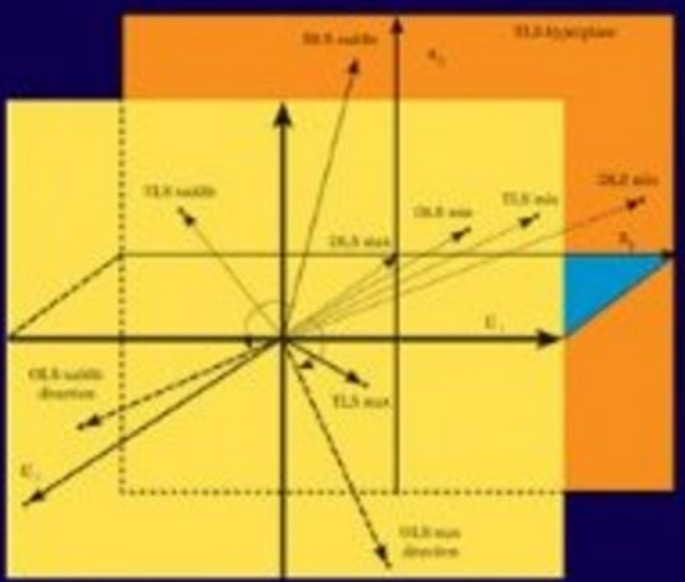
Wiley Series in Adaptive and Learning Systems for Signal Processing, Communication, and Control
Simon Haykin, Series Editor

Neural-Based Orthogonal Data Fitting

The EXIN Neural Networks



Giansalvo Cirrincione
Maurizio Cirrincione

WILEY

NEURAL-BASED ORTHOGONAL DATA FITTING

Adaptive and Learning Systems for Signal Processing, Communication, and Control

Editor: Simon Haykin

A complete list of titles in this series appears at the end of this volume.

NEURAL-BASED ORTHOGONAL DATA FITTING

The EXIN Neural Networks

Giansalvo Cirrincione

Université de Picardie–Jules Verne
Amiens, France

Maurizio Cirrincione

Université de Technologie de Belfort–Montbéliard
Belfort, France

WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Cirrincione, Giansalvo

Neural-based orthogonal data fitting: the EXIN neural networks / Giansalvo Cirrincione, Maurizio Cirrincione

Includes bibliographic references and index.

ISBN 978-0-471-32270-2

Printed in Singapore

10 9 8 7 6 5 4 3 2 1

To our parents Concetta Maria and Paolo

our sister Nuccetta

and little Paolo

CONTENTS

FOREWORD

The history of data analysis can be divided into two main periods: before the 1970–1980 decades and after. Before the 1970–1980 decades it was mainly the domain of applied mathematics and statistics. During this period, researchers developed linear techniques (now defined as "classical") under various predefined algorithms with some successful approaches to relatively standard problems and data structures.

During the 1970–1980 decades, a new technique, often addressed to non-mathematicians, appeared; it is known under the generic name of *neural networks*. The idea arose from the understanding that our (biological) neural networks are able to perform in a very short time very complex tasks on difficult problems, without a logically derived algorithm and without an "if–then–else" reasoning scheme. These networks are known to be mainly nonlinear and also to be capable of learning from the data.

Although they were oversimplifications of biological models, the first mathematical models of neural networks have proved capable of dealing with complex problems. The community of researchers became more and more interested in these models, progressively establishing a completely new approach to data processing and analysis. First considered cautiously by mathematicians, this new technique gave rise to increasing interest from the mathematical and statistical communities.

In the past decades, many international conferences have been devoted to neural networks per se, progressively establishing the foundations of a new technique. Mathematicians who formerly considered this domain rather questionable have now adopted it, and we can say that today it qualifies as one of many classical statistical tools.

In this book the authors address the problem of linear multidimensional regression from a completely new point of view. First, they proceed to an in-depth theoretical analysis of the problem of regression and show that many researchers, using ordinary least squares, have not properly considered the presence of noise in their data, thus obtaining erroneous results. Based on the fact that noise may be present in columns as well as in lines of a data matrix, the authors address the problems of total least squares and minor components analysis.

The authors propose an in-depth and interesting theoretical analysis of the dynamics of iterative approaches from the point of view of convergence and stability. They use an original neural network called "EXIN" to achieve safe convergence of the iterative process. They have used this network and its extensions and generalizations in numerous applications in which it proves to be very efficient, even on large high-dimensional data matrices.

The authors are widely renowned in the field of data analysis by means of neural networks; they have published many theoretical and application papers in international journals and have presented many papers at international conferences. It is my pleasure to warmly recommend their book to students and researchers interested in orthogonal regression who are seeking extensive insight into the problem.

JEANNY HERAULT
Professor Emeritus
University of Grenoble, France

PREFACE

Neural-Based Orthogonal Regression

Modeling systems using linear regression models is done by researchers nearly automatically. Most often they use ordinary least squares (OLS) techniques, which they can easily find already implemented, without posing any question about the assumptions of this approach. This is due both to the simplicity of the method and to its robustness: Even if OLS is used in a nonoptimal way, the result is often still satisfying. Another reason is the fact that most researchers know neither the linear regression theory (they use OLS because everyone uses it!) nor the existence of alternative methods. The total least squares (TLS) approach is not as well known. That is why it has been rediscovered a number of times by each community of researchers, and as a consequence, it has a lot of names, such as orthogonal regression or the errors-in-variables method. Thus, what is the point of using TLS instead of OLS? It depends on the assumptions made on data used in the model: If all data are noisy, TLS yields better estimates than OLS. However, the TLS statistical assumptions are very strict and data preprocessing is often needed. Furthermore, TLS suffers from some important drawbacks:

- It is very computationally expensive with respect to the OLS approach.
- It yields inconsistent results in the presence of outliers in the data.
- It does not work for very noisy data or the close-to-singular data matrix (*nongeneric TLS*).

In the latter case, a numerical rank determination is needed to solve the problem.

This book is not oriented toward all TLS methods, only to some iterative ones which in the literature are defined as neural. Among them, the EXIN neurons (see Figure P.1), created by the authors of this book in recent years, are the main topic. Here they are introduced, explained, analyzed (convergence, stability, transient, dynamics, numerical considerations), and compared to the other neural approaches. The work is mainly theoretical; that is, it deals with the mathematical and numerical aspects of the EXIN neurons. The simulations proposed are, above all, an illustration of the theory. However, many applications created by the authors already exist and will be the subject of our next book.

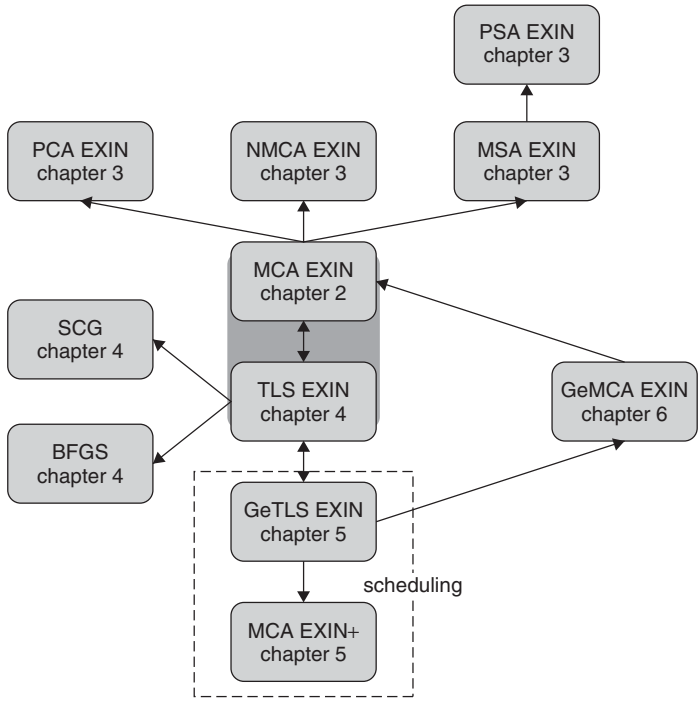


Figure P.1 Neural EXIN family.

In data analysis one very important linear technique for extracting information from data is *principal components analysis* (PCA). Here, the principal components are the directions in which the data have the largest variances and capture most of the information content of data. They correspond to the eigenvectors associated with the largest eigenvalues of the autocorrelation matrix of the data vectors. On the contrary, the eigenvectors that correspond to the smallest eigenvalues of the autocorrelation matrix of the data vectors are defined as the *minor components* and are the directions in which the data have the smallest variances (they represent the noise in the data). Expressing data vectors in terms of the minor components is called *minor components analysis* (MCA). MCA is often used to solve TLS problems, and there is a lot of confusion between them in the literature. In this book the equivalence between TLS and MCA is clearly stated by introducing the idea of a TLS hyperplane. For this reason we can speak of the neural EXIN family, where neurons for TLS can be used for MCA, and vice versa.

Neural EXIN Family

The most important neurons are the TLS EXIN and the MCA EXIN (see Figure P.1). The TLS EXIN is the first neuron created by the authors. The MCA

is an extension in a higher-dimensional space. However, from a theoretical point of view, the basic neural network is the MCA EXIN, and its theory is the basis of the book. The TLS EXIN can be derived as an MCA EXIN constrained on the TLS hyperplane. In Chapter 4 we show the derivation of the TLS EXIN from the MCA EXIN and the original demonstration (deterministic proof). From the MCA EXIN some neural networks are derived: the NMCA EXIN, which is a robust version of the MCA EXIN for dealing with outliers in the data; the MSA EXIN, for computing the minor subspace; and the corresponding tools for principal components analysis, the PCA EXIN and the PSA EXIN. They are all described in Chapter 3. The TLS EXIN can be accelerated when working in batch or block mode; the SCG and the BFGS EXIN are analyzed in Chapter 4. The GeTLS EXIN was born as an extension of the TLS EXIN to take into account the three regression problems: OLS, TLS, and DLS. In Chapter 5 we compare the GeTLS EXIN with other existing generalization techniques and demonstrate its superiority. However, the corresponding GeTLS EXIN theory comprises the TLS EXIN theory and yields a new insight into this problem. As a consequence, the TLS EXIN would better be considered as a particular case of the GeTLS EXIN, and a novel technique called scheduling is first justified and then exploited to improve the convergence to the regression solutions. The MCA EXIN+ uses this approach to improving the MCA EXIN. In Chapter 5 we describe the GeTLS EXIN and its scheduling. Despite this deep analysis, the meaning of the GeTLS EXIN for cases of regression outside OLS, TLS, and DLS is not taken into account because no error cost in form of a Rayleigh quotient is identified. This problem has been solved by introducing the GeMCA theory (see Chapter 6), which identifies the GeTLS problem as a generalized eigenvalue problem. As a consequence, GeTLS EXIN can be replaced by a novel MCA EXIN neuron in which data are scaled to be dependent on the same parameter as that used in GeTLS EXIN. However, it is important to note that the entire MCA EXIN theory can be used to explain the intermediate case of GeTLS EXIN.

To summarize: The theory shows a circular trajectory. It begins at MCA EXIN and arrives at the same neural network, but with a deeper insight into the approach. In this sense, we can say that in this book we propose a complete theory about orthogonal regression.

Applications of the EXIN Family

Many applications of the EXIN algorithms have been created by the authors. They deal with the following topics.

- *Computer vision:* TLS and CTLS (constrained TLS) for estimation of the essential/fundamental parameters in the structure from motion [25,26,32] and their use in CASEDEL EXIN, a method that deals with outliers in data [29,34]
- *Linear system identification:* described in detail in [36] and in Section 4.5

- *Sensorless control of induction machines by a neural speed observer:* TLS EXIN [38,39,41,44–46] and MCA EXIN+ [40,47]
- *Estimation of the induction motor parameters by TLS EXIN* (a nongeneric TLS problem) [37,42,43]

However, the description of these methods is outside the scope of the book and will be the subject of our next book.

Outline of the Book

Chapter 1 TLS problems are shown and a sketch of their history and applications is given. Most formulas are presented because they are used in later chapters. This is mainly an introduction to problems addressed in other chapters.

Chapter 2 The fundamental MCA theory is proposed. MCA EXIN is introduced and compared with other approaches. It is one of the most important chapters because its theory is also used for GeTLS EXIN. One of the most important results found by the authors is the identification of three possible divergences (sudden, dynamical, and numerical) in the neural MCA algorithms.

Chapter 3 This chapter deals with the extensions of MCA EXIN: in particular, NMCA EXIN is used when data are corrupted by outliers, and MSA EXIN is used to estimate the minor subspace.

Chapter 4 The TLS EXIN neuron and the SCG and BFGS acceleration techniques are introduced and compared with the only existing neuron, TLS GAO. Geometrical analysis of stability properties is also given.

Chapter 5 This is one of the most important chapters because the GeTLS EXIN theory for generalizing and unifying the regression problems is covered. This theory not only explains the neural behavior, but justifies a particular scheduling of the parameter, which is the basis of MCA EXIN+ theory. Very important stability theorems are given whose main result justifies the use of null initial conditions for guaranteeing the convergence of TLS EXIN. This choice also implies automatic computation of the TLS solution, even in the case of nongeneric TLS, which arises when the linear equations are highly conflicting.

Chapter 6 This chapter presents GeMCA theory, starting with the identification of GeTLS EXIN as a generalized eigenvalue problem. It is mainly theoretical and shows some important inequalities for intermediate values of the GeTLS EXIN parameter. This analysis consolidates previous theories and concludes the book by showing its link with MCA EXIN theory.

Potentiality

If fast and accurate techniques exist, the possible applications of TLS are much greater than existing applications. For example, in [24] it is shown that many linear techniques for solving the structure-from-motion problem have been abandoned because researchers have not understood that they were dealing with TLS problems. The possibility of using online TLS techniques can also lead to very important implementations, as in signal processing (e.g., harmonics retrieval), control, and system identification.

Originality of Our Approach

The theory proposed in this book studies the algorithms from a variety of points of view:

- As a differential geometry problem: The neural algorithms are equivalent outside their critical points but depend on the Riemannian metric at convergence, which allows a common framework for all the algorithms
- As a dynamic problem, which is solved by using the average approximating differential equation (the unique point of view of the existing theory)
- As a stochastic problem, which allows us to discover very important phenomena, such as sudden divergence
- As a numerical problem, which allows an analysis of the cost per iteration and of the numerical stability of the algorithms

As anticipated, this theory has uncovered new phenomena and novel aspects that are unknown in the literature. In particular, some of the main features are the following:

- Four different types of divergence discovered in every neural algorithm except our neural networks and their extension to the PCA algorithms
- Complete transient analysis endowed by a bias/variance study
- Introduction and analysis of a domain of convergence
- Automatic (i.e., without the need for a numerical rank evaluation) solution of the nongeneric TLS problem by TLS EXIN
- Loss of the step phenomenon in case of computation of more than one minor component
- Locus of attraction for the weights in order to speed up the neural algorithms
- Equivalence GeTLS EXIN MCA EXIN by means of the GeMCA theory

The algorithms proposed have excellent properties and allow interesting applications. In particular:

- In computer vision, TLS EXIN (in a constrained version [24] not explained here because it is outside the scope of the book) is able to yield correct solutions even if half of the input data are completely wrong.
- There exists an entire neural architecture for solving the structure-from-motion problem in computer vision [24], composed of neural networks that can easily be implemented in hardware for real-time use.
- These algorithms are very robust in case of colored noise and outliers.
- It is possible to solve the identification problem in real time.

In the literature, all existing neural techniques are tested on toy problems (four-dimensional data matrices at most). We have analyzed all the algorithms on real-time problems (large-dimensional data matrices) and shown that only our algorithms are able to yield an accurate solution (all other algorithms diverge!). We have also shown the superiority of our methods with respect to the existing nonneural techniques.

To Whom This Book Is Addressed

The high potentiality of the TLS approach and the possibility of using very reliable, accurate, and fast neural algorithms can be very attractive for researchers in many fields. In particular, the book we propose can be useful to:

- Statisticians, because of the new tools in data analysis
- Applied mathematics experts, because of the novel theories and techniques that we propose
- Engineers, above all for the applications in control, system identification, computer vision, and signal processing

How to Use the Book

This book was not conceived as a vulgarization of the current state of the art about orthogonal regression, but to present a novel theory for it. It is therefore suitable for a graduate-level course supplement or as a complement for a postgraduate course in statistics or numerical analysis dealing with regression. The book is useful for both the theoretician and the practitioner. The theoretician can find a complete theory explaining not only the EXIN approach but also the general regression problems; it is difficult to decide which sections can be skipped, but if he or she is only interested in MCA, only Chapter 3 need be read.

The practitioner can skip most theory and just read the theorems without their proofs; he or she will then find all of the algorithms that may be needed. In particular, the following sections should be read:

Chapter 1 Sections 1.1 and 1.2 for a first idea on TLS; Sections 1.4, 1.5, and 1.11 for the basic formulas of the three regression problems and Section 1.7

for the nongeneric TLS which often happens in applications; Section 1.13 for an overview of classical methods.

Chapter 2 The first two sections for the first ideas on MCA, Section 2.3 for the MCA EXIN neuron; Theorem 60 about its convergence and Section 2.6 for its dynamics (particularly important Section 2.6.2 about divergence); Section 2.8 on numerical considerations; and the last three sections regarding how MCA EXIN works.

Chapter 3 Section 3.2 if experimental data are corrupted by noise (NMCA EXIN); if the case, Section 3.3.

Chapter 4 The entire chapter except Sections 4.7 and 4.8.

Chapter 5 Section 5.1 for the weighted TLS; Section 5.2 for introducing GeTLS EXIN; Section 5.3.3 for important considerations and a benchmark problem; the fundamental theorem expressed by Theorem 111; nongeneric TLS illustrated in Section 5.4; Section 5.5 for the scheduling in GeTLS EXIN, with the important particular case MCA EXIN+ in Section 5.6; the considerations of Section 5.7; the simulations of Section 5.8.

Chapter 6 Skip, because it is basically a theoretical chapter.

Prerequisites

Readers should be familiar with basic linear algebra and numerical analysis as well as the fundamentals of statistics, such as the basics of least squares, and preferably, but not necessarily, stochastics algorithms (to read Chapter 3). Some notions about differential geometry could be of help but are not strictly necessary.

Although the book is focused on neural networks, these are presented only by their learning law, which is simply an iterative algorithm: therefore, no a priori knowledge of neural networks is required.

Acknowledgments

We are very grateful to Angelo Accetta, Ph.D. student at the University of Palermo, for his help in formatting the manuscript in LaTeX.

We are very grateful to Professor Jeanny Hérault at the University Joseph Fourier of Grenoble (France), who has been hugely supportive throughout the Ph.D. years of the first author, and to Professor Sabine Van Huffel at the Katholieke Universiteit Leuven (Belgium), thanks to whom the first author has had the chance to improve his theoretical background during his postdoctoral experience in Leuven.

We also express our gratitude to the CNR (Italian National Reseach Centre)–ISSIA (Institute of Studies on Intelligent Systems for Automation) of Palermo, Italy, particularly Dr. Marcello Pucci, for institutional support in encouraging joint work in a very creative environment.

Finally, we would like to thank Aaron Nimzowitsch, the famous chess master, whose methodology in his work has inspired ours.

GIANSALVO CIRRINCIONE
MAURIZIO CIRRINCIONE

1

TOTAL LEAST SQUARES PROBLEMS

1.1 INTRODUCTION

The problem of *linear parameter estimation* gives rise to an overdetermined set of linear equations $Ax \approx b$, where A is the *data matrix* and b is the *observation vector*. In the (classical) least squares (LS) approach there is the underlying assumption that all errors are confined to the observation vector. This assumption is often unrealistic: The data matrix is not error-free because of sampling errors, human errors, modeling errors, and instrument errors. Methods for estimating the effect of such errors on the LS solution are given in [90] and [177]. The method of *total least squares* (TLS) is a technique devised to compensate for data errors. It was introduced in [74], where it has been solved by using *singular value decomposition* (SVD), as pointed out in [76] and more fully in [71]. Geometrical analysis of SVD brought Staar [176] to the same idea. This method of fitting has a long history in the statistical literature, where the method is known as *orthogonal regression* or *errors-in-variables* (EIV)[1] regression. Indeed, the univariate line-fitting problem had been considered in the nineteenth century [3]. Some important contributors are Pearson [151], Koopmans [108], Madansky [126], and York [197]. About 40 years ago, the technique was extended to multivariate problems and later to multidimensional problems (they deal with more than one observation vector b; e.g., [70,175]).

[1]EIV models are characterized by the fact that the true values of the observed variables satisfy unknown but exact linear relations.

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

A complete analysis of the TLS problems can be found in [98], where the algorithm of [74] is generalized to all cases in which it fails to produce a solution (*nongeneric TLS*). Most of the following theoretical presentation of the TLS problems is based on [98].

1.2 SOME TLS APPLICATIONS

There are a lot of TLS applications in many fields:

- *Time-domain system identification and parameter estimation*. This includes deconvolution techniques: for example, renography [100], transfer function models [57], estimates of the autoregressive parameters of an ARMA model from noisy measurements [179], and structural identification [8].
- *Identification of state-space models from noisy input-output measurements*. Examples, including the identification of an industrial plant, can be found in [134] and [135].
- *Signal processing*. A lot of algorithms have been proposed for the *harmonic retrieval* problem: the Pisarenko harmonic decomposition [152], the linear prediction–based work of Rahman and Yu [158], the ESPRIT (estimation of signal parameters via rotational invariance techniques) algorithm of Roy and Kailath [163], and the Procrustes rotations–based ESPRIT algorithm proposed by Zoltowski and Stavrinides [202]. Zoltowski [201] also applied TLS to the minimum variance distortionless response (MVDR) beamforming problem.
- *Biomedical signal processing*. This includes signal parameter estimates in the accurate quantification of in vivo magnetic resonance spectroscopy (MRS) and the quantification of chromophore concentration changes in neonatal brain [94].
- *Image processing*. This includes the image reconstruction algorithm for computing images of the interior structure of highly scattering media (optical tomography) by using the conjugate gradient method [200], a method for removing noise from digital images corrupted with additive, multiplicative, and mixed noise [87], and the regularized constrained TLS method for restoring an image distorted by a linear space-invariant point-spread function which is not exactly known [130].
- *Computer vision*. This includes disparity-assisted stereo optical flow estimation [102] and robust and reliable motion analysis [24,137].
- *Experimental modal analysis*. Estimates of the frequency response functions from measured input forces and response signals are applied to mechanical structures [162].
- *Nonlinear models*. The true variables are related to each other nonlinearly (see [73]).

- *Acoustic radiation problems, geology, inverse scattering, geophysical tomographic imaging, and so on* See, respectively, [79,58,171,104].
- *Environmental sciences*. This includes the fitting of Horton's infiltration curve using real data from field experiments and linear and nonlinear rainfall-runoff modeling using real data from watersheds in Louisiana and Belgium [159].
- *Astronomy*. This includes the determination of preliminary orbit in celestial mechanics [19], its differential correction, and the estimation of various parameters of galactic kinematics [12].

1.3 PRELIMINARIES

The TLS method solves a set of m linear equations in $n \times d$ unknowns X represented in matrix form by

$$AX \approx B \tag{1.1}$$

where A is the $m \times n$ *data matrix* and B is the $m \times d$ *observation matrix*. If $m > n$, the system is *overdetermined*. If $d > 1$, the problem is *multidimensional*. If $n > 1$, the problem is *multidivariate*. X' is the $n \times d$ *minimum norm* least squares solution and $\hat{X}$ is the *minimum norm* total least squares solution of eq. (1.1).

Singular value decomposition (SVD) of the matrix A $(m > n)$ in eq. (1.1) is denoted by

$$A = U'\Sigma'V'^T \tag{1.2}$$

where

$$U' = \left[U'_1; U'_2\right], U'_1 = \left[u'_1, \ldots, u'_n\right], U'_2 = \left[u'_{n+1}, \ldots, u'_m\right],$$
$$u'_i \in \Re^m, U'^T U' = U'U'^T = I_m$$
$$V' = \left[v'_1, \ldots, v'_n\right], v'_i \in \Re^n, V'^T V' = V'V'^T = I_n$$
$$\Sigma' = \operatorname{diag}\left(\sigma'_1, \ldots, \sigma'_n\right) \in \Re^{m\times n}, \sigma'_1 \geq \cdots \geq \sigma'_n \geq 0$$

and the SVD of the $m \times (n + d)$ matrix $[A; B]$ $(m > n)$ in eq. (1.1) is denoted by

$$[A; B] = U\Sigma V^T \tag{1.3}$$

where

$$U = [U_1; U_2], U_1 = [u_1, \ldots, u_n], U_2 = \left[u_{n+1}, \ldots, u_m\right],$$
$$u_i \in \Re^m, U^T U = UU^T = I_m$$

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{matrix} n \\ d \end{matrix} = \left[v_1, \ldots, v_{n+d}\right], v_i \in \Re^{n+d}, V^T V = VV^T = I_{n+d}$$
$$\begin{matrix} n & d \end{matrix}$$

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} = \text{diag}\,(\sigma_1, \ldots, \sigma_{n+t}) \in \Re^{m\times(n+d)},$$
$$t = \min\{m-n, d\}, \qquad \Sigma_1 = \text{diag}\,(\sigma_1, \ldots, \sigma_n) \in \Re^{n\times n},$$
$$\Sigma_2 = \text{diag}\,(\sigma_{n+1}, \ldots, \sigma_{n+t}) \in \Re^{(m-n)\times d}, \qquad \sigma_1 \geq \cdots \geq \sigma_{n+t} \geq 0$$

For convenience of notation, $\sigma_i = 0$ if $m < i \leq n + d$. $\left(u_i', \sigma_i', v_i'\right)$ and (u_i, σ_i, v_i) are, respectively, the *singular triplet* of A and $[A; B]$.

Most of the book is devoted to the unidimensional ($d = 1$) case; that is,

$$Ax = b \tag{1.4}$$

where $A \in \Re^{m\times n}$ and $b \in \Re^m$. However in Section 1.6 we deal with the multidimensional case. The problem is defined as *basic* if (1) it is unidimensional, (2) it is solvable, and (3) it has a unique solution.

1.4 ORDINARY LEAST SQUARES PROBLEMS

Definition 1 (OLS Problem) *Given the overdetermined system (1.4), the least squares (LS) problem searches for*

$$\min_{b'\in\Re^m} \left\|b - b'\right\|_2 \qquad \text{subject to} \quad b' \in R(A) \tag{1.5}$$

where $R(A)$ is the column space of A. Once a minimizing b' is found, then any x' satisfying

$$Ax' = b' \tag{1.6}$$

is called an LS solution (*the corresponding LS correction is $\Delta b' = b - b'$*).

Remark 2 *Equations (1.5) and (1.6) are satisfied if b' is the orthogonal projection of b into $R(A)$.*

Theorem 3 (Closed-Form OLS Solution) *If rank(A)= n, eqs. (1.5) and (1.6) are satisfied for the unique LS solution given by*

$$x' = (A^T A)^{-1} A^T b = A^+ b \tag{1.7}$$

(*for an underdetermined system, this is also the minimal L_2 norm solution*).

Matrix A^+ is called the *Moore–Penrose pseudoinverse*. The underlying assumption is that errors occur *only* in the observation vector and that the data matrix is known *exactly*.

The OLS solution can be computed as the minimizer of the following error function:

$$E_{\text{OLS}} = \tfrac{1}{2}(Ax - b)^T(Ax - b) \tag{1.8}$$

1.5 BASIC TLS PROBLEM

Definition 4 (Basic TLS Problem) *Given the overdetermined set (1.4), the total least squares (TLS) problem searches for*

$$\min_{[\hat{A};\hat{b}]\in\Re^{m\times(n+1)}} \left\|[A; b] - [\hat{A}; \hat{b}]\right\|_F \qquad \text{subject to} \quad \hat{b} \in R(\hat{A}) \tag{1.9}$$

where $\|\cdots\|_F$ is the Frobenius norm. Once a minimizing $[\hat{A}; \hat{b}]$ is found, then any $\hat{x}$ satisfying

$$\hat{A}\hat{x} = \hat{b} \tag{1.10}$$

is called a TLS solution (*the corresponding TLS correction is* $[\Delta\hat{A}; \Delta\hat{b}] = [A; b] - [\hat{A}; \hat{b}]$).

Theorem 5 (Solution of the Basic TLS Problem) *Given eq. (1.2) [respectively, (1.3)] as the SVD of A (respectively, $[A; b]$), if $\sigma'_n > \sigma_{n+1}$, then*

$$[\hat{A}; \hat{b}] = U\hat{\Sigma}V^T \qquad \text{where} \quad \hat{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_n, 0) \tag{1.11}$$

with corresponding TLS correction matrix

$$[\Delta\hat{A}; \Delta\hat{b}] = \sigma_{n+1}u_{n+1}v_{n+1}^T \tag{1.12}$$

solves the TLS problem (1.9) and

$$\hat{x} = -\frac{1}{v_{n+1,n+1}}\left[v_{1,n+1}, \ldots, v_{n,n+1}\right]^T \tag{1.13}$$

exists and is the unique solution to $\hat{A}\hat{x} = \hat{b}$.

Proof. (An outline: for the complete proof, see [98].) Recast eq. (1.4) as

$$[A; b][x^T; -1]^T \approx 0 \tag{1.14}$$

If $\sigma_{n+1} \neq 0$, $\text{rank}[A; b] = n + 1$; then no nonzero vector exists in the orthogonal complement of $R_r([A; b])$, where $R_r(T)$ is the row space of matrix T. To reduce to n the rank, using the Eckart–Young–Mirsky matrix approximation theorem (see [54,131]), the best rank n TLS approximation $[\hat{A}; \hat{b}]$ of $[A; b]$, which minimizes the deviations in variance, is given by (1.11). The minimal TLS correction is then

$$\sigma_{n+1} = \min_{\text{rank}([\hat{A};\hat{b}])=n} \left\| [A; b] - [\hat{A}; \hat{b}] \right\|_F \tag{1.15}$$

and is attained for the rank 1 TLS correction (1.12). Then the approximate set $[\hat{A}; \hat{b}][x^T; -1]^T \approx 0$ is now compatible and its solution is given by the only vector v_{n+1} that belongs to the kernel of $[\hat{A}; \hat{b}]$. The TLS solution is then obtained by scaling v_{n+1} until its last component is -1. ■

Proposition 6 *The interlacing theorem for singular values* (*see* [*182*]) *implies that*

$$\sigma_1 \geq \sigma_1' \geq \cdots \geq \sigma_n \geq \sigma_n' \geq \sigma_{n+1} \tag{1.16}$$

Proposition 7 *As proved in* [*98, Cor. 3.4*],

$$\sigma_n' > \sigma_{n+1} \Longleftrightarrow \sigma_n > \sigma_{n+1} \quad \text{and} \quad v_{n+1,n+1} \neq 0$$

Remark 8 *If* $\sigma_{n+1} = 0$, $\text{rank}[A; b] = n \Longrightarrow$ (*1.14*) *is compatible and no approximation is needed to obtain the exact solution* (*1.13*).

The TLS solution is obtained by finding the *closest subspace* $R\left([\hat{A}; \hat{b}]\right)$ to the $n + 1$ columns of $[A; b]$ such that the sum of the squared perpendicular distances from each column of $[A; b]$ to $R\left([\hat{A}; \hat{b}]\right)$ is minimized, and each column is approximated by its orthogonal projection onto that subspace.

Theorem 9 (Closed-Form Basic TLS Solution) *Given* (*1.2*) [*respectively,* (*1.3*)] *as the SVD of A* (*respectively,* $[A; b]$), *if* $\sigma_n' > \sigma_{n+1}$, *then*

$$\hat{x} = \left(A^T A - \sigma_{n+1}^2 I\right)^{-1} A^T b \tag{1.17}$$

Proof. The condition $\sigma_n' > \sigma_{n+1}$ assures the existence and the uniqueness of the solution (see Proposition 7). Since the singular vectors v_i are eigenvectors of $[A; b]^T[A; b]$, $\hat{x}$ satisfies the following set:

$$[A; b]^T[A; b] \begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} = \begin{bmatrix} A^T A & A^T b \\ b^T A & b^T b \end{bmatrix} \begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} = \sigma_{n+1}^2 \begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} \tag{1.18}$$

Formula (1.17) derives from the top part of (1.18). ■

Remark 10 (*See* [*74*].) *The ridge regression is a way of regularizing the solution of an ill-conditioned LS problem* [*114, pp. 190ff.*]*; for example, the minimization of* $\|b - Ax\|_2^2 + \mu \|x\|_2^2$*, where* μ *is a positive scalar, is solved by* $x_{\mathrm{LS}}(\mu) = \left(A^T A + \mu I\right)^{-1} A^T b$*, and* $\|x_{\mathrm{LS}}(\mu)\|_2$ *becomes small as* μ *becomes large. But* $x_{\mathrm{TLS}} = x_{\mathrm{LS}}(-\sigma_{n+1}^2)$*, which implies that the TLS solution is a deregularizing procedure, a reverse ridge regression. It implies that the condition of the TLS problem is always worse than that of the corresponding LS problem.*

Remark 11 *Transforming* (*1.18*) *as*

$$\begin{bmatrix} \Sigma'^T \Sigma' & g \\ g^T & \|b\|_2^2 \end{bmatrix} \begin{bmatrix} z \\ -1 \end{bmatrix} = \sigma_{n+1}^2 \begin{bmatrix} z \\ -1 \end{bmatrix} \quad \text{with} \quad g = \Sigma'^T U'^T b,\ z = V'^T \hat{x} \tag{1.19}$$

Then $\left(\Sigma'^T \Sigma' - \sigma_{n+1}^2 I\right) z = g$ *and* $\sigma_{n+1}^2 + g^T z = \|b\|_2^2$*. Substituting* z *in the latter expression by the former yields*

$$\sigma_{n+1}^2 + g^T \left(\Sigma'^T \Sigma' - \sigma_{n+1}^2 I\right)^{-1} g = \|b\|_2^2 \tag{1.20}$$

This is a version of the TLS secular equation [*74,98*].

Remark 12 *If* $v_{n+1,n+1} \neq 0$*, the TLS problem is solvable and is then called generic.*

Remark 13 *If* $\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}$*, any vector in the space created by the right singular vectors associated with the smallest singular vector is a solution of the TLS problem* (*1.9*)*; the same happens in the case* $m < n$ (*underdetermined system*)*, since then the conditions* $\sigma_{m+1} = \cdots = \sigma_{n+1} = 0$ *hold.*

Remark 14 *The TLS correction* $\left\|[A; b] - [\hat{A}; \hat{b}]\right\|_F$ *is always smaller in norm than the LS correction* $\left\|b - b'\right\|_2$*.*

1.5.1 OLS and TLS Geometric Considerations (Row Space)

The differences between TLS and OLS are considered from a geometric point of view in the row space R_r ($[A; b]$) (see Figure 1.1 for the case $n = 2$).

If no errors are present in the data (EIV model), the set $Ax \approx b$ is compatible, $\mathrm{rank}[A; b] = n$, and the space R_r ($[A; b]$) is n-dimensional (hyperplane). If errors occur, the set is no longer compatible and the rows $r_1, r_2, \ldots, r_m$ are scattered around the hyperplane. The normal of the hyperplane, corresponding to the minor component of $[A; b]$, gives the corresponding solution as its intersection with the hyperplane $x_{n+1} = -1$.

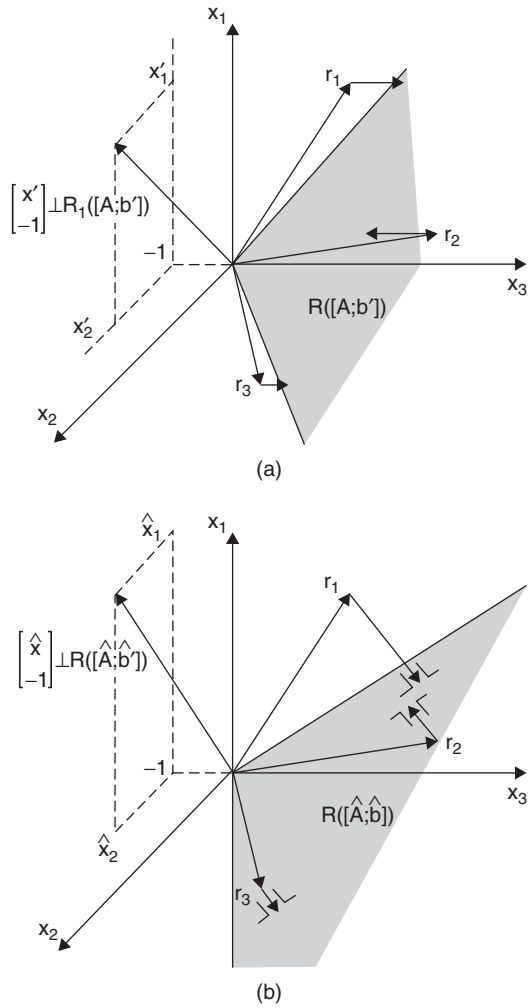


Figure 1.1 Geometry of the LS solution x' (a) and of the TLS solution $\hat{x}$ (b) for $n = 2$. Part (b) shows the TLS hyperplane.

Definition 15 *The TLS hyperplane is the hyperplane* $x_{n+1} = -1$.

The LS approach [for $n = 2$, see Figure 1.1(a)] looks for the best approximation b' to b satisfying (1.5) such that the space $R_r([A; b'])$ generated by the LS approximation is a hyperplane. Only the *last* components of $r_1, r_2, \ldots, r_m$ can vary. This approach assumes random errors along *one* coordinate axis only. The TLS approach [for $n = 2$, see Figure 1.1(b)] looks for a hyperplane $R_r([\hat{A}; \hat{b}])$ such that (1.9) will be satisfied. The data changes are not restricted to being along *one* coordinate axis x_{n+1}. All correction vectors Δr_i given by the rows of

$\left[\Delta\hat{A};\, \Delta\hat{b}\right]$ are *parallel* with the solution vector $\left[\hat{x}^T; -1\right]^T$ [it follows from (1.12) and (1.13)]. The TLS solution is parallel to the right singular vector corresponding to the minimum singular value of $[A; b]$, which can be expressed as a Rayleigh quotient (see Section 2.1):

$$\sigma_{n+1}^2 = \frac{\left\| [A; b] \left[\hat{x}^T; -1\right]^T \right\|_2^2}{\left\| \left[\hat{x}^T; -1\right]^T \right\|_2^2} = \frac{\sum_{i=1}^m \left| a_i^T \hat{x} - b_i \right|^2}{1 + \hat{x}^T \hat{x}} = \sum_{i=1}^m \| \Delta r_i \|_2^2 \tag{1.21}$$

with a_i^T the ith row of A and $\| \Delta r_i \|_2^2$ the square of the distance from $\left[a_i^T; b_i\right]^T \in \Re^{n+1}$ to the nearest point in the subspace:

$$R_r([\hat{A}; \hat{b}]) = \left\{ \begin{bmatrix} \tilde{a} \\ \tilde{b} \end{bmatrix} \middle| \tilde{a} \in \Re^n, \tilde{b} \in \Re,\ \tilde{b} = \hat{x}^T \tilde{a} \right\}$$

Thus, the TLS solution $\hat{x}$ minimizes the sum of the squares of the orthogonal distances (weighted squared residuals):

$$E_{\mathrm{TLS}}(x) = \frac{\sum_{i=1}^m |a_i^T x - b_i|^2}{1 + x^T x} \tag{1.22}$$

which is the Rayleigh quotient (see Section 2.1) of $[A; b]^T [A; b]$ constrained to $x_{n+1} = -1$. It can also be rewritten as

$$E_{\mathrm{TLS}}(x) = \frac{(Ax - b)^T (Ax - b)}{1 + x^T x} \tag{1.23}$$

This formulation is very important from the neural point of view because it can be considered as the energy function to be minimized for the training of a neural network whose final weights represent the TLS solution. This is the basic idea of the TLS EXIN neuron.

1.6 MULTIDIMENSIONAL TLS PROBLEM

1.6.1 Unique Solution

Definition 16 (Multidimensional TLS Problem) *Given the overdetermined set (1.1), the total least squares (TLS) problem searches for*

$$\min_{[\hat{A};\hat{B}] \in \Re^{m \times (n+d)}} \left\| [A; B] - [\hat{A}; \hat{B}] \right\|_F \qquad \text{subject to} \quad R(\hat{B}) \in R(\hat{A}) \tag{1.24}$$

Once a minimizing $[\hat{A}; \hat{B}]$ *is found, then any* $\hat{X}$ *satisfying*

$$\hat{A}\hat{X} = \hat{B} \tag{1.25}$$

is called a TLS solution (*the corresponding TLS correction is* $[\Delta\hat{A}; \Delta\hat{B}] = [A; B] - [\hat{A}; \hat{B}]$).

Theorem 17 (Solution of the Multidimensional TLS Problem) *Given* (*1.3*) *as the SVD of* $[A; B]$, *if* $\sigma_n > \sigma_{n+1}$, *then*

$$[\hat{A}; \hat{B}] = U \operatorname{diag}(\sigma_1, \ldots, \sigma_n, 0, \ldots, 0)\, V^T = U_1 \Sigma_1 \left[V_{11}^T; V_{21}^T\right] \tag{1.26}$$

with corresponding TLS correction matrix

$$[\Delta\hat{A}; \Delta\hat{B}] = U_2 \Sigma_2 \left[V_{12}^T; V_{22}^T\right] \tag{1.27}$$

solves the TLS problem (*1.24*) *and*

$$\hat{X} = -V_{12} V_{22}^{-1} \tag{1.28}$$

exists and is the unique solution to $\hat{A}\hat{X} = \hat{B}$.

Proof. See [98, p. 52]. ■

Theorem 18 (Closed-Form Multidimensional TLS Solution) *Given* (*1.2*) [*respectively,* (*1.3*)] *as the SVD of* $A([A; B])$, *if* $\sigma_n' > \sigma_{n+1} = \cdots = \sigma_{n+d}$, *then*

$$\hat{X} = \left(A^T A - \sigma_{n+1}^2 I\right)^{-1} A^T B \tag{1.29}$$

Proof. See [98, Th. 3.10]. ■

Proposition 19 (Existence and Uniqueness Condition) *See* [*98, p. 53*]*:*

$$\sigma_n' > \sigma_{n+1} \Longrightarrow \sigma_n > \sigma_{n+1} \quad \text{and} \quad V_{22} \text{ is nonsingular.}$$

Remark 20 *The multidimensional problem* $AX \approx B_{m\times d}$ *can also be solved by computing the TLS solution of each subproblem* $Ax_i \approx b_i$, $i = 1, \ldots, d$, *separately. The multidimensional TLS solution is better at least when all data are equally perturbed and all subproblems* $Ax_i \approx b_i$ *have the same degree of incompatibility.*

1.6.2 Nonunique Solution

Theorem 21 (Closed-Form Minimum Norm TLS Solution) *Given the TLS problem* (*1.24*) *and* (*1.25*), *assuming that* $\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+d}$ *with* $p \le n$,

let (*1.3*) *be the SVD of* $[A; B]$*, partitioning* V *as*

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{matrix} n \\ d \end{matrix}$$
$$\begin{matrix} p & n-p+d \end{matrix}$$

If V_{22} *is of full rank, the multidimensional minimum norm* (*minimum 2-norm and minimum Frobenius norm*) *TLS solution* $\hat{X}$ *is given by*

$$\hat{X} = -V_{12}V_{22}^{+} = \left(A^T A - \sigma_{n+1}^2 I\right)^{+} A^T B \tag{1.30}$$

Proof. See [98, p. 62]. ■

1.7 NONGENERIC UNIDIMENSIONAL TLS PROBLEM

The TLS problem (1.9) fails to have a solution if $\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}$, $p \leq n$ and all $v_{n+1,i} = 0$, $i = p+1, \ldots, n+1$ (nongeneric TLS problem). In this case the existence conditions of the preceding theorems are not satisfied. These problems occur whenever A is *rank-deficient* ($\sigma_p' \approx 0$) or when the set of equations is *highly conflicting* ($\sigma_p' \approx \sigma_{p+1}$ large). The latter situation is detected by inspecting the size of the smallest σ_i for which $v_{n+1,i} \neq 0$. If this singular value is large, the user can simply reject the problem as irrelevant from a linear modeling point of view. Alternatively, the problem can be made generic by adding more equations. This is the case if the model is EIV and the observation errors are statistically independent and equally sized (same variance). For $\sigma_p' \approx 0$, it is possible to remove the dependency among the columns of A by removing appropriate columns in A such that the remaining submatrix has full rank and then apply TLS to the reduced problem (*subset selection*, [88,89,95]). Although exact nongeneric TLS problems seldom occur, close-to-nongeneric TLS problems are not uncommon. The generic TLS solution can still be computed, but it is unstable and becomes even very sensitive to data errors when $\sigma_p' - \sigma_{p+1}$ is very close to zero [74].

Theorem 22 (Properties of Nongeneric Unidimensional TLS) (*See* [*98*].) *Let* (*1.2*) [*respectively,* (*1.3*)] *be the SVD of* A (*respectively,* $[A; b]$)*; let* b' *be the orthogonal projection of* b *onto* $R(A)$ *and* $[\hat{A}; \hat{b}]$ *the rank* n *approximation of* $[A; b]$*, as given by* (*1.11*)*. If* $V'(\sigma_j)$ [*respectively,* $U'(\sigma_j)$] *is the right* (*respectively, left*) *singular subspace of* A *associated with* σ_j*, then the following relations can be proven:*

$$v_{n+1,j} = 0 \Longleftrightarrow v_j = \begin{bmatrix} v' \\ 0 \end{bmatrix} \quad \text{with} \quad v' \in V'(\sigma_j) \tag{1.31}$$

$$v_{n+1,j} = 0 \Longrightarrow \sigma_j = \sigma_k' \quad \text{with} \quad k = j-1 \quad \text{or} \quad k = j \text{ and } 1 \leq k \leq n \tag{1.32}$$

$$v_{n+1,j} = 0 \Longrightarrow b \perp u' \qquad \text{with} \quad u' \in U'(\sigma_j) \tag{1.33}$$

$$v_{n+1,j} = 0 \Longleftrightarrow u_j = u' \qquad \text{with} \quad u' \in U'(\sigma_j) \tag{1.34}$$

$$v_{n+1,j} = 0 \Longrightarrow b' \perp u' \qquad \text{with} \quad u' \in U'(\sigma_j) \tag{1.35}$$

$$v_{n+1,j} = 0 \Longrightarrow \hat{b} \perp u' \qquad \text{with} \quad u' \in U'(\sigma_j) \tag{1.36}$$

If σ_j is an isolated singular value, the converse of relations (1.33) and (1.35) also holds.

Proof. See [92, Th. 1–3]. ■

Corollary 23 (*See* [98].) *If $\sigma_n > \sigma_{n+1}$ and $v_{n+1,n+1} = 0$, then*

$$\sigma_{n+1} = \sigma'_n, \quad u_{n+1} = \pm u'_n, \quad v_{n+1} = \begin{bmatrix} \pm v'_n \\ 0 \end{bmatrix}, \quad b, b', \hat{b} \perp u'_n \tag{1.37}$$

The generic TLS approximation and corresponding TLS correction matrix minimizes $\left\| \Delta\hat{A}; \Delta\hat{b} \right\|_F$ but does not satisfy the constraint $\hat{b} \in R(\hat{A})$ and therefore does not solve the TLS problem (1.9). Moreover, (1.37) yields

$$\left[\hat{A}; \hat{b}\right] v_{n+1} = 0 \Longrightarrow \hat{A} v'_n = 0 \tag{1.38}$$

Then the solution v'_n describes an approximate linear relation among the columns of A instead of estimating the desired linear relation between A and b. The singular vector v_{n+1} is called a nonpredictive multicollinearity in linear regression since it reveals multicollinearities in A that are of no (or negligible) value in predicting the response b. Also, since $b \perp u' = u_{n+1}$, there is no correlation between A and b in the direction of $u' = u_{n+1}$. The strategy of nongeneric TLS is to eliminate those directions in A that are not at all correlated with the observation vector b; the additional constraint $\left[\begin{smallmatrix} \hat{x} \\ -1 \end{smallmatrix}\right] \perp v_{n+1}$ is then introduced. *Latent root regression* [190] uses the same constraint in order to stabilize the LS solution in the presence of multicollinearities. Generalizing:

Definition 24 (Nongeneric Unidimensional TLS Problem) *Given the set (1.4), let (1.3) be the SVD of $[A; b]$; the nongeneric TLS problem searches for*

$$\begin{aligned} &\min_{[\hat{A};\hat{b}] \in \Re^{m \times (n+1)}} \left\| [A; b] - [\hat{A}; \hat{b}] \right\|_F \qquad \text{subject to} \quad \hat{b} \in R(\hat{A}) \quad \text{and} \\ &\begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} \perp v_j, \; j = p+1, \ldots, n+1 \qquad (\text{provided that } v_{n+1,p} \neq 0) \end{aligned} \tag{1.39}$$

Once a minimizing $[\hat{A}; \hat{b}]$ is found, then any $\hat{x}$ satisfying

$$\hat{A}\hat{x} = \hat{b} \tag{1.40}$$

is called a nongeneric TLS solution (*the corresponding nongeneric TLS correction is $[\Delta\hat{A}; \Delta\hat{b}] = [A; b] - [\hat{A}; \hat{b}]$*).

Theorem 25 (Nongeneric Unidimensional TLS Solution) *Let (1.3) be the SVD of* $[A; b]$*, and assume that* $v_{n+1,j} = 0$ *for* $j = p+1, \ldots, n+1$*,* $p \leq n$*. If* $\sigma_{p-1} > \sigma_p$ *and* $v_{n+1,p} \neq 0$*, then*

$$[\hat{A}; \hat{b}] = U \hat{\Sigma} V^T \qquad \text{where} \quad \hat{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_{p-1}, 0, \sigma_{p+1}, \ldots, \sigma_{n+1}) \tag{1.41}$$

with corresponding nongeneric TLS correction matrix

$$[\Delta\hat{A}; \Delta\hat{b}] = \sigma_p u_p v_p^T \tag{1.42}$$

solves the nongeneric TLS problem (1.39) and

$$\hat{x} = -\frac{1}{v_{n+1,p}} \left[v_{1,p}, \ldots, v_{n,p} \right]^T \tag{1.43}$$

exists and is the unique solution to $\hat{A}\hat{x} = \hat{b}$*.*

Proof. See [98, p. 72]). ■

Corollary 26 *If* $v_{n+1,n+1} = 0 \wedge v_{n+1,n} \neq 0 \wedge \sigma_{n-1} > \sigma_n$*, then (1.43) becomes*

$$\begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} = -\frac{v_n}{v_{n+1,n}} \tag{1.44}$$

Theorem 27 (Closed-Form Nongeneric TLS Solution) *Let (1.2) [respectively, (1.3)] be the SVD of A (respectively,* $[A; b]$*), and assume that* $v_{n+1,j} = 0$ *for* $j = p+1, \ldots, n+1$*,* $p \leq n$*. If* $\sigma_{p-1} > \sigma_p$ *and* $v_{n+1,p} \neq 0$*, the nongeneric TLS solution is*

$$\hat{x} = \left(A^T A - \sigma_p^2 I_n \right)^{-1} A^T b \tag{1.45}$$

Proof. See [98, p. 74]. ■

Remark 28 *The nongeneric TLS algorithm must identify the close-to-nongeneric or nongeneric situation before applying the corresponding formula. In the following it will be shown that the TLS EXIN neuron solves both generic and nongeneric TLS problems without changing its learning law (i.e., automatically).*

1.8 MIXED OLS–TLS PROBLEM

If n_1 columns of the $m \times n$ data matrix A are *known exactly*, the problem is called *mixed OLS–TLS* [98]. It is natural to require that the TLS solution not perturb the exact columns. After some column permutations in A such that $A = [A_1; A_2]$, where $A_1 \in \Re^{m \times n_1}$ is made of the exact n_1 columns and $A_2 \in \Re^{m \times n_2}$, perform n_1 Householder transformations Q on the matrix $[A; B]$ (QR factorization) so that

$$[A_1; A_2; B] = Q \begin{bmatrix} R_{11} & R_{12} & R_{1b} \\ 0 & R_{22} & R_{2b} \end{bmatrix} \begin{matrix} n_1 \\ m - n_1 \end{matrix} \qquad (1.46)$$
$$\begin{matrix} n_1 & n - n_1 & d \end{matrix}$$

where R_{11} is a $n_1 \times n_1$ upper triangular matrix. Then compute the TLS solution $\hat{X}_2$ of $R_{22}X \approx R_{2b}$. $\hat{X}_2$ yields the last $n - n_1$ components of each solution vector $\hat{x}_i$. To find the first n_1 rows $\hat{X}_1$ of the solution matrix $\hat{X} = \left[\hat{X}_1^T; \hat{X}_2^T\right]^T$, solve (OLS) $R_{11}\hat{X}_1 = R_{1b} - R_{12}\hat{X}_2$. Thus, the *entire* method amounts to a preprocessing step, a TLS problem, an OLS problem, and a *postprocessing* step (inverse row permutations) [72].

Theorem 29 (Closed-Form Mixed OLS–TLS Solution) *Let rank* $A_1 = n_1$; *denote by* σ' (*respectively,* σ) *the smallest* [*respectively,* $(n_2 + 1)th$] *singular value of* R_{22} (*respectively,* $[R_{22}; R_{2b}]$); *assume that the smallest singular values* $\sigma = \sigma_{n_2+1} = \cdots = \sigma_{n_2+d}$ *coincide. If* $\sigma' > \sigma$, *then the mixed OLS–TLS solution is*

$$\hat{X} = \left(A^T A - \sigma^2 \begin{bmatrix} 0 & 0 \\ 0 & I_{n_2} \end{bmatrix}\right)^{-1} A^T B \qquad (1.47)$$

Proof. It is a special case of [97, Th. 4]. ■

1.9 ALGEBRAIC COMPARISONS BETWEEN TLS AND OLS

Comparing (1.29) with the LS solution

$$X' = \left(A^T A\right)^{-1} A^T B \qquad (1.48)$$

shows that σ_{n+1} completely determines the difference between the solutions. Assuming that A is of full rank, $\sigma_{n+1} = 0$ means that both solutions coincide ($AX \approx B$ compatible or underdetermined). As σ_{n+1} deviates from zero, the set $AX \approx B$ becomes more and more incompatible and the differences between the TLS and OLS solutions become deeper and deeper.

If $\sigma'_n > \sigma_{n+1} = \cdots = \sigma_{n+d}$, then

$$\left\|\hat{X}\right\|_F \geq \left\|X'\right\|_F \tag{1.49}$$

σ_{n+1} also influences the difference in condition between the TLS and OLS problems and thus the difference in numerical accuracy of their respective solutions in the presence of worst-case perturbations. $\sigma'^2_n - \sigma^2_{n+1}$ is a measure of how close $AX \approx B$ is to the class of nongeneric TLS problems. Assuming that $A_0X \approx B_0$ is the corresponding unperturbed set, rank $A_0 = n$ and the perturbations in A and B have approximately the same size, the TLS solution is more accurate than the OLS solution, provided that the ratio $(\sigma_n - \sigma^0_{n+1})/\sigma'_n > 1$, where σ^0_{n+1} is the $(n+1)$th singular value of $[A_0; B_0]$. The advantage of TLS is more remarkable with increasing ratio: for example, when $\sigma'_n \approx 0$, when $\|B_0\|_F$ is large, or when X_0 becomes close to the singular vector v'_n of A_0 associated with its smallest singular value.

1.9.1 About the Residuals

Proposition 30 *Define the LS residua R' as $B - AX'$ and the TLS residual $\hat{R}$ as $B - A\hat{X}$; then*

$$\hat{R} - R' = \sigma^2_{n+1} A \left(A^T A - \sigma^2_{n+1} I\right)^{-1} X' \tag{1.50}$$

$$\left\|\hat{R}\right\|_F \geq \left\|R'\right\|_F \tag{1.51}$$

The TLS and LS residuals approach each other if:

1. σ_{n+1} is small (slightly incompatible set).
2. $\|B\|_F$ is small (i.e., the TLS solution becomes close to the LS solution).
3. $\sigma'_n \gg \sigma_{n+1}$ [i.e., A may not be (nearly) rank deficient].
4. B is close to the largest singular vectors of A.

Proposition 31 *If $\sigma_{n+1} = 0$, then $\hat{R} = R' = 0$.*

1.10 STATISTICAL PROPERTIES AND VALIDITY

Under the assumption that *all* errors in the augmented matrix $[A; B]$ are rowwise independently and identically distributed (i.i.d.) with *zero mean* and *common covariance matrix* of the form $\sigma^2_v \Xi_0$, Ξ_0 known and positive definite (e.g., the identity matrix), the TLS method offers the *best* estimate and is more accurate than the LS solution in estimating the parameters of a model. The most suitable model for the TLS concept is the *errors-in-variables* (EIV) model, which assumes an unknown but *exact* linear relation (*zero residual* problems) among the true variables that can only be observed with errors.

Definition 32 (Multivariate Linear EIV Model)

$$\begin{aligned} B_0 &= 1_m \alpha^T + A_0 X_0, \quad B_0 \in \Re^{m \times d}, \quad A_0 \in \Re^{m \times n}, \quad \alpha \in \Re^d \\ A &= A_0 + \Delta A \\ B &= B_0 + \Delta B \end{aligned} \tag{1.52}$$

where $1_m = [1, \ldots, 1]^T$. X_0 *is the* $n \times d$ *matrix of the true but unknown parameters to be estimated. The intercept vector* α *is either zero* (*no-intercept model*) *or unknown* (*intercept model*) *and must be estimated.*

Proposition 33 (Strong Consistency) *If, in the EIV model, it is assumed that the rows of* $[\Delta A; \Delta B]$ *are i.i.d. with common zero-mean vector and common covariance matrix of the form* $\Xi = \sigma_v^2 I_{n+d}$, *where* $\sigma_v^2 > 0$ *is unknown, then the TLS method is able to compute* strongly consistent *estimates of the unknown parameters* $X_0, A_0, \alpha,$ *and* σ_v.

EIV models are useful when:

1. The primary goal is to estimate the true parameters of the model generating the data rather than prediction and if there is not a priori certainty that the observations are error-free.
2. The goal is the application of TLS to the eigenvalue–eigenvector analysis or SVD (TLS gives the hyperplane that passes through the intercept and is parallel to the plane spanned by the first right singular vectors of the data matrix [174]).
3. It is important to treat the variables symmetrically (i.e., there are no independent and dependent variables).

The ordinary LS solution X' of (1.52) is generally an *inconsistent* estimate of the true parameters X_0 (i.e., LS is asymptotically *biased*). Large errors (large Ξ, σ_v), ill-conditioned A_0, as well as, in the unidimensional case, the solution oriented close to the lowest right singular vector v_n' of A_0 increase the bias and make the LS estimate more and more inaccurate. If Ξ is known, the asymptotic bias can be removed and a consistent estimator, called *corrected least squares* (CLS) can be derived [60,106,168]. The CLS and TLS asymptotically yield the same consistent estimator of the true parameters [70,98]. Under the given assumption about the errors of the model, the TLS estimators $\hat{X}, \hat{\alpha}, \hat{A},$ *and* $[d/(n+d)]\hat{\sigma}^2$ $[\hat{\sigma}^2 = (1/mt)\sum_{i=1}^{t} \sigma_{n+i}^2$ with $t = \min\{m-n, d\}]$ are the unique with probability 1 *maximum likelihood* estimators of $X_0, \alpha, A_0,$ *and* σ_v^2 [70].

