DATE DUE	DATE DUE	DATE DUE
[illegible]		

ALGORITHMS AND ARCHITECTURES FOR ADAPTIVE SET MEMBERSHIP-BASED SIGNAL PROCESSING

By

Souheil Farah Odeh

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

1990

ABSTRACT

ALGORITHMS AND ARCHITECTURES FOR ADAPTIVE SET MEMBERSHIP-BASED SIGNAL PROCESSING

By

Souheil Farah Odeh

This research is concerned with a class of *set membership (SM)* algorithms for estimating the parameters of linear system or signal models in which the error sequence is pointwise "energy bounded." Specifically, it is focused on the *set membership weighted recursive least squares (SM-WRLS)* algorithm which works with bounding hyperellipsoidal regions to describe the solution sets which are a consequence of the error bounds. SM-WRLS is based on the familiar WRLS algorithm with the SM considerations handled through a special weighting strategy. The original version of SM-WRLS is applicable to real scalar data. In this work, a generalized SM-WRLS algorithm that can handle complex vector-input vector-output data streams is developed which extends the applicability of this algorithm to virtually any signal processing problem involving parametric models. Further, a significant reduction in computational complexity can be achieved by employing a "suboptimal" test for information content in an incoming equation. This new strategy can be applied to virtually any version of the SM-WRLS algorithm to improve the computational complexity. The suboptimal check is argued to be a useful determiner of the ability of incoming data to shrink the ellipsoid.

The "unmodified" SM-WRLS algorithm has inherent adaptive capabilities in its own right. However, it is not possible to depend upon this algorithm to *reliably* behave in an adaptive manner, particularly in cases of quickly varying system dynamics. In

this work, *explicitly* adaptive SM-WRLS algorithms are developed. Adaptation is incorporated into SM-WRLS in a very general way by introducing a flexible mechanism by which the algorithm can forget the influence of past data. The general formulation permits the extension of SM-WRLS to a wide range of adaptation strategies.

The various SM-WRLS developments are tested on models derived from real speech data. Simulation results are presented which illustrate important points about the various methods and show that the adaptive algorithms yield accurate estimates using very few of the data and quickly adapt to fast variations in the signals dynamics. It is also significant that in preliminary experiments, most of the SM-WRLS algorithms are found to be robust in small (16-bit) wordlength environments.

Finally, a parallel architecture is developed that implements the various SM-WRLS algorithms in $\mathcal{O}(m)$ floating point operations per equation using $\mathcal{O}(m)$ cells, where m represents the number of parameters estimated. A detailed analysis of the computational complexity issues is carried out.

To my parents

Farah and Georgette Odeh

for their love, support, and sacrifice

Acknowledgments

I am gratefully indebted to my thesis advisor, Professor John R. Deller, Jr., for his invaluable guidance and generous support throughout the course of this research.

I would like to express my gratitude to all the members in my Ph.D. guidance committee, Professor Donnie K. Reinhard, Professor Lionel M. Ni, Professor Majid Nayeri, and Professor Paul M. Parker, for their time and effort in reviewing and discussing my dissertation. The late Professor Harriett B. Rigas was also an important source of inspiration and guidance throughout my education, especially in difficult times.

I would also like to give a special thanks to everyone in my family for their continuous love, understanding, and encouragement.

I also gratefully acknowledge the partial support of this work by a grant from the Whitaker Foundation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 General Objectives and Scope

This research is concerned with techniques for estimating the parameters of linear system or signal models. *Set membership (SM)* identification refers to a class of estimation techniques that uses certain *a priori* knowledge about a linear parametric model to constrain the solutions to certain sets. Based on certain set-theoretic checking criteria, SM algorithms select and use only the useful data to update the parameter estimates and refine the "membership sets" to which the true parameters must belong. When data do not help refine these membership sets, the effort of updating the parameter estimates at those points can be avoided. The power of SM algorithms becomes more apparent when implemented using parallel architectures due to the significant reduction in the computational complexity. Because of their strong potential for application and theoretical development in virtually any signal processing problem involving parametric models (such as speech recognition, image processing, beamforming, spectral estimation, and neural networks), SM algorithms have been the subject of intense research effort in recent years [1] - [26]. Much of the recent work on parametric models is closely related to previous papers by Schweppe [27], Witsenhausen [28], and Bertsekas and Rhodes [29] which study state space systems in the control and systems science domains.

The most widely studied class of SM algorithms involves the case in which the error sequence, say $v(n)$, is pointwise "energy bounded,"

$$\gamma(n)v^2(n) < 1 \tag{1.1}$$

where the sequence $\gamma(n)$ is known or can be estimated from the data. It is this problem with which this research is concerned. (Other interesting variations involve stability constraints [30], and other noise parameter bounds [31, 32]. Veres and Norton [33] have also investigated the effects of error bounding on model structure identification.) In addition to the many advantages of the algorithms to be developed from this information, the constraint (1.1) minimizes the necessary knowledge of the input. In particular, it is not necessary to know the form of the density function for $v(n)$.

Constraints of form (1.1), in conjunction with the model and data, imply pointwise "hyperstrip" regions of possible parameter sets in the parameter space which, when intersected over a given time range, usually form convex polytopes of permissible solutions for the "true" parameters. While exact descriptions of these polytopes are possible [1, 10, 11, 12, 17, 21, 24], algorithms of much lower complexity have been developed which work with a bounding hyperellipsoid, a tight superset of the polytope [5, 6, 7, 14, 20]. Such an "optimal bounding ellipsoid" (OBE) algorithm is the focus of this research. Recently, Deller [5, 6] has reformulated an OBE algorithm of Huang [3, 8, 14, 19, 20] so that it is exactly the familiar weighted recursive least squares solution [34, 35] with the SM considerations handled through a special weighting strategy. This algorithm is referred to as *set membership weighted recursive least squares (SM-WRLS)* to distinguish it from the original OBE algorithm. A review of SM theory and related topics is presented in the next section.

The main objectives of this research are:

1. To generalize the SM-WRLS algorithm to handle *complex vector-input vector-*

output data streams.

2. To reduce the computational complexity of the algorithm.
3. To make the estimator adaptive.
4. To perform simulation studies to research the performance of the SM-WRLS algorithms.
5. To develop parallel architectures to implement the SM-WRLS algorithms.

The first three objectives which are addressed in *Chapter 2* are concerned with algorithmic developments centered on SM-WRLS identification. Currently, the SM-WRLS algorithm can be used in the noted application areas if the data are *real* numbers. It will be advantageous if this powerful algorithm can be extended to work with *complex* numbers. Complex-valued data are encountered in many digital signal processing and image processing problems (e.g., see [36, Chs. 2 & 8]). The current version of SM-WRLS cannot handle complex-valued data. Also, SM-WRLS has the potential for application to a wide range of problems in which the data to be processed (at each instant of time) may take the form of a *vector* quantity. For example, a very important research area is *beamforming* [37], a spatial filtering task, to which the SM-WRLS algorithm may be applied. At every time interval, the array of sensors of a narrowband beamformer provide vector outputs that need to be processed by the beamformer using complex computations. There is also potential for applying SM-WRLS to *neural networks* in which both the input and the output are (complex) vectors [38]. The theoretical development of a generalized SM-WRLS algorithm that can handle complex vector-input vector-output data (objective 1) is the subject of *Section 2.2.*

The second main objective of this research is to develop a more efficient SM-WRLS algorithm which uses a ***suboptimal*** checking criterion to reduce the computational complexity of SM-WRLS at the expense of using "suboptimal" weights. This suboptimal strategy can be applied to any version, adaptive or "non-adaptive," of

the SM-WRLS algorithm to improve the computational efficiency. The theoretical development of the suboptimal SM-WRLS algorithm is presented in *Section 2.3*. *Section 2.4* demonstrates how to make the SM-WRLS algorithm explicitly *adaptive* (objective 3) by introducing a flexible mechanism by which it can "forget" the influence of past data. It is to be noted, however, that the basic (unmodified) SM-WRLS algorithm is inherently "adaptive" compared with the RLS algorithm. This adaptation is inherent in the use of data weights which are "optimal" in the SM sense.

Chapter 3 is concerned with objective 4 which discusses simulation studies of the various suboptimal and adaptive strategies presented in *Chapter 2*. These studies illustrate important points about these strategies and about the SM-WRLS algorithm in general.

One of the advantages of the SM-WRLS formulation (contrasted with Huang's OBE algorithm [20]) is that it immediately admits solution by contemporary parallel architectures. This is critical because it reduces the complexity of the algorithm from $\mathcal{O}(m^2)$ to $\mathcal{O}(m)$, where m is the number of parameters to be estimated. The significant reduction of computational complexity and parallel hardware implementation of SM algorithms improve their potential for real time applications. *Chapter 4* is devoted to parallel hardware implementations (objective 5) of the SM-WRLS and discussion of their advantages, particularly with regard to their improved computational complexity.

Finally, *Chapter 5* summarizes the main conclusions and contributions of this research and suggests possible directions for future research topics in the SM realm.

1.2 Set Membership Theory

A brief background on SM theory is presented in this section which is divided into three major subsections. The first contains a brief overview of the identification

problem and *least squares (LS)* estimation. The second gives an overview of the OBE algorithm. The third presents the "scalar case" of the SM-WRLS algorithm and its formulation based on *Givens rotations (GR's)*. However, the computational complexities of these algorithms (and others) are presented and compared in *Chapter 2*.

1.2.1 The Identification Problem and Least Squares Estimation

A well known identification problem is the estimation of the parameters of a general *autoregressive moving average with exogenous input (ARMAX(p,q))* [35] model of the form

$$y(n) = \sum_{i=1}^{p} a_i y(n-i) + \sum_{j=0}^{q} b_j w(n-j) + v(n) \tag{1.2}$$

in which $y(n)$ is a scalar output of the model; $w(n)$ is a measurable, uncorrelated, input sequence; $v(n)$ is an uncorrelated[1] driving (or error) sequence, known to be bounded as in (1.1), which is independent of $w(n)$; and a_i's and b_j's are the parameters to be identified. For convenience, the following vector notations are employed

$$\boldsymbol{x}^T(n) \doteq [y(n-1)y(n-2)\cdots y(n-p)w(n)w(n-1)\cdots w(n-q)] \tag{1.3}$$

and

$$\boldsymbol{\theta}_0^T \doteq [a_1 a_2 \cdots a_p b_0 b_1 \cdots b_q] \tag{1.4}$$

and hence,

$$y(n) \doteq \boldsymbol{\theta}_0^T \boldsymbol{x}(n) + v(n)\ . \tag{1.5}$$

[1] As will be noticed below, the SM-WRLS algorithm has no nominal constraint on $v(n)$ other than (1.1). However, if $v(n)$ is correlated, one would expect to obtain biased estimates since the solution is essentially based on the RLS method [35]. The issue of a correlated $v(n)$ with SM processing remains open for further research.

Note that this model has an important subcase of a purely order p autoregressive (AR(p)) model which is prevalent in many important applications. The dimension of $\boldsymbol{\theta}_0$ is defined as the integer m,

$$m = p + q + 1 \tag{1.6}$$

noting that m should be reduced to simply $m = p$ for the pure AR case.

Consider the LS problem [39]: Given data (or a system of observations) on the interval $i \in [1, n]$ ($n \geq m$), and some set of error minimization weights, say $\{\lambda(i)\}$, form the overdetermined system of equations

$$\begin{bmatrix} \sqrt{\lambda(1)}\boldsymbol{x}^T(1) & \rightarrow \\ \sqrt{\lambda(2)}\boldsymbol{x}^T(2) & \rightarrow \\ \vdots & \\ \sqrt{\lambda(n)}\boldsymbol{x}^T(n) & \rightarrow \end{bmatrix} \boldsymbol{\theta}_0 = \begin{bmatrix} \sqrt{\lambda(1)}y(1) \\ \sqrt{\lambda(2)}y(2) \\ \vdots \\ \sqrt{\lambda(n)}y(n) \end{bmatrix} \tag{1.7}$$

denoted

$$\mathbf{X}(n)\boldsymbol{\theta}_0 = \boldsymbol{y}(n) \ , \tag{1.8}$$

and find the LS estimate, say $\hat{\boldsymbol{\theta}}(n)$, for the vector $\boldsymbol{\theta}_0$.

There are well known methods to solve this problem. The first is the "batch" solution given by [39]

$$\hat{\boldsymbol{\theta}}(n) = \left[\mathbf{X}^T(n)\mathbf{X}(n)\right]^{-1} \mathbf{X}^T(n)\boldsymbol{y}(n) \tag{1.9}$$

with the matrix in brackets playing the role of the weighted covariance matrix, i.e.,

$$\mathbf{C}_x(n) = \left[\mathbf{X}^T(n)\mathbf{X}(n)\right] = \sum_{i=1}^{n} \lambda(i)\boldsymbol{x}(i)\boldsymbol{x}^T(i) \ . \tag{1.10}$$

The second is the recursive *matrix inversion lemma (MIL)*-based WRLS solution

given by [34, 35]

$$\mathbf{P}(n) = \mathbf{P}(n-1) - \lambda(n)\frac{\mathbf{P}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\mathbf{P}(n-1)}{1+\lambda(n)G(n)} \tag{1.11}$$

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \lambda(n)\mathbf{P}(n)\boldsymbol{x}(n)\epsilon_{n-1}(n) \tag{1.12}$$

where,

$$\mathbf{P}(n) \doteq \mathbf{C}_x^{-1}(n) \tag{1.13}$$

$$G(n) \doteq \boldsymbol{x}^T(n)\mathbf{P}(n-1)\boldsymbol{x}(n) \tag{1.14}$$

$$\epsilon_{n-1}(n) \doteq y(n) - \hat{\boldsymbol{\theta}}^T(n-1)\boldsymbol{x}(n) \tag{1.15}$$

in which $\mathbf{C}_x(n)$ is the weighted covariance matrix defined above, $\mathbf{P}(n)$ is its inverse; $\hat{\boldsymbol{\theta}}(n)$ is the parameter vector estimate using n points of data; $\epsilon_{n-1}(n)$ is the residual (or error) at time n based on $\hat{\boldsymbol{\theta}}(n-1)$; and $\lambda(n)$ is some error minimization weight. These recursions are theoretically equivalent to the batch solution given in (1.9) at each n.

1.2.2 The OBE Algorithm

The OBE algorithm is an SM algorithm developed by Fogel, Huang, and colleagues [3, 8, 14, 19, 20, 23, 25] which can be used to identify a general ARMAX(p,q) model. An overview of this algorithm and its origin is presented in this section (paraphrased from [5, 6, 7]).

Let us first present the general idea behind SM algorithms. Consider the case of a real m-dimensional parameter space; $\mathcal{R}^m$. These algorithms bound the parameters in a subset of $\mathcal{R}^m$ as follows: At any given time, the model, incoming datum, and constraint (1.1) define a pointwise "hyperstrip" region of possible parameter sets in $\mathcal{R}^m$. Over a given time range, the intersection of these hyperstrips forms a convex

polytope of permissible solutions for the "true" parameters; $\boldsymbol{\theta}_0$. The description of this polytope can be greatly simplified (mathematically) when approximated by a bounding hyperellipsoid, a tight superset of the polytope [5, 6, 7, 14, 20].

By recognizing the relationship between the conventional WRLS and SM identification, Fogel [25] showed that there is a membership set (hyperellipsoid) centered on the *unweighted* RLS estimate associated with the identification of *constant* unknown parameters of a linear system driven by uncorrelated noise, based on constraints of the form (1.1). Fogel also studied the convergence of the membership set to a single point. The subsequent papers by Fogel, Huang, and colleagues [3, 8, 14, 19, 20, 23] discuss a *weighted* approach (i.e., the OBE algorithm) based on the same principle. This algorithm uses energy constraints of form (1.1) to restrict the solutions of the linear parameters to ellipsoidal domains. At time n, the estimator, $\hat{\boldsymbol{\theta}}(n)$, is the center of an ellipsoidal region in $\mathcal{R}^m$, of the form

$$E(n) \doteq \left\{ \boldsymbol{\theta} \;\middle|\; \left[\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(n)\right]^T \boldsymbol{\Phi}^{-1}(n) \left[\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(n)\right] \leq 1 \right\} , \quad \boldsymbol{\theta} \in \mathcal{R}^m \tag{1.16}$$

which represents the smallest bounding ellipsoid of possible solutions, $\boldsymbol{\theta}$, to which the true parameters must belong. $\boldsymbol{\Phi}^{-1}(n)$ can be interpreted as a weighted covariance matrix on the observations,

$$\boldsymbol{\Phi}^{-1}(n) = \sum_{i=1}^{n} \lambda(i) \boldsymbol{x}(i) \boldsymbol{x}^T(i) \; . \tag{1.17}$$

The algorithm, in effect, seeks the $\lambda(n)$ which minimizes the "volume ratio" of the sequential ellipsoids, $E(n)$ and $E(n-1)$, given the incoming datum, and *subject to scaling of the previous weights.* For this reason, the technique is referred to as the OBE algorithm. Frequently, no such weight exists, indicating that the new datum is uninformative (in the SM sense) and the effort of updating the parameter estimates can be avoided.

It is important to note that the OBE algorithm is derived from geometric considerations and is centered on three recursions [20], two of which are remarkably similar to the conventional MIL-based WRLS algorithm [34]. Careful analysis of OBE reveals that it is "WRLS with time varying weights." This is alluded to above. Whereas the OBE's geometric approach solves a weighted LS problem on a point-by-point basis, its recursions cannot be exactly interpreted as conventional WRLS because of the fundamental difference in their development. Recently, Deller [5, 6] has reformulated OBE into a more *conventional* WRLS technique referred to as SM-WRLS. The SM-WRLS formulation, which incorporates SM considerations directly into the standard WRLS recursions, is treated in the next section.

1.2.3 The SM-WRLS Algorithm

1.2.3.1 MIL-based SM-WRLS Algorithm

In this section, the theoretical development of the basic MIL-based SM-WRLS algorithm is sketched. This algorithm, which is described in detail in [6, 7], accepts only scalar, real-valued data which makes it applicable to a specific class of problems. A brief overview of the SM theory and the main results of the basic SM-WRLS algorithm are summarized below (paraphrased from [6, 7]) to serve as a foundation for generalizing this algorithm to handle complex-valued vector-input vector-output data. The theoretical development of the generalized algorithm is presented in *Chapter 2.*

Suppose that, at time i, $y(i)$, $\boldsymbol{x}(i)$, and the model of form (1.5) are given. If there is no other information about the system, the parameter vector $\boldsymbol{\theta}_0$ can theoretically take any real vector value. A constraint on $v(i)$ like (1.1), however, restricts the possible range of values of $\boldsymbol{\theta}_0$. From (1.1) and (1.5), it is clear that (at time i)

$$v^2(i) = \left[y(i) - \boldsymbol{\theta}_0^T \boldsymbol{x}(i)\right]^2 < \frac{1}{\gamma(i)} \,. \tag{1.18}$$

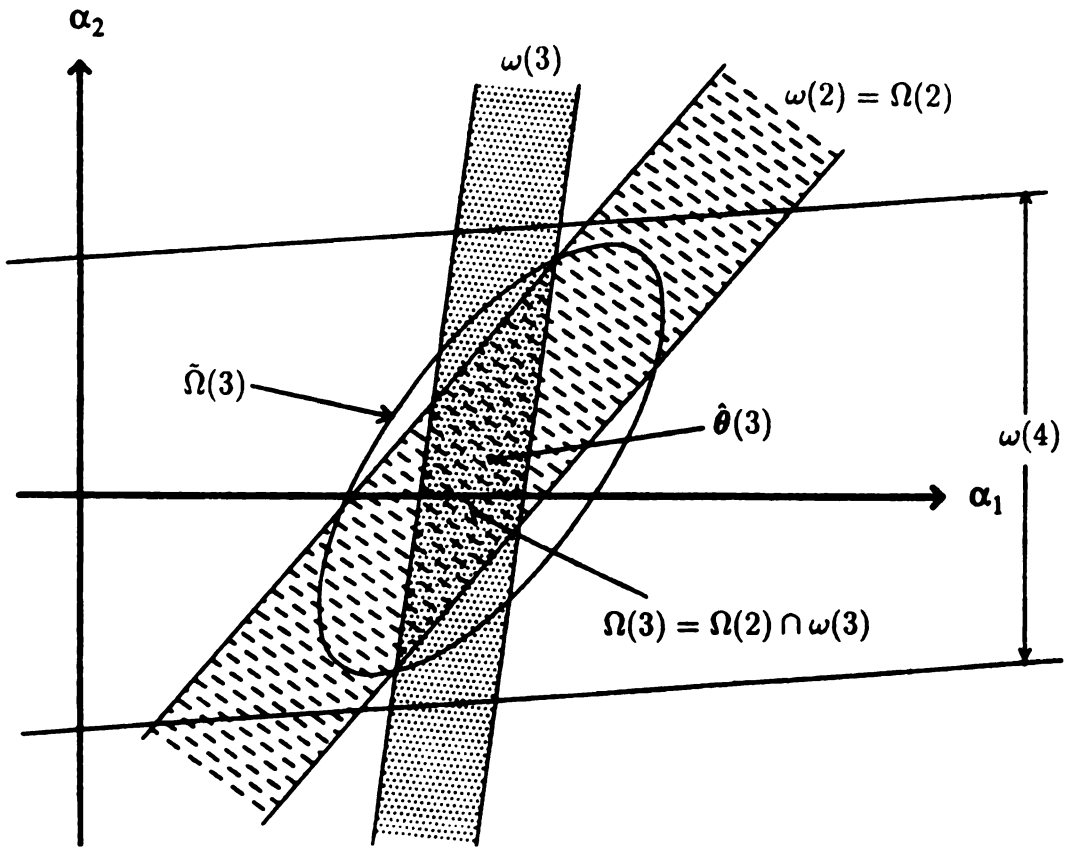


Figure 1.1: Local and global membership sets in 2-D.

It is assumed that the sequence of numbers, $\gamma(i)$, is known (or can be accurately estimated), and therefore, (1.18) restricts possible values of $\boldsymbol{\theta}_0$ to some range, say $\omega(i)$, that is called a "local membership set," to which $\boldsymbol{\theta}_0$ must belong at time i. If the data and the $\gamma(i)$ values are given over a range $[1, n]$, then n local sets $\omega(i)$, $i = 1, 2, \ldots, n$ (one for each observation) can be generated. Each of these takes the form of a "hyperstrip" in $\mathcal{R}^m$, the two-dimensional (2-D) case shown in Fig. 1.1. The parameter vector $\boldsymbol{\theta}_0$ must simultaneously belong to each of these sets, and therefore, must belong

to a *global membership set* given by

$$\Omega(n) = \bigcap_{i=1}^{n} \omega(i) \ . \tag{1.19}$$

$\Omega(i)$ will be a monotonically non-increasing set with i, and it will be the minimal (most restrictive) membership set known under the conditions of the problem. The global membership set, $\Omega(n)$, is the intersection of the individual strips $\omega(i)$ (see 1.19), which takes the form of a convex polytope in $\mathcal{R}^m$, as illustrated in Fig. 1.1 for a 2-D case. Note that the individual $\omega(i)$ does not necessarily help to refine $\Omega(i)$, i.e., it might be true that

$$\Omega(i) = \Omega(i-1) \cap \omega(i) = \Omega(i-1) \tag{1.20}$$

and the corresponding data are considered "unuseful" in the SM sense. See, for example, the case of $\Omega(4)$ in Fig. 1.1.

The set $\Omega(i)$, which requires high computational complexity algorithms [1, 10, 11, 12, 17, 21, 24] to describe and work with, provides a useful way of determining which data points are informative and which are not. Note that the center of this set provides a systematic estimate of the parameter vector $\boldsymbol{\theta}_0$ which would be expected to improve as i increases. However, neither $\Omega(i)$ nor its center is clearly or conveniently related to the WRLS estimation process of interest. There is, however, a related (but potentially larger) global membership set associated with the WRLS process at time n. This is derived from (1.1) and (1.18) by noting that the constraint (1.1) on the input implies an "accumulated inequality" given by

$$\sum_{i=1}^{n} \lambda(i) \left[y(i) - \boldsymbol{\theta}_0^T \boldsymbol{x}(i) \right]^2 < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} \tag{1.21}$$

which holds as long as the set of "weights" used, $\{\lambda(i)\}$, are *non-negative* (see [7,

Lemma 1]). This inequality leads to a global membership set, say $\tilde{\Omega}(n)$, to which $\boldsymbol{\theta}_0$ must belong. Since $\Omega(n)$ (from the discussion above) is the smallest known set, it must be true that $\Omega(n) \subseteq \tilde{\Omega}(n)$.

Two fundamental theorems underlie the basic SM-WRLS algorithm [5, 6, 7, 18]. The first indicates how the bounding ellipsoid is related to the conventional WRLS process and the second indicates how the optimal data weights are chosen. Proofs are found in [7] for the AR(p) case. The generalization to ARMAX(p, q) is straightforward.

Theorem 1 *[7] Let $\Omega(n) \subseteq \mathcal{R}^m$ be the set of all parameter vectors which are compatible with the data for $i \in [1, n]$ under constraint (1.1). Then there exists a superset of $\Omega(n)$, say $\tilde{\Omega}(n)$, a hyperellipsoid in $\mathcal{R}^m$, which is closely associated with the WRLS estimation process:*

$$\tilde{\Omega}(n) = \left\{ \boldsymbol{\theta} \mid \left[\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(n)\right]^T \frac{\mathbf{C}_x(n)}{\kappa(n)} \left[\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(n)\right] < 1 \right\} , \quad \boldsymbol{\theta} \in \mathcal{R}^m \tag{1.22}$$

where,

$$\kappa(n) = \hat{\boldsymbol{\theta}}^T(n)\mathbf{C}_x(n)\hat{\boldsymbol{\theta}}(n) + \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} \left(1 - \gamma(i)y^2(i)\right) \tag{1.23}$$

in which $\hat{\boldsymbol{\theta}}(n)$ is the conventional WRLS estimate of $\boldsymbol{\theta}_0$ at time n using weights $\{\lambda(i)\}$.

This theorem, which is derived from (1.21) by replacing $\boldsymbol{\theta}_0$ with a general vector $\boldsymbol{\theta}$, simply means that there is a hyperellipsoidal domain in $\mathcal{R}^m$ (i.e., all $\boldsymbol{\theta} \in \tilde{\Omega}(n)$) which is guaranteed to contain $\boldsymbol{\theta}_0$, and which is *centered on the WRLS estimate* $\hat{\boldsymbol{\theta}}(n)$.

The "volume" of the ellipsoid $\tilde{\Omega}(n)$ is inversely proportional to the determinant of the matrix $\mathbf{C}_x(n)/\kappa(n)$, and is a function of only one unknown, $\lambda(n)$. Therefore, a logical strategy for the selection of weights is to choose $\lambda(n)$ to maximally shrink the volume of $\tilde{\Omega}(n)$. If no such weight exists, the data at time n should be "rejected" ($\lambda(n)$ effectively set to zero), as it does not serve to refine the estimate of the parameters,

thereby saving the computational expense otherwise necessary to incorporate it into the estimate.

The ellipsoid volume is proportional to the quantity

$$\det \mathbf{B}(n) \doteq \det \kappa(n) \mathbf{C}_x^{-1}(n) \ . \tag{1.24}$$

A reasonable strategy is to find an *optimal* weight, $\lambda^*(n)$, at each step that minimizes the "volume ratio" of the ellipsoids at n and $n-1$:

$$V(\lambda(n)) = \frac{\det \mathbf{B}(n)}{\det \mathbf{B}(n-1)} \ . \tag{1.25}$$

Theorem 2 *[7] The weight, $\lambda^*(n)$, which minimizes the volume ratio (1.25), is the most positive root of the quadratic equation*

$$F(\lambda) = \alpha_2 \lambda^2 + \alpha_1 \lambda + \alpha_0 = 0 \tag{1.26}$$

where,

$$\begin{aligned} \alpha_2 &= (m-1)G^2(n) \\ \alpha_1 &= \left\{2m - 1 + \gamma(n)\epsilon_{n-1}^2(n) - \kappa(n-1)\gamma(n)G(n)\right\} G(n) \\ \alpha_0 &= m\left[1 - \gamma(n)\ \epsilon_{n-1}^2(n)\right] - \kappa(n-1)\gamma(n)G(n) \end{aligned}$$

in which all the quantities are defined above.

1.2.3.2 GR-based SM-WRLS Algorithm

In this section, a solution of the SM-WRLS algorithm that is amenable to parallel hardware implementation is presented, which integrates SM weights with a GR version of WRLS (paraphrased from [5, 6]).

The solution is based on the orthogonal triangularization (by GR's) of the $\mathbf{X}(n)$ matrix of (1.8) [5, 6, 39, 40, 41, 42, 43, 44]. The procedure, in principle, involves the application of a sequence of orthogonal operators (GR's) to (1.8) which leaves the system in the form

$$\begin{bmatrix} \mathbf{T}(n) \\ \hline \mathbf{0}_{(n-m)\times m} \end{bmatrix} \boldsymbol{\theta}_0 = \begin{bmatrix} \boldsymbol{d}_1(n) \\ \hline \boldsymbol{d}_2(n) \end{bmatrix} \tag{1.27}$$

where the matrix $\mathbf{T}(n)$ is an $m \times m$ upper triangular Cholesky factor [39] of $\mathbf{C}_x(n)$ (see (1.30) below), and $\mathbf{0}_{i\times j}$ denotes the $i \times j$ zero matrix. The system

$$\mathbf{T}(n)\hat{\boldsymbol{\theta}}(n) = \boldsymbol{d}_1(n) \tag{1.28}$$

is easily solved using back substitution [39] to obtain the LS estimate, $\hat{\boldsymbol{\theta}}(n)$. This formulation makes possible the solution (in terms of computation and implementation) of SM-WRLS on contemporary parallel architectures (developed in *Chapter 4*) for great speed advantages.

Some computational details need to be examined for future purposes. In computing $\lambda^*(n)$, it is noted that $F(\lambda)$ contains terms involving the inverse covariance matrix $\mathbf{C}_x^{-1}(n)$, which never occurs elsewhere in the GR-based algorithm. In particular, the computation of the scalar (see (1.14))

$$G(n) \doteq \boldsymbol{x}^T(n)\mathbf{C}_x^{-1}(n-1)\boldsymbol{x}(n) \tag{1.29}$$

requires $\mathcal{O}(m^2)$ operations which comprise the main computational load in determining $\lambda^*(n)$. This problem is easily resolved by noting that

$$\mathbf{C}_x(n) = \mathbf{X}^T(n)\mathbf{X}(n) = \mathbf{T}^T(n)\mathbf{T}(n) \tag{1.30}$$

because $\mathbf{T}(n)$ represents an orthogonal transformation on $\mathbf{X}(n)$. Therefore,

$$\begin{aligned} G(n) &= \boldsymbol{x}^T(n)\mathbf{T}^{-1}(n-1)\mathbf{T}^{-T}(n-1)\boldsymbol{x}(n) \\ &\doteq \boldsymbol{g}^T(n)\boldsymbol{g}(n) = \| \boldsymbol{g}(n) \|_2^2 \end{aligned} \tag{1.31}$$

in which $\| \cdot \|_2$ denotes the l_2 norm on $\mathcal{R}^m$. Since $\boldsymbol{x}(n) = \mathbf{T}^T(n-1)\boldsymbol{g}(n)$, and the matrix $\mathbf{T}^T(n-1)$ is lower triangular, $\boldsymbol{g}(n)$ is easily found from the available quantities at time n by back substitution.

A similarly inexpensive procedure for computing $\kappa(n)$ is available in the context of (1.27). Equation (1.23) can be written

$$\kappa(n) = \hat{\boldsymbol{\theta}}^T(n)\mathbf{C}_x(n)\hat{\boldsymbol{\theta}}(n) + \tilde{\kappa}(n) \tag{1.32}$$

where,

$$\begin{aligned} \tilde{\kappa}(n) &= \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} \left(1 - \gamma(i)y^2(i)\right) \\ &= \tilde{\kappa}(n-1) + \frac{\lambda(n)}{\gamma(n)} \left(1 - \gamma(n)y^2(n)\right) \end{aligned} \tag{1.33}$$

with $\tilde{\kappa}(0) = 0$. Also, from (1.27) and (1.30), the first term in (1.32) is easily shown to be $\| \boldsymbol{d}_1(n) \|_2^2$, and therefore, (1.32) can be written

$$\kappa(n) = \| \boldsymbol{d}_1(n) \|_2^2 + \tilde{\kappa}(n) \; . \tag{1.34}$$

Figure 1.2 summarizes this GR-based SM-WRLS algorithm.

Finally, the quantity $\det \mathbf{B}(n)$ can be conveniently monitored within the system (1.27), since, from (1.24) and (1.30),

$$\log \{\det \mathbf{B}(n)\} = m \; \log \{\kappa(n)\} - 2 \; \log \{\det \mathbf{T}(n)\} \tag{1.35}$$

INITIALIZATION: Fill $(m+1) \times (m+1)$ working matrix, $\mathbf{W}$, with zeros.
$\lambda(i) = 1, \quad i = 1, 2, \ldots, m+1$
$\tilde{\kappa}(0) = 0$

RECURSION: For $i = 1, 2, \ldots, n$,

STEP 1. (Skip[a] if $i \leq m+1$) Update $G(i)$, $\epsilon_{i-1}(i)$.
Solve $\mathbf{T}^T(i-1)\boldsymbol{g}(i) = \boldsymbol{x}(i)$ for $\boldsymbol{g}(i)$ by back substitution.
$G(i) = \| \boldsymbol{g}(i) \|_2^2$
$\epsilon_{i-1}(i) = y(i) - \hat{\boldsymbol{\theta}}^T(i-1)\boldsymbol{x}(i)$

STEP 2. (Skip if $i \leq m+1$) Compute optimal $\lambda(i)$, say $\lambda^*(i)$, by finding most positive root of quadratic (1.26).

STEP 3. (Skip if $i \leq m+1$) If $\lambda^*(i) \leq 0$, set
$\mathbf{T}(i) = \mathbf{T}(i-1)$
$\hat{\boldsymbol{\theta}}(i) = \hat{\boldsymbol{\theta}}(i-1)$
$\tilde{\kappa}(i) = \tilde{\kappa}(i-1)$
and go to STEP 7.
Otherwise, continue.

STEP 4. Update $\mathbf{T}(i)$.
Replace bottom row of $\mathbf{W}$ by $\sqrt{\lambda^*(i)}\left[\boldsymbol{x}^T(i) \mid y(i)\right]$.
Rotate this "new equation" into $\mathbf{W}$ using Givens rotations, leaving the result $[\mathbf{T}(i) \mid \boldsymbol{d}_1(i)]$ in the upper m rows of $\mathbf{W}$.
These rotations involve the scalar computations [40, 43]
$W'_{jk} = W_{jk}\sigma + W_{m+1,k}\tau\delta$ and $W'_{m+1,k} = -W_{jk}\tau\delta + W_{m+1,k}\sigma\delta$
for $k = j, j+1, \ldots, m+1$ and for $j = 1, 2, \ldots, m$;
where, $\sigma = W_{jj}/\rho$, $\tau = W_{m+1,k}/\rho$, $\rho = \sqrt{W_{jj}^2 + \delta W_{m+1,j}^2}$, δ is unity[b] and W_{jk} (W'_{jk}) is the j,k element of $\mathbf{W}$ pre- (post-) rotation.

STEP 5. (Skip if $i \leq m$) Update $\hat{\boldsymbol{\theta}}(i)$, solving $\mathbf{T}(i)\hat{\boldsymbol{\theta}}(i) = \boldsymbol{d}_1(i)$ by back substitution.

STEP 6. Update $\kappa(i)$ and $\tilde{\kappa}(i)$ according to
$\tilde{\kappa}(i) = \tilde{\kappa}(i-1) + \frac{\lambda^*(i)}{\gamma(i)}(1 - \gamma(i)y^2(i))$
$\kappa(i) = \| \boldsymbol{d}_1(i) \|_2^2 + \tilde{\kappa}(i)$
Compute and store only $\tilde{\kappa}(i)$ if $i \leq m$.

STEP 7. If $i \leq n$, increment i and return to Step 1.

[a] Generally $\mathbf{T}(i)$ does not become nonsingular until $i = m+1$. The first $\hat{\boldsymbol{\theta}}(i)$ cannot be computed until $i = m+1$ and the first $\lambda^*(i)$ at $i = m+2$. We arbitrarily set $\lambda(i) = 1$ on the initial range.

[b] δ is set to -1 to rotate an equation *out* of the estimate [40].

Figure 1.2: The SM-WRLS algorithm based on Givens rotations.

and $\det \mathbf{T}(n)$ is easily computed as the product of the diagonal elements of $\mathbf{T}(n)$.

Chapter 2

New Theoretical Results and Algorithms

2.1 Introduction

This chapter is divided into three major sections. The first is devoted to the theoretical development of a generalized SM-WRLS algorithm that can handle complex vector-input vector-output data.

In the case when it is critical to obtain the solutions of certain problems as fast as possible, it is essential to have a yet more efficient SM-WRLS algorithm that produces *acceptable* solutions faster. The theoretical development of this algorithm, which uses a *suboptimal* checking criterion to reduce the computational complexity of SM-WRLS at the expense of using "suboptimal" weights, is presented in *Section 2.3*.

Finally, *Section 2.4* demonstrates how to make the SM-WRLS algorithm explicitly *adaptive* with a very flexible mechanism by which it can "forget" the influence of past data. Three major subcases are identified and presented. This section also shows that exponential forgetting factor adaptation can be incorporated into SM-WRLS. The theoretical development of this algorithm and a discussion explaining why it has not been found to be effective for adaptation in preliminary experiments are presented.

2.2 A Generalized "Non-adaptive" SM-WRLS Algorithm

This section is devoted to the theoretical development of the generalized (for complex vector data) SM-WRLS algorithm. The developments here are guided by the work of Deller and Luk in [7] on the real scalar case and Deller in [45] on a special vector case.

Consider the general linear model of the form (cf. (1.2))

$$\boldsymbol{y}(n) = \sum_{i=1}^{p} \mathbf{A}_i^H \boldsymbol{y}(n-i) + \sum_{j=0}^{q} \mathbf{B}_j^H \boldsymbol{w}(n-j) + \boldsymbol{v}(n) \tag{2.1}$$

in which $\boldsymbol{y}(n) \in \mathcal{C}^k$ is a complex vector output sequence of the model; $\boldsymbol{w}(n) \in \mathcal{C}^k$ is a measurable, uncorrelated, complex vector input sequence to the model; $\boldsymbol{v}(n) \in \mathcal{C}^k$ is an uncorrelated[2] complex vector driving (or error) sequence, known to be bounded (as in (2.18) below), which is independent of $\boldsymbol{w}(n)$; $\mathbf{A}_i \in \mathcal{C}^{k \times k}$, $i = 1, 2, \ldots, p$, and $\mathbf{B}_j \in \mathcal{C}^{k \times k}$, $j = 0, 1, \ldots, q$, are complex matrices of "true" parameters to be identified; and superscript H is a standard notation for the *conjugate transpose (or hermitian)*. For convenience in the following, the dimensions of the vectors $\boldsymbol{y}(n)$ and $\boldsymbol{w}(n)$ are assumed to be equal (more on this below).

