PROJECTION METHODS FOR IMPROVED PERFORMANCE IN ADAPTIVE SYSTEMS

BY

ROBERT ATMARAM SONI

B.S., University of Cincinnati, 1992
M.S., University of Illinois at Urbana-Champaign, 1995

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1998

Urbana, Illinois

PROJECTION METHODS FOR IMPROVED PERFORMANCE IN ADAPTIVE SYSTEMS

Robert Atmaram Soni, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1999
W. Kenneth Jenkins, Advisor

The number and variety of adaptive algorithms available can overwhelm the adaptive system designer. Some attempts to date have been made to unify the various adaptive algorithms, but an underlying framework has been missing. The dissertation develops this framework. The development of this framework has led to the development of the new algorithms. The framework for these algorithms relates adaptive algorithms to *row-projection* methods for solving linear systems. This classical method of optimization due primarily to Kaczmarz '37 and Cimmino '39, has been periodically reinvented to solve a number of very different types of linear systems. This broad class of iterative methods does not require that the data available be well-conditioned and symmetric (a condition of many large iterative linear system solvers). They also allow a significant degree of parallelism and have been used with great success in large system solvers implemented in supercomputers.

The *row-projection* view of adaptive filtering helps to relate a number of different adaptive algorithms. In addition, this framework helps motivate the development of "new" adaptive algorithms which are scalable in performance and structure. This allows system designers the flexibility to design a system and change a parameter to improve performance and only pay a marginal cost in increased computation.

These methods encompass recursive least squares (RLS) techniques for adaptive filtering at one extreme choice of parameters and least mean square (LMS) techniques at the other extreme. RLS techniques offer rapid convergence rate and high computational complexity. These new algorithms function by reusing existing data. There are optimal methods of reusing this data, and this is discussed in detail in this thesis.

Further, there are parallel and serial methods for reusing the data. This allows the system designer even greater flexibility in the implementation of the adaptive system. These methods may be further enhanced by the use of acceleration techniques originally developed in a *row-projection* framework. Use of all these techniques is explored

in a number of applications including system identification, channel equalization, two-dimensional adaptive processing, adaptive interference suppression, and adaptive fault tolerance. In each application, significant advantages were discovered to using the projection algorithms and framework.

DEDICATION

To the Soni, Chandak, and Adams families.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. W. Kenneth Jenkins, for his tireless enthusiasm, understanding, patience, good humor, and support both financial and emotional. Only with his guidance, could I have achieved the grand goal of completing this dissertation.

I would also like to thank my co-advisor, Dr. Kyle Gallivan, for making time in the midst of his busy schedule and relocation to Florida. Through his insight, this dissertation became a reality. Though we often struggled with terminology, the relationship has been extremely fruitful.

I would like to thank Dr. David Munson Jr. for serving on my committee and helping me secure a teaching position when I arrived at the University of Illinois. With his timely assistance, I was able to attend this great university and collaborate and work with so many brilliant people. I am eternally grateful for his help.

I would also like to thank Dr. Douglas Jones for serving on my committee and providing me with useful advice on research and career development. Definitely, he is one of the real assets to students in the group in many different ways.

I would also like to thank Dr. Naresh Shanbhag for serving on my committee and sharing his helpful wisdom.

I would like to thank Chris Schmitz, my officemate, for providing the fuel for the fire of many “technical” discussions, and for being a dear friend for so many years. To my longtime roommate, Jonathan Locker, I owe a debt of eternal gratitude, and a blessing for his motivational skill.

I would also like to thank the numerous group mates and lab mates who are so numerous to mention that I will probably neglect to mention a few. They are: Swaroop Appadwedula, Samit Basu, Hyeokho Choi, Christophe Couvreur, Rich Czerwinksi, Dan Franzen, Trenton Haines, G. Harikumar, Andy Hull, Mike Kramer, Brian Krongold, Henry Kwok, In-Seop and Jungah Lee, Xiaohui Li, Robert Liu, Jeff Meunier, Leiming Qian, Yuhui Ren, Akbar Sayeed, Bernard Schnaufer, Jeff Strait, Raman Venkatarami, and Gilbert Yu. Thanks for helping to motivate me to finish this dissertation, and for providing technical assistance, and great friendship.

Finally, for providing the kind of support that only family members can offer, often in times that were extremely difficult, I would like to thank my "newly-wedded" wife, Rachita; my Dad, Dr. Atmaram Harilal Soni, and my Mom, Mrs. Ila A. Soni; and my father-in-law, Mr. Shrinarayan Chandak, my mother-in-law, Mrs. Asha Chandak; my sisters, Bina and Lisa; and my brother-in-law Matthew, and my sister-in-law, Rachana.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The adaptive approach to identifying unknown systems is a viable approach to solving a large number of telecommunications, system identification, and image processing problems. Classically, a system which identifies these unknown systems or their inverses must have some knowledge about the nature of the system. Is the system linear? Can it be modeled using a finite impulse response (FIR) filter? If these questions can be answered affirmatively, then a broad class of adaptive approaches may be applied with some success to solving these problems.

These approaches use linear FIR filters to model the systems in question. There are several approaches to using these FIR filters to solve this problem. Initially, this dissertation will discuss two basic approaches, the least mean square (LMS) algorithm, and the recursive least squares (RLS) algorithm. In terms of complexity and performance, the affine projection methods can achieve LMS performance and complexity with one choice of parameters, and RLS-like performance and complexity with another choice. These algorithms will be related to a class of data-reusing algorithms which can be viewed as low-rank or non-optimal approximations of the affine projection algorithms.

Using a broader class of optimization strategies known as *row projection*, a comprehensive framework for the explanation of the operation of the adaptive filtering algorithms can be developed. This framework is deterministic in nature. It allows the system designers to view the operation of stochastic gradient algorithms such as the LMS algorithm and quasi-Newton algorithm approach to adaptive filtering in a graphical or deterministic sense.

Associated with adaptive filtering algorithms, there are many key parameters that are considered to determine the overall effectiveness of the algorithm. Algorithms are chosen for applications based upon how these algorithms perform in terms of the three following categories: convergence rate, how fast the algorithm converges to the optimal solution; misadjustment, how well the adaptive filter coefficients match the optimal solution at convergence; computational complexity, how many multiplications are associated with the update of a single set of the adaptive filter coefficients; implementation complexity, how difficult the algorithm is to implement in fixed point as well as floating point.

Beginning with the LMS, RLS, affine projection, and data-reusing algorithms, this dissertation will examine how a number of different classes of adaptive algorithms are derived and developed. The key parameters associated with each algorithm will be discussed in greater detail, and the key differences between each algorithm will be illustrated and discussed. These algorithms will give motivation for the the "new" framework which follows.

1.1 LMS Algorithms

Traditional adaptive filtering applications have utilized the least mean square (LMS) approach with great success. The simplicity of implementation and low computational complexity have eased its use in numerous applications such as echo/noise cancellation, communication channel equalization, system/parameter identification, and signal enhancement and prediction.

The goal of the LMS algorithm is to minimize an error signal $e(n)$ in a mean square sense. In optimization terms, the performance function minimized is

$$J(n) = E[|e(n)|^2]. \tag{1.1}$$

For the system identification setting of adaptive filtering shown in Figure 1.1, $e(n) = d(n) - W^T(n)X(n)$, where the signal $d(n)$ represents the desired signal. The vector $W(n)$ represents the adaptive filter coefficients which change to minimize the least mean square error. The vector $X(n)$ represents a vector of the input signal which can be expressed as

$$X(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^T. \tag{1.2}$$

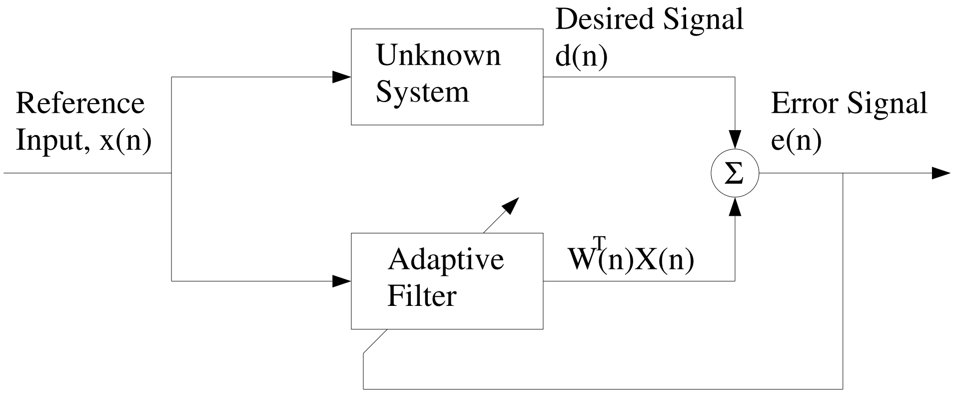


Figure 1.1 Adaptive Filtering Configuration, System Identification

The LMS algorithm is a stochastic steepest descent algorithm. The goal of the algorithm is to minimize $J(n)$ iteratively by approximating its gradient. As a stochastic gradient algorithm, its convergence behavior is highly sensitive to the coloring of the input signal $x(n)$ [1]. A larger eigenvalue ratio for the autocorrelation matrix $\mathbf{R}_{xx}(n)$ implies a slower convergence rate for the algorithm.

1.2 RLS Algorithms

The recursive least squares (RLS) algorithm takes a different approach to optimizing the adaptive filter coefficients. Instead of minimizing the performance function in Equation (1.1), the overall error is minimized in a weighted least squares sense. The performance function for a exponential window weighting scheme is

$$J(n) = \sum_{i=0}^{n} \lambda^{n-i} |e(i)|^2. \tag{1.3}$$

The convergence behavior of the RLS algorithm developed is highly impervious to the effects of colored or correlated inputs. Convergence to the optimal least squares solution occurs typically within $2N$ iterations, where N is the order of the adaptive filter. Unfortunately, this algorithm exhibits stability problems and numerically aberrant behavior. In addition, the algorithm has a much higher computational complexity than the LMS algorithm, requiring $O(N^2)$ multiplications per iteration versus $O(N)$ computations for the LMS algorithm. Slock, Kailath, Cioffi and others [2–4] have attempted to solve these issues, and have developed algorithms (termed fast transversal algorithms) which reduced the complexity and addressed the stability issues. However, they can require on the order of $16N$ multiplications per iteration.

1.3 Affine Projection Algorithms

Lying somewhere between the LMS algorithms and RLS algorithms are a class of algorithms which allow the system designer to choose complexity and performance by changing parameters without changing algorithms. The system designer may effectively trade increased computational complexity for improved convergence rate performance. In many scenarios, it is possible to achieve an improved convergence rate with only a marginal increase in computation.

In the body of this text, this class of algorithms will be described as *projection* algorithms. The *projection* algorithms iteratively project in an orthogonal way from one allowable set of solutions to the next updated set. They are iterative methods which solve a linear system by a method of successive approximation. Convergence is obtained by constraining the method of allowable projection.

Several of these algorithms are popularly referred to in the literature as "affine projection" adaptive filtering algorithms [5–9]. These algorithms add additional constraints to the basic constraint of minimizing the instantaneous error of the filter. Two basic approaches are surveyed here. The first employs the conjugate gradient algorithm to solve a constrained low-rank least squares problem. The second solves the same least squares problem using a direct rather than an iterative method. The second method developed by Slock has been redeveloped and reinvented by several different researchers. Some of these methods are merely "fast" methods or reduced computational complexity approaches to solving the same adaptive filtering problem. A brief summary of these methods is presented in Section 1.3.4.

1.3.1 Boray-Srinath algorithm

The Boray-Srinath conjugate gradient adaptive algorithm attempts to minimize repeatedly the performance index

$$J(W(n)) = \frac{1}{L} \sum_{i=n-L+1}^{n} (d(i) - W(n)^H X(i))^2, \tag{1.4}$$

where ($L \leq N$), $d(n)$ represents the desired signal, and $W(n)$ and $X(i)$ represent vectors of the adaptive filter coefficients and the input signal, respectively, which can be defined as

$$\begin{aligned} W(n) &= [W_0(n), W_1(n) \ldots, W_{N-2}(n), W_{N-1}(n)]^T \\ X(i) &= [x(i), x(i-1) \ldots, x(i-N+2), x(i-N+1)]^T. \end{aligned} \tag{1.5}$$

This index may be written in more compact vector notation as

$$J(W(n)) = \frac{1}{L} ||D(n) - \mathbf{X}(n)^H W(n)||^2, \tag{1.6}$$

where the vector $D(n)$ and the matrix $\mathbf{X}(n)$ are defined as

$$D(n) = [d(n), d(n-1), \ldots, d(n-L+1)]^T \tag{1.7}$$

$$\mathbf{X}(n) = [x(n), x(n-1), \ldots, x(n-L+1)]^T. \tag{1.8}$$

The parameter L determines the convergence rate performance of the algorithm developed by the optimization of Equation (1.4) or Equation (1.6). Increasing L improves the convergence rate performance, but also increases the computational complexity of the algorithm, as will be shown later.

Taking the gradient of the performance function given in Equations (1.4) or (1.6) yields the following system of normal equations:

$$\mathbf{X}(n)D(n) = \mathbf{X}(n)\mathbf{X}^H(n)W(n). \tag{1.9}$$

This system of equations is not full rank because the matrix $\mathbf{X}(n)$ has at most rank L, where $L < N$. Therefore, the matrix $\mathbf{X}(n)\mathbf{X}^H(n)$ cannot be inverted. Furthermore, this system has infinitely many solutions.

The Boray-Srinath adaptive algorithm repeatedly applies the conjugate-gradient algorithm to this system of equations. It restarts the conjugate-gradient algorithm every L steps with new data while using the previous result of the conjugate gradient algorithm as an initial condition. The Boray-Srinath adaptive algorithm can be summarized as

- **Step 1**. Initialization ($n = 1$):

$$W(0) = [0, \ldots, 0]^T .$$

- **Step 2**. Form normal equations:

$$\mathbf{X}(n)\mathbf{X}^H(n)W(n) = \mathbf{X}(n)D(n).$$

- **Step 3**. Iterate the conjugate gradient algorithm L steps on the following system:

$$\begin{aligned} x_0 &= W(n-1) \\ \mathbf{Q} &= \mathbf{X}(n)\mathbf{X}^H(n) \\ b &= \mathbf{X}(n)D(n). \end{aligned}$$

- **Step 4**. $W(n) = x_L$ and ($n = n + 1$) Go to **Step 2**.

Using Case 3 of Theorem 3 presented in Appendix A, Steps 3 and 4 of the Boray Srinath algorithm can be expressed in closed form to give an exact update expression for the adaptive filter coefficients $W(n)$:

$$W(n) = \mathbf{P}_{N(\mathbf{Q})}W(n-1) + \mathbf{Q}^{\dagger}b, \tag{1.10}$$

where $\mathbf{P}_{N(\mathbf{Q})}$ is the projection matrix for the null space of the matrix $\mathbf{Q}$ and $\mathbf{Q}^{\dagger}$ is the Moore-Penrose pseudo-inverse of the matrix $\mathbf{Q}$ (see Appendix A). Specifically, the vector $W(n)$ is the sum of the null-space residue of the previous coefficient vector, and the minimum norm solution to the normal equations formed in Step 3 (also given in Equation (1.9)).

If the conjugate gradient algorithm were not initialized with the old set of coefficient vectors $W(n-1)$ the algorithm would compute the minimum norm solution to the equations defined in Step 3 of the algorithm above and in Equation (1.9). At every n^{th} iteration, a new set of normal equations is defined. These normal equations are significantly related. They differ by only one row or set of equations. The solution to the previous set of normal equations $W(n-1)$ is very close to the solution of the normal equations associated with the n^{th} iteration. It may make sense to constrain the algorithm so that the difference (or distance in a "norm" sense) between successive iterations of the algorithm is minimized. Constraining the algorithm so that this modified minimum norm criterion is solved leads to the update expression defined by Slock [7, 8], as will be shown in subsequent sections.

1.3.2 Slock's affine projection algorithms

Slock [7, 8] proposed a new algorithm based upon the minimization of the following performance function:

$$J(W(n)) = ||D(n) - \mathbf{X}^H(n)W(n)||^2_{\mathbf{S_n}^{-1}} + ||W(n) - W(n-1)||^2_{\mathbf{T_n}^{-1}}, \qquad (1.11)$$

where $||x||^2_{\mathbf{T}} = x^H\mathbf{T}x$ is the weighted norm, and the matrices $\mathbf{S_n}$ and $\mathbf{T_n}$ are symmetric and positive definite. Taking the gradient of the expression in Equation (1.11) with respect to $W(n)$ and assuming that $W(n-1)$ is not a function of $W(n)$ yields the following:

$$\nabla J(W(n)) = \mathbf{X}(n)\mathbf{S_n}^{-1}[\mathbf{X}^H(n)W(n) - D(n)] + \mathbf{T_n}^{-1}[W(n) - W(n-1)]. \qquad (1.12)$$

Setting the gradient equal to zero, and solving for $W(n)$ yields

$$W(n) = [\mathbf{X}(n)\mathbf{S_n}^{-1}\mathbf{X^H}(n) + \mathbf{T_n}^{-1}]^{-1} \times (\mathbf{X}(n)\mathbf{S_n}^{-1}D(n) + \mathbf{T_n}^{-1}W(n-1)). \qquad (1.13)$$

Using the matrix inversion lemma

$$\begin{aligned}\mathbf{A} &= \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H \\ \mathbf{A}^{-1} &= \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^H\mathbf{B}^H \end{aligned} \tag{1.14}$$

on the first term of Equation (1.13) yields

$$[\mathbf{X}(n)\mathbf{S_n}^{-1}\mathbf{X}^H(n) + \mathbf{T_n}^{-1}]^{-1} = \mathbf{T_n} - \mathbf{T_n}\mathbf{X}(n)(\mathbf{S_n} + \mathbf{X}^H(n)\mathbf{T_n}\mathbf{X}(n))^{-1}\mathbf{X}^H(n)\mathbf{T_n}. \tag{1.15}$$

Simplifying the expression for the adaptive filter coefficients by applying Equation (1.15) to Equation (1.13) yields

$$W(n) = W(n-1) + \mathbf{T_n}\mathbf{X}(n)[\mathbf{S_n} + \mathbf{X}^H(n)\mathbf{T_n}\mathbf{X}(n)]^{-1}(D(n) - \mathbf{X}^H(n)W(n-1)). \tag{1.16}$$

For the class of algorithms which Slock calls "underdetermined sliding window covariance" algorithms, he chooses $\mathbf{T_n} = \mathbf{I}$ and $\mathbf{S_n} = \mu_n\mathbf{X}^H(n)\mathbf{X}(n)$. The adaptive filter coefficient update expression then becomes

$$W(n) = W(n-1) + \frac{1}{1+\mu_n}\mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}(D(n) - \mathbf{X}^H(n)W(n-1)). \tag{1.17}$$

This expression can be rearranged into the following form:

$$W(n) = [\mathbf{I} - \frac{1}{1+\mu_n}\mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}\mathbf{X}^H(n)]W(n-1) + \frac{1}{1+\mu_n}\mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}D(n). \tag{1.18}$$

In order to compare this algorithm to the Boray-Srinath approach, two lemmas are needed which relate quantities in Equation (1.18) to the system of equations given in Equation (1.9).

Lemma 1.1 *Given a system of equations* $\mathbf{X}(n)\mathbf{X}^H(n)W(n) = \mathbf{X}(n)D(n)$*, where* $\mathbf{X}(n)$ *has rank* M *and dimensions* $M \times N$*, the minimum-norm (MN) solution to these equations is given by*

$$W_{MN}(n) = \mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}D(n), \tag{1.19}$$

where W_{MN} *represents the minimum-norm solution.*

Proof: To prove this result, the vector $W_{MN}(n)$ can easily be shown to be a solution to the system of equations $\mathbf{X}(n)\mathbf{X}^H(n)W(n) = \mathbf{X}(n)D(n)$. To prove the uniqueness of this solution, and that it is in fact the minimum-norm least-squares solution to Equation (1.9), a result from the *minimum-norm least-squares* theorem presented in Arun [10] is used.

Theorem 1.1 *Among the vectors x that minimize $||\mathbf{A}x - b||$, the one with the smallest norm is the only one in $R(\mathbf{A}^H)$ and is unique. In addition, it is the only vector x that satisfies*

$$\begin{aligned} x &= \mathbf{A}^H z \\ \mathbf{A}^H\mathbf{A}\mathbf{A}^H z &= \mathbf{A}^H b. \end{aligned} \tag{1.20}$$

Proof: Given in Arun [10].□.

The vector $W_{MN}(n)$ satisfies Equation (1.20) by simple substitution. This fact is shown here:

$$\mathbf{X}(n)\mathbf{X}^H(n)\mathbf{X}(n)\mathbf{X}^H(n)W_{MN}(n) = \mathbf{X}(n)\mathbf{X}^H(n)\mathbf{X}(n)D(n). \tag{1.21}$$

The vector $W_{MN}(n)$ can be shown to lie in $R(\mathbf{X}(n)\mathbf{X}^H(n))$ by using a part of the corollary to the *fundamental theorem of linear algebra* given in Arun [10].

Corollary 1.1

$$\begin{aligned} R(\mathbf{A}\mathbf{A}^H) &= R(\mathbf{A}) \\ R(\mathbf{A}^H\mathbf{A}) &= R(\mathbf{A}^H) \\ N(\mathbf{A}\mathbf{A}^H) &= N(\mathbf{A}^H) \\ N(\mathbf{A}^H\mathbf{A}) &= N(\mathbf{A}) \end{aligned} \tag{1.22}$$

Clearly, $W_{MN}(n) \in R(\mathbf{X}(n))$. Applying the first part of the corollary, $W_{MN}(n) \in R(\mathbf{X}(n)\mathbf{X}^H(n))$. Hence, the update equation uses the minimum-norm solution as a component of the update expression.

Reusing the corollary, the projection matrix for the range space of $\mathbf{X}(n)\mathbf{X}^H(n)$ can be determined. The following lemma describes this.

Lemma 1.2 *For the matrix* $\mathbf{X}(n)\mathbf{X}^H(n)$*, the unique projection matrix* $\mathbf{P}_{R(\mathbf{X}(n)\mathbf{X}^H(n))}$ *is given by*

$$\mathbf{P}_{R(\mathbf{X}(n)\mathbf{X}^H(n))} = \mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}\mathbf{X}^H(n). \tag{1.23}$$

Proof: Again, we use a result about *existence* of projection matrices from Arun [10] which states:

Theorem 1.2 *Every subspace* S *of* C^N *has an associated projection matrix* $\mathbf{P}$ *which can be expressed in terms of a matrix* $\mathbf{B}$ *of the linearly independent basis vectors* b_i *which span the same subspace* S*.*

$$\mathbf{P} = \mathbf{B}(\mathbf{B}^H\mathbf{B})^{-1}\mathbf{B}^H. \tag{1.24}$$

Proof: Given in Arun [10].□.

Since $R(\mathbf{X}(n)\mathbf{X}^H(n)) = R(\mathbf{X}(n))$, and the vectors of $\mathbf{X}(n)$ can be assumed to be linearly independent for an independently distributed input signal, these vectors may be used as a basis for $R(\mathbf{X}(n))$ to yield the result

$$\mathbf{P}_{R(\mathbf{X}(n)\mathbf{X}^H(n))} = \mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}\mathbf{X}^H(n). \tag{1.25}$$

Finally, the mapping from $\mathbf{X}(\mathbf{n})D(n)$ to the minimum-norm least-squares solution of Equation (1.9) is given by the *Moore-Penrose* pseudo-inverse of $\mathbf{X}(n)\mathbf{X}^H(n)$. In terms of the minimum-norm solution to Equation (1.9) this yields

$$\begin{aligned} W_{MN}(n) &= \mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}D(n) \\ &= (\mathbf{X}(n)\mathbf{X}^H(n))^\dagger D(n), \end{aligned} \tag{1.26}$$

which completes the proof of Lemma 1.1. □

Using these combined results, and letting $\mathbf{Q} = \mathbf{X}(n)\mathbf{X}^H(n)$ and $b = \mathbf{X}(n)D(n)$, the adaptive filter coefficient updates can be rewritten as

$$W(n) = [\mathbf{I} - \frac{1}{1+\mu_n}\mathbf{P}_{\mathbf{R(Q)}}]W(n-1) + \frac{1}{1+\mu_n}\mathbf{Q}^\dagger b. \tag{1.27}$$

Heuristically, if the value μ_n approaches zero, the algorithm developed by Slock becomes equivalent to the algorithm developed by Boray and Srinath:

$$W(n) = \mathbf{P}_{\mathbf{N(Q)}} W(n-1) + \mathbf{Q}^{\dagger} b. \tag{1.28}$$

1.3.3 Vector normalized LMS algorithm

An alternative derivation of the affine projection technique may be developed by forming a vectorized form of the normalized LMS algorithm. This algorithm may be derived by re-visiting the standard Lagrange multiplier derivation of the normalized LMS algorithm [1]. For the vector NLMS algorithm, a set of constraining conditions can be developed in addition to forcing the *a-priori* error to be zero:

$$\begin{array}{cc} \min\limits_{W(n)} & ||W(n) - W(n-1)||_2^2 \\ \textit{subject to} & D(n) = \mathbf{X}^H(n) W(n). \end{array} \tag{1.29}$$

The closed-form solution to this system of equations may be formed by the Lagrange method of multipliers. More specifically, the following can be minimized:

$$J(W(n), \lambda) = [W(n) - W(n-1)]^H [W(n) - W(n-1)] + \lambda^H [D(n) - \mathbf{X}^H(n)] W(n). \tag{1.30}$$

To find the solution, the following derivatives are computed and set to zero:

$$\begin{aligned} \frac{\delta J(W(n), \lambda)}{\delta W(n)} &= W(n) + \mathbf{X}(n) = \lambda = 0 \\ \frac{\delta J(W(n), \lambda)}{\delta \lambda} &= D(n) - \mathbf{X}^H(n) W(n) = 0. \end{aligned} \tag{1.31}$$

These derivatives are

$$W(n) - W(n-1) + \mathbf{X}(n)\lambda = 0 \tag{1.32}$$

$$D(n) - \mathbf{X}^H(n) W(n) = 0. \tag{1.33}$$

Substituting Equation (1.32) into Equation (1.33) yields

$$D(n) - \mathbf{X}^H(n)[W(n-1) - \mathbf{X}(n)\lambda] = 0. \tag{1.34}$$

Or effectively, the following is the optimal choice of λ:

$$\lambda^* = -[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)W(n-1)], \tag{1.35}$$

assuming, of course, that the inverse $[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}$ exists. Substitution into Equation (1.32) yields an update expression for the tap-weights for the adaptive filtering coefficients:

$$W(n) = W(n-1) - \mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)W(n-1)]. \tag{1.36}$$

This algorithm is equivalent to the method developed by Slock [7, 8] with $\mu_n = 0$ and numerically equivalent to the approach of Boray and Srinath [11] (in infinite numerical precision).

1.3.4 Fast affine projection adaptive algorithms

While there are three equivalent derivations for the affine projection algorithm, there is also a large set of approximate implementations of this algorithm. The update expression for the affine projection algorithm of Slock, Boray-Srinath, and the algorithm developed here may be equivalently expressed as the following:

$$W(n) = W(n-1) + \frac{1}{1+\mu_n}\mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}E(n), \tag{1.37}$$

where $E(n) = D(n) - \mathbf{X}^H(n)W(n)$.

Slock [7, 8] and Montazeri and Duhamel [5, 6] prescribe an algorithm to efficiently compute the update of the adaptive filter coefficients in an exact fashion. They suggest that the *a-priori* errors can be computed with $O(NL)$ multiplications.

The matrix $\mathbf{R}(n) = \mathbf{X}^H(n)\mathbf{X}(n)$ can be updated from $\mathbf{R}(n-1)$ using $O(L \log_2 L)$ multiplications. This matrix is the instantaneous estimate of the autocorrelation matrix $\mathbf{R}_{xx}(n)$ of dimension $L \times L$.

Finally, the solution to the problem $\mathbf{R}(n)\eta_n = E(n)$ may be computed using the *generalized Levinson* algorithm with $O(L^2)$ multiplications. The update expression for the filter can be expressed in terms of the intermediate quantities:

$$W(n) = W(n-1) + \mathbf{X}(n)\eta_n. \tag{1.38}$$

The adaptive filter coefficients are then computable in $O(NL)$ multiplications. The overall complexity of this approach then becomes $O(NL) + O(Llog_2L) + O(L^2)$. Gay and Slock [7–9, 12] suggest that block implementations of this algorithm are reasonable for slowly-varying (in a statistical sense) data.

Further computational savings can be obtained by using methods motivated by developments used to reduce the computational complexity of traditional RLS techniques. Borrowing techniques from fast transversal filtering, algorithms are developed which can reduce the overall complexity to $O(L) + O(N)$ [9, 12–14]. For guaranteed stable versions of these techniques, $16N$ multiplications are required. These algorithms approximate the matrix $\mathbf{R}(n)$ in Toeplitz form rather than exact form to develop efficient update expressions.

An additional technique was developed by Douglas [15] using orthogonal transforms to develop an approximate affine projection adaptive algorithm. This method has an overall computational complexity of $O(L) + O(N) + O(Nlog_2N)$. This method approximates the autocorrelation matrix $\mathbf{R}(n)$ by using the vectors of the DFT or DCT matrix. Recently, an instrumental variable method for solving this problem was developed [16]. An instrumental variable is introduced in the computation of the inverse of the matrix $\mathbf{R}(n)$. While this method is computationally efficient, it is not provably stable. Several of the affine projection methods discussed here have been summarized by Morgan and Kratzer [17] and Papadias and Slock [18]. Further, Papadias [19] has developed an affine projection algorithm for updating the adaptive filter coefficients of a blind channel equalizer utilizing the constant modulus criterion.

While these techniques offer good performance for floating point algorithms, their performance in fixed point often leaves much to be desired due to their similarity to the fast transversal filtering algorithms [1]. In addition, their performance is significantly degraded by the presence of noise in the desired signal and correlation in the input signal [5, 6]. Many of these issues can be resolved by using a variant of the class of data-reusing algorithms. Unfortunately, these algorithms generally offer slower convergence-rate performance than the affine projection algorithms.

1.4 Data-Reusing Algorithms

The class of data-reusing algorithms in adaptive filtering was originally motivated by a need to exploit the use of "unused" cycles available on the processor. Assuming that the processor is awaiting samples in a traditional LMS adaptive filtering setting, reusing existing samples and data can lead to improved convergence performance compared with using the samples or data only once. There are several variants of data-reusing algorithms. The two most widely mentioned ones are the normalized and fixed-step-size data-reusing algorithms corresponding to normalized and fixed-step-size LMS algorithms.

1.4.1 Traditional data-reusing algorithms

The simplest class of data-reusing algorithms reuses the same sample again and again. Simply stated, the update expression for the LMS algorithm is given by

$$C_i = C_{i-1} + \mu e_i(n) X(n), \tag{1.39}$$

where $e_i(n) = d(n) - X^T(n) C_{i-1}$. The algorithm is initialized with $C_0 = W(n-1)$ and executed R steps, where R will be used throughout the dissertation to represent the number of reuses. After R steps, $W(n)$ is set to C_R. The parameter R is chosen by the system designer to match the adaptive filtering situation. In this manner, the algorithm reuses an existing data vector $X(n)$ and desired signal value $d(n)$ R times. For this algorithm, different choices of μ are more appropriate for this form of data reusing than traditional LMS filtering. For greater discussion of this algorithm, see Schnaufer [20].

This algorithm can possess improved convergence rate performance over the traditional fixed-step-size LMS adaptive algorithm. This approach is far from optimal, and can lead to significant stability problems if μ is not chosen appropriately.

