easily explained. Starting with an initial "guess" $\underline{h}_n$ for the solution, the first equation of the linear system is solved for the first component of the vector $\underline{h}$. A value for this component is computed by substituting the components of the vector $\underline{h}_n$ for the corresponding components of $\underline{h}$ on the right-hand side of the resulting equation. This value then becomes the first component of the vector $\underline{h}_{n+1}$. Each equation of the linear system in turn is treated in the same fashion. That is, the i^{th} equation is solved for the i^{th} component of the vector $\underline{h}$ and a value for the i^{th} component of $\underline{h}_{n+1}$ is computed by substituting the necessary components of the vector $\underline{h}_n$ for the corresponding components of $\underline{h}$ on the right-hand side of the resulting equation. Once all N components of $\underline{h}_{n+1}$ are obtained one step of the iteration has been completed. Equation (3.76) with $\beta=1$ is the relationship required to carry out the process just described.

Another way of viewing this algorithm is to note that the procedure (with $\beta=1$) finds the i^{th} component of the vector $\underline{h}_{n+1}$ in a way which would make the i^{th} component of the residual vector:

$$\underline{r}_n = \underline{\hat{R}}_{XX,n} - \hat{\Phi}_{XX,n}\,\underline{h}_n \qquad (3.77)$$

equal to zero if the i^{th} component of $\underline{h}_n$ was replaced by the i^{th} component of $\underline{h}_{n+1}$. However, if $\beta\neq1$ is used in (3.76), then the

i^{th} component of $\underline{h}_{n+1}$ is found in a way which would make the corresponding component of $\underline{r}_n$ not equal to zero if the i^{th} component of $\underline{h}_{n+1}$ is substituted for the i^{th} component of $\underline{h}_n$. Specifically, the i^{th} component of $\underline{r}_n$ computed using (3.77) with the i^{th} component of $\underline{h}_n$ replaced by the i^{th} component of $\underline{h}_{n+1}$ will be less than zero if $\beta > 1$ and greater than zero if $\beta < 1$. Thus, with $\beta > 1$ the algorithm is overcorrecting or "overrelaxing" and with $\beta < 1$ the algorithm is undercorrecting or "underrelaxing". If the iteration "converges", all components of the residual vector r will be essentially equal to zero. Obviously, $\underline{r}_n$ cannot be made exactly equal to zero in all cases with finite precision arithmetic. Thus, intuitively it can be seen that the progress that Jacobi Relaxation makes toward the solution vector at each iteration step is directly dependent on the value chosen for β. This conclusion is confirmed by the discussion on the rate of convergence of the algorithm presented in the last section.

The Jacobi Relaxation Method is one example of a fixed-point iterative algorithm which can be applied to the adaptive estimation problem. It has been shown that the algorithm converges provided that certain restrictions on the value of the relaxation parameter β are met. A bound on the value of β which will insure convergence of the algorithm was developed. This bound is independent of the statistics of the process. In addition, a bound on the rate of convergence of the algorithm is given. A comparison of the theoretical bounds on the rates of convergence of the Jacobi Relaxation Method, the LMS Gradient

Method and the Successive Over-Relaxation Method presented in the next section will be given in Chapter V. The relationship between the theoretical bounds and actual performance will be explored via computer simulation in Chapter VI.

3.4 The Successive Over-Relaxation Method for Adaptive Fixed-Point Iteration

The Jacobi Relaxation Algorithm was obtained by the choice of a particular diagonal matrix given by (3.60) for the C_n in the general Adaptive Fixed-Point Iteration formula (3.33). The diagonal form is easily invertable and the particular matrix chosen insured the convergence of the algorithm provided that the relaxation parameter β was chosen to be within a certain range. In the Successive Over-Relaxation Method (SOR), the matrix C_n is chosen to be the sum of a diagonal matrix and a lower triangular matrix. This form is also easily invertable and can be chosen so as to insure the convergence of the algorithm. That is, a matrix C_n of this form can be easily chosen so that the eigenvalues of the matrix:

$$(I-E[C_n]^{-1}\ \Phi_{XX}) \tag{3.78}$$

are strictly less than one in absolute value.

The symmetric matrix $\hat{\Phi}_{XX,n}$ can be decomposed as in (3.59) into the sum:

$$\hat{\Phi}_{XX,n} = L_n + D_n + L_n^T \tag{3.59}$$

where L_n is a strictly lower triangular matrix and D_n is a diagonal matrix. In the SOR method, the matrix C_n is chosen to be:

$$C_n = \frac{1}{\beta} D_n + L_n \tag{3.79}$$

where the constant parameter $\beta > 0$ is referred to as the relaxation parameter. Thus, the SOR method for Adaptive Fixed-Point Iteration is defined as:

$$\underline{h}_{n+1} = (I-\beta(D_n+\beta L_n)^{-1}\hat{\Phi}_{XX,n})\underline{h}_n+\beta(D_n+\beta L_n)^{-1}\hat{\underline{R}}_{XX,n}. \tag{3.80}$$

Therefore, by the results of Section 3.2.1, the SOR method will converge if and only if the eigenvalues of the matrix:

$$(I-\beta E[D_n+\beta L_n]^{-1}\Phi_{XX}) \tag{3.81}$$

are strictly less than one in absolute value. However, it is difficult to determine the eigenvalues of the matrix in (3.81). Fortunately, it is possible to prove that all the eigenvalues of the iteration matrix are less than one in absolute value for a range of the relaxation parameter β which does not depend on the eigenvalues. In order to prove the convergence of the SOR method, the following theorem is needed. A proof of the theorem can be found in [20].

Theorem 3.1

Let A be a real matrix. Then all the eigenvalues of A are strictly less than one in absolute value if and only if (I-A) is non-singular and there exists a real positive definite matrix Q such that:

$$P = HQ + QH^T \tag{3.82}$$

is positive definite, where:

$$H = (I-A)^{-1}(I+A) \quad . \tag{3.83}$$

Applying Theorem 3.1 to the SOR Method, the matrix A is given by:

$$A = (I-\beta E[D_n+\beta L_n]^{-1}\Phi_{XX}) \quad .$$

Thus, the matrices (I-A) and (I+A) are:

$$(I-A) = \beta E[D_n+\beta L_n]^{-1}\Phi_{XX} \tag{3.84}$$

$$(I+A) = 2I-\beta E[D_n+\beta L_n]^{-1}\Phi_{XX} \quad . \tag{3.85}$$

Obviously, the matrix (I-A) as given in (3.84) is non-singular since both $E[D_n+\beta L_n]^{-1}$ and Φ_{XX} are non-singular. Thus, the matrix $H = (I-A)^{-1}(I+A)$ for the SOR Method is:

$$H = \frac{1}{\beta}\,\Phi_{XX}^{-1}\,E[D_n+\beta L_n](2I-\beta E[D_n+\beta L_n]^{-1}\Phi_{XX})$$

or, equivalently:

$$H = 2/\beta\;\Phi_{XX}^{-1}\,E[D_n+\beta L_n] - I$$

$$H = 2/\beta\;\Phi_{XX}^{-1}[E[D_n]+\beta E[L_n]-\beta/2\;\Phi_{XX}] \quad .$$

Thus, H can be written as:

$$H = \Phi_{XX}^{-1}\;[\frac{2-\beta}{\beta}\,E[D_n]-E[L_n^T]+E[L_n]] \quad . \qquad (3.86)$$

Now, a positive definite matrix Q must be found so that the matrix in (3.83) is positive definite. The autocorrelation matrix is positive definite by Theorem 1.1. Thus, its inverse is also positive definite. If Φ_{XX}^{-1} is chosen for the matrix Q, then the expression in (3.83) may be rewritten as:

$$P = H\;\Phi_{XX}^{-1} + \Phi_{XX}^{-1}\;H^T \quad . \qquad (3.87)$$

Substituting from (3.86) for H in (3.87) yields:

$$P=\Phi_{XX}^{-1}[\frac{2-\beta}{\beta}E[D_n]-E[L_n^T]+E[L_n]]\Phi_{XX}^{-1}+\Phi_{XX}^{-1}[\frac{2-\beta}{\beta}E[D_n]-E[L_n^T]^T+E[L_n]^T]\Phi_{XX}^{-1}. \qquad (3.88)$$

But, $E[L_n^T]^T=E[L_n]$. Thus, (3.88) can be rewritten as:

$$P = 2\left(\frac{2-\beta}{\beta}\right) \Phi_{XX}^{-1} E[D_n] \Phi_{XX}^{-1} \qquad (3.89)$$

which is positive definite for any value of β in the range:

$$0 < \beta < 2 \quad . \qquad (3.90)$$

Thus, by Theorem 3.1, all the eigenvalues of the matrix (3.81) are strictly less than one in absolute value provided that the restrictions on β given by (3.90) are met. Thus, by Theorem 2.1, the SOR Method will converge for any value of the relaxation parameter between 0 and 2. Therefore, no knowledge of the statistics of the process is required to determine a value for β which will insure convergence of the SOR Method. However, the selection of the value of β has an important effect on the rate of convergence of the SOR Method as will be shown in the next section.

3.4.1 Rate of Convergence of the Mean-Square Error

In the previous section, it was shown that the SOR Method converges for any value of the relaxation parameter β such that $0 < \beta < 2$. However, the value of β also effects the rate of convergence of the algorithm. The convergence of the SOR Method can be studied by the use of equations (3.54) and (3.55) from Section 3.2.3. From these equations, it can be seen that for the SOR Method, a bound on the rate of convergence can be determined from:

$$MSE(n)-MSE^*(n) = \sum_{i=1}^{N} \lambda_i \tilde{y}_n^2(i) \qquad (3.91)$$

and:

$$\underline{\tilde{Y}}_{n+1} = (I-M^T\beta E[D_n+\beta L_n]^{-1}M\Lambda)\ \underline{\tilde{Y}}_n \qquad (3.92)$$

where λ_i and M are the eigenvalues and modal matrix respectively of the autocorrelation matrix Φ_{XX}. Now, in general, the modal matrix M of Φ_{XX} will not diagonalize the matrix $E[D_n+\beta L_n]^{-1}$. Thus, the alternate form for the rate of convergence given by (3.57) cannot be used to study the convergence of the SOR Method. However, equations (3.91) and (3.92) can be used to generate the bound sequentially even though the amount of computational effort required limits the usefulness of this approach when the length of the filter is large. The rate of convergence of the SOR Method for various values of β and its relationship to the rates of convergence of the LMS Gradient Algorithm and the Jacobi Relaxation Method will be considered in Chapter V.

3.4.2 Alternate Form for the Successive Over-Relaxation Method

The SOR Method for Adaptive Fixed-Point Iteration was given in the general form for fixed-point iteration in (3.80). The use of this vector form allows the SOR Method to be fitted into the general theory of fixed-point iteration. Specifically, use of (3.80) allows the application of the results on convergence in Section 3.2 to the SOR Method. However, when the implementation of the SOR Method is considered, an alternate form for the algorithm may be useful. The alternate form is

obtained by considering the i^{th} component $h_{i,n}$ of the unit-sample response vector $\underline{h}_n$. From (3.80), $\underline{h}_{n+1}$ may be written as:

$$\underline{h}_{n+1} = (1-\beta)\underline{h}_n - \beta D_n^{-1} L_n \underline{h}_{n+1} - \beta D_n^{-1} L_n^T \underline{h}_n + \beta D_n^{-1} \hat{\underline{R}}_{XX,n} \quad .$$

Thus, the i^{th} component $h_{i,n+1}$ of the vector $\underline{h}_{n+1}$ is given by:

$$h_{i,n+1} = (1-\beta)h_{i,n} - \frac{\beta}{\hat{\phi}_{ii,n}} \left[\sum_{j=1}^{i-1} h_{i,n+1} \hat{\phi}_{ij,n} + \sum_{j=i+1}^{N} h_{i,n} \hat{\phi}_{ij,n} \right]$$

$$+ \frac{\beta}{\hat{\phi}_{ii,n}} \hat{R}_{i,n} \qquad (3.93)$$

where $\hat{\phi}_{ij,n}$ and $\hat{R}_{i,n}$ are the i,j^{th} element of the matrix $\hat{\Phi}_{XX,n}$ and the i^{th} component of the vector $\hat{\underline{R}}_{XX,n}$ respectively. Since the matrix products have been broken down into scalar multiplications and additions, the alternate form (3.93) may be more useful for implementation than the vector form (3.80) for the SOR Method. In addition, the alternate form allows a more intuitive approach to understanding the algorithm.

In the special case where the relaxation parameter β equals one, equations (3.80) and (3.93) are simply alternate forms for the well-known Gauss-Seidel Iteration Algorithm. Thus, in the case where $\beta=1$, the algorithm can be easily explained and is in fact similar to the Jacobi Iteration Algorithm except that each new element of the unit-sample response vector immediately replaces $h_{i,n}$ and is used in computing the remaining elements of $\underline{h}_{n+1}$. Therefore, starting with an initial "guess"

$\underline{h}_n$ for the solution, the first equation is solved for the first component of the vector $\underline{h}$. A value for this component is then computed by substituting the components of the vector $\underline{h}_n$ for the corresponding components of $\underline{h}$ on the right-hand side of the resulting equation. This value then becomes the first component of the vector $\underline{h}_{n+1}$. The remaining N-1 equations are then treated in the following manner. The i^{th} equation is solved for the i^{th} component of $\underline{h}$ and a value for the i^{th} component of $\underline{h}_{n+1}$ is computed by substituting the first (i-1) elements of $\underline{h}_{n+1}$ and the last (N-i) elements of $\underline{h}_n$ for the corresponding elements of $\underline{h}$ on the right-hand side of the resulting equation. Once all N components of $\underline{h}_{n+1}$ have been computed, one step of the iteration has been completed. Equation (3.92) with $\beta=1$ is the recursive relationship for carrying out the process just described.

The SOR algorithm with $\beta=1$ can be viewed in another way by noting that the i^{th} component of the vector $\underline{h}_{n+1}$ is chosen in such a way as to make the i^{th} component of the residual vector

$$\underline{r}_n = \hat{\underline{R}}_{XX,n} - \hat{\Phi}_{XX,n}\,\underline{h}_n \tag{3.77}$$

equal to zero if the first i components of $\underline{h}_n$ were replaced by the first i components of $\underline{h}_{n+1}$. However, if $\beta\neq1$ is used for the relaxation parameter, then the i^{th} component of $\underline{h}_{n+1}$ is found in a way which would not make the corresponding component of $\underline{r}_n$ equal to zero if the first i components of $\underline{h}_n$ were re-

placed by the corresponding components of $\underline{h}_{n+1}$. Specifically, the i^{th} component of $\underline{r}_n$ computed as described above would be less than zero if $\beta > 1$ and greater than zero if $\beta < 1$. Therefore, with $\beta < 1$ the algorithm is undercorrecting or "underrelaxing" and with $\beta > 1$ the algorithm is overcorrecting or "overrelaxing". Convergence of the algorithm would imply that all components of $\underline{r}_n$ would approach zero. However, due to the limitations imposed by finite precision arithmetic, $\underline{r}_n$ cannot be made exactly equal to $\underline{0}$ in practice for most systems. Thus, intuitively it can be seen that the progress that the SOR Method makes toward a solution at each step is directly dependent on the parameter β. This observation is confirmed by the results concerning the rate of convergence of the algorithm presented in Section 3.4.1.

The SOR Method is another example of a fixed-point iterative algorithm which can be applied to the adaptive estimation problem. It has been shown that the SOR Method converges provided that the relaxation parameter β is chosen so that $0 < \beta < 2$. This result is independent of the statistics of the process X(n). In addition, a bound on the rate of convergence of the algorithm was given. A comparison of the theoretical bounds on the rates of convergence of the mean-square error for the SOR Method, the Jacobi Relaxation Method and the LMS Gradient Method will be presented in Chapter V. The relationship between the theoretical bounds and actual performance will be studied in Chapter VI via computer simulation.

3.5 Number of Operations Required by Adaptive Fixed-Point Iteration

As mentioned in Chapter II, one of the advantages of the LMS Gradient Algorithm is its simplicity. The number of arithmetic operations required at each sample time n is on the order of the length of the filter N. Obviously, the Adaptive Fixed-Point Iteration Method proposed here requires a considerably larger number of computations at each sample step. In this section, the number of operations required by the Adaptive Fixed-Point Iteration Method will be determined and methods for the reduction of the number of operations will be explored.

The Adaptive Fixed-Point Iteration Algorithm can be broken down into three procedures, the estimation of the autocorrelation matrix and vector, the fixed-point iteration, and the computation of the filter output. As mentioned in Section 3.1, the elements of the autocorrelation matrix and the autocorrelation vector are merely sample values of the autocorrelation function $R_{XX}(k)$ for stationary processes. Also, since Φ_{XX} has a Topelitz structure for stationary processes, only N samples of $R_{XX}(k)$ are required to determine all of the elements of the autocorrelation matrix. In addition N samples of $R_{XX}(k)$ are needed to make up the autocorrelation vector $\underline{R}_{XX}$. However, if the delay n_i is less than N, then only $(N+n_1)$ samples of $R_{XX}(k)$ are needed to determine both Φ_{XX} and $\underline{R}_{XX}$. But, if $n_1 > N$, then 2N samples of $R_{XX}(k)$ are required to determine both the autocorrelation matrix and vector. In Section 3.1.3 it was proposed that a moving average be computed recursively to obtain

an estimate of the autocorrelation function. From (3.25) it can be seen that 3 multiplications and 2 additions are required to update the estimate of $R_{xx}(k)$ for each required value of k. Thus, at each sample time n, the following number of operations are required to update the estimates of the autocorrelation matrix and the autocorrelation vector:

Delay Value	Additions	Multiplications
$n_1 \leq N$	$2(N+n_1)$	$3(N+n_1)$
$n_1 > N$	4N	6N

Figure 3.1 Required number of operations/sample for the estimation of Φ_{xx} and $\underline{R}_{xx}$ by the moving average proposed in Section 3.1.3.

Thus, it can be seen from Figure 3.1 that the number of operations required to estimate Φ_{xx} and $\underline{R}_{xx}$ is on the order of the length of the filter N for the moving average estimator proposed in Section 3.1.3.

Estimation of the autocorrelation matrix and vector requires on the order of N operations per sample for the estimate considered here. Unfortunately, the fixed-point iteration procedure requires a considerably larger number of operations per sample. It can be seen from (3.76) and (3.93) that if these expressions are used directly to implement the Jacobi Relaxation Method and the SOR Method respectively, the required number of

computations will be on the order of the square of the length of the filter, or N^2, per sample. Figure 3.2 gives the specific number of operations per sample required for the Jacobi Relaxation Method and the SOR method if they are implemented directly using (3.76) and (3.93) respectively.

Algorithm	Additions	Multiplications
Jacobi Relaxation	N(N+1)	N(N+1)
SOR	N(N+1)	N(N+1)

Figure 3.2 Required number of operations/sample for Jacobi Relaxation and SOR Methods computed directly using (3.76) and (3.93) respectively. N = filter length.

The final procedure of the Adaptive Fixed-Point Iteration method is the computation of the signal estimate or the output of the filter by the use of equation (1.4). This procedure requires N multiplications and (N-1) additions. Thus, the numbers of operations required for Adaptive Fixed-Point Iteration using the Jacobi Relaxation Method and the SOR Method are the sums of the numbers of operations required to implement each of the three procedures discussed above. Figure 3.3 gives the total required number of operations/sample for Adaptive Fixed-Point Iteration using the Jacobi Relaxation Method and the SOR Method.

Operation	Jacobi Relaxation		SOR Method	
	$n_1 \leq N$	$n_1 > N$	$n_1 \leq N$	$n_1 > N$
Additions	$N^2+4N+2n_1-1$	N^2+6N-1	$N^2+4N+2n_1-1$	N^2+6N-1
Multiplications	$N^2+5N+3n_1$	N^2+8N	$N^2+5N+3n_1$	N^2+8N

Figure 3.3 Total required number of operations/sample for the Jacobi Relaxation Method and the SOR Method computed directly using (3.76) and (3.93) respectively with estimation of the autocorrelation matrix and vector by the moving average proposed in Sec. 3.2.1. N = filter length, n_1 = delay in unit sample response.

From Figure (3.3) it can be seen that the $0(N^2)$ operations per sample required by the fixed-point iteration procedure dominates the expression for total number of operations/sample for both the Jacobi Relaxation and the SOR Methods. Thus, a reduction in the number of operations/sample for the fixed-point iteration methods would have a significant effect on the amount of computational time required for these algorithms. Methods for the reduction of the number of operations required for both the iterative methods considered here are discussed in the following sections.

3.5.1 Reduction of Number of Operations/Sample by Use of Partial Iteration Steps

As stated above, both the Jacobi Relaxation Method and the SOR Method require $0(N^2)$ operations/sample in the forms presented in (3.76) and (3.93). Since such computational

complexity may be prohibitive in many applications, it is of interest to consider methods for reducing the number of operations/sample. In contrast to the LMS Gradient Algorithm, the "adaptation" or the estimation of the statistics of the process X(n) is carried out independently of the updating of the unit-sample response. Thus, while the estimates of Φ_{XX} and $\underline{R}_{XX}$ are updated at every sample time n, it is not necessary to update all of the elements of $\underline{h}_n$ by the fixed-point iterative procedure at every sample. Therefore, the following procedure is proposed for updating the unit-sample response vector $\underline{h}_n$. After k samples have been received, ℓ elements of $\underline{h}_n$ will be updated and the process repeated every k samples. For example, with N=8, k=5, and ℓ=3, after 5 samples have been received (n=4), the first 3 components of $\underline{h}$ are updated (components h_1, h_2 and h_3). Then after 5 more samples have been received (n=9), components h_4, h_5 and h_6 will be updated. Finally, at n=14, the next three components (h_7, h_8 and h_1) will be updated. The process continues in a similar fashion with ℓ=3 elements of $\underline{h}$ being updated in order every k=5 samples. When the end of the vector is reached, the first element becomes the next in order. That is, h_1 follows h_N in being updated. If this scheme is used, then the average number of operations per sample for the fixed point iteration portion of the Adaptive Fixed-Point Iteration Method is on the order of:

$$\ell \cdot (N/k)$$

for both the Jacobi Relaxation and the SOR Methods. The specific average number of operations/sample under this scheme is given in Figure 3.4 for both algorithms considered here.

Operation	Jacobi Relaxation	SOR Method
Additions	ℓ/k (N+1)	ℓ/k (N+1)
Multiplications	ℓ/k (N+1)	ℓ/k (N+1)

Figure 3.4 Average number of operations/sample for Jacobi Relaxation and SOR Methods with Partial Iteration steps. N=filter length, ℓ/k=Partial Iteration Step parameter.

Obviously, reduction of the average number of operations by the Partial-Iteration-Step Method will slow the rate of convergence of the algorithms proportionately to the ratio ℓ/k). Thus, computational complexity and rate of convergence can be traded off under the Partial-Iteration-Step Method. The measurement of computational complexity by the average number of operations/sample can be justified by noting that the fixed-point iteration procedure can be carried out by spreading out the $(\ell \cdot N)$ computations over the next k samples since the estimation of the statistics and the iteration can be carried out independently.

Use of the fixed-point iteration method with partial iteration steps is one approach to the reduction of the number of arithmetic operations which must be performed at each sample time n. However, as mentioned previously, the reduction of

operations by the use of partial iteration steps reduces the rate of convergence of the algorithm. Furthermore, it is interesting to note that while the Partial-Iteration-Step Method discussed above was suggested as a means of decreasing the average number of operations per sample, the same procedure could be performed with $\ell/k > N$ so that the number of operations per sample and the rate of convergence is increased. The limiting case for this method would, of course, be the complete solution of the system of linear equations at every sample time n. With the generalization that ℓ and k can take on any integer values greater than 0, the method discussed above will be referred to as the Variable Iteration Step Method. Thus, Adaptive Fixed-Point Iteration using the Variable Iteration Step Method has the advantage that trade-offs can be made between the average number of operations/sample and the rate of convergence of the algorithm. In contrast, the LMS Gradient Algorithm requires a fixed number of operations per sample. The ability to vary the number of operations/sample in the Adaptive Fixed-Point Iteration Method is a very attractive feature since for varying applications different sampling rates will be used. The average number of operations/sample in the Variable Iteration Step Method can be chosen so as to fully utilize the time available between samples.

The theoretical performance of Jacobi Relaxation and the SOR Method under the Variable-Iteration-Step scheme will be studied in Chapter V using the bounds on convergence derived in Section 3.2.3. Specifically, equation (3.54) can be used to

generate the MSE bound recursively. However, the sequence ($\underline{h}_n = \underline{h}^*$) must be generated by the particular partial iteration method as discussed above. The rates of convergence for various values of ℓ and k will be compared to the rate of convergence of the LMS Gradient Algorithm. Relationships between the theoretical predictions and actual performance will be studied through the use of computer simulation in Chapter VI.

3.5.2 Implementation of the Jacobi Relaxation Method Using the Discrete Fourier Transform

The Variable-Iteration-Step Method discussed in the last section decreased the average number of operations/sample by choosing $\ell/k < N$ and thus carrying out a single iteration step over a number of samples rather than making a complete iteration step at each sample time n. This method reduces the number of operations/sample at the expense of slowing the rate of convergence. Obviously, it would be advantageous to reduce the number of operations required for a complete iteration step. This would allow more complete iterations to be performed under the Variable-Iteration-Step Method in a given number of sample periods while keeping the average number of operations/sample constant. In this section it will be shown that with the estimate of the autocorrelation matrix proposed in Section 3.1, the Jacobi Relaxation Method can be implemented using the Discrete Fourier Transform (DFT). Thus, a Fast Fourier Transform (FFT) algorithm can be used to obtain a substantial reduction in the number of operations required to implement Jacobi

Relaxation. This result arises due to the Topelitz structure of the estimate of the autocorrelation matrix and does not hold for the general case of a non-Topelitz autocorrelation matrix.

The alternate form for the Jacobi Relaxation Method is given in equation (3.76) as:

$$h_{i,n+1} = h_{i,n} - \frac{\beta}{\hat{\phi}_{iin}} \sum_{j=1}^{N} \hat{\phi}_{ij,n}\, h_{j,n} + \frac{\beta}{\hat{\phi}_{ii,n}} \hat{R}_{i,n}$$

where h_i, $\hat{\phi}_{ij,n}$ and $\hat{R}_{i,n}$ are the i^{th} component of the unit-sample response vector $\underline{h}$, the $(i,j)^{th}$ element of the estimated autocorrelation matrix $\hat{\Phi}_{XX,n}$ and the i^{th} element of the estimated autocorrelation vector $\underline{\hat{R}}_{XX,n}$ respectively. Now, from equations (3.6) and (3.7):

$$\hat{\phi}_{ij,n} = \hat{R}_{XX,n}(i-j) \qquad (3.6)$$

$$\hat{R}_{i,n} = \hat{R}_{XX,n}(n_1+i-1) \quad . \qquad (3.7)$$

Thus, substituting for $\hat{\phi}_{ij,n}$ and $\hat{R}_{i,n}$ from (3.6) and (3.7) into (3.76), (3.76) can be written as:

$$h_{i,n+1} = h_{i,n} - \frac{\beta}{\hat{R}_{XX,n}(0)} \sum_{j=1}^{N} h_{j,n}\, \hat{R}_{XX,n}(i-j)$$

$$+ \frac{\beta}{\hat{R}_{XX,n}(0)} \hat{R}_{XX,n}(n_1+i-1) \quad . \qquad (3.94)$$

Now, recalling from equation (1.5) that the i^{th} component of the unit-sample response vector is the i^{th} non-zero sample of the unit-sample response, $h_{i,n}$ can be written as:

$$h_{i,n} = h(n_1+i-1)_n \quad .$$

Thus, equation (3.76) for Jacobi Relaxation can be written in terms of the unit sample response h(i) as:

$$h(i+n_1)_{n+1} = h(i+h_1)_n - \frac{\beta}{\hat{R}_{XX,n}(0)} \sum_{j=0}^{N-1} h(j+n_1)_n \hat{R}_{XX,n}(i-j)$$

$$+ \frac{\beta}{\hat{R}_{XX,n}(0)} \hat{R}_{XX,n}(i+n_1)_n \qquad 0 \leq i \leq N-1 \quad . \qquad (3.95)$$

Consideration of equation (3.95) reveals the basis behind the use of the Discrete Fourier Transform for implementing the Jacobi Relaxation Method. The summation in equation (3.95):

$$\sum_{j=0}^{N-1} h(j+n_1)\hat{R}_{XX,n}(i-j)$$

requires N multiplications and (N-1) additions for each value of $0 \leq i \leq N-1$ if computed directly. However, this term is merely the linear convolution of two sequences $h(i+n_1)$ and $\hat{R}_{XX}(i)_n$. Thus, it can be computed using the DFT. However, (3.95) is not in the simplest form for computation. Equation (3.95) can be rewritten as:

$$h(i+n_1)_{n+1} = \frac{\beta}{\hat{R}_{XX,n}(0)} \hat{R}_{XX}(i+n_1)_n + (1-\beta)h(i+n_1)_n$$

$$- \frac{\beta}{R_{XX,n}(0)} \left[\sum_{j=0}^{i-1} h(j+n_1)_n \hat{R}_{XX,n}(i-j) + \sum_{j=i+1}^{N-1} h(j+h_1)_n \hat{R}_{XX,n}(i-j) \right] \quad (3.96)$$

If two new N-point sequences are defined as:

$$R_1(i) = \begin{cases} \hat{R}_{XX,n}(i) & 0 \leq i \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (3.97)$$

$$R_2(i) = \begin{cases} \hat{R}_{XX,n}(i) & 0 < i \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (3.98)$$

then (3.91) can be expressed as:

$$h(i+n_1)_{n+1} = \frac{\beta}{\hat{R}_{XX,n}(0)} \hat{R}_{XX}(i+n_1) + (1-\beta)h(i+n_1)_n$$

$$- \frac{\beta}{R_{XX,n}(0)} \left[\sum_{j=0}^{N-1} h(j+n_1)_n R_1(i-j) + \sum_{j=0}^{N-1} h(j+n_1)_n R_2(i-j) \right]. \quad (3.99)$$

Now, $h(i+n_1)_n$, $R_1(i)$ and $R_2(i)$ are N-point sequences. Therefore, the linear convolution of $h(i+n_1)_n$ and $R_1(i)$ or $R_2(i)$ is a 2N-1 point sequence. Thus, (3.9) can be rewritten as:

$$h(i+n_1)_{n+1} = \frac{\beta}{\hat{R}_{XX,n}(0)} \hat{R}_{XX,n}(i+n_1) + (1-\beta)h(i+n_1)_n$$

$$- \frac{\beta}{R_{XX,n}(0)} \left[h(i+n_1)_n * R_1(i) + h(i+n_1)_n^* R_2(i) \right] \text{RECT}_N(i) \quad (3.100)$$

where the * denotes linear convolution and the sequence $RECT_N(i)$ is defined as:

$$RECT_N(i) = \begin{cases} 1 & 0 \leq i \leq N-1 \\ 0 & \text{otherwise} \end{cases} .$$

Thus, in order to find the N-point sequence $h(i+n_1)_{n+1}$, it is necessary to compute the following 2N-1 point sequence:

$$y(i)_n = [h(i+n_1)*R_1(i)+h(i+n_1)*R_2(i)]$$

and truncate the sequence after the first N points. Thus, if the DFT is used to compute $y(i)_n$, then a DFT of at least 2N-1 points will be required to represent the linear convolution of the sequences. Suppose that Discrete Fourier Transforms of $p \geq 2N-1$ points are computed for the N-point sequences $h(i+n_1)$, $R_1(i)$ and $R_2(i)$ where the remaining (p-N) points are chosen to be zero. The p-point DFT A(k) of an $N < p$ point sequence A(i) padded with (p-N) zeroes is defined as:

$$A(k) = \sum_{i=0}^{N-1} A(i)\, e^{-j\frac{2\pi}{p}ki} \qquad k=0,1,\ldots,p-1 . \qquad (3.101)$$

Provided that the DFTs of the N-point sequence $h(i+n_1)_n$, $R_1(n)$ and $R_2(n)$ are computed as described above, the $p \geq 2N-1$ point DFT of the 2N-1 point sequence $y_n(i)$ is given by the product of the DFTs of $h(i+n_1)_n$ and $(R_1(i)+R_2(i))$. Thus, since the DFT is a linear operator:

$$Y(k)_n = H(k)_n(R_1(k)+R_2(k)) \quad . \tag{3.102}$$

Now, it can be shown that $R_2(k)=R_1^*(k)$ where * denotes the complex conjugate. Thus, (3.102) can be rewritten as:

$$Y(k)_n = H(k)_n(R_1(k)+R_1^*(k))$$

or equivalently:

$$Y(k)_n = 2H(k)_n \, Re[R_1(k)] \tag{3.103}$$

where $Re[R_1(k)]$ is the real part of $R_1(k)$. Thus, the following procedure is proposed for computing $\underline{h}_{n+1}$ by the Jacobi Relaxation Method assuming that $R_1(k)$ is obtained from the autocorrelation estimate proposed in Section 3.1:

Step 1: Compute the $p \geq 2N-1$ point DFT of the N-point sequence $h(i+n_1)_n$ to find $H(k)_n$.

Step 2: Compute $Y(k)_n$ by (3.103).

Step 3: Take the inverse DFT of $Y(k)_n$ to obtain the 2N-1 point sequence $y(i)_n$.

Step 4: Compute the N-point sequence $h(i+n_1)_{n+1}$ as:

$$h(i+n_1)_{n+1} = \frac{\beta}{\hat{R}_{XX}(0)_n}[\hat{R}_{XX}(i+n_1)_n - y(i)_n] + (1-\beta)h(i+n_1)_n \quad . \tag{3.104}$$

The reduction of the number of operations required to carry out one step of the Jacobi Relaxation Method comes when the DFT computations in the above method are made using an FFT

algorithm used for the computation. Usually, however, the number of additions and multiplications are on the order of p log(p). In the discussion that follows, it will be assumed that N is an integer power of 2 and that a 2N point DFT is to be used in the computations required to implement the Jacobi Relaxation Method as proposed above. It is further assumed that a decimation-in-time (DIT) FFT algorithm with in-place computations such as that described in [21] is used to compute the DFT. For a p-point DFT computed using the DIT FFT algorithm considered, $p \log_2(p)$ complex additions and $p/2 \log_2(p)$ complex multiplications are required. Thus, to compute a 2N-point DFT would require $2N (1+\log_2(N))$ complex additions and $n (1+\log_2(N))$ complex multiplications. However, for Step 1 of the procedure, the 2N-point DFT is taken for an N-point sequence padded with N zeroes. Thus, the number of operations required for Step 1 is less than that required for a general 2N-point DFT. Specifically, $2N \log_2(N)$ complex additions and $N \log_2(N)$ complex multiplications are required to carry out Step 1.

Step 2 is simply the multiplication of 2 2N-point transforms and thus requires 2N complex multiplications and 2N complex additions. The complex additions are used in lieu of a multiplication by the real constant 2.

Step 3 requires the computation of an inverse DFT which can also be carried out using the same 2N-point FFT algorithm discussed above. Thus, $2N (1+\log_2(N))$ complex additions and $N (1+\log_2(N))$ complex multiplications are required. However, the number of additions can be reduced by noting that only the first

N-points of the 2N-point result will be used in Step 4. Thus, the number of complex additions can be reduced to N $(1+\log_2(N))$.

Finally, Step 4 is simply the addition of three N-point sequences and the intermediate multiplications by two real constraints. Thus, 2(N-1) real additions and 2N real multiplications are required to complete Step 4.

The total number of operations required to implement the Jacobi Relaxation Method using the procedure proposed is the sum of the numbers of operations required for Steps 1, 2, 3, and 4. Since the other algorithms discussed here require only real operations, the number of complex operations required for the method proposed will be converted into the equivalent number of real operations. Since two real additions are required for each complex addition and four real multiplications and two real additions are required for each complex multiplication, the number of operations given in the above discussion must be increased accordingly. Figure 3.5 gives the number of real arithmetic operations required to implement each step of the proposed procedure.

Operation	Step 1	Step 2	Step 3	Step 4	Total
Additions	$6N \log_2 N$	8N	$6N(1+\log_2 N)$	2(N-1)	$16N+12N \log_2 N-2$
Multiplications	$4N \log_2 N$	8N	$4N(1+\log_2 N)$	2 N	$14N+8N \log_2 N$

Figure 3.5 Number of real operations required to implement Jacobi Relaxation using the FFT.

From Figures 3.2 and 3.5 it can be seen that for small values of the filter length N it is advantageous to do the required computations for Jacobi Relaxation directly using (3.76). Nowever, as N becomes larger ($N \geq 64$), substantial savings in the number of operations required can be obtained by the use of the FFT as outlined above. For example, with N=1024, approximately $1.048 \cdot 10^6$ real multiplications are required to carry out one step of Jacobi Relaxation directly using (3.76). In contrast, roughly $9.523 \cdot 10^4$ real multiplications are required when the FFT is employed. Thus, when impelmented using the FFT, the Jacobi Relaxation Method requires a number of arithmetic computations which is between N and N^2.

In the above discussion it was assumed that the DFT of the sequence $R_1(i)$ defined in (3.97) was known. It will now be shown that $R_i(k)$ can be determined recursively without having to perform an additional DFT at each step. From (3.23) and (3.97) it can be seen that:

$$R_1(i) = \begin{cases} \frac{1}{M} \sum_{\ell=n-M+1}^{N} X(\ell)X(\ell-i) & 0 < i \leq N-1 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, from the definition of the DFT (3.101), the p-point DFT of $R_1(i)_n$ padded with (p-N) zeroes can be written as:

$$R_1(k)_n = \sum_{i=1}^{N-1} \left[\frac{1}{M} \sum_{\ell=n-M+1}^{n} X(\ell)X(\ell-i)\right] e^{-j\frac{2\pi}{p}ki} \quad . \qquad (3.105)$$

By interchanging the order of summation, (3.105) can be rewritten as:

$$R_1(k)_n = \frac{1}{M} \sum_{\ell=n-M+1}^{n} X(\ell) \sum_{i=1}^{N-1} X(\ell-i) e^{-j\frac{2\pi}{p} ki}$$

or, equivalently,

$$R_1(k)_n = \frac{1}{M} \sum_{\ell=n-M+1}^{n} X(\ell) \left[\sum_{i=0}^{N-1} X(\ell-i) e^{-j\frac{2\pi}{p} ki} - X(\ell) \right]. \quad (3.106)$$

Now, define a new sequence as:

$$A_\ell(k) = \sum_{i=0}^{N-1} X(\ell-i) e^{-j\frac{2\pi}{p} ki} - X(\ell) \quad (3.107)$$

so that $R_1(k)_n$ may be expressed as:

$$R_1(k)_n = \frac{1}{M} \sum_{\ell=n-M+1}^{n} X(\ell) A_\ell(k). \quad (3.108)$$

Thus, from (3.108) it can be seen that $R_1(k)_{n+1}$ can be written as:

$$R_1(k)_{n+1} = R_1(k)_n + \frac{1}{M} [X(n+1)A_{n+1}(k) - X(n-M+1)A_{n-M+1}(k)]. \quad (3.109)$$

Therefore $R_1(k)_{n+1}$ can be obtained recursively if $A_n(k)$ can be obtained recursively. Now, from (3.107):

$$A_{n+1}(k) = \left(\sum_{i=0}^{N-1} X(n+1-i) e^{-j\frac{2\pi}{p} ki} \right) - X(n+1).$$

By introducing the change of variables $\ell=i-1$, the above expression can be rewritten as:

$$A_{n+1}(k) = e^{-j\frac{2\pi}{p}k} \sum_{\ell=0}^{N-1} X(n-\ell)e^{-j\frac{2\pi}{p}k\ell} - e^{-j\frac{2\pi}{p}kN} X(n-N+1)$$

or equivalently:

$$A_{n+1}(k) = e^{-j\frac{2\pi}{p}k} \sum_{\ell=0}^{N-1} X(n-\ell)e^{-j\frac{2\pi}{p}k\ell} - e^{-j\frac{2\pi}{p}kN} X(n-N+1).$$

From the above expression it can be seen that $A_{n+1}(k)$ can be expressed recursively as:

$$A_{n+1}(k) = e^{-j\frac{2\pi}{p}k}(A_n(k)+X(n)) - e^{-j\frac{2\pi}{p}kN}(X(n-N+1)). \quad (3.110)$$

Thus, $R_1(k)_n$ can be updated recursively requiring on the order of p operations/sample rather than the $0(p \log(p))$ operations required to compute $R_1(k)$ directly using an FFT algorithm. In the special case considered previously where p=2N, (3.110) may be reduced to the following form:

$$A_{N+1}(k) = e^{-j\frac{\pi}{N}k}(A_n(k)+X(n)) - (-1)^k X(n-N+1). \quad (3.111)$$

Therefore, updating $A_n(k)$ requires 4N real additions and 2N complex multiplications at each sample step.

By considering (3.109) and the above discussion, it can be seen that updating $Re[R_i(k)_n]$ requires 12N real multiplications and 8N real additions which is considerably less than the

$N \log_2(N)$ complex operations required to compute the DFT using a FFT algorithm when $N > 8$.

The computations required by the Jacobi Relaxation Method can be carried out using the FFT method described in this section with recursive updating of $Re[R_1(k)_n]$ to obtain a substantial reduction in the number of operations/iteration step over the direct computation of (3.76) for $N \geq 64$. For large N the number of operations required is dominated by the number of computations required for the FFT algorithm which is on the order of $N \log(N)$. Thus, Jacobi Relaxation realized by this method lies between the LMS Gradient Algorithm and the SOR method in terms of the number of operations required per iteration step. To further reduce the average number of operations/sample, the FFT method can be combined with the Variable Iteration Step Method proposed in Section 3.5.1. Since the majority of the operations required by the FFT method are executed in the computation of the FFT it is most useful to update all N terms of the unit sample response vector every k samples. That is $\ell=N$ in the Variable Iteration Step Method. In this case, the approximately $N \log_2(N)$ operations are distributed over k samples for an average of $N/k \log_2(N)$ operations per sample. Thus, the FFT method for Jacobi Relaxation can be used to reduce the required number of operations/sample either by itself or in conjunction with the Variable-Iteration-Step Method obtaining a substantial savings in computation time for large N provided that the autocorrelation estimate yields an autocorrelation matrix with a Topelitz structure.

3.5.3 Reduction of Computation Time and Parallel Processing

The FFT Method for Jacobi Relaxation and the Variable Iteration Step Method for the SOR Method and Jacobi Relaxation are aimed at the reduction of the number of operations required per sample as a means for reducing the amount of computational time required by the algorithms. Another approach to the reduction of computation time is to increase the speed with which the required group of operations can be carried out. One method for achieving this goal is to carry out several operations concurrently or, to process in parallel. Adaptive Fixed-Point Iteration is well suited to parallel processing since the required tasks can be divided into three distinct areas, the updating of the estimates $\hat{\Phi}_{XX,n}$ and $\hat{\underline{R}}_{XX,n}$, the fixed-point iteration, and the computation of the filter output. In addition, the fixed-point iteration algorithms discussed here are also well suited to parallel processing in themselves [22]. Thus, while not of primary interest here, parallel processing is a viable method for the reduction of computational time in Adaptive Fixed-Point Iteration.

Another approach for the reduction of computational time can be developed by noting that the majority of the computations required by the Jacobi Relaxation Method and the SOR Method are involved with computing sums of the form:

$$y_i = \sum_{i=0}^{N-1} h_i \hat{\Phi}_{ij}$$

which, in terms of the unit sample response and the autocorrelation function estimate proposed in Section 3.1, can be written as:

$$y(j) = \sum_{i=n_1}^{n_2} h(i)\hat{R}_{XX,n}(j-i+n_1) \quad . \qquad (3.112)$$

Now, the above expression is identical to the convolution sum required to produce the filter output except that X(i-j) is replaced by $\hat{R}_{XX,n}(j-i+n_1)$. Thus, the filter structure itself can be used to carry out part of the computations required to update the unit sample response. Since in many cases high speed special purpose hardware is used to implement the transversal filter, this approach may allow very rapid computation of the sums required.

Thus, parallel processing and use of any special hardware used for implementing the filter are two approaches to the reduction of the time required to carry out the necessary computations for Adaptive Fixed-Point Iteration using the Jacobi Relaxation Method and the SOR Method. Again, as was the case with the FFT method, either of these two approaches can be used in conjunction with the Variable Iteration Step Method to reduce the average number of computations per sample step.

3.6 Conclusion

Adaptive Fixed-Point Iteration has been proposed as a method for the approximate solution of the Wiener-Hopf equation

for a finite impulse response digital filter. A direct estimate which is consistent was chosen for the estimation of the autocorrelation matrix Φ_{XX} and the autocorrelation vector $\underline{R}_{XX}$ for the case of a stationary input process X(n). For the more general case, a moving average was proposed as an estimate of these statistics and a bound on the variance of this estimate as a function of the length M of the data window was derived. In this development it was assumed that the statistics of the process X(n) vary slowly when compared to M and to the filter length N. This estimate yields an estimated autocorrelation matrix having a Topelitz structure which is advantageous in processing. However, the Adaptive Fixed-Point Iteration approach is also applicable when other estimates which do not force such a structure on the autocorrelation matrix are used.

A general form for fixed-point iteration algorithms was presented and conditions for convergence in the mean were established. In addition a bound on the rate of convergence of the mean square error to the minimum MSE produced by the optimal filter was established. Two particular examples of fixed-point iteration were considered in detail. Specifically, the Jacobi Relaxation Method and the SOR Method were proposed as algorithms for the implementation of an Adaptive Fixed-Point Iteration scheme. Conditions on a parameter β present in both algorithms were determined to insure convergence; independent of the statistics of the process. Two schemes for the reduction of the average number of operations per sample were presented.

The Variable Iteration Step Method is applicable to both algorithms while the FFT Method applies only to Jacobi Relaxation used in conjunction with an autocorrelation matrix estimate which has a Topelitz structure. Also, the use of parallel processing and any available special purpose transversal filter hardware for the reduction of the computation time required was discussed briefly.

It is believed that Adaptive Fixed-Point Iteration offers an extremely attractive solution to the problem of adaptively determining an approximation to the optimal filter when some increase in complexity over the LMS Gradient Algorithm can be tolerated. In particular, the Jacobi Relaxation Method and the SOR Method can be made to converge by a choice of the relaxation parameter β which is independent of the statistics of the input process $X(n)$. In addition, the use of the Variable Iteration Step Method makes Adaptive Fixed-Point Iteration extremely flexible in that trade-offs can be made between the rate of convergence and the average number of operations required per sample. Comparisons between the theoretical performance of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Method will be considered in Chapter V. Relationships between the theoretical and actual performance of the three algorithms will be explored via computer simulation in Chapter VI. In all cases, rate of convergence and required number of operations per sample will be the variables of interest. It will be shown in these chapters that Adaptive Fixed-Point Iteration offers

significantly better performance than the LMS Gradient Algorithm without a cost in computational time which would be prohibitive in many practical problems.

CHAPTER IV

NON-ITERATIVE SOLUTION -- THE LEVINSON ALGORITHM

The LMS Gradient Algorithm discussed in Chapter II and the Adaptive Fixed-Point Iteration Method proposed in Chapter III take somewhat similar approaches to the approximate solution of the discrete finite impulse response Wiener-Hopf equation given by (1.13). These methods are similar in that both make estimates of the statistics of the process and attempt to find an approximate solution to the problem by the use of an iterative technique. The LMS Gradient Method uses an instantaneous estimate of the gradient of the mean-square error and applies a Steepest Descent type method at each sample point while the Adaptive Fixed-Point Iteration Algorithm uses estimates of the autocorrelation matrix and vector and attempts to solve a system of the form:

$$\hat{\Phi}_{XX}\,\underline{h} = \hat{\underline{R}}_{XX} \tag{3.26}$$

by applying a complete or partial step of a fixed-point iteration method every k sample periods. The estimates of the autocorrelation matrix and the autocorrelation vector used in (3.26) are restricted only in that they should be at least asymptotically

unbiased and have a small covariance. No special structure is imposed upon the estimate of the autocorrelation matrix in order for the Adaptive Fixed-Point Iteration Method to be applicable. One possible estimate for use in Adaptive Fixed-Point Iteration was proposed in Section 3.1. This particular estimate imposes a Topelitz structure on $\hat{\Phi}_{XX}$. That is, each element of $\hat{\Phi}_{XX}$ can be defined as:

$$\phi_{ij} = f(i-j) \tag{4.1}$$

where f is some function of (i-j). Specifically, for the estimate considered in Section 3.1, ϕ_{ij} is given by (3.6) as:

$$\phi_{ij} = \hat{R}_{XX}(i-j)_M \quad . \tag{3.6}$$

The Topelitz structure of the autocorrelation matrix estimate given by (3.6) led to a reduction of the number of operations required to implement each step of the Jacobi Relaxation Method as discussed in Section 3.5.2. However, the Topelitz structure also makes possible the use of very efficient non-iterative algorithms for the solution of the system of equations given by (3.26). In this chapter, the Levinson Algorithm for the solution of the system (3.26) where $\hat{\Phi}_{XX}$ is a Topelitz matrix will be considered [23]. This algorithm is non-iterative in that the solution of (3.26) is obtained in a fixed number of steps. As will be shown later, the Levinson

Algorithm requires on the order of N^2 operations to completely solve (3.26) where $\hat{\Phi}_{XX}$ is a Hermetian Topelitz matrix.