The key to generalizing the results of the real scalar case discussed in *Chapter 1* is the reformulation of (2.1) as

$$\boldsymbol{y}(n) = \sum_{l=1}^{k} \boldsymbol{\Theta}_l^H \boldsymbol{x}_l(n) + \boldsymbol{v}(n) \tag{2.2}$$

in which $\boldsymbol{\Theta}_l \in \mathcal{C}^{m \times k}$ is a complex matrix of the parameters to be identified which are

[2]Absence of both temporal correlation and component correlation are assumed here. This will be useful in formulating the energy bounding assumption (2.18) below. Also, see Footnote 1 in *Chapter 1*.

associated with the vector $\boldsymbol{x}_l(n) \in \mathcal{C}^m$. For convenience, the following vector/matrix notations are employed

$$\boldsymbol{x}_l^T(n) \doteq [y_l(n-1)y_l(n-2)\cdots y_l(n-p)w_l(n)w_l(n-1)\cdots w_l(n-q)] \tag{2.3}$$

and

$$\boldsymbol{\Theta}_l \doteq \begin{bmatrix} a_{1,l1} & a_{1,l2} & \cdots & a_{1,lk} \\ a_{2,l1} & a_{2,l2} & \cdots & a_{2,lk} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p,l1} & a_{p,l2} & \cdots & a_{p,lk} \\ b_{0,l1} & b_{0,l2} & \cdots & b_{0,lk} \\ b_{1,l1} & b_{1,l2} & \cdots & b_{1,lk} \\ \vdots & \vdots & \ddots & \vdots \\ b_{q,l1} & b_{q,l2} & \cdots & b_{q,lk} \end{bmatrix} \tag{2.4}$$

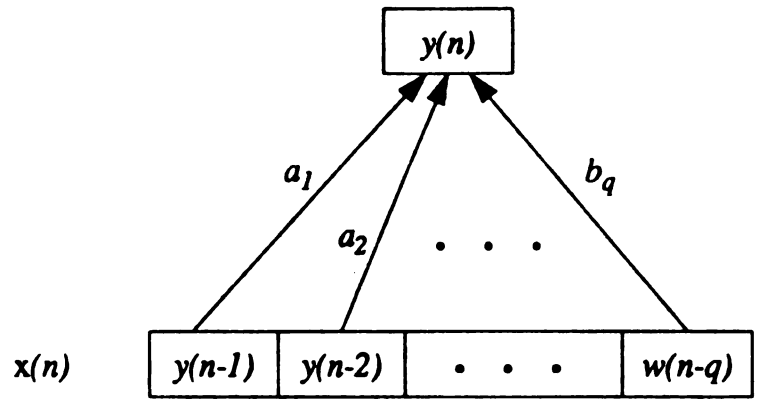
in which $w_l(n)$ is the l^{th} element of $\boldsymbol{w}(n)$; and $a_{i,lk}$'s and $b_{j,lk}$'s are the parameters to be identified which are associated with the l^{th} vector, $\boldsymbol{x}_l(n)$, and the k^{th} output element, $y_k(n)$. Figure 2.3 shows (a) the real scalar linear model of (1.2) and (b) the general complex vector linear model of (2.1). It is to be noted that the matrix $\mathbf{A}_i$ in (2.1) consists of the lk elements, $a_{i,lk}$, and the matrix $\mathbf{B}_j$ consists of the lk elements, $b_{j,lk}$ (see (2.4) and Fig. 2.3). Note that if the dimensions of the vectors $\boldsymbol{y}(n)$ and $\boldsymbol{w}(n)$ are not equal, then the corresponding elements in (2.2) are replaced by zeros.

The general model can be written

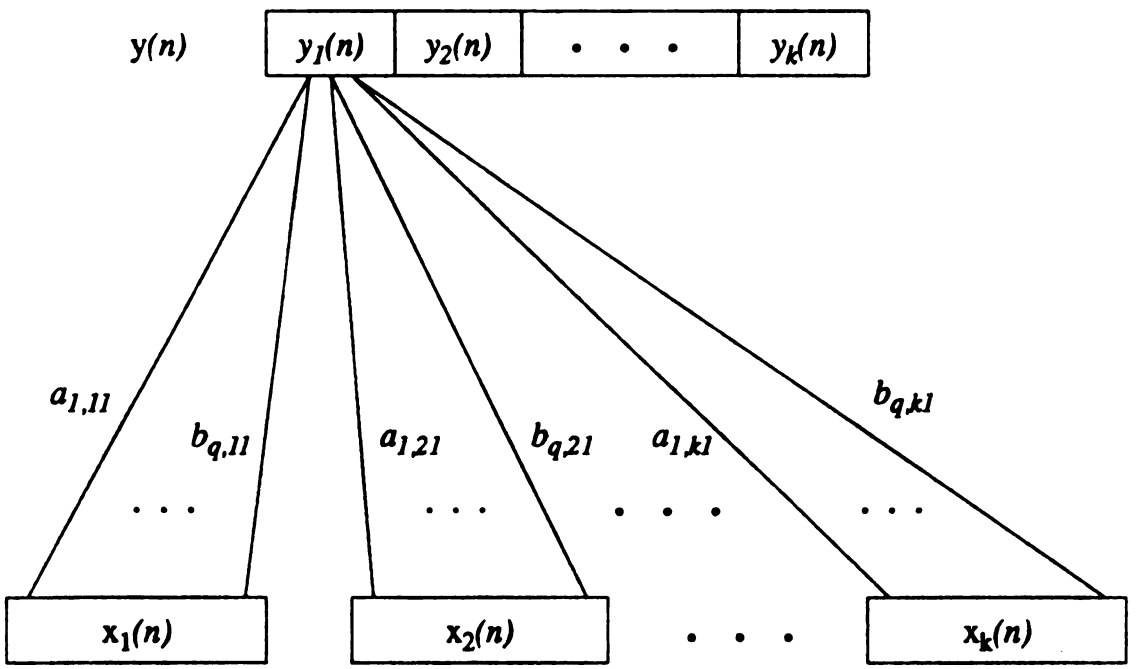
$$\boldsymbol{y}(n) = \boldsymbol{\Theta}_0^H \boldsymbol{x}(n) + \boldsymbol{v}(n) \tag{2.5}$$

where,

$$\boldsymbol{\Theta}_0^H \doteq \left[\boldsymbol{\Theta}_1^H \boldsymbol{\Theta}_2^H \cdots \boldsymbol{\Theta}_k^H\right] \tag{2.6}$$



(a)



(b)

Figure 2.3: (a) The real scalar linear model and (b) the general complex vector linear model.

and

$$\boldsymbol{x}^T(n) \doteq \left[\boldsymbol{x}_1^T(n)\boldsymbol{x}_2^T(n)\cdots\boldsymbol{x}_k^T(n)\right] \ . \tag{2.7}$$

Given a vector $\boldsymbol{x}(i) \in \mathcal{C}^{mk}$ and an output vector $\boldsymbol{y}(i) \in \mathcal{C}^k$ on the interval $i \in [1,n]$ $(n \geq m)$, and some set of error minimization weights, say $\{\lambda(i)\}$, the LS estimate, say $\hat{\boldsymbol{\Theta}}(n)$, of the parameter matrix $\boldsymbol{\Theta}_0 \in \mathcal{C}^{mk\times k}$ is the solution of the overdetermined system of equations of the form

$$\begin{bmatrix} \sqrt{\lambda(1)}\boldsymbol{x}^H(1) & \rightarrow \\ \sqrt{\lambda(2)}\boldsymbol{x}^H(2) & \rightarrow \\ \vdots & \\ \sqrt{\lambda(n)}\boldsymbol{x}^H(n) & \rightarrow \end{bmatrix} \boldsymbol{\Theta}_0 = \begin{bmatrix} \sqrt{\lambda(1)}\boldsymbol{y}^H(1) & \rightarrow \\ \sqrt{\lambda(2)}\boldsymbol{y}^H(2) & \rightarrow \\ \vdots & \\ \sqrt{\lambda(n)}\boldsymbol{y}^H(n) & \rightarrow \end{bmatrix} \tag{2.8}$$

denoted

$$\mathbf{X}^H(n)\boldsymbol{\Theta}_0 = \mathbf{Y}^H(n) \ . \tag{2.9}$$

The batch solution is given by [39]

$$\hat{\boldsymbol{\Theta}}(n) = \left[\mathbf{X}(n)\mathbf{X}^H(n)\right]^{-1}\mathbf{X}(n)\mathbf{Y}^H(n) \tag{2.10}$$

with the matrix in brackets called the (weighted) covariance matrix, i.e.,

$$\mathbf{C}_x(n) \doteq \left[\mathbf{X}(n)\mathbf{X}^H(n)\right] = \sum_{i=1}^{n}\lambda(i)\boldsymbol{x}(i)\boldsymbol{x}^H(i) \ . \tag{2.11}$$

The remaining matrix product (in (2.10)) is *cross-covariance* matrix for the vector inputs and outputs, denoted $\mathbf{C}_{xy}(n)$ and given by

$$\mathbf{C}_{xy}(n) \doteq \left[\mathbf{X}(n)\mathbf{Y}^H(n)\right] = \sum_{i=1}^{n}\lambda(i)\boldsymbol{x}(i)\boldsymbol{y}^H(i) \ . \tag{2.12}$$

The conventional recursions of the MIL-based WRLS solution, which we upgrade

here to the complex vector case, are given by

$$\mathbf{P}(n) = \mathbf{P}(n-1) - \lambda(n)\frac{\mathbf{P}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\mathbf{P}(n-1)}{1+\lambda(n)G(n)} \tag{2.13}$$

$$\hat{\boldsymbol{\Theta}}(n) = \hat{\boldsymbol{\Theta}}(n-1) + \lambda(n)\mathbf{P}(n)\boldsymbol{x}(n)\epsilon_{n-1}^H(n) \tag{2.14}$$

where,

$$\mathbf{P}(n) \doteq \mathbf{C}_{\boldsymbol{x}}^{-1}(n) \tag{2.15}$$

$$G(n) \doteq \boldsymbol{x}^H(n)\mathbf{P}(n-1)\boldsymbol{x}(n) \tag{2.16}$$

$$\epsilon_{n-1}(n) \doteq \boldsymbol{y}(n) - \hat{\boldsymbol{\Theta}}^H(n-1)\boldsymbol{x}(n)\ . \tag{2.17}$$

As in the real scalar case, it is assumed that the complex vector error sequence, $\boldsymbol{v}(n)$, is pointwise "energy bounded," i.e.,

$$\gamma(n)\ \mathrm{tr}\left\{\boldsymbol{v}(n)\boldsymbol{v}^H(n)\right\} < 1 \tag{2.18}$$

where the sequence $\gamma(n)$ is known or can be estimated from the data, and $\mathrm{tr}\{\mathbf{A}\}$ denotes the *trace* of the matrix $\mathbf{A}$. Since $\boldsymbol{v}(n)\boldsymbol{v}^H(n)$ is a hermitian matrix with *real* diagonal elements, the sequence $\gamma(n)$ is real numbers. This is useful in the proof of *Lemma 1* below.

The significance of the bounding sequence on $\mathrm{tr}\left\{\boldsymbol{v}(n)\boldsymbol{v}^H(n)\right\}$ is that it implies pointwise "local membership sets" to which any reasonable estimate for $\boldsymbol{\Theta}_0$ must belong. If the local membership set at time n is $\boldsymbol{\omega}(n)$, it follows immediately from (2.5) and (2.18) that

$$\boldsymbol{\omega}(n) = \left\{\boldsymbol{\Theta}\ \middle|\ \gamma(n)\ \mathrm{tr}\left\{\left[\boldsymbol{y}(n) - \boldsymbol{\Theta}^H\boldsymbol{x}(n)\right]\left[\boldsymbol{y}(n) - \boldsymbol{\Theta}^H\boldsymbol{x}(n)\right]^H\right\} < 1\right\}\ ,\quad \boldsymbol{\Theta} \in \mathcal{C}^{mk\times k} \tag{2.19}$$

where $\boldsymbol{\Theta}$ is a general matrix replacing $\boldsymbol{\Theta}_0$. The interpretation of this set becomes

clearer when considering a single output, $y_i(n)$, the i^{th} element of $\boldsymbol{y}(n)$. The related subset is

$$\boldsymbol{\omega}_i(n) = \left\{ \boldsymbol{\theta}_i \mid \gamma(n) \left[y_i(n) - \boldsymbol{\theta}_i^H \boldsymbol{x}(n) \right] \left[y_i(n) - \boldsymbol{\theta}_i^H \boldsymbol{x}(n) \right]^H < 1 \right\} , \quad \boldsymbol{\theta}_i \in \mathcal{C}^{mk} \tag{2.20}$$

in which $\boldsymbol{\theta}_i$ is the i^{th} column of the parameter matrix $\boldsymbol{\Theta}$. Each $\boldsymbol{\omega}_i(n)$ takes the form of a hyperstrip (or degenerate hyperellipsoid) in the parameter subspace $\mathcal{C}^{mk}$. If the data and the $\gamma(i)$ values are given over a range $[1,n]$, then n local sets $\boldsymbol{\omega}(i), i = 1,2,\ldots,n$ (one for each observation) can be generated. The parameter matrix Θ_0 must simultaneously belong to each of these sets, and therefore, must belong to a "global membership set" given by

$$\boldsymbol{\Omega}(n) = \bigcap_{i=1}^{n} \boldsymbol{\omega}(i) . \tag{2.21}$$

$\boldsymbol{\Omega}(i)$ will be a monotonically non-increasing set with i, and it will be the minimal (most restrictive) membership set known under the conditions of the problem.

Following the same arguments as in the real scalar case (see *Section 1.2.3.1*), a related (but potentially larger) global membership set associated with the WRLS process (at time n) can be derived. This is done by noting that the constraint (2.18) on $\boldsymbol{v}(n)$ implies that an "accumulated inequality" holds:

Lemma 1 *Condition (2.18) implies*

$$\sum_{i=1}^{n} \lambda(i)\ tr\left\{ \boldsymbol{v}(n)\boldsymbol{v}^H(n) \right\} \leq \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} \tag{2.22}$$

for any non-negative (real) sequence $\{\lambda(i)\}$. *The equality can be removed for* $n \geq i_0$, *where* i_0 *is the minimum* i *for which* $\lambda(i) \neq 0$.

Proof of Lemma 1: (Guided by Deller and Luk [7]). That the equality holds for $n < i_0$ is obvious. At i_0

$$\lambda(i_0)\ \mathrm{tr}\left\{\boldsymbol{v}(i_0)\boldsymbol{v}^H(i_0)\right\} < \frac{\lambda(i_0)}{\gamma(i_0)}\ , \tag{2.23}$$

which follows immediately from the positivity of $\lambda(i_0)$ and (2.18). But

$$\lambda(i_0)\ \mathrm{tr}\left\{\boldsymbol{v}(i_0)\boldsymbol{v}^H(i_0)\right\} = \sum_{i=1}^{i_0}\lambda(i)\ \mathrm{tr}\left\{\boldsymbol{v}(i)\boldsymbol{v}^H(i)\right\} \tag{2.24}$$

and

$$\lambda(i_0) = \sum_{i=1}^{i_0}\lambda(i)\ . \tag{2.25}$$

So,

$$\sum_{i=1}^{i_0}\lambda(i)\ \mathrm{tr}\left\{\boldsymbol{v}(i)\boldsymbol{v}^H(i)\right\} < \sum_{i=1}^{i_0}\frac{\lambda(i)}{\gamma(i)}\ . \tag{2.26}$$

Also

$$\lambda(i_0+1)\ \mathrm{tr}\left\{\boldsymbol{v}(i_0+1)\boldsymbol{v}^H(i_0+1)\right\} \le \frac{\lambda(i_0+1)}{\gamma(i_0+1)}\ . \tag{2.27}$$

Adding (2.26) and (2.27)

$$\sum_{i=1}^{i_0+1}\lambda(i)\ \mathrm{tr}\left\{\boldsymbol{v}(i)\boldsymbol{v}^H(i)\right\} < \sum_{i=1}^{i_0+1}\frac{\lambda(i)}{\gamma(i)}\ . \tag{2.28}$$

and so on, by induction. □

It is assumed, for convenience, that $\lambda(1) \neq 0$, and therefore, *Lemma 1* becomes

$$\sum_{i=1}^{n}\lambda(i)\ \mathrm{tr}\left\{\boldsymbol{v}(i)\boldsymbol{v}^H(i)\right\} < \sum_{i=1}^{n}\frac{\lambda(i)}{\gamma(i)}\ . \tag{2.29}$$

By inserting $\boldsymbol{y}(i) - \boldsymbol{\Theta}_0^H\boldsymbol{x}(i)$ for $\boldsymbol{v}(i)$, inequality (2.29) becomes

$$\sum_{i=1}^{n}\lambda(i)\ \mathrm{tr}\left\{\left[\boldsymbol{y}(i) - \boldsymbol{\Theta}_0^H\boldsymbol{x}(i)\right]\left[\boldsymbol{y}(i) - \boldsymbol{\Theta}_0^H\boldsymbol{x}(i)\right]^H\right\} < \sum_{i=1}^{n}\frac{\lambda(i)}{\gamma(i)}\ . \tag{2.30}$$

This inequality is a fundamental result which leads to a global membership set, say $\tilde{\Omega}(n)$, to which Θ_0 must belong. Since $\Omega(n)$ (from the discussion above) is the smallest known set, it must be true that $\Omega(n) \subseteq \tilde{\Omega}(n)$. The main result is stated as a theorem.

Theorem 3 *Let $\Omega(n) \subseteq \mathcal{C}^{mk \times k}$ be the set of all parameter matrices which are compatible with the data for $i \in [1,n]$ under constraint (2.18). Then there exists a superset of $\Omega(n)$, say $\tilde{\Omega}(n)$, a hyperellipsoid in $\mathcal{C}^{mk \times k}$, which is closely associated with the WRLS estimation process:*

$$\tilde{\Omega}(n) = \left\{ \Theta \;\middle|\; tr\left\{ \left[\Theta - \hat{\Theta}(n)\right]^H \frac{\mathbf{C}_x(n)}{\kappa(n)} \left[\Theta - \hat{\Theta}(n)\right] \right\} < 1 \right\} , \quad \Theta \in \mathcal{C}^{mk \times k} \tag{2.31}$$

where,

$$\kappa(n) = tr\left\{ \hat{\Theta}^H(n) \mathbf{C}_x(n) \hat{\Theta}(n) \right\} + \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} - tr\{\mathbf{C}_y(n)\} . \tag{2.32}$$

in which Θ is a general matrix replacing Θ_0.

As before, the interpretation of the set $\tilde{\Omega}(n)$ is simple when considering each scalar component of the output individually. The result is a corollary of the theorem.

Corollary 1 *Under the conditions of Theorem 3, all possible parameter vectors associated with output y_i, say θ_i, are confined to a hyperellipsoidal membership set which is centered on its current estimate, $\hat{\theta}_i(n)$,*

$$\tilde{\Omega}_i(n) = \left\{ \theta_i \;\middle|\; \left[\theta_i - \hat{\theta}_i(n)\right]^H \frac{\mathbf{C}_x(n)}{\kappa(n)} \left[\theta_i - \hat{\theta}_i(n)\right] < 1 \right\} , \quad \theta_i \in \mathcal{C}^{mk} \tag{2.33}$$

in which $\hat{\theta}_i(n)$ is the WRLS estimate of column i of the parameter matrix $\Theta(n)$ at time n using error minimization weights $\{\lambda(i)\}$.

This means simply that there is a hyperellipsoidal domain in the parameter subspace which contains all possible parameter vectors and which is centered on the WRLS estimate. Note that the ellipsoid associated with each y_i, $i = 1, 2, \ldots, k$, is identical to all others except for its center.

Proofs of Theorem 3 and Corollary 1: (Guided by Deller [45]) It follows immediately from *Lemma 1* that (see (2.30))

$$\sum_{i=1}^{n} \lambda(i)\ \mathrm{tr}\left\{\left[\boldsymbol{y}(i) - \boldsymbol{\Theta}^H \boldsymbol{x}(i)\right]\left[\boldsymbol{y}(i) - \boldsymbol{\Theta}^H \boldsymbol{x}(i)\right]^H\right\} < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} \ . \tag{2.34}$$

This constrains the possible parameter matrices to the set

$$\left\{\boldsymbol{\Theta} \ \mid \ \sum_{i=1}^{n} \lambda(i)\ \mathrm{tr}\left\{\left[\boldsymbol{y}(i) - \boldsymbol{\Theta}^H \boldsymbol{x}(i)\right]\left[\boldsymbol{y}(i) - \boldsymbol{\Theta}^H \boldsymbol{x}(i)\right]^H\right\} < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)}\right\} \ . \tag{2.35}$$

Expanding the trace term,

$$\left\{\boldsymbol{\Theta} \ \mid \ \sum_{i=1}^{n} \lambda(i)\ \mathrm{tr}\left\{\boldsymbol{y}(i)\boldsymbol{y}^H(i) - \boldsymbol{\Theta}^H \boldsymbol{x}(i)\boldsymbol{y}^H(i) - \boldsymbol{y}(i)\boldsymbol{x}^H(i)\boldsymbol{\Theta} + \boldsymbol{\Theta}^H \boldsymbol{x}(i)\boldsymbol{x}^H(i)\boldsymbol{\Theta}\right\} < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)}\right\} \ . \tag{2.36}$$

Moving the summation across terms,

$$\left\{\boldsymbol{\Theta} \ \mid \ \mathrm{tr}\left\{\mathbf{C}_y(n) - \boldsymbol{\Theta}^H \mathbf{C}_{xy}(n) - \mathbf{C}_{xy}^H(n)\boldsymbol{\Theta} + \boldsymbol{\Theta}^H \mathbf{C}_x(n)\boldsymbol{\Theta}\right\} < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)}\right\} \tag{2.37}$$

where, $\mathbf{C}_x(n)$ and $\mathbf{C}_{xy}(n)$ are defined in (2.11) and (2.12), and $\mathbf{C}_y(n)$ is defined in the same way as $\mathbf{C}_x(n)$. From (2.10),

$$\begin{aligned} \mathbf{C}_{xy}(n) &= \mathbf{C}_x(n)\hat{\boldsymbol{\Theta}}(n) \\ \text{or} \quad \mathbf{C}_{xy}^H(n) &= \hat{\boldsymbol{\Theta}}^H(n)\mathbf{C}_x^H(n) = \hat{\boldsymbol{\Theta}}^H(n)\mathbf{C}_x(n) \ . \end{aligned} \tag{2.38}$$

This substitution in (2.37) and some simple manipulation yields

$$\left\{\Theta \mid \operatorname{tr}\left\{\Theta^H \mathbf{C}_x(n)\Theta - \Theta^H \mathbf{C}_x(n)\hat{\Theta}(n) - \hat{\Theta}^H(n)\mathbf{C}_x(n)\Theta\right\} < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} - \operatorname{tr}\{\mathbf{C}_y(n)\}\right\} \tag{2.39}$$

Completing the square on the left side yields

$$\begin{aligned}\left\{\Theta \mid \operatorname{tr}\left\{\Theta^H \mathbf{C}_x(n)\Theta - \Theta^H \mathbf{C}_x(n)\hat{\Theta}(n) - \hat{\Theta}^H(n)\mathbf{C}_x(n)\Theta + \hat{\Theta}^H(n)\mathbf{C}_x(n)\hat{\Theta}(n)\right\}\right. \\ \left. < \sum_{i=1}^{n} \frac{\lambda(i)}{\gamma(i)} - \operatorname{tr}\{\mathbf{C}_y(n)\} + \operatorname{tr}\left\{\hat{\Theta}^H(n)\mathbf{C}_x(n)\hat{\Theta}(n)\right\} \doteq \kappa(n)\right\}\end{aligned} \tag{2.40}$$

from which it follows that the set is described by

$$\left\{\Theta \mid \operatorname{tr}\left\{\left[\Theta - \hat{\Theta}(n)\right]^H \mathbf{C}_x(n)\left[\Theta - \hat{\Theta}(n)\right]\right\} < \kappa(n)\right\} . \tag{2.41}$$

If the system is assumed to be *stationary* then $\mathbf{C}_x(n)$ is positive definite, and the left side of this inequality must be a positive number. $\kappa(n)$, therefore, must also be positive. Dividing both sides by $\kappa(n)$ yields (2.31). □

To prove *Corollary 1*, it is convenient to write

$$\operatorname{tr}\left\{\left[\Theta - \hat{\Theta}(n)\right]^H \mathbf{C}_x(n)\left[\Theta - \hat{\Theta}(n)\right]\right\} = \sum_{j=1}^{k} c_j \tag{2.42}$$

where c_j indicates the j^{th} diagonal element of $\left[\Theta - \hat{\Theta}(n)\right]^H \mathbf{C}_x(n)\left[\Theta - \hat{\Theta}(n)\right]$. Now it is clear that

$$c_i = \left[\theta_i - \hat{\theta}_i(n)\right]^H \mathbf{C}_x(n)\left[\theta_i - \hat{\theta}_i(n)\right] \tag{2.43}$$

for any i, where θ_i and $\hat{\theta}_i(n)$ are the i^{th} columns of Θ and $\hat{\Theta}(n)$. It is also true that all the c_j's are positive since $\mathbf{C}_x(n)$ is a positive definite matrix. Therefore,

$$c_i < \sum_{j=1}^{k} c_j < \kappa(n) \quad \text{for any } i = 1, 2, \ldots, k . \tag{2.44}$$

Dividing through by $\kappa(n)$ yields inequality (2.33). □

According to *Corollary 1*, all possible parameter vectors, $\boldsymbol{\theta}_i$, associated with the output y_i are guaranteed to be in a hyperellipsoidal set which is centered on $\hat{\boldsymbol{\theta}}_i(n)$, described by inequality (2.33). Further, the ellipsoids are *identical for each* i except for the centers. It therefore is reasonable to use $\lambda(n)$ which maximally shrinks this common ellipsoid if such can be found. Following the same arguments as in the real scalar case (see *Section 1.2.3.1*), the quantity

$$\det \mathbf{B}(n) \doteq \det \kappa(n)\mathbf{C}_x^{-1}(n) \tag{2.45}$$

is proportional to the volume of ellipsoid $\tilde{\boldsymbol{\Omega}}_i(n)$ of inequality (2.33). A reasonable strategy is to find $\lambda^*(n)$ at each step which minimizes the "volume ratio" of the ellipsoids at n and $n-1$:

$$V(\lambda(n)) = \frac{\det \mathbf{B}(n)}{\det \mathbf{B}(n-1)} \,. \tag{2.46}$$

Theorem 4 *The weight, $\lambda^*(n)$, which minimizes the volume ratio (2.46), is the most positive root of the quadratic equation*

$$F(\lambda) = \alpha_2\lambda^2 + \alpha_1\lambda + \alpha_0 = 0 \tag{2.47}$$

where,

$$\begin{aligned}
\alpha_2 &= (mk-1)G^2(n) \\
\alpha_1 &= \left[2mk - 1 + \gamma(n)\ tr\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\} - \kappa(n-1)\gamma(n)G(n)\right] G(n) \\
\alpha_0 &= mk\left[1 - \gamma(n)\ tr\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\}\right] - \kappa(n-1)\gamma(n)G(n) \,.
\end{aligned}$$

Proof of Theorem 4: (Guided by Deller [45]) Substituting the definition of $\mathbf{B}(n)$

into (2.13) results in

$$\frac{\mathbf{B}(n)}{\kappa(n)} = \frac{\mathbf{B}(n-1)}{\kappa(n-1)} - \lambda(n)\frac{\mathbf{B}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^H(n)\mathbf{B}(n-1)}{\kappa^2(n-1)\left[1+\lambda(n)G(n)\right]} \ . \tag{2.48}$$

Defining $h(n) \doteq 1 + \lambda(n)G(n)$ and $r(n) \doteq \kappa(n)/\kappa(n-1)$ yields

$$\mathbf{B}(n) = \mathbf{B}(n-1)r(n)\left\{\mathbf{I} - \frac{\lambda(n)\boldsymbol{x}(n)}{\kappa(n-1)h(n)}\left[\mathbf{B}(n-1)\boldsymbol{x}(n)\right]^H\right\} \ . \tag{2.49}$$

So,

$$\det\mathbf{B}(n) = \det\mathbf{B}(n-1)\ \det\left\{r(n)\mathbf{I} - \frac{r(n)\lambda(n)\boldsymbol{x}(n)}{\kappa(n-1)h(n)}\left[\mathbf{B}(n-1)\boldsymbol{x}(n)\right]^H\right\} \ . \tag{2.50}$$

Using the matrix identity [46] (for the complex case)

$$\det(c\mathbf{I} + \boldsymbol{y}\boldsymbol{z}^H) = c^{mk-1}(c + \boldsymbol{y}^H\boldsymbol{z}) \tag{2.51}$$

where $\boldsymbol{y}$ and $\boldsymbol{z} \in \mathcal{C}^{mk}$ and c is a real number. The term inside the braces becomes

$$r^{mk-1}(n)\left\{r(n) - \frac{r(n)\lambda(n)\boldsymbol{x}^H(n)}{\kappa(n-1)h(n)}\mathbf{B}(n-1)\boldsymbol{x}(n)\right\} \ . \tag{2.52}$$

Using (2.49), the term in (2.52) can be written as

$$r^{mk}(n)\left\{1 - \frac{\lambda(n)G(n)}{h(n)}\right\} = \frac{r^{mk}(n)}{h(n)} \ . \tag{2.53}$$

Therefore, to minimize the volume ratio (2.46) with respect to $\lambda(n)$, (2.53) is differentiated and the result is set to zero.

$$\frac{\partial}{\partial\lambda(n)}\left(\frac{r^{mk}(n)}{h(n)}\right) \equiv 0 \tag{2.54}$$

$$\Rightarrow \quad mkh(n)r^{mk-1}(n)\frac{\partial r(n)}{\partial \lambda(n)} - r^{mk}(n)G(n) \equiv 0 \; . \tag{2.55}$$

Since $r^{mk-1}(n) \neq 0$,

$$mkh(n)\frac{\partial r(n)}{\partial \lambda(n)} - r(n)G(n) = 0 \; . \tag{2.56}$$

The partial derivative of $r(n)$ is further expanded as follows: Inserting the right side of (2.14) into (2.32) for $\hat{\mathbf{\Theta}}(n)$ gives

$$\begin{aligned} \kappa(n) \;=\; & \mathrm{tr}\left\{\left[\hat{\mathbf{\Theta}}(n-1) + \lambda(n)\mathbf{P}(n)\boldsymbol{x}(n)\epsilon_{n-1}^{H}(n)\right]^{H}\mathbf{P}^{-1}(n)\right. \\ & \left.\left[\hat{\mathbf{\Theta}}(n-1) + \lambda(n)\mathbf{P}(n)\boldsymbol{x}(n)\epsilon_{n-1}^{H}(n)\right]\right\} + \sum_{i=1}^{n}\frac{\lambda(i)}{\gamma(i)} - \mathrm{tr}\left\{\mathbf{C}_{y}(n)\right\} \end{aligned} \tag{2.57}$$

$$\begin{aligned} = \; & \mathrm{tr}\left\{\hat{\mathbf{\Theta}}^{H}(n-1)\left[\mathbf{P}^{-1}(n-1) + \lambda(n)\boldsymbol{x}(n)\boldsymbol{x}^{H}(n)\right]\hat{\mathbf{\Theta}}(n-1) + \right. \\ & \lambda(n)\epsilon_{n-1}(n)\boldsymbol{x}^{H}(n)\hat{\mathbf{\Theta}}(n-1) + \lambda(n)\hat{\mathbf{\Theta}}^{H}(n-1)\boldsymbol{x}(n)\epsilon_{n-1}^{H}(n) + \\ & \left.\lambda^{2}(n)\epsilon_{n-1}(n)\boldsymbol{x}^{H}(n)\left[\mathbf{P}(n-1) - \lambda(n)\frac{\mathbf{P}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^{H}(n)\mathbf{P}(n-1)}{1+\lambda(n)G(n)}\right]\boldsymbol{x}(n)\epsilon_{n-1}^{H}(n)\right\} \\ & + \sum_{i=1}^{n}\frac{\lambda(i)}{\gamma(i)} - \mathrm{tr}\left\{\mathbf{C}_{y}(n)\right\} \end{aligned} \tag{2.58}$$

$$\begin{aligned} = \; & \kappa(n-1) + \frac{\lambda(n)}{\gamma(n)} - \lambda(n)\,\mathrm{tr}\left\{\boldsymbol{y}(n)\boldsymbol{y}^{H}(n)\right\} + \\ & \lambda(n)\,\mathrm{tr}\left\{\left[\boldsymbol{y}(n) - \boldsymbol{\epsilon}_{n-1}(n)\right]\left[\boldsymbol{y}(n) - \boldsymbol{\epsilon}_{n-1}(n)\right]^{H} + \boldsymbol{\epsilon}_{n-1}(n)\left[\boldsymbol{y}(n) - \boldsymbol{\epsilon}_{n-1}(n)\right]^{H} + \right. \\ & \left.\left[\boldsymbol{y}(n) - \boldsymbol{\epsilon}_{n-1}(n)\right]\epsilon_{n-1}^{H}(n) + \frac{\lambda^{2}(n)\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)G(n)}{1+\lambda(n)G(n)}\right\} \end{aligned} \tag{2.59}$$

$$= \; \kappa(n-1) + \frac{\lambda(n)}{\gamma(n)} - \lambda(n)\,\mathrm{tr}\left\{\boldsymbol{\epsilon}_{n-1}(n)\epsilon_{n-1}^{H}(n)\left[1 - \frac{\lambda(n)G(n)}{1+\lambda(n)G(n)}\right]\right\} \; . \tag{2.60}$$

Therefore,

$$\kappa(n) = \kappa(n-1) + \frac{\lambda(n)}{\gamma(n)} - \frac{\lambda(n)\,\mathrm{tr}\left\{\boldsymbol{\epsilon}_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}}{h(n)} \tag{2.61}$$

or,

$$r(n) = \frac{\kappa(n)}{\kappa(n-1)} = 1 + \frac{\lambda(n)}{\kappa(n-1)\gamma(n)} - \frac{\lambda(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}}{\kappa(n-1)h(n)} \ . \tag{2.62}$$

Differentiating this result with respect to $\lambda(n)$ yields

$$\frac{\partial r(n)}{\partial \lambda(n)} = \frac{1}{\kappa(n-1)}\left[\frac{1}{\gamma(n)} - \frac{\mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}}{h^2(n)}\right] \ . \tag{2.63}$$

Putting this result in (2.56) and replacing $\kappa(n)$ with the right side of (2.61) yields

$$\frac{mkh(n)}{\kappa(n-1)}\left\{\frac{1}{\gamma(n)} - \frac{\mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}}{h^2(n)}\right\} -$$
$$\left\{1 + \frac{\lambda(n)}{\kappa(n-1)\gamma(n)} - \frac{\lambda(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}}{\kappa(n-1)h(n)}\right\} G(n) = 0 \tag{2.64}$$

or,

$$mk\left[h^2(n) - \gamma(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}\right] -$$
$$\left[\kappa(n-1)\gamma(n)h(n) + \lambda(n)h(n) - \lambda(n)\gamma(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}\right] G(n) = 0 \ . \tag{2.65}$$

When $h(n)$ is replaced by its definition (see (2.49)), the following result is obtained after some manipulation

$$\lambda^2(n)\left\{(mk-1)G^2(n)\right\} +$$
$$\lambda(n)\left\{2mk - 1 + \gamma(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\} - \kappa(n-1)\gamma(n)G(n)\right\} G(n) +$$
$$mk\left[1 - \gamma(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^{H}(n)\right\}\right] - \kappa(n-1)\gamma(n)G(n) = 0 \tag{2.66}$$

which is the quadratic (2.47).

Finally, it is noted that the α_2 coefficient of the quadratic is always positive, so

that $F(\lambda)$ is concave upward. It follows immediately that the larger real root of $F(\lambda) = 0$ (if it exists) will correspond to a local minimum of $V(\lambda(n))$. This root must be real and positive to be a valid weight (see *Lemma 1*). □

Theorems 1 and 2 can be considered as special cases of *Theorems 3 and 4* in which the data are real scalar quantities. Another special case can also be derived when the data are complex scalar quantities and the fundamental results are stated below.

Consider the model of the form

$$y(n) \doteq \boldsymbol{\theta}_0^H \boldsymbol{x}(n) + v(n) \tag{2.67}$$

with the constraint

$$\gamma(n) \mid v(n) \mid^2 < 1 \tag{2.68}$$

and an error term

$$\epsilon_{n-1}(n) \doteq y(n) - \hat{\boldsymbol{\theta}}^H(n-1)\boldsymbol{x}(n) \; . \tag{2.69}$$

All the quantities above are complex except $\gamma(n)$ which is real.

The hyperellipsoidal membership set associated with this special case is easily defined by noting that *Corollary 1* applies directly to this case with $\boldsymbol{\theta}_i$ and $\hat{\boldsymbol{\theta}}_i$ replaced by $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$, and $\mathrm{tr}\{\mathbf{C}_y(n)\}$ (in *Theorem 3*) by $\sum_{i=1}^{n} \lambda(i) \mid y(i) \mid^2$. Similarly, the quadratic equation (2.47) can be applied with $\mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\}$ replaced by $\mid \epsilon_{n-1}(n) \mid^2$.

2.3 Suboptimal Tests for Innovation in SM-WRLS Algorithms

A significant reduction in computational complexity can be achieved by employing a "suboptimal" test for information content in an incoming equation. The proposed check is argued to be a useful determiner of the ability of incoming data to shrink the ellipsoid, but it does not rigorously determine the existence of an optimal SM weight in the sense described above. The main issue here is to avoid the computations of the quantities necessary at each step to construct and solve the quadratic (2.47) in cases in which the quadratic turns out only to be useful for the purpose of checking for the existence of a meaningful weight. Since most of the time these computations result in the rejection of incoming data, a more efficient test could significantly reduce the complexity of the algorithm.

The estimation error matrix at time n can be denoted by

$$\tilde{\mathbf{\Theta}}(n) \doteq \mathbf{\Theta}_0 - \hat{\mathbf{\Theta}}(n) \ . \tag{2.70}$$

The following inequality results immediately from (2.31),

$$\tilde{\mathbf{\Theta}}^H(n)\mathbf{C}_x(n)\tilde{\mathbf{\Theta}}(n) < \kappa(n) \ . \tag{2.71}$$

Using a similar inequality (for the real scalar case), Dasgupta and Huang [14] have noted that their $\kappa(n)$-like quantity provides a bound on the error vector (or matrix for the generalized case) sequence and have suggested minimizing this quantity with respect to $\lambda(n)$ in an effort to decrease computational complexity. However, this minimization does not, in general, imply an improvement in the estimate with respect to previous times, since both sides of the inequality (2.71) are dependent upon $\lambda(n)$. Further, the nonexistence of a minimum of $\kappa(n)$ with respect to $\lambda(n)$ is not very

informative in this sense. However, further arguments are presented here to provide support for this process in the SM-WRLS context.

Consider the usual volume quantity to be minimized at time n, defined in (2.45). Let us temporarily write the two key quantities there as functions of $\lambda(n)$: $\mathbf{C}_x(n, \lambda(n))$ and $\kappa(n, \lambda(n))$. It is assumed that enough equations have been included in the covariance matrix at time $n-1$ so that its elements are large with respect to the data in the incoming equation. Now the quantity $\det \mathbf{C}_x(n, \lambda(n))$ is readily shown to be monotonically increasing with respect to $\lambda(n)$ on $\lambda(n) \in [0, \infty)$ (see Appendix A), with $\mathbf{C}_x(n, 0) = \mathbf{C}_x(n-1, \lambda^*(n-1))$, where $\lambda^*(n-1)$ indicates the optimal weight at time $n-1$. Under the assumption above, $\det \mathbf{C}_x(n, \lambda(n))$ will not increase significantly over reasonably small values of $\lambda(n)$. The attempt to maximize $\det \mathbf{C}_x(n, \lambda(n))$ in (2.45) causes a tendency to increase $\lambda(n)$ in the usual optimization process. However, the attempt to minimize $\kappa(n, \lambda(n))$ generally causes a tendency toward small values of $\lambda(n)$, unless a minimum of $\kappa(n, \lambda(n))$ occurs at a "large" value of $\lambda(n)$. To pursue this idea and further points of the argument, key results about $\kappa(n, \lambda(n))$ are noted in the following.