1.4.2 New normalized data-reusing algorithms

Schnaufer [20, 21], described a family of algorithms for reusing existing data pairs for improving performance of the normalized and fixed-step-size LMS algorithms. By effectively reusing existing pairs of data, convergence to the Wiener-Hopf solution is obtained in a shorter time than possible using traditional normalized or fixed-step-size LMS algorithms. The normalized backward data-reusing algorithm for the system identification setting may be expressed as

- **Step 1.** *Initialization :* $i = 0$, $r = 0$, $C_0 = W(n)$.
- **Step 2.** *Loop 1:* (*While* $r < R$)

  Loop 2: (*While* $i < L$)

$$C_{i+1} = C_i + \alpha_i X(n-i) \tag{1.40}$$

$$\alpha_i = = \frac{d(n-i) - X^H(n-i)C_i}{\delta + X^H(n-i)X(n-i)} \tag{1.41}$$

$$i = i + 1 \tag{1.42}$$

  End Loop 2: i = 0, r = r + 1

  End Loop 1:

- **Step 3.** $W(n+1) = C_L$, $n = n+1$, and *Go to* **Step 1**,

where $W(n)$ is a vector of the adaptive filter coefficients, $X(n)$ is a vector of the previous N samples of the input data, and $d(n)$ represents the desired input signal. The vector C_i represents an intermediate iterate of the adaptive filter coefficient, $L-1$ represents the number of previously reused data pairs, and R represents the number of times that this data is reused. Depending upon the performance level and amount of computation desired, L and R are increased or decreased. For $L = 2$, the operation of this algorithm may be viewed graphically as shown in Figure 1.2 [20–22]. The normalized data-reusing

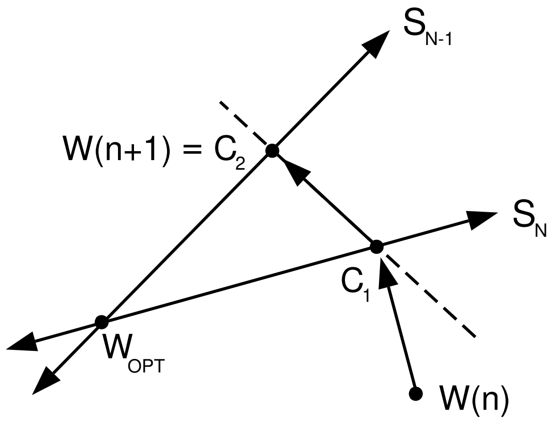


Figure 1.2 Graphical Description of the Normalized Data-Reusing LMS Algorithm ($L = 1$)

algorithm projects the current adaptive filter coefficient onto the hyperplane S_n defined by the zero instantaneous error condition $e(n) = 0 = d(n) - X^H(n)C_1$. The vector C_1 is chosen so that it is the orthogonal projection onto the translated subspace or hyperplane S_n.

This hyperplane S_n has dimension $N-1$, where N is the dimension of the filter. Note that at each iterate, the algorithm computes the orthogonal projection onto the previous solution hyperplane defined by the zero error conditions from the previous iterations, i.e., $e(n-1) = 0$, $e(n-2) = 0$, etc.

If the number of hyperplanes is fixed at two as in Figure 1.2, and the algorithm repeatedly computes the orthogonal projection in a direction orthogonal to each of these two convex hyperplanes, i.e., let $R \to \infty$, then it will converge to the intersection of the two sets which is labeled W_{opt}. For the purposes of illustration, W_{opt} is labeled as a point, but it is also a translated subspace or hyperplane with dimension $N-2$ for this example. This repeated projection operation is equivalent to letting the reuses of the two data pairs $(X(n), d(n))$ and $(X(n-1), d(n-1))$ approach infinity.

If the number of reused data pairs and hyperplanes L were set to N, then the intersection of the L hyperplanes would have dimension 1. At this point, the translated subspace W_{opt} is defined by a single vector. This vector would be unique if and only if the solution hyperplanes were not colinear. In the absence of measurement noise in $d(n)$ and colinear hyperplanes, W_{opt} would be the Wiener-Hopf solution [21].

The new normalized data-reusing algorithm approximates the behavior of the affine projection algorithms. As the reuse factor R approaches infinity while the number of hyperplanes L is kept constant, the algorithm will at each n^{th} iteration approach W_{opt}, the intersection of the valid solution hyperplanes. This operation is equivalent to directly projecting the current iterate $W(n-1)$ onto W_{opt}. The affine projection algorithms compute this projection directly in a finite number of steps. It is possible to come very close to the intersection using data-reusing methods in a finite number of steps without bearing the computational expense of the affine projection methods [21].

1.5 Overview

So far, several approaches to adaptive filtering have been presented. All of these approaches may be related using a concept of *row projection*. The relative advantages and disadvantages of each of these methods have been briefly discussed. The remainder of the thesis will focus on developing an underlying framework to relate these seemingly different approaches to adaptive filtering. In addition, using this framework, new methods for adaptive filtering will be developed and presented.

In Chapter 2, adaptive filtering is presented in the context of the method of *row projection*, a projection method for solving linear systems. Data-reusing and normalized LMS algorithms are placed into this class of projection methods and new methods are generated for data-reusing. Chapter 3 examines the use of acceleration methods in row projection and applies them to accelerate data-reusing adaptive filtering algorithms to develop some fundamentally new adaptive algorithms. Chapter 4 discusses the convergence properties of the *row projection* algorithms. In Chapter 5, a projection algorithm employing the fast quasi-Newton adaptive filtering algorithm developed by Marshall and Jenkins [23] using the conjugate gradient algorithm is presented in greater detail. Chapter 6 presents some applications of the projection-based adaptive filtering algorithms. These include two-dimensional adaptive processing, adaptive interference suppression, and fault tolerant adaptive filtering. Appendix A re-examines the conjugate gradient algorithm for the case of low-rank input matrices. It conjectures and proves what the conjugate gradient algorithm will compute for this case. Appendix B examines the issue of constant modulus adaptive algorithms and applies the *row projection* methods to the development of the new adaptive constant modulus algorithms.

CHAPTER 2

ROW PROJECTION METHODS APPLIED TO ADAPTIVE FILTERING

2.1 Introduction

Row-projection methods are a family of methods for solving linear systems described in [24, 25]. These methods are related to a number of adaptive filtering algorithms. They are an iterative class of methods for solving linear systems. Unlike direct methods, finite-order convergence is not always provable. But these methods offer significant advantages over direct methods in terms of computation and ease of implementation and relative insensitivities to ill-conditioning of data. These advantages help explain their efficacy in adaptive filtering algorithms.

The *row projection* methods may be described by first partitioning a linear system $\mathbf{A}x = b$ into m blocks as

$$\mathbf{A}^T = [A_1, A_2, \ldots, A_m]\,. \tag{2.1}$$

The vector b is also partitioned conformally. A *row projection* method is any method for solving the linear system which requires computing orthogonal projections $A_i A_i^{\dagger} x$ of a vector x onto the range spaces of the $A_i's$. Row projection methods allow for many different partitionings, e.g., $m \in [2, \ldots, M]$, of the matrix $\mathbf{A}$ which has dimension $M \times N$.

The *row projection* methods fall into two general categories, the Kaczmarz [26] or product forms, and the Cimmino [27] or sum forms. The operation of these methods can be explained using geometrical constructions which are not unlike the geometrical description of the normalized data-reusing LMS algorithm developed by Schnaufer [20, 21].

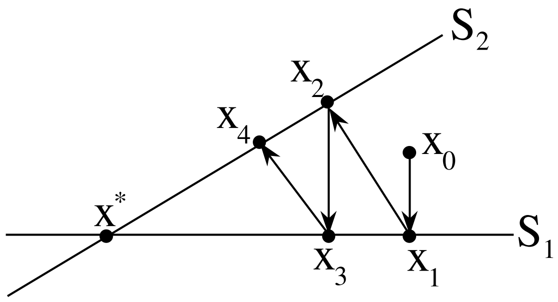


Figure 2.1 Repeated Row Projection by the Kaczmarz Method

2.2 The Kaczmarz Method for Row Projection

The Kaczmarz method updates the current solution by progressing through the matrix in a block-by-block fashion as follows:

$$x_{n+1} = (I - \mathbf{P}_i)x_n + \hat{b}_i, \tag{2.2}$$

where $\mathbf{P}_i = A_i(A_i^T A_i)^{-1} A_i^T$ for $i = 1, 2, \ldots, m$ and $\hat{b}_i = (A_i^T)^{\dagger} b_i$. By progressing repeatedly through the A_i's in an orderly fashion, convergence to the solution of $\mathbf{A}x = b$ is assured. Graphically, this repeated projection operation is presented in Figure 2.1 for the simple case where $m = 2$. In this figure, the Kaczmarz method iterates by repeatedly computing orthogonal projections between two hyperplanes which consist of the solution spaces of two different blocks, i.e., $S_1 = \{x : A_1^T x = b_1\}$ and $S_2 = \{x : A_2^T x = b_2\}$.

2.3 The Cimmino Method for Row Projection

Unlike the Kaczmarz method and the normalized data-reusing algorithms, which are serial or "in-place" algorithms, the Cimmino method is a *row projection* strategy which is highly parallel and effectively projects the current iterate onto a convex combination of the m range spaces of the A_i's. The generalized Cimmino method for *row projection* is given by

$$x_{k+1} = \left[I - 2 \sum_{i=1}^{m} \omega_i P_i \right] x_k + b_\sigma, \tag{2.3}$$

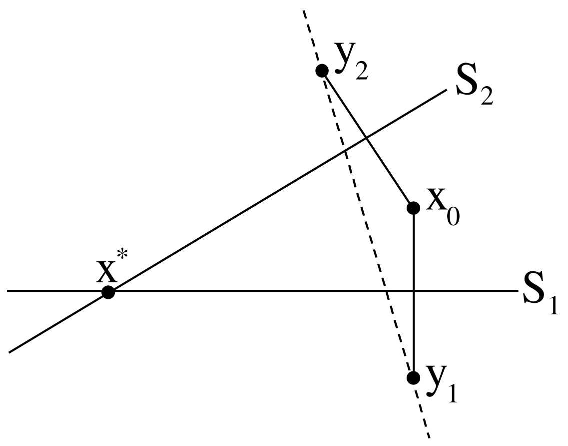


Figure 2.2 Cimmino Method for Row Projection

where

$$b_\sigma = 2\sum_{i=1}^{m} \omega_i \hat{b}_i, \tag{2.4}$$

and where $\hat{b}_i$ is defined as before. The ω_i's are chosen to ensure a convex combination of the projection matrices P_i's.

The Cimmino method may be understood by considering the homogeneous system $\mathbf{A}x = 0$. Instead of projecting x_0 onto $\mathbf{R}(A_i)$, let y_i be the reflection of x_0 through S_i. Then, geometrically, x_0 and each y_i lie on a circle centered at the solution $x^* = 0$. Any nontrivial convex combination of those points must lie strictly inside the circle and hence be closer to the solution. This forms the basis of the iterative method [24]. The Cimmino method is illustrated in Figure 2.2. In this example, the number of hyperplanes m is two. The first iterate is shown as a convex combination of the two reflection points y_1 and y_2. While this iterate lies at the midpoint of the bisecting line, any point along the convex line could be chosen to ensure convergence.

2.4 Connection between Row Projection Methods and Normalized LMS Algorithms

In the context of adaptive filtering, particularly for the normalized LMS algorithm, the adaptive filtering problem can be restated in terms of finding the solution to a linear system of equations. For traditional adaptive filtering applications, this system can be written as

$$\mathbf{X}^T(n)W(n) = D(n), \tag{2.5}$$

where $\mathbf{X}^T(n)$ is the matrix of input data signal. The matrix is given by

$$\mathbf{X}^T(n) = \begin{bmatrix} x(N-1) & x(N-2) & \dots & x(1) & x(0) \\ x(N) & x(N-1) & \dots & x(2) & x(1) \\ \vdots & & \ddots & & \vdots \\ x(n) & x(n-1) & \dots & x(n-N+2) & x(n-N+1) \end{bmatrix}. \tag{2.6}$$

The vector $W(n)$ is the adaptive filter coefficients, and $D(n)$ is a vector of the desired data which can be defined as

$$\begin{aligned} W(n) &= \begin{bmatrix} W_0(n), & W_1(n), & \dots & W_{N-1}(n) \end{bmatrix}^T \\ D(n) &= \begin{bmatrix} d(N-1), & d(N), & \dots & d(n) \end{bmatrix}^T. \end{aligned} \tag{2.7}$$

For the adaptive filtering problem, the number of columns is fixed at N, but the number of rows continues to grow as $n \to \infty$.

In order to relate the Kaczmarz method to the normalized LMS algorithm, the LMS algorithm can be rewritten in terms of projection matrices and pseudo-inverses. The relationship between the two algorithms becomes more clear:

$$\begin{aligned} W(n+1) &= W(n) + (d(n) - X^T(n)W(n)) \\ &\quad \times [X^T(n)X(n)]^{-1}X(n) & (2.8) \\ &= W(n) - X(n)[X^T(n)X(n)]^{-1}X^T(n)W(n) \\ &\quad + X(n)[X^T(n)X(n)]^{-1}d(n) & (2.9) \\ &= (\mathbf{I} - \mathbf{P}_{R(X(n))})W(n) + (X^T(n))^{\dagger}d(n), & (2.10) \end{aligned}$$

where $\mathbf{P}_{R(X(n))}$ is the unique projection matrix for the range space of $X(n)$.

The normalized LMS algorithm bears striking resemblance to the Kaczmarz method. The normalized LMS algorithm applies a Kaczmarz-type strategy to solving the linear system in Equation (2.5). However, the Kaczmarz method for solving linear systems is usually applied to finite dimensional linear systems. One additional note: *row projection* methods can project onto the respective row range spaces in any order, and convergence is still obtained [24]. Convergence of *row projection* methods, and more specifically, the adaptive algorithms developed in this dissertation are discussed in greater detail in Chapter 4.

2.5 Data-Reusing Strategies

Based upon the discussion of *row projection* methods, it is possible to develop a set of data-reusing algorithms which use the Kaczmarz and Cimmino *row projection* strategies. They are used in slightly modified form. As will be shown in the next subsection, the normalized data-reusing algorithms are slightly modified versions of the Kaczmarz algorithm. The Cimmino *row projection* algorithm will be used to derive a "new" form of data reusing. This "new" algorithm, as will be shown, can be related to LMS block filtering.

2.5.1 Kaczmarz normalized data-reusing methods

In addition to viewing the NLMS algorithm in terms of the *row projection* framework, the backward (or forward) normalized data-reusing algorithm can also be viewed in terms of this same ideology [28]. Rewriting the backward normalized data-reusing algorithm (special case of $R = 1$) in terms of projection matrices and pseudo-inverses to relate it to the Kaczmarz iteration yields the following expression:

$$W(n+1) = \prod_{i=0}^{L-1}(\mathbf{I} - \mathbf{P}_{X(n-i)})W(n) + D_\pi(n), \tag{2.11}$$

where $D_\pi(n)$ is given by

$$D_\pi(n) = (X^T(n))^\dagger d(n) + \sum_{i=1}^{L-1}\prod_{j=0}^{i}(\mathbf{I} - \mathbf{P}_{X(n-j)})X^T(n-i))^\dagger d(n-i). \tag{2.12}$$

Again, the same striking similarity to the Kaczmarz algorithm is visible. Yet, this algorithm projects onto each row L times. As the number of reused data pairs L for backward normalized data-reusing approaches infinity, the data-reusing algorithm becomes approximately the Kaczmarz method. However, there is one critical difference. Traditional *row projection* methods project on the range-spaces of the A_i's in a sequential fashion:

$$(A_1, \ldots, A_m, A_1, \ldots, A_m, A_1, \ldots) \, .$$

In backward normalized data-reusing, projections occur in the following order:

$$(A_L, \ldots, A_1, A_{L+1}, \ldots, A_2, A_{L+2}, \ldots) \, .$$

So, clearly, the iterates are not the same except for the limiting case, i.e., $L = M$, $m = M$, and $M \to \infty$.

2.5.2 Cimmino normalized data-reusing methods

The Cimmino method for *row projection* can be applied to formulate a "new" algorithm (special case $R = 1$) which resembles the data-reusing algorithm given in Equation (2.11):

$$\begin{aligned} W(n) &= \left[\mathbf{I} - \frac{2}{L} \sum_{i=0}^{L-1} \mathbf{P}_{X(n-i)} \right] W(n-1) \\ &\quad + \frac{2}{L} \sum_{i=0}^{L-1} (X^T(n-i))^\dagger d(n-i). \end{aligned} \tag{2.13}$$

where $\omega_i = 1/L$. This choice of ω_i ensures that the convex combination of the rows are taken. Further, this particular choice of ω_i will use the midpoint of the convex set formed by the L reflections of the previous point $W(n-1)$.

The Cimmino method for solving linear equations of the form $\mathbf{A}x = b$ forms the convex combination of all m blocks at every iteration. However, the data-reusing method utilizes only the previous L blocks or, for the adaptive filtering scenario, the previous L rows. Strictly speaking, this method for updating the adaptive filter coefficients is a hybrid approach combining both the Cimmino and Kaczmarz strategies of *row-projection*. Since the dimensions of the matrix $\mathbf{X}^T(n)$ are not fixed *a priori*, the convex combination cannot be formed of all of the rows.

2.5.2.1 Relation of the Cimmino method to block update LMS

The pseudo-inverses and the projection matrices in Equation (2.13) can be evaluated to gain further insight into how this algorithm is related to other LMS algorithms. Rewriting Equation (2.13) yields

$$\begin{aligned} W(n) &= \left[\mathbf{I} - \frac{2}{L}\sum_{i=0}^{L-1}\frac{X(n-i)X^T(n-i)}{X^T(n-i)X(n-i)}\right] \\ &\quad \times W(n-1) + \frac{2}{L}\sum_{i=0}^{L-1}\frac{X(n-i)d(n-i)}{X^T(n-i)X(n-i)} \\ &= W(n-1) + \frac{2}{L}\sum_{i=0}^{L-1}\frac{e(n-i)X(n-i)}{X^T(n-i)X(n-i)}, \end{aligned} \tag{2.14}$$

where $e(n-i) = d(n-i) - X^T(n-i)W(n-1)$. The Cimmino form of data-reusing can be related to the block LMS algorithm described by Haykin [1]. The block LMS algorithm for $L = N$ is given by the following expression:

$$W(n) = W(n-1) + \mu\sum_{i=0}^{L-1} X(nN-i)e(nN-i). \tag{2.15}$$

The block-LMS algorithm does not update the adaptive filter coefficients on a sample-by-sample basis. Instead, it forms a block estimate (average) of the gradient. The block LMS algorithm then updates the coefficients every Lth sample. The Cimmino-based method of data-reusing updates the tap-weight coefficients at every sampling instant.

The Cimmino data-reusing method can be used to gain some further insight into the operation of the block-LMS algorithm. Rewriting the data-reusing method in block processing form yields

$$W(n) = W(n-1) + \frac{2}{L}\sum_{i=0}^{L-1}\frac{e(nN-i)X(nN-i)}{X^T(nN-i)X(nN-i)}. \tag{2.16}$$

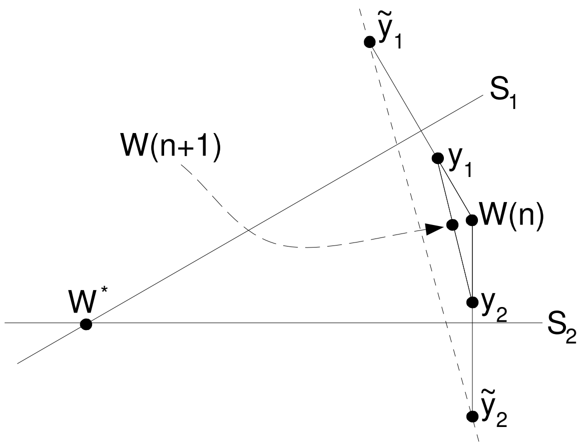


Figure 2.3 A Geometrical Interpretation of the Block LMS Algorithm

The Cimmino block-LMS method now begins to look very similar to the traditional block update LMS algorithm of Haykin [1] with a sample-dependent adjustment factor $\mu(i) = 1/X^T(nN-i)X(nN-i)$. Graphically, the block-update LMS algorithm can then be viewed in terms of the framework developed for the *row projection* methods. This is shown in Figure 2.3. For a fixed μ factor, the projections towards the two hyperplanes are no longer of uniform length. For a fixed μ factor and the simple case where $L = 2$, the solution will be the midpoint of the line-segment formed by the projections labeled y_1 and y_2. To compare the two methods, the Cimmino or normalized projections are shown in Figure 2.3 as $\tilde{y}_1$ and $\tilde{y}_2$. Any point along the line segment formed by $\tilde{y}_1$ and $\tilde{y}_2$ is closer to the intersection W^* of the two hyperplanes S_1 and S_2. Graphically, the Cimmino method appears to reach the intersection more rapidly.

The Cimmino data-reusing method offers several advantages over the Kaczmarz style of coefficient update. Both methods require the computation of the same number of projection matrices and pseudo-inverses. Yet, the Cimmino form requires only that the weighted sum of these projection matrices be computed rather than the product. In addition, the Cimmino data-reusing algorithm lends itself to parallelism. The projections may be computed independently and their weighted results combined to compute the adaptive filter coefficient update.

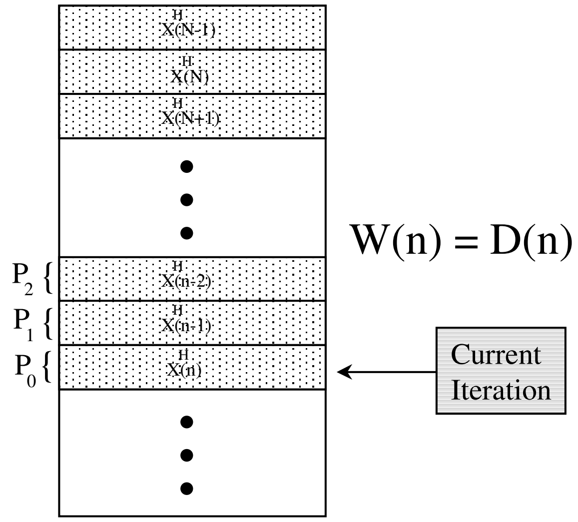


Figure 2.4 Matrix View of Data-Reusing Methods

2.6 Further Extensions: Inner versus Outer Loop Implementations

The Kaczmarz and Cimmino data-reusing algorithms have all assumed that the data is partitioned row by row, and that the methods attempt to approximate the best local projector to a set of L rows. This operation is best illustrated by Figure 2.4. At each iteration, the data-reusing method approximates the behavior of the local projector on the previous L rows. In the figure, $L = 3$. Three projectors are formed, and repeated projection onto the respective row range spaces leads to convergence to the projection of the combined range spaces of the three rows. This local set of iterations, which from now on will be called "inner loop" iterations, computes the approximation.

As an alternative, two rows could be grouped together, and the exact projector could be computed. This projector may be computed in a variety of ways, but generally it requires the use of an additional algorithm such as direct matrix inversion or the conjugate gradient algorithm [24].

This operation could be best illustrated by Figure 2.5. In this figure, two rows are grouped together to form one projector, a *block* projector. Now, the operation of the inner loop iteration for two rows has been subsumed by a single projection operation. However,

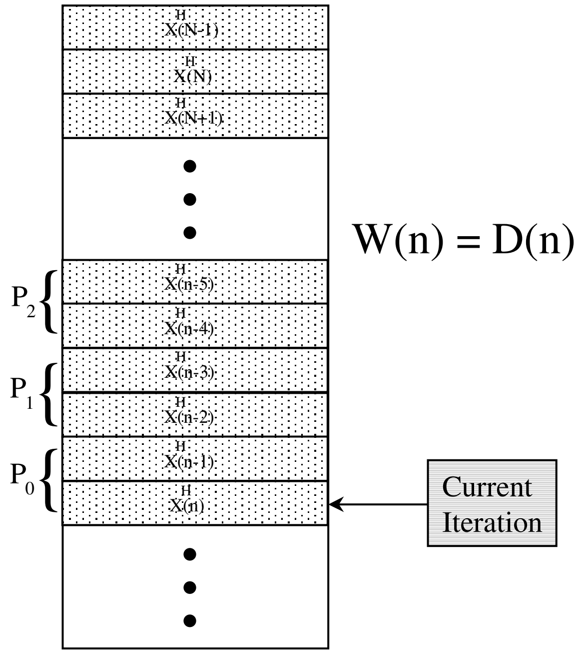


Figure 2.5 Matrix View of the Block Data-Reusing Methods

the projection to the three projectors P_0, P_1, and P_2 could be computed repeatedly. This iteration would become the new inner loop.

As discussed previously, the groupings of rows can lead to significantly different levels of performance. In each case, convergence to the ideal mean square error optimal solution occurs by the overall progression through each of the projectors (or *block* projectors). This operation of progressing through each set of inner loops will be referred to as the "outer loop" operation.

2.7 Experimental Comparison of Methods

The performances of the adaptive methods developed in this chapter were tested in a number of applications. Presented in the body of this dissertation are primarily system identification and channel equalization experiments. The system identification setting for adaptive filters is shown in Figure 1.1 in Chapter 1. Simulations for system identification are presented in Section 2.7.1.

Similar to system identification, channel equalization develops a model of the inverse of the unknown channel. Once a channel is properly equalized, communication across

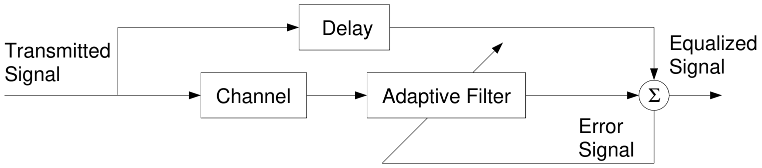


Figure 2.6 Adaptive Channel Equalization, Training Signal Present

the channel can be made reliable. A simple channel equalizer is illustrated in Figure 2.6. For the purposes of illustration, the details of the receiver are omitted.

The goal of the adaptive filter is to minimize the dispersive and distorting nature of the channel. It does this by forming an approximate inverse to the channel. In this way, the probability of error communicating across this channel is reduced. For the purposes of illustration, a training sequence is illustrated as available to the equalizer. This is often not the case.

This dissertation considers a special class of algorithms known as constant modulus algorithms. These property restoral algorithms assume something about the nature of the input signal. Primarily, they assume that the transmitted signal is binary phase shift keyed (BPSK) in nature, and possesses a constant envelope. The applications of the projection methods to the constant modulus algorithm are discussed in greater detail in Appendix B. Simulations of channel equalization are presented in Section 2.7.2.

2.7.1 System identification

Simulations were performed to illustrate the differences in performance between the Kaczmarz data-reusing (KDR) algorithm, the Cimmino data-reusing algorithm (CDR) and the normalized LMS (NLMS) algorithm. The parameters included the number of reused pairs, the level of reusing, and the coloring of the input noise sequence. Primarily, the experiments were designed to illustrate the improvements in performance that may be obtained by using the KDR and CDR methods. Further, they also help illustrate the relative performance differences between the two algorithms as a function of the choice of parameters the level of reusing R, and the number of reuses L.

In the first experiment, the performance of the algorithms was characterized for the case where the input signal for the system identification application was “white” noise.

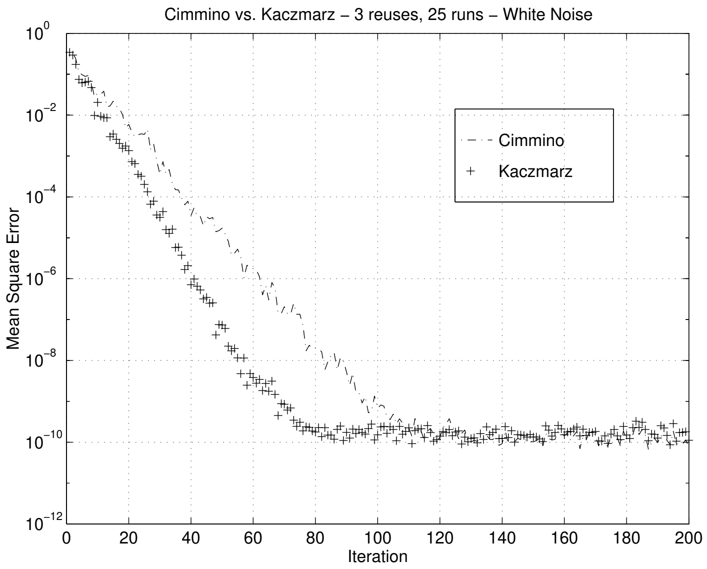


Figure 2.7 MSE Comparison: Cimmino versus Kaczmarz Algorithms, White Noise, L = 3, and 25 runs (avg.)

In Figure 2.7, the mean square error convergence in a system identification setting is plotted. The system has order $N = 8$. The unknown system is an 8-tap FIR model. The number of reuses L for both algorithms was three. The data is reused once ($R = 1$). For this example, the CDR method appears to exhibit slightly poorer performance than the KDR method, but the deviation is less than 20 iterations.

In Figure 2.8, the same system is again used for system identification, but the input signal is now colored by a low-pass filter which introduces a high degree of correlation. For this example, the KDR method appears to converge much more rapidly than the CDR method, converging in about 12000 iterations compared to 20000 iterations for the KDR method.

The convergence-rate performance of the Kaczmarz algorithm and the Cimmino algorithm can be made the same by increasing the number of times that the data is reused while keeping the amount of data constant. For three previous data pairs, the inner loop of the Kaczmarz and the Cimmino iteration is executed until the norm of the residual difference between iterations is less than 10^{-5}.

Figure 2.9 displays the progression of the mean square error for the colored noise input for both the Kaczmarz and the Cimmino iterates for a single run of data. The

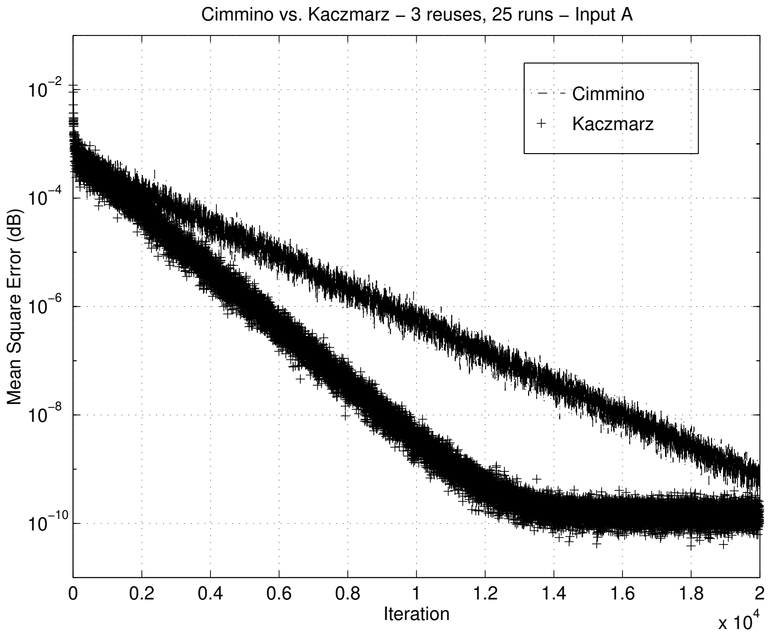


Figure 2.8 MSE Comparison: Cimmino versus Kaczmarz Algorithms, Colored Noise Input, L = 3, and 25 runs (avg.)

convergence histories appear to be very similar. Plotted directly below this in Figure 2.9 is the ratio of the number of iterations for both algorithms to reach convergence. The Cimmino algorithm consistently seems to take about four times as many iterations as the Kaczmarz form of data-reusing for this particular example. In Figure 2.9, the optimal performance for reusing three data pairs is obtained. However, note that to obtain this performance, the number of iterations of the inner loop must be very large (approximately 6000 in some cases).

If a significant number of multipliers is available for the computation of the update of the adaptive filter coefficients during the “dead-time” discussed earlier, then reasonably, the Cimmino-based algorithm may update the adaptive filter coefficients the same number of times that the Kaczmarz iteration does. For this situation, the progression of the mean square error for both algorithms is shown in Figure 2.10. Effectively, the inner loop of the Cimmino-based algorithm is exercised four iterations compared to once for the Kaczmarz-based algorithm. The disparity in convergence rates between the two data-reuse schemes has almost disappeared.