Remark 34 (Scaling) *The assumption about the errors seems somewhat restrictive: It requires that all measurements in A and B be affected by errors and, moreover, that these errors must be uncorrelated and equally sized. If these conditions are not satisfied, the classical TLS solution is no longer a consistent estimate of the model parameters. Provided that the error covariance matrix* Ξ *is known up*

to a factor of proportionality, the data $[A; B]$ *can be transformed to the new data* $[A^*; B^*] = [A; B]\,C^{-1}$, *where* C *is a square root* (*Cholesky factor*) *of* Ξ $(= C^T C)$, *such that the error covariance matrix of the transformed data is now diagonal with equal error variances. Then the TLS algorithm can be used on the new set, and finally, its solution must be converted to a solution of the original set of equations* [*69,70*].

Remark 35 (Covariance Knowledge) *The knowledge of the form of the error covariance matrix up to a constant scalar multiple may still appear too restrictive, as this type of information is not always available to an experimenter. Assuming that independent repeated observations are available for each variable observed with error, this type of replication provides enough information about the error covariance matrix to derive consistent unbiased estimates of* Ξ [*60,62*]. *Using these consistent estimates instead of* Ξ *does not change the consistency properties of the parameter estimators* [*60*].

The TLS estimators in both the intercept and no-intercept models are *asymptotically normally distributed*. For the unidimensional case, the covariance matrix of the TLS estimator $\hat{x}$ is larger than the covariance matrix of the LS estimator x', even if A is noisy [98].

Summarizing, a comparison about the accuracy of the TLS and LS solution with respect to their bias, total variance, and mean squared error (MSE; total variance + squared bias) gives:

- The bias of TLS is much smaller than the bias of LS and decreases with increasing m (i.e., increasing the degree of overdetermination).
- The total variance of TLS is larger than that of LS.
- At the smallest noise variances, MSE is comparable for TLS and LS; by increasing the noise in the data, the differences in MSE are greater, showing the better performance of TLS; the TLS solution will be more accurate, especially when the set of equations are more overdetermined, but the better performance is already good for *moderate* sample size m.

All these conclusions are true even if the errors are not Gaussian but are exponentially distributed or t-distributed with 3 degrees of freedom.

1.10.1 Outliers

The TLS, which is characterized by larger variances, is less stable than LS. This instability has quite serious implications for the accuracy of the estimates in the presence of *outliers* in the data (i.e., large errors in the measurements) [20,105]. In this case a *dramatic* deterioration of the TLS estimates happens. Also, the LS estimates encounter serious stability problems, although less dramatically; efficient *robust* procedures should be considered, which are quite efficient and rather insensitive to outliers (e.g., by down-weighting measurement samples that

have high residuals). In the following, robust nonlinear neurons are introduced to overcome this problem.

1.11 BASIC DATA LEAST SQUARES PROBLEM

The TLS problem can be viewed as an unconstrained perturbation problem because all columns of $[A; b]$ can have error perturbations. The OLS problem constrains the columns of A to be errorless; the opposite case is the data least squares (DLS) problem, because the error is assumed to lie only in the *data* matrix A.

Definition 36 (Basic DLS Problem) *Given the overdetermined set (1.4), the data least squares (DLS) problem searches for*

$$\min_{A'' \in \Re^{m \times n}} \left\| A - A'' \right\|_F \qquad \text{subject to} \quad b \in R(A'') \tag{1.53}$$

Once a minimizing A'' is found, then any x'' satisfying

$$A''x'' = b \tag{1.54}$$

is called a DLS solution (*the corresponding DLS correction is $\Delta A'' = A - A''$*).

The DLS case is particularly appropriate for certain deconvolution problems, such as those that may arise in system identification or channel equalization [51].

Theorem 37 *The DLS problem (1.53) is solved by*

$$x'' = \frac{b^T b}{b^T A v_{\min}} v_{\min} \qquad \left(b^T A v_{\min} \neq 0\right) \tag{1.55}$$

where $v_{\min}$ is the right singular vector corresponding to the smallest singular value of the matrix $P_b^{\perp} A$, where $P_b^{\perp} = \left(I - b\left(b^T b\right)^{-1} b^T\right)$ is a projection matrix that projects the column space of A into the orthogonal complement of b. If the smallest singular vector is repeated, then the solution is not unique. The minimum norm solution is given by

$$x'' = \frac{b^T b}{\left(b^T A V_{\min}\right)\left(V_{\min}^T A^T b\right)} V_{\min}\left(V_{\min}^T A^T b\right) \qquad \left(b^T A V_{\min}\right)\left(V_{\min}^T A^T b\right) \neq 0 \tag{1.56}$$

where $V_{\min}$ is the right singular space corresponding to the repeated smallest singular value of $P_b^{\perp} A$.

Proof. See [51], which derives its results from the constrained total least squares (CTLS) [1,2]. ■

1.12 PARTIAL TLS ALGORITHM

Considering the TLS solution of $AX \approx B$ to be deduced from a basis of the right singular subspace associated with the smallest singular values of $[A; B]$, considerable savings in computation time is possible by calculating only the basis vectors desired. This can be done in a *direct* way by modifying the SVD algorithm or in an *iterative* way by making use of start vectors. An improved algorithm PSVD (partial SVD) is presented in [98], which computes this singular subspace. There are three reasons for its higher efficiency vs. the *classical* SVD [75]:

1. The Householder transformations of the bidiagonalization are applied only to the basis vectors of the desired singular subspace.
2. The bidiagonal is only partially diagonalized.
3. An appropriate choice is made between QR and QL iteration steps [99].

Definition 38 (Motor Gap) *The* motor gap *is the gap between the singular values associated with the desired and undesired vectors.*

Depending on the motor gap (it must be large), the desired numerical accuracy, and the dimension of the desired subspace (it must be small), PSVD can be *three times as fast as* the classical SVD, while the same accuracy can be maintained. Incorporating the PSVD algorithm into the TLS computations results in an improved *partial TLS* (PTLS) [96]. Typically, PTLS reduces the computation time by a factor of 2.

1.13 ITERATIVE COMPUTATION METHODS

In the estimation of parameters of nonstationary systems that vary slowly with time, space, or frequency, a priori information is available for the TLS algorithm. Indeed, in this case, slowly varying sets of equations must be solved at each time instant and the TLS solution at the last step is usually a good initial guess for the solution at the next step. If the changes in these systems are of small norm but of *full* rank (e.g., when all elements of the data matrix change slowly from step to step), the computation time can be better reduced by using an *iterative* method. There are also other advantages in using this type of algorithm:

- Each step supplies a new and better estimate of the solution, permitting us to control the level of convergence depending on the perturbations of the data.
- It is easy to code.
- Some iterative routines use the given matrix over and over again without modification, exploiting its particular structure or sparsity.

1.13.1 Direct Versus Iterative Computation Methods

The direct computation methods are the classical TLS and the PTLS; their efficiency is determined essentially by the dimensions of the data matrix, the desired accuracy, the dimension p of the desired singular subspace, and the motor gap. The iterative methods are efficient in solving TLS problems if:

1. The start matrix is good and the problem is generic.
2. The desired accuracy is low.
3. The dimension p of the desired singular subspace is known.
4. The dimension d of the problem is low.
5. The data matrix dimensions are moderate.
6. The gap is sufficiently large.

In contrast to the iterative methods, the direct methods *always* converge to the desired solution.

In the sequel a list of the principal *nonneural* iterative methods is presented, together with their own field of application (see [98]).

1.13.2 Inverse Iteration

Many authors have studied the inverse iteration (II) method in (non)symmetric eigenvalue problems (see [75,150]). According to Wilkinson [193], it is the most powerful and accurate method for computing eigenvectors.

Being $S = C^T C$, where $C = [A; B]$, the *iteration matrix* Q_k is given by

$$Q_k = (S - \lambda_0 I)^{-k} Q_0 \tag{1.57}$$

with Q_0 a start matrix and λ_0 a chosen shift. Given the dimension p of the TLS eigensubspace of S, taking λ_0 zero or such that the ratio $|\sigma^2_{n-p} - \lambda_0|/|\sigma^2_{n-p+1} - \lambda_0|$ is high enough, matrix Q_k converges to the desired minor eigensubspace. The iteration destroys the structure of the matrix S (for details, see [98]).

Remark 39 (Convergence Property) *If the motor gap* (*the gap between* σ^2_{n-p} *and* σ^2_{n-p+1}) *is large,* fast convergence *occurs* (*this requirement is satisfied for many TLS problems*).

1.13.3 Chebyshev Iteration

When the motor gap is small, the convergence can be accelerated by applying the Chebyshev polynomials (see [136,150,164,193]), instead of the inverse power functions as before, to the matrix $C^T C$ for the *ordinary Chebyshev iteration* (OCI) and to the matrix $\left(C^T C\right)^{-1}$ for the *inverse Chebyshev iteration* (ICI).

The Chebyshev polynomials $T_k^{yz}(x)$ are orthogonal over the interval $[y, z]$ with respect to the density function $1/\sqrt{1-x^2}$. By choosing the interval $[y, z]$ as small as possible so that it contains all the *undesired* eigenvalues of a matrix S, the ordinary Chebyshev iteration method will converge to a basis of the eigensubspace associated with the remaining eigenvalues outside $[y, z]$: in particular, in multidimensional TLS problems, where the singular subspace associated with the p smallest singular values of C, $[y, z]$ must contain the $n - p$ largest eigenvalues of $S = C^T C = [A; B]^T [A; B]$. The inverse Chebyshev iteration is applied to the matrix $\hat{S} = \left(C^T C\right)^{-1}$ and then the interval $[\hat{z}, \hat{y}]$ must be chosen as small as possible such that it contains all undesired eigenvalues of $\hat{S}$ (i.e., the inverses of the squares of the singular values of $[A; B]$). Only OCI does not alter the matrix C.

Usually, for small motor gaps [typically, for ratios $\sigma_r/\sigma_{r+1} < 10$ for the gap between the singular values associated with the desired ($\leq \sigma_{r+1}$) and undesired ($\geq \sigma_r$) singular subspace of matrix C], ICI is recommended. Indeed, this method is proven [98, p. 160] always to converge faster than OCI. Generally, the gain in speed is very significant. In contrast to OCI and provided that the undesired singular value spectrum is not too small (e.g., $\sigma_1/\sigma_r \geq 2$), the convergence rate of ICI is hardly influenced by the spread of this spectrum as well as by the quality of its lower bound $\hat{z} \leq 1/\sigma_1^2$ [98, p. 152]. Moreover, ICI is proved [98, pp. 152–157] always to converge faster than II, provided that an optimal estimation of the bound $\hat{y}$ is given. The smaller the motor gap, the larger the gain in speed. OCI is shown [98, p. 139] to be efficient only in problems characterized by a very dense singular value spectrum, which is not the case for most TLS problems. However, it is recommended for solving very large, sparse, or structured problems, because it does not alter the matrix C. Table 1.1 summarizes these comparisons.

1.13.4 Lanczos Methods

Lanczos methods are iterative routines that bidiagonalize an arbitrary matrix C or tridiagonalize a symmetric matrix S directly without any orthogonal updates as in the Householder approach to the classical SVD. The original matrix structure is not destroyed and needs little storage; thus, these methods work well for large, sparse, or structured matrices [98]. They are used for TLS problems with a small motor gap; if the Lanczos process is applied to S(LZ), it has a minimally faster convergence than OCI with optimal bounds. Like OCI, the convergence

Table 1.1 Link between the singular value spectrum and the best convergence[a]

	II	OCI	ICI	RQI	LZ	ILZ
Motor gap	large	small	small	small	small	small
Undesired SV spread	indep.	small	indep.	indep.	small	indep.
Bounds	no	optimal	optimal	no	no	no

[a] indep., independent.

rate depends not only on the motor gap but also on the relative spread of the undesired singular value spectrum. If it is applied to S^{-1} [the inverse Lanczos method(ILZ)], it has the same convergence properties as ICI with optimal bounds. However, round-off errors make all Lanczos methods difficult to use in practice [75]. For a summary, see Table 1.1.

1.13.5 Rayleigh Quotient Iteration

The Rayleigh quotient iteration (RQI) can be used to accelerate the convergence rate of the inverse iteration process when the motor gap is small (see Table 1.1), no adequate shift λ_0 can be computed, and convergence to only *one* singular vector of the desired singular subspace is required. RQI is a variant of II by applying a variable shift $\lambda_0(k)$, which is the RQ $r(q_k)$ (see Section 2.1) of the iteration vector q_k, defined by

$$r(q_k) = \frac{q_k^T S q_k}{q_k^T q_k} \quad \text{minimizing} \quad f(\lambda) = \|(S - \lambda I)\, q_k\|_2 \tag{1.58}$$

Then the RQI is given by

$$(S - \lambda_0(k) I)\, q_k = q_{k-1} \tag{1.59}$$

With regard to II, RQI need not estimate the fixed shift λ_0, because it generates the *best shift* in each step automatically and therefore converges faster. Parlett [149] has shown that the RQI convergence is ultimately *cubic*. However, RQI requires more computations per iteration step than II. Furthermore, RQI applies only to square-symmetric matrices S and it is impossible to avoid the explicit formation of $S = C^T C$, which may affect the numerical accuracy of the solution. A good start vector q_0 should be available to allow the RQI to converge to the desired solution. For example, most time-varying problems are characterized by abrupt changes at certain time instants that seriously affect the quality of the start vector and cause RQI to converge to an undesired singular triplet (see [98]). If convergence to several basis vectors of the desired singular subspace is required, RQI extensions must be used (e.g., inverse subspace iteration with Ritz acceleration) [48].

1.14 RAYLEIGH QUOTIENT MINIMIZATION NONNEURAL AND NEURAL METHODS

From eqs. (1.22) and (1.21) it is evident that $E_{\mathrm{TLS}}(x)$ corresponds to the Rayleigh quotient (see Section 2.1) of $[A; b]$, and therefore the TLS solution corresponds to its minimization. This minimization is equal to the minor components analysis(MCA), because it corresponds to the search of the eigenvector associated with the minimum eigenvalue of $[A; b]$, followed by a scaling of the solution into the TLS hyperplane. This idea will be explained at greater length in the

following two chapters. Among the nonneural methods, the first to use such an idea in a recursive TLS algorithm was Davila [50]; there the desired eigenvector was updated with a correction vector chosen as a Kalman filter gain vector, and the scalar step size was determined by minimizing RQ. Bose et al. [11] applied recursive TLS to reconstruct high-resolution images from undersampled low-resolution noisy multiframes. In [200] the minimization is performed by using the conjugate gradient method, which requires more operations but is very fast and is particularly suitable for large matrices.

The neural networks applied to TLS problems can be divided into two categories: the neural networks for the MCA, which are described in Chapter 2, and the neural networks that iterate only in the TLS hyperplane and therefore give the TLS solution directly, which are described in Chapter 3. They are the following:

- *The Hopfield-like neural network of Luo, Li, and He* [120,121]. This network is made up of $3(m+n)+2$ neurons (m and n are the dimensions of the data matrix) grouped in a main network and in four subnetworks; the output of the main network gives the TLS solution, and the available data matrix and observation vector are taken directly as the interconnections and the bias current of the network; thus, the principal limit of this network is the fact that it is linked to the dimensions of the data matrix and cannot be used without structural changes for other TLS problems. The authors demonstrate[2] the stability of the network for *batch* operation: that is, working on all the equations together (as opposed to *sequential* or *online* operation, which updates at every equation presentation). The initial conditions cannot be null. The network is based on an analog circuit architecture which has continuous-time dynamics. The authors apply the network to the TLS linear prediction frequency estimation problem.
- *The linear neuron of Gao, Ahmad, and Swamy* [63,64]. This is a single linear neuron associated with a constrained anti-Hebbian learning law, which follows from the linearization of $E_{\mathrm{TLS}}(x)$, so it is correct enough for small gains and, above all, for *weight norms much smaller than 1*. The weight vector gives the TLS solution after the learning phase. It has been applied to adaptive FIR and IIR parameter estimation problems. In the first problem, after learning, the output of the neuron gives the error signal of the adaptive filter; this is a useful property because in a great many adaptive filter applications the error signal has the same importance as the filter parameters and other signal quantities. From now on, this neuron will be termed *TLS GAO*.
- *The linear neurons of Cichocki and Unbehauen* [21,22]. The authors propose linear neurons with different learning laws to deal with OLS, TLS, and DLS problems. For each algorithm an appropriate cost energy to be minimized

[2]This demonstration is not original, because the authors rediscover the well-known theorem of stability for gradient flows (see, e.g., [84, p. 19]).

is devised, which contains the classical minimization [e.g., the minimization of $E_{\mathrm{TLS}}(x)$ for the TLS] plus regularization terms for ill-conditioned problems and nonlinear functions for the robustness of the method. Then the system of differential equations describing the gradient flow of this energy is implemented in an analog network for the continuous-time learning law and in a digital network for the discrete-time learning law. The analog network consists of analog integrators, summers, and multipliers; the network is driven by independent source signals (zero-mean random, high frequency, uncorrelated i.i.d.) multiplied by the incoming data a_{ij}, b_i $(i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n)$ from $[A; b]$. The artificial neuron, with an on-chip adaptive learning algorithm, allows both complete processing of the input information simultaneously and the sequential strategy. In the digital neuron the difference equations of the gradient flow are implemented by CMOSswitched-capacitor (SC) technology. The neurons for the TLS do not work on the exact gradient flow, but on its linearization:[3] It gives the *same* learning law of TLS GAO for a particular choice of the independent signals. The DLS learning lawis introduced empirically, without justification. The examples of [22] are used as a benchmark for comparison purposes.

Remark 40 *TLS GAO and the linear neurons of Cichochi and Unbehauen for the TLS problems have learning laws which are not gradient flows of the error function because of the linearization, which forbids the use of acceleration techniques based on the Hessian of the error function and the conjugate gradient method.*

[3]It is evident that as for TLS GAO, the norm of the weight must be much less than unity for linearization; curiously, the authors forget to cite this limit.

2

MCA EXIN NEURON*

2.1 RAYLEIGH QUOTIENT

Definition 41 (RQ) *The* Rayleigh[1] quotient r *is a function that assigns to any* nonzero *complex vector u the scalar quantity*

$$r(u) \equiv r(u, C) \equiv \frac{u^* C u}{u^* u} = \frac{\sum_j \sum_k c_{jk} \overline{u}_j u_k}{\sum_i |u_i|^2} \tag{2.1}$$

where $C = [c_{jk}]$.

The most relevant properties of the RQ are the following (for the proofs, see [14,84,149,150]):

- *Homogeneity*:

$$r(\alpha u, \beta C) = \beta r(u, C) \qquad \forall \alpha, \beta \neq 0 \tag{2.2}$$

- *Translation invariance*:

$$r(u, C - \alpha I) = r(u, C) - \alpha \tag{2.3}$$

[1]John William Strutt (Lord Rayleigh), 1849–1919, born at Langford Grove in England, died at Terling Place.

*This chapter is reprinted from G. Cirrincione, M. Cirrincione, J. Hérault, and S. VanHuffel, "The MCA EXIN Neuron for the Minor Component Analysis," *IEEE Transactions on Neural Networks*, Vol. 13, No. 1, pp. 160–187, 2002. Copyright © 2002 IEEE. Reprinted with permission.

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

- *Boundedness*. As u ranges over all nonzero vectors, $r(u)$ fills a region in the complex plane which is called the *field of values* (also, the *numerical range*) of C. This region is closed, bounded, and convex. If $C = C^*$ (*self-adjoint matrix*), the field of values is the real interval bounded by the extreme eigenvalues $[\lambda_{\min}, \lambda_{\max}]$.
- *Orthogonality*:

$$u \perp (C - r(u) I) u \tag{2.4}$$

- *Minimal residual*: $\forall u \neq 0 \wedge \forall$ scalar μ,

$$\|(C - r(u) I) u\| \leq \|(C - \mu I) u\| \tag{2.5}$$

2.1.1 Critical Points

The following theorem gives, as a special case, the stationarity properties of RQ.

Theorem 42 (Courant–Fischer) *Let C be a Hermitian n-dimensional matrix with eigenvalues*

$$\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$$

Then (Fischer–Poincaré)

$$\lambda_k = \min_{V^{n-k+1}} \max_{u \in V^{n-k+1}} r(u, C) \tag{2.6}$$

and (Courant–Weyl)

$$\lambda_k = \max_{S^{n-k}} \min_{v \perp S^{n-k}} r(v, C) \tag{2.7}$$

$(k = 1, \ldots, n)$, where $V^{n-k+1}(S^{n-k})$ ranges over all subspaces of dimension $n - k + 1$ $(n - k)$.

Proposition 43 (Stationarity) *Let C be a real symmetric n-dimensional matrix with eigenvalues $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$ and corresponding unit eigenvectors $z_1, z_2, \ldots, z_n$. Then*

$$\lambda_1 = \max r(u, C) \tag{2.8}$$

$$\lambda_n = \min r(u, C) \tag{2.9}$$

More generally, the critical points and critical values of $r(u, C)$ are the eigenvectors and eigenvalues for C (for an alternative proof, see Section 2.5).

The *Hessian* matrix of the Rayleigh quotient is given by

$$H_r = -\frac{1}{\|u\|_2^2}\left(-r + 2\frac{ruu^T}{\|u\|_2^2} + \frac{u^T ruI + 2uu^T r}{\|u\|_2^2} - 4\frac{\left(u^T ru\right)uu^T}{\|u\|_2^4}\right)$$
$$= \frac{2}{\|u\|_2^2}\left(C - \text{grad}\,(r)\,u^T - u\,\text{grad}\,(r)^T - rI\right) \tag{2.10}$$

where u is real and

$$\text{grad}\,(r) = \frac{2}{\|u\|_2^2}\left(C - rI\right)u \tag{2.11}$$

It can be observed that $\forall i = 1, 2, \ldots, n$,

$$H_r\,(z_i) = C - \lambda_i I \tag{2.12}$$

Hence,

$$\det\left[H_r\,(z_i)\right] = \det\left[C - \lambda_i I\right] = 0 \tag{2.13}$$

which implies that $H_r\,(z_i)$ is singular $\forall z_i$. Furthermore,

$$H_r\,(z_i)\,z_j = \begin{cases} 0 & i = j \\ \left(\lambda_j - \lambda_i\right)z_j & i \neq j \end{cases} \tag{2.14}$$

So H_r, computed at the RQ critical points, has the same eigenvectors as C, but with different eigenvalues. $H_r\,(u)$ is positive semidefinite only when $u = z_{\min}$ (see, e.g., [81]).

Proposition 44 (Degeneracy) *The Rayleigh quotient critical points are* degenerate *because in these points the Hessian matrix is not invertible. Then the Rayleigh quotient is* not *a* Morse function[2] *in every open subspace of the domain containing a critical point.*

Remark 45 *Among the possible numerical techniques to minimize the Rayleigh quotient, it is not practical to use the variable metric (VM, also known as quasi-Newton) and the Newton descent techniques because the RQ Hessian matrix at the minimum is singular and then the inverse does not exist. On the other hand, the conjugate gradient can deal with the singular* H_r [*85, pp. 256–259; 196*].

[2]A function $f: U \to \Re$, U being an open subset of $\Re^n$, is called a *Morse function* if all its critical points $x_0 \in U$ are nondegenerate.

2.2 MINOR COMPONENTS ANALYSIS

Definition 46 (Minor Components) *The eigenvectors that correspond to the smallest eigenvalues of the autocorrelation matrix of the data vectors are defined as the* minor components [*195*].

The minor components are the directions in which the data have the smallest variances (the principal components are the directions in which the data have the largest variances). Expressing data vectors in terms of the minor components is called *minor components analysis* (MCA).

2.2.1 Some MCA Applications

There are a lot of MCA applications, especially in adaptive signal processing. MCA has been applied to frequency estimation [128,129], bearing estimation [167], digital beamforming [77], moving-target indication [107], and clutter cancellation [5]. It has also been applied to TLS algorithms for parameter estimation [63,64]; Xu et al. [195] have shown the relationship of MCA with the curve and surface fitting under the TLS criterion, and as a result, MCA is employed increasingly in many engineering fields, such as computer vision (see [24]).

2.2.2 Neural Answers

There are a lot of neural networks to solve the task of MCA. The only nonlinear network is the Hopfield network by Mathew and Reddy [128,129]. The authors develop a constrained energy function, using a penalty function, to minimize the RQ. The neurons have sigmoidal activation functions. However, the structure of the network is problem dependent (the number of neurons is equal to the dimension of the eigenvectors); in addition, it is necessary to estimate the trace of the covariance matrix for selecting appropriate penalty factors. All other existing neural networks are made up of *one* simple *linear* neuron.

2.2.3 MCA Linear Neurons: An Overview

In building a neural network, linear units are the simplest to use. They are often considered to be uninteresting because only linear functions can be computed in linear networks, and a network with several layers of linear units can always be collapsed into a linear network without hidden layers by multiplying the weights in the proper fashion. On the contrary, there are very important advantages. Oja [138] has found that a simple linear neuron with an unsupervised constrained Hebbian learning rule can extract the principal component from stationary input data. Later, Linsker [115–117] showed that in a layered feedforward network of linear neurons with random inputs and Hebbian learning, spatial opponent cells, orientation selective units, and orientation columns emerge in successive layers just like the organization of the mammalian primary visual cortex. Contrary to the

nonlinear neural networks, which are seriously plagued by the problem of local minima of their cost function, the linear networks have simpler and meaningful cost landscapes: The usual cost function of the linear feedforward network in autoassociation mode has a unique minimum corresponding to the projection onto the subspace generated by the first principal vectors, and all additional critical points are saddle points [4,15].

Consider a linear unit with input $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ and output $y(t)$:

$$y(t) = \sum_{i=1}^{N} w_i(t) x_i(t) = w^T(t) x(t) \tag{2.15}$$

where $w(t) = [w_1(t), w_2(t), \ldots, w_n(t)]^T$ is the weight vector. In MCA and PCA analysis, $x(t)$ is a bounded continuous-valued stationary ergodic (see Section 2.3 for the definition) data vector with finite second-order moments.

The existing learning laws for the MCA of the autocorrelation matrix $R = E\left[x(t) x^T(t)\right]$ of the input vector $x(t)$ are listed below.

2.2.3.1 Oja's Learning Laws Changing Oja's learning law for PCA [138] into a constrained anti-Hebbian rule, by reversing the sign, the following rule (*OJA*) [195] is given:

$$w(t+1) = w(t) - \alpha(t) y(t) \left[x(t) - y(t) w(t)\right] \tag{2.16}$$

$\alpha(t)$ being the positive learning rate. Its explicitly normalized version (*OJAn*) [195] (inspired by [110]) is

$$w(t+1) = w(t) - \alpha(t) y(t) \left[x(t) - \frac{y(t) w(t)}{w^T(t) w(t)}\right] \tag{2.17}$$

Substituting eq. (2.15) into the equations above gives, respectively,

$$\begin{aligned} w(t+1) = w(t) - \alpha(t) \big[&x(t) x^T(t) w(t) \\ &- w^T(t) x(t) x^T(t) w(t) w(t)\big] \end{aligned} \tag{2.18}$$

$$\begin{aligned} w(t+1) = w(t) - \alpha(t) \Big[&x(t) x^T(t) w(t) \\ &- \frac{w^T(t) x(t) x^T(t) w(t)}{w^T(t) w(t)} w(t)\Big] \end{aligned} \tag{2.19}$$

Under certain assumptions, using the techniques of the stochastic approximation theory (see [113,118,142]), the corresponding averaging differential equations

are, respectively,

$$\frac{dw(t)}{dt} = -Rw(t) + \left[w^T(t) Rw(t)\right] w(t) \tag{2.20}$$

$$\frac{dw(t)}{dt} = -Rw(t) + \underbrace{\frac{w^T(t) Rw(t)}{w^T(t) w(t)}}_{\text{Rayleigh quotient}} w(t) \tag{2.21}$$

The solutions $w(t)$ of eq. (2.16) [eq. (2.17)] tend to the asymptotically stable points of eq. (2.20) [eq. (2.21)].

In [195], the following theorem is presented.

Theorem 47 (Asymptotic Stability) *Let R be positive semidefinite with minimum eigenvalue λ_n of multiplicity 1 and corresponding unit eigenvector z_n. If $w^T(0) z_n \neq 0$, then:*

1. *For eq. (2.20), asymptotically $w(t)$ is parallel to z_n.*
2. *In the special case of $\lambda_n = 0$, it holds that $\lim_{t\to\infty} w(t) = \left(w^T(0) z_n\right) z_n$.*
3. *For eq. (2.21), it holds that $\lim_{t\to\infty} w(t) = \pm z_n$ and*

$$\lim_{t\to\infty} w^T(t) Rw(t) = \lambda_n \tag{2.22}$$

Proof. See [195, App., p. 455; 142, pp. 72–75]. ■

Another Oja's learning rule [141] (*OJA+*) is the following:

$$w(t+1) = w(t) - \alpha(t)\left[y(t) x(t) - (\, y^2(t) + 1 - \|w(t)\|_2^2 \,) w(t)\right] \tag{2.23}$$

The corresponding averaging ODE is given by

$$\frac{dw(t)}{dt} = -Rw(t) + \left[w^T(t) Rw(t)\right] w(t) + w(t) - \|w(t)\|_2^2 w(t) \tag{2.24}$$

The following theorem is also given.

Theorem 48 (Asymptotic Stability) *In eq. (2.23) assume that the eigenvalues of R satisfy $\lambda_1 > \lambda_2 > \cdots > \lambda_n > 0$ and $\lambda_n < 1$ and that the learning rate $\alpha(t)$ is positive. Then $\lim_{t\to\infty} w(t) = z_{\min}$ (or $-z_{\min}$) if $w^T(0) z_{\min}$ is positive (or negative).*

Proof. See [141, App., p. 935]. ■

Oja's linear neurons have been combined with PCA linear neurons for implementing the dual subspace pattern recognition (DSPR) method [194], in which a pattern class is represented not only by its subspace with its basis vectors being principal components, but also by its *complementary* subspace with its basis vectors being minor components.

2.2.3.2 Luo–Unbehauen–Cichocki Learning Law

Luo et al. [124] propose the following rule (*LUO*):

$$w(t+1) = w(t) - \alpha(t)\left[w^T(t)\, w(t)\, y(t)\, x(t) - y^2(t)\, w(t)\right] \tag{2.25}$$

with the corresponding averaging ODE,

$$\frac{dw(t)}{dt} = -w^T(t)\, w(t)\, Rw(t) + \left[w^T(t)\, Rw(t)\right] w(t) \tag{2.26}$$

together with the following theorem.

Theorem 49 (Asymptotic Stability) *If $w(0)$ satisfies $w^T(0)\, z_n \neq 0$ and λ_n is single, then*

$$\|w(t)\|_2^2 = \|w(0)\|_2^2 \qquad t > 0 \tag{2.27}$$

$$\lim_{t\to\infty} w(t) = \pm \|w(0)\|\, z_n \tag{2.28}$$

Proof. See [124, App., pp. 296–297]. ■

In [124, App., pp. 296–297], Luo et al. claim the discovery of eq. (2.27) and demonstrate its validity even for OJAn, without knowing that this equation was already well known and had already been discovered by one of the authors of OJAn [142].

In [18] an MCA algorithm has been presented; however, this is coincident with the LUO algorithm (see [123]).

2.2.3.3 Feng–Bao–Jiao Learning Law

In [56] the following rule (*FENG* or *FENG1*[3]) is introduced:

$$w(t+1) = w(t) - \alpha(t)\left[w^T(t)\, w(t)\, y(t)\, x(t) - w(t)\right] \tag{2.29}$$

The corresponding averaging ODE is given by

$$\frac{dw(t)}{dt} = -w^T(t)\, w(t)\, Rw(t) + w(t) \tag{2.30}$$

[3]Another rule is also given in the paper, but it does not work as well.

The global asymptotic convergence of eq. (2.30) is established by the following theorem.

Theorem 50 (Asymptotic Stability) *Let R be positive semidefinite with minimum eigenvalue λ_n of multiplicity 1 and corresponding unit eigenvector z_n. Then it holds that*

$$\lim_{t\to\infty} w(t) = \pm\frac{1}{\sqrt{\lambda_n}} z_n \tag{2.31}$$

Proof. See [56, pp. 2127–2129]. ■

In the simulations, the large oscillations of the weights around the solution are apparent. According to the authors, this sensitivity of the estimate is caused by the statistical fluctuation and error. In this chapter we show the true reasons for this phenomenon.

2.3 MCA EXIN LINEAR NEURON

There are various ways of demonstrating the MCA EXIN learning law [30]. It can be derived from the generalization of the TLS EXIN learning law [24,36] to a vectorial space of one more degree of freedom. Another way (followed here) is to find a stochastic learning law that has a well-defined averaging ODE, which is exactly the opposite of the reasoning regarding the other learning rules. This ODE represents the gradient flow of the Rayleigh quotient of the autocorrelation matrix R $(= E\left[x(t)\, x^T(t)\right])$ on $\Re^N - \{0\}$.

As already noted, the input vector x is assumed to be generated by such an *ergodic*[4] information source that the input sequence is an independent stochastic process governed by $p(x)$. According to the stochastic learning law, the weight vector w is changed in random directions and is uncorrelated with x. The averaged cost function is

$$E[J] = r(w, R) = \frac{w^T R w}{w^T w} = \frac{E\left[y^2\right]}{\|w\|_2^2} \tag{2.32}$$

where, as usual, $y = w^T x$. Then, up to a constant, the gradient flow of $E[J]$, which is then the averaging MCA EXIN ODE, is given by

$$\begin{aligned}\frac{dw(t)}{dt} &= -\frac{1}{\|w(t)\|_2^2}\left[R - \frac{w^T(t) R w(t)}{\|w(t)\|_2^2} I\right] w(t) \\ &= -\frac{1}{\|w\|_2^2}\left[R - r(w, R) I\right] w\end{aligned} \tag{2.33}$$

[4]The information source generating the vectors x is *ergodic* if the temporal average over a typical sequence $x(t)$ is the same as the average over the probability distribution $p(x)$.

(I is the identity matrix), which is the average version of the continuous-time differential equation

$$\frac{dw(t)}{dt} = -\frac{1}{\|w(t)\|_2^2}\left[x(t)x^T(t) - \frac{w^T(t)x(t)x^T(t)w(t)}{\|w(t)\|_2^2}\right]w(t) \tag{2.34}$$

which, after discretization, gives the MCA EXIN nonlinear stochastic learning rule:

$$w(t+1) = w(t) - \frac{\alpha(t)y(t)}{\|w(t)\|_2^2}\left[x(t) - \frac{y(t)w(t)}{\|w(t)\|_2^2}\right] \tag{2.35}$$

where the instantaneous cost function is given by

$$J = \frac{w^T x x^T w}{w^T w} = \frac{y^2}{\|w\|_2^2} \tag{2.36}$$

According to the stochastic approximation theory (see [113,118,142]), if some conditions are satisfied, eq. (2.33) represents eq. (2.35) effectively (i.e., their asymptotic paths are close with a large probability) and eventually the MCA EXIN solution tends with probability 1 to the uniformly asymptotically stable solution of the ODE. From a computational point of view, the most important conditions are the following:

1. $x(t)$ is zero mean, stationary, and bounded with probability 1.
2. $\alpha(t)$ is a decreasing sequence of *positive* scalars.
3. $\sum_t \alpha(t) = \infty$.
4. $\sum_t \alpha^p(t) < \infty$ for some p.
5. $\lim_{t\to\infty} \sup\left[\frac{1}{\alpha(t)} - \frac{1}{\alpha(t-1)}\right] < \infty$.

For example, the sequence $\alpha(t) = \text{const} \cdot t^{-\gamma}$ satisfies the four last conditions for $0 < \gamma \leq 1$. The fourth condition is less restrictive than the Robbins–Monro condition [161] $\sum_t \alpha^2(t) < \infty$, which is satisfied, for example, only by $\alpha(t) = \text{const} \cdot t^{-\gamma}$ with $\frac{1}{2} < \gamma \leq 1$.

MCA EXIN shares the same structure of LUO and OJAn:

$$w(t+1) = w(t) - \alpha(t) S_{\text{neuron}}(w(t), x(t)) \tag{2.37}$$

where

$$S_{\text{MCA EXIN}} = \frac{1}{\|w(t)\|_2^2} S_{\text{OJAn}} = \frac{1}{\|w(t)\|_2^4} S_{\text{LUO}} \tag{2.38}$$

The difference among these laws seems irrelevant, above all considering that according to eq. (2.27), the weight modulus is constant. Instead, they behave very differently. The reason for this is explained in the following section in Proposition 52 and in Theorem 68.

2.4 RAYLEIGH QUOTIENT GRADIENT FLOWS

Let $\Phi : M \to \Re$ be a smooth function defined on a manifold M and let $D\Phi : M \to T^*M$ denote the differential, that is, the section of the cotangent bundle[5] T^*M defined by

$$D\Phi(w) : T_wM \to \Re \qquad \xi \to D\Phi(w)\,\xi \equiv D\Phi|_w(\xi) \tag{2.39}$$

where $D\Phi(w)$ is the derivative of Φ at w. To be able to define the *gradient vector field* of Φ, a Riemannian metric[6] $\langle \cdot, \cdot \rangle_w$ on M must be specified. Then the consequent gradient $\operatorname{grad}\Phi : M \to TM$ is determined uniquely by the following two properties:

1. *Tangency condition:*
$$\operatorname{grad}\Phi(w) \in T_wM \qquad \forall w \in M \tag{2.40}$$

2. *Compatibility condition:*
$$D\Phi(w)\,\xi = \langle \operatorname{grad}\Phi(w), \xi \rangle \qquad \forall \xi \in T_wM \tag{2.41}$$

If $M = \Re^n$ is endowed with its standard Riemannian metric defined by $\langle \xi, \eta \rangle = \xi^T \eta \;\; \forall \xi, \eta \in \Re^n$, then the associated gradient vector is just

$$\nabla\Phi(w) = \left(\frac{\partial\Phi}{\partial w_1}(w), \frac{\partial\Phi}{\partial w_2}(w), \ldots, \frac{\partial\Phi}{\partial w_n}(w) \right)^T$$

If $Q{:}\Re^n \to \Re^{nxn}$ is a symmetric matrix denoting a smooth map which represents the Riemannian metric, then

$$\operatorname{grad}\Phi(w) = Q^{-1}(w)\,\nabla\Phi(w) \tag{2.42}$$

which shows the dependence of $\operatorname{grad}\Phi(w)$ on the metric $Q(w)$.

Let $S^{n-1} = \{ w \in \Re^n \mid \|w\| = 1 \}$ denote the unit sphere in $\Re^n$. Denote $r(w, R) : S^{n-1} \to \Re$ the restriction of the Rayleigh quotient of the autocorrelation matrix on the unit sphere. The tangent space is $T_wS^{n-1} =$

[5]The *tangent bundle* TM of a manifold M is the set-theoretic disjoint union of all tangent spaces T_wM of M (i.e., $TM = \cup_{w \in M} T_wM$). The *cotangent bundle* T^*M of M is defined as $T^*M = \cup_{w \in M} T_w^*M$, where $T_w^*M = \operatorname{Hom}(T_wM, \Re)$ is the dual (cotangent) vector space of T_wM.

[6]The *inner product* is determined uniquely by a positive definite symmetric matrix $Q \in \Re^{n \times n}$ such that $\langle u, v \rangle = u^T Q v$ holds $\forall u, v \in \Re^n$, a *Riemannian metric* on M is a family of nondegenerate inner products $\langle \cdot, \cdot \rangle_w$ defined on each tangent space T_wM, such that $\langle \cdot, \cdot \rangle_w$ depends smoothly on $w \in M$; thus, the Riemannian metric on $\Re^n$ is just a smooth map $Q : \Re^n \to \Re^{n \times n}$ such that $\forall w \in \Re^n$, $Q(w)$ is a real positive definite symmetric $n \times n$ matrix. Every nondegenerate inner product on $\Re^n$ defines a Riemannian metric on $\Re^n$. The converse is not true. A *Riemannian manifold* is a smooth manifold endowed with a Riemannian metric.

$\left\{\xi \in \Re^n \,\middle|\, w^T\xi = 0\right\}$. For any unit vector $w \in S^{n-1}$, the Fréchet derivative of $r(w,R)$ is the linear functional $Dr(w,R)\big|_w : T_wS^{n-1} \to \Re$ defined by

$$Dr(w,R)\big|_w(\xi) = 2\langle Rw, \xi\rangle = 2w^TR\xi$$

To define the gradient on the sphere, the standard Euclidean inner product on T_wS^{n-1} is chosen as a Riemannian metric on S^{n-1} [i.e., up to a constant, $\langle\xi,\eta\rangle \equiv 2\xi^T\eta \;\; \forall\xi,\eta \in T_wS^{n-1}$]. The gradient $\nabla r(w,R)$ is then determined uniquely if it satisfies the tangency condition and the compatibility condition:

$$\begin{aligned} Dr(w,R)|_w(\xi) &= \langle \nabla r(w,R), \xi\rangle \\ &= 2\nabla r^T(w,R)\,\xi \qquad \forall\xi \in T_wS^{n-1} \end{aligned}$$

which since $Dr(w,R)\big|_w(\xi) = 2w^TR\xi$ implies that $[\nabla r(w,R) - Rw]^T\xi = 0$. From the definition of tangent space, it follows that $\nabla r(w,R) = Rw + \lambda w$ with $\lambda = -w^TRw$, so that $w^T\nabla r(w,R) = 0$ to satisfy the tangency condition. Thus, the gradient flow for the Rayleigh quotient on the unit sphere S^{n-1} is

$$\frac{dw(t)}{dt} = -[R - r(w,R)I_n]\,w(t) \tag{2.43}$$

which is eq. (2.21) (i.e. the ODE of OJAn).

The Rayleigh quotient gradient flow restricted to the unit sphere extends to a flow on $\Re^n - \{0\}$. Deriving the expression of the RQ with respect to w gives the Fréchet derivative of $r(w,R) : \Re^n - \{0\} \to \Re$, which is

$$Dr(w,R)|_w(\xi) = \frac{2}{\|w\|_2^2}(Rw - r(w,R)\,w)^T\xi \tag{2.44}$$

Define a Riemannian metric on each tangent space $T_w(\Re^n - \{0\})$ as, $\forall\xi,\eta \in T_w(\Re^n - \{0\})$,

$$\langle\langle\xi,\eta\rangle\rangle \equiv \frac{2}{\|w\|_2^2}\xi^T\eta \overset{(2.42)}{\Longrightarrow} \underbrace{\frac{dw(t)}{dt} = -\|w\|_2^2\,\nabla\Phi(w)}_{\text{ODE LUO}} \tag{2.45}$$

The gradient of $r(w,R) : \Re^n - \{0\} \to \Re$ with respect to this metric is characterized by $\operatorname{grad} r(w,R) \in T_w(\Re^n - \{0\})$ and $Dr(w,R)|_w(\xi) = \langle\langle \operatorname{grad} r(w,R), \xi\rangle\rangle \;\; \forall\xi \in T_w(\Re^n - \{0\})$. The first condition imposes no constraint on $\operatorname{grad} r(w,R)$, and the second one is easily seen to be equivalent to

$$\operatorname{grad} r(w,R) = Rw - r(w,R)\,w$$

Then eq. (2.43) also gives the gradient of $r(w,R) : \Re^n - \{0\} \to \Re$.

Table 2.1 Gradient flows and corresponding Riemannian metric

ODE	Riemannian Metric	Tangent Space
OJAn	$2\xi^T\eta$	$T_w S^{n-1}$
LUO	$2\xi^T\eta \;/\; \|w\|_2^2$	$T_w\,(\Re^n - \{0\})$
EXIN	$2\,\|w\|_2^2\,\xi^T\eta$	$T_w\,(\Re^n - \{0\})$

The same analysis about LUO can be applied to MCA EXIN. In this case the derivative is taken parallel to the Fréchet derivative of $r\,(w, R) : \Re^n - \{0\} \to \Re$ [i.e., eq. (2.25) multiplied by $\|w\|_2^4$] and the Riemannian metric on each tangent space $T_w\,(\Re^n - \{0\}]$ is, $\forall \xi, \eta \in T_w\,(\Re^n - \{0\})$,

$$\langle\langle(\xi, \eta)\rangle\rangle \equiv 2\,\|w\|_2^2\,\xi^T\eta \overset{(2.42)}{\Longrightarrow} \underbrace{\frac{dw\,(t)}{dt} = -\,\|w\|_2^{-2}\,\nabla\Phi\,(w)}_{\text{ODE MCA EXIN}} \tag{2.46}$$

The analysis above is summarized in Table 2.1. Convergence and stability properties of gradient flows may depend on the choice of the Riemannian metric. In case of a nondegenerate critical point of Φ, the local stability properties of the gradient flow around that point do not change with the Riemannian metric. However, in case of a degenerate critical point, the qualitative picture of the local phase portrait of the gradient around that point may well change with the Riemannian metric [84,169].

Proposition 51 (Equivalence) *The ODEs of LUO, OJAn, and MCA EXIN are* equivalent *because they only differ from the Riemannian metric. This fact implies a similar stability analysis* (*except for the critical points*).

As seen in Proposition 44, the Rayleigh quotient critical points are degenerate. As a consequence, the phase portrait of the gradient flow has only degenerate straight lines [also recall the RQ homogeneity property (2.2) for $\beta = 1$] in the direction of the RQ eigenvectors (i.e., the critical points are not isolated). This fundamental observation, together with the analysis above, implies the following proposition and justifies the creation of MCA EXIN [24,30].

Proposition 52 (Local Stability) *The fact that the Rayleigh quotient is not a Morse function implies that the local stability properties* (*local phase portrait*) *of the gradient flow around the RQ critical points change with the Riemannian metric.*

2.5 MCA EXIN ODE STABILITY ANALYSIS

This analysis deals with the MCA EXIN ODE and is therefore constrained by the ODE approximation assumptions: It will be considered as a theory about the

first approximation of the neuron behavior [i.e., until the minimum is reached for the first time, as will be clear later (*first approximation* assumption)]. Recall that the first approximation is exact if the weights are constrained in a subspace of the weight vector space, just like hyperspheres or hyperplanes. Consider the MCA EXIN linear neuron

$$y(t) = w^T(t)\, x(t) \tag{2.47}$$

where $w(t), x(t) \in \Re^n$ are, respectively, the weight and the input vector. The averaged cost function is [eq. (2.32)]

$$E = r(w, R) = \frac{w^T R w}{w^T w} \tag{2.48}$$

where $R = E\left[x(t)\, x^T(t)\right]$ is the autocorrelation matrix of the input vector $x(t)$. Assume that R is *well behaved* (i.e., full rank with distinct eigenvalues). Being an autocorrelation matrix, it is symmetric, positive definite with orthogonal eigenvectors $z_n, z_{n-1}, \ldots, z_1$ and corresponding eigenvalues $\lambda_n < \lambda_{n-1} < \cdots < \lambda_1$. The cost function E is bounded from below (see Section 2.1). As seen in eq. (2.33), the gradient of the averaged cost function, unless a constant, is given by

$$\nabla E = \frac{1}{w^T w}\left[Rw - r(w, R)\, w\right] \tag{2.49}$$

which is used for the MCA EXIN learning law.

2.5.1 Critical Directions

The following reasonings are an alternative demonstration of Proposition 43 and introduce the notation. As a consequence of the assumptions above, the weight vector space has a basis of orthogonal eigenvectors. Thus,

$$w(t) = \sum_{i=1}^{n} \omega_i(t)\, z_i \tag{2.50}$$

From eqs. (2.49) and (2.50), it follows that the coordinates of the gradient along the principal components are

$$(\nabla E)^T z_i = \frac{\omega_i(t)}{w^T(t)\, w(t)}\left[\lambda_i - \frac{\sum_{j=1}^{n} \lambda_j \omega_j^2(t)}{\sum_{j=1}^{n} \omega_j^2(t)}\right] \tag{2.51}$$

Then the critical points of the cost landscape E are given by

$$\omega_i(t) = 0 \quad \text{or} \quad \lambda_i - \frac{\sum_{j=1}^{n} \lambda_j \omega_j^2(t)}{\sum_{j=1}^{n} \omega_j^2(t)} = 0 \tag{2.52}$$

The critical points can be deduced by looking at the number of nonzero $\omega_i(t)$ coordinates. If they are all null at time t, they will remain null. If two weight coordinates $\omega_{i_0}(t)$ and $\omega_{i_1}(t)$ on two different eigenvector directions are different from 0, (2.52) yields

$$\|w(t)\|_2^2 = \frac{\sum_{j=1}^{n} \lambda_j \omega_j^2(t)}{\lambda_{i_0}} = \frac{\sum_{j=1}^{n} \lambda_j \omega_j^2(t)}{\lambda_{i_1}} \tag{2.53}$$

which would imply that the two eigenvalues corresponding to these two nonzero coordinates are equal; this is not consistent with the cost assumptions. Furthermore, the same conclusion is reached under the assumption of more than two nonzero coordinates. Hence, there is at most one nonzero $\omega_i(t)$ coordinate. Excluding the zero weight because it is outside the domain of existence of the Rayleigh quotient ($\Re^n - \{0\}$), the n critical points are then parallel to the eigenvector directions. At these directions, the cost landscape takes the constant values given by the corresponding eigenvalues:

$$E(z_i) = \lambda_i \tag{2.54}$$

and therefore it is better to speak of *critical directions*.

2.5.1.1 Neighborhood of the Origin Suppose that we move in any direction in weight space from the origin, which is outside the cost domain. The new position $S(t)$ is given by

$$S(t) = \sum_i \varepsilon_i(t) z_i \qquad \text{with } 0 < \|S\|_2^2 \ll 1 \tag{2.55}$$

Around the origin, the cost of the perturbed point is upper bounded:

$$E(S(t)) = \frac{\sum_i \lambda_i \varepsilon_i^2(t)}{\sum_i \varepsilon_i^2(t)} < \frac{\sum_i \lambda_1 \varepsilon_i^2(t)}{\sum_i \varepsilon_i^2(t)} = \lambda_1 \tag{2.56}$$

which is obvious because every straight line from the origin has the same cost value [see eq. (2.2)] and λ_1 is an upper bound for the cost function (Proposition 43).

2.5.1.2 Stability Properties and Cones To study the acceptable critical directions, suppose that we move from one point in such a critical direction in various directions, written as linear combinations of the eigenvectors. As the simplest case, consider moving from the critical point (direction) corresponding to one eigenvector in the direction of the same eigenvector. As a consequence of the homogeneity property of the RQ [see eq. (2.2)], the error cost remains constant [see eq. (2.54)]. This implies an *indifferent equilibrium* (recall the *first*

approximation assumption). Consider now a perturbation from one point P^* in the critical direction i_0 to the critical direction i_1:

$$w(t) = \omega_{i_0}^*(t) z_{i_0} + \varepsilon_{i_1}(t) z_{i_1} \tag{2.57}$$

The corresponding cost error change is given by

$$\Delta E = E - E^* = \frac{\varepsilon_{i_1}^2(t)}{\|w(t)\|_2^2} \left(\lambda_{i_1} - \lambda_{i_0}\right) \tag{2.58}$$

Thus, the stability of the critical direction is determined by the difference between the eigenvalues associated with the corresponding eigenvectors. If the critical direction corresponds to the minor component direction, ΔE is always positive. However, ΔE can become negative when moving in any direction, corresponding to an eigenvector with a smaller eigenvalue than the critical direction considered. This demonstrates the following proposition.

Proposition 53 (Stability 1) *The critical direction is a global minimum in the direction of any eigenvector with a larger eigenvalue and a maximum in the direction of any eigenvector with a smaller eigenvalue.*

Consider moving from a critical direction (at point P^*) toward a linear combination of other eigenvectors. It implies that

$$w(t) = \omega_{i_0}^*(t) z_{i_0} + \sum_{j \neq i_0} \varepsilon_j(t) z_j \tag{2.59}$$

and

$$\Delta E = E - E^* = \frac{\sum_{j \neq i_0} \varepsilon_j^2(t) \left(\lambda_j - \lambda_{i_0}\right)}{\|w(t)\|_2^2} \tag{2.60}$$

Proposition 54 (Stability 2) *The critical direction is a global minimum in the direction of any linear combination of eigenvectors with larger eigenvalues and a maximum in the direction of any linear combination of eigenvectors with smaller eigenvalues.*

If the critical direction considered corresponds to the minor component direction, ΔE will always be positive. However, for any other eigenvector direction, it is possible to find a neighborhood such that ΔE is negative.

Consider the three-dimensional weight vector space ($n = 3$); there are three critical directions: the minimum direction (parallel to z_3), the saddle direction

(parallel to z_2), and the maximum direction (parallel to z_1). In every plane perpendicular to z_2, the locus of points with the same energy as z_2 (i.e., $\Delta E_2 = 0$) is given from eq. (2.60) as

$$\lambda_2 = \frac{\varepsilon_1^2 \lambda_1 + \varepsilon_3^2 \lambda_3}{\varepsilon_1^2 + \varepsilon_3^2} \tag{2.61}$$

where ε_1 and ε_3 are the two coordinates corresponding to the orthogonal axes parallel to e_1 and e_3. Thus,

$$(\lambda_1 - \lambda_2)\, \varepsilon_1^2 - (\lambda_2 - \lambda_3)\, \varepsilon_3^2 = 0 \tag{2.62}$$

which is a degenerate quadric (*cone*) composed of the following two planes:

$$\varepsilon_1 = \pm \sqrt{\frac{\lambda_2 - \lambda_3}{\lambda_1 - \lambda_2}} \varepsilon_3 \tag{2.63}$$

These planes intersect in the z_2 straight line and are symmetric with respect to the plane $z_2 z_3$. Their slope depends on the spectrum of R. These planes (defined as constant E_2 planes) divide the weight space into a (double) *volume of attraction* ($\Delta E_2 > 0$) and into a (double) *volume of repulsion* ($\Delta E_2 < 0$) (from now on they will be defined as *cone of attraction* and *cone of repulsion*, even if, actually, there is only one cone dividing the two volumes) from the point of view of the saddle direction z_2. Figure 2.1 illustrates this analysis.

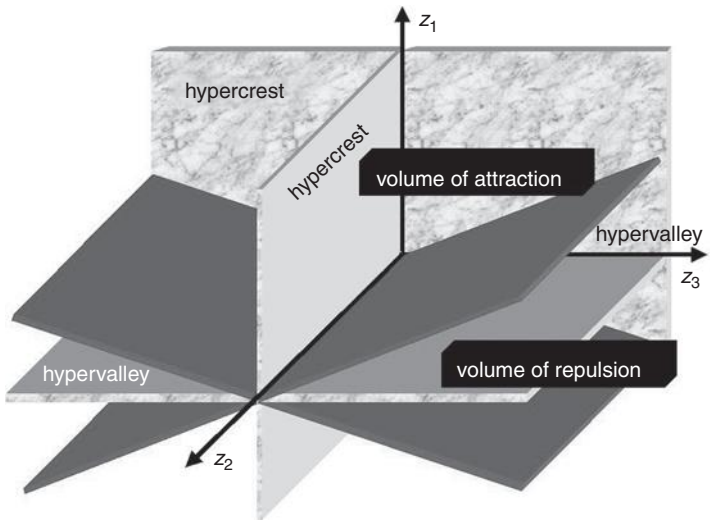


Figure 2.1 MCA EXIN ODE cost landscape stability analysis (three-dimensional case).

Consider now the case $n > 3$. Define the *gap* between the eigenvalue corresponding to the critical direction i_0 and the eigenvalue corresponding to the critical direction $j \neq i_0$ as $g_{ji_0} = \lambda_j - \lambda_{i_0}$. Then, in every hyperplane perpendicular to the direction i_0, the locus of points with the same energy as z_{i_0} (i.e., $\Delta E_{i_0} = 0$) is given by

$$-\sum_{j > i_0} |g_{ji_0}| \varepsilon_j^2 + \sum_{j < i_0} |g_{ji_0}| \varepsilon_j^2 = 0 \tag{2.64}$$

This homogeneous equation of the second degree specifies an $(n-1)$-dimensional *cone* whose vertex (center) is the intersection with the i_0 direction. If all g_{ji_0}'s are *of the same sign* (i.e., the minimum and maximum directions), then the cone is *imaginary* (only the center is a real point). The saddle directions have g_{ji_0}'s of different sign. In these cases the cone is called a *real* cone in the sense that it has real points besides the center. The saddles are of the following kind:

- *Saddle with index* $i_0 \geq (n+1)/2$: a saddle closer to the minimum than to the maximum. The hyperplane $\varepsilon_n = \text{constant} \neq 0$ intersects the cone (2.64) in an $(n-2)$-dimensional ellipsoid if $i_0 = n-1$ (i.e., the saddle closest to the minimum). It means that the minimum direction is the axis of the cone of repulsion of this saddle. The same hyperplane intersects the cone (2.64) in a hyperboloid if $i_0 < n-1$ (i.e., the inner saddles) (this case is not possible for $n \leq 4$). Every saddle repulsion cone intersects with the i_0 direction as center and consists of all possible straight lines passing through the center and through points of the surface in which it intersects the hyperplane $\varepsilon_n = \text{const} \neq 0$. The $(i_0 - 1)$-dimensional plane

$$\varepsilon_{i_0+1} = \text{const}_{i_0+1},\ \varepsilon_{i_0+2} = \text{const}_{i_0+2}, \ldots, \varepsilon_n = \text{const}_n \tag{2.65}$$

  intersects the cone (2.64) in an $(i_0 - 2)$-dimensional ellipsoid. It follows that the cone of repulsion contains all linear combinations of the eigenvectors with smaller eigenvalues (λ_j, $j > i_0$). Every reasoning can be repeated, but with respect to the $j < i_0$ directions, to describe the corresponding cones of attraction.
- *Saddle with index* $i_0 < (n+1)/2$: a saddle closer to the maximum than to the minimum. The same kind of reasoning also applies here, but with respect to the maximum; for example, the saddle closest to the maximum has a cone of attraction with the axis in the maximum direction and a cone of repulsion that contains all linear combinations of the eigenvectors with smaller eigenvalues (λ_j, $j > i_0$).

In every hyperplane perpendicular to the direction i_0, the cones are always the same, but the values of the error cost E are scaled by the denominator of eq. (2.60), which is a function of the weight component parallel to the i_0 direction.