4.1 The Levinson Algorithm

The exact solution of a system of N linear equations of the form (3.26) requires on the order of N^3 operations in the general case when using any of the standard methods for solving linear equations. Iterative methods require fewer operations if the convergence rate is high enough but do not yield exact solutions. However, if the system in (3.26) is Hermetian and Topelitz, then the solution can be found most efficiently by the use of Levinson's Algorithm [23]. In the non-Hermitian case, the Trench Algorithm can be used [24]. These algorithms utilize the fact that the first row and column of the inverse of a Topelitz matrix uniquely determine the remaining elements of the inverse. The use of the Levinson or Trench Algorithms leads to substantial savings in the number of operations required for the exact solution of (3.26).

Since the autocorrelation matrix estimate proposed in Section 3.1 produces a matrix $\hat{\Phi}_{XX}$ which is Hermitian and Topelitz, the Levinson Algorithm can be considered as a method for the solution of (3.26) when this estimate is used. The Levinson Algorithm is recursive in nature. However, the exact solution of (3.26) is obtained in a fixed number of steps. In the discussion that follows, it will be assumed that $\hat{\Phi}_{XX}$ and $\underline{\hat{R}}_{XX}$ are fixed during the application of the Levinson Agorithm and that

the following definitions apply:

$$\underline{h}_i = [h_1, h_2, \ldots, h_i]^T \tag{4.2}$$

$$\underline{a}_i = [\hat{R}_{XX}(1), \hat{R}_{XX}(2), \ldots, \hat{R}_{XX}(i)]^T \cdot \frac{1}{\hat{R}_{XX}(0)} \tag{4.3}$$

$$\tilde{\underline{a}}_i = [\hat{R}_{XX}(i), \hat{R}_{XX}(i-1), \ldots, \hat{R}_{XX}(1)]^T \cdot \frac{1}{\hat{R}_{XX}(0)} \tag{4.4}$$

$$\underline{g}_i = [g_1, g_2, g_3, \ldots, g_i]^T \tag{4.5}$$

$$\tilde{\underline{g}}_i = [g_i, g_{i-1}, \ldots, g_1]^T \quad . \tag{4.6}$$

As in any recursive algorithm initial values are assigned in the Levinson Algorithm before the recursion begins. These initial values are assigned as follows:

$$\underline{h}_1 = \hat{R}_{XX}(n_1)/\hat{R}_{XX}(0) \tag{4.7}$$

$$\tilde{\underline{g}}_1 = -\hat{R}_{XX}(1)/\hat{R}_{XX}(0) = -a_1 \quad . \tag{4.8}$$

With the above definitions and initial values, the recursion relationships for Levinson's Algorithm are as follows:

$$\theta_i = \hat{R}_{XX}(n_1+1)/\hat{R}_{XX}(0) - \underline{h}_i^T \tilde{\underline{a}}_i \tag{4.9}$$

$$h_i = -\hat{R}_{XX}(i+1)/\hat{R}_{XX}(0) - \underline{h}_i^T \tilde{\underline{g}}_i \tag{4.10}$$

$$\lambda_i = 1 + \underline{a}_i^T \underline{g}_i \tag{4.11}$$

$$\underline{h}_{i+1} = \begin{bmatrix} \underline{h}_i + \theta_i/\lambda_i \tilde{\underline{g}}_i \\ \theta_i/\lambda_i \end{bmatrix} \tag{4.12}$$

$$\underline{g}_{i+1} = \begin{bmatrix} \eta_i/\lambda_i \\ \tilde{\underline{g}}_i + \eta_i/\lambda_i \underline{g}_i \end{bmatrix} \tag{4.13}$$

When the recursive relationships in (4.9) through (4.12) are carried out for i = 1,2,3,...,N-1 the final result $\underline{h}_N$ is the solution to the system (3.26).

4.2.1 Proof of Levinson's Algorithm

The fact that $\underline{h}_N$ produced by Levinson's Algorithm is the solution to the N^{th} order system (3.26) is not obvious from an inspection of the recursive relationships in (4.9) through (4.12). Therefore, the following inductive proof is offered as a verification of the algorithm. It will be assumed without loss of generality that the system of (3.26) has been normalized by dividing both sides by $R_{XX}(0)$ so that the resulting normalized system is:

$$\Phi_N \underline{h}_N = \underline{R}_N \tag{4.14}$$

where the subscript N indicates that the system is of N^{th} order and the following definitions apply:

$$\Phi_N = \frac{1}{\hat{R}_{XX}(0)} \hat{\Phi}_{XX} \tag{4.15}$$

$$\underline{R}_N = \frac{1}{\hat{R}_{XX}(0)} \hat{\underline{R}}_{XX} \quad . \tag{4.16}$$

Now, consider the $(N+1)^{th}$ order normalized system given by:

$$\Phi_{N+1} \underline{h}_{N+1} = \underline{R}_{N+1} \quad . \tag{4.17}$$

This system can be partitioned as follows due to the Topelitz structure of Φ_{N+1} and Φ_N:

$$\begin{bmatrix} \Phi_N & \tilde{\underline{a}}_N \\ \tilde{\underline{a}}_N^T & 1 \end{bmatrix} \underline{h}_{N+1} = \begin{bmatrix} \underline{R}_N \\ \hat{R}_{XX}(n_1+N)/\hat{R}_{XX}(0) \end{bmatrix} \quad . \tag{4.18}$$

Substitution for $\underline{h}_{n+1}$ in (4.18) from (4.12) and expansion gives the following two equations:

$$\Phi_N \underline{h}_N + \Theta_N/\lambda_N \Phi_N \tilde{\underline{g}}_N + \Theta_N/\lambda_N \tilde{\underline{a}}_N = \underline{R}_N \tag{4.19}$$

$$\tilde{\underline{a}}_N^T \underline{h}_N + \Theta_N/\lambda_N \tilde{\underline{a}}_N^T \tilde{\underline{g}}_N + \Theta_N/\lambda_N = \hat{R}_{XX}(n_1+N)/\hat{R}_{XX}(0). \tag{4.20}$$

From the expressions for Θ_N and λ_N in equations (4.10) and (4.11) respectively, it can be seen that (4.20) is satisfied. Thus, $\underline{h}_{N+1}$ is the solution to (4.17) provided that (4.19) is satisfied. Suppose that $\underline{h}_N$ is a solution to (4.14). Then (4.19) reduces to:

$$\Phi_N \tilde{\underline{g}}_N + \tilde{\underline{g}}_N = \underline{0} \quad . \tag{4.21}$$

Therefore, $\underline{h}_{N+1}$ is a solution to (4.17) provided that $\underline{h}_N$ is a solution to (4.14) and (4.21) is satisfied. To see that (4.21) is satisfied for all N, consider the following inductive proof. Equation (4.21) can be partitioned as:

$$\begin{bmatrix} 1 & \underline{a}_{N-1}^T \\ \underline{a}_{N-1} & \Phi_{N-1} \end{bmatrix} \tilde{\underline{g}}_N + \begin{bmatrix} a_N \\ \tilde{\underline{a}}_{N-1} \end{bmatrix} = \underline{0} \quad . \tag{4.22}$$

Now, substituting for $\tilde{\underline{g}}_N$ in (4.22) from (4.13) and expanding yields the following two expressions:

$$\eta_{N-1}/\lambda_{N-1} + \underline{a}_{N-1}^T \tilde{\underline{g}}_{N-1} + \eta_{N-1}/\lambda_{N-1} \underline{a}_{N-1}^T \underline{g}_{N-1} + a_N = 0 \tag{4.23}$$

$$\underline{a}_{N-1} \eta_{N-1}/\lambda_{N-1} + \Phi_{N-1} \tilde{\underline{g}}_{N-1} + \eta_{N-1}/\lambda_{N-1} \Phi_{N-1} \underline{g}_{N-1} + \tilde{\underline{a}}_{N-1} = \underline{0}. \tag{4.24}$$

It can be seen from the expressions for η_{N-1} and λ_{N-1} in (4.10) and (4.11) respectively that (4.23) is satisfied. Thus, (4.21) will be satisfied provided that (4.24) holds. Equation (4.24) can be rewritten as:

$$\Phi_{N-1}\,\tilde{\underline{g}}_{N-1} + \tilde{\underline{a}}_{N-1} + \eta_{N-1}/\lambda_{N-1}\,(\Phi_{N-1}\,\underline{g}_{N-1} + \underline{a}_{N-1}) = \underline{0}\ . \tag{4.25}$$

Now, define the vector $\underline{b}_{N-1}$ as:

$$\underline{b}_{N-1} = \Phi_{N-1}\,\underline{g}_{N-1} + \underline{a}_{N-1}\ . \tag{4.26}$$

Due to the fact that Φ_{N-1} is symmetric and Topelitz, $\underline{b}_{N-1}$ can be written as:

$$\tilde{\underline{b}}_{N-1} = \Phi_{N-1}\,\tilde{\underline{g}}_{N-1} + \tilde{\underline{a}}_{N-1}\ .$$

Thus, (4.25) can be expressed as:

$$\tilde{\underline{b}}_{N-1} + \eta_{N-1}/\lambda_{N-1}\,\underline{b}_{N-1} = \underline{0}\ .$$

Thus, if it is assumed that (4.21) holds with N replaced by N-1, that is if:

$$\tilde{\underline{b}}_{N-1} = \Phi_{N-1}\,\tilde{\underline{g}}_{N-1} + \tilde{\underline{a}}_{N-1} = \underline{0}$$

then (4.25) will be satisfied. Thus, since it has been shown that (4.21) holds for all N provided that it is true for N-1, by induction (4.21) will be satisfied for all N provided that it is satisfied for the case N=1. That is if:

$$\Phi_1\,\tilde{\underline{g}}_1 + \tilde{\underline{a}}_1 = \underline{0}\ . \tag{4.27}$$

It can be seen from the initial value assigned in (4.8) to $\underline{\tilde{g}}_1$ that (4.27) holds. Thus, by induction (4.21) holds for all N.

It has been shown that $\underline{h}_{N+1}$ is the solution to (4.17) provided that $\underline{h}_N$ is a solution to (4.14). Thus, by induction $\underline{h}_N$ is a solution to (4.14) for all N provided that it is the solution in the case where N=1. That is if:

$$\Phi_1 \underline{h}_1 = \underline{R}_1 \quad . \tag{4.28}$$

Since the system (4.14) was assumed to be normalized, (4.28) reduces to:

$$\underline{h}_1 = \underline{R}_1 \quad .$$

It can be seen from the initial value assigned in (4.7) and the definition in (4.16) that (4.28) holds. Thus, it has been proved by induction that $\underline{h}_N$ is the solution to (4.14) and therefore to (3.26).

While not obvious from the recursive relationships in (4.9) through (4.13), $\underline{h}_N$ produced by Levinson's Algorithm is indeed the solution to the N^{th} order linear system of equations in (3.26) provided that $\hat{\Phi}_{XX}$ is constrained to be Hermitian and Topelitz as shown in the above proof. The fact that this algorithm can be used to solve the system (3.26) leads to substantial savings in the number of computations over other non-iterative methods as will be seen in the next section.

4.1.2 Number of Operations Required by Levinson's Algorithm

The solution of the N^{th} order system of linear equations (3.26) by a standard method such as Gaussian Elimination requires on the order of N^3 operations. Such computational effort becomes unreasonable for even moderately large values on N. However, in the special case where $\hat{\Phi}_{XX}$ is Hermitian and Topelitz, the use of Levinson's Algorithm can substantially reduce the amount of computational effort required.

Inspection of the recursive relationships (4.9) through (4.13) and the initial values (4.7) and (4.8) reveal that on the order of N^2 operations are required for the solution of the Hermitian Topelitz system (3.26) using Levinson's Algorithm. The specific number of real arithmetic operations required is given in Figure 4.1.

The number of operations required by Levinson's Algorithm is dominated by the term $2.5N^2$ for large N as can be seen in Figure 4.1. The required number of operations can be reduced somewhat by the use of a modification of Levinson's Algorithm

Additions	Multiplications	Divisions
$2.5N^2+0.5N-3$	$2.5N^2-2.5N$	$4N-2$

Figure 4.1 Number of Operations Required for the Solution of a Hermitian Topelitz System by Levinson's Algorithm.

due to Zohar [25]. This modification replaces equation (4.11) for the computation of the parameter λ_i with the recursive relationship:

$$\lambda_{i+1} = \lambda_i - \eta_i^2/\lambda_i \qquad (4.29)$$

where λ is initialized at the start of the algorithm to be:

$$\lambda_1 = 1 - (\hat{R}_{XX}(1)/\hat{R}_{XX}(0))^2 = 1 - a_1^2 \quad . \qquad (4.30)$$

With this modification, the solution of the system (3.26) requires approximately $2N^2$ operations for large N. The exact number of operations required by the Modified Levinson's Algorithm is given in Figure 4.2.

Additions	Multiplications	Divisions
$2N^2+N-2$	$2N^2+2N-4$	$4N-2$

Figure 4.2 Number of Operations Required for the Solution of a Hermitian Topelitz System by Levinson's Algorithm with Modification due to Zohar.

From Figures 4.1 and 4.2 it can be seen that the use of Levinson's Algorithm or the Modified Levinson's Algorithm allows the solution of a Hermitian Topelitz system of linear equations to be obtained with substantially less computational effort than that required by standard non-iterative methods. In fact,

the modified Levinson's Algorithm can be used to obtain the exact solution using approximately twice the number of operations required to carry out a single step of a fixed-point iterative technique such as Jacobi Relaxation or Successive Over Relaxation. However, the restriction that $\hat{\Phi}_{XX}$ be Topelitz makes Levinson's Algorithm less general than fixed-point iteration techniques.

4.2 Adaptive Estimation Using Levinson's Algorithm

The computational efficiency of the Levinson Algorithm makes it attractive for adaptive estimation when the estimated autocorrelation matrix $\hat{\Phi}_{XX}$ has a Topelitz structure as does the estimate proposed in Section 3.1. However, the $O(N^2)$ operations required by Levinson's Algorithm may be restrictive in many applications. In this case, a procedure analogous to the Variable Iteration Step Method for fixed-point iteration can be used. Specifically, the solution to the system (3.26) can be obtained by the use of Levinson's Algorithm every k samples while the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ are updated at each sample time n. Under this scheme, the average number of operations per sample required by Levinson's Algorithm can be reduced by a factor of k. The exact average number of operations per sample required by Levinson's Algorithm is given in Figure 4.3.

Figure 4.3 gives the average number of operations per sample required by both forms of Levinson's algorithm. However, to determine the average number of operations/sample

required by the adaptive estimation procedure, the number of operations needed to compute the estimated autocorrelation matrix and vector must be added to the average number of operations/sample required by the Levinson algorithm. If the estimate

Algorithm	Additions	Multiplications	Divisions
Levinson	$\frac{1}{k}(2.5N^2+0.5N-3)$	$\frac{1}{k}(2.5N^2-2.5N)$	$\frac{1}{k}(4N-2)$
Modified	$\frac{1}{k}(2N^2+N-2)$	$\frac{1}{k}(2N^2+2N-4)$	$\frac{1}{k}(4N-2)$

Figure 4.3 Average Number of Operations/Sample Required Levinson's Algorithm and the Modified Levinson Algorithm when carried out over k sample periods.

proposed in Section 3.1 is used in conjunction with Levinson's Algorithm to accomplish the adaptive estimation task, then the required average number of operations/sample can be obtained by adding the number of operations required to estimate Φ_{XX} and $\underline{R}_{XX}$ using (3.25) to the entries of Figure 4.3. The number of operations required to estimate Φ_{XX} and $\underline{R}_{XX}$ using (3.25) can be found in Figure 3.1. The total average number of operations/sample required for the solution of (3.26) using Levinson's Algorithm applied every k sample periods and updating $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ every sample period is given in Figure 4.4.

Comparing Figures 4.3 and 3.4, it can be seen that the use of the Levinson Algorithm produces the exact solution to the system (3.26) with an amount of computational effort which is on the same order as that required to carry out a single step of

Algorithm	Additions	Multiplications	Divisions
Levinson ($n_1 \leq N$)	$\frac{1}{k}(2.5N^2+0.5N-3)+2(N+n_1)$	$\frac{1}{k}(2.5N^2-2.5N)+3(N+n_1)$	$\frac{1}{k}(4N-2)$
Levinson ($n_1 > N$)	$\frac{1}{k}(2.5N^2+0.5N-3)+4N$	$\frac{1}{k}(2.5N^2-2.5N)+6N$	$\frac{1}{k}(4N-2)$
Modified ($n_1 \leq N$)	$\frac{1}{k}(2N^2+N-2)+2(N+n_1)$	$\frac{1}{k}(2N^2+2N-4)+3(N+n_1)$	$\frac{1}{k}(4N-2)$
Modified ($n_1 > N$)	$\frac{1}{k}(2N^2+N-2)+4N$	$\frac{1}{k}(2N^2+2N+4)+6N$	$\frac{1}{k}(4N-2)$

Figure 4.4 Total Average Number of Operations/Sample Required to Estimate Φ_{XX} and $\underline{R}_{XX}$ Using (3.25) and to Solve the System (3.26) Using Levinson's Algorithm Over k Sample Periods.

the Successive Over-Relaxation Method. Thus, Levinson's Algorithm is attractive for adaptive estimation when the estimated autocorrelation matrix has a Topelitz structure.

4.3 Conclusion

Levinson's Algorithm is an attractive approach to the solution of the system of linear equations (3.26) in that it determines the exact solution very efficiently in a non-iterative fashion. Using a procedure similar to the Variable-Iteration-Step Method, the average number of operations/sample can be reduced by solving the system using Levinson's Algorithm every k sample periods while updating the estimated autocorrelation matrix and vector at each sample time. This method gives a flexible approach to the adaptive estimation problem in that full utilization of the time available between samples can be made for computation.

Unfortunately, the Levinson Algorithm requires that the autocorrelation matrix estimate have a Topelitz structure. While the estimate $\hat{\Phi}_{XX}$ proposed in Section 3.1 has such a structure, other estimates of interest may not. In this respect, the Adaptive Fixed-Point Iteration Method is superior in that it is applicable to a more general class of matrices. In addition, the Levinson Algorithm is recursive in nature and not well suited to parallel processing as are the two fixed-point iteration methods discussed in the previous chapter. Suitability of an algorithm for parallel processing is of extreme importance

when real-time applications which involve large values of N are considered.

Levinson's Algorithm offers a viable approach to adaptive estimation for the somewhat restrictive class of problems to which it is applicable. It is very attractive in situations in which the sample period is large enough so that a large number of computations can be carried out between samples without the use of parallel processing. The performance of the adaptive estimation method using Levinson's Algorithm described in this chapter will be compared to that of the Adaptive Fixed-Point Iteration methods proposed in Chapter III and to that of the LMS Gradient Method via computer simulation in Chapter VI.

CHAPTER V

THEORETICAL PERFORMANCE OF THE LMS GRADIENT ALGORITHM AND ADAPTIVE FIXED-POINT ITERATION

Both the LMS Gradient Algorithm and the Adaptive Fixed-Point Iteration procedure proposed in Chapter 3 are iterative methods for the approximate solution of the discrete finite impulse response Wiener-Hopf equation given by:

$$\Phi_{XX}\,\underline{h} = \underline{R}_{XX} \quad . \tag{1.25}$$

Since these methods are iterative in nature, their rates of convergence are of importance. Theoretical bounds on the rates of convergence in the mean of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the Successive Over-Relaxation Method were derived in Chapters II and III. While only approximate, these bounds are useful for comparing convergence rates of the LMS Gradient Method over variations in the parameter μ and of the Jacobi Relaxation and SOR Methods over variations in the relaxation parameter β. In addition, these bounds are useful for evaluating the Jacobi Relaxation and SOR Algorithms when they are used with the Variable Iteration Method and for making general comparisons of the rates of convergence of the three iterative algorithms.

In this chapter, the rates of convergence of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Method are studied for a particular problem under varying conditions. The term "rate of convergence", as used in this chapter, will refer to the rate of convergence of the mean-square error produced by the filter determined by a particular algorithm to the minimum MSE which would be obtained by the use of an optimal filter. The derivation of the expressions used in computing the bounds considered in this chapter are given in Sections 2.3, 3.3.2 and 3.4.1. Comparisons between the theoretical bounds and "actual" performance will be investigated in Chapter VI by the use of computer simulation.

5.1 Statement of the Problem

The theoretical bounds on the rates of convergence of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Method derived in Chapters II and III require knowledge of the input autocorrelation matrix if they are to be used in the study of the convergence of these iterative algorithms. Therefore, some particular problem must be selected for which the autocorrelation matrix can be determined a priori. Sinusoidal signals comprize an important class of inputs for a Spectral Line Enhancer due to their widespread use. In addition, the autocorrelation function of a sinusoidal signal does not decay to zero as the amount of delay increases. Therefore, the sinusoid is the limiting case of a coherent signal. In the problem

considered in this chapter, the narrowband input to the Spectral Line Enhancer is assumed to be a single sinusoid of known amplitude and frequency but of unknown phase. This problem is often referred to as the "signal known except phase" (SKEP) problem. It will be assumed here that the phase is uniformly distributed on the interval $[-\pi,\pi]$. Therefore, in the problem considered here, the signal, or narrowband input to the Spectral Line Enhancer will be assumed to have the form:

$$S(n) = A \cos(\Omega n+\Theta) \qquad (5.1)$$

where: A = Signal Amplitude

Ω = Signal Frequency (Radians)

n = Time Index

Θ = Random Phase, uniformly distributed on $[-\pi,\pi]$

The Spectral Line Enhancer assumes that one component of the data is relatively narrowband compared to the other component. Therefore, the noise process chosen for this problem must be relatively wideband compared to the signal given in (5.1). In other words, the autocorrelation function of the noise must be essentially zero for delays of n_1 or greater. To satisfy this assumption for the problem considered, an autoregressive noise sequence was chosen as the process to be added to the signal. This type of noise sequence may be generated by the following relationship:

$$N(n) = \sum_{i=0}^{n_1-1} \alpha^i \, y(n-i) \qquad (5.2)$$

where y(i) are independent identically distributed random variables with zero mean and variance = σ^2, and $|\alpha| < 1$ is the autoregression parameter. As will be shown later, N(n) generated by using (5.2) has an autocorrelation function which is zero for delays of n_1 or greater which satisfies the assumption made in the formulation of the Spectral Line Enhancer.

Thus, the SKEP problem considered in this chapter is that of a single sinusoidal signal given by (5.1) in the presence of an additive autoregressive noise sequence generated according to (5.2). If it is assumed that the signal and the noise are orthogonal, then the determination of the input autocorrelation matrix is straight forward as will be shown in the next section.

5.1.1 Determination of the Autocorrelation Matrix

In order to investigate the convergence properties of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Method using the theoretical bounds derived in Chapters II and III, it is necessary to determine the input autocorrelation matrix for the problem considered. In the case of the SKEP problem considered, the determination of Φ_{XX} is fairly straightforward. However, the derivation will be included here for completeness.

Each element of the input autocorrelation matrix Φ_{XX} can be written as follows:

$$\phi_{ij} = E[X(i)X(j)] \quad . \tag{5.3}$$

However, since the signal sequence and the noise sequence are assumed to be orthogonal, (5.3) can be rewritten as:

$$\phi_{ij} = E[S(i)S(j)] + E[N(i)N(j)] \quad . \tag{5.4}$$

Therefore, determination of the matrix Φ_{XX} can be reduced to the computation of two autocorrelation functions, $R_{SS}(i,j)$ and $R_{NN}(i,j)$, where the following definitions apply:

$$R_{SS}(i,j) = E[S(i)S(j)] \tag{5.5}$$

$$R_{NN}(i,j) = E[N(i)N(j)] \quad . \tag{5.6}$$

The autocorrelation function of the signal can be obtained by substituting for S(i) and S(j) in (5.5) from equation (5.1). Equation (5.5) can then be written as:

$$R_{SS}(i,j) = A^2 \, E[\cos(\Omega i+\Theta)\cos(\Omega j+\Theta)] \quad . \tag{5.7}$$

Equation (5.7) can be reduced using a well-known trigonometric identity to:

$$R_{SS}(i,j) = \frac{A^2}{2} E[\cos(\Omega(i-j))+\cos(\Omega\,(i+j)+2\,\theta)] \quad . \tag{5.8}$$

However, the term $\cos(\Omega(i-j))$ is not a random quantity. There-

fore, (5.8) reduces to:

$$R_{SS}(i,j) = \frac{A^2}{2} \cos(\Omega(i-j)) + \frac{A^2}{2} E[\cos(\Omega(i+j)+2\theta)]. \quad (5.9)$$

Now, it is assumed that the phase θ was distributed uniformly on the interval $[-\pi,\pi]$. Therefore, the expected value term in (5.9) is defined as:

$$E[\cos(\Omega(i+j)+2\theta)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \cos(\Omega(i+j)+2\theta)d\theta \ . \quad (5.10)$$

But, the integral in the above expression is equal to zero. Thus:

$$E[\cos(\Omega(i+j)+2\theta)] = 0 \ .$$

Therefore, the autocorrelation function of the signal can be expressed as:

$$R_{SS}(i,j) = \frac{A^2}{2} \cos(\Omega(i-j)) \ . \quad (5.11)$$

It is obvious from (5.11) that the autocorrelation function of the signal for the SKEP problem is a stationary statistic since it is a function only of (i-j). Therefore, if $R_{NN}(i,j)$ is also a stationary statistic, then the input autocorrelation matrix will exhibit a Topelitz structure.

The autocorrelation function for the autoregressive noise sequence can be determined by considering equation (5.6) and substituting for N(i) and N(j) from (5.2). Equation (5.6) can

then be written as:

$$E[N(i)N(j)] = E[(\sum_{\ell=0}^{n_1-1} \alpha^{\ell}\, Y(i-\ell))(\sum_{k=0}^{n_1-1} \alpha^{k}\, Y(j-k))]. \quad (5.12)$$

Expanding (5.12) yields:

$$E[N(i)N(j)] = \sum_{\ell=0}^{n_1-1} \sum_{k=0}^{n_1-1} \alpha^{\ell+k}\, E[Y(i-\ell)Y(j-k)]. \quad (5.13)$$

Now, since the Y(i) were assumed to be independent, identically distributed random variables with zero mean, the expected value term $E[Y(i-\ell)Y(j-k)]$ is equal to zero except for the case when:

$$(i-\ell) = (j-k)$$

or, whenever:

$$(i-j) = (\ell-k) \quad . \quad (5.14)$$

Now, since $(\ell-k)$ can only take on integer values between the limits:

$$-(n_1-1) \leq (\ell-k) \leq (n_1-1)$$

$E[N(i)N(j)]$ must be equal to zero for all values of $(i-j)$ such that:

$$|i-j| \geq n_1 \quad . \quad (5.15)$$

Therefore, as required by the assumption made in the formulation of the Spectral Line Enhancer, noise samples which are separated by delays of n_1 or greater are orthogonal. In the case where $(i-j)=(\ell-k)$, $R_{NN}(i,j)$ can be determined by making the substitution $m=|i-j|$. With this substitution, (5.13) can be rewritten as:

$$E[N(i)N(j)] = \sum_{\ell=m}^{n_1-1} \alpha^{2\ell-m} E[Y(i-\ell)Y(i-\ell)] \quad . \qquad (5.16)$$

Now, since Y(i) was assumed to have zero mean and variance σ^2, (5.16) can be rewritten as:

$$E[N(i)N(j)] = \sigma^2 \alpha^{|i-j|} \frac{1-\alpha^{2(n_1-|i-j|)}}{1-\alpha^2} \quad . \qquad (5.17)$$

Thus, combining the results of (5.15) and (5.17) the autocorrelation function of the noise can be expressed as:

$$E[N(i)N(j)] = \sigma^2 \alpha^{|i-j|} \frac{1-\alpha^{2(n_1-|i-j|)}}{1-\alpha^2} \qquad |i-j| < n_1$$

$$= 0 \qquad \text{otherwise} \quad . \qquad (5.18)$$

Equations (5.11) and (5.18) can be substituted into (5.4) to determine the autocorrelation matrix of the input data. It is obvious from (5.11) and (5.18) that Φ_{XX} will have a Topelitz structure since both $R_{SS}(i,j)$ and $R_{NN}(i,j)$ are functions only of $(i-j)$. Once Φ_{XX} has been determined, the theoretical bounds derived in Chapters II and III can be used to study the convergence properties of the three iterative algorithms considered.

5.1.2 Initial Conditions and Parameter Values

If the input autocorrelation matrix Φ_{XX} for the SKEP problem is to be computed using (5.11) and (5.18), values for the various parameters must be assigned. In addition, the theoretical bounds on the rates of convergence of the iterative algorithms considered here depend on the initial value chosen for the unit sample response vector $\underline{h}$. Figure 5.1 gives the parameter values and initial vector $\underline{h}$ used in this chapter for the study of convergence properties of the three iterative algorithms. These parameters were chosen arbitrarily.

Parameter	Value
N	16
n_1	20
A	1
Ω	0.10π radians
α	0.25
$h_0(n+n_1)$	1 $0 \leq n \leq 7$ 0 otherwise

Figure 5.1 Parameter Values in Study of Theoretical Rates of Convergence of the Three Iterative Algorithms Considered

5.2 Theoretical Performance of the LMS Gradient Algorithm

A theoretical bound on the rate of convergence of the mean-square error produced by the filter determined by the LMS Gradient Algorithm was derived in Section 2.3 and was given by the relationship in (2.49). As discussed in Section 2.3, the rate of convergence of the MSE to the minimum MSE produced by the optimal filter depends on the parameter μ present in the LMS Gradient Algorithm. In this section, the rate of convergence of the LMS Gradient Algorithm for the SKEP problem is explored for various values of the parameter μ under several different signal-to-noise ratio (SNR) conditions. The SNR is defined here as the ratio of the signal variance to the noise variance on a per sample basis. That is, SNR is defined as:

$$\text{SNR} = \frac{\sigma_S^2}{\sigma_N^2} = \frac{A^2}{2\sigma_N^2} \quad . \tag{5.19}$$

The rates of convergence for the different values of μ are studied using the theoretical bound given by (2.49) and therefore "convergence" is interpreted as convergence in the mean.

Figure 5.2 shows computer-generated plots of the theoretical convergence bound given in (2.49) for the LMS Gradient Algorithm for various values of the parameter μ. Each plot is for a different condition of signal-to-noise ratio with the four curves in each plot corresponding to four different values of μ. The vertical axis in each figure is the MSE axis, while the horizontal axis corresponds to the number of sample periods. The values of μ which correspond to each of the four curves in each

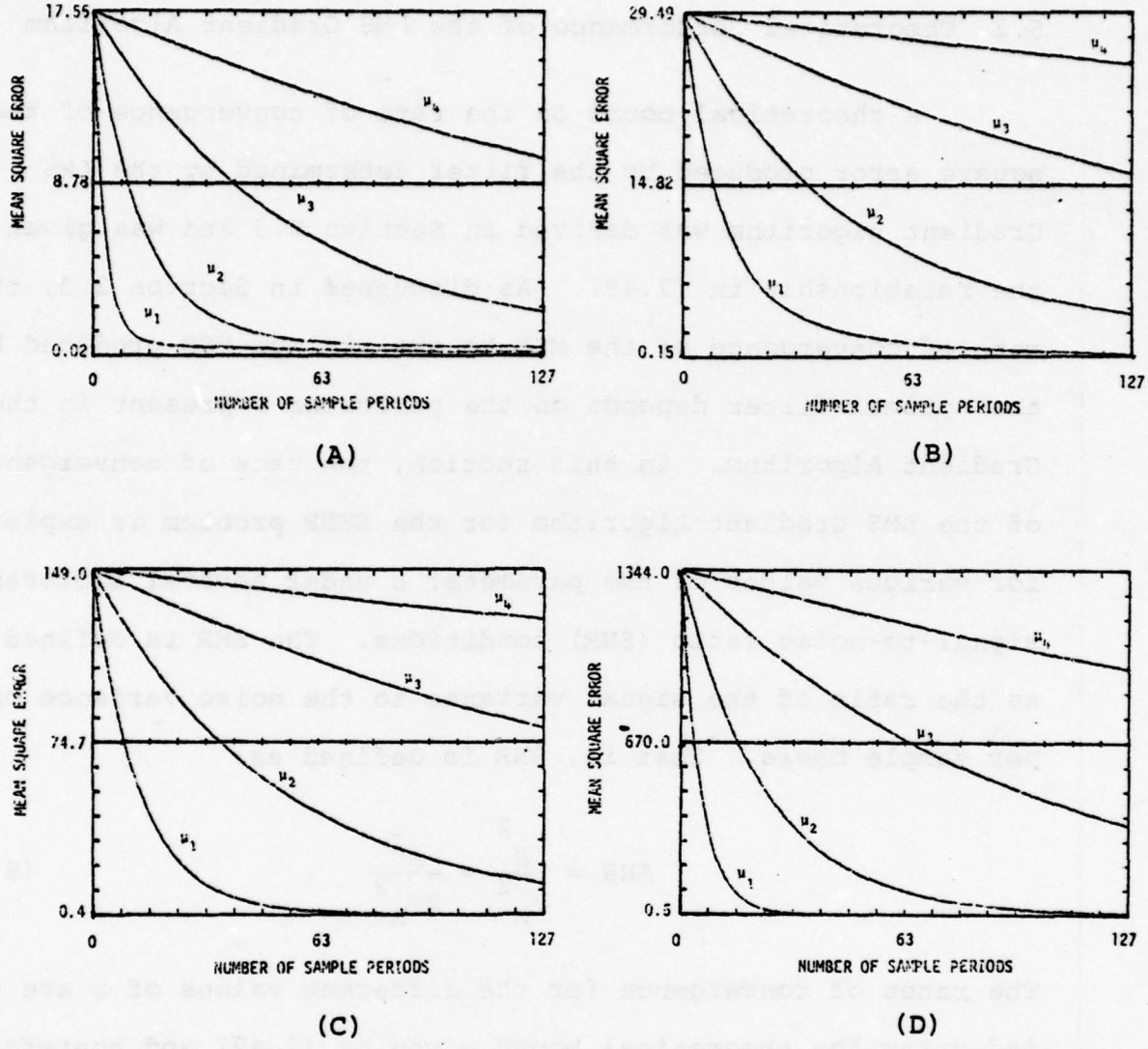


Figure 5.2 Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ Under Four Different SNR Conditions.

PLOT	SNR, db	μ_1	μ_2	μ_3	μ_4
A	+ 7.0	0.016	0.004	0.001	$2.5.10^{-4}$
B	- 3.0	0.004	0.001	$2.5.10^{-4}$	$6.25.10^{-5}$
C	-13.0	0.001	$2.5.10^{-4}$	$6.25.10^{-6}$	$1.56.10^{-5}$
D	-23.0	$2.5.10^{-4}$	$6.25.10^{-5}$	$1.56.10^{-5}$	$3.9.10^{-6}$

of the figures are values which might reasonably be used in practice. They have been selected so that there is a factor of four difference in the value of μ between any two adjacent curves in each of the four plots. As can be seen from the figures, the value of the parameter μ has a significant effect on the rate of convergence of the LMS Gradient Algorithm. Specifically, a decrease in the value of μ greatly increases the number of sample periods required for the LMS Gradient Algorithm to converge. The theoretical bounds considered here will be compared to the "actual" convergence properties of the LMS Gradient Algorithm via computer simulation in Chapter VI.

5.3 Theoretical Performance of the Jacobi Relaxation Method

Equation (3.75), derived in Section 3.3.2, gives a theoretical bound on the rate of convergence of the mean-square error produced by the filter determined by the Jacobi Relaxation Method to the minimum MSE which would be obtained by the use of the optimal filter. As discussed in Section 3.3.2, the relaxation parameter β directly effects the rate of convergence of the Jacobi Relaxation Method. In this section, the rate of convergence of the MSE produced by the filter determined by the Jacobi Relaxation Method to the MSE produced by the optimal filter for the SKEP problem is studied for various values of the relaxation parameter β under several different conditions of input signal-to-noise ratio. Signal-to-noise ratio is defined as in (5.19). In addition, the theoretical rate of convergence of

Jacobi Relaxation operating under the Variable Iteration Step Method is also studied for various SNR conditions in this section. In all cases, the theoretical convergence curves were generated using equation (3.75) as derived in Section 3.3.2. As was the case for the LMS Gradient Method, convergence of $\underline{h}_n$ produced by Jacobi Relaxation to the optimal solution is defined as convergence in the mean.

5.3.1 Theoretical Performance of Jacobi Relaxation for Varying β

As stated previously, the rate of convergence of Jacobi Relaxation depends directly on the value of the relaxation parameter β. To investigate the rate of convergence with various values of β for the SKEP problem, theoretical convergence curves were generated by a computer according to equation (3.75). Figure 5.3 shows the resulting curves. Each plot in the figure corresponds to a different signal-to-noise ratio condition with each curve in a plot corresponding to a different value of the relaxation parameter β. The values of β chosen are such that there is a factor of four difference in the value of β between any two adjacent curves. The largest value of β=0.14, corresponding to the left-most curve in each figure was chosen by applying the bound on β given in equation (3.72). This bound was used since its application does not require knowledge of the statistics of the input process which would not be available in practice. The vertical axis for each plot is the MSE axis while the horizontal axis gives the number of sample periods.

Note from Figure 5.3 that the value of the relaxation parameter β has a significant effect on the rate of convergence of the Jacobi Relaxation Method. Specifically, decreasing the value of β slows the rate of convergence or increases the number of sample periods required for the MSE produced by the filter determined by the Jacobi Relaxation Method to reach the minimum MSE produced by the optimal filter. The theoretical bounds considered here will be compared to the "actual" performance of the Jacobi Relaxation Method using computer simulation in Chapter VI.

5.3.2 Theoretical Performance of Jacobi Relaxation Under the Variable-Iteration Step Method

The Jacobi Relaxation Method requires on the order of N^2 computations per iteration step when implemented directly using equation (3.76) as can be seen in Figure 3.2. The Variable Iteration Step Procedure was proposed in Section 3.5.1 as a method for reducing the average number of operations per sample for the Fixed-Point Iteration Algorithms. In this procedure, the Jacobi Relaxation Method was used to update ℓ elements of the unit sample response vector every k samples. Therefore, if $\ell/k < N$, the average number of operations/sample can be reduced to:

$$\ell/k\ (N+1) < N(N+1) \quad . \tag{5.20}$$

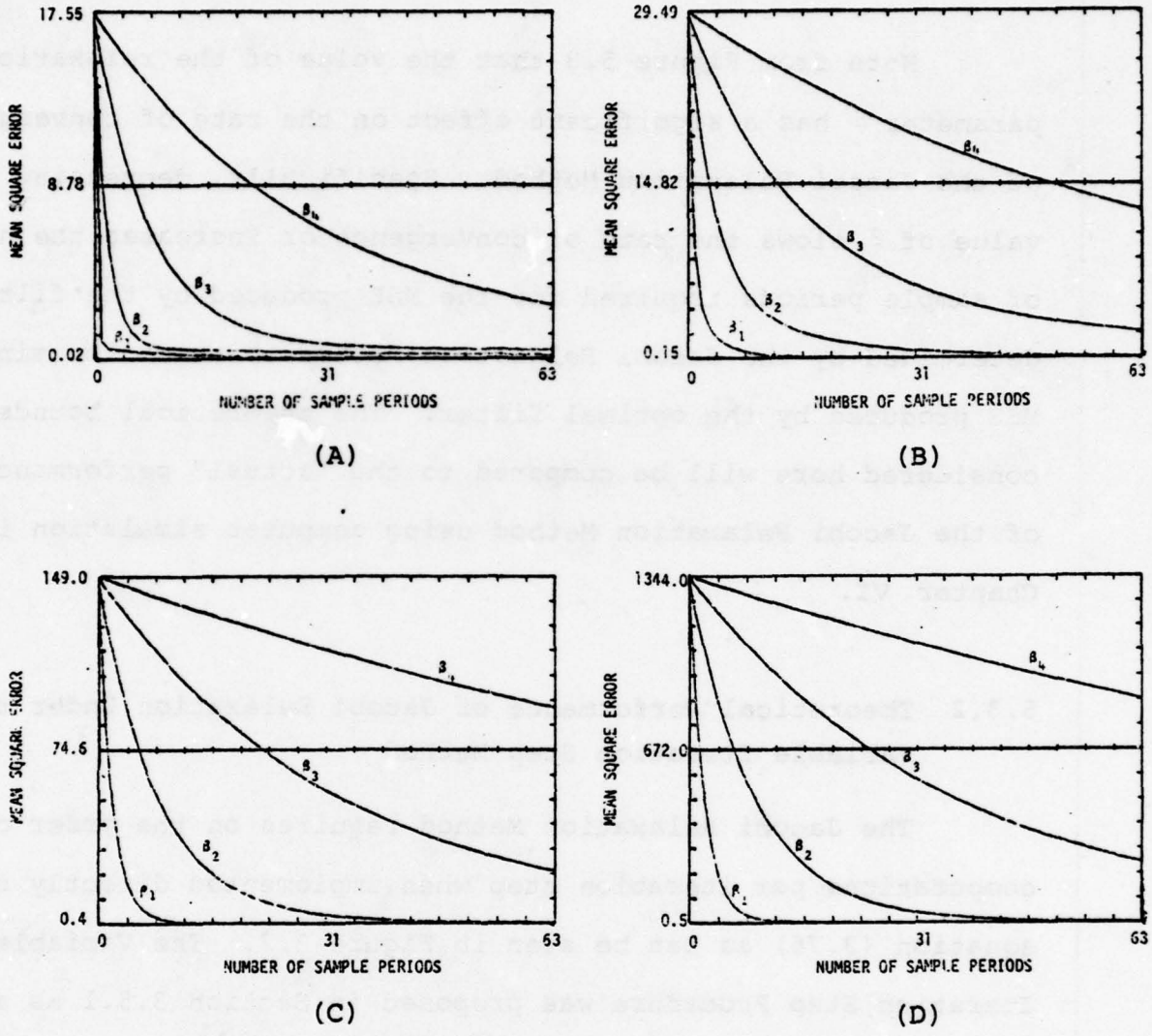


Figure 5.3 Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β Under Four Different SNR Conditions.

PLOT	SNR, db	β_1	β_2	β_3	β_4
A	+ 7.0	0.14	0.035	0.00875	0.00218
B	- 3.0	0.14	0.035	0.00875	0.00218
C	-13.0	0.14	0.035	0.00875	0.00218
D	-23.0	0.14	0.035	0.00875	0.00218

However, as mentioned in Section 3.5.1, such a reduction in the average number of operations per sample is accompanied by a corresponding increase in the number of sample periods required for convergence. Figure 5.4 shows computer-generated plots giving the theoretical bounds on the rate of convergence of the MSE produced by the filter determined by the Jacobi Relaxation Algorithm operating under the Variable Iteration Step Method. Each plot in the figure corresponds to a different SNR condition for the SKEP problem and each curve in a plot corresponds to a different value of ℓ/k. In all the plots, a curve corresponding to the ordinary Jacobi Relaxation Method ($\ell=N$, $k=1$) is included as a reference. In all cases, the relaxation parameter β was chosen to be $\beta=0.14$ which was determined from the bound given in equation (3.72) as discussed in the last section. The value of ℓ in the Variable Iteration Step Method has been fixed at $\ell=N=16$ since the FFT Method proposed for the implementation of Jacobi Relaxation updates all N elements of $\underline{h}$ at each iteration step. In all cases, the vertical axis is the MSE axis while the horizontal axis gives the number of sample periods.

As can be seen in Figure 5.4, decreasing the ratio ℓ/k causes a corresponding increase in the number of sample periods required for convergence. The increase is directly proportional to the parameter ℓ/k. The theoretical bounds considered here will be compared to the "actual" performance of the Jacobi Relaxation Algorithm operating under the Variable Iteration Step Method in Chapter VI using computer simulation.

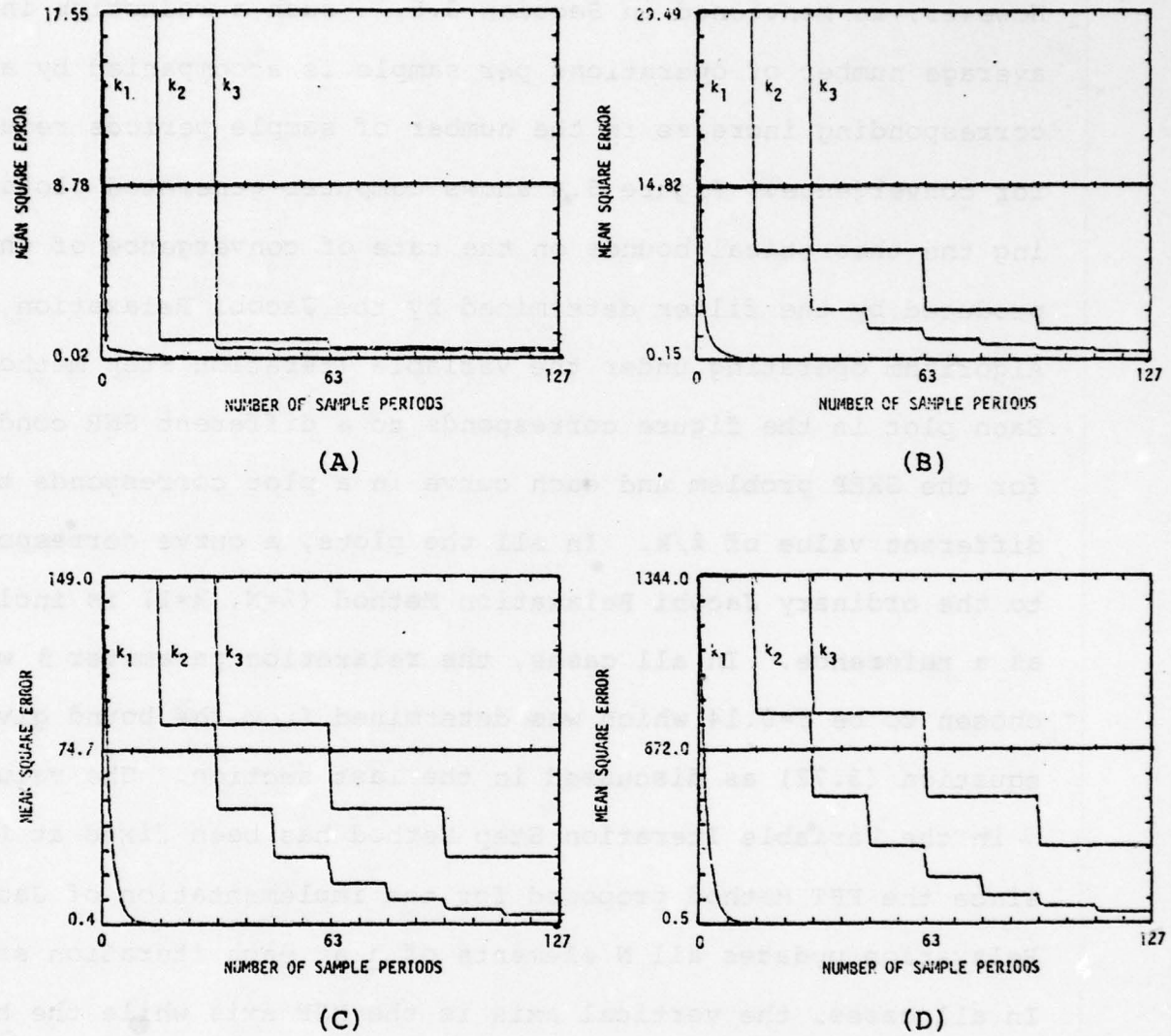


Figure 5.4 Theoretical Performance of the Jacobi Relaxation Algorithm Using the Variable Iteration Step Method for Various Values of the Parameter k Under Four Different SNR Conditions. $\ell=16, \beta=0.14$

PLOT	SNR, db	k_1	k_2	k_3
A	+ 7.0	1	16	32
B	- 3.0	1	16	32
C	-13.0	1	16	32
D	-23.0	1	16	32

5.4 Theoretical Performance of the SOR Method

A theoretical bound on the convergence of the mean-square error produced by the filter determined by the SOR Method was derived in Section 3.4.1 and is given by equations (3.91) and (3.92). As can be seen from these equations, the relaxation parameter β effects the rate of convergence of the SOR Method. In this section, the rate of convergence of the MSE produced by the filter determined by the SOR Method to the minimum MSE which would be obtained by the optimal filter for the SKEP problem is studied for various values of the relaxation parameter β under several different signal-to-noise ratio conditions. In addition, the theoretical rate of convergence of the MSE of the filter determined by the SOR Method operating under the Variable Iteration Step scheme is considered for various SNR conditions. In all cases the theoretical bounds were generated using equations (3.91) and (3.92) and convergence of $\underline{h}_n$ produced by the SOR Method to the optimal solution is interpreted as convergence in the mean.