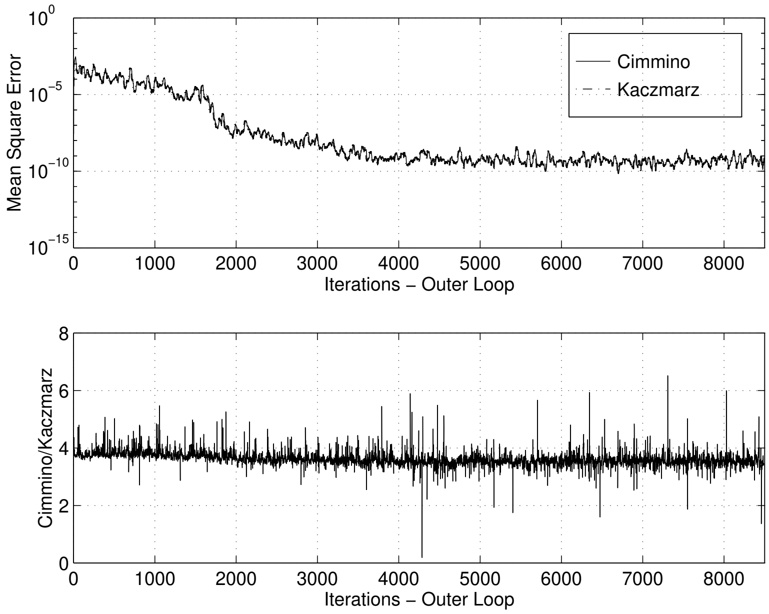


Figure 2.9 MSE Comparison: Cimmino versus Kaczmarz Algorithms, Colored Noise Input, L = 3, Residual-Based Reusing, Single run

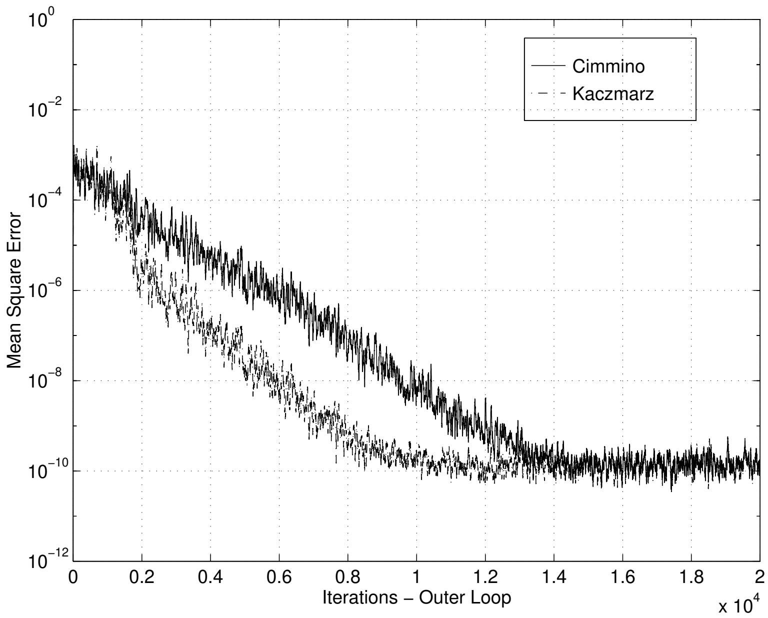


Figure 2.10 MSE Comparison: Cimmino (norm.) versus Kaczmarz, Colored Noise Input, L = 3

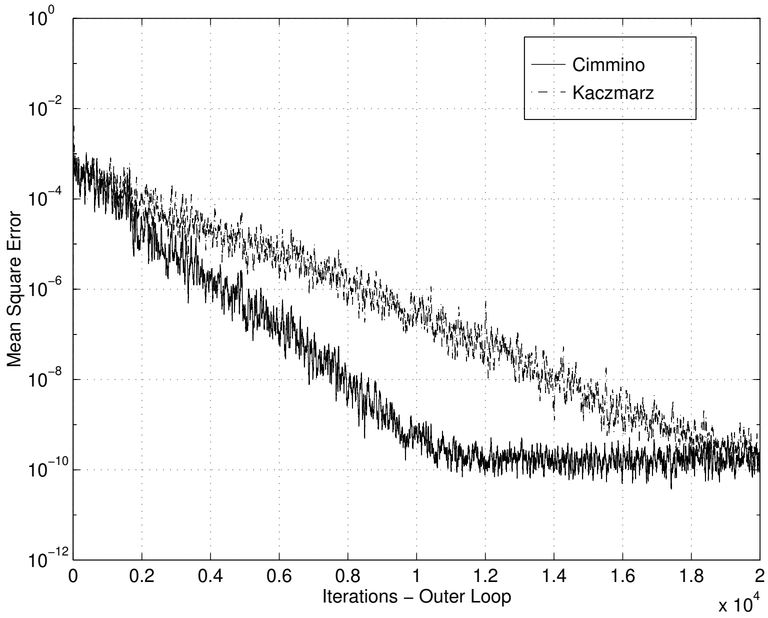


Figure 2.11 MSE Comparison: Cimmino (norm.) versus Kaczmarz, Colored Noise Input, L = 4, and 3, respectively

Finally, the simple case is considered where the Cimmino-based method is allowed to use more data than the Kaczmarz iterate. Three additional data pairs for the Cimmino algorithm are reused, and two additional pairs for the Kaczmarz algorithm are used. In addition, the Cimmino-based data-reusing scheme updates the coefficients the same number of times as the Kaczmarz iteration. For this situation shown in Figure 2.11, the Cimmino algorithm appears to converge faster than the Kaczmarz algorithm. It converges in 10000 iterations compared to the 20000 iterations required by the Kaczmarz iteration.

2.7.2 Channel equalization

Several experiments were performed to test the efficacy of the data-reusing methods in the equalization setting. They were compared against the performance of the LMS algorithm chosen with nearly experimentally optimal values of μ. The transmitted sequence was an equally likely set of zeros and ones. The channel model was a simple three-tap FIR filter with linear phase. The adaptive filter was chosen to be 11 coefficients. The

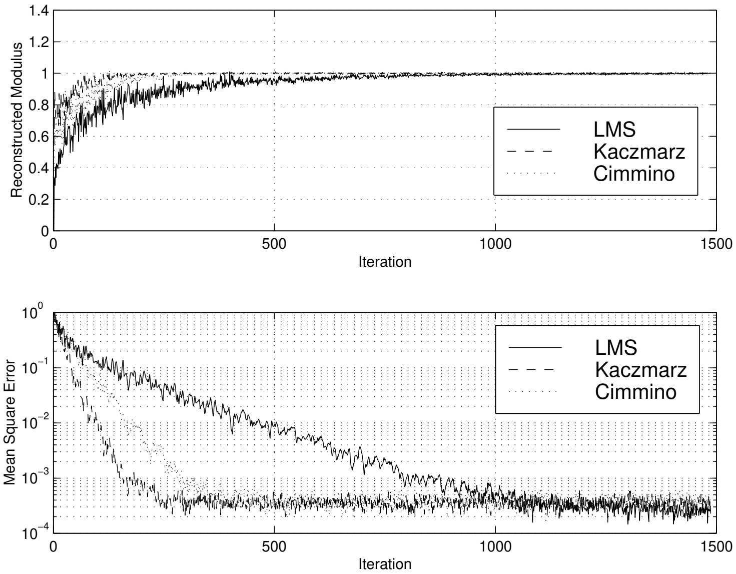


Figure 2.12 Channel Equalizer Performance, Training Signal Present

parameter L, the number of reused pairs, was chosen to be three, and the number of reuses R was chosen to be one.

In the first experiment, the training signal was present at the receiver. The resulting reconstructed modulus of the signal and the progression of the mean square error for 50 runs of the algorithm are displayed in Figure 2.12. For this example, the disparity in performance between the LMS algorithm and the data-reusing algorithms with a fairly low number of reuses, and reused pairs is clear. Again, the Kaczmarz algorithm performs slightly better than the Cimmino algorithm. For this example, the disparity is not as great as the system identification setting.

In the second experiment, the training signal was no longer present. In this case, the algorithm works as a property restoral algorithm (see Appendix B) and recovers the modulus of the signal and equalizes the channel. The resulting reconstructed modulus of the signal and the progression of the mean square error are displayed in Figure 2.13. Again, the benefits of using the data-reusing strategy are clear for this example. The difference in performance between the Cimmino and Kaczmarz algorithms has grown slightly over Figure 2.12, but both algorithms are still performing better than the LMS algorithm.

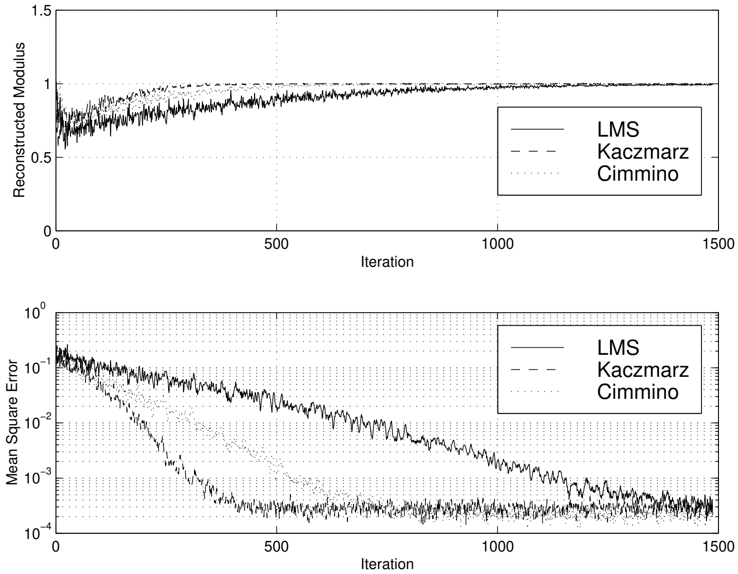


Figure 2.13 Channel Equalizer Performance, Constant Modulus Algorithm

2.7.3 Finite word length effects

Some experiments were run to illustrate the efficacy of these algorithms in finite precision. The simulations were run in a mixed domain. The inputs to all multiplication operations were prequantized, and the output of these multipliers was again quantized. Figure 2.14 illustrates the impact of using 8-bits of precision, and 16 bits of precision with the LMS algorithm in a system identification setting with 8 taps. The lower precision of operation implies a higher residual mean square error or misadjustment. The 8-bit precision algorithm converges to an overall misadjustment of 10^{-3}. The 16-bit precision algorithm converges to an overall misadjustment of 10^{-8}.

The Kaczmarz and Cimmino forms of data reusing were then compared to the performance of the LMS algorithm. Figure 2.15 illustrates the disparity between 8-bit and 16-bit precision for the Kaczmarz form of data reusing. Note that while the convergence rate has dramatically improved for this algorithm, the residual mean square error is about the same as for the LMS algorithm.

The performance of the Cimmino form of data reusing is illustrated in Figure 2.16. For this example, the Cimmino form of data reusing converges to a lower overall misadjustment than the Kaczmarz form. The convergence rate is slightly slower, however.

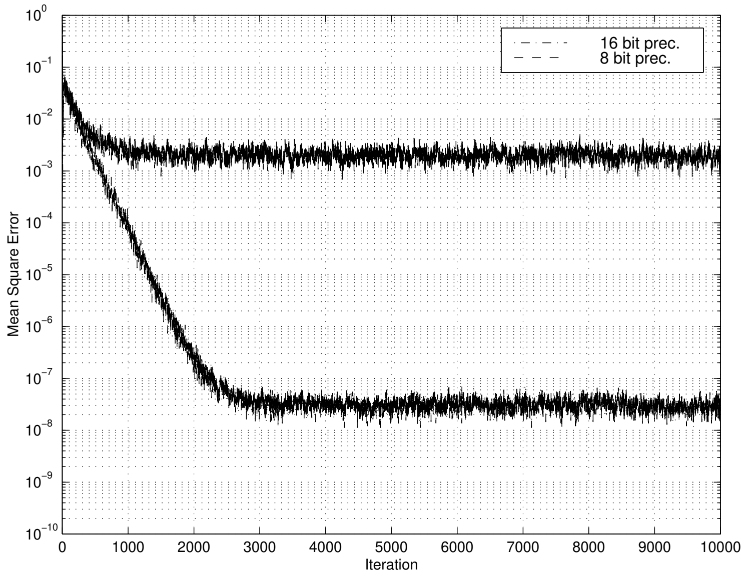


Figure 2.14 Convergence Performance, LMS Algorithm, Fixed Point Precision

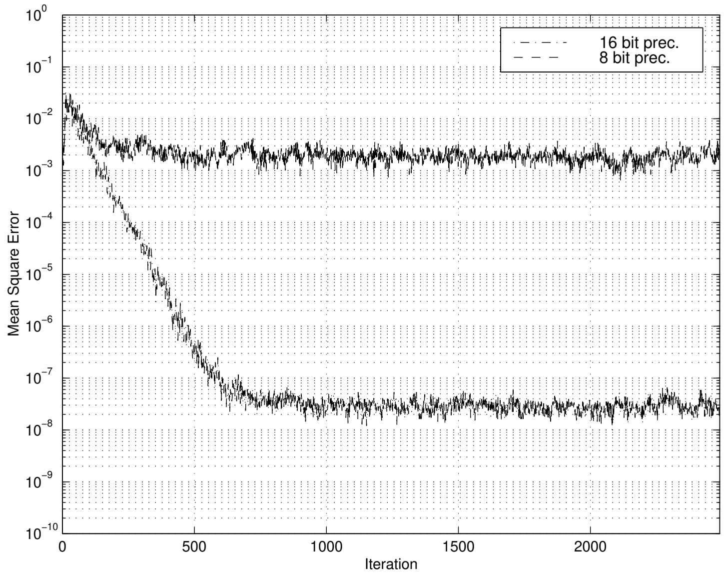


Figure 2.15 Convergence Performance, Kaczmarz Data-Reusing Algorithm, Fixed Point Precision, L = 3, R = 2

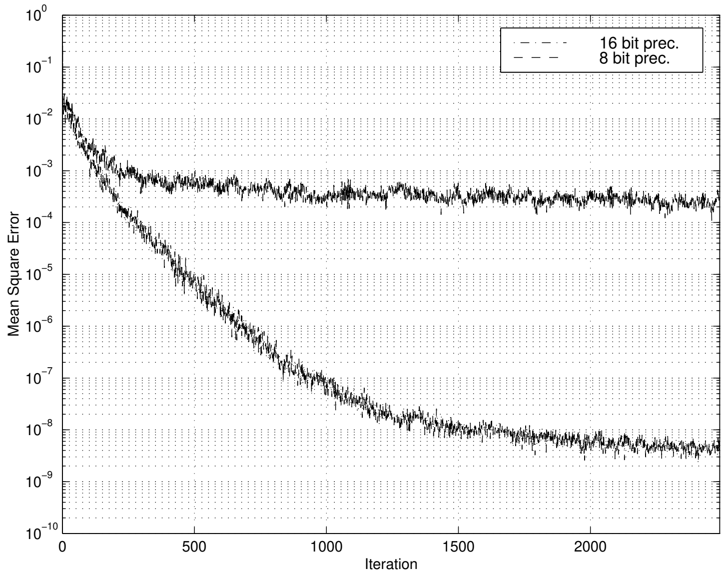


Figure 2.16 Convergence Performance, Cimmino Data-Reusing Algorithm, Fixed Point Precision, L = 3, R = 2

CHAPTER 3

ACCELERATION OF ROW PROJECTION AND APPLICATIONS TO ADAPTIVE FILTERING

While the performance of the data-reusing methods can be significantly better than the LMS or normalized LMS algorithms, further enhancements in convergence performance can be obtained. The Kaczmarz and Cimmino data-reusing adaptive algorithms can be accelerated by using methods which are analogous to those employed to accelerate *row projection* methods in general [29]. In the *row projection* framework, two common acceleration methods are the conjugate gradient method and Tchebyshev method [24]. These are iterative methods for solving linear systems that require the linear systems to be symmetric and positive definite to converge. The constraint on positive definiteness can be relaxed, but the systems must be symmetric. *Row projection* methods are a general class of methods for solving linear systems with few constraints. The acceleration methods are a prescription for translating the original nonsymmetric linear system problem into a symmetric and positive definite problem.

3.1 Acceleration of Row Projection Methods

The *row projection* methods can be parameterized as an iterative method in the following form:

$$x_{k+1} = \mathbf{Q}x_k + \tilde{b}. \tag{3.1}$$

This method will converge to the solution of the linear system:

$$(\mathbf{I} - \mathbf{Q})x = \tilde{b} \tag{3.2}$$

if and only if the matrix $\mathbf{Q}$ has eigenvalues contained in the interval $[0, 1]$.

Solving the system in Equation (3.2) has several advantages over solving the traditional $\mathbf{A}x = b$. The new system is symmetric and positive semidefinite. Thus, a broad class of iterative methods may be used to solve the system in Equation (3.2). The acceleration methods compute the solution of the positive semidefinite symmetric system described by Equation (3.2). Iterative methods with finite-order convergence properties are generally used to compute the solution to Equation (3.2). Otherwise, there would be few computational advantages over using the algorithm described in Equation (3.1).

Tanabe [25] has shown that the method described in Equation (3.1) will converge to the solution of Equation (3.2). The method of Equation (3.1) computes $x^* = (\mathbf{I} - \mathbf{Q})^\dagger \tilde{b}$ where † is notation representing the Moore-Penrose pseudo-inverse. Further, he presents proof that the solution x^* is also the minimum-norm least squares (MNLS) solution to the linear system $Ax = b$.

The Kaczmarz method of *row projection* as described earlier will not lead to a symmetric matrix $\mathbf{Q}$. To make the Kaczmarz method symmetric, a backward sweep of the rows is coupled with a forward sweep of the range spaces of the rows, $A_i's$. For this symmetric version of the Kaczmarz, the matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = (\mathbf{I} - \mathbf{P}_1)(\mathbf{I} - \mathbf{P}_2) \cdots (\mathbf{I} - \mathbf{P}_m) \cdots (\mathbf{I} - \mathbf{P}_2)(\mathbf{I} - \mathbf{P}_1). \tag{3.3}$$

This forces the matrix $\mathbf{Q}$ to be symmetric. The vector $\tilde{b}$ can be written as $\mathbf{T}b$ where the ith block column of $\mathbf{T}$ is given by

$$\prod_{j=1}^{i-1} (\mathbf{I} - \mathbf{P}_j) \left[I + \prod_{j=i}^{m} (\mathbf{I} - \mathbf{P}_j) \prod_{j=m-1}^{i+1} (\mathbf{I} - \mathbf{P}_j) \right] (A_i^T)^\dagger. \tag{3.4}$$

For the Cimmino method of *row projection*, the matrix $\mathbf{Q}$ is given by

$$\mathbf{Q} = \mathbf{I} - \frac{2}{L} \sum_{i=0}^{L} \mathbf{P}_i, \tag{3.5}$$

and $\tilde{b}$ is given by

$$\tilde{b} = \frac{2}{L} \sum_{i=0}^{L} A_i^\dagger b_i. \tag{3.6}$$

Note that the matrix $\mathbf{Q}$ is the sum of symmetric matrices which makes it symmetric. No further processing is required to make this method symmetric.This is an additional advantage of the Cimmino form of row projection. In the next section, the accelerated forms of *row projection* will be applied to data-reusing algorithms.

3.2 Acceleration of Data-Reusing Methods

The acceleration methods applied to standard *row-projection* algorithms can be applied to the adaptive filtering problem with data reusing [29]. For Kaczmarz data-reusing, a backward sweep of the previous $L-1$ rows (traditional normalized data reusing) is coupled with a forward sweep of the previous $L-1$ rows to yield the following update expression for a symmetric version of data reusing:

$$W(n+1) = \mathbf{Q}_K(n)W(n) + D_\pi(n), \tag{3.7}$$

where $\mathbf{Q}_K(n)$ is given by

$$\mathbf{Q}_K(n) = (\mathbf{I} - \mathbf{P}_{x(n)}) \cdots (\mathbf{I} - \mathbf{P}_{x(n-L)}) \cdots (\mathbf{I} - \mathbf{P}_{x(n)}) \tag{3.8}$$

and $D_\pi(n)$ can be written as $D_\pi(n) = \mathbf{T}D(n)$, where the ith block column of $\mathbf{T}$ is given by

$$\begin{aligned} &\prod_{j=0}^{i-1}(\mathbf{I} - \mathbf{P}_{X(n-j)})[\mathbf{I} + \prod_{j=i}^{L-1}(\mathbf{I} - \mathbf{P}_{X(n-j})\\ &\times \prod_{j=L-2}^{i+1}(\mathbf{I} - \mathbf{P}_{X(n-j)})](X(n-i)^T)^\dagger. \end{aligned} \tag{3.9}$$

Equivalently, the Cimmino data-reusing method may be accelerated. The matrix $\mathbf{Q}_C(n)$ is given by

$$\mathbf{Q}_C(n) = \left(\mathbf{I} - \frac{2}{L}\sum_{i=0}^{L}\mathbf{P}_{X(n-i)}\right). \tag{3.10}$$

The vector $D_\pi(n)$ is given by

$$D_\pi(n) = \frac{2}{L}\sum_{i=0}^{L}(X^T(n-i))^\dagger d(n-i). \tag{3.11}$$

For both data-reusing methods, acceleration methods are applied to the system:

$$[\mathbf{I} - \mathbf{Q}(n)]W(n) = D_\pi(n) \tag{3.12}$$

with one additional caveat: the matrix $(\mathbf{I}-\mathbf{Q}(n))$ is not full-rank. Solution of this system by itself will not lead to convergence to the Wiener-Hopf solution for the adaptive filter. If the linear system $\mathbf{A}x = b$ is full rank, then a consistent set of equations with the form of Equation (3.12) may be developed.

The affine projection algorithms, as discussed earlier in Chapter 1, solve a rank-deficient least squares problem. The system in Equation (3.12) is the same rank-deficient least squares problem transformed into a symmetric problem. The goal of the two algorithms is the same; they must both converge to the Wiener-Hopf solution for the adaptive filter. The affine projection algorithms solve a low-rank least squares problem subject to an additional constraint; the norm of the difference between successive iterations of adaptive filter coefficients must be minimized. Since the affine projection algorithms can also be viewed as a *row projection* algorithm, the same basic constraint applies to accelerated data reusing.

Therefore, the acceleration algorithms, conjugate gradient and Tchebyshev, should be initialized with the previous iterate of the adaptive filter coefficients, $W(n-1)$. This will insure convergence to the Wiener-Hopf solution (as will be discussed in greater detail in Chapter 4). Further, the solution of the following optimization problem will be computed:

$$\begin{aligned} \min_{W(n)} \quad & ||W(n) - W(n-1)||^2 \\ \textit{subject to} \quad & (\mathbf{I} - \mathbf{Q}(n))W(n) = D_\pi(n). \end{aligned} \tag{3.13}$$

In subsequent sections, the application of the conjugate gradient and Tchebyshev algorithm will be described and their use in accelerated data reusing will be demonstrated.

3.2.1 Conjugate gradient acceleration

The conjugate gradient algorithm, as described in Appendix A, is a method for solving linear symmetric positive semidefinite systems of equations. Appendix A shows that the algorithm may be applied to problems which do not satisfy the positive semidefinite property. The conjugate gradient method is a conjugate direction method. It iteratively defines a set of directions which are conjugate with respect to the system matrix [30]. Using these successively generated directions, it optimizes and finds the solution spanned by these direction vectors which is closest to the solution of the linear system.

The conjugate gradient method requires a finite number of iterations to converge. Assuming that the system has full rank, the algorithm will require N iterations to converge. However, if the linear system has less than full rank, i.e., the system has rank L, the conjugate gradient algorithm will converge in L steps. The rank of the matrix $[\mathbf{I} - \mathbf{Q}(n)]$ for both the Cimmino and Kaczmarz methods of data-reusing can reasonably be assumed to be L. Hence, applications of the conjugate gradient algorithm in data reusing will converge to the solution of Equation (3.12) in L steps. Based upon the results of Appendix A, the conjugate gradient method, if initialized with $W(n-1)$, will compute [30]:

$$W(n) = (\mathbf{I} - \mathbf{P}_{R(\mathbf{I}-\mathbf{Q}(n))})W(n-1) + [\mathbf{I} - \mathbf{Q}(n)]^{\dagger} d_{\pi}(n). \tag{3.14}$$

The general form of the conjugate gradient part of the accelerated data-reusing algorithm is summarized in Table 3.1. The conjugate gradient algorithm is initialized with $x_0 = W(n-1)$. After convergence in L steps, the adaptive filter coefficients $W(n)$ are set to the final output x_L of the conjugate gradient algorithm.

The conjugate gradient algorithm requires two matrix vector multiplications per step of the conjugate gradient algorithm. In addition, there are several vector multiplications. The overall computational complexity of this approach is $O(L^2N)$. In the next section, the Tchebyshev algorithm is explored as an alternative to the conjugate gradient algorithm.

Table 3.1 Summary of the Conjugate Gradient Algorithm

Initialization:	$g_0 = d_\pi(n) - (\mathbf{I} - \mathbf{Q}(n))x_0$ $d_0 = -g_0$
Repeat for $k = 1, 2, \ldots, L$:	$x_{k+1} = x_k + \alpha_k d_k$ $\alpha_k = \frac{g_k^T d_k}{d_k^T (\mathbf{I} - \mathbf{Q}(n)) d_k}$ $d_{k+1} = -g_{k+1} + \beta_k d_k$ $\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$

3.2.2 Tchebyshev algorithm acceleration

As previously stated, the conjugate gradient algorithm will converge in L steps. In addition, the algorithm requires two matrix-vector multiplications per step, which implies an overall complexity of $O(L^2 N)$. The number of matrix-vector multiplications may be reduced by using the Tchebyshev algorithm described in [24, 31].

The version of the Tchebyshev algorithm used by this dissertation for the acceleration of data-reusing methods is described in Bramley and is summarized in Table 3.2. This version of the algorithm computes the solution of the linear system $(\mathbf{I} - \mathbf{Q})x = b$. It also assumes that the maximal eigenvalue of the matrix $(\mathbf{I} - \mathbf{Q})$ is 1.

The parameters δ and γ, listed in Table 3.2, can be computed from the minimal eigenvalue ν as $\delta = (1 + \nu)/2$ and $\gamma = (1 - \nu)/2$. The algorithm assumes that their is an offline method of estimating the magnitude of the smallest eigenvalue. There are other methods which estimate the magnitude of the smallest eigenvalue in-line, but their use is not explored in this dissertation [24, 31].

3.2.2.1 Tchebyshev algorithm - background

The Tchebyshev method is a polynomial method for accelerating the computation of the solution to a system of linear equations. For any polynomial acceleration procedure applied to the basic iterative method of the form

$$x_k = \mathbf{G}x_{k-1} + k, \tag{3.15}$$

Table 3.2 Summary of the Tchebyshev Algorithm

Initialization:	$r_0 = \tilde{b} - (I - Q)x_0$
	$d_0 = r_0/\delta$
	$x_1 = x_0 + d_0$
	$\alpha_1 = 2\delta/(2\delta^2 - \gamma^2)$
Repeat for $k = 1, 2, \ldots, K$:	$r_k = \tilde{b} - (I - Q)x_k$
	$d_k = \alpha_k r_k + \beta_k d_{k-1}$
	$x_k = x_{k-1} + d_k$
	$\alpha_k = [\delta - \gamma^2\alpha/4]^{-1}$
	$\beta_k = \delta\alpha_k - 1$

the error vector associated with the procedure may be expressed as

$$\epsilon^{(k)} = Q_k(\mathbf{G})\epsilon^{(0)}, \tag{3.16}$$

where $Q_k(\mathbf{G}) = \alpha_{k,0}\mathbf{I} + \alpha_{k,1}\mathbf{G} + \cdots + \alpha_{k,k}\mathbf{G}^k$ is a matrix polynomial subject only to the condition that $\sum_{i=0}^{k} \alpha_{k,i} = 1$. Using the algebraic polynomial $Q_k(x)$ associated with $Q_k(\mathbf{G})$, the "virtual" spectral radius of the matrix $Q_k(\mathbf{G})$ can be defined:

$$\overline{\mathbf{S}}(Q_k(\mathbf{G})) = \max_{m(\mathbf{G}) \leq x \leq M(\mathbf{G})} |Q_k(x)|. \tag{3.17}$$

The quantities $M(\mathbf{G})$ and $m(\mathbf{G})$ denote, respectively, the algebraically largest and smallest eigenvalues of G. The particular polynomial method which is obtained by choosing the polynomial sequence $\{Q_k(\mathbf{G})\}$ which minimizes $\overline{\mathbf{S}}(Q_k(\mathbf{G}))$ for $k = 1, \ldots, n$ is called the Tchebyshev polynomial acceleration method.

The polynomial sequence which minimizes $\overline{\mathbf{S}}(Q_k(\mathbf{G}))$ can be defined in terms of the Tchebyshev polynomials. In order to explain this method, a brief outline of the useful properties of Tchebyshev acceleration is presented. The Tchebyshev polynomials may be defined by a basic recurrence relationship:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \end{aligned}$$

$$T_{n+1}(x) \quad = \quad 2xT_n(x) - T_{n-1}(x), n \geq 1. \tag{3.18}$$

It may also be shown by mathematical induction that $T_n(x)$ may be expressed as

$$T_n(x) = \cos(n \cos^{-1} x). \tag{3.19}$$

The fundamental property of Tchebyshev polynomials that is utilized most heavily in acceleration techniques is the following theorem [32].

Theorem 3.1 *Let n be a fixed integer and let d be any fixed real number such that $d > 1$. Defining $H_n(x)$ as*

$$H_n(x) = T_n(x)/T_n(d) \tag{3.20}$$

where $T_n(x)$ is the Tchebyshev polynomial, then

$$H_n(d) = 1 \tag{3.21}$$

and

$$\max_{-1 \leq x \leq 1} |H_n(x)| = 1/T_n(d). \tag{3.22}$$

Further, if $Q(x)$ is any polynomial of degree n or less such that $Q(d) = 1$ and

$$\max_{-1 \leq x \leq 1} |Q(x)| \leq \max_{-1 \leq x \leq 1} |H_n(x)| \tag{3.23}$$

then,

$$Q(x) = H_n(x) \tag{3.24}$$

Proof: See Young [31] or Flanders and Shortley [33]. □

The goal of polynomial acceleration is to minimize the virtual spectral radius of the matrix polynomial sequence $Q_n(\mathbf{G})$. Specifically, a polynomial $P_n(x)$ must be found such that $P_n(1) = 1$ and

$$\max_{m(\mathbf{G}) \leq x \leq M(\mathbf{G})} |P_n(x)| \leq \max_{m(\mathbf{G}) \leq x \leq M(\mathbf{G})} |Q_n(x)|. \tag{3.25}$$

Note that $Q_n(x)$ is any polynomial of degree n or less satisfying $Q_n(1) = 1$. The existence and definition of such a polynomial follows from the theorem.

If $w(x)$ is a one-to-one mapping function such that

$$w(x) = (2x - M(\mathbf{G}) - m(\mathbf{G}))/(M(\mathbf{G}) - m(\mathbf{G})), \tag{3.26}$$

which is a linear transformation which maps the interval $m(\mathbf{G}) \leq x \leq M(\mathbf{G})$ onto the interval $-1 \leq w \leq 1$. To utilize the results of the theorem a function $H_n(w)$ is defined in terms of Tchebyshev polynomials as

$$H_n(w) = T_n(w(x))/T_n(w(1)). \tag{3.27}$$

If a polynomial function $P_n(x)$ is defined as

$$P_n(n) = T_n\left(\frac{2x - M(\mathbf{G}) - m(\mathbf{G})}{M(\mathbf{G}) - m(\mathbf{G})}\right)/T_n\left(\frac{2 - M(\mathbf{G}) - m(\mathbf{G})}{M(\mathbf{G}) - m(\mathbf{G})}\right) \tag{3.28}$$

then obviously,

$$\max_{m(\mathbf{G}) \leq x \leq M(\mathbf{G})} |P_n(x)| = \max_{-1 \leq w \leq 1} |H_n(w)|. \tag{3.29}$$

It now follows from the theorem that $P_n(x)$ is the polynomial satisfying Equation (3.25). Thus we have the following result:

Theorem 3.2 *Let $\mathbf{S}_N$ be the set of polynomials $\{Q_n(x)\}$ of degree n or less satisfying $Q_n(1) = 1$. Then the polynomial $P_n(x)$ is the unique polynomial in the set $\mathbf{S}_N$ which satisfies*

$$\max_{m(\mathbf{G}) \leq x \leq M(\mathbf{G})} |P_n(x)| \leq \max_{m(\mathbf{G}) \leq x \leq M(\mathbf{G})} |Q_n(x)| \tag{3.30}$$

for any $\mathbf{Q}_n(x) \in \mathbf{S}_N$.

Proof: For the proof and a more detailed discussion of the Tchebyshev algorithm see Hageman and Young [32].□

3.2.2.2 Tchebyshev acceleration - data-reusing methods

Acceleration of the data-reusing methods using the Tchebyshev algorithm requires knowledge of the largest and smallest eigenvalues of $\mathbf{I}-\mathbf{Q}(n)$ or in-line estimators of these quantities. Fortunately, for the symmetric Kaczmarz data-reusing case, the matrix $\mathbf{I} - \mathbf{Q}(n)$ has a well-defined eigenstructure. For the purposes of the Kaczmarz implementation of this algorithm, several theorems governing the distribution of the eigenvalues have been developed.

In order to define the size of the maximum and minimum eigenvalue, a simple theorem gives containment on the eigenvalues. Using this theorem, the exact values of the eigenvalues can be readily proven.

Theorem 3.3 *The matrix* $(\mathbf{I} - \mathbf{Q}(n))$ *has eigenvalues contained in* $[0, 1]$.