Theorem 5 *$\kappa(n, \lambda(n))$ has the following properties:*

- *On the domain $\lambda(n) \in [0, \infty)$, $\kappa(n, \lambda(n))$ is either monotonically increasing or it has a single minimum.*
- *$\kappa(n, \lambda(n))$ has a minimum on $\lambda(n) \in [0, \infty)$ iff*

$$tr\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\} > \gamma^{-1}(n) \; . \tag{2.72}$$

Lemma 2 *Let $\lambda^*(n-1)$ denote the optimal weight in the sense of Theorem 4 (which might be zero) at time $n-1$. Then*

$$\kappa(n,\lambda(n)) = \kappa(n-1,\lambda^*(n-1)) + \frac{\lambda(n)}{\gamma(n)} - \left[\frac{\lambda(n)\ tr\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\}}{1+\lambda(n)G(n)}\right] . \quad (2.73)$$

Proof of Lemma 2: See (2.61).

Proof of Theorem 5: The minimum of $\kappa(n,\lambda(n))$ with respect to $\lambda(n)$ can be found by differentiating (2.73) and setting the result equal to 0,

$$\frac{\partial\kappa(n,\lambda(n))}{\partial\lambda(n)} = G^2(n)\lambda^2(n) + 2G(n)\lambda(n) + \left[1-\gamma(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\}\right] \equiv 0 . \quad (2.74)$$

This is a concave upward quadratic function with its minimum at

$$\lambda'(n) = -G^{-1}(n) < 0 . \quad (2.75)$$

Two *real* roots of (2.74) always exist,

$$\lambda_{roots}(n) = \frac{-1 \pm \sqrt{\gamma(n)\ \mathrm{tr}\left\{\epsilon_{n-1}(n)\epsilon_{n-1}^H(n)\right\}}}{G(n)} \quad (2.76)$$

the smaller corresponding to a maximum of $\kappa(n,\lambda(n))$, the larger to a minimum. Only the larger root can be positive since the lower root is bound to be less than $\lambda'(n)$. Therefore, it is only possible for $\kappa(n,\lambda(n))$ to exhibit a minimum or to be increasing on positive $\lambda(n)$. It is easy to use (2.76) to verify that the larger root is positive *iff* condition (2.72) is met. □

With these results, it can be argued that: If $\det \mathbf{C}_x(n,\lambda(n))$ is increasing, but not changing significantly over reasonably small values of $\lambda(n)$, then it is sufficient to seek $\lambda(n)$ which minimizes $\kappa(n,\lambda(n))$. If $\kappa(n,\lambda(n))$ is monotonically increasing on

$\lambda(n) \geq 0$, this value is $\lambda(n) = 0$ which corresponds to rejection of the equation at time n. It suffices, therefore to have a test for a minimum of $\kappa(n, \lambda(n))$ on positive $\lambda(n)$. As noted above, a simple test is embodied in condition (2.72). If this test is met, it is then cost effective to proceed with the standard optimization process centered on (2.47). Otherwise, the explicit construction and solution of (2.47) can be avoided.

It is to be noted that even if (2.72) is met, it is possible that the optimization procedure will still reject the datum. Perhaps more importantly, it is also possible for (2.72) to reject data which would have been accepted by the usual process. These ideas will be explored in the simulation studies (*Chapter 3*).

Finally, note that when the simplified test (2.72) accepts the new equation, there are tools to compute the weight which is "optimal" in the sense of minimizing $\kappa(n, \lambda(n))$. In particular, this would be the larger of the roots in (2.76). However, it clearly makes more sense to compute *the* optimal weight according to (2.47), since this computation is not much more expensive. The improvement in the computational complexity due to "suboptimal checking" is discussed in *Chapter 4*.

2.4 Adaptive SM-WRLS Algorithms

While the theoretical developments of the previous two sections, in principle, provide the background for further SM-WRLS developments with general complex vector inputs and outputs, it is upon the special case of real scalar data that most of the remaining work will focus. This special case was necessary in order to initiate a tractable study of the difficult issues of adaptation, algorithmic behavior, and architecture development. The work on this simpler case which is to follow, however, will lay the groundwork for future studies on the more general cases.

2.4.1 General Formulation

The adaptive algorithm presented here uses "back rotation" in order to partially or completely "forget" past information enabling it to track (potentially fast) time varying signals. Back rotation [40] is a Givens rotation-based technique that removes (or rotates out) a previously included equation from the system. In this section, the back rotation technique is modified such that a previous equation can be *partially* removed. This will permit a broader class of adaptive strategies. In SM terms, back rotation causes the ellipsoidal membership set to expand due to the removal of information. This expansion entices the algorithm to incorporate present data. The back rotation technique requires that all the weights with the corresponding equations (for weights other than zero) be stored for later use.

Recalling Fig. 1.2, it is seen that at each step in the SM-WRLS algorithm, the upper triangular system of simultaneous equations $\mathbf{T}(n)\hat{\boldsymbol{\theta}}(n) = \boldsymbol{d}_1(n)$, is solved (when data are accepted) to obtain the optimal estimate [6, 7, 9]. Suppose in approaching time n that the past equation to be (partially) removed is at time τ. Rotating this equation out of the system is accomplished by re-introducing it as though it were a new equation. A weight $\sqrt{\mu\lambda(\tau)}$, where μ is the fraction of the equation to be removed from the system, is used, and some sign changes in the rotation equations are necessary [40]. The system of equations with τ removed is referred to as the "downdated" system at time $n-1$, and the related quantities are labeled with subscript d, i.e.,

$$\mathbf{T}_d(n-1)\hat{\boldsymbol{\theta}}_d(n-1) = \boldsymbol{d}_{1,d}(n-1)\ . \tag{2.77}$$

The downdated ellipsoid matrix is $\mathbf{C}_d(n-1)/\kappa_d(n-1)$ where

$$\mathbf{C}_d(n-1) = \mathbf{T}_d^T(n-1)\mathbf{T}_d(n-1)\ , \tag{2.78}$$

$$\kappa_d(n-1) = \| \boldsymbol{d}_{1,d}(n-1) \|_2^2 + \tilde{\kappa}_d(n-1)\ , \tag{2.79}$$

with

$$\tilde{\kappa}_d(n-1) \doteq \tilde{\kappa}(n-1) - \frac{\mu\lambda(\tau)}{\gamma(\tau)}\left(1-\gamma(\tau)y^2(\tau)\right) \tag{2.80}$$

in which $\tilde{\kappa}(n-1)$ represents the updated value of $\tilde{\kappa}$ which includes the equation at time $n-1$. Equations (2.79) and (2.80) follow immediately from the definition of κ found in (1.23). These relations can be used repeatedly regardless of the number of equations (partially or completely) removed prior to time n. If more than one equation is removed prior to n, $\tilde{\kappa}(n-1)$ in the right side of (2.80) is replaced by the $\tilde{\kappa}_d(n-1)$ for all downdates after the first one. Following all necessary downdating just prior to time n, the algorithm uses the downdated system to compute the quantities $G_d(n)$ and $\epsilon_{n-1,d}(n)$ which are necessary to compute the optimal weight for the equation at n. To compute a downdated SM-WRLS estimate, therefore, it is only necessary to downdate the matrix $\mathbf{T}(n-1)$ and the vector $\boldsymbol{d}_1(n-1)$ and to solve for $\hat{\boldsymbol{\theta}}_d(n-1)$ prior to Step 1 in Fig. 1.2, then replace all relevant quantities in Step 1 by their downdated versions, i.e.,

$$G_d(n) = \boldsymbol{x}^T(n)\mathbf{T}_d^{-1}(n-1)\mathbf{T}_d^{-T}(n-1)\boldsymbol{x}(n) = \| \boldsymbol{g}_d(n) \|_2^2, \tag{2.81}$$

and

$$\epsilon_{n-1,d}(n) = y(n) - \hat{\boldsymbol{\theta}}_d^T(n-1)\boldsymbol{x}(n)\ . \tag{2.82}$$

$\kappa_d(n-1)$ and $\tilde{\kappa}_d(n-1)$ are downdated according to (2.79) and (2.80). Then $\lambda^*(n)$ is found in Step 2 using (1.26) with downdated quantities. Note that downdating is unnecessary if the equation τ was rejected by SM-WRLS. In this case $\mathbf{T}_d(n-1) = \mathbf{T}(n-1)$ and $\hat{\boldsymbol{\theta}}_d(n-1) = \hat{\boldsymbol{\theta}}(n-1)$. Conversely, when the "new" equation at n is rejected, then $\mathbf{T}(n) = \mathbf{T}_d(n-1)$ and $\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}_d(n-1)$.

A wide range of adaptation strategies is inherent in the general formulation described above. Three major subcases are identified in the following. In each of these subcases, the objective (in SM terms) is to expand the ellipsoidal region of possible

solutions in order to track fast time variations in the signal.

2.4.1.1 Windowing

Windowed adaptation is a special subcase of the general formulation which uses a sliding window of fixed length, l ($l \geq m+1$), so that the estimate at time n covers the range $[n-l+1, n]$. The windowing technique is made possible by the ability to completely (i.e., $\mu = 1$) and systematically remove equations at the trailing edge of the window. An illustration of this technique is shown in [5].

This technique implements the same procedure as that of the basic SM-WRLS algorithm for $i = 1, 2, \ldots, l$, and thus, exhibits similar performance. The initialization process (for this strategy and all other GR-based SM-WRLS strategies) is also the same as that of the basic SM-WRLS algorithm; i.e., the working matrix $\mathbf{W}$ (or $\mathbf{T}$) is filled with zeros, $\lambda(i) = 1$ for $i = 1, 2, \ldots, m+1$, and $\tilde{\kappa}(0) = 0$ (see Fig. 1.2). Windowed adaptation starts at time $l+1$ and works as follow: Prior to consideration of the equation at time n ($n \geq l+1$), it is simply necessary to remove the equation at time $\tau = n-l$ from the system by complete "downdating" as described above. All the theoretical results of the general formulation are valid with the value $\tilde{\kappa}_d(n-1)$ given by

$$\begin{aligned} \tilde{\kappa}_d(n-1) &= \sum_{i=n-l+1}^{n-1} \frac{\lambda(i)}{\gamma(i)} \Big(1 - \gamma(i) y^2(n)\Big) \\ &= \tilde{\kappa}(n-1) - \frac{\lambda(n-l)}{\gamma(n-l)} \Big(1 - \gamma(n-l) y^2(n-l)\Big) \end{aligned} \tag{2.83}$$

in which the value $n-l$ acts like a special τ for the windowed adaptation.

2.4.1.2 Graceful Forgetting

In this technique, only a fraction, μ (chosen such that μ^{-1} is an integer), of all previously included equations is removed at each n similarly to the exponential forgetting

factor conventionally used with WRLS[3]. This technique has a sliding window of fixed length, μ^{-1}, with the effective weights decreasing linearly when moving toward the trailing edge of the window. Hence, the equation at the trailing edge of the window has an effective weight of $\sqrt{\mu\lambda(n+1-\mu^{-1})}$ and the equation just rotated in has an effective weight of $\sqrt{\lambda(n)}$. Note that although each equation must be partially rotated out μ^{-1} times, only those equations that were previously accepted (in the past μ^{-1} recursions) need to be considered by the algorithm.

2.4.1.3 Selective Forgetting

This technique selectively chooses the equations to be (partially or completely) removed from the system based on certain user defined criteria in order to remove their influence from the system. The selection process can be, for example, to remove (or downweight) only the previously heavily weighted equations, to remove the equations that were accepted in regions of abrupt change in the signal dynamics, or to remove the equations starting from the first equation and proceeding sequentially. Whatever the criteria, a fundamental issue is to detect *when* adaptation is needed to improve the parameter estimates. This issue is further investigated in *Chapter 3.*

Therefore, the adaptive SM-WRLS algorithm and its extensions (e.g., the special subcases described above) are potentially very useful techniques that can be applied to identify models with fast time varying signals. Due to the flexible nature of this algorithm, however, various subcases can also be defined and tested. In any subcase, the objective (as noted above) is always to expand the ellipsoidal region of possible solutions. This objective can also be achieved in an unconventional but intuitive method. For example, an *ad hoc* adaptation strategy may be to inflate the ellipsoid (i.e., multiply the matrix $\mathbf{C}_x(n)/\kappa(n)$) by a scale factor, say α $(0 < \alpha < 1)$, when

[3]An exponentially forgetting factor can also be incorporated into SM-WRLS. Although the strategy is computationally very efficient, it has not been found to be effective for adaptation in preliminary experiments. See *Section 2.4.2* for theoretical development and discussion.

there is a need for adaptation.

It is important to note that the general adaptive formulation is amenable to the suboptimal strategy presented in *Section 2.3*. The performance of the adaptive, suboptimal, and *adaptive* suboptimal techniques are investigated in *Chapter 3*.

2.4.2 Exponential Forgetting Factor Adaptation

In this section, it is shown that exponential forgetting factor[4] adaptation can be incorporated into SM-WRLS. The theoretical development of this computationally efficient strategy is presented, followed by a discussion explaining why this method is not found to be effective for adaptation in simulations to date.

It is possible to make the *conventional* WRLS solution by GR's *adaptive* by multiplying the existing system of equations of the form $\mathbf{T}(n)\hat{\boldsymbol{\theta}}(n) = \boldsymbol{d}_1(n)$, through by a "forgetting factor," β, where, $0 < \beta < 1$, prior to rotation of the equation at time $n+1$ into the system [40, 43, 44]. Since this process nominally inflates the ellipsoid size by removing information, the optimization at the next time must be performed with respect to this "intermediate" system, say

$$\mathbf{T}_a(n)\hat{\boldsymbol{\theta}}(n) = \boldsymbol{d}_{1,a}(n) \ , \tag{2.84}$$

where $\mathbf{T}_a(n) = \beta\mathbf{T}(n)$ and $\boldsymbol{d}_{1,a}(n) = \beta\boldsymbol{d}_1(n)$. Note that in order to use (1.26), "adapted" versions of ϵ, G, and κ must be used. It is easy to see that

$$\epsilon_{n,a}(n+1) = \epsilon_n(n+1) \tag{2.85}$$

[4]This is a more conventional "forgetting factor" strategy than that suggested by Dasgupta and Huang in [14].

(since $\hat{\boldsymbol{\theta}}(n)$ is unchanged), and that

$$G_a(n+1) = \beta^{-2} G(n+1) \; . \tag{2.86}$$

Recalling (1.23) and the alternative expression for $\kappa(n)$ in (1.34), and noting that

$$\mathbf{C}_{x,a}(n) = \mathbf{T}_a^T(n)\mathbf{T}_a(n) = \beta^2 \mathbf{T}^T(n)\mathbf{T}(n) = \beta^2 \mathbf{C}_x(n) \; , \tag{2.87}$$

the leading term of $\kappa_a(n)$ is easily shown to be $\| \boldsymbol{d}_{1,a}(n) \|_2^2 = \beta^2 \| \boldsymbol{d}_1(n) \|_2^2$. To complete $\kappa_a(n)$, note the fact that, at time $n+1$, this method effectively assigns weight $\beta^{n+1-i}\sqrt{\lambda(i)}$ to the equation at time i, where $\sqrt{\lambda(i)}$ is the weight assigned to the equation when originally rotated into the system [44]. Therefore, using (1.34) and the discussion above, yields

$$\kappa_a(n) = \beta^2 \| \boldsymbol{d}_1(n) \|_2^2 + \tilde{\kappa}_a(n) \tag{2.88}$$

where,

$$\begin{aligned} \tilde{\kappa}_a(n) &= \sum_{i=1}^{n} \frac{(\beta^{n+1-i})^2 \lambda(i)}{\gamma(i)} \Big(1 - \gamma(i) y^2(n)\Big) \\ &= \beta^2 \tilde{\kappa}_a(n-1) + \frac{\beta^2 \lambda(n)}{\gamma(n)} \Big(1 - \gamma(n) y^2(n)\Big) \; . \end{aligned} \tag{2.89}$$

To introduce exponential forgetting factor adaptation into the SM-WRLS algorithm, therefore, it is only necessary to downweight the matrix $\mathbf{T}(n-1)$ and the vector $\boldsymbol{d}_1(n-1)$ by multiplying through by β prior to Step 1 in Fig. 1.2, then to replace all relevant quantities in Step 1 by their downweighted versions. $\kappa_a(n-1)$ and $\tilde{\kappa}_a(n-1)$ are updated according to (2.88) and (2.89). Then $\lambda^*(n)$ is found in Step 2 using (1.26) with downweighted quantities.

In preliminary experiments, it has been found that this strategy is not effective

for adaptation. Actually, this strategy performs well when applied to *slow* time varying systems but fails to track *fast* time variations. To explain this behavior, it is important to understand the effect of the "downweighting" process. As noted earlier, multiplying the system of equations (at each recursion) by β ($0 < \beta < 1$) inflates the ellipsoid volume by a factor that is inversely proportional to β. Therefore, a large value of β causes the ellipsoid volume to increase slightly which *lessens* the tendency to accept new equations and the ability to track fast time varying signals. On the other hand, a small value of β causes the ellipsoid volume to increase significantly, and therefore, a tendency to accept more equations[5]. However, every time an equation is accepted, the ellipsoid volume decreases accordingly. Hence, a small value of β may result in "bad" estimates due to the continuously expanding and shrinking ellipsoid and, more importantly, to the fact that the effective window length is very small. Therefore, it is desirable to choose β to be large but not to the extent of having insignificant effect on expanding the ellipsoid volume.

To pursue this discussion further, consider the problem of estimating a signal characterized as having *slow* time variations everywhere except in a region around time n at which the signal exhibits *fast* time variations (see, e.g., Fig. 3.3, where $n = 2000$). The fact that the signal is changing very slowly prior to n induces the algorithm to accept some points which, in turn, causes the ellipsoid volume to decrease. An increase in the "confidence" of the estimate results. Near time n, the ellipsoid volume becomes very small. When the signal moves rapidly away from its current location, it eventually moves outside the ellipsoid which is therefore no longer a valid bounding ellipsoid. The situation can be described as follows: The signal parameters are wandering outside of a small ellipsoid, but the incoming equations are being rejected (because of the high confidence in the existing estimate) until some of

[5]Note that one of the main advantages of SM-WRLS is the use of a small fraction of the data. Therefore, from the SM point of view, small values of β are least desirable.

the earlier influential equations are downweighted (or suppressed). Remember that the "downweighting" process is recursively being implemented. Hence, if β is large, it takes a long time to suppress the influence of previously accepted equations, or, in SM terms, the ellipsoid volume increases at a slow rate which may not be sufficient to track the fast time varying signal. This explains why the algorithm ceases accepting new equations for a period of time (which is dependent on β), and therefore, fails to track the signal.

The adaptive SM-WRLS algorithms (of *Section 2.4.1*) do not depend on a fixed factor, such as β, to expand the ellipsoid volume in order to adapt to the changing dynamics of the system. However, these algorithms expand the ellipsoid by (selectively) removing previously accepted influential equations from the system, either partially or completely, and therefore, relinquishing their influence from the current ellipsoid, thereby allowing it to expand and adapt to the changes in the signal dynamics.

2.5 A Survey of the Computational Complexities of Several Related Sequential Algorithms

The purpose of this section is to compare the computational complexities of several related sequential algorithms that solve the LS problem. The computational complexities are shown in Table 2.1 in which the first column indicates the algorithm under study, the second column gives the complexity (in floating point operations (flops) per equation) required to check for a valid SM weight, where one flop is defined as one floating point multiplication plus one floating point addition. The third column gives the complexity required to update (or downdate in the case of back rotation) the covariance matrix and the LS solution, and the fourth column gives the total number of flops per equation for a typical example described below.

There are two theoretically equivalent methods associated with conventional LS

Table 2.1: Computational complexities (in floating point operations (flops) per equation) for the real scalar sequential algorithms.

Algorithm	Checking	Covariance and Solution Update	Example (flops)
Batch Solution	—	$\mathcal{O}(m^3)$	> 1000
MIL-based WRLS	—	$3m^2 + 5m + 3$	353
MIL-based SM-WRLS	$m^2 + 2m + 13$	$2m^2 + 3m + 7$	180
Suboptimal MIL-based SM-WRLS	$(m+1) + s(m^2 + m + 12)$	$2m^2 + 3m + 7$	47
GR-based SM-WRLS	$.5m^2 + 2.5m + 13$	$2.5m^2 + 10.5m + 5$	160
Suboptimal GR-based SM-WRLS	$(m+1) + s(.5m^2 + 1.5m + 12)$	$2.5m^2 + 10.5m + 5$	55

solution; the "batch" solution [39] which requires $\mathcal{O}(m^3)$ flops per equation, and the MIL-based WRLS solution [34, 35] which requires $\mathcal{O}(m^2)$ flops per equation (see *Section 1.2.1*). The SM-WRLS algorithm can also solve the problem in $\mathcal{O}(m^2)$ flops per equation based on MIL (see *Section 1.2.3.1*), however, a GR-based solution (outlined in *Section 1.2.3.2*) is also considered because it is amenable to a systolic architecture implementation which reduces the complexity of the algorithm to $\mathcal{O}(m)$. The latter algorithm is the subject of *Chapter 4*.

Similar computational complexity expressions for the MIL- and GR-based SM-WRLS solutions can be derived for the suboptimal strategy of *Section 2.3*, and are shown in Table 2.1. Note that there is one square root operation associated with each of the SM-WRLS algorithms but has been dropped from the table since it does not have any significant effect on the comparison.

A note about the computational complexity of the OBE algorithm [20] is in order. It was noted by Huang [20] that the complexity of the OBE algorithm is in the order of m^2 multiplications (or flops) for the information evaluations (i.e., checking), while

updating the estimates requires $6m^2$ multiplications (or flops). However, a careful analysis of the OBE algorithm during the course of this work, taking into account the symmetric properties of some matrices, reveals that the complexity of this algorithm is m^2+2m+3 flops for checking and $2m^2+3m+16$ flops for updating the estimates. A similar analysis of a "suboptimal" OBE algorithm [14] shows that the complexity of this algorithm is $m+1$ flops for checking and $3m^2+4m+15$ flops for updating the estimates.

If the fraction of the data accepted by the SM-WRLS algorithm is denoted by r, the fraction of the data accepted by the suboptimal SM-WRLS algorithm by s ($s < r$), and the fraction of the data accepted by the SM-WRLS algorithm after passing the test (2.72) by t ($t \leq s$), then the total computational complexities of the SM-WRLS algorithms shown in Table 2.1 can be defined as follows. For the MIL-based SM-WRLS algorithm, the total computational complexity is given by

$$\left(m^2+2m+13\right)+r\left[2m^2+3m+7\right] \tag{2.90}$$

flops per equation. For the suboptimal MIL-based SM-WRLS algorithm, it is given by

$$(m+1)+s\left[m^2+m+12\right]+t\left[2m^2+3m+7\right] \tag{2.91}$$

flops per equation. For the GR-based SM-WRLS algorithm, it is given by

$$\left(.5m^2+2.5m+13\right)+r\left[2.5m^2+10.5m+5\right] \tag{2.92}$$

flops per equation. Finally, for the suboptimal GR-based SM-WRLS algorithm, it is given by

$$(m+1)+s\left[.5m^2+1.5m+12\right]+t\left[2.5m^2+10.5m+5\right] \tag{2.93}$$

flops per equation.

Let us consider a typical example to compare the complexities of the various algorithms. Assume that the model order $m = 10$, the fraction of the data accepted by the SM algorithms (both SM-WRLS and OBE), r, is 0.2, and the fraction of the data accepted by the suboptimal SM algorithms, s, is 0.1. To simplify the analysis, it is assumed that *all* the equations satisfying condition (2.72) for the suboptimal SM-WRLS algorithm are also accepted by the SM-WRLS algorithm ($s = t$). The total number of flops (per equation) is shown in the fourth column of Table 2.1. According to the computational complexities computed here, the OBE algorithm uses 173 flops and the "suboptimal" OBE algorithm uses 47 flops. The SM algorithms typically use $45 \sim 50\%$ the number of flops required by the MIL-based WRLS algorithm. The computational savings are mainly due to the infrequent updating of the covariance matrix and the LS solution. A suboptimal test which is presented in *Section 2.3* can be incorporated in virtually any version of the SM-WRLS algorithms to further improve the computational efficiency. When the suboptimal strategy is applied to a given algorithm, it reduces the computational complexity of the algorithm by $60 \sim 70\%$.

The adaptive GR-based SM-WRLS algorithms of *Section 2.4.1* use the same computational complexities as those of the "non-adaptive" GR-based SM-WRLS algorithms when performing back substitution to downdate the covariance matrix and the LS solution. The only exceptions being $\tilde{\kappa}_d$ (see (2.80)) which requires one extra flop per equation for cases when $\mu \neq 1$, and the fact that the LS solution needs to be computed (downdated) only after all necessary downdating of the covariance matrix and $\tilde{\kappa}$ at any given time (see *Section 2.4.1*). If the fractions of the data used by the adaptive GR-based SM-WRLS algorithms are denoted by the same symbols used above, and the fraction of the data removed from the system is denoted by u, then the total computational complexities of the adaptive SM-WRLS algorithms are as follows. For the windowed and selective forgetting SM-WRLS algorithms, the total

computational complexity is given by

$$\left(.5m^2 + 2.5m + 13\right) + r\left[2.5m^2 + 10.5m + 5\right] + u\left[2m^2 + 10m + 5\right] \qquad (2.94)$$

flops per equation. For the suboptimal windowed and selective forgetting SM-WRLS algorithms, it is given by

$$(m+1) + s\left[.5m^2 + 1.5m + 12\right] + t\left[2.5m^2 + 10.5m + 5\right] + u\left[2m^2 + 10m + 5\right] \quad (2.95)$$

flops per equation. For the graceful forgetting SM-WRLS algorithm, it is given by

$$\left(.5m^2 + 2.5m + 13\right) + r\left[2.5m^2 + 10.5m + 5\right] + \mu^{-1}u\left[2m^2 + 10m + 6\right] \qquad (2.96)$$

flops per equation. Finally, for the suboptimal graceful forgetting SM-WRLS algorithm, it is given by

$$(m+1) + s\left[.5m^2 + 1.5m + 12\right] + t\left[2.5m^2 + 10.5m + 5\right] + \mu^{-1}u\left[2m^2 + 10m + 6\right] \qquad (2.97)$$

flops per equation.

Consider the same example with the new assumption that the fraction of the data removed from the system, u, is half that of the data accepted (i.e., $u = 0.1$ for the SM-WRLS algorithm and 0.05 for the suboptimal strategy). Both the windowed and the selective forgetting strategies have the same complexity (191 flops per equation) which is reduced to 67 flops per equation when the adaptive suboptimal strategy is employed.

Using the same fractions of data and a μ value of 0.005, the graceful forgetting strategy use a total of 6280 flops per equation which is reduced to 2565 flops per equation for the adaptive suboptimal strategy. This larger number is due to the fact

Table 2.2: Computational complexities (in complex floating point operations (cflops) per equation) for the generalized sequential algorithms.

MIL-based Algorithm	Checking	Covariance and Solution Update	Example (cflops)
WRLS	—	$3(mk)^2 + (2k+3)mk + k + 2$	32312
SM-WRLS	$(mk)^2 + (k+1)mk + k + 12$	$2(mk)^2 + (k+2)mk + k + 6$	15365
Suboptimal SM-WRLS	$(mk^2 + k) + s\left[(mk)^2 + mk + 12\right]$	$2(mk)^2 + (k+2)mk + k + 6$	7276

that each equation must be partially rotated out μ^{-1} (or 200 in this case) times. Although this technique "forgets gracefully" and quickly adapts to the rapid changes in the signal dynamics, it is computationally expensive. However, the adaptive systolic architecture (of *Section 4.3*) reduces the computational complexity of the "sequential" adaptive GR-based algorithm by 60%.

In *Chapter 4*, the GR-based SM-WRLS algorithm is mapped into a systolic architecture for speed advantages. The (parallel) complexities of the parallel GR-based SM-WRLS algorithm and its suboptimal version are discussed in *Chapter 4*, however, it is worth noting that, compared to the complexity of the "sequential" GR-based algorithm, the systolic architecture implementation reduces the complexity of this algorithm by about 60%, and when the suboptimal strategy is employed, the complexity is reduced by 84%.

In order to find the computational complexity of the generalized SM-WRLS algorithm developed in *Section 2.2*, let us define a complex flop (cflop) as four real floating point multiplications plus four real floating point additions. The computational complexities of the MIL-based WRLS, SM-WRLS, and suboptimal SM-WRLS algorithms are shown in Table 2.2. The fourth column in this table shows the total

number of cflops (per equation) when using the same example used for the real scalar case with $k = 10$. The SM-WRLS algorithm typically uses $45 \sim 50\%$ the number of cflops required by the MIL-based WRLS algorithm which is consistent with the real scalar case (see Table 2.1). However, when the suboptimal strategy is applied, it reduces the computational complexity of the algorithm by $75 \sim 80\%$. The computational savings for the complex vector case are more than those reported for the real scalar case because of the fact that the suboptimal strategy performs "scalar" checking compared with "vector" updating. However, the computational complexity of the generalized sequential GR-based SM-WRLS algorithm is expected to be worse than that of the MIL-based algorithm due to the fact that the LS solution matrix has to be solved for one vector at a time.

Chapter 3

Simulation Studies

3.1 Introduction

SM algorithms have the potential for application to many real digital signal processing problems such as speech recognition, image processing, beamforming, spectral estimation, and neural networks. This chapter is concerned with testing the behavior of the various suboptimal and adaptive SM-WRLS strategies presented in *Chapter 2* for the real scalar case. This is done by conducting extensive simulation studies which illustrate important points about these strategies and about the SM-WRLS algorithm in general. The simulation studies are performed on models derived from real speech data. *Section 3.2* discusses the results using a model of order two so that the results can be easily illustrated. The performance of a more realistic model of order 14 is analyzed in *Section 3.3.* These simulation studies are carried out on a 32-bit machine, however, it is important to research the behavior of the SM-WRLS algorithm on a smaller wordlength. The performance of this algorithm is tested using a 16-bit wordlength and is discussed in *Section 3.4.*

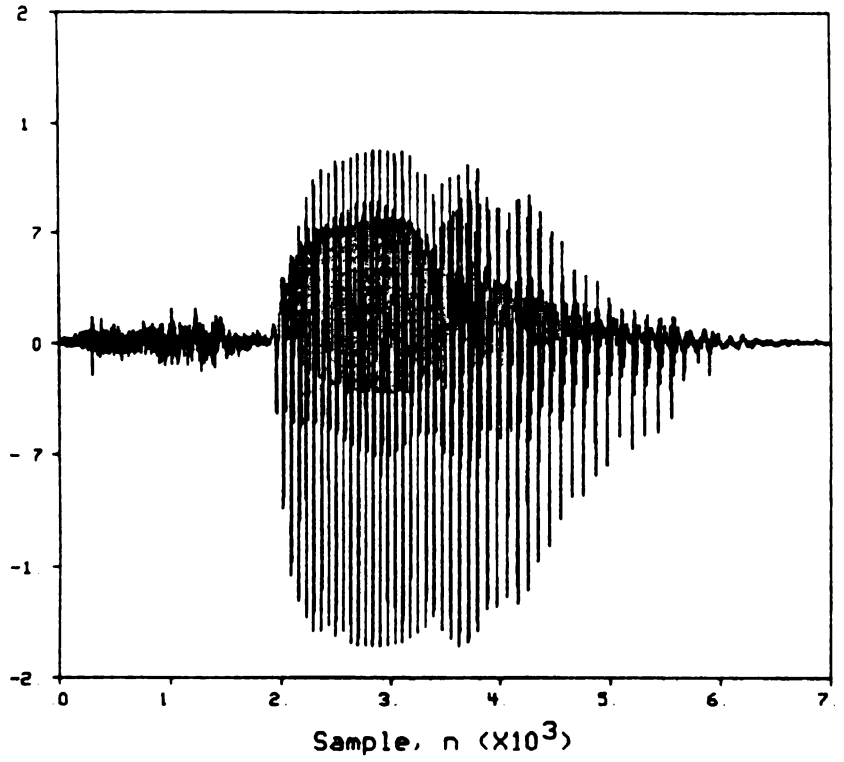


Figure 3.1: The acoustic waveform of the word "four".

3.2 Simulation Results of two AR(2) models

In this section, the identification of two time varying *AR(2)* models of the form

$$y(n) = a_1(n)y(n-1) + a_2(n)y(n-2) + v(n) \tag{3.1}$$

is considered. Two sets of AR parameters were derived using linear prediction (LP) analysis of order two on utterances of the words "four" and "six" by an adult male speaker. The acoustic waveforms of these two words are shown in Figs. 3.1 and 3.2. While more meaningful analysis of speech would involve model orders of 10-14 (see, e.g., [47]), this small number of parameters is used here so that the results are easily illustrated. The adaptive LP algorithm used to compute these parameters is described in [44]. The data were sampled at 10 kHz after 4.7 kHz lowpass filtering, and

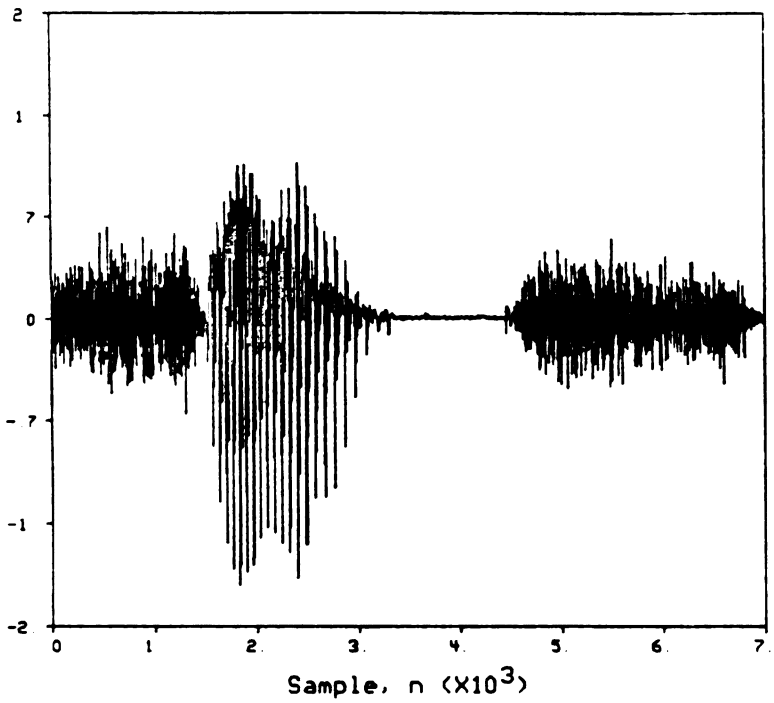
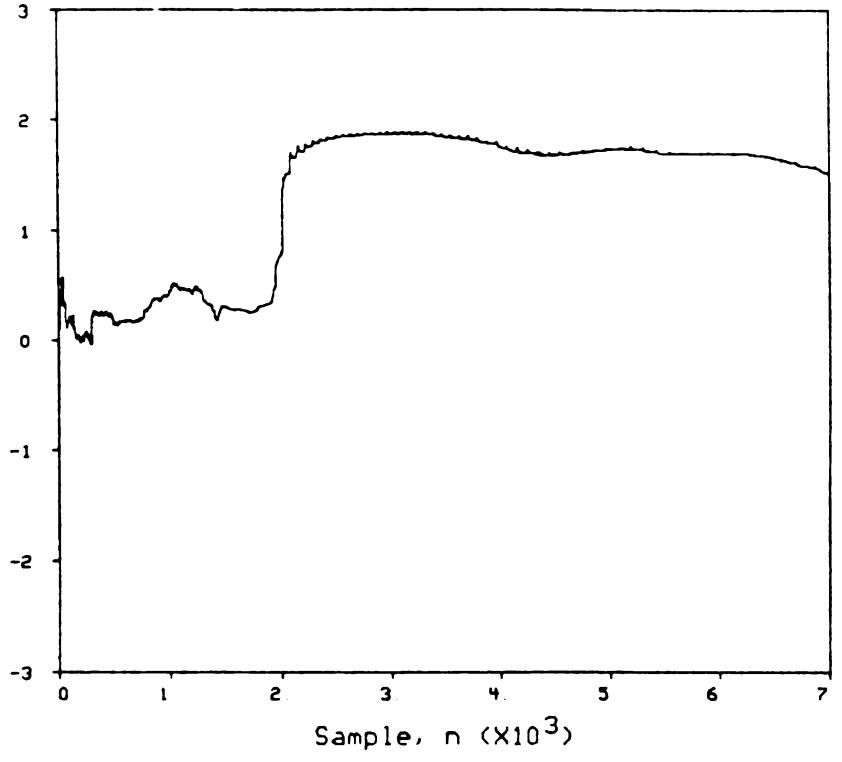


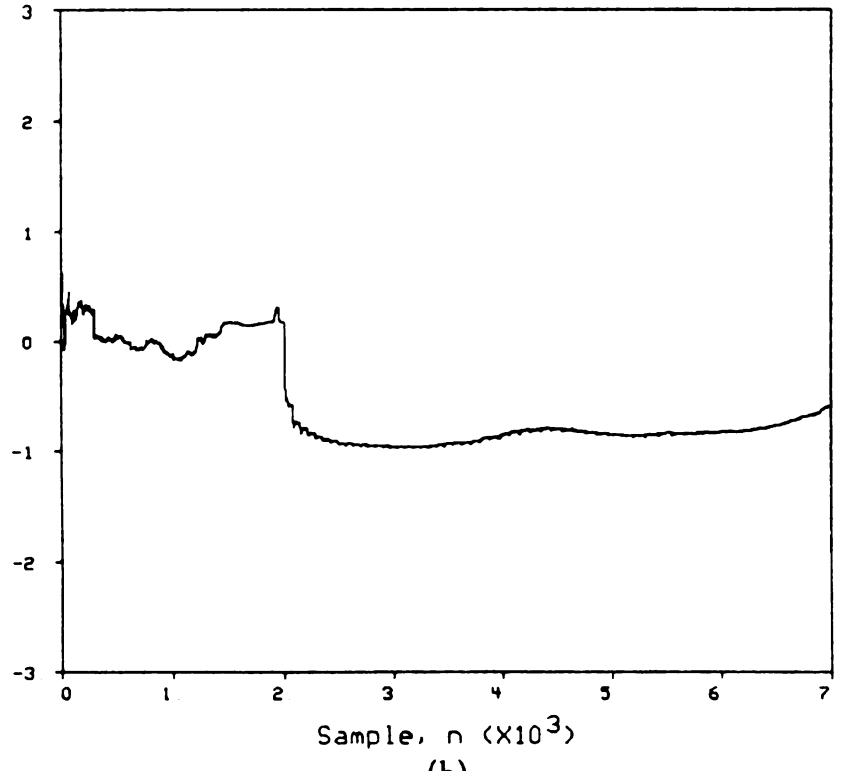
Figure 3.2: The acoustic waveform of the word "six".

the "forgetting factor" in the LP algorithm (see [44]) was $\beta^2 = 0.996$. A 7000 point sequence, $y(n)$, for each case ("four" and "six") was generated by driving the appropriate set of parameters with an uncorrelated sequence, $v(n)$, which was uniformly distributed on $[-1, 1]$. $v(n)$ in each case was generated using a random number generator based on a subtractive method [48]. In the simulations below, the conventional, adaptive, suboptimal, and adaptive suboptimal SM-WRLS algorithms are applied to the identification of the a_i parameters.