5.4.1 Theoretical Performance of Successive Over-Relaxation for Varying β

The rate of convergence of the SOR Method depends directly on the value of the relaxation parameter β as discussed previously. Theoretical convergence curves were generated for the SKEP problem by the use of equations (3.91) and (3.92) to investigate the rates of convergence of the MSE produced by the filters determined by the SOR Method with various values of β. The result-

ing curves are shown in Figure 5.5. Each plot in the figure corresponds to a different signal-to-noise ratio condition with each curve in a plot corresponding to a different value of the relaxation parameter β. Each of the three values of β chosen lie within the range $0 < \beta < 2$ which is required for convergence of the SOR Method. Figure 5.17 gives the SNR condition and values of β which correspond to each figure. The vertical axis for each plot is the MSE axis while the number of sample periods is along the horizontal axis.

Note from Figure 5.5 that the value of the relaxation parameter β has a significant effect on the rate of convergence of the SOR Method. However, as can be seen from the figures, the relationship between the rate of convergence and the value of β is not straightforward as was the case with the Jacobi Relaxation Method. In fact, except in certain special cases it is extremely difficult to determine the value of β which is most advantageous for convergence [20]. The "actual" performance of the SOR Method as determined by computer simulation will be compared to the theoretical bounds considered here in Chapter VI.

5.4.2 Theoretical Performance of Successive Over-Relaxation Under the Variable Iteration Step Method

The implementation of the SOR Method using equation (3.93) requires on the order of N^2 computations per iteration step as can be seen in Figure 3.2. The Variable Iteration Step Procedure proposed in Section 3.5.1 is a method for the reduction of the average number of operations per sample required by the

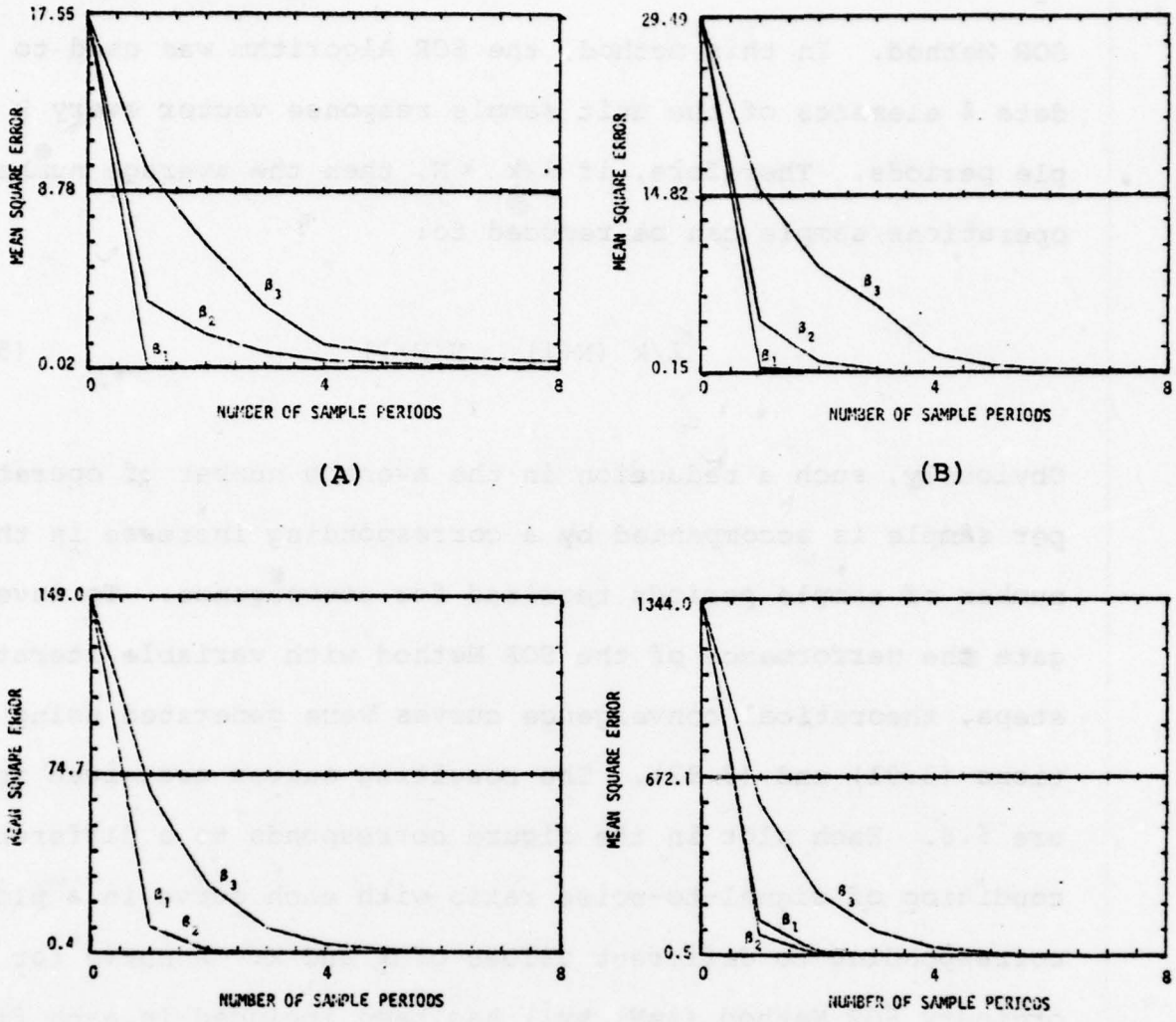


Figure 5.5 Theoretical Performance of the SOR Algorithm For Various Values of the Parameter β Under Four Different SNR Conditions.

PLOT	SNR, db	β_1	β_2	β_3
A	+ 7.0	0.5	1.0	1.5
B	- 3.0	0.5	1.0	1.5
C	-13.0	0.5	1.0	1.5
D	-23.0	0.5	1.0	1.5

SOR Method. In this method, the SOR Algorithm was used to update ℓ elements of the unit sample response vector every k sample periods. Therefore, if $\ell/k < N$, then the average number of operations/sample can be reduced to:

$$\ell/k\ (N+1) < N(N+1) \quad . \tag{5.21}$$

Obviously, such a reduction in the average number of operations per sample is accompanied by a corresponding increase in the number of sample periods required for convergence. To investigate the performance of the SOR Method with variable iteration steps, theoretical convergence curves were generated using equations (3.91) and (3.92). The resulting curves are given in Figure 5.6. Each plot in the figure corresponds to a different condition of signal-to-noise ratio with each curve in a plot corresponding to different values of ℓ and k. A curve for the ordinary SOR Method (ℓ=N, k=1) has been included in each figure as a reference. In all cases, the vertical axis is the MSE axis and the horizontal axis gives the number of sample periods.

As can be seen from Figure 5.6, the values of ℓ and k chosen for the variable iteration step method have a significant effect on the rate of convergence of the SOR Algorithm. Each plot in the figure contains three curves. The curve which exhibits the highest rate of convergence is the reference curve corresponding to ℓ=16 and k=1 as expected. The remaining two curves correspond to (ℓ,k) values which were chosen so that the ratio ℓ/k would equal one. In this case, the average number of

operations per sample for the SOR Method is on the order of N, the length of the filter. As can be seen from the figure, the curve for the case (ℓ=1, k=1) exhibits a higher rate of convergence than the case where (ℓ=16, k=16) since its convergence curve always lies below that for the case (ℓ=16, k=16). The theoretical bounds considered here will be compared to the "actual" performance of the SOR Algorithm operating under the Variable Iteration Step Method in Chapter VI by the use of computer simulation.

5.5 Comparison of the Three Algorithms

In the previous sections, the convergence properties of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Algorithm were studied under varying conditions using the theoretical bounds derived in Chapters II and III. While these bounds are only approximate and neglect covariance terms, comparisons among the three algorithms can still be made. In this section, theoretical convergence curves for the three iterative algorithms will be compared. The "actual" convergence of the algorithms will be studied via computer simulation in Chapter VI.

The LMS Gradient Algorithm requires fewer operations per sample than the two fixed-point iteration algorithms. Also, the estimate of the input statistics used by the LMS Gradient Algorithm has a larger variance than the estimate used by the Adaptive Fixed-Point Iteration Algorithms. Therefore, any meaningful comparison of the three algorithms should attempt to

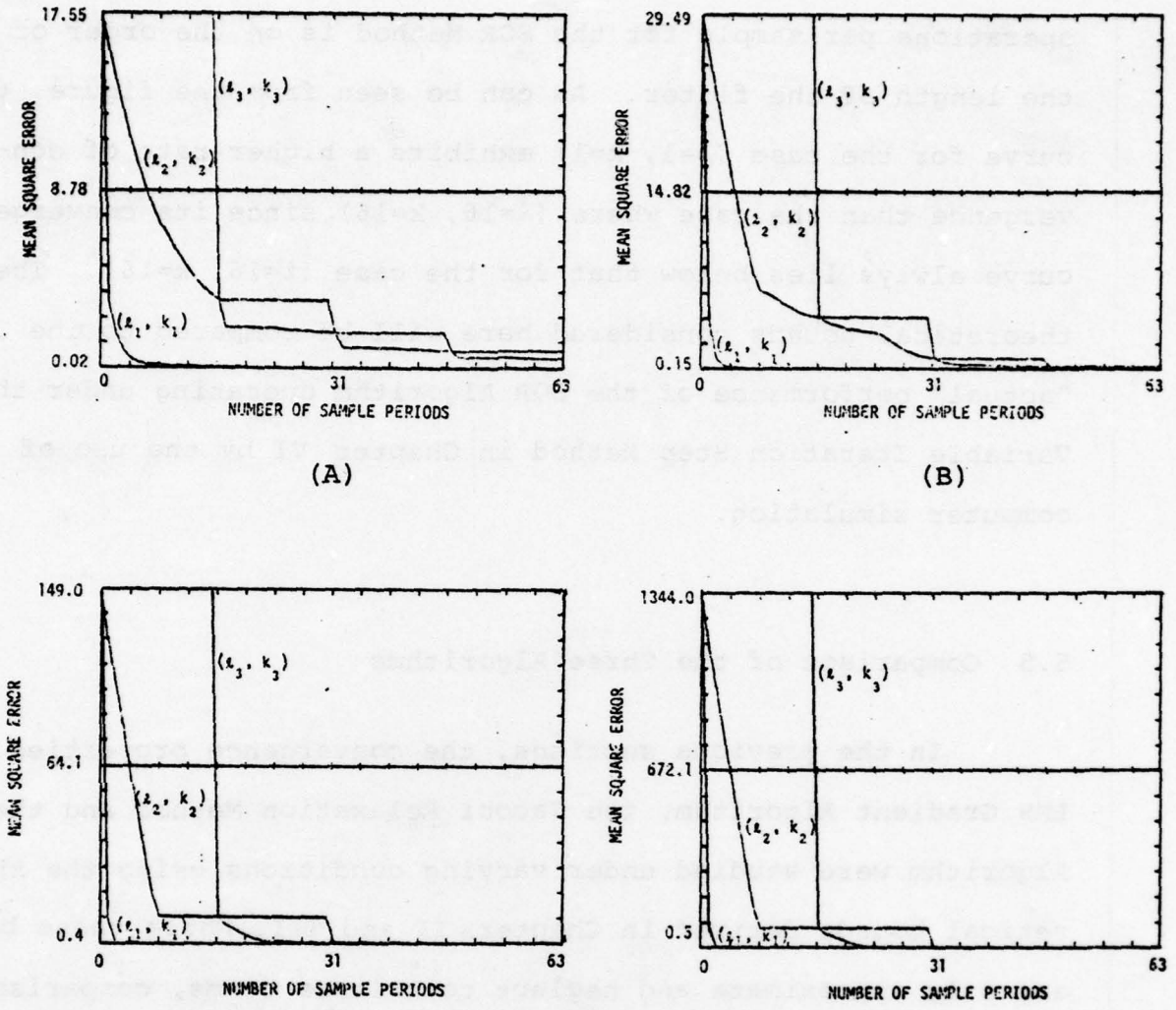


Figure 5.6 Theoretical Performance of the SOR Algorithm Using the Variable Iteration Step Method for Various Values of the Parameters ℓ and k Under Four Different SNR Conditions. β=1.0.

PLOT	SNR, db	ℓ_1	k_1	ℓ_2	k_2	ℓ_3	k_3
A	+ 7.0	16	1	1	1	16	16
B	- 3.0	16	1	1	1	16	16
C	-13.0	16	1	1	1	16	16
D	-23.0	16	1	1	1	16	16

equalize the number of operations/sample and the variance of the estimate of the statistics across the three algorithms. The average number of operations/sample can be equalized approximately by the use of the Variable Iteration Step Method with the Jacobi Relaxation Method and the SOR Algorithm. Under the Variable Iteration Step Method, the average number of operations/sample required by the fixed-point iteration algorithms can be reduced to O(N). Thus, in the comparisons that follow, the ratios ℓ/k are chosen to be one for both the Jacobi Relaxation and the SOR Methods. For the Jacobi Relaxation Method, ℓ and k are both chosen to be N, while for the SOR Method, ℓ and k are both chosen to be 1. Therefore, the average numbers of operations/sample are on the order of N for all three iterative algorithms considered. However, the Adaptive Fixed-Point Iteration methods still require more operations/sample than the LMS Gradient Algorithm due to the O(N) operations/sample required to estimate the autocorrelation matrix and autocorrelation vector.

The estimate of the statistics of the input process used by the LMS Gradient Method is given by (2.17) as:

$$\hat{\nabla}E[e^2(n)|\underline{h}] = 2(\hat{S}(n)-X(n))\underline{X}(n-n_1) \quad . \qquad (2.17)$$

Substituting for $\hat{S}(n)$, this estimate of the gradient of the MSE can be rewritten as:

$$\hat{\nabla}E[e^2(n)|\underline{h}] = 2[\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n - X(n)\underline{X}(n-n_1)] \qquad (5.22)$$

which may also be expressed as:

$$\hat{\nabla}_E[e^2(n)\,|\underline{h}] = 2(\hat{\Phi}_{XX}\,\underline{h}_n - \hat{\underline{R}}_{XX}) \qquad (5.23)$$

where:

$$\hat{\Phi}_{XX} = \underline{X}(n-n_1)\underline{X}^T(n-n_1) \qquad (5.24)$$

$$\hat{\underline{R}}_{XX} = X(n)\underline{X}(n-n_1) \quad . \qquad (5.25)$$

The estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ given in (5.24) and (5.25) are similar tothe estimates of the autocorrelation matrix and vector used in the two Adaptive Fixed-Point Iteration Algorithms. Therefore, the variance of each element of $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ is given by (3.22) with M=1. A comparison of the LMS Gradient Algorithm and the Jacobi Relaxation Algorithm reveals that the same basic iteration relationship is used in both cases. This iteration relationship is given by:

$$\underline{h}_{n+1} = \underline{h}_n - C(\hat{\Phi}_{XX}\,\underline{h}_n - \hat{\underline{R}}_{XX}) \quad . \qquad (5.26)$$

The difference between the two algorithms lies in the choice of the parameter C and the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$. In the comparisons which follow, the value of β for the Jacobi Relaxation Algorithm was chosen according to the bound given by equation (3.72). The parameter μ for the LMS Gradient Algorithm was then chosen so that the variance of each term of the matrix $2\mu\hat{\Phi}_{XX}$ and the vector $2\mu\hat{\underline{R}}_{XX}$ was equal to the variance of the corresponding matrix and vector in the Jacobi Relaxation Algorithm for a

data window of length M=512. Since for large M the variance of the estimates used in the Adaptive Fixed-Point Iteration Algorithms decrease at a rate of approximately 1/M, μ was selected to be:

$$\mu \doteq \frac{\beta}{2R_{XX}(0)} \cdot \frac{1}{\sqrt{512}} \qquad (5.27)$$

for all the comparisons considered in this section. Equation (5.27) yields values for μ which might realistically be used in practice for all the SNR conditions considered here.

Figure 5.7 gives the comparisons of the theoretical convergence curves for the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Method. Each plot in the figure corresponds to a different SNR condition for the SKEP problem. In all cases, the vertical axis is the MSE axis while the number of sample periods is displayed along the horizontal axis.

It can be seen from Figure 5.7(A) that for the high SNR condition, the three algorithms exhibit approximately the same rate of convergence. However, for the lower SNR conditions the SOR Method exhibits a significantly better convergence rate as can be seen in Figures 5.7(B) through 5.7(D). The convergence rate of the Jacobi Relaxation Method is slightly greater than that of the LMS Gradient Algorithm at the lower signal-to-noise ratios. Thus, it appears from these theoretical curves that the SOR Method can achieve significantly better performance than the LMS Gradient Method or Jacobi Relaxation while still having an average number of operations/sample which is on the order of the length of the filter N. A comparison of the "actual" rates of

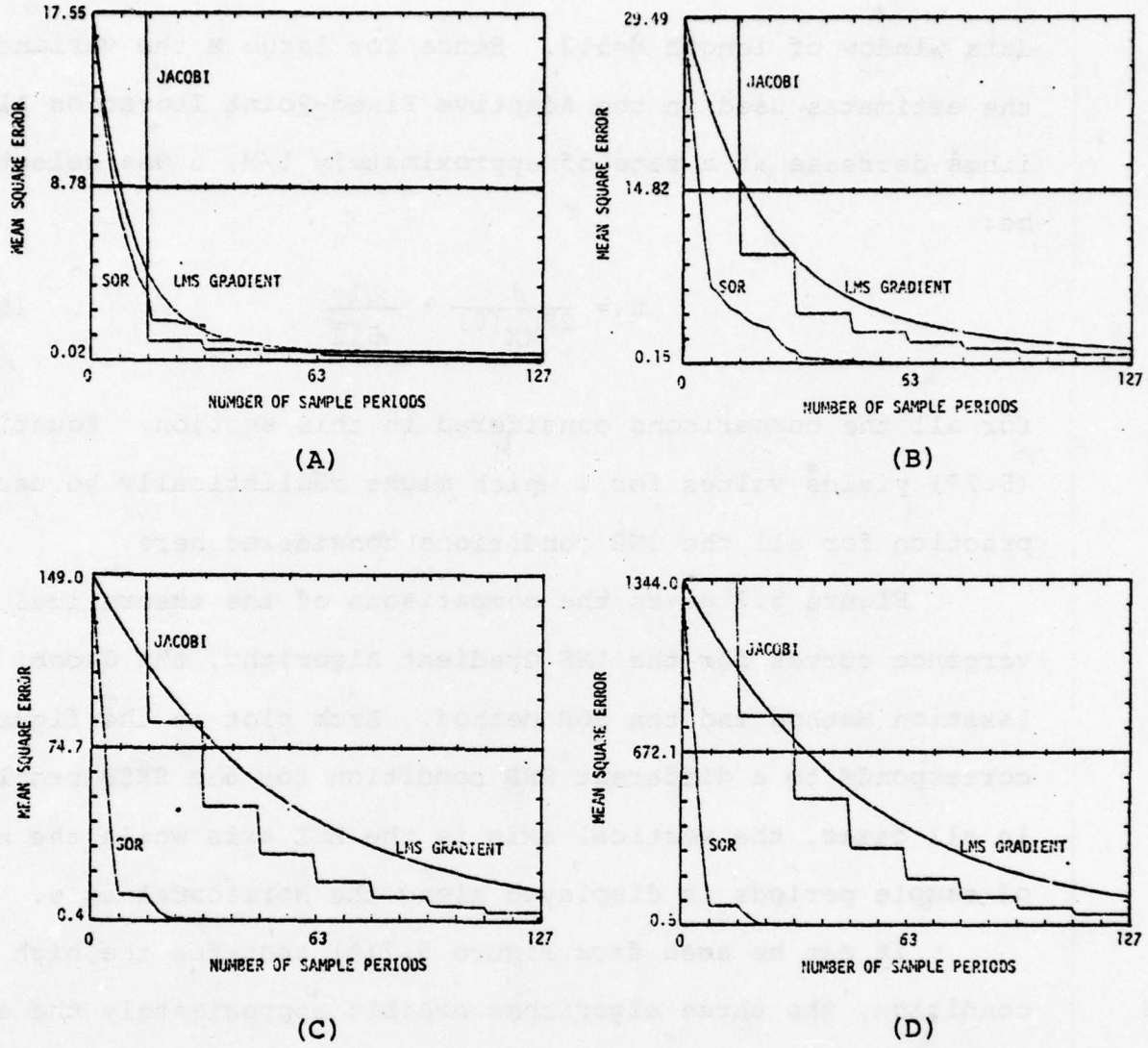


Figure 5.7 Comparison of the Theoretical Performance of the LMS Gradient, Jacobi Relaxation and SOR Algorithms Under Four Different SNR Conditions. The Variable Iteration Step Method has Been Used to Reduce the Number of Operations per Sample Required by the Jacobi and SOR Algorithms to O(N).

PLOT	SNR, db	LMS	JACOBI			SOR		
		μ	β	ℓ	k	β	ℓ	k
A	+ 7.0	0.005	0.14	16	16	1.0	1	1
B	- 3.0	0.002	0.14	16	16	1.0	1	1
C	-13.0	0.00025	0.14	16	16	1.0	1	1
D	-23.0	0.000029	0.14	16	16	1.0	1	1

convergence of the three algorithms will be made via computer simulation in Chapter VI.

5.6 Conclusion

The convergence properties of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Algorithm have been explored by the use of the theoretical bounds derived in Chapters 2 and 3. The dependence of the rate of convergence of the LMS Gradient Algorithm on the value of the parameter μ and of the rates of convergence of the Fixed-Point Iteration Algorithms on the value of the parameter β was demonstrated. A decrease in the value of μ in the LMS Gradient Algorithm or of β in the Jacobi Relaxation Method causes a corresponding increase in the number of sample periods required for the convergence of the respective algorithm. However, such a straightforward relationship between the value of β and the rate of convergence of the SOR Method cannot be obtained in the general case.

The convergence rates of the Jacobi Relaxation Method and the SOR Algorithm when the algorithms were operating under the Variable Iteration Step Method were also considered. As expected, a decrease in the value of the ratio ℓ/k caused a corresponding increase in the number of sample periods required for convergence. However, the Variable Iteration Step Method can be used to reduce the average number of operations/sample required by the two fixed-point iteration algorithms to the O(N) required by the LMS Gradient Method. The total average number of operations/sample

for Adaptive Fixed-Point Iteration is still slightly higher than that required by the LMS Gradient Algorithm due to the O(N) operations/sample required to estimate Φ_{XX} and $\underline{R}_{XX}$.

A comparison of the rates of convergence of the LMS Gradient Method and the two Fixed-Point Iteration Algorithms with ℓ/k chosen to yield O(N) operations/sample and μ chosen to equalize the variance of the estimates of the statistics revealed that the SOR Method was superior to both the LMS Gradient and the Jacobi Relaxation Method. This suggests that the SOR Method can offer significantly better performance than the LMS Gradient Method while still requiring only on the order of N operations per sample. This hypothesis will be tested further in Chapter VI where the "actual" performance of the three iterative algorithms will be compared using computer simulation.

CHAPTER VI

INVESTIGATION OF PERFORMANCE THROUGH COMPUTER SIMULATION

In the previous chapters, the well-known LMS Gradient Method and three other algorithms for use in adaptive linear estimation have been discussed. The LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Algorithm are iterative in nature. Therefore, their rates of convergence are of extreme importance in an evaluation of the performance of these algorithms. In Chapter V, the rates of convergence of these three iterative algorithms were studied for the SKEP problem using the theoretical bounds derived in Chapters II and III. The theoretical bounds give the exact rate of convergence provided that the estimates of the statistics of the input process used by the adaptive algorithm are unbiased and have negligable covariance (i.e., the estimates of the statistics equal their exact values). However, in practice the estimates of the statistics of the input process used by an adaptive iterative algorithm have non-zero covariance and "convergence" of the algorithm must be interpreted as convergence in the mean. If the assumptions made in the derivations of the theoretical bounds in Chapters II and III are reasonably valid, then these bounds should yield a good approximation to the convergence in the mean of the iterative

algorithms. In this chapter, the convergence properties of the LMS Gradient Method, the Jacobi Relaxation Algorithm and the SOR Method are explored via computer simulation. The results of the simulation are compared to the theoretical bounds derived in Chapters II and III to determine how well these bounds predict the performance of the adaptive algorithms for the problem considered.

In contrast to the three algorithms discussed above, Levinson's Algorithm can be used as a non-iterative approach to adaptive estimation. While Levinson's Algorithm is more complex than the three iterative algorithms discussed above, it determines the exact solution of the system:

$$\hat{\Phi}_{XX}\,\underline{h} = \hat{\underline{R}}_{XX} \qquad (6.1)$$

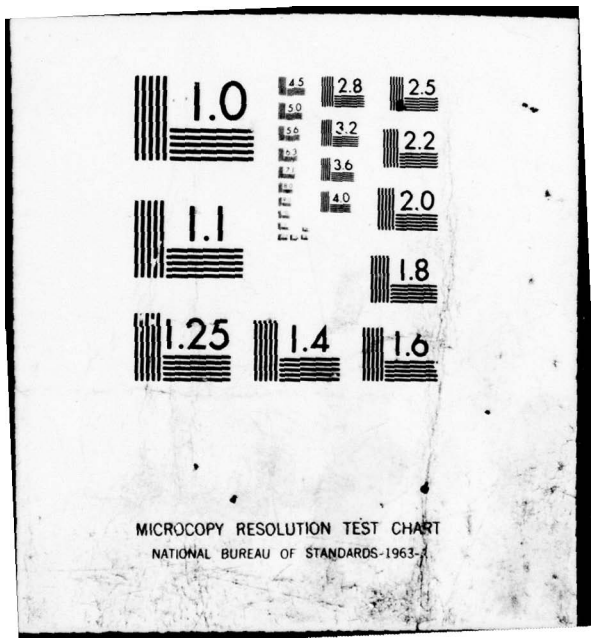
in a single step. However, if Levinson's Algorithm is to be used, $\hat{\Phi}_{XX}$ must be constrained to be a Topelitz matrix as discussed in Chapter IV. The "convergence" of the adaptive estimation procedure using Levinson's Algorithm, then, depends only on the properties of the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ and upon how often Levinson's Algorithm is applied to update $\underline{h}_n$. The "convergence" properties of the adaptive estimation procedure using Levinson's Algorithm proposed in Section 4.2 are also considered in this chapter using computer simulation.

The LMS Gradient Algorithm requires on the order of N operations/sample for its implementation, where N is the filter length. Unfortunately, the other three algorithms considered

AD-A068 760 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/3
ADAPTIVE LINEAR ESTIMATION ALGORITHMS APPLIED TO SPECTRAL LINE --ETC(U)
AUG 78 S D HUFFMAN N00014-75-C-0191
UNCLASSIFIED TR-15 NL

3 OF 4
AD
AO 68760




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

here require on the order of N^2 operations/sample when implemented directly. However, the average number of operations/sample can be reduced for the two Fixed-Point Iteration Algorithms by the use of the Variable Iteration Step Method proposed in Section 3.5.1. A similar approach can be taken for the adaptive estimation method using Levinson's Algorithm as discussed in Section 4.2. Therefore, in this chapter comparisons of the rate of convergence of the four adaptive algorithms will be made where the average number of operations/sample has been reduced to roughly O(N) for the Jacobi Relaxation Method, the SOR Algorithm and Levinson's Algorithm.

The computer simulation results given in this chapter allow the study and comparison of the rates of convergence of the four algorithms considered here under "practical" conditions. That is, no a priori knowledge of the signal or the noise is supplied to any of the algorithms. Thus, the data is actually processed adaptively and the results of the simulation can be used to verify the results and conclusions given in Chapter V which were based on the theoretical bounds derived in Chapters II and III.

6.1 The Problem and Simulation Method

In Chapter V, the convergence properties of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Algorithm were investigated for the signal-known-except-phase problem using the theoretical bounds given in Chapters II and

III. To verify the results and conclusions of Chapter V, the three adaptive algorithms above were simulated on a digital computer for the case of a single sinusoidal signal of fixed frequency and amplitude, but of random phase, in the presence of an autoregressive noise sequence. In addition, the adaptive estimation algorithm using Levinson's Algorithm proposed in Section 4.2 was also simulated as well as the optimal filter for the SKEP problem.

The single sinusoidal signal sequence used in the simulation was generated as:

$$S(n) = A \cos(\Omega n + \Theta) \tag{5.1}$$

where: A = Signal Amplitude

Ω = Signal Frequency (radians)

n = Time Index

Θ = Random phase distributed uniformly on $[-\pi, \pi]$.

The data sequence was then formed by adding an autoregressive noise sequence to the signal sequence. An autoregressive noise sequence was generated as described in Section 5.1 as:

$$N(n) = \sum_{i=0}^{n_1-1} \alpha^i Y(n-i) \tag{5.2}$$

where $Y(i)$ are independent identically distributed random variables with zero mean. For the simulation results given in this chapter, the random variables $Y(i)$ were drawn from a zero

mean, unit variance Gaussian distribution. Therefore, N(n) is also Gaussian with zero mean and variance which can be determined from Equation (5.18). The signal-to-noise ratio was varied by multiplying N(n) by a constant to change the noise variance of the resulting sequence according to the following relation:

$$\text{VAR}[C\ N(n)] = C^2\ \text{VAR}[N(n)] \quad . \tag{6.2}$$

The resulting data sequence given by:

$$X(n) = S(n) + C\ N(n) \tag{6.3}$$

was processed by the four adaptive algorithms considered as well as by the optimal filter for the SKEP problem.

It is important to note that while the data sequence generated for the simulation as described above is composed of a single sinusoidal signal sequence of fixed amplitude and frequency, but with a random phase, added to an autoregressive noise sequence, this information is not available to the adaptive algorithms. That is, while the statistics of the signal and noise are known to an observer, the processors have no a priori knowledge concerning the signal or the noise. Thus, the results in this chapter must be interpreted as the performance of a processor, which is given no a priori knowledge of the signal or noise, conditional to the fact that the data generated consists of a single sinusoid of fixed frequency and amplitude added to

an autoregressive noise sequence.

6.1.1 The Simulation Method

The simulation of four adaptive algorithms was carried out on an IBM 370/165 computer. A source listing of the PL-I program is given in Appendix D. The basic simulation method was to generate a data sequence as discussed in the last section and to allow the four adaptive algorithms to operate on this input data. The instantaneous squared estimation error given by:

$$e(n) = (S(n)-\hat{S}(n))^2 \qquad (6.4)$$

was computed at each value of n for each of the four algorithms. The estimated "mean-square error" was computed as the average of the instantaneous error sequence e(n) over an ensemble of 100 simulation runs. That is, the MSE was estimated as:

$$MSE(n) = \frac{1}{100} \sum_{i=1}^{100} (S_i(n)-\hat{S}_i(n))^2 \quad . \qquad (6.5)$$

A value for the signal phase Θ was chosen randomly before the start of each of the 100 runs. The "performance" curves, which are plots of the estimated mean-square error vs. the time index n were generated in the manner described above. Plots of time records of the signal estimates and plots of the associated spectra are the results of a single run and are not averaged over an ensemble. Thus, the simulation results presented in this chapter must be viewed in light of the fact that the curves

plotted are based on a finite number of realizations of the estimation process and therefore have non-zero variance.

6.2 Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ

A theoretical bound on the rate of convergence of the mean-square error produced by the filter determined by the LMS Gradient Algorithm was derived in Section 2.3. The rate of convergence of the LMS Gradient Algorithm for the SKEP problem was explored for various values of the parameter μ in Section 5.2 under several different SNR conditions using this theoretical bound. In this section, the convergence of the MSE for the LMS Gradient Algorithm is studied via computer simulation. The results of this simulation are compared to the theoretical bound derived in Section 2.3. The simulation was carried out according to the method discussed in Section 6.1. The parameter values required for the generation of the signal and noise sequences were chosen to be identical to those used in Chapter V. Figure 5.1 gives a summary of the parameter values used for generating the data sequence for the simulation. It is again emphasized that while the data generated corresponds to the SKEP problem, the LMS Gradient Algorithm is not given this information. Thus, the results of the computer simulation should be interpreted as the performance of the LMS Gradient Algorithm, which is given no a priori knowledge of the signal or the noise, conditional to the fact that the input data to the LMS Gradient

Algorithm is composed of a single sinusoid of fixed amplitude and frequency added to an autoregressive noise sequence.

The results of the computer simulation of the adaptive estimation procedure using the LMS Gradient Algorithm are given in the form of plots of the estimated mean-square error as a function of the time index n in Figures 6.1 through 6.4. Each figure corresponds to a different value of input signal-to-noise ratio and contains four separate plots. Each of the four plots within a figure shows the theoretical bound on the MSE (smooth curve), as well as the estimated MSE obtained from the computer simulation, for a different value of the parameter μ. In addition, the horizontal base line on each of the plots is the MSE which would result from the use of the optimal filter for the SKEP problem. In all cases, the vertical axis is the MSE axis and the horizontal axis gives the number of sample periods. The signal-to-noise ratios and the values of μ chosen for the simulation are identical to those used in Chapter V and are noted in the figures.

It can be seen from Figures 6.1 through 6.4 that the theoretical bound derived in Chapter II is close to the actual performance of the LMS Gradient Algorithm for the problem considered here. However, it should be noted that the difference between the theoretical bound and the simulation results increases as the SNR decreases. This is due to the fact that the variance of the gradient estimate used by the LMS Gradient Algorithm increases as the SNR decreases in this case. The primary difference between the simulation results and the

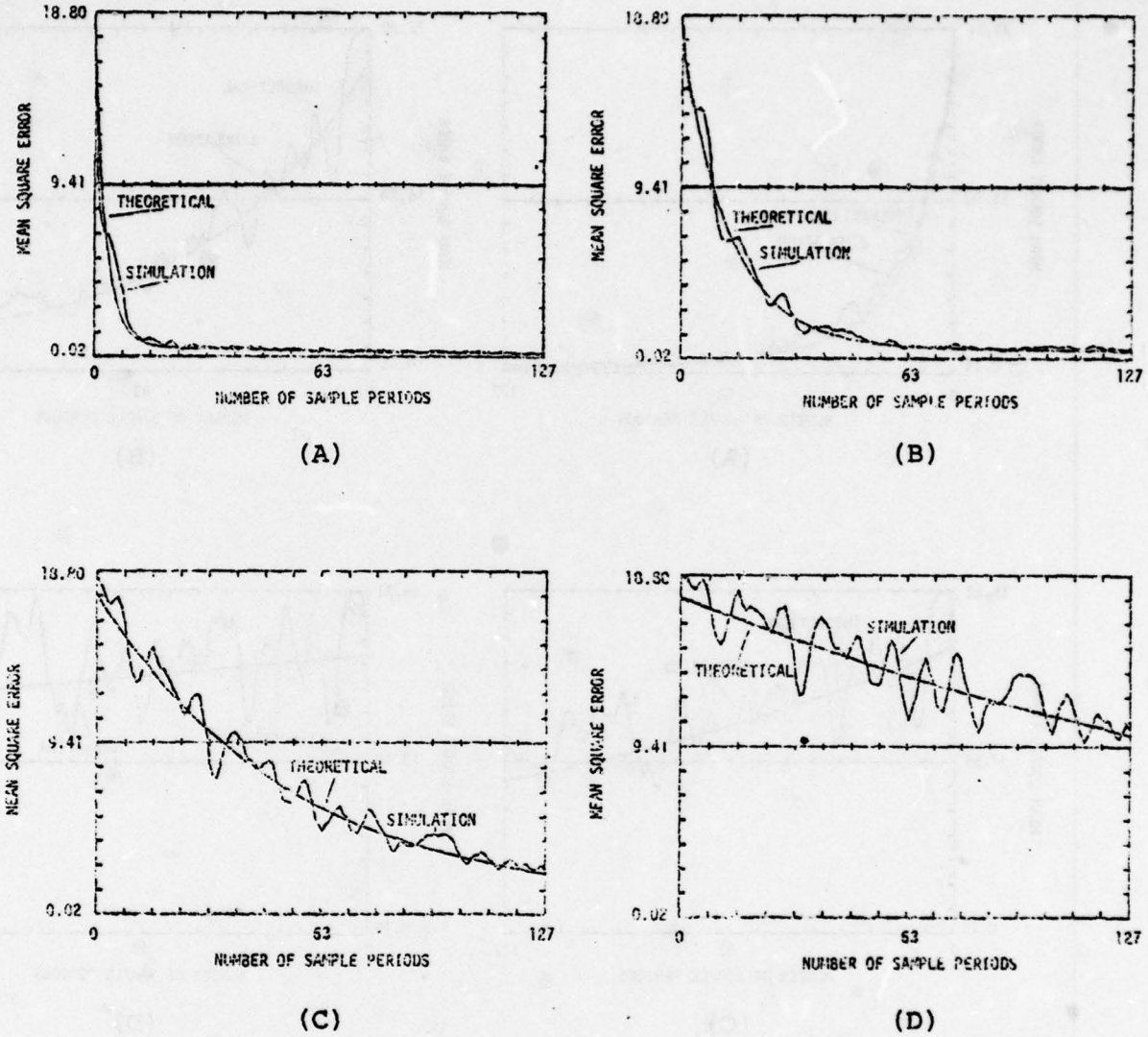


Figure 6.1 Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR = +7.0 db.

(A) $\mu = 0.016$

(B) $\mu = 0.004$

(C) $\mu = 0.001$

(D) $\mu = 0.00025$

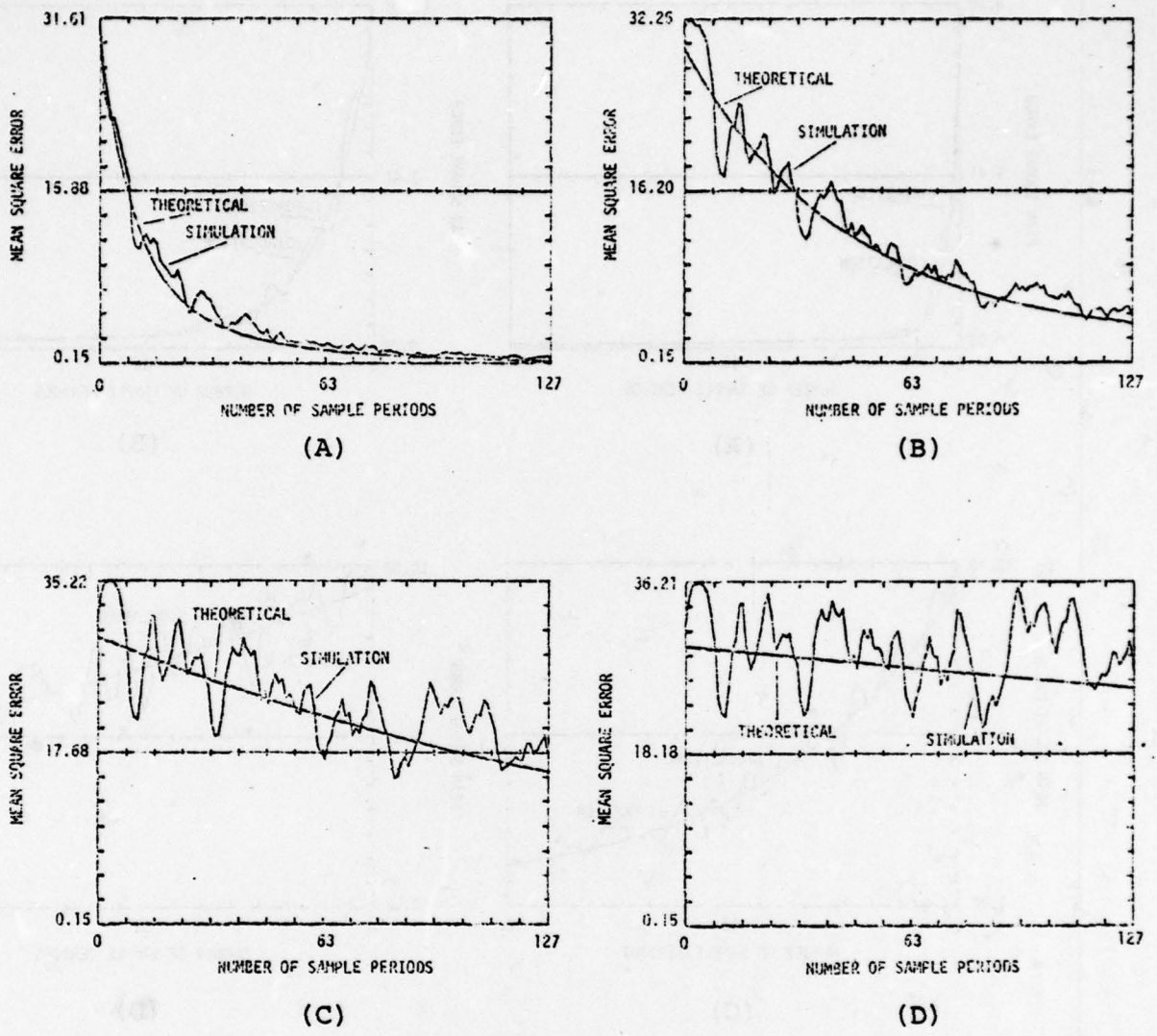


Figure 6.2 Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR = -3.0 db.

(A) $\mu = 0.004$

(B) $\mu = 0.001$

(C) $\mu = 0.00025$

(D) $\mu = 0.0000625$

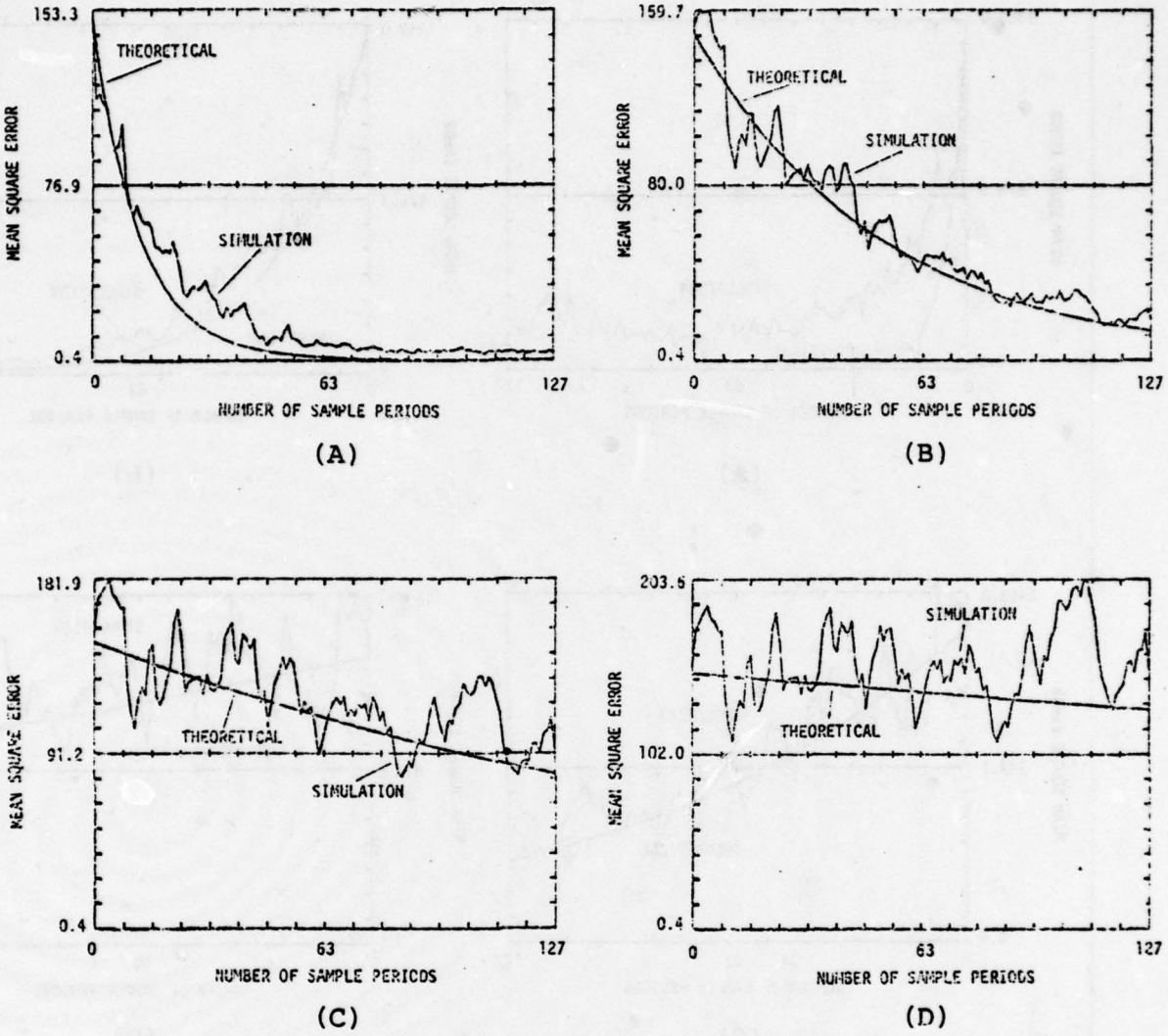


Figure 6.3 Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR = -13.0 db.

(A) $\mu = 0.001$

(B) $\mu = 0.00025$

(C) $\mu = 0.0000625$

(D) $\mu = 0.0000156$

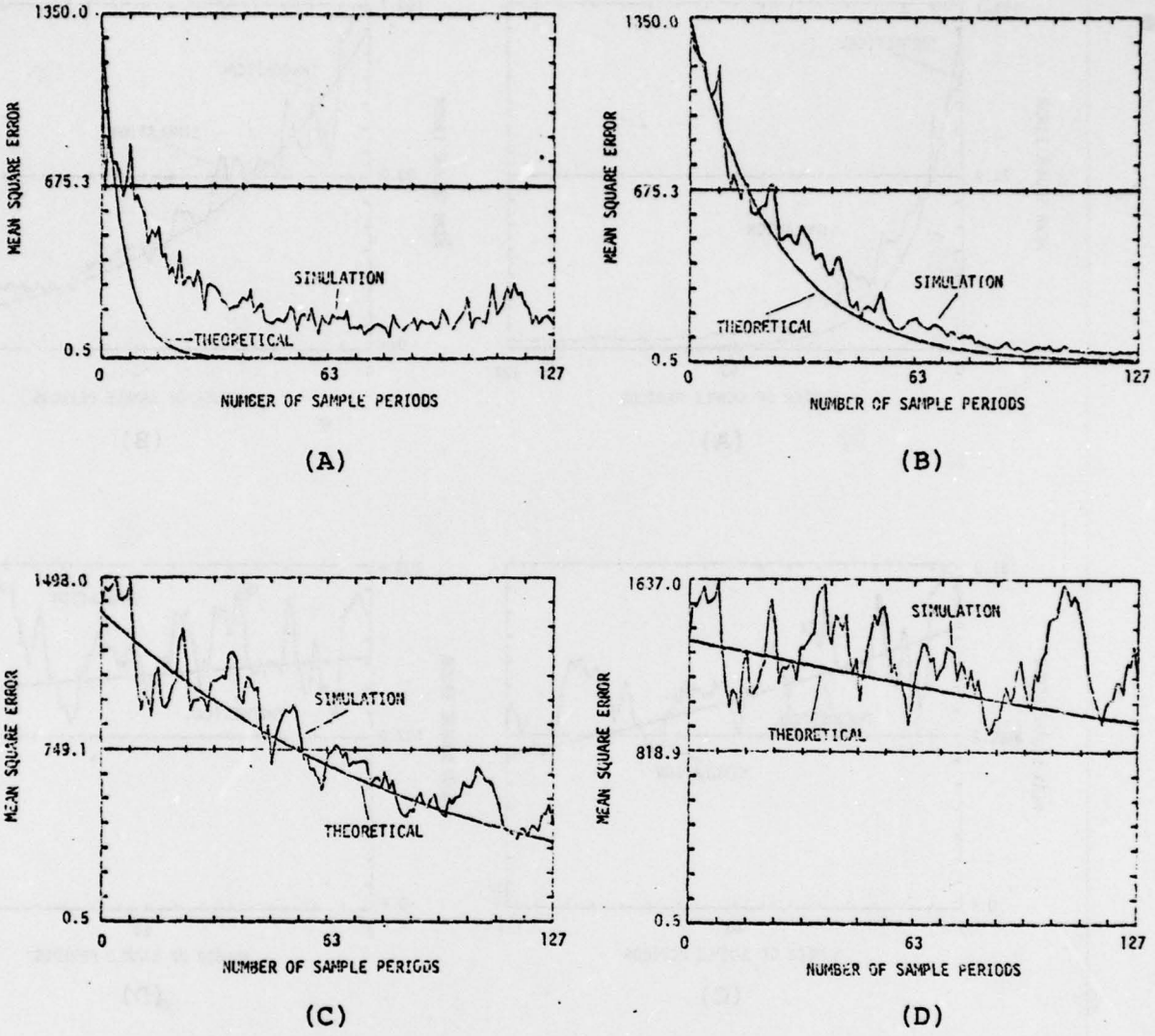


Figure 6.4 Comparison of Actual and Theoretical Performance of the LMS Gradient Algorithm for Various Values of the Parameter μ. SNR = -23.0 db.