It implies that far from the origin, the RQ cost landscape becomes flatter, and therefore in this case all the gradient flow methods are too slow. Summarizing:

Proposition 55 (Saddle Cones) *Every saddle direction has an infinity of cones of attraction, each with centers in this direction and containing the directions of the linear combinations of eigenvectors with bigger eigenvalues than those of the saddle eigenvalue. It also has an infinity of cones of repulsion, each with axes orthogonal to the axes of the cones of attraction, with centers in the saddle direction and containing the directions (dimensions of escape) of the linear combinations of the eigenvectors with smaller eigenvalues than the saddle eigenvalue.*

Remark 56 (Low Weights) *When using a gradient flow, it is better to work with low weights, to avoid the flatness of the RQ cost landscape.*

Consider now moving from a critical direction (at point P^*) toward a linear combination of this direction and of any other eigenvector:

$$w(t) = \omega_{i_0}^*(t) z_{i_0} + \sum_j \varepsilon_j(t) z_j \tag{2.66}$$

with corresponding energy:

$$E = \frac{\lambda_{i_0}\left[\omega_{i_0}^*(t) + \varepsilon_{i_0}(t)\right]^2 + \sum_{j\neq i_0}\lambda_j \varepsilon_j^2(t)}{\|w(t)\|_2^2} \tag{2.67}$$

If the considered critical direction corresponds to the smaller eigenvalue (MC), it follows that $\lambda_j = \lambda_{i_0} + k_j$ with $k_j > 0$. The energy becomes

$$E = \frac{\lambda_{i_0}\left[\omega_{i_0}^{*2}(t) + e\left(\varepsilon_j(t)\right)\right]^2}{\omega_{i_0}^{*2}(t) + \sum_j \varepsilon_j^2(t)} + \frac{\sum_{j\neq i_0} k_j \varepsilon_j^2(t)}{\omega_{i_0}^{*2}(t) + \sum_j \varepsilon_j^2(t)} \tag{2.68}$$

with $e\left(\varepsilon_j(t)\right) = 2\omega_{i_0}^*(t)\varepsilon_{i_0}(t) + \sum_j \varepsilon_j^2(t)$ being a linear combination of the ε_j's. The first term on the right-hand side of eq. (2.68) is equivalent to (2.54) in the neighborhood of the minimum direction, reached for $e = 0$ when $\varepsilon_j = 0\ \forall j \neq i_0$. The second term is a convex function of the ε_j's, minimized when $\varepsilon_j = 0\ \forall j \neq i_0$. Hence, the entire right-hand side is minimized when $\varepsilon_j = 0\ \forall j \neq i_0$. It implies that this critical direction is a minimum in any of the directions considered. If the critical direction does not correspond to the minimum eigenvalue, the condition $k_j > 0$ is no longer valid and the critical direction is not a minimum for the energy E.

Proposition 57 (General Facts About Stability) *The Rayleigh quotient (energy E) of the autocorrelation matrix R has n critical directions in the*

domain $\Re^n - \{0\}$. *Among these, all the directions associated with the nonminor components are saddle directions and correspond to ever-larger values of the cost function as the corresponding eigenvalues increase. They are minima in any direction of the space spanned by bigger eigenvectors* (*in the sense of eigenvectors associated with bigger eigenvalues*) *but are maximaz in any direction of the space spanned by smaller eigenvectors: While getting deeper (i.e., toward the minimum) in the cost landscape, the number of dimensions of escape from a saddle point becomes smaller. The critical direction associated with the minimum eigenvalue is the only global minimum.*

2.5.1.3 Hypercrests and Hypervalleys Using eqs. (2.48) and (2.50), the energy function can be recast as

$$E = \frac{\sum_{j=1}^{n} \lambda_j \omega_j^2 (t)}{\sum_{j=1}^{n} \omega_j^2 (t)} \tag{2.69}$$

The critical loci of this cost function *along* a critical axis indexed with i_0 are found by differentiating the function along the corresponding coordinate. The derivative is

$$\frac{\partial E}{\partial \omega_{i_0}} = \frac{2\omega_{i_0}}{\left(\sum_{j=1}^{n} \omega_j^2\right)^2} \underbrace{\left(\lambda_{i_0} \sum_{j \neq i_0} \omega_j^2 - \sum_{j \neq i_0} \omega_j^2 \lambda_j\right)}_{A} \tag{2.70}$$

$A = \sum_{j \neq i_0} \omega_j^2 \left(\lambda_{i_0} - \lambda_j\right) \neq 0$ always because of the assumption on the spectrum of R, except for $\omega_j = 0 \; \forall j \neq i_0$ (i.e., the i_0 direction). Thus, the critical loci are given by

$$\frac{\partial E}{\partial \omega_{i_0}} = 0 \tag{2.71}$$

$$\Leftrightarrow \begin{cases} \omega_{i_0} = 0 & \text{hyperplane} \perp i_0 \text{ direction} \\ \omega_j = 0 & i_0 \text{ direction } (\text{const } E) \\ \forall j \neq i_0 & \end{cases} \tag{2.72}$$

($\omega_j = 0 \; \forall j$, i.e., the origin, is out of the domain). The typology of these critical loci can be studied by checking the second derivative:

$$\frac{\partial^2 E}{\partial \omega_{i_0}^2} = \frac{2A}{\left(\sum_j \omega_j^2\right)^3} \left(\sum_j \omega_j^2 - 4\omega_{i_0}^2\right) \tag{2.73}$$

The straight line in the i_0 direction yields $A = 0$, and it is evident that this second derivative and all higher derivatives are null, just giving a constant E, which agrees with the previous analysis.

$$\left.\frac{\partial^2 E}{\partial \omega_{i_0}^2}\right|_{\omega_{i_0}=0} = \frac{2A}{\left(\sum_{j\neq i_0} \omega_j^2\right)^3}$$

$$\sum_{j\neq i_0} \omega_j^2 = \frac{2\sum_{j\neq i_0} \omega_j^2 \left(\lambda_{i_0} - \lambda_j\right)}{\left(\sum_{j\neq i_0} \omega_j^2\right)^2} \tag{2.74}$$

By the sign analysis of this formula [for i_0 representing the index of a saddle direction, the sign analysis of eq. (2.60) with reversed sign holds], the following proposition holds.

Proposition 58 (Hypercrests/Valleys) *The hyperplane through the origin and perpendicular to the minimum direction is made up of points which are maxima along the minimum direction (hypercrest along the minimum direction). The hyperplane through the origin and perpendicular to the maximum direction is made up of points which are that along the maximum direction* (*hypervalley along the maximum direction*). *The hyperplane through the origin and perpendicular to a saddle direction is made up of two types of points: points given by the intersection with the saddle cone of repulsion, which are minima along the saddle direction (hypervalley along the saddle direction) and points given by the intersection with the saddle cone of attraction (i.e., all the other points), which are maxima along the saddle direction* (*hypercrest along the saddle direction*).

Figure 2.1 shows the hypercrests and hypervalleys for the three-dimensional case.

2.5.2 Gradient Flows and Equilibria

The MCA EXIN ODE performs a gradient descent on the RQ cost function in weight space. In discrete form, it yields

$$w(t+1) = w(t) + \frac{\alpha(t)}{w^T(t)\, w(t)}\left[-Rw(t) + \frac{w^T(t)\, Rw(t)}{w^T(t)\, w(t)} w(t)\right] \tag{2.75}$$

Expressing the weight vector as $w(t) = \sum_{i=1}^{n} \omega_i(t)\, z_i$ yields

$$\omega_i(t+1) = \omega_i(t)\left[1 + \frac{\alpha(t)}{\|w(t)\|_2^4}\left(\sum_{j\neq i} \omega_j^2(t)\left(\lambda_j - \lambda_i\right)\right)\right] \tag{2.76}$$

Equation (2.76) describes an n-dimensional dynamical system whose equilibrium points are defined by $\omega_i(t+1) = \omega_i(t)$. In this case this property holds only if $\omega_j = 0\ \forall j \neq i_0$ (i.e., all the R eigenvectors are equilibrium directions). Consider a weight perturbation from an equilibrium direction as in eq. (2.57) [i.e. $w(t) = \omega_{i_0}(t)\, e_{i_0} + \varepsilon_{i_1}(t)\, e_{i_1}$]. In this case, eq. (2.76) yields

$$\omega_{i_0}(t+1) = \left[1 + \alpha(t) \frac{\varepsilon_{i_1}^2(t) \left(\lambda_{i_1} - \lambda_{i_0}\right)}{\left(\omega_{i_0}^2(t) + \varepsilon_{i_0}^2(t)\right)^2} \right] \omega_{i_0}(t) \tag{2.77}$$

$$\varepsilon_{i_1}(t+1) = \left[1 + \alpha(t) \frac{\omega_{i_0}^2(t) \left(\lambda_{i_0} - \lambda_{i_1}\right)}{\left(\omega_{i_0}^2(t) + \varepsilon_{i_0}^2(t)\right)^2} \right] \varepsilon_{i_1}(t) \tag{2.78}$$

For $\lambda_{i_0} > \lambda_{i_1}$, it is possible to find a small enough $\varepsilon_{i_1}(t)$ such that the state of the dynamical system will be farther from the equilibrium direction at time $t+1$ than at time t [i.e., $\omega_{i_0}(t+1) \approx \omega_{i_0}(t)$ and $\varepsilon_{i_1}(t+1) > \varepsilon_{i_1}(t)$]. This result confirms the fact that because the equilibrium direction considered is a local maximum in the direction considered, and because the algorithm is a gradient descent, the equilibrium must be unstable in this direction: *Equilibrium directions are unstable in the direction of the eigenvectors with smaller eigenvalues.* This result will remain valid for any linear combination of such eigenvectors. Around an equilibrium direction associated with an eigenvector, consider the region of weight space spanned by the eigenvectors with bigger eigenvalues. Recalling the basic assumption of the first approximation, in this region the continuous function E has a unique minimum at the equilibrium direction and is strictly decreasing since the dynamical system follows the gradient of E. Therefore, E is a Lyapunov function (this fact will be used in the proof of Theorem 60). Hence, the Lyapunov theorem justifies the fact that the equilibrium direction considered is asymptotically stable in the region of the space considered. Furthermore, for the equilibrium direction associated with the smallest eigenvalue, this region becomes the entire space (i.e., the global minimum is asymptotically stable in the n-dimensional weight space).

Remark 59 (Extension) *By considering the fact that OJAn and LUO have the same learning law structure as MCA EXIN* [*see eq.* (2.37)]*, all the theory about MCA EXIN ODE stability presented here is valid for them. Indeed, the consequences of the learning law are the same, because eqs.* (2.75) *to* (2.78) *would differ for a positive* (*and then irrelevant*) *quantity.*

2.5.3 Convergence to the Minor Component

This section deals with the asymptotic behavior of the MCA EXIN ODE, which can be considered only in the limits of validity of this asymptotic theory (see [118, Th. 3 and discussion, p. 556]).

Theorem 60 (Asymptotic Behavior) *Let R be the $n \times n$ autocorrelation matrix input data, with eigenvalues $0 \leq \lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$ and corresponding orthonormal eigenvectors $z_n, z_{n-1}, \ldots, z_1$. If $w(0)$ satisfies $w^T(0) z_n \neq 0$ and λ_n is single, then,* in the limits of validity of the ODE approximation*, for MCA EXIN it holds that*

$$w(t) \to \pm \|w(0)\|_2 z_n \quad \vee \quad w(t) \to \infty \tag{2.79}$$

$$\|w(t)\|_2^2 \approx \|w(0)\|_2^2 \qquad t > 0 \tag{2.80}$$

Proof. (also deals with OJAn and LUO) Multiplying eq. (2.33) by $w^T(t)$ on the left yields

$$\begin{aligned} \frac{d\|w(t)\|_2^2}{dt} &= \frac{2}{\|w(t)\|_2^4} \left[-\|w(t)\|_2^2 Rw(t) + \left(w^T(t) Rw(t)\right) \|w(t)\|_2^2 \right] \\ &= 0 \end{aligned} \tag{2.81}$$

Then

$$\|w(t)\|_2^2 = \|w(0)\|_2^2 \qquad t > 0 \tag{2.82}$$

The symbol $\approx$ in eq. (2.80) is a consequence of the fact that the ODE represents the learning law only as a first approximation. The same reasoning also applies to OJAn and LUO. As demonstrated in Section 2.4, OJAn, LUO, and MCA EXIN are gradient flows with different Riemannian metrics of the Rayleigh quotient. It is well known (see, e.g., [84,118]) that the gradient flows always converge to a minimum straight line. Indeed, for all three neurons, the Rayleigh quotient can be chosen as a Lyapunov function for their ODEs. Restricting the RQ to a sphere with the origin at the center justifies its use as a Lyapunov function $V(w)$ because it is definite positive and, unless a constant,

$$\begin{aligned} \frac{dV(w(t))}{dt} &= \left(\frac{dw(t)}{dt}\right)^T \left(\frac{dV(w)}{dw}\right)_{\Re^n - \{0\}} \\ &= \left(\frac{dw(t)}{dt}\right)^T \|w(t)\|_2^{-2} (\nabla V(w))_{S^{n-1}} \\ &= \left(\frac{dw(t)}{dt}\right)^T \|w(t)\|_2^{-2} Q(w(t)) \operatorname{grad} V(w) \\ &= -\left(\frac{dw(t)}{dt}\right)^T \|w(t)\|_2^{-2} Q(w(t)) \frac{dw(t)}{dt} \\ &= -Q(w(t)) \|w(t)\|_2^{-2} \left\| \frac{dw(t)}{dt} \right\|_2^2 \end{aligned}$$

$$= \begin{cases} -\|w(t)\|_2^{-4} \left\| \frac{dw(t)}{dt} \right\|_2^2 \leq 0 & \text{LUO} \\ -\|w(t)\|_2^{-2} \left\| \frac{dw(t)}{dt} \right\|_2^2 \leq 0 & \text{OJAn} \\ -\left\| \frac{dw(t)}{dt} \right\|_2^2 \leq 0 & \text{EXIN} \end{cases} \tag{2.83}$$

and for all three neurons $dV(t)/dt = 0$ for $dw(t)/dt = 0$ (i.e., $dV(t)/dt$ is negative semidefinite (i.e., the Lyapunov function is *weak*). For completeness the Lyapunov function time derivative is given for OJA [83]:

$$\frac{dV(w(t))}{dt} = -\|w(t)\|_2^{-2} \left[\left\| \frac{dw(t)}{dt} \right\|_2^2 + (1 - \|w(t)\|_2^2) \left[\left(w^T(t) Rw(t) \right)^2 - \|Rw(t)\|_2^2 \right] \right]$$

Then, for the Lyapunov direct method (see, e.g., [132]) the ODE has the points for $dw(t)/dt = 0$ as asymptotic stable points in the limit of the ODE approximation, but the global asymptotic stability theorem does not hold because $V(w)$ is *radially bounded*. According to Ljung [118, Cor. 2], the ODEs have the point $\{\infty\}$ as stationary, too. For all three neurons, at the stable state $w(t_{st})$,

$$\frac{dw(t_{st})}{dt} = 0 \Rightarrow \quad w^T(t_{st}) w(t_{st}) Rw(t_{st}) = w^T(t_{st}) Rw(t_{st}) w(t_{st})$$

Then

$$Rw(t_{st}) = \frac{w^T(t_{st}) Rw(t_{st})}{w^T(t_{st}) w(t_{st})} w(t_{st}) \tag{2.84}$$

That is, $w(t_{st})$ is one eigenvector of R, and its corresponding eigenvalue $\lambda(t_{st})$ is the Rayleigh quotient of R. It still has to be proven that $\lambda(t_{st}) = \lambda_n$. The weight $w(t)$ can be expressed as a function of the orthogonal vectors:

$$w(t) = \sum_{i=1}^{n} f_i(t) z_i \tag{2.85}$$

Replacing it into eq. (2.33), recalling the orthogonality of the eigenvectors, yields

$$\frac{df_i(t)}{dt} = \|w(t)\|_2^{-4} [\, w^T(t) Rw(t) f_i(t) - w^T(t) w(t) \lambda_i f_i(t) \,] \qquad \forall i = 1, 2, \ldots, n \tag{2.86}$$

Define, $\forall i = 1, 2, \ldots, n-1$,

$$\varphi_i(t) = \frac{f_i(t)}{f_n(t)} \tag{2.87}$$

which gives

$$\begin{aligned}\frac{d\varphi_i(t)}{dt} &= \frac{f_n(t)\frac{df_i(t)}{dt} - f_i(t)\frac{df_n(t)}{dt}}{(f_n(t))^2} \\ &\overset{(2.86)}{=} \|w(t)\|_2^{-2}(\lambda_n - \lambda_i)\varphi_i(t)\end{aligned} \tag{2.88}$$

whose solution on $[0, \infty)$ is

$$\varphi_i(t) = \exp\left[(\lambda_n - \lambda_i)\int_0^t \frac{dv}{\|w(v)\|_2^2}\right] \qquad \forall i = 1, 2, \ldots, n-1 \tag{2.89}$$

If λ_n is single, then $\varphi_i(t) \overset{t\to\infty}{\to} 0 \ \forall i = 1, 2, \ldots, n-1$ and then

$$\lim_{t\to\infty} f_i(t) = 0 \qquad \forall i = 1, 2, \ldots, n-1$$

which yields

$$\lim_{t\to\infty} w(t) = w(t_{\text{st}}) = \lim_{t\to\infty} f_n(t) z_n \tag{2.90}$$

Considering as a first approximation the constancy of the weight modulus, it follows that

$$\begin{aligned}\lim_{t\to\infty} f_n(t) &= \pm\|w(0)\|_2 \\ \lim_{t\to\infty} w(t) &= w(t_{\text{st}}) \\ &= \begin{cases} +\|w(0)\|_2 z_n & \forall w/w^T z_n > 0 \\ -\|w(0)\|_2 z_n & \forall w/w^T z_n < 0 \end{cases}\end{aligned}$$

■

Remark 61 (Approximation by the ODE) *As will be apparent in the following section (which deals mainly with the stochastic discrete MCA learning laws), eq. (2.80) is only approximately valid in the first part of the time evolution of the MCA learning laws (i.e., in approaching the minor component).*

Theorem 62 (Time Constant) *If the initial weight vector has modulus less than unity, the MCA EXIN weight vector reaches the minor component direction faster than OJAn, which is faster than LUO. The contrary happens if the modulus of the initial weight vector is more than unity. If the weight remains on the unit sphere,*

the three neurons have the same speed. Furthermore, the three time constants are inversely proportional to $\lambda_{n-1} - \lambda_n$; well-separated eigenvalues give a faster response.

Proof. From eq. (2.89), in the approximation of modulus constancy,

$$\varphi_i(t) = \exp\left[(\lambda_n - \lambda_i)\frac{t}{\|w(0)\|_2^2}\right] \qquad \forall i = 1, 2, \ldots, n-1 \tag{2.91}$$

That is, the MCA EXIN time constant $\tau_{\text{MCA EXIN}}$ (i.e., dominant) is given by

$$\tau_{\text{MCA EXIN}} = \frac{\|w(0)\|_2^2}{\lambda_{n-1} - \lambda_n} \tag{2.92}$$

Repeating the reasonings above for the other two neurons gives

$$\tau_{\text{OJAn}} = \frac{1}{\lambda_{n-1} - \lambda_n} \tag{2.93}$$

$$\tau_{\text{LUO}} = \frac{\|w(0)\|_2^{-2}}{\lambda_{n-1} - \lambda_n} \tag{2.94}$$

The following considerations can be made:

- $\|w(0)\|_2 > 1 \Rightarrow \tau_{\text{LUO}} < \tau_{\text{OJAn}} < \tau_{\text{MCA EXIN}}$
- $\|w(0)\|_2 = 1 \Rightarrow \tau_{\text{MCA EXIN}} = \tau_{\text{OJAn}} = \tau_{\text{LUO}}$
- $\|w(0)\|_2 < 1 \Rightarrow \tau_{\text{MCA EXIN}} < \tau_{\text{OJAn}} < \tau_{\text{LUO}}$ Q.E.D.

Alternative Proof. The Lyapunov function can be used to estimate the transient of the ODE [132] in the limit of its approximation. Indeed, if an upper bound of the Lyapunov function can be found, it is possible to estimate the speed of the transient.[7] In the case of the three neurons, there is no need of an upper bound for a comparison of their transients. Indeed, after defining

$$k \equiv -w^T(t)w(t)Rw(t) + \left[w^T(t)Rw(t)\right]w(t)$$

[7]If, for example, the following upper bound for the Lyapunov function is found:

$$\dot{V} \leq -\rho V$$

then

$$V(x(t)) \leq V(x_0)\, e^{-\rho(t-t_0)}$$

This is an upper bound on $V(x(t))$ $\forall t \geq t_0$ [i.e., for $t \geq t_0$, the state of the system will remain inside the contour $V(x(t)) = V(x_0)\, e^{-\rho(t-t_0)}$]. Therefore, a large ρ is indicative of a fast response toward the origin [132].

the time derivatives can be expressed, except for a constant 2, as

$$\frac{dV\ (w\ (t))}{dt} = \begin{cases} -\dfrac{1}{\|w\ (t)\|_2^4}\ \|k\|_2^2 & \text{for LUO} \\ -\dfrac{1}{\|w\ (t)\|_2^6}\ \|k\|_2^2 & \text{for OJAn} \\ -\dfrac{1}{\|w\ (t)\|_2^8}\ \|k\|_2^2 & \text{for EXIN} \end{cases} \tag{2.95}$$

Integrating the equations with respect to the time gives the same results as in the first proof. ■

Remark 63 (OJA's Time Constant) *OJA has the same time constant as OJAn* [*195, App., p. 455*].

Remark 64 (Low Initial Weights) *From Theorem 62 it may seem that OJAn and LUO can give better results than MCA EXIN by choosing large initial conditions. Unfortunately, this choice is not good because of the flatness of the RQ landscape* (*see Remark 56*)*, the too large fluctuations for FENG* (*see below*) *and the fact that it is to difficult to stop the algorithm reliably* (*see the divergence analysis*) *for all the other learning laws.*

According to the analysis above, the best choice for MCA EXIN would be null initial conditions. However, as can easily be checked, this is not possible. Not only, but too low weights may generate too high oscillations of the weights [see eq. (2.35)]. It is shown in [24] and in the next chapters that the MCA neuron can be endowed with a particular scheduling (DLS scheduling) which allows the neuron to start with infinitesimal initial conditions and remain for a certain time with low weights, just following a stable path in the weight phase diagram. This improved version has been called *MCA EXIN+*.

Remark 65 (Computation of λ_n) *It has been noted experimentally that all MCA learning laws yield the correct value of λ_n well before an accurate estimate of z_n.*

2.6 DYNAMICS OF THE MCA NEURONS

The purpose of this section is the analysis of the temporal behavior of all MCA neurons by using not only the ODE approximation but, above all, the stochastic discrete laws. Indeed, using only the ODE approximation does not reveal some of the most important features of these algorithms. For example, it will be shown that the constancy of the weight modulus [see eq. (2.27)] for OJAn, LUO, and MCA EXIN, which is a consequence of the use of the ODE, is not valid except, as a very first approximation, in approaching the minor component (see

Section 2.5.3). This study will also lead to the very important problem of sudden divergence [181].

The following analysis is original, rethinks and criticizes the existing theory, but above all, proves the superiority of MCA EXIN in the most general MCA problems.

2.6.1 Against the Constancy of the Weight Modulus

2.6.1.1 OJAn, LUO, and MCA EXIN The OJAn, LUO, and MCA EXIN *stochastic discrete learning laws* have a similar analysis, because of their common structure, here repeated for convenience:

$$\begin{aligned} w(t+1) &= w(t) + \delta w(t) = w(t) - \alpha(t) S_{\text{neuron}}(w(t), x(t)) \\ &= w(t) + -\alpha(t) s(p) \left\{ y(t)x(t) - \frac{y^2(t)}{p} w(t) \right\} \end{aligned} \tag{2.96}$$

where

$$s(p) = \begin{cases} p & \text{for LUO} \\ 1 & \text{for OJAn} \\ \dfrac{1}{p} & \text{for EXIN} \end{cases} \tag{2.97}$$

and $p = \|w(t)\|_2^2$. They have in common the following property:

$$w^T(t) \left\{ y(t)x(t) - \frac{y^2(t)}{w^T(t)\, w(t)} w(t) \right\} = 0 \tag{2.98}$$

That is, *the weight increment at each iteration is orthogonal to the weight direction*. This arises from the fact that they are gradient flows of the Rayleigh quotient and exploit the property of orthogonality (2.4). The squared modulus of the weight vector at instant $t+1$ is then given by

$$\|w(t+1)\|_2^2 = \begin{cases} \|w(t)\|_2^2 + \dfrac{\alpha^2(t)}{4} \|w(t)\|_2^2 \|x(t)\|_2^4 \sin^2 2\vartheta_{xw} & \text{for OJAn} \\ \|w(t)\|_2^2 + \dfrac{\alpha^2(t)}{4} \|w(t)\|_2^6 \|x(t)\|_2^4 \sin^2 2\vartheta_{xw} & \text{for LUO} \\ \|w(t)\|_2^2 + \dfrac{\alpha^2(t)}{4} \|w(t)\|_2^{-2} \|x(t)\|_2^4 \sin^2 2\vartheta_{xw} & \text{for MCA EXIN} \end{cases}$$

where ϑ_{xw} is the angle between the directions of $x(t)$ and $w(t)$. From these formulas the dependence of the squared modulus on the square of the learning rate, which is a consequence of the property (2.98), is apparent. As a consequence,

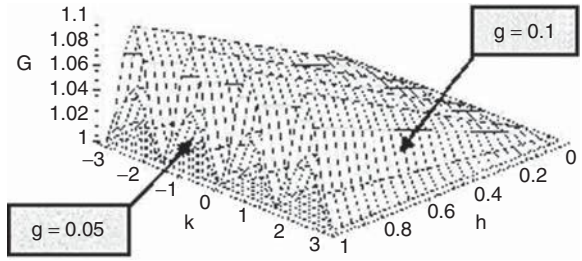


Figure 2.2 Weight modulus increment surfaces for $g = 0.1$ (exterior) and $g = 0.05$ (interior). k is expressed in radians.

for low learning rates (as near convergence), the increase of the weight modulus is less significant. Consider now the ratio G, defined as

$$G = \frac{\|w(t+1)\|_2^2}{\|w(t)\|_2^2} = 1 + gh\sin^2 2\vartheta_{xw}$$
$$= G\left(g\left(\alpha(t), \|w(t)\|_2\right), h, \vartheta_{xw}\right) \tag{2.99}$$

where $h = \|x(t)\|_2^4$ and

$$g = \begin{cases} +\dfrac{\alpha^2(t)}{4}\|w(t)\|_2^4 & \text{for LUO} \\ +\dfrac{\alpha^2(t)}{4}\|w(t)\|_2 & \text{for OJAn} \\ +\dfrac{\alpha^2(t)}{4}\|w(t)\|_2^{-4} & \text{for EXIN} \end{cases} \tag{2.100}$$

Figure 2.2 represents the portion of G greater than 1 (i.e., the modulus increment) as a function of h (here normalized) and $\vartheta_{xw} = k \in (-\pi, \pi]$, parameterized by g. It shows that the increment is proportional to g.

The following considerations can be deduced:

- Except for particular conditions, *the weight modulus always increases*:

$$\|w(t+1)\|_2^2 > \|w(t)\|_2^2 \tag{2.101}$$

  These particular conditions (i.e., all data in exact particular directions) are too rare to be found in a noisy environment.
- The weight modulus does not increase ($G = 1$) if the parameter g is null. In practice, this happens *only* for $\alpha(t) = 0$.
- The parameter g gives the size of the weight modulus increment. If $g \to 0$, then $G \to 1$. Equation (2.100) shows that g is related to the weight modulus according to a power depending on the kind of neuron. MCA EXIN has the biggest increment for weight moduli less than 1; this explains the fact that it

enters the MC direction faster, starting from null or small initial conditions. Furthermore, starting the divergence (it will be shown later that all three laws diverge) implies large moduli; in this case MCA EXIN has the lowest weight modulus increment. In a nutshell, once the MCA direction is reached, the negative power for the weight modulus in g acts as an *inertia*, unlike LUO and OJAn, which are recursively amplified.[8]

- $\sin^2 2\vartheta_{xw}$ is a positive function with four peaks in the interval $(-\pi, \pi]$. This is one of the possible interpretations of the oscillatory behavior of the weight modulus, which will be shown in the simulations.
- h and t depend *only* on the input noisy signal and on the relative positions among the data and the weight vector, so they are impossible to control.

Summarizing: For these three neurons, the property of constant modulus (2.27) is not correct. Notice that in [124] the choice of a unit starting weight vector in the simulations for LUO does not reveal the increase in the modulus.

2.6.1.2 OJA, OJA+, and FENG Following the same analysis as above, it holds that

$$\|w(t+1)\|_2^2 = \|w(t)\|_2^2 + 2\alpha(t)\,\Xi + O\left(\alpha^2(t)\right) \tag{2.102}$$

where

$$\Xi = \begin{cases} y^2(t)\left(\|w(t)\|_2^2 - 1\right) & \text{for OJA} \\ \left(y^2(t) - \|w(t)\|_2^2\right)\left(\|w(t)\|_2^2 - 1\right) & \text{for OJA+} \\ \|w(t)\|_2^2\left(1 - y^2(t)\right) & \text{for FENG} \end{cases}$$

For these neurons the property (2.98) is no longer valid and eq. (2.102) then depends on $\alpha(t)$. This implies a larger increment than for MCA EXIN, OJAn, and LUO (remember that α is in general less than 1). For OJA, the sign of Ξ depends only on the value of the squared modulus; hence, the increment in the squared modulus is always decreasing or increasing according to the initial conditions. This fact is no longer valid for OJA+ and FENG, where an appropriate value of y^2 can change the sign of Ξ.

Near convergence, $y = w^T x < \varepsilon$ with ε low, because w is pointing in the MC direction, whose projection on the data is as small as possible. In this case, for FENG it holds that

$$\|w(t+1)\|_2^2 \longrightarrow \|w(t)\|_2^2 + 2\frac{\alpha(t)}{\lambda_n} + O\left(\alpha^2(t)\right) \tag{2.103}$$

where eq. (2.31) has been used. If λ_n is low, the increase in the squared modulus can be significant. On the contrary, if $\lambda_n < 1$, the OJA+ weight modulus remains constant ($\|w(t)\|_2^2 \to 1$ at convergence).

[8] As is made clear later, a too strong divergence prevents the learning law from having a reliable stop criterion.

2.6.2 Divergence

Restating La Salle's principle of invariance, the Lyapunov function being *weak* (see proof of Theorem 60) yields:

Proposition 66 (Flow Solution Typology) *Let* $\Phi{:}M \to \Re$ *be a smooth function on a Riemannian manifold with compact sublevel sets; that is, for all* $c \in \Re$ *the sublevel set* $\{x \in M \mid \Phi(x) \leq c\}$ *is a compact subset of* M. *Then* [*84*] *every solution* $x(t) \in M$ *of the gradient flow* $\dot{x}(t) = -grad\,\Phi(x(t))$ *on* M *exists for all* $t \geq 0$. *Furthermore,* $x(t) \in M$ *converges to a nonempty compact and connected component of the set of critical points of* Φ *plus* ∞ [*118*] *as* $t \to \infty$.

Corollary 67 (Simple Convergence Behavior) *The solutions of a gradient flow have a particularly simple convergence behavior: There are no periodic solutions, strange attractors, or chaotic behaviors* [*84*].

These reasonings can be applied to $\Phi = r(w, R)$ (i.e., to the MCA learning laws, except FENG). Recalling the degeneracy property of the Rayleigh quotient (see Proposition 44 for the point of view of the Lyapunov direct method), the components of the set of critical points plus ∞ are straight lines.

In Section 2.6.1 we have shown that for MCA EXIN, OJAn, and LUO, the weight increment at each iteration is orthogonal to the weight direction [see eq. (2.98)] and the weight modulus always increases [see eq. (2.101)] (i.e., there is divergence). For initial conditions of modulus greater than 1, OJA also diverges, even if the weight increment is not orthogonal to the weight direction. Every critical point of the ODE for the minimum straight line has a basin of attraction given by the locus of constant modulus equal to the modulus of the critical point. If a critical point of this line is taken as an initial condition, a small perturbation on the weight vector given by the stochastic learning process suffices to move the weight into another basin of larger modulus (see Figure 2.3) and therefore the neuron will converge to another critical point of larger modulus. This happens until the learning rate is null. Considering that $s\left(w^T(t)\,w(t)\right)$ [see eq. (2.97)] is inversely proportional to $w^T(t)\,w(t)$ only for MCA EXIN, its weight increment is smaller and therefore the weight will converge to a nearer critical point with respect to OJAn and LUO. It is evident that the ODE approximation can be accepted only outside the minimum straight line, because of its degeneracy. The theorems (see [84, Prop. 3.6; 118, Cor. 2]) are no longer valid to infer the asymptotic behavior of the stochastic law. The dynamic behavior of the weight vector, except for OJA+ and FENG, can be described in the following way:

1. There is an initial transient.
2. There are fluctuations around the locus of constant modulus, but with an increasing bias; the fluctuations are a function of the learning rate, as will be shown later.
3. Arrival is in the direction desired.

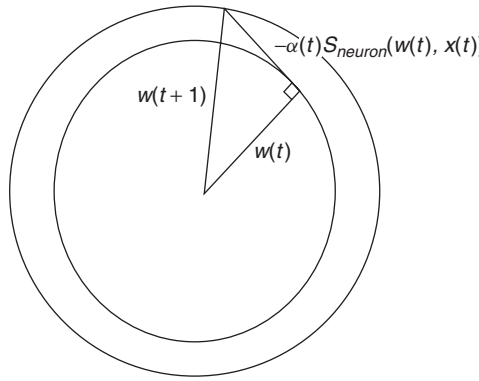


Figure 2.3 Evolution of the weight vector during updating.

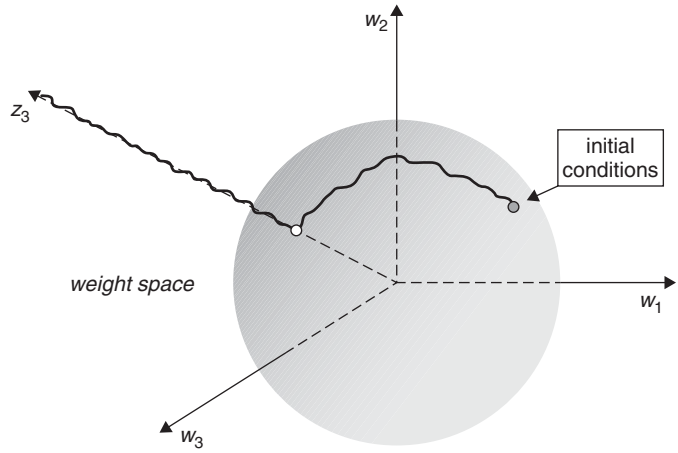


Figure 2.4 Dynamic behavior of the weights for MCA EXIN, LUO, OJAn, and OJA (for particular initial conditions): three-dimensional case.

4. By increasing the modulus, the fluctuations push the weight to another critical point, and so on until ∞.

This analysis is illustrated in Figure 2.4 for the three-dimensional case.

In the remaining part of this section, the divergence is studied in further detail and the sudden divergence phenomenon is illustrated.

2.6.2.1 OJAn, LUO, MCA EXIN, and the General Case All the analysis above justifies the following fundamental theorem:

Theorem 68 (MCA Divergence) *LUO, OJAn, and MCA EXIN do not converge. MCA EXIN has the slowest divergence; LUO has the fastest divergence.*

Averaging the behavior of the weight vector after the first critical point has been reached ($t \geq t_0$), the following formula can be written:

$$w(t) = \|w(t)\|_2 \, z_n \qquad \forall t \geq t_0 \tag{2.104}$$

where z_n is the unit eigenvector associated with the smallest eigenvalue of R. From eq. (2.96) the discrete law can easily be deduced for updating the weight modulus. This discrete law can be regarded as the discretization of the following ODE:

$$\frac{d\,w^T w}{dt} = s^2(w^T w) E\left[\left\| yx - \frac{y^2}{w^T w} w \right\|_2^2\right] \tag{2.105}$$

Without loss of generality, the input data are considered as Gaussian; after some matrix algebra, which can be found in [181], the following equation is derived:

$$\frac{dp}{dt} = s^2(p)\left[-\lambda_n^2 + \lambda_n \operatorname{tr}(R)\right] p \tag{2.106}$$

where λ_n is its smallest eigenvalue of R, the input data autocorrelation matrix. Solving this differential equation for $s(p) = 1$ (OJAn), with the initial condition $p(0) = 1$ for the sake of simplicity, yields

$$p(t) = e^{\left[-\lambda_n^2 + \lambda_n \operatorname{tr}(R)\right] t} \tag{2.107}$$

and considering that the quantity in brackets is never negative, it follows that

$$p(t) \to \infty \qquad \text{when} \quad t \to \infty \text{ exponentially} \tag{2.108}$$

Hence, the norm of $w(t)$ diverges very fast. This divergence also arises when $s(p) = p$ (LUO); indeed,

$$p(t) = \frac{1}{\sqrt{1 - 2\left[-\lambda_n^2 + \lambda_n \operatorname{tr}(R)\right] t}} \tag{2.109}$$

In this case the divergence happens in a *finite* time (*sudden divergence*); that is,

$$p(t) \to \infty \qquad \text{when} \quad t \to t_\infty \tag{2.110}$$

where

$$t_\infty = \frac{1}{2\left[-\lambda_n^2 + \lambda_n \operatorname{tr}(R)\right]} \tag{2.111}$$

Hence, t_∞ depends on the spreading of the eigenvalue spectrum of R; if the eigenvalues of R are clustered, the sudden divergence appears late. Furthermore, t_∞ is proportional to the inverse of λ_n (high λ_n means noisy data).

Finally, for $s(p) = 1/p$ (MCA EXIN),

$$p(t) = \sqrt{1 + 2\left[-\lambda_n^2 + \lambda_n \operatorname{tr}(R)\right] t} \tag{2.112}$$

still divergence, but at a slower rate. This confirms Theorem 68, which is based on the analysis of Section 2.6.1.

For the general case of eq. (2.96), it is easy to verify that the condition for sudden divergence is given by

$$\int_{p_0}^{\infty} \frac{dp}{p s^2(p)} < \infty \tag{2.113}$$

where p_0 is the value of p at $t = 0$ (note that $p_0 = 0$ is excluded because it is outside the domain of the RQ of R). If $s(p)$ is a polynomial in p of degree β, then

$$p\, q^2(p) \sim p^{2\beta+1} \qquad \text{for} \quad p \to \infty \tag{2.114}$$

Hence, for $\beta > 0$ the integral (2.113) converges and the sudden divergence holds.[9]

2.6.2.2 OJA The sudden divergence of OJA can be analyzed both with the analysis above (see [181]) and using the corresponding ODE (2.20). In this case it holds that

$$\frac{d \|w(t)\|_2^2}{dt} = 2\left[w^T(t) R w(t)\right]\left(\|w(t)\|_2^2 - 1\right) \tag{2.115}$$

Assuming that the MC direction has been approached and recalling Remark 65, eq. (2.115) can be approximated as

$$\frac{dp}{dt} = 2\lambda_n p\,(p - 1) \tag{2.116}$$

where $p = \|w(t)\|_2^2$, as usual. Notice the importance of $\operatorname{sgn}(p - 1)$ with respect to the increment of the weight modulus, as seen in Section 2.6.1. Define the instant of time in which the MC direction is approached as t_0 and the corresponding

[9]What is the best possible $\beta < 0$ that can be chosen in devising a novel MCA learning law? Certainly, as seen before, a large $|\beta|$ has a positive inertial effect on the weight modulus. In this case it can always be classified as RQ gradient flow with its own Riemannian metric. However, for increasing $|\beta|$, the flop cost per iteration quickly becomes prohibitive, above all for high-dimensional weight vectors. On the contrary, $\beta = 2$ (MCA EXIN) yields the best compromise between inertia and computational cost.

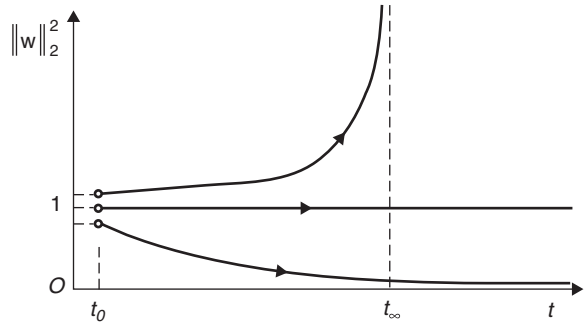


Figure 2.5 Asymptotic behavior of the OJA ODE for different values (circles) of the initial conditions.

value of the squared weight modulus as p_0.[10] The solution of eq. (2.116) is given by

$$p = \begin{cases} \dfrac{p_0}{p_0 + (1 - p_0)\, e^{2\lambda_n (t - t_0)}} & \text{if } p_0 \neq 1 \quad (2.117) \\ p_0 & \text{if } p_0 = 1 \quad (2.118) \end{cases}$$

Figure 2.5 shows these results for different values of p_0. If $p_0 > 1$, OJA diverges at the finite time t_∞ (sudden divergence) given by

$$t_\infty = t_0 + \frac{1}{2\lambda_n} \ln \frac{p_0}{p_0 - 1} \quad (2.119)$$

This expression shows that $p_0 \to 1^+$ is mandatory in order to have the sudden divergence as late as possible. Unfortunately, it is not always possible to choose the initial conditions (e.g., in slowly varying systems for real-time applications). Furthermore, a matrix R with high λ_n (noise in the data) worsens the sudden divergence problem.

If $p_0 < 1$,[11] the rate of decrease of p increases with λ_n (noisy data) and decreases with p_0. Too high a decrease of p toward 0 may create the same problems in stopping the algorithm as the sudden divergence does.

2.6.2.3 OJA+ By using the OJA+ ODE and repeating the procedure used for OJA, the following result is obtained:

$$\frac{dp}{dt} = 2\,(\lambda_n - 1)\, p\, (p - 1) \quad (2.120)$$

[10]The value p_0 is equal to the starting value of p in the ODE approximation.
[11]$p = p_0 = 1$ is unstable.

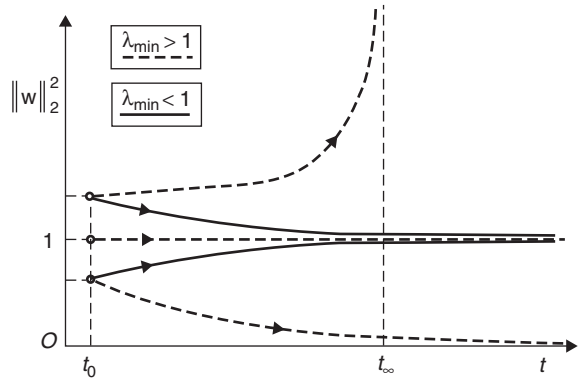


Figure 2.6 Asymptotic behavior of the OJA+ ODE for different values (circles) of the initial conditions.

whose solution is given by

$$p = \frac{p_0}{p_0 + (1 - p_0)\, e^{2(\lambda_n - 1)(t - t_0)}} \qquad \text{if } p_0 \neq 1 \tag{2.121}$$

The change in the time constant of the exponential with respect to OJA implies a fundamental difference. Indeed, if $\lambda_n > 1$, OJAn behaves as OJA (the OJA expressions are valid here simply by replacing λ_n with $\lambda_n - 1$). If $\lambda_n = 1$, $p = p_0$. If $\lambda_n < 1$, $p \to 1$, as expected from Theorem 48. Figure 2.6 summarizes the results for different values of p_0.

In summary: OJA+ is not divergent and it does not suffer from the sudden divergence, but, unfortunately, it requires the assumption that the smallest eigenvalue of the autocorrelation matrix of the input data is less than unity. If it cannot be assumed in advance (e.g., for noisy data), OJA+ may suffer from the sudden divergence.

2.6.2.4 Simulation Results for the MCA Divergence The previous analysis is illustrated by using, as a benchmark, the example in [124, pp. 294–295] and [181]. A zero-mean Gaussian random vector $x(t)$ is generated with the covariance matrix

$$R = \begin{bmatrix} 0.4035 & 0.2125 & 0.0954 \\ 0.2125 & 0.3703 & 0.2216 \\ 0.0954 & 0.2216 & 0.4159 \end{bmatrix} \tag{2.122}$$

and taken as an input vector. The learning rate is given by $\alpha(t) = \text{const} = 0.01$. The algorithms are initialized at the true solution; that is,

$$w(0) = [0.4389 \quad -0.7793 \quad 0.4473]^T$$

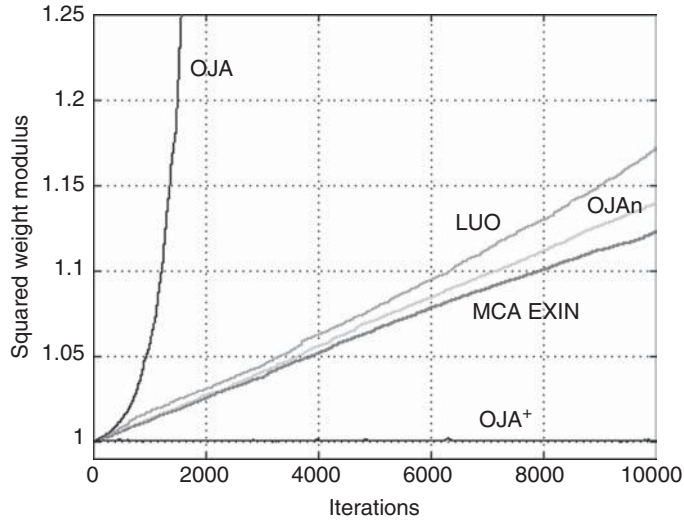


Figure 2.7 Divergences (represented by the evolution of the squared weight modulus) of the MCA linear neurons with initial conditions equal to the solution. (*See insert for color representation of the figure*.)

corresponding to $\lambda_n = 0.1235 < 1$. Figure 2.7 [181] shows the different divergences of $\|w(t)\|_2^2$: OJA yields the worst result (sudden divergence), MCA EXIN has the slowest divergence, and OJA+ converges. Figure 2.8 shows the deviation from the MC direction for LUO. The effect of the orthogonal weight increments and the start of the sudden divergence are apparent. Figure 2.9 shows the good approximation of the ODE (2.106) with respect to the LUO stochastic law and the good estimation of t_∞ by eq. (2.111).

2.6.2.5 Conclusions Except for FENG, whose limits are very serious and will be analyzed later, all other MCA neurons diverge and some of them are plagued with sudden divergence.

Proposition 69 (MCA Divergence) *LUO and OJA diverge at a finite time* (*sudden divergence*). *OJA+ behaves as OJA for noisy input data. OJAn and MCA EXIN diverge at infinite time, but the former at a faster rate.*

Proposition 70 (Noisy Input Data) *Noisy input data (i.e., high* λ_n*), worsen the neuron features. LUO and OJA diverge before the rate of divergence of OJAn* [*which depends exponentially on* λ_n*; see eq.* (*2.107*)], *which increases more than that of MCA EXIN* [*which depends on* $\sqrt{\lambda_n}$*; see eq.* (*2.112*)]. *OJAn behaves as OJA and then diverges at finite time.*

In the next section we analyze the problems caused by the divergence.

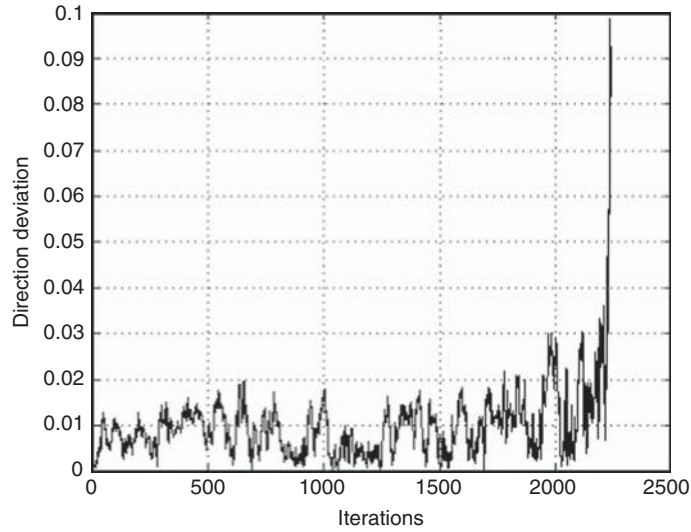


Figure 2.8 Deviation from the MC direction for LUO, measured as the squared sine of the angle between the weight vector and the MC direction.

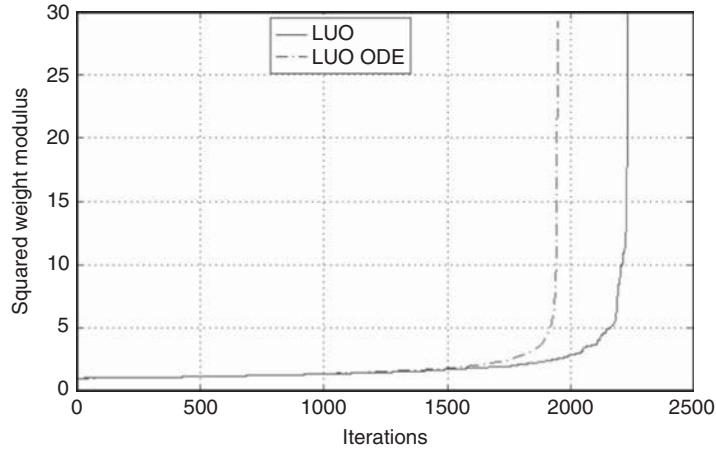


Figure 2.9 Check of the validity of the ODE approximation for LUO.

2.6.2.6 Stop Criteria It is impossible to use the usual stop criteria for the MCA learning laws, because the error cost does not go to zero at convergence. Indeed, the minimum of the MCA error cost is equal to the smallest eigenvalue of the autocorrelation matrix. Hence, it is necessary to detect the flatness of the curves representing either the weight components or the error function. This problem is complicated by the difficulties described in Proposition 69. So this type of detection is valid only for MCA EXIN, which has the slowest divergence.

In all the simulations of this chapter, the following numerical stop criterion has been chosen.

Define

$$\Delta w(k) = w(k) - w(k-1) \tag{2.123}$$

where $w \in \Re^n$ and k is the iteration; then the learning law stops when

$$\|\Delta w(k)\|_\infty \equiv \max_i |\Delta w_i(k)| < \varepsilon \tag{2.124}$$

for T iterations, where the L_∞ or Chebyshev norm has been taken into account; T is, in general, equal to the number of vectors of the training set or to a predefined number in case of online learning and ε is very small.

Another interesting stop criterion, which exploits the MCA EXIN ODE because of its slow divergence, has been devised, inspired by [78, p. 140]. Instead of testing the convergence of the sequence $\{r(w,R)\}$, which is not suitable for comparing different algorithms because of its dependence on the sequence $\{w(t)\}$, this criterion is based on the squared length of the gradient of $r(w,R)$:

$$\|\text{grad}\, r(w,R)\|_2^2 = \|w\|_2^{-2} \left[\|w\|_2^{-2} \|Rw\|_2^2 - r^2(w,R) \right] \tag{2.125}$$

Considering the fact that $\text{grad}\, r(w,R)$ is inversely proportional to $\|w\|_2^2$ and the stopping criterion cannot depend on $\|w\|_2$, a normalization by $\|w\|_2^2$ is necessary. Furthermore, to avoid the dependence on the scale of R [if R is multiplied by a constant, $r(w,R)$; see eq. (2.2), and $\text{grad}\, r(w,R)$ is also multiplied by the same constant], the following stop criterion holds:

$$\|\text{grad}\, r(w,R)\|_2^2 \|w\|_2^2 r^{-2}(w,R) = \left(\|w\|_2^{-2} \|Rw\|_2^2 r^{-2}(w,R) - 1 \right) < \varepsilon \tag{2.126}$$

where ε is a prescribed threshold. If the initial guess $w(t_0)$ is selected very near an eigenvector, the test values will be small in the first few iterations. Hence, it is recommended that the process not be stopped until the test values are below the threshold for several consecutive iterations. The same problem may happen even if the learning trajectory crosses the solution without staying on it.

2.6.3 Limits of the Feng Neuron

Even if it converges to a solution [see eq. (2.31)], the FENG learning law has some unacceptable problems: the high oscillations around the solution, which prevent the choice of a reliable stop criterion, and the difficulty in controlling the rate of convergence (see Theorem 71). Figure 2.10 shows a comparison with MCA EXIN in the computation of the smallest eigenvalue of a matrix R, whose

eigenvalues are 0, 1, and 2. The initial weight components have been chosen randomly in the interval $[0, 1]$. The OJA learning law (2.16) can be interpreted in the following way:

$$w(t+1) = w(t) \underbrace{-\alpha(t)\, y(t)\, x(t)}_{\text{anti-Hebbian}} + \alpha(t)\, w(t)\, y^2(t) \tag{2.127}$$

and compared with the FENG learning law (2.29):

$$w(t+1) = w(t) - \alpha(t)\, \|w(t)\|_2^2\, y(t)\, x(t) + \alpha(t)\, w(t) \tag{2.128}$$

In approaching convergence, (2.31) holds, so $\|w(t)\|_2^2 \to 1/\lambda_n$. Hence, the FENG law becomes

$$w(t+1) \approx w(t) \underbrace{-\gamma(t)\, y(t)\, x(t)}_{\text{anti-Hebbian}} + \lambda_n \gamma(t)\, w(t) \tag{2.129}$$

where $\gamma(t) = \alpha(t)/\lambda_n$. With regard to OJA, the term $y^2(t)$ does not appear. According to Oja [138], this term implies a normalization of the weight vector and has a stabilizing effect on the anti-Hebbian law. Indeed, for small γ,

$$\begin{aligned} w(t+1) &= \frac{w(t) - \gamma\, y(t)\, x(t)}{1-\gamma} \\ &\approx w(t) - \gamma\, y(t)\, x(t) + \gamma w(t) + O(\gamma^2) \end{aligned} \tag{2.130}$$

which means that the FENG learning law, being $0 \leq \gamma < 1$, *amplifies* the anti-Hebbian law, and this fact is more relevant just when it approaches the MC direction. From (2.130), the relative error of the FENG estimate can be deduced:

$$\left.\frac{\Delta w(t)}{w(t)}\right|_{\text{FENG}} = \frac{w(t+1) - w(t)}{w(t+1)} = \gamma \tag{2.131}$$

Then, decreasing γ, the oscillations around the MC direction decay. Indeed, the term $y^2 = w^T x x^T w$ in (2.127) tends, on average, to $w^T R w$. At very near convergence, $w^T R w \to w^T w \lambda_n$. If $w^T w \to 1/\lambda_n$, as for FENG [see eq. (2.31)], it can be stated that $y^2 \to 1$, so after a certain number of iterations, in general very large, the anti-Hebbian rule becomes as constrained as OJA. This phenomenon is apparent in Figure 2.11, which shows the last iterations of the example of Figure 2.10.

Theorem 71 (Time Constant) *In the limit of the FENG ODE approximation (2.30), that is, in the converging phase to the minor component, the decay time constant is the same as LUO and, close to convergence, depends only on the autocorrelation matrix spectrum.*

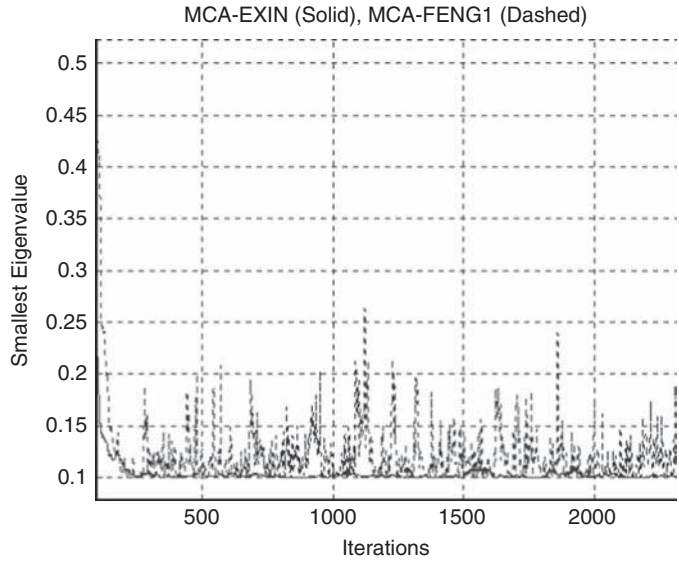


Figure 2.10 Smallest eigenvalue computation by MCA EXIN and FENG for a three-dimensional well-conditioned autocorrelation matrix.

Proof. Consider the FENG ODE (2.30) and replace $w(t)$ with the expression (2.85):

$$\frac{df_i(t)}{dt} = -w^T(t)\, w(t)\, \lambda_i f_i(t) + f_i(t) \qquad \forall i = 1, \ldots, n \tag{2.132}$$

Using the notation and the same analysis as in the proof of Theorem 16, it holds that

$$\frac{d\varphi_i(t)}{dt} = \|w(t)\|_2^2 (\lambda_n - \lambda_i)\, \varphi_i(t) \tag{2.133}$$

which is coincident with the formula for LUO [124, App.]. Recalling that, near convergence, $\|w(t)\|_2^2 \to 1/\lambda n \forall i = 1, \ldots, n-1$, it holds that

$$\tau_{\text{FENG}} = \frac{\|w(0)\|_2^{-2}}{\lambda_{n-1} - \lambda_n} \xrightarrow{\text{convergence}} \frac{\lambda_n}{\lambda_{n-1} - \lambda_n} \tag{2.134}$$

■

2.6.4 Weight Increments and RQ Minimal Residual Property

The MCA learning laws are iterative and so have the common form

$$w(t+1) = w(t) + \alpha(t)\, \Delta w(t) \tag{2.135}$$

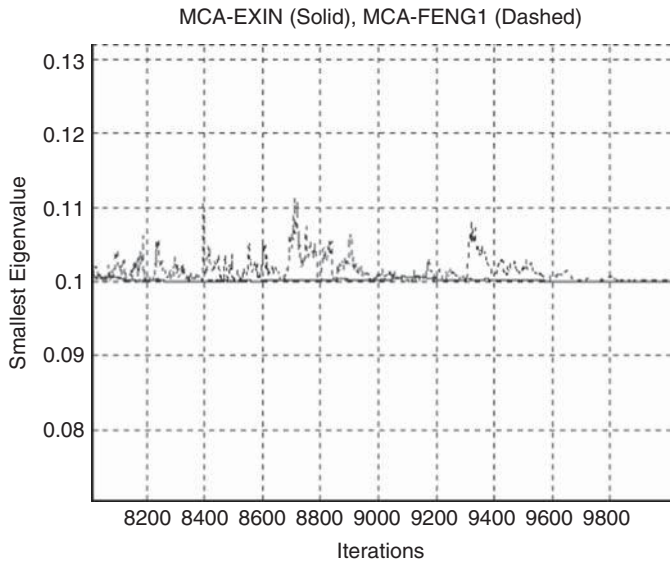


Figure 2.11 Smallest eigenvalue computation by MCA EXIN and FENG (last iterations).