In the following, the simulation results are shown and discussed. In each figure, there are two different frames, one for each parameter. Each frame shows two curves, one for the true parameter, the other for the estimate obtained by the algorithm under study. Additionally, the true parameters for the words "four" and "six" are shown in Figs. 3.3 and 3.4, respectively. These are provided as a reference in cases



(a)



(b)

Figure 3.3: The "true" parameters for the word "four". (a) Parameter a_1 and (b) Parameter a_2.

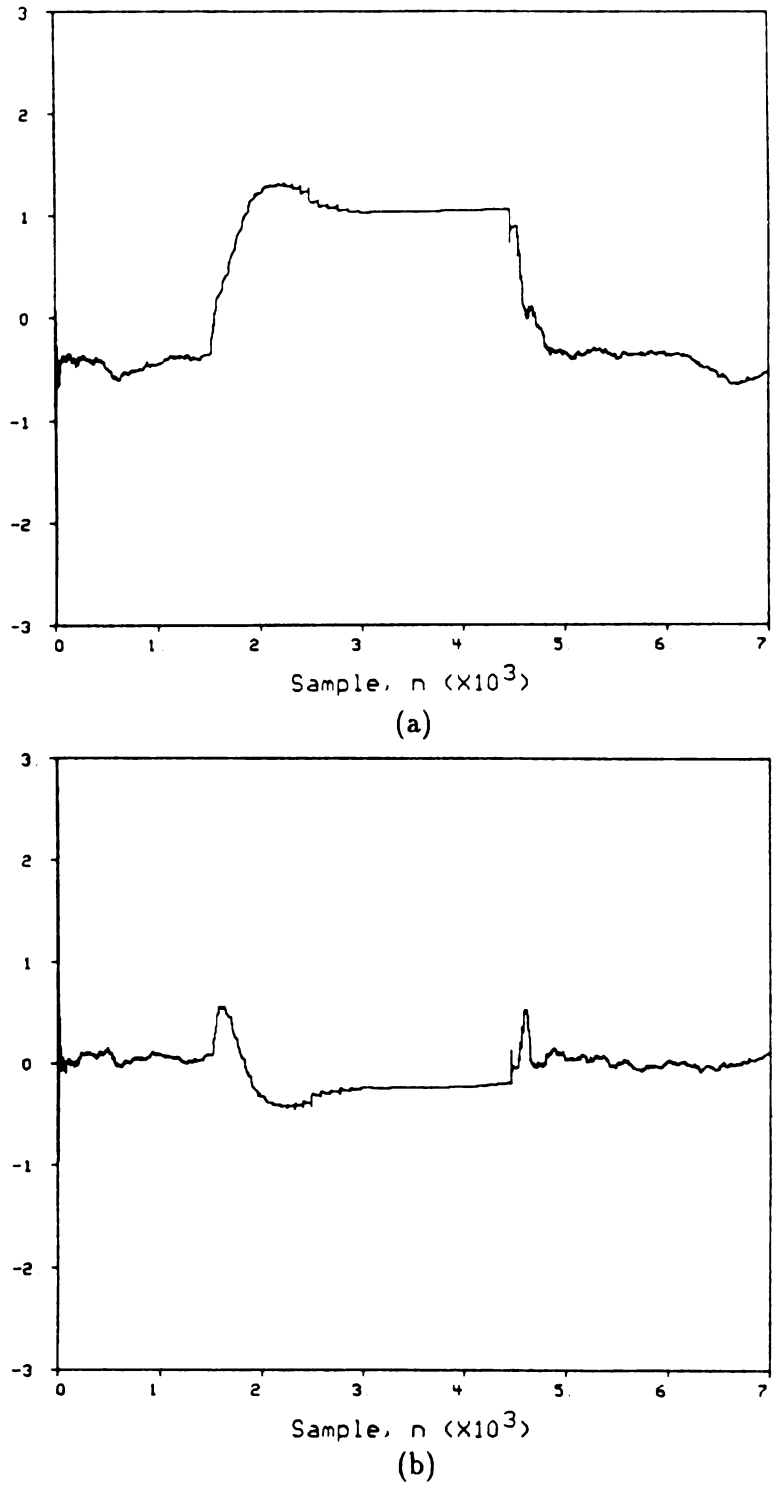


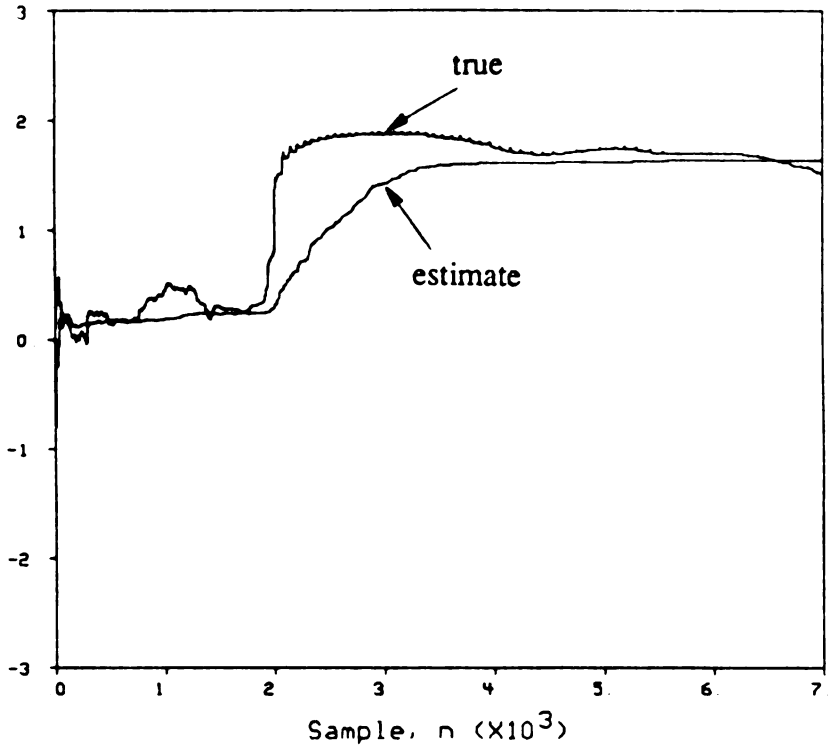
Figure 3.4: The "true" parameters for the word "six". (a) Parameter a_1 and (b) Parameter a_2.

where the true curves are difficult to discern.

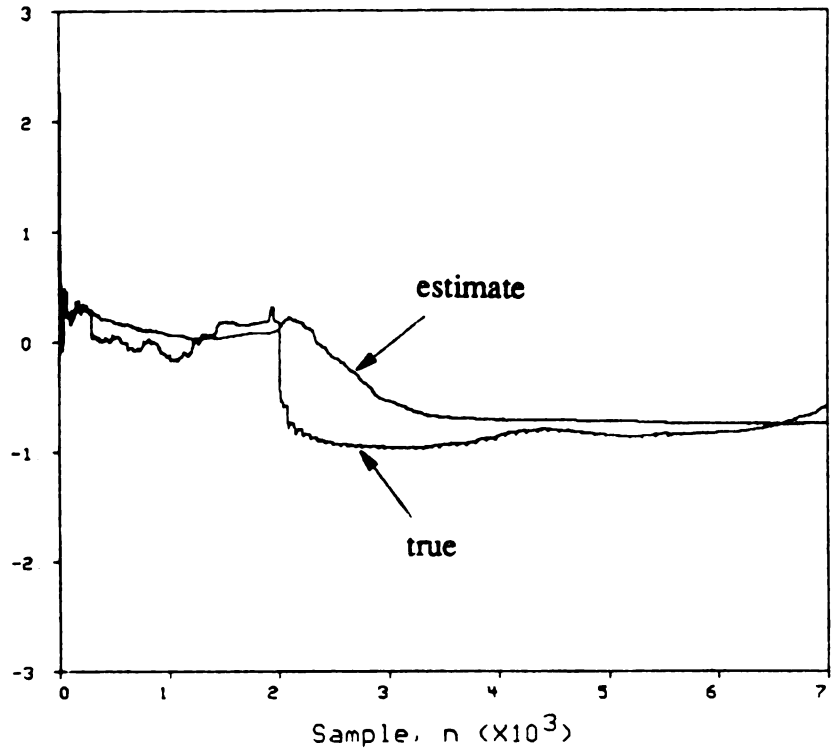
3.2.1 Conventional RLS and SM-WRLS Algorithms

The power of the SM-WRLS algorithm is evident when compared with the conventional RLS [35] algorithm. As a basis for further discussion, we first show this comparison. Figures 3.5 and 3.6 show the simulation results for the word four using the RLS and the SM-WRLS algorithms, respectively, and Figs. 3.7 and 3.8 show the simulation results for the word six. It is evident that SM-WRLS outperforms RLS in terms of its tracking capability, and it is critical to note that this improved performance comes with improved computational efficiency. In this case SM-WRLS uses only 1.86% and 2.16% of the data for the words four and six, respectively, and yet yields better parameters estimates almost all the time. It is important to note that SM-WRLS tracks the time varying parameters faster than RLS. This is manifest in both examples, especially the word six (see Figs. 3.7 and 3.8).

While the main theme of *Section 2.4* is the development of *adaptive* SM-WRLS methods, it is noted that the "unmodified" SM-WRLS algorithm apparently has adaptive capabilities in its own right. While SM-WRLS is developed under the assumption of stationary system dynamics, it is capable of behaving in this manner in certain circumstances because of the special weights used. Recall that these optimal data weights have the interpretation as parameters which minimize the ellipsoid volume. Note, however, that these weights multiply the corresponding equations, and therefore, different equations have different weights. The SM-WRLS algorithm determines the value of each weight depending on the "amount" of *new* information (from the SM point of view) contained in an equation. Hence, an equation with "no new information" is likely to be rejected ($\lambda(n) = 0$) whereas an equation with a significant amount of information (such as in the case of fast changing dynamics) is likely to be *heavily* weighted. This intuitively accounts for the inherent adaptation behavior of

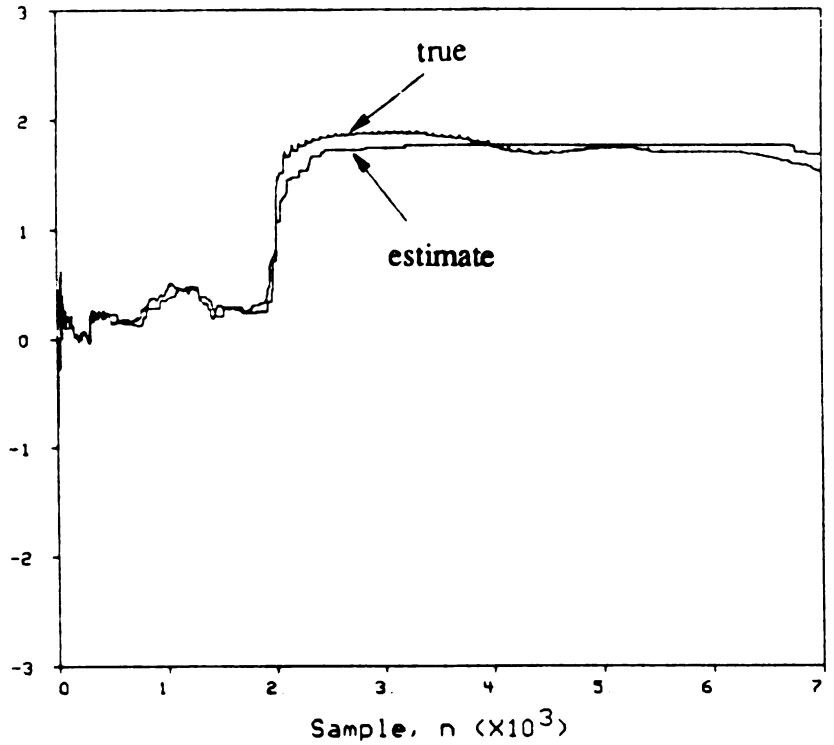


(a)

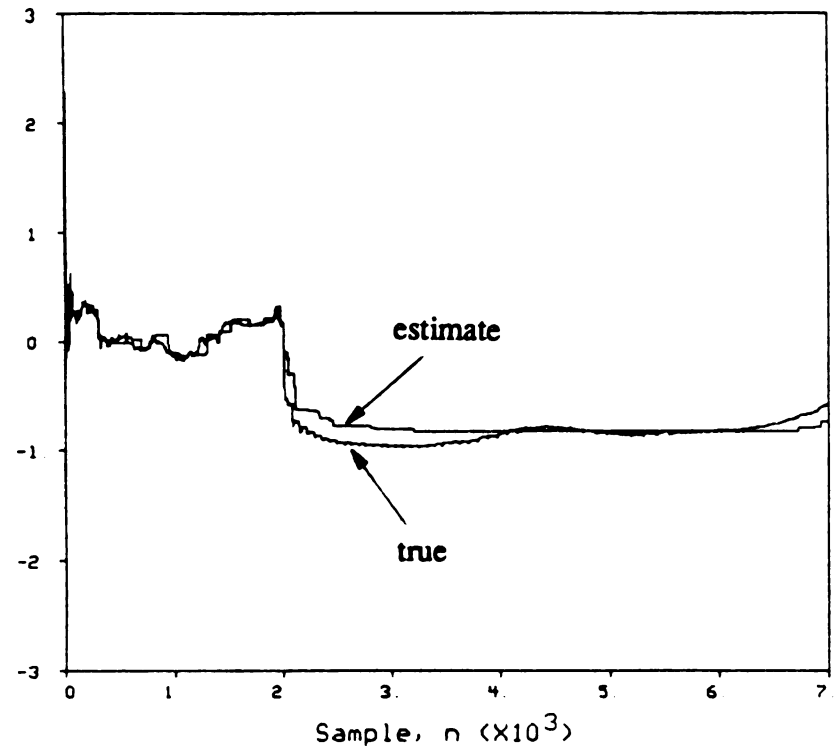


(b)

Figure 3.5: Simulation results of the conventional RLS algorithm for the word four.

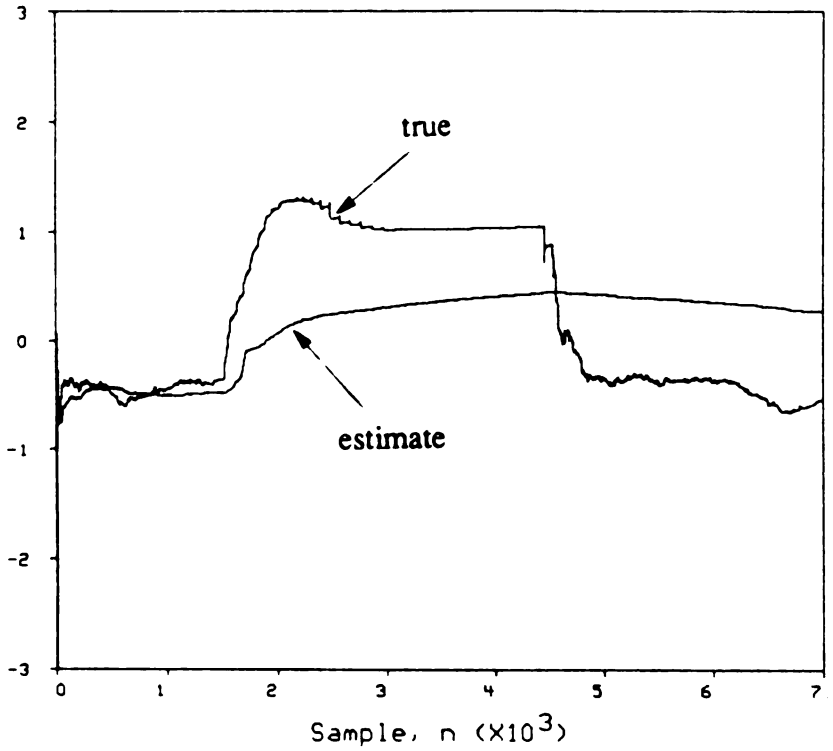


(a)

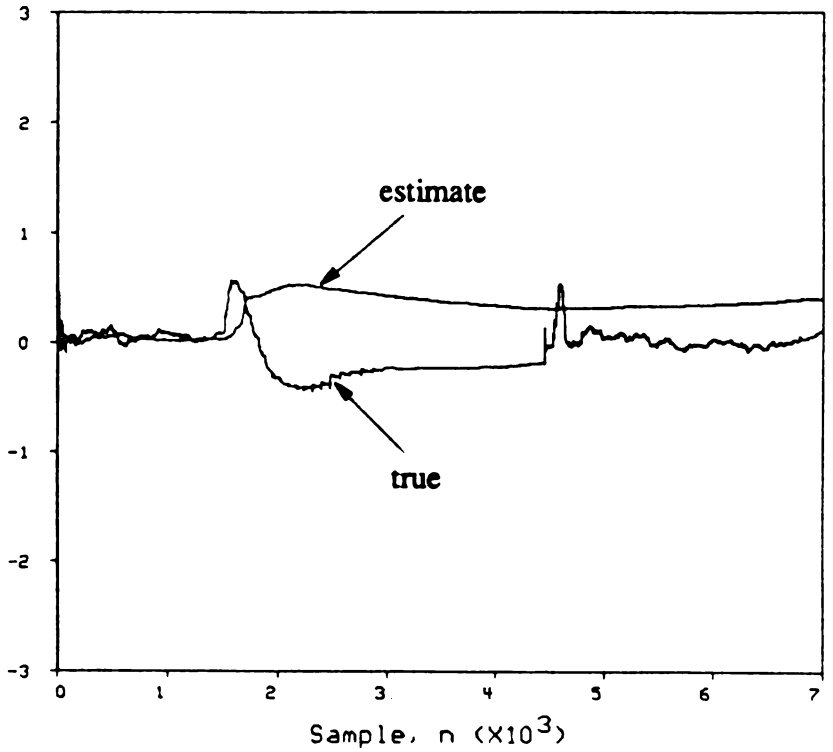


(b)

Figure 3.6: Simulation results of the SM-WRLS algorithm for the word four. 1.86% of the data is employed in the estimation process.

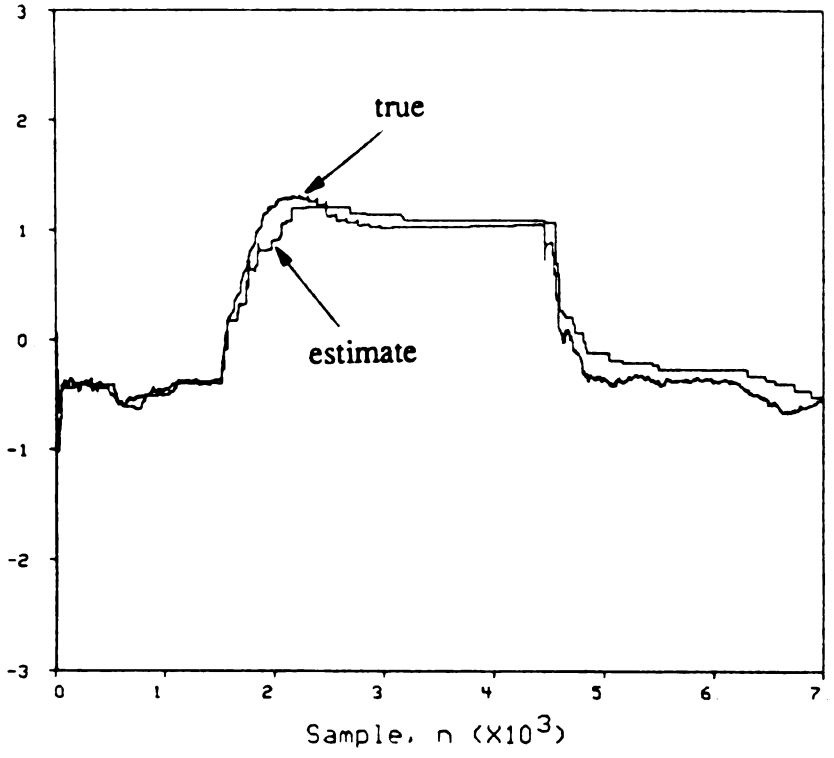


(a)

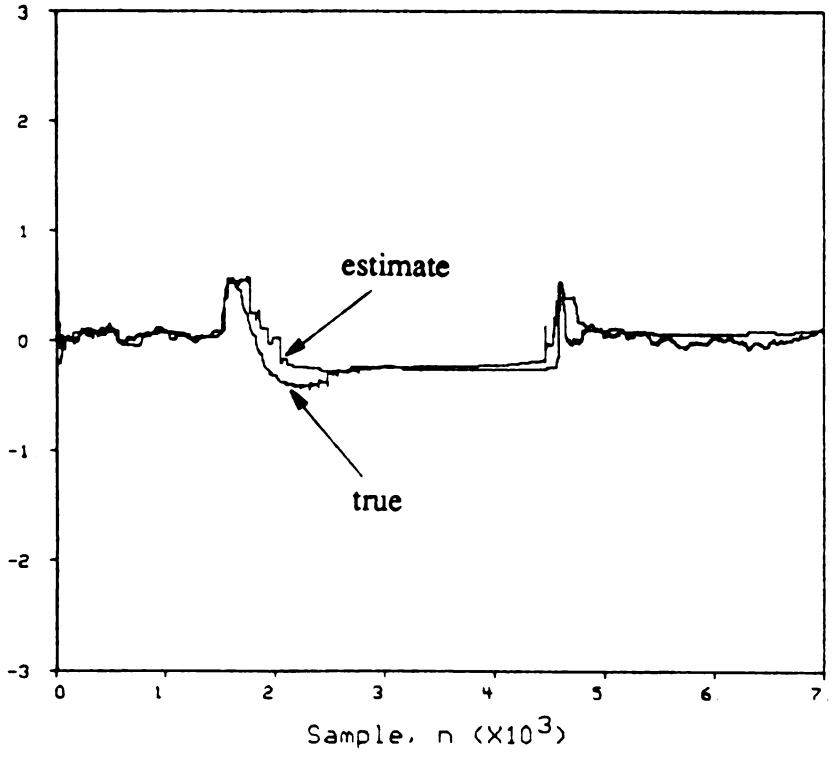


(b)

Figure 3.7: Simulation results of the conventional RLS algorithm for the word six.



(a)



(b)

Figure 3.8: Simulation results of the SM-WRLS algorithm for the word six. 2.16% of the data is employed in the estimation process.

the unmodified SM-WRLS algorithm.

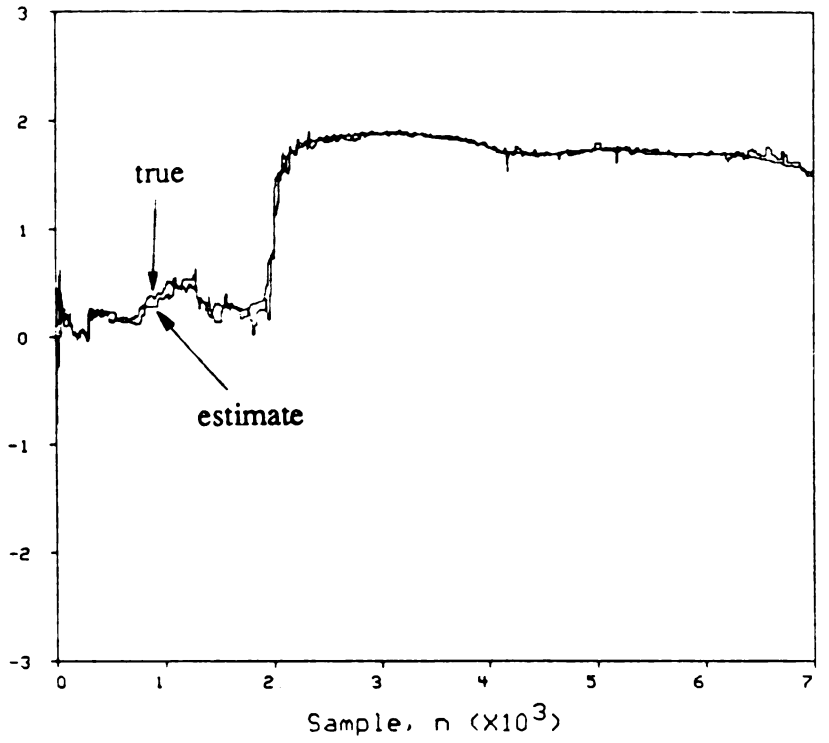
However, as will be seen below, it is not possible to depend upon SM-WRLS to reliably behave in this adaptive manner, particularly in cases of quickly varying system dynamics. Each time a new equation is accepted, the ellipsoid volume decreases and the "confidence" in the current estimate increases. In a situation in which the signal is varying rapidly and the parameters are moving away from their "current" locations, then the algorithm accepts incoming equations to incorporate the new information into the estimate, and the ellipsoid volume decreases rapidly, eventually becoming very small. As the parameters continue to move rapidly away from their current locations, they eventually move outside the shrinking ellipsoid which becomes an invalid bounding ellipsoid. This condition indicates that a violation of the theory has taken place, and therefore, the unmodified SM-WRLS algorithm is no longer guaranteed to work properly. However, it is noted (empirically) that the unmodified SM-WRLS algorithm performs well when applied to slow time varying systems.

Next, the simulation results of the several variations on the general adaptive SM-WRLS algorithm are shown. It is worth reiterating that when it becomes difficult to distinguish the true parameters from their estimates, the reader can refer to Figs. 3.3 and 3.4 for clarification. Also note that the figure captions will not contain the description for the two frames for it is implied that frame (a) shows parameter a_1 and frame (b) shows parameter a_2.

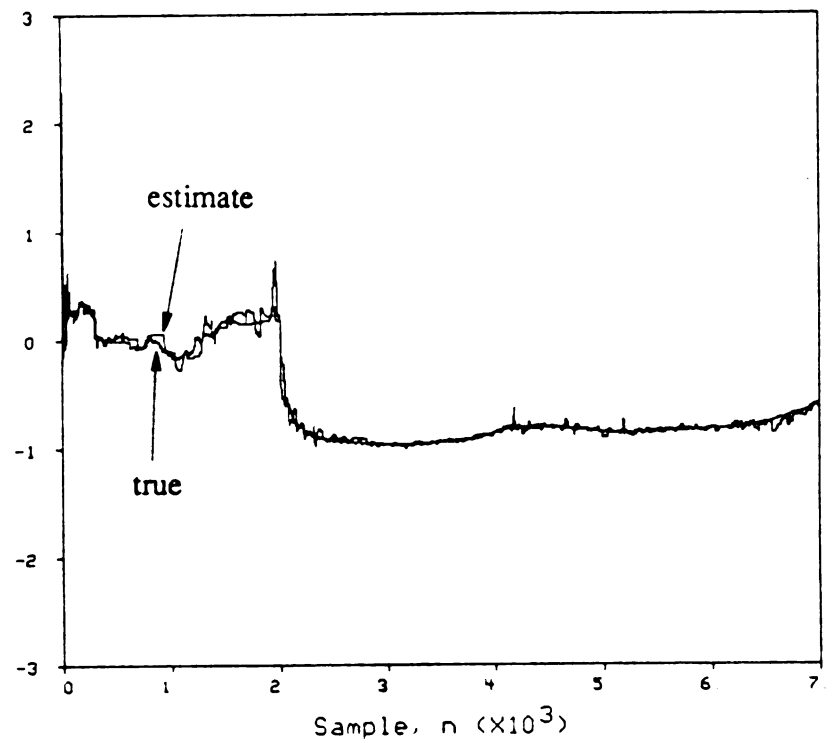
3.2.2 Adaptive SM-WRLS Algorithms

3.2.2.1 Windowing

Figures 3.9 and 3.10 show the simulation results of the windowed SM-WRLS algorithm for the words four and six, respectively, using a window of length 1000. This strategy uses only 5.69% and 5.44% of the data for the words four and six, respectively. Since each equation rotated in is eventually rotated out of the system, this strategy

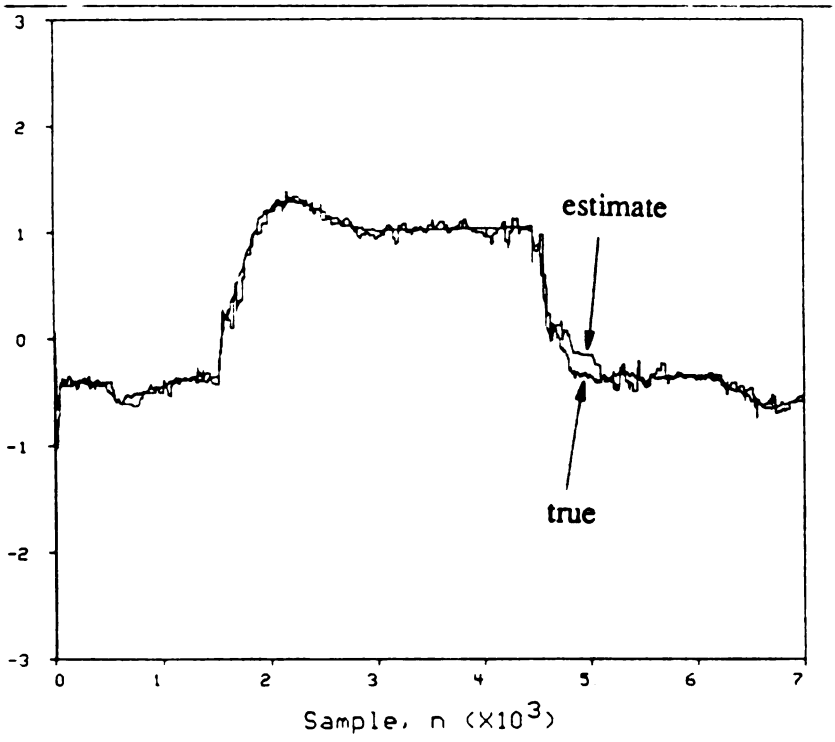


(a)

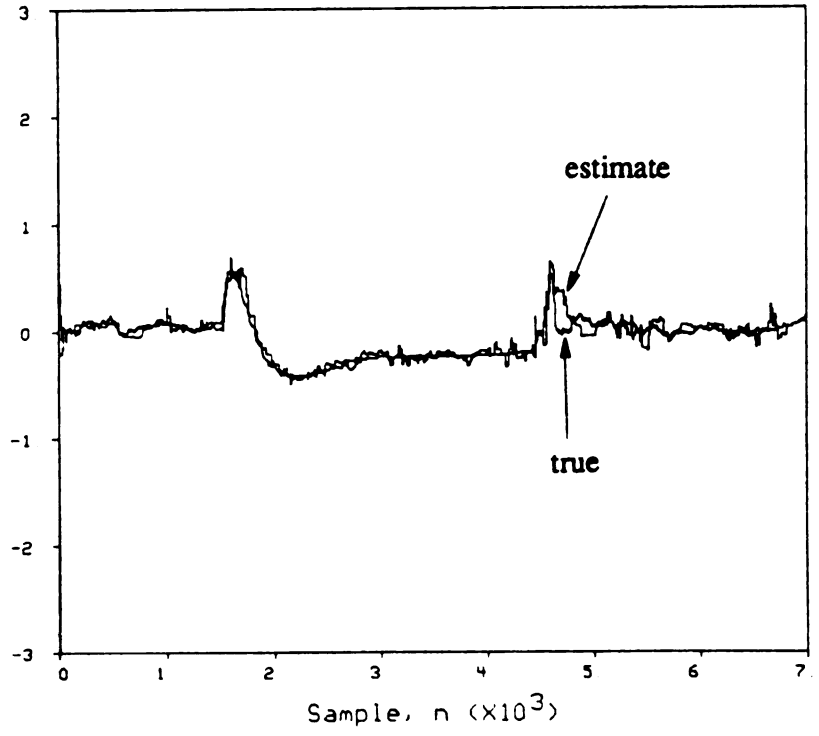


(b)

Figure 3.9: Simulation results of the windowed SM-WRLS algorithm for the word four ($l = 1000$). 5.69% of the data is employed in the estimation process.



(a)



(b)

Figure 3.10: Simulation results of the windowed SM-WRLS algorithm for the word six ($l = 1000$). 5.44% of the data is employed in the estimation process.

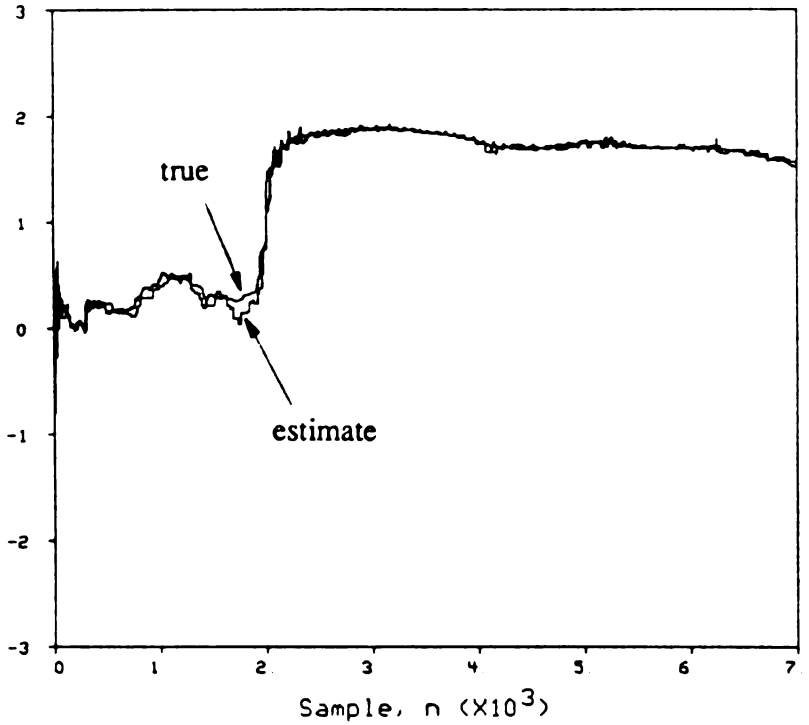
effectively uses about twice the number of equations rotated in. More data than with the unmodified SM-WRLS algorithm are used, but more accurate estimates result and the time varying parameters are tracked more quickly and accurately. This can easily be seen when the parameter dynamics change abruptly near the point 2000 for the word four (see Fig. 3.9) and near the points 2000 and 4500 for the word six (see Fig. 3.10).

3.2.2.2 Graceful Forgetting

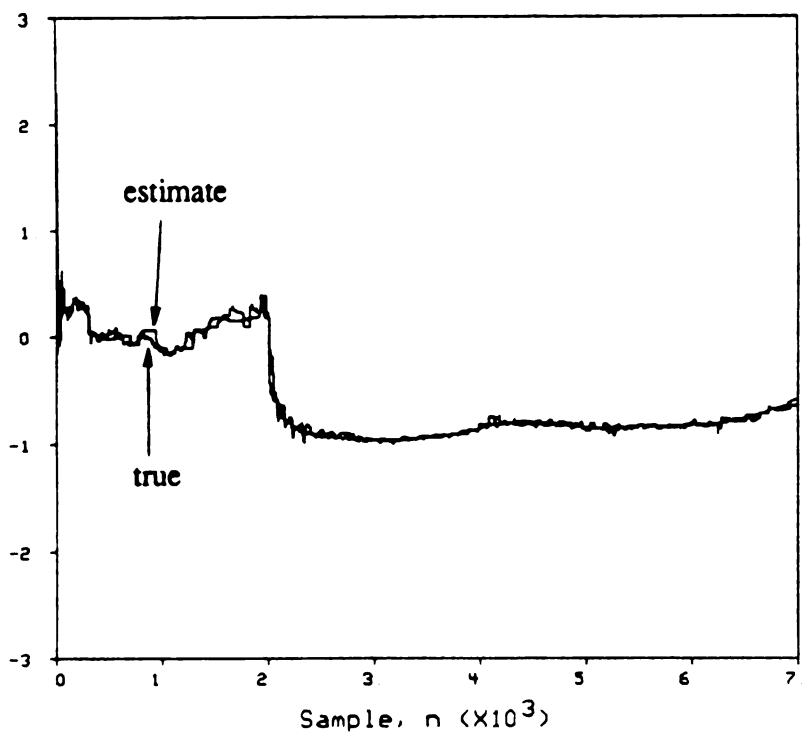
Figures 3.11 and 3.12 show the simulation results of the graceful forgetting SM-WRLS algorithm when rotating out 0.1% of each of the equations ($\mu = 10^{-3}$) that was accepted in the past 1000 recursions. This strategy uses only 6.19% and 4.89% of the data for the words four and six, respectively. Note that this technique uses comparable percentages of the data to those used by the windowed strategy and yields smoother estimates. Although the algorithm uses very small percentages of the data, the value of μ used here might not be practical because it means that the algorithm will rotate out each equation *that was initially accepted* 1000 times, which is clearly a computational burden. Therefore, this strategy effectively uses about 1000 (or μ^{-1}) times the number of equations rotated in. Depending on the nature of the problem, practical values of μ may range from 0.002 to 0.01 with an effective window of length 500 to 100.

3.2.2.3 Selective Forgetting

As noted in *Subsection 2.4.1.3*, the selective forgetting strategy selects the equations to be (partially or completely) removed from the system based on user defined criteria. The selection procedure used here is to remove the equations starting from the first accepted equation remaining in the estimate at a given time, and proceeding sequentially until some other condition is satisfied. The determination of when to apply the forgetting procedure and when to stop removing equations at a given time

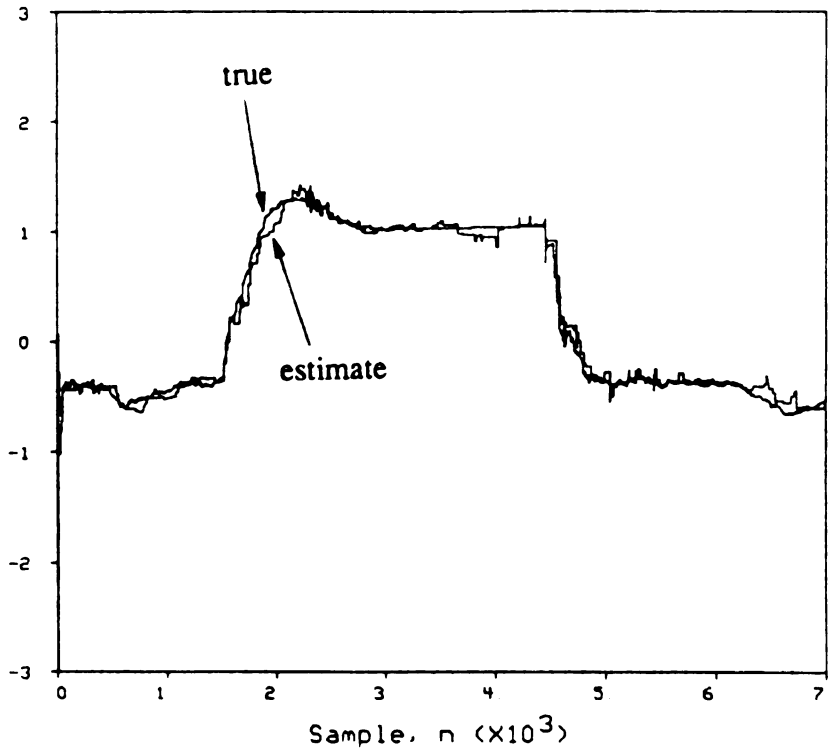


(a)

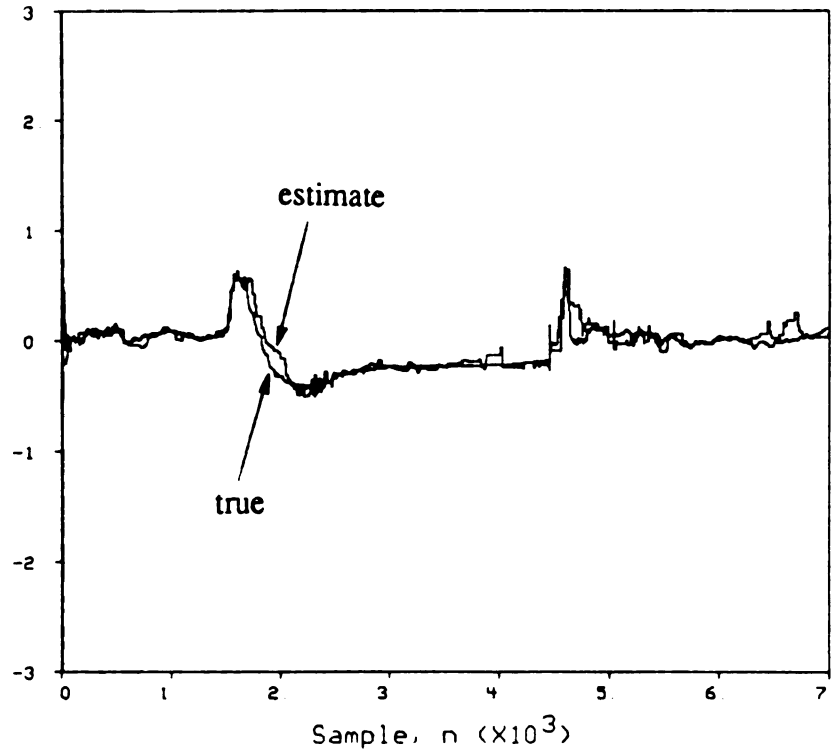


(b)

Figure 3.11: Simulation results of the graceful forgetting SM-WRLS algorithm for the word four ($\mu = 10^{-3}$). 6.19% of the data is employed in the estimation process.



