INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

INTERIOR POINT LEAST SQUARES ESTIMATION

BY
KAYWAN H. AFKHAMIE
MAY 2000

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF MCMASTER UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

INTERIOR POINT LEAST SQUARES ESTIMATION

Doctor of Philosophy (2000)
(Electrical & Computer Engineering)

McMaster University
Hamilton, Ontario

TITLE:	Interior Point Least Squares Estimation
AUTHOR:	Kaywan H. Afkhamie (B.Eng., M.Eng., McMaster University)
SUPERVISOR:	Dr. Zhi-Quan Luo
NUMBER OF PAGES:	ix, 97

Abstract

Adaptive filters are used in linear estimation problems when no *a priori* knowledge of signal statistics is available or when systems are time varying. They have become a popular signal processing tool and have found application in many diverse areas. The work in recent years has led many to regard the recursive least-squares algorithm (RLS) and its variants as the state of the art. The principal advantage of RLS over the rivalling least mean square algorithm (LMS) is its fast rate of convergence, which in many applications justifies the higher computational complexity of RLS.

Mathematically speaking, both RLS and LMS result from the application of particular iterative optimization algorithms to the minimization of the least-squares criterion. The purpose of this thesis is to investigate the applicability of a new class of optimization methods, interior point optimization algorithms, to the adaptive filtering problem. We develop a new recursive algorithm, called Interior Point Least Squares or IPLS, that can be efficiently implemented with a computational cost comparable to (but higher than) RLS. IPLS matches RLS in asymptotic performance, but has a faster transient convergence rate. This is significant because until now "... [RLS'] convergence speed [has been] considered to be optimal in practice, and [thus] a measure for comparison for other algorithms.", (Moustakides, in a paper in the *IEEE Transactions on Signal Processing*, October 1997). Additional properties of IPLS are its insensitivity to variations in initialization, numerical stability in the presence of limited precision arithmetic, and versatility in that it readily allows the inclusion of additional constraints on the weight vector. Some of the above mentioned properties are further investigated and exploited in applications to adaptive equalization and adaptive beamforming. Numerical simulations are used throughout the thesis to illustrate, confirm, and extend our analytical results.

Acknowledgements

I would like to thank my supervisor, Dr. Tom Luo, for his guidance and continued encouragement. His depth in interior point optimization theory provided the necessary impetus for the convergence analysis in the thesis.

I would also like to thank Dr. Tim Davidson, who was always generous with his time and advice; Dr. Max Wong, for critically reading many of my draft papers; and fellow students Dr. Nima Ahmadvand, Faisal Shad, and Kamran Rahbar with whom I also had many helpful discussions.

Many friends have helped make this time special. Thank you, Darryl, Hugh, Alpesh, Gaurav, Jay, Jean-René, Eliana, and Anne-Jeanne (who is now wreaking havoc in her native country); and from across campus Brian, Paul, Halit and Spiro, and the rest of the Ambasadurs.

Finally, I am most grateful to my family for their unfailing support, and also for putting a roof over my head during this time.

The research reported in this thesis has been made possible by the financial support from the Ontario Graduate Scholarship fund, the ECE Department of McMaster University, and CITO (Communications and Information Technology Ontario).

Contents

Abstract iii

Acknowledgements iv

1 Introduction 1
- 1.1 The Linear Filtering Problem . . . 2
- 1.2 Adaptive Filtering . . . 4
  - 1.2.1 The Least-Mean Square (LMS) Algorithm . . . 6
  - 1.2.2 The Recursive Least-Squares (RLS) Algorithm . . . 7
  - 1.2.3 The Kalman Filter . . . 9
  - 1.2.4 Set-Theoretic Approaches to Adaptive Filtering . . . 9
- 1.3 Adaptive Filtering Applications . . . 10
- 1.4 Interior Point Methods . . . 11
- 1.5 Contributions and Outline . . . 12

2 A First Approach: Interior Point Column Generation 14
- 2.1 Introduction . . . 14
- 2.2 Background on Interior Point Column Generation . . . 16
  - 2.2.1 The Analytic Center . . . 18
  - 2.2.2 Algorithm Details . . . 20
- 2.3 Adaptive Filtering Applications . . . 22
  - 2.3.1 IPCG Algorithms for Adaptive Filtering . . . 22
  - 2.3.2 Practical Modifications to the IPCG Algorithm . . . 23
  - 2.3.3 Simulations . . . 25
- 2.4 Discussion . . . 29

3 Adaptive Linear Filtering using Interior Point Optimization Techniques 31
- 3.1 Introduction . . . 31
- 3.2 An Analytic Center based Estimator . . . 33

3.3 Interior Point Least Squares Algorithm 36
3.3.1 Recursive Update of Newton Direction 38
3.4 Simulations 41
3.5 Discussion 46

4 Convergence Analysis of the Interior Point Least Squares Algorithm 48
4.1 Asymptotic Convergence Analysis of IPLS 49
4.2 Transient Convergence Analysis of IPLS 58
4.3 Discussion 67

5 MMSE Decision Feedback Equalization with short Training Sequences 68
5.1 Introduction 68
5.2 System Description 70
5.3 Simulation Results 72
5.4 Discussion 76

6 Numerically Stable Constrained Adaptive Estimation using IPLS 79
6.1 Introduction 79
6.2 Interior Point Least Squares for Constrained Adaptive Filtering 82
6.3 Simulation 84
6.4 Discussion 87

7 Summary and Discussion 88
7.1 Further Work 89

Bibliography 91

List of Figures

2.1 Five iterations of the IPCG Algorithm for Example 1.1 18
2.2 Example 1.2: (a) The region Ω; (b) Logarithmic barrier function 19
2.3 Noise causing a bad cut 24
2.4 Channel impulse response $h(n)$ 26
2.5 Experiment 2.1: (a) Weight vector error (high SNR) (b) Instantaneous squared error (c) Weight vector error (low SNR) (d) Weight vector error (high SNR, correlated inputs) 28
2.6 Experiment 2.2: (a) Weight vector error, (b) Adaptive thresholding 29

3.1 Example 3.1: Six iterations of analytic center based adaptive filtering. . . . 36
3.2 Comparison of convergence properties of RLS and IPLS (a) SNR = 40dB, Source: White Gaussian Noise (b) SNR = 40dB, Source: Correlated (c) SNR = 10dB, Source: White Gaussian Noise (d) SNR = 10dB, Source: Correlated. 43
3.3 Dependence of IPLS on β. (a) SNR = 40dB, Source: White Gaussian Noise (b) SNR = 10dB, Source: White Gaussian Noise 45
3.4 Dependence of IPLS on R. (a) SNR = 40dB, Source: White Gaussian Noise (b) SNR = 10dB, Source: White Gaussian Noise 45
3.5 Comparison of sliding-window versions of IPLS and RLS when channel characteristics change abruptly (at iteration 100). 46

5.1 Discrete-time model of CDMA downlink 70
5.2 Three ray Rayleigh Fading Channel with fading rate 0.005 71
5.3 Code Matched Filter - Chip rate DFE 71
5.4 Equalization performance in the single user, static channel case. (a) BER performance, (b) Probability of packet arrival. 74
5.5 Equalization performance in the 4 user, static channel case. (a) BER performance, (b) Probability of packet arrival. 75
5.6 Equalization performance versus length of training sequence: static channel. (a) BER performance, (b) Probability of packet arrival. 76

5.7 Mean-squared error versus time, $N_T = 2$, SNR=15dB. 77
5.8 Equalization performance versus length of training sequence: time-varying channel. (a) BER performance, (b) Probability of packet arrival. 77
5.9 Equalization performance in the single user, time-varying channel case. (a) BER performance, (b) Probability of packet arrival. 78
5.10 Equalization performance in the 4 user time-varying channel case. (a) BER performance, (b) Probability of packet arrival. 78

6.1 Mean-squared error in $\mathbf{w}_n$ 85
6.2 Frequency response of filters at Iteration 4000 85
6.3 Frequency response of $\mathbf{w}_n$ for LC-IPLS at iteration 100,000 ($\lambda = 0.9$) 86
6.4 Abrupt change at iteration 1000 86

List of Tables

2.1 IPCG parameter settings 26

3.1 Recursive Computation of Newton Direction $[\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})$. 41
3.2 Nominal parameter settings 42
3.3 Direct Comparison of RLS, IPLS 47

Chapter 1

Introduction

The application of the least-squares criterion to linear estimation has been so successful and ubiquitous that the terms *linear estimation*, *least-squares estimation* and *linear least-squares estimation* have become almost synonymous (in engineering we can also add the term *linear filtering*). The popularity of the least-squares criterion for linear estimation can be attributed to several factors. First, it leads to tractable mathematics and in particular, an objective function that is convex in the parameters (Haykin, 1998). Second, it has an intricate connection with many stochastic or deterministic problems in engineering and even in pure mathematics (Kailath, 1974). Finally, the topic of least-squares estimation has received the attention of several eminent researchers over the last 200 or so years. When Gauss (1809) (who is credited with originating linear estimation (Haykin, 1998)) studied the *motion of heavenly bodies* he invented the *method of least squares*. Kolmogorov (1941), Krein (1945a,b) and perhaps most notably Wiener (1949) used least-squares estimation in their studies of stochastic processes. Wiener's work is said to "have [brought] the statistical point of view into communication theory ... " (Kailath, 1974), which in turn may well have motivated Shannon (1948) to embark on his exploration of information theory. The least-squares cost function is thus established as an excellent choice of optimization criterion (over, for example, the 1-norm or the infinity norm of the error) for linear estimation. In this thesis we too concentrate on the least-squares approach.

We further limit the scope of our work by turning our attention to recursive solutions of the linear filtering problem. While Kolmogorov and Wiener assumed an infinite amount of data and stationary processes, engineering applications in the 1950's demanded methods that were able to update parameter estimates efficiently with an increasing number of observations. This led to three famous approaches that were all developed between 1950

and 1960. The *least mean square* (LMS) algorithm is a stochastic gradient algorithm for adaptively adjusting the tap weights of a transversal filter (Widrow and Hoff, 1960). The *recursive least-squares* (RLS) algorithm can be viewed as a Newton approach to minimizing the least-squares objective. This method has been developed independently by several investigators but credit for the original contribution is sometimes given to Plackett (1950). The third approach is the *Kalman Filter*, which takes advantage of a known finite-dimensional state-space model to estimate the internal state of a system. More details on each algorithm and on the relation that exists between them will be given later in this chapter.

Note that LMS, RLS, and special instances of the Kalman filtering algorithms are also referred to as adaptive filtering algorithms in signal processing. Because adaptive filtering algorithms use only the measurements and no *a priori* information of the underlying process statistics, they can be considered to be stochastic approximations to the optimum linear filtering (*i.e.*, Wiener) solutions. The differences in the methods result from the differences in the various iterative deterministic optimization methods that are used to minimize the least squares cost function. For example, the LMS algorithm is a gradient descent algorithm and RLS arises from an application of the Newton-Raphson method.

Fifteen years ago the seminal work of Karmarkar (1984) has propelled the study of a new class of optimization algorithms, Interior Point Methods. The subject of this thesis is the application of interior point optimization methods to adaptive filtering problems. Our investigation culminates in an algorithm called Interior Point Least Squares. The algorithm's asymptotic and transient convergence properties are analyzed and it is applied to two typical adaptive filtering applications.

The remainder of the chapter is organized as follows. In Section 1.1 we give the mathematical problem formulation of the linear filtering problem. Various solution methods that are available in the literature are discussed in Section 1.2, and some applications of adaptive filters are given in Section 1.3. A brief introduction to Interior Point methods and their literature is given in Section 1.4. We conclude the chapter with a preview of our contributions and an outline of the remainder of the thesis.

1.1 The Linear Filtering Problem

Let us now give the mathematical formulation of the linear filtering problem. While the problem applies to a wide class of applications it is set here in the context of a system identification application. The formulation can be easily modified to apply to, say the adaptive equalization problem (see Chapter 5).

We briefly establish some notational conventions. In the sequel, $\mathbb{R}$ and $\mathbb{C}$ represent the set of real and complex numbers, respectively. The superscripts $(\cdot)^T$ and $(\cdot)^H$ mean transposition and Hermitian transposition. We denote vectors with lower case bold and matrices with upper case bold symbols. The gradient of a function f is ∇f and $\nabla^2 f$ denotes its Hessian matrix. Time indexing is achieved by writing the time as a subscript, as in $\mathbf{w}_n$, or if the subscript is not available, by writing $\mathbf{w}(n)$. The notation $f(n) = O(g(n))$ means that there exist constants c and n_0, such that $f(n) \leq cg(n)$, for all $n \geq n_0$. Finally, statistical expectation is denoted by $E[\cdot]$. Consider a discrete-time linear system as follows:

$$y_i = \mathbf{x}_i^H \mathbf{w}_* + v_i, \qquad i = 1, 2, \ldots$$

where $y_i \in \mathbb{C}$ is the sequence of observations, $\mathbf{x}_i \in \mathbb{C}^M$ denotes the sequence of input vectors of size M, $\mathbf{w}_* \in \mathbb{C}^M$ is the unknown deterministic parameter vector or filter that we wish to estimate, and $v_i \in \mathbb{C}$ is the additive measurement noise. The linear filtering problem is to identify the parameter vector $\mathbf{w}_*$ from input/output pairs $\{\mathbf{x}_i, y_i\}$ and their respective statistics. When the estimate is to be updated sequentially with the arrival of new data, the problem is often referred to as adaptive filtering (Haykin, 1998).

Estimates of $\mathbf{w}_*$ are typically found by either minimizing a least-squares objective function or by minimizing the worst case error. For reasons explained earlier in this chapter we shall concentrate on the least-squares approach. Define the mean-squared error of a particular estimate $\mathbf{w}$ (at time n) as

$$\mathcal{V}_n(\mathbf{w}) = E\left[\left|y_n - \mathbf{x}_n^H \mathbf{w}\right|^2\right] = E[|y_n|^2] - \mathbf{w}^H \mathbf{p} - \mathbf{p}^H \mathbf{w} + \mathbf{w}^H \mathbf{R} \mathbf{w} \tag{1.1.1}$$

where $\mathbf{R} = E\left[\mathbf{x}_n \mathbf{x}_n^H\right]$, and $\mathbf{p} = E\left[\mathbf{x}_n y_n\right]$ are the input signal autocorrelation matrix and the cross-correlation vector of the observation with the input signal, respectively. The least-squares estimate of the parameter vector $\mathbf{w}_*$ is the estimate that minimizes $\mathcal{V}_n$. We may differentiate $\mathcal{V}_n(\mathbf{w})$ with respect to the complex valued tap-weight vector $\mathbf{w}$ using the convention described in (Haykin, 1998, Appendix B). Setting $\nabla \mathcal{V}_n(\mathbf{w}) = 0$ yields the well-known *normal equations*

$$\mathbf{R}\mathbf{w} = \mathbf{p},$$

which are solved by the Wiener solution (the optimum linear filter)

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}. \tag{1.1.2}$$

Thus, if the joint statistics of the (stationary) input and the desired output are known then (1.1.2) represents an effective solution of the filtering problem. The next section deals with the case where these statistics are not available *a priori*.

1.2 Adaptive Filtering

Suppose the desired statistical parameters for the calculation of the Wiener solution are not available. To obtain an alternative solution we may define the sample based mean-squared error function

$$\mathcal{F}_{n,\beta}(\mathbf{w}) = \sum_{i=1}^{n} \beta_{n,i} \left| y_i - \mathbf{x}_i^H \mathbf{w} \right|^2, \tag{1.2.3}$$

where the *weighting factors* $\beta_{n,i}$ are used to adapt the cost function to either a stationary or a non-stationary environment. In the stationary case, $\beta_{n,i}$ should weight each data sample equally. With $\beta_{n,i} = 1/n$, the cost function becomes

$$\mathcal{F}_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - \mathbf{x}_i^H \mathbf{w} \right|^2 = \frac{1}{n} \mathbf{y}_n^H \mathbf{y}_n - \mathbf{w}^H \mathbf{p}_{xy}(n) - \mathbf{p}_{xy}^H(n)\mathbf{w} + \mathbf{w}^H \mathbf{R}_{xx}(n)\mathbf{w} \tag{1.2.4}$$

where

$$\mathbf{y}_n = [y_1, y_2, \cdots, y_n]^T, \quad \mathbf{p}_{xy}(n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i y_i, \quad \mathbf{R}_{xx}(n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^H.$$

Here, $\mathbf{R}_{xx}(n)$ and $\mathbf{p}_{xy}(n)$ are estimates of the parameters $\mathbf{R}$ and $\mathbf{p}$, respectively. The linear filtering problem associated with $\mathcal{F}_n$ is

$$\min_{\mathbf{w} \in \mathbf{C}^M} \mathcal{F}_n(\mathbf{w}), \tag{1.2.5}$$

and its solution, $\mathbf{w} = \mathbf{R}_{xx}(n)^{-1}\mathbf{p}_{xy}(n)$ (obtained by solving $\nabla \mathcal{F}_n(\mathbf{w}) = 0$) is a *stochastic approximation* of the Wiener filter. Note that we are concerned predominantly with the linear filtering problem (1.2.5) (as opposed to the minimization of $\mathcal{V}_n$). Thus, in the sequel, we often refer to the quantities $\mathcal{F}_n, \mathbf{R}_{xx}(n), \mathbf{p}_{xy}(n)$ as the mean-squared error, the autocorrelation matrix, and the cross-correlation vector, respectively.

When operating in a non-stationary environment it is desirable to emphasize the more recent data and to gradually "forget" older data. In this case, $\beta_{n,i}$ is commonly used as an exponential forgetting factor $\beta_{n,i} = \lambda^{n-i}$. The cost function can then be expressed as

$$\mathcal{F}_{n,\lambda}(\mathbf{w}) = \sum_{i=1}^{n} \lambda^{n-i} y_i^* y_i - \mathbf{w}^H \mathbf{p}_{xy}(n) - \mathbf{p}_{xy}^H(n)\mathbf{w} + \mathbf{w}^H \mathbf{R}_{xx}(n)\mathbf{w} \tag{1.2.6}$$

where in this context

$$\mathbf{p}_{xy}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}_i y_i, \quad \mathbf{R}_{xx}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}_i \mathbf{x}_i^H.$$

Note that here the parameters $\mathbf{p}_{xy}(n)$ and $\mathbf{R}_{xx}(n)$ are defined differently as in the case of constant weighting. In the sequel it is always clear from the context which definition applies (depends on whether or not there is a forgetting factor).

To further emphasize the non-stationarity of the data, one may entirely disregard all data that is older than a given limit I , by setting

$$\beta_{n,i} = \begin{cases} \lambda^{n-i}, & i = n-I+1, \dots, n \\ 0, & i \leq n-I \end{cases}$$

This definition leads to obvious modifications in $\mathcal{F}_{n,\lambda}$, $\mathbf{p}_{xy}(n)$ and $\mathbf{R}_{xx}(n)$.

When we speak of *adaptive* filters we typically have the additional requirement of being able to update a solution efficiently with the arrival of more data. This calls for recursive implementations of stochastic methods. The most prominent recursive schemes for linear filtering are the LMS, RLS and Kalman filtering algorithms and their variants. Before introducing these and the less conventional *set-theoretic* methods for adaptive filtering, let us briefly list some criteria by which the performance of such methods is measured (Haykin, 1998).

Rate of convergence This indicates the speed at which an algorithm is able to converge to the Wiener solution in the mean-square sense. We also distinguish between *asymptotic* convergence, which describes the behaviour of an algorithm after having seen a "large" amount of data, and *transient* convergence, which refers to the convergence behaviour shortly after initialization.

Tracking While the rate of convergence is typically measured in a stationary environment, tracking performance is more important in a non-stationary environment, where an algorithm must follow or "track" the statistical variations of signals.

Misadjustment This measures the statistical bias of an estimator with respect to the Wiener solution.

Robustness Robustness can be measured with respect to many factors. In the literature the robustness of adaptive filtering algorithms often refers to the sensitivity of the algorithm to ill-conditioned input data, or its robustness to round-off errors (*i.e.*, numerical sensitivity). In the sequel we will also refer to the robustness of algorithms to their initialization. Specifically, the initialization of an algorithm should be easy to determine, and the algorithm should not be sensitive to small variations in its initialization.

Computational Complexity Computational complexity is typically measured by the amount of computation required per iteration and by the memory requirements of the algorithm.

Structure This criterion refers to the ease with which an algorithm's structure can be exploited for efficient implementation. For example, some algorithms lend themselves easily to parallel implementations saving significant computation time.

Numerical properties Digital implementations of an algorithm cause round-off errors due to the limited precision representation of numbers in a computer. Such round-off errors can in turn lead to issues of *numerical stability* in the recursive adaptive filtering algorithms. Another issue is *numerical accuracy* which refers to the degree to which an algorithm's precision is affected by the number of bits used to represent each quantity.

The literature on adaptive filtering is too extensive to be adequately surveyed in this section. In the following we give a brief outline of the major approaches; much more detailed presentations are given in the books by Haykin (1998), Kalouptidis and Theodoridis (1993), Proakis *et al.* (1992), Widrow and Stearns (1985). Also of interest in this context is a recent survey paper by Glentis *et al.* (1999).

1.2.1 The Least-Mean Square (LMS) Algorithm

The LMS algorithm is a form of steepest (or gradient) descent approach to minimizing the mean-squared error cost function. It was introduced by Widrow and Hoff (1960). The algorithm produces an estimator of the form

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \frac{1}{2}\mu_n \mathbf{d}_n$$

where μ_n is a step-size parameter and $\mathbf{d}_n$ is the direction in which the tap weight vector is updated. As implied by the title "gradient descent", $\mathbf{d}_n$ is determined by the negative of the instantaneous gradient of $\mathcal{F}_n$,

$$\begin{aligned} \mathbf{d}_n &= -\frac{\partial}{\partial \mathbf{w}} \left\{ \left[y_n - \mathbf{x}_n^H \mathbf{w}_{n-1} \right]^2 \right\} \\ &= 2\mathbf{x}_n \left(y_n - \mathbf{x}_n^H \mathbf{w}_{n-1} \right). \end{aligned}$$

Noting that $y_n - \mathbf{x}_n^H \mathbf{w}_{n-1}$ can be regarded as a forward prediction error e_n, we have the LMS update equation

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu_n \mathbf{x}_n e_n. \tag{1.2.7}$$

Equation (1.2.7) and the definition of the error e_n define the LMS algorithm. The method is highly popular to this date because of its simplicity and efficiency in computation. There are however several drawbacks. Most importantly, the method is slow to converge (*e.g.*, compared to the RLS algorithm) and sensitive to the eigenvalue spread in the signal autocorrelation matrix.

Over the years there have been many variants of the LMS algorithm. Better convergence rates are achieved with *Normalized* LMS, which utilizes a time-varying step-size parameter $\mu_n = 1/(\mathbf{x}_n^H \mathbf{x}_n)$. Some computational savings are attained by the so-called *sign*-LMS algorithm which quantizes the error to the three levels -1, 0, 1. The interested reader may consult Glentis *et al.* (1999) for a recent review and comparison of LMS algorithms. Finally, a surprising property of the LMS algorithm is its optimality in an H^∞ sense. This has been recently shown by Hassibi *et al.* (1996).

1.2.2 The Recursive Least-Squares (RLS) Algorithm

The recursive least-squares algorithm is another very well-known method for adaptive filtering. The reason for its popularity lies mainly in its improved convergence rate as compared to LMS, even though the better transient performance is achieved at a computational cost. In our opinion the RLS algorithm is situated quite centrally in the "forest" of adaptive filtering algorithms. It can be viewed as an extension of the LMS approach (via the Fast Newton Transversal Filters described below) and as a special case of the Kalman filter (Sayed and Kailath, 1994) (also see our discussion in the next section). In Chapter 3 of this thesis we will realize that the RLS algorithm is also quite similar to the interior point least squares algorithm.

The (forgetting factor) RLS algorithm can be expressed with the set of equations (Moustakides, 1997)

$$\mathbf{R}_{xx}(n) = \lambda \mathbf{R}_{xx}(n-1) + \mathbf{x}_n \mathbf{x}_n^H \tag{1.2.8}$$

$$e_n = y_n - \mathbf{x}_n^H \mathbf{w}_{n-1} \tag{1.2.9}$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} + e_n \mathbf{R}_{xx}^{-1}(n) \mathbf{x}_n \tag{1.2.10}$$

where λ is a forgetting factor ($0 \ll \lambda \leq 1$) and all other quantities are defined as before. Note that (1.2.10) would require the inversion of $\mathbf{R}_{xx}(n)$ which is an $O(M^3)$ operation (where M, once again, is the number of adjustable tap weights). Implementations of the RLS algorithm actually propagate the inverse factor $\mathbf{R}_{xx}^{-1}(n)$ to avoid the direct matrix inversion. The update of the inverse autocorrelation matrix can be achieved in $O(M^2)$ by a simple application of the Sherman-Morrison formula, also known as the matrix inversion lemma

(Golub and Van Loan, 1996). For the purposes of this review the more lucid description (1.2.8) — (1.2.10) of RLS is preferred.

RLS suffers from instability in the presence of finite-precision arithmetic. To address this issue, square-root versions of RLS have been developed that propagate a square-root factor of $\mathbf{R}_{xx}^{-1}(n)$. They achieve numerical stability essentially by preventing the loss of symmetry and positive-definiteness of matrix $\mathbf{R}_{xx}^{-1}(n)$ (Glentis *et al.*, 1999). Another matrix factorization leads to the well-known QR-decomposition based RLS algorithm (see, *e.g.*, Haykin, 1998). This algorithm too is more numerically stable than standard RLS.

Other modifications of RLS arise from approximations to the inverse autocorrelation matrix $\mathbf{R}_{xx}^{-1}(n)$. Since the RLS filter update equation (1.2.10) can be viewed as a Newton step in the minimization of the mean-squared error, these methods can be collectively regarded as *Quasi-Newton approaches* (Glentis *et al.*, 1999). The first Quasi-Newton approach replaces $\mathbf{R}_{xx}(n)$ with a Toeplitz approximation (Panda *et al.*, 1986; Marshal and Jenkins, 1992; Diniz *et al.*, 1995; de Campos and Antoniou, 1997). Other methods assume an AR model of the input signal $x(n)$. When the order of the AR process is much smaller than the order of the filter, the use of the resulting banded autocorrelation matrix leads to significant computational savings. These so-called *Fast Newton Transversal Filters* (FNTF) (Moustakides and Theodoridis, 1991; Theodoridis *et al.*, 1995; Mavridis and Moustakides, 1996; Farhang-Boroujeny, 1997) reduce to LMS when the AR process is of order zero and the autocorrelation matrix becomes diagonal. The third family of Quasi-Newton algorithms computes the pseudo-inverse of the limited-time sample covariance matrix, *i.e.*, only the last L samples are used to compute $\mathbf{R}_{xx}(n)$. These methods can be interpreted to compute the filter under the constraint that it optimally projects each of the L inputs onto its corresponding output value, and are thus often referred to as *affine projection algorithms* (Tanaka *et al.*, 1999; Rupp, 1998; Baykal and Constantinides, 1997).

So far, most of the RLS-type algorithms we discussed can be computed with $O(M^2)$ operations per iteration. They are thus more expensive than the $O(M)$ LMS algorithms and much effort has been expended into overcoming this drawback. Perhaps the most well known method in this direction is the so-called *Fast-RLS* algorithm (Ljung *et al.*, 1978; Carayannis *et al.*, 1982; Cioffi and Kailath, 1984). Fast-RLS exploits the shift-invariance property of the input data matrix (*i.e.*, at each iteration n only one element, $x(n)$, in the size M data vector $\mathbf{x}_n$ is new) to derive an $O(M)$ update algorithm. Although the method is theoretically equivalent to the RLS algorithm it exhibits much poorer numerical performance. Numerical instability problems were addressed in Botto and Moustakides (1989) and Slock and Kailath (1991). Other approaches to reduce the computational complexity include the FNTF already mentioned above and the so-called *Fast Quasi-Newton* approaches (Marshal

and Jenkins, 1992) which achieve computational savings by updating the approximation to $\mathbf{R}_{xx}^{-1}(n)$ only infrequently.

1.2.3 The Kalman Filter

Kalman filter theory was developed in the late 1950's (Kalman, 1960) to accommodate nonstationary input signals (which did not fit into the Wiener filtering framework). Kalman considered stochastic processes that admit a state-space representation as follows,

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{F}_n\mathbf{x}_n + \mathbf{G}_n\mathbf{r}_n \\ \mathbf{y}_n &= \mathbf{H}_n\mathbf{x}_n + \mathbf{v}_n \end{aligned}$$

where $\mathbf{y}_n$ is an N-dimensional observation vector, $\mathbf{x}_n$ is the M-dimensional state vector, and the matrices $\mathbf{F}_n, \mathbf{G}_n, \mathbf{H}_n$ are known and of appropriate dimensions. The vectors $\mathbf{r}_n$ and $\mathbf{v}_n$ are uncorrelated zero-mean noise processes with known covariance matrices. The Kalman filtering algorithm provides an efficient way to estimate the state $\mathbf{x}_{n+1}$ recursively.

Kalman filter theory was first applied to adaptive filtering in the early 1970's. While the problem was perhaps first considered by Lawrence and Kaufman (1971) credit often goes to Godard (1974). Godard rephrased the adaptive filtering problem in the state-space framework by regarding the unknown weight vector as the unknown state variable $\mathbf{x}_n$. Since the work of Godard it was clear that there existed a relationship between the Kalman filter and RLS, at least for the growing memory case (forgetting factor $\lambda = 1$). But it was only 20 years later that Sayed and Kailath (1994) were able to extend the connection of RLS and Kalman filtering to the case of exponentially decaying memory. In fact, Sayed and Kailath showed that virtually all existing RLS-type algorithms can be derived from their state-space counterparts.

1.2.4 Set-Theoretic Approaches to Adaptive Filtering

An alternative approach to adaptive filtering is given by set-theoretic methods that are based on the premise that noise can be assumed to be bounded,

$$|v_i| \leq \gamma, \qquad \text{for some } \gamma > 0.$$

The set of filters consistent with all observations up to a certain time n is called a *membership set* and takes the general form

$$\Omega_n = \bigcap_{i=1}^{n} \left\{ \mathbf{w} \in \mathbb{R}^M \;:\; -\gamma \leq y_i - \mathbf{x}_i^T\mathbf{w} \leq \gamma \right\}.$$

It is the goal of the adaptive algorithms to recursively update a "center" $\mathbf{w}_n$ of Ω_n. In general there are two approaches.

Recognizing that Ω_n describes an increasingly complex set (the number of vertices increase exponentially with n) most methods obtain an approximate outer bounding of the membership set by a simple-shaped set. Most often this simpler set is an ellipsoid (Dasgupta and Huang, 1987; Fogel and Huang, 1982; Deller, 1989) but it can also be an orthotope (Belforte *et al.*, 1990) or a parallelotope (Vicino and Zappa, 1996). Regardless of their shape, these outer bounding sets introduce a certain level of conservatism, because they have to be made larger than the original Ω_n. The advantage that is gained by the outer bounding is that now a center of the set can be computed with relative ease.

Other approaches work with Ω_n directly. As indicated above, the complexity of this set makes this strategy computationally prohibitive. Nevertheless, one recent approach proposes to compute the *analytic center* of Ω_n and produces an iterative algorithm to update this estimate (Bai *et al.*, 1999). While this algorithm is at least recursive, its computing time still grows as n increases (Bai *et al.*, 1998).

The approach of Bai *et al.* is of particular interest in this thesis as we too use the analytic center of a feasible region as our filter estimate (Chapters 2 and 3). Our approach, however, is fundamentally different from that of Bai *et al.* because (1) we do not impose the bounded noise restriction, and (2) our algorithm of Chapter 3 has a computational complexity comparable to that of more conventional least-squares approaches (like RLS).

1.3 Adaptive Filtering Applications

Linear filtering problems that require adaptive solutions arise in various fields such as control, communications, coding, digital image processing, radar, and geophysical exploration. Specific applications include system identification, adaptive equalization, linear predictive coding, echo cancellation, and beamforming. These are just a few examples. Details on these and many other problems can be found for example in Haykin (1998), or in Kalouptidis and Theodoridis (1993).

In this thesis we consider in detail the adaptive equalization of wireless communication channels in Chapter 5 and an example of adaptive beamforming in Chapter 6. Also, an application to system identification is considered in Chapters 2 and 3 to formulate and briefly motivate the development of our approaches to recursive linear filtering.