Equation (2.38) implies that for high $\|w(t)\|_2$, as during the divergence in the MC direction,

$$\Delta w(t)|_{\text{EXIN}} \leq \Delta w(t)|_{\text{OJAn}} \leq \Delta w(t)|_{\text{LUO}} \tag{2.136}$$

An analogous comparison can be made between MCA EXIN and the other neurons by using the RQ minimal residual property (2.5), which can be written as

$$\|(R - r(w(t))I)\, w(t)\| \leq \|(R - \mu I)\, w(t)\| \tag{2.137}$$

R being, as usual, the autocorrelation matrix. Discretizing, it holds that

$$\left\| y(t)x(t) - \frac{y^2(t)w(t)}{\|w(t)\|_2^2} \right\| \leq \|y(t)x(t) - \mu w(t)\| \tag{2.138}$$

Under the same assumption of high $\|w(t)\|_2$, it follows that

$$\Delta w(t)|_{\text{EXIN}} \leq \Delta w(t)|_{\text{OJA}} \tag{2.139}$$

by using (2.137) and $\mu = y^2(t)$,

$$\Delta w(t)|_{\text{EXIN}} \leq \Delta w(t)|_{\text{OJA+}} \tag{2.140}$$

by using (2.137) and $\mu = y^2(t) + 1 - \|w\,(t)\|_2^2$,

$$\Delta w(t)|_{\text{EXIN}} \leq \Delta w(t)|_{\text{FENG}} \tag{2.141}$$

by using (2.137) and $\mu = \|w\,(t)\|_2^{-2}$. In this case, as the iterations increase, $\mu \to \lambda_n$ [see (2.31)] and then $\Delta w(t)|_{\text{FENG}} \to \Delta w(t)|_{\text{EXIN}}$ as seen in Section 2.6.3.

2.7 FLUCTUATIONS (DYNAMIC STABILITY) AND LEARNING RATE

The learning rate $\alpha\,(t)$ must be quite small to avoid the instability and consequent divergence of the learning law. It implies some problems:

- A small learning rate gives a slow learning speed.
- It is difficult to find a good learning rate to prevent learning divergence.
- The transient and the accuracy in the solution are both affected by the choice of the learning rate.

Then the study of the stochastic discrete learning laws of the three neurons with respect to $\alpha\,(t)$ becomes an analysis of their dynamics. Define:

$$r' = \frac{\left|w^T\,(t+1)\,x\,(t)\right|^2}{\|w\,(t+1)\|_2^2} \tag{2.142}$$

$$r = \frac{\left|w^T\,(t)\,x\,(t)\right|^2}{\|w\,(t)\|_2^2} \tag{2.143}$$

$$\rho\,(\alpha) = \frac{r'}{r} \geq 0, \qquad p = \|w\,(t)\|_2^2, \qquad u = y^2\,(t)$$

The two scalars r' and r represent, respectively, the squared perpendicular distance from the input $x(t)$ to the data-fitting hyperplane whose normal is given by the weight and passes through the origin, after and before the weight increment. Recalling the definition of MC, it should hold that $r' \leq r$. If this inequality is not valid, this means that the learning law increases the estimation error due to disturbances caused by noisy data. When this increase is too large, it will make $w\,(t)$ deviate drastically from normal learning, which may result in divergence or *fluctuations* (implying an increased learning time). This problem is here called *dynamic instability* and the possible divergence is here defined as (*dynamic*) *instability divergence*. In the remaining part of this section we deal with the analysis of ρ for each MCA learning law.

2.7.1 OJAn, LUO, and MCA EXIN

By using eq. (2.96), the following formulas hold:

$$w^T(t+1)\,x(t) = y(t)\left[1-\alpha\, s(p)\left(\|x(t)\|_2^2 - \frac{u}{p}\right)\right] \tag{2.144}$$

$$\|w(t+1)\|_2^2 = p + \alpha^2 s^2(p)\, u\, q \tag{2.145}$$

where

$$q = \left\| x(t) - \frac{y(t)\, w(t)}{p} \right\|_2^2 \tag{2.146}$$

Notice that[12]

$$q = \left(\|x(t)\|_2^2 - \frac{u}{p}\right) = \|x(t)\|_2^2 \sin^2 \vartheta_{xw} \tag{2.147}$$

where ϑ_{xw} is the angle between the directions of $x(t)$ and $w(t)$. So eq. (2.144) can be rewritten as

$$w^T(t+1)\,x(t) = y(t)\left[1-\alpha\, s(p) q\right] \tag{2.148}$$

Hence,

$$\rho(\alpha) = \frac{p\left[1-\alpha\, s(p) q\right]^2}{p + \alpha^2 s^2(p)\, u\, q} \tag{2.149}$$

It follows that

$$\rho(\alpha) > 1 \Leftrightarrow p\,(1-\alpha\, s(p) q)^2 > p + \alpha^2 s^2(p)\, u\, q \tag{2.150}$$

which yields

$$\alpha(\alpha - \alpha_b) > 0 \tag{2.151}$$

being

$$\alpha_b = \frac{2p\, s^{-1}(p)}{pq - u} \tag{2.152}$$

In particular,

$$\alpha_{b\mathrm{EXIN}} = p\alpha_{b\mathrm{OJAn}} = p^2 \alpha_{b\mathrm{LUO}} \tag{2.153}$$

[12]From this point on, this analysis diverges from the analysis in [195, proof of Prop. 2], because of a mistake in the unnumbered equation about Δ^2 proceding eq. (A13). Then all the consequent theory in [195, proof of Prop. 2] is incorrect.

Hence,

$$\rho(\alpha) > 1 \Leftrightarrow \begin{cases} \alpha > \alpha_b(x, w(t)) & \text{if } \alpha_b > 0 \\ \alpha > 0 & \text{if } \alpha_b < 0 \end{cases} \tag{2.154}$$

Considering that it always holds that $\alpha \geq 0$, if α_b is negative, then r' is always greater than r independent of the value of the learning rate α. Here this phenomenon is called *negative instability*. The following cases can generate instability (fluctuations or divergence):

1. $\alpha_b < 0$ (negative instability). This case is equivalent to

$$pq - u = A(q, p, u) < 0 \tag{2.155}$$

Then there is negative instability when

$$q = \|x(t)\|_2^2 \sin^2 \vartheta_{xw} < \frac{u}{p} = \|x(t)\|_2^2 \cos^2 \vartheta_{xw} \tag{2.156}$$

That is,

$$\tan^2 \vartheta_{xw} < 1 \tag{2.157}$$

which means that there is negative instability for $-\pi/4 < \vartheta_{xw} < \pi/4$ and for $-3\pi/4 < \vartheta_{xw} < 3\pi/4$. This is illustrated in Figure 2.12, which shows the instability bounds in the weight space with respect to the input vector $x(t)$. Considering that in steady state, $\vartheta_{xw} \to \pm\pi/2$, because of the definition of MC (the dotted area in Figure 2.12), the negative instability is typically valid for the *transient state* and in case of *outliers*. This study holds for all three neurons.

2. $0 < \alpha_b \leq \gamma < 1$. In this case, $\alpha > \alpha_b$ generates instability. It implies that too low a value of γ certainly requires a lower learning rate α to avoid this problem. As a consequence, γ can be considered as an upper bound for the learning rate. This case is equivalent to $A(q, p, u) > 0$ and $B(q, p, u) > 0$, where

$$B(q, p, u) = (\gamma p) q - \left(\gamma u + \frac{2p}{s(p)}\right)$$

Hence, there are no fluctuations (instability) when

$$0 \leq \frac{u}{p} < q < \frac{u}{p} + \frac{1}{\gamma} \frac{2}{s(p)} \tag{2.158}$$

which becomes

$$1 < \tan^2 \vartheta_{xw} < 1 + \frac{1}{\gamma} \frac{2}{s(p)} \frac{1}{\|x(t)\|_2^2 \cos^2 \vartheta_{xw}} = 1 + \Delta_{\text{st}} \tag{2.159}$$

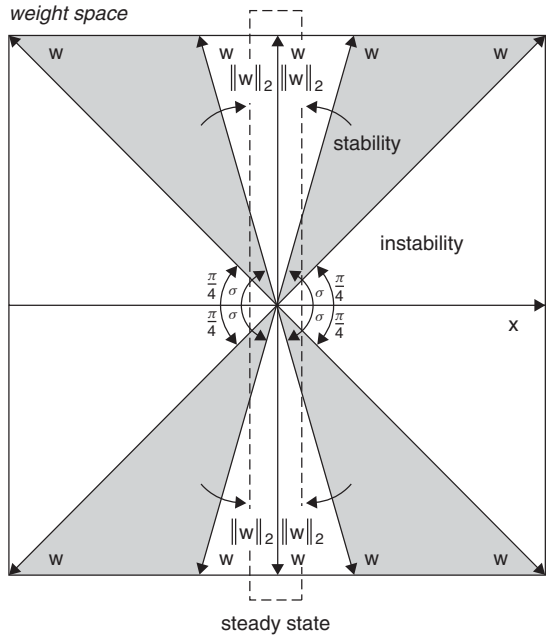


Figure 2.12 MCA EXIN: stability subspaces of the weight space with respect to the input vector (two-dimensional space). The angle σ is proportional to the weight modulus. The negative instability bounds are also valid for OJAn and LUO.

The positive scalar Δ_{st} represents the interval of absence of fluctuations (dynamic stability) and depends on the following quantities:

- γ. Near convergence, α is very low, because of the Robbins–Monro conditions [161]; then γ can be taken very low, and this choice improves the dynamic stability.
- $s(p)$. This quantity has different expressions according to the learning law [see eq. (2.97)]; as discussed before (see Section 2.6.2.1 and Remark 64), the initial weight vector must be chosen low; then, after a certain number of iterations, the weight vector diverges; near convergence, Δ_{st} for LUO decreases, so its dynamic stability worsens; on the contrary, the weight divergence has a positive effect on the dynamic stability of MCA EXIN (see Figure 2.12, where the angle σ increases for increasing p).
- $x(t)$. Data with a high modulus and close to the weight direction worsen the dynamic stability.

Remark 72 (Bias and Variance) *From the point of view of the variance/bias dilemma the MCA EXIN learning law is a high variance/low bias algorithm. Indeed, choosing low initial conditions implies a high variance in the first temporal part of the weight time evolution, but not at the expenses of the final accuracy,*

because, as seen before, in the divergence phase the fluctuations decrease to zero. The other algorithms do not have this feature; they need low initial conditions (which implies high variance) and have low bias, but with high oscillations around the solution for a longer time than MCA EXIN.

2.7.2 OJA

The analysis on OJA can be found in [195, proof of Prop. 2]. There is dynamic instability ($\rho > 1$) when

$$\cos^2 \vartheta_{xw} < \frac{1}{2p} \quad \wedge \quad \alpha > \alpha_b \tag{2.160}$$

being

$$\alpha_b = \frac{2}{\|x\,(t)\|_2^2 \left(1 - 2p \cos^2 \vartheta_{xw}\right)} \tag{2.161}$$

The first condition implies the absence of the negative instability (i.e., $\alpha_b > 0$). In reality, the first condition is contained in the second condition. Indeed, considering the case $0 < \alpha_b \leq \gamma < 1$, it holds that

$$\cos^2 \vartheta_{xw} \leq \frac{1}{2p} - \frac{1}{\gamma p \, \|x\,(t)\|_2^2} = \Upsilon \tag{2.162}$$

which is more restrictive than (2.178). Figure 2.13 shows this condition, where $\sigma = \arccos \sqrt{\Upsilon}$. The instability domain is the contrary of the domain of the neurons considered previously. The angle σ is proportional to $1/\sqrt{2p}$ and then an increasing weight modulus (as close to the MC direction for $\lambda_n > 1$) better respects this condition. Also, the decrease of γ has a similar positive effect. From Figure 2.13 it is apparent that in the transient (in general, low ϑ_{xw}), there are fewer fluctuations than in the neurons cited previously. Then OJA has a low variance but a higher bias than MCA EXIN [see eq. (2.139)].

2.7.3 FENG

From eq. (2.29) it follows that

$$w^T\,(t+1)\,x\,(t) = y\,(t)\,(1 - \alpha q) \tag{2.163}$$

where

$$q = p\,\|x\,(t)\|_2^2 - 1 \tag{2.164}$$

and

$$\|w\,(t+1)\|_2^2 = p\left[1 - 2\alpha\,(u-1) + \alpha^2\,(qu - u + 1)\right] \tag{2.165}$$

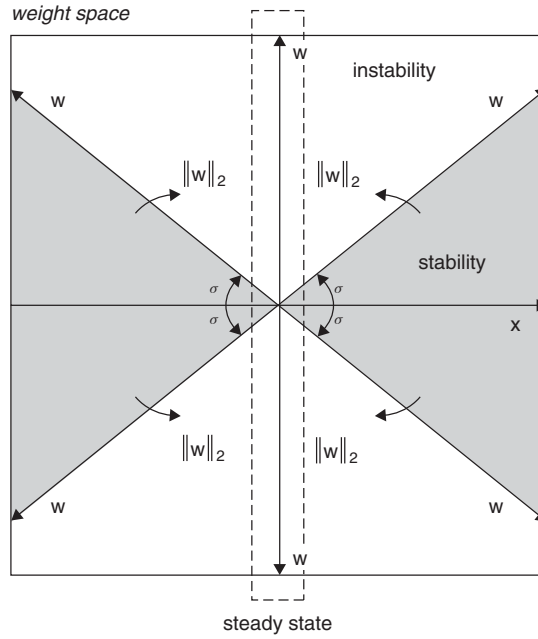


Figure 2.13 OJA and OJA+: stability subspaces of the weight space with respect to the input vector (two-dimensional space). The angle σ is proportional to the weight modulus.

Hence,

$$\rho(\alpha) = \frac{(1-\alpha q)^2}{1-2\alpha(u-1)+\alpha^2(qu-u+1)} \tag{2.166}$$

Then $\rho(\alpha) > 1$ (dynamic instability) iff

$$\alpha(\alpha-\alpha_b)\underbrace{\left(q^2-qu+u-1\right)}_{\Gamma} > 0 \tag{2.167}$$

where

$$\alpha_b = \frac{2}{q-1} = \frac{2}{p\|x(t)\|_2^2 - 2} \tag{2.168}$$

and

$$\Gamma = p\|x(t)\|_2^2 \sin^2\vartheta_{xw}\left(p\|x(t)\|_2^2 - 2\right) \tag{2.169}$$

The dynamic instability condition is then

$$\alpha > \alpha_b = \frac{2}{q-1} \quad \wedge \quad q > 1 \tag{2.170}$$

The second condition implies the absence of the negative instability (i.e., $\alpha_b > 0$). Following the strategy in Section 2.7.2.1, only the case $0 < \alpha_b \leq \gamma < 1$ is taken into consideration. There is a possible dynamic instability if

$$p > \frac{2}{\|x\,(t)\|_2^2}\left(1 + \frac{1}{\gamma}\right) \tag{2.171}$$

Hence, the instability does not depend on ϑ_{xw}, only on the weight modulus; so unlike the other neurons, the FENG stability does not improve when approaching the MC direction ($\vartheta_{xw} \to \pm\pi/2$). However, near convergence, γ can be taken very low, and this choice improves the dynamic stability. By considering that $p \to 1/\lambda_n$ [see eq. (2.31)], it follows that an autocorrelation matrix with too low a value of λ_n (near-singular matrix) can cause dynamic instability.

2.7.4 OJA+

For OJA+ the analysis is the same as above. It holds that

$$\rho(\alpha) = \frac{p\,(1 - \alpha q)^2}{p - 2\alpha\,(1 - p)\,(u - p) + \alpha^2 H} \tag{2.172}$$

where

$$q = \|x\,(t)\|_2^2 - u - 1 + p \tag{2.173}$$

and

$$H = u\,(q + p) + p^3 + (1 + u)\left((1 - p)\,(p - u) - p^2\right) \tag{2.174}$$

Then $\rho(\alpha) > 1$ (dynamic instability) iff

$$\alpha\,(\alpha - \alpha_b)\; p\,\|x\,(t)\|_2^2 \sin^2\vartheta_{xw}\left(2q - \|x\,(t)\|_2^2\right) > 0 \tag{2.175}$$

where

$$\alpha_b = \frac{2}{2q - \|x\,(t)\|_2^2} \tag{2.176}$$

The dynamic instability condition is then

$$\alpha > \frac{2}{2q - \|x\,(t)\|_2^2} \quad \wedge \quad 2q - \|x\,(t)\|_2^2 > 0 \tag{2.177}$$

The second condition implies the absence of the negative instability (i.e., $\alpha_b > 0$). It can be rewritten as

$$\cos^2\vartheta_{xw} \leq \frac{1}{2p} + \frac{1}{\|x\,(t)\|_2^2}\frac{p - 1}{p} \tag{2.178}$$

In reality, as for OJA, the second condition is contained in the first. Indeed, considering the case $0 < \alpha_b \leq \gamma < 1$, it holds that

$$\cos^2 \vartheta_{xw} \leq \left(\frac{1}{2p} + \frac{1}{\|x\,(t)\|_2^2} \frac{p-1}{p} \right) - \frac{1}{\gamma p \, \|x\,(t)\|_2^2} = \Pi \tag{2.179}$$

which is more restrictive than (2.178). Figure 2.13 shows this condition, where $\sigma = \arccos \sqrt{\Pi}$. This angle is proportional to $1/\sqrt{2p}$ (as OJA) and then an increasing weight modulus (as close to the MC direction for $\lambda_n > 1$) better respects this condition. The decrease in γ has a similar positive effect.

2.7.5 Conclusions

In summary: Dealing with the possible instabilities, there is a clear distinction between two groups of learning laws. The first contains LUO, OJAn, and MCA EXIN and is characterized by the presence of negative instability, which in general happens for values of ϑ_{xw} typical of the transient state. The second contains OJA, OJA+, and FENG and is characterized by stability in the transient state. This analysis justifies the following proposition.

Proposition 73 (Bias/Variance) *LUO, OJAn, and MCA EXIN are iterative algorithms with high variance and low bias. However, OJAn and, above all, LUO require many more iterations to converge than MCA EXIN, because they have more fluctuations around the MC direction and cannot be stopped earlier by a stop criterion that, as seen in Section 2.6.2.6, requires the flatness of the weight time evolution. On the contrary, OJA, OJA+, and FENG are algorithms with low variance and high bias. However, FENG has larger fluctuations and is unreliable for near-singular matrices.*

Obviously, for all the algorithms, the presence of outliers worsens the dynamic stability.

Remark 74 (Fluctuations Comparison) *When the weight modulus is quite a bit smaller than 1, the LUO neuron learning process has the smallest fluctuations (transient) with respect to OJAn and MCA EXIN (the worst transient). Increasing the weight modulus decreases the differences among the neurons, which are null on the unit sphere. For larger moduli, LUO is still the neuron with fewer fluctuations.*

Remark 75 (Fluctuations and Weight Moduli) *Working with weight moduli less than 1 gives fewer fluctuations.*

2.8 NUMERICAL CONSIDERATIONS

2.8.1 Computational Cost

The MCA learning laws are iterative algorithms and have a different computational cost per iteration, which is here evaluated in floating-point operations

Table 2.2 Cost per iteration of the MCA learning laws

	Flops per Iteration
OJA+	$8n + 3$
LUO	$8n + 1$
EXIN	$8n + 1$
OJAn	$8n$
FENG	$8n - 1$
OJA	$6n$

(flops[13]) and shown in Table 2.2. OJA has the lowest cost. All costs depend on the dimensionality n of the data. For high-dimensional data, all learning laws have the same cost except OJA, which has a saving per iteration of 33% in flops.

2.8.2 Quantization Errors

Limited precision (quantization) errors can degradate the solution of gradient-based algorithms with respect to the performance achievable in infinite precision. These errors accumulate in time without bound, leading in the long term (tens of millions of iterations) to an eventual overflow [23]. This type of divergence is here called *numerical divergence*. There are two sources of quantization errors:

1. *The analog-to-digital conversion used to obtain the discrete time-series input*. For a uniform quantization characteristics, the quantization is zero mean.
2. *The finite word length used to store all internal algorithmic quantities*. This error is not of zero mean. This mean is the result of the use of multiplication schemes that either truncate or round products to fit the given fixed word length.

The degradation of the solution is proportional to the conditioning of the input (i.e., to the spread of the eigenvalue spectrum of the input autocorrelation matrix). Hence, this problem is important for near-singular matrices (e.g., in the application of MCA for the computation of the translation in computer vision) [24]. For an example with FENG, see Figure 2.14, which shows the computation of the smallest eigenvalue of a singular matrix whose eigenvalues are 0, 1, and 1.5. Notice the finite time divergence caused here by numerical problems. The following remarks, demonstrated for the OLS algorithm in [23], are also valid for the MCA learning laws.

Remark 76 (Slow Learning Rate) *Decreasing the learning rate in the infinite-precision algorithm leads to improved performance. Nevertheless, this decrease increases the deviation from infinite-precision performance.*

[13]For a definition of flop, see [75, p. 19].

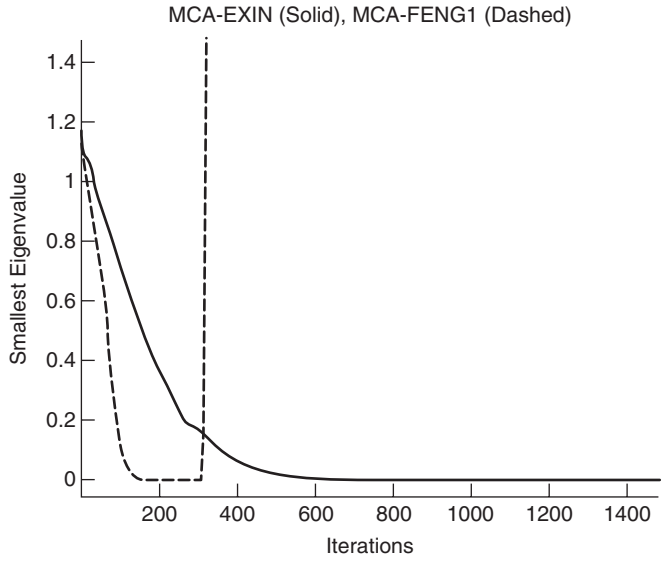


Figure 2.14 Numerical sudden divergence of FENG in a singular autocorrelation matrix.

Increasing the learning rate can also magnify numerical errors, so there is a trade-off in terms of numerical effects on the size of the learning rate [67, digital stopping rule].

Remark 77 (Small Eigenvalues) *Any quantization error that has a nonzero energy content along the first minor component of the input autocorrelation matrix is increased in magnitude by the inverse of this smallest eigenvalue. Furthermore, small eigenvalues further mandate that slow learning rates be used for infinite-precision stability* (*for the OLS algorithm, see* [*82,192*]).

Mutuated from the OLS theory, a technique known as *leakage* is proposed here to prevent the overflow at the expense of some increase in cost of implementation and at the expense of a further small degradation in performance (a small bias). This technique was introduced in [68,183,184,198]. The leakage method changes the MCA error criterion (2.32) as

$$E = \frac{w^T R w}{w^T w} + \delta \|w\|_2^2 \tag{2.180}$$

where δ is a constant scalar. The MCA EXIN, LUO, and OJAn learning laws (2.37) then become

$$\begin{aligned} w(t+1) &= (1 - \alpha\delta)\, w(t) - \alpha S_{\text{neuron}} \\ &= \eta w(t) - \alpha S_{\text{neuron}} \end{aligned} \tag{2.181}$$

where $\eta = 1 - \alpha\delta$ is the leakage factor and is chosen close to 1, $\eta < 1$. δ can also be a diagonal matrix, with the entries along the diagonal chosen to weight more heavily against unconstrained growth in regions where the input is energy deficient [23]. Note that if δ is chosen small, there is only a little performance degradation, but if it is too small, overflow can occur. This technique is *nearly* equivalent to the technique of adding a small (of power δ) uncorrelated noise component to the input [23,192]. The noise is generated by one of several available techniques [180] so as to be a stationary and ergodic process with zero mean and diagonal variance matrix with equal elements. This white noise has the effect of exciting all the ranges of the input. For examples of these techniques, see [24], where applications of MCA to the computation of the focus of expansion (FOE) in passive navigation are given.

2.8.3 Preprocessing and Preconditioning

The training sets have to be preprocessed to translate the centroid of the data to the origin of the coordinate axes (i.e., all inputs must have zero mean). This technique is also justified by the RQ properties (2.2) and (2.3).

The exact adaptive gradient methods require explicit knowledge of the autocorrelation matrix $R = E\left[x(t)\,x^T(t)\right]$ of the input vector $x(t)$. This is the case of the solution of overdetermined systems where all rows of the coefficient matrix compose the TS for the neuron. The instantaneous adaptive gradient algorithms replace the knowledge R with its rank 1 update $x(t)\,x^T(t)$ and are controlled by the learning rate, which also controls the memory length of the system. These instantaneous methods cannot be used to compute the eigenpairs of a fixed deterministic matrix. [14] According to [48], the gradient-based adaptive algorithms are unattractive when not used in their approximated instantaneous form. However, in [24] a constrained MCA EXIN algorithm, called TLS EXIN, is proposed and works very well for finding the TLS solution of overdetermined sets of equations. In these problems, it should also be considered a pre- and postconditioning of the autocorrelation matrix which corresponds to the data of the overdetermined set.

2.8.4 Acceleration Techniques

The MCA learning laws are instantaneous adaptive gradient algorithms and then work as sequential. Different methods, used in optimization theory and in neural theory to improve the learning process, can be applied equally here. Among these, the most interesting are:

1. Adding a *momentum* term [153]
2. The *bold driver* technique [7,185] which checks the error function to set the learning parameters online
3. The optimal learning-rate parameter estimation using the Hessian matrix of the error function [49]

[14] As a consequence, batch techniques don't work well for MCA neurons.

4. The *delta-bar-delta* rule [101], which considers a diagonal matrix instead of the usual scalar learning rate parameter (assuming that the weights evolve independently in time) and sets its diagonal elements as a function of the sign of the local gradient correspondent component
5. The *quickprop* rule [55], which considers the weights as quasi-independent and approximates the error surface by a quadratic polynomial

Other methods can be found in [21]. However, the batch techniques can better accelerate the MC computation. For MCA, if the incoming inputs are collected in *blocks* and are fed to the neuron, which changes its weights only after the entire block presentation, all methods typical of *batch* learning can be used (here block processing and batch processing are considered distinct because of the constancy of the block in the second process): for example, the conjugate gradient methods [59,66,85,86] (for its implementation as an acceleration method for MCA EXIN, see [24]).

Remark 78 (Best Block Technique) *Among the possible numerical techniques to minimize the Rayleigh quotient, it is not practical to use the variable metric* (*VM; also known as quasi-Newton*) *and the Newton descent techniques because the RQ Hessian matrix at the minimum is singular and then the inverse does not exist. On the other hand, the conjugate gradient approach can deal with the singular* H_r [*85, pp. 256–259; 196*] *and can be used for accelerating MCA EXIN in block mode* [*24*].

2.9 TLS HYPERPLANE FITTING

An important application of the MCA is hyperplane fitting [195] by means of the orthogonal regression (TLS). Given a set of N n-dimensional data points $x^{(i)}$, the problem is to find a hyperplane model

$$w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_0 = w^T x + w_0 = 0 \tag{2.182}$$

such that the sum of the squared perpendicular distances of the model from the data is minimized (total least squares criterion error E_{TLS}; see [98]). Hence,

$$E_{\text{TLS}} = \sum_{i=1}^{N} \frac{\left(w^T x^{(i)} + w_0\right)^2}{w^T w} = N \frac{w^T R w + 2 w_0 w^T e + w_0}{w^T w} \tag{2.183}$$

where $e = (1/N) \sum_{i=1}^{N} x^{(i)}$ and $R = (1/N) \sum_{i=1}^{N} x^{(i)} x^{(i)T}$ are the mean vector and the autocorrelation matrix of the data set. From $dE_{\text{TLS}}/dw = 0$ it follows that the critical points of E_{TLS} should satisfy

$$Rw + w_0 e - \lambda w = 0, \qquad \lambda = \frac{w^T R w + 2 w_0 w^T e + w_0}{w^T w} \tag{2.184}$$

Taking the expectation of both sides of eq. (2.182), the following relation holds:

$$w_0 = -w^T e \tag{2.185}$$

which, substituted into eq. (2.184) yields

$$\Sigma w - \lambda w = 0, \qquad \lambda = r\,(w, \Sigma) \tag{2.186}$$

where $\Sigma = R - ee^T$ is the covariance matrix of the data set. The MCA neurons can be used directly as optimal TLS fitting analyzers if $e = 0$. The parameters are given by the unit vector in the direction of the final solution (i.e., a division is needed). In the case of $e \neq 0$, these neurons need a *preprocessing* of the data (see Section 2.8.2.3). Noticing that $\Sigma = R - ee^T = E\left[(x - e)\,(x - e)^T\right]$, it suffices to subtract e from each data point. An extra operation is still needed after the neuron learning; the parameter $w_0 = -w^T e$ must be computed.

Remark 79 (Output of the Neuron) *After the preprocessing, if the weight modulus is equal to 1, after the presentation of the* ith *sample point, the absolute value of the output of the linear MCA neuron represents the orthogonal distance from the sample point to the current fitted hyperplane. Furthermore, the sign of its output also detects which side of the hypersurface the point is on.*

2.10 SIMULATIONS FOR THE MCA EXIN NEURON

The first simulations, presented here to illustrate the application of Section 2.9, deal with the benchmark problem in [195]: A line model $a_1 x + a_2 y = 1$ is fitted to an observation data set by the estimation of its parameters. Here, $a_1 = 0.25$ and $a_2 = 0.5$. First, 500 points are taken from this line with $0 < x < 5$ by uniform sampling. Then the noisy observation set is generated by adding Gaussian noise with zero mean and variance σ^2 to these points. As seen before, at first the points are preprocessed. Then the MCA EXIN, OJA, OJAn, and LUO learnings are compared (at each step the input vectors are taken from the training set with equal probability). All neurons have the same initial conditions $[0.1, 0.1]^T$ and the same learning rate [i.e., start $\alpha(t)$ at 0.01, reduce it linearly to 0.001 at the first 500 iterations, and then keep it constant]. In the end, the estimated solution must be renormalized. The index parameter, defined as

$$\rho = \frac{\sum_{i=1}^{n} \left(w_i(t) - w_i^*\right)^2}{n} \tag{2.187}$$

[$w_i(t)$ are the n components of the neuron weight vector and w_i^* are the n components of the desired parameters], is used as a measure of the accuracy. In the first experiment the noise level is low ($\sigma^2 = 0.1$). Figure 2.15 shows

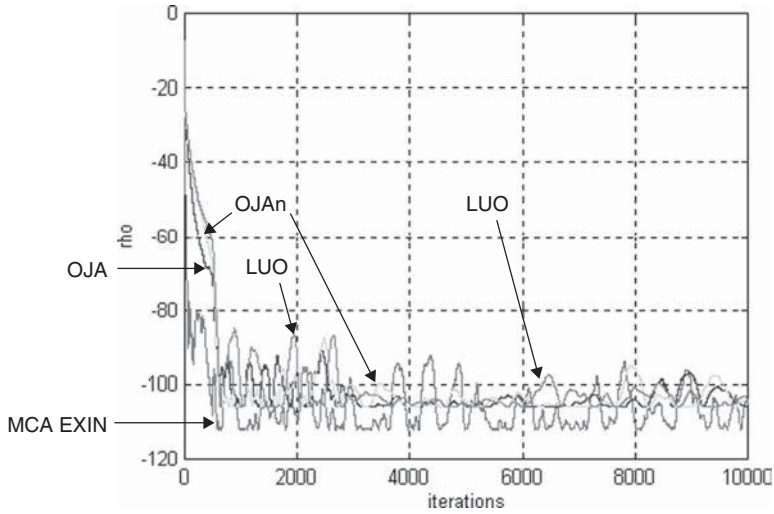


Figure 2.15 Line fitting for noise variance equal to 0.1: plot of the index parameter (expressed in decibels). The values are averaged using a temporal mask with width equal to 500 iterations, except in the first 500 iterations, in which the mask is equal to the maximum number of iterations. (*See insert for color representation of the figure*.)

the plot of the index parameter. The accuracy is very good and, in particular, MCA EXIN has the best ρ for most of iterations (recall that the initial weight norm is low, but this is the best choice, as a consequence of Remark 64). The figure also shows its bigger fluctuations. As anticipated by the theory, LUO has the worst behavior. Figure 2.16 represents the same plot for a higher level of noise ($\sigma^2 = 0.5$). Evidently, the accuracy is lower, but the analysis about the MCA neuron properties is still valid. Figure 2.17 shows the first iterations: MCA EXIN is the fastest and LUO is the slowest algorithm.

The following simulations use, as data, a zero mean Gaussian random vector $x(t)$ generated by an autocorrelation matrix R whose spectrum is chosen in advance. The goal of this approach is the analysis of the behavior of the MCA laws with respect to the dimensionality n of data and the conditioning of R. In the first group of simulations, the components of the initial weight vector are chosen randomly in $[0, 1]$. λ_n is always equal to 1. The other eigenvalues are given by the law $\lambda_i = n - i$; then the condition number $\kappa_2(R) = \lambda_1/\lambda_n$ increases with n, but R always remains a well-conditioned matrix. Table 2.3 shows, for four MCA laws, the best results,[15] in terms of total cost in flops, obtained for each value of n. Except for EXIN, all other laws diverge for low values of n: from OJA, which diverges for only $n = 7$, to LUO, which diverges for $n = 10$. This problem can be explained by the choice of initial conditions: increasing the number of components, the initial weight modulus increases and quickly becomes greater than

[15]For each value of n, several experiments have been done by changing the learning rate (initial and final values, monotonic decreasing law), and only the best result for each MCA law is reported.

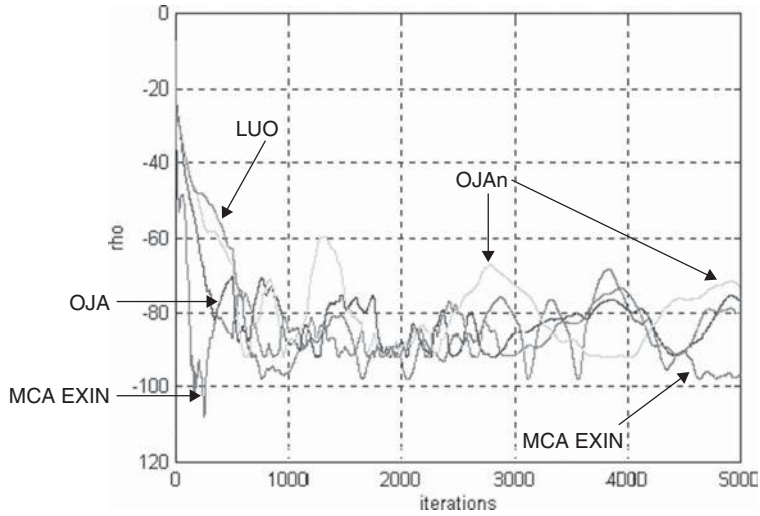


Figure 2.16 Line fitting for noise variance equal to 0.5: plot of the index parameter (expressed in decibels). The values are averaged using a temporal mask with width equal to 500 iterations, except in the first 500 iterations in which the mask is equal to the maximum number of iterations. (*See insert for color representation of the figure*.)

1. About OJA, there is sudden divergence. Indeed, from (2.119), t_∞ decreases for increasing p_0. It has been noticed that for increasing n, the sudden divergence is anticipated. About FENG, there is instability divergence. Indeed, as shown in (2.171), the fluctuations depend only on the modulus and the data inputs. The modulus is large and remains so because of the too large oscillations, which prevent the weights from approaching the MC direction. As a consequence, this generates more and more instability until the finite time divergence. Obviously, for increasing n, the divergence is anticipated. About LUO, two explanations are possible: sudden and instability divergence. However, in these experiments the divergence is of the instability type, because it is accompanied by very large oscillations and certainly anticipates the sudden divergence. Indeed, the weight modulus increases very much [see (2.100)], so it decreases the dynamic stability interval Δ_{st} accordingly [see (2.159)], generating large oscillations which can generate the divergence. However, the fact of having a dynamic instability subspace which is the contrary of the OJAs (see Figures 2.12 and 2.13) and the fact that the weight modulus increases depends only on α^2 (see Section 2.6.1) certainly explains the fact that LUO diverges for higher n than OJA. Obviously, for increasing n, the LUO divergence is anticipated. Figures 2.20 to 2.21 show some of these experiments for, respectively, $n = 5$, 9, 11, and 100. Figure 2.18 shows the large fluctuations of FENG and that MCA EXIN has the fastest transient. Figure 2.19 shows the very good accuracy of MCA EXIN, the sudden divergence of OJA, and the unacceptable accuracy of FENG. Figure 2.20 confirms that FENG undergoes the instability divergence (notice the peaks before the

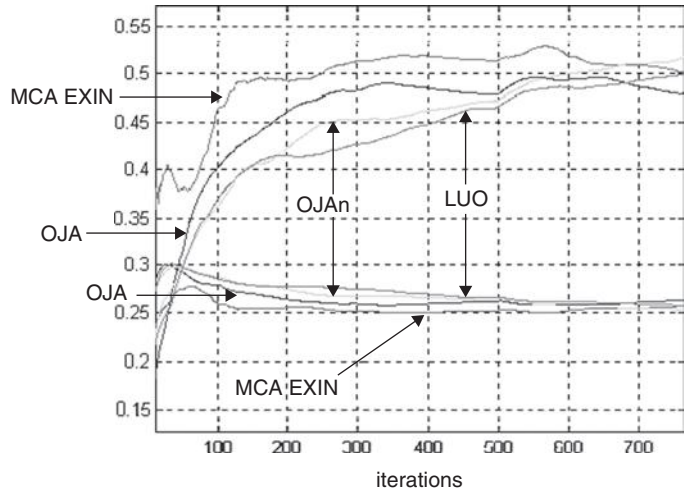


Figure 2.17 Line fitting for noise variance equal to 0.5: zoom of the plot of the weights for the transient analysis. (*See insert for color representation of the figure*.)

Table 2.3 Total cost of the MCA learning laws for autocorrelation matrices of increasing size*

dim	κ_2	EXIN	LUO	FENG	OJA
3	20	42,726	37,466	41,133	20,888
5	40	65,966	59,131	60,345	42,942
7	60	183,033	640,074	742,346	div.
8	70	205,968	706,075	830,000	div.
9	80	228,681	965,544	*	div.
10	90	252,002	1,061,098	div.	div.
15	140	366,255	div.	div.	div.
25	240	2,003,755	div.	div.	div.
50	490	3,725,868	div.	div.	div.
75	740	4,488,496	div.	div.	div.
100	990	6,258,001	div.	div.	div.

*Inaccurate result; div., divergence.

divergence). Figure 2.21 shows the good results obtained by MCA EXIN (here $\lambda_n = 0.01$).

To avoid starting with too high initial conditions, a second group of experiments has been done where the initial weight vector is the same as in the first experiment, but divided by 10 times its norm. Table 2.4 shows the results about divergence of all MCA laws except MCA EXIN, which always converges. Obviously, the divergence happens later than in the first group of experiments because the weights are lower. Figures 2.22 and 2.23 show some results. A possible interpretation of the divergence in these experiments is the following:

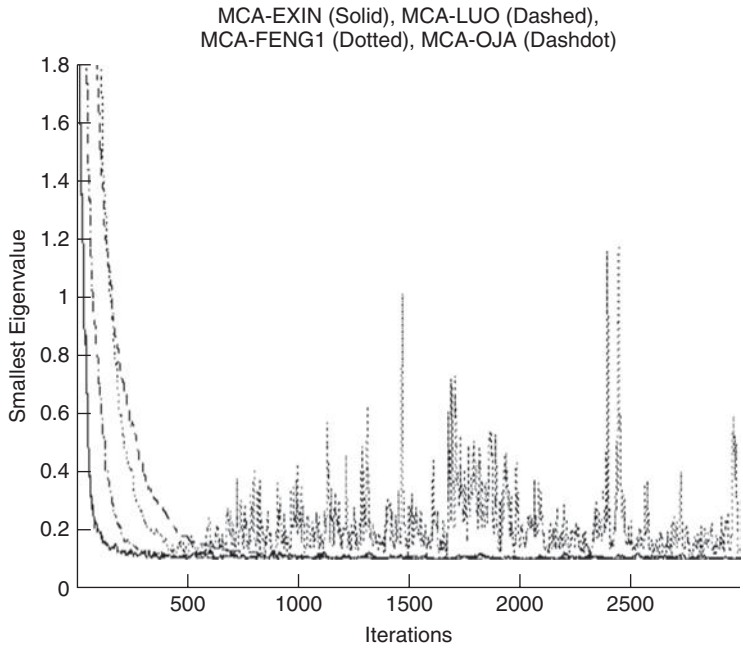


Figure 2.18 Computation of the smallest eigenvalue of a 5×5 autocorrelation matrix for the first choice of initial conditions.

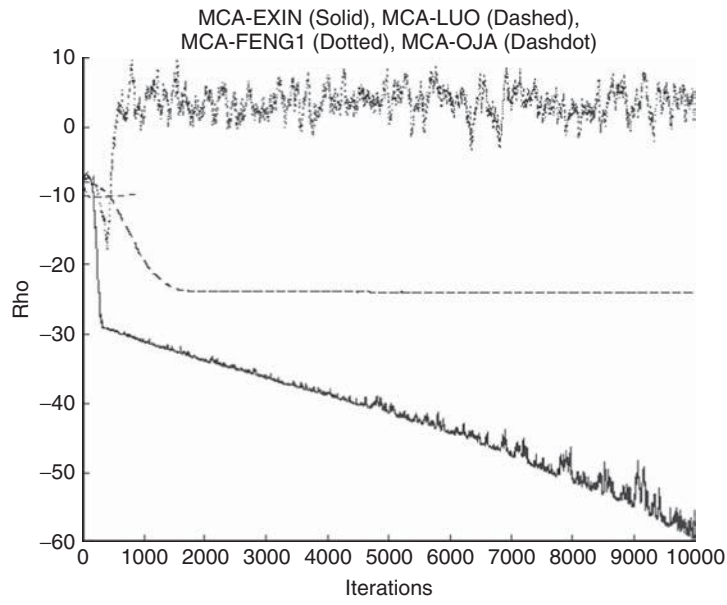


Figure 2.19 Computation of the accuracy parameter ρ of a 9×9 autocorrelation matrix for the first choice of initial conditions.

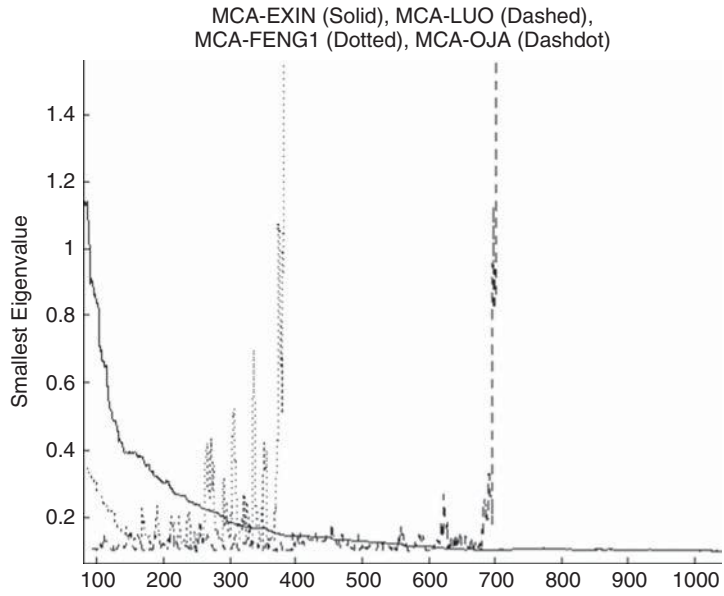


Figure 2.20 Computation of the smallest eigenvalue of a 11×11 autocorrelation matrix for the first choice of initial conditions.

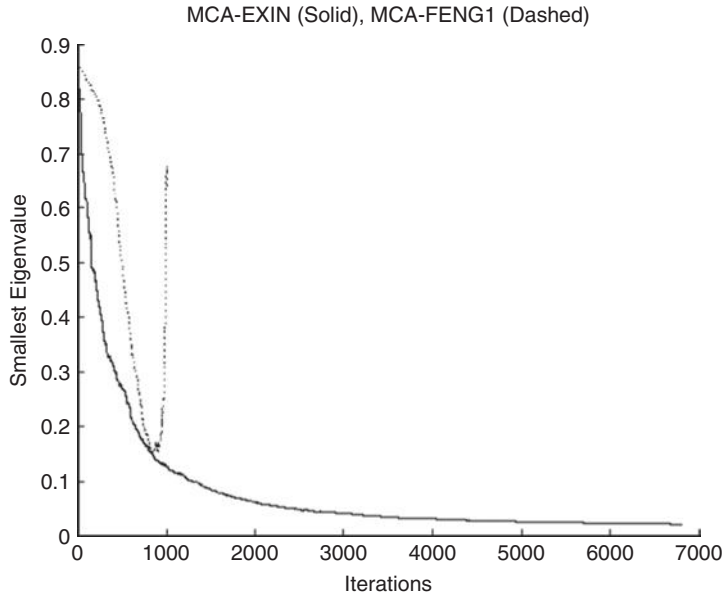


Figure 2.21 Computation of the smallest eigenvalue of a 100×100 autocorrelation matrix for the first choice of initial conditions.

Table 2.4 Divergence of the MCA learning laws for autocorrelation matrices of increasing size[a]

dim	LUO	FENG	OJA	OJAn	OJA+
3	conv.	conv.	conv.	conv.	conv.
7	conv.	conv.	conv.	conv.	conv.
9	conv.	413	conv.	conv.	conv.
10	conv.	460	conv.	conv.	conv.
15	conv.	531	conv.	conv.	conv.
18	conv.	371	conv.	conv.	conv.
20	conv.	700	180	conv.	65
25	conv.	27	27	conv.	26
30	conv.	27	27	1848	17
40	conv.	400	17	975	12
50	conv.	370	6	670	12
60	conv.	540	3	550	14
70	conv.	260	7	520	12
80	545	220	5	400	6
90	8	7	8	355	8
100	8	8	8	250	5

[a]The numbers show at which iteration the weights go to infinity; conv., convergence.

- *LUO*. The weight modulus increases as before, but starts from lower values. Hence, there is always instability divergence, but for higher n. The experiments have shown that it is very sensitive to the learning rate, as expected from (2.159).
- *OJA*. With this choice of initial conditions, the weight modulus quickly[16] decreases to 0 (see Section 2.6.2.2) and then the stability subspace is smaller (see Figure 2.13), which implies an increase in the oscillations and the instability divergence. For increasing n, the divergence is anticipated. Probably it is due to the fact that it is more difficult to fit data with a linear hyperplane because of the problem of empty space [10] caused by the increasing dimensionality of data and accentuated by the fact that this law is only an approximate RQ gradient, which implies that the OJA algorithm does not reach the MC direction and then diverges very soon.
- *OJAn*. The same analysis for LUO is valid here. However, for increasing n, since the first iterations, there are large fluctuations. This could be expected from eq. (2.159), which, unlike LUO, does not depend on p, so the algorithm cannot exploit the low initial conditions. This instability is worsened by the problem of the empty space; indeed, the algorithm does not reach the MC direction, so the oscillations increase because on average, ϑ_{xw} does not approach $\pi/2$ and Δ_{st} remains small enough to cause the instability divergence for values of n smaller than those in the case of LUO.

[16]Proportional to α (see Section 2.6.2.1).

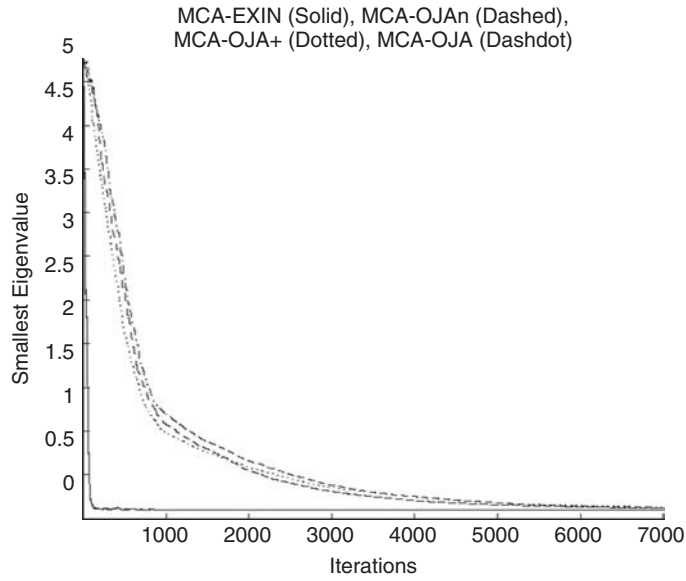


Figure 2.22 Computation of the smallest eigenvalue of a 7×7 autocorrelation matrix for the second choice of initial conditions.

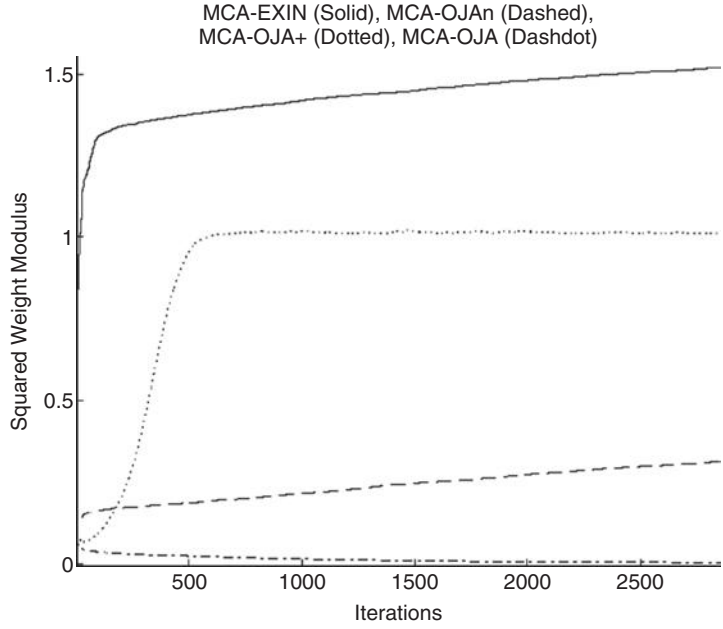


Figure 2.23 Plot of the squared weight moduli in the case of a 10×10 autocorrelation matrix for the second choice of initial conditions.

- *OJA+*. It behaves exactly as OJA as a consequence of the fact that because of the problem of the empty space for increasing n (it is not an exact RQ gradient), it does not converge to 1.
- *FENG*. There is always instability, but for higher n because of the advantage of starting with lower initial conditions. It is possible that for high n, there is the phenomenon of numerical divergence.

Other simulations can be found in [24].

2.11 CONCLUSIONS

Minor components analysis is becoming more and more important, not only in signal processing, but above all, in data analysis (orthogonal regression, TLS). Important applications in computer vision (estimation of the parameters of the essential matrix in structure from motion problems) may be found in [24] and are very promising.

This chapter is not only the presentation of a novel neural MCA algorithm that overcomes the drawbacks of the other neural MCA laws, but is above all a complete theory of the neural MCA. Indeed, the analysis is done from the stochastic, the asymptotic, the differential geometry, and the numerical point of view. Three types of divergence are presented and studied. An equivalence analysis is presented. The relevance of the RQ degeneracy property is shown, and very important consequences are deduced from it. For the first time all the algorithms are tested on problems of higher dimensionality. A real application is presented briefly.

From this chapter, the following conclusions can be deduced from each of the already existing MCA learning laws, by considering the choice of low initial conditions (justified by Remarks 56 and 64):

- *LUO*. It has a slow convergence to the MC direction and then diverges [order $O\left(\alpha^2\right)$]. It also suffers from the sudden divergence, which can be anticipated by noisy data. It cannot be stopped reliably and is very sensitive to outliers. It is a high variance/low bias algorithm, but it oscillates too much around the solution. It works badly for medium-dimensional data.
- *OJAn*. It has a slow convergence (faster than LUO) to the MC direction and then diverges [order $O\left(\alpha^2\right)$, but slower than LUO]. The rate of divergence depends exponentially on the level of noise in the data (λ). It cannot be stopped reliably and is very sensitive to outliers. It is a high variance/low bias algorithm, but it oscillates too much around the solution. It works badly for medium-dimensional data.
- *OJA*. It has a very slow convergence to the MC direction and its squared weight modulus change is of order $O(\alpha)$. The weights decrease to 0 (in case of initial conditions of modulus greater than 1, there is sudden divergence,

which is anticipated in case of noisy data). It cannot be stopped reliably and is very sensitive to outliers. It is a low variance/high bias algorithm and the computational cost per iteration is the best. It works very badly for medium-dimensional data.

- *OJA+*. It has a very slow convergence to the MC direction and its squared weight modulus change is of order $O(\alpha)$. The weights converge only for $\lambda_n < 1$ (in case of initial conditions of modulus greater than 1 and $\lambda_m > 1$, there is sudden divergence, which is anticipated in case of noisy data). It cannot be stopped reliably and is very sensitive to outliers. It is a low variance/high bias algorithm. It works very badly for medium-dimensional data.
- *FENG*. It is the worst learning law because of the too large oscillations (the dynamic stability does not improve when the MC direction is reached because it depends only on p and the learning rate). It has a slow convergence (same as LUO) to the MC direction and is very sensitive to the outliers. It cannot be stopped reliably and can diverge for near-singular matrices. It works very badly for medium-dimensional data.

MCA EXIN is by far the best MCA learning law. It has the best convergence to the MC direction, the slower divergence, and does not have problems of either sudden or instability divergence. It works very well in high-dimensional spaces and has been used in real applications. It is robust to outliers because of the presence of the squared modulus of the weight in the denominator of its weight increment. However, there also exists a variant (NMCA EXIN [28]), which is robust to outliers because it implements the theory of the M-estimators [91], and it is presented in Chapter 3. MCA EXIN can be stopped reliably. It is a high variance/low bias algorithm with very good dynamics because of its inertia for large weights. It has the same computational cost as the other MCA laws, except OJA.

The best possible choice for the initial conditions should be as low as possible. However, there is a limit for MCA EXIN: too large oscillations in the transient (not divergence!). In this case, a variant, called MCA EXIN+, has been presented in [24], which allows null initial conditions and works better than MCA EXIN. It is described in Chapter 5.

3

VARIANTS OF THE MCA EXIN NEURON*

3.1 HIGH-ORDER MCA NEURONS

High-order units have been advocated in [65]. The MCA neurons can be generalized into high-order neurons for fitting nonlinear hypersurfaces, which can be expressed as

$$w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x) + w_0 = 0 \tag{3.1}$$

where $f_i(x)$ is a function of x [e.g., for polynomial hypersurfaces $f_i(x) = x_1^{r_1} x_2^{r_2} \cdots x_n^{r_n}$ with integers $r_j \geq 0$]. A generalized high-order neuron is just like a linear neuron, but each weight is now regarded as a high-order connection, and each input is not the component of the data vector x but its high-order combination $f_i(x)$. From this point of view, the neuron can be considered as a *functional link* neuron [148]. Hence, the same reasoning of the hyperplane fitting can be made: Defining $f = \left[f_1(x), f_2(x), \ldots, f_n(x)\right]^T$, calculate $e_f = E(f)$; if it is different from zero, preprocess the data, and so on.

Remark 80 (MCA Statistical Limit) *If the observation datum x has Gaussian noise, the noise in the transformed input f is generally non-Gaussian* [*145*]. *It*

*Sections 3.1 and 3.2 are reprinted from G. Cirrincione and M. Cirrincione, "Robust Total Least Squares by the Nonlinear MCA EXIN Neuron," *IEEE International Joint Symposia on Intelligence and Systems (SIS98): IAR98, Intelligence in Automation & Robotics*, pp. 295–300, Rockville, MD, May 21–23, 1998. Copyright © 1998 IEEE. Reprinted with permission.

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

implies that the MCA solution (TLS) is not optimal (see [98]); a robust version of MCA EXIN (NMCA EXIN) that overcomes the problem is presented in the next section.

3.2 ROBUST MCA EXIN NONLINEAR NEURON (NMCA EXIN)

As shown in Section 1.10, the TLS criterion (which gives a solution parallel to the minor eigenvector) is not optimal if the data errors are impulsive noise or colored noise with unknown covariance or if some strong outliers exist. Instead of using the TLS criterion, Oja and Wang [143–145], suggest an alternative criterion that minimizes the sum of certain *loss functions* of the orthogonal distance:

$$J_f(w) = \sum_{i=1}^{N} f\left(\frac{w^T x^{(i)}}{\sqrt{w^T w}}\right) = \sum_{i=1}^{N} f(d_i) \qquad w_0 = 0 \tag{3.2}$$

This is the TLS version of the robust LS regression. In robust LS the vertical distances are taken instead of the perpendicular signed distances d_i. Solutions for the weights are among the M-estimates, which are the most popular of the robust regression varieties [91].

Definition 81 (Robust MCA) *The weight vector that minimizes the criterion $J_f(w)$ is called a* robust minor component.

The loss function f must be nonnegative, convex with its minimum at zero, increasing less rapidly than the square function, and even [91]. Convexity prevents multiple minimum points, and the slow increase dampens the effect of strong outliers on the solution. The derivative of the loss function, $g(y) = df(y)/dy$ is called the *influence function* [80]. Depending on the choice of the loss function, several different learning laws are obtained. The most important loss functions are [21]:

- *Logistic function:*

$$f(y) = \frac{1}{\beta} \ln \cosh \beta y \tag{3.3}$$

with corresponding influence function

$$g(y) = \tanh \beta y \tag{3.4}$$

- *Absolute value function:*

$$f(y) = |y| \tag{3.5}$$

with corresponding influence function

$$g(y) = \operatorname{sign} y \tag{3.6}$$

- *Huber's function:*

$$f(y) = \begin{cases} \dfrac{y^2}{2} & \text{for} \quad |y| \leq \beta \\ \beta\,|y| - \frac{\beta^2}{2} & \text{for} \quad |y| > \beta \end{cases} \tag{3.7}$$

- *Talvar's function:*

$$f(y) = \begin{cases} \dfrac{y^2}{2} & \text{for} \quad |y| \leq \beta \\ \dfrac{\beta^2}{2} & \text{for} \quad |y| > \beta \end{cases} \tag{3.8}$$

- *Hampel's Function:*

$$f(y) = \begin{cases} \dfrac{\beta^2}{\pi}\left(1 - \cos\dfrac{\pi y}{\beta}\right) & \text{for} \quad |y| \leq \beta \\ 2\dfrac{\beta^2}{\pi} & \text{for} \quad |y| > \beta \end{cases} \tag{3.9}$$

where β is a problem-dependent control parameter.