(a)



(b)

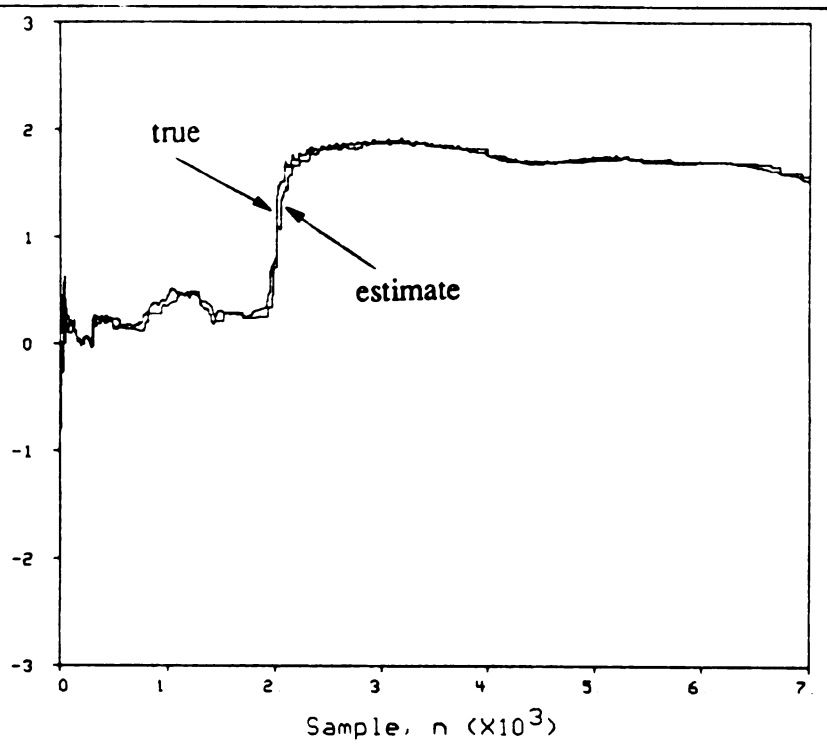
Figure 3.12: Simulation results of the graceful forgetting SM-WRLS algorithm for the word six ($\mu = 10^{-3}$). 4.89% of the data is employed in the estimation process.

is discussed in the following.

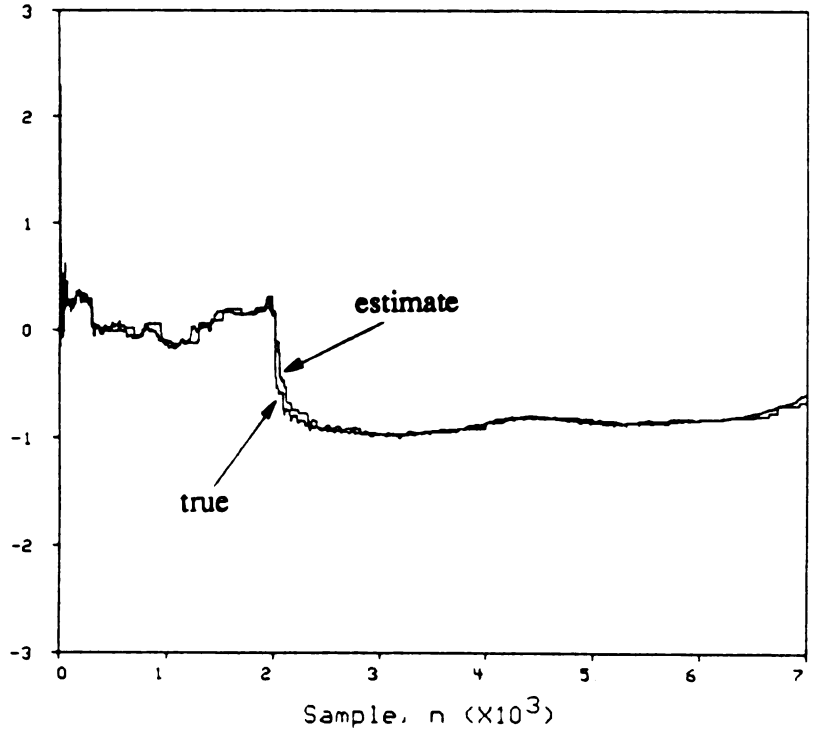
When the true parameters of the word four are inspected (see Fig. 3.3), for example, it is noted that they can be characterized as having *slow* time variations everywhere except in the region from 2000 to 2300 where they have *fast* time variations. The fact that the parameters are changing very slowly in the first 2000 points, induces the algorithm to accept some points which, in turn, causes the ellipsoid volume to decrease. An increase in the "confidence" of the estimate results. Near time 2000, the ellipsoid volume becomes very small. When the parameters move rapidly away from their current location, they eventually move outside the ellipsoid which is therefore no longer a valid bounding ellipsoid. An "animation" subroutine has been developed which allows the user to see the locations of the true parameters and their estimates with respect to the ellipsoid in the 2-D parameter space. When this condition happens, it eventually leads to a negative value of $\kappa(n)$, (see (1.23)). For a stationary system, $\kappa(n)$ is always positive [7], so that this condition indicates that a violation of the theory (in particular, the violation of the assumption of stationary dynamics) has taken place[6]. A similar condition was also reported by Dasgupta and Huang [14] while applying their OBE algorithm to nonstationary systems. In our simulation studies, it is found that a negative $\kappa(n)$ is an effective indicator of need for adaptation, and this criterion is used as the prompt to begin selective forgetting. Whenever accepting an equation causes $\kappa(n)$ to become negative, the algorithm starts rotating out the equations which are selected based on the selection procedure until $\kappa(n)$ becomes positive again.

Figures 3.13 and 3.14 show the simulation results of the selective forgetting strategy described here. For the words four and six, respectively, this technique uses only 3.6% and 2.83% of the data, 18.7% and 56.1% of which are rotated out during the adaptation process, and therefore, when counting the total number of equations ro-

[6]Mathematically, $\kappa(n) < 0$ indicates an ellipsoid of negative dimensions.

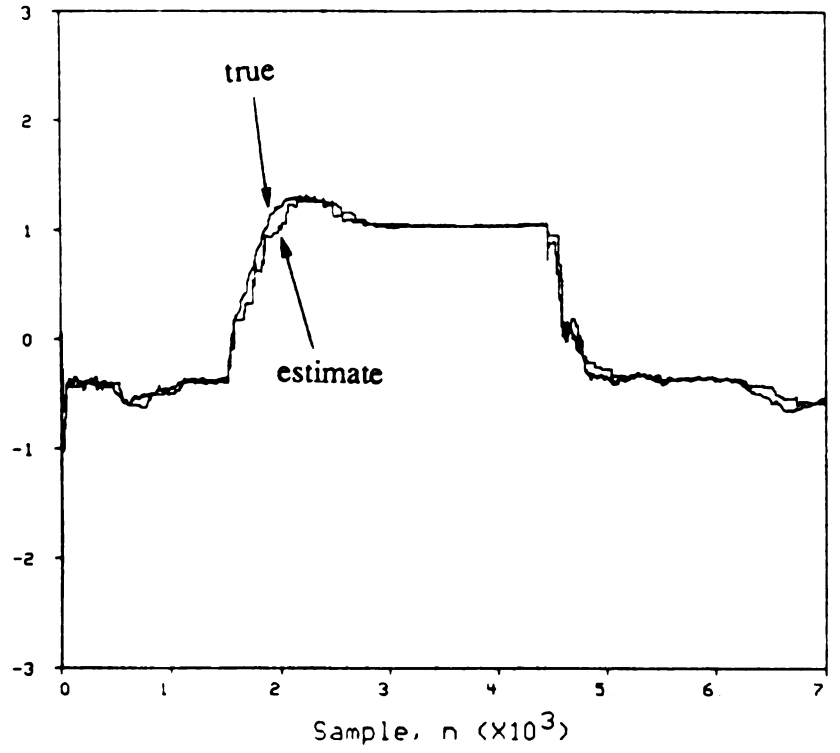


(a)

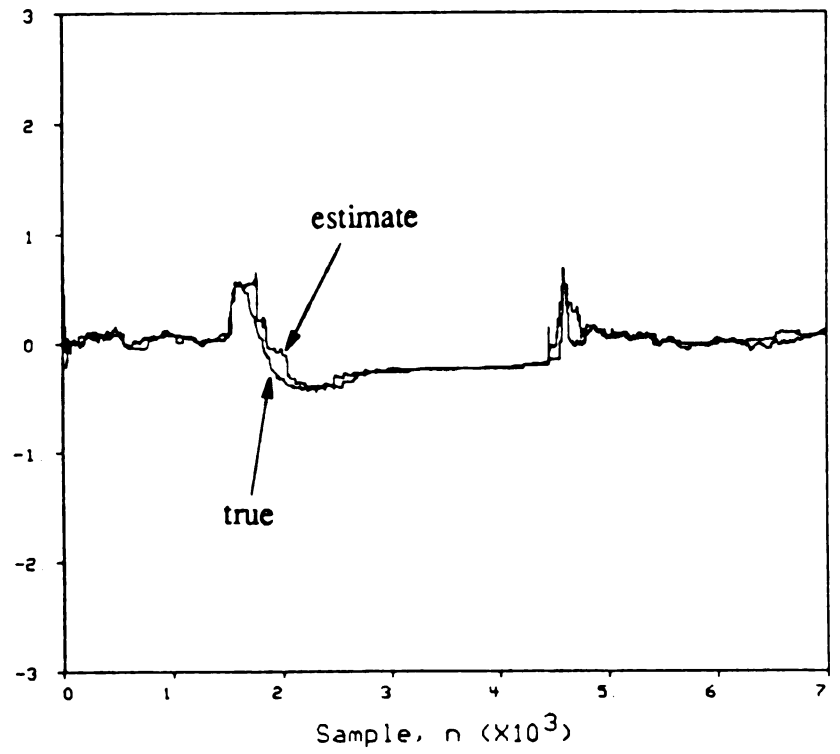


(b)

Figure 3.13: Simulation results of the selective forgetting SM-WRLS algorithm for the word four. 3.6% of the data is employed in the estimation process.



(a)



(b)

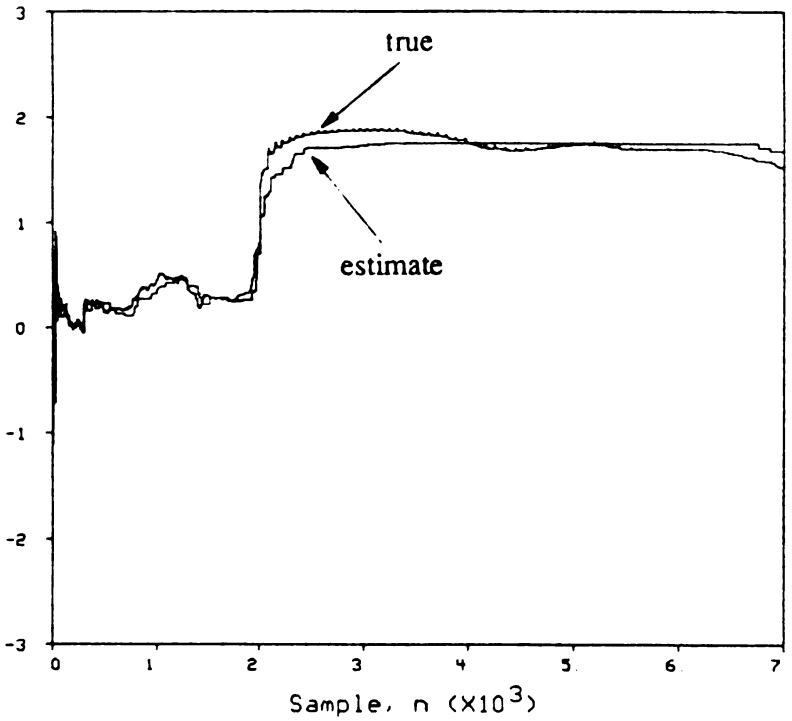
Figure 3.14: Simulation results of the selective forgetting SM-WRLS algorithm for the word six. 2.83% of the data is employed in the estimation process.

tated into and out of the system, this strategy effectively uses 4.27% and 4.41% of the data. Compared to the windowed and graceful forgetting adaptive strategies, the simulation results show that the selective forgetting strategy yields smoother estimates using even fewer data. It is important to recall that the percentages given in the windowed and graceful forgetting adaptive strategies represent the total number of equations used by these techniques, and do not account for the number of equations rotated out of the system. Detailed complexity comparisons are made in *Section 2.5.*

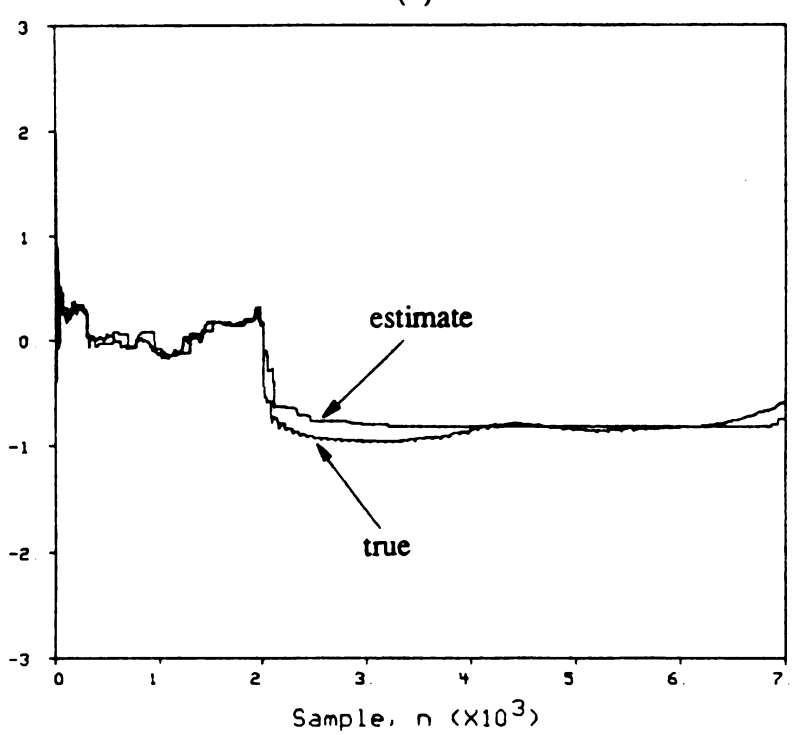
It should be pointed out that $\kappa(n) > 0$ is only a necessary condition for the true parameters to be inside the current ellipsoid. The fact that κ goes negative at a particular time does not precisely determine the point at which system dynamics began to change. In fact, $\kappa(n) < 0$ indicates a rather severe breakdown of the process indicating that the "true" parameters have moved well outside of the current ellipsoid. However, it is precisely in cases of *fast* changing dynamics that this "breakdown" occurs rapidly resulting in "$\kappa(n) < 0$" in fact being a good locator of changing dynamics which require "immediate" adaptation to preserve the integrity of the process. In cases of slowly changing dynamics where the theory can be violated without the appearance of negative κ, SM-WRLS seems to be sufficiently robust to these slow changes to make its own adjustments. The examples of Figs. 3.6 and 3.8 above illustrate this later point.

3.2.3 Suboptimal SM-WRLS

Figures 3.15 and 3.16 show the simulation results of the unmodified SM-WRLS algorithm with suboptimal data selection. In this case, only 1.19% and 1.53% of the data are used for the words four and six, respectively. Compared to the SM-WRLS algorithm (see Figs. 3.6 and 3.8), the suboptimal technique uses slightly fewer data but produces comparable estimates. It is interesting to note that *most* of the equations (97.6% for the word four and 94.4% for the word six) that are accepted by

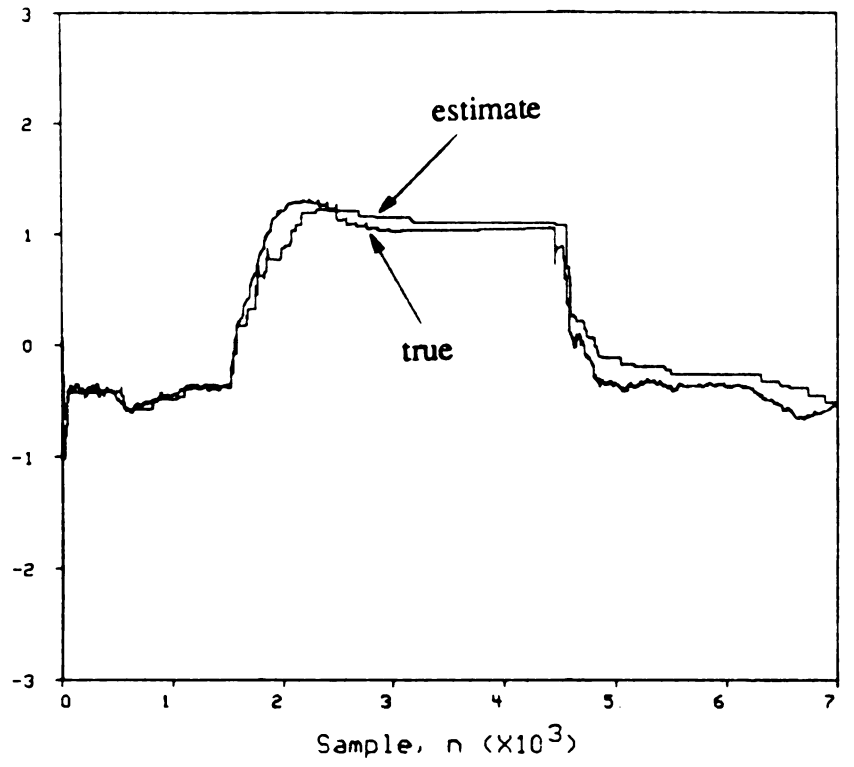


(a)

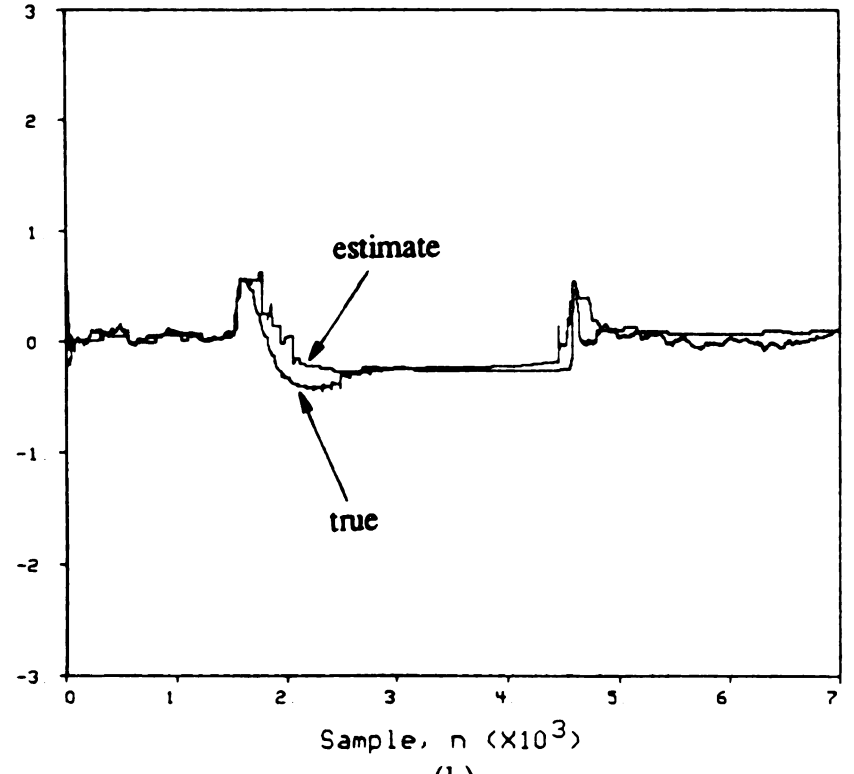


(b)

Figure 3.15: Simulation results of the SM-WRLS algorithm with suboptimal data selection for the word four. 1.19% of the data is employed in the estimation process.



(a)



(b)

Figure 3.16: Simulation results of the SM-WRLS algorithm with suboptimal data selection for the word six. 1.53% of the data is employed in the estimation process.

the suboptimal technique are also accepted by the SM-WRLS algorithm. It is also interesting to note that the equations that are accepted by the suboptimal technique but not by the SM-WRLS algorithm lie mostly in regions of fast changing dynamics.

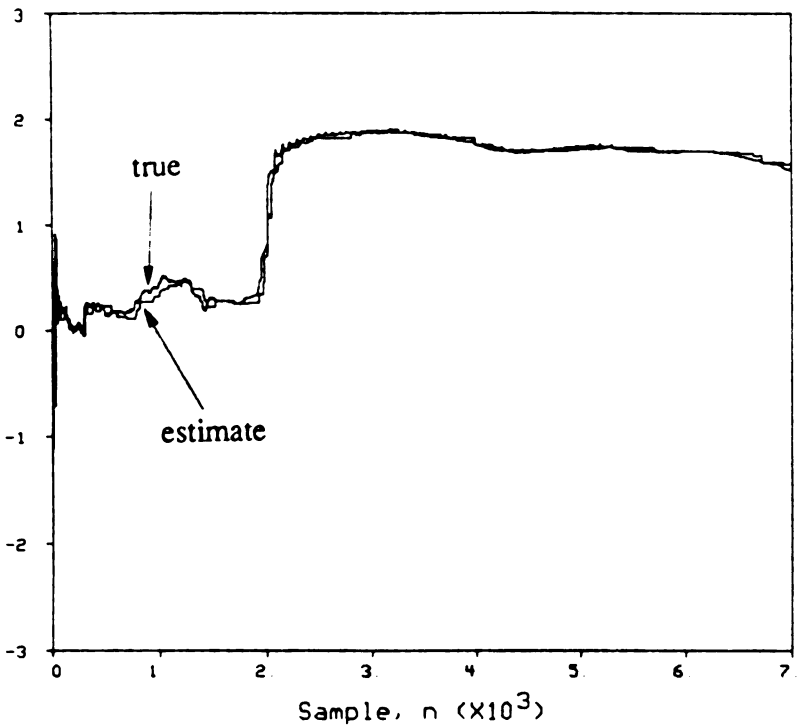
3.2.4 Adaptive Suboptimal SM-WRLS

It is noted in *Section 2.4.1* that the general formulation of the adaptive SM-WRLS algorithm is amenable to the suboptimal technique. The simulation results of the selective forgetting SM-WRLS technique with suboptimal data selection are shown in Figs. 3.17 and 3.18. This strategy uses only 1.89% and 1.86% of the data, 14.4% and 48.5% of which are rotated out during the adaptation process, and therefore, when counting the total number of equations rotated into and out of the system, this strategy effectively uses 2.16% and 2.76% of the data.

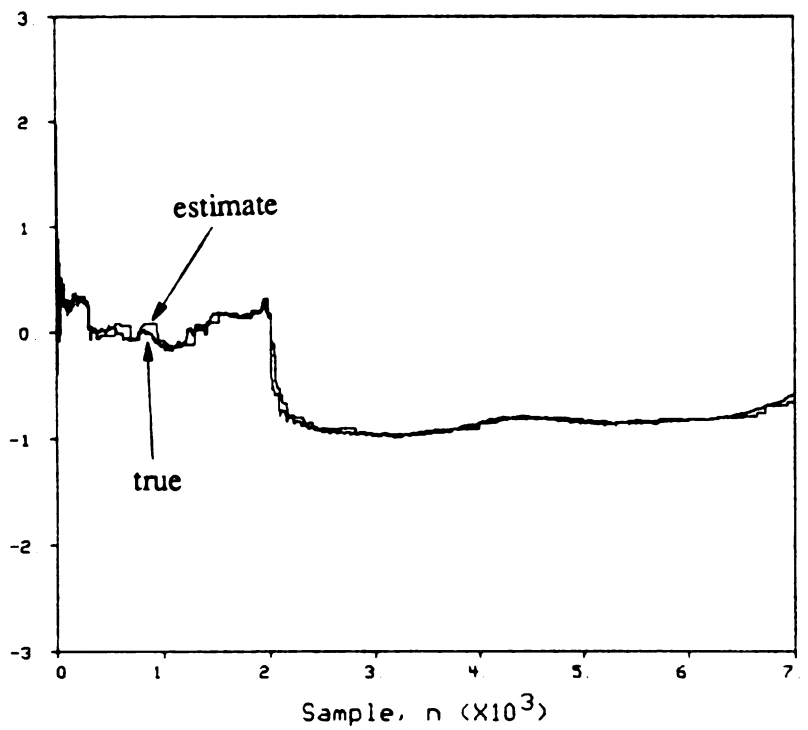
Compared to the selective forgetting strategy (Figs. 3.13 and 3.14), the selective forgetting technique with suboptimal data selection uses fewer data but produces comparable estimates. On the other hand, when compared to unmodified SM-WRLS with suboptimal data selection (Figs. 3.15 and 3.16), the selective forgetting suboptimal technique uses more data but produces better estimates.

3.3 Simulation Results of an AR(14) model

In this section, the identification of a time varying *AR(14)* model is considered. The same procedure of *Section 3.2* is used to generate the "true" AR parameters from an utterance of the word "seven" by an adult male speaker. The acoustic waveform of this word is shown in Fig. 3.19. In the following, the simulation results are shown and discussed using the same format as that used in *Section 3.2*, however, only one true parameter (a_4), which is the most challenging parameter for the algorithm to track, is used for illustration and is shown in Fig. 3.20.

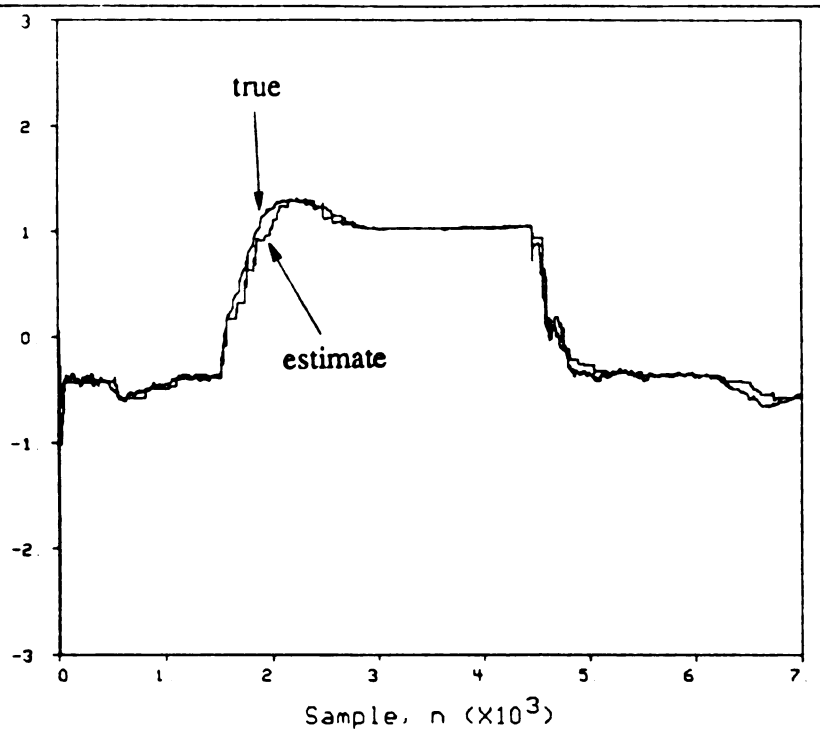


(a)

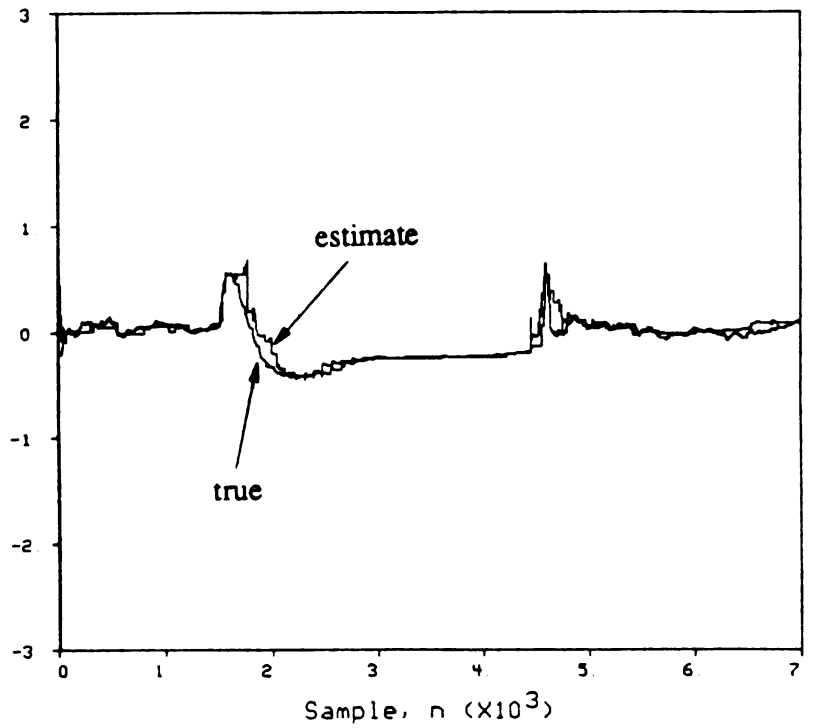


(b)

Figure 3.17: Simulation results of the selective forgetting SM-WRLS algorithm with suboptimal data selection for the word four. 1.89% of the data is employed in the estimation process.



(a)



(b)

Figure 3.18: Simulation results of the selective forgetting SM-WRLS algorithm with suboptimal data selection for the word six. 1.86% of the data is employed in the estimation process.

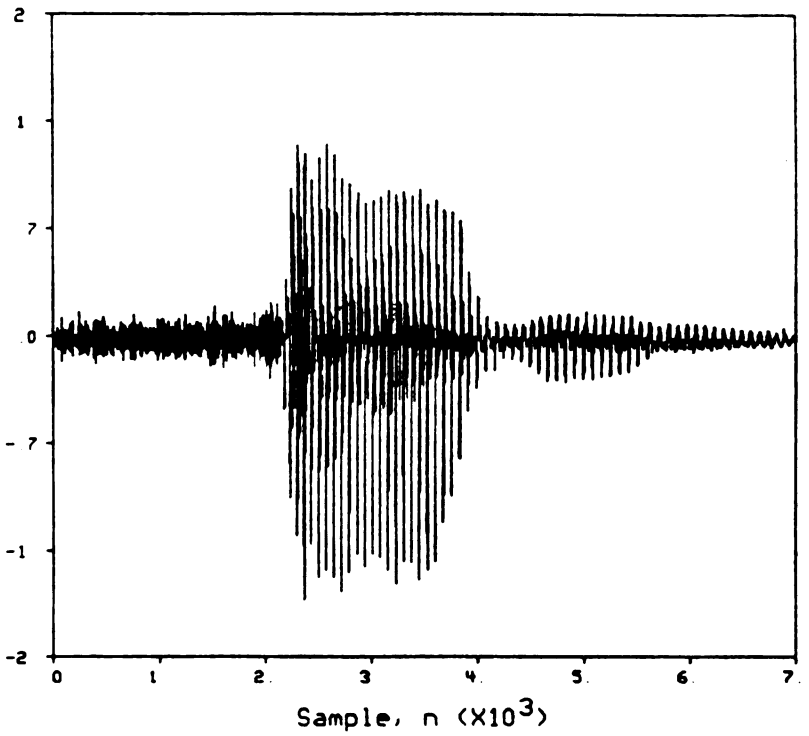


Figure 3.19: The acoustic waveform of the word "seven".

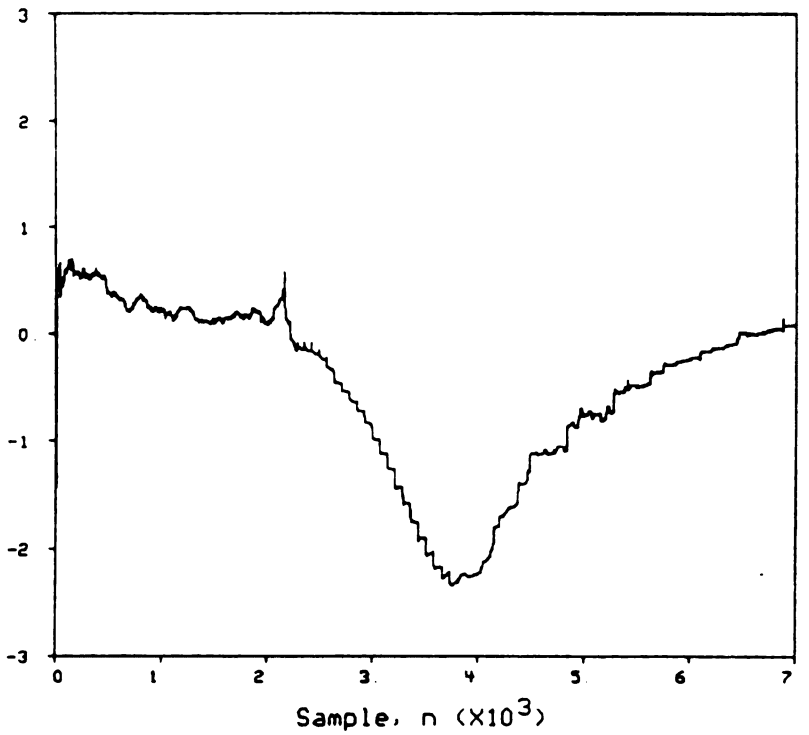


Figure 3.20: The fourth "true" parameter (a_4) for the word "seven".

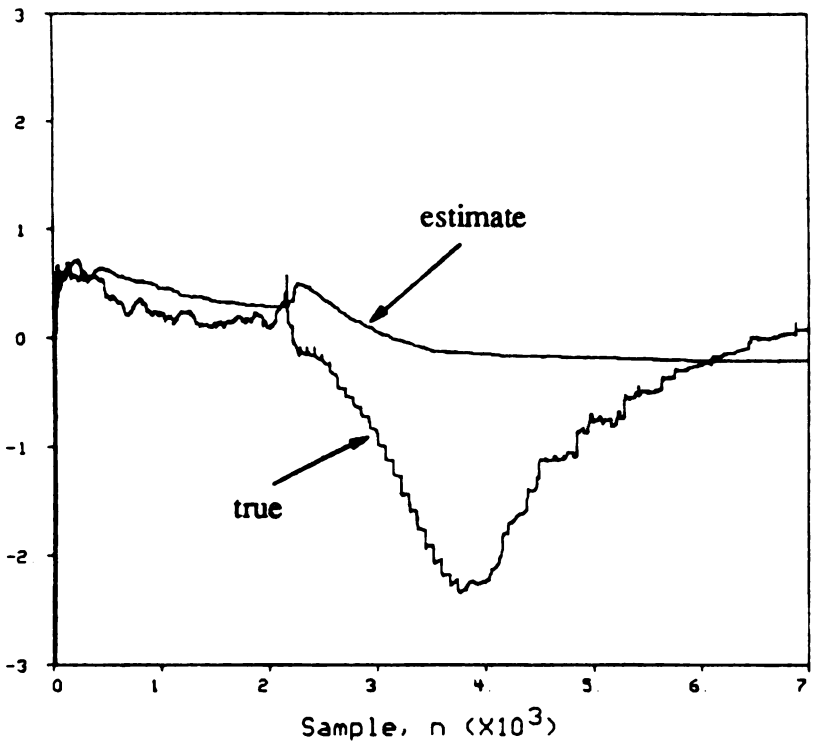


Figure 3.21: Simulation results of the conventional RLS algorithm for the word seven.

3.3.1 Conventional RLS and SM-WRLS Algorithms

Figures 3.21 and 3.22 show the simulation results for the word seven using the RLS and the SM-WRLS algorithms, respectively. Although both algorithms yield bad estimates and do not track the time varying parameters, the SM-WRLS algorithm which uses only 7.93% of the data yields better parameter estimates than those of the RLS algorithm most of the time.

In contrast with the AR(2) results of Figs. 3.6 and 3.8, the "unmodified" SM-WRLS algorithm for the AR(14) example shown in Fig. 3.22 fails to track the time varying parameters. This is expected since the signal is varying rapidly (especially in the range [2000–6000]) and the algorithm is no longer guaranteed to work properly (recall the discussion of *Section 3.2.1*). By inspecting Fig. 3.22 carefully, it is clear that the SM-WRLS algorithm loses its ability to track the signal in the range when

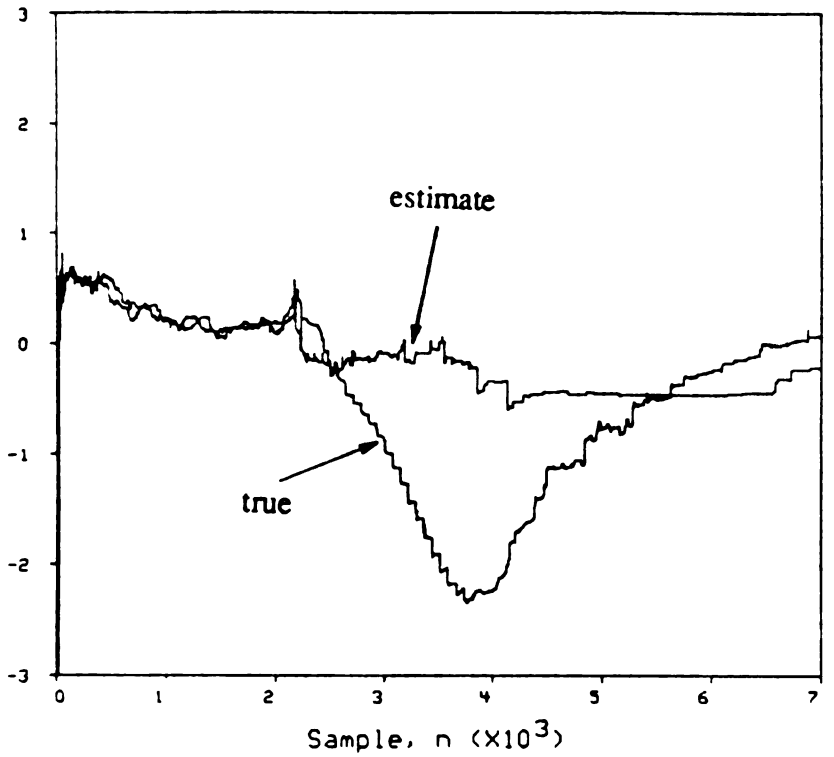


Figure 3.22: Simulation results of the SM-WRLS algorithm for the word seven. 7.93% of the data is employed in the estimation process.

the signal is varying rapidly.