1.4 Interior Point Methods

Interior point methods have fundamentally altered the landscape of mathematical programming theory ever since the seminal work by Karmarkar (1984). They presented a polynomial-time approach to linear programming that performed well in practice (the first polynomial-time algorithm for linear programming was given by Khachiyan, 1979). Karmarkar's paper and much of the ensuing research effort (which was extensive) addressed the application of interior point methods to linear programming and quadratic programming. In recent years some work also explored the extension of Karmarkar's first results to general convex programming. Most notably Nesterov and Nemirovskii (1994) presented a *unified theory* of polynomial-time interior point methods for all of convex programming.

Note that in this thesis we do not use any interior point method directly, rather we use some of the concepts and ideas they introduce. However, to convey to the reader a sense of how interior point methods are used in optimization it is useful to describe their operation in a general convex programming setting. Consider the following general convex program (Nesterov and Nemirovskii, 1994)

$$\begin{array}{lll} \text{CP:} & \min & f(x) \\ & \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \ldots, m \end{array}$$

where $f(x)$ is a linear objective function and the constraint functions $g_i(x)$ are convex. The constraining inequalities define a convex feasible set which has a nonempty interior

$$D = \{x \mid g_i(x) < 0, \quad i = 1, \ldots, m\}.$$

A "barrier function on D" is a function that tends to infinity as x approaches the boundary of D. If $B(x)$ is such a barrier function and $\mu \geq 0$, then the "barrier problem" associated with the CP can be expressed as

$$\begin{array}{lll} \text{BP}(\mu): & \min & f(x) + \mu B(x) \\ & \text{s.t.} & x \in D \end{array}$$

If $B(x)$ is strictly convex and $\mu > 0$, then BP(μ) has a unique solution $x^*(\mu)$. Variations in the parameter μ lead to the *central trajectory*, $\{x^*(\mu) : \mu \in [0, 1]\}$ of CP. It can be proved that as $\mu \to 0$ the central trajectory converges to a solution of the original CP, which usually lies on the boundary of D. The strategy for solving the CP above is efficient (polynomial-time) only if the barrier function $B(x)$ is chosen appropriately. An important contribution of Nesterov and Nemirovskii (1994) is that they have shown that the Newton method is a particularly efficient tool for solving a CP, when $f(x)$ and $B(x)$ are *self-concordant* functions.

In general, a function $f(x)$ is self-concordant if for some K

$$|f'''(x)| \leq 2K \left[f''(x)\right]^{3/2}$$

for all x in the domain of f. A popular barrier function for the CP above is $B(x) = -\sum_i^m \log g_i(x)$, which is appropriately called the *logarithmic barrier function*. It can now also be seen where interior point methods have obtained their name. The final solution is obtained by following the central path from the *interior* of D; this is in contrast to, for example, the simplex method for linear programming, which finds the solution by checking *boundary* points of the feasible set.

Interior point methods have been applied to mathematical programming problems (*e.g.*, linear and quadratic programming, geometrical programming, quadratically constrained quadratic programming) and to engineering problems. Vandenberghe and Boyd (1996) show how interior point methods can be efficiently applied to semidefinite programming problems that arise in control and information theory, maximum eigenvalue and matrix norm minimization, pattern classification, statistics, and combinatorial and non-convex optimization. More recently, Davidson *et al.* (1999) used semidefinite programming formulations for the optimization of orthogonal pulse shapes in communications.

To close this section we give some further references on interior point methods. An excellent starting point is the tutorial paper by Freund and Mizuno (1996). This paper references many recent books on the subject, for example, Ye (1997), Bertsimas and Tsitsiklis (1997), Wright (1997), Terlaky (1996), and Roos *et al.* (1997).

1.5 Contributions and Outline

In applying interior point optimization techniques to adaptive least-squares estimation we make a twofold contribution. From the signal processing point of view, we provide new recursive algorithms for linear least-squares filtering. For the interior point least squares (IPLS) algorithm of Chapter 3 we also present the full asymptotic and transient convergence analysis. The IPLS algorithm, in particular, also has several desirable properties which distinguish it from established methods such as RLS. It has an exponential transient convergence rate, and is robust to small variations in its initialization. It easily accommodates linear or convex quadratic constraints and remains numerically stable in the presence of such constraints. The computational complexity of this algorithm is higher than that of RLS, however, for the increase in computation one receives several major advantages as just described. From the point of view of interior point optimization, we have shown

the applicability and usefulness of interior point methods to dynamic problems in engineering. The *applicability* is shown by using interior point concepts in the development of our algorithms and the *usefulness* is shown by using the interior point theory, to prove the better performance of these algorithms as compared to methods based on other optimization techniques.

The remainder of this thesis is organized as follows. Our first steps in using interior point optimization techniques for adaptive filtering are documented in Chapter 2. We introduce an adaptive filtering approach that is based on the interior point column generation technique. This method shows promising results already, but in trying to avoid the restriction to a bounded noise case, it uses heuristic measures to achieve good performance, and thus its convergence properties become difficult to analyze. Chapter 2 is also used to introduce in more detail many of the concepts we need from the interior point literature. A more "mature" algorithm for adaptive filtering is developed in Chapter 3. This interior point least squares (IPLS) algorithm allows a detailed comparison to the recursive least-squares technique and a complete convergence analysis which follows in Chapter 4. The most important result shown here, is the exponential convergence rate of IPLS in the transient phase. The transient performance of IPLS is then exploited in Chapter 5 to show that it requires shorter training sequences than, for example, RLS to adjust the tap weights of an adaptive equalizer. Chapter 6 demonstrates another feature of the IPLS algorithm. It is shown in an adaptive beamforming context that IPLS can easily handle additional convex constraints, and that it remains numerically stable in the presence of such constraints. Chapter 7 contains a summary and some directions for future research.

Chapter 2

A First Approach: Interior Point Column Generation

In this chapter we present our first approach to adaptive filtering using interior point optimization techniques. The method is built on the concept of Interior Point Column Generation (IPCG). One of the key features of these algorithms is that they consider inequality constraints that define a convex problem one at a time. This can be exploited in applications that require a solution to be updated adaptively. We give some background on IPCG algorithms and how they are used to solve convex feasibility problems. Then we apply IPCG to the classical filtering problem of adaptive system identification. Simulation results here show the potential advantages IPCG has over the recursive least squares method, especially in terms of transient convergence behaviour.

2.1 Introduction

Interior Point Column Generation algorithms have been developed in recent years by the optimization community for the problem of finding a feasible point in a set defined by a (possibly infinite) number of convex inequalities (convex feasibility problems) (Goffin *et al.*, 1994; Luo and Sun, 1999; Luo, 1997). These algorithms are also known as interior point cutting plane methods. Their operation can be described in simple terms as follows. At each iteration they compute an approximate center of a current set defined by the inequalities generated in previous iterations. If this approximate center is not a feasible point (with respect to the original problem), then a new inequality (or cutting plane) is added to the current set, thus defining a new set with a new approximate center. As the number of

cuts increases, the set defined by their intersection shrinks and the algorithm gets closer to finding a point satisfying all the convex feasibility constraints. One important advantage of IPCG algorithms is that they do not require the complete knowledge of all constraints; the inequalities that define the feasible set are tested one at a time, and as soon as an inequality is found to be violated the solution is updated.

IPCG algorithms are well suited to adaptive filtering for several reasons. The error minimization criterion (1.2.5) used in the least squares formulation of the adaptive filtering problem can be easily transformed into a quadratically constrained convex feasibility problem. In particular, we can apply IPCG to find a filter $\mathbf{w}$, such that

$$\mathcal{F}_n(\mathbf{w}) \leq \tau^2, \qquad \forall\ n \geq M, \tag{2.1.1}$$

where the threshold τ^2 can be either a constant related to the average noise power or determined adaptively. Thus, the adaptive filtering problem can be formulated as the problem of finding a feasible point satisfying the convex quadratic inequalities given by (2.1.1). Notice that the inequalities (2.1.1) are generated dynamically as new data arrives in a sequential fashion. This makes it very natural to apply the IPCG algorithms to adaptive filtering.

Interior point algorithms have previously been successfully applied to engineering problems in control and information theory. Boyd *et al.* (1994) and Vandenberghe and Boyd (1996) used interior point methods to solve many optimal control and design problems involving linear matrix inequalities. However, their applications are static in nature in the sense that constraints must be specified before the start of the computation. The application of interior point methods to dynamic problems in engineering commenced in 1996. There were two research efforts: while the developments that are reported in this chapter were first published in (Afkhamie *et al.*, 1996), an independent interior point approach for parameter estimation was also developed by Bai *et al.* (1997). The significant difference between the two approaches is that while Bai *et al.* considered the bounded noise case, our method does not impose any such restrictions on the noise, and may thus be used in a wider variety of applications.

As it turns out, the application of IPCG algorithms to the adaptive filtering problem (1.2.5) is far from straightforward. This is partly because IPCG algorithms are designed for deterministic convex feasibility problems (Goffin *et al.*, 1994; Luo and Sun, 1999; Luo, 1997), while in adaptive filtering the data sequences $\mathbf{x}_n$ and y_n are stochastic and may be noise-corrupted. Certain modifications are thus necessary for a practical implementation of the IPCG algorithms to solve the adaptive filtering problem. For example, we need to introduce a mechanism for removing cuts that are obsolete or incorrect in the sense that

additive noise caused the cut to face a wrong direction (see Figure 2.3 for an illustration). Without this feature, IPCG algorithms may get stuck because noise-corrupted cuts given by (2.1.1) may sometimes mislead the IPCG algorithms to cut away the optimal filter.

In the next section, we briefly introduce the theory of Interior Point Column Generation. In Section 2.3, we formulate the adaptive filtering problem as a convex feasibility problem, and give a detailed description of an IPCG algorithm for solving it. Particular details will be given on the practical modifications added to the original IPCG algorithm, and we also present some encouraging simulation results which show that the convergence behaviour of the IPCG algorithm compares favorably to the RLS algorithm. We conclude the chapter with some discussion in Section 2.4.

2.2 Background on Interior Point Column Generation

Our focus in this section will be mainly on conveying an intuitive understanding of Interior Point Column Generation, as well as to cover all the major mathematical computations performed in the algorithm. An in-depth treatment of the algorithms summarized here can be found in (Goffin *et al.*, 1994; Luo and Sun, 1999; Luo, 1997).

The IPCG algorithm has been developed by the optimization theory community to solve the so called convex feasibility problem. There, the objective is to find a feasible point in a convex set $\Gamma \subset \mathbb{C}^M$, where we assume Γ contains an interior point. The complex region Γ may be bounded by convex inequalities of arbitrary shape. Their number may be finite or infinite and they may be defined implicitly. The convex feasibility problem includes as special cases the linear programming problem, the linear feasibility problem and the convex quadratic programming problem with quadratic constraints (Luo and Sun, 1999).

Let us outline how the IPCG algorithm solves the convex feasibility problem. We start with a sufficiently large "search region" Ω_0, such that $\Gamma \subset \Omega_0$. The algorithm iteratively cuts a portion away from the search region, thereby generating the sequence of shrinking sets:

$$\Omega_0 \supset \Omega_1 \supset \Omega_2 \supset \ldots \supset \Omega_n \supset \Omega_{n+1} \supset \ldots \supset \Gamma.$$

At each iteration n, the "analytic center" (the definition of this center will be given shortly, in Section 2.2.1) of Ω_n, $\mathbf{y}_n$, is tested for feasibility with respect to Γ. If $\mathbf{y}_n \in \Gamma$, then the algorithm stops. Otherwise, we have $\mathbf{y}_n \notin \Gamma$ and a cut is generated to form Ω_{n+1},

$$\Omega_{n+1} = \Omega_n \cap \left\{ \mathbf{y} \in \mathbb{C}^M : c_{n+1} - \langle \mathbf{a}_{n+1}, \mathbf{y} \rangle - \langle \mathbf{y}, \mathbf{a}_{n+1} \rangle - \frac{1}{2} \langle \mathbf{y}, \mathbf{Q}_{n+1} \mathbf{y} \rangle \geq 0, \right\},$$

where c_{n+1} is a scalar, $\mathbf{a}_{n+1}$ is an $M \times 1$ complex vector, and $\mathbf{Q}_{n+1}$ is an $M \times M$ Hermitian positive semi-definite matrix. The notation $\langle \mathbf{a}, \mathbf{b} \rangle$ denotes the inner product $\mathbf{a}^H \mathbf{b}$. Cuts are always placed through Ω_n in a way such that $\Gamma \subset \Omega_{n+1} \subset \Omega_n$. More specifically, at each iteration $n \geq 0$ the algorithm performs the following three steps:

Step 1. Feasibility test and column generation.
First test whether $\mathbf{y}_n$ is a feasible point. If yes, stop. Otherwise, identify an inequality that is violated by $\mathbf{y}_n$ and determine the corresponding $\mathbf{a}_{n+1}, \mathbf{Q}_{n+1}$.

Step 2. Update Ω_n to Ω_{n+1}.
Given $\mathbf{a}_{n+1}, \mathbf{Q}_{n+1}$ of a new inequality, compute a corresponding c_{n+1} that ensures an aggressive cut while leaving the previous analytic center $\mathbf{y}_n$ in the interior of Ω_{n+1} (for faster convergence in Step 3).

Step 3. Compute new analytic center.
Compute a new *approximate analytic center* $\mathbf{y}_{n+1}$ of Ω_{n+1} and go back to Step 1.

Let us further illustrate the principles of Interior Point Column Generation by a simple example in $\mathbb{R}^2$, employing linear cuts (*i.e.*, $\mathbf{Q}_n = \mathbf{0}$).

Example 1.1: IPCG algorithm

Suppose the feasible region Γ is given by a small circle located within the unit square in $\mathbb{R}^2$. The initial conditions for the IPCG algorithm are given as follows:

$$\begin{aligned} \Omega_0 &:= \text{unit square in } \mathbb{R}^2 \\ \mathbf{y}_0 &:= (0.5, 0.5) \end{aligned}$$

The algorithm receives $\mathbf{y}_0$ as an input and tests its feasibility with respect to Γ. If $\mathbf{y}_0$ is feasible, *i.e.*, $\mathbf{y}_0 \in \Gamma$, the algorithm terminates; otherwise it identifies a linear inequality that will cut away a fraction from the current set Ω_0, while making sure nothing is cut away from the feasible region Γ. With this additional cut and the cuts previously existing in Ω_0 we obtain an updated region Ω_1. Then the new center y_1 of this region is computed and the algorithm continues.

Figure 2.1 displays 5 iterations of the IPCG algorithm. The algorithm terminates after 5 iterations because the analytic center y_4 of Ω_4 is a feasible point inside the circle Γ. Note that in general cuts are not necessarily placed directly through the analytic centers. □

The above example illustrates all the important features of the IPCG algorithm. In particular, it should aid the reader in understanding the three steps of the algorithm. Before elaborating on the implementation of each step, we introduce the concept of the analytic center.

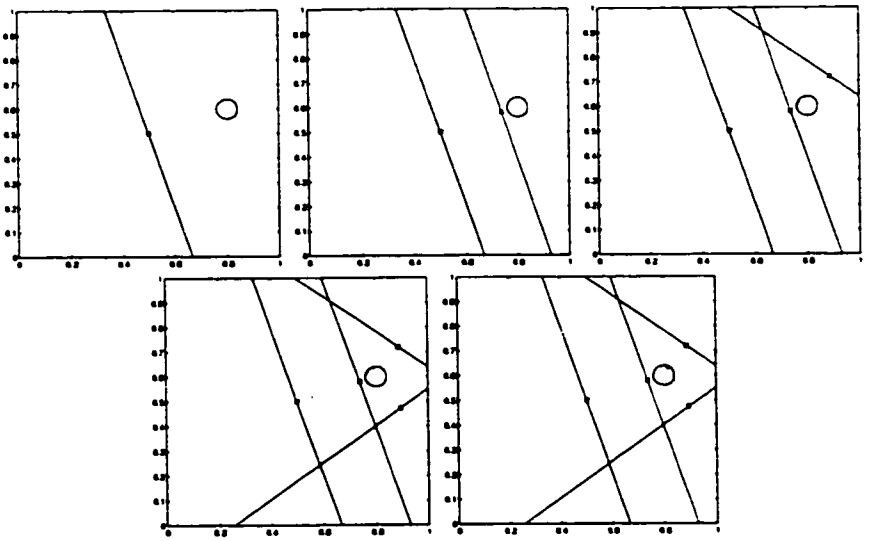


Figure 2.1: *Five iterations of the IPCG Algorithm for Example 1.1*

2.2.1 The Analytic Center

The success of a column generation algorithm depends critically on its choice of a "center" for each Ω_n. Intuitively, we would like $\mathbf{y}_n$ to be located at the "geometric center" of Ω_n (*i.e.*, the center of gravity) because, in this way, we can expect to cut away a sizable portion of the search region Ω_n when a new cut is placed at $\mathbf{y}_n$. As a result, we can ensure the volume of Ω_n is reduced at a geometric rate, thus ensuring a fast convergence of the column generation algorithm. However, the geometric center is usually difficult to compute. Thus, to achieve overall computational efficiency, we need to find a center for Ω_n that is not only located almost centrally in Ω_n, but also easily computable. One such choice is the so called *analytic center* of Ω_n. Unlike the geometric center of Ω_n which depends only on the geometric shape of Ω_n, the analytic center depends further on the algebraic representation of Ω_n.

Formally, the notion of analytic center can be defined as follows. Suppose Ω_n is an arbitrary bounded convex body in $\mathbb{R}^M$

$$\Omega_n = \left\{ \mathbf{y} \in \mathbb{C}^M : c_j - \langle \mathbf{a}_j, \mathbf{y} \rangle - \langle \mathbf{y}, \mathbf{a}_j \rangle - \frac{1}{2} \langle \mathbf{y}, \mathbf{Q}_j \mathbf{y} \rangle \geq 0,\ j = 1, \ldots, N_n \right\}.$$

Let us first define the *potential function* of Ω_n as

$$\phi_n(\mathbf{y}) = -\sum_{j=1}^{N_n} \log s_j, \tag{2.2.1}$$

where the slack variables s_j are measures of the distance from $\mathbf{y}$ to each of the boundaries of Ω_n,

$$s_j = c_j - \langle \mathbf{a}_j, \mathbf{y} \rangle - \langle \mathbf{y}, \mathbf{a}_j \rangle - \frac{1}{2}\langle \mathbf{y}, \mathbf{Q}_j \mathbf{y} \rangle, \quad j = 1, \ldots, N_n.$$

It is intuitive that a point deep in the interior of Ω_n will have a small potential while a point close to the boundary of Ω_n will have at least one very small s_j value, and have a large positive potential. Points on the boundary of Ω_n will have one $s_j = 0$, thus their potential goes to infinity. This explains why ϕ_n is also referred to as a *logarithmic barrier function.* It can be shown that the potential function is strictly convex and has a unique minimizer over Ω_n (Luo, 1997). Since the potential function ϕ_n depends on the analytic representation of Ω_n, we call this minimizer of ϕ_n the analytic center of Ω_n, and write

$$\mathbf{y}^a = \min_{\mathbf{y} \in \Omega_n} \phi_n(\mathbf{y}).$$

We illustrate the analytic center concept with the following simple example in $\mathbb{R}^2$.

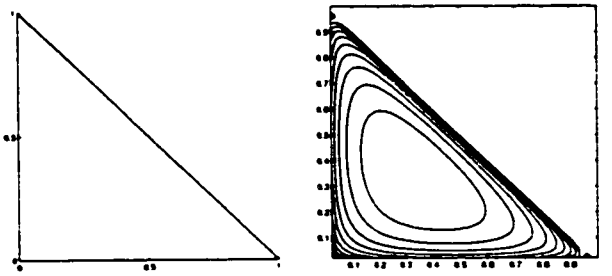
Example 1.2: Potential function and analytic center

Suppose we have

$$\Omega := \left\{ \mathbf{y} \in \mathbb{R}^2 \ : \ 1 - y_1 - y_2 \geq 0, \ y_1 \geq 0, \ y_2 \geq 0 \right\}.$$

Geometrically, Ω is given by a triangle in $\mathbb{R}^2$ as depicted in Figure 2.2a. If we evaluate the potential function (2.2.1) over the interior of Ω, we obtain

$$\phi(\mathbf{y}) = -\log(1 - y_1 - y_2) - \log y_1 - \log y_2.$$



(a) (b)

Figure 2.2: *Example 1.2: (a) The region Ω; (b) Logarithmic barrier function*

The contour plot of the logarithmic barrier function ϕ is given by Figure 2.2b. It can be seen that the potential function is steep at the outer borders and flat in the middle. The analytic center is located at $(1/3, 1/3)^T$. Notice that in this case the geometric center coincides with the analytic center. □

2.2.2 Algorithm Details

Having defined the notion of analytic centers, we can now examine each of the three steps of the IPCG algorithm in more detail. There is a separate module that performs the feasibility test and column generation of step 1. This module takes as an input the current feasible point $\mathbf{y}_n$ and tests it against its knowledge of the feasible region Γ. If the given point $\mathbf{y}_n$ is *not* feasible, the module will generate a convex quadratic inequality that is violated at $\mathbf{y}_n$ but is satisfied by all the points in Γ. Such an inequality is called a *separating inequality* (or *cut*) and can be written as

$$\bar{c}_{n+1} - \langle \mathbf{a}_{n+1}, \mathbf{y} \rangle - \langle \mathbf{y}, \mathbf{a}_{n+1} \rangle - \frac{1}{2}\langle \mathbf{y}, \mathbf{Q}_{n+1}\mathbf{y} \rangle \quad \begin{cases} = 0 & \mathbf{y} = \mathbf{y}_n \\ \geq 0 & \forall \mathbf{y} \in \Gamma. \end{cases} \tag{2.2.2}$$

The values of $\bar{c}_{n+1}$, $\mathbf{a}_{n+1}$ and $\mathbf{Q}_{n+1}$ are passed to step 2 of the IPCG algorithm for further processing. The choice of $\bar{c}_{n+1}$ such that the cut bisects Ω_n through $\mathbf{y}_n$ is the most aggressive cut possible. In many applications, the value of $\bar{c}_{n+1}$ may be unavailable or inappropriate. In this case, a new constant term c_{n+1} will be evaluated in step 2 of the IPCG algorithm. Note also that since Γ is convex and $\mathbf{y}_n \notin \Gamma$, separating inequalities or cuts always exist.

In the optimization literature, the module used in step 1 for feasibility test and cut generation is usually called an *oracle*. With the help of an oracle, step 1 of the IPCG algorithm can be easily carried out. We remark that the exact form of this oracle varies from application to application; see (2.3.3)–(2.3.4) for a description of the oracle in the case of adaptive filtering. Indeed, the oracle is the only place within the IPCG algorithm that requires (and possesses) knowledge of the feasible region Γ. In other words, to construct an IPCG algorithm for a certain application we only need to construct an appropriate oracle to be used in step 1 of the algorithm; steps 2 and 3 are independent of the specific application.

Step 2 of the IPCG algorithm deals with the updating of Ω_n. The values of $\bar{c}_{n+1}$, $\mathbf{a}_{n+1}$ and $\mathbf{Q}_{n+1}$ are available from the first step. In certain cases, the only operation required is to add the inequality (2.2.2) defined by these parameters to the set of inequalities that define Ω_n in order to form Ω_{n+1}. However, as we mentioned above, some oracles may generate only the linear and quadratic components, $\mathbf{a}_{n+1}$ and $\mathbf{Q}_{n+1}$. A suitable new constant c_{n+1} can be determined as we describe below. Generally speaking, a cut can be placed either directly through the current center $\mathbf{y}_n$ (aggressive cut) or one can allow for a small positive slack in determining the constant c_{n+1}. The main reason for the latter procedure is that it helps to ensure fast theoretical convergence of the re-centering iterations in step 3. Specifically, we can compute the constant c_{n+1} of the separating inequality as

$$c_{n+1} := \beta r + \langle \mathbf{a}_{n+1}, \mathbf{y}_n \rangle + \langle \mathbf{y}_n, \mathbf{a}_{n+1} \rangle + \frac{1}{2}\langle \mathbf{y}_n, \mathbf{Q}_{n+1}\mathbf{y}_n \rangle, \tag{2.2.3}$$

where $\beta \geq (\sqrt{2}+1)^2$ is a constant and

$$r := \langle \mathbf{h}, (\nabla^2 \phi_n(\mathbf{y}_n))^{-1}\mathbf{h}\rangle^{1/2}, \quad \mathbf{h} := \mathbf{a}_{n+1} + \mathbf{Q}_{n+1}\mathbf{y}_n$$

and $\phi_n(\cdot)$ is the logarithmic barrier function of Ω_n. With this new value of c_{n+1}, the separating inequality becomes

$$c_{n+1} - \langle \mathbf{a}_{n+1}, \mathbf{y}\rangle - \langle \mathbf{y}, \mathbf{a}_{n+1}\rangle - \frac{1}{2}\langle \mathbf{y}, \mathbf{Q}_{n+1}\mathbf{y}\rangle \geq 0.$$

It can be easily checked that $\mathbf{y}_n$ satisfies this inequality. Adding this separating inequality to Ω_n we obtain Ω_{n+1}:

$$\Omega_{n+1} = \left\{ \mathbf{y} \in \mathbb{C}^M \ : \ c_{n+1} - \langle \mathbf{a}_{n+1}, \mathbf{y}\rangle - \langle \mathbf{y}, \mathbf{a}_{n+1}\rangle - \frac{1}{2}\langle \mathbf{y}, \mathbf{Q}_{n+1}\mathbf{y}\rangle \geq 0 \right\} \cap \Omega_n.$$

Notice that we always have $\Gamma \subset \Omega_{n+1} \subset \Omega_n$.

The third step of the IPCG algorithm involves the computation of an approximate analytic center of Ω_{n+1}. Since the exact analytic center is too costly to compute, the IPCG algorithm tries to find an *approximate* analytic center of Ω_{n+1}. The proximity from a point $\mathbf{y}$ to the exact analytic center is measured by the norm of the Hessian scaled gradient:

$$\delta(\mathbf{y}) := \langle \nabla\phi_{n+1}(\mathbf{y}), (\nabla^2\phi_{n+1}(\mathbf{y}))^{-1}\nabla\phi_{n+1}(\mathbf{y})\rangle^{1/2},$$

where ϕ_{n+1} denotes the potential function of Ω_{n+1}, $\nabla\phi_{n+1}(\mathbf{y})$ denotes the gradient vector of ϕ_{n+1} evaluated at $\mathbf{y}$, and $\nabla^2\phi_{n+1}(\mathbf{y})$ is the Hessian matrix of ϕ_{n+1} evaluated at $\mathbf{y}$. Clearly, if $\mathbf{y}$ is the exact analytic center of Ω_{n+1}, we have $\nabla\phi_{n+1}(\mathbf{y}) = 0$, implying $\delta(\mathbf{y}) = 0$. The vector $\mathbf{y}$ is said to be an η-approximate analytic center of Ω_{n+1} if $\delta(y) \leq \eta$, for some fixed $\eta \in (0,1)$. Given any η-approximate analytic center $\mathbf{y}$ of Ω_{n+1}, we can obtain an η^2-approximate center $\mathbf{y}^+$ by performing the following Newton iteration:

$$\mathbf{y}^+ := \mathbf{y} - \alpha_N(\nabla^2\phi_{n+1}(\mathbf{y}))^{-1}\nabla\phi_{n+1}(\mathbf{y}), \tag{2.2.4}$$

where α_N is the Newton step size. Indeed, it can be shown (Luo and Sun, 1999) that the proximity measure is reduced quadratically after each Newton iteration: $\delta(y^+) \leq \delta(\mathbf{y})^2$. Thus, suppose we have a $\frac{1}{2}$-approximate center of Ω_{n+1}, we can obtain a $\frac{1}{4}$-approximate center by performing just *one* Newton iteration. In the IPCG algorithm, we typically require the iterate $\mathbf{y}_n$ to be an η-approximate center of Ω_n, for some fixed η (say, $\eta = 1/4$). When a new cut is added to Ω_n, we usually ensure that $\mathbf{y}_n$ remains an η'-approximate center for Ω_{n+1}, where $0 < \eta < \eta' < 1$ (say, $\eta' = 1/2$). Recall that the parameter β in (2.2.3) is introduced for just this purpose. Subsequently, we can perform a few Newton iterations starting from $\mathbf{y}_n$. This will re-center $\mathbf{y}_n$ and yield an η-center $\mathbf{y}_{n+1}$ of Ω_{n+1}. By the quadratic convergence of the proximity measure, this re-centering process is extremely efficient; it rarely requires more than one Newton iteration in the re-centering step.

2.3 Adaptive Filtering Applications

In this section we show how Interior Point Column Generation can be applied to the adaptive filtering problem (introduced in Chapter 1). We then discuss some practical modifications that are necessitated by the effects of noise. Section 2.3.3 contains some simulation results where IPCG is compared to RLS in a system identification application.

2.3.1 IPCG Algorithms for Adaptive Filtering

Our first step is to convert the optimization problem into a feasibility problem so that the IPCG algorithm can be applied. Then we specify how to identify a violated constraint and generate the corresponding $\mathbf{a}_{n+1}$ and $\mathbf{Q}_{n+1}$. These will be used in steps 2 and 3 of the IPCG algorithm.

We define the feasible region Γ_{ls} as follows,

$$\Gamma_{ls}(n) = \left\{\mathbf{w} \in \mathbb{C}^M \mid \mathcal{F}_n(\mathbf{w}) \leq \tau^2\right\} \tag{2.3.1}$$

Note that we must specify a threshold τ^2. Consider a stationary environment: in this case the noise is not filtered through the adaptive filter, thus we know that as $\mathbf{w}(n)$ approaches the optimal filter $\mathbf{w}^*$, $\mathcal{F}_n(\mathbf{w})$ approaches the noise power σ_N^2. The threshold should therefore be selected proportional to σ_N^2.

It is now easy to see how the oracle can determine whether a given filter $\mathbf{w}_n$ is feasible: it must test the condition given in $\Gamma_{ls}(n)$. If the inequality in (2.3.1) is satisfied, the tested filter $\mathbf{w}_n$ is feasible and the oracle is done. However, if $\mathbf{w}_n$ is found to be infeasible, the oracle must generate a quadratic cut. Recall that a generic quadratic cut takes on the form:

$$c - \langle \mathbf{a}, \mathbf{y} \rangle - \langle \mathbf{y}, \mathbf{a} \rangle - \frac{1}{2}\langle \mathbf{y}, \mathbf{Q}\mathbf{y} \rangle \geq 0. \tag{2.3.2}$$

The oracle is only required to specify the linear and quadratic terms $\mathbf{a}$ and $\mathbf{Q}$. Therefore, in our case, we need to determine $\mathbf{a}_{n+1}$ and $\mathbf{Q}_{n+1}$ for a new cut at time $n+1$. Since the current filter $\mathbf{w}_n$ is infeasible, the least squares error $\mathcal{F}_{n+1}(\mathbf{w}_n)$ exceeds τ^2. The constraint that is violated can be expressed using (1.2.4)

$$\begin{aligned} \mathcal{F}_{n+1}(\mathbf{w}_n) &= \frac{1}{n+1}\mathbf{y}_{n+1}^H\mathbf{y}_{n+1} - \mathbf{w}_n^H\mathbf{p}_{xy}(n+1) - \mathbf{p}_{xy}^H(n+1)\mathbf{w}_n + \mathbf{w}_n^H\mathbf{R}_{xx}(n+1)\mathbf{w}_n \\ &\leq \tau^2. \end{aligned}$$

We rearrange the terms in the equation above to obtain a separating inequality in the

generic format (2.3.2),

$$(\tau^2 - \frac{\mathbf{y}_{n+1}^H \mathbf{y}_{n+1}}{n+1}) - \langle -\mathbf{p}_{xy}(n+1), \mathbf{w}_n \rangle - \langle \mathbf{w}_n, -\mathbf{p}_{xy}(n+1) \rangle - \frac{1}{2}\langle \mathbf{w}_n, 2\mathbf{R}_{xx}(n+1)\mathbf{w}_n \rangle \geq 0.$$

The required information $\mathbf{a}_{n+1}$ and $\mathbf{Q}_{n+1}$ for this quadratic cut can be readily obtained as

$$\mathbf{a}_{n+1} = -\mathbf{p}_{xy}(n+1), \tag{2.3.3}$$

$$\mathbf{Q}_{n+1} = 2\mathbf{R}_{xx}(n+1). \tag{2.3.4}$$

Note that $\mathbf{Q}_{n+1}$ is given by a scaled autocorrelation matrix and thus is symmetric positive semidefinite as required by the column generation algorithm. The above definition of $\Gamma_{ls}(n)$ uses a fixed threshold τ^2. In applications where the noise power is unknown or the channel is time varying, we have found that an adaptive thresholding technique works well; see Section 3.2 for details.