Replacing the finite sum in eq. (3.2) by the theoretical expectation of f gives the following cost function:

$$J_{\text{robust EXIN}}(w) = E\left\{f\left(\frac{w^T x}{\sqrt{w^T w}}\right)\right\} \tag{3.10}$$

Then

$$\frac{d\,J_{\text{robust EXIN}}(w)}{d\,w} = E\left\{g\left(\frac{w^T x}{\sqrt{w^T w}}\right)\frac{\left(w^T w\right)x - \left(w^T x\right)w}{\left(w^T w\right)^{\frac{3}{2}}}\right\}$$

Considered that $g(y)$ is odd and assuming that $\left|w^T x\right| < \varepsilon \;\; \forall \varepsilon > 0$, the following approximation is valid:

$$g\left(\frac{w^T x}{\sqrt{w^T w}}\right) \sim \frac{g\left(w^T x\right)}{\sqrt{w^T w}} \tag{3.11}$$

which, in a stochastic approximation-type gradient algorithm in which the expectations are replaced by the instantaneous values of the variables, gives the *robust MCA EXIN learning law* (NMCA EXIN):

$$w(t+1) = w(t) - \frac{\alpha(t)}{w^T(t)\,w(t)}\left[g(y(t))\,x(t) - \frac{g(y(t))\,y(t)\,w(t)}{w^T(t)\,w(t)}\right] \tag{3.12}$$

If the absolute value function is chosen, then

$$w(t+1) = w(t) - \frac{\alpha(t)}{w^T(t)\,w(t)}\left[(\operatorname{sign} y(t))\,x(t) - \frac{|y(t)|\,w(t)}{w^T(t)\,w(t)}\right] \tag{3.13}$$

and the MCA EXIN neuron is nonlinear because its activation function is the hard-limiter [eq. (3.6)]. If the logistic function is chosen, then

$$w(t+1) = w(t) - \frac{\alpha(t)}{w^T(t)\,w(t)}\left[x(t)\tanh\beta y(t) - \frac{y(t)\,w(t)\tanh\beta y(t)}{w^T(t)\,w(t)}\right] \tag{3.14}$$

and the MCA EXIN neuron is nonlinear because its activation function is the sigmoid [eq. (3.4)]. In the simulations, only the sigmoid is taken in account and its steepness β is always chosen equal to 1.

Remark 82 (NMCA EXIN vs. MCA EXIN) *The MCA EXIN learning law is a particular case of the NMCA EXIN learning law for* $f(d_i) = d_i^2$. *As a consequence of the shape of the loss function, for small orthogonal distances* (*inputs not too far from the fitting hyperplane*), *the behaviors of the nonlinear and linear neuron are practically the same, but for larger orthogonal distances, the nonlinear neuron better rejects the noise. The nonlinear unit inherits exactly all the MCA EXIN properties and, in particular, the absence of the finite time* (*sudden*) *divergence phenomenon.*

Remark 83 (Robusting the MCA Linear Neurons) *The same considerations that give the NMCA EXIN learning law can be applied to OJA, OJAn, and LUO. It can be shown that these robust versions inherit the properties of the corresponding linear units. In particular, they still diverge because this phenomenon is not linked to the activation function chosen, but to the structure of the learning law, which implies orthogonal updates with respect to the current weight state. Furthermore, the presence of outliers anticipates the sudden divergence because the learning law is directly proportional to the squared weight modulus, which is amplified by the outlier; it increases the difficulty in choosing a feasible stop criterion.*

For details on this subject, see [28].

3.2.1 Simulations for the NMCA EXIN Neuron

Here a comparison is made between the NMCA EXIN neuron and the nonlinear neuron of [143–145] (NOJA+), which is the nonlinear counterpart of OJA+. Under the same very strict assumptions, NOJA+ still converges. The first simulation uses, as a benchmark, the example in Sections 2.6.2.4 and 2.10,

where $\lambda_{\min} = 0.1235 < 1$. For simplicity, $\alpha(t) = \text{const} = 0.01$. Every 100 steps, an outlier is presented. It is generated with the autocorrelation matrix

$$R' = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} \tag{3.15}$$

The norm of the difference between the computed weight and the right MC unit eigenvector is called ε_w and represents the accuracy of the neuron. After 10 trials under the same conditions (initial conditions at $[0.1, 0.1, 0.1]^T$, $\alpha(t) = \text{const} = 0.01$) and for $\sigma^2 = 100$, NMCA EXIN yields, in the average, after 10,000 iterations, $\lambda = 0.1265$ and $\varepsilon_w = 0.1070$ (the best result yields $\lambda = 0.1246$ and $\varepsilon_w = 0.0760$). A plot of the weights is given in Figure 3.1. Observe the peaks caused by the outliers and the following damping. Figure 3.2 shows the corresponding behavior of the squared weight norm, which is similar to the MCA EXIN dynamic behavior. In the same simulation for $\sigma^2 = 100$, NOJA+ yields, on the average, after $10,000$ iterations, $\lambda = 0.1420$ and $\varepsilon_w = 0.2168$. This poorer accuracy is caused by its learning law, which is an approximation of the true Rayleigh quotient gradient. In the presence of strong outliers, this simplification is less valid. In fact, for lower σ^2, the two neurons have nearly the same accuracy. Figure 3.3 shows the sudden divergence phenomenon for the robust version of LUO. Here the experiment conditions are the same, but the initial conditions are the exact solution: It shows that the solution is not stable even in the robust version.

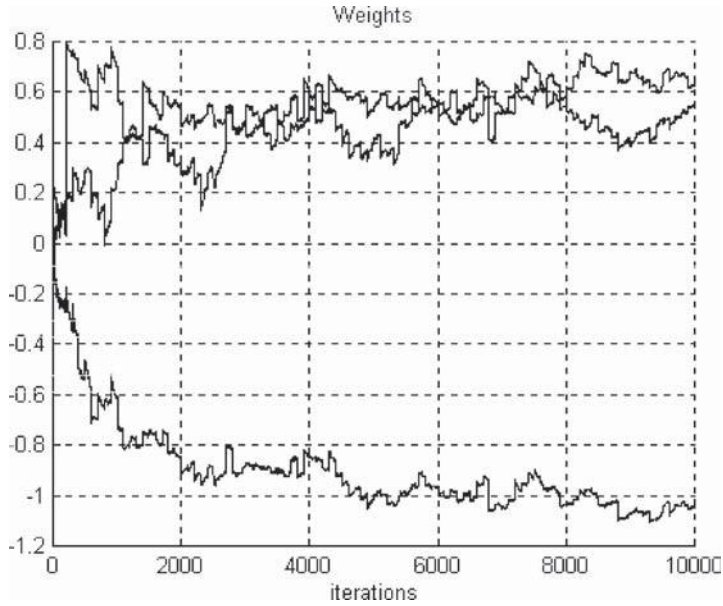


Figure 3.1 NMCA EXIN weights in the presence of outliers.

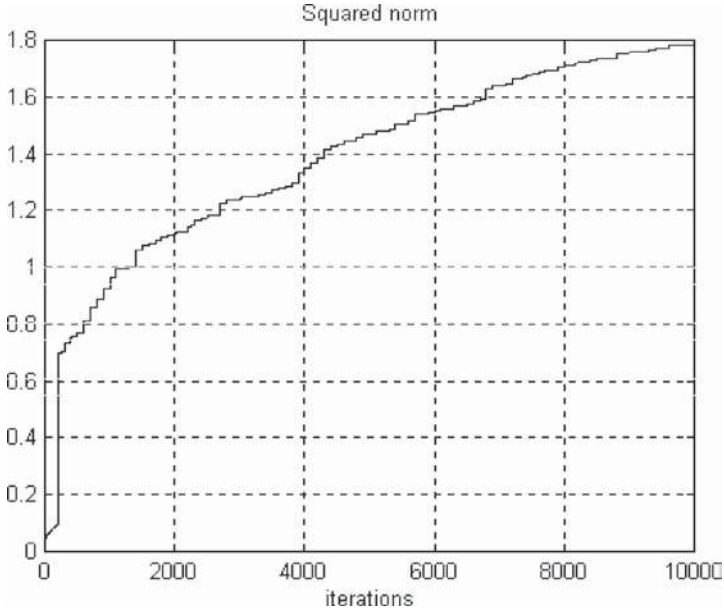


Figure 3.2 Divergence of the NMCA EXIN squared weight norm.

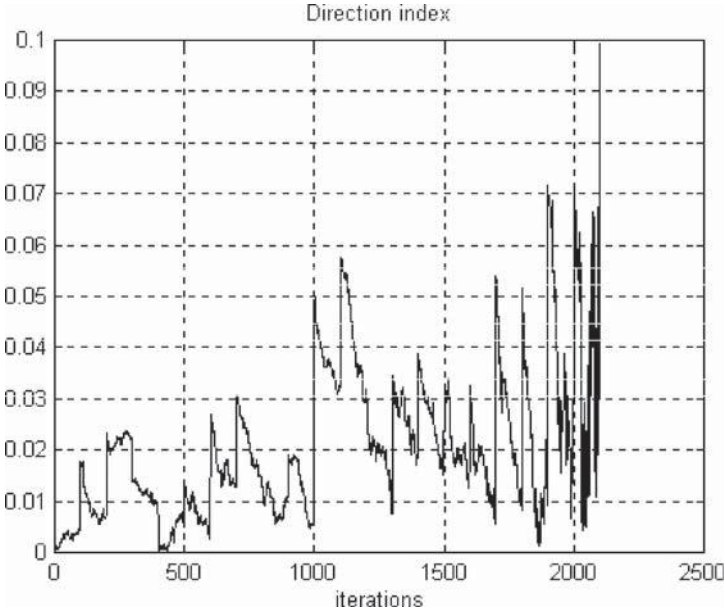


Figure 3.3 Deviation from the MC direction for NLUO, measured as the squared sine of the angle between the weight vector and the MC direction.

The second simulation deals with the benchmark problem in [195], already solved in Section 2.10. As seen before, at first the points are preprocessed. Then the NMCA EXIN and NOJA+ learnings are compared (at each step the input vectors are picked up from the training set with equal probability). Both neurons have the same initial conditions $[0.05, 0.05]^T$ and the same learning rate used in [195]: that is, start $\alpha(t)$ at 0.01, linearly reduce it to 0.0025 at the first 500 iterations, and then keep it constant. In the end, the estimated solution must be renormalized. Table 3.1 shows the results (averaged in the interval between iteration 4500 and iteration 5000) obtained for $\sigma^2 = 0.5$ and without outliers. They are compared with the singular value decomposition (SVD)–based solution and the LS solution. It can be concluded that the robust neural solutions have the same accuracy as that of the corresponding linear solutions. Second, another training set is created by using the same 500 points, but without additional noise. Then two strong outliers at the points $x = 22$, $y = 18$ and $x = 11.8$, $y = 9.8$ are added and fed to the neuron every 100 iterations. Table 3.2 shows the results: NOJA+ is slightly more accurate than NMCA EXIN. This example is of no practical interest because the assumption of noiseless points is too strong. The last experiment uses the same training set, but with different values of additional Gaussian noise and the same two strong outliers. The learning rate and the initial conditions are always the same. The results in Table 3.3 are averaged over 10 experiments and in the interval between iteration 4500 and iteration 5000. As the noise in the data increases, the NMCA EXIN neuron gives more and more accurate results with respect to the NOJA+ neuron. As in the first simulation,

Table 3.1 Line fitting with Gaussian noise

	a_1	a_2	ε_w
True values	0.25	0.5	
LS	0.233	0.604	0.105
SVD	0.246	0.511	0.011
OJA+	0.246	0.510	0.012
NOJA+	0.248	0.506	0.006
NMCA EXIN	0.249	0.498	$\backsim 0$

Table 3.2 Line fitting with strong outliers only

	a_1	a_2	ε_w
True values	0.25	0.5	
LS	0.216	0.583	0.089
SVD	0.232	0.532	0.036
OJA+	0.232	0.532	0.036
NOJA+	0.244	0.492	0.010
NMCA EXIN	0.241	0.490	0.013

Table 3.3 Line fitting with Gaussian noise and strong outliers

σ^2	NMCA EXIN ε_w	NOJA+ ε_w
0.1	0.0229	0.0253
0.2	0.0328	0.0393
0.3	0.0347	0.0465
0.4	0.0379	0.0500
0.5	0.0590	0.0880

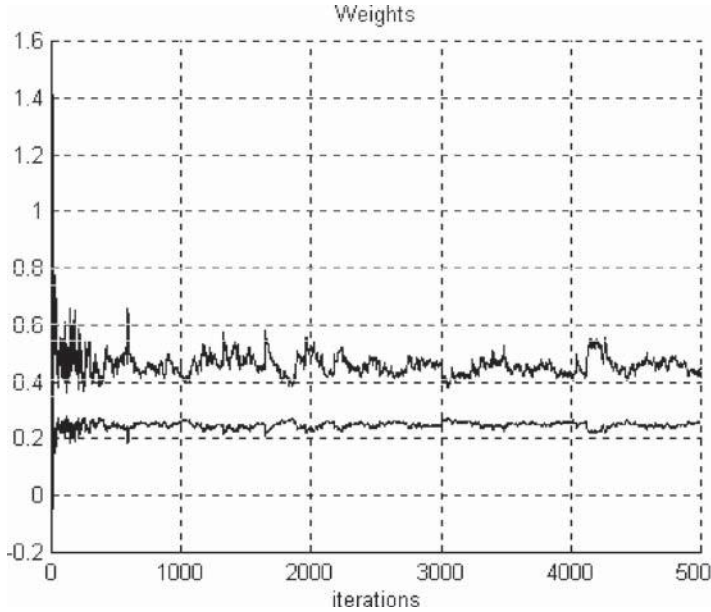


Figure 3.4 Plot of NMCA EXIN weights in a very noisy environment with strong outliers.

this fact is explained by the type of approximation of NOJA+. Figure 3.4 shows the plot of the NMCA EXIN weights for $\sigma^2 = 0.5$ for one of the 10 experiments with this level of noise.

3.3 EXTENSIONS OF THE NEURAL MCA

In this section the neural analysis for MCA is extended to minor subspace analysis (MSA), principal components analysis (PCA), and principal subspace analysis (PSA).

Let R be the $N \times N$ autocorrelation matrix input data, with eigenvalues $0 \leq \lambda_N \leq \lambda_{N-1} \leq \cdots \leq \lambda_1$ and corresponding orthonormal eigenvectors $z_N, z_{N-1}, \ldots, z_1$. The following definitions are given:

- z_1 is the first principal component.
- z_N is the first minor component.
- $z_1, z_2, \ldots, z_M$ $(M < N)$ are M principal components.
- span $\{z_1, z_2, \ldots, z_M\}$ is an M-dimensional principal subspace.
- $z_{N-L+1}, z_{N-L+2}, \ldots, z_N$ $(L < N)$ are L minor components.
- span $\{z_{N-L+1}, z_{N-L+2}, \ldots, z_N\}$ is an L -dimensional minor subspace.

The general neural architecture is given by a single layer of Q linear neurons with inputs $x(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^T$ and outputs $y(t) = \left[y_1(t), y_2(t), \ldots, y_Q(t)\right]^T$:

$$y_j(t) = \sum_{i=1}^{N} w_{ij}(t)\, x_i(t) = w_j^T(t)\, x(t) \tag{3.16}$$

for $j = 1, 2, \ldots, Q$ or $j = N, N-1, \ldots, N-Q+1$, where $wj(t) = \left[w_{1j}(t), \ldots, w_{Nj}(t)\right]^T$ is the connection weights vector. Hence, $Q = 1$ for PCA and MCA (in this case, $j = N$ only).

3.3.1 Minor Subspace Analysis

The minor subspace (dimension M) usually represents the statistics of the additive noise and hence is referred to as the *noise subspace* in many fields of signal processing, such as those cited in Section 2.2.1. Note that in this section, *MSA* represents both the subspace and the minor unit eigenvectors.

The existing MSA networks are:

- *Oja's MCA algorithm*. Oja [141] extended the OJA rule (2.16) as follows:

$$\begin{aligned} w_j(t+1) = w_j(t) &\overbrace{-\alpha(t)\, x(t)\, y_j(t)}^{A} \\ &\overbrace{+\alpha(t)\left[y_j^2(t) + 1 - w_j^T(t)\, w_j(t)\right] w_j(t)}^{B} \\ &\overbrace{-\vartheta\alpha(t) \sum_{i>j} y_i(t)\, y_j(t)\, w_i(t)}^{C} \end{aligned} \tag{3.17}$$

for $j = N, N-1, \ldots, N-M+1$. The corresponding averaging ODE is given by

$$\begin{aligned} \frac{dw_j(t)}{dt} = &-Rw_j(t) + \left[w_j^T(t)\, Rw_j(t)\right] w_j(t) + w_j(t) \\ &- w_j^T(t)\, w_j(t)\, w_j(t) - \vartheta \sum_{i>j} w_i^T(t)\, Rw_j(t)\, w_i(t) \end{aligned} \tag{3.18}$$

The term A in eq. (3.17) is the anti-Hebbian term. The term B is the constraining term (forgetting term). The term C performs the Gram–Schmidt orthonormalization. Oja [141] demonstrated that under the assumptions $w_j^T(0)\, z_j \neq 0$, $\|w_j(0)\|_2 = 1$, $\lambda_{N-M+1} < 1$, and $\vartheta > \lambda_{N-M+1}/\lambda_N - 1$, the weight vector in eq. (3.18) converges to the minor components $z_{N-M+1}, z_{N-M+2}, \ldots, z_N$. If ϑ is larger than but close to $\lambda_{N-M+1}/\lambda_N - 1$, the convergence is slow.

- *LUO MSA algorithms*. In [121], two learning laws are proposed. The first is

$$w_j(t+1) = w_j(t) - \alpha(t)\, y_j(t)\, w_j^T(t)\, w_j(t)\, x(t) + \alpha(t)\, y_j^2(t)\, w_j(t) + \beta\alpha(t)\, y_j(t)\, w_j^T(t)\, w_j(t) \sum_{i>j} y_i(t)\, w_i(t) \tag{3.19}$$

for $j = N, N-1, \ldots, N-M+1$. The corresponding averaging ODE is given by

$$\frac{dw_j(t)}{dt} = -w_j^T(t)\, w_j(t) \left[I - \beta \sum_{i>j} w_i(t)\, w_i^T(t) \right] Rw_j(t) + w_j^T(t)\, Rw_j(t)\, w_j(t) \tag{3.20}$$

where β is a positive constant affecting the convergence speed. Equation (3.20) does not possess the invariant norm property (2.27) for $j < N$. The second (MSA LUO) is the generalization of the LUO learning law (2.25):

$$w_j(t+1) = w_j(t) - \alpha(t)\, y_j(t) \left[w_j^T(t)\, w_j(t)\, x_j(t) - w_j^T(t)\, x_j(t)\, w_j(t) \right] \tag{3.21}$$

for $j = N, N-1, \ldots, N-M+1$ and

$$x_j(t) = x(t) - \sum_{i>j} y_i(t)\, w_i(t) \tag{3.22}$$

for $j = N-1, \ldots, N-M+1$. Equation (3.22) represents, at convergence, the projection of $x(t)$ on the subspace orthogonal to the eigenvectors $z_{j+1}, z_{j+2}, \ldots, z_N$. In this way, an adaptive Gram–Schmidt orthonormalization is done. Indeed, note that if

$$w^T(0)\, z_N = w^T(0)\, z_N = \cdots = w^T(0)\, z_i = 0 \tag{3.23}$$

in the assumptions for Theorem 49, the LUO weight vector converges to z_{i-1} (this is clear in the proof of Theorem 60). This idea has been proposed

in [141,186,194]. The corresponding averaging ODE is given by

$$\frac{dw_j(t)}{dt} = -w_j^T(t)\, w_j(t) \left[I - \sum_{i>j} w_i(t)\, w_i^T(t) \right] R w_j(t) + w_j^T(t) \left[I - \sum_{i>j} w_i(t)\, w_i^T(t) \right] R w_j(t)\, w_j(t) \quad (3.24)$$

Defining

$$R_j = R - \sum_{i>j} w_i(t)\, w_i^T(t)\, R \quad (3.25a)$$

$$\Psi_j(t) = w_j^T(t)\, w_j(t) \quad (3.25b)$$

$$\phi_j(t) = w_j^T(t)\, R_j\, w_j(t) \quad (3.25c)$$

eq. (3.23) becomes

$$\frac{dw_j(t)}{dt} = -\Psi_j(t)\, R_j w_j(t) + \phi_j(t)\, w_j(t) \quad (3.26)$$

Equation (3.26) is characterized by the invariant norm property (2.27) for $j < N$ (evidently, in a first approximation).

It is instructive to summarize the convergence theorems for MSA LUO. In [122], the following theorem appears.

Theorem 84 (MSA LUO Convergence 1) *If the initial values of the weight vector satisfy* $w_j^T(0) z_j \neq 0$ *for* $j = N, N-1, \ldots, N-M+1$, *then*

$$\text{span}\{w_{N-M+1}(f), wz_{N-M+2}(f), \ldots, w_N(f)\} = \text{span}\{z_{N-M+1}, z_{N-M+2}, \ldots, z_N\} \quad (3.27)$$

where $w_j(f) = \lim_{t\to\infty} w_j(t)$. *In other words, MSA LUO finds a basis for the minor subspace, but not the eigenvector basis.*

The proof of this theorem exploits the analogy of the MSA LUO learning law with the PSA LUO learning law cited in Section 3.3.3 and uses thec orresponding proof for convergence. Considering that only the part of demonstration concerning the orthogonality of the converged weight vectors to the remaining nonminor eigenvectors can be applied for MSA, the authors cannot also demonstrate the convergence to the minor components. Suddenly, in [121], the following theorem appears.

Theorem 85 (MSA LUO Convergence 2) *If the initial values of the weight vector satisfy* $w_j^T(0)z_j \neq 0$ *for* $j = N, N-1, \ldots, N-M+1$, *then the weight vectors provided by eq. (3.21) converge to the corresponding minor components; that is,*

$$\lim_{t\to\infty} w_j(t) = \pm \left\| w_j(0) \right\|_2 z_j \tag{3.28}$$

for $j = N, N-1, \ldots, N-M+1$.

This theorem, which looks like an extension of Theorem 84, is given without demonstration! In fact, in [121] it is justified by analogy with the PSA LUO learning law (i.e., the same demonstration as that of Theorem 84 is suggested). Considering that the analogy cannot be used entirely because the upper bounds used have no sense for MSA, it can be deduced that Theorem 85 is false. A qualitative analysis of the proof is given in the next section.

3.3.2 MSA EXIN Neural Network

Using the same adaptive Gram–Schmidt orthonormalization as in Section 3.3.1 and the MCA EXIN learning law (2.35), the MSA EXIN learning law is deduced [compare with eq. (3.21)]:

$$\begin{aligned} w_j(t+1) = w_j(t) - \frac{\alpha(t)\, y_j(t)}{w^T(t)\, w(t)} \Bigg[& x(t) - \sum_{i>j} y_i(t)\, w_i(t) \\ & - \frac{w_j^T(t) \left(x(t) - \sum_{i>j} y_i(t)\, w_i(t) \right) w_j(t)}{w^T(t)\, w(t)} \Bigg] \end{aligned} \tag{3.29}$$

for $j = N, N-1, \ldots, N-M+1$. The corresponding averaging ODE is given by

$$\begin{aligned} \frac{dw_j(t)}{dt} &= -\Psi_j^{-1}(t) R_j w_j(t) + \Psi_j^{-2}(t) \phi_j(t) w_j(t) \\ &= -\Psi_j^{-1} \left[R_j - r\left(w_j(t), R_j \right) I \right] w_j(t) \end{aligned} \tag{3.30}$$

where r represents the Rayleigh quotient. Multiplying eq. (3.30) by $w_j^T(t)$ on the left yields

$$\begin{aligned} \frac{d \left\| w_j(t) \right\|_2^2}{dt} &= -\Psi_j^{-1}(t)\, w_j^T(t) R_j w_j(t) + \Psi_j^{-2}(t) \phi_j(t)\, w_j^T(t)\, w_j(t) \\ &= -\Psi_j^{-1}(t) \phi_j(t) + \Psi_j^{-2}(t) \phi_j(t) \Psi_j(t) = 0 \end{aligned} \tag{3.31}$$

for $j = N, N-1, \ldots, N-M+1$. Then

$$\left\| w_j(t) \right\|_2^2 = \left\| w_j(0) \right\|_2^2, \qquad t > 0 \tag{3.32}$$

This analysis and the following considerations use the averaging ODE (3.30) and so are limited to this approximation (i.e., as seen earlier, to the first part of the weight vector time evolution).

The following theorem for the convergence of the learning law holds.

Theorem 86 (Convergence to the MS) *If the initial values of the weight vector satisfy* $w_j^T(0) z_j \neq 0$ *for* $j = N, N-1, \ldots, N-M+1$, *then the weight vectors provided by eq.* (*3.29*) *converge, in the first part of their evolution, to the minor subspace spanned by the corresponding minor components [i.e., eq.* (*3.27*) *holds].*

Proof. The proof is equal to the proof in [122] because MSA EXIN can be deduced only by dividing the learning rate of MSA by $\left\| w_j(t) \right\|_2^4$. In particular, the same analysis as already done for LUO and MCA EXIN can be repeated, and it is easy to notice that if the initial weight vector has modulus of less than 1, then MSA EXIN approaches the minor subspace more quickly than MSA LUO. ■

Now, a qualitative analysis of the temporal evolution of MSA EXIN and MSA LUO is given. The quantitative analysis will be the subject of future work. First, consider eq. (3.22), which represents an adaptive Gram–Schmidt orthonormalization and is input to the jth neuron. It means that the M neurons of the MSA LUO and EXIN neural networks are not independent, except the neuron for $j = N$ (*the driving neuron*), which converges to the first minor component. It can be argued that the neurons labeled by $j < N$ are driven by the weight vectors of the neurons with labels $i > j$. As will be clear from the analysis leading to Proposition 57, the cost landscape of the MCA neurons has N critical directions in the domain $\Re^N - \{0\}$. Among these, all the directions associated with the nonminor components are saddle directions, which are minima in any direction of the space spanned by bigger eigenvectors (in the sense of eigenvectors associated with bigger eigenvalues) but are maxima in any direction of the space spanned by smaller eigenvectors. The critical direction associated with the minimum eigenvalue is the only global minimum. With these considerations in mind, let's review the temporal evolution.

- *Transient*. The driving neuron weight vector tends to the minor component direction, staying approximately on a hypersphere of radius equal to the initial weight modulus. The other weight vectors stay in a first approximation on the respective hyperspheres [see eq. (3.32)] and tend to subspaces orthogonal to the eigenvectors associated with the smaller eigenvalues (the *lower subspaces*) because the adaptive orthogonalization eliminates degrees of freedom and the corresponding saddle in the cost landscape becomes a

minimum. MSA EXIN is faster than MSA LUO in approaching the minor subspace for initial weights of modulus less than 1.

- *MC direction.* The weight vectors of the driving neurons for both neural networks diverge, fluctuating around the first MC direction. Weight moduli soon become larger than unity. Hence, the fluctuations, which are orthogonal to the first MC direction, as shown in Section 2.6.2, are larger for LUO than for EXIN, because the latter has the positive smoothing effect of the squared weight norm at the denominator of the learning law. Hence, its weight vector remains closer to the first MC direction. The driven neurons have inputs with nonnull projections in the lower subspaces.
- *Loss of step.* If the driving neurons have too large (with respect to MC) orthogonal weight increments, the driven neurons lose their orthogonal direction, and their associated minima become saddles again. So the weights tend to the minor component directions associated with the lower eigenvalues until they reach the first minor component direction. This loss of the desired direction is called a loss of step, in analogy with electrical machines. As a result, this loss propagates from neuron $N-1$ to neuron $N-M+1$, and the corresponding weights reach the first MC direction in the same order. Because of its larger weight increments, MSA LUO *always* loses a step before MSA EXIN. Furthermore, MSA EXIN has weight vectors staying longer near their correct directions.[1]
- *Sudden divergence.* The driving neuron of MSA EXIN diverges at a finite time, as demonstrated in Section 2.6.2. The other neurons, once they have lost a step, have the same behavior as that of the driving neuron (i.e., they diverge at a finite time).

As a consequence of its temporal behavior, MSA LUO is useless for applications both because their weights do not stay long in their correct directions and for the sudden divergence problem.

3.3.2.1 Simulations The first example uses, as benchmark, Problem 6.7 in [121, p. 228]. Here,

$$R = \begin{bmatrix} 2.7262 & 1.1921 & 1.0135 & -3.7443 \\ 1.1921 & 5.7048 & 4.3103 & -3.0969 \\ 1.0135 & 4.3103 & 5.1518 & -2.6711 \\ -3.7443 & -3.0969 & -2.6711 & 9.4544 \end{bmatrix}$$

has distinct eigenvalues $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$, where $\lambda_1 = 1.0484$ with corresponding eigenvector $z_1 = [0.9052, 0.0560, -0.0082, 0.4212]^T$ and $\lambda_2 = 1.1048$ with corresponding eigenvector $z_2 = [-0.0506, 0.6908, -0.7212, 0.0028]^T$. MSA EXIN has common learning rate $\alpha(t) = \text{const} = 0.001$. The initial

[1]This phenomenon is different from the loss of step, but is not independent.

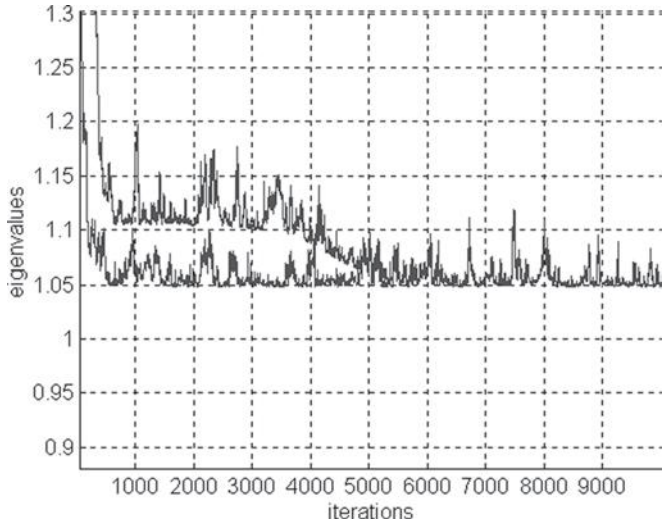


Figure 3.5 First example for MSA EXIN: eigenvalue estimation.

conditions are $w_1(0) = e_1$ and $w_2(0) = e_2$, where e_i is the coordinate vector with all zeros except the ith element, which is 1. MSA EXIN yields, after 1000 iterations, the following estimations:

$$\tilde{\lambda}_1 = \frac{\phi_1(1000)}{\Psi_1(1000)} = 1.0510 \qquad w_1(1000) = [0.9152, 0.0278, -0.0054, 0.4215]^T$$

$$\tilde{\lambda}_2 = \frac{\phi_2(1000)}{\Psi_2(1000)} = 1.1077 \qquad w_2(1000) = [0.0185, 0.7165, -0.7129, 0.0407]^T$$

which are in very good agreement with the true values. Define P_{ij} as $z_i^T w_j / \|z_i\|_2 \left\|w_j\right\|_2$. Figure 3.5 is a temporal plot of the eigenvalue estimation. Note the loss of step, which begins after about 3000 iterations. After 6000 iterations the two computed eigenvalues are equal because the two weight vectors, w_1 and w_2, are parallel. This can also be seen clearly in Figure 3.6, which is a plot of P_{12}. Figures 3.7 and 3.8 show the accurate recovery of the minor subspace. Figure 3.9 is an estimation of the eigenvalues when the experimental setup of Section 2.6.2.4 is used (i.e., the true solutions are taken as initial conditions; second example). The plot shows both that the loss of step happens after the minor component direction is reached and that there is no sudden divergence.

MSA LUO has the same learning rate as MSA EXIN and initial conditions $w_1(0) = e_1$ and $w_2(0) = e_2$. MSA LUO yields, after 1000 iterations, the following estimations:

$$\tilde{\lambda}_1 = \frac{\phi_1(1000)}{\Psi_1(1000)} = 1.0508 \qquad w_1(1000) = [0.9154, 0.0219, 0.0160, 0.4362]^T$$

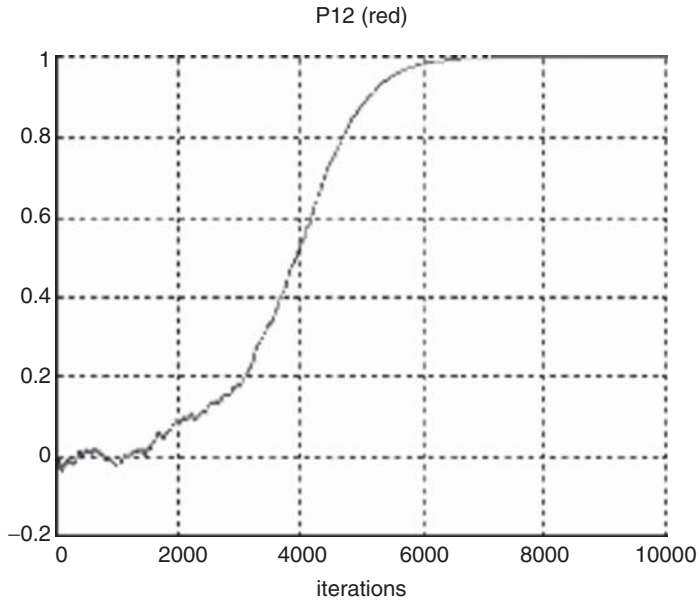


Figure 3.6 First example for MSA EXIN: P_{12}.

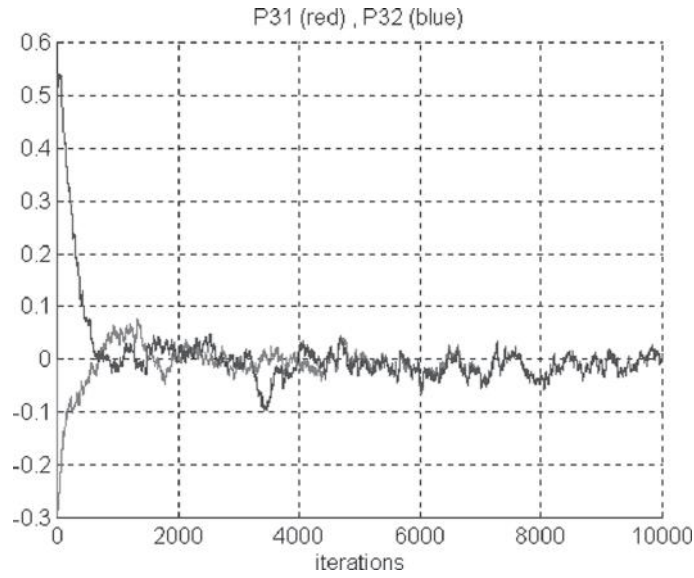


Figure 3.7 First example for MSA EXIN: P_{31} and P_{32}. (*See insert for color representation of the figure*.)

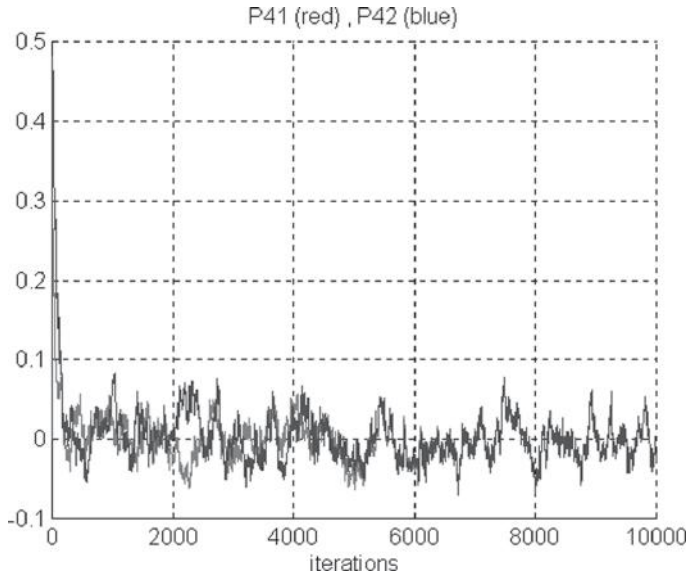


Figure 3.8 First example for MSA EXIN: P_{41} and P_{42}. (*See insert for color representation of the figure*.)

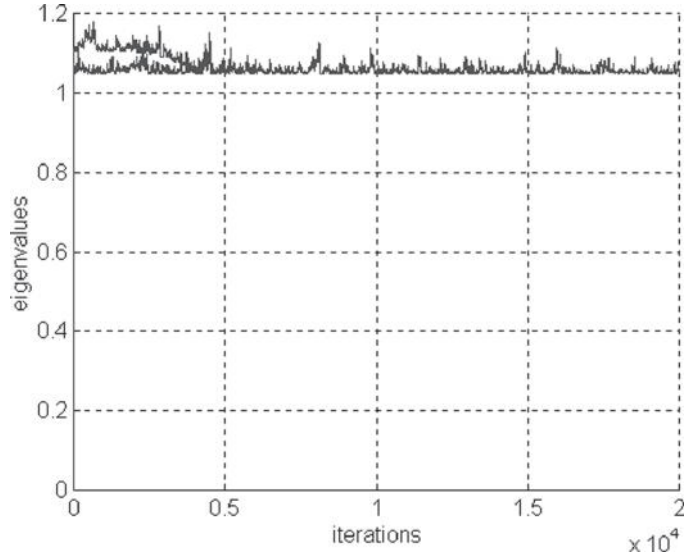


Figure 3.9 Second example for MSA EXIN: eigenvalue estimation.

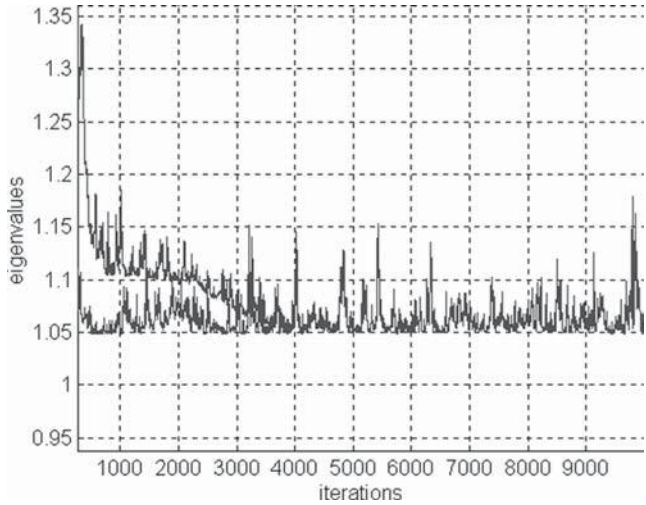


Figure 3.10 First example for MSA LUO: eigenvalue estimation.

$$\tilde{\lambda}_2 = \frac{\phi_2\,(1000)}{\Psi_2\,(1000)} = 1.1094 \qquad w_2\,(1000) = [0.0414, 0.7109, -0.7263, 0.0744]^T$$

which are similar to the MSA EXIN results. Figure 3.10 shows that the loss of step begins after only 1000 iterations. After 4000 iterations, w_1 and w_2, are parallel. This is also shown in Figure 3.11, which is a plot of P_{12}. Hence, the loss of step happens much earlier in MSA LUO than in MSA EXIN. Note in Figure 3.11 that there are no flat portions of the curve before the loss of step (in MSA EXIN there is a big portion with nearly null gradient); this fact does not allow us to devise a good stop criterion. Figures 3.12 and 3.13 show the accurate recovery of the minor subspace. Figures 3.14, 3.15, and 3.16 show the results for λ_1 and λ_2 together, for λ_2 alone, and for P_{12}, respectively, when the true solutions are taken as initial conditions. λ_2 goes to infinity at iteration $20,168$, λ_1 at iteration $21,229$.

The third example uses, as a benchmark, Problem 6.8 in [121, p. 229]. Here,

$$R = \begin{bmatrix} 2.5678 & 1.1626 & 0.9302 & -3.5269 \\ 1.1626 & 6.1180 & 4.3627 & -3.4317 \\ 0.9302 & 4.3627 & 4.7218 & -2.7970 \\ -3.5269 & -3.4317 & -2.7970 & 9.0905 \end{bmatrix}$$

has nearly nondistinct eigenvalues, $\lambda_1 = 1.0000$ and $\lambda_2 = 1.0001$, with corresponding eigenvectors

$$z_1 = [0.9070, 0.0616, 0.0305, 0.4433]^T$$
$$z_2 = [-0.2866, 0.6078, -0.7308, -0.1198]^T$$

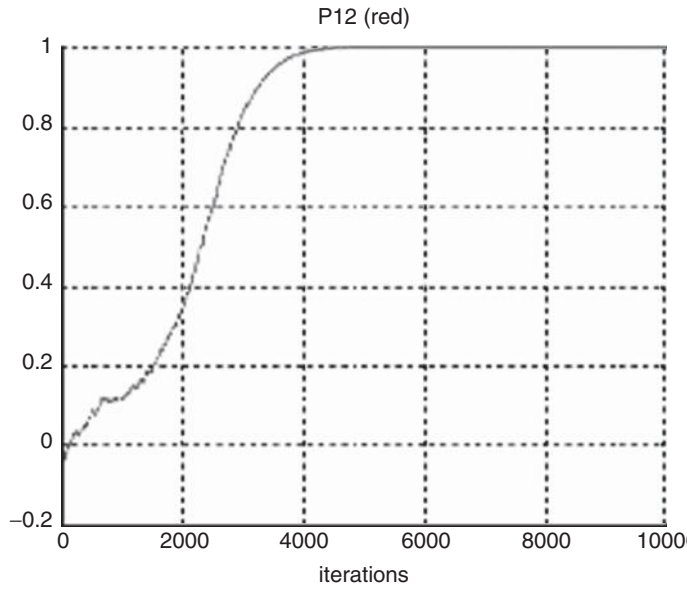


Figure 3.11 First example for MSA LUO: P_{12}.

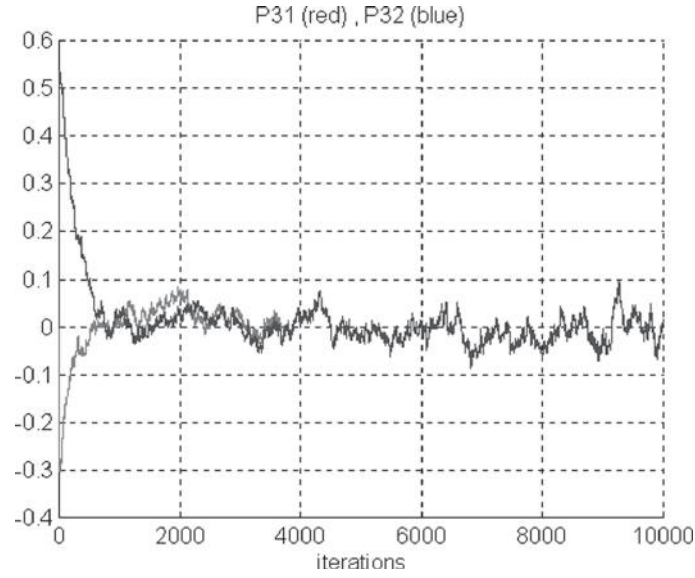


Figure 3.12 First example for MSA LUO: P_{31} and P_{32}. (*See insert for color representation of the figure*.)

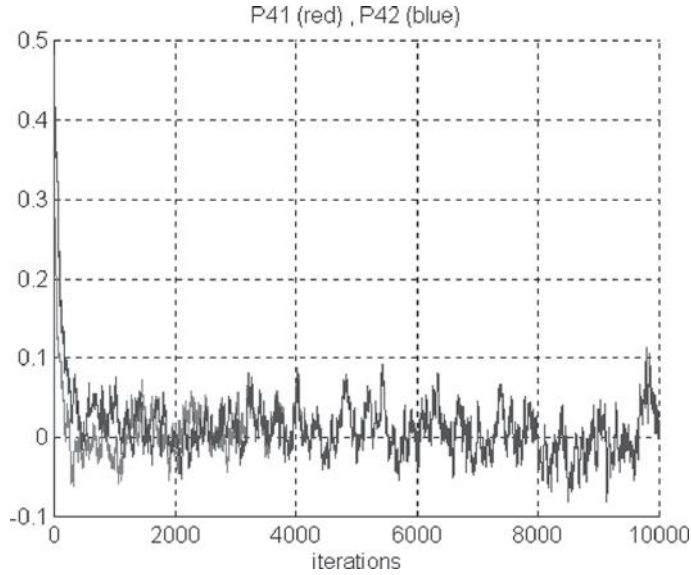


Figure 3.13 First example for MSA LUO: P_{41} and P_{42}. (*See insert for color representation of the figure*.)

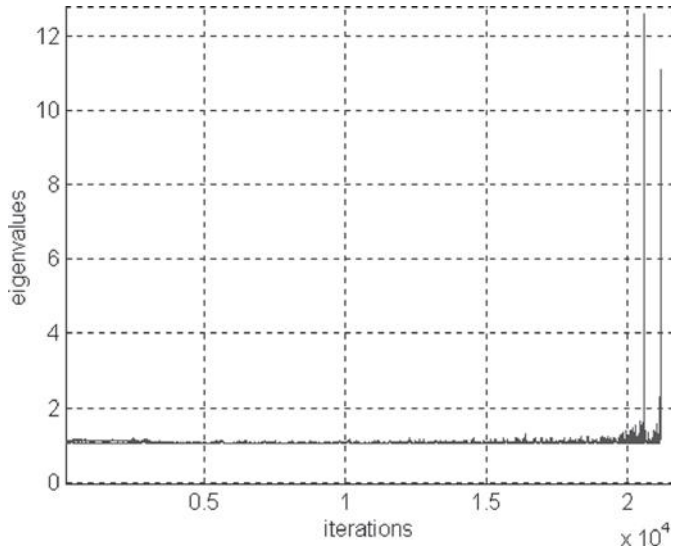


Figure 3.14 Second example for MSA LUO: eigenvalue estimation.

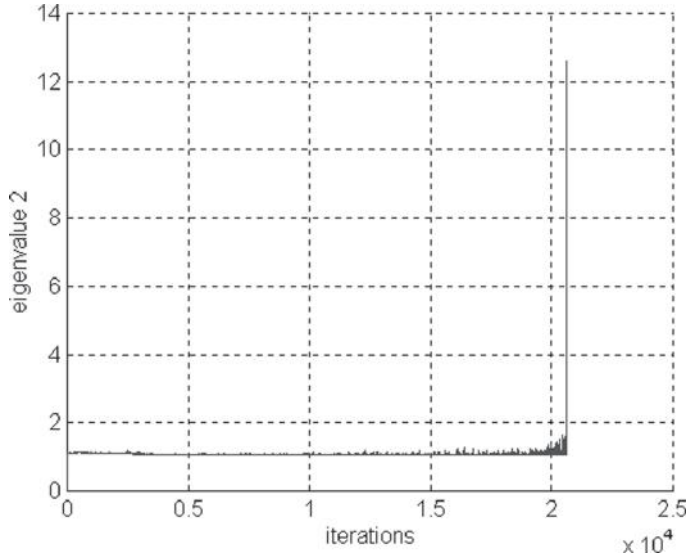


Figure 3.15 Second example for MSA LUO: estimation of λ_2.

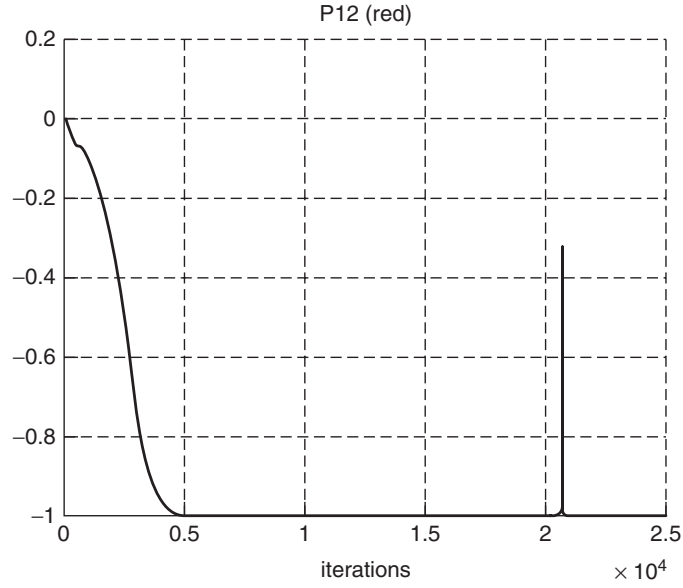


Figure 3.16 Second example for MSA LUO: P_{12}.

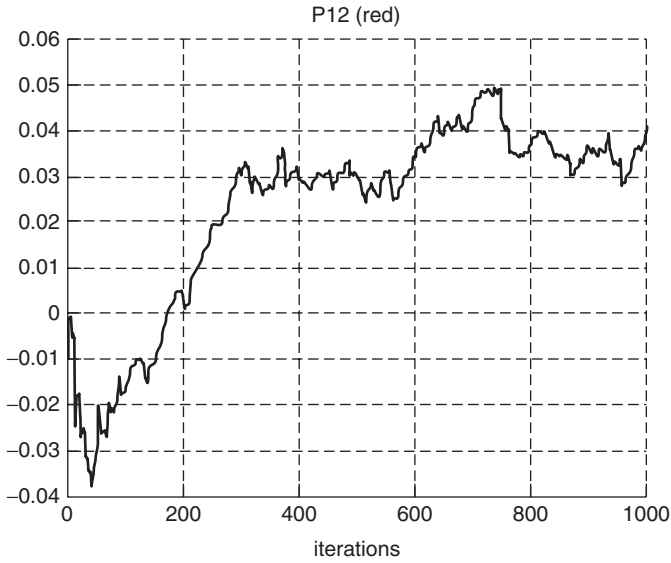


Figure 3.17 Second example for MSA EXIN: P_{12}.

MSA EXIN and MSA LUO have a common learning rate $\alpha(t) = \text{const} = 0.001$. The initial conditions are $w_1(0) = e_1$ and $w_2(0) = e_2$. MSA EXIN yields, after 1000 iterations, the following estimations:

$$\tilde{\lambda}_1 = \frac{\phi_1(1000)}{\Psi_1(1000)} = 1.0028 \qquad w_1(1000) = [0.9070, 0.0616, 0.0305, 0.4433]^T$$

$$\tilde{\lambda}_2 = \frac{\phi_2(1000)}{\Psi_2(1000)} = 1.0032 \qquad w_2(1000) = [0.0592, 0.6889, -0.7393, 0.0797]^T$$

MSA LUO yields, after 1000 iterations, the following eigenvalues:

$$\tilde{\lambda}_1 = \frac{\phi_1(1000)}{\Psi_1(1000)} = 1.0011 \quad \text{and} \quad \tilde{\lambda}_2 = \frac{\phi_2(1000)}{\Psi_2(1000)} = 1.0088$$

Figures 3.17 to 3.21 show some results for both neural networks. Their performances with respect to the nondesired eigenvectors are similar. More interesting is the behavior in the minor subspace. As is evident in Figure 3.16 for MSA EXIN and Figure 3.19 for MSA LUO, MSA EXIN better estimates the minor components and for a longer time. Hence, it can easily be stopped.

3.3.3 Principal Components and Subspace Analysis

This subject is very important, but outside the scope of this book, except from the point of view of a straight extension of the MCA and MSA EXIN learning laws.

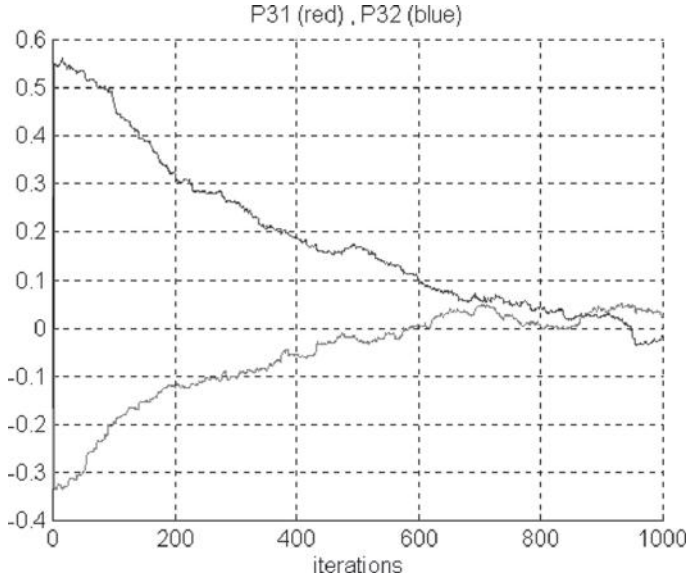


Figure 3.18 Second example for MSA EXIN: P_{31} and P_{32}. (*See insert for color representation of the figure*.)

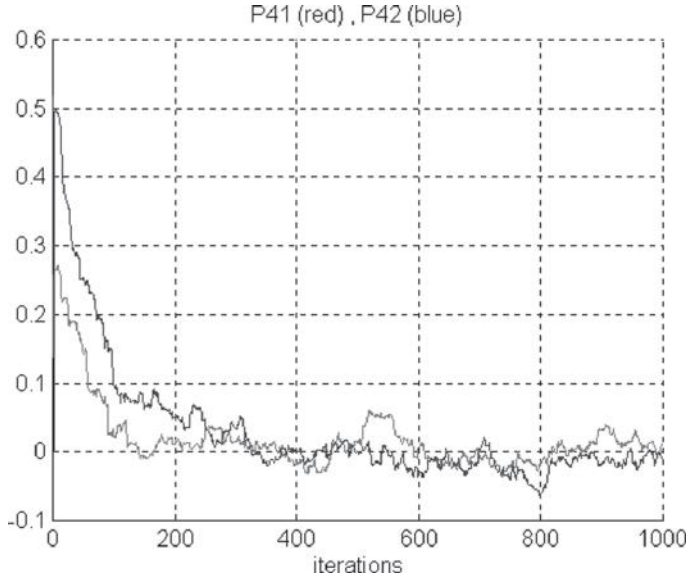


Figure 3.19 Second example for MSA EXIN: P_{41} and P_{42}. (*See insert for color representation of the figure*.)

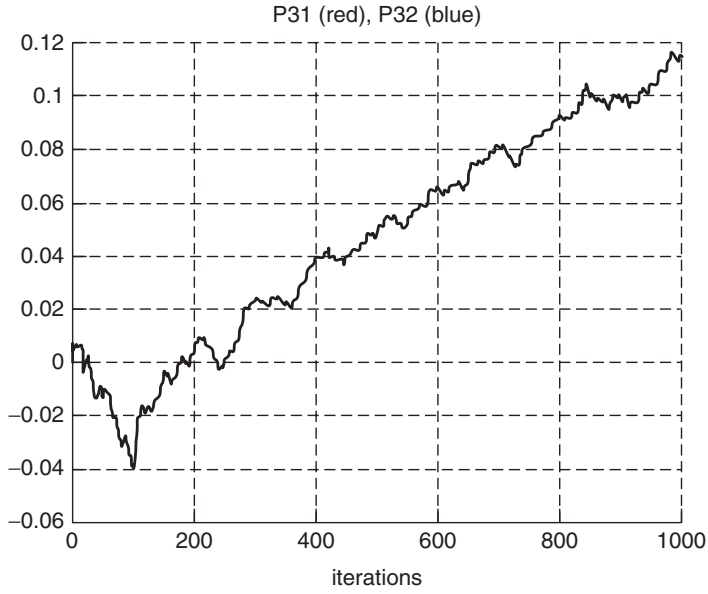


Figure 3.20 Second example for MSA LUO: P_{12}

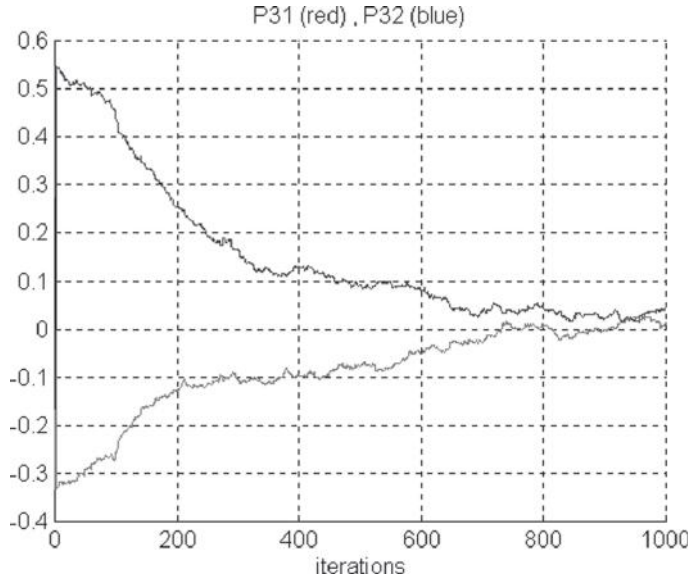


Figure 3.21 Second example for MSA LUO: P_{31} and P_{32}. (*See insert for color representation of the figure*.)

Table 3.4 PCA learning laws

	PCA OJA	PCA OJAn	PCA LUO	PCA EXIN
a_1	1	1	$\|w(t)\|_2^2$	$\|w(t)\|_2^{-2}$
a_2	1	$\|w(t)\|_2^{-2}$	1	$\|w(t)\|_2^{-4}$

Indeed, the formulation of the PCA and PSA (here PSA is intended as finding the principal components) EXIN learning laws derives from the corresponding equations for MCA and MSA simply by reversing a sign. As a consequence, all corresponding theorems are also valid here. The same can be said for OJA, OJAn, and LUO, even if, in reality, historically they were originally conceived for PCA. According to the general analysis for PCA in [121, p. 229], all these PCA algorithms can be represented by the general form

$$w(t+1) = w(t) + a_1\alpha(t)\, y(t)\, x(t) - a_2\alpha(t)\, y^2(t)\, w(t) \tag{3.33}$$

and the particular cases are given in Table 3.4. The corresponding statistically averaging differential equation is the following:

$$\frac{dw(t)}{dt} = a_1 R w(t) - a_2 w^T(t) R w(t)\, w(t) \tag{3.34}$$

From the point of view of the convergence, at a first approximation, it holds that

$$\lim_{t\to\infty} w(t) = \pm\sqrt{\frac{a_1}{a_2}} z_1 \tag{3.35}$$

which is valid on the condition $w^T(0)\, z_1 \neq 0$.

About the divergence analysis, eq. (2.105) still holds exactly for PCA. Hence, all consequences are still valid; in particular, PCA LUO diverges at the finite time given by eq. (2.111). On the contrary, eq. (2.116) is no longer valid, but for PCA OJA it must be replaced by

$$\frac{dp}{dt} = +2\lambda_{\min}(1-p)\, p \qquad\qquad p(0) = p_0 \tag{3.36}$$

whose solution is given by

$$p(t) = \frac{1}{1 + (1-p_0)e^{-2\lambda_{\min} t}/p_0} \tag{3.37}$$

Hence,

$$\lim_{t\to\infty} p(t) = 1 \tag{3.38}$$

and neither divergence nor sudden divergence happens.

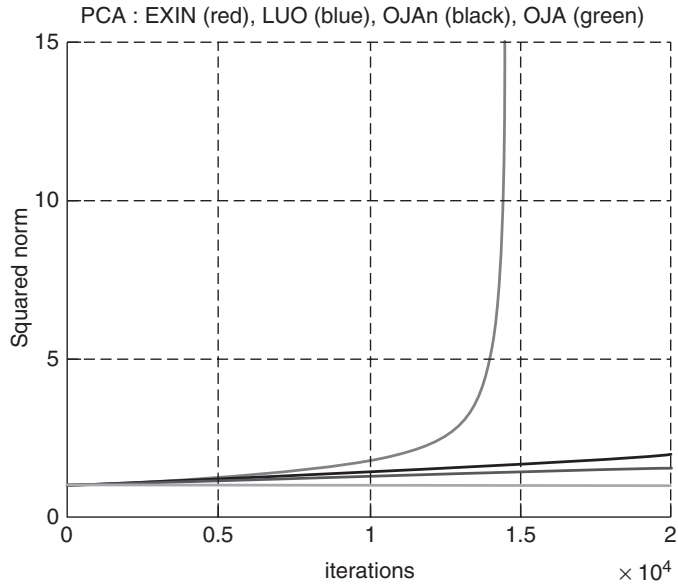


Figure 3.22 Evolution of the squared weight moduli of the PCA linear neurons with initial conditions equal to the solution. (*See insert for color representation of the figure*.)