Proof: First the matrix $\mathbf{I} - \mathbf{Q}(n)$ is nonnegative definite by the following argument:

$$\begin{aligned} v^H(\mathbf{I} - \mathbf{Q}(n))v = v^H v - v^H \mathbf{Q}(n)v \quad &= \\ ||v||^2 - ||(\mathbf{I} - \mathbf{P}_{L-1}) \cdots (\mathbf{I} - \mathbf{P}_0)v||^2. & \end{aligned} \tag{3.31}$$

This quantity will be greater than or equal to zero if $||v||^2 \geq ||(\mathbf{I} - \mathbf{P}_{L-1}) \cdots (\mathbf{I} - \mathbf{P}_0)v||^2$ $\forall v \in C^N$ with equality if and only if $v \in N(X^H(n)) \bigcap \cdots \bigcap N(X^H(n - L + 1))$. This is clear by observing the following:

$$\begin{aligned} ||(\mathbf{I} - \mathbf{P}_{L-1}) \ldots (\mathbf{I} - \mathbf{P}_0)v||^2 &\leq \\ ||(\mathbf{I} - \mathbf{P}_{L-2}) \ldots (\mathbf{I} - \mathbf{P}_0)v||^2 &\leq \\ &\vdots \\ ||(\mathbf{I} - \mathbf{P}_0)v||^2 &\leq ||v||^2. \end{aligned} \tag{3.32}$$

To show that the matrix $\mathbf{I} - \mathbf{Q}(n)$ has no eigenvalues greater than one, we construct a proof by contradiction. Assume that this matrix has an eigenvalue greater than one so that the following holds:

$$[\mathbf{I} - \mathbf{Q}(n)]v = \lambda v. \tag{3.33}$$

This implies that $(1-\lambda)v = \mathbf{Q}(n)v$. Again, the familiar eigenvalue relationship is visible. We assumed that $\lambda > 1$, yet we discover that for this to be true, $\mathbf{Q}(n)$ must have negative eigenvalues. If we investigate the "definiteness" of this matrix, we find a contradiction:

$$v^H\mathbf{Q}(n)v = ||(\mathbf{I}-\mathbf{P}_{L-1})\cdots(\mathbf{I}-\mathbf{P}_0)v||^2 \geq 0 \tag{3.34}$$

with equality if and only if

$$v \in \{N(X^H(n))\bigcap\cdots\bigcap N(X^H(n-L+1))\}^{\perp}. \tag{3.35}$$

Or equivalently,

$$v \in R(X^H(n))\bigcup\cdots\bigcup R(X^H(n-L+1)). \tag{3.36}$$

Thus, the matrix $\mathbf{Q}(n)$ is positive semidefinite, and the matrix $(\mathbf{I}-\mathbf{Q}(n))$ has no eigenvalues greater than one.$\Box$

Using this result, the values of the smallest and largest eigenvalue can be evaluated.

Lemma 3.1 *The matrix* $(\mathbf{I}-\mathbf{Q}(n))$ *has at least one eigenvalue equal to one.*

Proof: For the matrix $(\mathbf{I}-\mathbf{Q}(n))$ to have an eigenvalue equal to one, the following must hold: $(\mathbf{I}-\mathbf{Q}(n))v = v$. Clearly, if $v = X(n)$, the following will be true:

$$\begin{aligned}[\mathbf{I}-(\mathbf{I}-\mathbf{P}_0)\cdots(\mathbf{I}-\mathbf{P}_{L-1})\cdots(\mathbf{I}-\mathbf{P}_0)]X(n) &= \\ X(n)-(\mathbf{I}-\mathbf{P}_0)\cdots(\mathbf{I}-\mathbf{P}_{L-1})\cdots(\mathbf{I}-\mathbf{P}_0)X(n) &= \\ X(n). &\end{aligned} \tag{3.37}$$

Lemma 3.2 *The matrix* $(\mathbf{I}-\mathbf{Q}(n))$ *has at least one zero eigenvalue if the* $N \geq (L+1)$.

Proof: In order for one zero eigenvalue to exist, $(\mathbf{I}-\mathbf{Q}(n))v = 0$. This is equivalent to $\mathbf{Q}(n)v = v$. Or, the matrix $\mathbf{Q}(n)$ must have an eigenvalue of one. Equivalently, this can occur if $v \in R(\mathbf{I}-\mathbf{P}_0)\bigcap R(\mathbf{I}-\mathbf{P}_1)\bigcap\cdots\bigcap R(\mathbf{I}-\mathbf{P}_{L-1})$. Similarly,

$v \in N(X^H(n)) \bigcap \cdots \bigcap N(X^H(n-L+1))$. In matrix notation this is

$$\begin{bmatrix} X^H(n) \\ X^H(n-1) \\ \vdots \\ X^H(n-L+1) \end{bmatrix} v = 0 \tag{3.38}$$

letting $\mathbf{X(n)} = [X(n), X(n-1), \ldots, X(n-L+1)]$, $v \in N(\mathbf{X}^H(n))$. The dimension of the null space of the matrix $N(\mathbf{X}^H(n))$ is $N-L$ for $N > L$. This implies that the vector v corresponds to a zero eigenvector of the matrix $(\mathbf{I}-\mathbf{Q}(n))$ if and only if $N \geq (L+1)$.□

Based upon the development of these theorems, the Tchebyshev algorithm can be readily implemented. Experiments of the accelerated Kaczmarz data-reusing algorithm using the Tchebyshev algorithm are shown in the next section.

3.3 Connection to Affine Projection Methods

The accelerated data-reusing algorithms shown previously have a strong relationship with the affine projection algorithms discussed in Chapter 1. In Chapter 1, these methods were shown to have an equivalent set of update expressions:

$$W(n) = W(n-1) + \mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}E(n) \tag{3.39}$$

where $E(n) = D(n) - \mathbf{X}^H(n)W(n-1)$ is an $L \times 1$ vector of error conditions. An additional expression for the adaptive filter coefficient update would be given by

$$W(n) = [\mathbf{I} - P_{R(\mathbf{X}(n))}]W(n-1) + [\mathbf{X}^H(n)]^{\dagger}D(n). \tag{3.40}$$

Although the accelerated data-reusing algorithm and the algorithm expressed by Equation (3.40) differ in the method and amount of computation, the algorithms are functionally equivalent by the following lemma and by $[\mathbf{I}-\mathbf{Q}(n)]^{\dagger}D_{\pi}(n) = [\mathbf{X}^H(n)]^{\dagger}D(n)$ [25].

Lemma 3.3 $R(\mathbf{X}(n)) = R(\mathbf{I} - \mathbf{Q}(n))$

Proof: Let $v \in N(\mathbf{I}-\mathbf{Q}(n))$, then, $\mathbf{Q}(n)v = v$. This implies $v \in N(X^H(n)) \bigcap \cdots \bigcap N(X^H(n-L+1))$. Equivalently, this implies that $v \in N(\mathbf{X}^H(n))$, or

$$N(\mathbf{I} - \mathbf{Q}(n)) = N(\mathbf{X}^H(n)). \tag{3.41}$$

Since $\mathbf{C}^N = R(\mathbf{X}(n)) \oplus N(\mathbf{X}^H(n))$, this implies that $R(\mathbf{I} - \mathbf{Q}(n)) = R(\mathbf{X}(n))$.□

Using this result, the accelerated data-reusing methods can be shown to be functionally equivalent to the affine projection-based adaptive filtering algorithms. In an infinite precision environment with either the Tchebyshev or conjugate gradient algorithm used to accelerate either data-reusing method, the convergence behavior will be identical to the affine projection methods.

However, there are several key differences between the two algorithms' approaches to solving essentially the same problem. First, the accelerated data-reusing methods do not compute matrix inverses as the affine projection methods prescribe. In fact, the affine projection methods become more sensitive to the conditioning of the input sequence, as the number of reused data pairs L approaches N. Most of the affine projection methods incorporate regularization in the update of the inverses to compensate for this sensitivity. Unfortunately, this regularization degrades convergence rate performance.

Second, the algorithms differ in terms of computational structure. The computational complexities of the affine projection algorithms has been summarized in Section 3.4 of Chapter 1. The computational complexities of the various implementations vary. Exact implementations of the affine projection algorithms have $O(L^2N)$ for sample-by-sample updates. Block implementations reduce this complexity to $O(LN)$ as the computation becomes distributed over the block. However, the performance degrades as the block size grows. This algorithm also possesses a computational complexity of $O(L^2N)$. However, the computation of this method can be distributed in a number of different ways to exploit the parallel structure of the algorithms.

On a final note, the affine projection methods are also directly related to the Kaczmarz iteration and *row projection* methods. Just as the normalized LMS iteration can be viewed as a *row projection* method (see Section 2.3), the affine projection method can also be viewed as a *row projection* method. The affine projection method is equivalent

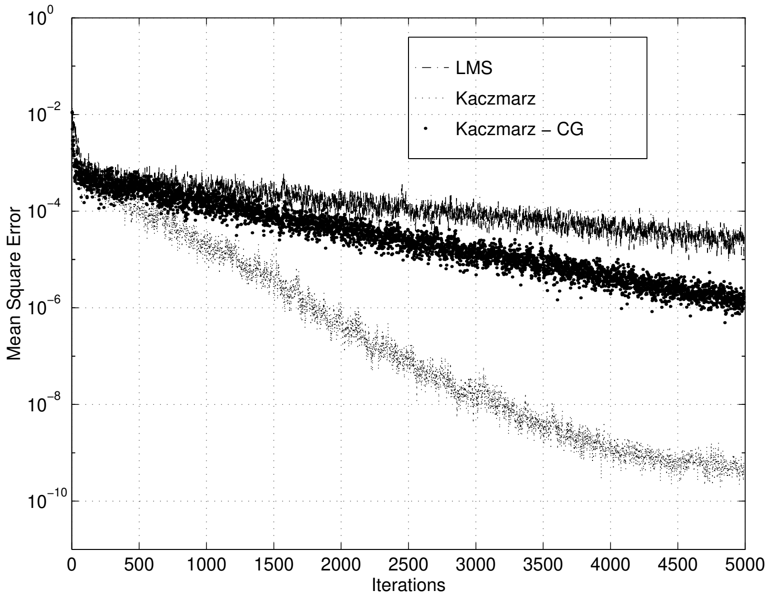


Figure 3.1 MSE Comparison of NLMS, KDR and CG KDR, Colored Noise Input, $L = 3$

to grouping L rows together and forming a projector which is the combination of the L rows.

3.4 Simulation Results

Several simulations were executed to compare the relative performance of the algorithms introduced in this dissertation. The experiments were designed to illustrate the improvements in convergence rate performance that may be obtained by accelerating the data reusing algorithms developed in Chapter 2. Numerous simulations were developed to illustrate the trends or representative behavior of the algorithms. The effects of accelerating the Kaczmarz and Cimmino data reusing algorithms using the conjugate gradient algorithm are explored. The effect of varying the number of steps in the Tchebyshev-based acceleration of the Kaczmarz method is also studied.

For the first example, the same traditional unknown eighth-order system is used. The progression of the mean square error (MSE) for the normalized LMS (NLMS), Kaczmarz data-reusing (KDR), and conjugate gradient accelerated Kaczmarz data-reusing (CG-KDR) displayed in Figure 3.1. For this example, three previous data pairs are reused or $L = 3$, and the NLMS algorithm is slowest to converge. The Kaczmarz algorithm

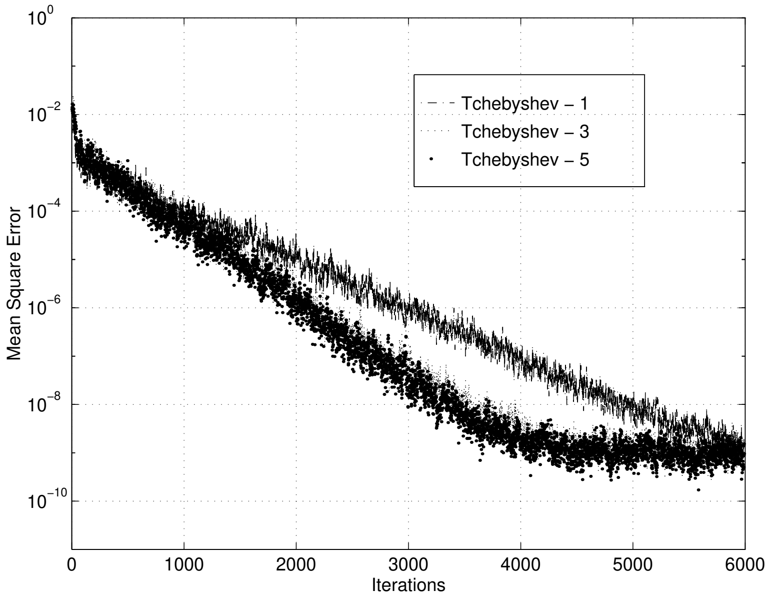


Figure 3.2 MSE Comparison of Tchebyshev-accelerated Kaczmarz Data-Reusing Algorithms, Varying Internal Iterations of the Tchebyshev Algorithm

may be employed to improve convergence by a factor of 2 as shown in [28]. But, convergence is still on the order of 12,000 iterations. Through the use of the conjugate gradient algorithm, convergence is obtained in about 3500 iterations, which is a significant improvement over traditional NLMS.

In Figure 3.2, the Tchebyshev algorithm is used to accelerate the performance of the Kaczmarz data-reusing algorithm. For this scenario, the magnitude of the smallest eigenvalue is estimated to be 10^{-10}. The Tchebyshev accelerated Kaczmarz data-reusing (T-KDR) algorithm is executed a variable number of steps. The resulting progression of the mean square error, shown in Figure 3.2, indicates that the Tchebyshev algorithm can be used to improve KDR performance with just a few iterations of its algorithm. In addition, performance may be made comparable to the conjugate gradient algorithm in about five steps.

The effects of accelerating the Cimmino form of data reusing are investigated in Figure 3.3. Again, the NLMS algorithm is plotted for comparison. The Cimmino-based data-reusing (CDR) scheme does offer some performance improvement over the NLMS scheme. The conjugate gradient accelerated version of CDR seems to offer significant improvement over un-accelerated CDR. CG-CDR appears to converge in about 4000

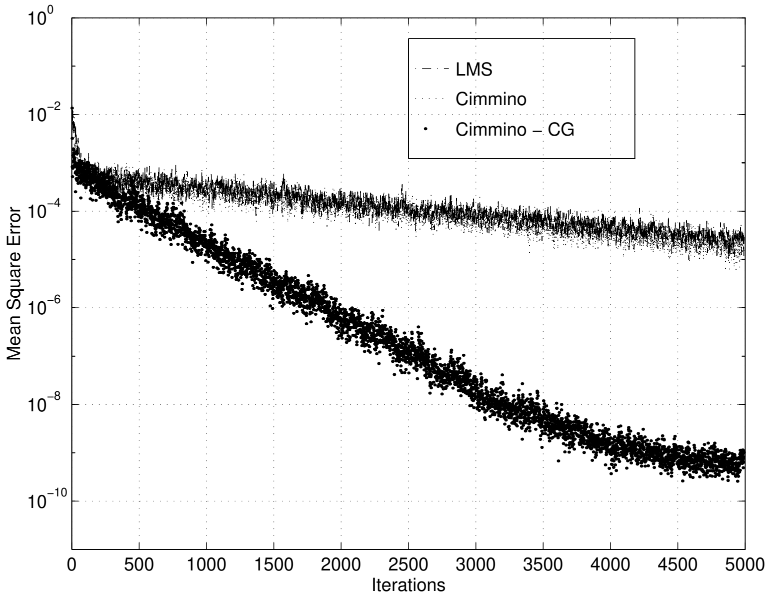


Figure 3.3 Comparison of NLMS, CDR, and CG CDR, Colored Noise Input, $L = 3$

iterations which is comparable in behavior to CG-KDR. Several experiments were performed to contrast and compare the performance of the accelerated data-reusing methods in the fixed-point domain. In the first example, the CG accelerated Kaczmarz algorithm is implemented in 16-bit precision. The resulting mean square error progression for 50 runs of the algorithm is displayed in Figure 3.4. The top half of the figure displays the mean square error progression. The lower half of the figure displays the quantization error associated with the CG algorithm. The output of the floating point version of the CG algorithm is compared to the fixed point version. The error is quite large and appears to be directly related to the noise floor of the algorithm.

In Figure 3.5, the resulting mean square error progression of a fixed point version of the Tchebyshev algorithm is displayed. The difference between the fixed point version of the Tchebyshev algorithm and a floating point version is also displayed. The Tchebyshev algorithm exhibits less quantization noise than the CG algorithm due to the lower number of matrix vector multiplications associated with this algorithm.

Finally, a CG-accelerated Cimmino algorithm is also implemented in 16-bit precision. Again, the algorithm exhibits fairly large noise floor which can be attributed to quantization effects of the fixed point CG algorithm. The error associated with a fixed point

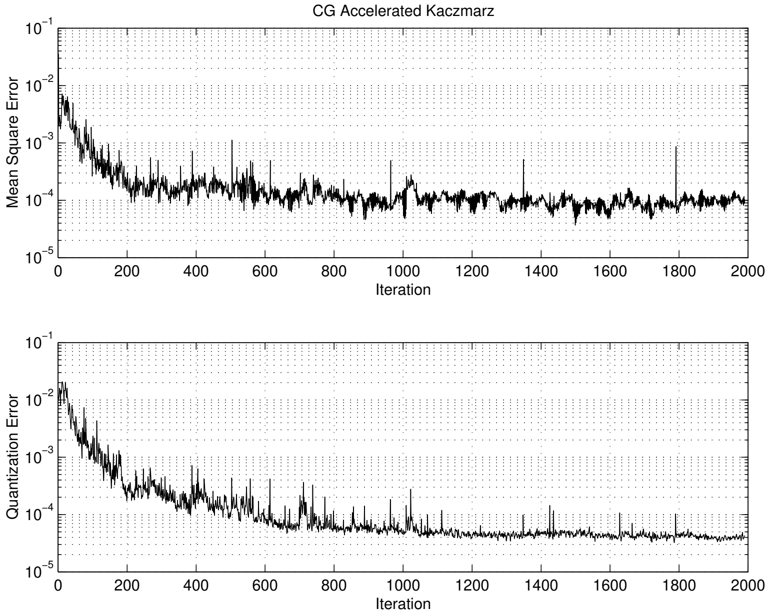


Figure 3.4 CG-Accelerated Kacmarz, 16-bit precision, L = 3

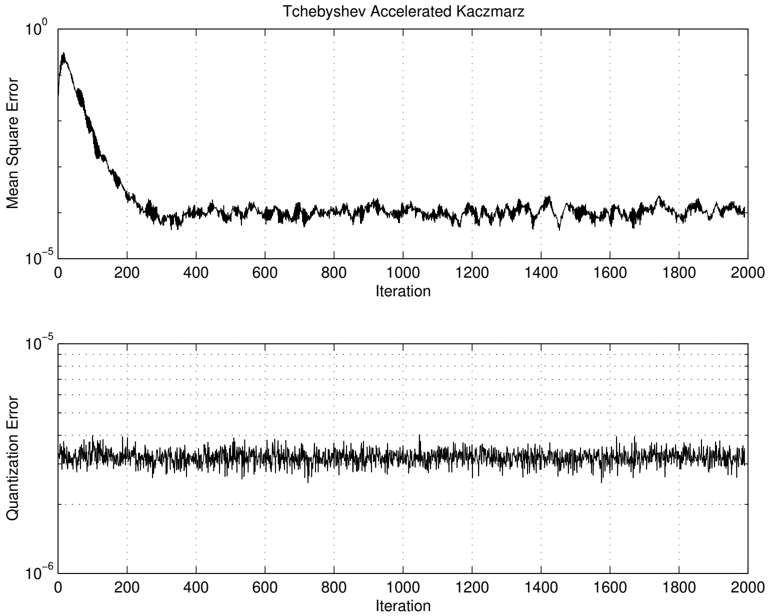


Figure 3.5 Tchebyshev-Accelerated Kaczmarz, 16-bit precision, L = 3, R = 4

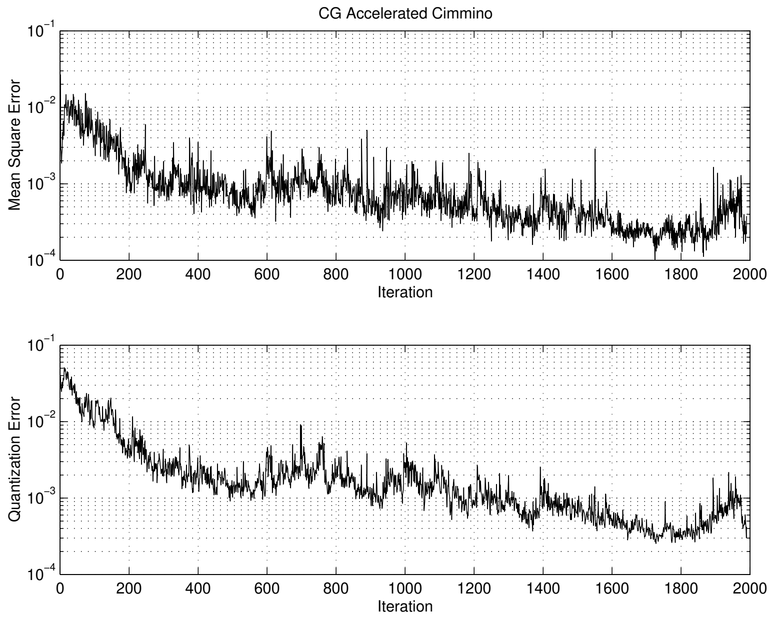


Figure 3.6 CG-Accelerated Cimmino, 16-bit precision, L = 3

implementation of the CG algorithm is displayed in Figure 3.6 along with progression of the mean square error of the fixed point version of the algorithm. The Cimmino accelerated algorithm appears to exhibit the largest degree of misadjustment of the floating point algorithms.

CHAPTER 4

CONVERGENCE ANALYSIS OF ROW PROJECTION METHODS

4.1 Introduction

Analytical convergence analysis of adaptive algorithms is in general a difficult task. Numerous papers have been written about convergence theory even for simple LMS or normalized LMS algorithms. Few if any convergence results exist for the affine projection and data-reusing methods [5–7, 20].

In this chapter, several new convergence results are presented for the *row projection*-based adaptive algorithms. There are several basic elements to convergence theory that concern system designers. There is the transient convergence rate, how quickly the adaptive algorithm reaches the optimal value; steady-state convergence performance, how well the adaptive filter matches the optimal value in the presence of noise; and tracking ability, how well an adaptive algorithm is able to adapt to and track a nonstationary input. These issues are discussed and addressed in this chapter.

Initially, a convergence analysis of the inner-loop of the data-reusing adaptive filtering algorithms is developed. Analysis of data-reusing algorithms has proven to be a difficult exercise [20, 34]. In this dissertation, some results from *row projection* are applied to the data-reusing methods developed here to arrive at some not-so-startling conclusions about the convergence rate performance of the inner loop of the data reusing methods. Primarily, this analysis was developed to determine if the deterministic view of the adaptive filtering operation would lead to some new insights into the convergence operation of the adaptive algorithms. While this analysis provided some insight, it failed to provide the comprehensive convergence analysis that was first envisioned.

For this reason, a traditional convergence analysis was used, employing the popular independence theory assumptions developed by Widrow and Stearns [35]. Utilizing these assumptions, several results are developed for understanding the steady-state convergence performance of the projection algorithms. Some similar results for the affine projection algorithms developed by Slock [7, 8] and Montazeri and Duhamel [5, 6] are included for comparison.

Finally, some simulation results illustrate the strength of the conclusions and the analysis in Sections 4.2 and 4.4. Primarily, the simulations illustrate that the excess mean square error and the misadjustment are increasing functions of the number of reuses and reused data pairs. However, the data-reusing methods are less subject to this effect than the accelerated data-reusing and affine projection algorithms.

4.2 *Row Projection* Convergence Analysis

The convergence behavior of the deterministic *row projection* methods applied to solving linear systems has been well studied [24, 25, 36]. This convergence theory can be put to use in understanding the operation of the adaptive *row projection*-based algorithms. Before applying these results, some background in this area is presented to aid in the understanding some of the key concepts in *row projection* convergence theory. These results are then applied to the methods derived in Chapter 2.

Nelson and Neumann [36] categorize the performance of the *row projection* methods in terms of a single convergence or contraction constant. To define this constant, some basic definitions are needed. First, Nelson and Neumann [36] defines the subdominant or controlling eigenvalue of a matrix $\mathbf{B}$ as

$$\gamma(\mathbf{B}) \doteq max\left\{|\lambda| \ni \lambda \in 0 \cup \sigma(\mathbf{B})\backslash\{1\}\right\}, \tag{4.1}$$

where $\sigma(\mathbf{B})$ represents the spectrum of $\mathbf{B}$. Second, an $N \times N$ matrix $\mathbf{B}$ is called paracontracting if the spectral norm of the matrix $\mathbf{B}$ is bounded by unity and if

$$0 \neq x \in [N(\mathbf{I} - \mathbf{B})]^{\perp} \Rightarrow ||\mathbf{B}x||_2 < ||x||_2. \tag{4.2}$$

Third, if $\mathbf{B}$ is paracontracting, then the contraction constant which governs the overall convergence rate of *row projection* methods may be defined as

$$c(\mathbf{B}) \doteq \inf\left\{c \in [0,\infty) | \forall x \in [N(\mathbf{I}-\mathbf{B})]^{\perp} ||\mathbf{B}x||_2 \leq c||x||_2\right\}. \tag{4.3}$$

The paracontraction constant satisfies $0 \leq c(\mathbf{B}) < 1$.

If $\mathbf{B}$ is Hermitian then $c(\mathbf{B}) = \gamma(\mathbf{B})$. In particular, $c(\mathbf{P}) = 0$ for any orthogonal projection $\mathbf{P}$. More generally, the paracontracting constant is given by $c(\mathbf{B}) = |1-\omega|$ if $\mathbf{B} = \mathbf{I} - \omega(\mathbf{I}-\mathbf{P})$,$0 < \omega < 2$ and $\mathbf{P} \neq \mathbf{I}$ is an orthogonal projection.

Additionally, the matrix $\mathbf{B}$ is called *semiconvergent* if the powers of $\mathbf{B}$ converge [37]. According to a characterization due to Oldenburger [37], $\mathbf{B}$ is semiconvergent if and only if the following conditions hold:

(i) $N(\mathbf{I}-\mathbf{B})$ and $R(\mathbf{I}-\mathbf{B})$ are complementary subspaces of the the n-dimensional complex subspace C^N, and

(ii) $\gamma(\mathbf{B}) < 1$.

Referring to (i) and (ii), the paracontracting matrices are semiconvergent. Additionally, if x is an eigenvector of $\mathbf{B}$ corresponding to the subdominant eigenvalue, then $x \in R(\mathbf{I}-\mathbf{B})$ and $||\mathbf{B}x||_2 = \gamma(\mathbf{B})||x||_2$. Hence,

$$\gamma(\mathbf{B}) \leq c(\mathbf{B}) < 1. \tag{4.4}$$

In general, the procedure to evaluate the contraction constant of a *row projection* applied to a linear system problem can be difficult. Nelson and Neumann [36] developed a theorem bounding the contraction constants which will be put to use in the development of the understanding of the convergence behavior of the data-reusing methods.

Theorem 4.1 *Let* $\mathbf{B} = \mathbf{B}_k \ldots \mathbf{B}_1$ *be the product of* k *paracontracting matrices* $\mathbf{B}_i \in C^{n,n}$. *Then* $\mathbf{B}$ *is paracontracting. Furthermore,*

$$N(\mathbf{I}-\mathbf{B}) = \bigcap_{i=1}^{k} N(\mathbf{I}-\mathbf{B}_i) \tag{4.5}$$

and

$$c(\mathbf{B}) \leq \left\{ 1 - \prod_{i=1}^{k}[1 - c^2(\mathbf{B}_i)] \prod_{i=1}^{k-1} sin^2(\theta_i) \right\}, \tag{4.6}$$

where θ_i denotes the angle between the subspaces $N(I - \mathbf{B_I})$ and $\bigcap_{j=i+1}^{k} N(\mathbf{I} - \mathbf{B}_j)$. In particular, if $\gamma(\mathbf{B}_i) = c(\mathbf{B}_i)$ for $i = 1, \ldots, k$ then

$$\gamma(\mathbf{B}) \leq \left\{ 1 - \prod_{i=1}^{k}[1 - \gamma^2(\mathbf{B}_i)] \prod_{i=1}^{k-1} sin^2(\theta_i) \right\}. \tag{4.7}$$

Proof: See Nelson and Neumann [36]. □

4.3 Applications to *Row Projection*-based Adaptive Methods

The results of the previous section can be applied to gain more insight into the *row projection*-inspired adaptive filtering algorithms. As was well-described in Chapter 2, these adaptive methods solve low-rank systems of linear equations. These linear systems of equations are given by

$$\mathbf{X}^H(n)W(n) = D(n). \tag{4.8}$$

In addition, the algorithm satisfies an additional constraint that the distance between successive iterates of the adaptive filtering algorithm is minimized:

$$\min_{W(n)} ||W(n) - W(n-1)||_2^2. \tag{4.9}$$

The *row projection* algorithms solving this constrained set of linear equations may be expressed as

$$W(n+1) = \overline{\mathbf{Q}}(n)W(n) + \overline{D}_\pi(n), \tag{4.10}$$

where $\overline{\mathbf{Q}}(n) = \mathbf{Q}_K(n)$ or $\mathbf{Q}_C(n)$, and $\mathbf{D}_\pi(n) = \mathbf{D}_\pi^K(n)$ or $\mathbf{D}_\pi^C(n)$, where the superscripts represent the Kaczmarz or Cimmino form of data reusing, respectively. Reusing data to a factor R can be thought of as holding the matrix $\overline{\mathbf{Q}}(n)$ and the vector $\overline{D}_\pi(n)$ constant for R iterations. For the Kaczmarz algorithm, the matrix $\mathbf{Q}_K(n)$ may be expressed as

$$\mathbf{Q}_K(n) = (\mathbf{I} - \mathbf{P}_{X(n)}) \cdots (\mathbf{I} - \mathbf{P}_{X(n-L+1)}), \tag{4.11}$$

and the vector $D_\pi^K(n)$ can be expressed as

$$D_\pi^K(n) = [X(n)^H]^\dagger + \sum_{i=1}^{L-1}\prod_{j=1}^{i}(\mathbf{I} - \mathbf{P}_{X(n-j)})[X(n-i)^H]^\dagger. \tag{4.12}$$

For the Cimmino algorithm, the matrix $\mathbf{Q}_C(n)$ may be expressed as

$$\mathbf{Q} = \left(\mathbf{I} - \frac{2}{L}\sum_{i=0}^{L-1}\mathbf{P}_i\right), \tag{4.13}$$

and the vector $D_\pi^C(n)$ can be expressed as

$$D_\pi^C(n) = \frac{2}{L}\sum_{i=0}^{L-1}[X^H(n-i)]^\dagger d(n-i). \tag{4.14}$$

The matrix $\mathbf{Q}_K(n)$ is the product of L paracontracting matrices and is therefore paracontracting. The matrix $\mathbf{Q}_C(n)$ is similarly paracontracting, but the proof is omitted here for brevity. Hence, the convergence of the data-reusing method for variable reusing factor R is controlled by the angle between the subspaces $N(\mathbf{I} - (\mathbf{I} - \mathbf{P}_{X(n-i)}))$ and $\bigcap_{j=i+1}^{L-1} N(\mathbf{I} - (\mathbf{I} - \mathbf{P}_{X(n-j)}))$.

While this is a useful insight into the convergence behavior, further interpretation is warranted in terms of the range space of the individual vectors $X(n)$. First, a useful relationship is needed to relate this unusual null space $N(\mathbf{I} - (\mathbf{I} - \mathbf{P}_{X(n-i)}))$ to the range space of $X(n)$.

Lemma 4.1

$$\begin{aligned} N(\mathbf{I} - (\mathbf{I} - \mathbf{P}_{R(X(n))})) &= N(\mathbf{P}_{R(X(n)}) && (4.15)\\ &= N(X^H(n)). && (4.16)\end{aligned}$$

Proof: $x \in N(\mathbf{P}_{R(X(n))}) \Rightarrow \mathbf{P}_{R(X(n))}x = 0$. This statement implies that $x \in R(X(n))^\perp$ and $R(X(n))^\perp = N(X^H(n))$. $\square$

Using this lemma, the rate of convergence for variable R is determined to be controlled by the smallest angle θ_i between the $N(X^H(n-i))$ and $\bigcap_{j=i+1}^{L-1} N(X^H(n-j))$. Further discussion on how to compute these angles can be obtained in Bjorck and Golub [38].

An upper bound on the angles θ_i's can be evaluated for a simple case of $L = 2$ and $N = 2$. The rate of convergence for this case is upper-bounded by the sine of the angle between $N(X^H(n))$ and $N(X^H(n-1))$. The cosine of the angle between these two null spaces can be defined as [36, 38]

$$\cos\theta = \frac{X^H(n)X(n-1)}{\|X(n)\|\|X(n-1)\|}, \tag{4.17}$$

where the row-vector $X^H(n)$ is assumed to form a basis for $N(X^H(n))$ and the row-vector $X^H(n-1)$ is assumed to form a basis for $N(X^H(n-1))$. This is reasonable as long as the two rows of data $X^H(n)$ and $X^H(n-1)$ are linearly independent.

For the limited case that the two planes are colinear, θ is equal to 0. Equivalently, the sine of the angle is 0. This implies that the contraction constant is 1 or as large as it can be. When $X(n)$ and $X(n-1)$ are orthogonal, the angle θ is $90°$, and the contraction constant c is 0 or as small as it can be. Intuitively, this orthogonal case corresponds to rapid convergence, and the colinear case corresponds to slow convergence and a large contraction constant.