(A) $\mu = 0.00025$

(B) $\mu = 0.0000625$

(C) $\mu = 0.0000156$

(D) $\mu = 0.0000029$

theoretical bounds is that for the smaller SNR conditions, the theoretical bound and the simulation results do not appear to converge to the same value. This difference is often referred to as the "misadjustment error" in the literature [7]. Aside from the effects of the misadjustment error, the theoretical bound derived in Section 2.3 is a good approximation to the "actual" MSE(n) produced by the filter determined by the LMS Gradient Algorithm for the problem considered.

6.3 Comparison of the Theoretical Bound and Simulation Results for the Jacobi Relaxation Method.

Equation (3.75) derived in Section 3.3.2 gives a theoretical bound on the rate of convergence of the MSE produced by the filter determined by the Jacobi Relaxation Method. Through the use of this bound, the rate of convergence of the Jacobi Relaxation Method was studied for various values of the relaxation parameter β under several different conditions of input SNR in Section 5.3. In addition, the convergence of the Jacobi Relaxation Method operating under the Variable Iteration Step procedure was studied in Section 5.3.2 for various values of the parameters ℓ and k. In this section, the convergence of the MSE for the filter determined by the Jacobi Relaxation Algorithm is explored through the use of computer simulation. The results of the computer simulation are compared to the theoretical bound derived in Section 3.3.2.

The simulation was carried out according to the method outlined in Section 6.1. The Jacobi Relaxation Method was implemented directly using Equation (3.76) from Section 3.3.3. The autocorrelation matrix and the autocorrelation vector were estimated using Equation (3.5) in Section 3.1. The parameter values required for the generation of the data sequence are identical to those used in Chapter V and are summarized in Figure 5.1. It should again be noted that the results of the simulation must be interpreted as the performance of the Jacobi Relaxation Algorithm, which is not given any a priori knowledge concerning the signal or the noise, conditional to the fact that the input data to the algorithm is composed of a single sinusoid of fixed amplitude and frequency in the presence of an additive autoregressive noise sequence.

6.3.1 Performance of the Jacobi Relaxation Method for Various Values of the Relaxation Parameter β

The convergence of the Jacobi Relaxation Method as a function of the relaxation parameter β was studied in Section 5.3 for various SNR conditions through the use of the theoretical bound on convergence derived in Section 3.3.2. To determine the usefulness of this theoretical bound in predicting the rate of convergence of the Jacobi Relaxation Method, the algorithm was simulated using the same parameter values for the signal and noise as used in Chapter V. Figure 5.1 gives a summary of the parameter values used in generating the data sequence for the simulation.

The results of the computer simulation of the adaptive estimation method using the Jacobi Relaxation Algorithm with various values of the relaxation parameter β are given in Figures 6.5 through 6.8. Each figure is for a different condition of SNR ratio with each plot in a figure corresponding to the estimated mean-square error vs. the time index n for a different value of the relaxation parameter β. In addition, the theoretical bound (smooth curve) for the particular value of β is also shown in each plot to facilitate comparisons. The horizontal base line in each plot is the MSE which would result from the use of the optimal filter for the SKEP problem. In all cases, the vertical axis is the MSE axis and the horizontal axis gives the time index n. The signal-to-noise ratios and the values of β used in the simulation are identical to those used in Chapter V and are noted in the figures.

Note from Figures 6.5 through 6.8 that the "actual" performance of the Jacobi Relaxation Method is close to the theoretical bound derived in Chapter III for the problem considered here. However, as the SNR is decreased, the difference between the theoretical bound and the simulation results increases. The primary difference between the theoretical bound and the simulation results for the lower SNR conditions is that the "actual" MSE appears to converge to a higher value than does the theoretical bound. Since the variance of the estimate of the autocorrelation function given in Section 3.1 decreases roughly as 1/n, the variance decreases rapidly initially, but

slowly for larger values of n. Thus, the actual convergence of the Adaptive Fixed-Point Iteration Method using Jacobi Relaxation is governed initially by the convergence rate of Jacobi Relaxation. However, as n increases, convergence is very slow due to the fact that the variance of the estimate of the statistics decreases slowly. This effect is more noticeable at the lower SNR conditions due to the fact that the variance of the estimate of the autocorrelation function given in Section 3.1 increases as the SNR decreases in this case. Aside from the differences caused by the effect mentioned above, the theoretical bound derived in Section 3.3.2 is a good approximation to the "actual" MSE(n) produced by the filter determined by the Jacobi Relaxation Method, using the estimates of Φ_{XX} and $\underline{R}_{XX}$ proposed in Section 3.1, for the problem considered.

6.3.2 Performance of Jacobi Relaxation Using the Variable-Iteration-Step Method

The Variable-Iteration-Step Method was proposed in Section 3.5.1 as a method for the reduction of the average number of operations/sample for the Fixed-Point Iteration Algorithms. In this procedure, the Jacobi Relaxation Method was used to update ℓ elements of the unit-sample response vector every k sample periods. Thus, if $\ell/k < N$, the average number of operations/sample can be reduced to:

$$\ell/k\ (N+1) < N(N+1)\ . \qquad (5.20)$$

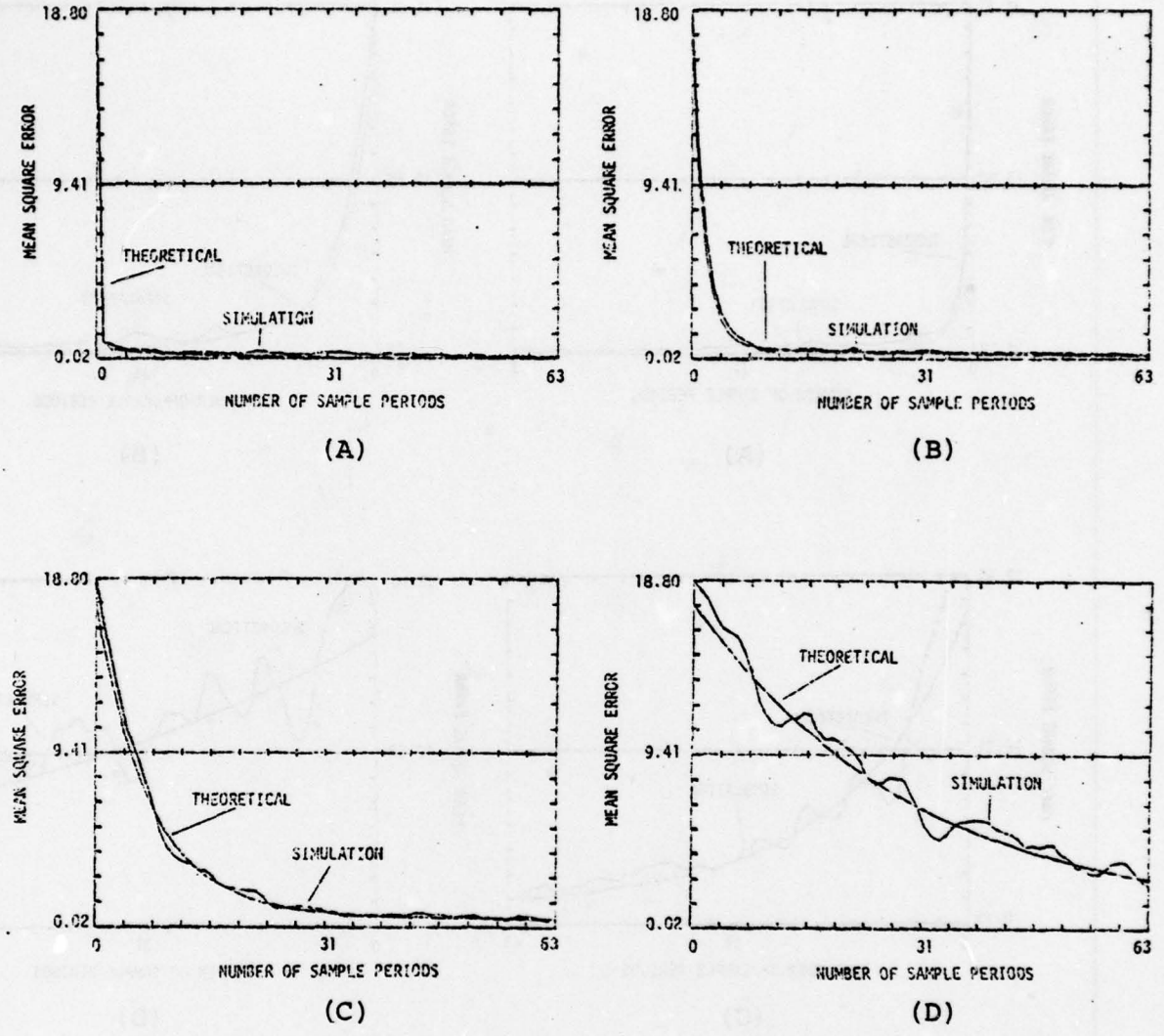


Figure 6.5 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR = +7.0 db.

(A) $\beta = 0.14$

(B) $\beta = 0.035$

(C) $\beta = 0.00875$

(D) $\beta = 0.00218$

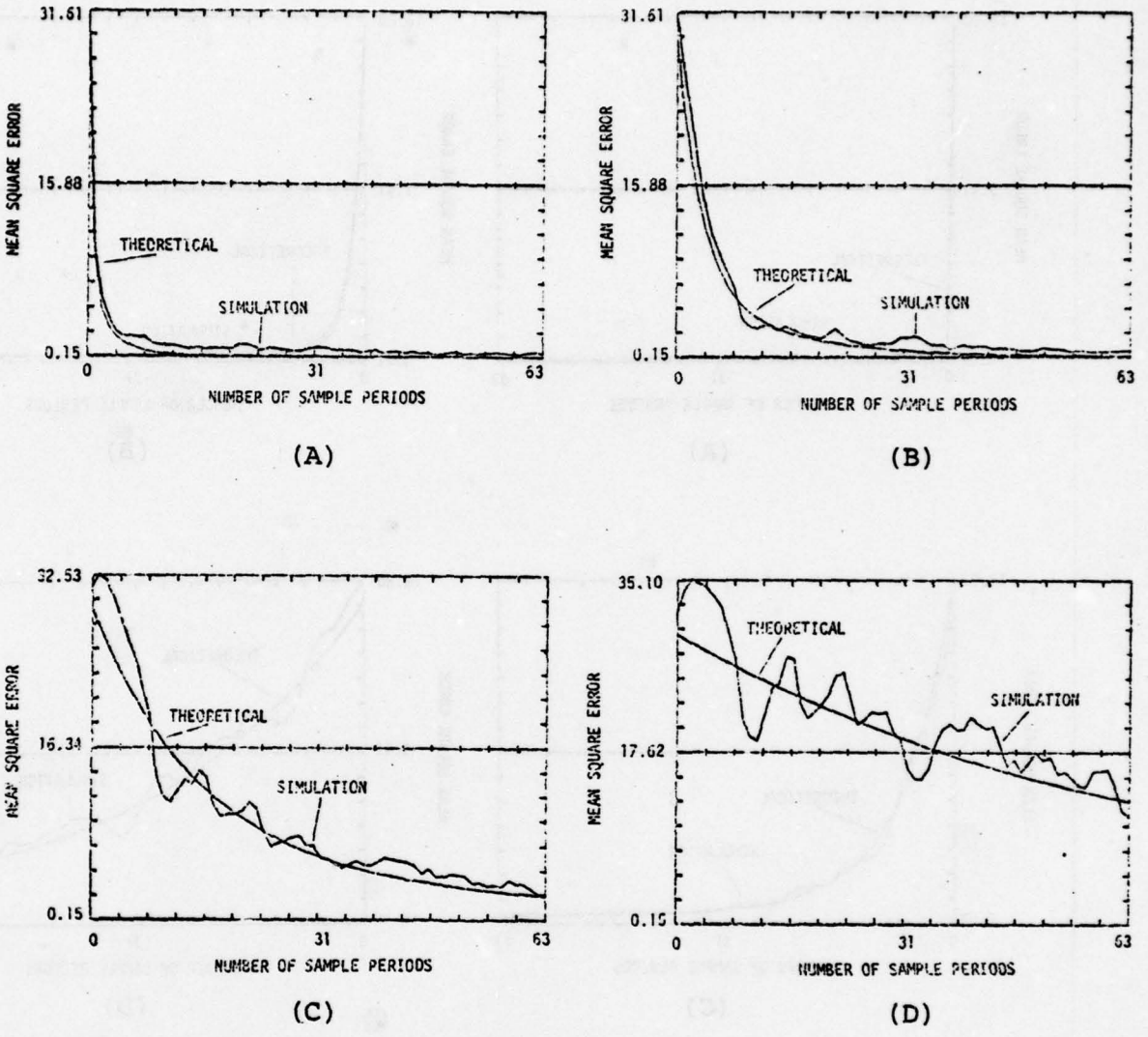


Figure 6.6 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR = -3.0 db.

(A) $\beta = 0.14$

(B) $\beta = 0.035$

(C) $\beta = 8.75\ 10^{-3}$

(D) $\beta = 2.18\ 10^{-3}$

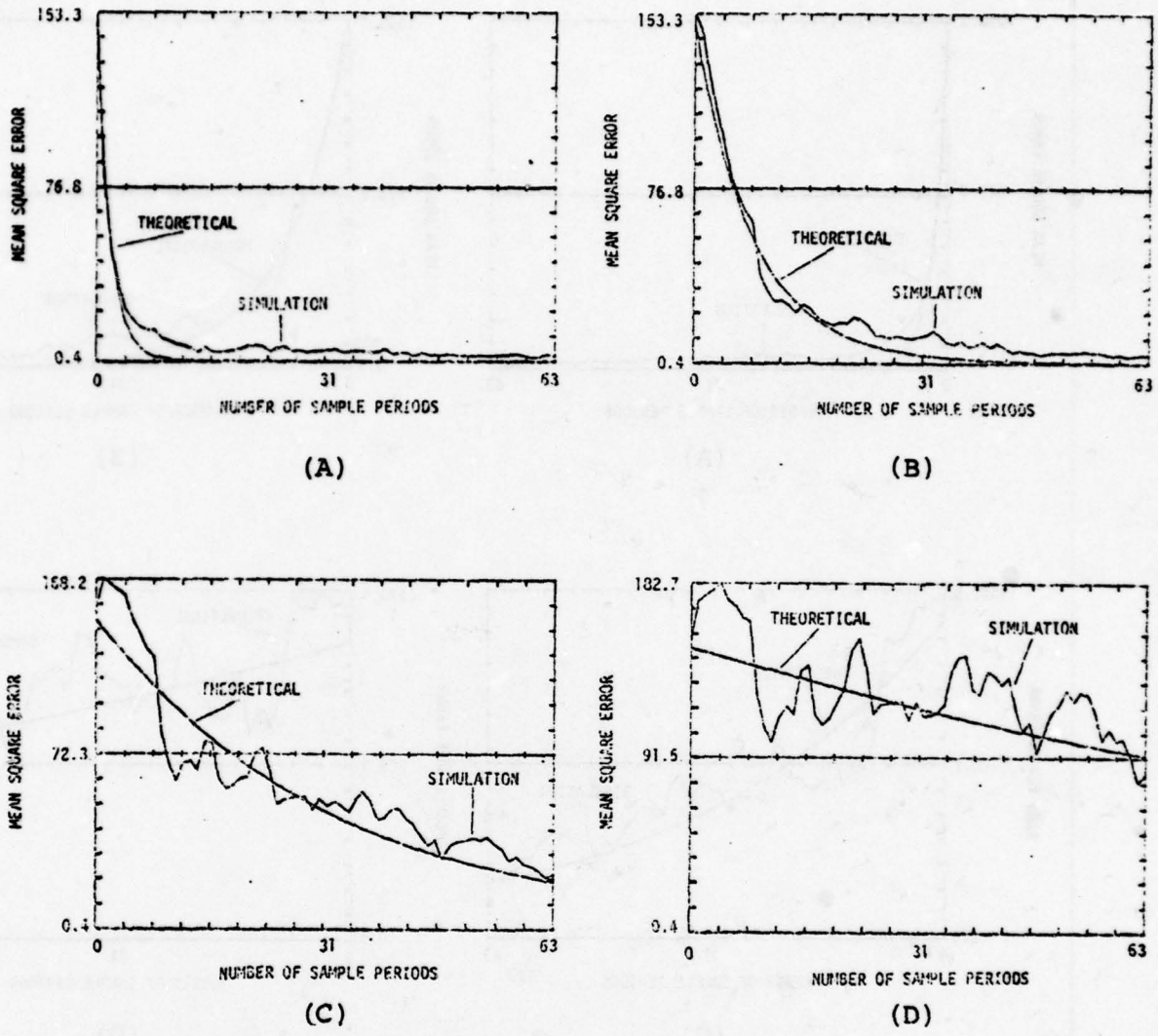


Figure 6.7 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR = -13.0 db.

(A) $\beta = 0.14$

(B) $\beta = 0.035$

(C) $\beta = 8.75\ 10^{-3}$

(D) $\beta = 2.18\ 10^{-3}$

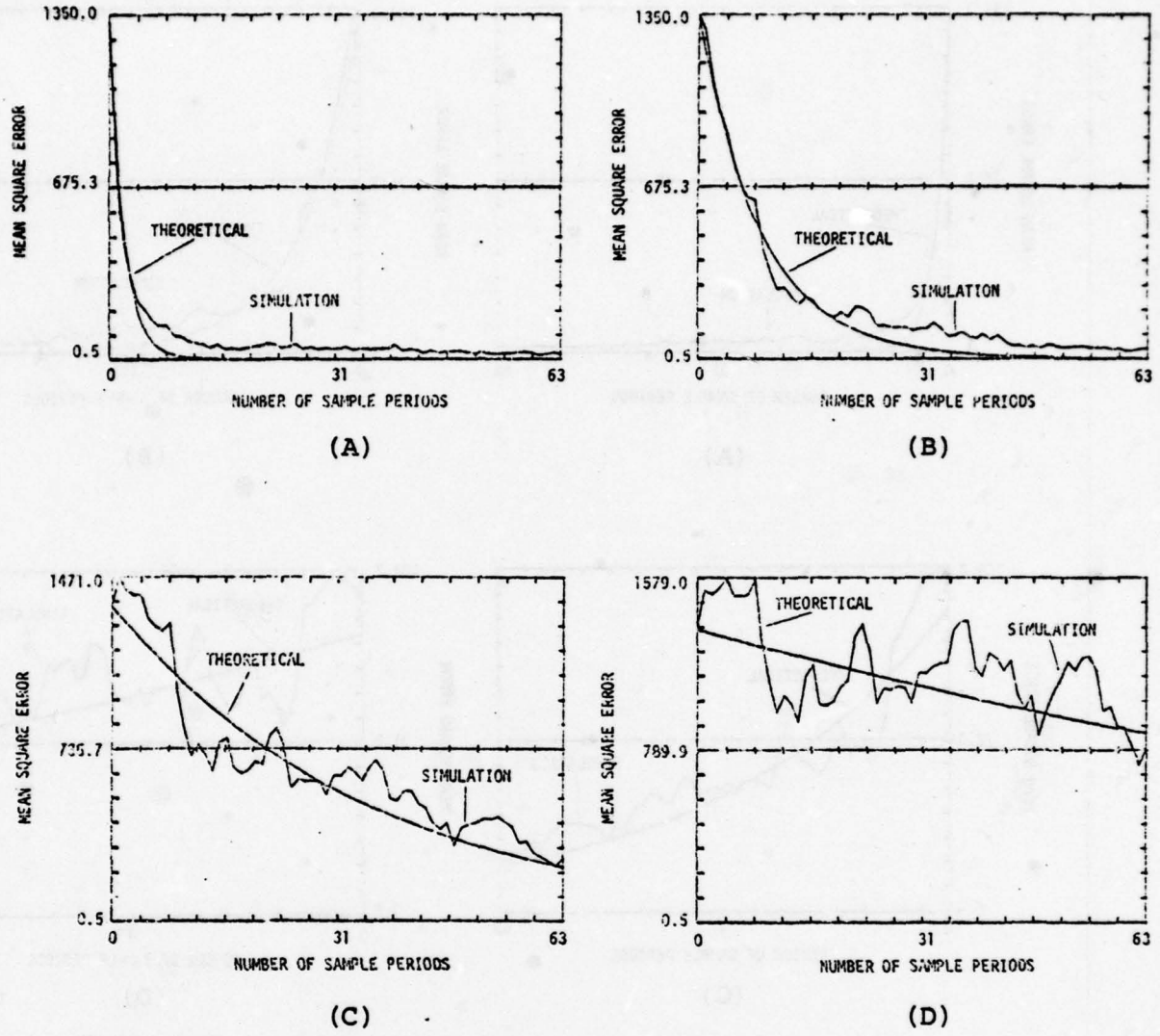


Figure 6.8 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm for Various Values of the Parameter β. SNR = -23.0 db.

(A) $\beta = 0.14$

(B) $\beta = 0.035$

(C) $\beta = 8.75\ 10^{-3}$

(D) $\beta = 2.18\ 10^{-3}$

However, such a reduction in the average number of operations/ sample is accompanied by a corresponding increase in the number of sample periods required for convergence. The rate of convergence of the Jacobi Relaxation Method under the Variable Iteration Step Method was studied in Section 5.3.2 using the theoretical bound derived in Chapter III. The usefulness of this theoretical bound was explored by a computer simulation of the algorithm. As in Section 5.3.2, the value of the relaxation parameter β was chosen to be $\beta=0.14$ as determined from Equation (3.72). The value of ℓ was chosen to be $\ell=N=16$ in all cases and the algorithm was simulated for k=16 and k=32 as was the case in Section 5.3.2.

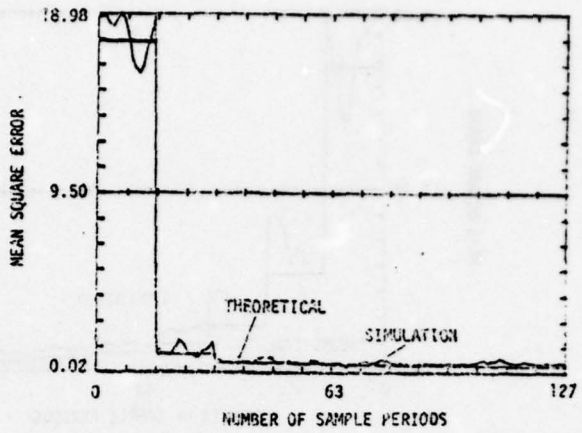
The results of the computer simulations are given in Figures 6.9 through 6.12. Each figure is for a different SNR condition with the plots in the figures giving the "actual" MSE(n) for the Jacobi Relaxation Algorithm using the Variable Iteration Step Method. The two plots in each figure are for the cases k=16 and k-32 respectively. To facilitate comparisons, the theoretical bound for the particular value of k is also included on each plot. The horizontal base line in each plot is the MSE which would result from the use of the optimal filter for the SKEP problem. The vertical axis is the MSE axis in all cases and the horizontal axis gives the number of sample periods. The SNR conditions and k values corresponding to each plot are noted in the figures.

As can be seen from Figures 6.9 through 6.12, the "actual" performance of the Jacobi Relaxation Algorithm using the Variable

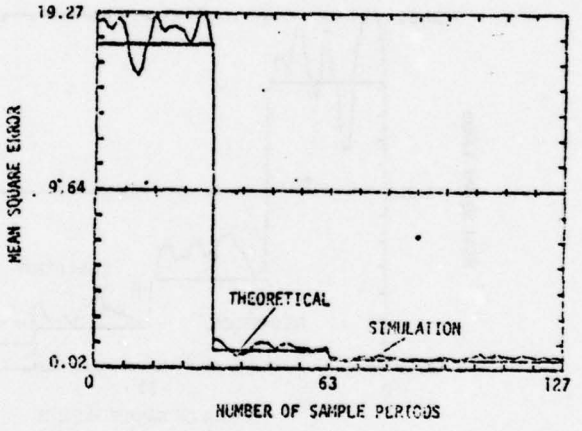
Iteration Step Method is close to that predicted using the theoretical bound. Again, however, the difference between the theoretical bound and the "actual" MSE(n) increases as the SNR decreases. This is again due to the fact that the variance of the estimate of the autocorrelation function given in Section 3.1 increases as the SNR is decreased in this case. Thus, the effect of the rate at which the variance of the autocorrelation function estimate decreases is more noticeable at the lower SNR conditions as discussed in the previous section. Aside from this effect, which causes the "actual" MSE(n) to appear to converge to a value higher than that predicted, the theoretical bound gives a good approximation to the "actual" MSE(n) produced by the filter determined by the Jacobi Relaxation Algorithm using the Variable Iteration Step Method.

6.4 Comparison of the Theoretical Bound and Simulation Results for the Successive Over-Relaxation Method

A theoretical bound on the convergence of the mean-square error produced by the filter determined by the SOR Method was derived in Section 3.4.1 and is given by equations (3.91) and (3.92). The convergence properties of the SOR Method for various values of the relaxation parameter β were investigated for several different SNR conditions using this bound in Section 5.4.1. Also, the convergence of the SOR Algorithm operating under the Variable Iteration Step Method was studied in Section 5.4.2 for various values of the parameters ℓ and k. In this



(A)

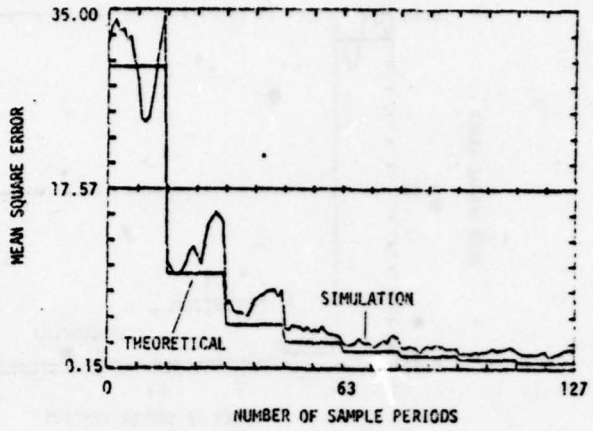


(B)

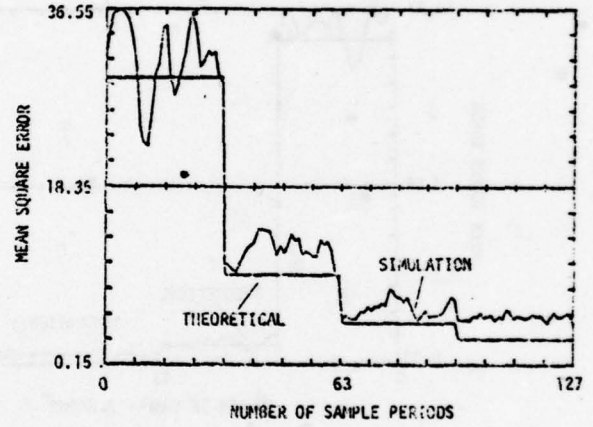
Figure 6.9 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm Using the Variable-Iteration-Step Method. β=0.14. SNR = +7.0 db.

(A) ℓ=16, k=16

(B) ℓ=16, k=32



(A)

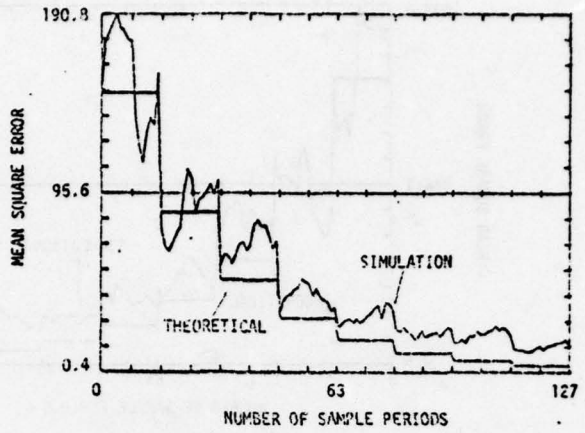


(B)

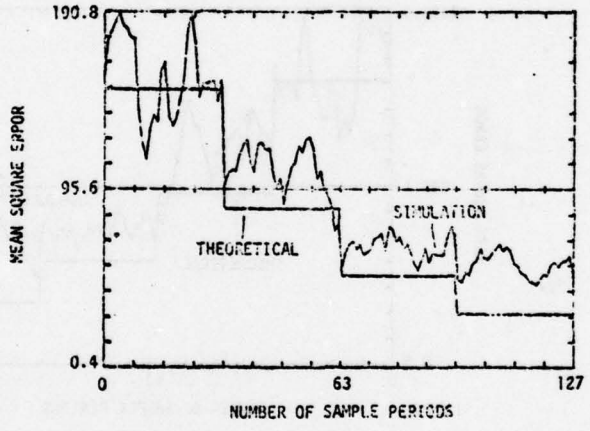
Figure 6.10 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm Using the Variable-Iteration-Step Method. β=0.14. SNR = -3.0 db.

(A) ℓ=16, k=16

(B) ℓ=16, k=32



(A)

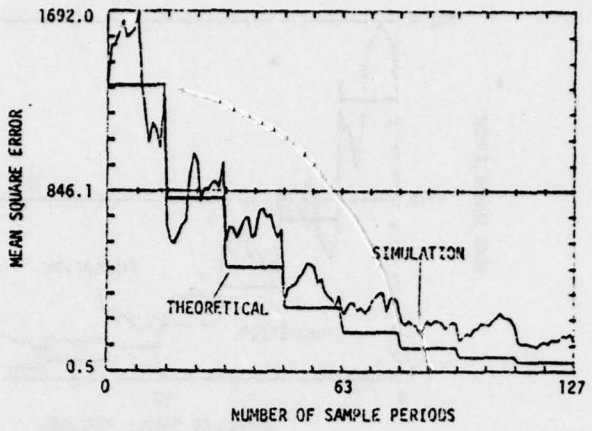


(B)

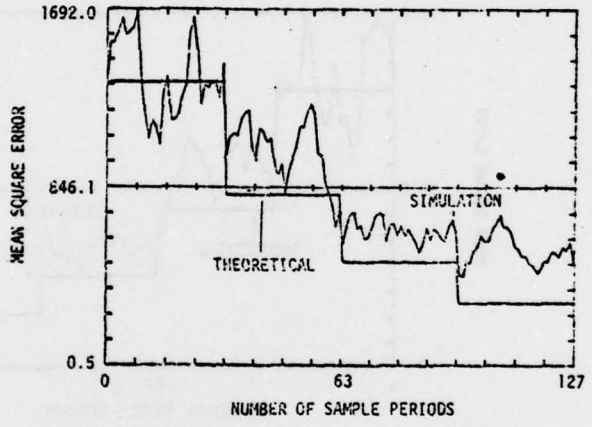
Figure 6.11 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm Using the Variable-Iteration-Step Method. $\beta=0.14$. SNR = -13.0 db

(A) $\ell=16$, $k=16$

(B) $\ell=16$, $k=32$



(A)



(B)

Figure 6.12 Comparison of Actual and Theoretical Performance of the Jacobi Relaxation Algorithm Using the Variable-Iteration-Step Method. β=0.14. SNR = -23.0 db

(A) ℓ=16, k=16

(B) ℓ=16, k=32

section, the convergence of the MSE for the filter determined by the SOR Algorithm is explored through the use of computer simulation. The results of the computer simulation are compared to the theoretical bounds derived in Section 3.4.1.

The method outlined in Section 6.1 was used in carrying out the simulation of the SOR Algorithm. The SOR Method was implemented using Equation (3.85) from Section 3.4.2 and the autocorrelation matrix and vector were estimated using Equations (3.6) and (3.7) from Section 3.1. The data sequence used in the simulation was generated using parameter values identical to those used in Chapter V, which are summarized in Figure 5.1. As in the case of the computer simulations of the LMS Gradient and Jacobi Relaxation Algorithms, the results of this simulation should be interpreted as the performance of the SOR Method, which has no a priori knowledge of the signal or the noise, conditional to the fact that the input data to the algorithm is composed of a single sinusoid of fixed frequency and amplitude in the presence of an additive autoregressive noise sequence.

6.4.1 Performance of the SOR Method for Various Values of the Relaxation Parameter β

The theoretical bound derived in Section 3.4.1 was used to study the convergence properties of the SOR Algorithm for various values of the relaxation parameter β under several SNR conditions in Section 5.4. The SOR Algorithm was simulated using the same parameter values for the signal and noise as used in Chapter V to determine the usefulness of this theoretical

bound in predicting the convergence of the SOR Algorithm.

Figures 6.13 through 6.16 give the results of the computer simulations of the adaptive estimation method using the SOR Algorithm for various values of the relaxation parameter β. Each figure is for a different condition of SNR with each plot in a figure corresponding to the estimated MSE vs. the time index n for a different value of the relaxation parameter β. In addition, the theoretical bound corresponding to the particular value of β is also included in each plot to facilitate comparisons. The MSE which would result from the use of the optimal filter for the SKEP problem is given by the horizontal base line in each plot. In all cases, the MSE is given along the vertical axis and the number of sample periods is given along the horizontal axis. The values of the parameter β and the SNR conditions used in the simulation are identical to those used in Chapter V and are noted in the figures.

It can be seen from Figures 6.13 through 6.16 that the actual performance of the SOR Algorithm is close to the theoretical bound derived in Chapter III for the case considered here. However, it appears from the figures that the "actual" MSE(n) converges to a value which is greater than that predicted by the theoretical bound. This situation is more noticeable at the lower values of SNR and is due to the effect of the rate at which the variance of the estimate of the autocorrelation function given in Section 3.1 decreases as was the case with the Jacobi Relaxation Algorithm. This effect is more noticeable in the curves for the SOR Algorithm due to its high rate of

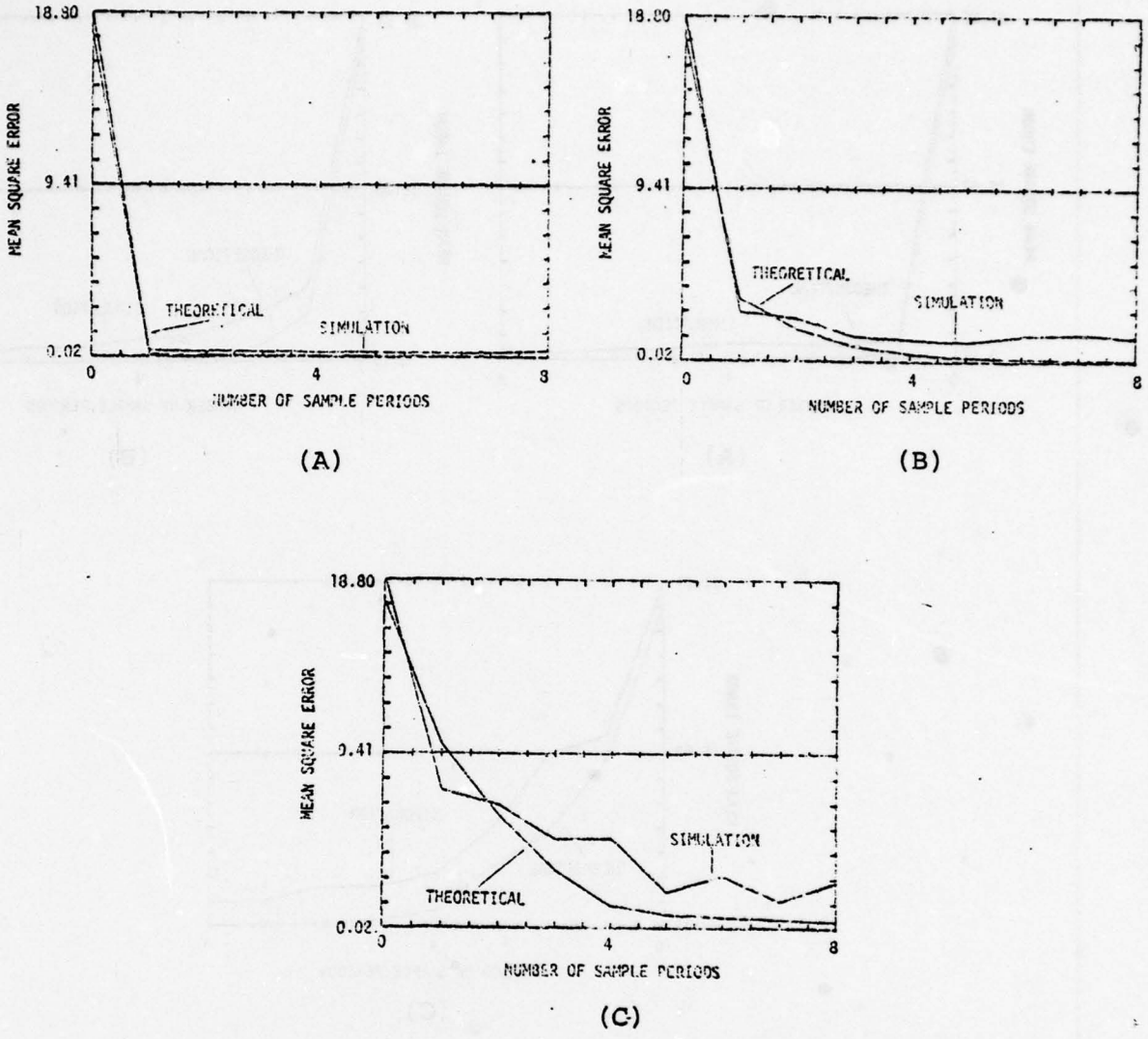


Figure 6.13 Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=+7.0 db

(A) β = 0.5

(B) β = 1.0

(C) β = 1.5

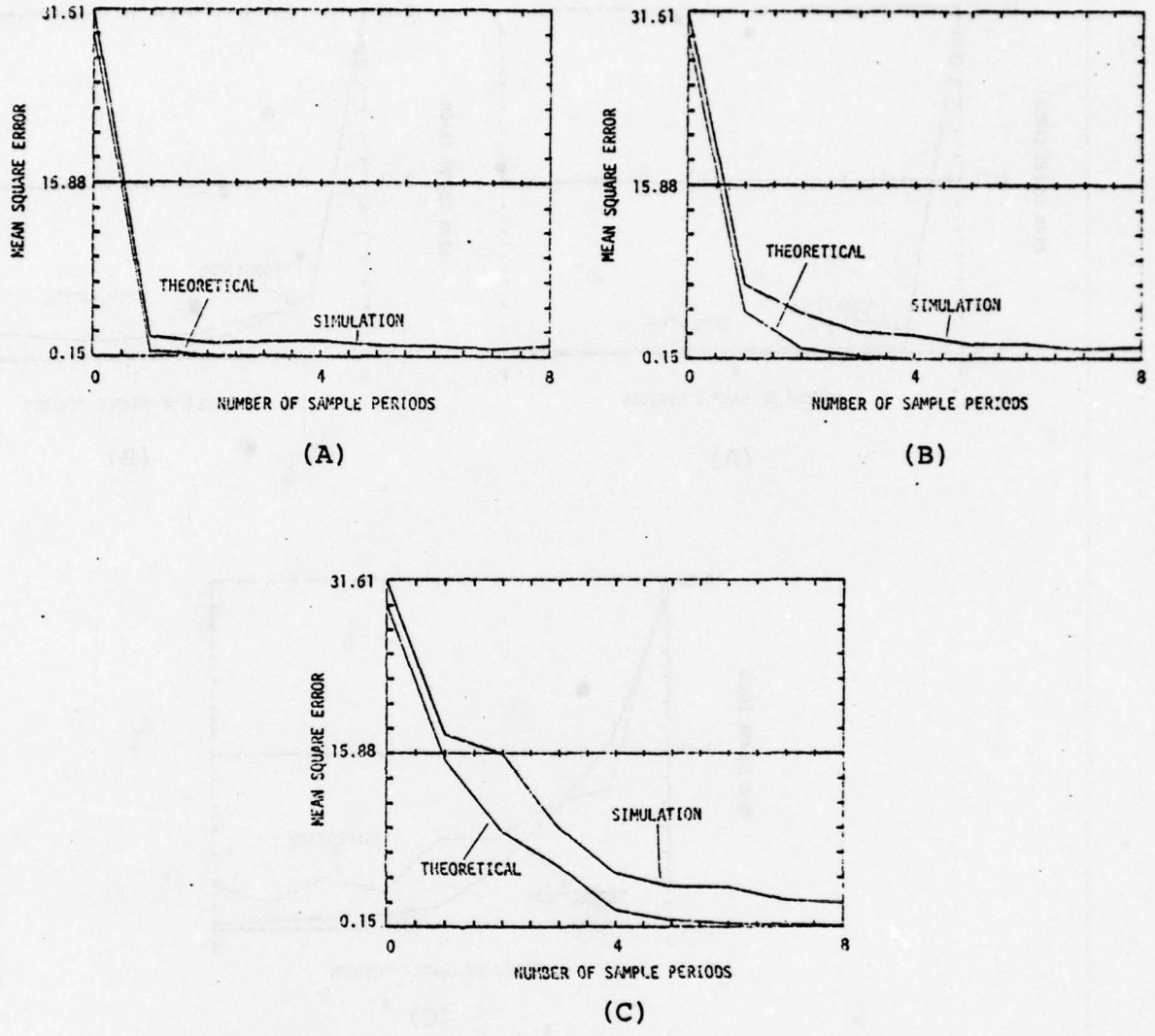


Figure 6.14 Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=-3.0 db

(A) $\beta = 0.5$

(B) $\beta = 1.0$

(C) $\beta = 1.5$

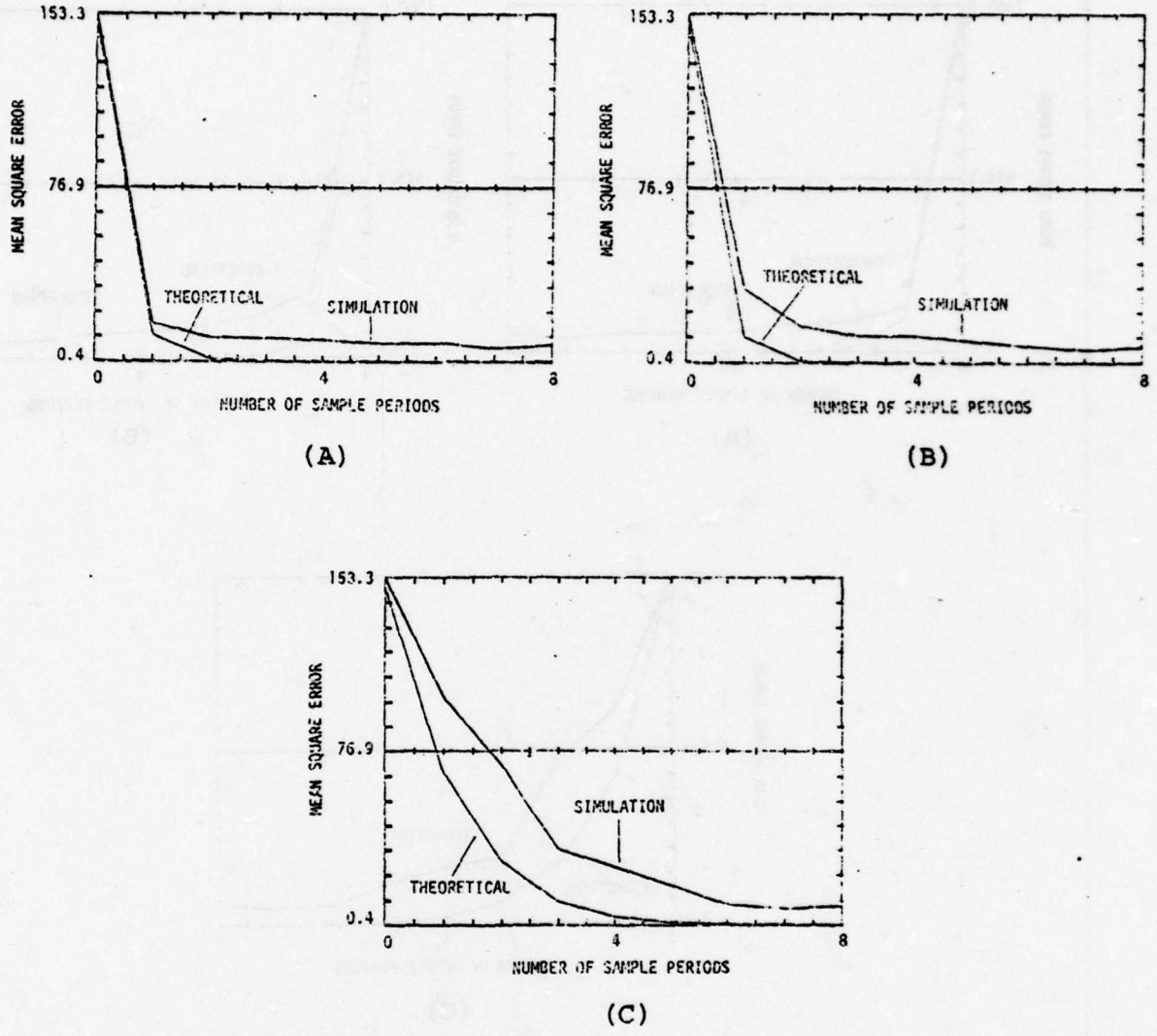


Figure 6.15 Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=-13.0 db

(A) $\beta = 0.5$

(B) $\beta = 1.0$

(C) $\beta = 1.5$

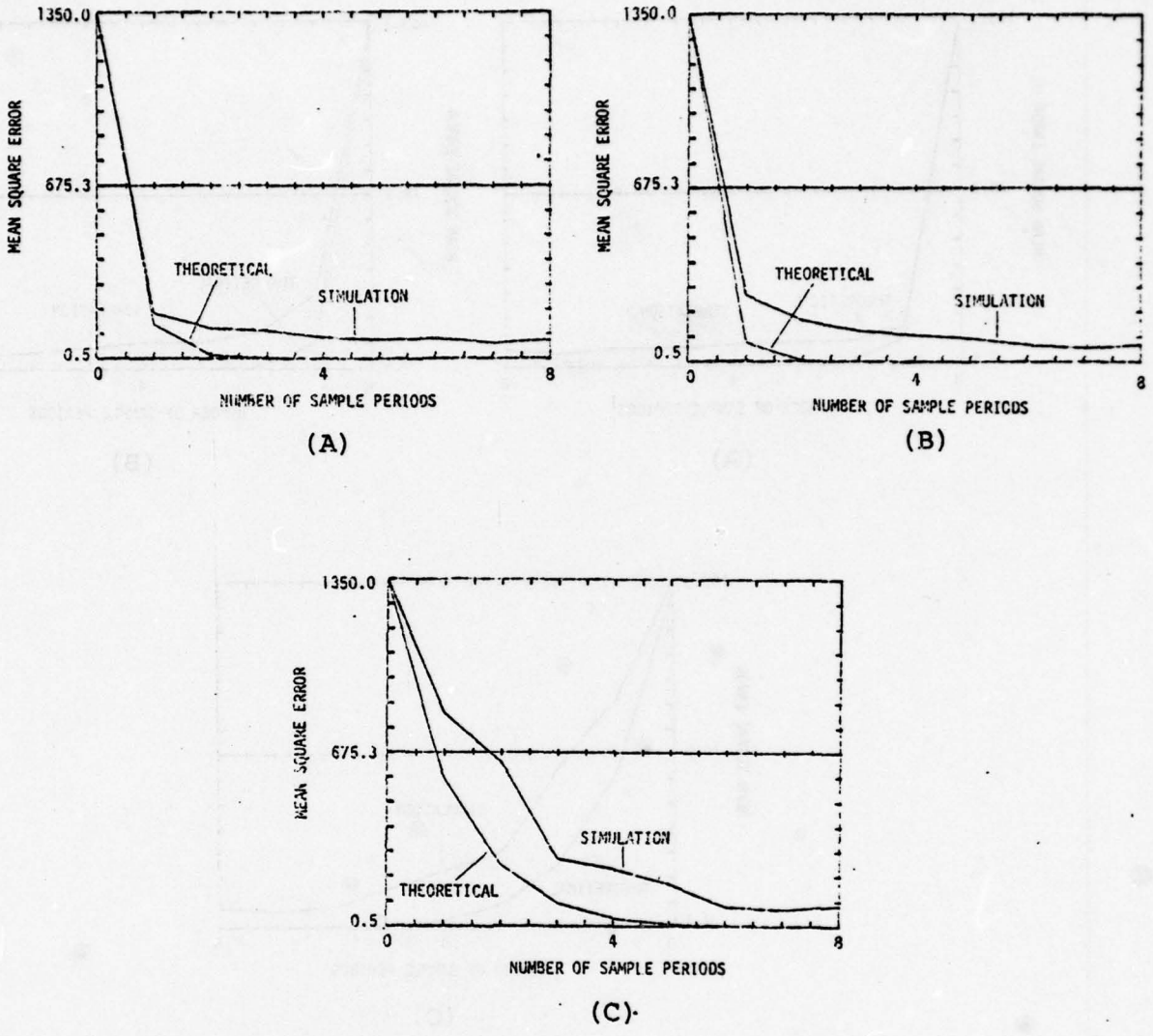


Figure 6.16 Comparison of Actual and Theoretical Performance of the SOR Algorithm for Various Values of the Parameter β. SNR=-23.0 db

(A) $\beta = 0.5$

(B) $\beta = 1.0$

(C) $\beta = 1.5$

convergence (note that the plots only cover a range of $0 \leq n \leq 8$). Aside from differences due to this effect, the theoretical bound derived in Section 3.4.1 is a good approximation to the "actual" MSE(n) produced by the filter determined by the SOR Algorithm.