Note that our approach can easily be modified to include a forgetting factor for tracking time-varying systems, or to a sliding-window implementation, where the error is only accumulated over a finite observation window. It is sufficient to modify the formulation of the mean-squared error used in (2.3.1). If $\lambda \leq 1$ denotes the forgetting factor and I is the interval of consideration for the finite memory case, the cost function can be expressed as

$$\mathcal{F}_{n,\lambda,I}(\mathbf{w}) = \sum_{i=n-I+1}^{n} \lambda^{n-i} \left| y_i - \mathbf{x}_i^H \mathbf{w} \right|^2$$

We will see in the simulations of Section 2.4 that $\mathcal{F}_{n,\lambda,I}(\mathbf{w})$ allows for better tracking in the non-stationary case. Moreover, it requires less computational effort than the infinite memory approach.

2.3.2 Practical Modifications to the IPCG Algorithm

The presence of noise in adaptive systems necessitates some practical changes to the IPCG algorithm. Because of noise it is possible that at some iteration n the region $\Gamma_{ls}(n)$ is located far away from the true system w_* (as shown in Figure 2.3). If at such a time the current solution $\mathbf{w}_n$ lies somewhere in-between $\mathbf{w}_*$ and $\Gamma_{ls}(n)$, the oracle may return a cut removing $\mathbf{w}_*$ permanently from Ω_n.

In a practical implementation of the IPCG algorithm we thus take several precautions. To avoid the generation of bad cuts we want the error threshold τ^2 to be slightly larger than the average value of the error $\mathcal{F}_n(\mathbf{w})$. If the threshold was lower we would generate unnecessarily many cuts without actually being able to minimize $\mathcal{F}_n(\mathbf{w})$ further. If the

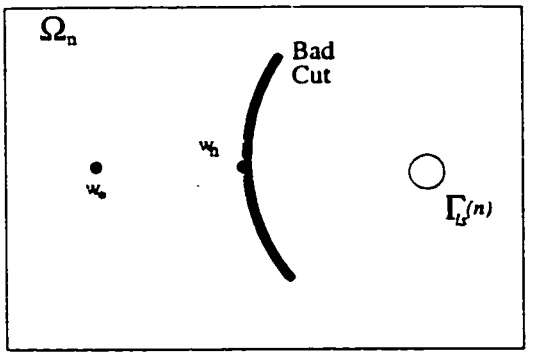


Figure 2.3: *Noise causing a bad cut*

threshold was much larger, then the steady-state mean-square error in $\mathbf{w}_n$ may be unacceptable. Assuming no *a priori* knowledge of the output noise power, one way of determining the threshold τ^2 is an adaptive thresholding technique as outlined below.

1. Start with a very small threshold value to promote early cuts for faster convergence.
2. Define a threshold adaptation parameter, α_τ (usually $\alpha_\tau \approx 0.05$).
3. If the current error $\mathcal{F}_n(\mathbf{w})$ falls into the interval

$$(1 - 2\alpha_\tau)\tau^2 \leq \mathcal{F}_n(\mathbf{w}) \leq (1 + \alpha_\tau)\tau^2$$

   the threshold remains unchanged for the next iteration.
4. Threshold reduction: *if* $\mathcal{F}_n(\mathbf{w}) < (1 - 2\alpha_\tau)\tau^2$, *then* $\tau^2 := (1 - \alpha_\tau)\tau^2$.
5. Incrementing threshold: *if* $\mathcal{F}_n(\mathbf{w}) > (1 + \alpha_\tau)\tau^2$, *then* $\tau^2 := (1 + 2\alpha_\tau)\tau^2$.

Points 4 and 5 purposely reflect a certain bias of the method to select a threshold slightly higher than the mean-squared error. In this way we hope to minimize the overall number of cuts that need to be generated.

In the case of unbounded additive disturbances, such as additive white Gaussian noise, a bad cut (as described above) may occur no matter how high the threshold. Such cuts should periodically be purged from the set of inequalities defining the current Ω_n. After a cut has been purged, the set Ω_n is enlarged and $\mathbf{w}_n$ is no longer an approximate analytic center of the resulting set. Re-centering $\mathbf{w}_n$, however, is not necessary. Either $\mathbf{w}_n$ will

be found to be feasible in the next iteration, or, if not, a cut will be added and Newton iterations (2.2.4) will update $\mathbf{w}_n$ to the new approximate center of Ω_{n+1}. Cut removal can be implemented by maintaining only a fixed number of inequalities and removing one whenever the maximum is exceeded. Cuts to be removed can be identified, for example, as those most opposed to the most recent direction of update (of $\mathbf{w}_n$). Alternatively, one could simply delete the oldest cut each time the limit is attained. Maintaining a small number of cuts in Ω_n has the advantage that the updating to a new approximate analytic center is made easier. Removing old cuts also represents a natural mechanism for tracking a time-varying system, where old cuts may be obsolete after the system has undergone substantial changes.

Another implementation issue concerns the transient phase of adaption. We may encounter the scenario that a given filter $\mathbf{w}_n$ remains feasible for many iterations $n' > n$. While this "resting time" saves computations (no updates are necessary) it also causes a deterioration in transient performance. To ensure the continued adjustment of $\mathbf{w}_n$ we therefore define a cut interval parameter I_c that signifies that a cut is made every I_c iterations regardless of the current error.

2.3.3 Simulations

In this section we present numerical simulation results to evaluate the performance of the IPCG algorithm as compared with the RLS method in the system identification scenario. We are interested mainly in the speed and accuracy of convergence of the IPCG and RLS methods. To this end we use the following performance measures. We consider the instantaneous squared error,

$$\varepsilon_y(n) = |y(n) - \mathbf{x}^H(n)\mathbf{w}_n|^2. \tag{2.3.5}$$

and the error in the weight vector $\mathbf{w}_n$ itself. The mean squared weight vector error is defined as

$$\varepsilon_w(n) = \|\mathbf{w}_n - \mathbf{w}_*\|^2, \tag{2.3.6}$$

where $\mathbf{w}_*$ is the plant to be identified.

In implementing the RLS and the IPCG algorithms we must select certain parameters. In the case of RLS this is relatively straightforward: the forgetting factor λ must be chosen and the algorithm is initialized by choosing $\mathbf{w}_0$, and a matrix $\mathbf{P}(0)$ which is equal to the inverse of the estimated autocorrelation matrix. The initialization affects the transient performance of RLS but is not crucial to the convergence if the data length is large (Haykin,

1998). In the case of IPCG algorithms a larger number of parameters can be adjusted. To limit the scope of simulation, we fix these parameters to the values shown in Table 2.1.

Table 2.1: *IPCG parameter settings*

Parameter	Value	Description
β	2.0	indicates aggressiveness of cut
δ_{th}	10^{-4}	limit for distance between approximate and true analytic centers
α_N	1.0	Newton step size
τ^2	0.03	initial threshold for mean squared error $\varepsilon(n)$
α_τ	0.03	threshold adaptation parameter (see section 3.2)
n_{cuts}	2	maximum number of cuts maintained
R	100	radius of Ω_0
I_c	20	Interval at which a cut will be made for sure

2.3.3.1 System models

The unknown plant is modeled as a three-ray multipath communication channel characterized by the impulse response shown in Figure 2.5. The impulse response has length $M = 24$ and is composed of three delayed and amplified raised cosine pulses with roll-off factor 0.11.

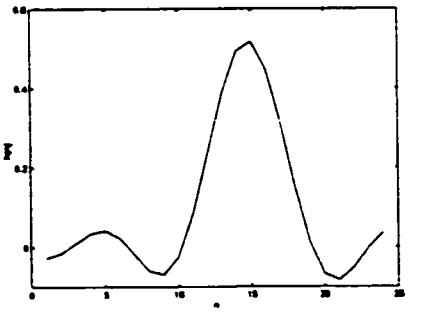


Figure 2.4: *Channel impulse response* $h(n)$

Noise is assumed to be additive white Gaussian at a specified signal-to-noise ratio. We use two different source models: the first is a white Gaussian noise sequence, and the second source model is a correlated noise sequence. The correlated source sequence is obtained by taking a 4 sample average of the white noise sequence $\{v(n)\}$,

$$v_c(n) = \frac{1}{2}(v(n) + v(n-1) + v(n-2) + v(n-3)), \qquad n = 4, 5, \ldots$$

By using such a coloring filter we increase the eigenvalue spread of the correlation matrix of the signal.

2.3.3.2 Comparison of IPCG and RLS for Adaptive Filtering

We examine the convergence behaviour of the new IPCG algorithm in two separate experiments. The first experiment investigates the robustness of IPCG to variations in source correlation and SNR levels. An abrupt change in the system is considered in Experiment 2.2. For reference, an implementation of the RLS algorithm was tested on the same data. All results were obtained by repeating 200 Monte Carlo trials and averaging the respective errors.

Experiment 2.1: System Identification

We first test both algorithms with white noise at the input and the channel of Figure 2.4. Noise is added at an SNR of 20 dB. The IPCG algorithm is configured with the parameter settings of Table 1. The RLS algorithm is run with a forgetting factor of $\lambda = 1.0$. The weight vector and mean squared errors for both algorithms are plotted in Figures 2.5a and 2.5b. It can be seen that in this case IPCG closely matches the RLS performance. In fact, the mean squared errors not only converge to the same amplitude but appear to be identical in steady state (*cf.* Figure 2.5b).

We repeat the same test at an SNR of 5dB. The performance of both algorithms deteriorates as shown in Figure 2.5c. They still converge to the same steady state error, but IPCG displays clearly better transient performance. It should be noted here, that RLS' performance could be improved by assuming a higher noise level in the initialization stage. This, however, would require *a priori* knowledge of the noise power. In this experiment we did not assume such knowledge for either of the algorithms.

In the third test scenario we use the correlated source and increased the SNR to 15 dB. The results of this run are depicted in Figure 2.5d. We observe again that IPCG show better initial convergence than RLS. While IPCG converges to its steady state error in less than 50 iterations, RLS takes over 100 iterations to reach the same low error level of IPCG. Figure 2.5d was obtained using $\beta = 20$. Note that optimizing the initialization of RLS in this case hardly improves its convergence. IPCGs better performance persists as long as algorithms are initialized in their normal operating regions. The computational complexity of IPCG in this experiment was on average 15 to 20 times that of RLS.

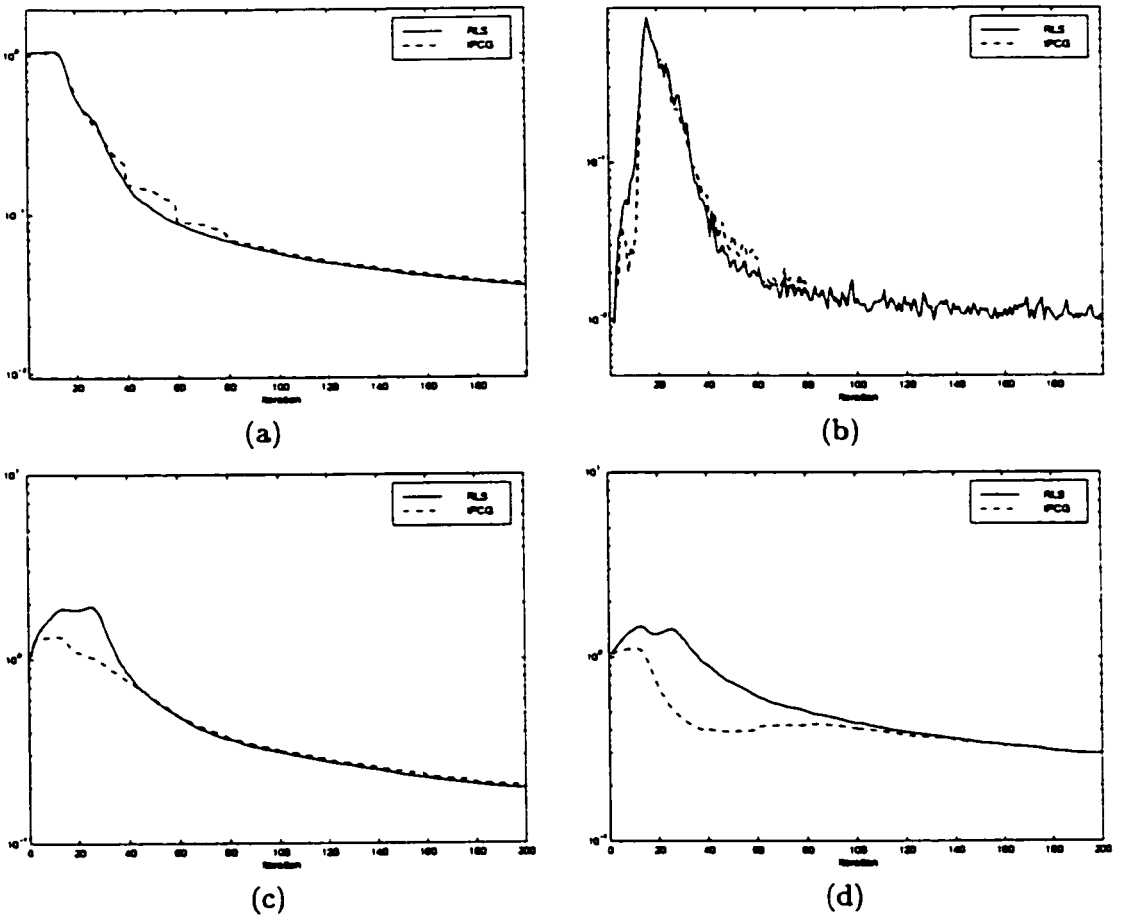


Figure 2.5: *Experiment 2.1: (a) Weight vector error (high SNR) (b) Instantaneous squared error (c) Weight vector error (low SNR) (d) Weight vector error (high SNR, correlated inputs)*

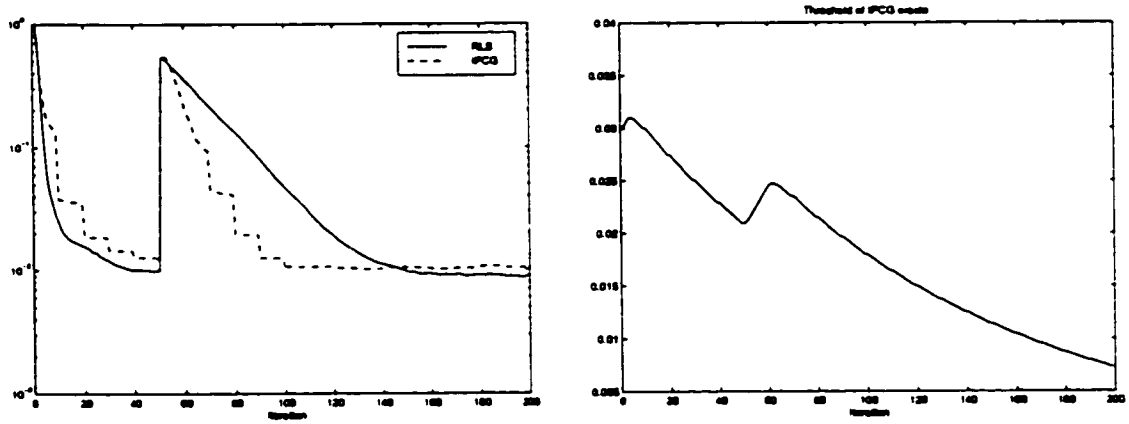
Experiment 2.2: System Identification (abrupt change)

As was seen in Experiment 2.1, IPCG can be more accurate especially during the transient phase of the training period. We now want to investigate whether this advantage can be exploited for the tracking of time-varying systems. We thus test the ability of both algorithms to track a sudden change in the unknown system that occurs during the identification process.

The input sequence is an uncorrelated Gaussian signal. This signal is convolved first simply with a delta function, and after 50 iterations with the channel impulse response of Figure 2.4. White Gaussian noise is added at an SNR of 30 dB.

Both RLS and IPCG use a forgetting factor $\lambda = 0.95$. Further parameter setting for IPCG were as in Table 2.1 with the exception that $\alpha_\tau = 0.01$. Also, for IPCG we used an oracle based on the finite interval error $\mathcal{F}_{n,\lambda,I}$ and the interval length was set to $I = 10$.

The results of the simulation are depicted in Figure 2.6. We can see that after the abrupt change (at Iteration 50), IPCG reacquires the unknown system almost twice as fast as RLS. In a practical scenario, this could for example mean a 50% saving in the length of a training signal required, after the receiver has lost lock on a current communication channel. Figure 2.7b is included to illustrate the adaptive thresholding procedure described in Section 2.3.2.



(a) (b)

Figure 2.6: *Experiment 2.2: (a) Weight vector error, (b) Adaptive thresholding*

2.4 Discussion

We demonstrated the applicability of Interior Point Column Generation algorithms to a dynamic problem in engineering, adaptive filtering. Our simulation results are encouraging: in adverse conditions the transient convergence of IPCG is superior to RLS. When conditions are less hostile, the IPCG algorithm consistently matches the RLS performance. Results also indicate that the IPCG algorithm is robust to changes in input signal and channel characteristics.

The IPCG algorithm for adaptive filtering bears a certain resemblance to set-theoretic approaches (see Section 1.2.4) in that it always maintains a feasible set of filters and chooses as its estimate a filter that is near the center of this convex feasible set. An important difference is that the set-theoretic approaches in the literature are always formulated in the

context of a *bounded error* environment, and thus of limited usefulness in such applications such as telecommunications, where a noise bound can not easily be imposed.

While the performance of the IPCG algorithm encourages further investigation, there are important issues to be addressed. Practical modifications (*e.g.*, adaptive thresholding, cut removal) were introduced in Section 2.3.2 to handle the problems arising with the presence of unbounded noise. These are heuristic procedures, and no claim for optimality can be made. Furthermore, they hinder the theoretical analysis of the algorithm. For example, no convergence results could be obtained. The algorithm is also computationally intensive. The update of the analytic center at each iteration is obtained by taking at least one Newton step, which is $O(M^3)$. In the case of unlimited cuts, the complexity depends on time as well. The Interior Point Least Squares algorithm of Chapter 3 achieves improvements on all these issues.

Chapter 3

Adaptive Linear Filtering using Interior Point Optimization Techniques

We propose a new approach for the linear adaptive filtering problem using techniques from interior point optimization. Similar to the IPCG approach of Chapter 2, this method formulates a convex feasibility problem at each iteration and obtains as an estimate a filter near the center of the feasible region. The difference lies in how the convex feasible region is defined. It now only contains two inequalities throughout, and it is by controlling the relative importance of these inequalities that convergence of the estimator is achieved.

In this chapter we concentrate on the algorithm and its practical implementation. It is shown that the algorithm can be made recursive with a per-sample complexity of $O(M^{2.2})$, where M is the filter length. The potential of the algorithm for practical applications is demonstrated via numerical simulations where the new algorithm is shown to have superior transient behaviour and improved robustness to the source signal statistics when compared to the recursive least-squares (RLS) method.

3.1 Introduction

The linear filtering problem (1.2.5) has been the subject of active research over the past few decades (Haykin, 1998). This is because of the important role of adaptive filtering in signal processing and data communication where it is used to solve a wide range of

problems including system identification, adaptive channel equalization, echo cancellation, linear predictive speech coding and adaptive differential coding. The work in recent years has led many to regard the recursive least-squares (RLS) algorithm and its variants as the state of the art. The RLS algorithm computes the solution $(\frac{\delta}{n}\mathbf{I} + \mathbf{R}_{xx}(n))^{-1}\mathbf{p}_{xy}(n)$ recursively, requiring $O(M^2)$ operations per sample. The extra term $\frac{\delta}{n}\mathbf{I}$, where δ is a *fixed* initialization constant, is used to handle the problem of ill-conditioning (or singularity) of the matrix $\mathbf{R}_{xx}(n)$, as caused by, for example, the lack of observation data in the initial phase. The choice of δ significantly affects filtering performance since it determines the exact amount of regularization for the remainder of the adaptive process. In the absence of estimation uncertainties about $\mathbf{R}_{xx}(n)$ and $\mathbf{p}_{xy}(n)$ it is the regularization term that determines the rate at which the RLS estimator converges to the optimum linear filter (see also Moustakides, 1997).

In this chapter we reformulate the adaptive filtering problem as a convex feasibility problem and propose to use the approximate *analytic center* of the feasible region as an estimate of the optimal filter $\mathbf{R}_{xx}(n)^{-1}\mathbf{p}_{xy}(n)$. This estimator takes the form of $(\alpha_n\mathbf{I} + \mathbf{R}_{xx}(n))^{-1}\mathbf{p}_{xy}(n)$, where α_n is a *time-varying* parameter. We introduce a recursive Interior Point Least Squares (IPLS) algorithm, with per-sample complexity $O(M^{2.2})$, to compute this estimator efficiently. Our algorithm is based on recently developed interior point optimization techniques and can be shown to retain the $O(1/n)$ asymptotic convergence rate of the RLS algorithm (Chapter 4). Most importantly, the algorithm is able to adjust the regularization parameter α_n intelligently, ensuring that α_n is relatively large when the estimated filter is far from the optimum linear filter and small when the estimated filter is close to the optimum linear filter. As a result, the algorithm is observed to be robust to variations in its initialization and possesses superior (exponentially decaying) transient behaviour compared to RLS. The transient convergence of IPLS is demonstrated experimentally by computer simulations. The theoretical analysis of transient convergence follows in Chapter 4.

Previously, interior point optimization approaches to adaptive filtering have been based on the idea of Interior Point Column Generation (see previous chapter or Bai *et al.*, 1999). Their strategy is to iteratively add cuts to a feasible region, such that a "shrinking" sequence of feasible regions is obtained. Eventually, the center of the last feasible region will be close to the optimum linear filter. These types of algorithms encounter difficulties when the measurements are noisy, because noise can in fact cause the optimum filter to be cut from the feasible region. This difficulty is remedied in the IPCG algorithm of Chapter 2 by maintaining only a fixed number of cuts in the description of feasible sets. Should a "bad" cut occur it would be eliminated eventually; however the sequence of feasible sets is no longer

shrinking, thus convergence becomes difficult to establish. Alternatively, Bai *et al.* (1999) assume bounded noise to circumvent this problem. Their method suffers from growing computational complexity: the number of operations required to update the filter estimate grows with the number of observations n. In contrast, the IPLS algorithm proposed here has a low per-sample complexity of $O(M^{2.2})$ which does not grow with n and its asymptotic convergence can be established.

The remainder of the chapter is organized as follows. In Section 3.2, we introduce an analytic center based estimator. The Interior Point Least Squares (IPLS) algorithm for adaptive filtering is introduced in Section 3.3 to approximately implement the estimator described in Section 3.2. We also develop a recursive updating scheme in this section to efficiently compute the IPLS estimator. Then in Section 3.4 we provide some numerical simulations that show the promise of the IPLS algorithm, especially in terms of its transient convergence for $n < M$. We close the chapter with some discussion in Section 3.5.

3.2 An Analytic Center based Estimator

Recall that the linear filtering problem is formulated as finding a weight vector $\mathbf{w}$ that minimizes the cost $\mathcal{F}_n(\mathbf{w})$ (*cf.* 1.2.5). Consequently, we can define the following convex feasibility region:

$$\Omega_n = \{\mathbf{w} \in \mathbb{C}^M \mid \mathcal{F}_n(\mathbf{w}) \leq \tau_n,\ \|\mathbf{w}\|^2 \leq R^2\}, \tag{3.2.1}$$

where $\tau_n > 0$ is an appropriately chosen scalar and $R > 0$ is a fixed number. Any vector $\mathbf{w} \in \Omega_n$ is said to be feasible at time n. Obviously, we would like τ_n to become small quickly, because then any feasible vector $\mathbf{w}$ will approximately minimize $\mathcal{F}_n(\mathbf{w})$. The additional inequality $\|\mathbf{w}\|^2 \leq R^2$ is important since it ensures that the region Ω_n is bounded.

For the above convex feasible region Ω_n, the *logarithmic barrier function* becomes

$$\phi_n(\mathbf{w}) = -\log(\tau_n - \mathcal{F}_n(\mathbf{w})) - \log(R^2 - \|\mathbf{w}\|^2), \qquad \mathbf{w} \in \Omega_n.$$

Recall that this barrier function is strictly convex over Ω_n, and thus admits a unique minimizer: the analytic center $\mathbf{w}_n^a$. Consider $\mathbf{w}_n^a$ as an estimator for the optimal solution of the linear filtering problem. Intuitively, if τ_n and R are chosen in a way to emphasize the first constraint, then the analytic center $\mathbf{w}_n^a$ will be a close approximation of the minimizer of $\mathcal{F}_n(\mathbf{w})$. Setting $\nabla\phi_n(\mathbf{w}_n^a) = 0$, we obtain

$$\frac{\nabla\mathcal{F}_n}{s_n(\mathbf{w}_n^a)} + \frac{2\mathbf{w}_n^a}{t_n(\mathbf{w}_n^a)} = 0 \tag{3.2.2}$$

where $s_n(\mathbf{w}) := \tau_n - \mathcal{F}_n(\mathbf{w})$ and $t_n(\mathbf{w}) := (R^2 - \|\mathbf{w}_n^a\|^2)$ are the slacks associated with the two quadratic constraints of Ω_n. It should be noted here that the gradient is taken with respect to the complex valued tap weight vector $\mathbf{w}$ (following the convention of Haykin, 1998, Appendix B). Substituting

$$\nabla\mathcal{F}_n(\mathbf{w}) = -2\mathbf{p}_{xy}(n) + 2\mathbf{R}_{xx}(n)\mathbf{w}$$

in the above relation and rearranging, we further obtain

$$\mathbf{w}_n^a = \left(\frac{s_n}{t_n}\mathbf{I} + \mathbf{R}_{xx}(n)\right)^{-1}\mathbf{p}_{xy}(n) = (\alpha_n\mathbf{I} + \mathbf{R}_{xx}(n))^{-1}\mathbf{p}_{xy}(n). \tag{3.2.3}$$

The term $\alpha_n\mathbf{I}$ can be viewed as a regularization term for the matrix $\mathbf{R}_{xx}(n)$ (which may be poorly conditioned). The above expression for $\mathbf{w}_n^a$ suggests that the estimator $\mathbf{w}_n^a$ is similar to the RLS estimator which takes the form of $(\frac{\delta}{n}\mathbf{I}+\mathbf{R}_{xx}(n))^{-1}\mathbf{p}_{xy}(n)$. The difference lies in that for the analytic center $\mathbf{w}_n^a$, the value of α_n changes (actually depends on $\mathbf{w}_n^a$) with data. The initialization of α_n, however, depends on the filter $\mathbf{w}_{n-1}^a$) of the previous iteration. In our proposed algorithm, $\alpha_n(\mathbf{w}_{n-1}^a)$ is proportional to the quantity $\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1}^a)\|_2$. As a result, $\alpha_n\mathbf{I}$ will provide sufficient regularization to $\mathbf{R}_{xx}(n)$ in the initial phase when it is poorly conditioned. As n becomes large, $\mathbf{R}_{xx}(n)$ becomes better conditioned, $\mathcal{F}_n(\mathbf{w})$ will be approximately minimized and $\nabla\mathcal{F}_n(\mathbf{w}_{n-1}^a)$ approaches zero. This implies $\alpha_n \to 0$ and the effect of regularization is gradually eliminated as desired.

It remains to specify how τ_n is updated in each iteration. An important consideration here is to ensure the proportionality of α_n to $\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1}^a)\|_2$ which is essential to the convergence behaviour of our algorithm. This can be accomplished by choosing

$$\begin{aligned}\tau_n &:= \mathcal{F}_n(\mathbf{w}_{n-1}^a) + \beta r_n \\ &:= \mathcal{F}_n(\mathbf{w}_{n-1}^a) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1}^a)\|_2,\end{aligned} \tag{3.2.4}$$

where $\beta > 0$ is a constant chosen by the user. With this choice of τ_n, the added "slack" becomes $s_n(\mathbf{w}_{n-1}^a) = \beta r_n = \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1}^a)\|_2$ which in turn guarantees $\alpha_n(\mathbf{w}_{n-1}^a) \sim \|\nabla\mathcal{F}_n(\mathbf{w}_{n-1}^a)\|_2$.

Another important consequence of (3.2.4) is that $\mathbf{w}_{n-1}^a$ remains feasible with respect to the updated region Ω_n so that $\mathbf{w}_{n-1}^a$ can be used as a starting point for a Newton procedure to find the analytic center $\mathbf{w}_n^a$ of Ω_n. The condition $\mathcal{F}_n(\mathbf{w}_{n-1}^a) \leq \tau_n$ is easily verified with (3.2.4). The $(R/\sqrt{2})$ term in s_n provides a normalization that is required to show the convergence of the algorithm (see Chapter 4). The parameter β is typically used in interior point methods to control the "aggressiveness" with which constraints are generated. For example, a small β corresponds to a small allowed slack, *viz*, $\mathbf{w}_{n-1}^a$ is left

close to the boundary of Ω_n. By increasing β we allow more slack and hence define Ω_n more conservatively.

Let us again use an example in $\mathbb{R}^2$ to illustrate the principles of the algorithm.

Example 3.1

At time zero the feasible region Ω_0 is determined only by the norm condition $\|\mathbf{w}\|^2 \leq R^2$ with $R = 100$. The initial filter estimate is $\mathbf{w}_0^a = [0 \quad 0]^T$. When the first data point $\{\mathbf{w}_1^a, y_1\}$ is observed we can update Ω_0 to Ω_1 by calculating $\mathbf{R}_{xx}(1)$, $\mathbf{p}_{xy}(1)$. We also update τ_n according to (3.2.4). Now the new filter estimate can be found by computing the new analytic center of Ω_1. We can see in Figure 3.1 that after one iteration the inequality $\mathcal{F}_1(\mathbf{w}) \leq \tau_1$ still defines an unbounded region. This is so because with one data point the autocorrelation matrix is still rank deficient. In this case, the presence of the norm constraint ensures that among all filters that yield a small mean squared error we select one that also minimizes the norm. After two iterations $\mathbf{R}_{xx}(2)$ has become full rank and defines a bounded ellipse. The size of the ellipse is determined by $\|\nabla\mathcal{F}_2(\mathbf{w}_1^a)\|_2$: the "better" $\mathbf{w}_1^a$ was, the smaller the ellipse, or the more confident we are that $\mathbf{w}_2^a$ should be somewhere near $\mathbf{w}_1^a$. From the sequence of plots in Figure 3.1 we can see that the ellipses defined by the first inequality become progressively smaller, which indicates that we are becoming more and more confident in the analytic center filter estimate $\mathbf{w}_n^a$. □

To summarize, we have proposed the following procedure for the adaptive filtering problem:

An Analytic Center Based Adaptive Filtering Procedure

Step 1: Initialization. Select $R > 0$, $\beta > 0$. Let $\mathcal{F}_0(\mathbf{w}) = 0$ and $\Omega_0 = \{\|\mathbf{w}\|^2 \leq R^2\}$. Let $\mathbf{w}_0^a = 0$.

Step 2: Updating. For $n \geq 1$, update the feasible region Ω_n using the new data sample $(\mathbf{x}_n, y_n)$ and the new value τ_n according to (3.2.4).

Step 3: Recentering. Compute the analytic center $\mathbf{w}_n^a$ of Ω_n using the previous center $\mathbf{w}_{n-1}^a$. Set $n := n + 1$ and return to Step 2.