This result can further be interpreted in a statistical fashion by taking the expectation of the relationship given in Equation (4.17):

$$\begin{aligned} E[\cos\theta] &= E[\frac{X^H(n)X(n-1)}{\|X(n)\|\|X(n-1)\|}] \\ &\approx \rho(n, n-1), \end{aligned} \tag{4.18}$$

where $\rho(n, n-1)$ is the correlation between successive samples. The expression for $\rho(n, n-1)$ is given by

$$\rho(n, n-1) = \frac{E[x(n)x(n-1)]}{\sqrt{E[x(n)^2]E[x(n-1)^2]}}. \tag{4.19}$$

This is intuitively satisfying. The correlation between successive samples controls the convergence behavior of the data-reusing method for variable R. Highly correlated data implies θ is close to 0. Uncorrelated or no correlation between successive samples implies that θ is close to $\pi/2$.

4.3.1 Extension to higher-order case

While the bounds on the convergence rate developed for the case where $L = N = 2$ sheds some light on convergence behavior, further study is warranted for arbitrary L and N. For general L, the analysis can begin with the first angle θ_1. This is the principle angle between the $N(X^H(n))$ and the $N(X^H(n-1)) \bigcap \cdots \bigcap N(X^H(n-L+1))$. The principle angle between these null spaces will be zero if the two intersect. The intersecting null space, $N(X^H(n)) \bigcap N(X^H(n-1) \bigcap \cdots \bigcap N(X^H(n-L+1))$, can be examined to get further insight. Finding the intersection of these different null spaces is equivalent to finding a solution to the following equation:

$$\begin{bmatrix} X^H(n) \\ X^H(n-1) \\ \vdots \\ X^H(n-L+1) \end{bmatrix} v = 0. \tag{4.20}$$

Equivalently, if this system of equations has a solution, this implies that $N(X^H(n))$ intersects $N(X^H(n-1)) \bigcap \cdots \bigcap N(X^H(n-L+1))$. Further if this intersection exists, then the angle θ_1 is equal to zero. Clearly, this system will have a null space if and only if $N > L$ (this assumes that all received rows are linearly independent). Furthermore, all of the angles θ_i will be zero if $N > L$. This implies that the upper bound on the contraction constant c for $N > L$ can be evaluated for the forward-backward data-reusing method as 1. Or equivalently, the transformed data matrix $\mathbf{Q}(n)$ has the following property:

$$||\mathbf{Q}(n)W(n-1)||_2 \leq ||W(n-1)||_2. \tag{4.21}$$

This is similar to the norm-reducing property of the projection matrix, but the matrix $\mathbf{Q}(n)$ is not a projection matrix.

While the theorem gives an upper bound on the contraction constant c and guarantees that the inner loop of the algorithm converges, a tighter bound on the contraction constant may be determined. Restating the definition for the contraction constant in terms of the matrix $\mathbf{Q}(n)$, the following is obtained:

$$c(\mathbf{Q}(n)) \doteq \inf \left\{ c \in [0, \infty] | \forall x \in [N(\mathbf{I} - \mathbf{Q}(n))]^{\perp}, ||\mathbf{Q}(n)x||_2 \leq c||x||_2 \right\}. \tag{4.22}$$

Letting $y \in N(\mathbf{I} - \mathbf{Q})$, then the following can be shown:

$$(\mathbf{I} - \mathbf{Q}(n))y = 0 \rightarrow \mathbf{Q}(n)y = y. \tag{4.23}$$

Further, if $\mathbf{Q}(n)y = y$, this implies that $y \in N(X^H(n)) \bigcap \cdots \bigcap N(X^H(n - L + 1)) = N(\mathbf{X}^H(n))$. More directly, using the fundamental theorem of linear algebra, $\mathbf{C}^N = R(A) \bigoplus N(A^H)$, it is possible to show $[N(\mathbf{I} - \mathbf{Q}(n))]^{\perp} = R(\mathbf{X}(n))$. Using the definition of $c(\mathbf{Q}(n))$, the value of c may be evaluated directly $\forall x \in R(\mathbf{X}(n))$. For any and all such x, the following will hold:

$$\mathbf{Q}(n)x = 0. \tag{4.24}$$

Thus, the value of $c(\mathbf{Q}(n))$ will be zero.

Note that in the case where $L = 1$, and $N = 2$, the intersection of the null spaces $N(X^H(n))$ and $N(X^H(n-1))$ does not exist, so this result does not apply to the result which was determined earlier. Further, this result does not apply to the case where $L \geq N - 1$.

This result seems to indicate that this contraction constant developed for the *row projection*-inspired adaptive data-reusing algorithms does not offer any insight into the convergence behavior of the inner loop of these algorithms. Further, this constant may not be an accurate predictor of performance of the algorithm. However, it does provide some insight into the operation for the simple case.

4.4 Steady-State Statistical Convergence Analysis

The deterministic convergence analysis was only able to offer limited insight into the performance of the adaptive filtering algorithms. Further convergence study is warranted. Most importantly, can the algorithms be said to converge? In what sense? Do they converge in the mean? Do they converge in the mean square? What is the variance of the solutions? Sometimes, the steady state variance in the solutions is referred to as *misadjustment* [1]. Misadjustment is defined as the excess mean square error [1]. The affine projection algorithms described in the literature (See Section 1.3) are well known to possess some fairly large misadjustment or large excess mean square error [5–8].

In this section, some basic background concerning the independence theory is presented to help guide the reader through the development of the convergence results for the projection and data-reusing algorithms. These basic assumptions about the independence of data are used to develop some new results about the convergence behavior of the algorithms developed and discussed in this dissertation.

4.4.1 Independence theory assumptions

Statistical analysis of the LMS and RLS algorithms is based upon four basic assumptions concerning the independence of data. Primarily, these four points, which are summarized in Haykin [1], are:

- The tap-input vectors $X(1), X(2), \ldots, X(N)$ constitute a sequence of statistically independent vectors.
- At time n, the tap-input vector $X(n)$ is statistically independent of all previous samples of the desired response, namely, $d(1), d(2), \ldots, d(n-1)$.
- At time n, the desired response $d(n)$ is dependent on the corresponding tap-input vector $X(n)$, but statistically independent of all previous samples of the desired response.
- The tap-input vector $X(n)$ and the desired response $d(n)$ consist of *mutually Gaussian-distributed* random variables for all n.

4.4.2 Fallacies in the independence theory

Unfortunately, the assumptions about the independence of the data do not always hold. However, the use of these assumptions has been shown to be an accurate tool in the analytical prediction of convergence performance of numerous adaptive filtering algorithms [1].

The first item listed does not hold due the tapped-delay line nature of the FIR filter. The sequence of vectors $X(n)$ is statistically dependent due to the shifting property of the input data sequence $x(n)$. The vector $X(n)$ is given by

$$X(n) = \left[\begin{array}{cccc} x(n), & x(n-1), & \ldots, & x(n-N+1) \end{array} \right]. \tag{4.25}$$

The vector $X(n-1)$ is given by

$$X(n) = \left[\begin{array}{cccc} x(n+1), & x(n), & \ldots, & x(n-N) \end{array} \right]. \tag{4.26}$$

Clearly, the vectors $X(n)$ and $X(n+1)$ are statistically dependent.

In fact, these assumptions do not hold for a wide variety of inputs. However, the results developed using these assumptions match the experimental behavior of the algorithms studied. The results summarized here are not intended to provide concrete proof of convergence, but offer some interesting insights into the behavior of the data-reusing algorithms.

4.4.3 Convergence in the mean of the projection algorithms

A sufficient and necessary condition for convergence in the mean of the projection algorithm is

$$E[\epsilon(n)] = E[W(n) - W_0(n)] \to 0, \tag{4.27}$$

where $W_0(n)$ represents the optimal Wiener-Hopf solution. The evolution of the error between the adaptive weights and the optimal weights $W_0(n)$ using Equation (4.10) in generic form for $R = 1$ is

$$\epsilon(n) = \mu\overline{\mathbf{Q}}(n)W(n-1) + \overline{D}_\pi(n), \tag{4.28}$$

where μ has been added to the update expression for the coefficients and the evolution of the error to aid in the proof of convergence. The evolution of the error may be expressed for arbitrary reuse factor R as

$$\epsilon(n) = \tilde{\mathbf{Q}}(n)W(n-1) + \tilde{D}_\pi(n), \tag{4.29}$$

where the matrix $\tilde{\mathbf{Q}}(n)$ is given by

$$\tilde{\mathbf{Q}}(n) = [\mu\overline{\mathbf{Q}}(n)]^R, \tag{4.30}$$

and the vector $\tilde{D}_\pi(n)$ may be expressed as

$$\tilde{D}_\pi(n) = \sum_{i=0}^{R-1}[\mu\overline{\mathbf{Q}}(n)]^i\overline{D}_\pi(n). \tag{4.31}$$

For convergence in the mean, two necessary and sufficient conditions are required:

- The sequence, $\tilde{D}_\pi(n)$ must be zero mean.
- The eigenvalues of the matrix $\tilde{\mathbf{Q}}(n)$ must be contained in the interval [0,1).

For both the Kaczmarz and Cimmino matrices, the vector $\tilde{D}_\pi(n)$ is a linear function of the input sequence $d(n)$. Therefore, the sequence $\tilde{D}_\pi(n)$ must be zero mean if the sequence $d(n)$ is zero mean. For the Kaczmarz algorithm, the matrix $\mathbf{Q}_K(n)$ was shown to have eigenvalues contained in the interval $[0,1]$ for the symmetric case. A similar proof may be developed for the nonsymmetric case, but the proof is omitted for brevity. The choice of $\mu < 1$ satifies the second condition for an arbitrary value of R. Thus, the Kaczmarz algorithm is convergent in the mean. For the Cimmino algorithm, the eigenvalues of the matrix $\mu\mathbf{Q}_C(n)$ must be contained in the interval $[0,1)$. Further, the matrix $\tilde{\mathbf{Q}}(n)$ must be contained in the interval $[0,1)$. The following theorem aids in the proof of convergence of the Cimmino form of data reusing.

Theorem 4.2 *The eigenvalues of the matrix $\mathbf{Q}_C(n)$ are contained in the interval $[0,1]$.*

Proof: By symmetry arguments, the matrix $\mathbf{Q}_C(n)$ clearly has eigenvalues greater than or equal to zero. To prove that the eigenvalues are less than or equal to one, the matrix

$\mathbf{Q}_C(n)$ must be re-expressed as

$$\mathbf{Q}_C(n) = \frac{1}{L}\sum_{i=0}^{L-1}[\mathbf{P}_{N(\mathbf{X}^H(n-i))} - \mathbf{P}_{R(\mathbf{X}(n-i))}]. \tag{4.32}$$

The spectral norm σ_S of the matrix $\mathbf{Q}_C(n)$ may be computed as

$$\sigma_S = ||\mathbf{Q}_C(n)||_S = \max_{z\neq 0}\frac{|\mathbf{Q}_C(n)z|}{z}. \tag{4.33}$$

Without loss of generality, this norm may be computed as

$$\begin{array}{cc} \max\limits_{z} & |\mathbf{Q}_C(n)z| \\ \textit{subject to} & ||z|| = 1. \end{array} \tag{4.34}$$

The spectral norm of the matrix $\mathbf{Q}_C(n)$ may be upper-bounded in the following way:

$$||\frac{1}{L}\sum_{i=0}^{L-1}[\mathbf{P}_{N(\mathbf{X}^H(n-i))} - \mathbf{P}_{R(\mathbf{X}(n-i))}]||_S \leq \frac{1}{L}\sum_{i=0}^{L-1}||\mathbf{P}_{N(\mathbf{X}^H(n-i))} - \mathbf{P}_{R(\mathbf{X}(n-i))}]||_S. \tag{4.35}$$

The generalized triangle equality has been used to move the sum outside the norm operator. The norm inside the sum may be evaluated by re-examining the quantity

$$||\mathbf{P}_{N(\mathbf{X}^H(n))} - \mathbf{P}_{R(\mathbf{X}(n))}]||_S = \max_{z}|\mathbf{P}_{N(\mathbf{X}^H(n))}z - \mathbf{P}_{R(\mathbf{X}(n))}z|. \tag{4.36}$$

This quantity is maximized for the choice that $z \in R(\mathbf{X}(n))$ or that $z \in N(\mathbf{X}^H(n))$. Either choice will lead to the direct evaluation of the quantity expressed in Equation (4.36) as unity. Hence, the spectral norm of the matrix $\mathbf{Q}_C(n)$ may be expressed as

$$||\mathbf{Q}_C(n)||_S \leq \frac{1}{L}\sum_{i=0}^{L-1}1 = 1. \tag{4.37}$$

Therefore, the spectral radius of the matrix $\mathbf{Q}_C(n)$ is contained in the interval $[0, 1]$ $\square$.

To obtain convergence in the mean for the Cimmino method, the scaling factor μ must be less than one. Using this choice of μ, and the results of the theorem, it is possible to conclude that the Cimmino form of data reusing converges in the mean.

4.4.4 Convergence of the projection methods in the mean square

While the proof of convergence in the mean is an important yet fairly weak form of convergence, a better characterization of the variation in the behavior of the tap weights is needed. Hence, the behavior of the covariance matrix $E[\epsilon(n)\epsilon^H(n)] = \mathbf{K}(n)$ of the deviation of the optimal error is very critical. For the *row projection*-based data-reusing algorithms, the following expression describes the evolution of the covariance matrix:

$$\mathbf{K}(n) = \tilde{\mathbf{Q}}(n)\mathbf{K}(n-1)\tilde{\mathbf{Q}}(n) + E[D_\pi(n)D_\pi^H(n)]. \tag{4.38}$$

The cross products associated with the update of the covariance matrix have been eliminated by using the assumptions about the independence of data. These cross-products are

$$E[D_{\pi(n)}\epsilon^H(n-1)\tilde{\mathbf{Q}}^H(n)] = 0. \tag{4.39}$$

For mean square convergence of the adaptive algorithms developed in this dissertation, the following must hold [5, 6]:

$$\|\tilde{\mathbf{Q}}(n)\mathbf{K}(n-1)\tilde{\mathbf{Q}}^H(n)\|_2^2 \leq \|\mathbf{K}(n-1)\|_2^2. \tag{4.40}$$

For both the Kaczmarz and Cimmino forms of data reusing, the matrices $\tilde{\mathbf{Q}}(n)$ have eigenvalues in the interval $[0, 1]$. For convergence in the mean square, the constant μ must be chosen so that $0 < \mu < 1$. With this choice of μ, the data-reusing algorithm converges in the mean square.

4.4.5 Learning curves of the projection and data-reusing algorithms

The expressions for the covariance matrix can be used to develop approximate expressions for the learning curves for the data-reusing algorithms. For the data-reusing and affine projection algorithms, the learning curve may be defined in terms of the mean square of the norm of the *a-posteriori* errors. This performance function can be defined as follows [5–8]:

$$J'(n) = E[||E_f(n)||^2] = E[||D(n) - \mathbf{X}^H(n)W(n)||^2], \tag{4.41}$$

where $E_f(n)$ represents a vector of the *a-posteriori errors*. Evaluation of the expectation given in Equation (4.41) leads to the following:

$$\begin{aligned} J'(n) &= E[||E_f(n)||^2] = E[E_f(n)^H E_f(n)] = E[tr(E_f(n)E_f(n)^H)] \\ &= E[tr(D(n) - \mathbf{X}^H(n)\epsilon(n-1))(D(n) - \mathbf{X}^H(n)\epsilon(n-1))^H)] \\ &= tr(E[D(n)D^H(n)]) + tr(E[\mathbf{X}^H(n)\epsilon(n-1)\epsilon(n-1)\mathbf{X}(n)]) \\ &= N\sigma_d^2 + tr[\mathbf{X}^H(n)\mathbf{K}(n-1)\mathbf{X}(n)]. \end{aligned} \tag{4.42}$$

Utilizing the relationship derived earlier for $\mathbf{K}(n)$, the mean square error becomes:

$$J'(n) \approx N\sigma_d^2 + tr[\mathbf{X}^H(n)E[\tilde{D}_\pi(n-1)\tilde{D}_\pi^H(n-1)]\mathbf{X}(n)], \tag{4.43}$$

where the driving term in the recurrence is assumed to dominate for the covariance matrix $\mathbf{K}(n)$ evolution [5, 6]. More specifically,

$$\mathbf{K}(n) \approx E[\tilde{D}_\pi(n)\tilde{D}_\pi^H(n)]. \tag{4.44}$$

For the Kaczmarz data-reusing algorithm, the matrix $\mathbf{K}(n)$ evaluates to the following:

$$\begin{aligned} \mathbf{K}(n) &\approx [\sum_{l=0}^{R-1} \mathbf{Q}_K^l(n)]E[D_\pi^K(n)(D_\pi^K(n))^H][\sum_{l=0}^{R-1} \mathbf{Q}_K^l(n)]^H \\ &= \sigma_d^2 \mathbf{Q}_n^R \sum_{i=0}^{L-1} \prod_{j=0}^{i-1} \mathbf{P}_j [X^H(n-i)]^\dagger \{[X^H(n-i)]^\dagger\}^H \prod_{l=i-1}^{0} \mathbf{P}_l [\mathbf{Q}_n^R]^H \\ &= \sigma_d^2 \mathbf{Q}_n^R \sum_{i=0}^{L-1} \prod_{j=0}^{i-1} \mathbf{P}_j \frac{\mathbf{P}_i^\perp}{||X(n-i)||^2} \prod_{l=i-1}^{0} \mathbf{P}_l [\mathbf{Q}_n^R]^H, \end{aligned} \tag{4.45}$$

where $\mathbf{P}_j = \mathbf{P}_{N(X^H(n-j))}$ and $\mathbf{Q}_n^R$ is a polynomial of the matrix $\mathbf{Q}(n)$. Applying the result in Equation (4.45) yields the following expression for the learning curve:

$$J_K(n) \approx N\sigma_d^2 + \sigma_d^2 \sum_{m=0}^{L-1} || \sum_{i=0}^{L-1} \frac{\mathbf{P}_i^{\perp}}{||X(n-i)||} \prod_{l=i-1}^{0} \mathbf{P}_l [\mathbf{Q}_n^R]^H X(n-m)||^2. \qquad (4.46)$$

For the Cimmino data-reusing case, this result may be similarly computed:

$$\begin{aligned} \mathbf{K}(n) &\approx \sigma_d^2 \mathbf{Q}_n^R \sum_{i=0}^{L-1} [X^H(n-i)]^{\dagger}([X^H(n-i)]^{\dagger})^H\}[\mathbf{Q}_n^R]^H \\ &= \sigma_d^2 \mathbf{Q}_n^R \sum_{i=0}^{L-1} \frac{\mathbf{P}_i^{\perp}}{||X(n-i)||^2} [\mathbf{Q}_n^R]^H, \end{aligned} \qquad (4.47)$$

where $\mathbf{Q}_n^R = \sum_{i=0}^{R-1} \mathbf{Q}_c^i(n)$. Again, the same observations can be made; as L and R increase, so does the amplification of the term σ_d^2. Similarly, the Cimmino learning curve may be expressed as the following:

$$J_C(n) \approx N\sigma_d^2 + \sigma_d^2 \sum_{m=0}^{L-1} || \sum_{i=0}^{L-1} \frac{\mathbf{P}_i^{\perp}}{||X(n-i)||} [\mathbf{Q}_n^R]^H X(n-m)||^2. \qquad (4.48)$$

For both the Kaczmarz and Cimmino data-reusing methods, the measurement noise σ_d gets amplified as the number of reuses R and the number of reused data pairs L are increased. Approximate expressions for the learning curves for the affine projection algorithms were developed by Montazeri and Duhamel [5, 6] and Slock [7, 8]. The covariance matrix $\mathbf{K}(n)$ can be approximated for the affine projection methods:

$$\mathbf{K}(n) \approx [\mathbf{X}(n)[\mathbf{X}^T(n)\mathbf{X}(n)]^{-2}\mathbf{X}^T(n)]. \qquad (4.49)$$

In this expression, the covariance matrix is dependent upon the conditioning of the matrix $[\mathbf{X}^T(n)\mathbf{X}(n)]^{-2}$. The learning curve is similarly derived as before:

$$J_{AP}(n) \approx N\sigma_d^2 + \mu^2 N\sigma_d^2 tr[\mathbf{X}^T(n)\mathbf{X}^T(n-1)[\mathbf{X}^T(n-1)\mathbf{X}(n-1)]^{-2}\mathbf{X}^T(n-1)\mathbf{X}(n)]. \qquad (4.50)$$

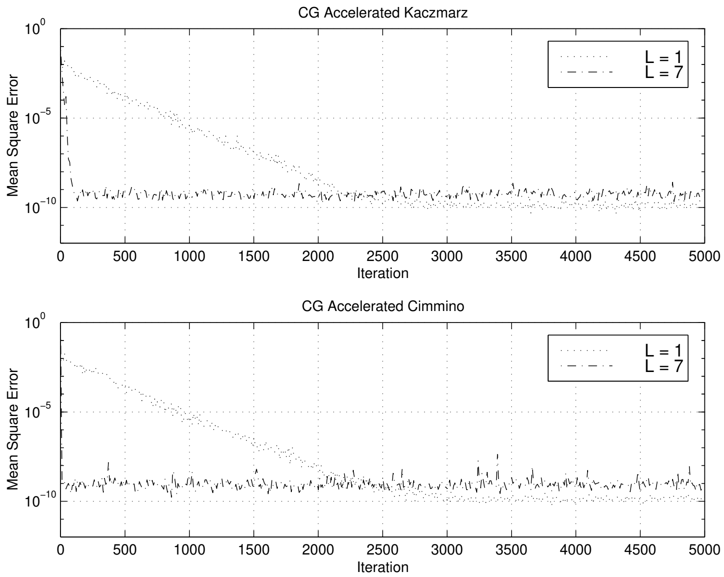


Figure 4.1 Mean Square Error Progressions, Kaczmarz and Cimmino Accelerated Data-Reusing Algorithms

In this expression, the learning curve has a strong dependence on the conditioning of the matrix $[\mathbf{X}^T(n)\mathbf{X}(n)]$. As the number of reused pairs L grows larger, the matrix becomes more ill-conditioned. Further, the excess mean square error or second term in the expression for the learning curve grows larger. This happens as the data becomes more correlated. The data-reusing methods are not as vulnerable to this effect.

4.5 Experimental Verification

Several experiments were designed to illustrate that the derived expressions for the learning curves will predict the relative performance of the data-reusing and affine projection algorithms. For a base of comparison, the accelerated data-reusing algorithms, which have similar performance to the affine projection algorithms, were used to identify an eighth-order system which was excited with colored noise. The resulting progression of the mean square error is displayed in Figure 4.1. For this experiment, the effect of varying the window size was seen to have great impact on the excess mean square error. For a window size $L = 7$, the misadjustment has risen to 10^{-8} above a 10^{-10} noise floor.

The experiment was repeated with the data-reusing methods, and the resulting progression of the mean square error is displayed in Figure 4.2. For a window size of $L = 7$

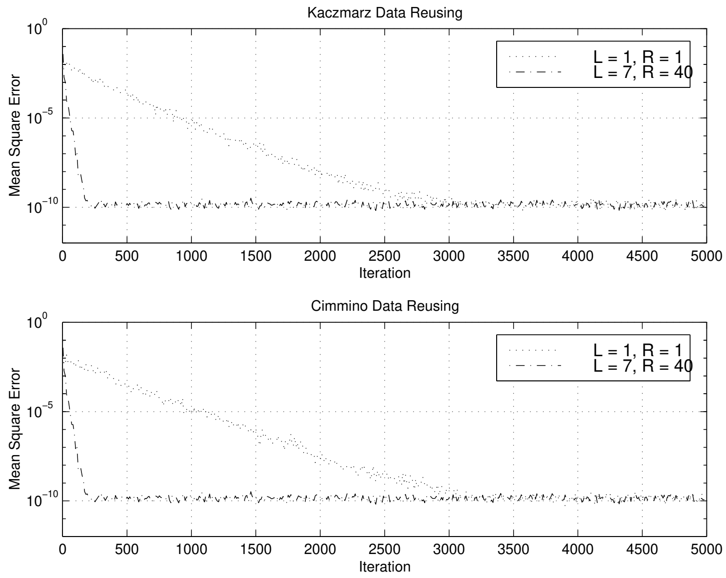


Figure 4.2 Mean Square Error Progressions, Kaczmarz and Cimmino Data-Reusing Algorithms

and a reuse factor $R = 40$, the excess mean square error is negligible and comparable in performance to a window size of $L = 1$. To illustrate that the data-reusing algorithms were capable of displaying large misadjustments when the reusing factor and the window size are increased, an experiment was performed which illustrated the algorithm was capable of amplifying noise measured in the desired input. Figure 4.3 displays the results of using a very high reusing factor, $R = 1000$, with a relatively large noise source of 10^{-5} in the desired input.

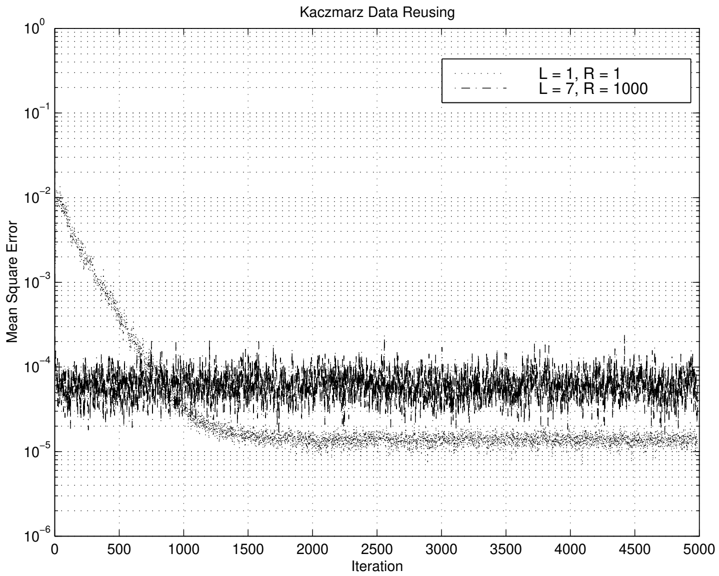


Figure 4.3 Mean Square Error Progressions, Kaczmarz and Cimmino Data-Reusing Algorithms

CHAPTER 5

FAST QUASI NEWTON METHODS

An additional class of algorithms which is related to the *row projection* class of algorithms is the fast quasi-Newton algorithm. The fast quasi-Newton algorithm uses an estimate of the auto-correlation matrix of the input sequence $x(n)$ to update the adaptive filter coefficients and effectively improve the convergence rate performance of the traditional LMS algorithm [23, 39].

The standard quasi-Newton method for iteratively finding the solution to the minimization of a performance function $f(x)$ is given by

$$x_{k+1} = x_k + \alpha_k \mathbf{S}_k^{-1} \nabla f(x_k)^T \tag{5.1}$$

where $\nabla f(x_k)^T$ is the gradient of the performance function $f(x)$ and $\mathbf{S}_k$ is an estimate of the inverse of the Hessian matrix of the performance function $f(x)$ [30].

For the least mean square (LMS) method, the performance function is $E[e^2(n)]$, and the goal is to minimize the mean square error. The gradient can then be approximated as the expectation of $e(n)X(n)$, and the Hessian may be approximated accordingly as $\mathbf{R}_{xx}(n)$, the auto-correlation matrix of the input sequence $x(n)$.

Based upon these approximations, the fast LMS quasi-Newton algorithm for adaptive filtering may be expressed as

$$W(n+1) = W(n) + \mu_n e(n) \tilde{\mathbf{R}}_{xx}^{-1}(n) X(n), \tag{5.2}$$

where $\mu_n = \frac{1}{1+X^T(n)\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)}$, and $\tilde{\mathbf{R}}_{xx}$ is an estimate of the true autocorrelation matrix $\mathbf{R}_{xx}(n)$ of the input sequence $x(n)$.

Several methods exist for estimation of the inverse of the matrix $\tilde{\mathbf{R}}_{xx}(n)$. For real-valued wide-sense stationary sequences, $\mathbf{R}_{xx}(n)$ has Toeplitz structure. Assuming this structure, the Levinson algorithm may be used to compute the quantity $\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)$, sometimes referred to as the Kalman gain [40]. Marshall [39] observed a relationship between the fast quasi-Newton algorithm and the transform domain adaptive filtering algorithm where the optimal transformation is employed. This transformation is the *Karhunen-Loeve* transformation, and can represent the eigen-decomposition of the auto-correlation matrix $\mathbf{R}_{xx}(n)$.

In the transform domain adaptive algorithm, the input sequence $x(n)$ is transformed by an orthogonal linear transformation such as the discrete Fourier transform (DFT), discrete cosine transform (DCT) or the Walsh-Hadamard transform (WHT). Using the transformed version of the input, an algorithm is developed in the new domain which updates the better[1] conditioned input signals. The transform domain algorithm may be expressed as

$$\tilde{W}(n) = \tilde{W}(n-1) + 2\mu e(n)\mathbf{\Lambda}^{-2}\mathbf{T}^{*}X^{*}(n), \tag{5.3}$$

where $\tilde{W}(n)$ is the transformed version of the adaptive filter coefficient vector $\tilde{W}(n) = \mathbf{T}W(n)$ and the matrix $\mathbf{T}$ is the matrix version of the linear transformation. The matrix $\mathbf{\Lambda}^{-2}$ represents an estimate of the power associated with the i^{th} row of the matrix $\mathbf{T}$.

Choosing the optimal or Karhunen-Loeve transformation and observing the algorithm in the time-domain rather than the equivalent transformed domain algorithm leads to the following update expression for the adaptive filter coefficients:

$$\begin{aligned} W(n) &= W(n-1) + 2\mu e(n)[\mathbf{T}^{T}\mathbf{\Lambda}^{-2}\mathbf{T}^{*}]X^{*}(n) \\ &\approx W(n-1) + 2\mu e(n)\mathbf{R}_{XX}^{-1}X^{*}(n). \end{aligned} \tag{5.4}$$

[1] Conceivably, the choice of transformation could worsen the performance of the algorithm and make the input sequence more correlated. For more discussion of the transform domain adaptive algorithm see Marshall [39] and Marshall and Jenkins [23]

Using this interpretation of the two algorithms, the operation of the fast quasi-Newton algorithm may now be described as forming the solution to the linear system

$$\mathbf{X}^H(n)\mathbf{T}^*\mathbf{\Lambda}^{-\mathbf{2}}\mathbf{T}W(n) = D(n), \tag{5.5}$$

where the matrix $\mathbf{T}^*\mathbf{\Lambda}^{-2}\mathbf{T}^H$ is the eigendecomposition of the autocorrelation matrix. From *row projection* theory, the use of the optimal transformation which makes the rows of the matrix $\mathbf{X}^H(n)$ orthogonal will lead to the convergence of the algorithm in M steps where M is the number of rows. This convergence behavior is typical of Newton algorithms where the convergence of the algorithms can occur in approximately $2N$ iterations.

In this chapter, the conjugate gradient algorithm will be used to compute the Kalman gain factor $\tilde{\mathbf{R}}_{XX}(n)X(n)$. Using the conjugate gradient algorithm will allow the fast quasi-Newton algorithm to approximate the behavior of the other *row projection* techniques developed. Specifically, the use of this algorithm will lead to algorithms which are scalable in performance and computational complexity.

5.1 Conjugate Gradient Methods Employed to Solve Kalman Gain Problem

The conjugate gradient (CG) algorithm may be used to perform the computation for the Kalman gain $\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)$ [41, 42] instead of the Levinson algorithm as prescribed by Marshall [39] and Marshall and Jenkins [23]. The CG algorithm is an iterative method which requires exactly N steps to converge to $\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)$. The overall complexity of a fast quasi-Newton algorithm employing the conjugate gradient method is $O(N^2log_2(N))$. However, significant improvement over the traditional normalized LMS algorithm performance can be accomplished without too much increase in computational complexity by executing the CG algorithm a small number of steps. In this case, the complexity is $O(LNlog_2(N))$ where L is the number of steps.

So, the question remains: What happens when the CG algorithm is executed a few steps on a full-rank matrix $\mathbf{A}$, associated with the linear system $\mathbf{A}x = b$? The CG algorithm iteratively minimizes the following performance function:

$$J(x) = \min_x x^T \mathbf{A} x - b^T x. \tag{5.6}$$

The CG algorithm minimizes the performance function $J(x)$ over the k-dimensional linear variety $x_0 + \beta_k$ where β_k is the subspace spanned by the direction vectors $\{d_0, d_1, \ldots, d_k\}$. Using a principal result from the *conjugate gradient theorem* [30], this basis can be expressed in terms of the matrix Q and the vector b. This is given by the following relationship:

$$R(\mathbf{D}) = span[d_0, d_1, \ldots d_k] = span[b, \mathbf{A}b, \ldots, \mathbf{A}^{k-1}b], \tag{5.7}$$

where $R(\mathbf{D})$ is the range space of the direction vectors d_i and it forms the basis for the solution subspace. This assumes that the CG algorithm is initialized with a zero initial condition $x_0 = 0$.