6.4.2 Performance of the SOR Algorithm Using the Variable Iteration Step Method

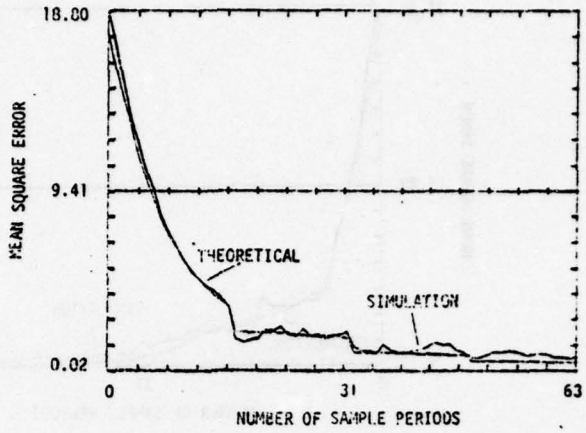
The Variable-Iteration-Step Method was proposed in Section 3.5.1 as a method for the reduction of the average number of operations/sample required by the fixed-point iteration algorithms. In this procedure, the SOR Algorithm is used to update ℓ elements of the unit sample response every k samples. Therefore, if $\ell/k < N$, then the average number of operations/sample can be reduced to:

$$\ell/k \ (N+1) < N(N+1) \quad . \tag{5.21}$$

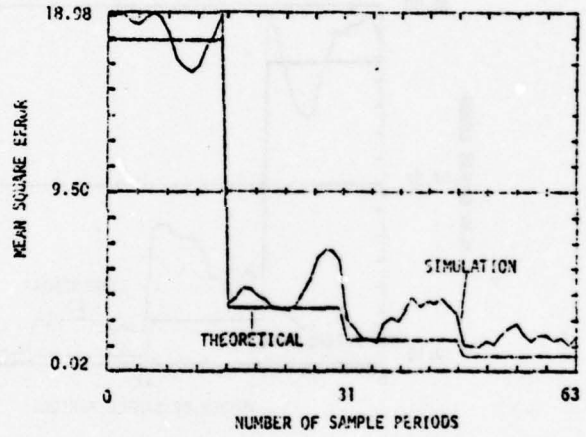
Such a reduction is, obviously, accompanied by a corresponding increase in the number of sample periods required for convergence. The performance of the SOR Algorithm using the Variable-Iteration-Step Method was investigated in Section 5.4.2 through the use of the theoretical bound given by Equations (3.91) and (3.92). The usefulness of this theoretical bound was studied by the use of a computer simulation of the algorithm. For the simulation, the value of the relaxation parameter was chosen to be $\beta = 1.0$ and the algorithm was simulated for the two cases of (ℓ=1, k=1) and (ℓ=16, k=16) as in Section 5.4.2.

Figures 6.17 through 6.20 give the results of the computer simulation of the SOR Algorithm using the Variable-Iteration-Step Method. Each figure corresponds to a different SNR condition with the plots in each figure giving the "actual" MSE(n) for the SOR Algorithm using the Variable-Iteration-Step Method. The two plots in each figure are for the cases of (ℓ=1, k=1) and (ℓ=16, k=16) respectively, the theoretical bound for the particular values of ℓ and k is included in each plot to facilitate comparisons. The MSE which would result from the use of the optimal filter for the SKEP problem is given by the horizontal base line in each plot. In all cases, the vertical axis is the MSE axis and the horizontal axis gives the number of sample periods.

Note from the figures that the "actual" performance of the SOR Algorithm using the Variable-Iteration-Step Method is close to that predicted using the theoretical bound. Also note that the difference between the theoretical bound and the "actual" MSE(n) increases as the SNR decreases and that the two curves appear to "converge" to different values. These differences are again due to the fact that the variance of the estimate of the autocorrelation function given in Section 3.1 increases as the SNR decreases in this case making the effect of the rate at which the variance decreases more noticeable at the lower SNR conditions as discussed in previous sections. Aside from the differences caused by this effect, the theoretical bound gives a good approximation to the "actual" MSE(N) produced by the filter determined by the SOR Algorithm using the



(A)

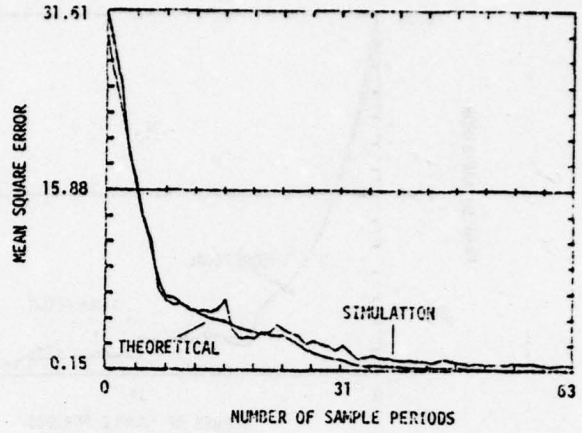


(B)

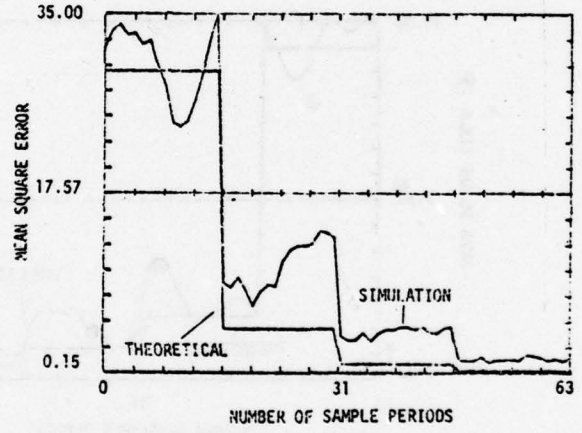
Figure 6.17 Comparison of Actual and Theoretical Performance of the SOR Algorithm Using the Variable Iteration Step Method. β=1.0. SNR=+7.0 db.

(A) ℓ=1, k=1

(B) ℓ=16, k=16



(A)

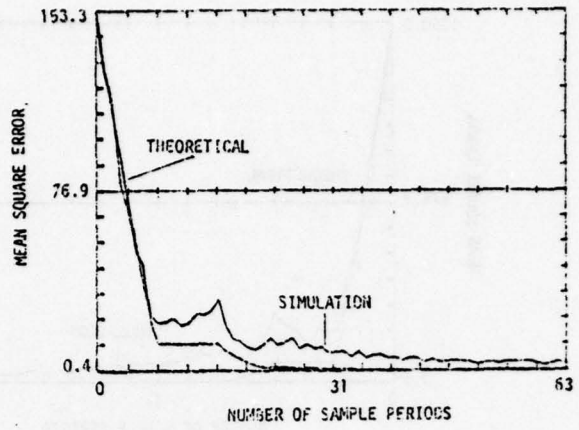


(B)

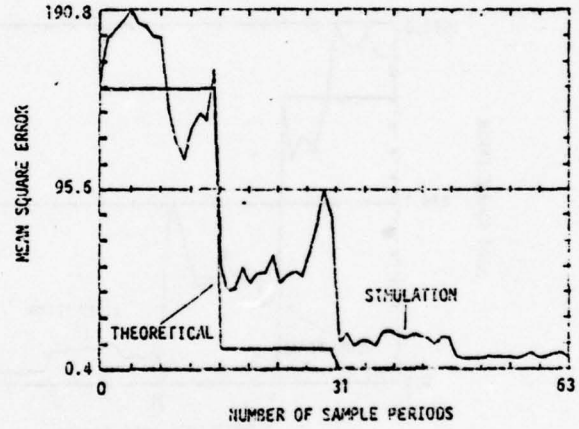
Figure 6.18 Comparison of Actual and Theoretical Performance of the SOR Algorithm Using the Variable Iteration Step Method. β=1.0. SNR=-3.0 db.

(A) ℓ=1, k=1

(B) ℓ=16, k=16



(A)

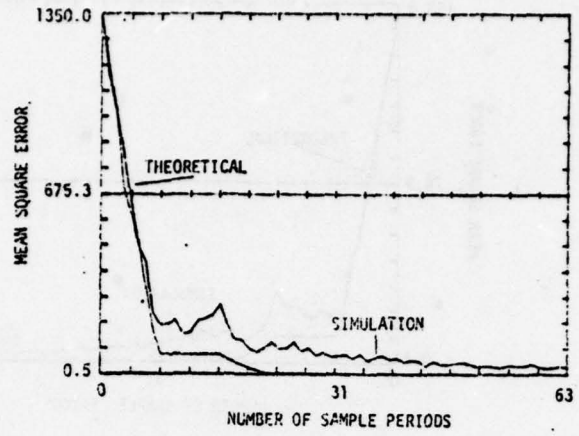


(B)

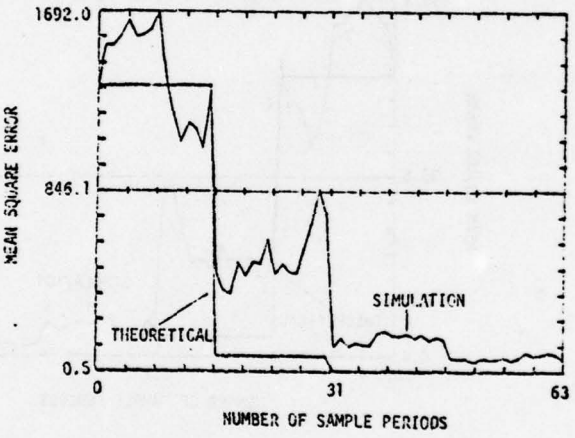
Figure 6.19 Comparison of Actual and Theoretical Performance of the SOP Algorithm Using the Variable Iteration Step Method. β=1.0. SNR=-13.0 db.

(A) ℓ=1, k=1

(B) ℓ=16, k=16



(A)



(B)

Figure 6.20 Comparison of Actual and Theoretical Performance of the SOR Algorithm Using the Variable Iteration Step Method. β=1.0. SNR=-23.0 db.

(A) ℓ=1, k=1

(B) ℓ=16, k=16

Variable-Iteration-Step Method for the problem considered here.

6.5 Comparison of Adaptive Estimation Algorithms Through Computer Simulation

In the preceeding chapters four algorithms for adaptive linear estimation have been considered. The convergence properties of the LMS Gradient Algorithm, the Jacobi Relaxation Algorithm and the SOR Algorithm have been studied under various conditions via computer simulation in the previous section of this chapter. In this section, the "actual" convergence properties of the LMS Gradient Algorithm, the Jacobi Relaxation Method, the SOR Algorithm and Levinson's Algorithm will be compared. The "actual" curves of MSE vs. the time index n for the four algorithms were generated by a computer simulation carried out according to the method outlined in Section 6.1. As mentioned previously, the simulation results should be interpreted as the performance of adaptive algorithms, which have no a priori knowledge of the signal or the noise, conditional to the fact that the input data to the algorithms is composed of a single sinusoid of fixed frequency and amplitude added to an autoregressive noise sequence.

The LMS Gradient Algorithm requires fewer operations/sample than the two fixed-point iteration algorithms or Levinson's Algorithm. Also, the estimate of the input statistics used by the LMS Gradient Algorithm has a larger variance than the estimate proposed in Section 3.1 which was used with the other three algorithms. Therefore, any meaningful comparison of

the four algorithms should attempt to equalize the number of operations/sample and the variance of the estimate of the statistics across the four algorithms. This problem was addressed in Section 5.5, where the theoretical bounds derived in Chapters II and III were used to compare the performance of the LMS Gradient Algorithm, the Jacobi Relaxation Algorithm and the SOR Algorithm. In that section, the Variable-Iteration-Step Method was used with the Jacobi Relaxation and SOR Algorithms to approximately equalize the average number of operations/sample. An attempt to equalize the variance of the estimates of the statistics of the input process was made by selecting the value of the parameter according to Equation (5.27). In the computer simulations considered here, the parameter values chosen for the LMS Gradient Algorithm, the Jacobi Relaxation Algorithm and the SOR Algorithm are identical to those used in Section 5.5. Levinson's Algorithm was implemented as outlined in Chapter IV and was applied at every k=2N=32 sample periods in the simulation to reduce the average number of operations per sample to O(N) as discussed in Section 4.2. The parameter values used for the algorithms for each SNR condition simulated as well as the total average number of operations/sample for each algorithm are summarized in Figure 6.21. As was the case in the computer simulations discussed in the previous sections of this chapter, the parameter values required for the generation of the input data are identical to those summarized in Figure 5.1.

Figures 6.22 through 6.33 give the results of the computer simulation of the four adaptive estimation algorithms considered

here for various SNR conditions. Figures 6.22, 6.25, 6.28 and 6.31 give plots of the estimated MSE vs. the time index n for the four algorithms for SNR conditions of +7.0 db, -3.0 db, -13.0 db and -23.0 db respectively. These curves were obtained by taking the average of the instantaneous squared error sequence given by Equation (6.3) over an ensemble of 100 simulation runs. The true signal sequence, optimal signal estimate and signal estimates produced by the four adaptive algorithms are shown in Figures 6.23, 6.26, 6.29 and 6.32. These estimates are the results of a single simulation run and are not averaged over an ensemble. The results given in these figures for SNR conditions of +7.0 db, -3.0 db, -13.0 db and -23.0 db respectively. Figures 6.24, 6.27, 6.30 and 6.33 give the magnitude of the Discrete Fourier Transforms of the true signal sequence, optimal signal estimate and signal estimates produced by the four adaptive algorithms. These DFT's are 1024 point DFT's of the last 20 points of the sequences shown in Figures 6.23, 6.26, 6.29 and 6.32. These 20 points, corresponding to one period of the signal, are repeated 51 times, giving a 1020 point sequence, before the 1024 point DFT is computed. Since the last four points of the 1024 point DFT are equal to zero, there is a very slight windowing effect evident in the plots of the DFT's.

Note from Figure 6.22 that for the high SNR condition, the rates of convergence for the LMS Gradient, Jacobi Relaxation and SOR Algorithms are approximately the same. However, for the lower SNR conditions, the SOR Algorithm exhibits a significantly higher rate of convergence than the LMS or Jacobi Relaxation

Algorithms as can be seen in Figures 6.25, 6.28 and 6.31. Thus, it appears taht the SOR Algorithm can achieve significantly better performance than the LMS Gradient or Jacobi Relaxation Algorithms while still requiring only on the order of N operations/sample.

An interesting effect can be noted in the plots of MSE(n) for the Levinson Algorithm in Figures 6.22, 6.25, 6.28 and 6.31. In these simulations, Levinson's Algorithm is used to update the unit-sample response vector $\underline{h}$ at sample times n=31, n=63 and n=95 based on estimates of the autocorrelation matrix and vector which are made at sample times n=0, n=31 and n=63 respectively so that the computations may be spread out over k=2N=32 samples to reduce the average number of operations/sample. Note from the figures that the mean-square error is actually increased after the first time that $\underline{h}$ is updated. This is due to the fact that the estimates of the autocorrelation matrix and vector are very poor at sample time n=0. The MSE is then dramatically decreased after the second time that $\underline{h}$ is updated due to the fact that the estimates $\hat{\phi}_{XX}$ and $\underline{\hat{R}}_{XX}$ have improved with the addition of 32 more sample points. Aside from this effect, the Levinson Algorithm reduces the MSE more rapidly than the LMS Gradient and Jacobi Relaxation Algorithms at the lower signal-to-noise ratios.

The ability of the adaptive estimation algorithms to perform the estimation task can also be judged by considering Figures 6.24, 6.27, 6.30 and 6.33 which show the magnitude of the DFT of the signal estimates. How "good" the estimate is can be judged by comparing the amount of signal component in the esti-

mate to the amount of the other frequency components. As can be seen from the figures, the SOR and Levinson Algorithms produce "better" estimates than do the LMS Gradient and Jacobi Relation Methods. However, the plots of the magnitude of the DFT should not be weighted heavily in the evaluation of the algorithms since they are the results of a single simulation run and therefore have a high variance, especially at the lower SNR conditions.

Thus, it appears from the results of the simulation that the SOR and Levinson Algorithms perform significantly better than the LMS Gradient or Jacobi Relaxation Algorithms for the conditions of the computer simulation.

6.6 Conclusion

The convergence properties of the four adaptive estimation algorithms considered here have been explored through the use of computer simulation. The results of the simulation have shown that the theoretical bounds derived in Chapters II and III give good approximations to the actual MSE(n) produced by the filters determined by the LMS Gradient, Jacobi Relaxation and SOR Algorithms. However, due to the effect of the increase in the variance of the estimates of the statistics with a decrease in SNR, the MSE produced by the adaptive algorithms differs from the theoretical bound by a larger amount at the lower SNR conditions considered.

It is also evident from the results of the simulation that the SOR Algorithm and Levinson's Algorithm can give substantially better performance than the LMS Gradient Algorithm and the Jacobi Relaxation Method when the Variable-Iteration-Step Method has been used to reduce the number of operations to O(N) and the parameter μ in the LMS Gradient Method has been chosen to equalize the variance of the estimates of the statistics. This suggests that the SOR Algorithm and Levinson's Algorithm can be used to perform the adaptive estimation task with substantially better performance than the LMS Gradient Algorithm with only a modest increase in the average number of operations/sample required.

SNR	Algorithm	Parameter	ℓ	k	Operations/Sample
+ 7.0 db	LMS	μ=0.005	16	1	2N+1 = 33
	Jacobi	β=0.14	16	16	6N+1 = 97
	SOR	β=1.0	1	1	6N = 96
	Levinson	-	16	32	6N = 96
- 3.0 db	LMS	μ=0.002	16	1	2N+1 = 33
	Jacobi	β=0.14	16	16	6N+1 = 97
	SOR	β=1.0	1	1	6N = 96
	Levinson	-	16	32	6N = 96
-13.0 db	LMS	μ=0.00025	16	1	2N+1 = 33
	Jacobi	β=0.14	16	16	6N+1 = 97
	SOR	β=1.0	1	1	6N = 96
	Levinson	-	1	1	6N = 96
-23.0 db	LMS	μ=0.000029	16	1	2N+1 = 33
	Jacobi	β=1.14	16	16	6N+1 = 97
	SOR	β=1.0	1	1	6N = 96
	Levinson	-	16	32	6N = 96

Figure 6.21 Parameter Values and Average Number of Operations/Sample for the Four Algorithms as Implemented in the Computer Simulation.

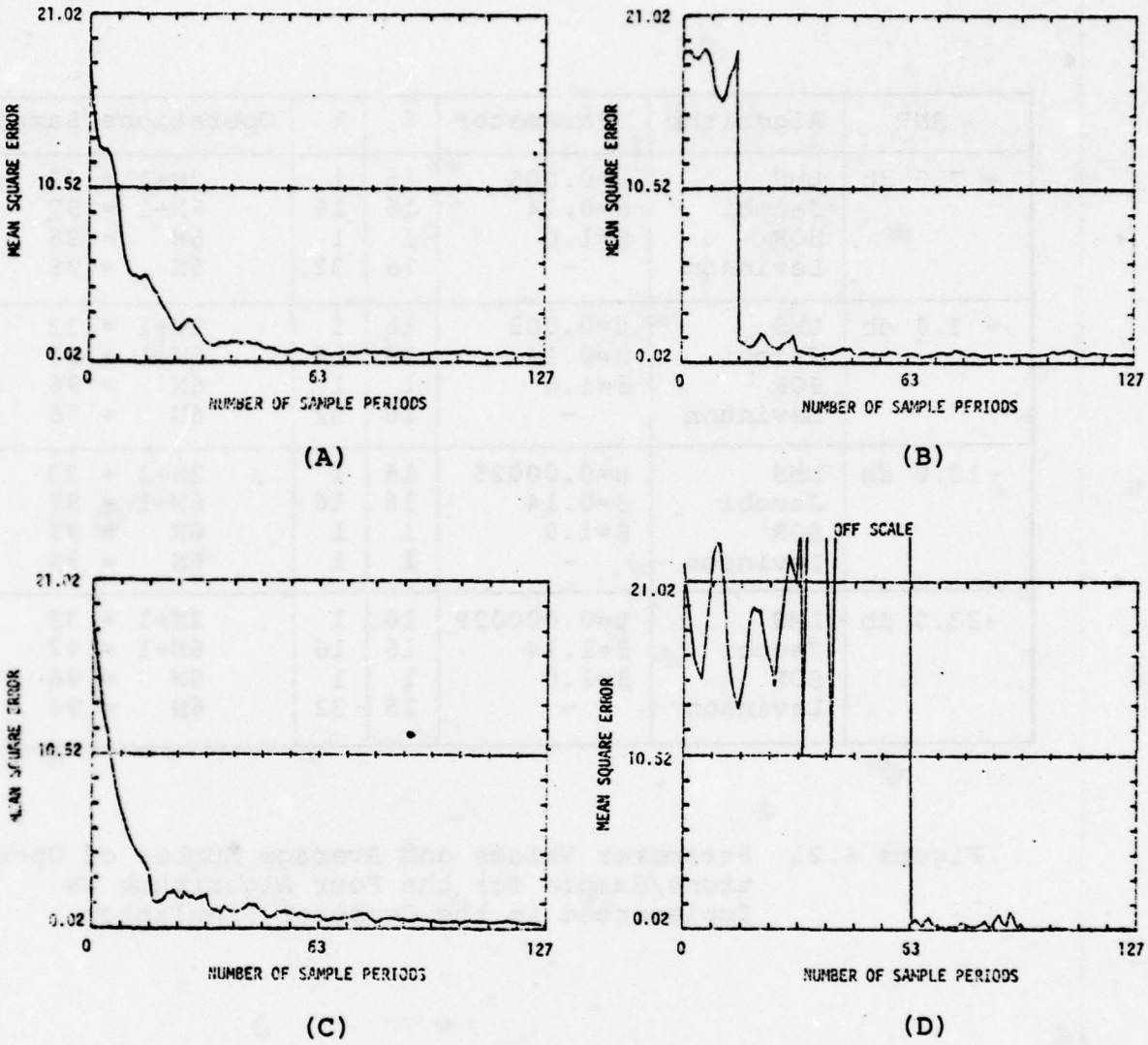


Figure 6.22 Comparison of Actual Performance of the Four Adaptive Algorithms. SNR = +7.0 db

(A) LMS, μ=0.005

(B) Jacobi, β=0.14, ℓ=16, k=16

(C) SOR, β=1.0, ℓ=1, k=1

(D) Levinson, k=32

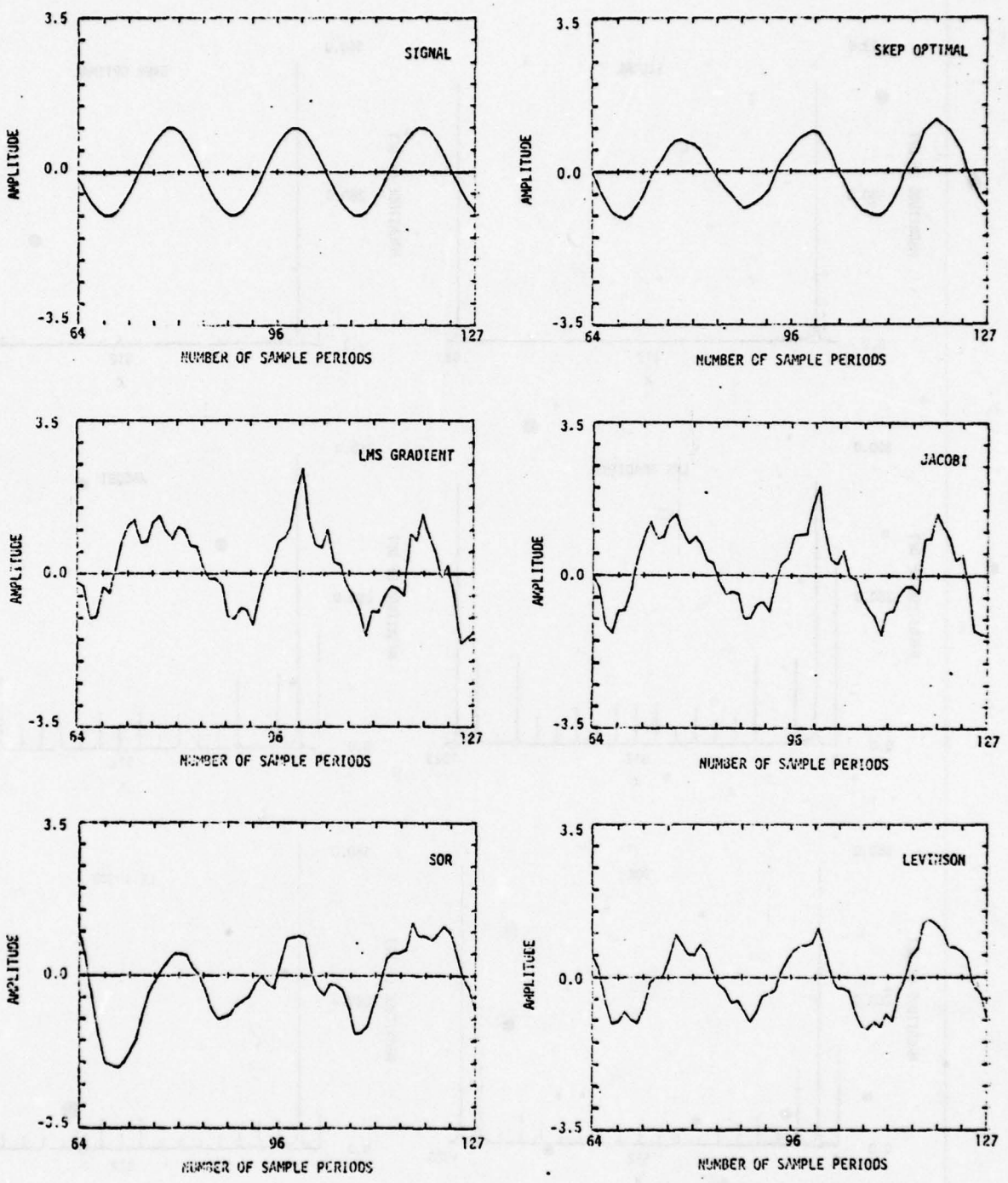


Figure 6.23 Comparison of True Signal, Optimal Signal Estimate (SKEP), and Signal Estimates for The Four Adaptive Agorithms. SNR = +7.0 db

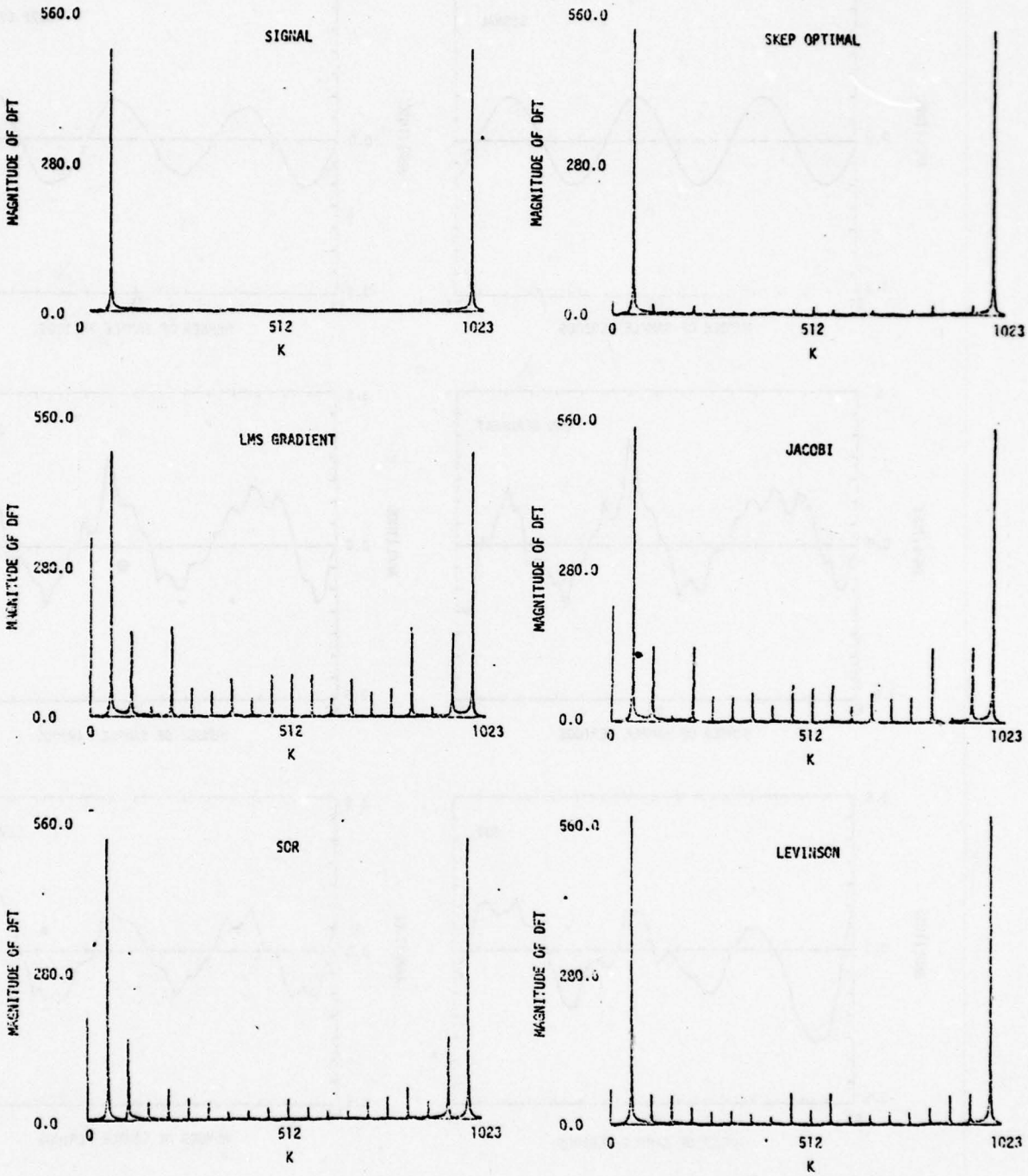


Figure 6.24 Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR = +7.0 db

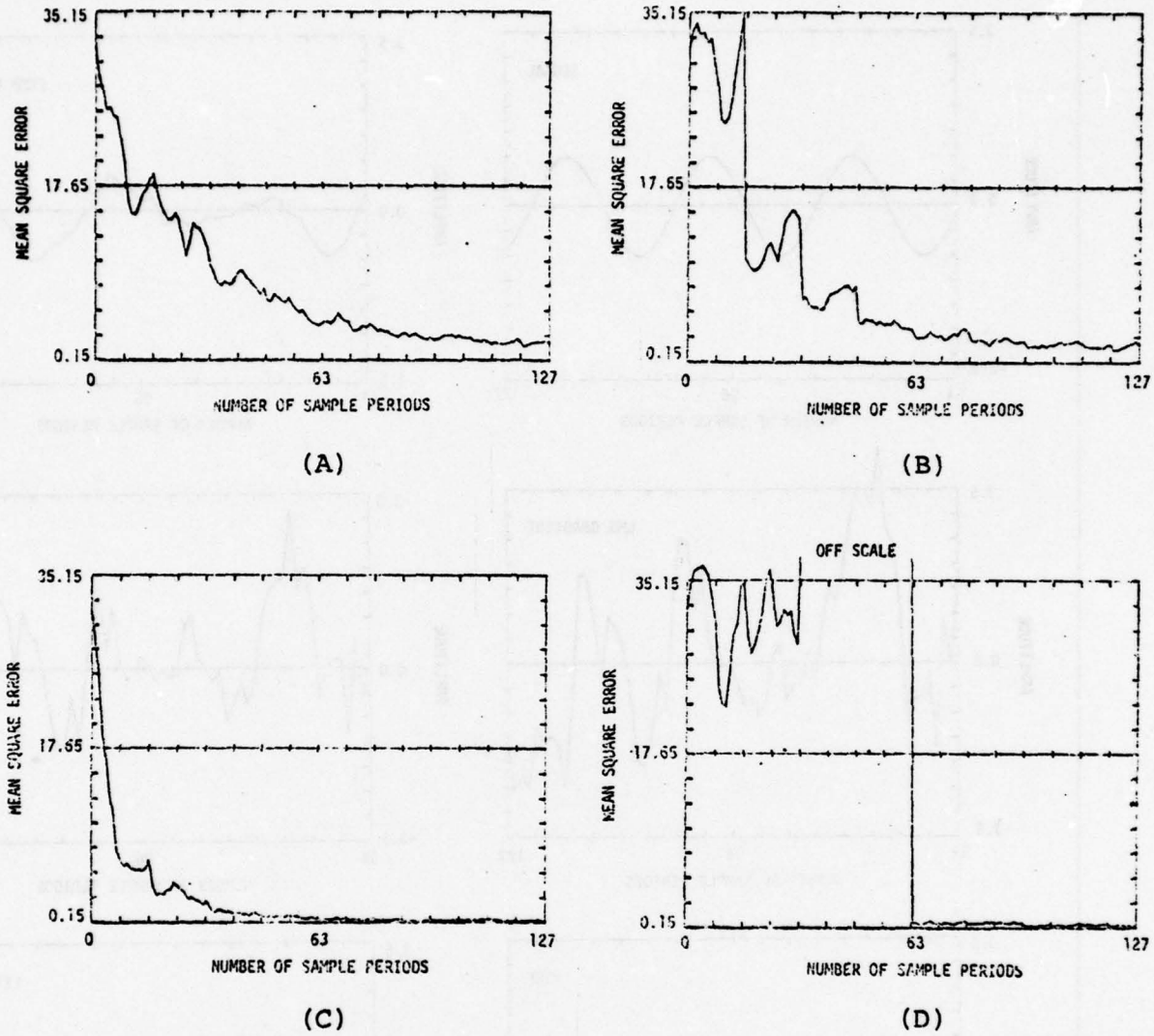


Figure 6.25 Comparison of Actual Performance of the Four Adaptive Algorithms. SNR = -3.0 db

(A) LMS, μ=0.002

(B) Jacobi, β=0.14, ℓ=16, k=16

(C) SOP, β=1.0, ℓ=1, k=1

(D) Levinson, k=32

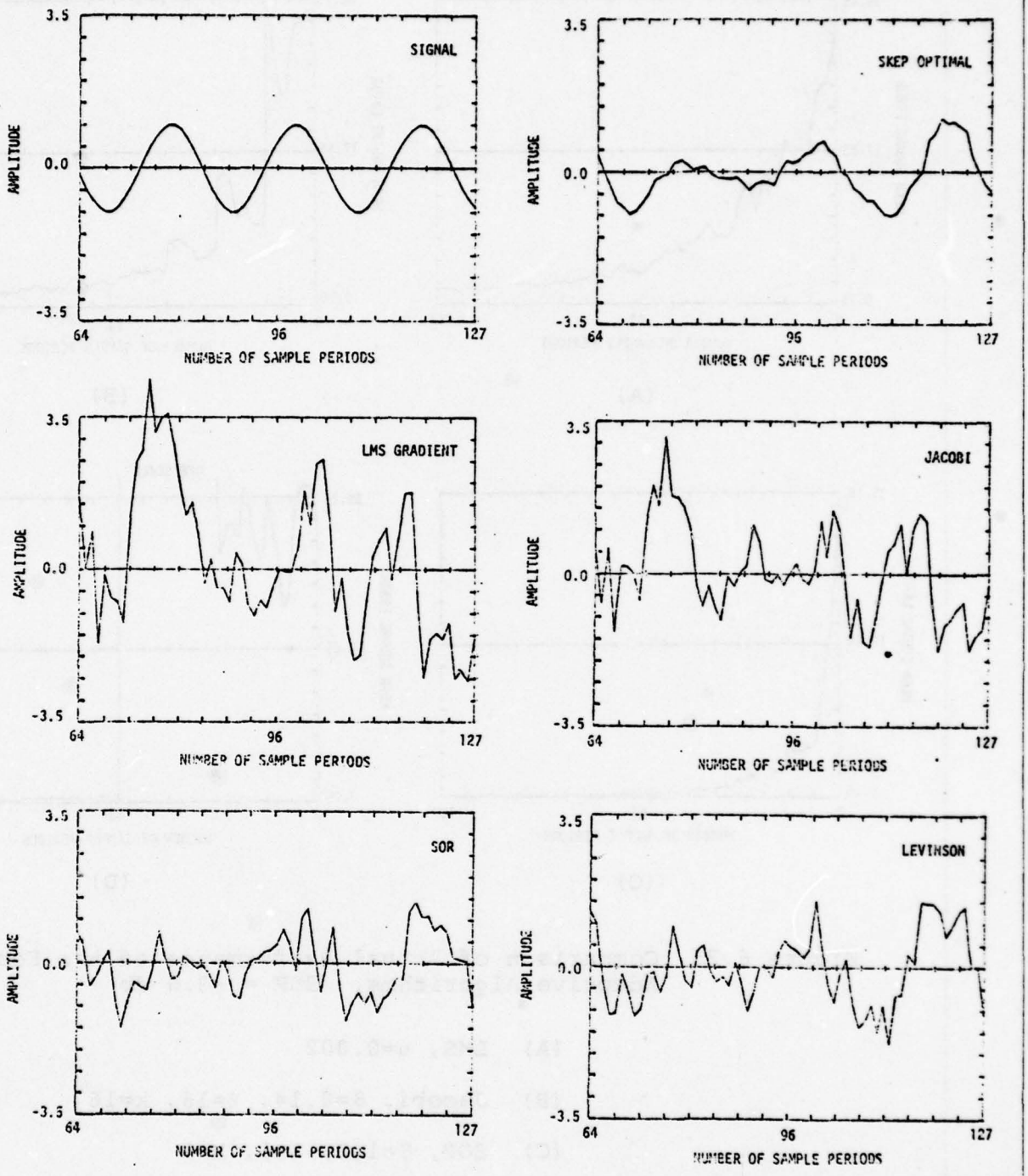


Figure 6.26 Comparison of True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced By the Four Adaptive Algorithms. SNR = -3.0 db

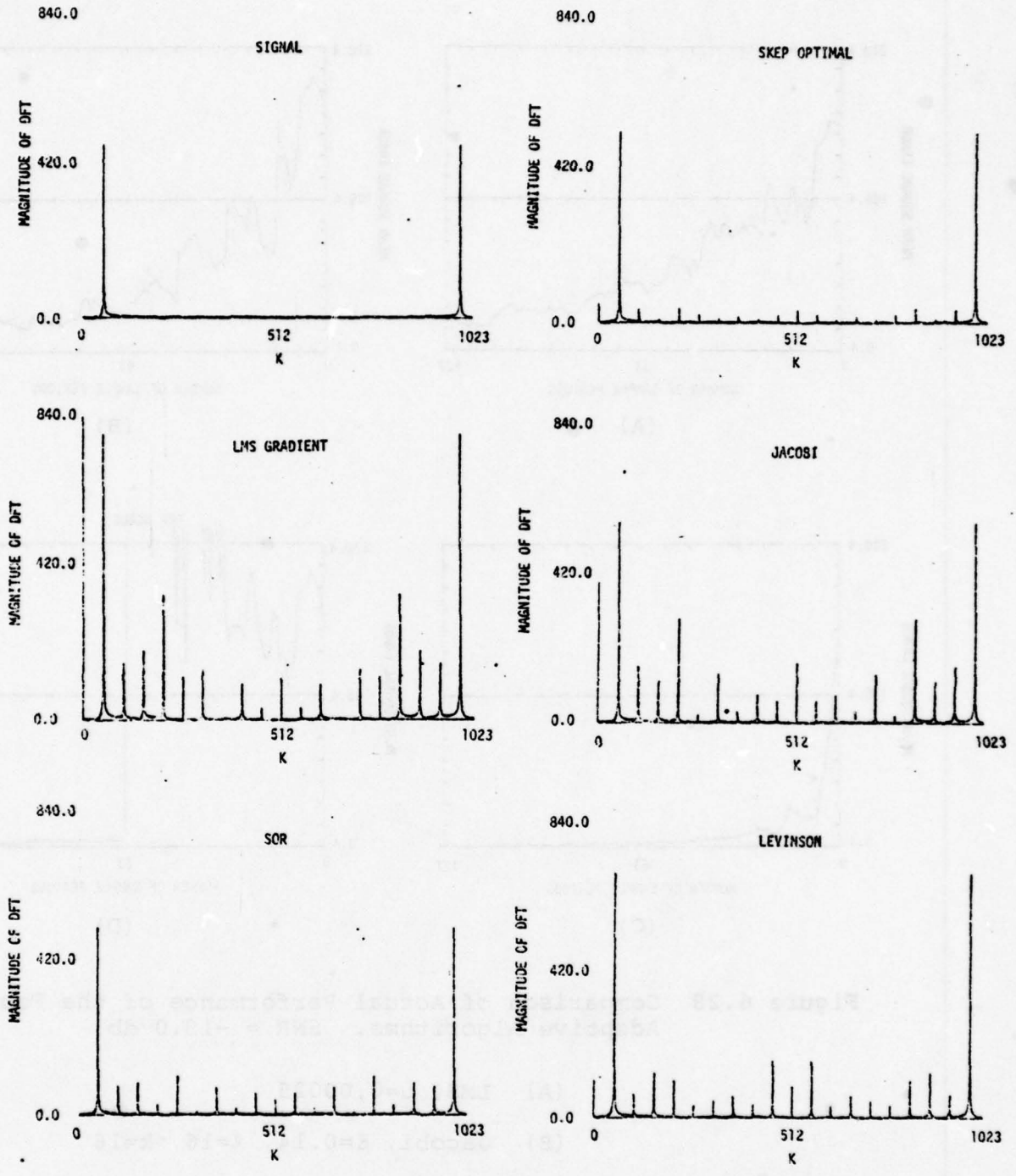


Figure 6.27 Comparison of the DFT's of the True Signal Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR = -3.0 db

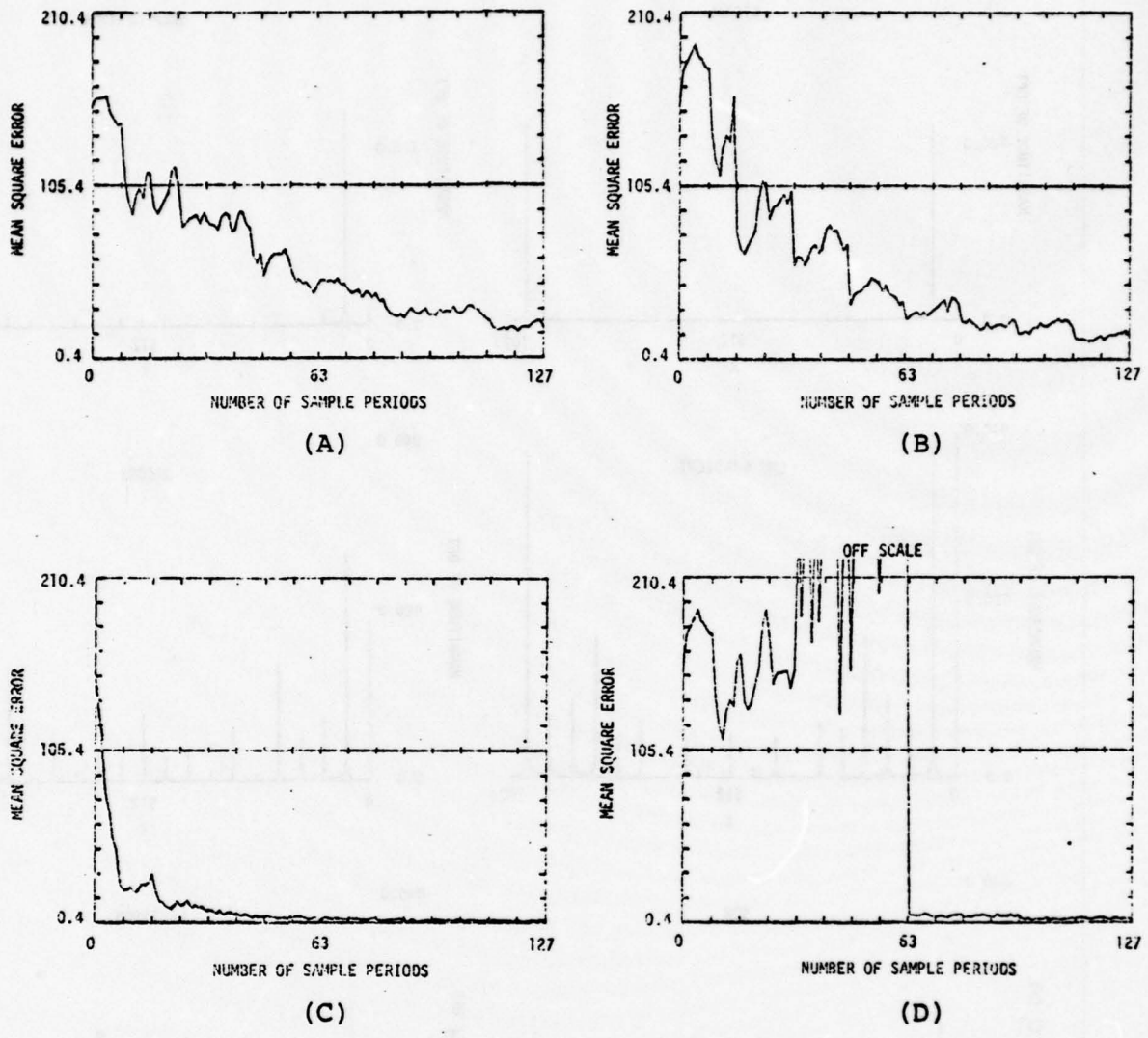


Figure 6.28 Comparison of Actual Performance of the Four Adaptive Algorithms. SNR = -13.0 db

(A) LMS, μ=0.00025

(B) Jacobi, β=0.14, ℓ=16, k=16

(C) SOR, β=1.0, ℓ=1, k=1

(D) Levinson, K=32

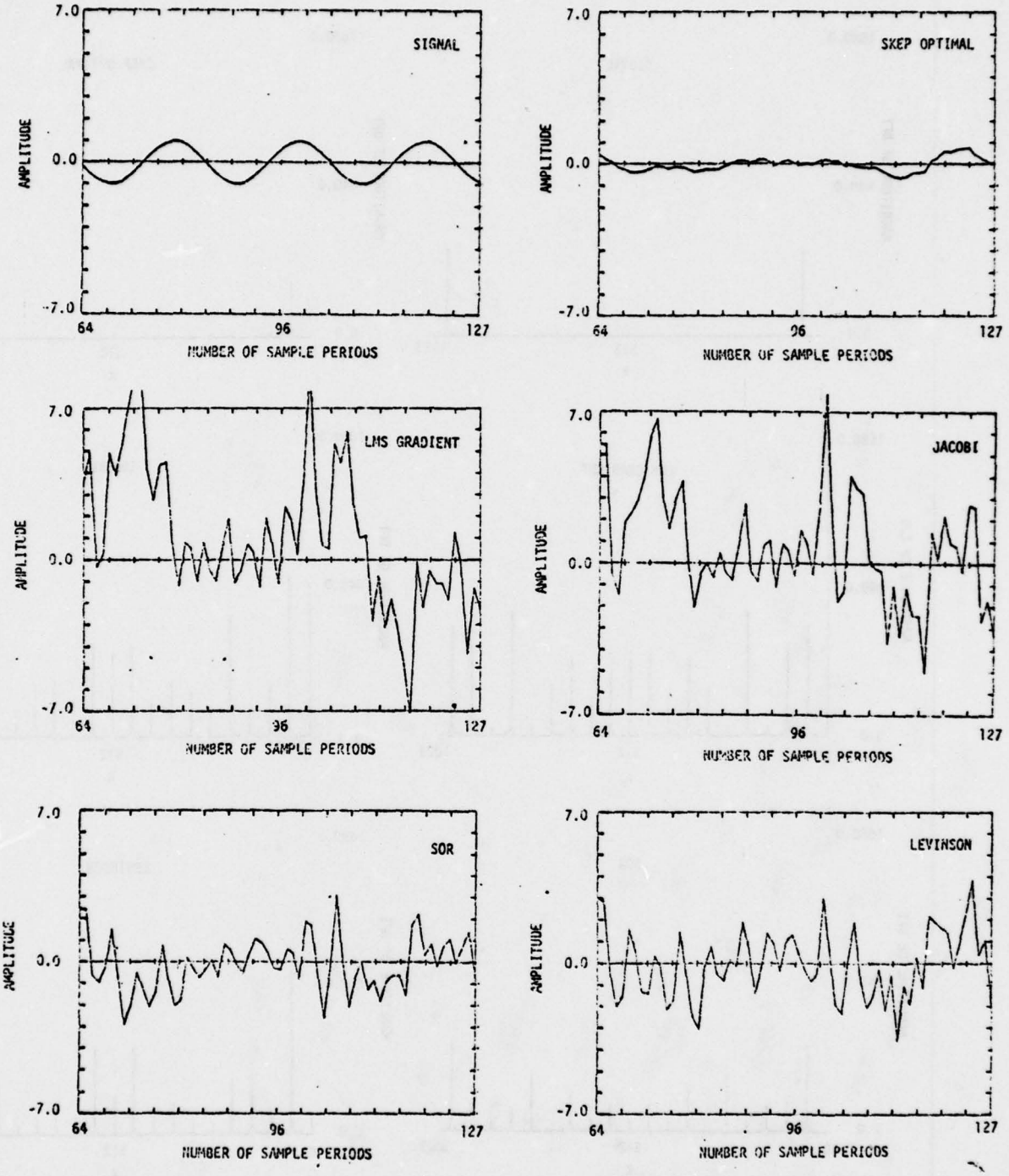


Figure 6.29 Comparison of True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR = -13.0 db