The following simulation deals with the same example of Section 2.6.2.4, but now the initial condition is given by the first principal component:

$$w(0) = [0.5366, 0.6266, 0.5652]^T$$

and $\lambda_{\max} = 0.7521$. The learning rate is the same. Figures 3.22 and 3.23 confirm the previous analysis regarding the PCA divergence. Figure 3.24 shows an estimation of the largest eigenvalue using the PCA neurons with initial conditions chosen randomly, for every component, in $[-0.1, 0.1]$. PCA EXIN is the fastest, but has sizable oscillations in the transient.

In [140], a neural network is proposed to find a basis for the principal subspace. For the adaptive extraction of the principal components, the stochasting gradient ascent (SGA) algorithm [139,142], the generalized Hebbian algorithm (GHA) [166], the LEAP algorithm [16,17], the PSA LUO [the learning law is eq. (3.21) but with the reversed sign of the weight increment] [125], and the APEX algorithm [53,111,112] are some of the most important methods. In particular, for PSA LUO, the convergence is guaranteeed if the moduli of the weight vectors are greater than 1. A PSA EXIN learning law can be conceived by reversing the

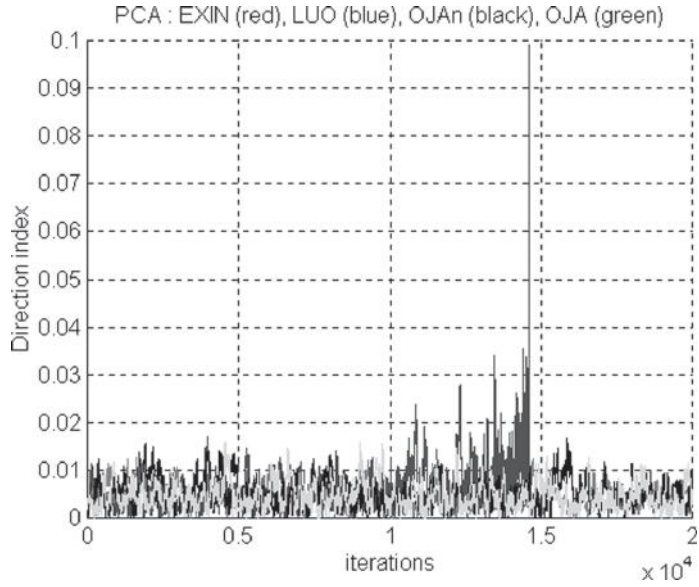


Figure 3.23 Deviation from the PC direction for the PCA neurons, measured as the squared sine of the angle between the weight vector and the PC direction. (*See insert for color representation of the figure*.)

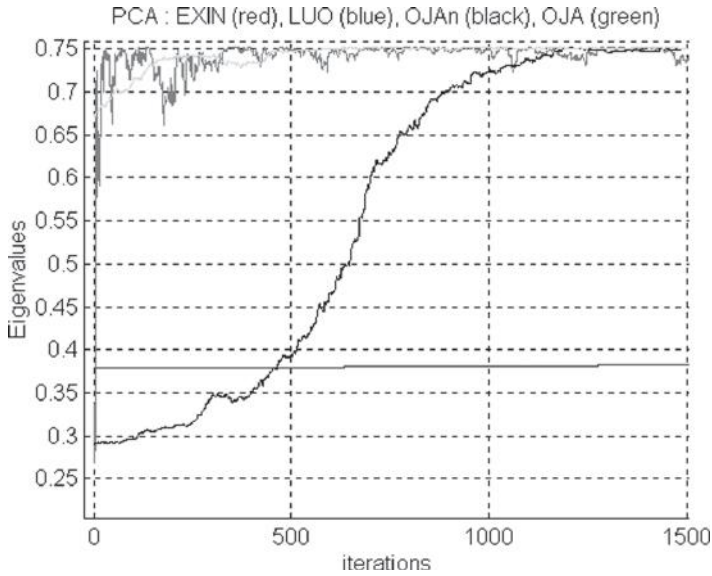


Figure 3.24 Computation of the maximum eigenvalue using PCA neurons. (*See insert for color representation of the figure*.)

sign of the weight increment in eq. (3.29):

$$w_j(t+1) = w_j(t) + \frac{\alpha(t)\, y_j(t)}{w^T(t)\, w(t)} \left[x(t) - \sum_{i>j} y_i(t)\, w_i(t) - \frac{w_j^T(t) \left(x(t) - \sum_{i>j} y_i(t)\, w_i(t) \right) w_j(t)}{w^T(t)\, w(t)} \right] \tag{3.39}$$

This network has not been used in this book and will be the subject of future work. Because of its similarity to the PSA LUO learning law, the same convergence theorem applies, but we think that this convergence theorem can be extended to a more general constraint for the weight moduli because it is based only on upper bounds for the averaging ODE and subsequent formulas.

4

INTRODUCTION TO THE TLS EXIN NEURON*

4.1 FROM MCA EXIN TO TLS EXIN

As seen in Section 1.5.1, given the system $Ax = b$, where $A \in \Re^m \times n$ and $b \in \Re^m$, the TLS solution mimimizes the following cost function:

$$E_{\text{TLS}}(x) = \frac{(Ax - b)^T (Ax - b)}{1 + x^T x} = \frac{\left\| [A; b] \left[x^T; -1 \right]^T \right\|_2^2}{\left\| \left[x^T; -1 \right]^T \right\|_2^2} \tag{4.1}$$

which is the Rayleigh quotient of $[A; b]^T [A; b]$ constrained to the TLS hyperplane (i.e., $x_{n+1} = -1$).

Hence, the TLS solution is parallel to the right singular vector ($\in \Re^{n+1}$) corresponding to the minimum singular value of $[A; b]$. Define

$$\xi_i = \left[a_i^T; b_i \right]^T \quad \vee \quad y_i = \Psi^T \xi_i \tag{4.2}$$

with a_i^T the ith row of A, where y_i is the output of the MCA linear neuron of weight vector $\Psi \in \Re^{n+1}$, input $\xi_i \in \Re^{n+1}$, and learning law minimizing the Rayleigh quotient of the autocorrelation matrix of the input data R, which is *equivalent* to $[A; b]^T [A; b]/m$. Then, to find the TLS solution, the MCA solution

*Sections 4.3.1, 4.3.2, 4.4, and 4.5 are reprinted from G. Cirrincione and M. Cirrincione, "Linear Systems Identification by Using the TLS EXIN Neuron," *Neurocomputing*, Vol. 28, Issue 1–3, pp. 53–74, 1999. Copyright © 1999 Elsevier. Reprinted with permission.

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

must be normalized to have the last component of the solution equal to -1. This way of finding the TLS solution has two fundamental problems:

1. An additional operation of division must be made because of the normalization of the output.
2. All the problems on the behavior of the MCA neurons cited above are valid here.

If the weight vector is constrained to lie on the TLS hyperplane (i.e., losing a degree of freedom), it can be argued that all the convergence problems, which are the consequence of the degeneracy property of the Rayleigh quotient, are no longer valid, because, on the TLS hyperplane , the only critical points are the intersections with the critical straight lines, and therefore they are *isolated* (it implies *the global asymptotic stability* of the solution on the hyperplane plus the eventual solution ∞). This reasoning, illustrated in Figure 4.1 for the three-dimensional Ψ-space, derives from Section 2.5 and will be extended further in this and the next two chapters. Furthermore, no normalization operation would be used.

The following MCA EXIN learning rule solves the TLS problem:

$$\Psi(t+1) = \Psi(t) - \frac{\alpha(t)y(t)}{\Psi^T(t)\Psi(t)}\left[\xi(t) - \frac{y(t)\Psi(t)}{\Psi^T(t)\Psi(t)}\right] \tag{4.3}$$

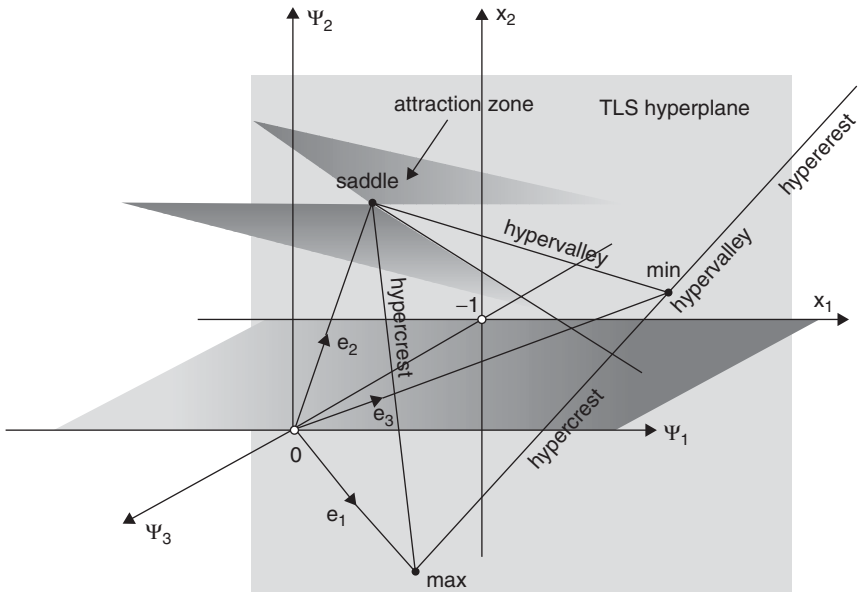


Figure 4.1 Cost landscape stability analysis for MCA EXIN and TLS EXIN (TLS hyperplane in light gray) in $\Re^3$ space.

Replacing $\Psi(t)$ with $\left[x^T(t); -1\right]^T$, $\xi(t)$ with $\left[a_i^T; b_i\right]^T$, i being the index of the row of $[A; b]$ input to the neuron at the time t, and taking only the first n components yields

$$x(t+1) = x(t) - \alpha(t)\,\gamma(t)a_i + \left[\alpha(t)\,\gamma^2(t)\right]x(t) \tag{4.4}$$

where

$$\gamma(t) = \frac{\delta(t)}{1 + x^T(t)x(t)} \tag{4.5}$$

and

$$\delta(t) = x^T(t)a_i - b_i \tag{4.6}$$

This is the *TLS EXIN learning law*. The TLS EXIN neuron is a linear unit with n inputs (vector a_i), n weights (vector x), one output (scalar $y_i = x^T a_i$) and one training error [scalar $\delta(t)$]. With this typology, the training is considered as *supervised*, b_i being the target. The quantity in brackets is positive; it implies that the second term in eq. (4.4) is a *reverse ridge regression* (see Remark 10).

About the input of the neuron, two cases have to be considered:

1. The equations (rows of $[A; b]$) are not known in advance and therefore are fed directly at their presentation.
2. The equations (rows of $[A; b]$) are known in advance and are fed either in a random sequence or in a cyclical order.

In both cases, the stochastic approximation is valid and the learning process is called *online* or *sequential* (as seen above, the same consideration is valid in *block* mode). The second demonstration of the TLS EXIN learning law, which does not use the MCA EXIN learning law and is presented in the next section, will justify its use even in the *batch* mode.

4.2 DETERMINISTIC PROOF AND BATCH MODE

From eq. (1.22),

$$E_{\text{TLS}}(x) = \frac{(Ax-b)^T(Ax-b)}{1+x^Tx} = \frac{\sum_{i=1}^m \left|a_i^Tx - b_i\right|^2}{1+x^Tx} = \sum_{i=1}^m E^{(i)}(x) \tag{4.7}$$

where

$$E^{(i)}(x) = \frac{\left(a_i^Tx - b_i\right)^2}{1+x^Tx} = \frac{\sum_{j=1}^n \left(a_{ij}x_j - b_i\right)^2}{1+x^Tx} = \frac{\delta^2}{1+x^Tx} \tag{4.8}$$

Hence,

$$\frac{dE^{(i)}}{dx} = \frac{\delta a_i}{1 + x^T x} - \frac{\delta^2 x}{\left(1 + x^T x\right)^2} \tag{4.9}$$

and the corresponding steepest descent discrete-time formula is just eq. (4.4), which can be considered as the sequential learning law for solving the TLS problem.

Given the system $Ax = b$, where $A \in \Re^{m \times n}$ and $b \in \Re^m$, the *training set* of the neuron is made of the m rows a_i and the m corresponding targets b_i. In the *batch* mode, the m equations are presented to the neuron, which builds the global error gradient,

$$\frac{dE}{dx} = \sum_{i=1}^{m} \frac{dE^{(i)}}{dx} \tag{4.10}$$

using eq. (4.9), while keeping its weights constant. The presentation of the whole set is an *epoch*. After each epoch the weights are updated using the simple steepest descent method:

$$x(t+1) = x(t) - \alpha(t) \frac{dE}{dx} \tag{4.11}$$

See [21, Sec. 3.7.5] for a comparison of online and batch procedures. The most important considerations are:

- The online approach has to be used if all training examples are not available before the learning starts and an adaptation to the actual stream of training patterns is desired.
- If high precision is required, the batch procedure can be the method of choice.
- The batch approach lends itself to straightforward applications to the more sophisticated optimization procedures, as shown in the next section.

4.3 ACCELERATION TECHNIQUES

4.3.1 Scaled Conjugate Gradient TLS EXIN

Several ways of accelerating the convergence rate of MCA EXIN (in block mode) and TLS EXIN (in block and batch mode) have been implemented. As pointed in remark [45], the conjugate gradient method (CG 6,7,59,66,85,86 103,109,127,155,157,170,187,189) can deal with singular Hessian matrices, so it can be applied in the case of MCA EXIN, and, a fortiori, in the case of TLS EXIN, where the Hessian is no longer singular, as shown in the next section.

For both neurons, the search directions of the algorithm are taken conjugate with respect to the Hessian matrix:

$$d_i^T H d_j = 0 \qquad \forall i \neq j \tag{4.12}$$

where the d_i's are the search directions (at time instants i). Hence, E being the cost function, the CG algorithm can be formulated as

$$w(t+1) = w(t) + \alpha(t) d(t) \tag{4.13}$$

$$d(0) = -\nabla E(w(0)) \tag{4.14}$$

$$d(t+1) = -\nabla E(w(t+1)) + \beta(t) d(t) \tag{4.15}$$

$$\beta(t) = \frac{\nabla E^T(w(t+1))\,[\nabla E(w(t+1)) - \nabla E(w(t))]}{d^T(t)\nabla E(w(t)} \tag{4.16}$$

where eq. (4.16) is called the *Hestenes–Stiefel formula* (other formulations are possible). The learning rate parameter is defined as

$$\alpha(t) = -\frac{d^T(t)\nabla E(w(t))}{d^T(t) H d(t)} \tag{4.17}$$

and in the case of these two neurons, there is no need to avoid computation of the Hessian matrix using a line minimization because of the a priori knowledge of this matrix [see eq. (2.10)]. The CG algorithm has been derived on the assumption of a quadratic error function with a positive definite Hessian matrix: In this case it finds the minimum after at most n iterations, with n the dimension of the weight vector. This clearly represents a significant improvement on the simple gradient descent approach, which could take a very large number of steps to minimize even a quadratic error function. In the case of MCA EXIN and TLS EXIN, the error function is not quadratic and, for MCA EXIN, the corresponding Hessian matrix near the minimum is not positive definite; this implies the possibility of nondescent directions. To improve the method, the scaled conjugate gradient (SCG) algorithm [133] has been implemented. It combines the CG approach with the model trust region approach of the Levenberg–Marquardt algorithm. It adds some multiple (λ) of the unit matrix to the Hessian matrix, giving the following learning rate:

$$\alpha(t) = -\frac{d^T(t)\nabla E(w(t))}{\delta(t)} = -\frac{d^T(t)\nabla E(w(t)}{d^T(t) H d(t) + \lambda(t)\,\|d(t)\|_2^2} \tag{4.18}$$

For a positive definite Hessian matrix, $\delta(t) > 0$. If this is not the case, the value of $\lambda(t)$ must be increased to make the denominator positive. Let the raised value of $\lambda(t)$ be called $\overline{\lambda}(t)$. In [133] the following value is proposed:

$$\overline{\lambda}(t) = 2\left(\lambda(t) - \frac{\delta(t)}{\|d(t)\|_2^2}\right) \tag{4.19}$$

If the CG local quadratic approximation is poor, the following prescription for $\lambda(t)$ is given 59:

$$\text{if} \quad \Delta(t) > 0.75 \quad \text{then} \quad \lambda(t+1) = \frac{\lambda(t)}{2} \tag{4.20}$$

$$\text{if} \quad \Delta(t) < 0.25 \quad \text{then} \quad \lambda(t+1) = 4\lambda(t) \tag{4.21}$$

where

$$\Delta(t) = \frac{E(w(t)) - E(w(t) + \alpha(t)d(t))}{E(w(t)) - E_Q(w(t) + \alpha(t)d(t))} \tag{4.22}$$

where $E_Q(w(t))$ is the local quadratic approximation to the error function in the neighborhood of the point $w(t)$. If $\Delta(t) < 0$, the weights are not updated. The two stages of increasing $\lambda(t)$ aim to ensure that the denominator is positive and to validate that the local quadratic approximation is applied in succession after each weight update. If the conjugacy of the search directions tends to deteriorate, the algorithm is restarted by resetting the search vector to the negative gradient direction. The SCG approach is fully automated because it includes no critical user-dependent parameters. In a great many problems it offers a significant improvement in speed compared to conventional CG algorithms. In [133] it is asserted that it is faster than the BFGS method. In the simulations for the SCG TLS EXIN and the BFGS TLS EXIN, the contrary will be shown.

4.3.2 BFGS TLS EXIN

Theorem 87 (Hessians) *The $n \times n$ Hessian matrix of the TLS cost function eq. (4.7) is positive definite (which implies the nonsingularity) at the cost function minimum.*

Proof. The Hessian matrix of the Rayleigh quotient $r(u, C)$, given in eq. (2.10), is

$$H_r = \frac{2}{\|u\|_2^2}\left(C - (\nabla r)\, u^T - u\,(\nabla r)^T - rI\right) \tag{4.23}$$

In the TLS approach, the Hessian matrix of the Rayleigh quotient $r\left(u, [A; b]^T [A; b]\right)$ constrained to

$$u_{n+1} = -1 (u_{\text{TLS}} = \left[x^T; -1\right]^T \qquad x \in \Re^n) \tag{4.24}$$

is

$$\begin{aligned} H_{\text{TLS}} &= \frac{2}{1 + x^T x}\left([A; b]^T [A; b] - (\nabla E)\left[x^T; -1\right] - \left[x^T; -1\right]^T (\nabla E)^T - EI_{n+1}\right) \\ &= \frac{2}{1 + x^T x}\begin{bmatrix} A^T A - EI_n - (\nabla E)\, x^T - x\,(\nabla E)^T & A^T b - \nabla E - kx \\ b^T A - kx^T + (\nabla E)^T & b^T b - E + 2k \end{bmatrix} \end{aligned}$$

where

$$E = r\left(u, [A; b]^T [A; b]\right) \tag{4.25}$$

is $E_{\text{TLS}}(u)$ and $k = \partial E / \partial u_{n+1} \big|_{u_{n+1}=-1}$. In the critical points $\left[x_c^T; -1\right]^T$ of the RQ,[1] corresponding to the singular values σ_c of $[A; b]$ (i.e., the eigenvalues $\lambda_c = \sigma_c^2$ of $[A; b]^T [A; b]$) the Hessian matrix is

$$\begin{aligned} H_{\text{TLS}}^c &= \frac{2}{1 + x_c^T x_c} \begin{bmatrix} A^T A - \sigma_c^2 I_n & A^T b \\ b^T A & b^T b - \sigma_c^2 \end{bmatrix} \\ &\overset{syst.(1.18)}{=} \frac{2}{1 + x_c^T x_c} \begin{bmatrix} A^T A - \sigma_c^2 I_n & \left(A^T A - \sigma_c^2 I_n\right) x_c \\ b^T A & b^T A x_c \\ n & 1 \end{bmatrix} \begin{matrix} n \\ 1 \end{matrix} \end{aligned} \tag{4.26}$$

So H_{TLS}^c is singular because the $(n+1)$th column is a linear combination (by x_c) of the first n columns. The $n \times n$ submatrix

$$h_{\text{TLS}}^c = \frac{2}{1 + x_c^T x_c} \left[A^T A - \sigma_c^2 I_n\right] \tag{4.27}$$

is positive definite in the critical point corresponding to the minimum eigenvalue. Indeed, respecting the notation of Section 1.3, given the EVD of $A^T A = V' \Lambda V'^T$, where $\Lambda = \text{diag}\left(\sigma_1'^2, \sigma_2'^2, \ldots, \sigma_n'^2\right)$, eq. (4.27) yields

$$h_{\text{TLS}}^c = \frac{2}{1 + x_c^T x_c} V' \left[\Lambda - \sigma_c^2 I_n\right] V'^T \tag{4.28}$$

Recalling the interlacing property (Theorem 6),

$$\sigma_1 \geq \sigma_1' \geq \cdots \geq \sigma_n \geq \sigma_n' \geq \sigma_{n+1} \tag{4.29}$$

if the TLS existence condition $\sigma_n' > \sigma_{n+1}$ is valid, then *in the TLS hyperplane, all critical points are a maximum (negative semidefinite Hessian matrix), saddles (indefinite Hessian matrix), and a minimum (positive definite Hessian matrix).* The submatrix h_{TLS}^c is the Hessian matrix of the TLS cost function,

$$E_{\text{TLS}}(x) = \frac{(Ax - b)^T (Ax - b)}{1 + x^T x} = E_{\text{TLS}}(u) \tag{4.30}$$

as a function of $x \in \Re^n$. Indeed,

$$\nabla_x E_{\text{TLS}}(x) = \frac{2}{1 + x^T x} \left[A^T (Ax - b) - x E_{\text{TLS}}\right] \tag{4.31}$$

[1]At the critical point the RQ assumes the value of the corresponding eigenvalue (see Proposition 43).

and

$$H_{\mathrm{TLS}}(x) = \frac{2}{1+x^T x}\left(A^T A - (\nabla_x E_{\mathrm{TLS}})\, x^T - x\,(\nabla_x E_{\mathrm{TLS}})^T - E_{\mathrm{TLS}} I_n\right) \tag{4.32}$$

which, in the critical points, is

$$H_{\mathrm{TLS}}(x_c) = \frac{2}{1+x_c^T x_c}\left[A^T A - \sigma_c^2 I_n\right] = h_{\mathrm{TLS}}^c \qquad \blacksquare$$

As a consequence, the inverse of the Hessian matrix can be computed. This implies the feasibility of Newton's method, which uses the Newton direction

$$d(t) = -H^{-1}\nabla E(w(t)) \tag{4.33}$$

H being the Hessian matrix of $E(w)$ computed at time t. However, the Hessian matrix must be inverted at each step and it is computationally prohibitive; more interesting are the *quasi-Newton* or *variable metric methods* [157,187,189], which are based on eq. (4.33), but instead of calculating H directly and then evaluating its inverse, they build up an approximation to the inverse over a number of steps (compare with Remark 45). This approach generates a sequence of matrices $G(t)$ which represent increasingly accurate approximations to H^{-1} using only information on the first derivatives of the error function. The two most commonly used update formulas for $G(t)$ are the *Davidson–Fletcher–Powell* (DFP) and the *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) procedures. Here, only the BFGS is given because it is generally regarded as superior:

$$w(t+1) = w(t) + \alpha(t)d(t) = w(t) - \alpha(t)G(t)\nabla E(w(t)) \tag{4.34a}$$

$$G(0) = I \tag{4.34b}$$

$$G(t+1) = G(t) + \frac{pp^T}{p^T v} - \frac{(G(t)v)\, v^T G(t)}{v^T G(t) v} + \left(v^T G(t) v\right) uu^T \tag{4.34c}$$

where

$$p = w(t+1) - w(t) \tag{4.34d}$$

$$v = \nabla E(w(t+1)) - \nabla E(w(t)) \tag{4.34e}$$

$$u = \frac{p}{p^T v} - \frac{G(t)v}{v^T G(t) v} \tag{4.34f}$$

where $\alpha(t)$ is found by line minimization. Derivations of this procedure can be found in many standard texts on optimization methods, such as Polak [154] and Luenberger [119]. An important advantage of the quasi-Newton approach over the conjugate gradient method is that the line search does not need to be very accurate since it does not form a critical factor in the algorithm (the line minimizations for the CG must be performed accurately to ensure that the system of conjugate directions and orthogonal gradients is set up correctly). Hence, the simple *backtracking* line search has been implemented for TLS EXIN.

4.3.2.1 Backtracking Algorithm *The constants* $\eta \in (0,1)$ *and* $\tau, \tau'(0 < \tau < \tau')$ are given.

1. *Set* $\alpha = 1$.
2. *Test the relation* $E\left(w(t) + \alpha(t)d(t)\right) \leq E\left(w(t)\right) + \eta\alpha\nabla E^T(w(t))d(t)$.
3. *If the relation is not satisfied, choose a new* α *in* $\left[\tau\alpha, \tau'\alpha\right]$ *and go to* 2. *If it is satisfied, set* $\alpha(t) = \alpha$ *and* $w(t+1) = w(t) + \alpha(t)d(t)$.

Several procedures have been utilized to choose a new trial value of α in step 3. The classical Goldstein–Armijo method simply multiplies the old value of α by $\frac{1}{2}$ or some other constant in $(0,1)$. It is also possible to choose the new α as the minimizer of a polynomial interpolating already computed function and derivative values, subject to being in the interval $\left[\tau\alpha, \tau'\alpha\right]$. The BFGS method with backtracking is globally and superlinearly convergent on uniformly convex problems [13].

For an n-dimensional quadratic form, the sequence of matrices $G(t)$ is guaranteed to converge exactly to the true Hessian matrix inverse after n steps, and the quasi-Newton algorithm would find the exact minimum of the quadratic form after n steps, provided that the line minimizations are performed exactly. In the following, simulations of the BFGS TLS with backtracking will be shown.

4.4 COMPARISON WITH TLS GAO

As anticipated in Section 1.14, the only existing neural network giving the TLS solution directly is the linear neuron of Gao et al. [63,64]. Using the notation of eq. (4.2), the $(n+1)$th-dimensional TLS GAO learning discrete law is

$$\Psi(t+1) = \Psi(t) - \alpha(t)\, y(t)\left[\xi(t) + \Psi(t)\,\xi_{n+1}(t)\right] \tag{4.35}$$

The corresponding ODE is

$$\frac{d\Psi(t)}{dt} = -R\Psi(t) - \Psi^T(t)\,\overline{r}\,\Psi(t) \tag{4.36}$$

where

$$\overline{r} = E\left[\xi_{n+1}(t)\xi(t)\right] = \begin{bmatrix} r \\ \Gamma \end{bmatrix} \tag{4.37}$$

$$R = E\left[\xi(t)\xi^T(t)\right] = \begin{bmatrix} \overline{R} & r \\ r^T & \Gamma \end{bmatrix} \tag{4.38}$$

and $\overline{R} = E\left(a_i a_i^T\right)$, $r = E\left(b_i a_i\right)$, $\Gamma = E\left(b_i^2\right)$. In [64, Theorem 2] it is shown that taking $\Psi_{n+1}(0) = -1$ as an initial condition, $\Psi(t)$ always lies in the TLS

hyperplane and therefore only the first n components of eqs. (4.35) and (4.36) are meaningful. The nth-dimensional TLS GAO ODE is

$$\frac{dx}{dt} = -\left(\overline{R}x - r\right) - \left(r^T x - \Gamma\right) x \tag{4.39}$$

Considering the derivation of the TLS GAO learning law in [64, p. 722] as a gradient descent algorithm using a linearization (it implies the assumptions of small gains and, above all, weight norms much smaller than 1) of the local gradient vector of the instantaneous estimation of the TLS energy function (4.8), it is easy to demonstrate the following proposition.

Proposition 88 (TLS GAO as a TLS EXIN Simplification) *The TLS GAO learning law can be derived as a linearization of the TLS EXIN learning law for weight norms much smaller than 1.*

The most important differences between TLS EXIN and TLS GAO are:

1. The operation cost per iteration is smaller for TLS GAO.
2. As pointed in Remark 40, the TLS GAO learning law does not represent the gradient flow of an error function, not allowing batch methods as CG, SCG, and BFGS as for TLS EXIN.
3. The dependence of TLS GAO on strongly constraining assumptions limits its dynamic behavior. Indeed, violation of the small gain assumption, typical in the transient, implies that TLS EXIN has a better transient and a faster and more accurate response than TLS GAO. Violation of the small weight assumption limits the initial conditions' field of validity of TLS GAO, in the sense of accuracy and dynamical behaviour.

4.5 TLS APPLICATION: ADAPTIVE IIR FILTERING

From the point of view of linear system identification, the TLS neurons can be applied for the parameter estimation of adaptive FIR[2] and IIR filters when the data observed are both contaminated with additive noise [27]. Adaptive IIR filters have lower filter orders and effectively model a wider variety of systems [172] than those of FIR filters. According to the choice of the prediction error, there are two approaches to adaptive IIR filtering:

1. The *output-error method*, which minimizes the output error. If the minimization is performed using the LS criterion, it gives a consistent result but

[2]For the FIR parameter estimation, the TLS neurons are fed with the input signal $s(t)$ and its delayed time $n-1$ values $s(t-1), s(t-2), \ldots, s(t-n+1)$; the desired signal is $b(t)$. After learning, the weights give the FIR parameters, and the error signal δ represents the error signal of the adaptive filter, which, in many adaptive filter applications, has the same importance as the filter parameters [63,64].

may either converge to a local minimum or lead to undesirable nonlinear computations [165,172].

2. The *equation-error method*, which minimizes the equation error. The LS criterion gives a biased estimate. The TLS criterion gives a strong consistent estimate (see Proposition 33).

With the second formulation, the IIR filter output is given by

$$y(t) = \sum_{i=1}^{N-1} s_i(t) d(t-i) + \sum_{i=0}^{M-1} c_i(t) u(t-i) \tag{4.40}$$

with $\{u(t)\}$ and $\{d(t)\}$ the measured input and output sequences from the unknown system. The error signal is defined as

$$e(t) = y(t) - d(t) \tag{4.41}$$

The input vector for the TLS neurons is

$$a(t) = [d(t-1), \ldots, d(t-N+1), u(t), \ldots, u(t-M+1)]^T \tag{4.42}$$

$y(t)$ and $d(t)$ are, respectively, the neuron output and the target. At convergence, the weights yield the unknown parameters $s_i(t)$ and $c_i(t)$. The neuron output error $e(t)$ yields the IIR error signal.

As a benchmark problem [63,64], consider that the unknown system parameters are

$$[s_1, s_2, c_0, c_1, c_2]^T = [1.1993, -0.5156, 0.0705, 0.1410, 0.0705]^T \tag{4.43}$$

The unknown system is driven by a zero-mean uniformly distributed random white signal with a variance of 1.

The initial conditions are null (it agrees with the TLS GAO assumption). The index parameter ρ, introduced in Section 2.10, is used as a measure of the accuracy. The learning rate $\alpha(t)$ is a decreasing third-order polynomial from 0.2 to 0.01, reached at null tangent at iteration 6000; then it is a constant equal to 0.01. The TLS EXIN neuron is compared with the TLS GAO neuron and the recursive least squares (RLS) algorithm [82] based on the equation-error formulation. In the first simulations, the additive input and output white noises are distributed uniformly over $\left[-\kappa/2, \kappa/2\right]$. Figures 4.2 to 4.6 show the dynamic behavior of the weights for the two TLS neurons for $\kappa = 0.4$. They confirm the TLS EXIN features cited above with respect to the behavior of TLS GAO. Figure 4.7 shows the index parameter ρ. An improvement of more than 50 dB is obtained with respect to TLS GAO. Table 4.1 (σ^2 = variance) summarires the results obtained for TLS EXIN and shows the results obtained for TLS GAO and RLS, as given in [63] and [64]. Despite a certain computational burden, TLS EXIN outperforms

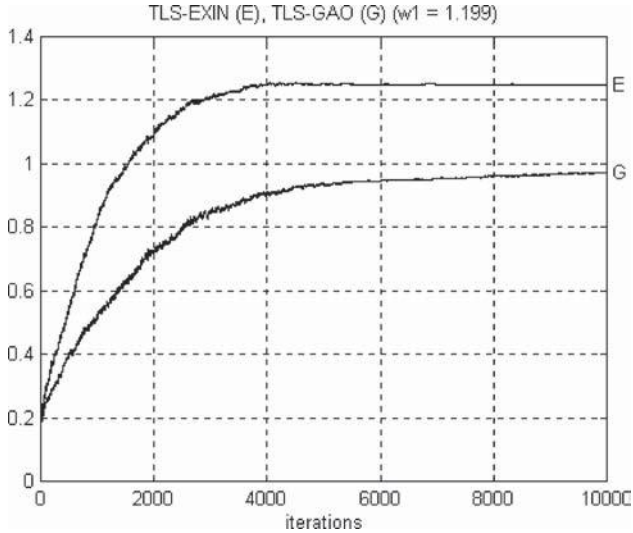


Figure 4.2 $\kappa = 0.4$; averaged over 80 independent trials for w_1.

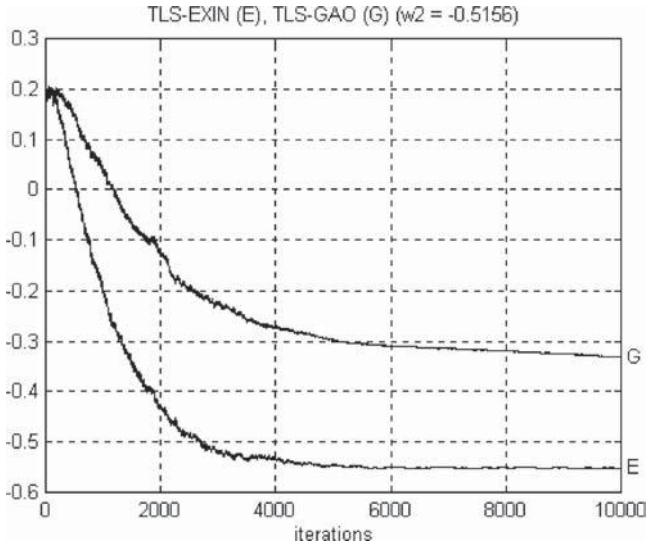


Figure 4.3 $\kappa = 0.4$; averaged over 80 independent trials for w_2.

the other two methods. Figure 4.8 shows, for the same problem, the very good performance of the TLS EXIN acceleration methods. In particular, BFGS has a much superior accuracy. The epochs are actually *blocks*, in order to use these methods online. These acceleration techniques give good results even for high noise levels (see Figure 4.9 for $\kappa = 1$; RLS and TLS GAO yield, respectively,

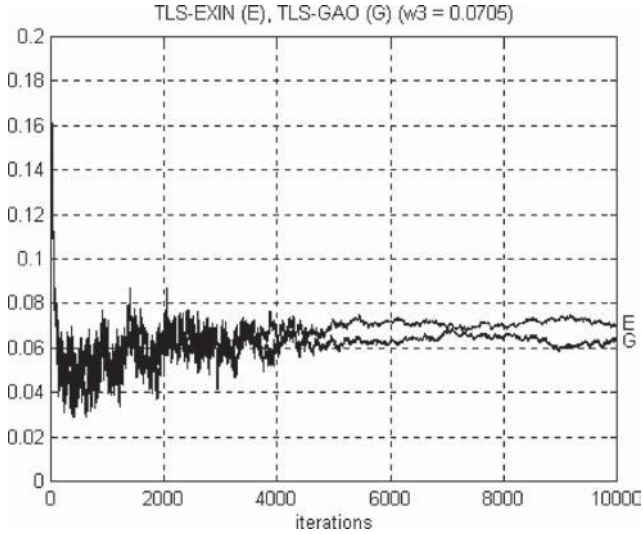


Figure 4.4 $\kappa = 0.4$; averaged over 80 independent trials for w_3.

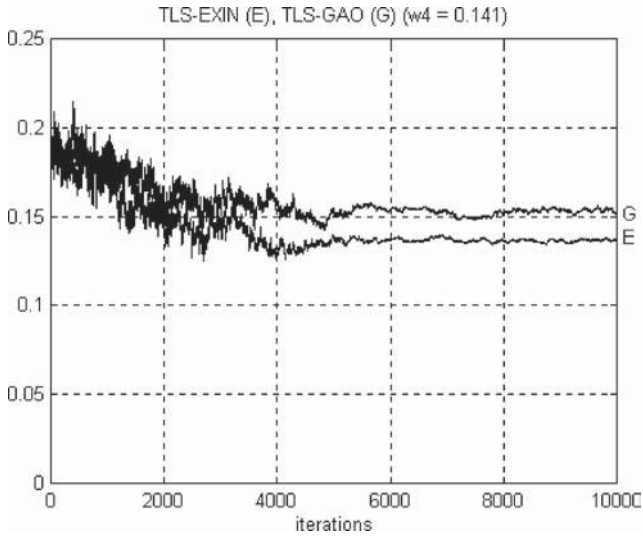


Figure 4.5 $\kappa = 0.4$; averaged over 80 independent trials for w_4.

-8 and -26 dB). Figure 4.10 shows the index parameter ρ for $\kappa = 0.2$. The difference between the two learnings is smaller because, with less noise, the TLS GAO linearization is more acceptable. In the last simulations, the additive input and output noises are zero-mean white Gaussian noise with variance σ^2. Figure 4.11 shows the case for $\sigma^2 = 0.1$: TLS EXIN shows an improvement of

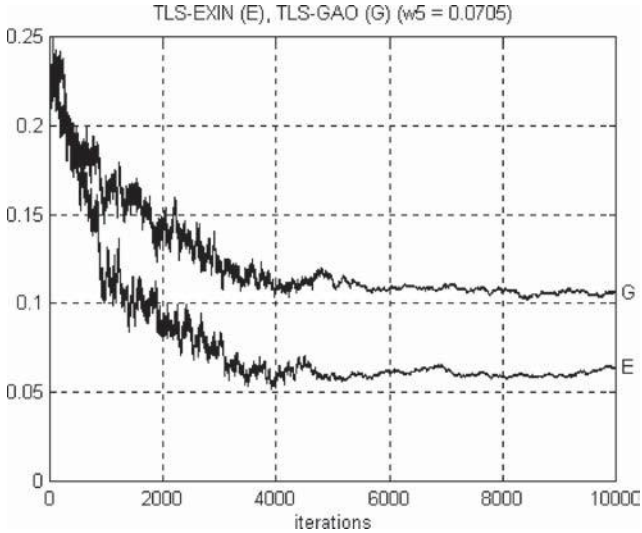


Figure 4.6 $\kappa = 0.4$; averaged over 80 independent trials for w_5.

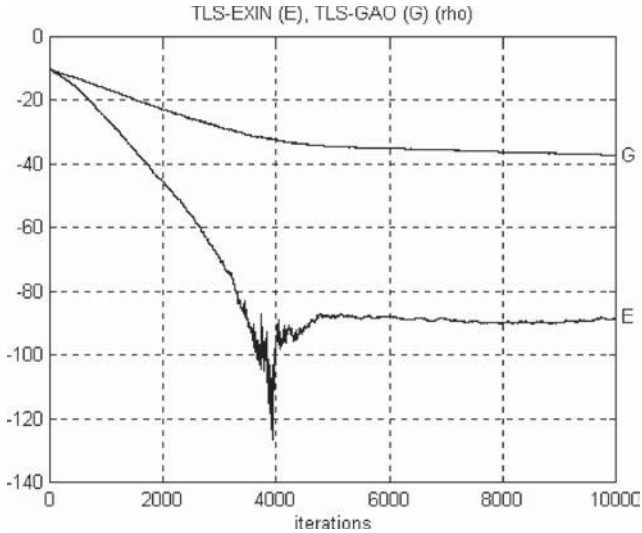


Figure 4.7 $\kappa = 0.4$; averaged over 80 independent trials for the index parameter (expressed in decibels).

slightly less than 15 dB with respect to TLS GAO. Interestingly, as σ^2 increases over 0.2, TLS GAO diverges. This phenomenon can be explained considering that because of its learning law linearization during the transient and in presence of the high noise, the neuron weights are so far from the solution to find themselves

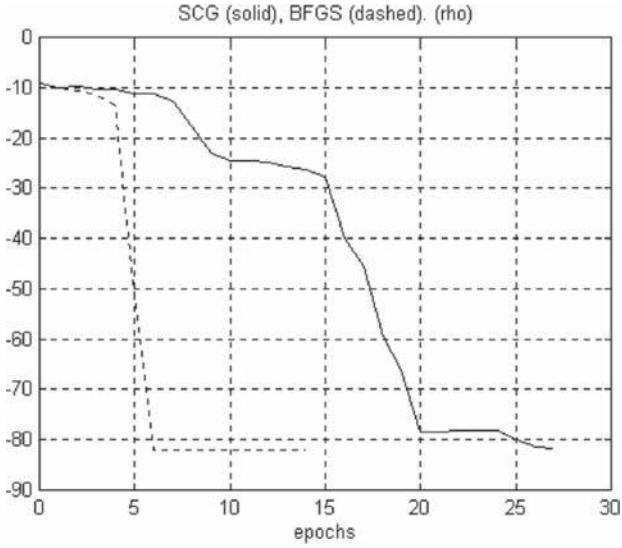


Figure 4.8 TLS EXIN acceleration methods for $\kappa = 0.4$; an epoch represents a block of 500 training vectors.

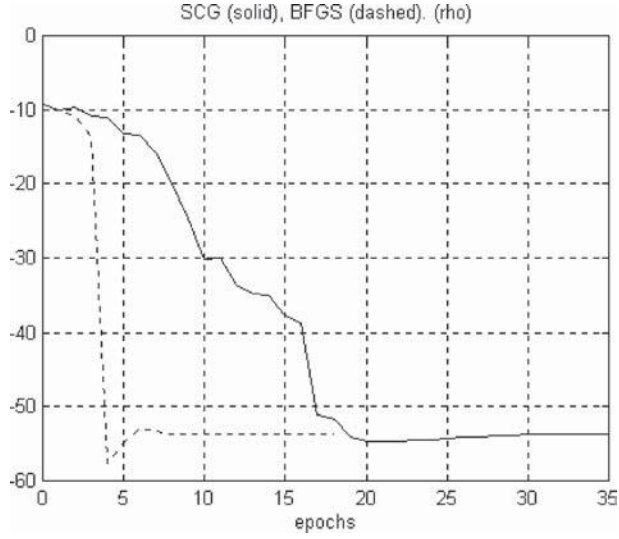


Figure 4.9 TLS EXIN acceleration methods for $\kappa = 1$; an epoch represents a block of 1200 training vectors.

in the divergence domain as seen in the analysis of convergence. TLS EXIN is still convergent, but better results are obtained by its own acceleration techniques. Figure 4.12 shows that even in the presence of very high noise, the BFGS and SCG methods still yield satisfactory results.

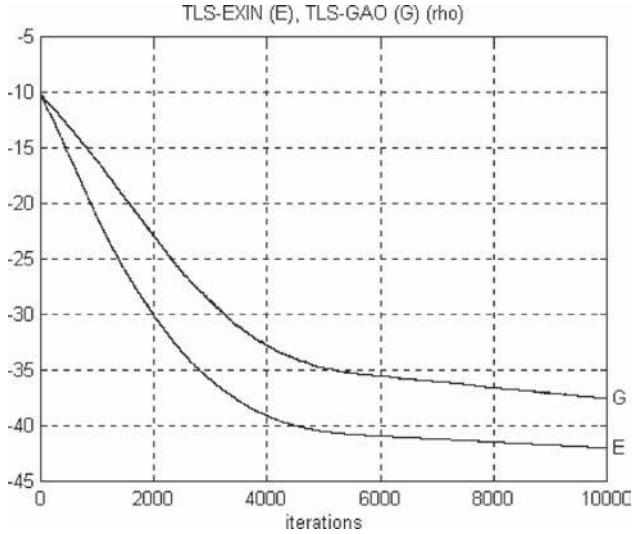


Figure 4.10 $\kappa = 0.2$; averaged over 60 independent trials for the index parameter (expressed in decibels).

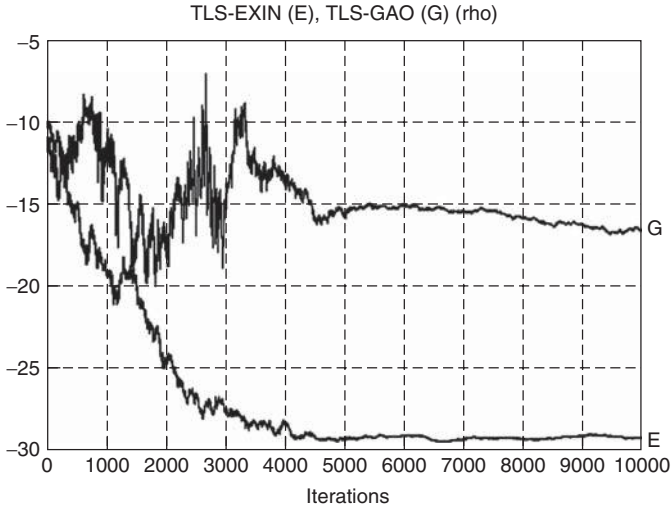


Figure 4.11 Gaussian noise with variance 0.1; averaged over 10 independent trials for the index parameter (expressed in decibels).

4.6 NUMERICAL CONSIDERATIONS

Define $C = [A; b] \in \Re^m \times (n + 1)$ and consider the (one-dimensional) TLS solution. The total number of operations (flops) required by TLS EXIN after the presentation of all the rows of C is given by $2m\,(3n + 1)$. The ratio $k\,(m, n)$ of the operation counts of the partial SVD (see [98, p. 146] with $s = 2$ [76], where

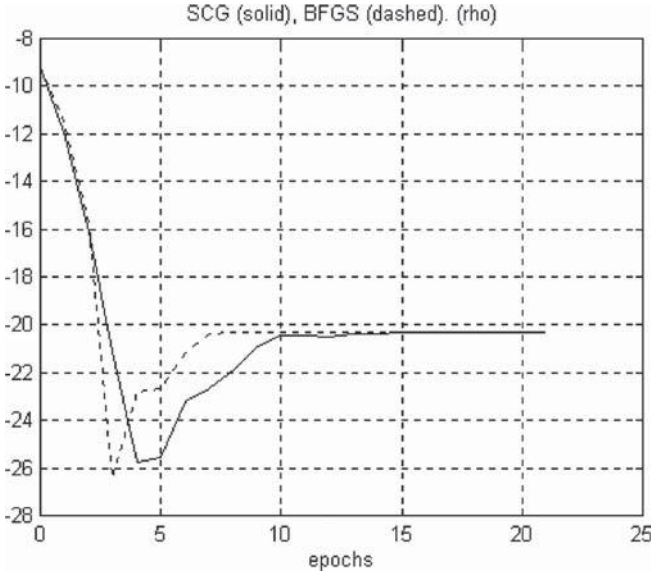


Figure 4.12 TLS EXIN acceleration methods for variance 1; an epoch represents a block of 1000 training vectors.

Table 4.1 Best solutions for adaptive IIR filtering[a]

	s_1	s_2	c_0	c_1	c_2
Desired	1.1993	−0.5156	0.0705	0.1410	0.0705
RLS m.e.	0.6862	0.5516	0.0002	0.0483	0.1155
RLS σ^2	0.0072	0.0064	0.0024	0.0025	0.0028
GAO m.e.	0.0768	0.0617	0.0087	0.0071	0.0155
GAO σ^2	0.0260	0.0192	0.0219	0.0241	0.0241
EXIN m.e.	0.0113	0.0073	0.0018	0.0019	0.0029
EXIN σ^2	0.0253	0.0148	0.0002	0.0003	0.0017

[a] m.e., mean error.

s is the average number of required QR/QL iteration steps for convergence to one basis vector) to that of TLS EXIN is given by

$$k(m,n) = \begin{cases} \dfrac{1}{2m(3n+1)}\left(2mn^2 - \dfrac{2}{3}n^3 + 11n^2 + \dfrac{128}{3}n - 2m - 55\right) & \text{for } m < \dfrac{5n}{3} \\ \dfrac{1}{2m(3n+1)}\left(mn^2 + n^3 + \dfrac{23}{2}n^2 + mn + \dfrac{83}{2}n - 62\right) & \text{for } m \geq \dfrac{5n}{3} \end{cases} \tag{4.44}$$

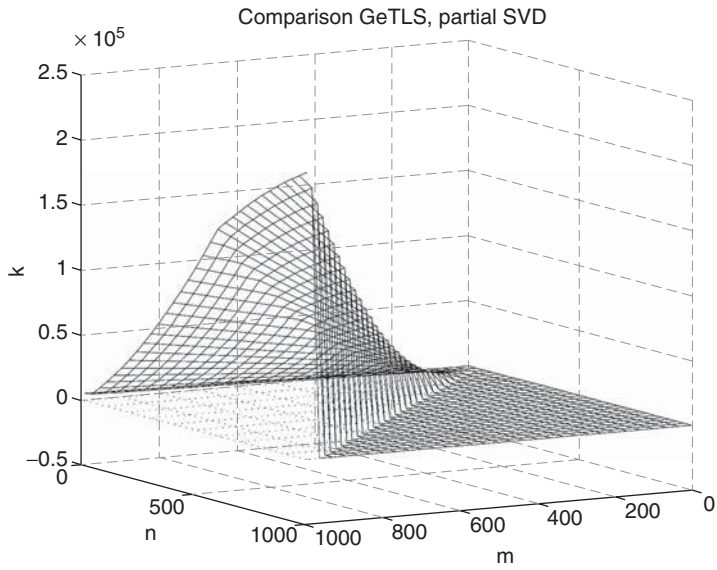


Figure 4.13 Computational cost: partial SVD vs. TLS EXIN.

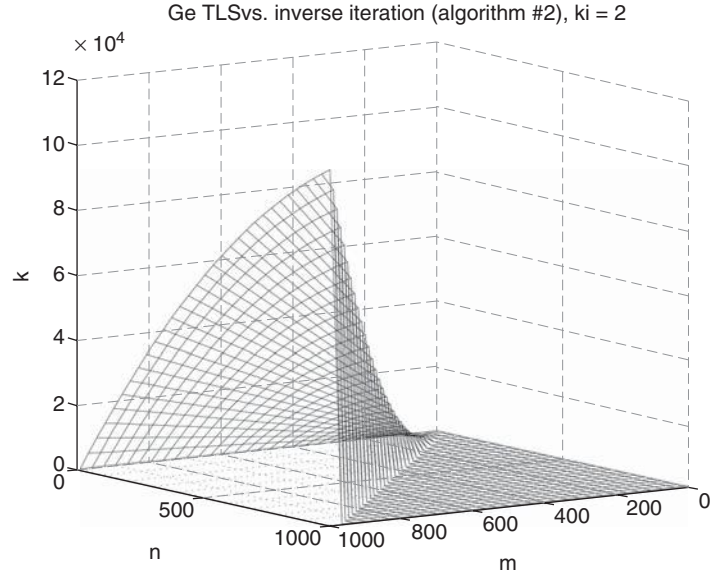


Figure 4.14 Computational cost: inverse iteration method vs.TLS EXIN ($K_i = 2$ is assumed).

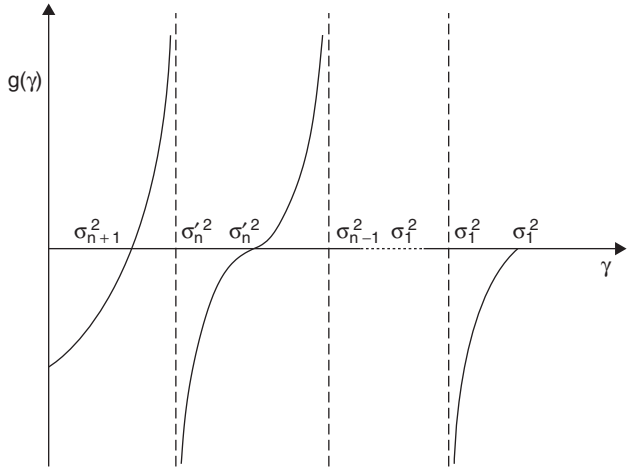


Figure 4.15 Plot of g as a function of the E_{TLS} level. (*See insert for color representation of the figure*.)

and the ratio of the operation counts of the inverse iteration method (see Section 1.13.2) to that of TLS EXIN is given by [98, p. 146 alg. 5.2]

$$k\,(m,n) = \frac{1}{2m\,(3n+1)}\left(\frac{1}{6}n^3 + \frac{3}{4}n^2 + \frac{7}{3}n + n\,(n+7)\,K_i\right) \tag{4.45}$$

where K_i is the number of inverse iteration steps for given C. The two ratios are plotted in Figures 4.13 and 4.14, respectively. Notice the better behavior of the method proposed here for large C.

4.7 TLS COST LANDSCAPE: GEOMETRIC APPROACH

This discussion is based on [156]. Consider the system $Ax = b$, where $A \in \Re^{m\times n}$ and $b \in \Re^m$. The TLS cost function (2.122) results in the ratio of positive definite quadratic forms $Q_N\,(x)$ and $Q_D\,(x)$:

$$E_{\text{TLS}}\,(x) = \frac{(Ax-b)^T\,(Ax-b)}{1+x^Tx} = \frac{x^TA^TAx - 2b^TAx + b^Tb}{1+x^Tx} = \frac{Q_N\,(x)}{Q_D\,(x)} \tag{4.46}$$

Form $Q_N\,(x)$ generally contains second-order terms, first-order terms, and a constant term. It is positive definite, as it is the squared modulus of the nonnull vector $Ax - b$. The form $Q_D\,(x)$ contains only the squared modulus of x plus 1. Thus, $Q_D\,(x)$ is greater or equal to 1.

The family of the equilevel hypersurfaces

$$E_{\text{TLS}}(x) = \gamma \tag{4.47}$$

is introduced, where γ plays the role of the family parameter. According to eq. (4.46), these loci are coincident with the sets of vectors x solving the parametric equation

$$Q_N(x) - \gamma Q_D(x) = x^T A^T A x - 2b^T A x + b^T b - \gamma\left(1 + x^T x\right) = 0 \tag{4.48}$$

So the equilevel hypersurfaces constitute a family of hyperconics with γ as a parameter, because any hypersurface is given by a quadratic form equal to zero.

Equation (4.48) is recast in the form

$$x^T\left[A^T A - \gamma I_n\right]x - 2b^T A x + b^T b - \gamma = 0 \tag{4.49}$$

where I_n is the $n \times n$ identity matrix. In (4.49) the unknown vector x is substituted by $y + x_c$ (translation), where

$$x_c(\gamma) = \left[A^T A - \gamma I_n\right]^{-1} A^T b \tag{4.50}$$

coincides with the center of the hyperconic with level γ. Then a parametric equation in $y \in \Re^n$ follows without first-order terms,

$$y^T\left[A^T A - \gamma I_n\right]y - b^T A\left[A^T A - \gamma I_n\right]^{-1} A^T b + b^T b - \gamma = 0 \tag{4.51}$$

Given the assumption of $\text{rank}(A) = n$, the right singular vectors of A are given by the columns of the orthogonal matrix $V' \in \Re^{n\times n}$, given by the EVD of $A^T A$:

$$A^T A = V' \Lambda V'^T \tag{4.52}$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) = \Sigma'^T \Sigma'$ [see eq. (1.2)], being $\lambda_n < \lambda_{n-1} \leq \cdots \leq \lambda_1$ ($\lambda_i = \sigma_i'^2$, where σ_i' is the ith singular value of A: see Section 1.3). Replacing (4.52) into (4.51) yields

$$y^T\left[V'(\Lambda - \gamma I_n)V'^T\right]y - b^T A V'\left[\Lambda - \gamma I_n\right]^{-1} V'^T A^T b + b^T b - \gamma = 0 \tag{4.53}$$

A rotation $z = V'^T y$ can be performed so that

$$z^T\left[\Lambda - \gamma I_n\right]z - b^T A V'\left[\Lambda - \gamma I_n\right]^{-1} V'^T A^T b + b^T b - \gamma = 0 \tag{4.54}$$

By introducing

$$g(\gamma) = b^T A V'\left[\Lambda - \gamma I_n\right]^{-1} V'^T A^T b - b^T b + \gamma \tag{4.55}$$

eq. (4.54) is rewritten in the simpler form

$$z^T \left[\Lambda - \gamma I_n\right] z = g\,(\gamma) \tag{4.56}$$

The study of this hyperconic family requires analysis of the behavior of the known term $g\,(\gamma)$ vs. γ. For $\gamma = 0$,

$$\begin{aligned} g\,(0) &= b^T A \left[A^T A\right]^{-1} A^T b - b^T b \\ &= -b^T \left[A \left(A^T A\right)^{-1} A^T - I_n\right]^T \left[A \left(A^T A\right)^{-1} A^T - I_n\right] b \leq 0 \end{aligned} \tag{4.57}$$

where the equal sign is valid only for a compatible system of linear equations. For an incompatible system, the negativeness of $g\,(0)$ implies that no hyperconics of the family in eq. (4.56) exist for $\gamma = 0$.

Definition 89 *By introducing the vector* $q \equiv \left[q_1\, q_2\, \cdots\, q_n\right]^T = V'^T A^T b$, *eq.* (*4.55*) *becomes*

$$g\,(\gamma) = \sum_{i=1}^{n} \frac{q_i^2}{\lambda_i - \gamma} + \gamma - b^T b = q^T \left(\Lambda - \gamma I_n\right)^{-1} q + \gamma - b^T b \tag{4.58}$$

Definition 90 (Convergence Keys) *The components* q_i *of the vector* q *will be called the convergence keys because they play a fundamental role in the analysis of the TLS and GTLS domain of convergence.*

Recalling the definition of g in eqs. (1.19) and (1.2), it follows that

$$q = V'^T A^T b = V'^T \left(V' \Sigma'^T U'^T\right) b = \Sigma'^T U'^T b = g \tag{4.59}$$

and therefore eq. (4.58) coincides with eq. (1.20) for $\gamma = \sigma_{n+1}^2$. It can be concluded that eq. (4.58) is a version of the TLS secular equation [74,98] and that this zero of $g\,(\gamma)$ gives the level of the TLS solution [eq. (1.21)].