In terms of the fast quasi-Newton algorithm, an approximation or projection of the product $\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)$ is computed. The CG algorithm searches for a solution vector $g(X(n), \tilde{\mathbf{R}}_{xx}(n))$ which minimizes

$$||g(X(n), \tilde{\mathbf{R}}_{xx}(n)) - \tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)||^2 \tag{5.8}$$

such that

$$g(X(n), \tilde{\mathbf{R}}_{xx}(n)) \in span[X(n), \tilde{\mathbf{R}}_{xx}(n)X(n), \ldots, \tilde{\mathbf{R}}_{xx}^{k}(n)X(n)], \tag{5.9}$$

where k is the number of steps in which the CG algorithm is executed. This can also be stated more tersely as

$$g(X(n), \tilde{\mathbf{R}}_{xx}(n)) = P_{R(\mathbf{D})}[\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)], \tag{5.10}$$

where $P_{R(\mathbf{D})}$ is the projection operator on $R(\mathbf{D})$.

Using this expression, the conjugate gradient version of the fast quasi-Newton algorithm (CG-FQN) can be expressed in the following form:

$$W(n+1) = W(n) + e(n)\frac{P_{R(\mathbf{D})}[\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)]}{1 + X^T(n)P_{R(\mathbf{D})}[\tilde{\mathbf{R}}_{xx}^{-1}(n)X(n)]}, \tag{5.11}$$

where $R(\mathbf{D}) = span[X(n), \tilde{\mathbf{R}}_{xx}(n)X(n), \ldots, \tilde{\mathbf{R}}_{xx}^k(n)X(n)]$.

Using the CG-FQN algorithm, the designer of the adaptive filtering algorithm may choose performance level (i.e., misadjustment and convergence rate) by choosing the number of conjugate direction vectors. Alternatively, a stopping criterion may be defined by comparing the gradient of the performance function to some predefined tolerance value [40].

5.2 Use of Preconditioners to Accelerate Convergence

The rate of convergence of the conjugate gradient algorithm is directly related to the eigenvalue spread of the matrix $\tilde{\mathbf{R}}_{xx}(n)$. The convergence can be accelerated by applying a preconditioning step at each iteration of the conjugate gradient algorithm [40, 43]. Instead of solving the system $\tilde{\mathbf{R}}_{xx}(n)K(n) = X(n)$ the preconditioned system is solved:

$$\bar{\mathbf{R}}_{xx}(n)\bar{K}(n) = \bar{X}(n), \tag{5.12}$$

where $\bar{\mathbf{R}}_{xx}(n) = \mathbf{P}^{-1}\tilde{\mathbf{R}}_{xx}(n)\mathbf{P}^{-1}$, $\bar{K}(n) = \mathbf{P}K(n)$, and $\bar{X}(n) = \mathbf{P}^{-1}X(n)$. Use of the preconditioned CG method requires that the matrix $\mathbf{P}^{-1}$ can be computed efficiently. For this reason, the matrix $\mathbf{P}$ is generally chosen to be circulant, so that a fast discrete Fourier transform (FFT) may be used to efficiently compute the inverse. Ng and Plemmons [43] and Hull [40] independently developed this method of adaptive filtering. Hull [40] uses several preconditioners developed by others [44, 45].

Ng and Plemmons [43] augment the system $\tilde{\mathbf{R}}_{xx}(n)K(n) = X(n)$ to make it circulant. Then using the FFT, the inverse of the augmented version of the auto-correlation matrix from the previous iteration $\tilde{\mathbf{R}}_{xx}(n-1)$ is inverted and used as a preconditioner. They have shown that their method converges with probability 1 and requires $O(N \log_2 N)$ operations per adaptive filter input.

The convergence rate of the preconditioned CG method is governed by the eigenvalue spread of $\mathbf{P}^{-1}\mathbf{R}_{xx}(n)$. A smaller condition number implies a faster convergence rate. In this dissertation, the performances of four different forms of the matrix $\mathbf{P}$ are compared. The preconditioner developed by Chan and Strang [45] uses the central half of the diagonals of the original matrix $\tilde{\mathbf{R}}_{xx}(n)$. Chan and Strang [45] showed for a large class of matrices known as Wiener class matrices that the eigenvalues of the matrix $\mathbf{P}^{-1}\tilde{\mathbf{R}}_{xx}(n)$ are clustered around one except for a finite number of outliers. This implies that the PCG method will converge to the optimum solution in fewer than N iterations for this class [30]. Each of these methods is compared to the limiting case $\mathbf{P} = \mathbf{I}$.

The preconditioner developed by Chan [44] minimizes the Frobenius norm $||\mathbf{P} - \tilde{\mathbf{R}}_{xx}(n)||_F^2$ between the original matrix $\tilde{\mathbf{R}}_{xx}(n)$ and all circulant matrices $\mathbf{P}$. This solution to this minimization has a closed-form expression, so the preconditioner may be easily computed.

Ku and Kuo [46] developed a broad class of preconditioning matrices which attempt to use all of the original elements of the matrix $\tilde{\mathbf{R}}_{xx}(n)$. The preconditioners developed by Chan and Strang are special cases of two of the four classes of preconditioners developed by Ku and Kuo. One case of the first class of preconditioner developed by Ku and Kuo is included for comparison. The preconditioned conjugate gradient algorithm is summarized in Table 5.1.

5.3 Simulations

Several simulations were executed to demonstrate the efficacy of this approach to adaptive filtering. System identification and channel equalization experiments were performed to illustrate the impact of the algorithm in two different applications. The algorithms were studied under a diverse set of input conditions. Primarily, the effects of varying the preconditioner and the number of iterations of the CG algorithm were studied and compared with the normalized LMS algorithm.

Table 5.1 Summary of the Preconditioned CG Algorithm

Initialization:	$K_0 = 0$ $D_0 = G_0 = X - \tilde{\mathbf{R}}_{xx}(n)K_0$ $\beta_1 = 0$
Step 1:	Preconditioning - Solve for z_{j-1} $\mathbf{P}Z_{j-1} = G_{j-1}$
Step 2:	CG Iteration - Compute $\beta_j = \frac{Z_{j-1}^T G_{j-1}}{Z_{j-2}^T G_{j-2}}$ $D_j = Z_{j-1} + \beta_j D_{j-1}$ $\alpha_j = \frac{Z_{j-1}^T G_{j-1}}{D_j^T \mathbf{R_{xx}} D_j}$ $K_j = K_{j-1} + \alpha_j D_j$ $G_j = G_{j-1} - \alpha_j \tilde{\mathbf{R}}_{xx}(n) D_j$
Step 3:	if $\|\|G_j\|\|^2 > TOL$, or $j < M \leq N$ return to Step 1

In the channel equalization setting, the quasi-Newton acceleration technique was explored as a method to improve the convergence behavior of the notoriously "slow" LMS constant modulus adaptive algorithm employed in equalization. The constant modulus algorithm has been described in some detail in Chapter 2 and in Appendix B where data-reusing algorithms are explored as a method to improve their behavior. For greater detail about the LMS version of the constant modulus algorithm and more discussion of the *row projection*-like behavior of the constant modulus algorithm, see Appendix B.

The two different forms of the fast quasi-Newton algorithm employed in the simulation experiments in this chapter, the MSE or LMS algorithm, and the constant modulus equalization algorithm are summarized in Table 5.2.

Two system identification examples are presented here to illustrate the effects of using different preconditioners, the choice of L, and the number of steps in which the preconditioned CG algorithm is executed. For both examples, experiments were run in the traditional system identification setting. The unknown system can be modeled as an eight-tap FIR filter. The input signal was colored using a 32-tap FIR filter referred to

Table 5.2 Adaptation Algorithms

Initialization:
$W_i = 0, i = 0, \ldots, N-1$
if CMA, $W_{ref} = 1$, $ref = arbitrary$
$r(i) = 0, i = 1, \ldots, N-1$
$r(0) = 1$
Vector and Matrix Definitions:
$\mathbf{A} = diag[1\alpha^{1/2} \ldots \alpha^{N-1/2}]$
$R(n) = [r(0), r(1), \ldots r(N-1)]^T$
$\mathbf{R_{xx}}(n) = toeplitz\{R(n)\}$
Recursion:
$y(n) = W^T(n)X(n)$
$\epsilon(n) = (\lvert y(n)\rvert^2 - 1)y(n)$
$e(n) = d(n) - y(n)$
$R(n) = \alpha R(n-1) + x(n)AX(n)$
$K(n) = \mathbf{R_{xx}^{-1}(n)}X(n)$ - via the PCG method
$\mu(n) = \frac{1}{4(\delta + X^T(n)\mathbf{R_{xx}^{-1}(n)}X(n))}$
Tap Weight Updates:
$W(n+1) = W(n) - \mu(n)\epsilon(n)K(n)$ – CMA Algorithm
$W(n+1) = W(n) + \mu(n)e(n)K(n)$ – LMS Algorithm

as Filter A in Marshall's text [39]. This filter introduces a strong correlation between successive samples, and the eigenvalue ratio is fairly high. The coloring filter is high-pass.

The resulting progression of the mean square error for the system identification setting is shown in Figure 5.1 for the choice of all four preconditioners. The preconditioned CG-FQN algorithms which employ the preconditioners developed by Chan and Strang [45] and Chan [44] appear to exhibit similar performance for this case. The algorithm with the Ku and Kuo [46] preconditioner appears to lag behind these two in performance. The identity matrix used as a preconditioner exhibits the slowest convergence rate performance of the four.

In the next example, the effects of executing the preconditioned CG algorithm in L steps are demonstrated. Shown in Figure 5.2, the mean square error progressions of

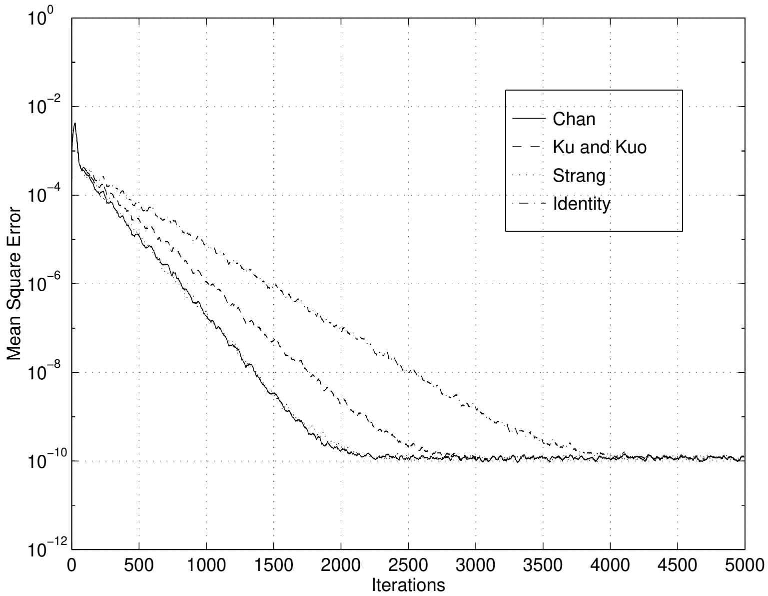


Figure 5.1 Mean Square Error Progressions, System Identification, Colored Noise Input, Various Preconditioners

the various L factors are shown. The preconditioner developed by Chan [44] is the only one explored here. As the factor L increases, or the number of CG steps increase, the convergence rate appears to increase. The most significant difference is between the $L = 1$ and $L = 3$. Increases beyond $L = 3$ do not appear to make as significant a difference. In fact, it is difficult to discern the difference in performance between $L = 5$ and $L = 7$.

In the first equalization experiment, the adaptive filter is attempting to model the inverse of the channel which has the impulse response [1]:

$$h(n) = \delta(n) + 0.3\delta(n-2). \tag{5.13}$$

This channel mildly colors the transmitted sequence. The eigenvalue spread of the transmitted BPSK sequence of equally likely + or - 1's was changed from 1 to 3.07 by the channel.

For the LMS-constant modulus algorithm, the parameter μ was set to 1/25. For the PCG-Newton method, α was set to 0.95, and the normalizing factor δ in the self-adjusting μ factor was set to 0.05. The additive noise floor was fixed at -100 dB. The resulting mean square error and the reconstructed modulus of the signal conjugate gra-

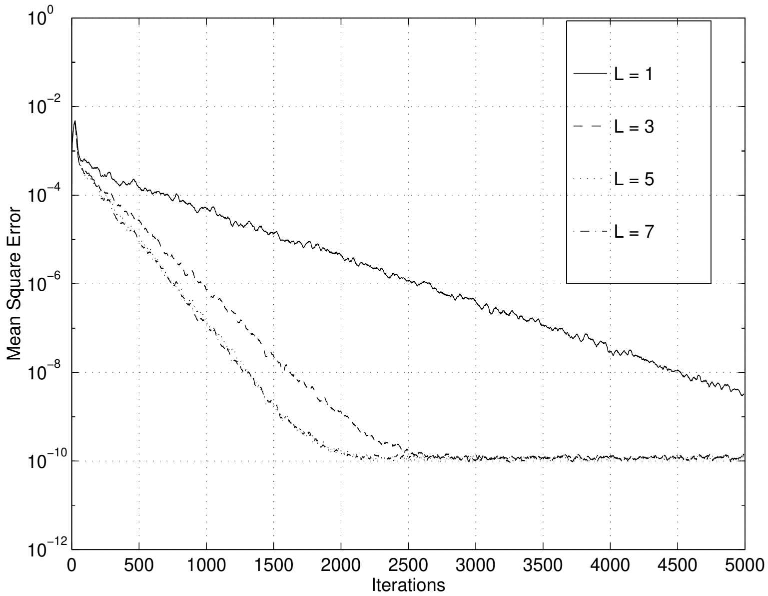


Figure 5.2 Mean Square Error Progressions, System Identification, Colored Noise Input, Different Values of L

dient algorithm are plotted in Figure 5.3. The conjugate gradient algorithm was limited to three iterations ($M = 3$) in both the CMA and MSE algorithms. The tolerance value or stopping condition chosen for the gradient expression was 10^{-4}. For this example, the LMS algorithm appears to exhibit convergence and misadjustment performance similar to the three preconditioned conjugate gradient algorithms employing preconditioners by Chan and Strang [45], Chan [44], and Ku and Kuo [46]. In Figure 5.4, the results of the MSE algorithm appear to illustrate that the preconditioners have little impact in either changing the misadjustment or the convergence rate of the adaptive filter. Interestingly, the excess mean square error for the MSE algorithm is higher than the misadjustment for the CMA algorithm.

In the second equalization experiment, the adaptive filter models a channel which highly correlates the data and has a resulting eigenvalue spread of 100. The model for the channel is given by [1]

$$h(n) = \tfrac{1}{2}[1 + \cos(2\pi(i-2)/W)], \quad i = 1, 2, 3, \tag{5.14}$$

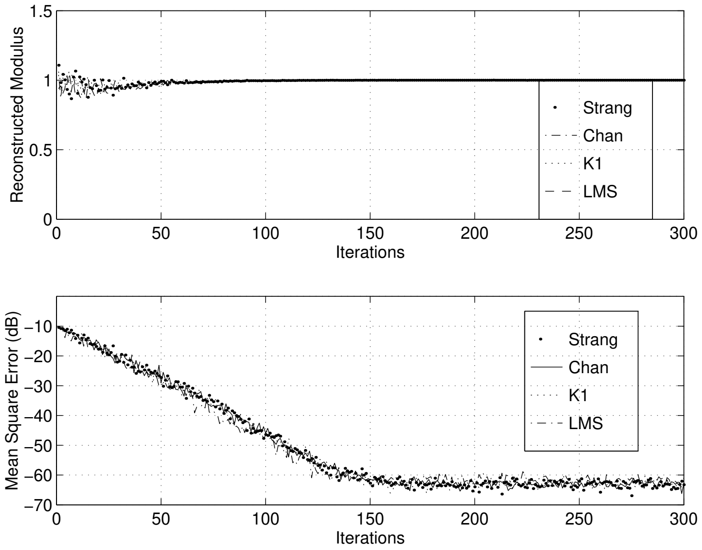


Figure 5.3 Reconstructed Modulus and MSE, CMA Algorithm, Channel A

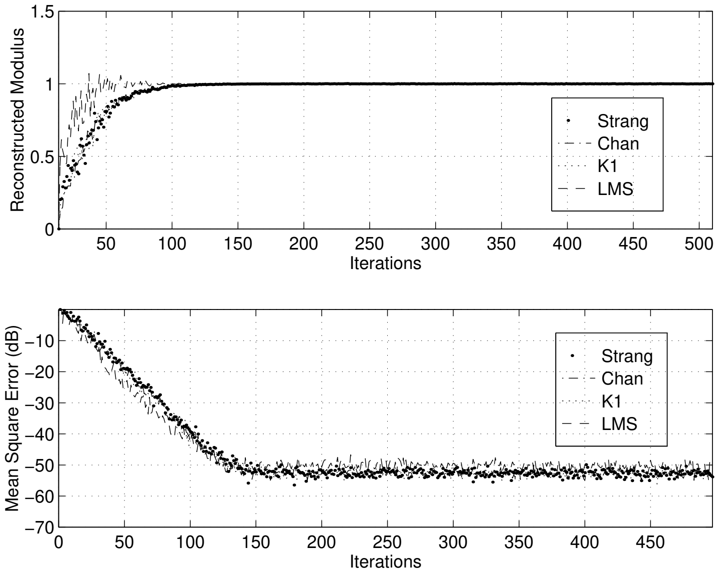


Figure 5.4 Reconstructed Modulus and MSE, MSE Algorithm, Channel A

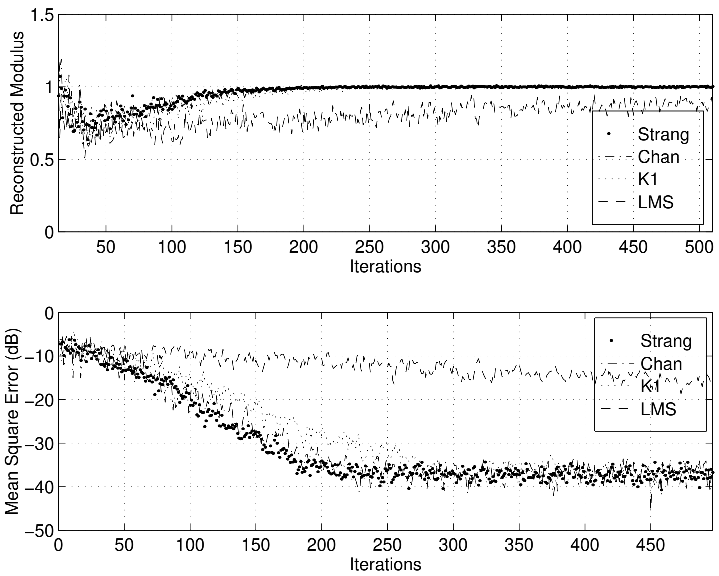


Figure 5.5 Reconstructed Modulus and MSE, CMA Algorithm, Channel B

where W was set to 3.5. For this algorithm, μ was set to $1/50$ for the CMA algorithm and $1/20$ for the MSE algorithm. The parameters for the PCG-Newton method remained unchanged.

Figure 5.5 appears to show that the use of the PCG-Newton method is strongly warranted in the CMA algorithm. The LMS algorithm exhibits a fairly slow convergence rate, and a fairly high misadjustment, which is not displayed in order to illustrate the different convergence behaviors of the three different preconditioners. The filter converges to around -30 db, which is about 10 dB higher than the PCG-Newton algorithms. The difference in convergence among the PCG-Newton methods with different preconditioners are small. Chan's preconditioner lags slightly behind that of Strang and that of Ku and Kuo. The difference appears to be only a few iterations relative to the hundreds of additional iterations required by LMS to reach a higher overall misadjustment level. Figure 5.6 appears to demonstrate again that the MSE performance of the PCG Newton-based algorithm is superior to the LMS-based algorithm. The MSE algorithm does, however, appear to converge more rapidly than the CMA algorithm for this same model. However, the LMS algorithm still requires several hundred additional iterations over any of the preconditioner-based algorithms. The preconditioned conjugate gradient algorithm for channel equalization was observed to greatly enhance convergence and

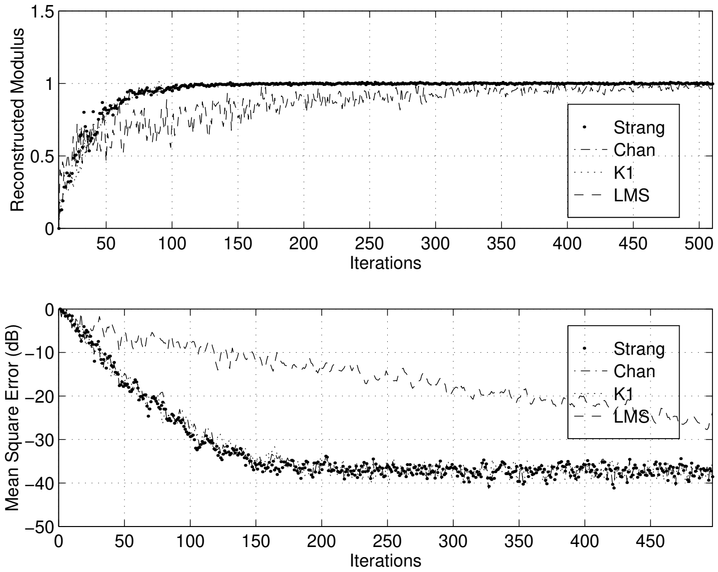


Figure 5.6 Reconstructed Modulus and MSE, MSE Algorithm, Channel B

misadjustment performance over traditional LMS-based mean square error and constant modulus algorithms. The use of preconditioners offers the designer a choice of performance level by choosing the stopping condition of the conjugate gradient algorithm either by tolerance or a fixed number of iterations. By choosing a relatively small number of fixed iterations of the algorithm, the computational complexity of the preconditioned CG method was observed to be $O(NMlog(N))$, where M is the number of iterations of the inner loop, and N is the order of the adaptive filter. The constant modulus algorithm that traditionally suffers from poor performance relative to the MSE algorithm appeared to have its convergence rate made comparable to the traditional MSE algorithm for channels which highly correlate data.

CHAPTER 6

APPLICATIONS OF PROJECTION METHODS IN ADAPTIVE FILTERING

6.1 Introduction

There are a number of important applications of the *row projection* methods that have been developed in this dissertation. Three additional applications are explored here. The algorithms studied and developed in this dissertation are applied to two dimensional adaptive filtering, adaptive interference suppression, and the use of *row projection* methods in fault tolerant adaptive filtering methods. The application of *row projection* methods to these applications is motivated by some common needs. In each of these applications, LMS-level performance is unacceptable. Further, RLS methods are either not directly applicable or not computationally feasible.

For two-dimensional adaptive processing, the extreme cost of switching from an LMS to an RLS method is prohibitive. LMS methods possess $O(N^2)$ computational complexity where the filter's region of support is defined by an $N \times N$ window of coefficients. To implement the recursive least squares algorithm in two dimensions, $O(N^4)$ multiplications are required. Fortunately, the use of the *row projection*-based adaptive algorithms allows the development of a number of algorithms which offer significant performance improvements at a marginal increase in computational complexity.

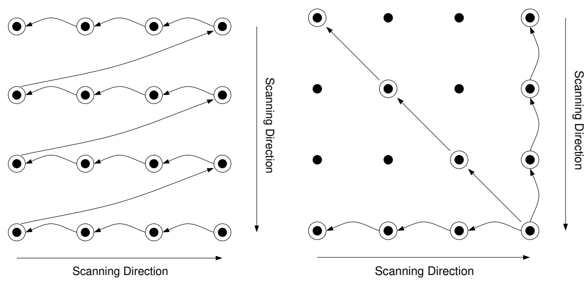
In wireless communication settings, the presence of other users can degrade the accuracy of mean square optimal detectors. An adaptive interference suppression system has been explored as a potential method of reducing the impact of interfering users. RLS adaptive techniques have been shown to offer advantages over traditional LMS techniques. This particular application uses an input sequence with an autocorrelation matrix which is dominated by L eigenvalues which could lead to numerical problems for RLS algorithms. Affine projection techniques and, by analogy, *row projection* techniques have been shown to be well-suited to this type of input sequence, especially when the number of dominant eigenvalues is matched to the number of reused data pairs.

Using a fault tolerant model developed by Schnaufer [21], a number of algorithms were developed which are impervious to stuck-at faults occurring in the adaptive filtering block. Specifically, a register which contains an adaptive filter coefficient is stuck at a particular value and is unable to update the coefficients to minimize the mean square error. LMS techniques can be developed which incorporate redundancy in the adaptive filter structure. However, post-fault convergence rates can be extremely slow. Recursive least squares techniques are not applicable because the redundancy in the tap weights leads to a rank-deficient set of normal equations. Jenkins et al. [47] developed a technique which improved the performance of the algorithm over traditional LMS algorithms. This algorithm can be described by the *row projection* framework developed in this dissertation. Further, new algorithms for fault tolerant adaptive algorithms are developed.

6.2 Two-Dimensional Adaptive Filtering

The convergence rate performance of two-dimensional adaptive filtering algorithms is limited by the coloring of the input sequence (or the relative conditioning of the autocorrelation matrix) [1]. The convergence rate can significantly limit the effectiveness and usefulness of adaptive filtering in two dimensions in applications such as image enhancement, image coding, and adaptive noise cancellation [48]. Use of recursive least squares algorithms, fast lattice-based algorithms and fast quasi-Newton methods [49] can augment performance at considerable computational and storage complexity costs.

For one-dimensional adaptive filtering applications, it is possible to improve performance by employing either a data-reusing strategy or a projection algorithm which add marginally to the overall computational complexity. Similarly, in two dimensions convergence rate performance may be augmented with little additional cost in computation by reusing existing data for adaptive filter coefficient updates.



(a) (b)

Figure 6.1 Data-Reusing in Two Dimensions, (a) Rectangular Configuration (b) Axial Configuration

6.2.1 Two-dimensional data-reusing algorithms in adaptive filtering

For two-dimensional applications, there are a number of ways that data can be reused. One possible scheme for data-reusing is to reuse data in the fashion shown in Figure 6.1(a). For one example of this scheme, we could reuse the previous 16 data pairs. This corresponds to a neighborhood, L, of four. At each pixel iterate, the adaptive algorithm effectively steps back over the previous 16 pixels defined in Figure 6.1(a). For a neighborhood of L, the computational complexity of this algorithm is $O(L^2N^2)$. There are a number of other schemes which are possible as well. One other scheme is presented in Figure 6.1(b). In this case, the data is reused along two major axes and the diagonal axis for an assumed rectangular scanning pattern. For this neighborhood, the complexity becomes $O(LN^2)$.

6.2.2 Alternative data-reusing strategies

Ostensibly, there are two categories of normalized data-reusing strategies. Both are well described in this dissertation and in [29, 41]. The traditional normalized data-reusing algorithm described by Schnaufer [20] can be formulated in two dimensions as the following algorithm:

- **Step 1.** *Initialization*, $i = 0$

$$c_{00} = W(n_1, n_2)$$

- **Step 2.** *Loop 1* (*While* $i < L$), $j = 0$
- **Step 3.** *Loop 2* (*While* $j < L$)

$$\begin{aligned}
c_{i,j+1} &= c_{i,j} + \alpha_{i,j} X(n_1 - i, n_2 - j) \\
\alpha_{i,j} &= \frac{e(i,j)}{X^T(n_1 - i, n_2 - j) X(n_1 - i, n_2 - j)} \\
e(i,j) &= d(n_1 - i, n_2 - j) - X^T(n_1 - i, n_2 - j) c_{i,j} \\
j &= j + 1
\end{aligned}$$

- **Step 4.** *end Loop 1*, $i = i + 1$
- **Step 5.** *end Loop 2*, $W(n_1, n_2 + 1) = c_{LL}$ Go to **Step 1**

Note that this implementation assumes that the data is reused in a rectangular window as described in Figure 6.1(a). While this form of data-reusing can lead to significant performance improvements over the the traditional normalized LMS algorithm, it requires the hardware to essentially function at $L^2\times$ faster rate. For many situations, this is not feasible.

The other category of data-reusing methods referred to as Cimmino normalized data-reusing algorithm in [29, 41], is a scheme which can be implemented in parallel fashion as discussed earlier. The coefficient update expression is given by

$$\begin{aligned}
W(n_1, n_2 + 1) &= W(n_1, n_2) + \\
&\sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \mu_{i,j} e(i,j) X(n_1 - i, n_2 - j),
\end{aligned} \tag{6.1}$$

where $\mu_{i,j}$ is an auto-adjusting scale factor. It is given by

$$\mu_{i,j} = \frac{1}{X^T(n_1 - i, n_2 - j) X(n_1 - i, n_2 - j)}. \tag{6.2}$$

The error signal $e(i,j)$ is given by

$$e(i,j) = d(n_1-i, n_2-j) - X^T(n_1-i, n_2-j)W(n_1, n_2). \tag{6.3}$$

For this method, the number of multiplies and accumulates associated with the computation of this algorithm is equivalent to the number in traditional normalized data-reusing method. But, this method may be implemented using more parallel structure. The requirement on the hardware speed may be relaxed if additional hardware is available.

6.2.3 Projection methods in two-dimensional adaptive filtering

The two-dimensional projection algorithm minimizes the distance between successive iterates of the adaptive filter coefficients $W(n_1, n_2+1)$ and $W(n_1, n_2)$ subject to a set of constraining conditions. The constraining conditions for a simple neighborhood $L = 1$ can be expressed as

$$E(n_1, n_2) = D(n_1, n_2) - \mathbf{X}^T(n_1, n_2)W(n_1, n_2), \tag{6.4}$$

where $E(n_1, n_2)$ and $D(n_1, n_2)$ are vectors of the error signal and desired signal values. These vectors may be expressed as

$$E(n_1, n_2) = \begin{bmatrix} e(n_1, n_2) \\ e(n_1 - 1, n_2) \\ e(n_1, n_2 - 1) \\ e(n_1 - 1, n_2 - 1), \end{bmatrix} \tag{6.5}$$

$$D(n_1, n_2) = \begin{bmatrix} d(n_1, n_2) \\ d(n_1 - 1, n_2) \\ d(n_1, n_2 - 1) \\ d(n_1 - 1, n_2 - 1) \end{bmatrix}. \tag{6.6}$$

The matrix of the input signal $\mathbf{X}(n_1, n_2)$ contains four regions of support of the filter. These regions of support are individually concatenated into a vector. A sample vector

$X(n_1, n_2)$ would be

$$X(n_1, n_2) = \begin{bmatrix} x(n_1, n_2) \\ x(n_1 - 1, n_2) \\ \vdots \\ x(n_1 - N + 1, n_2 - N + 1) \end{bmatrix}. \tag{6.7}$$

The matrix $\mathbf{X}^T(n_1, n_2)$ can be expressed in terms of the four concatenated vectors:

$$\mathbf{X}^T(n_1, n_2) = \begin{bmatrix} X^T(n_1, n_2) \\ X^T(n_1 - 1, n_2) \\ X^T(n_1, n_2 - 1) \\ X^T(n_1 - 1, n_2 - 1) \end{bmatrix}. \tag{6.8}$$

The projection algorithm computes the orthogonal projection of $W(n_1, n_2)$ onto the intersection of solution hyperplanes defined by $e(n_1, n_2) = e(n_1 - 1, n_2) = e(n_1, n_2 - 1) = e(n_1 - 1, n_2 - 1) = 0$. Various efficient methods for the rectangular reuse window allow this algorithm to be implemented in $O(L^4 N^2)$ [7, 11]. Effectively, these methods update the adaptive filter coefficients in the following manner:

$$\begin{aligned} W(n_1, n_2 + 1) &= (I - P_{N(\mathbf{X}(n_1,n_2))})W(n_1, n_2) \\ &\quad + [\mathbf{X}^T(n_1, n_2)]^\dagger D(n_1, n_2), \end{aligned} \tag{6.9}$$

where $[\mathbf{X}^T(n_1, n_2)]^\dagger$ is the Moore-Penrose pseudo-inverse of this matrix.