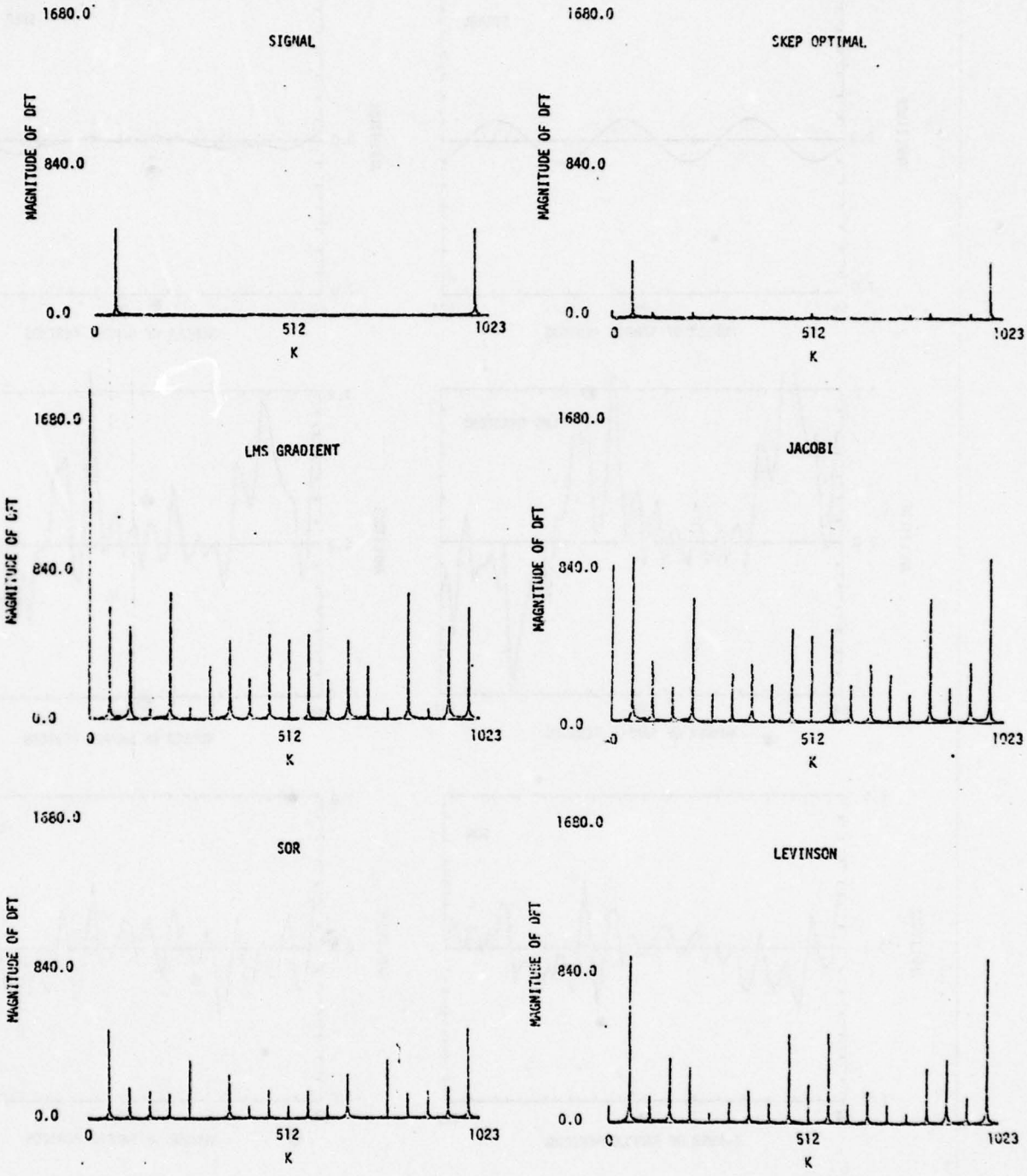


Figure 6.30 Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR = -13.0 db

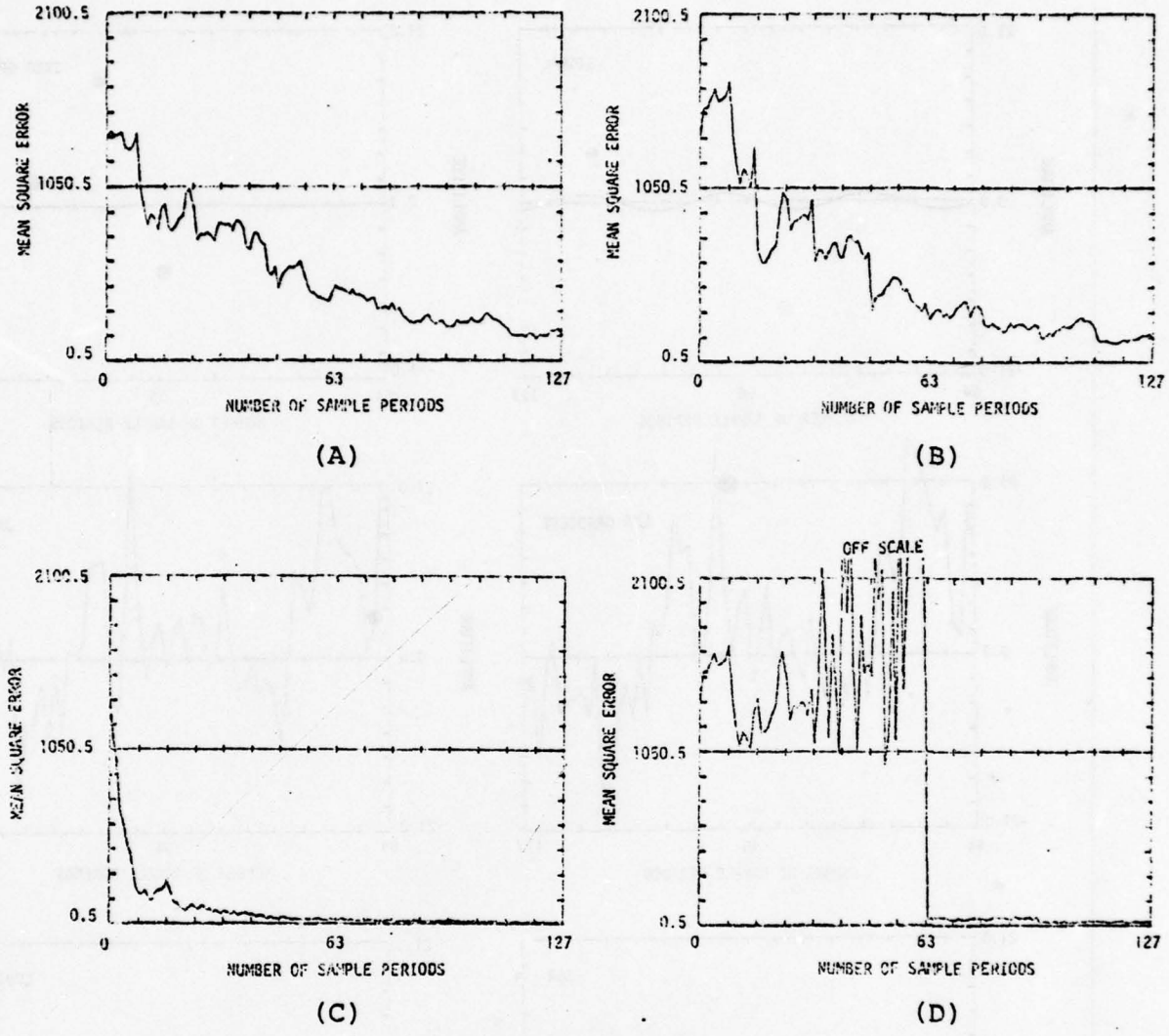


Figure 6.31 Comparison of Actual Performance of the Four Adaptive Algorithms. SNR = -23.0 db

(A) LMS, μ=0.000029

(B) Jacobi, β=0.14, ℓ=16, k=16

(C) SOR, β=1.0, ℓ=1, k=1

(D) Levinson, k=32

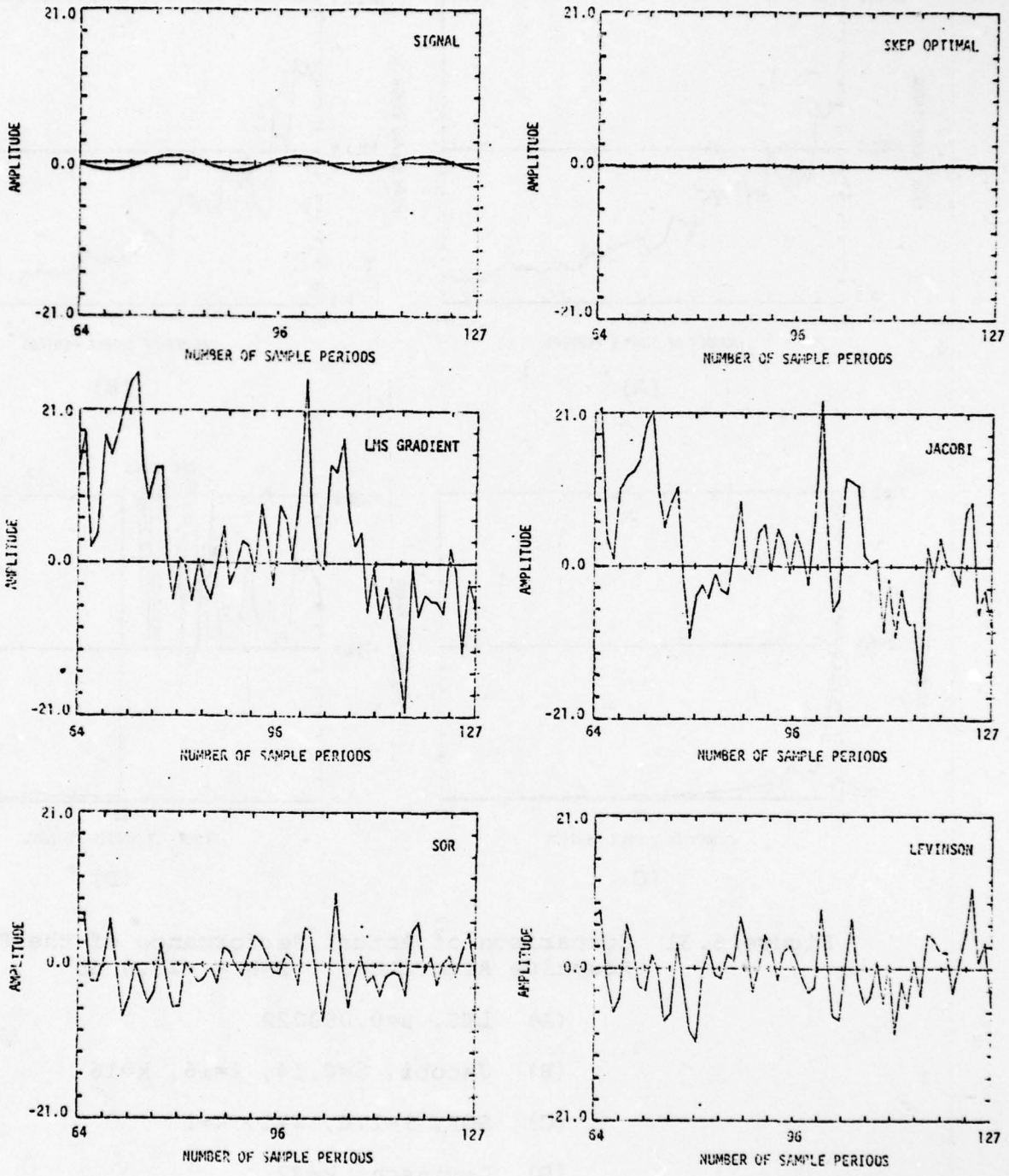


Figure 6.32 Comparison of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR = -23.0 db

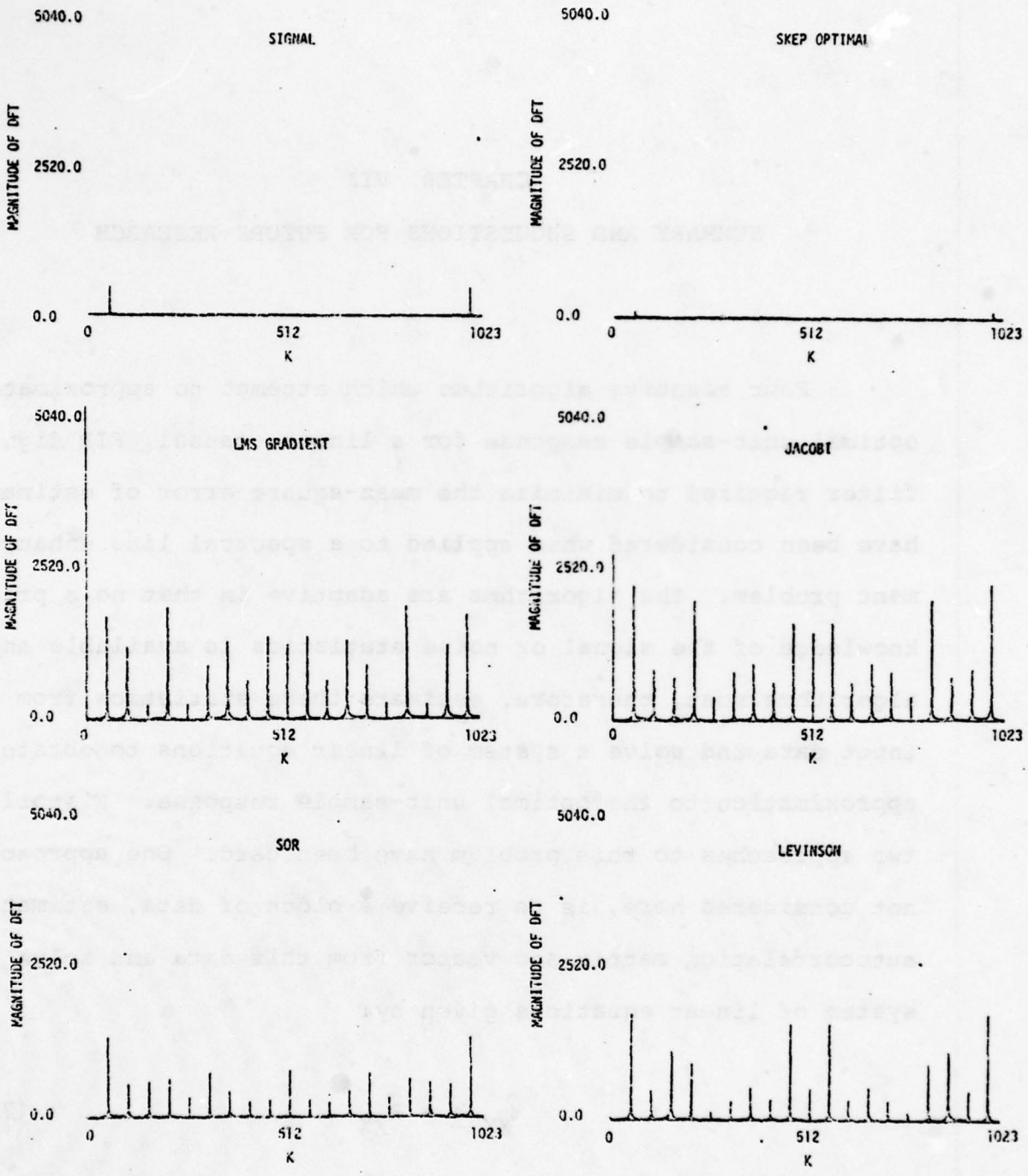


Figure 6.33 Comparison of the DFT's of the True Signal, Optimal Signal Estimate (SKEP) and Signal Estimates Produced by the Four Adaptive Algorithms. SNR = -23.0 db.

CHAPTER VII

SUMMARY AND SUGGESTIONS FOR FUTURE RESEARCH

Four adaptive algorithms which attempt to approximate the optimal unit-sample response for a linear, causal, FIR digital filter required to minimize the mean-square error of estimation have been considered when applied to a spectral line enhancement problem. The algorithms are adaptive in that no a priori knowledge of the signal or noise statistics is available and the algorithms must, therefore, estimate these statistics from the input data and solve a system of linear equations to obtain an approximation to the optimal unit-sample response. Historically, two approaches to this problem have been used. One approach, not considered here, is to receive a block of data, estimate the autocorrelation matrix and vector from this data and solve the system of linear equations given by:

$$\hat{\mathbf{C}}_{XX}\,\underline{h} = \hat{\mathbf{R}}_{XX} \quad . \tag{7.1}$$

The resulting unit-sample response is then used to filter the block of received data. Due to the delay required to implement this procedure, it is usually an off-line approach. The second approach has been to use a real-time adaptive algorithm such

as the LMS Gradient Algorithm. The LMS Gradient Method and three algorithms which lie somewhere between the two historical methods in terms of computational complexity and performance have been explored in this dissertation.

7.1 Summary

The LMS Gradient Method is a widely used adaptive algorithm for the solution of the system in Equation (7.1). Chapter II outlines the LMS Gradient Algorithm, which is based on the Method of Steepest Descent for the minimization of a function of several variables. An instantaneous estimate of the gradient of the MSE is used as an estimate of the statistics of the input process. A parameter μ present in the LMS Gradient Algorithm must be chosen to be smaller than a certain bound, which depends upon the statistics of the input process, in order for the algorithm to converge. A theoretical bound on the rate of convergence of the LMS Gradient Algorithm was developed in Chapter II by deriving a bound on the MSE as a function of the time index n. This bound revealed that the rate of convergence of the LMS Gradient Algorithm was dependent on the parameter μ. Another view of the LMS Gradient Algorithm was given in Chapter II as a result of the derivation, at least formally, of another algorithm based on an attempt to approximately satisfy the orthogonality principle using direct estimates of the expected value terms involved. The resulting algorithm was very similar to the LMS Gradient Method except that the constant parameter μ

was replaced by a data-dependent term. This derivation gives additional insight into the operation of the LMS Gradient Algorithm.

As an alternative to the LMS Gradient Algorithm, a family of Adaptive-Fixed-Point Iteration Methods was proposed. In this approach, a direct estimate of the autocorrelation matrix and vector is made and continually updated so that the linear system of equations (7.1) results. This system is solved by applying a single step of a fixed-point iterative method at each sample time n. The estimates of the autocorrelation matrix and vector are arbitrary with the restriction that they be unbiased and have a reasonably small variance. However, a moving average estimate was suggested in Section 3.1 for the estimation of these statistics.

Two particular fixed-point iteration algorithms were considered in detail in Chapter III. The Jacobi Relaxation and Successive Over-Relaxation Algorithms were shown to be convergent in the mean provided that certain restrictions were placed on the value of the relaxation parameter β present in both algorithms. These restrictions are independent of the statistics of the input data. Theoretical bounds on the MSE produced by the filters determined by each of these two algorithms were derived as a function of the time index n. These bounds reveal that the rates of convergence of the Jacobi Relaxation and SOR Algorithms depend on the value of the relaxation parameter β used in each algorithm.

The Jacobi Relaxation and SOR Algorithms require on the order of N^2 operations/sample if implemented directly as opposed to the O(N) operations/sample required by the LMS Gradient Algorithm. Since such computational complexity may be prohibitive in real-time applications, methods for the reduction of the average number of operations/sample were considered in Chapter III. In the Variable-Iteration-Step Method, a fixed-point iteration algorithm is used to update ℓ of the elements of the unit-sample response vector every k sample periods. Thus, if $\ell/k < N$, the average number of operations can be reduced to on the order of:

$$\ell/k \ (N) \ . \qquad (7.2)$$

However, such a decrease in computational complexity brings about a corresponding increase in the number of sample periods required for convergence. This approach is extremely attractive in that it allows the time available between samples to be fully utilized for computation since the number of operations/sample can be adjusted. In addition to the Variable-Iteration-Step Method, a procedure for carrying out the Jacobi Relaxation Algorithm using the Fast Fourier Transform was proposed for cases in which the autocorrelation matrix estimate exhibits a Topelitz structure as does the estimate proposed in Section 3.1. This method can lead to substantial savings in computation time for larger values on N. Thus, the Adaptive Fixed-Point Iteration Algorithms proposed in Chapter III offer a very flexible approach to the solution of the adaptive estimation problem.

When the estimate of the autocorrelation matrix exhibits a Topelitz structure, Levinson's Algorithm can be used to solve the system of equations given by (7.1) in a very efficient manner. Levinson's Algorithm was outlined in Chapter IV and is a non-iterative algorithm. Levinson's Algorithm requires on the order of $2N^2$ operations for its implementation. Therefore, to reduce the average number of operations/sample it was proposed that the estimates $\hat{\Phi}_{XX}$ and $\hat{\underline{R}}_{XX}$ be updated as each data sample is received and that Levinson's Algorithm be used to solve the system (7.1) at intervals of k sample periods. In this case, the average number of operations/sample required for the solution of (7.1) can be reduced to on the order of:

$$\frac{2N^2}{k} \quad . \tag{7.3}$$

Thus, Levinson's Algorithm is an attractive approach in the special case where the estimated autocorrelation matrix has a Topelitz structure.

The four adaptive algorithms considered here are summarized in Figure 7.1. As can be seen from the figure, the LMS Gradient Algorithm has the advantage of being very simple and requiring only O(N) operations/sample. However, the other three algorithms have the advantage of allowing a variable number of operations/sample which, while greater than that required by the LMS Gradient Method, can still be reduced to O(N). Levinson's Algorithm, which is the most complex of the four, has the advantage of being non-iterative in nature. That is, the "exact"

	ESTIMATION OF STATISTICS	ADAPTATION OF UNIT SAMPLE RESPONSE	SIGNAL ESTIMATE	NUMBER OF OPERATIONS
LMS GRADIENT	$\hat{\nabla}\left[e^2(n)\,\middle|\,\underline{h}_n\right] = 2\left(\hat{S}(n) - X(n)\right)\underline{X}(n-n_1)$	$\underline{h}_{n+1} = \underline{h}_n - 2\mu\,\hat{\nabla}\left[e^2(n)\,\middle|\,\underline{h}_n\right]$	$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n$	O(N)
JACOBI RELAXATION	$\hat{R}_{XX,n+1}(\cdot)_M = \hat{R}_{XX,n}(\cdot)_M + \frac{1}{M}\left[X(n+1)X(n+1-\lvert\cdot\rvert) - X(n-M+1)X(n-M+1-\lvert\cdot\rvert)\right]$. $\hat{\phi}_{ij,n} = \hat{R}_{XX,n}(i-j)_M$ $\hat{R}_{i,n} = \hat{R}_{XX,n}(n_1+i-1)_M$	$h'_{i,m+1} = h'_{i,m} - \frac{\beta}{\hat{\phi}_{ii,mk}}\sum_{j=1}^{N}\hat{\phi}_{ij,n}\,h'_{j,mk} + \frac{\beta}{\hat{\phi}_{ii,n}}\hat{R}_{i,mk}$ $m = 0, 1, 2, \ldots$ $i = \mathrm{MOD}_N(\ell m), \mathrm{MOD}_N(\ell m+1), \ldots \mathrm{MOD}_N\left(\ell(m+1)-1\right)$ $h_{i,n} = h'_{i,\mathrm{INT}(n/k)}$	$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n$	$O\left(\frac{\ell \cdot N}{k}\right)$
SOR	$\hat{R}_{XX,n+1}(\cdot)_M = \hat{R}_{XX,n}(\cdot)_M + \frac{1}{M}\left[X(n+1)X(n+1-\lvert\cdot\rvert) - X(n-M+1)X(n-M+1-\lvert\cdot\rvert)\right]$. $\hat{\phi}_{ij,n} = \hat{R}_{XX,n}(i-j)_M$ $\hat{R}_{i,n} = \hat{R}_{XX,n}(n_1+i-1)_M$	$h'_{i,m+1} = (1-\beta)h'_{i,m} - \frac{\beta}{\hat{\phi}_{ii,mk}}\left[\sum_{j=1}^{i-1} h'_{i,m+1}\hat{\phi}_{ij,mk} + \sum_{j=i+1}^{N} h'_{i,n}\hat{\phi}_{ij,m} - \hat{R}_{i,mk}\right]$ $m = 0, 1, 2, \ldots$ $i = \mathrm{MOD}_N(\ell m), \mathrm{MOD}_N(\ell m+1), \ldots \mathrm{MOD}_N\left(\ell(m+1)-1\right)$ $h_{i,n} = h'_{i,\mathrm{INT}(n/k)}$	$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n$	$O\left(\frac{\ell \cdot N}{k}\right)$
LEVINSON	$\hat{R}_{XX,n+1}(\cdot)_M = \hat{R}_{XX,n}(\cdot)_M + \frac{1}{M}\left[X(n+1)X(n+1-\lvert\cdot\rvert) - X(n-M+1)X(n-M+1-\lvert\cdot\rvert)\right]$. $R'_{XX}(\cdot) = \hat{R}_{XX,mk}(\cdot)_M \quad m = 0, 1, 2, \ldots$	$\varepsilon_i = R'_{XX}(n_1+i)/R'_{XX}(0) - \underline{h}'^T_{i,n}\,\underline{\tilde{e}}_i$; $\eta_i = -R'_{XX}(i+1)/R'_{XX}(0) - \underline{h}'^T_{i,m}\,\underline{\tilde{g}}_i$ $\underline{h}'_{i+1,m} = \begin{bmatrix}\underline{h}_{i,m} + \theta_i/\lambda_i\,\underline{\tilde{g}}_i \\ \theta_i/\lambda_i\end{bmatrix}$; $\underline{g}_{i+1} = \begin{bmatrix}\eta_i/\lambda_i \\ \underline{\tilde{g}}_i + \eta_i/\lambda_i\,\underline{g}_i\end{bmatrix}$; $\lambda_{i+1} = \lambda_i - \eta_i^2/\lambda_i$ $i = 1, 2, 3, \ldots, N-1$ $\underline{h}_n = \underline{h}'_{N,\mathrm{INT}(n/k)}$	$\hat{S}(n) = \underline{X}^T(n-n_1)\underline{h}_n$	$O\left(\frac{2N^2}{k}\right)$

Figure 7.1 Summary of Equations Required for the Implementation of the Four Adaptive Algorithms.

solution of (7.1) is found in a fixed number of steps. However, the usefulness of Levinson's Algorithm is limited due to the constraint that the estimated autocorrelation matrix have a Toeplitz structure. In addition, the Levinson Algorithm is sequential in nature and not well suited to parallel processing as are the Jacobi Relaxation and SOR Algorithms.

The performance of the LMS Gradient Algorithm, the Jacobi Relaxation Method and the SOR Algorithm was studied in Chapter V using the theoretical bounds derived in Chapters II and III. The dependence of the rate of convergence of the LMS Gradient Algorithm on the parameter μ and of the Jacobi Relaxation and SOR Algorithms on the relaxation parameter β was demonstrated. A reduction in the value of μ for the LMS Gradient Algorithm or in the value of β for the Jacobi Relaxation Algorithm lead to a corresponding increase in the number of sample periods required for convergence of the respective algorithm. However, such a straightforward relationship between the value of β and the convergence rate of the SOR algorithm does not exist. In addition to the effect of the value of β, it was demonstrated that a decrease in the value of the ratio ℓ/k led to a corresponding increase in the number of sample periods required for the convergence of the Jacobi Relaxation and SOR Algorithms when the Variable-Iteration-Step Method was used. A comparison of the rates of convergence of the three iterative algorithms was made using the theoretical bounds with the parameters ℓ and k for the Variable-Iteration-Step Method chosen so that approximately N operations/sample were required by the two fixed-point iteration

algorithms. In addition, the value of the parameter μ was chosen according to Equation (5.27) in an attempt to equalize the variance in the estimates of the statistics of the input process. The results of this comparison indicated that for a high SNR condition of +7.0 db, the three algorithms have essentially the same rate of convergence. However, at the lower SNR conditions of -3.0 db, -13.0 db and -23.0 db, the SOR Algorithm exhibited a substantially higher rate of convergence than did the LMS Gradient Algorithm and the Jacobi Relaxation Method. The LMS Gradient and Jacobi Relaxation Algorithms converge at approximately the same rate with the rate of convergence of the Jacobi Relaxation Algorithm being slightly higher. Thus, the results of the comparisons of the theoretical bounds indicate that the SOR Algorithm can give substantially better performance than the LMS Gradient or Jacobi Relaxation Method while still requiring only on the order of N operations/sample.

The performance of the four adaptive algorithms considered here was investigated through extensive computer simulation. The results of this simulation, given in Chapter VI, indicate that the theoretical bounds on the MSE as a function of the time index n derived in Chapters II and III give good approximations to the actual MSE(n) produced by the filters determined by the LMS Gradient, Jacobi Relaxation and SOR Algorithms. However, the difference between the theoretical bound and the actual performance of the algorithms increase as the SNR decreases. This is due to the fact that the variance of the estimates of the input statistics increases as the SNR decreases in this case.

Due to the "mis-adjustment error" the actual MSE(n) for the LMS Gradient Algorithm does not converge to the value predicted by the theoretical bound. Similarly, due to the effect of the rate at which the variance of the estimate of the statistics used by the Jacobi Relaxation and SOR Algorithms in the simulation decreases, the actual MSE(n) produced by the filter determined by these algorithms appears to converge to a value which is slightly greater than that predicted by the theoretical bound.

A comparison of the results of the computer simulations of the four adaptive algorithms was presented in Chapter VI. For this comparison, the number of operations/sample was chosen to be on the order of the length of the filter for the Jacobi Relaxation, SOR and Levinson Algorithms. In addition the value of the parameter μ for the LMS Gradient Algorithm was chosen according to Equation (5.27) in an attempt to equalize the variance of the estimates of the input statistics across the four algorithms. The results of this comparison indicate that the SOR Algorithm and Levinson's Algorithm offer substantially better performance than the LMS Gradient and Jacobi Relaxation Methods while still requiring on the order of N operations/sample.

From the results of the computer simulation and from a consideration of the theoretical bounds derived in Chapters II and III it is evident that Adaptive Fixed-Point Iteration using the SOR Algorithm offers an extremely attractive solution to the problem of adaptive estimation for Spectral Line Enhancement. The SOR Algorithm using the Variable-Iteration-Step Method gives substantially better performance than the LMS Gradient Method or

the Jacobi Relaxation Algorithm while still requiring on the order of N operations/sample. In addition, the variable number of operations/sample allowed by the Variable-Iteration-Step Method enables the processor to make full use of the time available between samples. Also, the Adaptive Fixed-Point Iteration Method is well suited to parallel processing.

Levinson's Algorithm also performed well in the simulation study. However, the restriction that the estimated autocorrelation matrix have a Topelitz structure makes it less attractive than the SOR Method as a general algorithm.

Thus, of the four algorithms considered, Adaptive Fixed-Point Iteration using the SOR Algorithm and the Variable Iteration Step Method appears to offer the most attractive solution to the problem of adaptive linear estimation for Spectral Line Enhancement.

7.2 Suggestions for Future Research

Methods for the solution of the adaptive linear estimation problem have been proposed in this thesis which, in light of the results given here, appear to yield better performance than the LMS Gradient Algorithm without a prohibitive increase in the number of operations/sample required for implementation. The performance of the LMS Gradient Algorithm, the Jacobi Relaxation Method, the SOR Algorithm and Levinson's Algorithm has been studied for a specific set of input conditions. That is, it was assumed here that the input to the Spectral Line

Enhancer is in the form of a single sinusoidal signal of fixed amplitude and frequency, but of random phase, added to an autoregressive noise sequence. While the performance of the LMS Gradient Algorithm for other input conditions has been reported on in the literature, an investigation of the performance of the Adaptive Fixed-Point Iteration Algorithms and Levinson's Algorithm under various conditions of signal and noise is needed. Of obvious interest are the problems of a signal composed of multiple sinusoids, signals of time-varying frequency and/or amplitude and signals in time-varying noise environments.

In general, it is suggested that future work be directed towards a better understanding of the performance of the algorithms proposed here and include evaluations of the performance of these methods as compared to that of more established procedures such as the LMS Gradient Algorithm for a variety of signal and noise conditions.

APPENDIX A

PROOF OF THEOREM 2.1

Theorem 2.1

A necessary and sufficient condition for the convergence of the fixed-point iteration given by:

$$\underline{Y}_{n+1} = B\,\underline{Y}_n + \underline{d} \qquad \text{(A.1)}$$

is that the magnitude of the eigenvalues of the iteration matrix B be less than 1.

Proof:

By Jordan's Theorem, there exists a non-singular matrix M and a matrix J of Jordan Form such that:

$$B = M^{-1}\,J\,M \quad . \qquad \text{(A.2)}$$

Thus:

$$M\,\underline{Y}_{n+1} = J\,M\,\underline{Y}_n + M\,\underline{d} \quad . \qquad \text{(A.3)}$$

Since M is non-singular, the mapping:

$$\underline{Z}_n = M\,\underline{Y}_n \tag{A.4}$$

is one-to-one and continuous. Thus, a necessary and sufficient condition for the convergence of (A.1) is that:

$$\underline{Z}_{n+1} = J\,\underline{Z}_n + M\,\underline{d} \tag{A.5}$$

converges. Consider partitioning (A.5) as follows:

$$\begin{bmatrix} \underline{Z}_1 \\ \underline{Z}_2 \\ \vdots \\ \underline{Z}_k \end{bmatrix}_{n+1} = \begin{bmatrix} J_1 & 0 & 0 & \cdots & 0 \\ 0 & J_2 & \cdots & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & & J_k \end{bmatrix} \begin{bmatrix} \underline{Z}_1 \\ \underline{Z}_2 \\ \vdots \\ \underline{Z}_k \end{bmatrix}_n + M\,\underline{d} \tag{A.6}$$

where:

$$J_i = \begin{bmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ 0 & 0 & \lambda_i & \cdots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & 0 & \cdots & & \lambda_i \end{bmatrix}$$

λ_i = eigenvalue of the matrix B.

Thus, (A.6) can be expanded into K independent equations:

$$\underline{Z}_{i,n+1} = J_i \, \underline{Z}_{i,n} + (M \, \underline{d})_i \qquad (A.7)$$

where:

$(M \, \underline{d})_i$ = i^{th} partition of the vector $(M \, \underline{d})$.

Therefore, a necessary and sufficient condition for the convergence of (A.5) is that (A.7) converge for all $1 \leq i \leq K$. Consider the last equation of the system (A.7) given by:

$$z_{\ell,n+1} = \lambda i \; z_{\ell,n} + (M \, \underline{d})_\ell \qquad (A.8)$$

which is independent of the other $(\ell-1)$ equations. Equation (A.8) can be rewritten as:

$$z_{\ell,n+1} = \lambda i^{n+1} \; z_{\ell,0} + \sum_{j=0}^{n} \lambda i^j \; (M \, \underline{d})_\ell$$

or:

$$z_{\ell,n+1} = \lambda i^{n+1} \; z_{\ell,0} + \frac{1+\lambda_i^n}{1+\lambda_i} \; (M \, \underline{d})_\ell \; . \qquad (A.9)$$

Obviously, (A.9) converges if $|\lambda_i| < 1$. Thus, it is necessary that $|\lambda_i| < 1$ for (A.8) to converge. Now consider the next to the last equation in (A.7) given by:

$$z_{\ell-1,n+1} = \lambda_i \; z_{\ell-1,n} + z_{\ell,n} + (M \, \underline{d})_{\ell-1}$$

which can be rewritten as:

$$z_{\ell-1,n+1} = \lambda_i^{n+1} z_{\ell-1,0} + \frac{(1+\lambda_i^n)}{1+\lambda_i} [{}_{\ell,n} + (M\underline{d})_{\ell-1}]. \qquad (A.10)$$

It is obvious that a sufficient condition for the convergence of (A.10) is that (A.9) converge and $|\lambda_i| < 1$. It can be seen that a sufficient condition for the convergence of $z_{\ell-j,n}$ is that $z_{\ell-j-1,n}$ converge and that $|\lambda_i| < 1$.

Thus, a necessary and sufficient condition for the convergence of (A.7) is that $|\lambda_i| < 1$. Therefore, it is necessary and sufficient that $|\lambda_i| < 1$ for all $1 \leq i \leq k$ to insure the convergence of (A.5) and, thus, of (A.2).

APPENDIX B

APPROXIMATION OF MSE(n) FOR THE LMS GRADIENT METHOD

Equation (2.31) gives an approximation to the mean-square error produced by the filter determined by the LMS Gradient Algorithm. This appendix contains the development of this equation. The true MSE produced by the LMS Gradient Algorithm can be written as:

$$MSE(n)=E[S^2(n)]-2E[S(n)\underline{X}^T(n-n_1)\underline{h}_n]+E[\underline{h}_n^T\underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n]. \quad (B.1)$$

Consider the scalar term:

$$E[\underline{h}_n^T\ \underline{X}(n-n_1)\underline{X}^T(n-n_1)\underline{h}_n] \quad (B.2)$$

which can be expanded as:

$$E\ \sum_{i=1}^{N}\sum_{j=1}^{N} h_i\ X_i\ X_j\ h_j$$

or:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} E[h_i\ X_i\ X_j\ h_j]\ . \quad (B.3)$$

Now, consider the term inside the summation given by:

$$E[h_i\ X_i\ X_j\ h_j] \quad . \tag{B.4}$$

Assume that h_i, h_j, X_i and X_j are jointly Gaussian. Then, (B.4) can be rewritten as:

$$E[h_i\ X_i\ h_j\ X_j] = E[h_i\ X_i]E[h_j\ X_j] + E[h_i\ X_j]E[X_i\ h_j]$$

$$+ E[h_i\ h_j]E[X_i\ X_j] \quad . \tag{B.5}$$

Now, Widrow [7] has shown that for small μ and low input SNR, $\underline{h}$ and $\underline{X}$ exhibit little statistical correlation. Thus, (B.5) can be rewritten as:

$$E[h_i\ X_i\ X_j\ h_j] = E[h_i]E[X_i]E[X_j]E[h_j] + E[h_i]E[X_j]E[X_i]E[h_j]$$

$$+ E[h_i\ h_j]E[X_i\ X_j] \quad . \tag{B.6}$$

Consider the covariance of h_i and h_j given by:

$$COV[h_i,h_j] = E[(h_i-\bar{h}_i)(h_j-\bar{h}_j)] = E[h_i\ h_j]-\bar{h}_i\bar{h}_j. \tag{B.7}$$

Now, Widrow [7] has shown that for uncorrelated input data sequences:

$$COV[h_i,h_j] = \begin{cases} \mu \cdot E_{min} & i=j \\ 0 & \text{otherwise} \end{cases} \quad . \tag{B.8}$$

where:

E_{min} = MSE produced by the optimal filter.

Thus, from (B.7) and (B.8) it can be seen that if $\mu \cdot E_{min}$ is "small", (B.6) can be rewritten as:

$$E[h_i\ X_i\ X_j\ h_j] = E[h_i]E[X_i]E[X_j]E[h_j] + E[h_i]E[X_j]E[X_i]E[h_j]$$

$$+ E[h_i]E[h_j]E[X_i\ X_j] \quad . \qquad (B.6)$$

Therefore, under the assumptions of uncorrelated input data, and small μ, (B.2) can be rewritten as:

$$E[\underline{h}_n^T]E[\underline{X}(n-n_1)\underline{X}^T(n-n_1)]E[\underline{h}_n] \quad . \qquad (B.7)$$

Now, consider the term:

$$E[S(n)\underline{X}^T(n-n_1)\underline{h}] = E[S(n) \sum_{i=1}^{N} X_i\ h_i]$$

or:

$$E[S(n)\underline{X}^T(n-n_1)\underline{h}] = \sum_{i=1}^{N} E[S(n)X_i\ h_i] \quad . \qquad (B.8)$$

Expanding $S(n)\ X_i\ h_i$ into a Taylor series about $(E[S(n)X_i], E[h_i])$ and retaining up to second order terms yields:

$$E[S(n)X_i\ h_i] \approx E[S(n)X_i]E[h_i] + 1/2\ E[h_i - E[h_i]]$$

$$+ 1/2\ E[S(n)X_i - E[S(n)X_i]]$$

or:

$$E[S(n)X_i\ h_i] \approx E[S(n)X_i]E[h_i]\ . \qquad (B.9)$$

Thus, from (B.9) it can be seen that if the random variables $S(n)X_i$ and h_i have a probability density which is concentrated near its center of gravity, then (B.8) can be rewritten as:

$$E[S(n)\underline{X}^T(n-n_1)\underline{h}] = E[S(n)\underline{X}^T(n-n_1)]E[\underline{h}]\ . \qquad (B.10)$$

Combining the results of (B.7) and (B.10) gives:

$$MSE(n) \approx E[S^2(n)] - 2E[S(n)\underline{X}^T(n-n_1)]E[\underline{h}_n] + E[\underline{h}_n^T]\,\Phi_{XX}\,E[\underline{h}_n]. \qquad (2.31)$$

APPENDIX C

MEAN AND VARIANCE OF THE TERM

$$\frac{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}{\underline{X}(n+1-n_1)\underline{X}(n+1-n_1)}$$

In this Appendix, a second order approximation to the mean of the term:

$$\frac{\underline{X}^T(n-n_1)\underline{X}(n-n_1)}{\underline{X}^T(n+1-n_1)\underline{X}(n+1-n_1)} \tag{C.1}$$

is derived assuming that the process X(n) is stationary. Consider the sequence of *r.v. defined as:*

$$Y_i = X^2(n+1-n_1-i) \quad . \tag{C.2}$$

Equation (C.1) can then be rewritten as:

$$f(\underline{Y}) = \frac{\sum_{i=-1}^{N-2} Y_i}{\sum_{i=0}^{N-1} Y_i} = 1 + \frac{Y_{-1}-Y_{N-1}}{\sum_{i=0}^{N-1} Y_i} \quad . \tag{C.3}$$

Consider expanding the above expression in a Taylor series about the point:

$$E[\underline{Y}]$$

where:

$$\underline{Y} = [Y_{-1}\ Y_0\ Y_1\ \cdots\ Y_{N-1}]^T$$

Thus, retaining only through second order terms, (C.3) can be rewritten as:

$$f(\underline{Y}) \approx f(E[Y])+\nabla f(\underline{Y}-E[\underline{Y}])+(\underline{Y}-E[\underline{Y}])^T\nabla^2 f(\underline{Y}-E[Y]) \qquad (C.4)$$

where:

∇f = gradient of f evaluated at $E[\underline{Y}]$

$\nabla^2 f$ = Hessian Matrix of f evaluated at $E[\underline{Y}]$.

Consider the components of the gradient vector ∇f evaluated at the point $E[\underline{Y}]$; recalling that X was assumed stationary:

$$\frac{2f}{2Y_i}\bigg|_{E[\underline{Y}]} = \frac{Y_{N-1}-Y_{-1}}{\sum_{K=0}^{N-1} Y_I^2}\bigg|_{E[\underline{Y}]} = 0 \qquad \begin{matrix} i\neq 1 \\ i\neq N-1 \end{matrix} \qquad (C.5)$$

$$\frac{2f}{2Y_{-1}}\bigg|_{E[\underline{Y}]} = \frac{1}{\sum_{K=0}^{N-1} Y_K}\bigg|_{E[\underline{Y}]} = \frac{1}{NE[Y]} \qquad (C.6)$$

$$\frac{2f}{2Y_{N-1}}\bigg|_{E[\underline{Y}]} = \frac{\sum_{j=-1}^{N-2} Y_j}{\sum_{K=0}^{N-1} Y_K^2}\bigg|_{E[\underline{Y}]} = \frac{-1}{NE[Y]} \quad . \qquad (C.7)$$

The elements of the Hessian Matrix $\nabla^2 f$ evaluated at the point $E[\underline{Y}]$ are as follows:

$$\left.\frac{\partial^2 f}{\partial Y_i \partial Y_j}\right|_{E[\underline{Y}]} = \left.\frac{2[Y_{-1}-Y_{N-1}]}{\left(\sum_{K=0}^{N-1} Y_K\right)^3}\right|_{E[\underline{Y}]} = 0 \qquad \begin{matrix} i,j\neq 1 \\ i,j=N-1 \end{matrix} \tag{C.8}$$

$$\left.\frac{\partial^2 f}{\partial Y_i \partial Y_{-1}}\right|_{E[\underline{Y}]} = \left.\frac{-1}{\left(\sum_{K=0}^{N-1} Y_K\right)^2}\right|_{E[\underline{Y}]} = \frac{-1}{Y^2 E^2[y]} \quad i\neq -1 \tag{C.9}$$

$$\left.\frac{\partial^2 f}{\partial Y_i \partial Y_{N-1}}\right|_{E[\underline{Y}]} = \left.\frac{\sum_{K=0}^{N-1} 2 - 2(Y_{N-1}-Y_{-1})\sum_{K=0}^{N-1} Y_K}{\left(\sum_{K=0}^{N-1} Y_K\right)^4}\right|_{E[\underline{Y}]} = \frac{1}{N^2 E^2[\underline{Y}]}$$

$$\begin{matrix} i\neq -1 \\ i\neq N-1 \end{matrix} \tag{C.10}$$

$$\left.\frac{\partial^2 f}{\partial^2 Y_{-1}}\right|_{E[\underline{Y}]} = 0 \tag{C.11}$$

$$\left.\frac{\partial^2 f}{\partial^2 Y_{N-1}}\right|_{E[\underline{Y}]} = \frac{-1}{\left(\sum_{j=0}^{N-1} Y_j\right)^2} = \frac{-1}{N^2 E^2[\underline{Y}]} \tag{C.12}$$

From Equations (C.5) through (C.12) it can be seen that (C.4) can be written as:

$$f(\underline{Y}) \approx f(E[\underline{Y}]) + \frac{(Y_{-1}-E[Y_{-1}])-(Y_{N-1}-E[Y_{N-1}])}{N\,E[Y]}$$

$$+ \frac{2(Y_{N-1}-Y_{-1})}{N^2E^2[Y]} \sum_{i=0}^{N-2} (Y_i-E[Y_i])$$

$$- \frac{(Y_{N-1}-E[Y_{N-1}])(Y_{N-1}+Y_{-1}-2E[Y])}{N^2\,E^2[Y]}$$

or:

$$f(\underline{Y}) \approx f(E[\underline{Y}]) + \frac{(Y_{-1}-Y_{N-1})}{N\,E[Y]} - \frac{2(Y_{-1}-Y_{N-1})}{N^2E^2[Y]} \sum_{i=0}^{N-2} (Y_i-E[Y_i])$$

$$- \frac{(Y_{N-1}-E[Y_{N-1}])}{N^2\,E^2[Y]} (Y_{N-1}+Y_{-1}-2E[Y]) \quad . \qquad (C.13)$$

Thus, the mean of $f(\underline{Y})$ can be written as:

$$E[f(\underline{Y})] \approx 1 - \frac{2}{N^2E^2[Y]} \sum_{i=0}^{N-2} (E[Y_{-1}Y_i]-E[Y_{N-1}Y_i])$$

$$- \frac{E[Y_{N-1}^2]+E[Y_{-1}Y_{N-1}]-E^2[Y_{N-1}]}{N^2\,E^2[Y]} \qquad (C.14)$$

or if X is assumed Gaussian:

$$E[f(\underline{Y})] \approx 1 - \frac{4}{N^2R_{XX}^2(0)} \sum_{i=0}^{N-2} (R_{XX}^2(i-1)-R_{XX}^2(i-N+1))$$

$$- \frac{3R_{XX}^2(0)+2R_{XX}^2(N-1-i)}{N^2\,R_{XX}^2(0)} \quad .$$

Since $R_{XX}(K)$ has a maximum at K=0:

$$|E[f(\underline{Y})]-1| \lesssim \frac{8(N-1)}{N^2} + \frac{5}{N^2} \quad . \tag{C.15}$$

Thus, it can be seen from (C.15) that $E[f(\underline{Y})] \approx 1$ for large values of N.

Determination of the variance of $f(\underline{Y})$ is difficult. However, intuitively, it can be argued that the variance becomes small as N becomes large. From Equation (C.3), it can be seen that $f(\underline{Y})$ can also be written as:

$$f(\underline{Y}) = \frac{\frac{1}{N}\sum_{i=-1}^{N-2} Y_i}{\frac{1}{N}\sum_{i=0}^{N-1} Y_i} \quad . \tag{C.16}$$

or, in terms of the sample autocorrelation function R_{XX}, discussed in Section 3.1:

$$f(\underline{Y}) = \frac{R_{XX}(0)_{N,n}}{R_{XX}(0)_{N,n+1}} \quad . \tag{C.17}$$

Now, it was shown in Section 3.1 that the variance of R_{XX} approaches zero as $N \to \infty$. Thus, both the numerator and denominator in (C.17) approach constants with probability 1 as N approaches ∞. Thus, intuitively, the variance of $f(\underline{Y})$ becomes small as N becomes large.