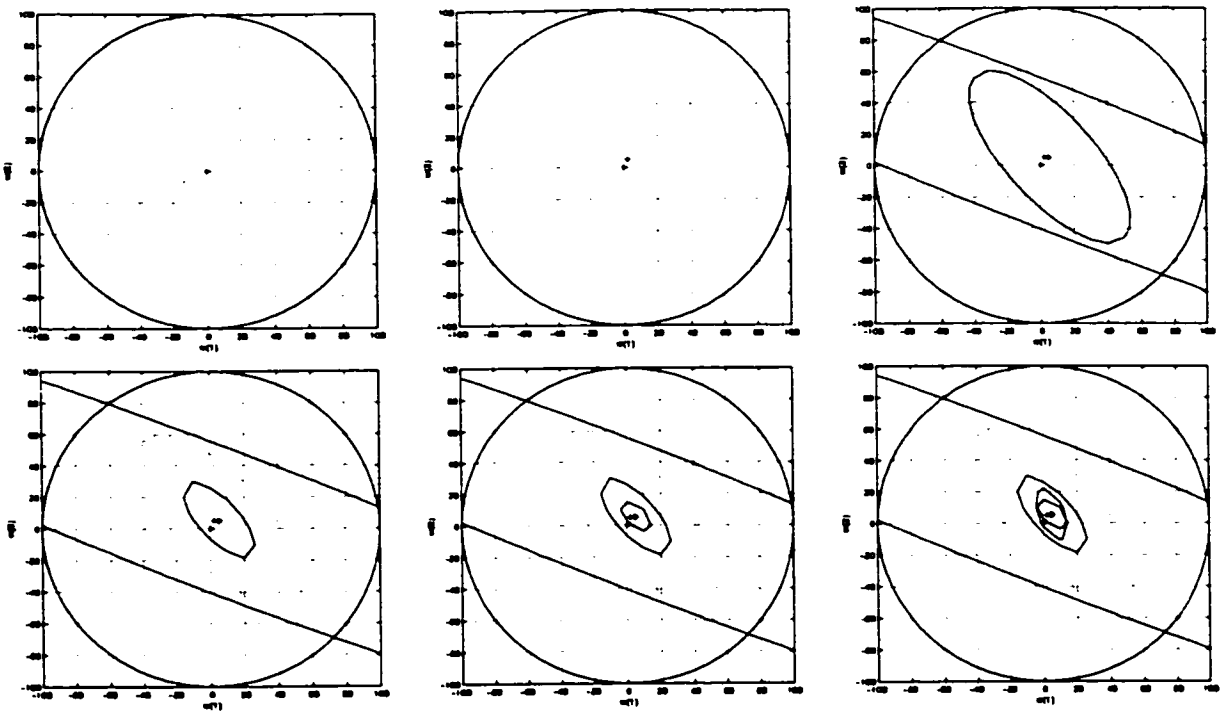


Figure 3.1: *Example 3.1: Six iterations of analytic center based adaptive filtering.*

3.3 Interior Point Least Squares Algorithm

The analytic center based adaptive filtering procedure proposed in Section 3.2 requires the calculation of the exact analytic center $\mathbf{w}_n^a$ of Ω_n for $n \geq 1$. This calculation involves the minimization of the logarithmic function $\phi_n(\mathbf{w})$ which typically requires an infinite number of iterations. Thus, the aforementioned adaptive filtering procedure is difficult to implement in practice. Fortunately, it can be shown (Chapter 4) that an *approximate* analytic center of Ω_n is indeed sufficient in the adaptive filtering procedure. Moreover, this approximate analytic center can always be updated by a *single* Newton iteration in the minimization of $\phi_n(\mathbf{w})$. The Newton iteration can be described as follows:

$$\mathbf{w}_n := \mathbf{w}_{n-1} - (\nabla^2 \phi_n(\mathbf{w}_{n-1}))^{-1} \nabla \phi_n(\mathbf{w}_{n-1}), \tag{3.3.1}$$

where by $\mathbf{w}_n$ we denote the approximate analytic center of Ω_n.

To compute (3.3.1) we require the gradient and Hessian of $\phi_n(\mathbf{w})$

$$\nabla\phi_n(\mathbf{w}) = \frac{\nabla\mathcal{F}_n}{s_n(\mathbf{w})} + \frac{2\mathbf{w}}{t_n(\mathbf{w})}, \tag{3.3.2}$$

$$\nabla^2\phi_n(\mathbf{w}) = \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^H}{s_n^2(\mathbf{w})} + \frac{\nabla^2\mathcal{F}_n}{s_n(\mathbf{w})} + \frac{4\mathbf{w}\mathbf{w}^H}{t_n^2(\mathbf{w})} + \frac{2\mathbf{I}}{t_n(\mathbf{w})} \tag{3.3.3}$$

where $s_n(\mathbf{w}) := \tau_n - \mathcal{F}_n(\mathbf{w})$ and $t_n(\mathbf{w}) := R^2 - \|\mathbf{w}\|^2$ are the slacks of $\mathbf{w}$ with respect to the first and the second inequalities in Ω_n, respectively. The gradient and Hessian of $\mathcal{F}_n(\mathbf{w})$ are given by

$$\begin{aligned}\nabla\mathcal{F}_n &= -2\mathbf{p}_{xy}(n) + 2\mathbf{R}_{xx}(n)\mathbf{w},\\ \nabla^2\mathcal{F}_n &= 2\mathbf{R}_{xx}(n).\end{aligned}$$

The threshold τ_n is now defined in terms of the approximate analytic center,

$$\tau_n = \mathcal{F}_n(\mathbf{w}_{n-1}) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|_2, \tag{3.3.4}$$

This provides all the information that is necessary for the Newton step.

By replacing the exact analytic center $\mathbf{w}_n^a$ by an approximate analytic center $\mathbf{w}_n$ in the Analytic Center Adaptive Filtering Procedure given in Section 3.2, we obtain an implementable Interior Point Least Squares (IPLS) algorithm where the approximate analytic centers are updated by a single Newton iteration. We summarize the steps of the IPLS algorithm as follows:

Interior Point Least Squares (IPLS) Algorithm

Step 1: Initialization. Let β, R be given. Set

$$\mathbf{w}_0 = \mathbf{0}, \quad \mathbf{p}_{xy}(0) = \mathbf{0}, \quad \mathbf{R}_{xx}(0) = \mathbf{0}, \quad \nabla\mathcal{F}_0(\mathbf{0}) = \mathbf{0}.$$

Step 2: Updating. For $n \geq 1$, acquire new data $\mathbf{x}_n, y_n$. Then recursively update

$$\begin{aligned}\mathbf{p}_{xy}(n) &= \frac{n-1}{n}\mathbf{p}_{xy}(n-1) + \frac{1}{n}\mathbf{x}_n y_n,\\ \mathbf{R}_{xx}(n) &= \frac{n-1}{n}\mathbf{R}_{xx}(n-1) + \frac{1}{n}\mathbf{x}_n\mathbf{x}_n^H.\end{aligned} \tag{3.3.5}$$

Update

- $\nabla\mathcal{F}_n(\mathbf{w}_{n-1}) = -2\mathbf{p}_{xy}(n) + 2\mathbf{R}_{xx}(n)\mathbf{w}_{n-1}$
- $s_n(\mathbf{w}_{n-1}) = \tau_n - \mathcal{F}_n(\mathbf{w}_{n-1}) = \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|_2$

- $t_n(\mathbf{w}_{n-1}) = R^2 - \|\mathbf{w}_{n-1}\|^2$
- $\nabla\phi_n(\mathbf{w}_{n-1})$, $\nabla^2\phi_n(\mathbf{w}_{n-1})$ using equations (3.3.2), (3.3.3)

Step 3: Recentering. The new center of Ω_n is obtained by taking just one Newton iteration starting at $\mathbf{w}_{n-1}$:

$$\mathbf{w}_n := \mathbf{w}_{n-1} - (\nabla^2\phi_n(\mathbf{w}_{n-1}))^{-1}\nabla\phi_n(\mathbf{w}_{n-1})$$

Set $n := n + 1$, and return to Step 2.

The choice of constant parameters β and R will be discussed in Chapter 4. Basically, the appropriate choices of β and R ensure the asymptotic convergence of $\mathbf{w}_n$ to the optimum filter. Note also that the update of the slack variable s_n in Step 2 implies the update of the threshold τ_n according to (3.3.4). We have chosen to update s_n directly in order to save the computation of $\mathcal{F}_n(\mathbf{w}_{n-1})$.

It can be easily seen that Step 2 (the recursive updating step) of IPLS algorithm requires $O(M^2)$ arithmetic operations per iteration. This is analogous to the computational requirement per iteration of the basic RLS algorithm. However, Step 3 (the Recentering step) involves the calculation of the Hessian inverse $(\nabla^2\phi_n(\mathbf{w}_{n-1}))^{-1}$ which requires $O(M^3)$ operations. Therefore the overall computational requirement per iteration is in the order of $O(M^3)$. Though this complexity is independent of n, it compares unfavorably to the RLS algorithm which has a per iteration complexity of $O(M^2)$. Next we outline a procedure to update the Newton direction $[\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\phi_n(\mathbf{w}_{n-1})$ in $O(M^{2.2})$ operations. With this procedure, the overall complexity per iteration of IPLS is reduced to $O(M^{2.2})$.

3.3.1 Recursive Update of Newton Direction

Suppose $\mathbf{A}$ is a square matrix of size $M \times M$ and $\mathbf{U}, \mathbf{V} \in \mathbb{C}^{M\times k}$. We first recall the following three facts from numerical linear algebra:

1. The Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1996) gives a convenient way of computing the rank k update of the inverse of A:

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^H)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^H\mathbf{A}^{-1}.$$

In particular, suppose $\mathbf{A}^{-1}$ is available, then the above updating formula takes $O(kM^2)$ operations (as opposed to $O(M^3)$ operations needed to invert $(\mathbf{A} + \mathbf{U}\mathbf{V}^H)$ from scratch). Notice that the RLS algorithm is based precisely on the above updating scheme (for $k = 1$).

2. Let $\Lambda \in \mathbb{C}^{M\times M}$ be a symmetric banded matrix of band-width $2s+1$, *viz*, its entries $\Lambda_{i,j}$ satisfy $\Lambda_{i,j} = 0$, $|i-j| > s$. Solving the linear system $\Lambda\mathbf{x} = \mathbf{b}$ takes $O(Ms^2)$ operations (Golub and Van Loan, 1996).

3. Let $\mathbf{P} \in \mathbb{C}^{M\times M}$ be a hermitian matrix that transforms $\mathbf{A}$ into a banded form, *viz*, $\mathbf{P}^H\mathbf{A}\mathbf{P} = \Lambda$, where Λ is a banded matrix of band-width $2s+1$. If $\mathbf{A}$ is modified by a rank-one matrix $\mathbf{u}\mathbf{u}^T$ and if $s = M^{3/5}$, it is possible to update $\mathbf{P}$ to a new hermitian matrix $\mathbf{P}'$ and Λ to a new $(2s+1)$-banded matrix Λ' in $O(M^{2.2})$ operations, such that

$$(\mathbf{P}')^H(\mathbf{A}+\mathbf{u}\mathbf{u}^H)\mathbf{P}' = \Lambda'.$$

In other words, the symmetric rank-one update of the transformation of $\mathbf{A}$ into a $(2s+1)$-banded matrix can be achieved in $O(M^{2.2})$ operations.

Facts 1 and 2 above are well known. Fact 3 has been treated by Powell (1998), where it is shown that the decomposition of a matrix into a banded structure after a rank-one update requires the computation of $\frac{3}{4}(M-1)^2/(s+1)$ Givens matrices. To jointly minimize the amount of work needed to update Λ and $\mathbf{P}$, s is set to $M^{3/5}$. Note that with $s = M^{3/5}$ the complexity of solving the banded system (Fact 2) also becomes $O(M^{2.2})$.

Let us now examine the structure inherent in the Hessian $\nabla^2\phi_n(\mathbf{w}_{n-1})$. Rearranging (3.3.3) and using the relation $\nabla^2\mathcal{F}_n = 2\mathbf{R}_{xx}(n)$ we obtain

$$\nabla^2\phi_n(\mathbf{w}_{n-1}) = \frac{2\mathbf{R}_{xx}(n)}{s_n(\mathbf{w}_{n-1})} + \frac{2\mathbf{I}}{t_n(\mathbf{w}_{n-1})} + \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^H}{s_n^2(\mathbf{w}_{n-1})} + \frac{4\mathbf{w}_{n-1}\mathbf{w}_{n-1}^H}{t_n^2(\mathbf{w}_{n-1})}. \tag{3.3.6}$$

Suppose that we have a hermitian matrix $\mathbf{P}_n$ that reduces $\mathbf{R}_{xx}(n)$ to a matrix Λ_n of band-width $2s+1$, *viz*,

$$\mathbf{P}_n\mathbf{R}_{xx}(n)\mathbf{P}_n^H = \Lambda_n, \qquad \mathbf{P}_n\mathbf{P}_n^H = \mathbf{I}. \tag{3.3.7}$$

We can then re-write the Hessian

$$\begin{aligned}
\nabla^2\phi_n(\mathbf{w}_{n-1}) &= \mathbf{P}_n^H\left(\frac{2\Lambda_n}{s_n(\mathbf{w}_{n-1})} + \frac{2\mathbf{I}}{t_n(\mathbf{w}_{n-1})}\right)\mathbf{P}_n + \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^H}{s_n^2} + \frac{4\mathbf{w}_{n-1}\mathbf{w}_{n-1}^H}{t_n(\mathbf{w}_{n-1})^2} \\
&= \mathbf{P}_n^H\left[\mathbf{T}_n + \frac{(\mathbf{P}_n\nabla\mathcal{F}_n)(\mathbf{P}_n\nabla\mathcal{F}_n)^H}{s_n(\mathbf{w}_{n-1})^2} + \frac{4(\mathbf{P}_n\mathbf{w}_{n-1})(\mathbf{P}_n\mathbf{w}_{n-1})^H}{t_n(\mathbf{w}_{n-1})^2}\right]\mathbf{P}_n \\
&= \mathbf{P}_n^H\mathbf{G}_n\mathbf{P}_n,
\end{aligned}$$

where

$$\mathbf{T}_n := \left(\frac{2\Lambda_n}{s_n(\mathbf{w}_{n-1})} + \frac{2\mathbf{I}}{t_n(\mathbf{w}_{n-1})}\right)$$

and

$$\mathbf{G}_n := \mathbf{T}_n + \frac{(\mathbf{P}_n \nabla \mathcal{F}_n)(\mathbf{P}_n \nabla \mathcal{F}_n)^H}{s_n(\mathbf{w}_{n-1})^2} + \frac{4(\mathbf{P}_n \mathbf{w}_{n-1})(\mathbf{P}_n \mathbf{w}_{n-1})^H}{t_n(\mathbf{w}_{n-1})^2}.$$

Since Λ_n is banded with bandwidth $2s+1$, so is $\mathbf{T}_n$. Thus by Fact 2 a linear system with the coefficient matrix $\mathbf{T}_n$ can be efficiently solved in $O(M^{2.2})$ if we choose $s = M^{3/5}$. Now, $\mathbf{G}_n$ differs from $\mathbf{T}_n$ only by a rank-two modification $(\mathbf{P}_n\mathbf{U}_n)(\mathbf{P}_n\mathbf{U}_n)^H$ where $\mathbf{U}_n$ is an $M \times 2$ matrix given by

$$\mathbf{U}_n = \left[\frac{\nabla \mathcal{F}_n}{s_n(\mathbf{w}_{n-1})};\ \frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})} \right].$$

Therefore we can use Fact 1 cited above to express $\mathbf{G}_n^{-1}$ in terms of $\mathbf{T}_n^{-1}$. The Newton direction needed in the IPLS algorithm can then be written as

$$\begin{aligned} [\nabla^2 \phi_n(\mathbf{w}_{n-1})]^{-1} \nabla \phi_n(\mathbf{w}_{n-1}) &= \mathbf{P}_n^H \mathbf{G}_n^{-1} \mathbf{P}_n \nabla \phi_n(\mathbf{w}_{n-1}) \\ &= \mathbf{P}_n^H \mathbf{T}_n^{-1} \mathbf{P}_n \nabla \phi_n(\mathbf{w}_{n-1}) + \mathbf{P}_n^H \mathbf{T}_n^{-1} \mathbf{P}_n \mathbf{U}_n \qquad (3.3.8) \\ &\quad \cdot \left(\mathbf{I} + \mathbf{U}_n^H \mathbf{P}_n^H \mathbf{T}_n^{-1} \mathbf{P}_n \mathbf{U}_n\right)^{-1} \mathbf{U}_n^H \mathbf{P}_n^H \mathbf{T}_n^{-1} \mathbf{P}_n \nabla \phi_n(\mathbf{w}_{n-1}). \end{aligned}$$

We outline an $O(M^{2.2})$ recursive procedure for updating the Newton direction (3.3.8) in Table 1. For each step in the procedure we also indicate the approximate number of arithmetic operations required. Here, by approximate we mean that the constants in the computational complexity estimate have been evaluated only for the $O(M^{2.2})$ and $O(M^2)$ terms. We see from Table 1 that the per iteration complexity for IPLS can be noticeably higher than for RLS which requires approximately $4M^2$ operations per iteration. Thus the performance improvement gained by using IPLS comes at the price of increased complexity. It should be noted, however, that the complexity estimates are based only on a crude and un-optimized implementation of the procedure of Table 1. Considering that a significant amount of the computational effort involves the calculation and application of Givens rotations (to compute the banded matrix Λ_n) a parallel implementation may be one way to achieve computational savings.

Furthermore, it should be recognized that the calculation of the IPLS estimator is complicated mainly because of the regularization term that changes in each iteration. As we have argued in Section 3.2 it is precisely this term that leads to IPLS' improved transient behaviour. In a practical implementation one could easily use the full IPLS algorithm only during transient phase and then switch to an RLS like estimator in steady state.

Table 3.1: *Recursive Computation of Newton Direction* $[\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})$

Step	Computations
Recursively update $\mathbf{p}_{xy}(n)$, $\mathbf{R}_{xx}(n)$, $s_n(\mathbf{w}_{n-1})$, $t_n(\mathbf{w}_{n-1})$, $\nabla\mathcal{F}_n(\mathbf{w}_{n-1})$ and $\nabla\phi_n(\mathbf{w}_{n-1})$ in exactly the same way as outlined in the IPLS algorithm.	$3M^2$
Maintain the decomposition of $\mathbf{R}_{xx}(n)$ into a banded system: $$\mathbf{P}_n^H\mathbf{R}_{xx}(n)\mathbf{P}_n = \Lambda_n, \qquad \text{for } n \geq 1,$$ by recursively updating $\mathbf{P}_n$ and Λ_n. Since $\mathbf{R}_{xx}(n-1)$ and $\mathbf{R}_{xx}(n)$ differ by only a symmetric rank-one matrix (*cf.* 3.3.5), it follows from Fact 3 above that we can compute the matrices $\mathbf{P}_n$, Λ_n from $\mathbf{P}_{n-1}$, Λ_{n-1} in $O(M^{2.2})$ operations on average.	$9.5M^{2.2} + 25M^2$
Use $\mathbf{P}_n$, Λ_n obtained above to update the Newton direction according to (3.3.8).	$6M^{2.2} + 4M^2$
Total	$15.5M^{2.2} + 32M^2$

3.4 Simulations

We present numerical experiments to compare the performance of the analytic center estimator $\mathbf{w}_n$ (obtained from the IPLS algorithm) with the least-squares estimator $\mathbf{w}_n^{rls}$ obtained from the RLS algorithm. Throughout we shall use the mean-squared errors in the filter,

$$\varepsilon_{ip}(n) = \|\mathbf{w}_n - \mathbf{w}_*\|^2 \quad \text{and} \quad \varepsilon_{rls}(n) = \|\mathbf{w}_n^{rls} - \mathbf{w}_*\|^2$$

as a measure of performance, where $\mathbf{w}_*$ represents the coefficients of the unknown system to be identified.

In our experiments we employ two different sources. In each of the sources the input vector satisfies $\mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-M+1}]^T$, where the sequence $\{x_n\}$ can be generated either as white Gaussian noise, or by passing white Gaussian noise first through an IIR system with transfer function (adopted from Moustakides (1997))

$$H(z) = \frac{1 + 2z^{-1} + 3z^{-2}}{(1 - 1.1314z^{-1} + 0.64z^{-2})(1 + 0.9z^{-1})}. \tag{3.4.1}$$

The unknown FIR system $\mathbf{w}_*$ is a random vector of length M (where M will be specified in

each experiment) and each entry takes on a value in the interval $[-1, +1]$. Zero mean white Gaussian noise will be added to the output of the system at specified signal-to-noise ratios.

We present three separate experiments. In Experiment 3.1 we examine the basic convergence behaviour of our algorithm (and compare to RLS). Experiment 3.2 is designed to illustrate how IPLS changes with different choices of parameters β and R. Finally, in the third experiment, we consider an abrupt change in the FIR system $\mathbf{w}_*$. In each experiment our results are obtained by averaging errors over 500 independent Monte Carlo trial.

Experiment 3.1

The order of the FIR system is $M = 20$, and source sequence length is $N = 200$. The parameters of both the RLS and IPLS algorithm are set to the nominal values listed in Table 3.2. For RLS, λ is the usual forgetting factor, and δ is a parameter that controls the amount of regularization added to the autocorrelation matrix $\mathbf{R}_{xx}(n)$. The widely used "rule of thumb" is to set $\delta \ll 0.01\sigma_x^2$, where σ_x^2 is the average signal power (Haykin, 1998).

Table 3.2: *Nominal parameter settings*

RLS	$\lambda = 1, \quad \delta = 10^{-4}$
IPLS	$\beta = 2, \quad R = 1000$

The nominal values are chosen such that both algorithms perform well in the nominal scenario of high signal-to-noise ratio (SNR=40dB), and white Gaussian signal source. Figure 3.2a confirms that in this case both RLS and IPLS efficiently identify the unknown system. Their respective identification errors ε_{ip} and ε_{rls} are almost identical. We then compare both algorithms in the case where the underlying conditions have changed, *i.e.*, the SNR drops to 10dB, and/or the source becomes correlated and the source power changes (by being generated according to (3.4.1)). Figures 3.2b, 3.2c, and 3.2d show that now, RLS has a significantly worse transient convergence behaviour while asymptotically, both RLS and IPLS are still equivalent. It should be noted that the degradation in RLS is not so much due to the source being correlated as to the fact that signal power has changed and RLS is now poorly initialized. In other words, RLS can be re-initialized to perform well in the scenario of Figure 3.2d, however, then its performance in Figure 3.2a (high SNR, white Gaussian source) suffers considerably.

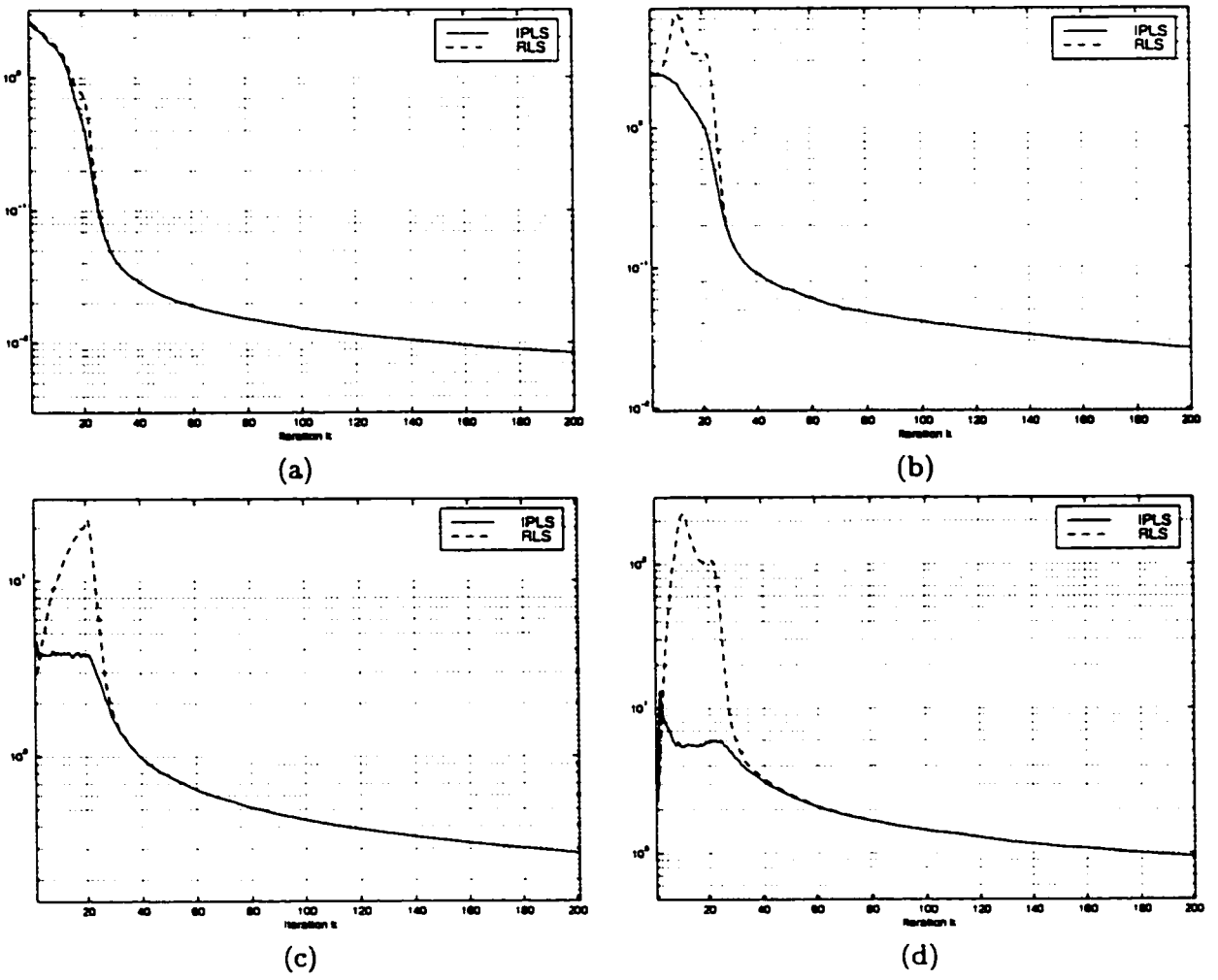


Figure 3.2: *Comparison of convergence properties of RLS and IPLS (a) SNR = 40dB, Source: White Gaussian Noise (b) SNR = 40dB, Source: Correlated (c) SNR = 10dB, Source: White Gaussian Noise (d) SNR = 10dB, Source: Correlated.*

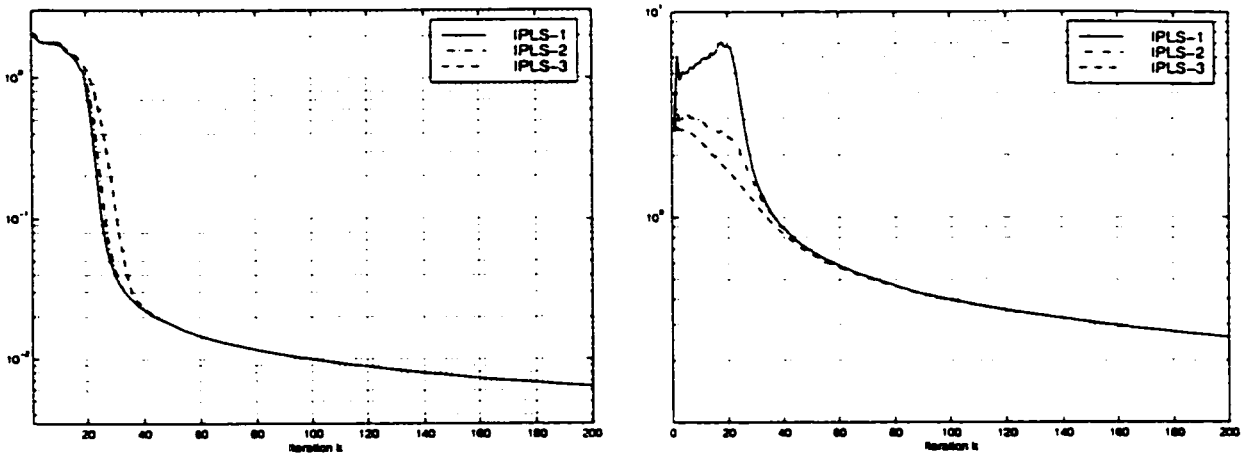
Experiment 3.2

We now examine the sensitivity of the algorithm to the initialization of parameters β and R. We use once again a size $M = 20$ filter, and source sequences of length $N = 200$. Let the three algorithms IPLS-1, IPLS-2, IPLS-3 have $R = 1000$ but three different values of β, $\beta_1 = 0.1, \beta_2 = 10$, and $\beta_3 = 100$, respectively. Recall from Section 3.2 that a smaller value of β corresponds to a smaller slack, and thus a smaller parameter α_n. Figure 3.3a illustrates that when the SNR is high, a smaller regularization term results in slightly faster convergence. This is intuitively satisfying because as the noise is small, the true filter should be located more central in the feasible region. On the other hand, when the SNR is low, the smaller $\alpha_n \mathbf{I}$ term is insufficient to contain the error during the transient stage of the algorithm as seen Figure 3.3b.

To examine the sensitivity of IPLS to the value of R, we keep β constant ($\beta = 2$, as in Experiment 3.1) and initialize the three different IPLS algorithms (IPLS-1, 2, and 3) with $R_1 = 10, R_2 = 1000$ and $R_3 = 100,000$, respectively. Note that since the elements of $\mathbf{w}_*$ are constrained within the interval $[-1, +1]$, $\|\mathbf{w}_*\| \leq \sqrt{M} \leq R$ for all three values R_i. The simulation results depicted in Figure 3.4 are best interpreted by examining the relation of R to α_n in (3.2.3). When R is small $\alpha_n \mathbf{I}$ is large, thus the conditioning of $\mathbf{R}_{xx}(n)$ is much improved by the regularization term. Thus the better performance of IPLS-3 in Figure 3.4b. When the SNR is high, $\mathbf{R}_{xx}(n)$ is better conditioned and does not require the augmentation by $\alpha_n \mathbf{I}$. It is important to note that the initialization of parameters β and R affect only the transient behaviour of IPLS. Asymptotically the effects of β and R vanish in comparison with that of the term $\|\nabla \mathcal{F}_n\|$.

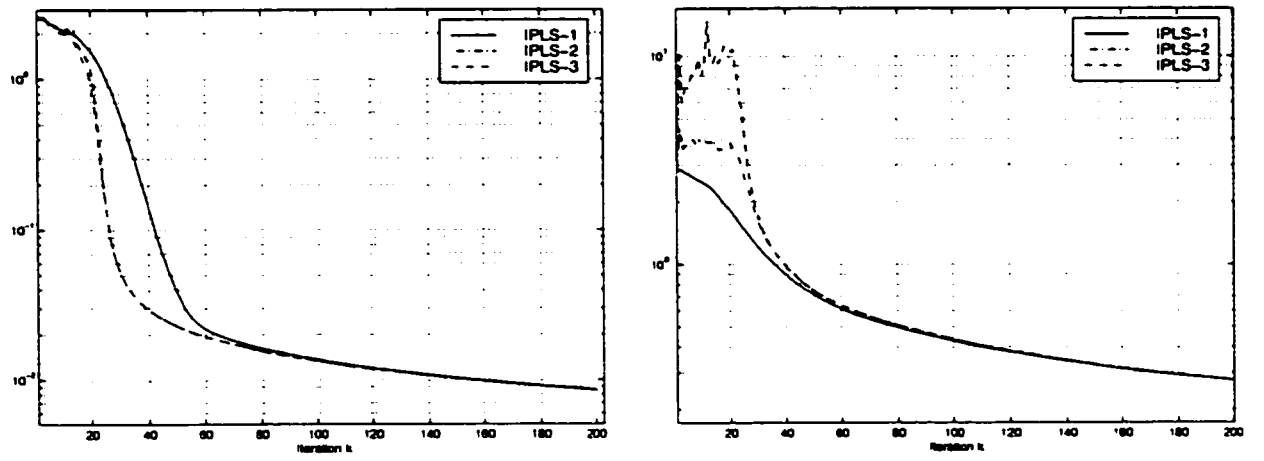
Experiment 3.3

The transient convergence of an adaptive filtering algorithm becomes even more significant when the parameter vector to be identified changes in time. In this experiment the parameter vector changes to a new set of values at iteration 100. It is well known that the standard RLS algorithm performs poorly after an abrupt change, because as the change occurs the state of $\mathbf{R}_{xx}^{-1}$ corresponds to a poor initialization of the algorithm (Moustakides, 1997). The problem is addressed using a sliding-window approach in which all data outside the current window is erased from memory. We employ the method of Liu and He (1995) to adapt RLS to a sliding-window format. Note that implementing a sliding-window IPLS is quite straight forward. One simply computes the cross-correlation vector and autocorrelation matrix over a window of data. For the sliding-window RLS (or SW-RLS) it is nchaow also necessary to reduce the forgetting factor λ to a value less than one. We use



(a) (b)

Figure 3.3: *Dependence of IPLS on β. (a) SNR = 40dB, Source: White Gaussian Noise (b) SNR = 10dB, Source: White Gaussian Noise*



(a) (b)

Figure 3.4: *Dependence of IPLS on R. (a) SNR = 40dB, Source: White Gaussian Noise (b) SNR = 10dB, Source: White Gaussian Noise*

$\lambda = 0.99$ but all other parameters remain as in Table 3.2. In the simulation of Figure 3.5 we let $M = 10$ and use a sliding-window length of 15. This window length is relatively short to improve (and emphasize) the transient responses of both algorithms tested. Noise is added at an SNR of 40dB. After the abrupt change it takes 15 iterations to "flush out" the old and incorrect data from the statistics. Thus, any convergence behaviour beyond iteration 115 is purely due to a decaying regularization. It can be seen in Figure 3.5 that the regularization becomes a limiting factor for RLS which needs about 200 iterations (after the change) to reach steady state. IPLS, on the other hand, is only limited by the length of the sliding window, and needs no further iterations to reduce any regularization bias. Due to the forgetting factor $\lambda < 1$, $\mathbf{w}_n^{rls}$ does not converge to the optimum linear filter, thus the higher steady-state error of RLS.