6.3 Computer Simulations

Several experiments were designed to illustrate the effectiveness of these approaches. The number of reuses was varied. In addition, the performance of the rectangular and diagonal windows was compared. In the first example, an 8×8 FIR filter is used for both the model and the adaptive filter. White noise is used as the input signal. Figure 6.2 illustrates the progression of the mean square error (MSE) for neighborhoods of $L = 1$, 3, and 5 for the traditional data-reusing method. The neighborhood of $L = 0$ corresponds

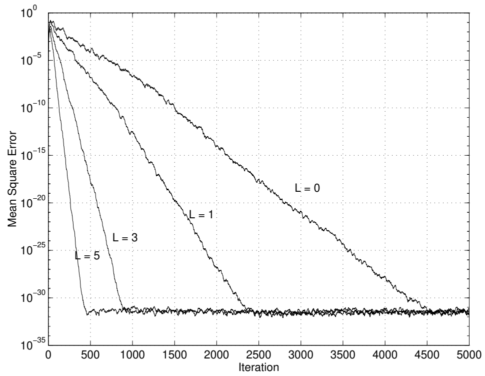


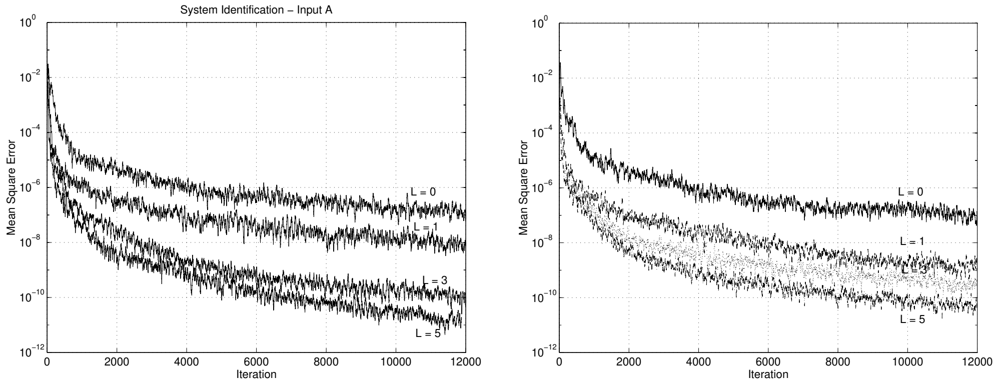
Figure 6.2 MSE Progression, Normalized Data-Reusing Algorithm, White Noise Input, Rectangular Window

to using only one data pair, and it is functionally the same as the normalized LMS algorithm. Figure 6.2 appears to illustrate that as L gets larger, the convergence speed of the algorithm is greatly improved.

In the second example, a low-pass coloring filter is used to change the statistics of the input signal. Figure 6.3(a) illustrates MSE progressions of the data-reusing methods with the use of the rectangular window. For this limiting example, the normalized LMS algorithm corresponding to $L = 0$ appears to take considerable time to converge. The Kaczmarz methods appear to offer significant improvement in convergence speed as L grows larger. Figure 6.3(b) illustrates the impact of using the reduced-order data window. For this particular experiment, the axial window described earlier is used. It appears that comparable performance for the rectangular window may be obtained by using the reduced-order window.

In Figure 6.4 the Cimmino normalized data-reusing strategy results are displayed. The results for different rectangular windows are shown. There is not a large disparity between varying values of L. The mean square errors for $L = 3$ and $L = 5$ appear to have very similar convergence behavior slightly better than the $L = 1$ case. Overall, this data-reusing strategy has slightly better behavior than the normalized LMS algorithm.

The results of using the higher-complexity accelerated data-reusing/affine projection algorithm are displayed in Figure 6.5(a). This higher-complexity projection algorithm again exhibits the similar trend of improved performance for increasing window sizes. However, the improvement in convergence rate for this particular examples is marginal



(a) (b)

Figure 6.3 MSE Progression, Normalized Data-Reusing Algorithm, Colored Noise Input, (a) Rectangular Window, (b) Axial Window

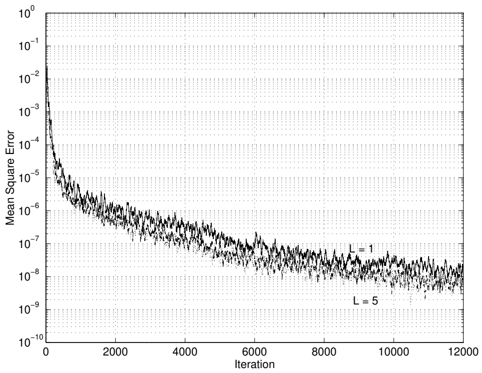
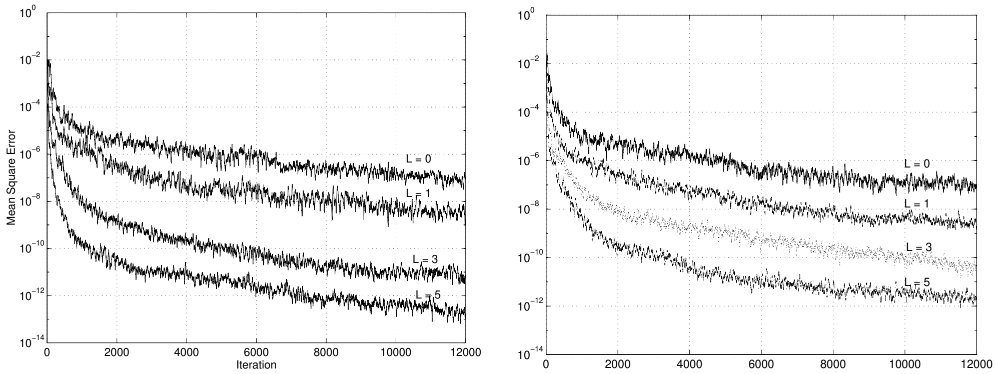


Figure 6.4 MSE Progression, Cimmino Data-Reusing Algorithm, Colored Noise Input, Rectangular Window



(a) (b)

Figure 6.5 MSE Progressions, Projection Algorithm, Colored Noise Input, (a) Rectangular Window, (b) Axial Window

over the data-reusing strategy. In Figure 6.5(b), the results of using the reduced order window for the projection method are displayed. Similarly, the reduced-order window performs similarly to the full rectangular window.

6.4 Adaptive Interference Suppression

The explosion in research in wireless technology has been motivated by the desire to increase bandwidth efficiency and the number of users in a cellular system. Multiple access techniques through the use of unique signature sequences associated with each user have been shown to be an effective way for users to simultaneously communicate. Identification of these sequences and optimal detection of the transmitted sequences in the presence of fast-fading/multipath channels has been shown to be a difficult problem. Compounding this problem is the mere presence of the other users, who often have combined power greater than the desired user [50],[51].

Adaptive minimum mean square error (MMSE) interference suppression methods have been shown to be an effective method for solving the near-far problem associated with code division multiple access (CDMA) communication systems. For this problem, the bit-error rate of the conventional receiver is considered to be sensitive to differences between the received energies of the desired user (considered "far") and the interfering users (considered "near"). Reliable demodulation is impossible unless stringent power control is exercised [50],[51]. There are a number of techniques for solving this problem.

They primarily rely on stochastic gradient or LMS-type techniques to derive the optimal correlator for the received sequence. Further refinement in these algorithms has led to the development of RLS-type techniques for the improved detection and tracking of received sequences, albeit at a somewhat higher computational cost. In this dissertation, the *row-projection* framework is used to develop new algorithms for adaptive interference suppression. These algorithms exploit the inherent structure of the autocorrelation matrix of these multiple access systems and offer reduced computational complexity over these RLS techniques.

6.4.1 System models

A simple discrete system model for the N-dimensional vectors of the received sequence for the multiple access synchronous CDMA system using a BPSK model can be expressed as

$$R(n) = b_0(n)U_0 + \sum_{i=1}^{L-1} b_i(n)U_i + \sigma N(n), \tag{6.10}$$

where U_i is $A_i(n) * S_i$, where $A_i(n)$ is a time-varying amplitude assumed not to vary significantly enough over the N samples of the code, and S_i is the signature sequence associated with the i-th user. The received transmitted sequence information is conveyed in the sequence $b_i(n)$.

Assuming that the signature sequences are chosen to be strict-sense orthogonal, not always the choice, the autocorrelation matrix for the received sequence may be expressed using a similarity transformation as

$$\mathbf{R}_{rr}(n) = \mathbf{U} \begin{bmatrix} P_0 + \sigma^2 & & & & & \\ & \ddots & & & & \\ & & P_{L-1} + \sigma^2 & & & \\ & & & \sigma^2 & & \\ & & & & \ddots & \\ & & & & & \sigma^2 \end{bmatrix} \mathbf{U}^H, \tag{6.11}$$

where P_i is the power of the i'th user. Further, the matrix $\mathbf{U}$ may be decomposed as $\begin{bmatrix} \mathbf{U}_S & \mathbf{U}_N \end{bmatrix}$, where $\mathbf{U}_S$ is a set of eigenvectors corresponding to the signaling subspace or the range space spanned by $[S_0, \ldots S_{L-1}]$, and $\mathbf{U}_N$ is a set of eigenvectors corresponding to the noise subspace. Further, by the joint independence of $b_i(n)$ and $N(n)$, these subspaces may always be chosen to be *orthogonal*. More complex asynchronous system models may be developed, but this inherent structure where L dominant eigenvalues (associated with L users in the multiple axis system) are present will always exist.

6.4.2 Projection-based adaptive interference suppression

The affine projection algorithm, data reusing, and accelerated data reusing adaptive filtering methods can be shown to be highly suited to this dominant eigenvalue problem. They have been shown to perform well in the problem of acoustic echo cancellation where this type of structure is present in the auto-correlation matrix, i.e. speech excitation [6, 9, 12].

A deterministic view of the affine projection and *row projection* methods has been presented which shows that these interference suppression algorithms are derived with one basic idea: to minimize the difference between successive iterations of the adaptive filter iterates subject to a basic constraint. In more concise terms, the following optimization problem is solved:

$$\begin{gathered} \min_{C(n)} ||C(n) - C(n-1)|| \\ subject\ to\ E(n) = 0, \end{gathered} \tag{6.12}$$

where $C(n)$ represents the time-varying adaptive coefficients. In the standard adaptive filtering context, $E(n)$ represents a vector of zero errors:

$$E(n) = [e(n), \ldots, e(n-L+1)]^T, \tag{6.13}$$

where $e(i) = d(i) - C^H(n)R(i)$. In this detection framework, the constraining condition is replaced with the following operation:

$$E(n) = b_i(n) - sign(real(C^H(n)R(n)) = 0. \tag{6.14}$$

When a training sequence exists, the adaptive receiver converges to the MMSE detector of the desired user $b_i(n)$. Since it is not practical to continue to train the detector, most schemes switch to decision directed mode after achieving some minimum mean square error. However, this can create some difficulty in deep fading channels. Further study is warranted into the effectiveness of these approaches in the presence of fast-fading channels. Other blind methods have been developed which minimize the output energy of the correlation structure $sign(real(C^H(n)R(n))$. However, this dissertation focuses on convergence rate and computational complexity issues, and these other applications are not discussed here.

6.4.3 Stochastic interpretation of projection-based adaptive filtering

The affine projection algorithms described in Gay and Tavatia [9] and *row projection* algorithms described by Soni et al. [28, 29] and in this dissertation can be expressed as an iterative algorithm as

$$C(n) = C(n-1) + \mathbf{R}(n)[\mathbf{R}^H(n)\mathbf{R}(n)]^{-1}E(n), \tag{6.15}$$

where $\mathbf{R}(n)$ is a matrix consisting of consecutive received vectors associated with the received sequence

$$\mathbf{R}(n) = [R(n), \ldots R(n-L+1)]. \tag{6.16}$$

Slock [8] and Gay and Tavatia [9] conjectured that $\mathbf{R}^H(n)\mathbf{R}(n)$ can be approximated as a windowed estimate of the $L \times L$ dimensional part of the $N \times N$ autocorrelation matrix. They exploit this conjecture to develop fast algorithms based on fast transversal filtering concepts which approximate this matrix as “nearly Toeplitz” [7]. This approximate relationship is expressed as

$$\tilde{\mathbf{R}}_{RR}(n) \approx \mathbf{R}^H(n)\mathbf{R}(n) = \mathbf{U}_s\Sigma_s\mathbf{U}_s^H, \tag{6.17}$$

where $\Sigma_s = diag[P_0 + \sigma^2, \ldots, P_{L-1} + \sigma^2]$ is the signal plus noise power matrix. In this dissertation, the conjugate gradient or the Tchebyshev method was used to exactly compute the update in Equation (6.15). This can be viewed as pre-whitening the input $R(n)$ by an L-th order pre-whitening filter [8]. Further, the computational complexity of the algorithms associated with the computation of the adaptive detector outputs is $O(L^2N)$.

6.4.4 Application to CDMA-based system

For the adaptive detection problem, the inclusion of $L-1$ users in the reception of the multiple access data leads to an autocorrelation matrix with L dominant eigenvalues. In the adaptive filtering framework, the convergence rate of the filter is controlled by the eigenvalues of the input sequence autocorrelation matrix $\mathbf{R}_{xx}$ where $x(n)$ is the input sequence. The evolution of the error in the tap-weights of the adaptive filter for LMS and stochastic gradient algorithms can be expressed in terms of the autocorrelation matrix as

$$\epsilon(n) = (\mathbf{I} - \mu\mathbf{R}_{xx})\epsilon(n-1), \tag{6.18}$$

where $\epsilon(n) = C(n) - C_{opt}$, and C_{opt} is the MSE optimal value of the coefficients [1].

Premultiplying by a matrix $\mathbf{Q^H}$ composed of the eigenvectors of the autocorrelation matrix $\mathbf{R}_{xx}$ yields the following expression for the evolution of the translated error $V(n)$:

$$V(n) = (\mathbf{I} - \mu\mathbf{\Lambda})V(n-1), \tag{6.19}$$

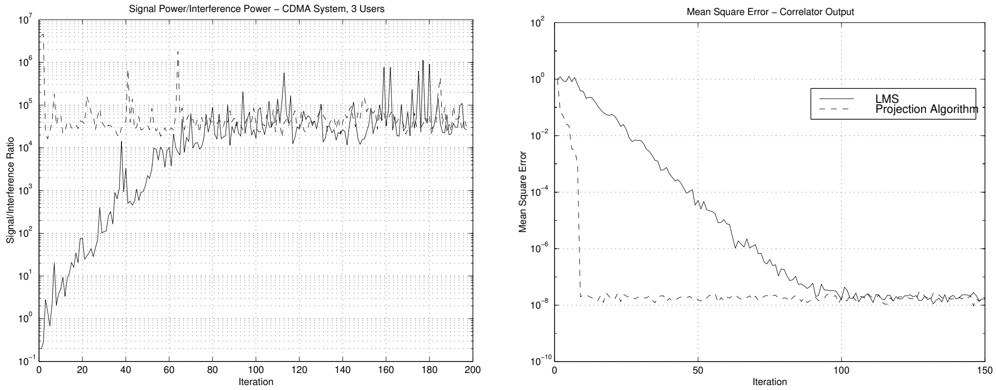
where $V(n) = \mathbf{Q}^H(n)C(n)$. The matrix $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues λ_i's of the autocorrelation matrix.

In terms of the initial value of the error, this becomes

$$V(n) = (\mathbf{I} - \mathbf{\Lambda})^n V(0). \tag{6.20}$$

Clearly, for larger diagonal elements, the λ_i's will lead to slower convergence to C_{opt}.

In the case of the adaptive detector, the L dominant eigenvalues associated with the L users control the rate of convergence of the adaptive receiver. Thus, isolating and pre-whitening these eigenvalues associated with the slower more dominant modes associated with the L users appears to be a promising way to improve the performance of adaptive detectors in the presence of interfering users.



(a) (b)

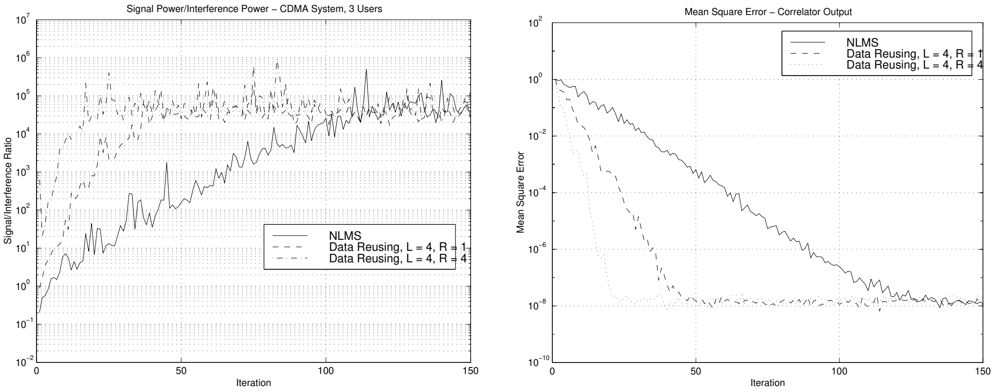
Figure 6.6 LMS and Projection Adaptive Algorithms, (a) SIR, (b) MSE

6.4.5 Simulation results

Training sequence-based estimation of the signature sequences for the simple synchronous CDMA system discussed earlier is presented here. For the signature sequences, simple Hadamard codes are used which are strict-sense *orthogonal* leading to the structured autocorrelation matrix structure discussed earlier. The optimal detector for the desired user in the presence of three additional users which have a total interference power of 10.8 dB is computed. The detector has a processing gain N of 16 coefficients. Additional thermal white Gaussian noise is introduced into the receiver. Relative to this noise power, the SNR of the desired user is 30 dB. For this example, fading is not included, and the effects of the channel are negligible. The resulting signal-to-interference power ratio (SIR) and mean square error (MSE) are plotted for the received sequences in Figures 6.6(a) and 6.7(b), respectively. The signal power is simply defined as the output of the adaptive correlator squared. The interference power is the output of the correlator due to the thermal noise and the interfering users.

The exact implementation of Equation 6.15 is utilized to compute the adaptive coefficients of the correlator as described in Soni and Jenkins [41], rather than the approximate technique of Slock [8] and Gay and Tavatia [9].

This projection technique clearly outperforms the LMS technique albeit at a slightly higher overall mean square error. The computational cost of this technique is $O(L^2N)$



(a) (b)

Figure 6.7 Data-Reusing Adaptive Algorithms, (a) SIR, (b)MSE

where N is the processing gain of the receiver, and L is the number of users. Where $L << N$, this compares favorably to the RLS technique which possesses $O(N^2)$ complexity.

Figures 6.7(a) and 6.7(b) display the SIR ratio and MSE, respectively, for the data-reusing methods. As the number of reuses, M increases, the performance of the data-reusing algorithm more closely approximates the performance of the projection algorithms.

6.5 Fault Tolerant Projection Methods

Increasing concerns over the reliability of fast adaptive algorithms and their implementations has spawned interest and research in the development of algorithms which are capable of adapting to optimal or "nearly" optimal solutions in the presence of faults. Adaptive filtering coefficients are made robust to faults by making the coefficients redundant. In order to compensate for a single fault, a set of coefficients may be made fault-tolerant by adding a single additional adaptive filter coefficient. To compensate for up to R faults, R additional coefficients are added to the set N.

However, the inputs to the R coefficients are different from the other N. They are typically composed of the linear combination of the N delayed input samples. In order to illustrate some of the possibilities, the output of the adaptive filter for the Nth-order case with no faults can be expanded:

$$y(n) = W^H(n) \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} X(n), \tag{6.21}$$

where $W(n)$ is a vector of adaptive filter coefficients and $X(n)$ is a vector of the input data. To incorporate protection for one fault, the following modified filtering structure may be used:

$$y(n) = W^H(n) \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ 1 & \cdots & 1 \end{bmatrix} X(n), \tag{6.22}$$

where the vector of adaptive filter coefficients now has $N + 1$ coefficients [47]. This approach may be generalized further by writing the output of the adaptive filter as $y(n) = W^H(n)\mathbf{U}^H X(n)$. The matrix $\mathbf{U}$ insures fault tolerance to R faults if all sets of any of the N columns of the $M \times N$ dimensional matrix $\mathbf{U}$ are linearly independent where $M = N + R$ [47]. This set may be constructed by taking the first N columns of the M-dimensional DFT or DCT matrices. Alternatively, this can also be viewed as zero-padding the N-dimensional input vector to M values, and taking the DFT or DCT, i.e., by taking the DFT of the vector $X_e(n)$ where $X_e(n)$ is given by

$$X_e(n) = \left[X^T(n), 0, \ldots, 0\right]^T. \tag{6.23}$$

Note that many other constructions are possible for the matrix $\mathbf{U}$ [52].

Postfault convergence rates for LMS-based approaches to updating the adaptive filter coefficients will generally be poor for highly colored or correlated inputs. One possible solution to this problem would be to incorporate the vector NLMS-based approach (described in Section 1.3.3) or the *row projection* approach to updating the coefficients. For either of these approaches, the following minimization problem should be solved:

$$\begin{array}{rl} \min\limits_{W(n)} & ||W(n) - W(n-1)||_2^2 \\ \textit{subject to} & D(n) = \mathbf{X}^H(n)\mathbf{U}W(n). \end{array} \tag{6.24}$$

The *row projection*-based adaptive algorithms developed in this dissertation may be directly applied to the transformed system. And then the data reusing algorithms and accelerated data reusing algorithms may be used to improve the performance of this algorithm.

Alternatively, using the vector NLMS approach discussed in Section 1.3.3, an exact update expression for the coefficients can be expressed as

$$W(n) = W(n-1) - \mathbf{U}^H\mathbf{X}(n)[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1)]. \tag{6.25}$$

6.5.1 Incorporating fault knowledge in adaptive algorithms

The algorithm developed in Equation (6.25) does not have knowledge of the exact fault locations. Conceivably, there might be a fault detection mechanism designed into the system [20]. In such a case, an update expression for the adaptive filter coefficients, where knowledge of such faults is assumed, may be developed. Assuming that the goal of the algorithm is to compensate for faults occurring in up to R taps, the vectorized NLMS algorithm may be restated to include the effects of the R faults:

$$\begin{array}{rl} \min\limits_{W(n)} & ||W(n) - W(n-1)||_2^2 \\ \textit{subject to} & D(n) = \mathbf{X}^H(n)\mathbf{U}W(n) \\ & \mathbf{E}^H W(n) = C, \end{array} \tag{6.26}$$

where the matrix $\mathbf{E}$ is given by

$$\mathbf{E} = \left[e_{k_1}, \ldots, e_{k_R}\right], \tag{6.27}$$

where e_i is a vector with a one in the ith position and zeros elsewhere, and $\{k_1, \ldots k_R\}$ is an indexed set of integers belonging to $[0, M-1]$ which define the location of the R

failed taps. Ideally, we would like the optimization in Equation (6.25) to converge to the Wiener-Hopf solution regardless of the location of the k_j's.

If the faults can be identified, the accelerated *row projection* methods may be augmented directly to solve this alternative system. The vector NLMS with fault knowledge replaces $D(n)$ and $X(n)$ with $\tilde{D}(n)$ and $\tilde{X}(n)$, respectively, where

$$\begin{aligned} \tilde{D}(n) &= \begin{bmatrix} D(n) \\ C \end{bmatrix} \\ \tilde{\mathbf{X}}^H(n) &= \begin{bmatrix} \mathbf{X}^H(n)\mathbf{U} \\ \mathbf{E}^H \end{bmatrix}. \end{aligned} \tag{6.28}$$

The update for the adaptive coefficients in the vector NLMS case using fault knowledge becomes

$$\begin{aligned} W(n) &= W(n-1) - \begin{bmatrix} \mathbf{U}^H\mathbf{X}(n) & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n) & \mathbf{X}^H\mathbf{U}\mathbf{E} \\ \mathbf{E}^H\mathbf{U}^H\mathbf{X} & \mathbf{E}^H\mathbf{E} \end{bmatrix}^{-1} \\ &\times \begin{bmatrix} D(n) - \mathbf{X}_e^H(n)\mathbf{U}W(n-1)] \\ 0 \end{bmatrix}. \end{aligned} \tag{6.29}$$

While this direct implementation is easy enough to visualize, it can be computationally expensive to implement, particularly the computation of the inverse associated with this update. As an alternative, the CG algorithm can be used to solve the system $\mathbf{A}X = B$, where $\mathbf{A} = \tilde{\mathbf{X}}(n)\tilde{\mathbf{X}}^H(n)$ and $B = \tilde{\mathbf{X}}(n)\tilde{D}(n)$, similar to the approach of Boray and Srinath [11]. As an alternative, an implementation directly incorporating the faults as constraining conditions may also be developed. This approach will also solve the optimization problem stated in Equation (6.26) and will be developed in Section 6.5.2.

Row projection methods may be used to express the adaptive filtering process as the solution of a constrained solution of a linear system. The linear system associated with the adaptive filtering process is the set of zero error conditions $\mathbf{X}^H(n)W(n) = D(n)$ and the fault knowledge $\mathbf{E}^H W(n) = C$. Earlier, numerous methods have been described which can solve this type of linear system with data-reusing and accelerated-data reusing adaptive techniques.

6.5.2 Fault tolerant adaptive filtering algorithms (faults known)

To formulate a reduced closed-form solution for this optimization problem, the Lagrange multiplier method of optimization can be used. The following function can then be minimized:

$$\begin{aligned} J(W(n), \lambda_1, \lambda_2) &= [W(n) - W(n-1)]^H [W(n) - W(n-1)] + \\ &\quad \lambda_1^H [D(n) - \mathbf{X}^H(n)\mathbf{U}W(n)] + \lambda_2^H [\mathbf{E}^H W(n) - C]. \end{aligned} \tag{6.30}$$

Following the general prescription of Lagrange multiplier optimization, the derivatives of Equation (6.30) are taken with respect to $W(n)$, λ_1, and λ_2 and set equal to zero:

$$\frac{\delta J(W(n), \lambda_1, \lambda_2)}{\delta W(n)} = W(n) - W(n-1) + \mathbf{U}^H \mathbf{X}(n)\lambda_1 + \mathbf{E}\lambda_2 = 0 \tag{6.31}$$

$$\frac{\delta J(W(n), \lambda_1, \lambda_2)}{\delta \lambda_1} = [D(n) - \mathbf{X}^H(n)\mathbf{U}W(n)] = 0 \tag{6.32}$$

$$\frac{\delta J(W(n), \lambda_1, \lambda_2)}{\delta \lambda_2} = [\mathbf{E}^H W(n) - C] = 0. \tag{6.33}$$

Substituting Equation (6.31) into Equation (6.32) and Equation (6.33) yields the following two equations:

$$\begin{aligned} D(n) &= \mathbf{X}^H(n)\mathbf{U}[W(n-1) - \mathbf{U}^H \mathbf{X}(n)\lambda_1 - \mathbf{E}\lambda_2] \\ C &= \mathbf{E}^H [W(n-1) - \mathbf{U}^H \mathbf{X}(n)\lambda_1 - \mathbf{E}\lambda_2]. \end{aligned} \tag{6.34}$$

Solving first for λ_1 and λ_2 yields

$$\begin{aligned} \lambda_1 &= -[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H \mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1) + \mathbf{X}^H(n)\mathbf{U}\mathbf{E}\lambda_2] \\ \lambda_2 &= -[\mathbf{E}^H \mathbf{E}]^{-1}[C - \mathbf{E}^H W(n-1) + \mathbf{E}^H \mathbf{U}^H \mathbf{X}(n)\lambda_1]. \end{aligned} \tag{6.35}$$

Since the fault has occurred at the previous instant $C - \mathbf{E}^H W(n-1) = 0$, and $\mathbf{E}^H\mathbf{E} = \mathbf{I}$, the expression for λ_2 simplifies greatly to

$$\lambda_2 = -\mathbf{E}^H \mathbf{U}^H \mathbf{X}(n)\lambda_1. \tag{6.36}$$

Substituting this into the expression for λ_1 yields

$$\lambda_1 = -[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1) - \mathbf{X}^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)\lambda_1]. \tag{6.37}$$

Solving for λ_1^* yields

$$\begin{aligned}\lambda_1^* &= -[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}\mathbf{X}^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)]^{-1}[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1} \\ &\quad \times[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1)]. \end{aligned} \tag{6.38}$$

Further, λ_2^* is obtained

$$\begin{aligned}\lambda_2^* &= \mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}\mathbf{X}^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)]^{-1} \\ &\quad \times[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1)]. \end{aligned} \tag{6.39}$$

Using λ_1^* and λ_2^*, the optimal update expression for $W(n)$, becomes

$$\begin{aligned}W(n) &= W(n-1) + \mathbf{U}^H\mathbf{X}(n)[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}\mathbf{X}^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)]^{-1} \\ &\quad \times[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1)] \\ &\quad -\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}\mathbf{X}^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)]^{-1} \\ &\quad \times[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1)]. \end{aligned} \tag{6.40}$$

This expression simplifies to

$$\begin{aligned}W(n) &= W(n-1) + [\mathbf{I} - \mathbf{E}\mathbf{E}^H]\mathbf{U}^H\mathbf{X}(n)[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}\mathbf{X}^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}(n)]^{-1} \\ &\quad \times[\mathbf{X}^H(n)\mathbf{U}\mathbf{U}^H\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}^H(n)\mathbf{U}W(n-1)]. \end{aligned} \tag{6.41}$$

If $\mathbf{U}^H$ is chosen as the first N columns of the M-dimensional *unitary* DFT, and the data vectors are zero-padded to enforce this choice, the update expression becomes

$$\begin{aligned} W(n) &= W(n-1) + [\mathbf{I} - \mathbf{E}\mathbf{E}^H]\mathbf{U}^H\mathbf{X}_e(n)[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}\mathbf{X}_e^H(n)\mathbf{U}\mathbf{E}\mathbf{E}^H\mathbf{U}^H\mathbf{X}_e(n)]^{-1} \\ &\quad \times[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}_e^H(n)\mathbf{U}W(n-1)]. \end{aligned} \tag{6.42}$$

Further defining the projection matrices $\mathbf{P}_{R(\mathbf{E})} = \mathbf{E}\mathbf{E}^H$ and $\mathbf{P}_{N(\mathbf{E}^H)} = [\mathbf{I} - \mathbf{E}\mathbf{E}^H]$ results in the following update expression:

$$\begin{aligned} W(n) &= W(n-1) + \mathbf{P}_{N(\mathbf{E}^H)}\mathbf{U}^H\mathbf{X}_e(n)[\mathbf{I} - [\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}\mathbf{X}_e^H(n)\mathbf{U}\mathbf{P}_{R(\mathbf{E})}\mathbf{U}^H\mathbf{X}_e(n)]^{-1} \\ &\quad \times[\mathbf{X}^H(n)\mathbf{X}(n)]^{-1}[D(n) - \mathbf{X}_e^H(n)\mathbf{U}W(n-1)]. \end{aligned} \tag{6.43}$$

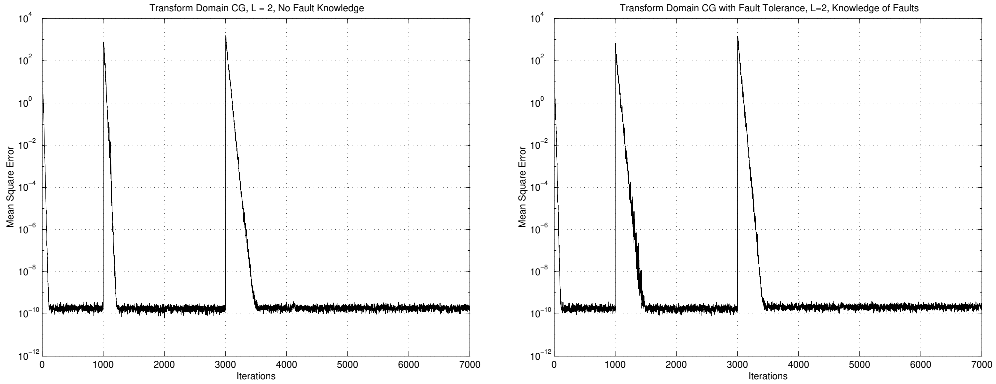
The update expression now appears similar to the update expression for the algorithm with no fault knowledge. The update in the coefficients is projected away from the span of the faulty coefficients. Potentially, this method might offer some benefit in terms of reduced computational complexity over the method in Equation 6.29, but it may also require a more complicated implementation. While this expression is simplified, the resulting implementation can still be difficult.

Potentially, the *row projection* methods developed by the authors or the CG algorithm of Boray and Srinath [11] offers the simplest level of understanding to the system designer attempting to realize a fault tolerant system.

6.5.3 Experimental evaluation

In order to determine if knowledge of the faults made any difference in the performance of the adaptive filter convergence rate, the performance of the accelerated *row projection* method developed in [29] in the presence and absence of fault knowledge is examined. When the fault is known, the accelerated data reusing algorithms solve the optimization problem stated in Equation (6.26). When the faults are unknown, the accelerated data reusing algorithms solve the optimization problem associated with Equation (6.24).