APPENDIX D

COMPUTER PROGRAMS FOR THE STUDY OF FOUR ADAPTIVE LINEAR ESTIMATION ALGORITHMS

This appendix describes the computer programs used in the study of the LMS Gradient, Jacobi Relaxation, SOR and Levinson Algorithms. Four separate programs were used in the study. One program was used for the simulation of the four adaptive algorithms; two were used in computing the theoretical bounds derived in Chapters II and III and for the comparison of these bounds with the simulation results, and the fourth program was used to compute the Discrete Fourier Transform of data generated by the simulation program.

D.1 Simulation Program

The four adaptive algorithms as well as the optimal filter for the SKEP problem were simulated on an IBM 370/165 computer. The basic simulation method was to generate a signal-plus-noise data sequence and to process this data sequence using each of the four algorithms as well as the optimal filter. The instantaneous squared error given by equation (6.4) was computed for each of the five filters implemented. The mean-square error was estimated as the ensemble average of the instantaneous error

over an ensemble of 100 signal estimate records.

D.1.1 Specifications of the Problem

The specifications of the problem are discussed in Section 6.1 and are summarized below:

1. Single frequency sinusoidal signal with random phase uniformly distributed on $[-\pi,\pi]$ specified by equation (5.1).
2. Zero-mean Gaussian autoregressive noise sequence with autocorrelation function given by equation (5.18).
3. Filter length N = 16.
4. Parameter values for data generation summarized in Figure 5.1.

D.1.2 Program Structure

The structure of the simulation program is illustrated in Figure D.1. Each program segment is described below:

1. ESTIM8 -- Main program of the simulation, contains segments ALLOC, SIMULATE and OUTPUT.
2. ALLOC -- Reads data used in controlling the simulation and parameter values for simulation, allocates required storage, initializes storage locations.

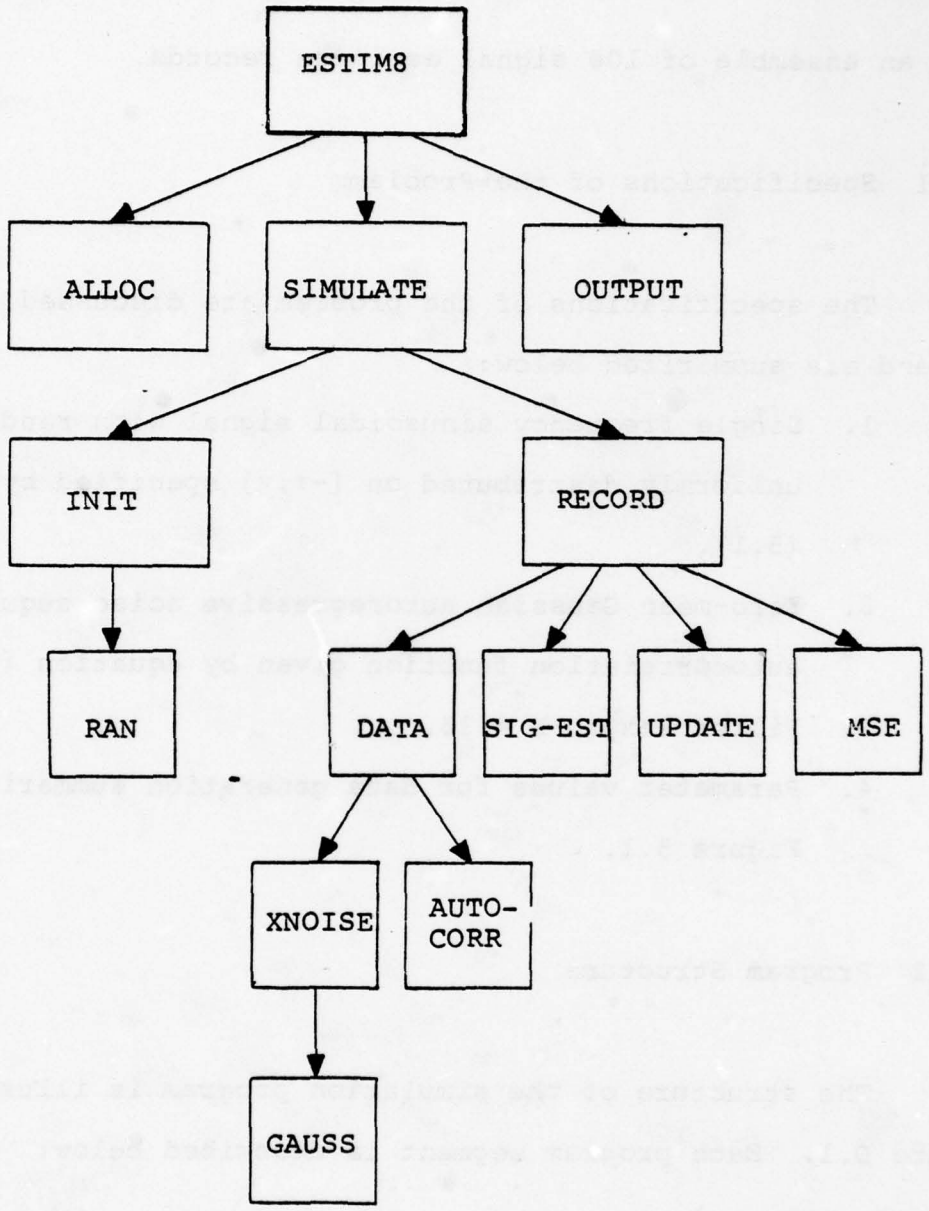


Figure D.1 Structure of the Simulation Program ESTIM8.

INPUTS:

N --	Filter length.
REL-LEN --	Number of samples/record.
#REC --	Number of records to be processed.
DELTA --	Delay value n_1.
ALPHA --	Autoregression parameter of noise sequence.
SNR --	Signal-to-noise ratio.
LJAC --	Variable-Iteration-Step parameter ℓ for Jacobi Relaxation Method.
KJAC --	Variable-Iteration-Step parameter k for Jacobi Relaxation Method.
LSOR --	Variable-Iteration-Step parameter ℓ for SOR Method.
KSOR --	Variable-Iteration-Step parameter k for SOR Algorithm.
KLEV --	Parameter k for Levinson's Algorithm.
BJAC --	Parameter β for Jacobi Relaxation Algorithm.
BSOR --	Parameter β for SOR Algorithm.
MU --	Parameter μ for LMS Gradient Algorithm.
H-INIT(N) --	Initial unit sample response vector.
AMP --	Amplitude of the signal.

FREQ -- Frequency of the signal (radians).

H-OPT(N) -- Optimal unit sample response vector.

OUTPUTS:

Same as Inputs.

3. SIMULATE -- Simulation segment. Produces an ensemble of #REC signal estimate records. Contains segments INIT and RECORD.

INPUTS:

Same as ALLOC.

OUTPUTS:

SIGNUL(n) -- Last signal record computed in the simulation.

SEOPT(n) -- Last signal estimate record produced by the Optimal Filter in the simulation.

SELMS(n) -- Last signal estimate record produced by the LMS Gradient Algorithm in the simulation.

SEJAC(n) -- Last signal estimate record produced by the Jacobi Relaxation Algorithm in the simulation.

SESOR(n) -- Last signal estimate record produced by the SOR Algorithm in the simulation.

SELEV(n) -- Last signal estimate record produced by the Levinson Algorithm in the simulation.

MSEOPT(n) -- Estimated MSE produced by the optimal filter.

MSELMS(n) -- Estimated MSE produced by the LMS Gradient Method.

MSEJAC(n) -- Estimated MSE produced by the Jacobi Relaxation Algorithm.

MSESOR(n) -- Estimated MSE produced by the SOR Algorithm.

MSELEV(n) -- Estimated MSE produced by the Levinson Algorithm.

4. OUTPUT -- Outputs the results of the simulation.

INPUTS:

Same as outputs of SIMULATE.

OUTPUTS:

Same as Inputs.

5. INIT -- Initializes variables prior to the generation of each record in the simulation. Calls Procedure RAN.

OUTPUTS:

PHASE -- Phase of the sinusoid.

In addition all unit sample response vectors and autocorrelation estimates are initialized.

6. RECORD -- Produces a single signal estimate record for each of the four adaptive algorithms as well as for the optimal filter. Contains segments DATA, SIG-EST, UPDATE and MSE.

INPUTS:

PHASE -- Phase of the sinusoid.

Initialized unit sample response vectors and autocorrelation estimates.

OUTPUTS:

Signal estimate records for each of the five filters. Estimates of MSE(n) for each of the five filters based on all the data previously processed.

7. RAN(IY) -- Produces Uniform [0,1] random numbers using the power residue method.

INPUTS:

IY -- Seed number, changed by RAN.

OUTPUTS:

RAN(IY) -- Uniform [0,1] number.

8. DATA -- Generates data sample, computes autocorrelation estimate. Contains segment AUTO-CORR. Calls Procedure XNOISE.

INPUTS:

N -- Filter length.

AMP -- Signal amplitude.

FREQ -- Signal frequency (radians).

PHASE -- Signal phase (radians).

DELTA -- Delay value n_1.

ALPHA -- Autoregression parameter for noise sequence.

SNR -- Signal-to-noise ratio.

#SAMP -- Time index.

OUTPUTS:

SIGNUL(n) -- Signal sample at time n.

X(n) -- Data sample at time n. The array X operates as a stack.

RXX(k) -- Estimate for Topelitz autocorrelation matrix.

AUTO(k) -- Estimate for autocorrelation vector.

9. SIG-EST -- Generates signal estimate samples for each of the five filters simulated.

INPUTS:

N -- Filter length.

X(n) -- Data vector.

H-OPT -- Optimum unit sample response vector.

H-LMS -- Unit sample response for LMS Gradient Algorithm.

H-JAC -- Unit sample response for Jacobi Relaxation.

H-SOR -- Unit sample response for SOR Algorithm.

H-LEV -- Unit sample response for Levinson's Algorithm.

SEOPT -- Optimal signal estimate.

SEJAC -- Jacobi Relaxation signal estimate.

SELMS -- LMS Gradient Signal estimate.

SESOR -- SOR signal estimate.

SELEV -- Levinson signal estimate.

10. UPDATE -- Updates the unit sample response vectors for the four adaptive algorithms.

INPUTS:

N -- Filter length.

H-LMS -- Unit sample response vector for LMS Gradient Algorithm at time n.

H-JAC -- Unit sample response vector for Jacobi Relaxation Algorithm at time n.

H-SOR -- Unit sample response vector for SOR Algorithm at time n.

H-LEV -- Unit sample response vector for Levinson's Algorithm at time n.

OUTPUTS:

H-LMS -- Unit sample response vector for LMS Gradient Algorithm at time n+1.

H-JAC -- Unit sample response vector for Jacobi Relaxation Algorithm at time n+1.

H-SOR -- Unit sample response vector for SOR Algorithm at time n+1.

H-LEV -- Unit sample response vector for Levinson's Algorithm at time n+1.

11. MSE -- Computes estimated mean-square error based on all data records previously processed.

INPUTS:

SIGNUL(n) -- Signal sample at time n.

SEOPT(n) -- Optimal signal estimate at time n.

SELMS(n) -- LMS Gradient Signal estimate at time n.

SEJAC(n) -- Jacobi Relaxation signal estimate at time n.

SESOR(n) -- SOR Algorithm signal estimate at time n.

SELEV(n) -- Levinson's Algorithm signal estimate at time n.

OUTPUTS:

MSEOPT(n) -- Estimated MSE for optimal filter at time n.

MSELMS(n) -- Estimated MSE for LMS Gradient Algorithm at time n.

MSEJAC(n) -- Estimated MSE for Jacobi Relaxation at time n.

MSESOR(n) -- Estimated MSE for SOR Algorithm at time n.

MSELEV(n) -- Estimated MSE for Levinson's Algorithm at time n.

12. XNOISE -- Recursively generates autoregressive noise sequence. Calls Procedure GAUSS.

INPUTS:

ALPHA --	Autoregression parameter for noise sequence.
DELTA --	Delay value n_1.
#SAMP --	Time index.
Y --	Vector containing previous DELTA Gaussian random numbers.
IY --	SEED number.

OUTPUTS:

XNOISE --	Autoregressive noise sequence sample.
Y --	Previous DELTA Gaussian random numbers.
IY --	New Seed number.

13. AUTO-CORR -- Computes autocorrelation estimates.

INPUTS:

X(n) --	Data vector.
N --	Filter length.

OUTPUTS:

RXX(k) --	Topelitz autocorrelation matrix estimate.
AUTO(k) --	Autocorrelation vector estimate.

14. GAUSS -- Generates approximately Gaussian random numbers by summing 12 uniformly distribuited random numbers.

INPUTS:

IY -- Seed number.

OUTPUTS:

GAUSS -- Approximately Gaussian random number.

IY -- New seed number.

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

D.1.3 Source Program

```
                ADAPTIVE LINE ENHANCER SIMULATION 6/13/78
*****************************************************************

 THIS PROGRAM SIMULATES A  SPECTRAL LINE ENHANCER USING THE LMS
 GRADIENT PROCEDURE PROPOSED BY WIDROW ET.AL.   (IEEE PROC. DEC.
 1975).  IN ADDITION, ALGORITHMS WHICH ESTIMATE THE INPUT AUTO-
 CORRELATION MATRIX AND SOLVE FOR THE FILTER IMPULSE RESPONSE
 USING THE JACOBI RELAXATION, SOR OR LEVINSON ALGORITHMS ARE ALSO
 SIMULATED AS WELL AS THE OPTIMUM FILTER FOR FOR THE SKEP PROBLEM.
 THE INPUT TO ALL OF THE SIMULATED PROCESSORS CONSISTS OF A SINGLE
 SINUSOID (WITH USER SPECIFIED FREQ AND AMPLITUDE) ADDED TO AN
 AUTOREGRESSIVE NOISE SEQUENCE.  THE SIGNAL AND NOISE ARE ASSUMED
 TO BE UNCORRELATED AND NOISE SAMPLES A DISTANCE DELTA OR GREATER
 APART ARE ALSO ASSUMED UNCORRELATED.  NOISE SAMPLES ARE GENERATED
 BY PROCEDURE XNOISE AND ARE ASSUMED TO HAVE MEAN=0 AND VAR=1 AS
 WELL AS THE CORRELATION PROPERTIES INDICATED ABOVE.

****************************** INPUT DATA ***************************

 THE FIRST INPUT RECORD IS IN A DATA FORM AND SPECIFIES THE
 FOLLOWING PARAMETERS:

 N = # OF WEIGHTS FOR FOR ALL ALGORITHMS (INTEGER)
 REC_LEN = RECORD LENGTH FOR SIMULATION RUN (INTEGER)
 #REC = # OF REC_LEN RECORDS TO BE AVERAGED (INTEGER)
 DELTA = # SAMPLES FOR DELAY--SEE ABOVE (INTEGER)
 SNR = DESIRED SIGNAL-TO-NOISE RATIO (REAL)
 MEAN = DESIRED MEAN OF THE NOISE (REAL)
 ALPHA = AUTOREGRESSION PARAMETER (REAL) (SEE COMMENTS IN XNOISE)
 LJAC = VARIABLE ITERATION STEP PARAMETER L FOR JACOBI METHOD(INT)
 KJAC = VARIABLE ITERATION STEP PARAMETER K FOR JACOBI METHOD(INT)
 LSOR = VARIABLE ITERATION STEP PARAMETER L FOR SOR METHOD (INT)
 KSOR = VARIABLE ITERATION STEP PARAMETER K FOR SOR METHOD (INT)
 KLEV = # OF SAMPLES BETWEEN APPLICATION OF LEVINSON METHOD (INT)
 BSOR = PARAMETER BETA FOR THE SOR ALGORITHM (REAL)
 BJAC = PARAMETER BETA FOR THE JACOBI RELAXATION METHOD (REAL)

 THE NEXT SET OF INPUT DATA IS IN A LIST FORM.  THE FOLLOWING
 DATA IS READ:

 AMP = AMPLITUDE OF SIGNAL (REAL)
 FREQ = FREQUENCY OF SIGNAL IN RADIANS (REAL)
 IY = SEED NUMBER FOR RANDOM NUMBER GENERATION (ODD INTEGER)

 THE LAST INPUT RECORD GIVES PARAMETERS USED IN THE EACH OF THE
 ALGORITHMS.  A LIST FORM IS USED TO ENTER:

 MU = VALUE OF MU TO BE USED IN LMS PROCEDURE (REAL)
 W_INIT(N) = INITIAL VALUES OF THE N FILTER WEIGHTS (REAL)
 H_OPT = OPTIMAL UNIT SAMPLE RESPONSE (REAL)

*****************************************************************
************************ END OF COMMENTS ************************
```

```
*****************************************************************

                    BEGIN MAIN PROGRAM

*****************************************************************

ESTIM8: PROC OPTIONS(MAIN);

        DCL (N,REC_LEN,#REC,DELTA,REC) FIXED BIN;
        DCL #SAMP FIXED BIN INIT (0);
        DCL SIGPOW FLOAT BIN INIT(0.0);
        DCL (POINT,DPOINT) FIXED BIN;
        DCL (I,J,K,L,M,LEN_X) FIXED BIN;
        DCL IY FIXED BIN(31) INIT(3);
        DCL (KGS,KJAC,LJAC,LSOR,KSOR,KLEV) FIXED BIN;
        DCL KJA FIXED BIN;
        DCL (A,B,C,D,Z,MU,ERROR,T,SNR) FLOAT BIN;
        DCL(MEAN,ALPHA) FLOAT BIN;
        DCL (SORER,LMSER) FLOAT BIN;
        DCL JACER FLOAT BIN;
        DCL (BSOR,BJR) FLOAT BIN;
        DCL (AMP,FREQ,PHASE,WT) FLOAT BIN;
        DCL (XLAM,THETA,XNU,OPTER,LEVER) FLOAT BIN;
        ON ENDPAGE B=1.0;

        /* READ DATA FROM CARDS */

        GET DATA(N,REC_LEN,#REC,DELTA,LJAC,KJAC,LSOR,KSOR,KLEV,SNR,
                 MEAN,ALPHA,BSOR,BJR);
        PUT DATA(N,REC_LEN,#REC,DELTA,LJAC,KJAC,LSOR,KSOR,KLEV,SNR,
                 MEAN,ALPHA,BSOR,BJR);

        /* COMPUTE ARRAY DIMENSIONS */

        I=REC_LEN-1;
        LEN_X=N+DELTA;
        J=LEN_X-1;
        K=N-1;
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
/*****************************************/
/*                                       */
/* ALLOCATE STORAGE AND INITIALIZE ARRAYS */
/*                                       */
/*****************************************/

ALLOC:
  BEGIN;
     DCL SIGNUL(0:I) FLOAT BIN;
     DCL X(0:J) FLOAT BIN  INIT((LEN_X)0.0);
     DCL SEOPT(0:I) FLOAT BIN;
     DCL HOPT(0:K) FLOAT BIN;
     DCL MSEOPT(0:I) FLOAT BIN INIT((REC_LEN)0.0);
     DCL HLMS(0:K) FLOAT BIN INIT((N)0.0);
     DCL SELMS(0:I) FLOAT BIN;
     DCL MGS(0:K) FLOAT BIN;
     DCL HSOR(0:K) FLOAT BIN;
     DCL SESOR(0:I) FLOAT BIN;
     DCL MSESOR(0:I) FLOAT BIN INIT((REC_LEN)0.0);
     DCL MSESOR(0:I) FLOAT BIN INIT((REC_LEN)0.0);
     DCL WJTMP(0:K) FLOAT BIN;
     DCL WJAC(0:K) FLOAT BIN INIT((N)0.0);
     DCL HJAC(0:K) FLOAT BIN;
     DCL SEJAC(0:I) FLOAT BIN;
     DCL MSEJAC(0:I) FLOAT BIN INIT((REC_LEN)0.0);
     DCL HLEV(N) FLOAT BIN;
     DCL WLV(N) FLOAT BIN;
     DCL G(N) FLOAT BIN;
     DCL GDUM(N) FLOAT BIN;
     DCL SELEV(0:I) FLOAT BIN;
     DCL MSELEV(0:I) FLOAT BIN INIT((REC_LEN)0.0);
     DCL Y(0:DELTA-1) FLOAT BIN;
     DCL RXX(0:K) FLOAT BIN;
     DCL RXXN(0:K) FLOAT BIN;
     DCL AUTO(0:K) FLOAT BIN INIT((N) 0.0);
     DCL W_INIT(0:K) FLOAT BIN;

/*  END OF STORAGE ALLOCATION  */

/*  INPUT AMPLITUDE AND FREQUENCY FOR SKEP PROBLEM */

GET LIST(AMP,FREQ,IY);
PUT SKIP DATA(AMP,FREQ,IY);

/* COMPUTE SIGNAL POWER */

SIGPOW=AMP**2/2.0;

/* INPUT MU AND INITIAL W VECTOR */

GET LIST(MU,(W_INIT(I) DO I=0 TO N-1));
PUT SKIP LIST(MU,(W_INIT(I) DO I=0 TO N-1));

/* INPUT OPTIMAL UNIT SAMPLE RESPONSE */

GET LIST((HOPT(I) DO I=0 TO N-1));
PUT LIST((HOPT(I) DO I=0 TO N-1));

*************** END OF DATA ENTRY AND INITIALIZATION *********
```

```
/*****************************************************************/
 SIMULATION BEGINS HERE.  ALL FOUR ALGORITHMS AS WELL AS THE
 OPTIMAL FILTER FOR THE SKEP PROBLEM PROCESS THE SAME DATA AND ARE
 USED TO GEN. ESTIMATES OF THE SIGNAL FOR REC_LEN POINTS.  #REC
 SUCH RECORDS ARE GENERATED AND THE SQUARED ERRORS ARE AVERAGED
 OVER THIS ENSEMBLE.
/*****************************************************************/

     SIMULATE:
       DO REC=0 TO #REC-1;

          /* INITIALIZE PARAMETERS BEFORE EACH RUN */

          HLMS=W_INIT;
          WJAC=W_INIT;
          HJAC=W_INIT;
          WGS=W_INIT;
          HSOR=W_INIT;

          DO K=0 TO N-1;
            HLEV(K+1)=W_INIT(K);
            WLV(K+1)=W_INIT(K);
          END;

          X=0.0;
          KGS=0;
          KJA=0;
          #SAMP=0;
          RXX=0.0;
          AUTO=0.0;
          PHASE=6.2831853*RAN(IY);

          RECORD:
            DO I=0 TO REC_LEN-1;

                  /* GENERATE SIGNAL & DATA SAMPLE */

               DO WHILE ((#SAMP-I) < LEN_X);
                    SIGNUL(I)=AMP*COS(FREQ*#SAMP+PHASE);
                    POINT=MOD((#SAMP+LEN_X),LEN_X);
                    X(POINT)=SIGNUL(I)+XNOISE(ALPHA,DELTA,#SAMP,Y,IY)*
                             SQRT(SIGPOW/SNR)+MEAN;

                    AUTOCORR:
                      DO J=0 TO N-1 WHILE(#SAMP>=J);
                        M=MOD(POINT-J+LEN_X,LEN_X);
                        RXX(J)=RXX(J)*(#SAMP-J)/(#SAMP-J+1)+X(POINT)*
                               X(M)/(#SAMP-J+1);
                        IF (#SAMP-DELTA)>=J
                          THEN DO;
                            DPOINT=MOD(POINT-DELTA-J+LEN_X,LEN_X);
                            M=#SAMP-DELTA-J;
                            AUTO(J)=AUTO(J)*M/(M+1)+X(DPOINT)*X(POINT)/
                                    (M+1);
                          END;
                    END AUTOCORR;

                    #SAMP=#SAMP+1;

               END;
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
/*****************************************/
/*                                       */
/* UPDATE UNIT SAMPLE RESPONSE FOR JACOBI */
/* RELAXATION, SOR ALGORITHM AND LEVINSON */
/* ALGORITHM.                            */
/*                                       */
/*****************************************/

/* SOR ALGORITHM WITH VARIABLE ITERATION  */
/*  STEPS.                               */

IF MOD(I,KSOR)=0
  THEN DO;
    MSOR=WGS;

    SOR:
      DO K=0 TO LSOR-1;
        IF KGS>N-1
          THEN KGS=0;
        WGS(KGS)=(1.-BSOR)*WGS(KGS)*RXX(0)/BSOR;
        DO J=0 TO N-1;
          M=ABS(KGS-J);
          IF J^=KGS
            THEN WGS(KGS)=WGS(KGS)-RXX(M)*WGS(J);
        END;
        WGS(KGS)=(WGS(KGS)+AUTO(KGS))*BSOR/RXX(0);
        KGS=KGS+1;
    END SOR;

END;

/* JACOBI RELAXATIN METHOD WITH VARIABLE  */
/* ITERATION STEPS.                      */

IF MOD(I,KJAC)=0
  THEN DO;
    MJAC=WJAC;
    WJTMP=WJAC;

    JACOBI:

      DO K=0 TO LJAC-1;
        IF KJA>N-1
          THEN KJA=0;
        WJAC(KJA)=0.0;
        DO J=0 TO N-1;
          M=ABS(KJA-J);
          WJAC(KJA)=WJAC(KJA)-RXX(M)*WJTMP(J);
        END;
        WJAC(KJA)=WJTMP(KJA)+BJR*(WJAC(KJA)+AUTO(KJA))
                  /RXX(0);
        KJA=KJA+1;

      END JACOBI;

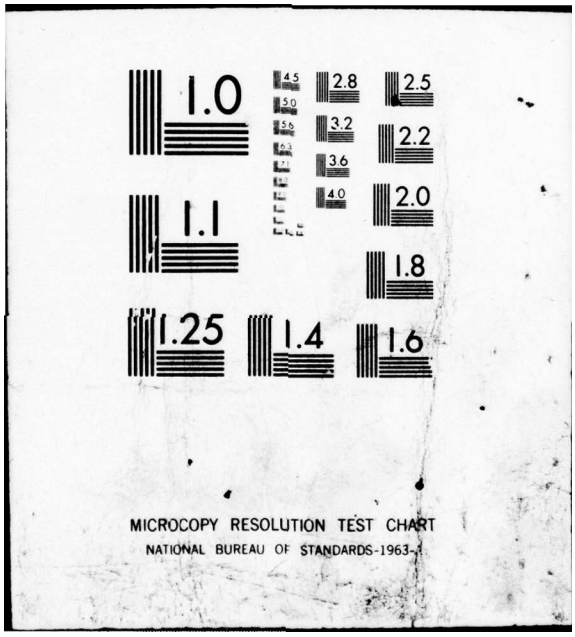
END;
```

AD-A068 760 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/3
ADAPTIVE LINEAR ESTIMATION ALGORITHMS APPLIED TO SPECTRAL LINE --ETC(U)
AUG 78 S D HUFFMAN N00014-75-C-0191
UNCLASSIFIED TR-15 NL

4 OF 4
AD
AO 68760



END
DATE
FILMED
6 -79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
/* LEVINSONS ALGORITHM */

IF MOD(I,KLEV)=0
  THEN DO;
    HLEV=WLV;

    /* INITIALIZE VECTORS */

    RXXN=RXX/RXX(0);
    WLV=0.0;
    G=0.0;
    WLV(1)=AUTO(0)/RXX(0);
    G(1)=-RXXN(1);
    XLAM=1.-RXXN(1)**2;

    LEVINSON:

      DO K=1 TO N-1;
        THETA=AUTO(K)/RXX(0);
      IF (K+1)<=(N-1)
        THEN XNU=-RXXN(K+1);
      DO J=1 TO K;
        THETA=THETA-WLV(J)*RXXN(K-J+1);
        XNU=XNU-RXXN(J)*G(J);
      END;
      DO J=1 TO K;
        WLV(J)=WLV(J)+THETA/XLAM*G(J);
        GDUM(J+1)=G(J)+XNU/XLAM*G(K-J+1);
      END;
      G(1)=XNU/XLAM;
      DO J=2 TO K+1;
        G(J)=GDUM(J);
      END;
      WLV(K+1)=THETA/XLAM;
      XLAM=XLAM-XNU**2/XLAM;

    END LEVINSON;

END;

/* END OF WEIGHT VECTOR UPDATE  */

/*******************************/
/*                             */
/* COMPUTE SIGNAL ESTIMATES */
/*                             */
/*******************************/

SEOPT(I)=0.0;
SELMS(I)=0.0;
SESOR(I)=0.0;
SEJAC(I)=0.0;
SELEV(I)=0.0;

SIGEST:
  DO J=0 TO N-1;
    DPOINT=MOD(POINT-DELTA-J+LEN_X,LEN_X);
    SEOPT(I)=SEOPT(I)+HOPT(J)*X(DPOINT);
    SELMS(I)=SELMS(I)+HLMS(J)*X(DPOINT);
    SEJAC(I)=SEJAC(I)+HJAC(J)*X(DPOINT);
    SESOR(I)=SESOR(I)+HSOR(J)*X(DPOINT);
    SELEV(I)=SELEV(I)+HLEV(J+1)*X(DPOINT);
END SIGEST;

/* END OF SIGNAL ESTIMATION */
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
/*****************************/
/*                           */
/* UPDATE LMS WEIGHT VECTOR  */
/*                           */
/*****************************/

ERROR=X(POINT)-SELMS(I);
UPDATE:
  DO J=0 TO N-1;
    DPOINT=MOD(POINT-DELTA-J+LEN_X,LEN_X);
    HLMS(J)= HLMS(J)+2.*MU*ERROR*X(DPOINT);
END UPDATE;

/*************************************/
/*                                   */
/* COMPUTE MSE FOR ALL  ALGORITHMS   */
/*                                   */
/*************************************/

    OPTER=(SIGNUL(I)-SEOPT(I))**2;
    LMSER=(SIGNUL(I)-SELMS(I))**2;
    SORER=(SIGNUL(I)-SESOR(I))**2;
    JACER=(SIGNUL(I)-SEJAC(I))**2;
    LEVER=(SIGNUL(I)-SELEV(I))**2;
    MSEOPT(I)=MSEOPT(I)+OPTER/#REC;
    MSELMS(I)=MSELMS(I)+LMSER/#REC;
    MSEJAC(I)=MSEJAC(I)+JACER/#REC;
    MSESOR(I)=MSESOR(I)+SORER/#REC;
    MSELEV(I)=MSELEV(I)+LEVER/#REC;

  END RECORD;

END SIMULATE;

********** END OF SIMULATION LOOP **********

*****************************************************

 THIS SECTION TAKES THE RESULTS OF THE PROCESSING OF THE LAST
 DATA RECORD  AND PREPARES AND PLOTS THE PROGRAM OUTPUT

*****************************************************

OUT:
  DO I=0 TO REC_LEN-1;
    PUT SKIP EDIT(SIGNUL(I),SEOPT(I),SELMS(I),SEJAC(I),
      SESOR(I),SELEV(I),MSEOPT(I),MSELMS(I),MSEJAC(I),
      MSESOR(I),MSELEV(I))((11)E(13,6));
END OUT;

********** END MAIN PROGRAM **********
```

```
/*****************************************************************

   XNOISE GENERATES AN AUTOREGRESSIVE SEQUENCE WHEN CALLED
   RECURSIVELY.  XNOISE=SUM((ALPHA**K)*Y(K)) K=0. DELTA-1.  WHERE
   Y(K) ARE INDEPENDENT, IDENTICALLY DISTRIBUITED RANDOM VARIABLES
   TO SAVE EXECUTION TIME, NOISE SAMPLE IS ACTUALLY COMPUTED IN A
   RECURSIVE FASHION.

*****************************************************************/

XNOISE: PROC(ALPHA,DELTA,#SAMP,Y,IY) RETURNS(FLOAT BIN);
        DCL(ALPHA,XOLD,YOLD) FLOAT BIN;
        DCL XNEW FLOAT BIN STATIC;
        DCL(DELTA,#SAMP,K,POINT) FIXED BIN;
        DCL IY FIXED BIN(31);
        DCL Y(*) FLOAT BIN;
        IF #SAMP = 0
          THEN DO;                  /* INITIALIZE SEQUENCE ON FIRST CALL
                 XNEW=0.0;
                 DO K=0 TO DELTA-1;
                   Y(K)=GAUSS(IY);
                   XNEW=ALPHA*XNEW+Y(K);
                 END;
               END;
          ELSE DO;
                 XOLD=XNEW;
                 POINT=MOD(#SAMP+DELTA-1,DELTA);
                 YOLD=Y(POINT);
                 Y(POINT)=GAUSS(IY);
                 XNEW=ALPHA*XOLD+Y(POINT)-YOLD*ALPHA**DELTA;
               END;

        /*  SCALE SO THAT VAR(XOLD)=1  */

        XOLD=XNEW*(1-ALPHA**(2*(DELTA-1)))/(1-ALPHA**2);
        RETURN(XOLD);

END XNOISE;

/*****************************************************************

   GAUSS GENERATES NORMAL(0,1) RANDOM VARIABLES BY TAKING THE SUM
   OF 12 INDEPENDENT UNIFORM(0,1) RANDOM VARIABLES GENERATED BY TH
   POWER RESIDUE METHOD.  SINCE THE MEAN OF U(0,1)=1/2 AND THE VAR
   IANCE OF U(0,1)=1/12, THE SUM OF 12 OF THESE INDEPENDENT RANDOM
   VARIABLES HAS:  MEAN(SUM)=SUM(MEANS)=6 AND VAR(SUM)=SUM(VAR)=1.
   THEREFORE, THE R.V. X=SUM-6 HAS MEAN=0 AND VARIANCE=1 AND IS
   APPROXIMATELY NORMAL BY THE CENTRAL LIMIT THEOREM.

*****************************************************************/

(NOSIZE,NOFIXEDOVERFLOW):
GAUSS:  PROC(IY) RETURNS(FLOAT);
        DCL I FIXED BIN;
        DCL IY FIXED BIN(31);
        DCL A FLOAT BIN;
        A=0.0;
        DO I=1 TO 12;
          IY=IY*65539;
          IF IY < 0
            THEN IY=IY+2147483647+1;
          A=IY*0.4656613E-9+A;
        END;
        RETURN(A-6.0);
END GAUSS;
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
/*****************************************************************/

 SUBROUTINE RAN GENERATES UNIFORM (0,1) RANDOM VARIABLES
   USING THE POWER RESIDUE METHOD.

/*****************************************************************/

(NOSIZE,NOFIXEDOVERFLOW):
RAN: PROC(IY) RETURNS(FLOAT BIN);
        DCL IY FIXED BIN(31);
        DCL A FLOAT BIN;
        IY=IY*65539;
        IF IY < 0
          THEN IY=IY+2147483647+1;
        A=IY*0.4656613E-9;
        RETURN(A);
END RAN;

END ALLOC;

END ESTIM8;
```

D.2 Computation of Eigenvalues and Eigenvectors

The theoretical bounds derived in Chapters II and III require knowledge of the autocorrelation matrix and its eigenvalues. Expressions for the autocorrelation matrix and vector for the SKEP problem were derived in Chapter V. The computer program described here determines Φ_{XX} and its eigenvalues and eigenvectors as well as the regular and uncoupled optimal unit sample response vectors. The uncoupled initial unit sample response is also computed. The program was run on an IBM 370/165 computer and is written in the PL-I language.

D.2.1 Specifications of the Problem.

The specifications of the problem are discussed in Section 6.1 and are summarized below:

1. Single frequency sinusoidal signal with random phase uniformly distributed on $[-\pi,\pi]$ specified by equation (5.1).
2. Zero-mean Gaussian autoregressive noise sequence with autocorrelation function given by equation (5.18).
3. Filter length N = 16.
4. Parameter values for data generation summarized in Figure 5.1.

D.2.2 Program Structure

The structure of the program is illustrated in Figure D.2. Each program segment is described below:

1. EIGEN -- Main program. Contains program segments ALLOC, AUTO-CORR, USR, MMSE and OUTPUT. Calls procedure MSDU.

   INPUTS:

N --	Filter length.
DELTA --	Delay value n_1.
ALPHA --	Autoregression parameter for noise sequence.
SNR --	Signal-to-noise ratio.
AMP --	Amplitude of signal.
FREQ --	Frequency of signal (radians).
H-INIT --	Initial unit sample response vector.

2. ALLOC -- Allocates required storage.

   INPUTS:

N --	Filter length.

3. AUTO-CORR -- Computes autocorrelation matrix and vector.

   INPUTS:

N --	Filter length.
DELTA --	Delay value n_1.
ALPHA --	Autoregression parameter for noise sequence.

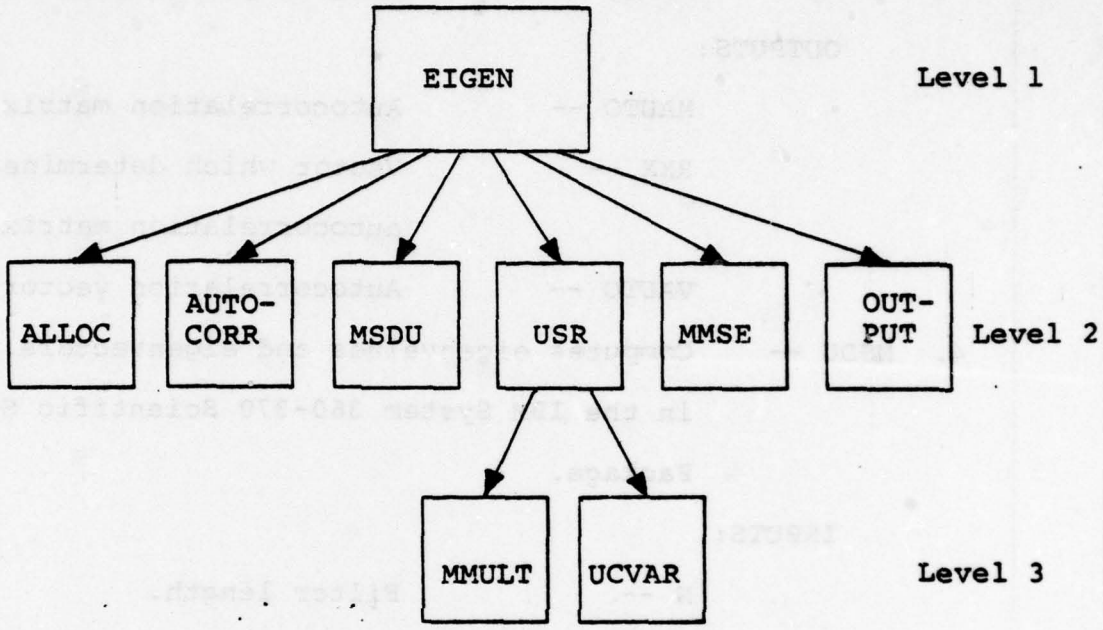


Figure D.2 Structure of Program EIGEN -- for the Computation of the Eigenvalues of ζ_{XX}.

SNR -- Signal-to-noise ratio.

AMP -- Amplitude of signal.

FREQ -- Frequency of signal (radians).

OUTPUTS:

MAUTO -- Autocorrelation matrix.

RXX -- Vector which determines Topelitz autocorrelation matrix.

VAUTO -- Autocorrelation vector.

4. MSDU -- Computes eigenvalues and eigenvectors. Available in the IBM System 360-370 Scientific Subroutine Package.

INPUTS:

N -- Filter length.

MAUTO -- Autocorrelation matrix.

OUTPUTS:

MAUTO -- Matrix contains eigenvalues along the diagonal.

MODAL -- Modal matrix of the system.

5. USR -- Computes optimal unit sample response, uncoupled optimal unit sample response and uncoupled initial unit sample response. Contains segment UCVAR. Calls procedure MMULT.

INPUTS:

MAUTO -- Matrix containing eigenvalues along diagonal.

MODAL -- Modal matrix of the system.

N -- Filter length.

HINIT -- Initial unit sample response.

OUTPUTS:

H -- Optimal unit sample response vector.

HUC -- Uncoupled optimal unit sample response vector.

HIUC -- Uncoupled initial unit sample response vector.

6. MMSE -- Computes MSE produced by the optimal filter.

INPUTS:

MAUTO -- Matrix containing eigenvalues along the diagonal.

HUC -- Uncoupled optimal unit sample response vector.

AMP -- Signal amplitude.

OUTPUTS:

MSE -- MSE produced by the optimal filter.

7. OUTPUT -- Outputs results of program.

INPUTS:

MAUTO -- Matrix containing eigenvalues along the diagonal.

H -- Optimal unit sample response vector.

HUC -- Uncoupled optimal unit sample response vector.

HINIT -- Initial unit sample response vector.

HIUC -- Uncoupled unit sample response vector.

RXX -- Vector which determines the Topelitz autocorrelation matrix.

N -- Filter length.

DELTA -- Delay value n_1.

ALPHA -- Autoregression parameter for noise sequence.

AMP -- Signal amplitude.

FREQ -- Signal frequency (radians).

SNR -- Signal-to-noise ratio.

MSE -- Mean-square error produced by optimal filter.

OUTPUTS:

N -- Filter length.

DELTA -- Delay value n_1.

AMP -- Signal amplitude.

FREQ -- Signal frequency (radians).

ALPHA -- Autoregression parameter for noise sequence.

MSE -- MSE produced by optimal filter.

MAUTO(I,I)-- Eigenvalues of autocorrelation matrix.

H -- Optimal unit sample response - Initial unit sample response.

HUC -- Uncoupled optimal unit sample response - Uncoupled unit sample response.

RXX -- Vector which determines the Topelitz autocorrelation matrix.

8. MMULT -- Multiplies an NxN matrix and an N-vector.

INPUTS:

MATRIX -- Matrix for multiplication.

V1 -- Vector for multiplication.

N -- Filter length.

K -- K=0 gives MATRIX·V1.
K=1 gives $MATRIX^{T}$·V1.

OUTPUTS:

VR -- Vector result of the multiplication.

9. UCVAR -- Solves system of equations in terms of uncoupled variables.

INPUTS:

MAUTO -- Matrix containing eigenvalues along the diagonal.

VAUTO -- Autocorrelation vector.

N -- Length of filter.

OUTPUTS:

HUC -- Uncoupled optimal unit sample response vector.

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

D.2.3 Source Listing

```
/*****************************************************/
/*                                                   */
/* THIS PROGRAM COMPUTES THE EIGENVALUES OF THE      */
/* AUTOCORRELATION MATRIX FOR THE SKEP PROBLEM       */
/* THE NORMAL AND UNCOUPLED OPTIMAL UNIT SAMPLE      */
/* RESPONSES AND THE UNCOUPLED INITIAL UNIT          */
/* SAMPLE RESPONSE ARE ALSO COMPUTED AS WELL         */
/* AS THE AUTOCORRELATION VECTOR.                    */
/*                                                   */
/*****************************************************/

EIGEN:  PROC OPTIONS(MAIN);

        DCL (AMP,FREQ,SNR,ALPHA,T) FLOAT BIN;
        DCL (DELTA,I,J,K,M,N) FIXED BIN;
        DCL (RSS,RNN,MSE) FLOAT BIN;

        /* READ DATA FROM CARDS */

        GET DATA(N,DELTA,AMP,FREQ,T,ALPHA,SNR);

        /* ALLOCATE STORAGE */

        ALLOC:
          BEGIN;
            DCL MAUTO(N,N) FLOAT BIN;
            DCL VAUTO(N) FLOAT BIN;
            DCL H(N) FLOAT BIN INIT((N)0.0);
            DCL HUC(N) FLOAT BIN INIT((N)0.0);
            DCL MODAL(N,N) FLOAT BIN;
            DCL HINIT(N) FLOAT BIN INIT((N)0.0);
            DCL HIUC(N) FLOAT BIN INIT((N)0.0);
            DCL RXX(N) FLOAT BIN;
            DCL MSDU ENTRY((*,*) FLOAT BIN,(*,*) FLOAT BIN,FIXED BIN,
                  FIXED BIN);

        /* END OF STORAGE ALLOCATION */

        /* INPUT INITIAL UNIT SAMPLE RESPONSE */

        GET LIST((HINIT(I) DO I=1 TO N));

        /* COMPUTE AUTOCORRELATION MATRIX AND VECTOR */

        AUTO_CORR:
          DO I=1 TO N;
            DO J=1 TO I;
              K=ABS(I-J);
                RSS=0.5*(AMP**2)*COS(6.283185*FREQ*T*K);
              IF K< DELTA
                THEN RNN=(ALPHA**K)*((1.-ALPHA**(2*DELTA-2*K))/
                          (1.-ALPHA**2))*(AMP**2)/(2.*SNR);
                ELSE RNN=0.0;
              MAUTO(I,J)=RSS+RNN;
              MAUTO(J,I)=MAUTO(I,J);
            END;
            VAUTO(I)=0.5*(AMP**2)*COS(6.283185*FREQ*T*(DELTA+I-1));
            RXX(I)=MAUTO(I,1);
          END AUTO_CORR;

        /* COMPUTE EIGENVALUES AND EIGENVECTORS */

        K=0;
        CALL MSDU(MAUTO,MODAL,N,K);
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
        /* DETERMINE UNCOUPLED INITIAL UNIT SAMPLE RESPONSE */

        K=1;
        CALL MMULT(MODAL,HINIT,HIUC,N,K);

        /* SOLVE EQNS IN TERMS OF UNCOUPLED VARIABLES */

        UCVAR:
          DO I=1 TO N;
            DO J=1 TO N;
              HUC(I)=HUC(I)+MODAL(J,I)*VAUTO(J);
            END;
            HUC(I)=HUC(I)/MAUTO(I,I);
        END UCVAR;

        /* FIND SOLUTION IN TERMS OF COUPLED VARIABLES */

        K=0;
        CALL MMULT(MODAL,HUC,H,N,K);

        /* FIND MINIMUM MSE */

        MSE=(AMP**2)*0.5;

        MMSE:
          DO I=1 TO N;
            MSE=MSE-MAUTO(I,I)*HUC(I)**2;
        END MMSE;

        /* OUTPUT RESULTS */

        H=HINIT-H;
        HUC=HIUC-HUC;
        PUT SKIP EDIT(N,DELTA,ALPHA,AMP)((2)F(5),(2)E(13,6));
        PUT SKIP EDIT(FREQ,T,SNR,MSE)((4)E(13,6));
        DO I=1 TO N;
          PUT SKIP EDIT(MAUTO(I,I),H(I),HUC(I),RXX(I))((4)E(13,6));
        END;
/*****************************************************/
/*                                                   */
/* MMULT MULTIPLIES AN N X N MATRIX BY A VECTOR      */
/*                                                   */
/*****************************************************/