3.3.2 Adaptive SM-WRLS Algorithms

3.3.2.1 Windowing

This subsection tests and compares the simulation results of the windowed SM-WRLS algorithm for the word seven using three different window lengths. Figures 3.23, 3.24, and 3.25 show the simulation results using windows of lengths 500, 1000, and 1500, and using 22.1%, 17.04%, and 14.34% of the data, respectively. In all three tests, more data than with the unmodified SM-WRLS algorithm are used, but much more accurate estimates result and the time varying parameters (one of which is shown) are tracked more quickly and accurately. When comparing the effect of the window length

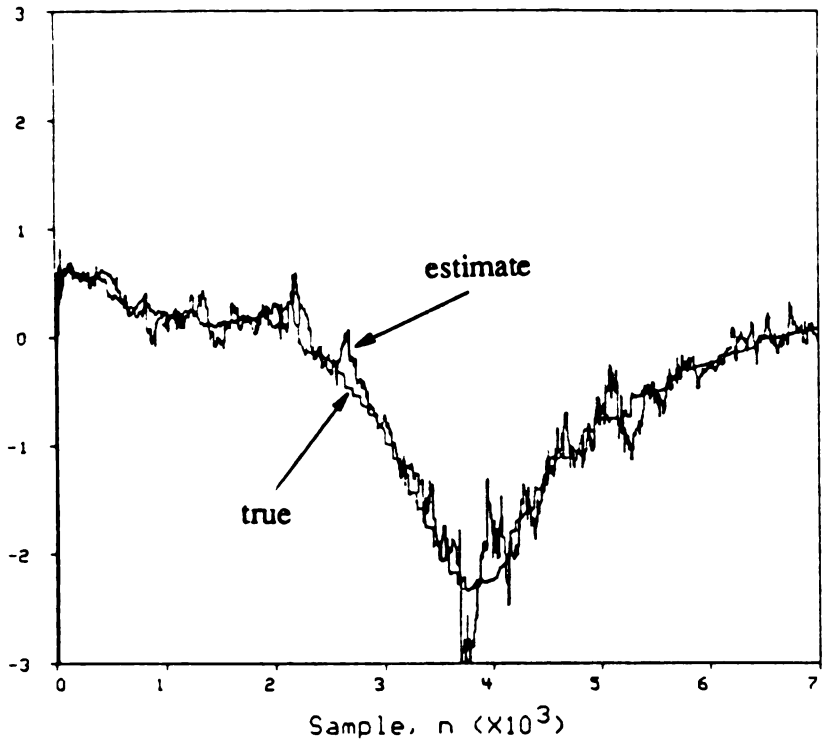


Figure 3.23: Simulation results of the windowed SM-WRLS algorithm for the word seven ($l = 500$). 22.1% of the data is employed in the estimation process.

on the algorithm, it is noted that shorter window lengths use *more* data, but yield more accurate estimates (most of the time) and track the time varying parameters more quickly and accurately. However, if the window length becomes very short (< 300), the variance of the estimates becomes very large because these estimates involve a small window length which is effectively even smaller because of the small fraction of data accepted.

3.3.2.2 Graceful Forgetting

This subsection tests and compares the simulation results of the graceful forgetting SM-WRLS algorithm for the word seven using three different μ values. Recall that μ represents the fraction of the equations removed from the system at each time. Figures 3.26, 3.27, and 3.28 show the simulation results using μ values of 10^{-3}, 2×10^{-3}, and 4×10^{-3} (or effective windows of lengths 1000, 500, and 250), and using 18.66%,

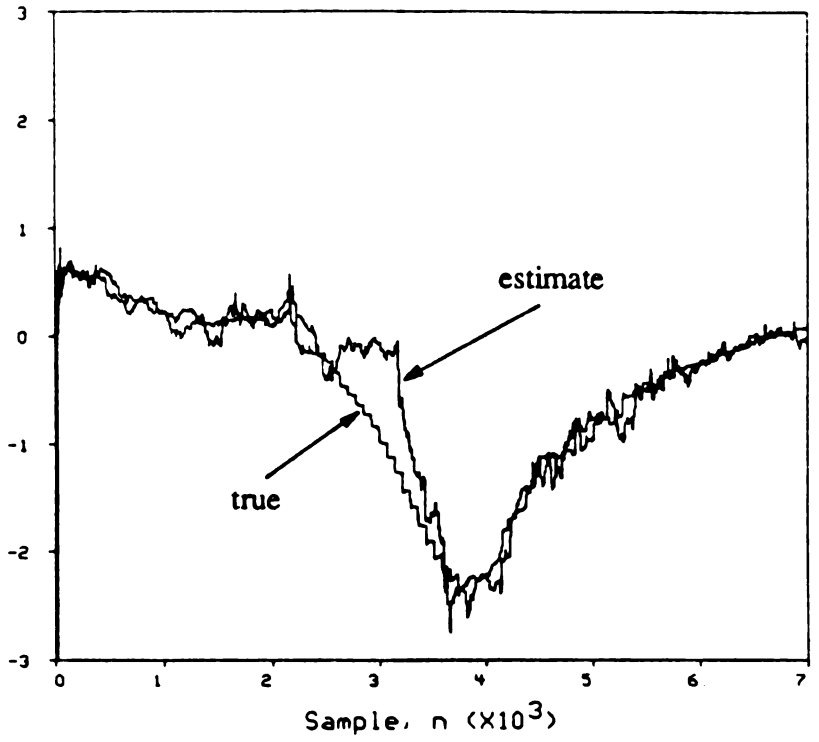


Figure 3.24: Simulation results of the windowed SM-WRLS algorithm for the word seven ($l = 1000$). 17.04% of the data is employed in the estimation process.

27.63%, and 35.59% of the data, respectively. Note that this strategy is initiated at time 100 to allow for the estimation to stabilize.

It is clear that as the value of μ increases (or the effective window length decreases), the algorithm uses more data (as expected), tracks the time varying parameters more quickly, but yields inaccurate estimates at certain points. The latter point is more evident in Fig. 3.28 where the effective window length is only 250 points. When comparing the results of the graceful forgetting technique with those of the corresponding windowed technique (for example, compare Figs. 3.24 and 3.26 in which the effective window length is 1000), it is noted that the graceful forgetting technique uses more data and yields "comparable" estimates.

3.3.2.3 Selective Forgetting

Using the same selection criterion as that used in *Subsection 3.2.2.3*, the selective

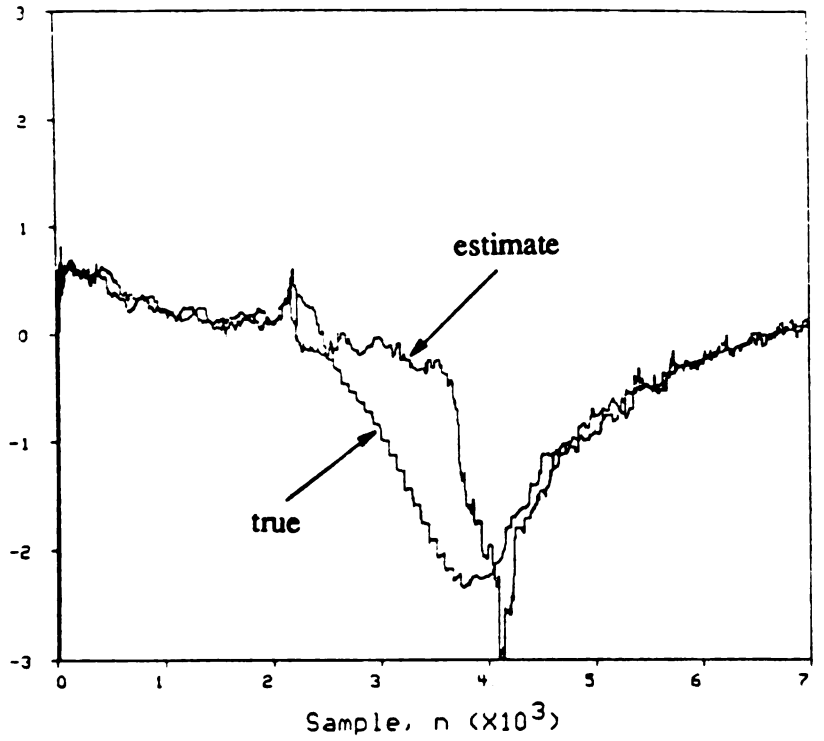


Figure 3.25: Simulation results of the windowed SM-WRLS algorithm for the word seven ($l = 1500$). 14.34% of the data is employed in the estimation process.

forgetting strategy yields the simulation result shown in Fig. 3.29. This technique uses only 12.89% of the data, 72.9% of which are rotated out during the adaptation process. Therefore, it effectively uses (or rotates into and out of the system) 22.29% of the data. It is noted that this technique *slowly* tracks the rapidly varying parameters and yields estimates that are not as accurate as those produced by the other adaptation strategies but uses fewer data. Recall that this technique produces highly accurate estimates in the AR(2) examples (see Figs. 3.13 and 3.14), however, the signal variations in these examples are not as severe as those in the AR(14) example.

3.3.3 Suboptimal SM-WRLS

Figure 3.30 shows the simulation results of the unmodified SM-WRLS algorithm with suboptimal data selection which uses only 4.74% of the data. Compared to the SM-

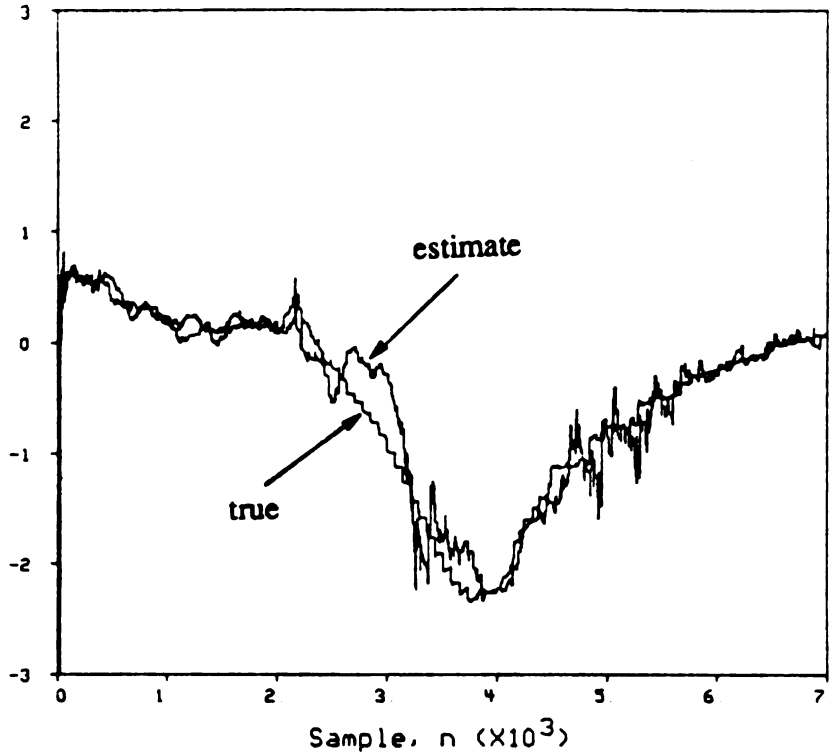


Figure 3.26: Simulation results of the graceful forgetting SM-WRLS algorithm for the word seven ($\mu = 10^{-3}$). 18.66% of the data is employed in the estimation process.

WRLS algorithm (see Fig. 3.22), the suboptimal technique uses fewer data (60% of the data used by SM-WRLS) but produces comparable estimates. It is noted that most (91.3%) of the equations that are accepted by the suboptimal technique are also accepted by the SM-WRLS algorithm. It is also noted that the equations that are accepted by the suboptimal technique but not by the SM-WRLS algorithm lie mostly in regions of fast changing dynamics.

3.3.4 Adaptive Suboptimal SM-WRLS

Two adaptive suboptimal techniques are tested in this section. The first technique is the windowed SM-WRLS algorithm with suboptimal data selection. Figures 3.31 and 3.32 show the simulation results of this technique which uses only 10.93% and 8.71% of the data when using a window of length 500 and 1000, respectively. Compared

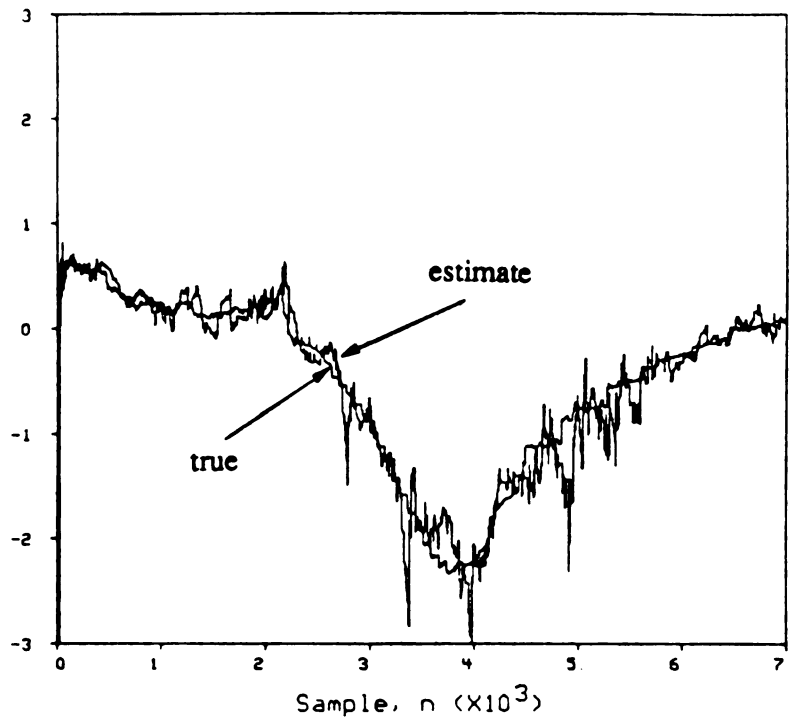


Figure 3.27: Simulation results of the graceful forgetting SM-WRLS algorithm for the word seven ($\mu = 2 \times 10^{-3}$). 27.63% of the data is employed in the estimation process.

to the windowed SM-WRLS algorithm (see Figs. 3.23 and 3.24), the corresponding adaptive suboptimal technique uses fewer data (50% of the data used by the windowed SM-WRLS algorithm), however, it produces estimates that are comparable to but not as smooth as those of the windowed SM-WRLS algorithm (see Figs. 3.31 and 3.32).

The second technique is the selective forgetting SM-WRLS algorithm with suboptimal data selection. The simulation result of this technique is shown in Fig. 3.33 which uses only 8.8% of the data, 63.3% of which are rotated out during the adaptation process, therefore, it effectively uses 14.37% of the data. This technique produces comparable estimates to those of the selective forgetting SM-WRLS algorithm shown in Fig. 3.29 using fewer data.

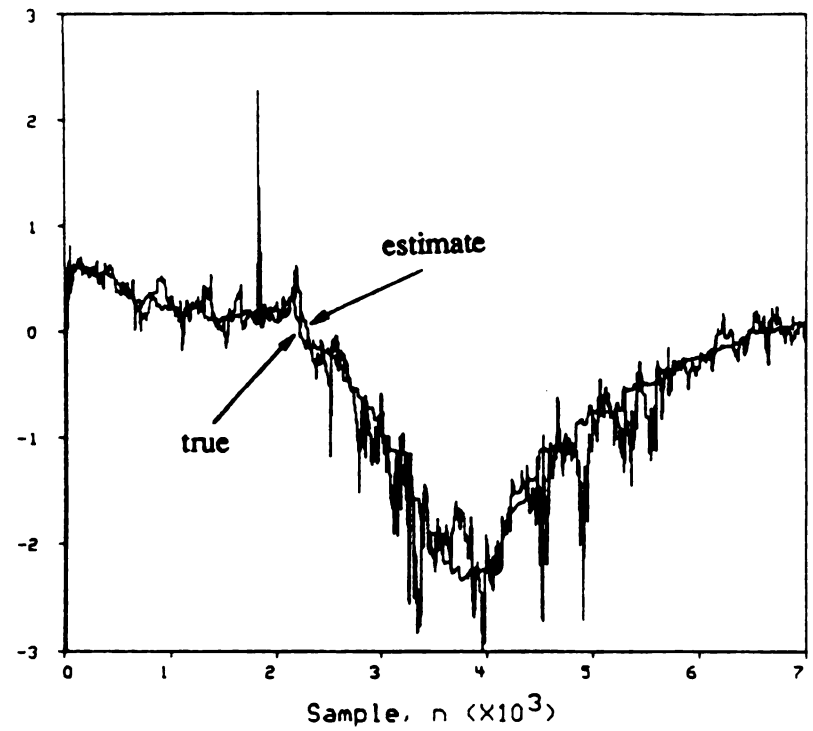


Figure 3.28: Simulation results of the graceful forgetting SM-WRLS algorithm for the word seven ($\mu = 4 \times 10^{-3}$). 35.59% of the data is employed in the estimation process.

3.4 Roundoff Error Analysis

Recently, there has been an increasing interest in the performance of adaptive algorithms in small wordlength environments [8, 49, 50, 51]. Rao and Huang [8] have investigated the effect of small wordlengths on one of the OBE algorithms presented in [14]. Their simulation studies were performed in integer arithmetic using a fixed point implementation of the OBE algorithm. They have shown that the OBE algorithm yields consistently good estimates over a large range of wordlength and performs better than the conventional RLS algorithm for small wordlengths [8].

Marshall and Jenkins [49] have presented a fast quasi-Newton (FQN) adaptive filtering algorithm which is quite robust with respect to small wordlength implemen-

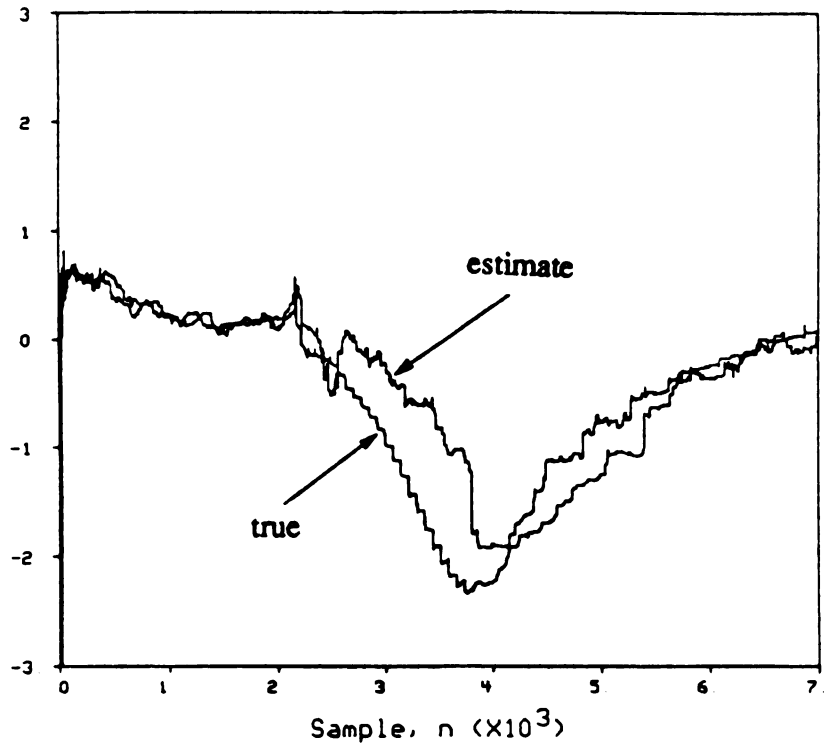


Figure 3.29: Simulation results of the selective forgetting SM-WRLS algorithm for the word seven. 12.89% of the data is employed in the estimation process.

tation and has comparable performance to that of RLS. However, this algorithm is developed based on the assumption that the input is real and wide-sense stationary which yields a symmetric and Toeplitz autocorrelation matrix. The FQN algorithm appears to avoid the numerical problems reported for several fast RLS techniques [51, 52]. The numerical problems consist of numerical inaccuracy in the results (performance degradation) and numerical instability (overflows or underflows) which are caused by finite wordlength computations [50].

All the simulations presented in the previous sections are performed by using C with 32-bit, single precision, floating point arithmetic on a VAX 8600 mainframe running under a Unix operating system. It is the main purpose of this section to test the effect of smaller wordlength computations on the performance of the GR-based SM-WRLS algorithm. The simulations presented here are performed with 16-

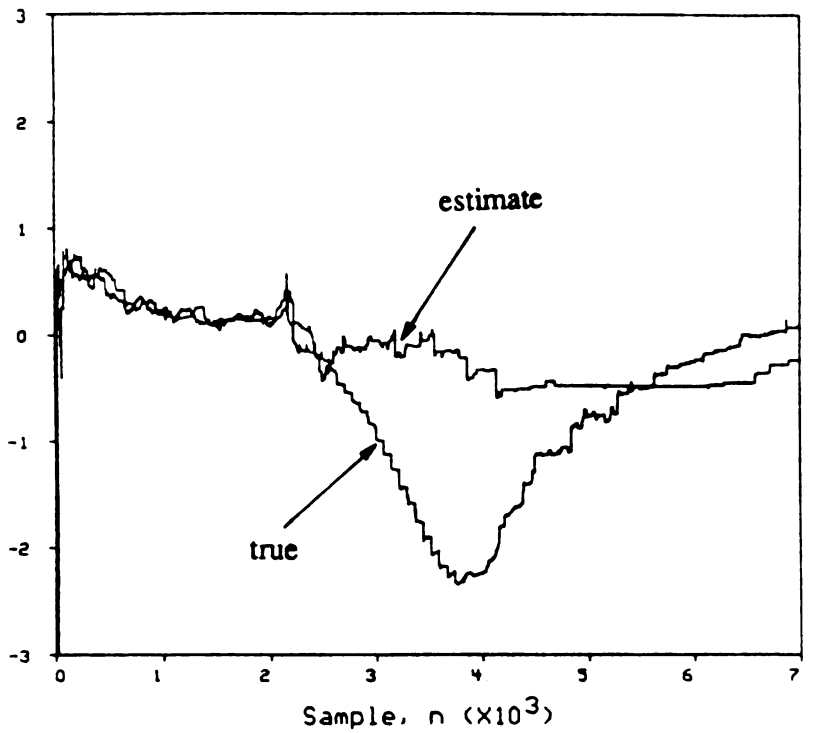


Figure 3.30: Simulation results of the SM-WRLS algorithm with suboptimal data selection for the word seven. 4.74% of the data is employed in the estimation process.

bit, single precision, unnormalized floating point arithmetic. Whenever an overflow (underflow) occurs, the algorithm sets the detected value to the maximum (minimum) possible value which is determined by the wordlength used.

The roundoff error analyses are performed on the same AR(2) models used in *Section 3.2.* Figures 3.34 and 3.35 show the simulation results of the SM-WRLS algorithm for the words four and six using only 1.96% and 2.17% of the data, respectively. Compared to the results of the SM-WRLS algorithm for the word four obtained when using a wordlength of 32-bit (see Fig. 3.6), this algorithm uses slightly more data and yields slightly *better* estimates when a 16-bit wordlength is used. The simulation results for the word six when a 16-bit wordlength is used are almost identical to those obtained when using a wordlength of 32-bit (see Fig. 3.8).

Figure 3.36 shows the simulation results of the windowed algorithm for the word

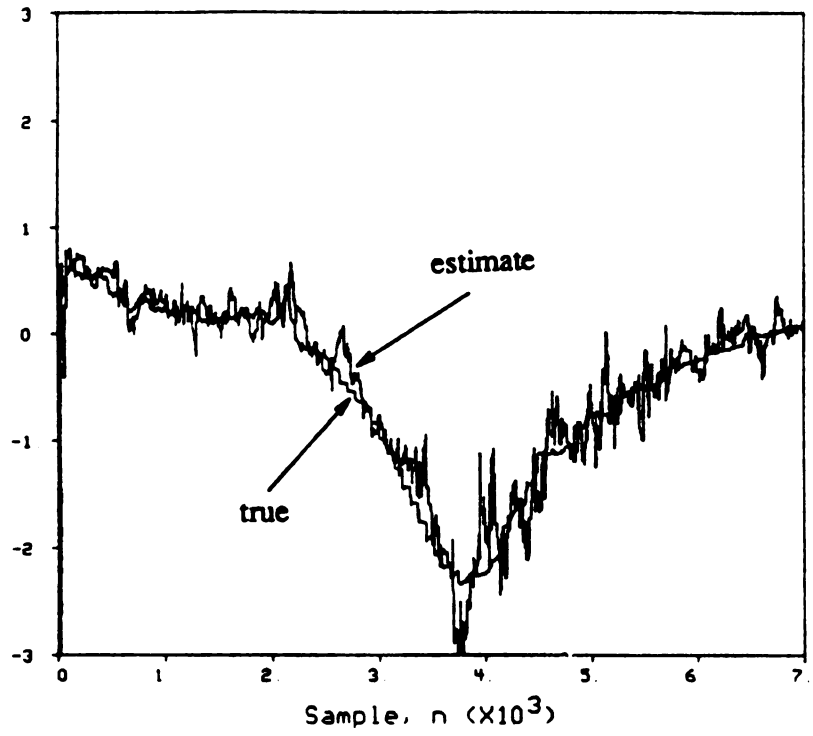


Figure 3.31: Simulation results of the windowed SM-WRLS algorithm with suboptimal data selection for the word seven ($l = 500$). 10.93% of the data is employed in the estimation process.

four using a window length of 1000 and a wordlength of 16-bit. This strategy uses slightly fewer data (5.4%) and produces comparable but slightly less accurate estimates compared with those of the corresponding results obtained when using a wordlength of 32-bit (see Fig. 3.9).

Figure 3.37 shows the simulation results of the graceful forgetting SM-WRLS algorithm when rotating out 0.1%, and using a wordlength of 16-bit. This strategy uses about half the data (3.27%) but produces unacceptable estimates compared with those of the corresponding results obtained when using a wordlength of 32-bit (see Fig. 3.11).

Finally, Figure 3.38 shows the simulation results of the selective forgetting SM-WRLS algorithm using a wordlength of 16-bit. This technique uses only 3.27% of

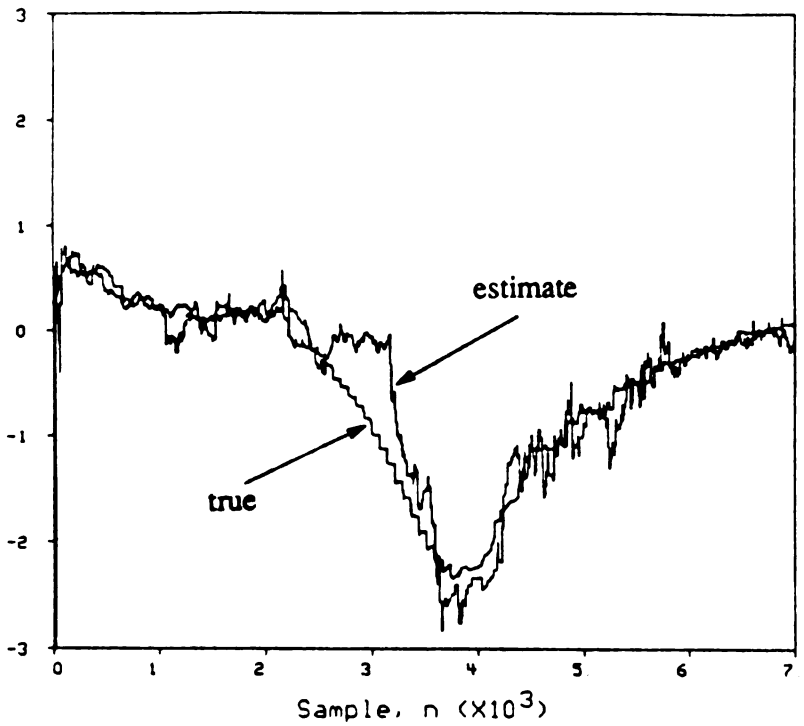


Figure 3.32: Simulation results of the windowed SM-WRLS algorithm with suboptimal data selection for the word seven ($l = 1000$). 8.71% of the data is employed in the estimation process.

the data, 10.5% of which are rotated out during the adaptation process. Therefore, it effectively uses (or rotates into and out of the system) 3.61% of the data. This strategy uses slightly fewer data and produces comparable but slightly less accurate estimates compared with those of the corresponding results obtained when using a wordlength of 32-bit (see Fig. 3.13).

Except for the graceful forgetting strategy, the simulation results using a wordlength of 16-bit are very encouraging. This is due to the infrequent updating behavior inherent in the SM-WRLS algorithms, and hence, slower accumulations of roundoff errors. Also, note that the orthogonal rotation used in the GR-based SM-WRLS algorithms is known to be a numerically stable operation under the assumption that the orthogonal matrices are produced in the absence of roundoff errors [39, 50]. However, the

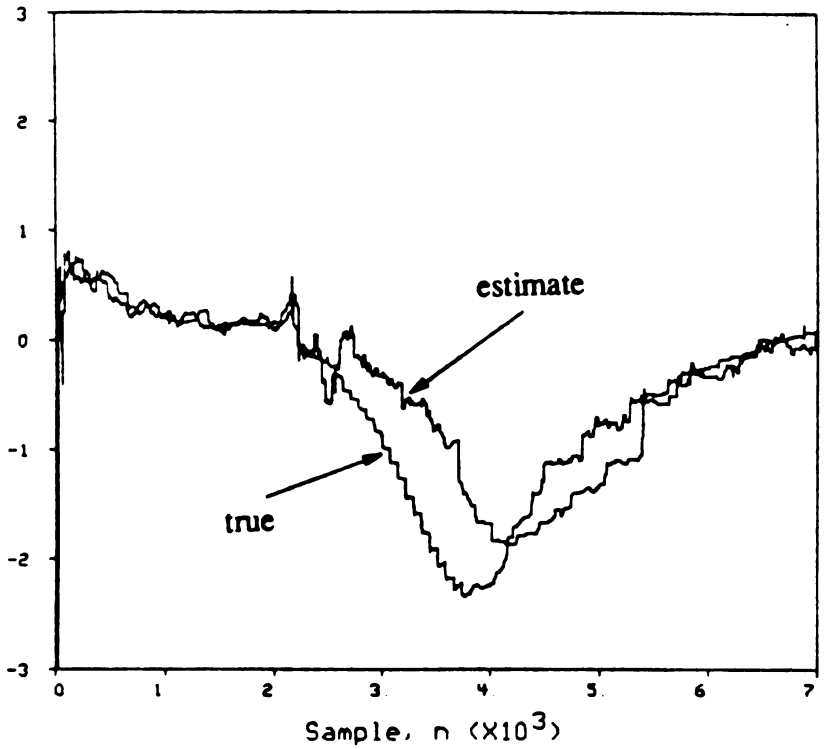
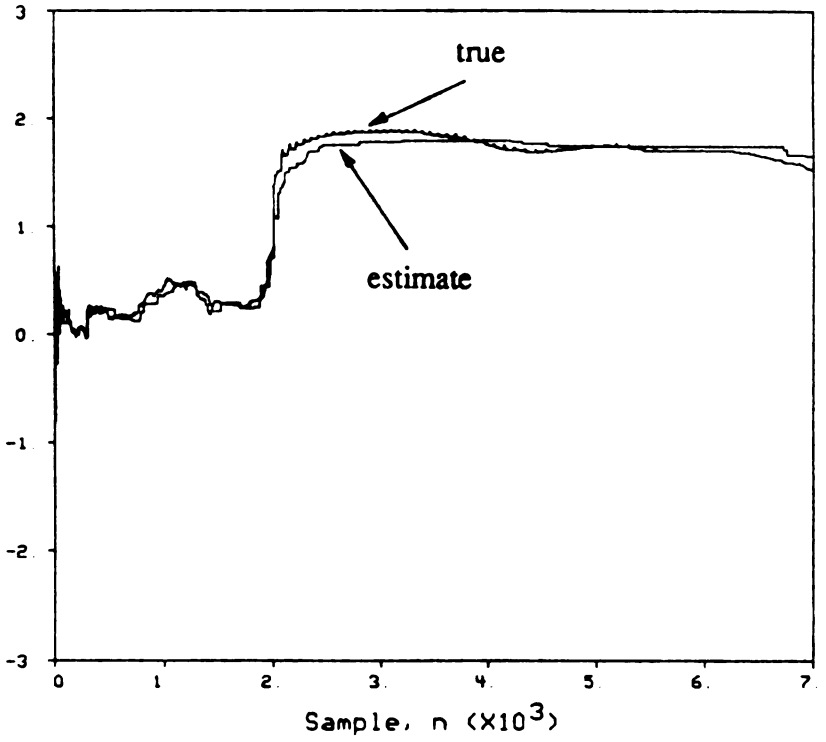


Figure 3.33: Simulation results of the selective forgetting SM-WRLS algorithm with suboptimal data selection for the word seven. 8.8% of the data is employed in the estimation process.

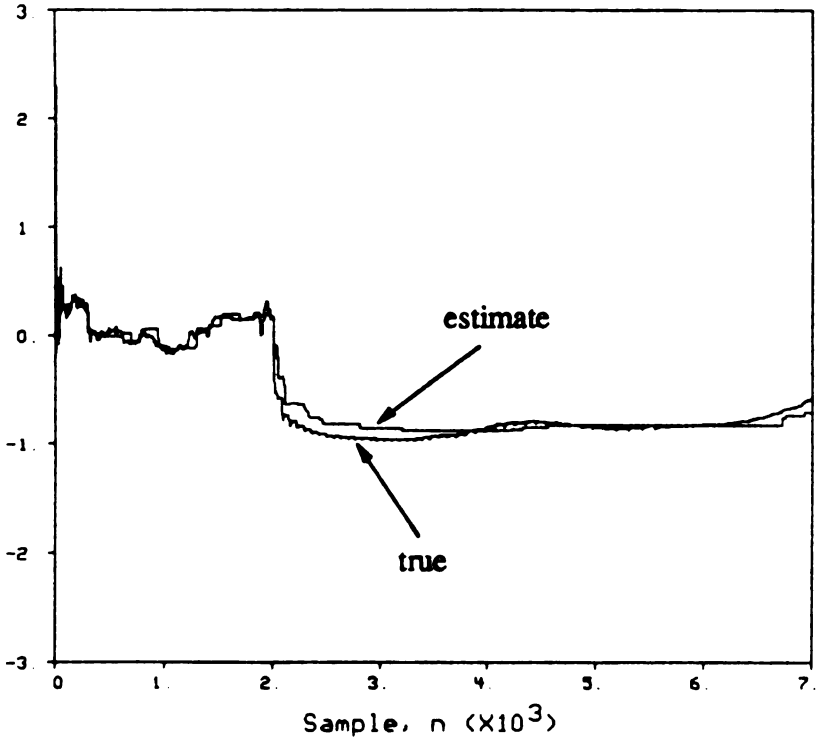
graceful forgetting strategy requires performing a large number of downdates for each previously included equation, which has a severe effect in degrading the performance of the algorithm due to the accumulation of roundoff errors per iteration.

The most widely discussed fast identification algorithm is the Fast Transversal Filter (FTF) [52] which is a fast RLS algorithm, requiring $8m + 15$ flops per iteration (or $\mathcal{O}(m^2)$ for small m) in its more stable form. It includes a "rescue" procedure to prevent divergence due to finite precision effects, and to ensure numerical stability. The rescue procedure is a fast initialization procedure, requiring $3m + 10$ flops per invocation, in which all the accumulated quantities are sacrificed.

It is true that roundoff error experiments performed in this section are $\mathcal{O}(m^2)$, however, it is clear from the simulation results presented in *Sections 3.2 and 3.3*

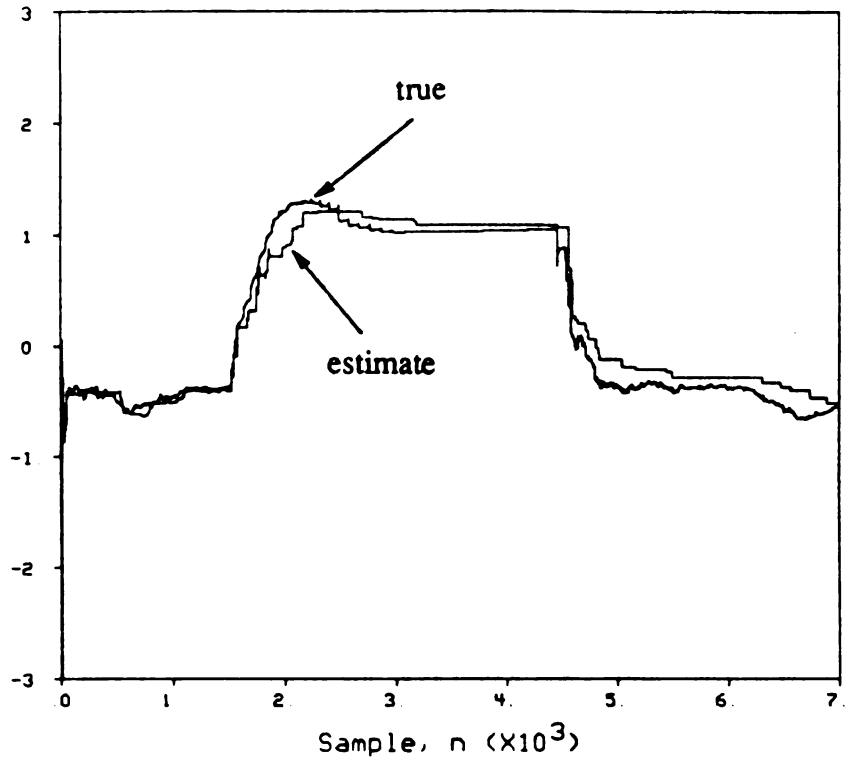


(a)

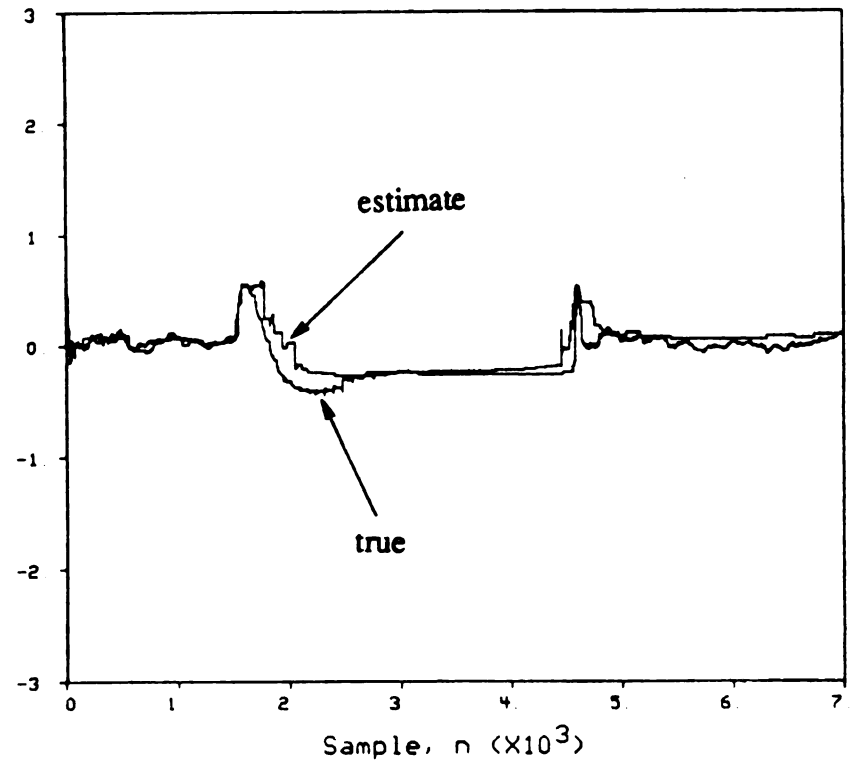


(b)

Figure 3.34: Simulation results of the SM-WRLS algorithm for the word four using a 16-bit wordlength. 1.96% of the data is employed in the estimation process.