A simple set of system identification experiments was performed where stuck-at value faults occurred. The unknown system had eight taps and R was set to 2. This system



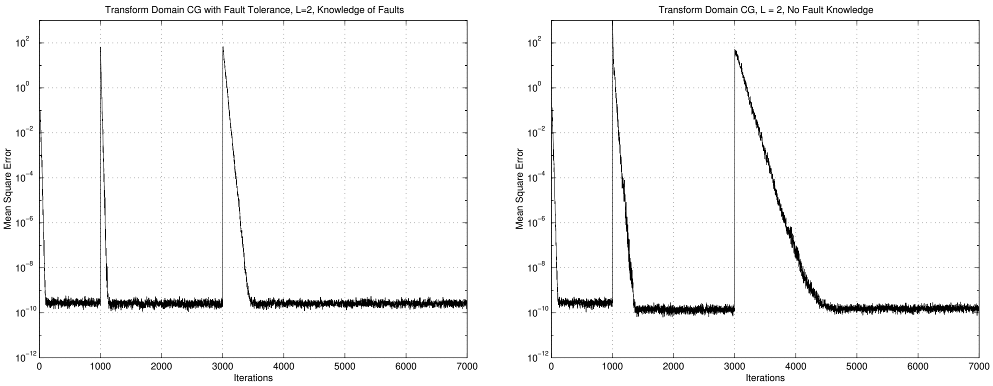
(a) (b)

Figure 6.8 System Identification, White Noise Input, (a) No Fault Knowledge and (b) Fault Knowledge

could in theory compensate for up to two faults. The results of the accelerated *row projection* algorithm where faults were both unknown and known for white noise are displayed in Figures 6.8(a) and 6.8(b), respectively.

The data from these plots does not indicate a significant disparity for the case of white noise input. But, the knowledge of faults does slightly improve the postfault convergence rate after the second fault. However, it appears to lead to a slightly slower convergence rate after the first fault.

The same experiment was repeated for the case where the input signal was colored noise. The resulting mean square error plots in Figures 6.9(a) and 6.9(b) illustrate the large disparity in the system that occurs when the knowledge of the faults is completely unknown. For this simple case, the postfault convergence after the second fault illustrates that knowledge of the fault can improve convergence behavior over the system where no fault knowledge is assumed.



(a) (b)

Figure 6.9 System Identification, Colored Noise Input, (a) Fault Knowledge, (b) No Fault Knowledge

CHAPTER 7

CONCLUSION

The principle contribution of this dissertation has been the development of a "new" framework for explaining the operation of a broad class of adaptive algorithms. This framework has its origins in an optimization strategy which has existed for over 50+ years. Using this framework, several fundamentally new algorithms have been developed which allow system designers the flexibility to decide how much performance they desire as a function of the computational complexity that they are willing to support.

Essentially, it provides a deterministic framework for viewing the operation of adaptive algorithms which were derived using stochastic considerations. The LMS algorithm and the RLS or fast quasi-Newton algorithms were shown to be linked using this framework. Using a generalized algorithm notation, the LMS algorithm was shown to offer low computational complexity and implied lesser performance at one extreme, and the RLS algorithm offered higher performance and computational complexity at the other extreme.

This framework explains the operation of the new algorithms in a projection framework. Convergence occurs to the optimal solution by constraining the method and direction of the projection. The normalized LMS algorithms were seen to be the simplest implementation and lowest rank performers of the various algorithms demonstrated. Normalized data-reusing algorithms were shown to offer scalable performance improvements as a function of increased computational complexity. Acceleration of the data-reusing methods was shown to offer even better performance in terms of convergence rate.

Fast quasi-Newton algorithms were also placed in this framework. Using the conjugate gradient algorithm to solve the matrix inversion problem associated with the computation of the Kalman gain relates the two outwardly appearing different approaches to solving the adaptive filtering problem. The conjugate gradient algorithm allowed the operation

Table 7.1 Summary of Computational Complexity of Algorithms

Normalized LMS	$O(N)$
RLS	$O(N^2)$
Data Reusing	$O(RN)$
Kaczmarz Data Reusing	$O(RLN)$
Cimmino Data Reusing	$O(RLN)$
CG Accelerated Kaczmarz Data Reusing	$O(L^2N)$
Tchebyshev Accelerated Kaczmarz Data Reusing	$O(KLN)$
CG Accelerated Cimmino Data Reusing	$O(L^2N)$
Fast Quasi Newton via CG	$O(L^2N)$

of the fast quasi-Newton algorithm to be viewed as an algorithm which utilized repeated projections.

The computational complexities of the various algorithms discussed in this dissertation are summarized in Table 7.1. Each of the new algorithms presented offers the ability to trade computational complexity for performance. The parameters R and L are the scaling parameters associated with each algorithm. Increasing R and L improves the convergence rate performance but also increases the computational complexity. In addition, as already shown, it raises the excess mean square error associated with the algorithm. Further research opportunities exist in the applications of this framework to further relate and explain additional algorithms. In addition, there are additional algorithms which may be translated from a traditional *row projection* scenario to adaptive filtering applications to offer further computational and performance advantages.

These algorithms have all been applied exclusively to the direct-form finite impulse response (FIR) adaptive structures. Further work may be done to apply these results to infinite impulse response (IIR) adaptive structures. IIR filters have attracted much attention for use in adaptive systems, and the use of these filters can reduce the number of coefficients required to realize a system. Unfortunately, IIR filters can be slow to converge and become unstable. Hopefully, these algorithms will lend themselves to algorithms which converge rapidly and offer robustly stable performance. Future researchers can hopefully apply these techniques without difficulty to a variety of filtering structures.

A fairly complete survey of the new techniques has been provided. A detailed explanation of the framework needed to explain the new techniques in the context of the existing adaptive filtering literature has also been presented. Applications of these techniques to many existing adaptive filtering problems such as system identification, channel equalization, interference suppression, two-dimensional adaptive processing, and fault tolerant adaptive filtering problems have been presented and discussed. Use of the new techniques and the new framework have given new insight into existing problems and presented system designers with an abundance of new approaches to solve adaptive filtering problems.

APPENDIX A

CONJUGATE GRADIENT ALGORITHM APPLIED TO LOW-RANK PROBLEMS

The conjugate gradient algorithm is a method of minimization which requires no matrix inversion and computes the solution to the matrix problem

$$\mathbf{Q}x = b. \tag{A.1}$$

More specifically, the conjugate gradient algorithm minimizes the quadratic function

$$J(x) = \frac{1}{2}x^H\mathbf{Q}x - b^H x. \tag{A.2}$$

Generally, discussions of the conjugate gradient algorithm presume that the matrix $\mathbf{Q}$ is symmetric and strictly positive definite. Yet, the conjugate gradient algorithm may be readily applied to systems where $\mathbf{Q}$ is not full-rank. In those cases, the question then becomes, what is the convergence point of the conjugate gradient algorithm? Does it converge to the min-norm solution?

A.1 Conjugate Gradient Algorithm: New Definitions and Algorithm

For the answers to the questions posed about the conjugate gradient algorithm, some basic definitions for the conjugate gradient algorithm are restated [30, 53] and modified to apply them to cases where the matrix $\mathbf{Q}$ is not full-rank. The first definition is a restatement of the definition of conjugate direction vectors.

Definition A.1 *Given a symmetric matrix $\mathbf{Q}$, two vectors d_1 and d_2 are said to be $\mathbf{Q}$-orthogonal, or conjugate with respect to $\mathbf{Q}$, if $d_1^H\mathbf{Q}d_2 = 0$*

This original theorem helps develop the general result conjugate gradient algorithm where the system is not full-rank.

Theorem A.1 *The number of nonzero conjugate directions vectors of a symmetric matrix* $\mathbf{Q}$ *contained in the range space of* $\mathbf{Q}$ *which are linearly independent is equal to* L *where* $L > 1$ *is the rank of the matrix* $\mathbf{Q}$.

Proof: The dimension of $\mathbf{R}(\mathbf{Q})$ is L where L linearly independent vectors may be used to describe the $\mathbf{R}(\mathbf{Q})$. Let d_0, d_1, ..., d_{L-1} be $\mathbf{Q}$-conjugate vectors contained in the range space of $\mathbf{Q}$. To complete the proof, we must show that these vectors constitute a linearly independent set. Suppose that there are constants α_i, $i = 0, \ldots, L-1$ such that

$$\alpha_0 d_0 + \alpha_1 d_1 + \ldots + \alpha_{L-1} d_{L-1} = 0.$$

Premultiplying by $\mathbf{Q}$ and taking the scalar product with d_i yields

$$\alpha_i d_i^H \mathbf{Q} d_i = 0.$$

Since $d_i \in \mathbf{R}(\mathbf{Q}) = \mathbf{R}(\mathbf{Q}^H)$, this implies that $d_i \notin \mathbf{N}(\mathbf{Q})$ and $d_i^H \mathbf{Q} d_i > 0$. Thus, we have $\alpha_i = 0$ which implies the linear independence of the vectors.

Note that this theorem does not include the case for $L = 1$. For this case, there clearly is no such thing as $\mathbf{Q}$-conjugacy. Further, the definition does not quantify the number of vectors which lie in the $\mathbf{N}(\mathbf{Q})$. Clearly, any two vectors observe $\mathbf{Q}$-conjugacy if either one of them lies in the null space of $\mathbf{Q}$. $\square$

The conjugate gradient algorithm may be easily described by the following set of equations. Starting at any $x_0 \in \mathbf{C}^N$, define $d_0 = -g_0 = b - \mathbf{Q}x_0$ and

$$\begin{aligned}
x_{k+1} &= x_k + \alpha_k d_k && \text{(A.3)}\\
\alpha_k &= -\frac{g_k^H d_k}{d_k^H \mathbf{Q} d_k} && \text{(A.4)}\\
d_{k+1} &= -g_{k+1} + \beta_k d_k && \text{(A.5)}\\
\beta_k &= \frac{g_{k+1}^H g_{k+1}}{g_k^H g_k}, && \text{(A.6)}
\end{aligned}$$

where $g_k = \mathbf{Q}x_k - b$.

A.2 Convergence Results

A theorem has been developed which describes the convergence behavior of the conjugate gradient algorithm for the special case where $\mathbf{Q}$ is rank-deficient, i.e., $\rho(\mathbf{Q}) < N$.

Theorem A.2 *The conjugate gradient algorithm computes the solution x^*_{cg} in L steps for the N-dimensional system $\mathbf{Q}x = b$, where x_0 is the initial condition for the CG algorithm if b is contained in the range space of $\mathbf{Q}$, $\mathbf{R}(\mathbf{Q})$, if the rank of $\mathbf{Q}$ is L.*

- **Case 1:** $x_0 = 0$
$$x^*_{cg} = x_{MN} = \mathbf{Q}^\dagger b,$$
where $\mathbf{Q}^\dagger$ is the Moore-Penrose pseudo-inverse of $\mathbf{Q}$ and this represents the minimum-norm solution to the system $\mathbf{Q}x = b$.

- **Case 2:** $x_0 \in \mathbf{R}(\mathbf{Q})$
$$x^*_{cg} = x_{MN} = \mathbf{Q}^\dagger b.$$

- **Case 3:** $x_0 = \tilde{x}_0 + \tilde{x}_0^\dagger$ *where* $\tilde{x}_0 = \mathbf{P}_{\mathbf{R}(\mathbf{Q})} x_0$, $\tilde{x}_0^\dagger = \mathbf{P}_{\mathbf{N}(\mathbf{Q})} x_0$, *and* $\tilde{x}_0^H \tilde{x}_0^\dagger = 0$.
$$x^*_{cg} = x_{MN} + \tilde{x}_0^\dagger = \mathbf{Q}^\dagger b + \tilde{x}_0^\dagger.$$

Proof: First, the special case where $L = 1$ is examined. In this case, the $\mathbf{Q}$ conjugacy of vectors cannot be used to describe convergence behavior. Let x_0 be defined as in *Case 3* of the above theorem. In this case, the algorithm can be evaluated as follows:

$$d_0 = b - \mathbf{Q}x_0 = b - \mathbf{Q}\tilde{x}_0 \tag{A.7}$$
$$g_0 = -d_0 \tag{A.8}$$
$$\alpha_0 = \frac{(b - \mathbf{Q}\tilde{x}_0)^H (b - \mathbf{Q}\tilde{x}_0)}{(b - \mathbf{Q}\tilde{x}_0)^H \mathbf{Q} (b - \mathbf{Q}\tilde{x}_0)} \tag{A.9}$$
$$x_1 = \tilde{x}_0 + \tilde{x}_0^\dagger + \alpha_0 d_0. \tag{A.10}$$

The matrix $\mathbf{Q}$ can be written in terms of a vector outer product since the rank is 1 as $v^H v$. Also, $b, \tilde{x}_0 \in \mathbf{R}(\mathbf{Q})$ which means that α_0 can be simplified to $\alpha_0 = \frac{1}{v^H v}$. Then, the expression for x_1 becomes

$$\begin{aligned} x_1 &= \tilde{x}_0 + \tilde{x}_0^\dagger + \frac{1}{v^H v}(b - \mathbf{Q}\tilde{x}_0) && \text{(A.11)} \\ &= \tilde{x}_0^\dagger + (\mathbf{I} - v[v^H v]^{-1} v^H)\tilde{x}_0 + \frac{1}{v^H v} b && \text{(A.12)} \\ &= \tilde{x}_0^\dagger + (\mathbf{I} - \mathbf{P}_{\mathbf{R}(\mathbf{Q})})\tilde{x}_0 + \frac{1}{v^H v} b && \text{(A.13)} \\ &= \tilde{x}_0^\dagger + \mathbf{P}_{\mathbf{N}(\mathbf{Q})}\tilde{x}_0 + \frac{1}{v^H v} b && \text{(A.14)} \\ &= \tilde{x}_0^\dagger + \mathbf{Q}^\dagger b, && \text{(A.15)} \end{aligned}$$

where $\mathbf{P}_{\mathbf{R}(\mathbf{Q})} = v[v^H v]^{-1} v^H$ is used to simplify Equation (A.12) to Equation (A.13). Further the fact that $\mathbf{Q}^\dagger b = \frac{1}{v^H v} b$ for $b \in \mathbf{R}(\mathbf{Q})$ is used to reduce Equation (A.14) to Equation (A.15).

In terms of the sequence of the nonzero $\mathbf{Q}$-conjugate vectors d_0, d_1, $\ldots$, d_{L-1} which are contained in $\mathbf{R}(\mathbf{Q})$, the difference between the minimum norm solution x_{MN} and the initial value of x can be written as

$$x_{MN} - x_0 = \alpha_0 d_0 + \alpha_1 d_1 + \ldots + \alpha_{L-1} d_{L-1} + \tilde{x}_0^\dagger. \tag{A.16}$$

By premultiplying Equation A.16 by $\mathbf{Q}$ and taking the scalar product with d_k, the optimal value of α_k, the step size, may be derived as

$$\alpha_k = \frac{d_k^H \mathbf{Q}(x_{MN} - x_0)}{d_k^H \mathbf{Q} d_k}. \tag{A.17}$$

Following the iterative process from x_0 to x_k gives

$$x_k - x_0 = \alpha_0 d_0 + \alpha_1 d_1 + \ldots + \alpha_{L-1} d_{k-1} + \tilde{x}_0^\dagger. \tag{A.18}$$

By the $\mathbf{Q}$-conjugacy of the d_k's, the following can be shown:

$$d_k^H \mathbf{Q}(x_k - x_0) = 0. \tag{A.19}$$

Substituting Equation (A.19) into Equation (A.17) yields

$$
\begin{aligned}
\alpha_k &= \frac{d_k^H \mathbf{Q}(x_{MN} - x_k)}{d_k^H \mathbf{Q} d_k} \\
&= \frac{d_k^H (b - \mathbf{Q} x_k)}{d_k^H \mathbf{Q} d_k} \\
&= -\frac{g_k^H d_k}{d_k^H \mathbf{Q} d_k},
\end{aligned} \tag{A.20}
$$

which is the same α given in the conjugate gradient algorithm description.

Finally, in order to complete the proof, we must show that the vectors generated by the conjugate gradient algorithm are $\mathbf{Q}$-conjugate. To do this, we must apply the general results of the *conjugate gradient theorem* described in Luenberger [30]. Specifically, we must show that the direction vectors generated by this algorithm are, in fact, $\mathbf{Q}$-conjugate. $\square$

APPENDIX B

CHANNEL EQUALIZATION AND THE CONSTANT MODULUS ALGORITHM

The use of the adaptive filters to identify an unknown channel has become an important application of adaptive filters. In order to identify the channel, a training sequence is usually broadcast prior to beginning transmission. If the channel changes significantly, then the training sequence may be periodically retransmitted. Unfortunately, increasing bandwidth efficiency and difficulties with re-synchronization with the training sequence have motivated system designers to research and develop methods which do not require a training sequence. First and foremost of these methods is the constant modulus algorithm. The constant modulus algorithm has a performance function defined as follows:

$$J(n) = E[||y(n)|^p - |\delta|^p|^q] \tag{B.1}$$

where $y(n) = X^T(n)W(n)$ is the output of the adaptive filter, and δ is the expected modulus of the signal. The pioneering form of this algorithm was developed by Sato [54] in 1975 for the special case where $p = 1$, and $q = 2$. In 1980, Godard [55] formulated algorithms for the case where p and q were arbitrary. Treichler and Agee [56] studied the case where $p = q = 2$, which is the most popular form of the algorithm and the form considered in this appendix. Numerous versions of the stochastic gradient version of this algorithm have been developed. Transform domain, quasi-Newton, lattice versions, and a variety of other methods exist to update the coefficients on a sample-by-sample basis. Schirtzinger [57] and Papadias [19] have developed comprehensive summaries of available techniques for updating these coefficients. In his dissertation, Papadias [19] developed an affine projection approach to solving the constant modulus problem.

The focus of this appendix is to relate the stochastic gradient solutions of the constant modulus function to the solution of a linear system. Using this approach, the algorithms developed using a *row projection* framework may be utilized to develop new algorithms for the constant modulus approach to adaptive filtering. The Kaczmarz and Cimmino data reusing methods, and the affine projection-based adaptive filtering method, discussed in Chapter 1, may all be applied to this adaptive filtering problem.

The stochastic gradient algorithm leads to minimizing the instantaneous constant modulus error $y^2(n) - \delta^2$. The equivalent quadratic system for this instantaneous error will be

$$|X^T(n)W(n)|^2 = \delta^2. \tag{B.2}$$

The resulting stochastic gradient or LMS algorithm for this approach is

$$W(n) = W(n-1) - \mu[|X^H(n)W(n)|^2 - \delta^2][X^H(n)W(n)]X(n). \tag{B.3}$$

The Kaczmarz data reusing strategy can be used to improve the performance of the stochastic gradient algorithm. The equivalent Kazcmarz strategy for the constant modulus algorithm can be implemented as follows:

- **Step 1:** *Initialization :* $i = 0$

$$C_0(n) = W(n). \tag{B.4}$$

- **Step 2:** *Loop:* (*While* $i < L+1$)

$$C_{i+1} = C_i + \alpha_i[X^H(n-i)C_i]X^H(n-i) \tag{B.5}$$

$$\alpha_i = \frac{|X^H(n-i)c_i|^2 - \delta^2}{X^H(n-i)X(n-i)} \tag{B.6}$$

$$i = i+1. \tag{B.7}$$

- **Step 3:** $W(n+1) = C_{L+1}$, $n = n+1$, and *goto Step 1.*

The Cimmino data-reusing strategy can be applied to the constant modulus algorithm as well. For the Cimmino algorithm, the update can be given as

$$W(n) = W(n-1) + \frac{2}{L}\sum_{i=0}^{L-1}\alpha_i[X^H(n-i)W(n-1)]X^H(n-i), \tag{B.8}$$

where

$$\alpha_i = \frac{|X^H(n-i)W(n-1)|^2 - 1}{X^H(n-i)X(n-i)}. \tag{B.9}$$

The accelerated data-reusing algorithms are difficult to apply to this performance criterion due to the inability to manipulate the update expressions for the adaptive coefficients into a form like

$$W(n) = \mathbf{Q}(n)W(n) + D_\pi(n). \tag{B.10}$$

However, ideas from the affine projection techniques may be used to develop a higher performance algorithm. Similar algorithms have been developed in several different sources [19, 57]. In this dissertation, the constant modulus algorithm will be examined from a *row projection* framework. Similar to the MSE criterion commonly associated with adaptive filter updates, the constant modulus criterion is satisfied at the current instant, but also with the data from the previous instant. In the notation of this section, the following conditions are satisfied:

$$\begin{aligned} |X^H(n)W(n)|^2 &= \delta^2 \\ |X^H(n-1)W(n)|^2 &= \delta^2 \\ &\vdots \\ |X^H(n-L+1)W(n)|^2 &= \delta^2. \end{aligned} \tag{B.11}$$

Combining these L equations yields the following quadratic system:

$$W^H(n)\mathbf{X}(n)\mathbf{X}^H(n)W(n) = \delta^2 L. \tag{B.12}$$

This system is not a consistent set of equations. Due to the structure, the solution to this set of equations is not unique. For illustration purposes, let $\mathbf{U}\Sigma\mathbf{U}^H = \mathbf{X}(n)\mathbf{X}^H(n)$ be the singular value decomposition of the L-rank matrix $\mathbf{X}(n)\mathbf{X}^H(n)$. Further, let the vector $Z(n) = \mathbf{U}^H W(n)$. Note that the dimension of $Z(n)$ is $L \times 1$. The set of equations in Equation (B.12) reduces to

$$\sum_{i=0}^{L-1} \sigma_i^2 Z_i^2(n) = \delta^2 L. \tag{B.13}$$

This is a polynomial of L variables with an infinite number of valid solutions. Hence, no solution to the system given in Equation (B.12) is unique. Rather than just solve these equations which are not uniquely determined, an additional constraint is incorporated into the optimization. More specifically, the following must be satisfied:

$$\min_{W(n)} ||W(n) - W(n-1)||_2^2. \tag{B.14}$$

Together, these two constraints uniquely determine the update expression for the adaptive filter coefficients. Based upon this constraint, the conjugate gradient algorithm will yield the solution to the quadratic system in Equation (B.12). Use of the conjugate gradient algorithm will insure that the minimum distance constraint stated in Equation (B.14) will be satisfied. The system in Equation (B.12) must be restated in terms of a minimization problem. More specifically, it may be stated as

$$J(W(n)) = [|\mathbf{X}^H(n)W(n)|^2 - \delta^2 L]^2. \tag{B.15}$$

Now, the gradient and Hessian of $J(W(n))$ may be readily computed for the implementation of the CG algorithm. They are derived in [57] and are restated here for reference:

$$\begin{aligned} g(n) &= 4[|X^H(n)W(n)|^2 - \delta^2](X^H(n)W(n))X(n) \\ H(n) &= 4X(n)X^H(n)[3|X^H(n)W(n)|^2 - \delta^2], \end{aligned} \tag{B.16}$$

where $g(n)$ and $H(n)$ represent the gradient and Hessian, respectively, of the performance function in Equation (B.15).

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory.* Upper Saddle River NJ: Prentice-Hall 1996.

[2] J. Cioffi and T. Kailath, "Fast, recursive-least squares transversal filters for adaptive filtering," *IEEE Trans. Acoust. Speech, and Signal Proc.*, vol. 32, no. 2, pp. 304-337, Apr. 1984.

[3] J. Cioffi and T. Kailath, "Windowed fast transversal adaptive algorithms with normalization," *IEEE Trans. Acoust., Speech, and Signal Proc.*, vol. 33, no. 3, pp. 607-625, Jun. 1985.

[4] D. Slock, "Fast transversal filters with data sequence weighting," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. 37, no. 3, pp. 346-359, Mar. 1989.

[5] M. Montazeri and P. Duhamel, "A set of algorithms linking NLMS and block RLS algorithms," in *Proc. IEEE ICASSP*, Minneapolis MN, vol. 3, 1993, pp. 412-415.

[6] M. Montazeri and P. Duhamel, "A set of algorithms linking NLMS and block RLS algorithms," *IEEE Trans. Signal Processing*, vol. 43, no. 2, pp. 444-453, Feb. 1995.

[7] D. Slock, "The block underdetermined covariance (BUC) fast transversal (FTF) algorithm for adaptive filtering," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, Asilomar CA., Nov. 1992, pp. 550-554.

[8] D. Slock, "Underdetermined growing and sliding window covariance fast transversal filter RLS algorithms," in *Proc. EUSIPCO '92 VIth European Signal Processing Conference*, Brussels, Belgium, Aug. 1992, pp. 1169-1172.

[9] S. Gay and S. Tavatia, "The fast affine projection algorithm," in *Proc. IEEE ICASSP*, Detroit, MI, 1995, pp. 3023-3026.

[10] K. S. Arun, *Fundamentals of Digital Signal and Spectral Analysis*, Course Notes, University of Illinois at Urbana-Champaign, Urbana, IL, 1993.

[11] G. Boray and M. Srinath, "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. Circuits Systems*, vol. 39, no. 1, pp. 1-10, Jan. 1992.

[12] S. Gay "A fast converging, low complexity adaptive filtering algorithm," *Lannion*, Sept. 1993.

[13] M. Tanaka et al. "Fast projection algorithm and its step size control," in *Proc. IEEE ICASSP*, Detroit MI, 1995, pp. 945-948.

[14] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using and orthogonal projection to an affine subspace and its properties," *Electron. Commun. Jpn.* vol. 67-A, no. 5, 1984.

[15] S. Douglas, "Efficient approximate implementations of the fast affine projection algorithm using orthogonal transforms," in *Proc. IEEE ICASSP*, Atlanta, GA, 1996, pp. 1656-1659.

[16] K. Maouche and D. Slock, "A fast instrumental variable affine projection algorithm," in *Proc. IEEE ICASSP*, Seattle, WA, 1998.

[17] D. Morgan and S. Kratzer, "On a class of computationally efficient, rapidly converging, generalized NLMS algorithms," *IEEE Signal Proc. Letters*, vol. 3, no. 8, pp. 245-247, Aug. 1996.

[18] C. Papadias and D. Slock, "Normalized sliding window constant modulus and decision-directed algorithms: A link between blind equalization and classical adaptive filtering," *IEEE Trans. Signal Proc.* vol. 45, no. 1. pp. 231-235. Jan. 1997.

[19] C. Papadias, "Methods for blind equalization and identification of linear channels," Ph.D. dissertation, Ecole Nationale Superieure des Telecommunications, France, 1995.

[20] B. Schnaufer, "Practical techniques for rapid and reliable real-time adaptive filtering," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[21] B. Schnaufer, "New data-reusing LMS Algorithms for improved convergence," in *Proc. Twenty-Seventh Asilomar Conf. Signals Systems, and Computers*, Pacific Grove, CA, 1993.

[22] G. Goodwin, and K. S. Sin, *Adaptive Filtering, Prediction, and Control.* Englewood Cliffs, NJ: Prentice-Hall 1984.

[23] D. Marshall and W. Jenkins, "A fast quasi-Newton adaptive filtering algorithm," *IEEE Trans. Signal Proc.* vol. 40, no. 7, pp. 1652-1662, Jul. 1992.

[24] R. Bramley, "Row projection methods for linear systems," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1989.

[25] K. Tanabe, "Projection method for solving a singular system of linear equations and its applications," *Numer. Math.*, vol. 17, pp. 203-214, 1971.

[26] S. Kaczmarz, "Angenäherte auflösung von systemen linear engleichungen," *Bull. Int. Acad. Polon. Sci. Lett. Cl. Sci. Math.. Nat. A*, 1937.

[27] G. Cimmino, "Calcolo approssimato per le soluzioni dei sistemi di equazionii lineari," *Ric. Sci.. Prog. Techn. Econom. Naz.*, vol. 9, pp. 326-333, 1939.

[28] R. Soni, W. Jenkins, and K. Gallivan, "Applications of row-projection methods for improved performance in adaptive filtering," in *Proc. European Conf. on Circuit Theory and Design*, Budapest, Hungary. Sep. 1997.

[29] R. Soni, K. Gallivan, and W. Jenkins, "Projection methods for improved performance in FIR adaptive filtering," in *Proc. Midwest Symp. on Circuits and Systems*, Sacramento CA. Aug. 1997.

[30] D. Luenberger, *Linear and Non-Linear Programming.* Reading, MA: Addison-Wesley, 1984.

[31] D. Young, *Iterative Solution of Large Linear Systems*, New York NY: Academic Press, 1971.

[32] L. Hageman and D. M. Young, *Applied Iterative Methods.* New York NY: Academic Press 1981.

[33] D. Flanders and G. Shortley, "Numerical determination of fundamental modes," *Journal of Applied Physics*, vol. 21, pp. 1326-1332, 1950.

[34] G. Cheung, "Efficiency of data reusing methods," M.S. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1994.

[35] B. Widrow and S. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall 1984.

[36] S. Nelson and Neumann, "Generalizations of the projection method with applications to SOR theory for Hermitian positive semidefinite linear systems," *Numer. Math.*, vol. 51, pp. 123-141, 1987.

[37] R. Oldenburger, "Infinite powers of matrices and characteristic roots," *Duke Math Journal*, vol. 6, pp. 357-361, 1940.

[38] A. Bjorck and G. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of Computation*, vol. 127, number 123, pp. 579-594. Jul. 1973.

[39] D. Marshall, "Computationally efficient techniques for rapid convergence of adaptive digital filters," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1988.

[40] A. W. Hull, "Orthogonalization techniques for adaptive filters," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1994.

[41] R. Soni and W. Jenkins, "Channel equalization with adaptive filtering and the preconditioned conjugate gradient algorithm," in *Proc. Int'l Symp. on Circuits and Systems*, Hong Kong, Jun. 1996.

[42] R. Soni, T. Schirtzinger, and W. Jenkins, "Learning strategies for adaptive equalizers using the constant modulus criterion," to appear in *The Int'l Journal on Adaptive Control and Signal Processing.*

[43] M. Ng and R. Plemmons, "LMS-Newton adaptive filtering using FFT-based conjugate gradient iterations," *Elect. Trans. on Numer. Anal.*, vol. 4, pp. 14-36, Mar. 1996.

[44] T. Chan, "An optimal circulant preconditioner for Toeplitz systems," *SIAM Journal Sci. Stat. Comput.*, vol. 9, no. 4, pp. 766-771, Jul. 1988.

[45] R. Chan and G. Strang, "Toeplitz equations by conjugate gradients with circulant preconditioner," *SIAM Journal Sci. Stat. Comput.* vol. 10, no. 1, pp. 104-119, Jan., 1989.

[46] T. Ku and C. Kuo, "Design and analysis of toeplitz preconditioners," *IEEE Trans. Signal Process.*, vol. 40, no. 1, pp. 129-141, Jan. 1992.

[47] W. Jenkins, A. Hull, J. Strait, B. Schnaufer, and X. Li, *Advanced Concepts in Adaptive Signal Processing.* Norwell, MA:Kluwer Press 1996.

[48] M. Howard, R. Soni, and W. Jenkins, "Two-dimensional transform domain adaptive filtering," in *Proc. Twenty-Eighth Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, 1994.

[49] R. Soni and W. Jenkins, "Fast two-dimensional adaptive IIR algorithms," in *Proc. Int'l Symp. on Circuits and Systems*, Atlanta GA. May, 1996.

[50] M. Honig, U. Madhow, and S. Verdu, "Blind adaptive multiuser detection," *IEEE Trans. on Info. Theory*, vol. 41, no. 4, pp. 944-960, Jul. 1995.

[51] L. Zhu and U. Madhow, "Adaptive interference suppression for DS-CDMA over a Rayleigh fading channel," in *Proc. Conf. Inform. Sci. Sys. (CISS'97)*, Johns Hopkins University, Baltimore, MD, March 1997.

[52] B. Schnaufer and W. Jenkins, "Adaptive fault tolerance for reliable LMS adaptive filtering," *IEEE Trans. Circ. and Sys., Part II: Analog and Digital Signal Proc.* vol. 44, no. 12, pp. 1001-1014, Dec. 1997.

[53] G. Golub and C. Van Loan, *Matrix Computations*, Baltimore, MD: The Johns Hopkins University Press 1996.

[54] Y. Sato, "A method of self-recovering equalization for multi-level amplitude modulation systems," *IEEE Trans. on Comm.*, vol. 23, pp. 679-682, 1975.

[55] D. N. Godard, "Self recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. on Comm.*, vol. 28, pp. 1867-1875, 1980.

[56] J. Treichler and B. Agee, "A new approach to multipath correction of constant modulus signals," *IEEE Trans. on Acoust. Speech, and Signal Proc.*, vol. 28, pp. 349-472, 1983.

[57] T. Schirtzinger, "Designing adaptive equalizers based upon the constant modulus criterion," M. S. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL 1994.

VITA

Robert Atmaram Soni was born on November 20, 1969. He received the B.S.E.E. degree at the University of Cincinnati in Cincinnati, Ohio, in 1992 and the M.S. degree in electrical engineering from the University of Illinois in 1995. In August 1992, he joined the Graduate College at the University of Illinois at Urbana-Champaign. From August 1992 until May 1996, he was a teaching assistant in the Department of Electrical and Computer Engineering teaching undergraduate students in the Measurements Laboratory, the Electronics Laboratory, and the Senior Design Laboratory. In August 1995, he was appointed a research assistant in the Coordinated Science Laboratory of the University of Illinois at Urbana-Champaign. He has recently accepted a position as a Member of the Technical Staff at Lucent Technologies and Bell Laboratories in Whippany, New Jersey. His research interests include adaptive filtering, digital signal processing for telecommunications, high-speed channel equalization, adaptive interference suppression, antenna arrays, and multidimensional digital signal processing.