Disregard the nonrealistic case $q_n = 0$. A direct inspection of eq. (4.58) establishes that the term $g\,(\gamma)$ strictly increases from the negative value at $\gamma = 0$ to ∞ for $\gamma \to \lambda_n^-$. Then one and only one value $\gamma_{\min}$ of γ exists in the interval $(0, \lambda_n)$ such that $g\,(\gamma_{\min}) = 0$. Consequently, the equilevel hyperconics in eq. (4.56) are hyperellipsoids [$(n-1)$-dimensional ellipsoids] $\forall \gamma \in (\gamma_{\min}, \lambda_n)$. These hyperellipsoids are of the same type and dimensionality of the intersection between the cone of the saddle nearer the minimum and the hyperplane $\varepsilon_N = \varepsilon_{n+1} = \text{const} \neq 0$ in the MCA $(n-1)$-dimensional space (see Section 2.5.1.2). For $\gamma = \gamma_{\min}$, the hyperconic collapses into a unique point $z = y = 0$. This point corresponds to the unique minimum of $E_{\text{TLS}}\,(x)$ with position given by

$$\hat{x} = x_c\,(\gamma_{\min}) = \left[A^T A - \gamma_{\min} I_n\right]^{-1} A^T b = \left(A^T A - \sigma_{n+1}^2 I\right)^{-1} A^T b \tag{4.60}$$

[i.e., the closed-form solution of the unidimensional basic TLS problem (1.17)].

Theorem 91 ($g(\gamma)$ Zeroe) *The zeros of eq.* (*4.58*) *coincide with the squared singular values of* $[A; b]$.

Proof. Consider (1.2) [respectively, (1.3)] as the SVD of A (respectively, $[A; b]$) and $\sigma'_n > \sigma_{n+1}$. Since the $n+1$ singular vectors v'_i are eigenvectors of $[A; b]^T[A; b]$, its intersection $[\overline{v}_i'^T; -1]^T$ with the TLS hyperplane satisfies the following set:

$$[A; b]^T[A; b]\begin{bmatrix} \overline{v}'_i \\ -1 \end{bmatrix} = \begin{bmatrix} A^TA & A^Tb \\ b^TA & b^Tb \end{bmatrix}\begin{bmatrix} \overline{v}'_i \\ -1 \end{bmatrix} = \sigma_i^2\begin{bmatrix} \overline{v}'_i \\ -1 \end{bmatrix} \tag{4.61}$$

Transforming (4.61) as

$$\begin{bmatrix} \Sigma'^T\Sigma' & g \\ g^T & \|b\|_2^2 \end{bmatrix}\begin{bmatrix} z \\ -1 \end{bmatrix} = \sigma_i^2\begin{bmatrix} z \\ -1 \end{bmatrix} \quad \text{with} \quad g = \Sigma'^T U'^T b, z = V'^T\overline{v}'_i \tag{4.62}$$

then $\left(\Sigma'^T\Sigma' - \sigma_i^2 I\right) z = g$ and $\sigma_i^2 + g^T z = \|b\|_2^2$. Substituting z in the latter expression by the former yields

$$\sigma_i^2 + g^T\left(\Sigma'^T\Sigma' - \sigma_i^2 I\right)^{-1} g = \|b\|_2^2 \tag{4.63}$$

Recalling that $g = q$ [see eq. (4.59)], this equation coincides with eq. (4.58) and shows that the zeros of $g(\gamma)$ are the squared singular values of $[A; b]$. There are no other zeros because the characteristic equation (4.61) from which the secular equation originates is satisfied only by its eigenvalues. Q.E.D.

Alternative Proof. The characteristic equation of $[A; b]^T[A; b]$ is

$$\begin{aligned} &\det\left[[A; b]^T[A; b] - \gamma I_{n+1}\right] = \det\left[A^TA - \gamma I_n\right] \\ &\left[b^Tb - \gamma - b^TA\left(A^TA - \gamma I_n\right)^{-1}A^Tb\right] = 0 \end{aligned} \tag{4.64}$$

The second factor on the right-hand side coincides with $-g(\gamma)$ in eq. (4.55). Hence, if the squared singular values of $[A; b]$ are distinct from the squared singular values of A, the zeros of $g(\gamma)$ coincide with the eigenvalues of $[A; b]$ (i.e., the squared singular values of $[A; b]$). ■

Corollary 92 (E_{TLS} Critical Points) *The zeros of eq.* (*4.58*) *coincide with the levels of* $E_{\text{TLS}}(x)$ *at its critical points.*

Proof. The squared singular values of $[A; b]$ are the values of the Rayleigh quotient $E_{\text{TLS}}(x)$ at its critical points. ■

The asymptotes of $g(\gamma)$ are given by the squared singular values $\sigma_i'^2 = \lambda_i$ of the data matrix A. They are positioned among the zeros σ_i^2, just respecting the interlacing theorem [eq. (1.16)], as illustrated in Figure 4.15 .

4.8 FIRST CONSIDERATIONS ON THE TLS STABILITY ANALYSIS

By using the theory of Section 2.5 and Figure 4.1, two remarks on the stability of TLS iterative algorithms can be deduced.

Remark 93 (TLS Stability) *The critical directions intersect the TLS hyperplane in points, which are critical for the TLS problem with the same typology of the corresponding critical direction. The cones become volumes of attraction and repulsion for each saddle. The intersections of the hypervalleys and hypercrests with the TLS hyperplane give* $(N-2)$*-dimensional planes, which are, respectively, made up of minima or maxima with respect to the direction of their normal. In the three-dimensional weight space, the critical directions become critical points, and the TLS plane is divided into a zone of attraction and a zone of repulsion (shown only partially in the figure). The straight line passing through the saddle and the maximum is a crest along the minimum direction. The straight line passing through the saddle and the minimum is a valley along the maximum direction. The straight line passing through the minimum and the maximum is a crest along the saddle direction, except for the intersection with the zone of repulsion, which is a valley along the saddle direction.*

Remark 94 (TLS Barrier) *The hypercrest along the minimum direction is not a barrier* (*limit of the basin of attraction*) *for the minimum. Indeed, if the initial conditions for the learning law are chosen in either of the half-spaces delimited by the hypercrest, the weight vector always converges to the minimum direction, orienting according to the half-space containing the initial conditions. The same reasoning is no more valid in the TLS case* (*i.e., the crest is a barrier*) *because there is only one minimum point and therefore it is mandatory to choose the initial conditions in the half-space containing this point. Later we demonstrate the very important property peculiar to TLS EXIN alone that the TLS hyperplane origin* (*intersection of the TLS hyperplane with the axis corresponding to the last coordinate of* $\Re^{n+1}$) *is in the same half space as the minimum. This crest is not a unique TLS barrier; it must be considered in conjunction with the saddle cones.*

5

GENERALIZATION OF LINEAR REGRESSION PROBLEMS

5.1 INTRODUCTION

The three basic problems of linear regression from the point of viw of noisy data are the OLS, the TLS, and the DLS, which have been described in earlier chapters. The problem addressed in this chapter is the generalization of these three cases and a common framework. It will be shown that this analysis will lead to a global stability study of these cases. In the literature, apart from the technique introduced in this chapter, the following approach is the most interesting.

5.1.1 Weighted (Scaled) TLS

Here a special class of weighted TLS [146,147,160] is considered. The problem is the following:

$$\min \|[\Delta A; \alpha \Delta b]\|_F \qquad \text{subject to} \quad b + \Delta b \in R(A + \Delta A) \tag{5.1}$$

where ΔA and Δb are perturbations of A and b, respectively and $\alpha$is a scalar. When $\alpha \to 0$, the solution approaches the OLS solution. When $\alpha = 1$, it coincides with the TLS formulation. Now consider $\alpha \to \infty$. Perturbing b by even a small amount will result in a large value of the cost function. Thus, the optimum solution will be to perturb A such that b lies in the range $A + \Delta A$. This is the same as the DLS problem posed in eq. (1.53). Hence, the α-parameterization unifies and compacts the three basic problems. Let A have full rank (n). Define

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

$C = [A; \alpha b]$. If $b \notin R(A)$, then $\text{rank}(A) = n + 1$. The rank n approximation of C, say $\hat{C}$, is found such that the Frobenius norm of the correction is minimal. The vector in $\ker(\hat{C})$ scaled as $\left[x^T, -1/\alpha\right]^T$ yields a solution of the weighted TLS problem. This formulation has the drawback of requiring an infinite value of α for the DLS problem and holds only asymptotically for the OLS problem.

5.2 GENERALIZED TOTAL LEAST SQUARES (GeTLS EXIN) APPROACH

As illustrated in Proposition 33, in the unidimensional EIV model, under the assumption that the rows of $[\Delta A; \Delta b]$ (i.e., Δa_{ij}, Δb_i, $i = 1, \ldots, m$, $j = 1, \ldots, n$) are i.i.d. with common zero-mean vector and common covariance matrix of the form $\Xi = \sigma_v^2 I_{n+1}$, the TLS method is able to compute strongly consistent estimates of the unknown parameters. In many cases of practical interest, the assumption above is not realistic. For instance, the relative errors $\Delta a_{ij}/a_{ij}$ and $\Delta b_i/b_i$ may be i.i.d. random variables with zero-mean value and then the previous assumption no longer holds. The generalized GeTLS EXIN deals with the case in which the errors Δa_{ij} are i.i.d. with zero mean and common variance σ_a and the errors Δb_i are also i.i.d. with zero mean and common variance $\sigma_b \neq \sigma_a$. This formulation is justified because b and A represent different physical quantities and therefore are often subject to different measurement methods with different accuracy. Solving GeTLS [24,31,35,156], implies finding the vector $\overline{x}$ minimizing

$$\zeta \, \|\Delta A\|_F^2 + (1 - \zeta) \, \|\Delta b\|_F^2 \qquad \text{with} \quad 0 \leq \zeta \leq 1 \tag{5.2}$$

where

$$\frac{\zeta}{1 - \zeta} = \frac{\sigma_a^2}{\sigma_b^2} \tag{5.3}$$

Recalling the OLS, TLS, and DLS formulations:

$$\text{OLS}: \quad \min_{b' \in \Re^m} \left\| b - b' \right\|_2 \quad \text{subject to} \quad b' \in R(A) \tag{5.4a}$$

$$\text{TLS}: \quad \min_{[\hat{A}; \hat{b}] \in \Re^{m \times (n+1)}} \left\| [A; b] - [\hat{A}; \hat{b}] \right\|_F \quad \text{subject to} \quad \hat{b} \in R(\hat{A}) \tag{5.4b}$$

$$\text{DLS}: \quad \min_{A'' \in \Re^{m \times n}} \left\| A - A'' \right\|_F \quad \text{subject to} \quad b \in R(A'') \tag{5.4c}$$

the formulation (5.2) comprises these problems. Indeed, $\zeta = 0$ results in the OLS formulation because $\sigma_a^2 = 0$, whereas $\zeta = 0.5$ results in the TLS formulation because $\sigma_a^2 = \sigma_b^2 = \sigma_v^2$, and $\zeta = 1$ results in the DLS formulation because $\sigma_b^2 = 0$. Equation (5.3) defines the problem for intermediate values of ζ: in an

experiment, variances σ_a^2 and σ_b^2 are estimated under the spherical Gaussian assumption in order to compute the corresponding ζ by (5.3):

$$\zeta = \frac{\sigma_a^2}{\sigma_a^2 + \sigma_b^2} \tag{5.5}$$

The optimal solution of problem (5.2) is given by the minimization of the following cost function:

$$E_{\text{GeTLS EXIN}}(x) = \frac{1}{2}\frac{(Ax-b)^T(Ax-b)}{(1-\zeta)+\zeta x^T x} \tag{5.6}$$

The difference with the weighted (scaled) TLS, which would need a unique scalar parameter going to infinity in order to have the DLS solution, as shown in Section 5.1.1, is apparent in eq. (5.6). The same holds true for the one-parameter generalization in [21]. A similar problem can be found if the augmented matrix $[A; b]$ is pre- and postmultiplied by particular matrices, as in [74]. However, the need for a finite parameter in the numerical algorithms limits the accuracy. This does not happen in the GeTLS EXIN formulation because DLS is represented by a finite value, as a consequence of the fact that the parameter appears directly in the denominator of the cost function (5.6).

For $\zeta = 0$ this cost is the classical OLS sum-of-squared error; for $\zeta = 0.5$ this cost is eq. (1.22). The validity of this expression for the DLS is demonstrated in Section 5.2.4.

5.2.1 GeTLS EXIN Linear Neuron

The function (1.22) can be regarded as the cost function of a linear neuron whose weight vector is $x(t)$. The weight vector converges to the required solution during training. In particular,

$$E_{\text{GeTLS EXIN}}(x) = \sum_{i=1}^{m} E^{(i)}(x) \tag{5.7}$$

where

$$E^{(i)}(x) = \frac{1}{2}\frac{\left(a_i^T x - b_i\right)^2}{(1-\zeta)+\zeta x^T x} = \frac{1}{2}\frac{\delta^2}{(1-\zeta)+\zeta x^T x} \tag{5.8}$$

Hence,

$$\frac{dE^{(i)}}{dx} = \frac{\delta a_i}{(1-\zeta)+\zeta x^T x} - \frac{\delta^2 \zeta x}{\left[(1-\zeta)+\zeta x^T x\right]^2} \tag{5.9}$$

and the corresponding steepest descent discrete-time formula is

$$x(t+1) = x(t) - \alpha(t)\gamma(t)a_i + \left[\zeta\alpha(t)\gamma^2(t)\right]x(t) \tag{5.10}$$

where

$$\gamma(t) = \frac{\delta(t)}{(1-\zeta) + \zeta x^T(t)x(t)} \tag{5.11}$$

Equation (5.10) represents the training law of the GeTLS EXIN linear neuron. It is a linear unit with n inputs (vector a_i), n weights (vector x), one output (scalar $y_i = a_i^T x$), and one training error [scalar $\delta(t)$]. With this architecture, training is considered as *supervised*, b_i being the target. The same is true for the TLS EXIN neuron (see Section 4.1).

Proposition 95 (n-Dimensional GeTLS ODE) *The nth-dimensional ODE of GeTLS EXIN is*

$$\frac{dx}{dt} = \frac{1}{(1-\zeta) + \zeta x^T x} \left\{ -\left(\overline{R}x - r\right) + \frac{\zeta\left[\left(x^T\overline{R}x\right)x - 2\left(x^T r\right)x + \Gamma x\right]}{(1-\zeta) + \zeta x^T x} \right\} \tag{5.12}$$

where $x \in \Re^n$, $\overline{R} = E\left(a_i a_i^T\right)$, $r = E\left(b_i a_i\right)$, and $\Gamma = E\left(b_i^2\right)$.

Proof. Replacing eq. (5.11) in eq. (5.10) yields

$$\begin{aligned} x(t+1) = x(t) - \frac{\alpha(t)}{\left[(1-\zeta) + \zeta x^T(t)x(t)\right]^2}[&(1-\zeta)a_i a_i^T x(t) - (1-\zeta)ba_i \\ &+ \zeta x^T(t)x(t)a_i a_i^T x(t) - \zeta b_i x^T(t)x(t)a_i - \zeta x^T(t)a_i a_i^T x(t)x(t) \\ &+ \zeta b_i x(t)x^T(t)a_i - \zeta b_i^2 x(t)] \end{aligned}$$

Defining $\overline{R} = E\left(a_i a_i^T\right)$, $r = E\left(b_i a_i\right)$, and $\Gamma = E\left(b_i^2\right)$ and averaging according to the stochastic approximation theory yields eq. (5.12). ■

As a particular case, the nth-dimensional TLS EXIN ODE is

$$\frac{dx}{dt} = \frac{1}{1+x^T x} \left\{ -\left(\overline{R}x - r\right) + \frac{1}{1+x^T x}\left[\left(x^T\overline{R}x\right)x - 2\left(x^T r\right)x + \Gamma x\right] \right\} \tag{5.13}$$

5.2.2 Validity of the TLS EXIN ODE

Given $\xi(t) = \left[a_i^T; b_i\right]^T \in \Re^{n+1}$ as input to the MCA EXIN (see Section 4.1), its autocorrelation matrix becomes

$$R = E\left[\xi(t)\xi^T(t)\right] = \begin{bmatrix} \overline{R} & r \\ r^T & \Gamma \end{bmatrix} \approx \frac{1}{m}\begin{bmatrix} A^T A & A^T b \\ b^T A & b^T b \end{bmatrix} = \frac{[A;b]^T[A;b]}{m} \tag{5.14}$$

Theorem 96 (Validity) *The* $(n+1)$th*-dimensional MCA EXIN ODE constrained on the TLS hyperplane is the* $(n+1)$th*-dimensional TLS EXIN ODE and its first n components are eq.* (*5.13*)*, while its* $(n+1)$th *component is asymptotically an identity. This justifies the use of eq.* (*5.13*) *as the ODE and the respect of the hyperplane constraint by the discrete TLS EXIN learning law.*

Proof. From the MCA EXIN ODE,

$$\frac{d\Psi(t)}{dt} = -\frac{1}{\Psi^T(t)\,\Psi(t)}\left[R - \frac{\Psi^T(t)\,R\Psi(t)}{\Psi^T(t)\,\Psi(t)}\right]\Psi(t)$$

where R is given by eq. (5.14) and $\Psi \in \Re^{n+1}$. Replacing $\Psi(t)$ with $\left[x^T(t); -1\right]^T$ yields

$$\begin{pmatrix} dx/dt \\ 0 \end{pmatrix} = \frac{1}{1+x^Tx}\left[\begin{pmatrix} -\left(\overline{R}x - r\right) \\ \Gamma - r^Tx \end{pmatrix} + \frac{1}{1+x^Tx}\begin{pmatrix} \left(x^T\overline{R}x - 2x^Tr + \Gamma\right)x \\ -x^T\overline{R}x + 2x^Tr - \Gamma \end{pmatrix}\right] \tag{5.15}$$

Its first n components are eq. (5.13). Define Δ as the $(n+1)$th component of the right-hand side and Δ' as the expression

$$\begin{aligned}\Delta' &= m\left(1+x^Tx\right)\Delta = \left(x^Tx\right)\left(b^Tb\right) - \left(x^Tx\right)b^TAx - x^TA^TAx + b^TAx \\ &= \left(x^Tx\right)\left(b^Tb - b^TAx\right) - x^T\left(A^TAx - A^Tb\right)\end{aligned} \tag{5.16}$$

At the TLS solution (minimum), eq. (1.18) holds:

$$[A;b]^T[A;b]\begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} = \begin{bmatrix} A^TA & A^Tb \\ b^TA & b^Tb \end{bmatrix}\begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} = \sigma_{n+1}^2\begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} \tag{5.17}$$

Hence,

$$\Delta' \overset{t\to\infty}{\to} \left(\hat{x}^T\hat{x}\right)\sigma_{n+1}^2 - \hat{x}^T\left(\sigma_{n+1}^2\hat{x}\right) = 0 \tag{5.18}$$

showing that the $(n+1)$th component of the MCA EXIN ODE, in the limit, carries no information. ■

5.2.3 Analysis of the OLS EXIN ODE

From eq. (5.12), for $\zeta = 0$ (OLS),

$$\frac{dx}{dt} = -\overline{R}x - r \tag{5.19}$$

Its critical point (a minimum because the cost function is quadratic) is given by

$$x' = \overline{R}^{-1} r \tag{5.20}$$

($\overline{R}$ is invertible because it is positive definite). From eq. (5.14), $\overline{R} = A^T A/m$ and $r = A^T b/m$. Replacing these expressions in eq. (5.20) gives

$$x' = (A^T A)^{-1} A^T b = A^+ b$$

which is the closed-form OLS solution [see eq. (1.7)]. This confirms the validity of eq. (5.12) for $\zeta = 0$.

5.2.4 Convergence of DLS EXIN

In the data least squares problem, $\zeta = 1$. Thus, using the formulas of this section, the error function becomes

$$E_{\text{DLS}}(x) = \frac{1}{2} \frac{(Ax - b)^T (Ax - b)}{x^T x} \qquad \forall x \in \Re^n - \{0\} \tag{5.21}$$

The DLS EXIN learning law becomes

$$x(t+1) = x(t) - \alpha(t)\gamma(t) a_i + \left[\alpha(t)\gamma^2(t)\right] x(t) \tag{5.22}$$

where

$$\gamma(t) = \frac{\delta(t)}{x^T(t)x(t)} \tag{5.23}$$

and the corresponding nth-dimensional ODE is

$$\frac{dx}{dt} = \frac{1}{x^T x} \left\{ -\left(\overline{R}x - r\right) + \frac{\left(x^T \overline{R} x\right) x - 2\left(x^T r\right) x + \Gamma x}{x^T x} \right\} \tag{5.24}$$

Two questions arise:

1. Is the DLS error cost minimized by the DLS solution (1.55)?
2. Does the DLS EXIN learning law converge to the right solution?

The following theorem addresses these problems.

Theorem 97 (DLS EXIN Convergence) *The DLS EXIN learning law, expressing the minimization of the error cost* (*5.21*) *converges, under the usual stochastic approximation assumptions, to the DLS solution* [*i.e., eq.* (*1.55*)].

Proof. Express eq. (5.24) as

$$\frac{dx}{dt} = \frac{1}{\left(x^T x\right)^2} Q\left(x, \overline{R}, r, \Gamma\right) \tag{5.25}$$

where

$$Q\left(x, \overline{R}, r, \Gamma\right) = -\left(\overline{R}x - r\right) x^T x + \left(x^T \overline{R} x\right) x - 2\left(x^T r\right) x + \Gamma x \tag{5.26}$$

The critical points are given by $Q\left(x, \overline{R}, r, \Gamma\right) = 0$, which, under the transformation

$$x = \left(r^T v\right)^{-1} \Gamma v \tag{5.27}$$

becomes

$$Q\left(v, \overline{R}, r, \Gamma\right) = 0 \tag{5.28}$$

which implies that

$$\begin{aligned}
&\left(\overline{R}^T v\right)^{-3} \Gamma^3 \left(v^T v\right) \overline{R} v - \left(\overline{R}^T v\right)^{-2} \Gamma^2 \left(v^T v\right) \overline{R} \\
&\quad = \left(\overline{R}^T v\right)^{-3} \Gamma^3 \left(v^T \overline{R} v\right) v - 2\left(\overline{R}^T v\right)^{-2} \Gamma^2 \left(v^T \overline{R}\right) v + \left(\overline{R}^T \overline{R}\right)^{-1} \Gamma^2 v
\end{aligned}$$

After some algebra,

$$\left(\overline{R} - \frac{\overline{R}\overline{R}^T}{\Gamma}\right) v = \underbrace{\frac{v^T \left(\overline{R} - \overline{R}\overline{R}^T / \Gamma\right) v}{v^T v}}_{\overline{R}\text{ayleigh quotient}} v \tag{5.29}$$

Hence, the critical points v_c are the eigenvectors of the matrix $\overline{R} - \overline{R}\overline{R}^T / \Gamma$. Following the proof of Theorem 60, it is easy to show that in the transformed space, under the usual assumptions, the DLS EXIN learning law converges to the eigenvector $v_{\min}$ associated with the minimum eigenvalue. Using eq. (5.14), it holds that, on average,

$$\overline{R} - \frac{\overline{R}\overline{R}^T}{\Gamma} \to A^T A - \frac{\left(A^T b\right)\left(b^T A\right)}{b^T b} = A^T \left(I - b\left(b^T b\right)^{-1} b^T\right) A \tag{5.30}$$

But

$$A^T \left(I - b \left(b^T b \right)^{-1} b^T \right) A = \left\| \left(I - b \left(b^T b \right)^{-1} b^T \right) A \right\|_2^2 = \left\| P_b^{\perp} A \right\|_2^2 \tag{5.31}$$

So $v_{\min}$ is the right singular vector corresponding to the smallest singular value of the matrix $P_b^{\perp} A$, where $P_b^{\perp} = \mathrm{I} - b \left(b^T b \right)^{-1} b^T$ is a projection matrix that projects the column space of A into the orthogonal complement of b (see Theorem 37 and [51]). Reversing the transformation yields

$$x_{\min} = \left(r^T v \right)^{-1} \Gamma v_{\min} = \frac{b^T b}{b^T A v_{\min}} v_{\min} = x'' \tag{5.32}$$

This is the DLS solution (1.55). ■

This theorem confirms the validity of eq. (5.21) as the error function for a DLS neuron.

Remark 98 (DLS Null Initial Conditions) *As shown in eq. (5.21), the DLS error cost is not defined for null initial conditions. To allow this choice of initial conditions, which allows the neuron better properties and convergence, DLS scheduling is introduced (discussed in Section 5.5).*

5.2.5 Error Functions: A Summary

Solving the OLS problem requires minimization of the cost function:

$$E_{\mathrm{OLS}}(x) = \tfrac{1}{2} (Ax - b)^T (Ax - b) \tag{5.33}$$

The TLS solution minimizes the sum of the squares of the orthogonal distances (weighted squared residuals):

$$E_{\mathrm{TLS}}(x) = \frac{(Ax - b)^T (Ax - b)}{1 + x^T x} \tag{5.34}$$

DLS requires minimization of the cost function:

$$E_{\mathrm{DLS}}(x) = \frac{1}{2} \frac{(Ax - b)^T (Ax - b)}{x^T x} \tag{5.35}$$

These three error functions derive from the GeTLS error function (5.6) for the values $\zeta = 0$, 0.5, and 1, respectively.

5.2.6 GeTLS EXIN MADALINE

The GeTLS EXIN adaptive linear neuron (ADALINE) can also be applied to multidimensional problems $AX \approx B_{m \times d}$. In this case, d neurons

(MADALINE, i.e., many ADALINES; see [191]) solve each subproblem $Ax_i \approx b_i$, $i = 1, \ldots, d$ *separately*. The GeTLS EXIN MADALINE is made up of a single layer of d GeTLS EXIN neurons, each having the same input (row $a_i \in \Re^n$, $i = 1, \ldots, m$), a weight vector ($x_j \in \Re^n$, $j = 1, \ldots, d$) and a target ($b_{ij} \in \Re^{m \times d}$, $i = 1, \ldots, m^n$, $j = 1, \ldots, d$). At convergence, the solution is $\hat{X} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_d]$. The instantaneous cost function is

$$E^{(i)}(X) = \frac{1}{2} \sum_{j=1}^{d} \frac{\left(a_i^T x_j - b_{ij}\right)^2}{(1-\zeta) + \zeta \left\| x_j \right\|_2^2} \tag{5.36}$$

and the global cost function is

$$E_{\text{GeTLS EXIN MAD}}(x) = \sum_{i=1}^{m} E^{(i)}(x) \tag{5.37}$$

The GeTLS EXIN MADALINE discrete learning law is then

$$X(t+1) = X(t) - \alpha(t)\, a_i \kappa^T(t) + \zeta \alpha(t)\, X(t) \Omega(t) \tag{5.38}$$

where

$$\delta_j(t) = a_i^T x_j(t) - b_{ij} \tag{5.39}$$

$$\gamma_j(t) = \frac{\delta_j(t)}{(1-\zeta) + \zeta \left\| x_j(t) \right\|_2^2} \tag{5.40}$$

$$\kappa(t) = \left[\gamma_1(t), \gamma_2(t), \ldots, \gamma_d(t)\right]^T \in \Re^d \tag{5.41}$$

$$\Omega(t) = \text{diag}\ \left(\gamma_1^2(t), \gamma_2^2(t), \ldots, \gamma_d^2(t)\right) \in \Re^{d \times d} \tag{5.42}$$

This network has independent weight vectors. As seen in Remark 20, a network with *dependent* weight vectors is better, at least when all data are equally perturbed and all subproblems $Ax_i \approx b_i$ have the same degree of incompatibility.

5.3 GeTLS STABILITY ANALYSIS

5.3.1 GeTLS Cost Landscape: Geometric Approach

For convenience, the GeTLS EXIN cost function (5.6) is repeated here:

$$E_{\text{GeTLS EXIN}}(x) = \frac{1}{2} \frac{(Ax-b)^T (Ax-b)}{(1-\zeta) + \zeta x^T x}$$

Repeating the same reasoning as that of Section 4.7, analogously to eq. (4.48), search for the sets of vectors x solving the two-parameter equation in the n-dimensional vector x:

$$x^T A^T A x - 2b^T A x + b^T b - 2\gamma(1-\zeta) - 2\gamma\zeta x^T x = 0 \tag{5.43}$$

Similar to eq. (4.50), the center of the hyperconic x_c is given by

$$x_c(\gamma,\zeta) = \left[A^T A - 2\gamma\zeta I_n\right]^{-1} A^T b \tag{5.44}$$

which, using the EIV decomposition (4.52) of $A^T A$, becomes

$$x_c(\gamma,\zeta) = V'\left[\Lambda - 2\gamma\zeta I_n\right]^{-1} q \tag{5.45}$$

As in eq. (4.51), the following two-parameter equation in the n-dimensional vector y, without first-order terms, is obtained:

$$y^T\left[A^T A - 2\gamma\zeta I_n\right] y - b^T A\left[A^T A - 2\gamma\zeta I_n\right]^{-1} A^T b + b^T b - 2\gamma(1-\zeta) = 0 \tag{5.46}$$

By using eq. (4.52), eq. (5.46) becomes

$$\begin{aligned} &y^T\left[V'(\Lambda - 2\gamma\zeta I_n)V'T\right] y - b^T A V' \\ &\left[\Lambda - 2\gamma\zeta I_n\right]^{-1} V'^T A^T b + b^T b - 2\gamma(1-\zeta) = 0 \end{aligned} \tag{5.47}$$

By introducing

$$g(\gamma,\zeta) = b^T A V'\left[\Lambda - 2\gamma\zeta I_n\right]^{-1} V'^T A^T b - b^T b + 2\gamma(1-\zeta) \tag{5.48}$$

the equations corresponding to eqs. (4.56) and (4.58) are

$$z^T\left[\Lambda - 2\gamma\zeta I_n\right] z = g(\gamma,\zeta) \tag{5.49}$$

and

$$\begin{aligned} g(\gamma,\zeta) = &\sum_{i=1}^{n} \frac{q_i^2}{\lambda_i - 2\gamma\zeta} + 2\gamma(1-\zeta) - b^T b = q^T \\ &(\Lambda - 2\gamma\zeta I_n)^{-1} q + 2\gamma(1-\zeta) - b^T b \end{aligned} \tag{5.50}$$

respectively. Again, for any ζ, one and only one value $\gamma_{\min}$ of γ exists in the interval $(0, \lambda_n/2\zeta)$ such that $g(\gamma_{\min},\zeta) = 0$. This point corresponds to the unique minimum of the cost function $E_{\text{GeTLS EXIN}}(x)$, with position given by

$$x_{\text{GeTLS sol}} = x_c(\gamma_{\min},\zeta) = \left[A^T A - 2\gamma_{\min}\zeta I_n\right]^{-1} A^T b \tag{5.51}$$

Figure 4.15 is also valid for the GeTLS problem if $\sigma_i'^2$ is replaced by $\lambda_i/2\zeta$.

5.3.2 GeTLS Hyperconic Family

Recast the GeTLS hyperconic family (5.49) as

$$z^T \left[\Lambda - 2\gamma\zeta I_n\right] z = g\left(\gamma, \zeta\right) \rightarrow \sum_{i=1}^{n} \frac{z_i^2}{k_i} = 1 \tag{5.52}$$

where $k_i = g\left(\gamma, \zeta\right)/\lambda_i - 2\gamma\zeta$ and z_i is the ith component of the vector z and is the coordinate along the direction of the eigenvector v_i' associated with $\lambda_i = \sigma_i'^2$ [see eq. (1.2)], because of the rotation $z = V'^T y$. As seen in the last two sections, $\forall\gamma \in \left[0, \gamma_{\min}\right)$ the k_i's are all negative and the family is composed of imaginary ellipsoids.

5.3.2.1 The $\gamma_{\min} \leq \gamma \leq \lambda_n/2\zeta$ Case Here, $k_i > 0\ \forall i$, and therefore the family is composed of hyperellipsoids [$(n-1)$-dimensional ellipsoids] with axes parallel to the A eigenvectors. As γ increases, the hyperellipsoid becomes larger (for $\gamma = \gamma_{\min}$, its center, a unique point, represents the minimum point) and its semiaxis parallel to v_n' grows much more than the others, giving ellipsoids more and more stretched in this direction. The density of these equilevel hypersurfaces is proportional to the gap between $\gamma_{\min}$ and $\lambda_n/2\zeta$ (e.g., for a small gap, the cost landscape is flat). The infinity in the direction of v_n' has a value of $\lambda_n/2\zeta$ for the cost function:

$$\lim_{s \to \pm\infty} E_{\text{GeTLS EXIN}}\left(x_0 + s v_n'\right) = \frac{\lambda_n}{2\zeta} \qquad \forall x_0 \in \Re^n \tag{5.53}$$

because the hyperellipsoids tend to infinity as $\gamma \rightarrow \lambda_n/2\zeta^-$. For $\zeta = 0.5$ (TLS), $\gamma_{\min} = \sigma_{n+1}^2$.

5.3.2.2 The $\lambda_n/2\zeta \leq \gamma \leq \lambda_{n-1}/2\zeta$ Case Figure 4.15 shows the existence of one value of γ (γ^*), which is a zero for $g\left(\gamma, \zeta\right)$.

1. $\lambda_n/2\zeta \leq \gamma \leq \gamma^*$. Here, $k_n \geq 0$ and $k_i \leq 0\ \forall\, i \neq n$. The family is composed of two-sheeted hyperboloids (for $n \geq 2$) consisting of two parts located in the half-spaces $z_n \geq k_n$ and $z_n \leq -k_n$ (i.e., along the v_n' direction). The hyperplanes $z_n = \text{const}$ ($|z_n| > k_n$) intersect in $(n-2)$-dimensional ellipsoids, the remaining hyperplanes $z_i = \text{const}$ ($i = 1, \ldots, n-1$) along two-sheeted hyperboloids. If $\gamma = \gamma^*$, the hyperconic is made up of only one point. As $\gamma \rightarrow \lambda_n/2\zeta^+$, the hyperboloids tend to stretch on the v_n' direction.
2. $\gamma^* < \gamma \leq \lambda_{n-1}/2\zeta$. Here the k_i's change sign. The family is now composed of one-sheeted hyperboloids along the v_n' direction. All hyperplanes $z_n = \text{const}$ intersect it along $(n-2)$-dimensional ellipsoids, the remaining hyperplanes $z_i = \text{const}$ ($i = 1, \ldots, n-1$) along hyperboloids or cones. These one-sheeted hyperboloids tend to stretch on the v_{n-1}' direction as

$\gamma \to \lambda_{n-1}^{-}/2\zeta$. The infinity in the direction of v'_{n-1} has a value of $\lambda_{n-1}/2\zeta$ for the cost function:

$$\lim_{s\to\pm\infty} E_{\text{GeTLS EXIN}}\left(x_0 + s\,v'_{n-1}\right) = \frac{\lambda_{n-1}}{2\zeta} \qquad \forall x_0 \in \Re^n \tag{5.54}$$

because the hyperboloids tend to infinity as $\gamma \to \lambda_{n-1}^{-}/2\zeta$.

The density of these equilevel hypersurfaces is proportional to the gap between $\lambda_n/2\zeta$ and $\lambda_{n-1}/2\zeta$ (e.g., for a small gap, the cost landscape is flat). The point $\gamma = \gamma^*$ is very important from a geometric point of view because it represents the passage from a two-sheeted hyperboloid to the opposite one-sheeted hyperboloid. Hence, it represents a *saddle* point. Indeed, every positive k_i represents a minimum in the corresponding direction; so, for levels of the cost function under the point of level γ^*, only the v'_n direction is a minimum, the others being maxima. For levels above γ^*, all A eigenvector directions are minima, just attracting in the corresponding level range. For $\zeta = 0.5$ (TLS), $\gamma^* = \sigma_n^2$.

5.3.2.3 The $\lambda_i/2\zeta \le \gamma \le \lambda_{i-1}/2\zeta$ Case The same considerations of the last case cited can be made here. $g(\gamma, \zeta)$ is strictly increasing from $-\infty$ for $\gamma = \lambda_i^{+}/2\zeta$ to $+\infty$ for $\gamma = \lambda_{i-1}^{-}/2\zeta$ this implies a zero for g at $\gamma = \gamma^{**}$. For each level of the cost function, the family is composed of hyperboloids (for $n \ge 4$) with at least 2 positive and negative k_i's. Hence, they intersect each of the hyperplanes $z_i = \text{const.}$ along some hyperboloid of smaller dimensions (even one degenerating into a cone). For $\lambda_i/2\zeta \le \gamma \le \gamma^{**}$, the equilevel hyperboloids intersect the $(n - i + 1)$-dimensional plane

$$z_{i-1} = \text{const}_{i-1},\ z_{i-2} = \text{const}_{i-2}, \ldots,\ z_1 = \text{const}_1 \tag{5.55}$$

in a $(n - i)$-dimensional ellipsoid. Hence, the space spanned by $v'_n, v'_{n-1}, \ldots, v'_i$ is a minimum for the cost function. For $\gamma^{**} \le \gamma \le \lambda_{i-1}/2\zeta$, the k_i's change of sign and now the $(i - 1)$-dimensional complementary plane of eq. (5.55) is a minimum. Thus, $\gamma = \gamma^{**}$ represents the level of a saddle. For $\zeta = 0.5$ (TLS), $\gamma^{**} = \sigma_i^2$. The equilevel hyperboloids tend to stretch on the v'_i direction as $\gamma \to \lambda_i/2\zeta^{+}$ and tend to stretch on the v'_{i-1} direction as $\gamma \to \lambda_{i-1}^{-}/2\zeta n^{-}$. The density of these hypersurfaces is proportional to the gap between $\lambda_i/2\zeta$ and $\lambda_{i-1}/2\zeta$. The infinity in the direction of v'_{i-1} has a value of $\lambda_{i-1}/2\zeta$ for the cost function:

$$\lim_{s\to\pm\infty} E_{\text{GeTLS EXIN}}\left(x_0 + s\,v'_i\right) = \frac{\lambda_{i-1}}{2\zeta} \qquad \forall x_0 \in \Re^n \tag{5.56}$$

because the hyperboloids tend to infinity as $\gamma \to \lambda_{i-1}^{+}/2\zeta^{+}$.

5.3.2.4 The $\lambda_1/2\zeta \leq \gamma \leq \gamma_{\max}$ Case

$g(\gamma, \zeta)$ is strictly increasing from $-\infty$ for $\gamma = \lambda_1/2\zeta^+$ to $+\infty$ for $\gamma = +\infty$; this implies a zero for g at $\gamma = \gamma^{***}$. This zero necessarily represents the maximum level of the cost function ($\gamma^{***} = \gamma_{\max}$) because all k_i's have the same sign and only for $g(\gamma, \zeta) \leq 0$ are the hyperconics real hyperellipsoids [$(n-1)$-dimensional ellipsoids] with axes parallel to the A eigenvectors [$\forall \gamma \in (\gamma^{***}, \infty)$ the k_i's are all negative and the family is composed of imaginary ellipsoids]. As $\gamma \to \lambda_1/2\zeta^+$, the hyperellipsoids tend to stretch in the v_1' direction. As $\gamma \to \gamma_{\max}$, the hyperellipsoids tend to collapse into a point, but their shape spreads out. The density of these hypersurfaces is proportional to the gap between $\lambda_1/2\zeta$ and $\gamma_{\max}$. For $\zeta = 0.5$ (TLS), $\gamma^{***} = \sigma_1^2$.

5.3.3 Considerations and the Two-Dimensional Case

Theorem 99 (GTLS EXIN Critical Points) *The GeTLS EXIN cost function of an n-dimensional unknown has $n+1$ critical points: a minimum, $n-1$ saddles, and a maximum.*

Remark 100 (Link with the MCA EXIN Stability Analysis) *All the previous considerations about the TLS EXIN stability analysis and cost landscape are very similar to the corresponding considerations for MCA EXIN stability, in particular Section 2.5.1.2. This was expected because, in the MCA space, the only difference lies in the fact that the TLS EXIN problem takes into consideration only intersections with a hyperplane perpendicular to the last Cartesian coordinate (TLS hyperplane) instead of hyperplanes perpendicular to the $[A; b]$ eigenvectors.*

In the following, much insight will be dedicated to the two-dimensional case (i.e., $n = 2$) because it is easy to visualize. Figure 4.1 shows this general case in the TLS hyperplane. The following overdetermined system of linear equations in two variables, taken from [22, Ex. 3], will be the main benchmark for the GeTLS EXIN neuron from now on.

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix} \tag{5.57}$$

The exact results, obtained by using MATLAB, are

$$x' = [0.7, 0.3]^T \tag{5.58a}$$

$$\hat{x} = [0.8544, 0.2587]^T \tag{5.58b}$$

$$x'' = [1.1200, 0.1867]^T \tag{5.58c}$$

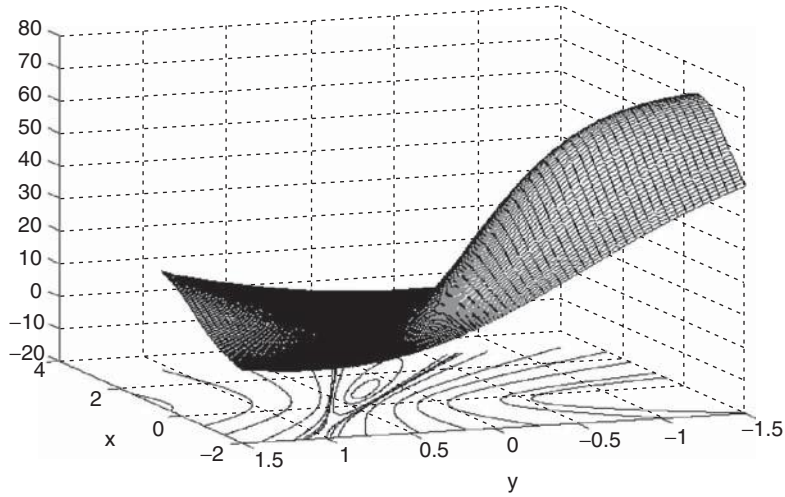


Figure 5.1 TLS cost landscape and corresponding equilevel curves for a benchmark overdetermined system of the two unknowns x and y.

where x', $\hat{x}$, and x'' are, respectively, the OLS, TLS, and DLS solution. The TLS cost landscape for this problem is given in Figure 5.1. Its critical points are

$$\min = [0.8544, 0.2587]^T \tag{5.59a}$$

$$\text{saddle} = [-1.4442, 0.9037]^T \tag{5.59b}$$

$$\max = [-0.5629; -2.0061]^T \tag{5.59c}$$

The equilevel curves are better visualized in Figure 5.2, which also shows the critical point locations and one section of the cost landscape along the straight line passing through the saddle and the minimum, which is a valley in the maximum direction (see Remark 93). Observe the barrier given by the saddle in this section: a gradient method with initial conditions on the left of the saddle cannot reach the minimum.

5.3.4 Analysis of the Convergence Keys

Theorem 101 (First Characterization) *The convergence keys are the components of the OLS solution $x' = A^+ b$ with respect to the unidimensional eigenspaces of $A^T A$, weighted by the corresponding eigenvalues.*

Proof. From the definition of the vector v_j as a right singular vector of A,

$$A^T A v_j' = \lambda_j v_j'$$

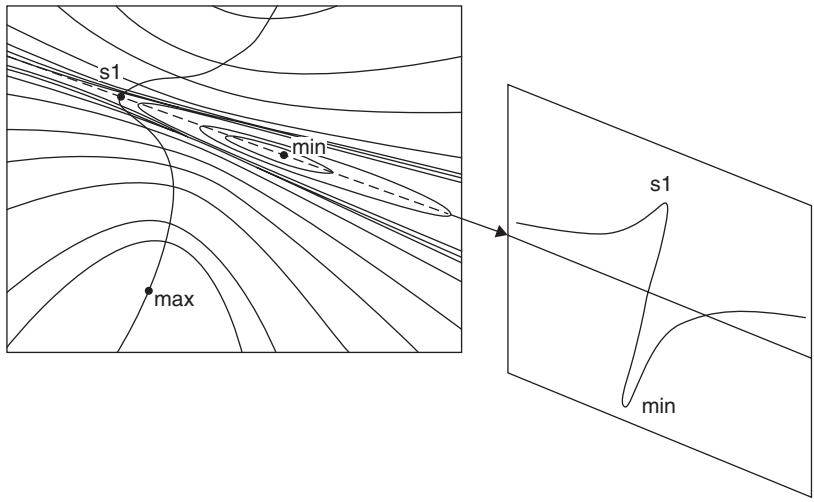


Figure 5.2 TLS equilevel curves for a benchmark overdetermined system with the corresponding critical points and the barrier of convergence, together with a section along the straight line passing through the minimum and the saddle.

the following deductions are found, assuming A^TA nonsingular:

$$v_j'^T A^T A = \lambda_j v_j'^T \Rightarrow v_j' T = \lambda_j v_j'^T \left(A^T A\right)^{-1} \Rightarrow v_j'^T A^T = \lambda_j v_j'^T \left(A^T A\right)^{-1} A^T$$

Recalling the definition of pseudoinverse of a matrix, we have

$$v_j'^T A^T = \lambda_j v_j'^T A^+ \tag{5.60}$$

Hence, from the definition of q_j,

$$q_j = \left(v_j'^T A^T\right) b = \lambda_j v_j' T A^+ b = \lambda_j \left[\left(A^+ b\right) \cdot v_j'\right] \tag{5.61}$$

where $\cdot$ denotes a scalar product. Recalling that $x_{\text{OLS}} = A^+ b$, it follows that

$$q_j = \lambda_j \left[x' \cdot v_j'\right] \tag{5.62}$$

that is, the convergence keys are the components of the OLS solution $x' = A^+ b$ with respect to the right singular vectors v_j' of A, weighted by the corresponding squared singular values λ_j. ■

Theorem 102 (Second Characterization) *The convergence keys are the components of $A^T b$ with respect to the unidimensional eigenspaces of $A^T A$.*

Proof. It follows directly from the definition of q:

$$q = V'^T A^T b \Rightarrow q_j = v_j'^T A^T b = \left(A^T b\right) \cdot v_j' \qquad \forall j = 1, \ldots, n \tag{5.63}$$

where $\cdot$ denotes a scalar product. ■

Corollary 103 (Characterization Equivalence) *The two convergence key characterizations are equivalent.*

Proof. Using the SVD of A, the pseudoinverse A^+ can be expressed as

$$A^+ = \sum_j \frac{1}{\sigma_j'} v_j' u_j'^T \tag{5.64}$$

Hence, the OLS solution can be expressed as

$$x' = A^+ b = \sum_j \frac{1}{\sigma_j'} v_j' u_j'^T b \tag{5.65}$$

From eq. (5.62),

$$q_j = \lambda_j \left[x' \cdot v_j'\right] = \lambda_j \left[\sum_k \frac{1}{\sigma_k'} v_k' u_k'^T b \cdot v_j'\right] = \lambda_j \frac{1}{\sigma_j'} u_j'^T b = \sigma_j' u_j'^T b \tag{5.66}$$

From the definition of q,

$$q_j = v_j'^T A^T b = \sigma_j' u_j'^T b \tag{5.67}$$

Then eq. (5.66) is equivalent to eq. (5.67). ■

Proposition 104 (Convergence Key Property) *The following property for the convergence keys holds:*

$$b^T b \geq \sum_{i=1}^{n} \frac{q_i^2}{\lambda_i} \tag{5.68}$$

($\lambda_i = \sigma_i'^2$), where the sign $>$ is valid for overdetermined systems.

Proof. Recalling eq. (5.67), the expression on the right-hand side of the inequality (5.68) becomes

$$\sum_{i=1}^{n} \frac{q_i^2}{\lambda_i} = \sum_{i=1}^{n} \frac{\left(\sigma_i' u_i'^T b\right)^2}{\sigma_i'^2} = \sum_{i=1}^{n} \left(u_i'^T b\right)^2 \tag{5.69}$$

where $b, u_i' \in \Re^m$. Considering that the basis of the m unit vectors u_i' spans the entire $\Re^m$ vector space but in this last expression only the n unit vectors u_i' corresponding to the nonnull singular values σ_i' are to be taken into account, it follows that for $m > n$, $\sum_{i=1}^{n} \left(u_i'^T b\right)^2$ expresses the squared norm of the projection of the vector b onto the subspace spanned by n unit vectors u_i'. Hence, this squared norm is less than the squared norm of b except when $m = n$, where this expression represents the squared norm of b, so eq. (5.68) is valid with the sign equal. ■

5.3.5 Approximated Analysis of the Relative GeTLS Solution Positions

The GeTLS solutions comprise solutions of the GeTLS problems $\forall \zeta \in [0, 1]$, in particular the OLS, TLS, and DLS solutions. Here, the basic assumptions are:

1. An overdetermined system of only slightly incompatible equations (i.e., $\gamma_{\min} \ll 1$).
2. In the expression of $g(\gamma, \zeta)$, the nth term of $\sum_{i=1}^{n} q_i^2/\lambda_i - 2\gamma\zeta$ is dominant (i.e., λ_n and λ_{n-1} are enough distant) for $\gamma < \lambda_n/2\zeta$.

With these assumptions, $g(\gamma, \zeta)$ becomes

$$g(\gamma, \zeta) \approx \frac{q_n^2}{\lambda_n - 2\gamma\zeta} + 2\gamma(1 - \zeta) - b^T b \tag{5.70}$$

Its zero for $\gamma < \lambda_n/2\zeta$ gives the solution (minimum) for the corresponding GeTLS problem. Hence, the GeTLS solution solves the approximated second-degree equation

$$4\zeta(1 - \zeta)\gamma^2 - 2\left[\zeta b^T b + (1 - \zeta)\lambda_n\right]\gamma + \lambda_n b^T b - q_n^2 \approx 0 \tag{5.71}$$

which, by using the first assumption, can be simplified and gives the approximated solution:

$$\gamma_{\min} \approx \frac{1}{2}\frac{\lambda_n b^T b - q_n^2}{\zeta b^T b + (1 - \zeta)\lambda_n} = \frac{1}{2}\frac{k}{\zeta} b^T b + (1 - \zeta)\lambda_n \tag{5.72}$$

where $k \geq 0$ for the property (5.68). The three main solutions are

$$\gamma_{\min} = \begin{cases} \gamma_{\text{OLS}} \approx \dfrac{k}{2\lambda_n} & (\zeta = 0) \\ \gamma_{\text{TLS}} \approx \dfrac{k}{b^T b + \lambda_n} & (\zeta = 0.5) \\ \gamma_{\text{DLS}} \approx \dfrac{k}{2b^T b} & (\zeta = 1) \end{cases} \tag{5.73}$$

Introducing the parameter

$$t = 2\zeta\gamma \tag{5.74}$$

which will be shown later to be very important, it follows that

$$t_{\mathrm{DLS}} \geq t_{\mathrm{TLS}} \geq t_{\mathrm{OLS}} = 0 \tag{5.75}$$

where the equal sign is valid only for compatible systems ($\gamma_{\min} = 0$). In the x reference system, the position of the solution is given by eq. (5.45) for $\gamma = \gamma_{\min}$:

$$x_{\mathrm{sol}}\left(\gamma_{\min}, \zeta\right) = V'\left[\Lambda - 2\gamma_{\min}\zeta I_n\right]^{-1} q \tag{5.76}$$

which in the rotated z reference system (no translation) becomes

$$z_{\mathrm{sol}}\left(\gamma_{\min}, \zeta\right) = \left[\Lambda - 2\gamma_{\min}\zeta I_n\right]^{-1} q \tag{5.77}$$

Thus,

$$z_{\mathrm{sol}\,n} \approx \begin{cases} \dfrac{q_n}{\lambda_n} & \left(z'_n\right) \\ \dfrac{q_n}{\lambda_n - \gamma_{\mathrm{TLS}}} & \left(\hat{z}_n\right) \\ \dfrac{b^T b}{q_n} & \left(z''_n\right) \end{cases} \tag{5.78}$$

where $\gamma_{\mathrm{TLS}} = k/b^T b + \lambda_n > 0$ and therefore $|\hat{z}_n| \geq \left|z'_n\right|$.

Proposition 105 (Relative Positions) $\forall i = 1, \ldots, n,$

$$\left|z''_i\right| \geq |\hat{z}_i| \geq \left|z'_i\right| \tag{5.79}$$

which implies, in the x reference system, the same inequalities for the modules:

$$\left\|x''\right\|_2 \geq \|\hat{x}\|_2 \geq \left\|x'\right\|_2 \tag{5.80}$$

where the equal sign is valid only for compatible systems.

Proof. From eq. (5.78),

$$\hat{z}_n \approx \frac{q_n}{\lambda_n - \dfrac{1}{2}\dfrac{\lambda_n b^T b - q_n^2}{b^T b + \lambda_n}} = H q_n \tag{5.81}$$

where

$$H = 2\frac{b^T b + \lambda_n}{\lambda_n b^T b + 2\lambda_n^2 + q_n^2} \tag{5.82}$$

Recast z_n'' as

$$z_n'' \approx Gq_n \tag{5.83}$$

where

$$G = \frac{b^T b}{q_n^2} \tag{5.84}$$

It follows that

$$G > H \tag{5.85}$$

Indeed, using property (5.68),

$$H = 2\frac{b^T b + \lambda_n}{\lambda_n b^T b + 2\lambda_n^2 + q_n^2} < \frac{b^T b + \lambda_n}{q_n^2 + \lambda_n^2} < \frac{b^T b}{q_n^2} = G \tag{5.86}$$

Hence, $\left|z_n''\right| > |\hat{z}_n|$.

It is easy to extend the inequalities to the other z components. Indeed, $\forall i = 1, \ldots, n$,

$$z_{\text{sol}\,i}\,(\gamma_{\min}, \zeta) = \frac{q_i}{\lambda_i - 2\gamma_{\min}\zeta I_n} \approx \begin{cases} \dfrac{q_i}{\lambda_i} & \left(z_i'\right) \\ \dfrac{q_i}{\lambda_i - \lambda_n + H^{-1}} & (\hat{z}_i) \\ \dfrac{q_i}{\lambda_i - \lambda_n + G^{-1}} & \left(z_i''\right) \end{cases} \tag{5.87}$$

Thus:

- $\left|z_i''\right| > \left|z_i'\right|$ because $\lambda_i - \lambda_n + G^{-1} < \lambda_i$ [$G^{-1} < \lambda_n$ for the property (5.68)].
- $\left|z_i''\right| > |\hat{z}_i|$ because $G^{-1} < H^{-1}$.
- $|\hat{z}_i| > \left|z_i'\right|$ because $\lambda_i - \lambda_n + H^{-1} < \lambda_i$. Indeed, $|\hat{z}_n| > \left|z_n'\right| \Rightarrow H^{-1} < \lambda_n$. These properties, a fortiori, justify eq. (5.80). ■

5.3.6 Existence of the OLS Solution and the Anisotropy of the OLS Energy

A direct consequence of property (5.68) is the existence of the OLS solution. Indeed, from eq. (5.50),

$$g\,(\gamma_{\min}, 0) = \sum_{i=1}^{n} \frac{q_i^2}{\lambda_i} + 2\gamma_{\min} - b^T b = 0 \tag{5.88}$$

and the zero (energy height at the minimum) is nonnegative iff $b^T b \geq \sum_{i=1}^{n} q_i^2/\lambda_i$ (i.e., the above-mentioned property).

For $\gamma > \gamma_{\min}$, the equilevel family is only composed of hyperellipsoids [$(n-1)$-dimensional ellipsoids] with axes parallel to the A eigenvectors, the same center,

$$x_c = x' = V'\Lambda^{-1}q \tag{5.89}$$

and analytical expression

$$z^T \Lambda z = 2(\gamma - \gamma_{\min}) \tag{5.90}$$

Hence, the squared semiaxes k_i's [see eq. (5.52)] are given by

$$k_i = 2\frac{\gamma - \gamma_{\min}}{\lambda_i} \tag{5.91}$$

From the point of view of a neural algorithm in the form of a gradient flow, it is better to have isotropy of the hyperellipsoids just to avoid preferred unknown directions. It implies a clustering of the squared singular values of A. Too distant λ_i's, as in the case of an ill-conditioned matrix ($\lambda_1 \gg \lambda_n$), give strong hyperellipsoid anisotropy.

5.3.7 Barrier of Convergence

5.3.7.1 Two-Dimensional Case For $\lambda_2/2\zeta \leq \gamma \leq \lambda_1/2\zeta$, the family of equilevel curves for the cost function is made up of hyperbolas with focal axes parallel to the v_2' direction for heights less than the saddle height and of hyperbolas with focal axes parallel to the v_1' direction for heights more than the saddle height. This last subfamily is important from the point of view of the convergence domain of the steepest descent learning law. In fact, the locus of the intersections of the hyperbolas with the corresponding focal axes gives the crest for the cost function (i.e., the barrier for the gradient method). Note that this analysis does not take into account the influence of the maximum on the crest, which is valid for the inferior crest branch (i.e., the branch toward the maximum). Therefore, the following analytical considerations are exact only for the superior crest branch and for the part of the inferior crest branch nearer the saddle point. Figure 5.3 shows the benchmark problem (5.57).

For $\gamma_{\text{saddle}} \leq \gamma \leq \lambda_1/2\zeta$ the hyperbolas have the analytical expression [see eq. (5.52)]

$$\sum_{i=1}^{2} \frac{z_i^2}{k_i} = \frac{z_1^2}{k_1} + \frac{z_2^2}{k_2} = 1 \tag{5.92}$$

where $k_i = g(\gamma, \zeta)/\lambda_i - 2\gamma\zeta$ ($k_2 < 0$, $k_1 > 0$). The intersection with the z_1 axis is given by $\sqrt{k_1}$. Recalling that

$$x = V'z + x_c \tag{5.93}$$

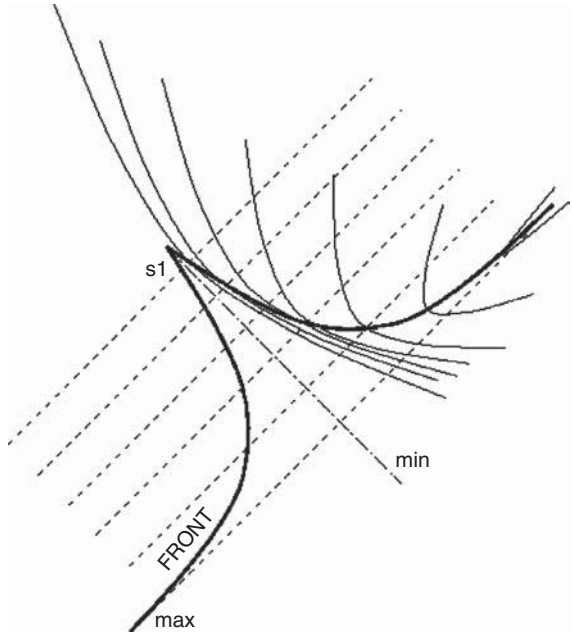


Figure 5.3 Barrier (frontier) for a two-dimensional benchmark; the focal axes of the equilevel hyperbolas are evidentiated.

where $x_c(\gamma)$ is the center of the corresponding equilevel curve, the barrier locus is given by the two-dimensional vector

$$x_{\text{bar}} = \pm\sqrt{\frac{g\,(\gamma,\zeta)}{\lambda_1 - 2\gamma\zeta}}\,v_1' + x_c(\gamma,\zeta) \tag{5.94}$$

where v_1' is the first column of V' (i.e., the principal component of A^TA). Hence, the barrier follows the v_1' direction. The subfamily of hyperbolas for $\lambda_2/2\zeta \leq \gamma \leq \gamma_{\text{saddle}}$ is not relevant from the convergence point of view because it will be demonstrated in the following that the relative position between the saddle and the minimum implies that only the first barrier is valid. This analysis is valid for every gradient flow of the energy cost considered.