MMULT:  PROC(MATRIX,VI,VR,N,K);
        DCL MATRIX(*,*) FLOAT BIN;
        DCL VI(*) FLOAT BIN;
        DCL VR(*) FLOAT BIN;
        DCL (I,J,N,K) FIXED BIN;
        MULT:
          DO I=1 TO N;
            VR(I)=0.0;
            DO J=1 TO N;
              IF K=0
                THEN VR(I)=VR(I)+MATRIX(I,J)*VI(J);
                ELSE VR(I)=VR(I)+MATRIX(J,I)*VI(J);
            END;
        END MULT;
        RETURN;
END MMULT;
        END ALLOC;
END EIGEN;
```

D.3 Program for Evaluation of Theoretical Bounds

The program described in this section was used for the evaluation of the theoretical bounds derived in Chapters II and III. In addition, the plots of the theoretical bounds and of the results of the results of the simulation were made using this program. A PDP 11/45 computer with storage tube graphics display was used and the data generated by the simulation program and the eigenvalue program were stored as data files on the PDP 11/45 magnetic disk for access by this program. The program was written in the FORTRAN IV language and operates in an iteractive fashion.

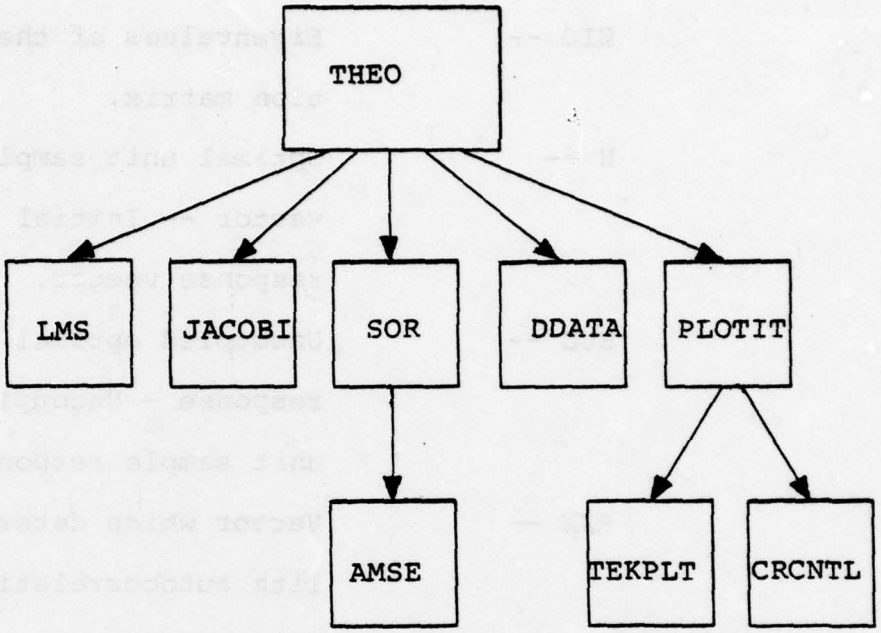
D.3.1 Program Structure

The structure of the program is illustrated in Figure D.3. Each program segment is described below:

1. THEO -- Main program for evaluation of theoretical bounds and display of data. Calls Subroutines LMS, JACOBI, SOR, DDATA and PLOTIT.

   INPUTS:

N --	Filter length
DELTA --	Delay value n_1.
ALPHA --	Autoregression parameter for noise sequence.
AMP --	Amplitude of signal.
FREQ --	Frequency of signal (radians).



D.3 Structure of Program THEO -- For the Computation of Theoretical Bounds and for Display of Data.

SNR -- Signal-to-noise ratio.

EMIN -- MSR produced by the otpimal filter.

EIG -- Eigenvalues of the autocorrelation matrix.

H -- Optimal unit sample response vector -- Initial unit sample response vector.

HUC -- Uncoupled optimal unit sample response - Uncoupled initial unit sample response.

RXX -- Vector which determines the Topelitz autocorrelation matrix.

OUTPUTS:

Same as inputs.

2. LMS -- Computes theoretical bound on MSE(n) for the LMS Gradient Algorithm.

INPUTS:

U -- μ value for LMS Gradient Method.

N -- Filter length.

NP -- Number of samples to be computed.

HUC -- Uncoupled optimal unit sample response - Uncoupled initial unit sample response.

EIG -- Eigenvalues of autocorrelation matrix.

EMIN -- MSE produced by the optimal filter.

I -- Row of array X in which results

are to be stored.

OUTPUTS:

X(I,·) -- Row of array containing samples of the theoretical bound for the LMS Gradient Algorithm.

3. JACOBI -- Computes theoretical bound on MSE(n) for the Jacobi Relaxation Method.

INPUTS:

B -- Parameter β for the Jacobi Relaxation Method.

N -- Filter length.

NP -- Number of samples to be computed.

HUC -- Uncoupled optimal unit sample response - Uncoupled initial unit sample response.

EIG -- Eigenvalues of the autocorrelation matrix.

EMIN -- MSE produced by the optimal filter.

I -- Row of the array X in which results are to be stored.

NS -- k value for the Variable Iteration Step Method.

OUTPUTS:

X(I,·) -- Row of array containing samples of the theoretical bound for the Jacobi Relaxation Method.

4. SOR -- Computes theoretical bound on MSE(n) for the SOR Algorithm. Calls function AMSE.

INPUTS:

B --	Parameter β for the SOR Algorithm.
N --	Filter length.
NP --	Number of samples to be computed.
H --	Optimal unit sample response - Initial unit sample response.
RXX --	Vector which determines the Topelitz autocorrelation matrix.
EMIN --	MSE produced by the optimal filter.
I --	Row of the array X in which results are to be stored.
NW --	ℓ value for the Variable Iteration Step Method.
NS --	k value for the Variable Iteration Step Method.

OUTPUTS:

X(I,·) --	Row of array containing samples of the theoretical bound for the SOR Algorithm.

5. DDATA -- Reads data file produced by the Simulation program.

INPUTS:

Filename for the data file containing the simulation results.

NC --	Particular variable which is desired.

I -- Row of the array X in which results are to be stored.

OUTPUTS:

X(I,·) -- Row of array containing the desired data from the results of the simulation.

6. PLOTIT -- Plots data on storage tube graphics terminal. Calls subroutines TEKPLT and CRCNTL.

INPUTS:

NP -- number of points to be plotted per curve.

NV -- Number of curves to be plotted on same graph.

NS -- NS=0 for manual scaling, NS=1 for automatic scaling.

PVAR -- Array containing indices of the rows of array X to be plotted.

X(I,J) -- Array containing data to be plotted.

NSTART -- Number of point at which plot is to begin.

NFIN -- Number of point at which plot is to end.

IDEV -- Unit number of graphics device.

YS -- Vertical scale factor if manual scaling used.

YINT -- Y-axis intercept if manual scaling used.

OUTPUTS:

Plot of the contents of the selected row(s) of the array X.

7. AMSE -- Computes MSE for the SOR Method given the value of the unit sample response vector.

INPUTS:

HW -- Unit sample response vector.

RXX -- Vector which determines Topelitz autocorrelation matrix.

EMIN -- MSE produced by optimal filter.

N -- Filter length.

OUTPUTS:

AMSE -- MSE using the unit sample response vector HW.

8. TEKPLT -- System subroutine for vector graphics.

INPUTS:

M -- M=1 for visible vector, M=0 for invisible vector.

IX1 -- X coordinate of first endpoint.

IY1 -- Y coordinate of first endpoint.

IX2 -- X coordinate of second endpoint.

IY2 -- Y coordinate of second endpoint.

IDEV -- Unit number of graphics device.

OUTPUTS:

Vector drawn from (IX1, IY1) to (IX2, IY2).

9. CRCNTL -- System subroutine for generating carriage control characters for graphics device.

INPUTS:

IDEV --	Unit number of graphics device.

OUTPUTS:

ISP --	Character for single space.
IDSP --	Character for double space.
IPLUS --	Character for +.
IDOLL --	Character for $.

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

D.3.2 Source Program

```
      INTEGER DELTA
      COMMON X(8,512)
      DIMENSION NVAR(8),NCRD(8),PUC(16),P(16),EIG(16),RXX(16)
      DATA BLANK/4H    /
C
C ASK FOR KEYBOARD DEVICE NUMBER
C
      WRITE(6,2600)
      READ(6,1000) IDEV
C
C ASSIGN DISK FILE CONTAINING EIGENVALUES TO UNIT 1
C
    5 WRITE(6,2000)
      PAUSE
      DEFINE FILE 1(18,8,U,INDX)
C
C READ EIGENVALUE DATA FROM DISK
C
      READ(1'1) RN,RDEL,ALPHA,AMP
      READ(1'2) FREQ,T,SNR,EMIN
      N=INT(RN)
      DELTA=INT(RDEL)
      DO 10 I=1,N
        K=I+2
        READ(1'K) EIG(I),P(I),PUC(I),RXX(I)
   10 CONTINUE
      END FILE 1
C
C WRITE EIGENVALUE DATA TO LINE PRINTER
C
      WRITE(5,2100) N,DELTA,ALPHA,AMP,FREQ,T,SNR,EMIN
      WRITE(5,2200)
      DO 15 I=1,N
        WRITE(5,2300) EIG(I),P(I),PUC(I),RXX(I)
   15 CONTINUE
      END FILE 5
C
C ASK WHICH CURVES TO BE GENERATED
C
   17 WRITE(6,2400)
      READ(6,1000) (NCRD(I),I=1,8)
C
C ASK # POINTS TO BE GENERATED
C
      WRITE(6,2500)
      READ(6,1100) NP
C
C CALL APPROPRIATE SUBROUTINES TO GENERATE CURVES
C
      DO 60 I=1,8
        IF(NCRD(I).EQ.0) GO TO 60
        K=NCRD(I)
        GO TO (20,30,40,50,55),K
   20   CALL LMS(I,N,NP,PUC,EIG,EMIN)
        GO TO 60
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
   30    CALL JACOBI(I,N,NP,HUC,EIG,RXX,EMIN)
         GO TO 60
   40    CALL SOR(I,N,NP,H,RXX,EMIN)
         GO TO 60
   50    CALL DDATA(I,NP)
         GO TO 60
   55    DO 57 IN=1,NP
           X(I,IN)=EMIN
   57    CONTINUE
   60 CONTINUE
C
C PLOT DATA
C
  300 WRITE(6,2650)
      READ(5,1000)(NVAR(I),I=1,8)
      WRITE(6,2700)
      READ(5,1100) NSTART,NFIN
      WRITE(6,2750)
      READ(5,1000) NS
      CALL PLOTIT(512,8,NS,NVAR,NSTART,NFIN,ICEV)
      WRITE(6,2800)
      READ(5,1000) NOPT
      IF(NOPT.EQ.1) GO TO 300
      WRITE(6,2650)
      READ(5,1000) NOPT
      IF(NOPT.EQ.1) GO TO 17
      WRITE(6,2900)
      READ(5,1000) NOPT
      IF(NOPT.EQ.1) GO TO 5
 1000 FORMAT(1X,8(I1,2X))
 1100 FORMAT(1X,I3,2X,I3)
 2000 FORMAT(1H ,'ASSIGN EIGENVALUE DISK FILE, EXAMPLE:'/1X,
     1       '$AS LB0:XXXXXX.XXX,1',/1X,'$CC')
 2100 FORMAT(1H ,1X,'N',2X,'DELTA',4X,'ALPHA',8X,'AMP',8X,'FREQ',
     1       10X,'T',10X,'SNR',8X,'EMIN',//1X,I3,1X,I4,1P6E12.4//)
 2200 FORMAT(1H ,2X,'EIGENVALUES',4X,'(INITIAL U-S RESP.-OPTIMAL',
     1' U-S RESP.)',8X,'RXX(1)',/1X,22X,'COUPLED',10X,'UNCOUPLED',//)
 2300 FORMAT(1H ,1X,1PE12.4,3(6X,1PE12.4))
 2400 FORMAT(1H ,'ENTER UP TO 8 CURVES TO BE GENERATED',/1X,
     1 '0=NO CURVE',/1X,'1=LMS GRADIENT',/1X,'2=JACOBI RELAXATION',
     2 /1X,'3=SOR METHOD',/1X,'4=DATA FILE FROM DISK',/1X,
     3 '5=MINIMUM MSE BOUND FOR SKEP',//1X,
     4 '( )( )( )( )( )( )( )( )')
 2500 FORMAT(1H ,'NUMBER OF POINTS TO BE GENERATED',/1X,'(   )')
 2650 FORMAT(1H ,'ENTER UP TO 8 VARIABLES FOR PLOTTING',/1X,
     1       '( )( )( )( )( )( )( )( )')
 2700 FORMAT(1H ,'ENTER STARTING AND FINISH POINTS FOR PLOT',/1X,
     1      'NSTART NFIN',/1X,'(   )(   )')
 2750 FORMAT(1H ,'ENTER 1 FOR AUTOSCALE',/1X,'( )')
 2800 FORMAT(1H ,'ENTER CONSOLE DEVICE (6=TEK, 9=DEKSCOPE)',/1X,
     1       '( )')
 2850 FORMAT(1H ,'ENTER 1 TO CONTINUE PLOTTING',/1X,'( )')
 2900 FORMAT(1H ,'ENTER 1 TO GENERATE NEW SET OF CURVES',/1X,'( )')
 2950 FORMAT(1H ,'ENTER 1 TO READ NEW EIGENVALUE DATA',/1X,'( )')
      END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
      SUBROUTINE LMS(T,N,NP,HLC,EIG,EMIN)
      COMMON X(8,512)
      DIMENSION EIG(N),HLC(N),A(32),EIG2(32)
C
C ASK VALUE OF MU
C
      WRITE(6,2000)
      READ(5,1000) U
C
C COMPUTE CONVERGENCE CURVE
C
      X(I,1)=EMIN
      DO 50 J=1,N
        EIG2(J)=(1.-2.*U*EIG(J))**2
        A(J)=EIG(J)*HLC(J)**2
        X(I,1)=X(I,1)+A(J)
   50 CONTINUE
      DO 100 K=2,NP
        X(I,K)=EMIN
        DO 75 J=1,N
          A(J)=A(J)*EIG2(J)
          X(I,K)=X(I,K)+A(J)
   75   CONTINUE
  100 CONTINUE
      RETURN
 1000 FORMAT(1X,F10.0)
 2000 FORMAT(1H ,'ENTER MU VALUE FOR LMS GRADIENT METHOD',/1X,
     1 '(           )')
      END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
      SUBROUTINE JACCEI(I,N,NP,FUC,EIG,RXX,EMIN)
      COMMON X(8,512)
      DIMENSION EIG(N),FUC(N),A(32),EIG2(32),RXX(N)
C
C  ASK VALUE OF BETA
C
      WRITE(6,2000)
      READ(6,1002) B
C
C  ASK NO. SAMPLES BETWEEN ITERATION STEPS
C
      WRITE(6,2200)
      READ(6,1100) NS
C
C  COMPUTE CONVERGENCE CURVE
C
      X(I,1)=EMIN
      DO 50 J=1,N
        EIG2(J)=(1.-B*EIG(J)/RXX(1))**2
        A(J)=EIG(J)*FUC(J)**2
        X(I,1)=X(I,1)+A(J)
   50 CONTINUE
      NCOUNT=0
      DO 100 K=2,NP
        NCOUNT=NCOUNT+1
        IF(NCOUNT.LT.NS) GO TO 80
        NCOUNT=0
        X(I,K)=EMIN
        DO 75 J=1,N
          A(J)=A(J)*EIG2(J)
          X(I,K)=X(I,K)+A(J)
   75   CONTINUE
        GO TO 100
   80   X(I,K)=X(I,K-1)
  100 CONTINUE
      RETURN
 1002 FORMAT(1X,F10.2)
 1100 FORMAT(1X,I2)
 2000 FORMAT(1H ,'ENTER BETA VALUE FOR JACOBI RELAXATION METHOD',
     1 '(          )')
 2200 FORMAT(1H ,'ENTER NO. SAMPLE STEPS PER ITERATION STEP FOR',
     1 ' JACOBI RELAXATION',/1X,'(  )')
      END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
      SUBROUTINE SCR(I,N,NP,H,RXX,EMIN)
      COMMON X(8,512)
      DIMENSION H(N),RXX(N),HW(32)
C
C ASK VALUE OF BETA
C
      WRITE(6,2000)
      READ(5,1100) BETA
C
C ASK NO. OF WEIGHTS UPDATED PER ITERATION STEP
C
      WRITE(6,2100)
      READ(5,1100) NW
C
C ASK NO. OF SAMPLES BETWEEN ITERATION STEPS
C
      WRITE(6,2200)
      READ(5,1100) NS
C
C LOAD WORKING ARRAY
C
      DO 100 J=1,N
        HW(J)=H(J)
  100 CONTINUE
C
C COMPUTE CONVERGENCE CURVE
C
      X(I,1)=AMSE(HW,RXX,EMIN,N)
      NCOUNT=0
      NST=1
C
      DO 500 M=2,NP
C
C CARRY OUT SCR METHOD WITH PARTIAL ITERATION STEPS
C
      NCOUNT=NCOUNT+1
      IF(NCOUNT.LT.NS) GO TO 400
      NCOUNT=0
      NUP=0
  250 IF(NST.GT.N) NST=1
      HW(NST)=(1.-BETA)*HW(NST)*RXX(1)/BETA
      DO 300 J=1,N
        IF(J.EQ.NST) GO TO 300
        K=IABS(J-NST)+1
        HW(NST)=HW(NST)-HW(J)*RXX(K)
  300 CONTINUE
      HW(NST)=HW(NST)*BETA/RXX(1)
      NST=NST+1
      NUP=NUP+1
      IF(NUP.LT.NW) GO TO 250
      X(I,M)=AMSE(HW,RXX,EMIN,N)
      GO TO 500
  400 X(I,M)=X(I,M-1)
  500 CONTINUE
      RETURN
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
1000 FORMAT(1X,F10.2)
1100 FORMAT(1X,I2)
2000 FORMAT(1H ,'ENTER BETA FOR SCR ALGORITHM',/1X,'(          )')
2100 FORMAT(1H ,'ENTER NO. WEIGHTS UPDATED PER ITERATION STEP FOR',
     1 ' SCR ALGORITHM',/1X,'(  )')
2200 FORMAT(1H ,'ENTER NO. SAMPLE STEPS PER ITERATION STEP FOR',
     1 ' SCR ALGORITHM',/1X,'(  )')
     END




     FUNCTION AMSE(HW,RXX,EMIN,N)
     DIMENSION HW(N),RXX(N)
     ASET=EMIN
     DO 200 J=1,N
       A=0.0
       DO 100 K=1,N
         L=IABS(J-K)+1
         A=A+RXX(L)*HW(K)
 100   CONTINUE
       ASET=ASET+A*HW(J)
 200 CONTINUE
     AMSE=ASET
     RETURN
     END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
      SUBROUTINE DDATA(I,NP)
      COMMON X(8,512)
      DIMENSION Y(11),Z(512)
C
C ASK FOR DISK FILE ASSIGNMENT
C
      WRITE(6,2000)
      PAUSE
      DEFINE FILE 2(512,22,U,INDEX)
C
C ASK WHICH COLUMN OF DATA FILE IS DESIRED
C
      WRITE(5,2100)
      READ(6,1000) ND
C
C READ DATA FILE
C
      DO 100 J=1,NP
        READ(2'J)(Y(K),K=1,11)
        Z(J)=Y(ND)
  100 CONTINUE
      END FILE 2
  150 DO 200 J=1,NP
        X(I,J)=Z(J)
  200 CONTINUE
      RETURN
 1000 FORMAT(1X,I1)
 2000 FORMAT(1H ,'ASSIGN DATA FILE TO BE PLOTTED TO UNIT 2, EXAMPLE'
     1 /1X,'$AS DK0:XXXXXX.XXX,2',/1X,'SCO')
 2100 FORMAT(1H ,'ENTER DATA RECORD TO BE PLOTTED',/1X,'1=SIGNAL',
     1'2=OPTIMAL SIGNAL ESTIMATE',/1X,'3=LMS SIGNAL ESTIMATE',/1X,
     2'4=JACOBI SIGNAL ESTIMATE',/1X,'5=SOR SIGNAL ESTIMATE',/1X,
     3'6=LEVINSON SIGNAL ESTIMATE',/1X,'7=MSE FOR OPTIMAL FILTER',
     4'8=MSE FOR LMS METHOD',/1X,'9=MSE FOR JACOBI METHOD',/1X,
     5'10=MSE FOR SOR METHOD',/1X,'11=MSE FOR LEVINSON ALGORITHM',
     6'( )')
      END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
      SUBROUTINE PLOTIT(NF,NV,NS,PVAR,NSTART,NFIN,ICEV)
      INTEGER PVAR
      COMMON X(8,512)
      DIMENSION XMAX(8),XMIN(8),XMEAN(8),PVAR(NV)
      CALL CRCNTL(ISF,ICSF,IPLUS,ICOLL,ICEV)
C
C   SCALE DATA
C
      NST=NSTART+1
      DO 30 J=1,NV
      XMEAN(J)=0.0
      XMAX(J)=0.0
      DO 10 I=NSTART,NFIN
      XMEAN(J)=XMEAN(J)+X(J,I)/FLOAT(NFIN-NSTART)
   10 XMAX(J)=AMAX1(XMAX(J),X(J,1))
      XMIN(J)=XMAX(J)
      DO 15 I=NSTART,NFIN
   15 XMIN(J)=AMIN1(XMIN(J),X(J,I))
      IF(XMAX(J)) 20,20,30
   20 XMAX(J)=XMIN(J)
      DO 25 I=NSTART,NFIN
   25 XMAX(J)=AMAX1(XMAX(J),X(J,I))
   30 CONTINUE
      K=1
      DO 32 J=1,NV
        IF((PVAR(J).GT.0).AND.(PVAR(J).LE.NV)) K=PVAR(J)
   32 CONTINUE
      XMAX1=XMAX(K)
      XMIN1=XMIN(K)
      DO 27 J=1,NV
      I=PVAR(J)
      IF((I.LT.1).OR.(I.GT.NV)) GO TO 27
      XMAX1=AMAX1(XMAX1,XMAX(I))
      XMIN1=AMIN1(XMIN1,XMIN(I))
   27 CONTINUE
      IF(NS.EQ.0) GO TO 50
      YINT=(XMAX1-XMIN1)/2.+XMIN1
   40 IF(ABS(XMAX1-YINT).NE.0.0) GO TO 45
      YS=1.
      GO TO 50
   45 YS=315./ABS(XMAX1-YINT)
   50 XS=900./FLOAT(NFIN-NSTART)
      IF(NS.NE.0) GO TO 65
      CALL TEKPLT(0,0,0,0,0,ICEV)
      WRITE(6,2300)
      DO 60 J=1,NV
      I=PVAR(J)
      IF((I.LT.1).OR.(I.GT.NV)) GO TO 60
      WRITE(6,2400) I,XMAX(I),XMIN(I),XMEAN(I)
   60 CONTINUE
      WRITE(6,2500)
      READ(6,1100) YS
      YS=45./YS
      WRITE(6,2600)
      READ(6,1100) YINT
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
   65 CALL TEKPLT(0,0,0,0,0,IDEV)
      CALL TEKPLT(2,50,10,850,10,IDEV)
      CALL TEKPLT(2,50,640,850,640,IDEV)
      CALL TEKPLT(2,50,10,50,640,IDEV)
      CALL TEKPLT(2,850,10,850,640,IDEV)
      CALL TEKPLT(2,50,325,850,325,IDEV)
      DO 100 J=1,15
      IY=45.*(15-J)+10.
      CALL TEKPLT(2,50,IY,60,IY,IDEV)
      CALL TEKPLT(2,840,IY,850,IY,IDEV)
      XP=YINT+(45./YS)*FLOAT(8-J)
      IY=IY-6
      CALL TEKPLT(3,855,IY,0,0,IDEV)
  100 WRITE(IDEV,2002) IPLUS,XP
      DO 150 J=1,15
      IX=50*J+50
      CALL TEKPLT(2,IX,10,IX,20,IDEV)
      CALL TEKPLT(2,IX,320,IX,330,IDEV)
  150 CALL TEKPLT(2,IX,630,IX,640,IDEV)
      DO 800 N=1,NV
      J=PVAR(N)
      IF((J.LT.1).OR.(J.GT.NV)) GO TO 800
      IYY=(X(J,NSTART)-YINT)*YS+325.
      CALL TEKPLT(3,5,IYY,5,IYY,IDEV)
      WRITE(IDEV,2700) IPLUS,J
      DO 800 I=NST,NFIN
      RIY1=(X(J,I-1)-YINT)*YS+325.
      RIY2=(X(J,I)-YINT)*YS+325.
      IF(ABS(RIY1).GT.3.E+4) RIY1=3.E+4
      IF(ABS(RIY2).GT.3.E+4) RIY2=3.E+4
      IY1=INT(RIY1)
      IY2=INT(RIY2)
      K=I-NST
      IX1=(K*XS)+50
      IX2=(K+1)*XS+50
      CALL TEKPLT(2,IX1,IY1,IX2,IY2,IDEV)
  800 CONTINUE
 1000 FORMAT(1X,I5)
 1100 FORMAT(1X,F10.0)
 2000 FORMAT(A2,1PE10.3)
 2300 FORMAT(/1X,'VARIABLE',5X,'YMAX',11X,'YMIN',11X,'MEAN'//)
 2400 FORMAT(1X,I5,3(5X,1PE10.3))
 2500 FORMAT(//1X,'ENTER Y-AXIS SCALE'/1X,'(          )')
 2600 FORMAT(//1X,'ENTER X-AXIS POSITION'/1X,'(          )')
 2700 FORMAT(A2,I2)
 2200 FORMAT(///)
      RETURN
      END
```

D.4 Program for Computation of DFT's of Signal Estimate Records

This program was used for the computation and plotting of the DFT of the signal and signal estimate records as described in Section 6.5. The last 20 samples of the data record from the results of the simulation program are repeated 51 times for a total of 1020 samples. The 1024 point DFT is then taken of the resulting sequence. This procedure was followed since the FFT program is a radix-2 algorithm and it was desired to reduce the effect of windowing present in the output. The program was written in Fortran IV and run on a PDP 11/45 computing system in an interactive fashion.

D.4.1 Program Structure

The structure of the program for computing and plotting the magnitude of the DFT is illustrated in Figure D.4. The program segments are described on the following pages.

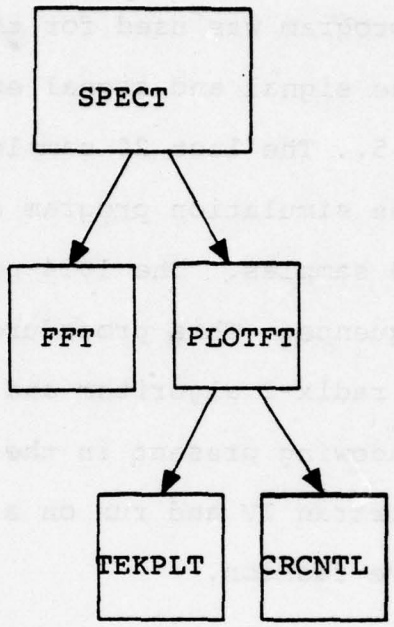


Figure D.4 Structure of Program SPECT -- for the computation of the Magnitude of the DFT of the Signal Estimates.

1. SPECT -- Main Program. Calls program FFT and PLOTFT.

   INPUTS:

IDEV --	Logical unit number of graphics terminal.
NP --	Number of records in data file.
NR --	Identifier for the particular variable desired.

   Data file name assigned through monitor command.

   OUTPUTS:

Z --	Array containing magnitude of DFT.

2. FFT -- Computes the DFT using the FFT algorithm by Brenner.

   INPUTS:

DATA --	Array containing real and imaginary parts of data stored in alternating locations.
NN --	Number of DFT points desired.
ISIGN --	-1 for transform, +1 for inverse transform.

   OUTPUTS:

DATA --	Array containing real and imaginary parts of result stored in alternating locations.

3. PLOTFT -- Plots magnitude of DFT, calls subroutines TEKPLT and CRCNTL.

   INPUTS:

X --	Data to be plotted.
NP --	Number of points to be plotted.

NV -- NV=1.

NS -- NS=0 for manual scaling, NS=1 for autoscaling.

NSTART -- Location in array X at which plot is to begin.

NFIN -- Location in array X at which plot is to end.

IDEV -- Logical unit number of graphics device.

OUTPUTS:

Plot of the Array X.

4. TEKPLT -- System subroutine for vector graphics. See section D.3.2 for description of inputs and outputs.

5. CRCNTL -- System subroutine for generating carriage control characters. See Section D.3.2 for description of inputs and outputs.

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
C
C PROGRAM FOR SPECTRAL ANALYSIS OF ALE DATA
C SDH 6/22/78
C
      DIMENSION X(2048),REC(11),Y(20),Z(1024)
C
C ASK KEYBOARD DEVICE NUMBER
C
      WRITE(6,2000)
      READ(6,1000) IDEV
C
C DEFINE DATA FILE ON UNIT 2
C
   20 WRITE(6,2100)
      PAUSE
      DEFINE FILE 2(256,22,U,INDX)
C
C READ NUMBER OF RECORDS IN FILE
C
      WRITE(6,2200)
      READ(6,1100) NF
C
C READ DATA FILE DESIRED
C
      WRITE(6,2300)
      READ(6,1000) NN
C
C READ DATA
C
      DO 100 I=1,20
        K=NF-20+I
        READ(2'K) (REC(J),J=1,11)
        Y(I)=REC(NR)
  100 CONTINUE
      END FILE 2
C
C ZERO ARRAY
C
      DO 50 I=1,2048
        X(I)=0.0
   50 CONTINUE
C
C LOAD ARRAY FOR FFT PROCESSING
C
      DO 200 I=1,51
        DO 150 J=1,20
          K=40*(I-1)+2*J-1
          X(K)=Y(J)
  150   CONTINUE
  200 CONTINUE
C
C COMPUTE FOURIER TRANSFORM
C
      CALL FOURI(X,1024,-1)
C
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
C TAKE MAGNITUDE OF RESULT
C
      DO 250 I=1,1024
        K=2*I-1
        Z(I)=SQRT(X(K)**2+X(K+1)**2)
  250 CONTINUE
C
C PLOT RESULTS
C
C ASK FOR AUTO SCALE
C
  300 WRITE(6,2420)
      READ(6,1020) NS
C
C PLOT SPECTRUM
C
      CALL PLOTFT(Z,1024,1,NS,1,1024,ICDEV)
      WRITE(6,2500)
      READ(6,1000) NST
      IF(NST.EQ.1) GO TO 300
      WRITE(6,2600)
      READ(6,1000) NST
      IF(NST.EQ.0) GO TO 20
 1000 FORMAT(1X,I1)
 1100 FORMAT(1X,I3)
 2000 FORMAT(1H ,'ENTER KG: CEVICE (6=TEK, 9=DEKSCOPE)',/1X,'( )')
 2100 FORMAT(1H ,'ASSIGN DATA FILE TO UNIT 2')
 2200 FORMAT(1H ,'NUMBER OF RECORDS IN FILE',/1X,'(   )')
 2300 FORMAT(1H ,'ENTER DATA RECORD DESIRED',/1X,'1=SIGNAL',/1X,
     1 '2=OPTIMAL SIGNAL ESTIMATE',/1X,'3=LMS SIGNAL ESTIMATE',/1X,
     2 '4=JACOBI SIGNAL ESTIMATE',/1X,'5=SOR SIGNAL ESTIMATE',/1X,
     3 '6=LEVINSON SIGNAL ESTIMATE',//1X,'( )')
 2500 FORMAT(1H ,'ENTER 1 TO PLOT AGAIN',/1X,'( )')
 2600 FORMAT(1H ,'ENTER 1 TO EXIT FROM PGM',/1X,'( )')
 2420 FORMAT(1H ,'ENTER 1 FOR AUTO SCALE',/1X,'( )')
      END
```

```
      SUBROUTINE PLOTFT(X,NP,NV,NS,NSTART,NFIN,ICEV)
      DIMENSION X(1)
      CALL CRCNTL(ISP,ICSP,IPLUS,ICOLL,ICEV)
C
C   SCALE DATA
C
      NST=NSTART+1
      XMEAN=0.0
      XMAX=0.0
      DO 10 I=NSTART,NFIN
      XMEAN=XMEAN+X(I)/FLOAT(NFIN-NSTART)
   10 XMAX=AMAX1(XMAX,X(I))
      XMIN=XMAX
      DO 15 I=NSTART,NFIN
   15 XMIN=AMIN1(XMIN,X(I))
      IF(XMAX) 20,20,30
   20 XMAX=XMIN
      DO 25 I=NSTART,NFIN
   25 XMAX=AMAX1(XMAX,X(I))
   30 CONTINUE
      XMAXI=XMAX
      XMINI=XMIN
      IF(NS.EQ.0) GO TO 50
      YINT=(XMAXI-XMINI)/2.+XMINI
   40 IF(ABS(XMAXI-YINT).NE.0.0) GO TO 45
      YS=1.
      GO TO 50
   45 YS=315./ABS(XMAXI-YINT)
   50 XS=800./FLOAT(NFIN-NSTART)
      IF(NS.NE.0) GO TO 55
      CALL TEKPLT(0,0,0,0,0,IDEV)
      WRITE(6,2300)
      WRITE(6,2400) I,XMAX,XMIN,XMEAN
      WRITE(6,2500)
      READ(6,1100) YS
      YS=45./YS
      WRITE(6,2600)
      READ(6,1100) YINT
   65 CALL TEKPLT(0,0,0,0,0,IDEV)
      CALL TEKPLT(2,50,10,850,10,IDEV)
      CALL TEKPLT(2,850,10,850,640,IDEV)
      DO 100 J=1,15
      IY=45.*(15-J)+10.
      CALL TEKPLT(2,640,IY,850,IY,IDEV)
      XP=YINT+(45./YS)*FLOAT(8-J)
      IY=IY-6
      CALL TEKPLT(3,855,IY,0,0,IDEV)
  100 WRITE(IDEV,2000) IPLUS,XP
      IYY=(X(NSTART)-YINT)*YS+325.
      CALL TEKPLT(3,5,IYY,5,IYY,IDEV)
      WRITE(IDEV,2700) IPLUS,J
      DO 800 I=NST,NFIN
      RIY1=(X(I-1)-YINT)*YS+325.
      RIY2=(X(I)-YINT)*YS+325.
      IF(ABS(RIY1).GT.3.E+4) RIY1=3.E+4
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
      IF(ABS(RIY2).GT.3.E+4) RIY2=3.E+4
      IY1=INT(RIY1)
      IY2=INT(RIY2)
      K=I-NS
      IX1=(K*XS)+50
      IX2=(K+1)*XS+50
      CALL TEKPLT(2,IX1,IY1,IX2,IY2,ICLR)
  800 CONTINUE
 1000 FORMAT(1X,I5)
 1100 FORMAT(1X,F10.0)
 2000 FORMAT(A2,1PE10.3)
 2300 FORMAT(//1X,'VARIABLE',5X,'YMAX',11X,'YMIN',11X,'MEAN'//)
 2400 FORMAT(1X,I5,3(5X,1PE12.3))
 2500 FORMAT(//1X,'ENTER Y-AXIS SCALE'/1X,'(          )')
 2600 FORMAT(//1X,'ENTER X-AXIS POSITION'/1X,'(          )')
 2700 FORMAT(A2,I2)
 2200 FORMAT(///)
      RETURN
      END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
C
C   THIS IS THE FOURIER TRANSFORM FROM MIT BY BRENNER.
C
C   DATA IS STORED IN THE 2*NN X 1 ARRAY DATA(I) BY
C   ALTERNATING REAL AND IMAGINARY PARTS WITH ODD
C   SUBSCRIPTED ELEMENTS CONTAINING REAL PART AND
C   THE FOLLOWING EVEN SUBSCRIPTED ELEMENT CONTAINING
C   THE CORRESPONDING IMAGINARY PART.  NN IS THE
C   LENGTH OF THE TRANSFORM AND MUST BE A INTEGER
C   POWER OF 2.  ISIGN SHOULD BE EQUAL TO -1 FOR A
C   FORWARD TRANSFORM AND +1 FOR AN INVERSE TRANSFORM.
C   THE RESULTING TRANSFORM IS RETURNED IN THE ARRAY
C   DATA(I) STORED IN THE SAME FASHION AS DESCRIBED ABOVE.
C
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
      DIMENSION DATA(1)
      N=2*NN
      J=1
      DO 5 I=1,N,2
        IF(I-J) 1,2,2
    1   TEMPR=DATA(J)
        TEMPI=DATA(J+1)
        DATA(J)=DATA(I)
        DATA(J+1)=DATA(I+1)
        DATA(I)=TEMPR
        DATA(I+1)=TEMPI
    2   M=N/2
    3   IF(J-M) 5,5,4
    4   J=J-M
        M=M/2
        IF(M-2) 5,3,3
    5   J=J+M
      MMAX=2
    6 IF(MMAX-N) 7,12,10
    7 ISTEP=2*MMAX
      DO 9 M=1,MMAX,2
        THETA=3.14159265358*FLOAT(ISIGN*(M-1))/FLOAT(MMAX)
        WR=COS(THETA)
        WI=SIN(THETA)
        DO 8 I=M,N,ISTEP
          J=I+MMAX
          TEMPR=WR*DATA(J)-WI*DATA(J+1)
          TEMPI=WR*DATA(J+1)+WI*DATA(J)
          DATA(J)=DATA(I)-TEMPR
          DATA(J+1)=DATA(I+1)-TEMPI
          DATA(I)=DATA(I)+TEMPR
          DATA(I+1)=DATA(I+1)+TEMPI
    8   CONTINUE
    9 CONTINUE
      MMAX=ISTEP
      GO TO 6
   10 RETURN
      END
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

REFERENCES

1. N. Wiener, Extrapolation, Interpolation and Smoothing of Stationary Time Series, J. Wiley and Sons, New York, 1949.

2. A. Papolous, Signal Analysis, McGraw-Hill, New York, 1977.

3. A. Papolous, Probability, Random Variables and Stocastic Processes, McGraw-Hill, New York, 1965.

4. D. Luenberger, Introduction to Linear and Non-Linear Programming, Addison-Wesley, Menlo Park, Calif., 1973.

5. W. Feller, An Introduction to Probability Theory and Its Applications, Vol. II, John Wiley and Sons, 1966.

6. B. Widrow, "Adaptive Filters," in Aspects of Network and System Theory, Part IV, R.E. Kalman and N. DeClaris, Editors, Holt, Rinehart and Winston, New York, 1971.

7. Widrow, et al., "Adaptive Noise Cancelling: Principles and Applications", Proceedings of the IEEE, Vol. 63, No. 12, December 1975, pp. 1672-1716.

8. J. Treichler, "The Spectral Line Enhancer -- The Concept, an Implementation, and an Application," Ph.D. Dissertation, Stanford University, Stanford, Calif., 1977.

9. B.S. Atal and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoustical Society of America, Vol. 50, No. 2, 1971, pp. 637-655.

10. J.R. Welch and J.D. Oetting, "Formant Extraction Hardware Using Adaptive Linear Predictive Coding," Proceedings National Telecommunications Conference, Paper 29C, November 1973.

11. R.J. Wang and S. Treitel, "The Determination of Digital Wiener Filters by Means of Gradient Methods," Geophysics, Vol. 38, No. 2, April 1973, pp. 310-326.

12. J.D. Markel and A.H. Gray, Jr., "An Autocorrelation Equation as Applied to Speech Analysis," IEEE Transactions on

Audio and Electroacoustics, Vol. AU-21, No. 2, April 1973, pp. 69-79.

13. B. Widrow, et al., "Adaptive Antenna Systems," Proceedings of IEEE, Vol. 55, No. 12, December 1967, pp. 2143-2159.

14. H.R. Schwarz, H. Rutishauser, E. Stiefel, Numerical Analysis of Symmetric Matrices, Prentice-Hall, 1973.

15. S.D. Conte and C. de Boor, Numerical Analysis: An Algorithmic Approach, McGraw-Hill, New York, 1972.

16. G.M. Jenkins and D.G. Watts, Spectral Analysis and its Applications, Holden-Day, San Francisco, 1968.

17. E.J. Hannan, Time Series Analysis, Wiley, New York, 1962.

18. J. Bendat and A.G. Piersol, Random Data: Analysis, Measurement and Procedures, Wiley-Interscience, 1971.

19. Noel Gastinel, Linear Numerical Analysis, Academic Press, New York, 1970.

20. D.M. Young, Iterative Solution of Larger Linear Systems, Academic Press, New York, 1971.

21. A.V. Oppenheim and R.W. Shaffer, Digital Signal Processing, Prentice-Hall, Englewood Cliffs, N.J., 1975.

22. V. Conrad and Y. Wallach, "Iterative Solution of Linear Equations on a Parallel Processor System," IEEE Transactions on Computers, Vol. C-26, No. 9, September 1977.

23. N. Levinson, "The Wiener RMS Error Criterion in Filter Design and Prediction," J. Math. and Physics, Vol. 25, No. 4, January 1947, pp. 261-278.

24. W.F. Trench, "Weighting Coefficients for the Prediction of Stationary Time Series from the Finite Past," SIAM J. Applied Math., Vol. 15, No. 6, November 1967, pp. 1502-1510.

25. S. Zohar, "The Solution of Topelitz Set of Linear Equations," J. of the Association for Computing Machinery, Vol. 12, No. 2, April 1974, pp. 272-276.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR-15	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Adoptive Linear Estimation Algorithms Applied to Spectral Line Enhancement		5. TYPE OF REPORT & PERIOD COVERED Technical 6. PERFORMING ORG. REPORT NUMBER TR-15
7. AUTHOR(s) Stephen D. Huffman		8. CONTRACT OR GRANT NUMBER(s) N000 14-75-C-0191
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering Duke University Durham, North Carolina 27706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N Subelement 31-Oceanography Task Area 386-000 Acoustics Work Unit 386-907
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE August 1978 13. NUMBER OF PAGES 312
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Signal Processing	Underwater Acoustics	Adaptive Line Enhancer
Estimation	Underwater Sound	Adaptive Algorithm
Estimation Theory	Adaptive Signal Processing	LMS Gradient Algorithm
Detection	Adaptive Processing	Noise Cancelling

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The estimation of a signal in the presence of an additive noise process is a basic problem in communications theory. When the processor chosen to carry out the estimation task is a linear digital filter with a finite unit sample response, the design problem is to determine the unit sample response which will produce the best possible estimate of the signal according to some criterion. If the criterion chosen as a measure of optimality is the mean-square error of estimation and if the statistics of the signal and noise are known, the optimal unit sample response can be found by the solution of a system of linear

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. equations. The resulting processor is, then, an example of the classical Wiener filter.

When the statistics of the signal and noise are not known a priori, the unit sample response of the Wiener filter cannot be determined and some method for approximating the optimal filter is required. One approach to this problem is to use an adaptive processor which attempts to "learn" the characteristics of the signal and the noise and to use this information to obtain an approximation to the optimal filter.

In this report, the performance characteristics of the LMS Gradient Algorithm, two Adaptive Fixed-Point Iteration Algorithms and of a non-iterative method based on Levinson's Algorithm are considered for the case where an adaptive algorithm is used to determine the unit sample response for a system which attempts to discriminate between the signal and noise processes on the basis of bandwidth. Such a system is often referred to as a Spectral Line Enhancer. Theoretical bounds on the mean-square error as a function of the time index n are derived for each of the three iterative methods. A comparison of these bounds is made for the case where the input data to the Spectral Line Enhancer is composed of a single sinusoid of random phase in the presence of an additive autoregressive noise sequence. The results of extensive computer simulations of the adaptive algorithms considered are used to determine the usefulness of the theoretical bounds and to make comparisons of the performance of the four methods.

UNCLASSIFIED

DISTRIBUTION LIST

Office of Naval Research
800 N. Quincy Street
Arlington, Virginia 22217
Attn: Code 222 — 2
102 OS — 1
480 — 1

Director
Naval Research Laboratory
Technical Information Division
4555 Overlook Avenue S.W.
Washington, D.C. 20375 — 6

Director
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, California 91106 — 1

Office of Naval Research
San Francisco Area Office
760 Market Street Room 447
San Francisco, California 94102 — 1

Director
Office of Naval Research Branch Office
495 Summer Street
Boston, Massachusetts 02210 — 1

Office of Naval Research
New York Area Office
207 West 24th Street
New York, New York 10011 — 1

Commanding Officer
Office of Naval Research Branch Office
Box 39
FPO New York 09510 — 1

Director
Office of Naval Research Branch Office
536 South Clark Street
Chicago, Illinois 60605 — 1

Commander
Naval Surface Weapons Center
Acoustics Division
Silver Spring, Maryland 20910
Attn: Dr. Zaka Slawsky — 1

Officer in Charge
Naval Ship Research & Development Center
Annapolis Laboratory
Annapolis, Maryland 21402 — 1

Commander
Naval Sea Systems Command
Department of the Navy
Washington, D.C. 20362
Attn: SEA 037 — 1
Carey Smith, 06H1 — 1
David F. Bolka, 06H2

Commanding Officer
Fleet Numerical Weather Central
Monterey, California 93940 — 1

Defense Documentation Center
Cameron Station
Alexandria, Virginia 22314 — 12

Director of Navy Laboratories
Chief of Naval Material
2211 Jefferson Davis Highway
Crystal Plaza #5
Arlington, Virginia 20360
Attn Dr. James Probus NAVMAT 03L — 1

Commander
Naval Electronic Systems Command
2511 Jefferson Davis Highway
Arlington, Virginia 20360
Attn: CDR A. R. Miller NAVELEX 320 — 1

Commander
Naval Ship Research & Development Center
Department of the Navy
Bethesda, Maryland 20084
Attn: Mr. Craig Olson — 1
Unclassified Library — 1

Chief of Naval Operations
Room 4D518, Pentagon
Washington, D.C. 20350
Attn: CAPT A. H. Gilmore — 1

Commander
Naval Ocean Systems Center
Department of the Navy
San Diego, California 92132
Attn: Dr. Dan Andrews — 1
Mr. Henry Aurand — 1
Dr. Dean Hanna — 1

Superintendent
Naval Research Laboratory
Underwater Sound Reference Division
P.O. Box 8337
Orlando, Florida 32806 1

Commanding Officer
Naval Underwater Systems Center
New London Laboratory
New London, Connecticut 06320
Attn: Dr. A. Nuttall 1
Mr. A. Ellinthorpe 1
Dr. D. M. Viccione 1

Commander
Naval Air Development Center
Department of the Navy
Warminster, Pennsylvania 18974
Attn: Unclassified Library 1

Superintendent
Naval Postgraduate School
Monterey, California 93940
Attn: Unclassified Library 1

Commanding Officer
Naval Coastal Systems Laboratory
Panama City, Florida 32401
Attn: Unclassified Library 1

Commanding Officer
Naval Underwater Systems Center
Newport Laboratory
Newport, Rhode Island 02840
Attn: Unclassified Library 1

Superintendent
U.S. Naval Academy
Annapolis, Maryland 21402
Attn: Library 1

Commanding Officer
Naval Intelligence Support Center
4301 Suitland Road
Suitland, Maryland 20390
Attn: Dr. Johann Martinek 1
Mr. E. Bissett 1

Commander
Naval Sea Systems Command
Washington, D.C. 20362
Attn: Unclassified Library, SEA 03E 1

Office of the Assistant Secretary of the Navy
for Research, Engineering & Systems
Room 4E732, Pentagon
Washington, D.C. 20350
Attn: Mr. Gerald Cann 1

Special Assistant for ASW
Office of the Assitant Secretary of the Navy
for Research, Engineering & Systems
Washington, D.C. 20350
Attn: Dr. D. Hyde 1

Dr. Melvin J. Jacobson
Rennselaer Polytechnic Institute
Troy, New York 12181 1

Dr. Charles Stutt
General Electric Company
P.O. Box 1088
Schenectady, New York 12301 1

Dr. Alan Winder
MSB Systems, Inc.
25 Sylvan Road South
West Point, Connecticut 06880 1

Dr. T. G. Birdsall
Cooley Electronics Laboratory
University of Michigan
Ann Arbor, Michigan 48105 1

Dr. Harry DeFerrari
University of Miami
Rosentiel School of Marine & Atmospheric Sciences
4600 Rickenbacker Causeway
Miami, Florida 33149 1

Mr. Robert Cunningham
Bendix Electronics Center
15825 Roxford Street
Sylmar, California 91342 1

Dr. Stephen Wolff
Johns Hopkins University
Baltimore, Maryland 21218 1

Dr. M. A. Basin
S.D.P., Inc
15250 Ventura Boulevard
Suite 518
Sherman Oaks, California 91403 1

Dr. Walter Duing
University of Miami
Rosentiel School of Marine & Atmospheric Sciences
4600 Rickenbacker Causeway
Miami, Florida 33149 1

Dr. David Middleton
127 East 91st Street
New York, New York 10028 1

Dr. Donald W. Tufts
University of Rhode Island
Kingston, Rhode Island 02881 1

Dr. Loren Nolte
Duke University
Department of Electrical Engineering
Durham, North Carolina 27706 1

Mr. S. W. Autrey
Hughes Aircraft Company
P.O. Box 3310
Fullerton, California 92634 1

Dr. Thomas W. Ellis
Texas Instruments, Inc
13500 North Central Expressway
Dallas, Texas 75231 1

Dr. Terry Ewart
Applied Physics Laboratory
University of Washington
1013 Northeast Fortieth Street
Seattle, Washington 98195 1

Institute for Acoustical Research
Miami Division of the Palisades Geophysical Institute
615 S.W. 2nd Avenue
Miami, Florida 33130
Attn: Mr. M. Kronengold 2
Dr. J. Clark
Dr. W. Jobst
Dr. S. Adams

Mr. Carl Hartdegen
Palisades Geophysical Institute
Sofar Station
FPO New York 09560 — 1

Mr. Charles Loda
Institute for Defense Analyses
400 Army-Navy Drive
Arlington, Virginia 22202 — 1

Mr. Beaumont Buck
Polar Research Laboratory
123 Santa Barbara Avenue
Santa Barbara, California 93101 — 1

Dr. M. Weinstein
Underwater Systems, Inc
8121 Georgia Avenue
Silver Spring, Maryland 20910 — 1

Dr. Thomas G. Kincaid
General Electric Company
P.O. Box 1088
Schenectady, New York 12301 — 1

Applied Research Laboratories
University of Texas at Austin
P.O. Box 8029
10000 FM Road
Austin, Texas 78712
Attn: Dr. Lloyd Hampton — 2
Dr. Charles Wood

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543
Attn: Dr. Paul McElroy — 1
Dr. R. Porter
Dr. R. Spindel

Dr. John Bouyoucos
Hydroacoustics, Inc
321 Northland Avenue
P.O. Box 3818
Rochester, New York 14610 — 1

Systems Control, Inc
260 Sheridan Avenue
Palo Alto, California 94306
Attn: Mr. Robert Baron — 1

Atlantic Oceanographic & Meteorological Laboratories
15 Rickenbacker Causeway
Miami, Florida 33149
Attn: Dr. John Proni 1

Dr. C. N. K. Mooers
University of Delaware
Newark, Delaware 19711 1

Westinghouse Electric Corporation
Advanced Development Program
Marketing Department MS 227
P.O. Box 746
Baltimore, Maryland 21203
Attn: F. J. Frissyn 1

Oak Ridge National Laboratory
Union Carbide Corporation
Nuclear Division
P.O. Box X
Oak Ridge, Tennessee 37830 1

Professor Neil Bershad
University of California, Irvine
School of Engineering
Irvine, California 92664 1