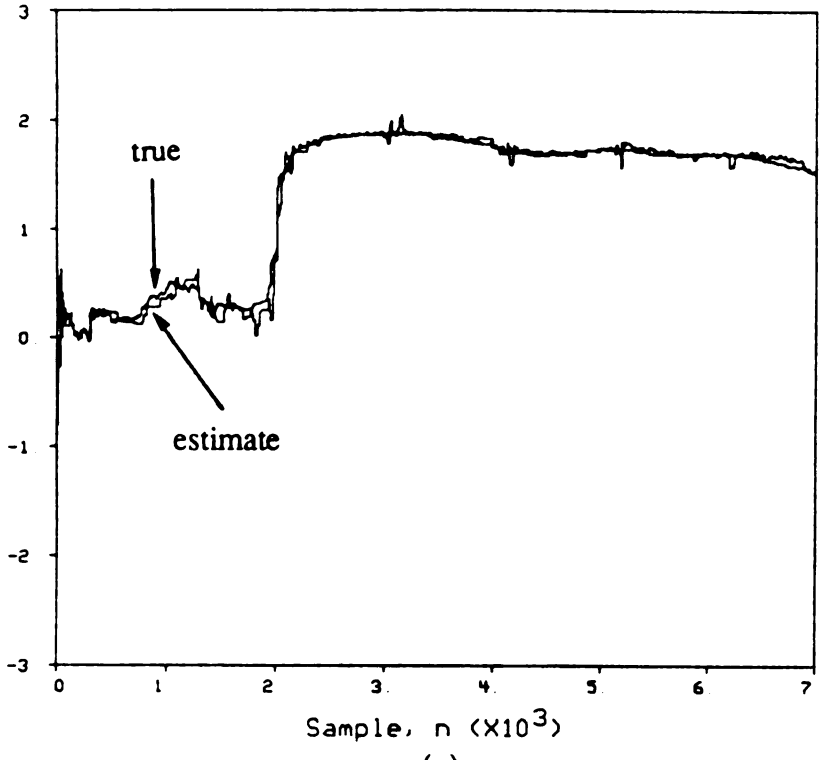


(a)

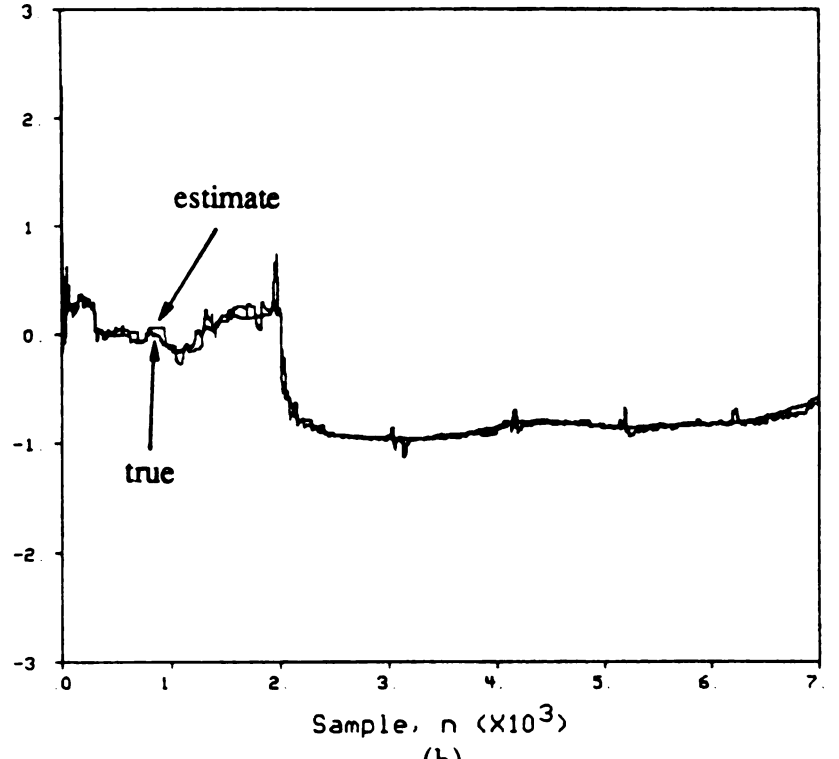


(b)

Figure 3.35: Simulation results of the SM-WRLS algorithm for the word six using a 16-bit wordlength. 2.17% of the data is employed in the estimation process.

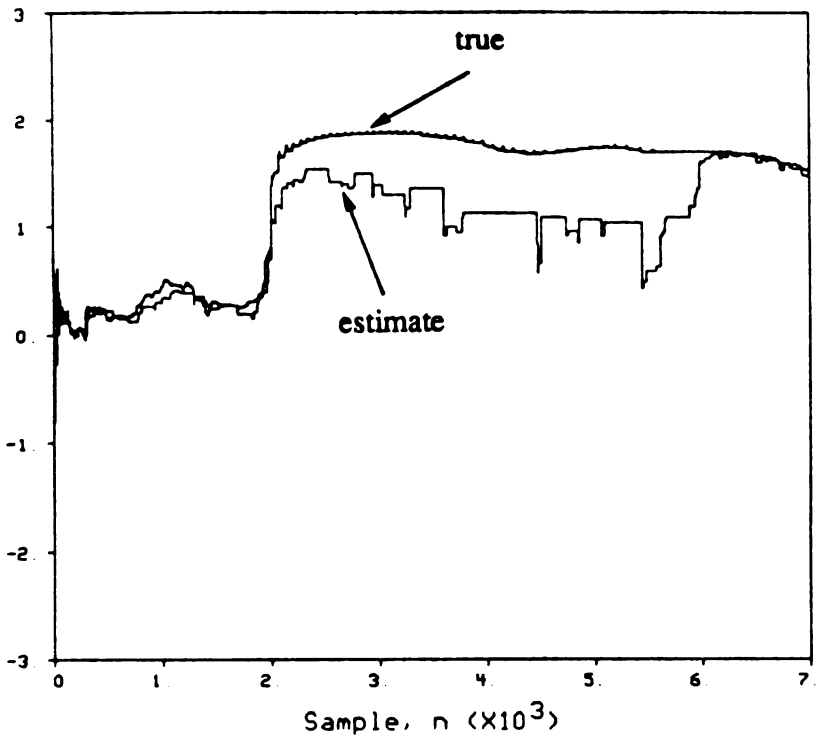


(a)

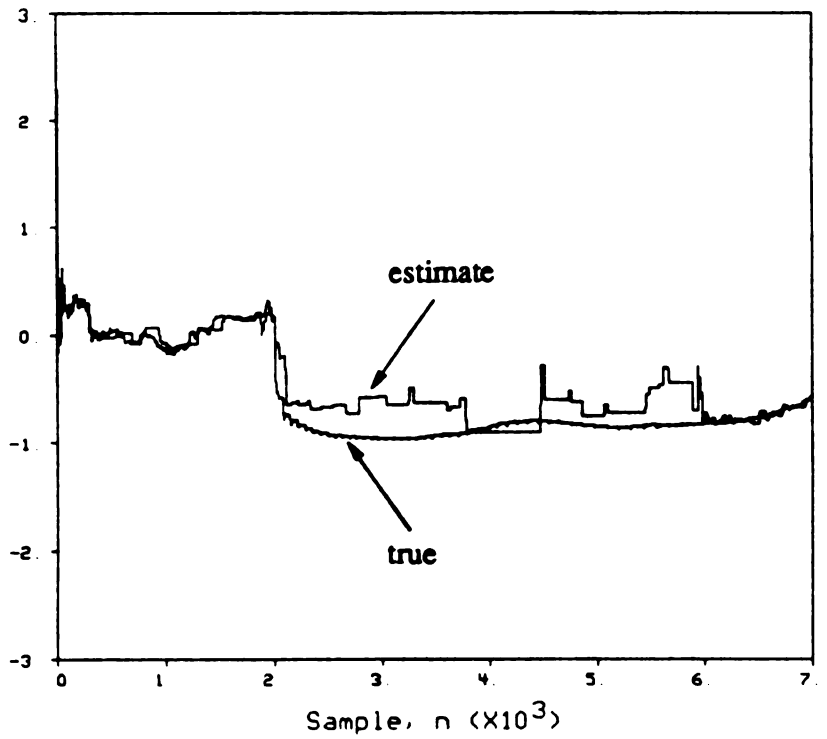


(b)

Figure 3.36: Simulation results of the windowed SM-WRLS algorithm for the word four ($l = 1000$) using a 16-bit wordlength. 5.4% of the data is employed in the estimation process.

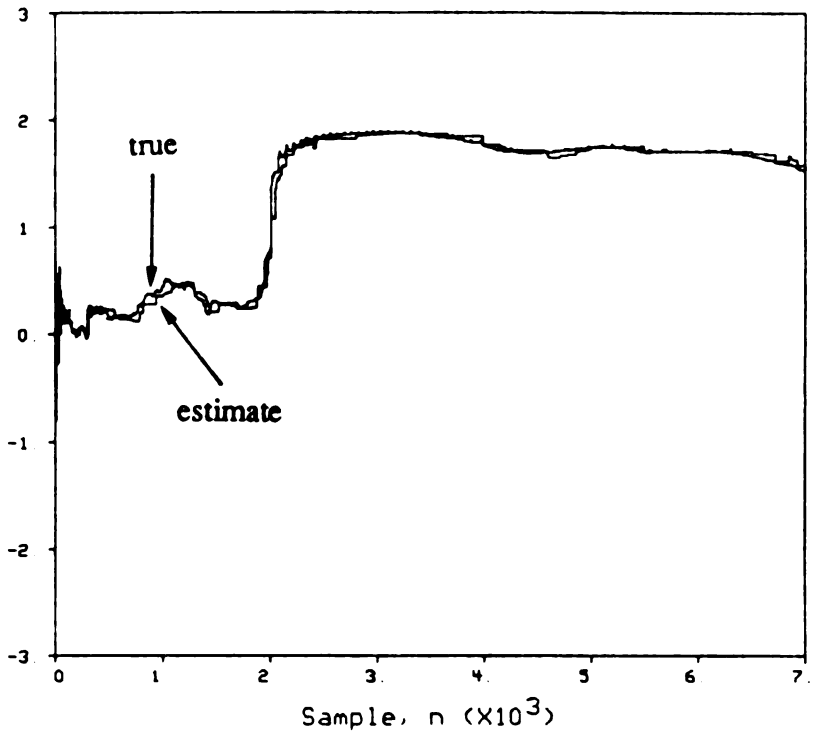


(a)

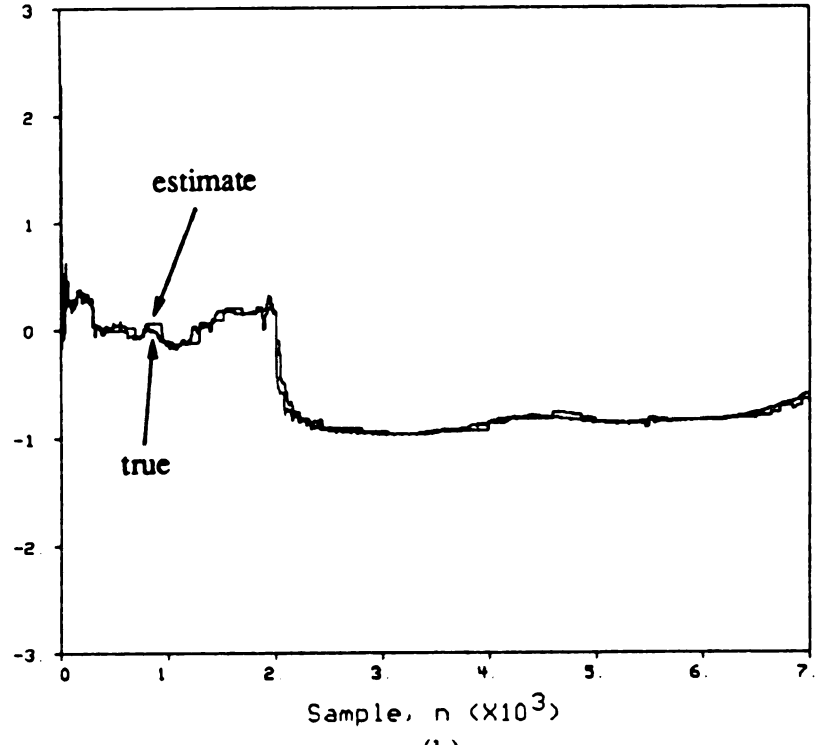


(b)

Figure 3.37: Simulation results of the graceful forgetting SM-WRLS algorithm for the word four ($\mu = 10^{-3}$) using a 16-bit wordlength. 3.27% of the data is employed in the estimation process.



(a)



(b)

Figure 3.38: Simulation results of the selective forgetting SM-WRLS algorithm for the word four using a 16-bit wordlength. 3.27% of the data is employed in the estimation process.

that the suboptimal strategy produces estimates which are not very different from those of the SM-WRLS algorithm and yet requires only m flops per checking, which represents a significant computational savings with respect to FTF. However, it is essential to note that the suboptimal strategy is used to select points and is not directly involved in the update procedure. Therefore, the results of the suboptimal strategy with respect to small wordlength effects are not expected to be significantly different from those of the SM-WRLS presented in this section. In fact, since even fewer computations are generally used in the suboptimal strategy, even *better* finite precision characteristics could be expected.

Chapter 4

Architectures and Complexity Issues

4.1 Introduction

It is noted in *Chapter 1* that one of the reasons why the GR-based SM-WRLS formulation is desirable is that it is amenable to contemporary computing architectures. This chapter is devoted to the development of parallel hardware implementations of the real scalar SM-WRLS algorithm and discussion of their advantages, particularly with regard to their improved computational complexity which improve their potential for real time applications. *Section 4.2* presents a parallel architecture that implements the SM-WRLS algorithm. *Section 4.3* develops an *adaptive compact* parallel architecture that implements virtually any version of the real scalar SM-WRLS algorithm. Finally, a detailed analysis of computational complexity issues is carried out in *Section 4.4*.

4.2 Parallel Architecture for SM-WRLS

The use of a parallel machine benefits the processing in terms of computational complexity by reducing the process to one requiring $\mathcal{O}(m)$ time, rather than $\mathcal{O}(m^2)$

required for the conventional version of the algorithm [5, 6], where m is the number of parameters in the system model. This speed-up is due to the parallel processing inherent in the design.

The main computational requirements of the GR-based SM-WRLS formulation are a GR processor (to effectively execute orthogonal triangularization) to update the matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$ at each step, and a back substitution (BS) processor to solve for the scalar $G(n)$ and also for the estimate $\hat{\boldsymbol{\theta}}(n)$ at each n. Systolic processors for these operations, based on the original work of Gentleman and Kung [42] and Kung and Leiserson [53], are well known. It is the purpose of this section to manifest this algorithm as a parallel architecture based on these processors.

The SM-WRLS algorithm of Fig. 1.2 is mapped into a parallel architecture. The need for implementing the SM-WRLS algorithm on a parallel architecture arises from the fact that portions of the algorithm are compute-bound, specifically, updating the matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$ and computing the value $G(n)$ and the parameter vector $\hat{\boldsymbol{\theta}}(n)$. The architecture that speeds up the computation of these quantities and satisfies the desirable characteristics of systolic arrays (SA's) is shown in Fig. 4.1. Note that although this architecture is designed based on SA design methodologies, it is used *here* to process one equation at a time (more on this below), and therefore, is not used as a SA. This architecture provides an improvement over that described in [9] by replacing the global buses with local buses for communication between adjacent cells. *For simplicity of notation, the figure shows a purely autoregressive case with p=3; AR(3). Once the processor is understood, it should be clear that the architecture is perfectly capable of handling the general ARMAX(p,q) case.* In the discussion below, the vectors $\boldsymbol{x}(n)$ and $\hat{\boldsymbol{\theta}}(n)$ are used, however, the architecture of Fig. 4.1 uses the vectors $\boldsymbol{y}(n)$ and $\hat{\mathbf{a}}(n)$ instead to denote the special case *AR(3)*, where $\boldsymbol{y}(n) = [y(n-1)y(n-2)y(n-3)]^T$.

The architecture is composed of two SA's, several memory management units (i.e.,

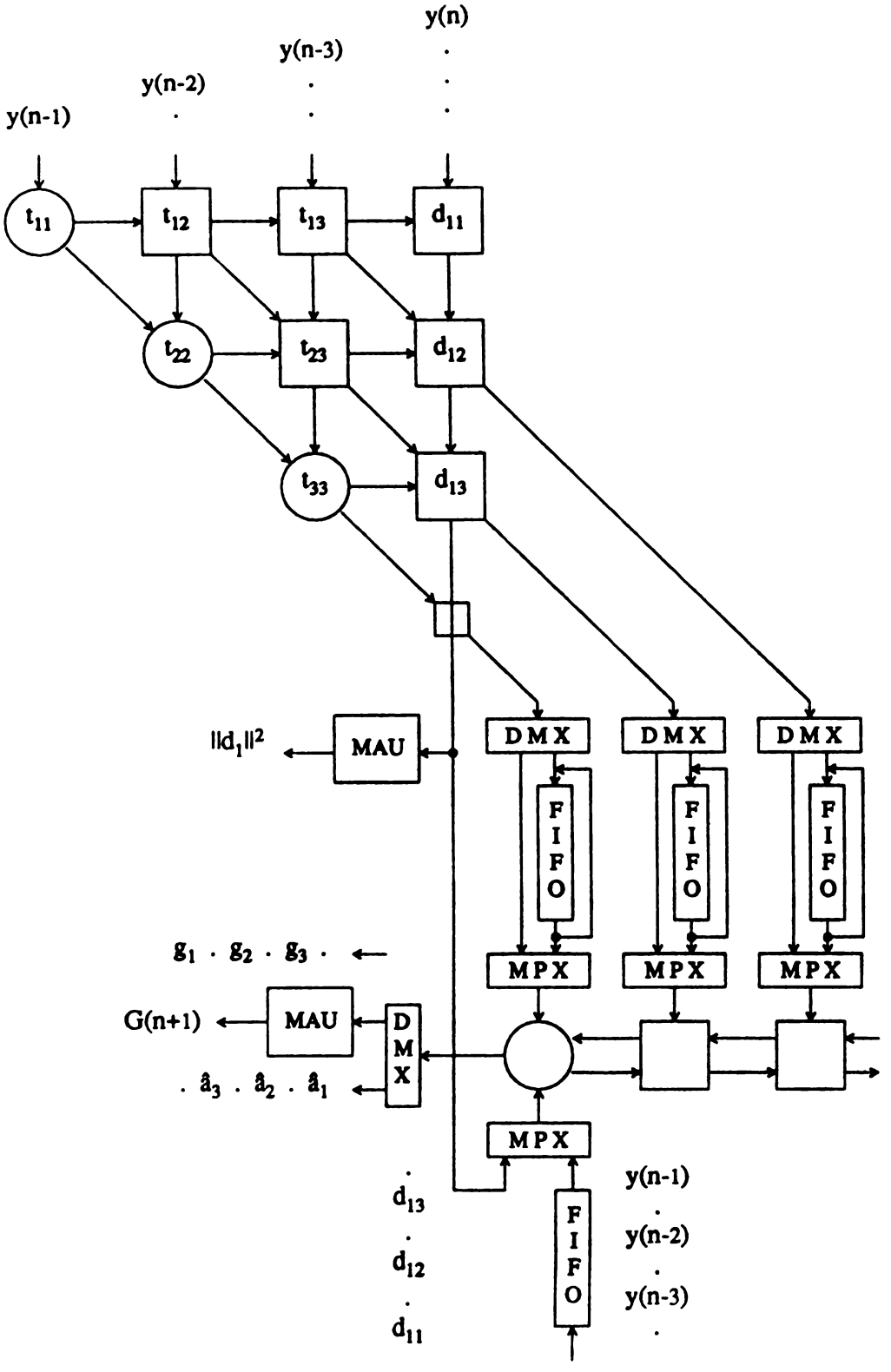
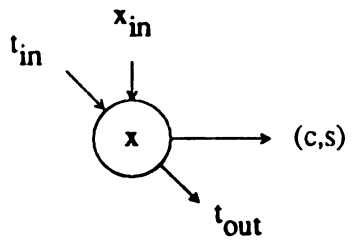


Figure 4.1: Systolic array implementation of the Givens rotation-based SM-WRLS algorithm. For simplicity of notation but without loss of generality, the figure shows a purely autoregressive case with *p=3; AR(3).*

First-in First-out (FIFO) and *Last-in First-out* (LIFO) stacks[7]), multiply-add units (MAU's), multiplexers (MPX's), and demultiplexers (DMX's). The first SA is a *triangular* array that performs orthogonal triangularization using GR's [42, 43] which are particularly suitable for solving recursive linear LS problems. The diagonal (circular) cells perform the "Givens generation" (GG) operations and all other (square) cells in the triangular array perform the GR operations. There is a delay element at the lower right-hand corner of the triangular array that is used to synchronize the flow of the generated entries into the FIFO stacks and to simplify the control of these stacks once they are filled and ready to output their contents to the BS array. The operations performed by this array are shown in Fig. 4.2 [42, 43]. Therefore, the triangular array rotates the new equation into the upper triangular matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$, where the t_{ij} cells update the matrix $\mathbf{T}(n)$ and the right-hand column (d_{1j}) cells update the vector $\boldsymbol{d}_1(n)$. The element t_{ij} denotes the ij^{th} element of the matrix $\mathbf{T}(n)$ and the element d_{1j} denotes the j^{th} element of the vector $\boldsymbol{d}_1(n)$.

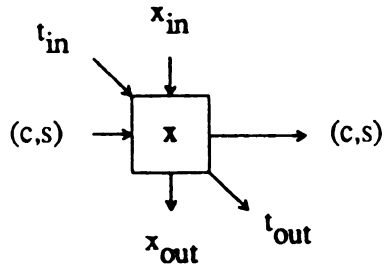
The second array is a linear array that performs the BS operations shown in Fig. 4.3 [53]. Note that the same BS array is used to solve for the vectors $\boldsymbol{g}(n+1)$ and $\hat{\boldsymbol{\theta}}(n)$ with the data provided to the appropriate cells in the required order by the FIFO and LIFO stacks. The FIFO stacks feed the lower triangular matrix $\mathbf{T}^T(n)$ to solve for the vector $\boldsymbol{g}(n+1)$, and hence, the value $G(n+1)$. The LIFO stacks feed the upper triangular matrix $\mathbf{T}(n)$ to solve for the parameter vector $\hat{\boldsymbol{\theta}}(n)$. The values $G(n+1) = \| \boldsymbol{g}(n+1) \|_2^2$ and $\| \boldsymbol{d}_1(n) \|_2^2$ are generated by the MAU's shown in Fig. 4.4. The number of segments in each stack is equal to the number of elements the stack holds. Therefore, the leftmost stack consists of m segments, whereas the rightmost stack has only one segment.

[7]Note that the architecture shown in Fig. 4.1 does not include any of the LIFO stacks that were used to hold the matrix $\mathbf{T}(n)$ in the architecture reported in [9]. This is achieved by slightly increasing the complexity of the cells used in the triangular array so that they can be used as storage elements as well. This is facilitated by the diagonal interconnections between adjacent cells which now constitute the LIFO stacks.



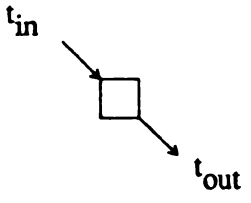
(a)

If $(x_{in} = 0)\{$
  $c = 1$
  $s = 0$
  $t_{out} = t_{in}$
$\}$
else$\{$
  temp $= [x^2 + (x_{in})^2]^{1/2}$
  $c = x$ / temp
  $s = x_{in}$ / temp
  $x =$ temp
  $t_{out} = x$
$\}$



(b)

If $(x_{in} = 0$ & $c = 1$ & $s = 0)$
  $t_{out} = t_{in}$
else$\{$
  $x = c\,x + s\,x_{in}$
  $x_{out} = -s\,x + c\,x_{in}$
  $t_{out} = x$
$\}$



(c)

$t_{out} = t_{in}$

When the array is used as LIFO stacks: First cycle: $t_{out} = x$ (t_{ij} cells)
$x_{out} = x$ (d_1 cells)
All following cycles: $t_{out} = t_{in}$ (t_{ij} cells)
$x_{out} = x_{in}$ (d_1 cells)

Figure 4.2: The operations performed by the cells used in the triangular array of Fig. 4.1. (a) The Givens generation (GG) cells, (b) the Givens rotation (GR) cells, and (c) the delay element.

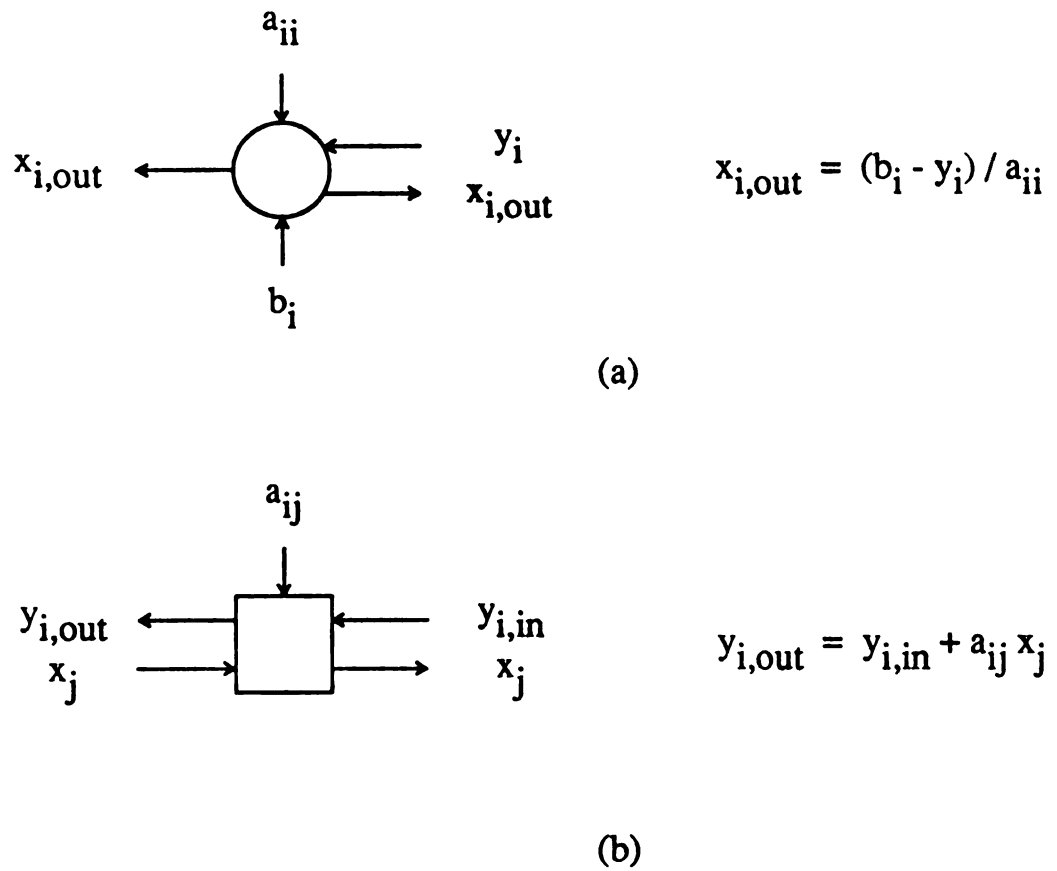


Figure 4.3: The operations performed by the back substitution array. (a) The left-end processor and (b) the multiply-add units. The initial $y_{i,in}$ entering the rightmost cell is set to 0.

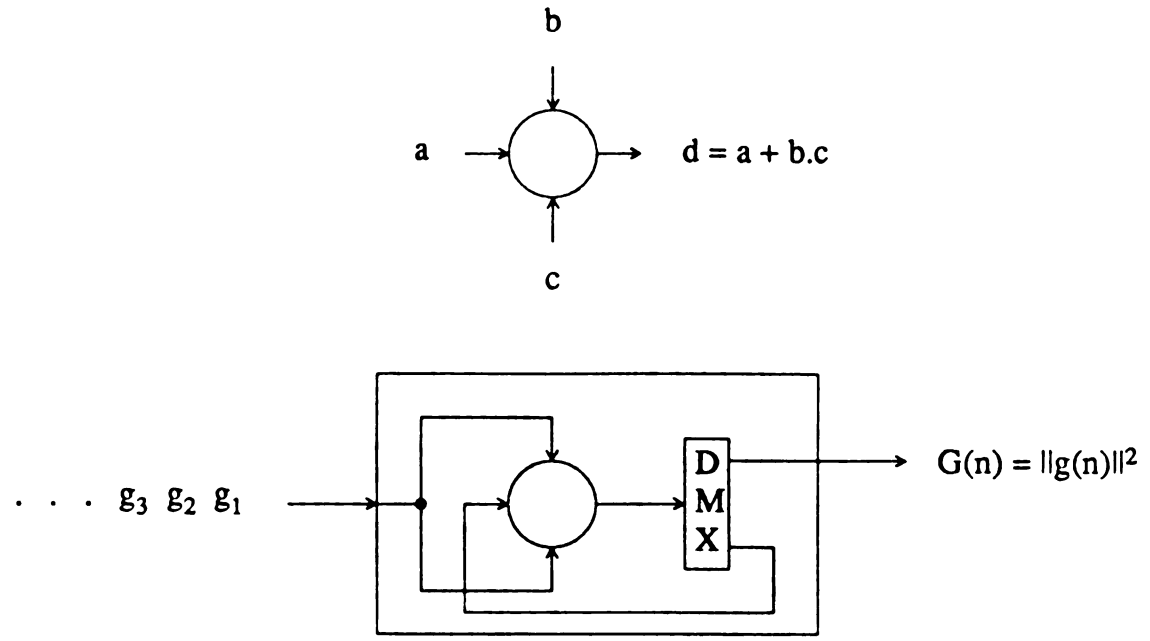


Figure 4.4: The multiply-add unit used in Fig. 4.1.

The system shown in Fig. 4.1 works as follows. The first m+1 equations (with appropriate weights) enter the triangular array (from the top) in a skewed order, and the matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$ is generated and stored inside the cells. A shift register with appropriate feedback connection and data sequencing can be used to hold and feed the equation to the array. The initial upper triangular matrix residing in the array, and corresponding to the first $m+1$ equations, is ready after $3m+1$ GG time cycles. The GG time cycle is that of the triangular array performing the GG operations without square roots, which is the time required to perform five floating point operations (flops) [43, 55], where one flop is defined as one floating point multiplication plus one floating point addition. Note that in order to prevent data collision, the flow of data in the triangular array moves along a corresponding wavefront and is controlled by the slowest cells in the array, *viz.*, GG cells. Note that the data are fed to the array one (skewed) equation at a time, therefore, the contents of each cell remain constant

after the completion of the current recursion. After the new equation is rotated into the matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$, the vectors $\boldsymbol{g}(n+1)$ and $\hat{\boldsymbol{\theta}}(n)$ are computed. All the t_{ij} cells in the triangular array load their contents on the t_{out} lines ($t_{out} \leftarrow x$), and then pass these elements across the diagonal lines ($t_{out} \leftarrow t_{in}$) (see Figs. 4.1 and 4.2). This obviates the LIFO stacks. The FIFO stacks are still needed, however, to compute the vector $\boldsymbol{g}(n+1)$. The FIFO stacks are filled with the elements of the lower triangular matrix $\mathbf{T}^T(n)$ as they are generated. This is done by loading the t_{ij} entry on the t_{out} line ($t_{out} \leftarrow x$) when it is generated. This entry propagates down the diagonal cells (with the function $t_{out} \leftarrow t_{in}$) until it arrives at and fills the appropriate FIFO stack. For the cells in the right-hand column, which generate the vector $\boldsymbol{d}_1(n)$, the operations are different because it is this column that constitutes the LIFO stack for the vector $\boldsymbol{d}_1(n)$. Hence, after the new equation is rotated into the array, all the cells in the right-hand column load their contents on the x_{out} lines ($x_{out} \leftarrow x$), and then they pass these elements down the column ($x_{out} \leftarrow x_{in}$) (see Figs. 4.1 and 4.2). Note that the output x_{out} leaving the bottom cell in this column passes through the delay element and is routed to both the MAU and the MPX feeding the d_{1j} elements to the BS array. Note that the elements d_{1m} and t_{mm} leave the triangular array at the same time because of this delay element. The timing diagram of the triangular array is shown in Table 4.1. In this table, the inputs refer to the elements fed to the cells in the top row. The circle (○) represents the GG cell and the square (□) represents the GR cell (see Fig. 4.1). The outputs refer to the elements that are produced in the array cells and are written column wise; i.e., the first column in the table represents the first column in the array, and so on.

The BS array is used to solve for the vectors $\boldsymbol{g}(n+1)$ and $\hat{\boldsymbol{\theta}}(n)$. The vector $\boldsymbol{g}(n+1)$ is solved using (1.31) and the parameter vector $\hat{\boldsymbol{\theta}}(n)$ using (1.28). Therefore, the vector $\boldsymbol{g}(n+1)$ is generated from the matrix $\mathbf{T}^T(n)$, which is residing in the FIFO stacks, and the vector $\boldsymbol{x}(n+1)$ which is available. The entries are fed to the

Table 4.1: The timing diagram of the triangular array of Fig. 4.1.

Time	Inputs ○	□	□	□	Outputs $[\mathbf{T}(n) \mid \boldsymbol{d}_1]$			
0	$y(n-1)$							
1		$y(n-2)$			t_{11}			
2			$y(n-3)$			t_{12}		
3				$y(n)$		t_{22}	t_{13}	
4							t_{23}	d_{11}
5							t_{33}	d_{12}
6								d_{13}

BS array every other BS time cycle, where the BS time cycle is the time required to perform one flop. As the g_i entries are output from the left-end processor of the BS array, they enter the MAU to generate the value $G(n+1)$ after $2m+1$ BS time cycles. Likewise, the parameter vector $\hat{\boldsymbol{\theta}}(n)$ is generated using the matrix $\mathbf{T}(n)$ and the vector $\boldsymbol{d}_1(n)$ which are stored in the triangular array. Starting one BS time cycle after the initiation of the first BS operation, the appropriate entries (of the second BS operation) are also fed to the BS array every other BS time cycle. The parameter vector $\hat{\boldsymbol{\theta}}(n)$ is output from the left-end processor of the BS array in reversed order and interleaved with the vector $\boldsymbol{g}(n+1)$ as shown in Fig. 4.1. The value $\| \boldsymbol{d}_1(n) \|_2^2$ is generated using a MAU one BS time cycle after the last (m^{th}) element of the vector $\boldsymbol{d}_1(n)$ is generated. The timing diagram of the BS array is shown in Table 4.2 in which the inputs refer to the elements fed to the shown cells, and the outputs refer to the elements produced by the left-end processor in the array.

The values $\kappa(n)$ and $\epsilon_n^2(n+1)$ are then computed, and hence, the value λ_{n+1} which determines whether the new equation is to be accepted or not. If the new equation is

Table 4.2: The timing diagram of the back substitution array of Fig. 4.1.

Time	Inputs ○	□	□	Outputs ○
0	$t_{11}, y(n-1)$			
1	t_{33}, d_{13}	t_{12}		g_1
2	$t_{22}, y(n-2)$	t_{23}	t_{13}	$\hat{a}_3$
3	t_{22}, d_{12}	t_{23}	t_{13}	g_2
4	$t_{33}, y(n-3)$	t_{12}		$\hat{a}_2$
5	t_{11}, d_{11}			g_3
6				$\hat{a}_1$

accepted, then the *weighted* new equation enters the triangular array and the same procedure described above takes place producing a new $[\mathbf{T}(n+1) \mid \boldsymbol{d}_1(n+1)]$ matrix after $2m+1$ GG time cycles, and therefore, an updated $G(n+2)$, $\hat{\boldsymbol{\theta}}(n+1)$, and $\kappa(n+1)$. On the other hand, if the new equation is rejected, then the triangular array preserves its contents (hold state), but the value $G(n+2)$ is updated to make the decision concerning the next equation. In the latter case, the same $\mathbf{T}^T(n+1)$ matrix is used as the previous $\mathbf{T}^T(n)$ matrix, and hence, the feedback on the FIFO stacks. This procedure is repeated for every new equation.

4.3 An Adaptive Compact Parallel Architecture

The basic idea behind the compact architecture is to map the triangular array of Fig. 4.1 into a linear array (called the GR array), that is, mapping all of the GG cells into one GG cell and all the GR cells that are on the same diagonal into one GR cell. This constitutes a permissible schedule because the projection vector, $\vec{d}$, is parallel to

the schedule vector, $\vec{s}$, and all the dependency arcs flow in the *same* direction across the hyperplanes [36, Ch. 3]. In other words, this schedule satisfies the conditions

$$\vec{s}^T\vec{d} \;>\; 0 \tag{4.1}$$

$$\text{and} \quad \vec{s}^T\vec{e} \;\geq\; 0, \quad \text{for any dependence arc } \vec{e}. \tag{4.2}$$

The compact architecture implementation of the adaptive SM-WRLS algorithm is shown in Fig. 4.5. The operations performed by this architecture are similar to those of Fig. 4.1 with the exception that the GG and GR cells are now capable of performing back rotation (see Fig. 4.6) and are embedded in a slightly more complicated modules needed for scheduling. These modules are called GG′ and GR′, and are shown in Fig. 4.7.

This architecture uses $\mathcal{O}(m)$ cells (one GG′ cell and m GR′ cells) compared with $\mathcal{O}(m^2)$ cells (m GG cells and $\frac{m^2+m}{2}$ GR cells) used in the architecture shown in Fig. 4.1, and yet has the same computational efficiency (per equation). Note however that the LIFO stacks that were embedded in the triangular array of Fig. 4.1 are now needed to hold the matrix $\mathbf{T}(n)$.