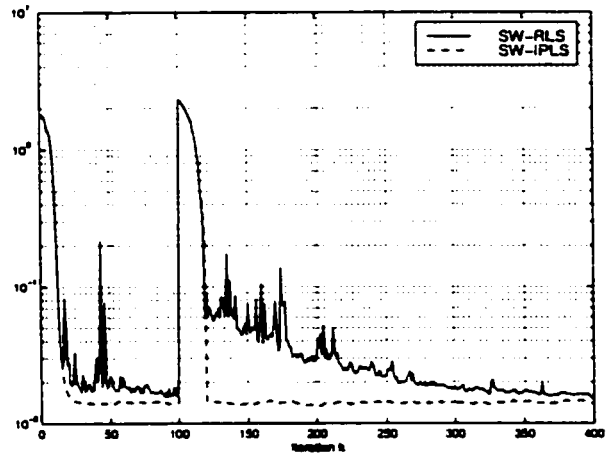


Figure 3.5: *Comparison of sliding-window versions of IPLS and RLS when channel characteristics change abruptly (at iteration 100).*

3.5 Discussion

We have presented a novel approach to adaptive linear filtering. The IPLS algorithm introduced in this chapter uses a convex feasible region with just two inequalities throughout: one that is designed to minimize the mean-square error and a norm constraint that is shown to provide regularization in the case of an ill-conditioned autocorrelation matrix. The algorithm is efficient in that it uses only one Newton step to update its estimate in each iteration. Moreover, this Newton step can be computed recursively in a way that reduces its computational complexity from $O(M^3)$ to $O(M^{2.2})$. The potential of our method has been demonstrated in several numerical experiments where the IPLS algorithm is observed

to be less sensitive to initialization and to have superior (exponentially decaying) transient behaviour. We point out that the IPLS algorithm can be readily extended to include an exponential forgetting factor or be limited to a finite data window. The recursive updating procedure can easily be modified to account for these changes.

The algorithm has of course not been completely characterized yet. Several important issues are left for later chapters: asymptotic and transient convergence are analyzed in Chapter 4 and numerical stability is addressed in Chapter 6. The reader may find it helpful, however, to see a somewhat detailed comparison of IPLS and RLS at this point. Those key properties that have been investigated in this thesis are summarized in Table 3.3. A quick glance over the table allows the general observation that: in comparison with RLS, IPLS trades off a slight increase in computational complexity for advantages in (1) transient convergence, (2) robustness to initialization, (3) numerical stability, (4) versatility with respect to additional constraints.

It should be mentioned that by taking advantage of structural properties in adaptive filtering problems, RLS can sometimes be implemented with $O(M)$ complexity (see Section 1.2.2). Whether the same structure can also be exploited for 'fast' implementations of IPLS, has not been considered in this thesis.

Table 3.3: *Direct Comparison of RLS, IPLS*

Property	RLS	IPLS
Asymptotic Convergence	$O(1/n)$	$O(1/n)$
Computational Complexity	$O(M^2)$	$O(M^{2.2})$
Transient Convergence	$O(1/n)$	$O(R^{-n})$
Robustness to Initialization	no	yes
Additional constraints	requires new algorithm	easily accommodated
Numerical Stability (in constrained case)	problems occur when λ is small (λ : forgetting factor)	stable even at small values of λ, and in cases of limited precision calculations
Sliding window implementation	can be accommodated	easily accommodated

Chapter 4

Convergence Analysis of the Interior Point Least Squares Algorithm

Generally speaking, the convergence behaviour of an estimation algorithm is influenced by two factors: the speed of the statistical averaging process and the decay rate of the initialization parameters. The former is usually dictated by the law of large numbers and therefore exhibits $O(1/n)$ type of convergence rate. The latter is dependent on how fast the estimation algorithm can phase out the effect of initialization and is therefore more algorithm specific. We shall refer to this algorithm property as its *transient behaviour*. This is consistent with other work in which the initialization of RLS has been identified as the most dominant factor governing transient convergence (Moustakides, 1997). Fast transient convergence is important for accurate estimation of time-varying parameters.

The two types of convergence behaviour can be nicely separated as follows. To isolate the statistical averaging process, we assume that the algorithm is initialized in such a way that there is no need for the phasing out of effects of initialization. On the other hand, to isolate the transient behaviour, we assume that the generated data has no statistical fluctuation so that the statistical averages are equal to the true estimated values throughout the algorithm computation. In this case, the convergence rate is entirely dictated by the transient behaviour of the estimation algorithm.

We first consider the RLS algorithm and briefly analyze its convergence behaviour in a simple example. Suppose the data is generated by the following discrete time-invariant

system in $\mathbb{R}^1$

$$y_i = w_* x_i + v_i, \quad \text{with } x_i = w_* = 1, \qquad i = 1, 2, \ldots \tag{4.0.1}$$

As discussed above, to understand the influence of the statistical averaging process on the convergence speed of the RLS algorithm, we assume RLS is perfectly initialized: the regularization term δ is equal to zero, so that the RLS estimate is given by

$$\mathbf{R}_{xx}(n)^{-1}\mathbf{p}_{xy}(n) = 1 + \frac{v_1 + v_2 + \cdots + v_n}{n}.$$

By the law of large numbers, the noise average goes to zero at the rate $O(1/n)$ (in the mean square sense). Therefore the RLS estimates converge to the correct value $w_* = 1$ at the rate $O(1/n)$. To understand the transient behaviour of the RLS algorithm, we assume that the noise is absent. In this case, $\mathbf{R}_{xx}(n) = 1$ and $\mathbf{p}_{xy}(n) = 1$ for all $n \geq 1$. Thus the RLS estimate at time n is $(\frac{\delta}{n} + 1)^{-1}$ which shows that the effect of the initialization decays at $O(1/n)$.

Now consider the analytic center based estimation method described earlier. Similar to RLS, the statistical averaging process converges at the rate of $O(1/n)$, thus giving the method an asymptotic convergence rate of $O(1/n)$. To understand its transient behaviour, we set the noise in (4.0.1) to zero. In this case, $\mathcal{F}_n(w) = (w-1)^2$ for all $n \geq 1$. In light of (3.2.4), we get

$$\tau_n = (w_{n-1}^a - 1)^2 + \beta\frac{R}{\sqrt{2}}|w_{n-1}^a - 1|.$$

By property (3.2.2) of the analytic center, we have

$$\frac{2(w_n^a - 1)}{\tau_n - (w_n^a - 1)^2} + \frac{2w_n^a}{R^2 - (w_n^a)^2} = 0, \qquad n \geq 2.$$

The above two relations imply $|w_n^a - 1| = O(R^{-1})|w_{n-1}^a - 1|$. This shows the exponential decay of the transient process when R is sufficiently large.

In the following two sections we rigorously analyze both the asymptotic and the transient convergence behaviour of the IPLS algorithm.

4.1 Asymptotic Convergence Analysis of IPLS

We now analyze the asymptotic convergence, *i.e.*, the steady state behaviour of the IPLS algorithm. The following two conditions will be required in the analysis.

Condition 1 (*Bounded Autocorrelation Matrix*) *There exist* $n_0 > 0, \sigma_1 > 0, \sigma_2 > 0$ *such that for every realization* $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$

$$\sigma_1 \mathbf{I} \leq \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \leq \sigma_2 \mathbf{I}, \qquad \forall n \geq n_0.$$

Condition 2 (*Bounded Outputs*) *There exists a fixed* ρ_y *such that for all* $n > n_0$ *there holds*

$$\frac{1}{n} \sum_{i=1}^{n} y_i^2 \leq \rho_y.$$

The left inequality in Condition 1 is known as *weak persistent excitation* condition and is considered standard in the analysis of adaptive linear filtering algorithms. The remaining inequalities stated in the conditions refer to the boundedness of inputs and outputs.

Lemma 4.1 *Suppose Conditions 1 and 2 are both satisfied. Then*

$$\mathcal{F}_n(\mathbf{w}) \leq \frac{1}{2\sigma_1} \|\nabla \mathcal{F}_n(\mathbf{w})\|^2 + \rho_y, \qquad \forall n \geq n_0.$$

Proof. Evaluating $\mathcal{F}_n(\mathbf{w})$ at the least-squares solution $\mathbf{w}_n^{ls} = \mathbf{R}_{xx}^{-1}(n)\mathbf{p}_{xy}(n)$ yields

$$\mathcal{F}_n(\mathbf{w}_n^{ls}) = \frac{1}{n} \mathbf{y}_n^H \mathbf{y}_n - \mathbf{p}_{xy}^H(n) \mathbf{R}_{xx}^{-1}(n) \mathbf{p}_{xy}(n) \leq \rho_y$$

where Condition 2 is used. By Taylor expansion and $\nabla \mathcal{F}_n(\mathbf{w}_n^{ls}) = 0$, we can write $\mathcal{F}_n(\mathbf{w})$ as:

$$\mathcal{F}_n(\mathbf{w}) = \frac{1}{2} (\mathbf{w} - \mathbf{w}_n^{ls})^H \nabla^2 \mathcal{F}_n (\mathbf{w} - \mathbf{w}_n^{ls}) + \mathcal{F}_n(\mathbf{w}_n^{ls}).$$

Taking the gradient with respect to $\mathbf{w}$ yields

$$\begin{aligned} \nabla \mathcal{F}_n(\mathbf{w}) &= (\nabla^2 \mathcal{F}_n)(\mathbf{w} - \mathbf{w}_n^{ls}) \\ \|\nabla \mathcal{F}_n(\mathbf{w})\|^2 &= (\mathbf{w} - \mathbf{w}_n^{ls})^H \left(\nabla^2 \mathcal{F}_n\right)^2 (\mathbf{w} - \mathbf{w}_n^{ls}) \end{aligned} \tag{4.1.2}$$

Then

$$\begin{aligned} \mathcal{F}_n(\mathbf{w}) &= \frac{1}{2} (\mathbf{w} - \mathbf{w}_n^{ls})^H (\nabla^2 \mathcal{F}_n)(\mathbf{w} - \mathbf{w}_n^{ls}) + \mathcal{F}_n(\mathbf{w}_n^{ls}) \\ &\leq \frac{1}{2\sigma_1} (\mathbf{w} - \mathbf{w}_n^{ls})^H \left(\nabla^2 \mathcal{F}_n\right)^2 (\mathbf{w} - \mathbf{w}_n^{ls}) + \rho_y \\ &= \frac{1}{2\sigma_1} \|\nabla \mathcal{F}_n(\mathbf{w})\|^2 + \rho_y, \end{aligned}$$

where the inequality uses Condition 1, and the equality follows by substituting (4.1.2). □

Lemma 4.1 is a consequence only of Conditions 1 and 2 and is thus true for any estimate $\mathbf{w}$ independent of the algorithm. The next lemma establishes some key preliminary results that are required in the proof of Lemma 4.3.

Lemma 4.2 *Let the quantities τ_n, $s_n(\mathbf{w})$ and $\mathbf{w}_n$ be generated according to the IPLS algorithm:*

$$\begin{aligned} \tau_n &:= \mathcal{F}_n(\mathbf{w}_{n-1}) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|, \\ s_n(\mathbf{w}) &:= \tau_n - \mathcal{F}_n(\mathbf{w}), \\ \mathbf{w}_n &:= \mathbf{w}_{n-1} - [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1}). \end{aligned}$$

Assume that $\tau_n \le t$ for some constant $t > \rho_y$. Then

1. $\|\mathbf{w}_n\| = O(1)$ *and* $\|\mathbf{w}_{n-1}\| = O(1)$.
2. *For large enough β and R, $\mathbf{w}_{n-1}$ is an approximate analytic center of Ω_n in the sense*
$$\left\langle \nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\right\rangle^{\frac{1}{2}} \le \frac{1}{\beta} + O(R^{-1}) \le 0.9.$$
3. $s_n(\mathbf{w}_n) = O\left(s_n(\mathbf{w}_{n-1})\right)$ *and* $s_n(\mathbf{w}_{n-1}) = O\left(s_n(\mathbf{w}_n)\right)$.

Here and throughout the big "O" notation refers to an upper bound with constant factor independent of R, β, but may depend on t.

Proof. We treat each claim of the lemma in sequence.

Claim 1

By the update rule for τ_n, we have $\mathcal{F}_n(\mathbf{w}_{n-1})$ is bounded by τ_n. Also, $\mathcal{F}_n(\mathbf{w}_n) < \tau_n$ since $\mathbf{w}_n$ must have a lower potential $\phi_n(\mathbf{w}_n)$ than $\mathbf{w}_{n-1}$ after the Newton update. Using the assumption $\tau_n \le t$, we then have that $\mathcal{F}_n(\mathbf{w}_{n-1}) \le t$ and $\mathcal{F}_n(\mathbf{w}_n) \le t$. On the other hand we can write as in Lemma 4.1,

$$\mathcal{F}_n(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}_n^{ls})^H(\nabla^2\mathcal{F}_n)(\mathbf{w} - \mathbf{w}_n^{ls}) + \mathcal{F}_n(\mathbf{w}_n^{ls}) \ge \frac{\sigma_1}{2}\|\mathbf{w} - \mathbf{w}_n^{ls}\|^2.$$

Using the bounds for $\mathcal{F}_n(\mathbf{w}_n)$ and $\mathcal{F}_n(\mathbf{w}_{n-1})$ argued above we obtain

$$\frac{\sigma_1}{2}\|\mathbf{w} - \mathbf{w}_n^{ls}\|^2 \leq t, \qquad \text{for } \mathbf{w} = \mathbf{w}_n,\ \mathbf{w} = \mathbf{w}_{n-1}.$$

This further implies

$$\|\mathbf{w}_{n-1}\| \leq \sqrt{2t/\sigma_1} + \|\mathbf{w}_n^{ls}\| \quad \text{and} \quad \|\mathbf{w}_n\| \leq \sqrt{2t/\sigma_1} + \|\mathbf{w}_n^{ls}\|.$$

By Conditions 1 and 2, the least-squares solution $\mathbf{w}_n^{ls}$ is clearly bounded by a constant (dependent on ρ_y and σ_1 only), the above immediately implies the claim.

Claim 2

Denote by LHS the quantity we want to bound in this proof.

$$\begin{aligned}
\text{LHS} &= \langle \nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\rangle^{\frac{1}{2}} \\
&= \left\langle \frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})} - \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}, [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\left(\frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})} - \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}\right)\right\rangle^{\frac{1}{2}} \\
&\leq \left\langle \frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})}, [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})}\right\rangle^{\frac{1}{2}} \\
&\quad + \left\langle \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}, [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}\right\rangle^{\frac{1}{2}} \qquad (4.1.3)
\end{aligned}$$

where $s_n(\mathbf{w}) := \tau_n - \mathcal{F}_n(\mathbf{w})$ and $t_n(\mathbf{w}) := R^2 - \|\mathbf{w}\|^2$. Here the second step follows directly from the definition of the gradient of the potential function $\phi_n(\mathbf{w})$, and the last step is obtained by using the triangular inequality. To continue, we first bound the Hessian of $\phi_n(\mathbf{w})$ which is given by

$$\nabla^2\phi_n(\mathbf{w}) = \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^H}{s_n^2(\mathbf{w})} + \frac{\nabla^2\mathcal{F}_n}{s_n(\mathbf{w})} + \frac{4\mathbf{w}\mathbf{w}^H}{t_n(\mathbf{w})^2} + \frac{2\mathbf{I}}{t_n(\mathbf{w})}. \qquad (4.1.4)$$

Since all terms in (4.1.4) are positive semidefinite, the Hessian of $\phi_n(\mathbf{w}_{n-1})$ can be bounded by the last term,

$$\nabla^2\phi_n(\mathbf{w}_{n-1}) \geq \frac{2\mathbf{I}}{t_n(\mathbf{w})} = \frac{2\mathbf{I}}{R^2 - \|\mathbf{w}_{n-1}\|^2} \geq \frac{2}{R^2}\mathbf{I},$$

or equivalently,

$$[\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1} \leq \frac{R^2}{2}\mathbf{I}. \qquad (4.1.5)$$

Using (4.1.5) in both terms of (4.1.3), we then obtain

$$\begin{aligned} \text{LHS} &\leq \frac{R}{\sqrt{2}}\left[\left\langle \frac{2\|\mathbf{w}_{n-1}\|}{t_n(\mathbf{w}_{n-1})}, \frac{2\|\mathbf{w}_{n-1}\|}{t_n(\mathbf{w}_{n-1})}\right\rangle^{\frac{1}{2}} + \left\langle \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}, \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}\right\rangle^{\frac{1}{2}}\right] \\ &= \frac{R}{\sqrt{2}}\left[\frac{2\|\mathbf{w}_{n-1}\|}{t_n(\mathbf{w}_{n-1})} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})}\right] \end{aligned}$$

But $s_n(\mathbf{w}_{n-1}) = \tau_n - \mathcal{F}_n(\mathbf{w}_{n-1}) = \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1}\|$ as stipulated by the updating rule for τ_n. Therefore

$$\text{LHS} \leq \frac{R}{\sqrt{2}}\left[\frac{2\|\mathbf{w}_{n-1}\|}{R^2 - \|\mathbf{w}_{n-1}\|^2} + \frac{1}{\beta\frac{R}{\sqrt{2}}}\right] = \frac{1}{\beta} + O(R^{-1})$$

where in the final step we made use of the first result in the lemma.

Claim 3

By selecting R and β sufficiently large, Claim 2 implies

$$\langle\nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\rangle^{\frac{1}{2}} < 0.9,$$

indicating that $\mathbf{w}_{n-1}$ is an approximate minimizer of $\phi_n(\mathbf{w})$ and an *approximate analytic center* of Ω_n. We can therefore apply a result by Nesterov and Nemirovskii (1994) that says

$$0 \leq \phi_n(\mathbf{w}_{n-1}) - \phi_n(\mathbf{w}_n) \leq c, \qquad \text{for some constant } c. \tag{4.1.6}$$

This result allows us to bound $s_n(\mathbf{w}_n)$ and $s_n(\mathbf{w}_{n-1})$ in terms of each other. Consider first the right inequality in (4.1.6). Substituting the definition of $\phi_n(\mathbf{w})$ we obtain

$$-\log\left(R^2 - \|\mathbf{w}_{n-1}\|^2\right) - \log s_n(\mathbf{w}_{n-1}) + \log\left(R^2 - \|\mathbf{w}_n\|^2\right) + \log s_n(\mathbf{w}_n) \leq c$$

or

$$\frac{1}{R^2 - \|\mathbf{w}_{n-1}\|^2} \cdot \frac{1}{s_n(\mathbf{w}_{n-1})} \cdot \left(R^2 - \|\mathbf{w}_n\|^2\right) \cdot (s_n(\mathbf{w}_n)) \leq e^c.$$

Solving for $s_n(\mathbf{w}_n)$ yields

$$\begin{aligned} s_n(\mathbf{w}_n) &\leq \frac{R^2 - \|\mathbf{w}_{n-1}\|^2}{R^2 - \|\mathbf{w}_n\|^2} \cdot e^c \cdot s_n(\mathbf{w}_{n-1}) \\ &\leq \frac{R^2}{R^2 - t^2} \cdot e^c \cdot s_n(\mathbf{w}_{n-1}) \\ &= O(s_n(\mathbf{w}_{n-1})). \end{aligned}$$

A similar argument using the left inequality in (4.1.6) yields

$$s_n(\mathbf{w}_{n-1}) \leq \frac{R^2 - \|\mathbf{w}_n\|^2}{R^2 - \|\mathbf{w}_{n-1}\|^2} \cdot s_n(\mathbf{w}_n) = O(s_n(\mathbf{w}_n)).$$

□

One important consequence of Claim 2 of Lemma 4.2 is that if β and R are chosen appropriately then $\mathbf{w}_{n-1}$ remains an analytic center of Ω_n. As a consequence the Newton step will quadratically reduce the error in the criterion $\langle \nabla\phi_n(\mathbf{w}), [\nabla^2\phi_n(\mathbf{w})]^{-1}\nabla\phi_n(\mathbf{w})\rangle$, which is instrumental in the proof of Lemma 4.3.

Lemma 4.3 *Assume the same conditions as Lemma 4.2 for some fixed n. Let $\mathbf{w}_n$ and $s_n(\mathbf{w})$ be obtained by the IPLS algorithm, then we have the following:*

1. *There holds*

$$\frac{\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} = \frac{\sqrt{2}}{\beta R} = O(R^{-1}), \qquad \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|}{s_n(\mathbf{w}_n)} = O(R^{-2}). \tag{4.1.7}$$

2. *If R and n are large,*

$$\tau_{n+1} := \mathcal{F}_{n+1}(\mathbf{w}_n) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_{n+1}(\mathbf{w})\| = O\left(R^{-1} + \frac{1}{n}\right) + \rho_y \leq t.$$

Proof. By assumption, all the conclusions of Lemma 4.2 hold. Therefore

$$\|\mathbf{w}_n\| = O(1) \quad \text{and} \quad \|\mathbf{w}_{n-1}\| = O(1) \tag{4.1.8}$$

and if we choose R and β sufficiently large

$$\langle \nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\rangle^{\frac{1}{2}} < 0.9. \tag{4.1.9}$$

The first part of Claim 1 of Lemma 4.3 follows directly from the updating rule of τ_n. To show the second part of the first claim of Lemma 4.3, we need to establish some bounds on the Hessian of ϕ_n. Recall

$$\nabla^2\phi_n(\mathbf{w}) = \frac{2\mathbf{I}}{t_n(\mathbf{w})} + \frac{2\mathbf{w}\mathbf{w}^H}{t_n(\mathbf{w})^2} + \frac{\nabla^2\mathcal{F}_n}{s_n(\mathbf{w})} + \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^H}{s_n^2(\mathbf{w})}.$$

Using Condition 1 and $t_n(\mathbf{w}_n) = R^2 - \|\mathbf{w}_n\|^2 = O(R^2)$, we obtain the following bounds

$$\nabla^2\phi_n(\mathbf{w}_{n-1}) \geq \frac{\sigma_1}{s_n(\mathbf{w}_{n-1})}\mathbf{I},$$

and

$$\begin{aligned}\nabla^2\phi_n(\mathbf{w}_n) &\leq \left(O(R^{-2}) + O(R^{-4}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)\mathbf{I} \\ &\leq \left(O(R^{-2}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)\mathbf{I}.\end{aligned}$$

By (4.1.9), $\mathbf{w}_{n-1}$ is an approximate analytic center, so there holds quadratic convergence (Nesterov and Nemirovskii, 1994):

$$\langle\nabla\phi_n(\mathbf{w}_n), [\nabla^2\phi_n(\mathbf{w}_n)]^{-1}\nabla\phi_n(\mathbf{w}_n)\rangle \leq \langle\nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\rangle^2.$$

Denote the left and right sides of the above inequality by LHS and RHS respectively. Then

$$\begin{aligned}\text{LHS} &\geq \|\nabla\phi_n(\mathbf{w}_n)\|^2 \cdot \left(O(R^{-2}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)^{-1} \\ \text{RHS} &\leq \|\nabla\phi_n(\mathbf{w}_{n-1})\|^4 \cdot \frac{s_n^2(\mathbf{w}_{n-1})}{\sigma_1^2}.\end{aligned}$$

Combining the above we obtain

$$\|\nabla\phi_n(\mathbf{w}_n)\|^2 \leq \frac{s_n^2(\mathbf{w}_{n-1})}{\sigma^2}\left(O(R^{-2} + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)\|\nabla\phi_n(\mathbf{w}_{n-1})\|^4.$$

Notice that $s_n(\mathbf{w}) \leq \tau_n \leq t = O(1)$ and that the quotient $\frac{s_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_n)}$ is $O(1)$ (due to Lemma 4.2 (Claim 3)). The above inequality further simplifies to

$$\|\nabla\phi_n(\mathbf{w}_n)\|^2 \leq \left(O(1) + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)\|\nabla\phi_n(\mathbf{w}_{n-1})\|^4. \tag{4.1.10}$$

Note that from (4.1.8) and

$$\frac{\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} = \frac{\sqrt{2}}{\beta R} = O(R^{-1}), \qquad \nabla\phi_n(\mathbf{w}_{n-1}) = \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})} + \frac{2\mathbf{w}_{n-1}}{R^2 - \|\mathbf{w}_{n-1}\|^2},$$

we can immediately deduce

$$\|\nabla\phi_n(\mathbf{w}_{n-1})\| = O(R^{-1}).$$

Using this result and the definition of $\nabla\phi_n$ in (4.1.10) we obtain

$$\left\|\frac{\nabla\mathcal{F}_n(\mathbf{w}_n)}{s_n(\mathbf{w}_n)} + \frac{2\mathbf{w}_n}{R^2 - \|\mathbf{w}_n\|^2}\right\|^2 \leq \left(O(1) + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)O(R^{-4}).$$

Applying the inequality $\|a+b\|^2 \geq \frac{1}{2}\|a\|^2 - \|b\|^2$ and noting $\|2\mathbf{w}_n\|^2/(R^2 - \|\mathbf{w}_n\|^2)^2 = O(R^{-4})$, we further obtain

$$\frac{1}{2}\frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)} - \frac{\|2\mathbf{w}_n\|^2}{(R^2 - \|\mathbf{w}_n\|^2)^2} \leq \left(O(1) + \frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right) O(R^{-4})$$

Solving for the $\frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}$ term yields

$$\frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)} = O(R^{-4}) \tag{4.1.11}$$

which immediately verifies the second part of the first claim in the lemma.

It remains to show the second claim that the updated threshold τ_{n+1} will still be bounded by t. We first note that the update formula for τ_{n+1} is

$$\begin{aligned}
\tau_{n+1} &= \mathcal{F}_{n+1}(\mathbf{w}_n) + \beta\frac{R}{\sqrt{2}}\|\nabla \mathcal{F}_{n+1}(\mathbf{w}_n)\| \\
&= \mathcal{F}_{n+1}(\mathbf{w}_n) + \beta\frac{R}{\sqrt{2}}s_n(\mathbf{w}_n)O(R^{-2}) \\
&= \mathcal{F}_{n+1}(\mathbf{w}_n) + \frac{\beta}{\sqrt{2}}O(R^{-1})O(1)
\end{aligned}$$

where the second line follows from (4.1.11), and the third line holds because $s_n(\mathbf{w}) \leq \tau_n \leq t = O(1)$. We next bound $\mathcal{F}_{n+1}(\mathbf{w}_n)$,

$$\begin{aligned}
\mathcal{F}_{n+1}(\mathbf{w}_n) &= \frac{1}{n+1}\sum_{i=1}^{n+1}(y_i - \mathbf{x}_i^H\mathbf{w}_n)^2 \\
&= \frac{n}{n+1}\mathcal{F}(\mathbf{w}_n) + \frac{1}{n+1}(y_{n+1} - \mathbf{x}_{n+1}^H\mathbf{w}_n)^2 \\
&= \frac{n}{n+1}\mathcal{F}(\mathbf{w}_n) + O\left(\frac{1}{n}\right)
\end{aligned}$$

where the last step follows from using Conditions 1 and 2 and the fact $\|\mathbf{w}_n\| = O(1)$. But from Lemma 4.1 we have

$$\mathcal{F}_n(\mathbf{w}_n) \leq \frac{1}{2\sigma}\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2 + \rho_y \leq \frac{1}{2\sigma}O(R^{-4})s_n^2(\mathbf{w}_n) + \rho_y = O(R^{-4}) + \rho_y.$$

Combining the last two results and gives

$$\begin{aligned}
\mathcal{F}_{n+1}(\mathbf{w}_n) &\leq \frac{n}{n+1}\left(O(R^{-4}) + \rho_y\right) + O\left(\frac{1}{n+1}\right) \\
&= O\left(R^{-4} + \frac{1}{n}\right) + \rho_y.
\end{aligned}$$

And finally, substituting back into our update formula we obtain

$$\tau_{n+1} = O(R^{-1}) + O\left(R^{-4} + \frac{1}{n}\right) + \rho_y \leq t, \qquad n,\ R \text{ large.}$$

□

The main result here is the first claim, which states that by taking one Newton step in the IPLS algorithm, the size of the quantity $\|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\|/s_n(\mathbf{w}_{n-1})$ reduces from $O(R^{-1})$ to $O(R^{-2})$. This will later help to prove our main convergence result in Theorem 4.1. We point out that the estimate (4.1.7) holds only under the assumption $\tau_n \leq t$ (as assumed in Lemma 4.2). However, Claim 2 of Lemma 4.3 ensures that $\tau_{n+1} \leq t$, thus Lemmas 4.2 and 4.3 can be applied again at iteration $n+1$. This fact will be used in the proof of our main convergence result below.

Theorem 4.1 *Let the sequence of estimates* $\{\mathbf{w}_n, n = 1, 2, 3, \dots\}$ *be generated by the IPLS algorithm. Then*

$$\|\nabla \mathcal{F}_n(\mathbf{w}_n)\| = 2\|\mathbf{R}_{xx}(n)\mathbf{w}_n - \mathbf{p}_{xy}(n)\| = O\left(1/n\right).$$

Consequently, $\|\mathbf{w}_n - \mathbf{R}_{xx}(n)^{-1}\mathbf{p}_{xy}(n)\| = O(1/n)$.

Proof. Fix some $t > \rho_y$ so that $\tau_n \leq t$ for some large n_1. Suppose β, R are chosen large enough. Then Lemma 4.3 holds for this n_1 so that $\tau_{n_1+1} \leq t$. But this implies Lemma 4.3 should hold for $n_1 + 1$. In general, a simple inductive argument shows Lemma 4.3 should hold for all $n \geq n_1$. Thus, we have

$$\frac{\|\nabla \mathcal{F}_n(\mathbf{w}_{n-1}\|}{s_n(\mathbf{w}_{n-1})} = \frac{\sqrt{2}}{\beta R} \quad \text{and} \quad \frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|}{s_n(\mathbf{w}_n)} = O(R^{-2}), \quad \forall n \geq n_1.$$

Therefore,

$$\begin{aligned} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| &= R^{-1}\, s_n(\mathbf{w}_n)\, O(R^{-1}) \\ &= \frac{\|\nabla \mathcal{F}(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} \frac{\beta}{\sqrt{2}}\, s_n(\mathbf{w}_n)\, O(R^{-1}) \\ &= \|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\| O(R^{-1}) \end{aligned} \tag{4.1.12}$$

where the last step is due to Lemma 4.2 (Claim 3). By the recursive update of $\nabla \mathcal{F}_n(\mathbf{w})$, we have

$$\begin{aligned} \|\nabla \mathcal{F}_{n+1}(\mathbf{w}_n)\| &= \left\| \frac{n}{n+1} \nabla \mathcal{F}_n(\mathbf{w}_n) + \frac{2(\mathbf{w}_n^H \mathbf{x}_{n+1} - y_{n+1})}{n+1} \mathbf{x}_{n+1} \right\| \\ &\leq \frac{n}{n+1} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| + O\left(\frac{1}{n}\right). \end{aligned}$$

Using (4.1.12) in the equation above we can then write

$$\|\nabla \mathcal{F}_{n+1}(\mathbf{w}_n)\| \leq \|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\| O(R^{-1}) + O\left(\frac{1}{n}\right).$$

This implies that $\|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\| = O\left(\frac{1}{n}\right)$ and due to (4.1.12)

$$\|\nabla \mathcal{F}_n(\mathbf{w}_n)\| = O\left(\frac{1}{n}\right).$$

The remaining claim follows due to Condition 1. □

Theorem 4.1 establishes the asymptotic convergence of the IPLS algorithm. This rate of convergence matches that of the RLS algorithm which computes $\mathbf{w}_n^{rls} := (\frac{\delta}{n}\mathbf{I}+\mathbf{R}_{xx}(n))^{-1}\mathbf{p}_{xy}(n)$ at each iteration, for some fixed δ. This is because $\mathbf{R}_{xx}(n)$ and $\mathbf{p}_{xy}(n)$ converge to their true values at the rate of $O(1/n)$. Note that the convergence of $\mathbf{R}_{xx}(n)$ and $\mathbf{p}_{xy}(n)$ together with the result of Theorem 1 also imply that the IPLS estimate $\mathbf{w}_n$ is asymptotically unbiased, in the sense that its expected value converges to the Wiener solution.

4.2 Transient Convergence Analysis of IPLS

Note that in the example at the beginning of the chapter we have shown that the actual analytical center converges exponentially to the true filter in the noise free case. In the following we extend this argument to show the same convergence result for the estimator obtained by the IPLS algorithm of Chapter 3. In addition to the bounded autocorrelation condition (Condition 1 in Section 4.1) we now require

Condition 3 (*No Noise*) *The system is free from measurement noise,* i.e.,

$$y_i = \mathbf{x}_i^H \mathbf{w}_*, \qquad i = 1, 2, \ldots .$$

Condition 3 is included to allow us to isolate the estimation bias and therefore the transient convergence behaviour of any estimator. In the noise free case the least-squares solution is always given by the true filter $\mathbf{w}_*$, since $\mathcal{F}_n(\mathbf{w}_*) = 0$. Thus any residual error can be caused only by a non-zero regularization term. Note that given the basic assumption that $\|\mathbf{w}_*\|$ is bounded, the combination of Conditions 1 and 3 implies that Condition 2 also holds.

To facilitate the proof of our transient convergence result (given in Theorem 4.2 below) we first show some preliminary results summarized in three more lemmas. These lemmas represent the corresponding results to Lemmas 4.1, 4.2 and 4.3 above, for the noise-free case.

Lemma 4.4 *Suppose Conditions 1 and 3 are satisfied. Then*

$$\mathcal{F}_n(\mathbf{w}) \leq \frac{1}{4\sigma_1}\|\nabla\mathcal{F}_n(\mathbf{w})\|^2, \qquad \forall n \geq n_0,$$

and

$$\|\mathbf{w} - \mathbf{w}_*\|^2 \leq \frac{1}{4\sigma_1^2}\|\nabla\mathcal{F}_n(\mathbf{w})\|^2, \qquad \forall n \geq n_0.$$

Proof. In the noise free case we may simplify the expression of the mean-squared error as follows

$$\begin{aligned}
\mathcal{F}_n(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \mathbf{x}_i^H\mathbf{w}\right|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{x}_i^H(\mathbf{w}_* - \mathbf{w})\right]^2 \\
&= (\mathbf{w} - \mathbf{w}_*)^H\mathbf{R}_{xx}(n)(\mathbf{w} - \mathbf{w}_*).
\end{aligned}$$

Taking the gradient with respect to w yields

$$\begin{aligned}
\nabla\mathcal{F}_n(\mathbf{w}) &= 2\mathbf{R}_{xx}(n)(\mathbf{w} - \mathbf{w}_*) \\
\|\nabla\mathcal{F}_n(\mathbf{w})\|^2 &= (\mathbf{w} - \mathbf{w}_*)^H\left(4\mathbf{R}_{xx}^2(n)\right)(\mathbf{w} - \mathbf{w}_*) \qquad (4.2.13)
\end{aligned}$$

Then

$$\begin{aligned}
\mathcal{F}_n(\mathbf{w}) &= (\mathbf{w} - \mathbf{w}_*)^H\mathbf{R}_{xx}(n)(\mathbf{w} - \mathbf{w}_*) \\
&\leq (\mathbf{w} - \mathbf{w}_*)^H\frac{1}{\sigma_1}\mathbf{R}_{xx}^2(n)(\mathbf{w} - \mathbf{w}_*) \\
&= \frac{1}{4\sigma_1}\|\nabla\mathcal{F}_n(\mathbf{w})\|^2,
\end{aligned}$$

where the inequality uses Condition 1, and the equality follows by substituting (4.2.13). Similarly, applying the condition $\mathbf{R}_{xx}(n) \geq \sigma_1\mathbf{I}$ twice to (4.1.2) yields

$$\|\mathbf{w} - \mathbf{w}_*\|^2 \leq \frac{1}{4\sigma_1^2}\|\nabla\mathcal{F}_n(\mathbf{w})\|^2.$$

□

Lemma 4.4 is a consequence only of Conditions 1 and 3 and is thus true for any estimate $\mathbf{w}$ independent of the algorithm. The next lemma establishes some key preliminary results that are required in the proof of Lemma 4.6.

Lemma 4.5 *Let the quantities τ_n, $s_n(\mathbf{w})$ and $\mathbf{w}_n$ be generated according to the IPLS algorithm:*

$$\begin{aligned} \tau_n &:= \mathcal{F}_n(\mathbf{w}_{n-1}) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|, \\ s_n(\mathbf{w}) &:= \tau_n - \mathcal{F}_n(\mathbf{w}), \\ \mathbf{w}_n &:= \mathbf{w}_{n-1} - [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1}). \end{aligned}$$

Assume that $\tau_n \leq t$ for some constant t. Then

1. $\|\mathbf{w}_n\| = O(1)$ *and* $\|\mathbf{w}_{n-1}\| = O(1)$.
2. *For large enough β and R, $\mathbf{w}_{n-1}$ is an approximate analytic center of Ω_n in the sense*

$$\left\langle \nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\right\rangle^{\frac{1}{2}} \leq \frac{1}{\beta} + O(R^{-1}) \leq 0.9.$$

3. $s_n(\mathbf{w}_n) = O\left(s_n(\mathbf{w}_{n-1})\right)$ *and* $s_n(\mathbf{w}_{n-1}) = O\left(s_n(\mathbf{w}_n)\right)$.

Here and throughout the big "O" notation refers to an upper bound with a constant factor which is independent of R, β, but may depend on t.

Proof. We treat each claim of the lemma in sequence.

Claim 1

By the update rule for τ_n, we have $\mathcal{F}_n(\mathbf{w}_{n-1})$ is bounded by τ_n. Also, $\mathcal{F}_n(\mathbf{w}_n) < \tau_n$ since $\mathbf{w}_n$ must remain feasible with respect to Ω_n. Using the assumption $\tau_n \leq t$, we then have that $\mathcal{F}_n(\mathbf{w}_{n-1}) \leq t$ and $\mathcal{F}_n(\mathbf{w}_n) \leq t$. On the other hand we can write as in Lemma 4.4,

$$\mathcal{F}_n(\mathbf{w}) = (\mathbf{w} - \mathbf{w}_*)^H\mathbf{R}_{xx}(n)(\mathbf{w} - \mathbf{w}_*) \geq \sigma_1\|\mathbf{w} - \mathbf{w}_*\|^2.$$

Using the bounds for $\mathcal{F}_n(\mathbf{w}_n)$ and $\mathcal{F}_n(\mathbf{w}_{n-1})$ argued above we obtain

$$\|\mathbf{w} - \mathbf{w}_*\|^2 \leq \frac{1}{\sigma_1}\mathcal{F}_n(\mathbf{w}) \leq \frac{1}{\sigma_1}t, \qquad \text{for } \mathbf{w} = \mathbf{w}_n,\ \mathbf{w} = \mathbf{w}_{n-1}.$$

This further implies

$$\|\mathbf{w}_{n-1}\| \leq \sqrt{t/\sigma_1} + \|\mathbf{w}_*\| \quad \text{and} \quad \|\mathbf{w}_n\| \leq \sqrt{t/\sigma_1} + \|\mathbf{w}_*\|.$$

Assuming boundedness of the unknown system $\mathbf{w}_*$ by a constant, the above immediately implies the claim.

Claim 2

Denote by LHS the quantity we want to bound in this proof.

$$\begin{aligned}\text{LHS} &= \langle \nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\rangle^{\frac{1}{2}} \\ &= \left\langle \frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})} + \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}, [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\left(\frac{2\mathbf{w}_{n-1}}{t_n(\mathbf{w}_{n-1})} + \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}\right)\right\rangle^{\frac{1}{2}}\end{aligned}$$

where $s_n(\mathbf{w}) := \tau_n - \mathcal{F}_n(\mathbf{w})$ and $t_n(\mathbf{w}) := R^2 - \|\mathbf{w}\|^2$. The second step follows directly from the definition of the gradient of the potential function $\phi_n(\mathbf{w})$. To continue, we first bound the Hessian of $\phi_n(\mathbf{w})$ which is given by

$$\nabla^2\phi_n(\mathbf{w}) = \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^H}{s_n^2(\mathbf{w})} + \frac{\nabla^2\mathcal{F}_n}{s_n(\mathbf{w})} + \frac{4\mathbf{w}\mathbf{w}^H}{t_n(\mathbf{w})^2} + \frac{2\mathbf{I}}{t_n(\mathbf{w})}.$$

Since all terms in the above summation are positive semi-definite, the Hessian of $\phi_n(\mathbf{w}_{n-1})$ can be bounded by the last term,

$$\nabla^2\phi_n(\mathbf{w}_{n-1}) \geq \frac{2\mathbf{I}}{t_n(\mathbf{w})} = \frac{2\mathbf{I}}{R^2 - \|\mathbf{w}_{n-1}\|^2} \geq \frac{2}{R^2}\mathbf{I},$$

or equivalently,

$$[\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1} \leq \frac{R^2}{2}\mathbf{I}.$$

Extracting the Hessian term and subsequently applying the triangular inequality LHS can be further bounded by

$$\begin{aligned}\text{LHS} &\leq \frac{R}{\sqrt{2}}\left[\left\langle \frac{2\|\mathbf{w}_{n-1}\|}{t_n(\mathbf{w}_{n-1})}, \frac{2\|\mathbf{w}_{n-1}\|}{t_n(\mathbf{w}_{n-1})}\right\rangle^{\frac{1}{2}} + \left\langle \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}, \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})}\right\rangle^{\frac{1}{2}}\right] \\ &= \frac{R}{\sqrt{2}}\left[\frac{2\|\mathbf{w}_{n-1}\|}{t_n(\mathbf{w}_{n-1})} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})}\right]\end{aligned}$$

But $s_n(\mathbf{w}_{n-1}) = \tau_n - \mathcal{F}_n(\mathbf{w}_{n-1}) = \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|$ as stipulated by the updating rule for τ_n. Therefore

$$\text{LHS} \leq \frac{R}{\sqrt{2}}\left[\frac{2\|\mathbf{w}_{n-1}\|}{R^2 - \|\mathbf{w}_{n-1}\|^2} + \frac{1}{\beta\frac{R}{\sqrt{2}}}\right] = \frac{1}{\beta} + O(R^{-1})$$

where in the final step we made use of the first result in the lemma.

Claim 3

By selecting R and β sufficiently large, Claim 2 implies

$$\left\langle \nabla\phi_n(\mathbf{w}_{n-1}), [\nabla^2\phi_n(\mathbf{w}_{n-1})]^{-1}\nabla\phi_n(\mathbf{w}_{n-1})\right\rangle^{\frac{1}{2}} < 0.9,$$

indicating that $\mathbf{w}_{n-1}$ is an approximate minimizer of $\phi_n(\mathbf{w})$ and an *approximate analytic center* of Ω_n. We can therefore apply a result by Nesterov and Nemirovskii (1994) that says

$$0 \leq \phi_n(\mathbf{w}_{n-1}) - \phi_n(\mathbf{w}_n) \leq c, \qquad \text{for some constant } c. \tag{4.2.14}$$

This result allows us to bound $s_n(\mathbf{w}_n)$ and $s_n(\mathbf{w}_{n-1})$ in terms of each other. Consider first the right inequality in (4.2.14). Substituting the definition of $\phi_n(\mathbf{w})$ we obtain

$$-\log\left(R^2 - \|\mathbf{w}_{n-1}\|^2\right) - \log s_n(\mathbf{w}_{n-1}) + \log\left(R^2 - \|\mathbf{w}_n\|^2\right) + \log s_n(\mathbf{w}_n) \leq c$$

or

$$\frac{1}{R^2 - \|\mathbf{w}_{n-1}\|^2} \cdot \frac{1}{s_n(\mathbf{w}_{n-1})} \cdot \left(R^2 - \|\mathbf{w}_n\|^2\right) \cdot \left(s_n(\mathbf{w}_n)\right) \leq e^c.$$

Solving for $s_n(\mathbf{w}_n)$ yields

$$\begin{aligned} s_n(\mathbf{w}_n) &\leq \frac{R^2 - \|\mathbf{w}_{n-1}\|^2}{R^2 - \|\mathbf{w}_n\|^2} \cdot e^c \cdot s_n(\mathbf{w}_{n-1}) \\ &\leq \frac{R^2}{R^2 - t^2} \cdot e^c \cdot s_n(\mathbf{w}_{n-1}) \\ &= O(s_n(\mathbf{w}_{n-1})). \end{aligned}$$

A similar argument using the left inequality in (4.2.14) yields

$$s_n(\mathbf{w}_{n-1}) \leq \frac{R^2 - \|\mathbf{w}_n\|^2}{R^2 - \|\mathbf{w}_{n-1}\|^2} \cdot s_n(\mathbf{w}_n) = O(s_n(\mathbf{w}_n)).$$

□

One important consequence of Claim 2 of Lemma 4.5 is that if β and R are chosen appropriately then $\mathbf{w}_{n-1}$ is close enough to the analytic center of Ω_n and *one* Newton step will suffice to recenter $\mathbf{w}_{n-1}$ to a new approximate center of Ω_n.

Lemma 4.6 *Assume the same conditions as Lemma 4.5 for some fixed n. Let $\mathbf{w}_n$ and $s_n(\mathbf{w})$ be obtained by the IPLS algorithm, then we have the following:*

1. *There holds*

$$\frac{\|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} = \frac{\sqrt{2}}{\beta R} = O(R^{-1}), \qquad \frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|}{s_n(\mathbf{w}_n)} = O(R^{-2}). \tag{4.2.15}$$

2. *If R is large,*

$$\tau_{n+1} := \mathcal{F}_{n+1}(\mathbf{w}_n) + \beta \frac{R}{\sqrt{2}} \|\nabla \mathcal{F}_{n+1}(\mathbf{w})\| = O(R^{-1}) \leq t.$$

Proof. By assumption, all the conclusions of Lemma 4.5 hold. Therefore

$$\|\mathbf{w}_n\| = O(1) \quad \text{and} \quad \|\mathbf{w}_{n-1}\| = O(1) \tag{4.2.16}$$

and if we choose R and β sufficiently large

$$\left\langle \nabla \phi_n(\mathbf{w}_{n-1}), [\nabla^2 \phi_n(\mathbf{w}_{n-1})]^{-1} \nabla \phi_n(\mathbf{w}_{n-1}) \right\rangle^{\frac{1}{2}} < 0.9. \tag{4.2.17}$$

The first part of Claim 1 of Lemma 4.6 follows directly from the updating rule of τ_n. To show the second part of the first claim of Lemma 4.6, we need to establish some bounds on the Hessian of ϕ_n. Recall

$$\nabla^2 \phi_n(\mathbf{w}) = \frac{2\mathbf{I}}{t_n(\mathbf{w})} + \frac{2\mathbf{w}\mathbf{w}^H}{t_n(\mathbf{w})^2} + \frac{\nabla^2 \mathcal{F}_n}{s_n(\mathbf{w})} + \frac{(\nabla \mathcal{F}_n)(\nabla \mathcal{F}_n)^H}{s_n^2(\mathbf{w})}.$$

Using Condition 1 and $t_n(\mathbf{w}_n) = R^2 - \|\mathbf{w}_n\|^2 = O(R^2)$, we obtain the following bounds

$$\nabla^2 \phi_n(\mathbf{w}_{n-1}) \geq \frac{\sigma_1}{s_n(\mathbf{w}_{n-1})} \mathbf{I},$$

and

$$\begin{aligned} \nabla^2 \phi_n(\mathbf{w}_n) &\leq \left(O(R^{-2}) + O(R^{-4}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)} \right) \mathbf{I} \\ &\leq \left(O(R^{-2}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)} \right) \mathbf{I}. \end{aligned}$$

By (4.2.17), $\mathbf{w}_{n-1}$ is an approximate analytic center, so there holds quadratic convergence Nesterov and Nemirovskii (1994):

$$\left\langle \nabla \phi_n(\mathbf{w}_n), [\nabla^2 \phi_n(\mathbf{w}_n)]^{-1} \nabla \phi_n(\mathbf{w}_n) \right\rangle \leq \left\langle \nabla \phi_n(\mathbf{w}_{n-1}), [\nabla^2 \phi_n(\mathbf{w}_{n-1})]^{-1} \nabla \phi_n(\mathbf{w}_{n-1}) \right\rangle^2.$$

Denote the left and right sides of the above inequality by LHS and RHS respectively. Then

$$\begin{aligned} \text{LHS} &\geq \|\nabla\phi_n(\mathbf{w}_n)\|^2 \cdot \left(O(R^{-2}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)^{-1} \\ \text{RHS} &\leq \|\nabla\phi_n(\mathbf{w}_{n-1})\|^4 \cdot \frac{s_n^2(\mathbf{w}_{n-1})}{\sigma_1^2}. \end{aligned}$$

Combining the above we obtain

$$\|\nabla\phi_n(\mathbf{w}_n)\|^2 \leq \frac{s_n^2(\mathbf{w}_{n-1})}{\sigma_1^2}\left(O(R^{-2}) + \frac{\sigma_2}{s_n(\mathbf{w}_n)} + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)\|\nabla\phi_n(\mathbf{w}_{n-1})\|^4.$$

Notice that $s_n(\mathbf{w}_n) \leq \tau_n \leq t = O(1)$ and that the quotient $\frac{s_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_n)}$ is $O(1)$ (due to Lemma 4.5 (Claim 3)). The above inequality further simplifies to

$$\|\nabla\phi_n(\mathbf{w}_n)\|^2 \leq \left(O(1) + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)\|\nabla\phi_n(\mathbf{w}_{n-1})\|^4. \tag{4.2.18}$$

Note that from (4.2.16) and

$$\frac{\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} = \frac{\sqrt{2}}{\beta R} = O(R^{-1}), \qquad \nabla\phi_n(\mathbf{w}_{n-1}) = \frac{\nabla\mathcal{F}_n(\mathbf{w}_{n-1})}{s_n(\mathbf{w}_{n-1})} + \frac{2\mathbf{w}_{n-1}}{R^2 - \|\mathbf{w}_{n-1}\|^2},$$

we can immediately deduce

$$\|\nabla\phi_n(\mathbf{w}_{n-1})\| = O(R^{-1}).$$

Using this result and the definition of $\nabla\phi_n$ in (4.2.18) we obtain

$$\left\|\frac{\nabla\mathcal{F}_n(\mathbf{w}_n)}{s_n(\mathbf{w}_n)} + \frac{2\mathbf{w}_n}{R^2 - \|\mathbf{w}_n\|^2}\right\|^2 \leq \left(O(1) + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)O(R^{-4}).$$

Applying the inequality $\|a+b\|^2 \geq \frac{1}{2}\|a\|^2 - \|b\|^2$ and noting $\|2\mathbf{w}_n\|^2/(R^2 - \|\mathbf{w}_n\|^2)^2 = O(R^{-4})$, we further obtain

$$\frac{1}{2}\frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)} - \frac{\|2\mathbf{w}_n\|^2}{(R^2 - \|\mathbf{w}_n\|^2)^2} \leq \left(O(1) + \frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}\right)O(R^{-4})$$

Solving for the $\frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)}$ term yields

$$\frac{\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2}{s_n^2(\mathbf{w}_n)} = O(R^{-4}) \tag{4.2.19}$$

which immediately verifies the second part of the first claim in the lemma.

It remains to show the second claim that the updated threshold τ_{n+1} will still be bounded by t. We first note that the update formula for τ_{n+1} is

$$\begin{aligned}\tau_{n+1} &= \mathcal{F}_{n+1}(\mathbf{w}_n) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_{n+1}(\mathbf{w}_n)\| \\ &= \mathcal{F}_{n+1}(\mathbf{w}_n) + \beta\frac{R}{\sqrt{2}}s_n(\mathbf{w}_n)O(R^{-2}) \\ &= \mathcal{F}_{n+1}(\mathbf{w}_n) + \frac{\beta}{\sqrt{2}}O(R^{-1})O(1)\end{aligned}$$

where the second step follows from (4.2.19), and the third line holds because $s_n(\mathbf{w}_n) \leq \tau_n \leq t = O(1)$. We next bound $\mathcal{F}_{n+1}(\mathbf{w}_n)$,

$$\begin{aligned}\mathcal{F}_{n+1}(\mathbf{w}_n) &= \frac{1}{n+1}\sum_{i=1}^{n+1}\left|y_i - \mathbf{x}_i^H\mathbf{w}_n\right|^2 \\ &= \frac{n}{n+1}\mathcal{F}_n(\mathbf{w}_n) + \frac{1}{n+1}[\mathbf{x}_{n+1}^H(\mathbf{w}_* - \mathbf{w}_n)]^2 \\ &\leq \frac{n}{n+1}\mathcal{F}_n(\mathbf{w}_n) + (\mathbf{w}_* - \mathbf{w}_n)^H\mathbf{R}_{xx}(n+1)(\mathbf{w}_* - \mathbf{w}_n) \\ &\leq \frac{n}{n+1}\mathcal{F}_n(\mathbf{w}_n) + \sigma_2\|\mathbf{w}_* - \mathbf{w}_n\|^2.\end{aligned}$$

Note that each term above is bounded by Lemma 4.4, thus

$$\begin{aligned}\mathcal{F}_{n+1}(\mathbf{w}_n) &\leq \frac{n}{n+1}\left(\frac{1}{4\sigma_1}\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2\right) + \sigma_2\left(\frac{1}{4\sigma_1^2}\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2\right) \\ &\leq \frac{1+\sigma_2}{4\sigma_1^2}\|\nabla\mathcal{F}_n(\mathbf{w}_n)\|^2 \\ &= \frac{1+\sigma_2}{4\sigma_1^2}O(R^{-4})s_n^2(\mathbf{w}_n) \;=\; O(R^{-4}).\end{aligned}$$

Finally, substituting back into our update formula we obtain

$$\tau_{n+1} = O(R^{-4}) + O\left(R^{-1}\right) \leq t.$$

□

The main result here is the first claim, which states that by taking one Newton step in the IPLS algorithm, the size of the quantity $\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|/s_n(\mathbf{w}_{n-1})$ reduces from $O(R^{-1})$ to $O(R^{-2})$. This will later help to prove our main convergence result in Theorem 4.2. We point out that the estimate (4.2.15) holds only under the assumption $\tau_n \leq t$ (as assumed in Lemma 4.5). However, Claim 2 of Lemma 4.6 ensures that $\tau_{n+1} \leq t$, thus Lemmas 4.5 and 4.6 can be applied again at iteration $n+1$. This fact will be used in the proof of our main convergence result below.

Theorem 4.2 *Let the sequence of estimates* $\{\mathbf{w}_n, n = 1, 2, 3, \dots\}$ *be generated by the IPLS algorithm. Suppose Lemmas 4.4, 4.5 and 4.6 hold at some iteration* n_1. *Then*

$$\|\mathbf{w}_n - \mathbf{w}_*\| = O(R^{-1})\|\mathbf{w}_{n-1} - \mathbf{w}_*\|, \qquad n \geq n_1.$$

Proof. Fix some t so that $\tau_n \leq t$ for some n_1. Suppose β, R are chosen large enough. Then Lemma 4.6 holds for this n_1 so that $\tau_{n_1+1} \leq t$. But this implies Lemma 4.6 should hold for $n_1 + 1$. A simple inductive argument shows Lemma 4.6 should hold for all $n \geq n_1$. Thus, we have

$$\frac{\|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} = \frac{\sqrt{2}}{\beta R} \quad \text{and} \quad \frac{\|\nabla \mathcal{F}_n(\mathbf{w}_n)\|}{s_n(\mathbf{w}_n)} = O(R^{-2}), \quad \forall n \geq n_1.$$

Therefore,

$$\begin{aligned} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| &= R^{-1}\, s_n(\mathbf{w}_n)\, O(R^{-1}) \\ &= \frac{\|\nabla \mathcal{F}(\mathbf{w}_{n-1})\|}{s_n(\mathbf{w}_{n-1})} \frac{\beta}{\sqrt{2}}\, s_n(\mathbf{w}_n)\, O(R^{-1}) \\ &= \|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\| O(R^{-1}) \end{aligned} \tag{4.2.20}$$

where the last step is due to Lemma 4.5 (Claim 3). By the recursive update of $\nabla \mathcal{F}_n(\mathbf{w})$, we have

$$\begin{aligned} \|\nabla \mathcal{F}_{n+1}(\mathbf{w}_n)\| &= \left\| \frac{n}{n+1} \nabla \mathcal{F}_n(\mathbf{w}_n) + \frac{2(y_{n+1} - \mathbf{x}_{n+1}^H \mathbf{w}_n)}{n+1} \mathbf{x}_{n+1} \right\| \\ &= \frac{n}{n+1} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| + \left\| \frac{2}{n+1} \left(\mathbf{x}_{n+1} \mathbf{x}_{n+1}^H\right) (\mathbf{w}_* - \mathbf{w}_n) \right\| \\ &\leq \frac{n}{n+1} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| + 2\sigma_2 \|\mathbf{w}_* - \mathbf{w}_n\| \\ &\leq \frac{n}{n+1} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| + \frac{2\sigma_2}{2\sigma_1} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| \\ &\leq \frac{\sigma_1 + \sigma_2}{\sigma_1} \|\nabla \mathcal{F}_n(\mathbf{w}_n)\| \end{aligned}$$

where we used the right inequality of Condition 1 in step 3 and Lemma 4.4 in step 4. Using (4.2.20) in the equation above we can then write

$$\|\nabla \mathcal{F}_{n+1}(\mathbf{w}_n)\| \leq \|\nabla \mathcal{F}_n(\mathbf{w}_{n-1})\| O(R^{-1}).$$

Due to the boundedness of the autocorrelation matrix imposed in Condition 1, this is equivalent to

$$\|\mathbf{w}_n - \mathbf{w}_*\| = O(R^{-1})\|\mathbf{w}_{n-1} - \mathbf{w}_*\|,$$

which shows the exponential decay of the transient process. □

Theorem 4.2 establishes the exponential transient convergence of the IPLS algorithm.

4.3 Discussion

The results above establish important convergence behaviour of IPLS under the condition that the estimated autocorrelation matrix is full rank (Condition 1). Another important aspect of transient behaviour is the convergence of an algorithm when $n < M$. In this case the autocorrelation matrix can have at most rank n, thus Condition 1 is certainly not satisfied. Recall that in the simulations of Chapter 3 we observed better transient behaviour of IPLS in precisely the case described above, *i.e.*, there are fewer observations than there are parameters. This implies that IPLS has also better convergence in the cost function (or criterial convergence) than RLS. We have considered this convergence issue briefly from a theoretical point of view, without any noteworthy results. This therefore remains an interesting topic for future investigation.

Chapter 5

MMSE Decision Feedback Equalization with short Training Sequences

In many communication systems training sequences are used to help the receiver identify and/or equalize the channel. The amount of training data required depends on how fast an equalizer can adjust to the observed data samples. Consequently, when adaptive filtering algorithms are used for the computation of equalizer tap-weights, it is their convergence properties that dictate the length of the training sequence. The Interior Point Least Squares (IPLS) algorithm was shown to have excellent transient convergence properties in Chapter 4. In this chapter we exploit this feature of IPLS to update the weight vector for a minimum-mean-square-error decision-feedback equalizer (MMSE-DFE) in a downlink code division multiple access (CDMA) scenario. Numerical simulations show that when training sequences are short IPLS consistently outperforms RLS in terms of system bit-error-rate and packet error rate. As the training sequence gets longer IPLS matches the performance of the RLS algorithm.

5.1 Introduction

In a wireless mobile communication system, the transmitted signal typically travels through a time-dispersive multipath channel which gives rise to intersymbol interference (ISI) at the receiver. Such ISI is known to be one of the main causes for signal distortion and

therefore must be canceled by some form of channel equalization. Since mobile channels are also time-varying, the channel equalization process must be adaptive to account for the nonstationary characteristics of the channel (Qureshi, 1985; Proakis, 1991). The decision-feedback equalizer (DFE) is a well-established and effective approach for the mitigation of the ISI. It provides improved performance compared to the linear transversal equalizer and reduced complexity (at almost the same performance) compared to maximum-likelihood sequence estimation (MLSE) (Proakis, 1983). We consider a DFE that is implemented to minimize the mean-square error criterion (MMSE-DFE). The MMSE-DFE has been identified as the best among the suboptimal (compared to MLSE) receivers for wireless applications by Klein *et al.* (1996) and has been well studied theoretically by Cioffi *et al.* (1995) and Al-Dhahir and Cioffi (1995).

When the DFE is updated iteratively, the tap weights are usually computed by an adaptive algorithm of the least-mean-square (LMS) or recursive-least-squares (RLS) family. These algorithms rely on a training period during which the receiver has *a priori* knowledge of the transmitted symbols (Proakis, 1991). The tap weights obtained at the end of a training period are used to initialize the channel equalizer in the subsequent transmission period until a new training period begins. To maximize system capacity, it is of course desirable to use as few training symbols as possible. Note that there are numerous approaches that can identify and equalize the channel "blindly", without the requirement of training data. Blind algorithms, however, cannot compete with training based methods in terms of performance and computational complexity. An alternative approach to minimize the length of the training sequence is to concentrate on the transient performance of the adaptive filtering algorithm that is used to initialize the equalizer tap weights. If transient performance is measured by the number of observations (training symbols) it takes an algorithm to attain "sufficiently" low estimation error, then transient performance essentially dictates what the minimum length of the training sequence should be.

The fast transient convergence behaviour of IPLS allows it to make effective use of limited training data. In a downlink CDMA scenario we used the IPLS algorithm to update the weight vector for a minimum-mean-square-error decision-feedback equalizer (MMSE-DFE). Numerical simulations show that when training sequences are short (less training symbols than DFE tap weights) IPLS consistently outperforms RLS in terms of system bit-error-rate. As the training sequence gets longer IPLS matches the performance of the RLS algorithm.

Below we first describe in more detail the components of the particular communication system we simulate in this chapter. The following section presents the results of numerical simulations.

5.2 System Description

In our simulation example we model the (synchronous) downlink portion of a CDMA system (see Figure 5.1). There are K users whose symbol rate signals are given by $\{u_i(n)\}, i = 1, \ldots, K$. The base station upsamples these signals by a spreading factor L and applies a particular L-tap FIR code filter $C_i(z)$ to each of the user signals. The chip rate signals are then all summed before transmission through a multipath channel $H(z)$. The spreading sequences are chosen to be Walsh codes of length 16 (Proakis, 1983) (hence $L = 16$). These codes have zero cross-correlation when there is only one received path and equal delay between users. For the system under consideration these codes represent a reasonable (not necessarily optimal) choice.

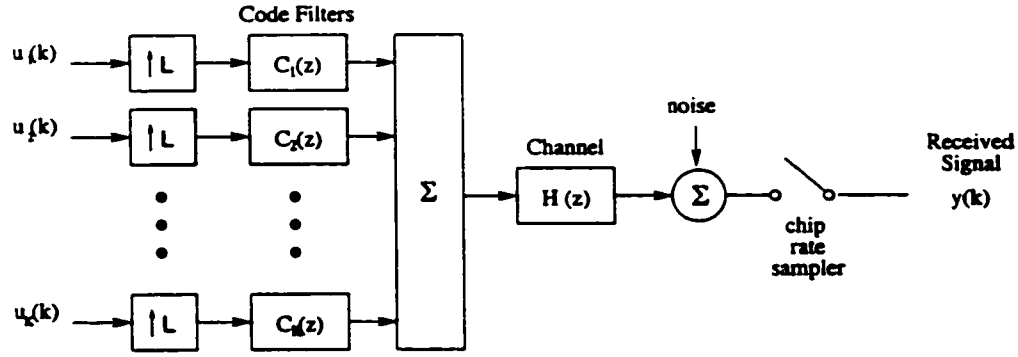


Figure 5.1: *Discrete-time model of CDMA downlink*

The user signals are modeled as QPSK, where each point in the alphabet $\{1+i, 1-i, -1+i, -1-i\}$ has equal probability. The channel impulse response is composed of three different taps, each undergoing Rayleigh fading independently. The first tap is considered to be the line-of-sight component and is on average 5 dB stronger than each of the multipath taps. The delays for each of the paths is determined randomly for each simulated channel, but the delay spread (delay between first and last tap of the channel) never exceeds $6T_c$, where T_c is the chip period. The time varying nature of the channel can be characterized in many different ways. In our case we selected a normalized fading rate $f_D T_c$ to be 0.005 to characterize a slowly fading channel. Figure 5.2 shows the time-variation of the tap amplitudes of one realization of this channel (the phase of each path is uniform between $[0, 2\pi]$).