The system shown in Fig. 4.5 works as follows. Each equation (with its optimal weight) enters the GR array (from the top) in a skewed order, and the matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$ is generated and stored in the appropriate memory units. Note that the GR array can operate in two modes, forward ($\delta = +1$) and backward ($\delta = -1$) rotation modes (see Fig. 4.6). In the backward rotation mode, the equation to be (partially or completely) removed is re-introduced to the GR array with the appropriate weight (see *Section 2.4.1*). At the end of each recursion, the FIFO stacks contain the lower triangular matrix $\mathbf{T}^T(n)$ needed to solve for the vector $\boldsymbol{g}(n+1)$, and hence, the value $G(n+1)$. The LIFO stacks contain the upper triangular matrix $\mathbf{T}(n)$ needed to solve for the parameter vector $\hat{\boldsymbol{\theta}}(n)$. The values $G(n+1) \;=\; \| \boldsymbol{g}(n+1) \|_2^2$ and

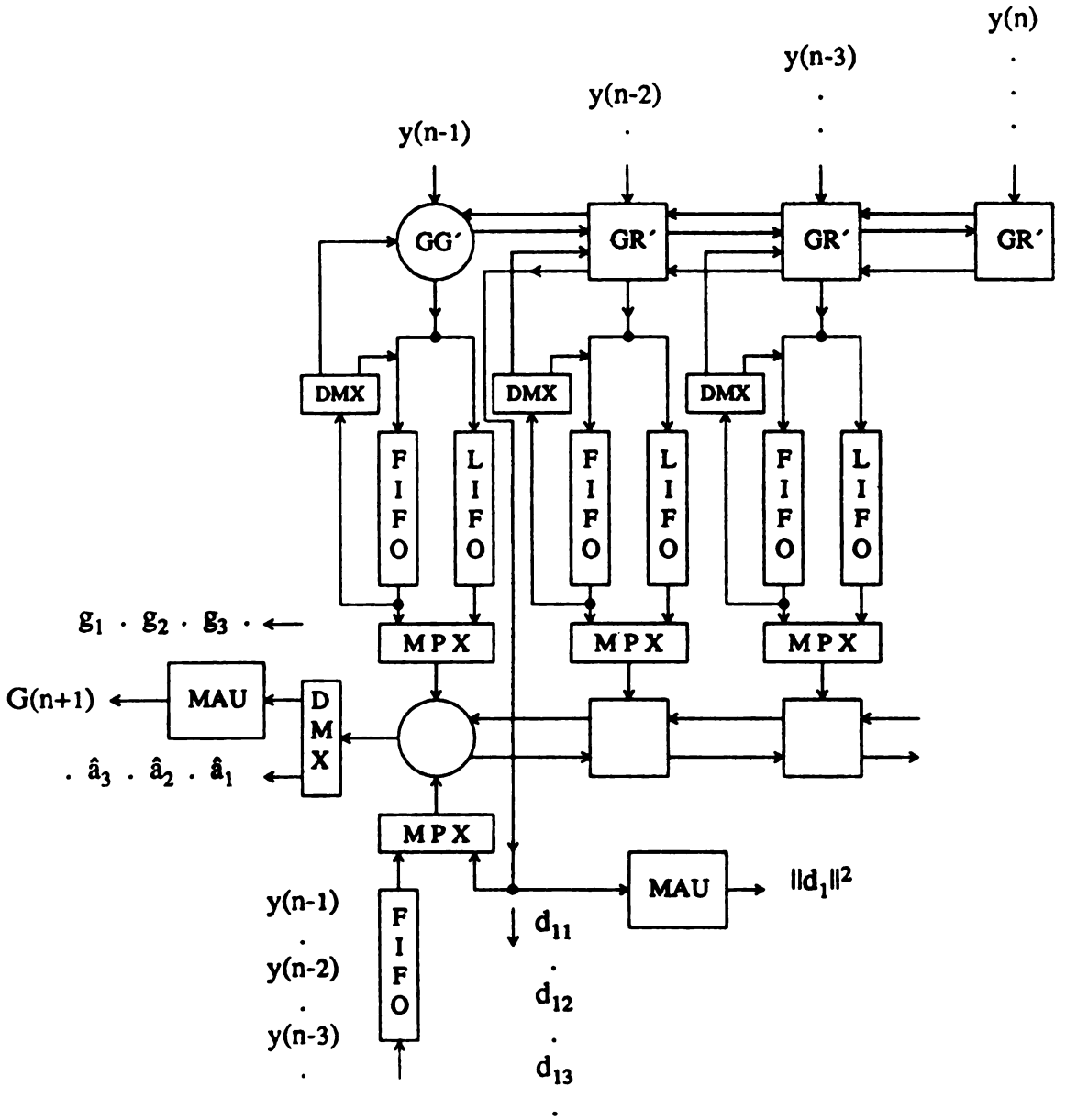
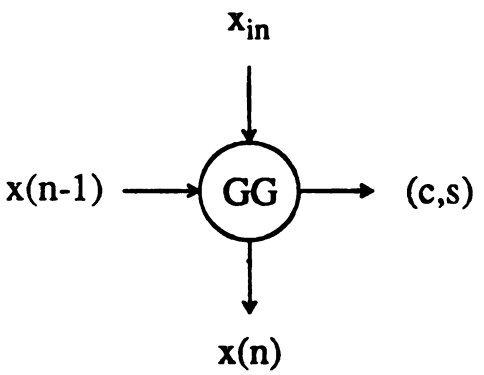
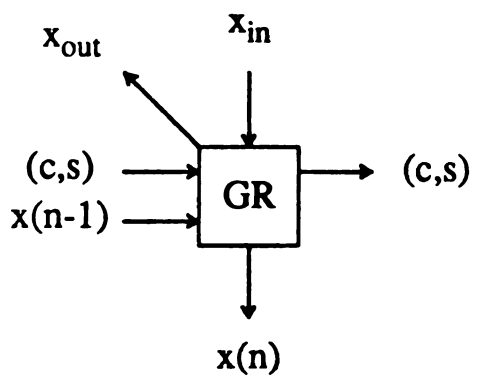


Figure 4.5: A compact architecture implementation of the adaptive SM-WRLS algorithm.



If ($x_{in} = 0$){
  c = 1
  s = 0
}
else{
  $x(n) = [x(n-1)^2 + \delta(x_{in})^2]^{1/2}$
  c = x(n-1) / x(n)
  $s = x_{in}$ / x(n)
}

(a)



$x(n) = c\,x(n-1) + s\,x_{in}\delta$

$x_{out} = -s\,x(n-1)\delta + c\,x_{in}\delta$

(b)

Figure 4.6: The operations performed by (a) the GG and (b) the GR cells used in the modules of Fig. 4.7. $\delta = +1\ (-1)$ for rotating the equation into (out of) the system.

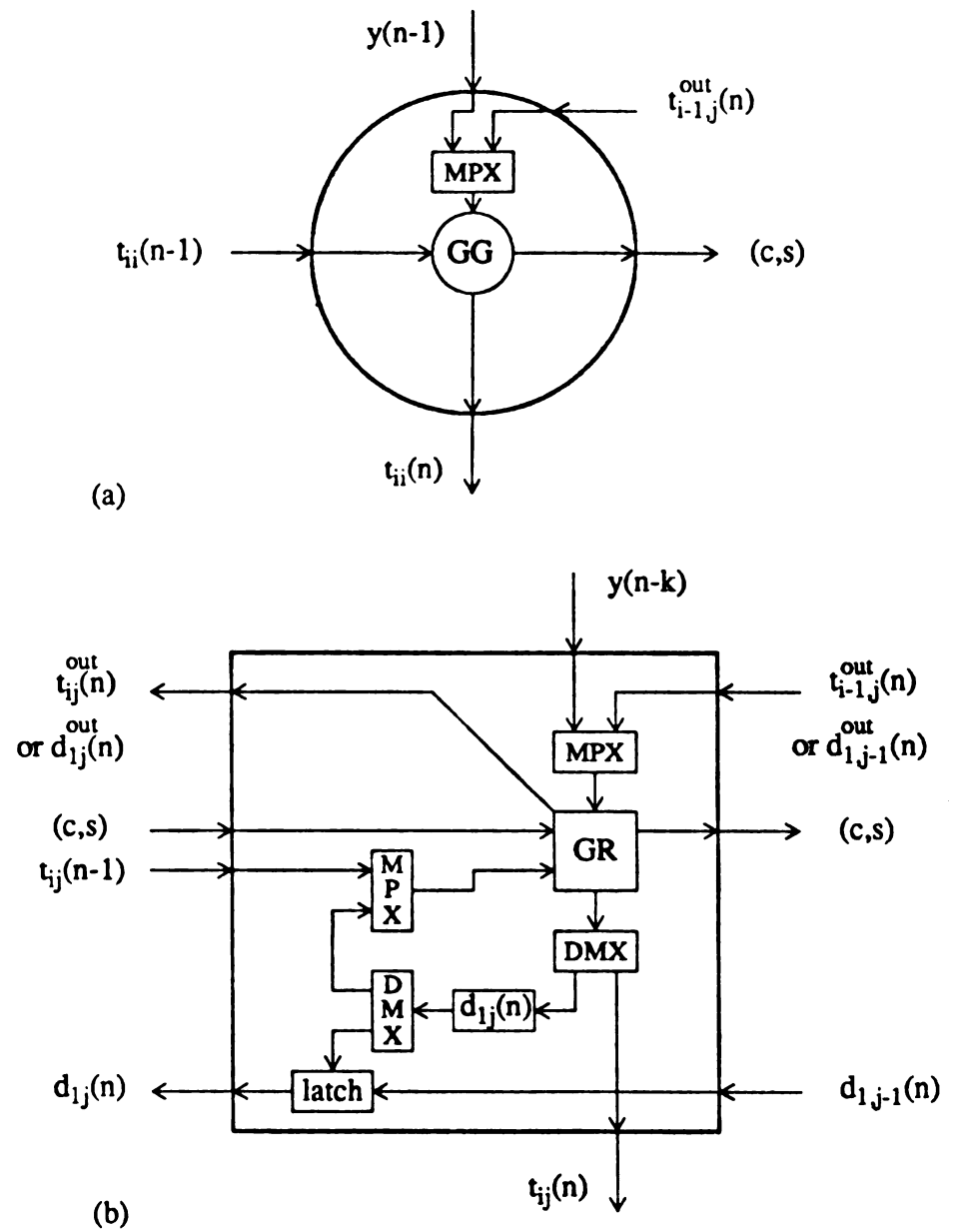


Figure 4.7: (a) The GG′ module and (b) the GR′ module used in the architecture of Fig. 4.5.

Table 4.3: The timing diagram of the Givens rotation (GR) array of Fig. 4.5.

Time	Inputs ○	□	□	□	Outputs ○	□	□	□
0	$y(n-1)$							
1		$y(n-2)$			t_{11}			
2			$y(n-3)$			t_{12}		
3				$y(n)$	t_{22}		t_{13}	
4						t_{23}		d_{11}
5					t_{33}		d_{12}	
6						d_{13}		

$\| \boldsymbol{d}_1(n) \|_2^2$ are generated by the MAU's. Note that the values which were propagating downward in the triangular array of Fig. 4.1 are now propagating leftward due to the new scheduling. Note also that the vector $\boldsymbol{d}_1(n)$ is treated differently from the matrix $\mathbf{T}(n)$. When the element d_{1j} is computed, it is stored in an internal register in the GR′ cell (see Fig. 4.7). After generating and storing the matrix $[\mathbf{T}(n) \mid \boldsymbol{d}_1(n)]$, the processor is ready to compute the vectors $\boldsymbol{g}(n+1)$ and $\hat{\boldsymbol{\theta}}(n)$ using the BS array. The vector $\boldsymbol{d}_1(n)$ is downloaded into the latches which serve as a LIFO stack used in conjunction with the other LIFO stacks (containing the matrix $\mathbf{T}(n)$) to solve for the parameter vector $\hat{\boldsymbol{\theta}}(n)$. The timing diagram of the GR array is shown in Table 4.3 in which the input (output) columns show the elements that are input (output) to (from) the corresponding GG (○) or GR (□) cells. Compared to the triangular array of Fig. 4.1, it is noted that the cell utilization per update (or downdate) has increased by a factor of 2.25 for the case when $m = 3$, or by $\frac{.5m^2+1.5m}{m+1}$ in general. The operations and timing diagram of the BS array is described in detail in *Section 4.2.*

The architectures of Figs. 4.1 and 4.5 can also be used for the suboptimal SM-

WRLS algorithm, however, they are not utilized to the same extent as they are in the SM-WRLS algorithm because the suboptimal SM-WRLS typically uses fewer data. The infrequent updating feature of this algorithm (and virtually all SM-WRLS algorithms) might provide processing advantages in the systolic (and other parallel processing) schemes by permitting the sharing of processing time and resources.

The complex scalar case can also be implemented using the same architectures of Figs. 4.1 and 4.5 which now perform complex GG and GR operations. These operations are well-defined and are found in [56]. However, the generalized complex vector case is not readily mapped into similar architectures. The generalized architecture that efficiently implements this case requires further research.

4.4 Computational Complexities

The computational complexities (in flops per equation) for the scalar sequential GR-based SM-WRLS algorithm and for that implemented using the architecture of Fig. 4.1 are shown in Table 4.4. Note that the complexities of the parallel GR-based SM-WRLS algorithm shown in Table 4.4 are *parallel* complexities in the sense that they denote the *effective* number of operations per equation, though many processors can be performing this number of operations simultaneously. Accordingly the parallel complexity indicates the time it takes the parallel architecture to process the data regardless of the total number of operations performed by the individual cells. The GG and GR operations constitute the main computational load of the algorithm as shown in Table 4.5. In this table, the number of flops associated with the GR's is multiplied by five to account for the GG cycle time (see *Section 4.2*). These operations are avoided when the equation is rejected, and thus, a significant savings in computation time. Since this technique uses $\mathcal{O}(m)$ flops per equation when implemented using the parallel architecture, it is clearly advantageous with respect to the

Table 4.4: Computational complexities (in flops per equation) for the real scalar sequential and parallel GR-based SM-WRLS algorithms.

SM-WRLS Algorithm	Checking	Covariance and Solution Update	Example (flops)
Sequential GR-based	$.5m^2 + 2.5m + 13$	$2.5m^2 + 10.5m + 5$	160
Sequential Suboptimal GR-based	$(m+1) + s(.5m^2 + 1.5m + 12)$	$2.5m^2 + 10.5m + 5$	55
Parallel GR-based	$3m + 14$	$11m + 10$	68
Parallel Suboptimal GR-based	$(m+1) + s(2m + 13)$	$11m + 10$	26

Table 4.5: Parallel computational complexities (in flops per equation) for the various SM-WRLS algorithms using the implementations of Figs. 4.1 and 4.5.

Element Computed	flops per equation
$\epsilon_n^2(n+1)$	$m+1$
Coefficients of quadratic (1.26)	7
$\lambda^*(n+1)$	$5 + \sqrt{}$
$G(n+1)$ and $\hat{\theta}(n)$	$2m+1$
If the equation is accepted:	
Updating the equation	$m + 1 + \sqrt{}$
Givens rotations	$5(2m+1)$
$\kappa(n)$	4

original $\mathcal{O}(m^2)$ sequential formulations [5, 6, 20].

If the fraction of the data accepted by the SM-WRLS is r (r is typically less than 30% [7]), then the total parallel computational complexity is given by

$$(3m + 14) + r[11m + 10] \tag{4.3}$$

flops per equation.

The adaptive compact architecture of Fig. 4.5, which has slightly more complicated cells, is as efficient as the architecture of Fig. 4.1. The only difference is that the architecture can be used to rotate out (part of) an equation which was previously rotated in. Therefore, the parallel computational complexity (per equation) does not change. However, in the windowed technique, for example, there might be a need to go through updating the system *twice* for a single equation; first to rotate an equation out (if it *was* accepted) and then to rotate the new equation in (if it *is* accepted). The architecture (of Fig. 4.5) can be visualized as operating in two modes, the first mode is when it is rotating an equation into the system ($\delta = +1$) and the second mode is when it is rotating an equation out of the system ($\delta = -1$). Note that the two modes have the same parallel computational complexity with some addition operations in one mode replaced by subtraction operations in the other mode (see Fig. 4.6).

The total parallel computational complexities of the general adaptive SM-WRLS algorithms (see *Section 2.4.1*) depend on the adaptive strategy employed, the percentage of the data accepted, and the number of times the algorithm rotates out an equation from the system (whether partially or completely). Consider the *windowed* adaptation, for example, which effectively slides a window of fixed length through the data by appropriate sequencing of rotating particular equations into and out of the system. Suppose that the fraction of the data accepted (rotated in) by the windowed SM-WRLS algorithm is r and the fraction of the data removed (rotated out) from

the system is u ($u < r$), then the total parallel computational complexity is given by

$$(3m + 14) + (r + u)[11m + 10] \tag{4.4}$$

flops per equation. This expression also holds for the selective forgetting strategy, however, the graceful forgetting strategy uses the same expression with u replaced by $\mu^{-1}u$. It is important to note that the adaptive techniques typically use more data but produce better estimates.

To show the computational savings when using the "suboptimal" SM-WRLS algorithm, it is noted in *Section 2.3* that at each recursion, we only need to compute $\epsilon^2_{n-1}(n)$ and check if following condition holds (written here for the real scalar case)

$$\epsilon^2_{n-1}(n) > \gamma^{-1}(n) \ , \tag{4.5}$$

whereas in the SM-WRLS algorithm, the coefficients of the quadratic (1.26), $\epsilon^2_{n-1}(n)$, $G(n)$, and $\lambda(n)$ must be computed before making the decision. In the suboptimal case, these quantities are computed *only if* condition (4.5) is met, and then the new equation is accepted if the optimal weight is positive.

To calculate the total computational complexity for the suboptimal SM-WRLS algorithm, let us denote the fraction of the data satisfying the condition (4.5) by s ($s < r$) and the fraction of the data accepted by the SM-WRLS algorithm after passing the test (4.5) by t ($t \leq s$). Then the total parallel computational complexity for the suboptimal algorithm is given by

$$(m + 1) + s[2m + 13] + t[11m + 10] \tag{4.6}$$

flops per equation, which clearly shows significant improvement over that of the SM-WRLS algorithm (cf. (4.3)). The total parallel computational complexity for the

suboptimal windowed and selective forgetting strategies is given by

$$(m+1)+s[2m+13]+(t+u)[11m+10] \tag{4.7}$$

flops per equation, with u replaced by $\mu^{-1}u$ for the suboptimal graceful forgetting strategy.

The fourth column in Table 4.4 shows the total number of flops per equation for a typical example with $m = 10$, $r = 0.2$, and $s = t = 0.1$. It shows that the parallel architecture reduces the complexity of the algorithm by about 60%, and when the suboptimal strategy is employed, the complexity is reduced by 84%.

The performance of the SM-WRLS algorithm in terms of its adaptive behavior and tracking capability, solution quality, and fraction of data used requires further research; however, it is important to note that the gain in the computational complexity of the GR-based algorithm, when implemented on a sequential machine, is only two to three times when compared to that of the MIL-based WRLS (see Table 2.1), and five to six times when implemented on a parallel machine (see Table 4.4). It is the suboptimal technique that gives an order of magnitude (13 to 14 times) gain in the computational complexity when implemented on a parallel machine, and gives six to seven times gain when implemented on a sequential machine. Therefore, it makes more sense to use the suboptimal technique for speed advantages since the estimates are not very different from those of the SM-WRLS algorithm.

The computational complexity of the generalized complex vector case of the SM-WRLS algorithm developed in *Section 2.2* when computed on a sequential machine is discussed in *Section 2.5*. However, the parallel computational complexity of the generalized parallel GR-based algorithm depends on the architectural implementation of this algorithm which requires further research.

Chapter 5

Conclusions and Further Work

5.1 Algorithmic Developments

5.1.1 A Generalized SM-WRLS Algorithm

This research has been concerned with a class of SM algorithms for estimating the parameters of linear system or signal models in which the error sequence was pointwise "energy bounded." Specifically, it was focused on the SM-WRLS algorithm which works with bounding hyperellipsoidal regions to describe the solution sets which are a consequence of the error bounds. SM-WRLS is based on the familiar WRLS solution with the SM considerations handled through a special weighting strategy. However, the original theoretical development of this algorithm made it applicable to real scalar data. Due to the strong potential for using this powerful algorithm in virtually any signal processing problem involving parametric models, this algorithm has been extended to work with a wider range of problems. The theoretical development of a generalized SM-WRLS algorithm that can handle complex vector-input vector-output data streams has been presented. The original SM-WRLS algorithm is a special case of the generalized algorithm.

5.1.2 Suboptimal Tests for Innovation

A new strategy has been developed which can be applied to virtually any version of the SM-WRLS algorithm to improve the computational complexity. A significant reduction in computational complexity is achieved by employing a "suboptimal" test for information content in an incoming equation. The suboptimal check has been argued to be a useful determiner of the ability of incoming data to shrink the ellipsoid, but one which does not rigorously determine the existence of an optimal SM weight in the SM-WRLS sense. The main issue is to avoid the computations of an $\mathcal{O}(m^2)$ checking procedure required to check for the existence of a meaningful weight. Since most of the time these computations would result in the rejection of incoming data, a more efficient test significantly reduces the complexity of the algorithm.

5.1.3 Adaptive SM-WRLS Algorithms

It has been argued that the "unmodified" SM-WRLS algorithm has inherent adaptive capabilities in its own right. However, it is not possible to depend upon this algorithm to *reliably* behave in an adaptive manner, particularly in cases of quickly varying system dynamics. In this work, explicitly adaptive SM-WRLS algorithms have been developed. Adaptation was incorporated into SM-WRLS in a very general way by introducing a flexible mechanism by which the algorithm can forget the influence of past data. The general formulation permitted the extension of SM-WRLS to a wide range of adaptation strategies. Three different adaptation techniques have been presented and tested on models derived from real speech data. *Windowing* is a simple way to make the algorithm adaptive by effectively sliding a window of fixed length through the data by appropriate sequencing of "adding" or "removing" equations. The *Graceful Forgetting* technique removes only a fraction of all previous equations at each time. The *Selective Forgetting* technique chooses the equations to be (partially or completely) removed from the system based on certain user defined criteria.

A survey of the computational complexities of several related sequential algorithms has been presented which shows the computational savings obtained when using the SM-based algorithms compared with the conventional WRLS algorithms. The differences in the computational complexities among the various SM-based algorithms have been discussed.

5.2 Simulation Studies

The SM-WRLS algorithms have been tested on models derived from real speech data representing the words "four," "six," and "seven" using an AR(2) model for the first two words and AR(14) model for the third. The simulation results presented illustrate important points about the various methods and show that the adaptive algorithms yield accurate estimates using very few of the data and quickly adapt to fast variations in the signals dynamics. It is significant that in preliminary experiments, most of the SM-WRLS algorithms have been found to be robust in small (16-bit) wordlength environments.

5.3 Architectures and Complexity Issues

The "nonadaptive" SM-WRLS algorithm has been formulated to be run on a parallel architecture. An architecture has been developed that implements this algorithm in $\mathcal{O}(m)$ flops per equation. Then, this architecture (which uses $\mathcal{O}(m^2)$ cells) was mapped into a compact architecture in order to increase the cell utilization at the expense of using slightly more complicated cells (needed for scheduling). The compact architecture uses only $\mathcal{O}(m)$ cells which is clearly advantageous with respect to the $\mathcal{O}(m^2)$ cells architecture. The cells used by the compact architecture were upgraded to implement the adaptive strategies. It was also noted that the same architectures could be used to implement the "suboptimal" strategy.

Finally, a detailed analysis of the computational complexity issues was carried out which clearly shows the significant computational savings when implementing the SM-WRLS algorithm using the parallel architectures. The computational complexities of the adaptive SM-WRLS algorithms depend on the adaptive strategy employed, the percentage of the data accepted (which is typically more than that of the nonadaptive algorithms), and the number of times the algorithm rotates out an equation from the system. The analysis also shows that the suboptimal strategy (whether implemented sequentially or using the parallel architectures) provides significant improvements in the computational efficiencies of the various algorithms.

5.4 Further Work

This research has been concerned with problems in which the error sequence was pointwise energy bounded (see constraint (1.1)). It is of great interest to study other classes of SM algorithms that use different constraints. Other interesting variations involve stability constraints [30], and other noise parameter bounds [31, 32].

The optimization criterion used in this research was based on minimizing the "volume ratio" of the ellipsoids at n and $n-1$ (see (1.25)). It might be useful to use different optimization criteria in order to minimize the ellipsoid volume. For example, it may be possible to define a strategy that efficiently minimizes the longest axis of the ellipsoid.

The adaptive SM-WRLS algorithm presented in *Section 2.4* works with a very flexible "forgetting" scheme. Three different techniques were presented and tested, however, it is possible to define (and test) various other adaptation strategies.

It is true that the SM-WRLS algorithm was tested on models derived from speech data (since they are available in the Speech Processing Laboratory), however, this algorithm has the potential for application to a wide range of problems. It is of great

interest and significance to study the performance of this algorithm when applied to beamforming, neural networks, and other important applications.

Finally, the computational complexity of the GR-based SM-WRLS algorithm can be further improved by incorporating more parallelism (in hardware) within the cells of the architectures of *Chapter 4*. Also, a generalized architecture is needed to efficiently implement the complex vector case.

Souheil F. Odeh

East Lansing, Michigan

May, 1990

Appendix

Appendix A

The Relationship between $\mathbf{C}_x(n)$ and $\lambda(n)$

The theoretical developments in this appendix are for the real scalar case. The generalization to the complex vector case is straightforward.

It was noted in *Section 1.2.3.1* that the system of equations (1.8) (or (2.9) for the generalized case) can be reduced to an upper triangular system (1.27) by applying a sequence of orthogonal operators (GR's). Suppose that a new equation is accepted with an optimal weight $\lambda(n)$, it can be rotated into the upper triangular system by inserting it in the $m+1^{st}$ row, i.e.,

$$\begin{bmatrix} \mathbf{T}(n-1) \\ \hline \sqrt{\lambda(n)}\boldsymbol{x}^T(n) \rightarrow \end{bmatrix} \boldsymbol{\Theta}_0 = \begin{bmatrix} \boldsymbol{d}_1(n-1) \\ \hline \sqrt{\lambda(n)}y(n) \end{bmatrix} \tag{A.1}$$

and applying another sequence of m GR's leaving the system in the form

$$\begin{bmatrix} \mathbf{T}(n) \\ \hline \mathbf{0}_{1\times m} \end{bmatrix} \boldsymbol{\Theta}_0 = \begin{bmatrix} \boldsymbol{d}_1(n) \\ \hline d_2(n) \end{bmatrix} \tag{A.2}$$

where the matrix $\mathbf{T}(n)$ is an $m \times m$ upper triangular Cholesky factor [39] of $\mathbf{C}_x(n)$

(see (1.30)). This sequence of GR's is given by

$$\mathbf{Q}(n) = \mathbf{J}_m \mathbf{J}_{m-1} \cdots \mathbf{J}_2 \mathbf{J}_1 \tag{A.3}$$

where $\mathbf{J}_i$ denotes an $(m+1) \times (m+1)$ orthogonal matrix that annihilates element i of the $m+1^{st}$ row of (A.1).

It can be shown that $\mathbf{Q}(n)$ takes the form

$$\begin{bmatrix}
c_1 & 0 & 0 & \cdots & 0 & s_1 \\
-s_2 s_1 & c_2 & 0 & \cdots & 0 & s_2 c_1 \\
-s_3 c_2 s_1 & -s_3 s_2 & c_3 & \cdots & 0 & s_3 c_2 c_1 \\
-s_4 c_3 c_2 s_1 & -s_4 c_3 s_2 & -s_4 s_3 & \cdots & 0 & s_4 c_3 c_2 c_1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
-s_m c_{m-1} \cdots c_2 s_1 & -s_m c_{m-1} \cdots c_3 s_2 & -s_m c_{m-1} \cdots c_4 s_3 & \cdots & c_m & s_m c_{m-1} \cdots c_1 \\
-c_m c_{m-1} \cdots c_2 s_1 & -c_m c_{m-1} \cdots c_3 s_2 & -c_m c_{m-1} \cdots c_4 s_3 & \cdots & s_m & c_m c_{m-1} \cdots c_1
\end{bmatrix}$$

in which c_i (s_i) is the cosine (sine) term associated with the i^{th} rotation. This form of $\mathbf{Q}(n)$ simplifies the generation of the matrix $\mathbf{T}(n)$ from $\mathbf{T}(n-1)$ and is useful in finding $\det \mathbf{C}_x(n)$ below. Since the matrix $\mathbf{T}(n)$ is upper triangular, then

$$\det \mathbf{T}(n) = \prod_{i=1}^{m} t_{ii}(n) \tag{A.4}$$

and

$$\det \mathbf{C}_x(n) = [\det \mathbf{T}(n)]^2 = \prod_{i=1}^{m} t_{ii}^2(n) \tag{A.5}$$

in which t_{ii} denotes the i^{th} diagonal element of the matrix $\mathbf{T}(n)$.

After some (tedious) algebraic manipulation, it follows that:

Case $m = 1$:

$$\begin{aligned}\det \mathbf{C}_x(n) &= t_{11}^2(n) \\ &= t_{11}^2(n-1) + \lambda(n)x_1^2(n)\end{aligned}$$

Case $m = 2$:

$$\begin{aligned}\det \mathbf{C}_x(n) &= t_{22}^2(n)t_{11}^2(n) \\ &= t_{22}^2(n-1)t_{11}^2(n) + \lambda(n)\left[t_{11}(n-1)x_2(n) - t_{12}(n-1)x_1(n)\right]^2\end{aligned}$$

Case $m = 3$:

$$\begin{aligned}\det \mathbf{C}_x(n) &= t_{33}^2(n)t_{22}^2(n)t_{11}^2(n) \\ &= t_{33}^2(n-1)t_{22}^2(n)t_{11}^2(n) + \\ &\quad \lambda(n)\left\{t_{22}(n-1)\left[t_{11}(n-1)x_3(n) - t_{13}(n-1)x_1(n)\right] - \right. \\ &\qquad \left. t_{23}(n-1)\left[t_{11}(n-1)x_2(n) - t_{12}(n-1)x_1(n)\right]\right\}^2\end{aligned}$$

Case $m = 4$:

$$\begin{aligned}\det \mathbf{C}_x(n) &= t_{44}^2(n)t_{33}^2(n)t_{22}^2(n)t_{11}^2(n) \\ &= t_{44}^2(n-1)t_{33}^2(n)t_{22}^2(n)t_{11}^2(n) + \\ &\quad \lambda(n)\left\{t_{33}(n-1)\left\{t_{22}(n-1)\left[t_{11}(n-1)x_4(n) - t_{14}(n-1)x_1(n)\right] - \right.\right. \\ &\qquad \left.\left. t_{24}(n-1)\left[t_{11}(n-1)x_2(n) - t_{12}(n-1)x_1(n)\right]\right\} - \right. \\ &\qquad \left. t_{34}(n-1)\left\{t_{22}(n-1)\left[t_{11}(n-1)x_3(n) - t_{13}(n-1)x_1(n)\right] - \right.\right. \\ &\qquad \left.\left. t_{23}(n-1)\left[t_{11}(n-1)x_2(n) - t_{12}(n-1)x_1(n)\right]\right\}\right\}^2\end{aligned}$$

and so on. Therefore, $\det \mathbf{C}_x(n)$ is a monotonically increasing function of $\lambda(n)$. □

Bibliography

Bibliography

[1] S.H. Mo and J.P. Norton, "Fast and robust algorithm to compute exact polytope parameter bounds," *Mathematics and Computers in Simulation*, to appear in 1990.

[2] J.P. Norton and S.H. Mo, "Parameter bounding for time-varying systems," *Mathematics and Computers in Simulation*, to appear in 1990.

[3] A.K. Rao, Y.F. Huang, and S. Dasgupta, "ARMA parameter estimation using a novel recursive estimation algorithm with selective updating," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 38, pp. 447-457, March 1990.

[4] S.F. Odeh and J.R. Deller, Jr., "A systolic algorithm for adaptive set membership identification," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '90*, Albuquerque, NM, vol. 5, pp. 2419-2422, April 1990.

[5] J.R. Deller, Jr., "A 'systolic array' formulation of the optimal bounding ellipsoid algorithm," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 37, pp. 1432-1436, Sept. 1989.

[6] J.R. Deller, Jr., "Set membership identification in digital signal processing," *IEEE ASSP Magazine*, vol. 6, no. 4, pp. 4-20, Oct. 1989.

[7] J.R. Deller, Jr. and T.C. Luk, "Linear prediction analysis of speech based on set-membership theory," *Computer Speech and Language*, vol. 3, no. 4, pp. 301-327, Oct. 1989.

[8] A.K. Rao and Y.F. Huang, "Analysis of finite precision effects on an OBE algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '89*, Glasgow, vol. 2, pp. 853-856, May 1989.

[9] J.R. Deller, Jr. and S.F. Odeh, "Implementing the optimal bounding ellipsoid algorithm on a fast processor," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '89*, Glasgow, vol. 2, pp. 1067-1070, May 1989.

[10] E. Walter and H. Piet-Lahanier, "Estimation of parameter bounds from bounded-error data: A survey," *Proc. 12th IMACS World Congress on Scientific Computation*, Paris, pp. 467-472, July 1988.

[11] E. Walter and H. Piet-Lahanier, "OMNE versus least squares for estimating parameters of a biological model from short data records," *Proc. 12th IMACS World Congress on Scientific Computation*, Paris, pp. 85-87, July 1988.

[12] H. Piet-Lahanier and E. Walter, "Practical implementation of an exact and recursive algorithm for characterizing likelihood sets," *Proc. 12th IMACS World Congress on Scientific Computation*, Paris, July 1988.

[13] R. Tempo, "Robust estimation and filtering in the presence of bounded noise," *IEEE Trans. Automatic Control*, vol. AC-33, pp. 864-867, 1988.

[14] S. Dasgupta and Y.F. Huang, "Asymptotically convergent modified recursive least squares with data-dependent updating and forgetting factor for systems with bounded noise," *IEEE Trans. Information Theory*, vol. IT-33, pp. 383-392, 1987.

[15] J.P. Norton, "Identification of parameter bounds of ARMAX models from records with bounded noise," *Int. J. Control*, vol. 45, pp. 375-390, 1987.

[16] J.P. Norton, "Identification and application of bounded-parameter models," *Automatica*, vol. 23, pp. 497-507, 1987.

[17] E. Walter and H. Piet-Lahanier, "Exact and recursive description of the feasible parameter set for bounded error models," *Proc. 26th IEEE Conf. Decision and Control*, Los Angeles, pp. 1921-1922, 1987.

[18] J.R. Deller, Jr. and T.C. Luk, "Set-membership theory applied to linear prediction analysis of speech," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '87*, Dallas, vol. 2, pp. 653-656, 1987.

[19] Y.F. Huang and A.K. Rao, "Application of a recursive estimation algorithm with information-dependent updating to ARMAX models and ARMA models with unknown inputs," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '87*, Dallas, vol. 2, pp. 1007-1010, 1987.

[20] Y.F. Huang, "A recursive estimation algorithm using selective updating for spectral analysis and adaptive signal processing," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 1331-1334, 1986.

[21] V. Broman and M.J. Shensa, "Polytopes, a novel approach to tracking," *Proc. 25th IEEE Conf. Decision and Control*, Athens, pp. 1749-1752, 1986.

[22] M.L. Feron and J.R. Deller, Jr., "'Membership set' identification applied to the impulse train excited AR model" (abstract), *Final Program: SIAM 1983 National Mtg.*, Denver, p. 30, 1983.

[23] E. Fogel and Y.F. Huang, "On the value of information in system identification — Bounded noise case," *Automatica*, vol. 18, pp. 229-238, 1982.

[24] M. Milanese and G. Belforte, "Estimation theory and uncertainty intervals evaluation in presence of unknown but bounded errors: Linear families of models and estimators," *IEEE Trans. Automatic Control*, vol. AC-27, pp. 408-414, 1982.

[25] E. Fogel, "System identification via membership set constraints with energy constrained noise," *IEEE Trans. Automatic Control*, vol. AC-24, pp. 752-758, 1979.

[26] B.R. Barmish and J. Sankaran, "The propagation of parametric uncertainty via polytopes," *IEEE Trans. Automatic Control*, vol. AC-24, pp. 346-349, 1979.

[27] F.C. Schweppe, "Recursive state estimation: Unknown but bounded errors and system inputs," *IEEE Trans. Automatic Control*, vol. AC-13, pp. 22-28, 1968.

[28] H.S. Witsenhausen, "Sets of possible states of linear systems given perturbed observations," *IEEE Trans. Automatic Control*, vol. AC-13, pp. 556-568, 1968.

[29] D.P. Bertsekas and I.B. Rhodes, "Recursive state estimation for a set-membership description of uncertainty," *IEEE Trans. Automatic Control*, vol. AC-16, pp. 117-128, 1971.

[30] P.L. Combettes and H.J. Trussell, "Stability of the linear prediction filter: A set theoretic approach," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '88*, New York, vol. 4, pp. 2288-2291, 1988.

[31] P.L. Combettes and H.J. Trussell, "General order moments in set-theoretic estimation," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '89*, Glasgow, vol. 4, pp. 2101-2104, May 1989.

[32] P.L. Combettes, *Set Theoretic Estimation in Digital Signal Processing* (Ph.D. Dissertation), North Carolina State University, Raleigh, 1989.

[33] S.M. Veres and J.P. Norton, "Structure identification of parameter-bounding models by use of noise-structure bounds," *Int. J. Control*, vol. 50, pp. 639-649, 1989.

[34] D. Graupe, *Time Series Analysis, Identification, and Adaptive Systems*, Malabar, FL: Krieger, Ch. 5, 1989.

[35] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, Cambridge, MA: MIT Press, pp. 14-15 & Sect. 2.2.1., 1983.

[36] S.Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[37] B.D. Van Veen and K.M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE ASSP Magazine*, pp. 4-24, April 1988.

[38] T. Kohonen, *Self-Organization and Associative Memory* (2^{nd} ed.), Berlin, Heidelberg: Springer-Verlag, Chs. 6 & 9, 1988.

[39] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Baltimore, MD: Johns-Hopkins Univ. Press, Chs. 5 & 6, 1983.

[40] J.R. Deller, Jr. and G.P. Picaché, "Advantages of a Givens rotation approach to temporally recursive linear prediction analysis of speech," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 37, pp. 429-431, March 1989.

[41] T.C. Luk and J.R. Deller, Jr., "A nonclassical WRLS algorithm," *Proc. 23rd Ann. Allerton Conf.*, pp. 732-741, 1985.

[42] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays," *Proc. Soc. Photo-Optical Instrumentation Engineers: Real-Time Signal Processing IV*, vol. 298, pp. 19-26, 1981.

[43] J.G. McWhirter, "Recursive least squares minimization using a systolic array," *Proc. Soc. Photo-Optical Instrumentation Engineers: Real-Time Signal Processing VI*, vol. 431, pp. 105-112, 1983.

[44] J.R. Deller, Jr. and D. Hsu, "An alternative adaptive sequential regression algorithm and its application to the recognition of cerebral palsy speech," *IEEE Trans. Circ. and Syst.*, vol. CAS-34, pp. 782-787, July 1987.

[45] J.R. Deller, Jr., Unpublished research notes, Michigan State University, East Lansing, 1990.

[46] M.L. Feron, "Membership set system identification with pulse train input," M.S. Thesis, Illinois Institute of Technology, Chicago, 1982.

[47] J. Makhoul, "Linear prediction: A tutorial review," *Proc. of the IEEE*, vol. 63, pp. 561-580, 1975.

[48] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge, MA: Cambridge Univ. Press, pp. 209-213, 1988.

[49] D.F. Marshall and W.K. Jenkins, "A fast quasi-Newton adaptive filtering algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '88*, New York, NY, vol. 3, pp. 1377-1380, 1988.

[50] J.M. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. Circ. and Syst.*, vol. CAS-34, pp. 821-833, July 1987.

[51] S. Ljung and L. Ljung, "Error propagation properties of recursive least squares adaptation algorithms," *Automatica*, vol. 21, pp. 157-167, 1985.

[52] J.M. Cioffi and T. Kailath, "Fast, recursive least squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-32, pp. 304-337, April 1984.

[53] H.T. Kung and C. Leiserson, "Algorithms for VLSI processor arrays," Rpt. No. MU-CS-79-103, Dept. of Computer Sci., Carnegie-Mellon Univ., 1978 (Reprinted in reference [54]).

[54] C. Mead and L. Conway, *Introduction to VLSI Systems*, Reading, MA: Addison-Wesley, Section 8.3, 1980.

[55] W.M. Gentleman, "Least squares computations by Givens transformations without square roots," *J. Inst. of Math. and its Appl.*, Vol. 12, pp. 329-336, 1973.

[56] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall, Ch. 10, 1986.