Alternatively, we also consider a static channel, where the tap weights for each path remain constant for the duration of the adaptive filtering procedure. The tap weights for the static channel are determined by sampling the fading channel at a random instant. Both

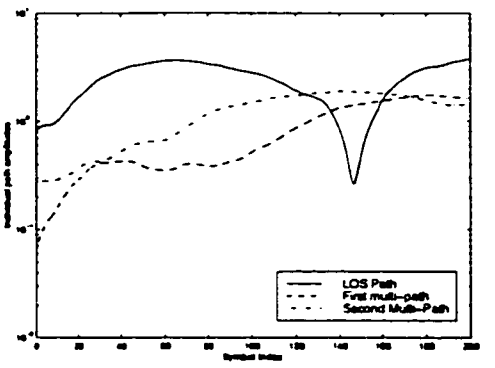


Figure 5.2: *Three ray Rayleigh Fading Channel with fading rate 0.005*

types of channel also include additive white Gaussian noise.

The receiver configuration for user 1 is depicted in Figure 5.3. The chip rate sampled received signal $y(n)$ is first passed through a code matched filter which implies that the receiver assumes knowledge only of the desired user's spreading code. The output of the code matched filter $r(n)$ is received by a decision feedback equalizer (DFE) that has a feedforward filter operating at the chip rate, and a symbol rate feedback loop. All simulations were carried out using 14 taps in the forward filter and 2 taps in the feedback filter of the DFE. For the given receiver structure and channel conditions, these filter lengths were found to yield good results (Abdulrahman *et al.*, 1994).

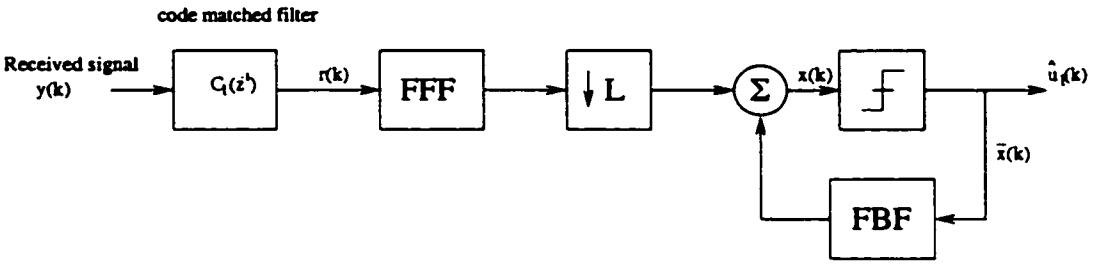


Figure 5.3: *Code Matched Filter - Chip rate DFE*

It is useful to introduce some variables at this point:

$y(n)$:	the sampled received signal;
$r(n)$:	the input to the feedforward filter (FFF);
$x(n)$:	the input to the decision device in the DFE;
$\bar{x}(n)$:	the input to the feedback filter (FBF);
$c_{k,0}, \ldots, c_{k,\ell}$:	the $\ell + 1$ tap coefficients of the FFF at time k;
$b_{k,1}, \ldots, b_{k,m}$:	the m tap coefficients of the FBF at time k.

The input to the decision device $\{x(n)\}$ can be computed by

$$x_n = \sum_{i=0}^{\ell} c_{n-1,i} r_{n-i} - \sum_{i=1}^{m} b_{n-1,i} \bar{x}_{n-i} = \mathbf{r}_n^H \mathbf{w}_{n-1}$$

where

$$\begin{aligned} \mathbf{r}_n &= [r_n, r_{n-1}, \cdots, r_{n-\ell}, -\bar{x}_{n-1}, \cdots, -\bar{x}_{n-m}]^H \\ \mathbf{w}_{n-1} &= [c_{n-1,0}, c_{n-1,1}, \cdots, c_{n-1,\ell}, b_{n-1,1}, \cdots, b_{n-1,m}]^T \end{aligned}$$

are complex vectors. $\mathbf{w}_{n-1}$ represents the adaptive filter tap coefficients based on observations up to time $n-1$. The equalization error at time n is defined as

$$\epsilon_n = d_n - x_n = d_n - \mathbf{r}_n^H \mathbf{w}_{n-1}$$

where d_n is the corresponding desired signal.

This defines all the quantities of interest for the adaptive filtering algorithms. The vector sequence $\{\mathbf{r}_n\}$ is used as the input sequence and obviously the sequence of filters $\{\mathbf{w}_n\}$ estimates the optimum equalizer tap weights. The error sequence $\{\epsilon_n\}$ is used by RLS to control the step size of the adaptation. Once the training sequence has been received the DFE is operated in decision-directed mode, where the quantized equalizer output is used instead of the training symbol d_n. In the simulations below, we always continue the decision-directed adaptation using the same algorithm that was used during training. Note that in practice one may desire to save in computation by switching to a less complex algorithm, such as LMS, once the adaptive equalizer has been initialized.

5.3 Simulation Results

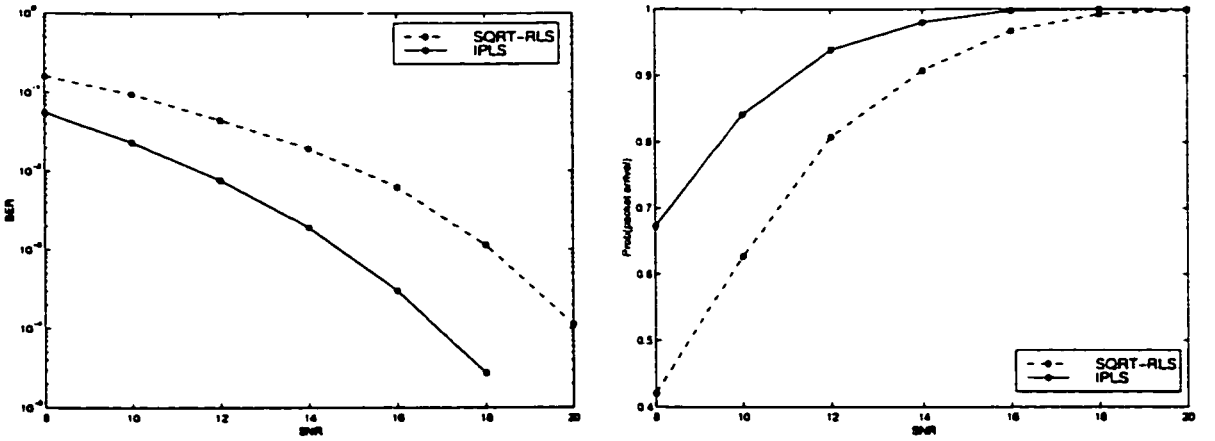
Two different algorithms are compared for the equalization of multipath channels: the IPLS algorithm and the square-root RLS algorithm. The choice of square-root RLS as a reference is motivated by two factors: RLS methods are generally considered to have better convergence properties than LMS based methods, and the square-root version of the algorithm is more numerically stable than regular RLS. For a comprehensive reference on the family of RLS algorithms see Sayed and Kailath (1994). All experimental results shown in this section were obtained by averaging over 5000 independent Monte Carlo trials.

Experiment 5.1: Static Channel

Let us first consider the static channel case. The tap weights of the unknown channel remain constant for the duration of the message signal. We set the length of the message signal to $N = 200$, and use the first $N_T = 10$ symbols as a training signal. The overhead due to training is thus only 5% of the total bandwidth. The algorithms initialize the regularization term with $\delta = R^{-1} = 10^{-4}$. The IPLS algorithm furthermore initializes $\beta = 2$. To simplify the simulation, we have used a fixed equalizer delay (equal to one symbol period), even though this may be suboptimal for some channels.

To compare the performance of the two algorithms we measure the bit-error-rate (BER) at various signal-to-noise ratios (SNR). Here and in subsequent experiments the BER is computed as the average probability of (bit) error over all Monte Carlo runs. The result for the case where only one user is transmitting is shown in Figure 5.4a. It can be seen that IPLS performs significantly better than the RLS algorithm. Specifically, RLS requires about 4 dB higher SNRs to achieve the same error performance as IPLS. Because of the error propagation inherent in decision feedback equalization we also evaluated the probability of packet failure to ensure a fair comparison between the two algorithms. The message signal of 200 symbols (including the training symbols) is regarded to be a "data packet", and is considered to be successfully received if the bit-error-rate over the data part of the packet is less than a certain threshold, in this case 10^{-2}. The probability of (successful) packet arrival for the single user case is shown in Figure 5.4b. Here the SNR gain of the IPLS algorithm over RLS is about 3 dB. These and subsequent results are obtained without any error-correction coding, which is typically used in wireless CDMA applications. Thus the results serve more as a comparison between different algorithms for adapting the DFE rather than absolute performance results of a complete receiver.

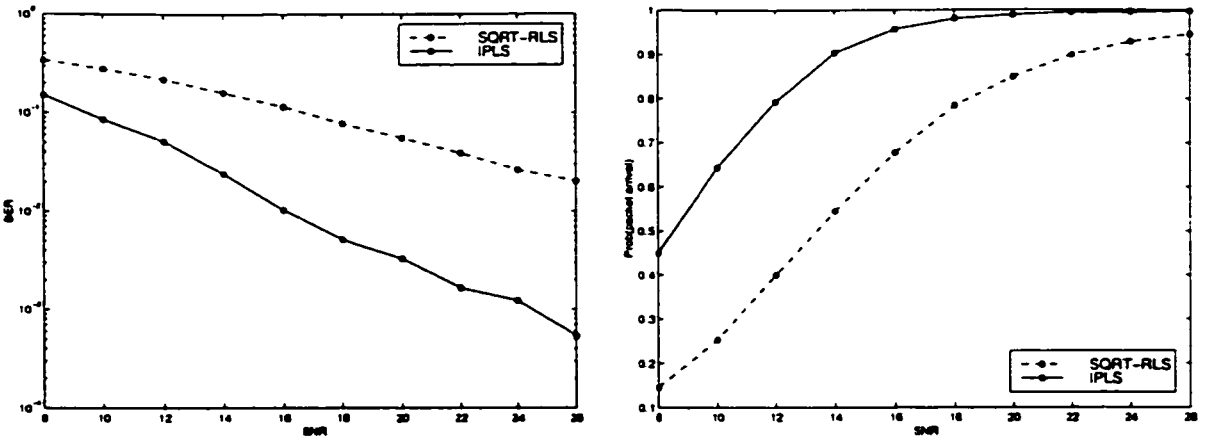
We next consider the multi-user case. The base station signal is comprised of four different user signals, and we show the results for one receiver. It is the purpose of the code matched filter to reduce the co-channel interference (CCI) from the signals of the "other" users. However, since the channel introduces intersymbol interference, the orthogonality of the codes no longer guarantees perfect cancellation of interference. As is shown in Figure 5.5a, the additional interference causes a degradation in equalization performance for both the RLS and IPLS algorithms. The performance gap between the two methods, however, has increased, indicating that IPLS is more robust to the remaining CCI than RLS. Figure 5.5b shows the corresponding plot of probability of packet arrival. It should be noted that multiuser detection techniques exist by use of which the receiver should attain essentially the performance of the single-user case (Madhow and Honig, 1994).



(a) (b)

Figure 5.4: *Equalization performance in the single user, static channel case. (a) BER performance, (b) Probability of packet arrival.*

Of course, the performance gap that was shown in Figures 5.4 and 5.5 is dependent mainly on the fact that we are using relatively short training sequences. To illustrate the dependence of each algorithm on the length of the training sequence N_T, we fix the signal-to-noise ratio to 12 dB and vary the number of training symbols used. Figure 5.6 shows results of this simulation for the single user case. The curves in Figure 5.6a can be divided into two segments. At their tail end the curves show significantly improved equalization performance when $N_T > 16$. Since the equalizer has a total of $M = 16$ taps, a minimum of 16 observations are required for the autocorrelation matrix to attain full rank. Hence, there is a sharp dropoff in BER at $N_T = 18$. The difference between RLS and IPLS is the greatest when $N_T \leq M$. In this region RLS gradually improves with the increasing number of training symbols. IPLS on the other hand is virtually independent of N_T, as long as there is at least *some* training available. The advantage of IPLS in this case is clearly due to its better transient behaviour. However, the results depicted in Figure 5.6 go even beyond the theoretical results presented in Chapter 4. When $N_T < M$, the weak persistent excitation condition is not fulfilled, hence exponential convergence in the parameter is theoretically not guaranteed. The fact that we still observe equalization then indicates that convergence in the cost function (*i.e.*, the mean squared error) is sufficient in some sense. This in turn implies that IPLS has also better convergence in the cost function (or criterial convergence) than RLS (this has been previously observed in Chapter 3). We provide some simulation evidence of the difference in convergence in cost in Figure 5.7. Fixing the training length to $N_T = 2$ and SNR to 15 dB, we plot the mean-squared error versus time for each algorithm. It can be seen here that even in the first few iterations, IPLS has a mean-squared error



(a) (b)

Figure 5.5: *Equalization performance in the 4 user, static channel case. (a) BER performance, (b) Probability of packet arrival.*

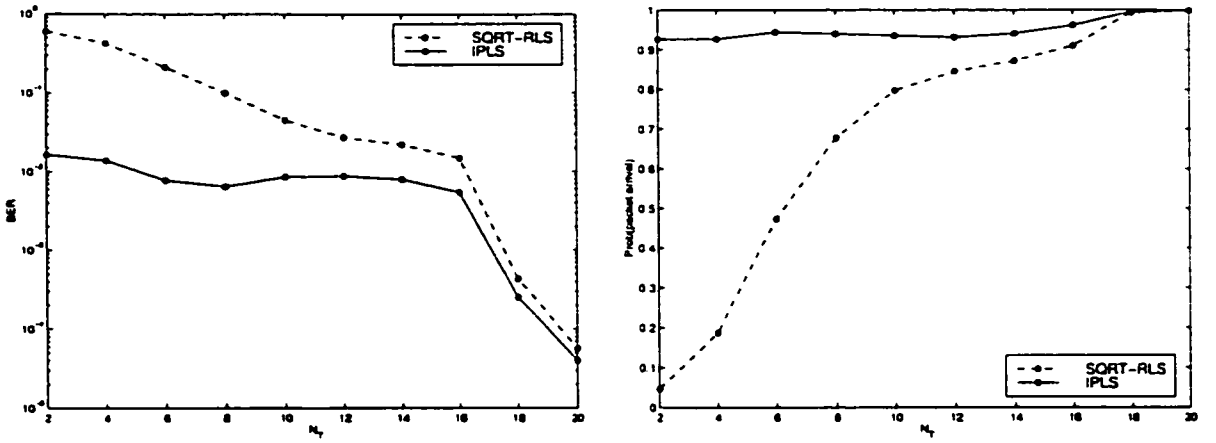
significantly below that of RLS.

Experiment 5.2: Time-Varying Channel

We now consider a time-varying multipath channel as discussed in Section 5.2. To deal with the time-varying nature of the channel, we use an exponentially decaying memory by employing a forgetting factor $\lambda < 1$.

As we have done in Experiment 5.1 we investigate how the equalization performance of each algorithm depends on the number of training symbols. The plots in Figure 5.8 were obtained at a fixed SNR of 18 dB. There is only one user in the system and both algorithms use a forgetting factor $\lambda = 0.85$. From the BER plot of Figure 5.8a we can see that as in the static channel, IPLS maintains a distinct advantage in BER performance in the region $0 < N_T \leq M$. At this SNR even as few as two training symbols are sufficient for IPLS to achieve a 90% success rate in equalizing packets. RLS requires at least 18 symbols to ensure the same kind of performance.

We now fix the training length and test how the equalization performance varies with the SNR. Figure 5.9 shows four curves in total; both RLS and IPLS were run with $N_T = 2$ and $N_T = 10$. The forgetting factor λ is set to 0.85 throughout. As suggested already in Figure 5.8, we observe that RLS depends heavily on the length of the training sequence. In the case of $N_T = 2$ the SQRT-RLS algorithm simply fails to equalize the channel, no matter what the SNR. IPLS is able to equalize the channel with 2 training symbols already.



(a) (b)

Figure 5.6: *Equalization performance versus length of training sequence: static channel. (a) BER performance, (b) Probability of packet arrival.*

In the case of the IPLS algorithm, increasing the training length to $N_T = 10$ results in an improvement especially at higher SNRs where ISI is the dominant cause of error.

Figure 5.10 depicts results for one user in the simulation of the 4 user scenario. Since length 2 training sequences now yielded inconsistent results we only show the case of $N_T = 10$. The advantage of the IPLS algorithm is around 6 dB according to Figure 5.10a and approximately 5 dB in Figure 5.10b. Both curves in Figure 5.10a seem to level out above 26 dB. This indicates that in that SNR range signal reception is limited by the amount of residual ISI, that could not be removed by the equalization algorithm.

5.4 Discussion

The fast convergence property of IPLS is exploited in an application to adaptive equalization of time-dispersive channels encountered in wireless communications. The IPLS algorithm is able to equalize channels using fewer training symbols than are required by the RLS algorithm. In particular, a minimum mean-squared-error decision feedback equalizer (MMSE-DFE) with 14 feedforward and 2 feedback taps can be effectively trained using as few as two training symbols. Depending on the channel conditions, the length of the training signal and number of co-channel user signals the gain of using IPLS over the RLS algorithm can range from 3-4 dB to well over 10 dB.

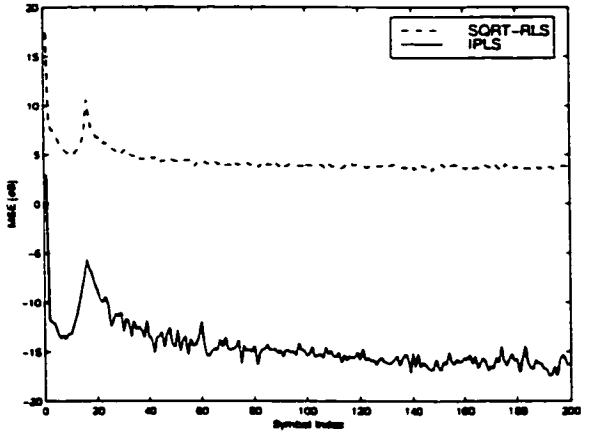
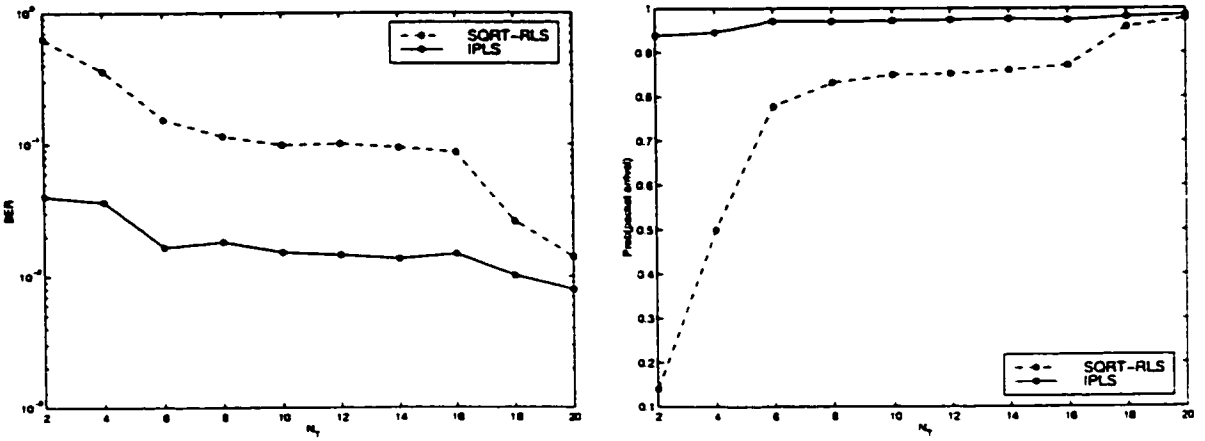


Figure 5.7: *Mean-squared error versus time, $N_T = 2$, SNR=15dB.*



(a) (b)

Figure 5.8: *Equalization performance versus length of training sequence: time-varying channel. (a) BER performance, (b) Probability of packet arrival.*

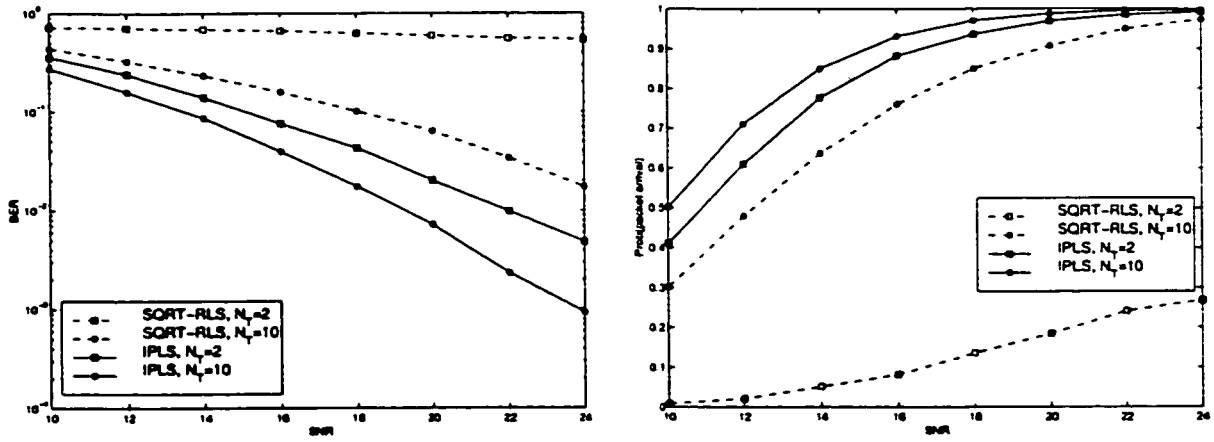


Figure 5.9: *Equalization performance in the single user, time-varying channel case. (a) BER performance, (b) Probability of packet arrival.*

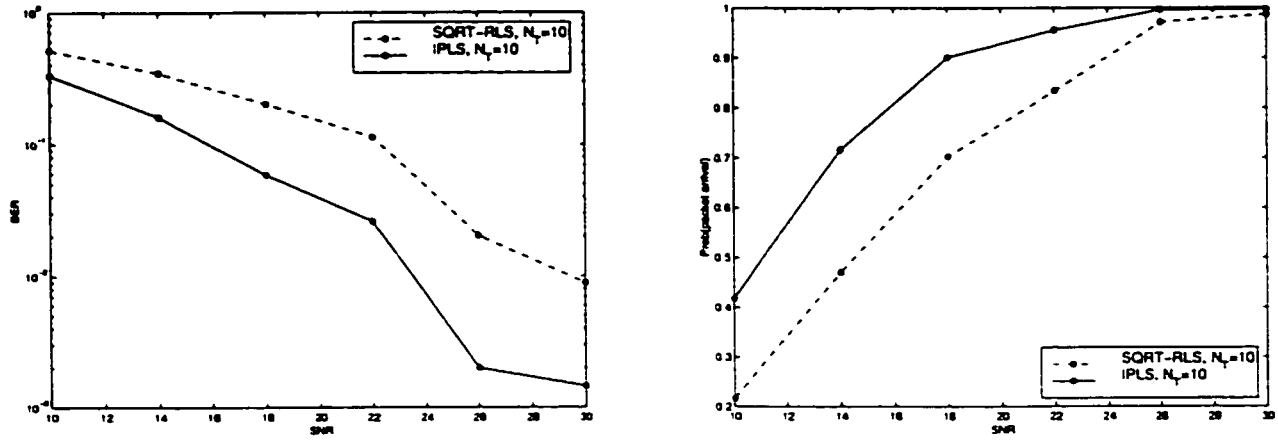


Figure 5.10: *Equalization performance in the 4 user time-varying channel case. (a) BER performance, (b) Probability of packet arrival.*

Chapter 6

Numerically stable Constrained Adaptive Estimation using IPLS

Linearly constrained adaptive filtering problems occur in applications such as adaptive digital beamforming. The main issues with existing algorithms are slow convergence in the case of constrained LMS (Frost, 1972) and numerical instability (RLS based algorithms). In this chapter we investigate the applicability of the interior point least squares (IPLS) algorithm to adaptive beamforming. The IPLS based method is shown to match RLS in terms of convergence speed and there seem to be no problems with numerical instability.

The chapter is organized as follows. In Section 6.1 we give the adaptive beamforming problem formulation and summarize some existing solution methods. Our proposed algorithm is described in Section 6.2 and in the following section we present some numerical simulations to illustrate the potential of our method. As usual we close the chapter with some discussion in Section 6.4.

6.1 Introduction

In many adaptive filtering problems the estimated parameters must also satisfy some linear constraints. For example, in the context of telecommunications, the most frequently quoted application is the *adaptive digital beamforming* problem. It should be noted that beamforming finds application in many other fields as well (*e.g.*, radar, sonar, geophysics, and biomedical signal processing (Haykin, 1998)). In essence, a beamformer is a *spatial* filter that is used to distinguish between the spatial properties of signal and noise. The aim

often is to adaptively place nulls in the directions of the interferences (or jammers) while protecting the target signal (steering capability). When the receiver is a linear array of M identical sensors, the beamformer can essentially be modeled as an adaptive transversal filter (Frost, 1972; Haykin, 1998). The requirement for steering capability gives rise to some linear constraints.

The linearly constrained adaptive filtering problem we consider in this chapter is stated as follows. Given a sequence of input vectors $\mathbf{x}_i$ and a desired response sequence y_i, the objective is to identify a linear filter $\mathbf{w}$ of length M, to minimize the mean-squared error $\mathcal{F}_n(\mathbf{w})$, subject to some linear constraints (imposed by target directions). Using an exponentially decaying memory the problem formulation becomes

$$\begin{aligned} \min \quad & \mathcal{F}_n = \sum_{i=1}^{n} \lambda^{n-i} |y_i - \mathbf{x}_i^T \mathbf{w}|^2, \\ \text{s.t.} \quad & \mathbf{C}^T \mathbf{w} = \mathbf{f}, \ \mathbf{w} \in \mathbb{R}^M, \end{aligned} \tag{6.1.1}$$

where λ is the forgetting factor; the matrix $\mathbf{C}$ and vector $\mathbf{f}$ are of appropriate dimensions and they represent the linear (side) constraints for the adaptive filter.

When the filter is trained with "signal-free observations" the problem above reduces to minimizing the output power, while keeping the same constraints. The resulting formulation is a special case of the Minimum Variance Distortionless Response (MVDR) beamformer,

$$\begin{aligned} \min \quad & \mathbf{w}^T \mathbf{R}_{xx}(n) \mathbf{w} \\ \text{s.t.} \quad & \mathbf{C}^T \mathbf{w} = \mathbf{f}, \ \mathbf{w} \in \mathbb{R}^M, \end{aligned}$$

where $\mathbf{R}_{xx}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}_i \mathbf{x}_i^T$ is the exponentially weighted autocorrelation matrix. This formulation also enjoys several *robustness* characteristics. The inclusion of the point constraints $\mathbf{C}^T \mathbf{w} = \mathbf{f}$ is one of the most popular methods to make the beamformer robust to pointing and calibration errors (Gershman, 1999). Such robustness is also achieved by *diagonally loading* the autocorrelation matrix, *viz*, $\mathbf{R}_{xx}(n)$ is augmented by a term $\alpha \mathbf{I}$. The loading factor arises from imposing a quadratic constraint on the norm of the weight vector, thus this technique is also referred to as *quadratically constrained* beamforming. Diagonal loading also leads to robustness of the beamformer to small sample sizes (Gershman, 1999).

Perhaps the easiest and most widely used technique for solving (6.1.1) is the constrained LMS algorithm by Frost (1972). Here, the weight vector is updated according to

$$\mathbf{w}_{n+1} = \mathbf{P}[\mathbf{w}_n - \mu \hat{y}(n) \mathbf{x}_n^T] + \mathbf{F},$$

where $\mathbf{P} = \mathbf{I}_M - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T$ and $\mathbf{F} = \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{f}$ are used to project the unconstrained estimate onto the space of vectors that do satisfy the constraints. This method inherits

LMS' simplicity and computational efficiency, but also its convergence problems in the case of correlated input signals.

Other methods are based on the adaptation gain (or Kalman Gain) readily computed by the RLS algorithm. We follow some of the development of Resende *et al.* (1996) to derive an RLS based beamforming algorithm. Solving the constrained minimization (6.1.1) by the method of Lagrange multipliers leads to the following solution,

$$\mathbf{w}_n = \mathbf{R}_{xx}^{-1}(n)\mathbf{p}_{xy}(n) + \mathbf{R}_{xx}^{-1}(n)\mathbf{C}[\mathbf{C}^T\mathbf{R}_{xx}^{-1}(n)\mathbf{C}]^{-1}(\mathbf{f} - \mathbf{C}^T\mathbf{R}_{xx}^{-1}(n)\mathbf{p}_{xy}(n)), \tag{6.1.2}$$

where $\mathbf{p}_{xy}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}_i y_i$ is the exponentially weighted cross-correlation vector of the desired signal with the input signal. In the case of signal-free observations this vector is zero (which leads to an obvious simplification of (6.1.2)).

The term $\mathbf{R}_{xx}^{-1}(n)\mathbf{p}_{xy}(n)$ can be identified as the solution of the unconstrained problem and the second term in (6.1.2) is a correction in order to satisfy the constraints. Using the definitions

$$\begin{aligned} \mathbf{w}_n^{unc} &= \mathbf{R}_{xx}^{-1}(n)\mathbf{p}_{xy}(n), \\ \Gamma_n &= \mathbf{R}_{xx}^{-1}(n)\mathbf{C}, \\ \Psi_n &= \mathbf{C}^T\Gamma_n, \end{aligned}$$

we rewrite the Lagrange solution as

$$\mathbf{w}_n = \mathbf{w}_n^{unc} + \Gamma_n\Psi_n^{-1}[\mathbf{f} - \mathbf{C}^T\mathbf{w}_n^{unc}]. \tag{6.1.3}$$

It is well known that the adaptation (or Kalman) gain, $\mathbf{g}_n$ can be used to update $\mathbf{w}_n^{unc}$,

$$\mathbf{w}_{n+1}^{unc} = \mathbf{w}_n^{unc} + \mathbf{g}_{n+1}[y_{n+1} - \mathbf{w}_n^{unc}\mathbf{x}_{n+1}]. \tag{6.1.4}$$

RLS computes $\mathbf{g}_n$ as

$$\mathbf{g}_n = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}_n}{1 + \lambda^{-1}\mathbf{x}_n^T\mathbf{P}(n-1)\mathbf{x}_n},$$

where $\mathbf{P}(n)$ is the inverse of the autocorrelation matrix $\mathbf{R}_{xx}(n)$ and updated by $\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{g}_n\mathbf{x}_n^T\mathbf{P}(n-1)$. Since Γ_n and Ψ_n^{-1} are closely related to $\mathbf{R}_{xx}^{-1}(n)$ we can easily derive formulae for their recursive updates as well (Resende *et al.*, 1996)

$$\begin{cases} \Gamma_{n+1} = \dfrac{1}{\lambda}[\Gamma_n - \mathbf{g}_{n+1}\mathbf{x}_{n+1}^T\Gamma_n], \\ \Psi_{n+1}^{-1} = \dfrac{1}{\lambda}[\Psi_n^{-1} + \mathbf{l}_{n+1}\mathbf{x}_{n+1}^T\Gamma_n\Psi_n^{-1}], \end{cases} \tag{6.1.5}$$

where

$$\mathbf{l}_{n+1} = \frac{1}{\lambda}[\mathbf{C}^T\Gamma_{n+1}]^{-1}\mathbf{C}^T\mathbf{g}_{n+1}. \tag{6.1.6}$$

Equations (6.1.3)–(6.1.6) already represent an algorithm for solving (6.1.1). Notice, however, that the constraint vector $\mathbf{f}$ never enters the algorithm except at the initialization (not shown here), thus the algorithm would soon diverge due to round-off errors. A mechanism to enforce the linear constraints is suggested by Resende *et al.* (1996).

A new term (just like in (6.1.3)) is added to correct numerical errors, this time using the explicit error in the constraints $[\mathbf{f} - \mathbf{C}^T\mathbf{w}_n]$ as the direction for correction. Suppose $\mathbf{w}_{bc}$ is the parameter vector *before correction*. Then the final estimate is found as

$$\mathbf{w}_{n+1} = \mathbf{w}_{bc} + \Gamma_{n+1}\Psi_n^{-1}[\mathbf{f} - \mathbf{C}^T\mathbf{w}_n]. \tag{6.1.7}$$

We shall refer to this algorithm as linearly constrained recursive least squares (LC-RLS).

In the next section we present a novel technique for the linearly constrained adaptive estimation problem based on the IPLS algorithm of Chapter 3. Its convergence speed matches that of the RLS based method and there seem to be no problems with numerical instability. The numerical stability is maintained even for relatively low forgetting factors, thus one can trade steady state performance for adaptation speed by changing the forgetting factor.

6.2 Interior Point Least Squares for Constrained Adaptive Filtering

The only difference between the problem (6.1.1) and the standard least-squares estimation problem we considered previously in Chapter 3, is that now we have some additional linear constraints on the filter. At each iteration $n \geq 1$ we require a filter that satisfies

$$\begin{cases} \mathcal{F}_n(\mathbf{w}) & \leq & \tau_n, \\ \|\mathbf{C}^T\mathbf{w} - \mathbf{f}\|^2 & \leq & \epsilon^2, \\ \|\mathbf{w}\|^2 & \leq & R^2, \end{cases}$$

where τ_n is a dynamic threshold as before and R is a normalizing constant to ensure $\mathbf{w}$ is bounded. The parameter ϵ^2 specifies the tolerance within which the additional point constraints must be satisfied.

The additional linear constraints lead to a straightforward modification of the algorithm of Chapter 3. The feasible region in the constrained case is

$$\Omega_n = \{\mathbf{w} \in \mathbb{R}^M \mid \mathcal{F}_n(\mathbf{w}) \leq \tau_n, \ \|\mathbf{C}^T\mathbf{w} - \mathbf{f}\|^2 \leq \epsilon^2, \ \|\mathbf{w}\|^2 \leq R^2\},$$

giving rise to the new *logarithmic barrier function*

$$\phi_n(\mathbf{w}) = -\log(\tau_n - \mathcal{F}_n(\mathbf{w})) - \log(\epsilon^2 - \|\mathbf{C}^T\mathbf{w} - \mathbf{f}\|^2) - \log(R^2 - \|w\|^2).$$

As before, ϕ_n is convex and approaches infinity on the boundary of Ω_n. The gradient and Hessian of ϕ_n are now given by

$$\begin{aligned}
\nabla\phi_n(\mathbf{w}) &= \frac{\nabla\mathcal{F}_n}{s_1} + \frac{2\mathbf{C}(\mathbf{C}^T\mathbf{w} - \mathbf{f})}{s_2} + \frac{2\mathbf{w}}{s_3}, \\
\nabla^2\phi_n(\mathbf{w}) &= \frac{(\nabla\mathcal{F}_n)(\nabla\mathcal{F}_n)^T}{s_1^2} + \frac{\nabla^2\mathcal{F}_n}{s_1} + \frac{4\mathbf{C}(\mathbf{C}^T\mathbf{w} - \mathbf{f})(\mathbf{C}^T\mathbf{w} - \mathbf{f})^T\mathbf{C}^T}{s_2^2} \\
&\quad + \frac{2\mathbf{C}\mathbf{C}^T}{s_2} + \frac{4\mathbf{w}\mathbf{w}^T}{s_3^2} + \frac{2\mathbf{I}}{s_3},
\end{aligned}$$

where the slack variables s_i are defined by $s_1 := \tau_n - \mathcal{F}_n(\mathbf{w})$, $s_2 := \epsilon^2 - \|\mathbf{C}^T\mathbf{w} - \mathbf{f}\|^2$, and $s_3 := R^2 - \|\mathbf{w}\|^2$, and the gradient and Hessian of $\mathcal{F}_n(\mathbf{w})$ are computed as in Chapter 3.

Note that it is geometrically intuitive that the extra constraints change the feasible region of $\mathbf{w}$. The optimization technique used to find the center of the feasible region, however, is the same as before. Thus the update equation for the threshold parameter τ_n is the same as in Chapter 3,

$$\tau_n = \mathcal{F}_n(\mathbf{h}_{w-1}) + \beta\frac{R}{\sqrt{2}}\|\nabla\mathcal{F}_n(\mathbf{w}_{n-1})\|_2,$$

and $\mathbf{w}$ is still updated with one exact Newton step,

$$\mathbf{w}_n = \mathbf{w}_{n-1} - (\nabla^2\phi_n(\mathbf{w}_{n-1}))^{-1}\nabla\phi_n(\mathbf{w}_{n-1}).$$

We remark that the analytical convergence results of Chapter 4 do not hold automatically for the linearly constrained IPLS algorithm described above. Similarly, we have not addressed the question whether it is still possible to compute the Newton direction efficiently (in $O(M^{2.2})$, or similar). A quick inspection of the expression for the Hessian matrix $\nabla^2\phi_n(\mathbf{w})$, allows the following observations: the additional linear constraints cause two new terms to appear. One of them is a rank one term, which can easily be handled in $O(M^2)$ operations. The second term is $2\mathbf{C}\mathbf{C}^T/s_2$. Since ϵ^2 should be a very small number we could certainly achieve efficient updates by approximating $s_2 = \epsilon^2 - \|\mathbf{C}^T\mathbf{w} - \mathbf{f}\|^2 \approx \epsilon^2$.

6.3 Simulation

The experiment we use is based on the one presented by Resende *et al.* (1996). The input signal is composed of three interferences at normalized frequencies 0.15, 0.1625, and 0.35, and white noise at an SNR of 40 dB,

$$x(n) = \sin(0.3n\pi) + \sin(0.325n\pi) + \sin(0.7n\pi) + b(n).$$

The filter has length $M = 11$ and is constrained to have unity response at target frequencies 0.1 and 0.25. This produces the constraint parameters

$$\mathbf{C}^T = \begin{bmatrix} 1 & \cos(0.2\pi) & \dots & \cos((M-1)0.2\pi) \\ 1 & \cos(0.5\pi) & \dots & \cos((M-1)0.5\pi) \\ 0 & \sin(0.2\pi) & \dots & \sin((M-1)0.2\pi) \\ 0 & \sin(0.5\pi) & \dots & \sin((M-1)0.5\pi) \end{bmatrix}$$

and

$$\mathbf{f} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T.$$

We tested our proposed algorithm against Frost's constrained LMS and the RLS type algorithm as outlined in Section 6.1. The RLS implementation is stabilized in that the symmetry of the autocorrelation matrix is always assured (Haykin, 1998, Chapter 19). The three algorithm shall be referred to as LC-LMS, LC-RLS and LC-IPLS, respectively.

To emphasize the stability issue, some of the higher level computations were limited to four decimal digits of numerical precision. This corresponds to an accuracy of at least 13 bits in a hardware implementation. Figure 6.1 shows the convergence of the estimated filter sequence to the ideal Lagrange solution. Clearly, both the LC-RLS and LC-IPLS converge much quicker than LC-LMS. But after about 1000 iterations the LC-RLS algorithm becomes unstable, whereas the new method shows no sign of instability. The forgetting factor here is $\lambda = 0.99$, and LC-LMS uses a step size of $\mu = 0.1$ for "fast" convergence. The result is averaged over 10 independent trials. Figure 6.2 shows the frequency responses of the filters after 4000 iterations.

We also tested the proposed algorithm in a more difficult scenario, as far as stability is concerned. Reducing the forgetting factor to 0.9 we ran an input sequence of length 100,000. Due to the small λ, the estimated filter does not separate the two closely spaced frequencies anymore, however, the adaptation remains stable and matches the constraints throughout the length of the simulation (see Figure 6.3).

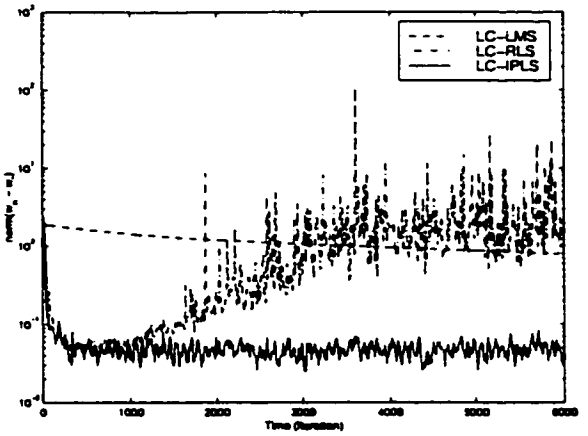


Figure 6.1: *Mean-squared error in* $\mathbf{w}_n$

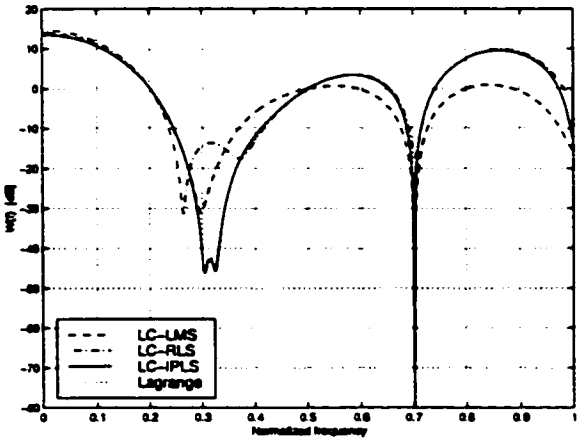


Figure 6.2: *Frequency response of filters at Iteration 4000*

We have shown the stability of LC-IPLS regardless of the forgetting factor λ. This property can be further exploited in applications where the system can undergo abrupt changes, and tracking performance becomes important. With the LC-IPLS method we may sacrifice some estimation precision for faster adaptation (by lowering the forgetting factor) without fear of instability. Figure 6.4 shows the convergence of all three algorithms after an abrupt change occurs at iteration 1000. Before $n = 1000$ the signal frequencies are 0.325, 0.350, 0.375 and then they switch to the familiar values of previous experiments. It was necessary in this case to reduce the step size μ in the LC-LMS to 0.03. At $\mu = 0.1$, LC-LMS would diverge immediately. The forgetting factors are 0.995 and 0.95 for LC-RLS and LC-IPLS, respectively. This allows LC-IPLS to adjust to the new signal within about 100 samples, while LC-RLS takes much longer to converge. Note that for this last experiment

the numerical precision of computation was set to 6 digits. Otherwise, LC-RLS would become unstable almost immediately after the change in the input signal at iteration 1000. A potential drawback of IPLS may be the oscillatory nature it exhibits in Figure 6.4. We cannot offer an explanation for this behaviour at this point, however, in this experiment the oscillations do not lead to instabilty, even for long sets of signals.

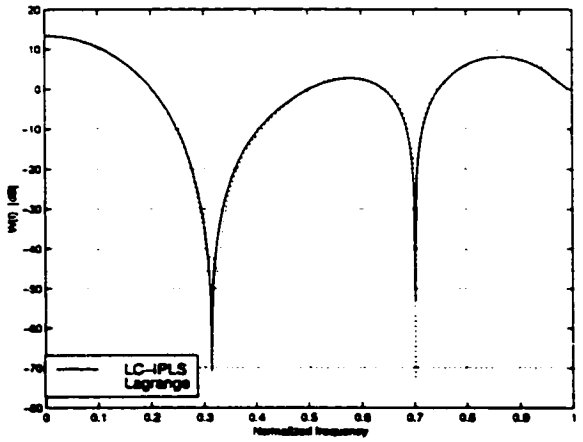


Figure 6.3: *Frequency response of* $\mathbf{w}_n$ *for LC-IPLS at iteration 100,000* ($\lambda = 0.9$)

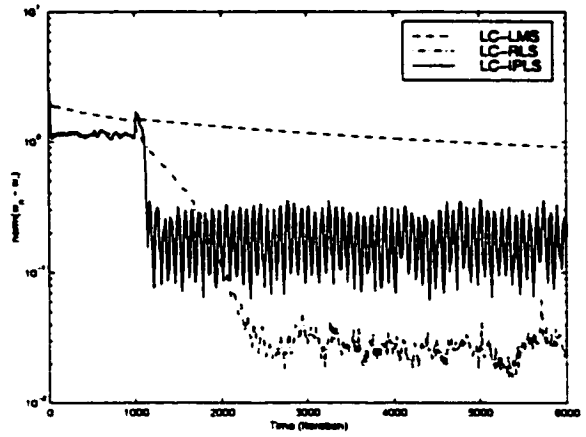


Figure 6.4: *Abrupt change at iteration 1000*

A note about computational complexity: the current implementation of the LC-IPLS algorithm requires about twice the computation time of the LC-RLS algorithm. Both algorithms are significantly more expensive to compute than Frost's LC-LMS algorithm.

6.4 Discussion

We have proposed a novel approach for the estimation of adaptive filters under linear constraints. A comparison with the work of Resende *et al.* (1996) shows that our method matches the convergence speed of RLS based methods, however, the interior point method leads to a numerically more stable solution of the constrained filtering problem. The method is flexible as it can handle an arbitrary number of either linear or convex quadratic constraints.

The inclusion of additional constraints causes certain technical issues. Namely, the convergence results of Chapter 4 do not automatically hold for the algorithm derived in this chapter, and we have also not established that the Newton iteration that leads to the updated filter estimate can be computed efficiently. A close approximation, however, can be determined in $O(M^{2.2})$.

Finally, the stability that we claim for our LC-IPLS algorithm is based purely on observation, as no rigorous results have yet been obtained. Also, the results of the chapter should not be interpreted as to say that our algorithm is numerically more stable than any RLS based algorithm. In fact, there exist a multitude of other RLS algorithms in the literature that may very well be better than the one used here. One certain benefit of the IPLS algorithm is, however, that the inclusion of extra linear or convex quadratic constraints is very simple. The IPLS framework allows for a unified description of both constrained and unconstrained algorithms.

Chapter 7

Summary and Discussion

The objective of this work was to investigate the applicability of interior point optimization methods to the adaptive filtering problem. In a first approach, interior point column generation algorithms were considered. The iterative nature with which these algorithms operate to find solutions of static problems (with potentially a large number of constraints) seemed attractive for the adaptive filtering problem, which also requires an iterative solution. Problems arose due to the unbounded nature of noise, which led to heuristic modifications of the algorithm. These modifications encumbered the theoretical analysis of the method. The simulations of this algorithm were promising, however, suggesting that it may have advantages over the recursive least-squares (RLS) algorithm in terms of transient performance.

Aided by the experience gained in applying the column generation methods to adaptive filtering, we next developed the interior point least squares (IPLS) approach. This algorithm is simpler and more elegant than the one based on column generation. It requires only one Newton step in each iteration and can be implemented efficiently with complexity $O(M^{2.2})$ per iteration, where M denotes the filter size. Moreover, IPLS does not rely on any heuristic measures to deal with the noise. Consequently, the method is more mathematically tractable, which is reflected in the fact that both asymptotic and transient convergence have been established. The exponential transient convergence rate is among the algorithms principal advantages, and it was exploited in an application to channel equalization, where we showed that IPLS can equalize channels with fewer training symbols than required by RLS. Another attractive property of IPLS is its numerical stability in the presence of additional constraints and limited precision arithmetic. This was shown in the context of adaptive beamforming, where again we compared our method to a (stabilized) RLS algorithm.

In summary, it can be said that the IPLS algorithm and its analysis and applications presented in this thesis establish the importance of interior point optimization to dynamic problems in engineering. Moreover, the IPLS algorithm has merits of its own as it proved to be a fast converging algorithm that is inherently numerically stable. In our opinion, IPLS may become a viable alternative to LMS and RLS in many adaptive filtering applications.

7.1 Further Work

The work in this thesis can be continued or extended in at least two general directions. First, we can complete the characterization of the IPLS algorithm by investigating more of its properties. Several open issues in this context were identified in the discussion sections of previous chapters: *i)* in Chapter 3 we mentioned that it is worth exploring whether one can exploit structural properties of the adaptive filtering problem to derive "fast IPLS" algorithms; *ii)* the discussion in Chapter 4 raised the question whether it is possible to show any analytical convergence results without invoking the weak persistent excitation condition; and *iii)* the analytical treatment of convergence and numerical stability of the linearly constrained IPLS algorithm in Chapter 6 also remain open problems. Another important property that should be analyzed in the future is the tracking performance of IPLS.

The second category of future research is concerned with more fundamental or conceptual questions. An open problem here is the exact relationship of IPLS to the Kalman filter. Kalman filter theory has been used by Sayed and Kailath (1994) to derive essentially all RLS-type algorithms in the literature. Since IPLS is similar to RLS in that it differs only in the way the regularization is determined, it would be interesting to investigate whether it too can be fitted into the Kalman filtering framework. And if so, what type of Kalman filter does IPLS corresponds to ? Throughout the thesis we have restricted our treatment of adaptive filtering to the least-squares case. Lately, the H^∞ criterion (*viz*, worst case error, or minimax criterion) has received considerable attention, especially in context with Kalman filter theory. Can an interior point algorithm be formulated to (efficiently) minimize an H^∞ criterion ? How would this algorithm compare to the H^∞-optimal LMS ?

It is our hope that this thesis represents only the first step in the study of adaptive interior point algorithms for engineering problems.

Postscript

Parts of the work reported in this thesis have previously appeared in the literature or have been submitted for publication as follows. The IPCG algorithm of Chapter 2 was first presented at a workshop on semidefinite programming in the Netherlands (Afkhamie *et al.*, 1996). Different parts of the work on the IPLS algorithm of Chapter 3 were presented in conferences (Afkhamie *et al.*, 1998b, 1999a) and as a full paper (Afkhamie *et al.*, 2000a), which also contains the complete asymptotic convergence analysis of IPLS (given here in Chapter 4). The adaptive equalization work of Chapter 5 appears in (Afkhamie *et al.*, 2000b) and together with the transient analysis of IPLS in (Afkhamie *et al.*, 1999b). Finally, the constrained IPLS algorithm of Chapter 6 has previously been presented in form of a conference paper (Afkhamie *et al.*, 1998a).

Bibliography

ABDULRAHMAN, M., SHEIKH, A. U. H., and FALCONER, D. D. (1994). Decision feedback equalization for CDMA in indoor wireless communications. *IEEE Journal on Selected Areas in Communications*, 12(4):698–706.

AFKHAMIE, K. H., DAVIDSON, T. N., and LUO, Z.-Q. (1998a). Constrained adaptive estimation using interior point optimization. In *Proceedings of the 1998 Symposium on Image, Speech, Signal Processing and Robotics*, pp. 21–26, Hong Kong. Invited.

AFKHAMIE, K. H., LUO, Z.-Q., and WONG, K. M. (1996). Interior Point Column Generation algorithms for adaptive filtering. presented in *The International Workshop on Semidefinite Programming and Applications*, Delft University of Technology, Delft, The Netherlands.

AFKHAMIE, K. H., LUO, Z.-Q., and WONG, K. M. (1998b). Adaptive system identification using interior point optimization. In *Proceedings of the* 9^{th} *IEEE Workshop on Statistical Signal and Array Processing*, Portland, Oregon.

AFKHAMIE, K. H., LUO, Z.-Q., and WONG, K. M. (1999a). Adaptive parameter estimation using interior point optimization techniques: convergence analysis. In *Proceedings of ICASSP 1999*, pp. 1681–1684, Phoenix, Arizona.

AFKHAMIE, K. H., LUO, Z.-Q., and WONG, K. M. (1999b). MMSE decision-feedback equalization using short training sequences: an application of Interior Point Least Squares. *IEEE Transactions on Signal Processing*. Submitted for publication.

AFKHAMIE, K. H., LUO, Z.-Q., and WONG, K. M. (2000a). Adaptive linear filtering using interior point optimization techniques. *IEEE Transactions on Signal Processing*. To appear (June 2000).

AFKHAMIE, K. H., LUO, Z.-Q., and WONG, K. M. (2000b). Interior Point Least Squares Estimation: exploiting transient convergence in MMSE decision-feedback equalization. In *Proceedings of ICASSP 2000*, Istanbul, Turkey.

AL-DHAHIR, N. and CIOFFI, J. M. (1995). MMSE decision feedback equalizers: finite-length results. *IEEE Transactions on Information Theory*, 41(4):961–975.

BAI, E., FU, M., TEMPO, R., and YE, Y. (1998). Analytic center approach to parameter estimation: convergence analysis. In *Proceedings of ICASSP 1998*, pp. 2293–2296, Seattle, WA.

BAI, E., YE, Y., and TEMPO, R. (1997). Bounded error parameter estimation: a sequential analytic center approach. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pp. 732–737.

BAI, E., YE, Y., and TEMPO, R. (1999). Bounded error parameter estimation: a sequential analytic center approach. *IEEE Transactions on Automatic Control*, 44(6):1107–1117.

BAYKAL, B. and CONSTANTINIDES, A. (1997). Underdetermined order recursive least squares adaptive filtering: the concept and algorithm. *IEEE Transactions on Signal Processing*, 45(2):346–362.

BELFORTE, G., BONA, B., and CERONE, V. (1990). Parameter estimation algorithms for a set-membership description of uncertainty. *Automatica*, 26(5):887–898.

BERTSIMAS, D. and TSITSIKLIS, J. (1997). *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA.

BOTTO, J. and MOUSTAKIDES, G. (1989). Stabilizing the fast Kalman algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP–37(9):1342–1348.

BOYD, S., VANDENBERGHE, L., and GRANT, M. (1994). Efficient convex optimization for engineering design. In *Proceedings IFAC Symposium on Robust Control Design*, Rio de Janeiro, Brazil.

CARAYANNIS, G., KALOUPTSIDIS, N., and MANOLAKIS, D. (1982). Fast recursive algorithms for a class of linear equations. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP–30(2):227–239.

CIOFFI, J. and KAILATH, T. (1984). Fast recursive LS transversal filters for adaptive processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP–32(2):304–337.

CIOFFI, J. M., DUDEVOIR, G. P., EYUBOGLU, M. V., and FORNEY JR., G. D. (1995). MMSE decision-feedback equalizers and coding—Part I: equalization results. *IEEE Transactions on Communications*, 43(10):2582–2594.

DASGUPTA, S. and HUANG, Y. (1987). Asymptotically convergent modified recursive least squares with data dependent updating and forgetting for systems with bounded noise. *IEEE Transactions on Information Theory*, 33(3):383–392.

DAVIDSON, T. N., LUO, Z.-Q., and WONG, K. M. (1999). Design of orthogonal pulse shapes for communications via semidefinite programming. *IEEE Transactions on Signal Processing*. To appear.

DE CAMPOS, M. and ANTONIOU, A. (1997). A new Quasi-Newton adaptive filtering algorithm. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, 44(11):924–934.

DELLER, J. R. (1989). Set membership identification in digital signal processing. *IEEE Signal Processing Magazine*, 6(4):4–19.

DINIZ, P., DE CAMPOS, M., and ANTONIOU, A. (1995). Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor. *IEEE Transactions on Signal Processing*, 43(3):617–628.

FARHANG-BOROUJENY, B. (1997). Fast LMS/Newton algorithms based on autoregressive modeling and their application to acoustic echo cancellation. *IEEE Transactions on Signal Processing*, 45(8):1987–2000.

FOGEL, E. and HUANG, Y. (1982). On the value of information in system identification — bounded noise case. *Automatica*, 18(2):229–238.

FREUND, R. and MIZUNO, S. (1996). Interior point methods: current status and future directions. *OPTIMA — Mathematical Programming Society Newsletter No. 51*. Also available from `http://web.mit.edu/rfreund/www/ipm.ps`.

FROST, O. L. (1972). An algorithm for linearly constrained adaptive array processing. *Proceedings of the IEEE*, 60(8):926–935.

GAUSS, C. F. (1809). *Theory of the motion of heavenly bodies*. Dover, New York. (English translation (1963) of "Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum").

GERSHMAN, A. B. (1999). Robust adaptive beamforming in sensor arrays. *International Journal of Electronics and Communications*. To appear.

GLENTIS, G.-O., BERBERIDIS, K., and THEODORIDIS, S. (1999). Efficient least squares adaptive algorithms for FIR transversal filtering. *IEEE Signal Processing Magazine*, 16(4):13–41.

GODARD, D. N. (1974). Channel equalization using a Kalman filter for fast data transmission. *IBM Journal of Research and Development*, 18(3):267–273.

GOFFIN, J.-L., LUO, Z.-Q., and YE, Y. (1994). *On the Complexity of a Column Generation Algorithm for Convex or Quasi-Convex Feasibility Problems*, pp. 182–189. Kluwer Academic Publishers. W. W. Hager, D. W. Hearn and P. M. Pardalos, Editors.

GOLUB, G. H. and VAN LOAN, C. F. (1996). *Matrix Computations*. Johns Hopkins University Press, Baltimore and London, 3rd edition.

HASSIBI, B., SAYED, A. H., and KAILATH, T. (1996). H^∞ optimality of the LMS algorithm. *IEEE Transactions on Signal Processing*, 44(2):267–280.

HAYKIN, S. (1998). *Adaptive Filter Theory*. Prentice Hall, NJ, 3rd edition.

KAILATH, T. (1974). A view of three decades of linear filtering theory. *IEEE Transactions on Information Theory*, IT-20(2):146–181.

KALMAN, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME; J. Basic. Eng.*, 82:35–45.

KALOUPTIDIS, N. and THEODORIDIS, S. (1993). *Adaptive System Identification and Signal Processing Algorithms*. Prentice Hall, NJ.

KARMARKAR, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

KHACHIYAN, L. G. (1979). A polynomial algorithm for linear programming. *Doklady Akad. Nauk USSR*, 244:1093–1096. Translated in *Soviet Math. Doklady*, 20:191–194, 1979.

KLEIN, A., KALEH, G. K., and BAIER, P. W. (1996). Zero forcing and minimum mean-square-error equalization for multiuser detection in code-division multiple-access channels. *IEEE Transactions on Vehicular Technology*, 45(2):276–287.

KOLMOGOROV, A. N. (1941). Stationary sequences in Hilbert space. *Bull. Math. Univ. Moscow*, 2(6). (A translation by N. Artin is available in many libraries).

KREIN, M. G. (1945a). On a generalization of some investigations of G. Szegö, W. M. Smirnov, and A. N. Kolmogorov. *Dokl. Akad. Nauk SSSR*, 46:91–94.

KREIN, M. G. (1945b). On a problem of extrapolation of A. N. Kolmogorov. *Dokl. Akad. Nauk SSSR*, 46:306–309.

LAWRENCE, R. E. and KAUFMAN, H. (1971). The Kalman filter for the equalization of a digital communication channel. *IEEE Transactions on Communication Technology*, COM-19(6):1137–1141.

LIU, H. and HE, Z. (1995). A sliding-exponential window RLS adaptive filtering algorithm: properties and applications. *Signal Processing*, 45(3):357–368.

LJUNG, L., MORF, M., and FALCONER, D. D. (1978). Fast calculations of gain matrices for recursive estimations schemes. *International Journal of Control*, 27(1):1–19.

LUO, Z.-Q. (1997). Analysis of a cutting plane method that uses weighted analytic center and multiple cuts. *SIAM Journal on Optimization*, 7(3):697–716.

LUO, Z.-Q. and SUN, J. (1999). An analytic center based column generation algorithm for convex quadratic feasibility problem. *SIAM Journal on Optimization*, 9(1):217–235.

MADHOW, U. and HONIG, M. (1994). MMSE interference suppression for direct-sequence spread spectrum CDMA. *IEEE Transactions on Communications*, 42(12):3178–3188.

MARSHAL, D. and JENKINS, W. (1992). A fast Quasi-Newton adaptive filtering algorithm. *IEEE Transactions on Signal Processing*, 40(7):1652–1662.

MAVRIDIS, P. and MOUSTAKIDES, G. (1996). Simplified Newton-type adaptive estimation algorithms. *IEEE Transactions on Signal Processing*, 44(8):1932–1940.

MOUSTAKIDES, G. and THEODORIDIS, S. (1991). Fast Newton transversal algorithms: a new class of adaptive estimation algorithms. *IEEE Transactions on Signal Processing*, 39(10):2184–2193.

MOUSTAKIDES, G. V. (1997). Study of the transient phase of the forgetting factor RLS. *IEEE Transactions on Signal Processing*, 45(10):2468–2476.

NESTEROV, Y. and NEMIROVSKII, A. (1994). *Interior Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics, Philadelphia, PA.

PANDA, G., MULGREW, B., COWAN, C., and GRANT, P. (1986). A self-orthogonalizing efficient block adaptive filter. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP–34:1573–1582.

PLACKETT, R. L. (1950). Some theorems in least squares. *Biometrica*, 37:149–157.

POWELL, M. J. D. (1998). The use of band matrices for second derivative approximations in trust region algorithms. In Yuan, Y., editor, *Advances in Nonlinear Programming*, pp. 3–28. Kluwer Academic Publishers, Dordrecht.

PROAKIS, J. G. (1983). *Digital Communications*. McGraw-Hill, New York.

PROAKIS, J. G. (1991). Adaptive equalization for TDMA digital mobile radio. *IEEE Transactions on Vehicular Technology*, 40(2):333–341.

PROAKIS, J. G., RADER, C. M., LING, F., and NIKIAS, C. L. (1992). *Advanced Digital Signal Processing*. Macmillan Publishing Co., NY.

QURESHI, S. U. H. (1985). Adaptive equalization. *Proceedings of the IEEE*, 73(9):1340–1387.

RESENDE, L. S., ROMANO, J. M. T., and BELLANGER, M. G. (1996). A fast least-squares algorithm for linearly constrained adaptive filtering. *IEEE Transactions on Signal Processing*, 44(5):1168–1174.

ROOS, C., TERLAKY, T., and VIAL, J.-P. (1997). *Theory and Algorithms for Linear Optimzation: An Interior Point Approach*. John Wiley & Sons, New Jersey.

RUPP, M. (1998). A family of adaptive filter algorithms with properties. *IEEE Transactions on Signal Processing*, 46(3):2771–2774.

SAYED, A. H. and KAILATH, T. (1994). A state-space approach to adaptive RLS filtering. *IEEE Signal Processing Magazine*, 11(3):18–60.

SHANNON, C. E. (1948). The mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656.

SLOCK, D. T. M. and KAILATH, T. (1991). Numerically stable fast transversal filters for recursive least-squares adaptive filtering. *IEEE Transactions on Signal Processing*, 39(1):92–114.

TANAKA, M., MAKINO, S., and KOJIMA, J. (1999). A block exact fast affine projection algorithm. *IEEE Transactions on Speech and Audio Processing*, 7(1):79–87.

TERLAKY, T. (1996). *Interior Point Methods of Mathematical Programming*. Kluwer Academic Publishers, Dordrecht.

THEODORIDIS, S., MOUSTAKIDES, G., and BERBERIDIS, K. (1995). A fast Newton multichannel algorithm for decision feedback equalization. *IEEE Transactions on Signal Processing*, 43(1):327–331.

VANDENBERGHE, L. and BOYD, S. (1996). Semidefinite programming. *SIAM Review*, 38(1):49–95.

VICINO, A. and ZAPPA, G. (1996). Sequential approximation of feasible parameter sets for identification with set membership uncertainty. *IEEE Transactions on Automatic Control*, 41(6):774–785.

WIDROW, B. and HOFF, J. (1960). Adaptive switching circuits. *IRE WESCON Conv. Rec.*, pp. 96–104.

WIDROW, B. and STEARNS, S. D. (1985). *Adaptive Signal Processing*. Prentice Hall, NY.

WIENER, N. (1949). *Extrapolation, Interpolation and Smoothing of Stationary Time Series, with Engineering Applications*. Technology Press and Wiley, New York. (Originally issued in February 1942 as a classified Nat. Defense Res. Council Rep.).

WRIGHT, S. (1997). *Primal-Dual Interior Point Algorithms*. SIAM Publications, Philadelphia, PA.

YE, Y. (1997). *Interior Point Algorithms: Theory and Analysis*. John Wiley & Sons, New York.