5.3.7.2 Higher-Dimensional Case For $\lambda_{j+1}/2\zeta \leq \gamma \leq \lambda_j/2\zeta$, the family is composed of two subfamilies of hyperboloids according to their level with respect to the saddle level. Extending these reasonings, focus on the subfamily for $\gamma_{\text{saddle}} \leq \gamma \leq \lambda_j/2\zeta$, made up of hypersurfaces of the form

$$\sum_{i=1}^{n} \frac{z_i^2}{k_i} = 1 \tag{5.95}$$

where $k_i = g(\gamma, \zeta)/(\lambda_i - 2\gamma\zeta)$ ($k_i < 0$ only for $i > j$).

$$k_i = \begin{cases} \dfrac{q_j^2}{(\lambda_j - 2\gamma\zeta)(\lambda_i - 2\gamma\zeta)} + \displaystyle\sum_{k\neq j} \frac{q_k^2}{(\lambda_k - 2\gamma\zeta)(\lambda_i - 2\gamma\zeta)} \\ \quad + 2\dfrac{\gamma(1-\zeta)}{\lambda_i - 2\gamma\zeta} - \dfrac{b^T b}{\lambda_i - 2\gamma\zeta} & \text{for } i \neq j \\ \dfrac{q_j^2}{(\lambda_j - 2\gamma\zeta)^2} + \displaystyle\sum_{k\neq j} \frac{q_k^2}{(\lambda_k - 2\gamma\zeta)(\lambda_j - 2\gamma\zeta)} \\ \quad + 2\dfrac{\gamma(1-\zeta)}{\lambda_j - 2\gamma\zeta} - \dfrac{b^T b}{\lambda_j - 2\gamma\zeta} & \text{for } i = j \end{cases} \tag{5.96}$$

For $\gamma \to \lambda_j/2\zeta^-$,

$$k_i \to \begin{cases} o(k_j) & \text{for } i \neq j \\ \dfrac{q_j^2}{(\lambda_j - 2\gamma\zeta)^2} & \text{for } i = j \end{cases} \tag{5.97}$$

Hence, the asymptotic form of the hyperboloid is

$$\frac{z_j^2}{k_j} \approx 1 \tag{5.98}$$

Recalling eq. (5.93), the barrier locus is given by

$$x_{\text{bar}} \approx \pm\sqrt{k_j}\, v_j' + x_c(\gamma, \zeta) = \pm\frac{q_j}{\lambda_j - 2\gamma\zeta} v_j' + x_c(\gamma, \zeta) \tag{5.99}$$

where v_j' is the jth column of V'. Consider the expression (5.45) for $x_c(\gamma, \zeta)$ in the limit

$$x_c(\gamma, \zeta) \overset{\gamma\to\lambda_j/2\zeta^-}{\longrightarrow} \frac{1}{\lambda_j - 2\gamma\zeta} v_j' v_j'^T A^T b = \frac{1}{\lambda_j - 2\gamma\zeta} \text{proj}_{v_j'}(A^T b) \tag{5.100}$$

where $v_j' v_j'^T$ is the n-dimensional projection matrix defining the vector parallel to v_j' with norm given by the orthogonal projection onto the line of orientation v_j' passing through the origin. Recalling Theorem 102, we have

$$x_c(\gamma, \zeta) \overset{\gamma\to\lambda_j/2\zeta^-}{\longrightarrow} \frac{q_j v_j'}{\lambda_j - 2\gamma\zeta} \tag{5.101}$$

Thus, the general form of the *barrier locus* is given by

$$x_{\text{bar}} \approx \pm \frac{2}{\lambda_j - 2\gamma\zeta} q_j v'_j \tag{5.102}$$

The same considerations of the two-dimensional case can be applied here. In particular, the following theorem results.

Theorem 106 (Barrier Asymptotic Behavior) *The GeTLS convergence barrier asymptotically follows the z directions* (*i.e., the directions of the v'_j eigenvectors for $j \neq n$*).

5.3.8 Critical Loci: Center Trajectories

Equation (5.45) represents the locus of the centers of the equilevel hypersurfaces in the x reference system; in the z reference system, rotated by the matrix V' around the origin, this equation becomes

$$z_c(\gamma, \zeta) = \left[\Lambda - 2\gamma\zeta I_n\right]^{-1} q \tag{5.103}$$

that is, for each component ($i = 1, \ldots, n$),

$$z_{ci} = \frac{q_i}{\lambda_i - 2\gamma\zeta} \tag{5.104}$$

In each plane $z_j z_i$ this locus is an equilateral hyperbola translated from the origin:

$$z_i = \frac{q_i z_j}{(\lambda_i - \lambda_j) z_j + q_j} \tag{5.105}$$

All these loci pass through the z origin, which coincides with the x origin. Its asymptotes are given by

$$z_j = -\frac{q_j}{\Delta\lambda} \quad \text{and} \quad z_i = \frac{q_i}{\Delta\lambda} \tag{5.106}$$

where $\Delta\lambda = \lambda_i - \lambda_j$.

Definition 107 (Asymptotes) *The asymptote of the hyperbolas parallel to z_i* (*the coordinate along the direction of the eigenvector v'_i associated to $\lambda_i = \sigma'^2_i$*) *is defined as the z_i* asymptote.

Assuming distinct singular values for A, these loci tend to degenerate into two corresponding orthogonal straight lines for $q_j \to 0$ or $q_i \to 0$.

The global locus (5.104) is parameterized by the product $\gamma\zeta$. The parameter $t = 2\gamma\zeta$ will be used [see eq. (5.74)]. If a priori assumptions are not made, the

effects of γ and ζ cannot be separated. The hyperbolas pass through the origin for both $t = -\infty$ and $t = +\infty$. The following interpretations for the locus are possible:

1. If $\zeta = \text{const}$, the hyperbolas, which refer to couples of coordinates, represent the locus of the corresponding coordinates of the centers of the equilevel hypersurfaces for the energy cost of the GeTLS problem at hand.
2. If $\gamma = \text{const}$, the hyperbolas express the way the variability of ζ moves the centers of the equilevel hypersurfaces, just giving an idea of the difficulties of the various GeTLS problems.
3. If $\forall\zeta$ the zeros of $g(\gamma, \zeta)$ are computed, the hyperbola is divided into loci (*critical loci*), representing the critical points.

Among these interpretations, the third is the most interesting from the point of view of the study of the domain of convergence and for the introduction of the principle of the GeTLS scheduling. The critical loci (see Figure 5.4) are:

- The *solution locus* composed of the points having $t \leq \lambda_n$ and given $\forall\zeta$ by $2\zeta\gamma_{\min}$ [$\gamma_{\min}$ is the smallest zero of $g(\gamma, \zeta)$]. This locus extends itself to infinity only if one plane coordinate is z_n; if it is not the case, it represents the part of the hyperbola branch just until λ_n.
- The *saddle locus*,composed of the points having $\lambda_j \leq t \leq \lambda_{j-1}$ and given $\forall\zeta$ by $2\zeta\gamma_{\text{saddle}}$ [γ_{saddle} is the zero of $g(\gamma, \zeta)$ in the corresponding interval]; it represents this saddle for every GeTLS problem. It is represented by an entire branch of the hyperbola only in the plane $z_j z_{j-1}$; in all other planes it represents the part of the branch between $t = \lambda_j$ and $t = \lambda_{j-1}$.
- The *maximum locus,* composed of the points having $t \geq \lambda_1$ and given $\forall\zeta$ by $2\zeta\gamma_{\max}$ [$\gamma_{\max}$ is the largest zero of $g(\gamma, \zeta)$]. This locus extends itself to infinity only if one plane coordinate is z_1; if it is not the case, it represents the part of the hyperbola branch from λ_1 to ∞, corresponding to the origin. This point is attained only by the DLS maximum, because in this case the zero of $g(\gamma, 1)$ is at infinity; eq. (5.50) shows that $g(\gamma, 1)$ is monotonically increasing for $t > \lambda_1$ and tends to $-b^T b$ for $t \to \infty$.

The position in the curve for $t = 0$ (origin of the parametric curve) coincides with the OLS solution and is the only center for OLS equilevel hypersurfaces. The hyperbolas can be described as a function of this point. Indeed, from eqs. (5.104) and (5.89),

$$z_{ci} = \frac{\lambda_i}{\lambda_i - 2\gamma\zeta}\frac{q_i}{\lambda_i} = \frac{\lambda_i}{\lambda_i - 2\gamma\zeta} z_i' \tag{5.107}$$

Thus,

$$z_c(\gamma, \zeta) = \Gamma z' \tag{5.108}$$

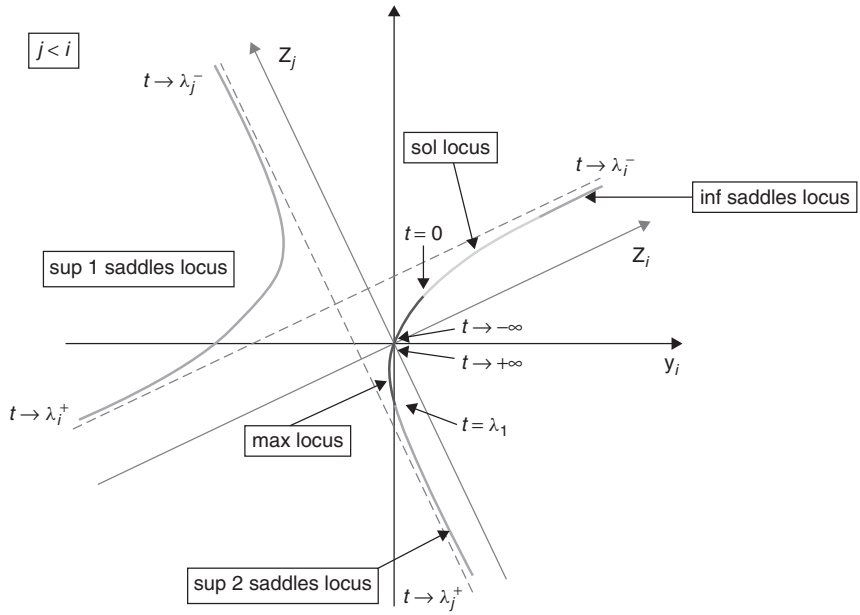


Figure 5.4 Critical loci in the plane $z_i z_j$. (*See insert for color representation of the figure.)*

where

$$\Gamma = \text{diag}\left(\frac{\lambda_1}{\lambda_1 - 2\gamma\zeta}, \frac{\lambda_2}{\lambda_2 - 2\gamma\zeta}, \ldots, \frac{\lambda_n}{\lambda_n - 2\gamma\zeta}\right)$$

has only positive elements in the solution locus. This confirms the reasoning that yields Proposition 105 (i.e., the GeTLS solutions follow the OLS solution in the locus and their distance on the curve is proportional to ζ). These two reasonings introduce the following theorem by explaining its first part.

Theorem 108 (Loci Relative Positions) *The value of the parameter t of a GeTLS solution (position in the solution locus with respect to the origin, which corresponds to the OLS solution) is proportional to the value of ζ. In particular, it means that $t_{DLS} \geq t_{TLS} \geq t_{OLS} = 0$. In every saddle locus and in the maximum locus, the critical points have a position t on the corresponding locus proportional to the parameter ζ of the GeTLS problem.*

Proof. The entire theorem can be demonstrated by the help of the graphical analysis of the zeros of $g(\gamma, \zeta)$. These zeros are given by the intersections between $h(t, \zeta)$ and $l(t, \zeta)$, where

$$h(t, \zeta) = b^T b + t(1 - \zeta^{-1}) \tag{5.109}$$

and

$$l(t,\zeta) = \sum_{i=1}^{n} \frac{q_i^2}{\lambda_i - t} \tag{5.110}$$

Note that $h(0,\zeta) \geq l(0,\zeta)$ [see inequality (5.68)]. Figure 5.6 illustrates the three-dimensional case. The straight line $h(t,0)$ (OLS) is vertical and has only a unique intersection at $t = 0$. The straight line $h(t,\zeta)$ rotates counterclockwise around $\left(0, b^T b\right)$ as ζ increases. As a consequence of the fact that $l(t,\zeta)$ is always strictly increasing, the saddles and maximum intersections have bigger and bigger parameters t as ζ increases.

The first part of the theorem is also demonstrated as follows.

The solution locus is given by the first zero (minimum) of $g(\gamma,\zeta)$ for every possible ζ:

$$g(\gamma_{\min},\zeta) = \sum_{i=1}^{n} \frac{q_i^2}{\lambda_i - 2\gamma_{\min}\zeta} + 2\gamma_{\min}(1-\zeta) - b^T b = 0 \tag{5.111}$$

For the points of the solution locus, $t_{\min} = 2\gamma_{\min}\zeta < \lambda_1$. Hence,

$$g(\gamma_{\min},\zeta) = \sum_{i=1}^{n} \frac{q_i^2}{\lambda_i}\left(1 + \frac{2\gamma_{\min}\zeta}{\lambda_i} + \cdots\right) + 2\gamma_{\min}(1-\zeta) - b^T b = 0 \tag{5.112}$$

Neglecting the higher-order terms yields

$$\gamma_{\min} = \frac{1}{2}\frac{b^T b - \sum_{i=1}^{n} q_i^2/\lambda_i}{1 + \left(\sum_{i=1}^{n}(q_i^2/\lambda_i) - 1\right)\zeta} = \gamma_{\min}(\zeta) \tag{5.113}$$

Correspondingly,

$$t_{\min} = \frac{A\zeta}{1 + (B-1)\zeta} \tag{5.114}$$

where $B = \sum_{i=1}^{n} q_i^2/\lambda_i > 0$ and $A = \frac{1}{2}\left(b^T b - \sum_{i=1}^{n} q_i^2/\lambda_i\right) \geq 0$ [see inequality (5.68)]. Figure 5.6 shows that the solution locus parameter $t_{\min}$ is always an increasing function for every GeTLS problem $\zeta \in [0,1]$. In particular:

- $\zeta = 0 \Rightarrow t_{\min} = 0$.
- $\zeta = 0.5 \Rightarrow t_{\min} = A/(B+1)$.
- $\zeta = 1 \Rightarrow t_{\min} = A/B$. ■

Hence, $t_{\mathrm{DLS}} \geq t_{\mathrm{TLS}} \geq t_{\mathrm{OLS}} = 0$.

Figure 5.5 also shows the DLS line (dashed), which is horizontal. This has two consequences:

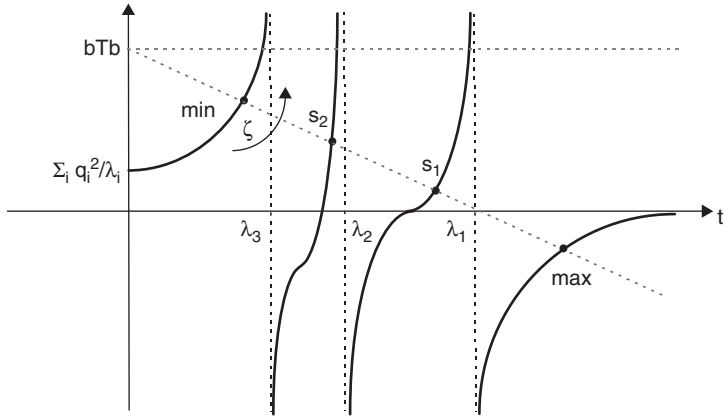


Figure 5.5 Graphical method for the determination of critical point parameters for a three-dimensional case; $h(t)$ is given by the nonvertical dashed straight line for two different GTLS problems (the horizontal line is the DLS); the solid curve is $l(t)$.

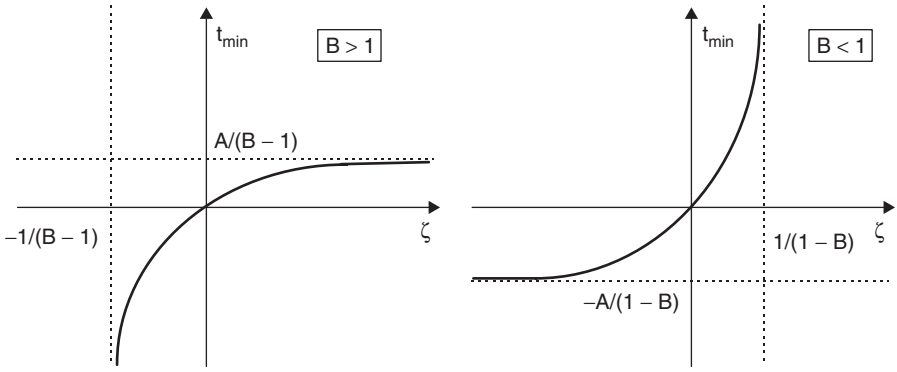


Figure 5.6 Plots of the solution locus parameter as a function of the type of GeTLS problem for all possible A and B.

1. The intersection for the maximum holds at $t = +\infty$; it agrees with the observation that the maximum for the DLS problem is at the TLS hyperplane origin.
2. The distance of a saddle to the asymptote at which its locus tends depends on $b^T b$.

The positions of the branches of the hyperbolas in the plane $z_i z_j$ (see Figures 5.7 and 5.8 for the two-dimensional case) depend on the signs of the corresponding q_i and q_j [i.e., the signs of the coordinates of the OLS solution z' (see Theorem 101) because $\lambda_i \geq 0 \; \forall \lambda_i$]. Hence, knowledge of the OLS solution indicates the quadrant containing the solution locus for every plane $z_i z_j$.

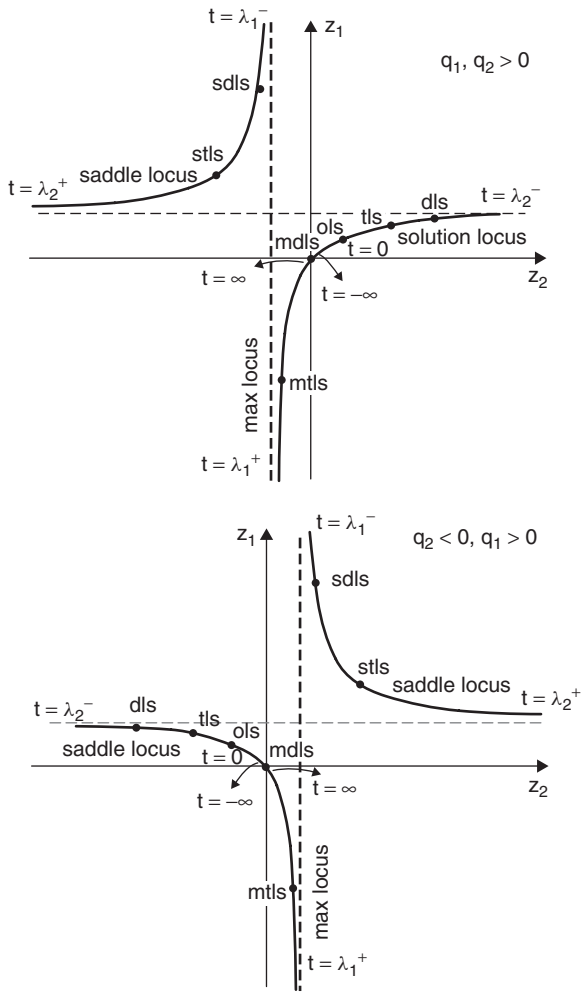


Figure 5.7 Two-dimensional case: hyperbolas and critical loci for $q_1 > 0$, $q_2 > 0$ and $q_1 > 0$, $q_2 < 0$.

Remark 109 (Scheduling First Step) *To solve a GeTLS problem, a possible technique is to begin with some iterations with $\zeta = 0$ (i.e., the corresponding OLS problem). This method places the weight vector in the solution locus, just avoiding entering the divergence zone.*

Remark 110 (An EVD Technique) *If the eigenvalue decomposition of $A^T A$ is known (i.e., the orthogonal matrix V'), the vector $q \equiv \begin{bmatrix} q_1 \, q_2 \, \cdots \, q_n \end{bmatrix}^T$ and the position of the z axes can be computed. Therefore, the position of the solution*

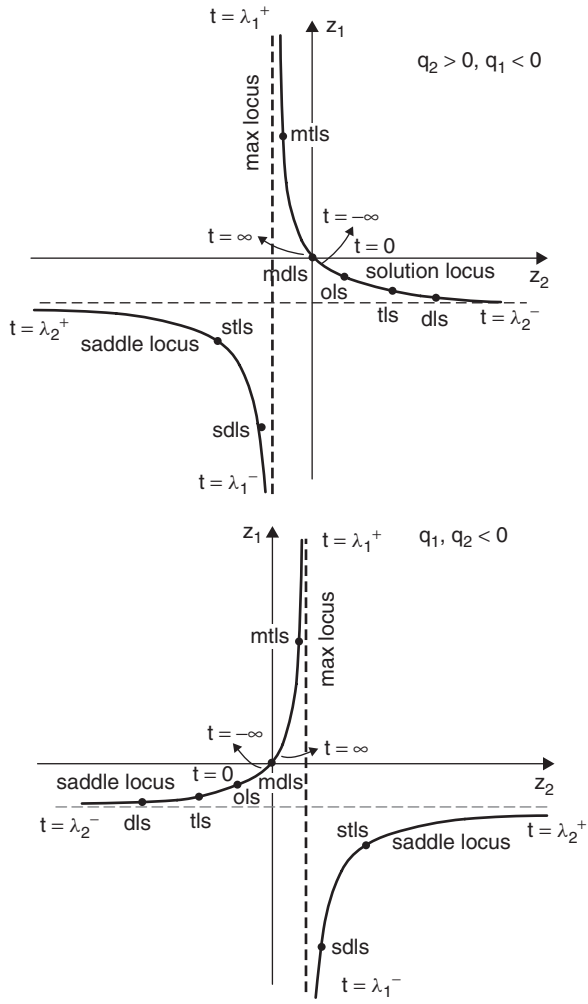


Figure 5.8 Two-dimensional case: hyperbolas and critical loci for $q_1 < 0$, $q_2 > 0$ and $q_1 < 0$, $q_2 < 0$.

locus can be determined, and this gives the best choice for the initial conditions for the GeTLS neuron. A direct consequence is knowledge of the exact position of the convergence barrier.

The origin of the x space, which coincides with the origin of the z space, has some very important features as a consequence of the existence of the hyperbolas:

1. In every parametric hyperbola, the origin corresponds to the point for $t = -\infty$; recalling that there are no zeros between this point and the GeTLS

solution, it follows that the origin is in the same branch of the solution locus, and no convergence barriers (caused by saddles) exist between the origin and this locus.

2. In every parametric hyperbola, the origin also corresponds to the point for $t = +\infty$; it follows that the origin is the superior limit of the maximum locus.

From these considerations the fundamental result for the choice of the initial conditions follows:

Theorem 111 (Fundamental for the Initial Conditions) *The originof the TLS hyperplane is the best choice for the initial conditions of a TLS neuron if no a priori information is given.*

Theorem 112 (Maximum Position) *The maximum is always nearer the minimum (GeTLS solution) than every saddle in every plane $z_n z_i$. In every plane $z_i z_j$ the maximum is nearer the minimum than all saddles corresponding to $t > \lambda_j$.*

The origin is always positioned between the maximum and the solution locus, and therefore, remarks about the maximum position are, a fortiori, even more valid for the origin. From the point of view of the convergence, the group of saddles for $t < \lambda_j$ causes no problems if the origin is chosen as an initial condition because if the learning rate is not too large, the weight vector approximately follows the solution locus (it coincides with the locus of the centers of the equilevel hyperellipsoids).

5.3.9 Domain of Convergence

Except for ζ, the TLS and GeTLS problems depend on the same parameters: the vector q and the $A^T A$ EVD:

- The positions of the asymptotes are proportional to the components of the vector q in the corresponding z hyperplanes [eq. (5.106)]. These asymptotes delimit the saddle loci and therefore give the positions of the barriers (recall the coincidence between the critical locus and the center locus). In particular, in every plane $z_n z_i$ the z_i asymptote is the border of the half-plane representing the domain of convergence. Therefore, increasing q_i enlarges the domain, thus having a positive effect. Resuming, the vicinity to the degeneration for the hyperbolas worsens the domain. In the TLS approach, the close-to-nongeneric condition, as will be seen later, implies, at least, $q_n \to 0$, just moving the vertical z_i asymptote toward the z_i axis in every $z_n z_i$ plane. This is the worst possible case for the domain of convergence; note that the origin is still in the domain, just on the barrier. It will be shown that the domain consists only of this barrier.

- In every $z_i z_j$ plane, the positions of the asymptotes are inversely proportional to the differences between the corresponding $A^T A$ eigenvalues λ_i and λ_j [eq. (5.106) . Hence, the positions of the barriers depend on the clustering of these eigenvalues (a big gap has a negative effect on the domain). In particular, in the $z_n z_1$ plane, the position of the barrier given by the first saddle (the highest) is inversely proportional to the distance between λ_1 and λ_n, and therefore an ill-conditioned data matrix A worsens the domain.

5.3.10 TLS Domain of Convergence

The TLS domain of convergence is affected by the action of the following borders:

1. *The barriers (asymptotes)*. This was shown in Section 5.3.7. Furthermore, in every plane $z_i z_j$ with $j < i$ there is a unique barrier for the v_i' direction, parallel to v_j', generated by the first saddle having parameter $t > \lambda_i$. Note that the solution locus stays on the hyperbola branch approaching the horizontal asymptote for $t \to \lambda_i^-$, and the saddle locus giving rise to the barrier begins from the opposite infinite point on the asymptote. Hence, the origin is always between the minimum and the barrier.
2. *The saddle cone projections in the TLS hyperplane*. These repulsive cone projections border that part of the frontier between the saddle and the corresponding barrier.
3. *The maximum locus tangent*. If the maximum is near the vertical asymptote, it corresponds closely to this asymptote. With respect to a vertical asymptote corresponding to a far eigenvalue λ_i ($\lambda_i \ll \lambda_1$), its effect is not negligible.
4. *The saddle-maximum hypercrest projection*. In the TLS hyperplane, the hyperplane representing the maxima in the minimum direction (see Section 2.5.1) projects to an $(n-1)$-dimensional plane which corresponds to a straight line in every plane $z_i z_j$.

The equation of the hypercrest in the $(n+1)$-dimensional space Ψ (MCA space, see Figure 4.1 for the three-dimensional case) is given by

$$v_{n+1}^T \Psi = \sum_{i=1}^{n+1} v_{i,n+1} \Psi_i = 0 \tag{5.115}$$

where v_{n+1} is the "smallest" right singular vector of $[A; b]$. In the TLS hyperplane, the minimum is given by the intersection with the straight line parallel to v_{n+1}:

$$x_{\min} = \hat{x} = \frac{-1}{v_{n+1,n+1}} \left[v_{1,n+1}, \ldots, v_{n,n+1} \right]^T \tag{5.116}$$

The intersection of the hypercrest with the TLS hyperplane is given by the solution of the following system:

$$\sum_{i=1}^{n+1} v_{i,n+1}\Psi_i = 0$$
$$\Psi_{n+1} = -1 \tag{5.117}$$

Hence, the equation of the $(n-1)$-dimensional plane in the TLS hyperplane (x space) is

$$\sum_{i=1}^{n} \frac{v_{i,n+1}}{v_{n+1,n+1}} x_i = 1 \tag{5.118}$$

which can be expressed as

$$\sum_{i=1}^{n} \frac{1}{-1/x_{i,\min}} x_i = 1 \tag{5.119}$$

Thus, the intercepts of this $(n-1)$-dimensional plane for every x_i axis of the TLS hyperplane are of opposite sign with respect to the x system origin. This demonstrates the following theorem.

Theorem 113 (Origin and Saddle-Maximum Barrier) *The origin of the TLS hyperplane is always between the saddle-maximum hypercrest projection and the TLS solution.*

Figure 5.9 shows the case for $n = 2$ and the change in the position of the projection as a function of the TLS minimum position. All the analysis of this section, Theorem 111, and all the observations about the TLS convergence demonstrate the following fundamental theorem.

Theorem 114 (Origin and TLS Domain of Convergence) *The TLS origin belongs to the TLS domain of convergence.*

As an example, Figures 5.10 and 5.11 show the domain of convergence of the generic TLS benchmark problem given by the system (5.57). Figure 5.10 illustrates the domain of convergence just showing a certain number of initial conditions for the sequential TLS EXIN neuron (it is also valid for every gradient flow of the TLS energy): The initial conditions for which the neuron converges are green. The dark blue straight line represents the asymptote/barrier z_1. The bottom part of the frontier is given by the asymptote, because the maximum is near to it. For increasing x_2, the frontier is given by the action of the saddle-maximum line and the repulsion area of the saddle. The latter is responsible for

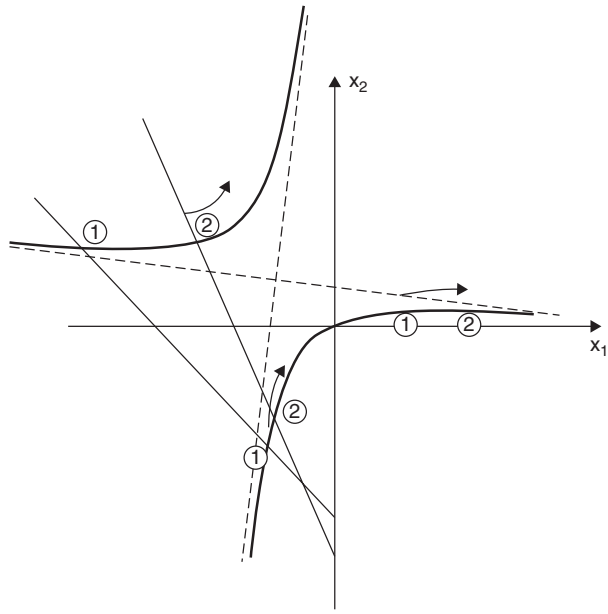


Figure 5.9 Saddle-maximum hypercrest projection and its evolution for two different TLS solutions (minima) in the two-dimensional TLS hyperplane.

the upper part until the top part, given by the asymptote. Figure 5.11 shows the corresponding weight vector evolution by means of the blue lines. All the trajectories, beginning from the green points, go to the straight line through the minimum and the saddle (TLS projection of the saddle-minimum hypervalley) at the right side of the saddle and then converge to the minimum. All the trajectories beginning from the red points go to the same straight line at the left side of the saddle and then diverge in the direction of the straight line.

5.3.11 GeTLS Domain of Convergence

The GeTLS domain of convergence is similar to the TLS domain of convergence (i.e., it is bordered by the same barriers and asymptotes and the same maximum locus tangent). The following remark explains the differences.

Remark 115 (GTLS Characterization) *Unlike the TLS problem, which can be extended to a higher-dimensional problem just to exploit the MCA, the GeTLS problem can be studied and solved only without a change in dimensionality because its formulation* [*eq.* (*5.6*)] *does not have a corresponding Rayleigh quotient.*

As a consequence, the MCA stability analysis cannot be applied. Then there are no hypercrest projections and the saddle cones are the volumes of attraction

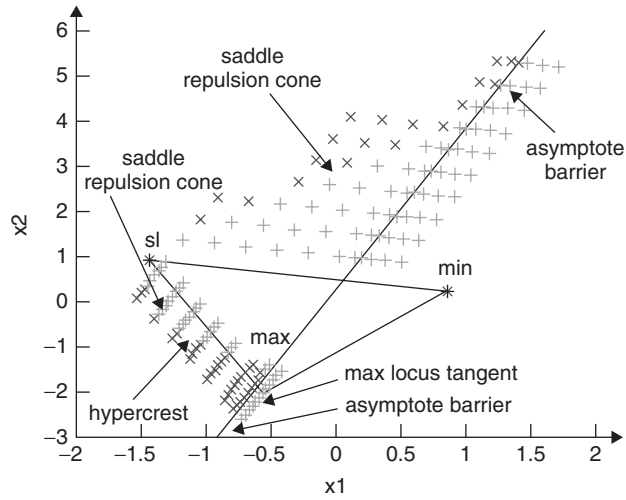


Figure 5.10 Initial conditions map for the generic TLS benchmark problem. Green represents the initial conditions by which a gradient flow algorithm converges (i.e., the domain of convergence). Red indicates divergence. The dark blue triangle has the critical points as vertices. The dark blue line represents the asymptote or barrier z_1. (*See insert for color representation of the figure*.)

and repulsion of its own critical points (the same shape as for the TLS case). In Chapter 6 the GeMCA theory is introduced: It transforms the GeTLS EXIN problem in a MCA EXIN problem, thus allowing us to use the MCA stability analysis.

For $\zeta = 0$ (OLS), the domain of convergence is the entire x space. To increase ζ, the domain of convergence becomes smaller and smaller. Indeed, in every saddle locus, the saddles tend to the next asymptote (t tends to the first superior eigenvalue) and this decreases the nearly triangular convergence area between the saddle and the corresponding barrier for every plane $z_n z_i$. In the maximum locus, the maximum is far from the vertical asymptote (for the DLS problem it coincides with the origin) and then the maximum locus tangent rotates toward the solution locus, thus decreasing the domain of convergence.

5.3.12 Skew Distances and GeTLS

Total least squares correspond to orthogonal regression because it minimizes the orthogonal distances (see Section 1.5.1 for TLS viewed from the row space and Figure 1.1). What is the meaning of the GeTLS EXIN error cost? Can we still speak of distances, perhaps skew, between the rows r_i of the augmented data matrix $[A; b]$ and the row space

$$R_r([A; b]) = \left\{ \begin{bmatrix} a \\ b \end{bmatrix} \middle| a \in \Re^n, b \in \Re,\ b = x^T a \right\}$$

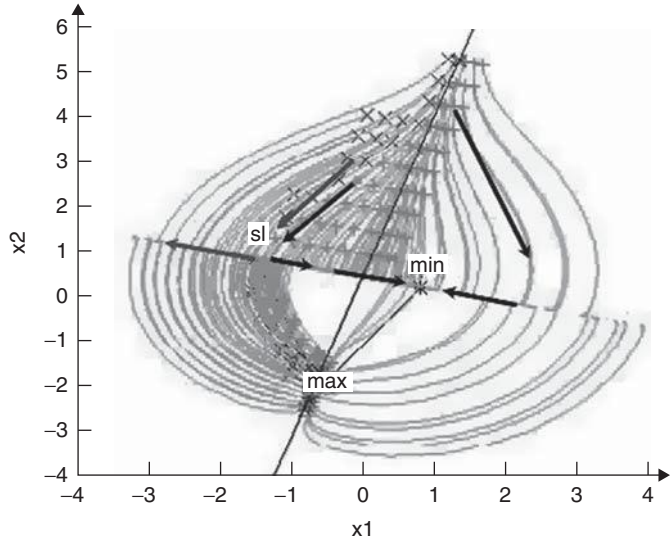


Figure 5.11 Temporal evolution of the sequential TLS EXIN neuron for the generic TLS benchmark problem. (*See insert for color representation of the figure*.)

or does this type of representation not exist? The answer is given by the following two theorems, which consider the problem from the reference of the orthogonal distance $d_\perp$ and the OLS distance d_{OLS}, respectively (see Figure 5.12).

Theorem 116 (Orthogonal Distance as Reference) *If the distances are expressed with respect to the orthogonal distances, the GeTLS error function* (*5.6*) *does not exactly represent a sum of squared skew distances, except for* $\zeta = 0.5$ (*TLS case*).

Proof. From Figure 5.12 it follows that

$$d_{\text{GeTLS}} = \frac{d_\perp}{\cos\alpha}$$

Hence, if it is assumed that the GeTLS error function (5.6) represents a sum of squared distances,

$$E_{\text{GeTLS}} = \sum d_{\text{GeTLS}}^2 = \sum \left(\frac{d_\perp}{\cos\alpha}\right)^2$$
$$= \frac{1}{\cos^2\alpha} E_{\text{TLS}} = \frac{1}{\cos^2\alpha} \frac{2E_{\text{OLS}}}{1+p}$$

where $p = x^T x$. Recalling eq. (5.6), we have

$$E_{\text{GeTLS}} = \frac{1}{2} \frac{2E_{\text{OLS}}}{(1-\zeta)+\zeta p}$$

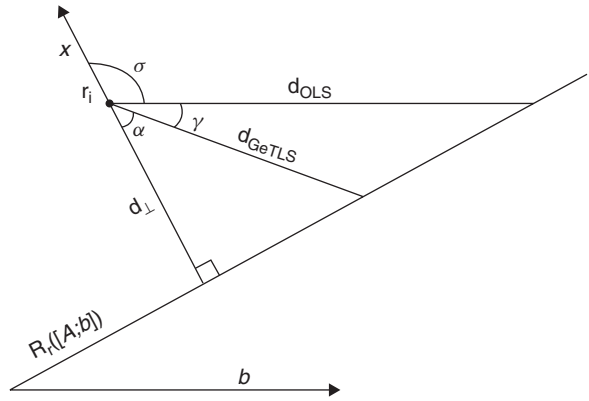


Figure 5.12 Geometry of the skew distance d_{GeTLS} in the row space.

It follows that

$$
\begin{aligned}
(1+p)\cos^2\alpha &= 2(1-\zeta)+2\zeta p = 2-2\zeta(1-p) \\
\cos^2\alpha &= \frac{2}{1+p}-\frac{1-p}{1+p}2\zeta \\
\sin^2\alpha &= 1-\cos^2\alpha = 1-\frac{2}{1+p}+\frac{1-p}{1+p}2\zeta \\
&= \frac{1-p}{1+p}(2\zeta-1)
\end{aligned}
\tag{5.120}
$$

which depends on p except for $\zeta = 0.5$, which implies that $\alpha = 0$. That is the TLS case, in which the orthogonal distances are used; indeed, we know that in this case α does not depend on p. ■

From eq. (5.120), other considerations can be drawn. From

$$\sin^2\alpha \overset{p\longrightarrow 0}{\longrightarrow} 2\zeta - 1 \qquad \text{with } \zeta \in [0.5, 1] \tag{5.121}$$

$$\sin^2\alpha \overset{p\longrightarrow \infty}{\longrightarrow} 1 - 2\zeta \qquad \text{with} \zeta \in [0, 0.5] \tag{5.122}$$

it can be deduced that during the first part of the transient, in the case of null initial conditions, p is close to 0, and for $\zeta \in [0.5, 1]$, as a first approximation, the GeTLS error cost represents a sum of skew distance whose slope depends on the value of ζ. The same is true for large p, but for $\zeta \in [0, 0.5]$. These two linear functions of ζ are visible in Figure 5.13 (they are the borders of the hypersurface), which plots eq. (5.120) together to the hyperplane $\sin^2\alpha = 0$, which is the lowest limit of the admissible space (the squared sine cannot be negative). Obviously, when $p = 1$, the distances are orthogonal. Figure 5.14 shows the derivative of

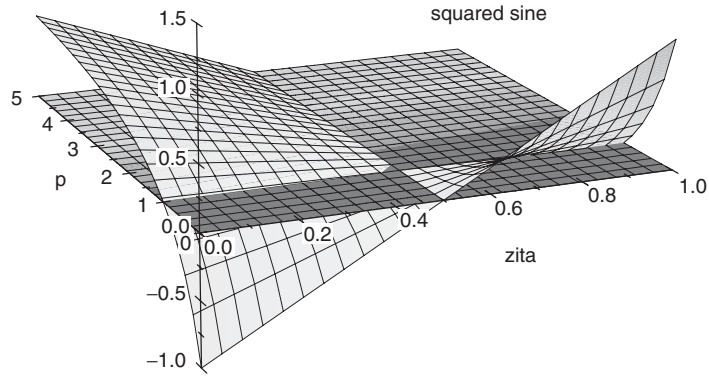


Figure 5.13 $\sin^2 \alpha$ as a function of p and ζ; its negative values determine the forbidden volume for the skew distance representation. (*See insert for color representation of the figure*.)

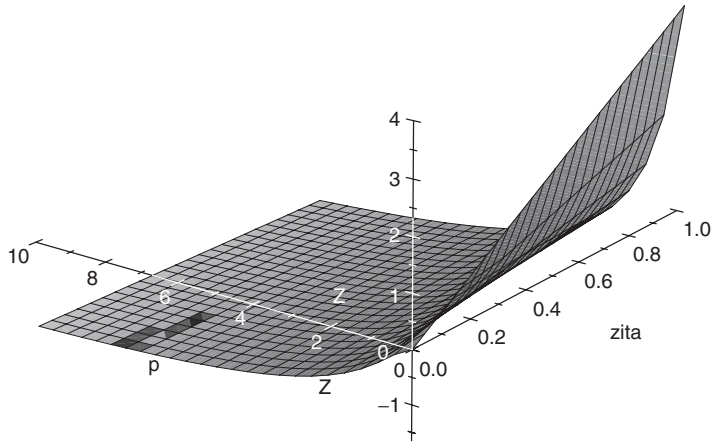


Figure 5.14 Plot of $\sin 2\alpha$. (*See insert for color representation of the figure*.)

eq. (5.120), which corresponds to $\sin 2\alpha$.It is very flat when p exceeds 1, which means that p does not greatly influence the value of α. Hence, in a first approximation, for large p, eq. (5.6) represents the sum of skew distances.

Theorem 117 (OLS Distance as Reference) *If the distances are expressed with respect to the OLS distances, the GeTLS error function (5.6) does not exactly represent a sum of squared skew distances, except for* $\zeta = 0$ *(OLS case).*

Proof. This proof is similar to the proof of Theorem 116 and uses the notation of Figure 5.12. Using the sine theorem, it follows that

$$\frac{\sin(\sigma - \pi/2)}{d_{\text{GeTLS}}} = \frac{\sin(\alpha + \pi/2)}{d_{\text{OLS}}}$$

where σ is the angle between x and its $(n+1)$th component (in the figure, vector b represents this last direction, which is parallel to the OLS residuals). Then

$$\frac{-\cos\sigma}{d_{\text{GeTLS}}} = \frac{\cos\alpha}{d_{\text{OLS}}}$$

Considering that $\alpha = \pi - (\sigma + \gamma)$, we obtain

$$\frac{\cos\sigma}{d_{\text{GeTLS}}} = \frac{\cos(\sigma+\gamma)}{d_{\text{OLS}}}$$

Hence, if it assumed that the GeTLS error function (5.6) represents a sum of squared distances,

$$\begin{aligned} E_{\text{GeTLS}} &= \sum d_{\text{GeTLS}}^2 = \sum d_{\text{OLS}}^2 \frac{\cos^2\sigma}{\cos^2(\sigma+\gamma)} \\ &= 2E_{\text{OLS}} \frac{\cos^2\sigma}{\cos^2(\sigma+\gamma)} = \frac{1}{2}\frac{2E_{\text{OLS}}}{(1-\zeta)+\zeta p} \end{aligned}$$

where $p = x^T x$. It follows that

$$\cos^2(\sigma+\gamma) = \cos^2\sigma\left[(1-\zeta)+\zeta p\right] \tag{5.123}$$

This formula does not depend on p, except for $\zeta = 0$, which implies that $\gamma = 0$ (OLS). ■

Summarizing: The GeTLS error function can be represented approximately as a sum of skew distances only with regard to an orthogonal regression framework.

5.4 NEURAL NONGENERIC UNIDIMENSIONAL TLS

In Section 1.7 we presented the theoretical basis of the nongeneric unidimensional TLS problem (i.e., when $\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}$, $p \le n$ and all $v_{n+1,i} = 0$, $i = p+1, \ldots, n+1$). From a geometric point of view, it means that the last $n-p+1$ critical points (the lowest) go to infinity [i.e., the corresponding zeros in $g(\gamma)$ coincide with the first $n-p+1$ asymptotes]. The saddle associated to σ_p^2 is the first critical point not coincident with an asymptote. Indeed,

$$\sigma'_{p-1} > \sigma_p > \sigma'_p = \sigma_{p+1} = \cdots = \sigma_{n+1} \tag{5.124}$$

(see property (1.32) and [(98, p. 75)]. This saddle is the nongeneric TLS solution, because its position in the TLS hyperplane [eq. (4.50)] corresponds to the closed-form solution (1.45). The singular value σ_{p+1} represents the degree of

incompatibility of the TLS problem $Ax \approx b$, indicating just how closely b is linearly related to A. A large σ_{p+1} means highly conflicting equations.

Remark 118 (Highly Conflicting Equations) *If $\sigma_{p+1} \to \sigma'_p$ (highly conflicting equations), the domain of convergence worsens because of the approach of the saddle to the corresponding asymptote (good equations have faraway saddles). Furthermore, the motor gap of the saddles is smaller, thus worsening the convergence speed of the iterative methods, and the maximum and the origin are much farther from the saddle/solution.*

Since $\sigma_{p+1} = \sigma'_p$ is the smallest singular value, it follows that A is nearly rank-deficient, or else the set of equations is highly incompatible. The various techniques devised to solve this problem are described in Section 1.7. In practice, close-to-nongeneric TLS problems are more common. In these cases the generic TLS solution can still be computed, but it is unstable and becomes very sensitive to data errors when σ'_p approaches σ_{p+1}. Identifying the problem as nongeneric stabilizes the solution, making it insensitive to data errors. Unfortunately, a numerical rank determination of a noisy matrix is required.

Remark 119 (No Observation Vector Identification) *If no matter which column is used as the right side, only an estimation of a linear relationship among the columns of $[A; b]$ is needed and $v_{n+1,n+1} = 0$, there is no need to solve the TLS problem as nongeneric. It suffices to replace b by any column a_i of A provided that $v_{i,n+1} \neq 0$.*

5.4.1 Analysis of Convergence for $p = n$

For $p = n$, the lowest critical point associated with σ_{n+1} goes to infinity in the direction of v'_n. Recalling (1.37), it holds that

$$v_{n+1} = \begin{bmatrix} \pm v'_n \\ 0 \end{bmatrix} \quad \text{and} \quad b \perp u'_n \tag{5.125}$$

The first property expresses the fact that the $(n+1)$-dimensional vector v_{n+1} is parallel to the TLS hyperplane, just in the direction of v'_n. From the second property, it follows that

$$q_n = v_n'^T A^T b = \sigma'_n u_n'^T b = 0 \tag{5.126}$$

which implies that for every hyperbola [eq. (5.104) with $\zeta = 0.5$

$$z_{cn} = \frac{q_n}{\lambda_i - \gamma} = \begin{cases} 0 & \text{nongeneric case} \\ \approx 0 & \text{close-to-nongeneric case} \end{cases} \tag{5.127}$$

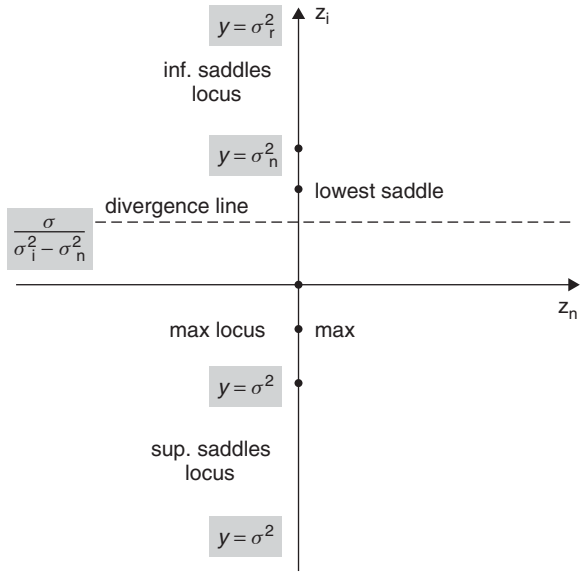


Figure 5.15 Solution and center locus in the plane $z_n z_i$ for a nongeneric TLS. The dotted line represents the divergence line.

Hence, the solution is contained in the hyperplane (*nongeneric TLS subspace*) through the origin and normal to the $A^T A$ eigenvector v_n'. It confirms the additional constraint $\begin{bmatrix} \hat{x} \\ -1 \end{bmatrix} \perp v_{n+1}$.Furthermore, in every plane $z_n z_i$ the hyperbolas degenerate in two straight lines given by $z_n = 0$ and $z_i = q_i/(\lambda_i - \lambda_n)$ (see Figure 5.15). In every plane $z_n z_i$ the latter straight line is the *divergence straight line* because it attracts all divergent weight trajectories. Indeed, it represents the equilevel surfaces center locus for $t < \lambda_n$.

Proposition 120 (Nongeneric TLS Subspace) *The nongeneric TLS subspace has dimension $n - 1$, n being the dimension of the TLS hyperplane. In this subspace, the lower saddle (i.e., the critical point corresponding to σ_n) loses its dimension of escape and then represents a minimum.*

The most important consequence of this proposition is the possibility of repeating exactly all the TLS stability analysis on this lower-dimensional subspace. Figure 5.15 shows that for every plane $z_n z_i$ the saddle/minimum is always closer to the asymptote than the other inferior ($t < \lambda_i$) saddles. It implies the following propositions.

Proposition 121 (Null Initial Conditions as Universal Choice) *The origin is always between the saddle/solution and the maximum. Hence, the choice of*

initial conditions for every TLS gradient flow, just like TLS EXIN, assures the convergence.

Proposition 122 (Divergence Straight Line) *In every plane $z_n z_i$, the divergence straight line is between the saddle/minimum and the maximum.*

5.4.2 Analysis of Convergence for $p < n$

In the case where $\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}$, $p \leq n$ (σ_{n+1} has multiplicity $n - p + 1$) and all $v_{n+1,i} = 0$, $i = p + 1, \ldots, n + 1$, there are $n - p + 1$ critical points going to infinity in the direction of the eigenvectors associated with the eigenvalues of the corresponding coincident asymptotes. Hence, there is an $n - p + 1$ *divergence plane* spanned by these eigenvectors. The complementary subspace (*nongeneric TLS subspace*) is the $(p - 1)$-dimensional plane through the origin and orthogonal to the divergence plane. All the reasonings about the case $p = n$ are also valid here, just recalling that in this case, the lower saddle (minimum) corresponds to σ_p. In particular, $q_p = q_{p+1} = \cdots = q_n = 0$. In the Ψ space (MCA vector space), the $(n + 1)$-dimensional vectors $v_{p+1}, \ldots, v_{n+1}$ are parallel to the TLS hyperplane.

5.4.3 Simulations

The first benchmark set of equations ([98, Ex. 3.1]) is

$$\begin{bmatrix} \sqrt{6} & \sqrt{6} \\ \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} 2 \\ 0 \\ -\sqrt{3} \end{bmatrix} \tag{5.128}$$

Here, $v_3 = [\sqrt{2}/2, -\sqrt{2}/2, 0]^T$. Note that $v_{n+1,n+1} = v_{3,3} = 0$. The solution (using the SVD) is

$$\hat{x} = \left[\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}} \right]^T \tag{5.129}$$

being $\left[\hat{x}^T; -1\right]^T \perp v_3$. All experiments with TLS EXIN and TLS GAO use a constant learning rate equal to 0.05. Figure 5.16 shows the phase diagram of TLS EXIN for different initial conditions: the red (black) initial conditions give convergent (divergent) trajectories. The black straight line passing through the red points contains the saddle/solution locus and coincides with the axis z_1. *The domain of convergence is given by the half-line with origin the maximum* $(-1.2247, -1.2247)$ *and containing the saddle/solution*. All the nonconverging trajectories tend to the divergence straight line (dark blue thick line), which is

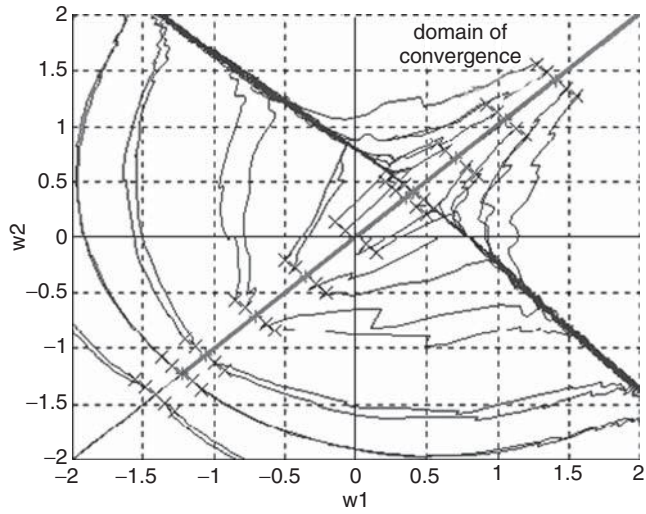


Figure 5.16 Evolution map of a sequential TLS EXIN for the first nongeneric TLS benchmark problem. The initial conditions for the converging trajectories are in red; the others are in black. The straight line passing through the red points contains the saddle/solution locus. The thick dark blue line is the divergence line. (*See insert for color representation of the figure*.)

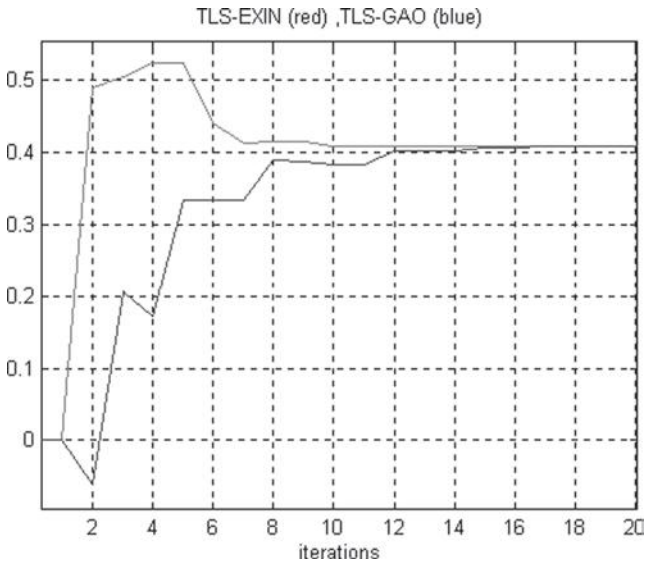


Figure 5.17 Plot of the weights of TLS EXIN and TLS GAO for the first nongeneric TLS benchmark problem. The initial conditions are null. The two learning laws stop at different iterations. (*See insert for color representation of the figure*.)

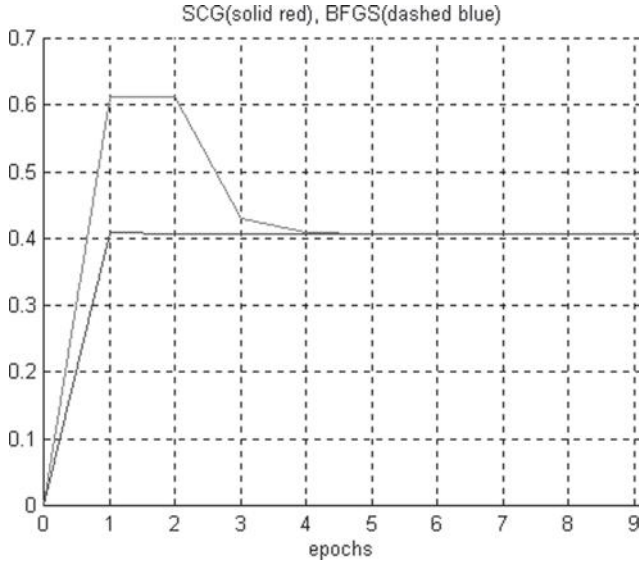


Figure 5.18 Plot of the weights of SCG and BFGS TLS EXIN for the first nongeneric TLS benchmark problem. The initial conditions are null. The two learning laws stop at different iterations. (*See insert for color representation of the figure*.)

the asymptote z_2 and then diverge in the v_2' direction. Figure 5.17 shows the dynamic behavior of TLS EXIN and TLS GAO for null initial conditions: The two weights w_1 and w_2 are always coincident because the weights trajectory is the bisector of the first quadrant. TLS EXIN is faster and more accurate. Figure 5.18 shows the behavior of SCG and BFGS TLS EXIN for null initial conditions (the weight components are coincident). They are faster than the other two neurons (1 epoch = 3 iterations) and, as usual, BFGS is best.

The second benchmark set of equations [93, Ex. 1] is

$$\begin{bmatrix} 1 & 0 \\ 0 & 10^{-4} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \tag{5.130}$$

Here, $v_3 = [0, -1, 0]^T$ and is parallel to the TLS hyperplane and to its axis x_2. Note that $v_{n+1,n+1} = v_{3,3} = 0$. The solution (using the SVD) is

$$\hat{x} = \left[\frac{2}{\sqrt{5} - 1}, 0 \right]^T \tag{5.131}$$

being $\left[\hat{x}^T; -1\right]^T \perp v_3$. SCG TLS EXIN and BFGS TLS EXIN, with null initial conditions, reach the solution (precision $= 10^{-15}$), in, respectively, six and four epochs. To explain problems regarding the close-to-nongeneric case, perturb

Table 5.1 Second nongeneric TLS benchmark: perturbed system

	x_1	x_2
SVD-based nongeneric TLS	$\frac{2}{\sqrt{5}-1}$	-2.618×10^{-8}
SCG TLS EXIN (six epochs)	$\frac{2}{\sqrt{5}-1} + 2.65 \times 10^{-6}$	2.7557×10^{-5}
BFGS TLS EXIN (four epochs)	$\frac{2}{\sqrt{5}-1} - 7.2 \times 10^{-8}$	2.4267×10^{-6}

the data matrix with $\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 10^{-8} \end{bmatrix}$ and the observation vector with $\begin{bmatrix} 0 \\ 10^{-8} \\ 0 \end{bmatrix}$. The SVD of the perturbed set of equations yields $v_{n+1,n+1} = v_{3,3} \approx 10^{-8}$ and $\sigma'_n - \sigma_{n+1} = O(10^{-8})$. The generic TLS solution yields a very sensitive, unstable solution given by $\left[1, 10^8 + 10^4\right]^T$. Hence, the nongeneric TLS solution must be computed. Using the SVD, the solution is

$$\hat{x} = \left[\frac{2}{\sqrt{5}-1}, \frac{2 \times 10^{-8}}{\sqrt{5}-3}\right]^T \tag{5.132}$$

Note that $\|\Delta\hat{x}\|_2 = 10^{-8}$ and then the perturbed solution is stable.

Contrary to the SVD-based techniques, TLS EXIN and its acceleration versions, for null initial conditions, are *stable* because they solve the nongeneric problem without changing the learning law. The results obtained (weights after convergence) are listed in Table 5.1 together with the number of epochs for the convergence and the SVD-based result. Figure 5.19 shows the phase diagram of TLS EXIN for different initial conditions. It has the same interpretation as that for Figure 5.16. In particular, *the domain of convergence is given by the horizontal half-line with origin the maximum* $(-0.618034, 0)$ *and containing the saddle/solution*. The divergence line is the asymptote z_2 and is vertical (parallel to x_2).

5.5 SCHEDULING

The ζ parameterization has been conceived to unify and generalize the three basic regression problems. The continuity of the solutions $\forall\zeta$ (i.e., the solution locus cited above) suggests a novel numerical technique for the solution of the basic problems, which further justifies the intermediary ζ cases. This technique is defined here as *scheduling* because it requires a predefined programmation of the parameter ζ. It uses the solution locus just like an *attraction locus* for the trajectory of the solution estimate in the x space and is conceived especially for solving the DLS problem, which is the most difficult.

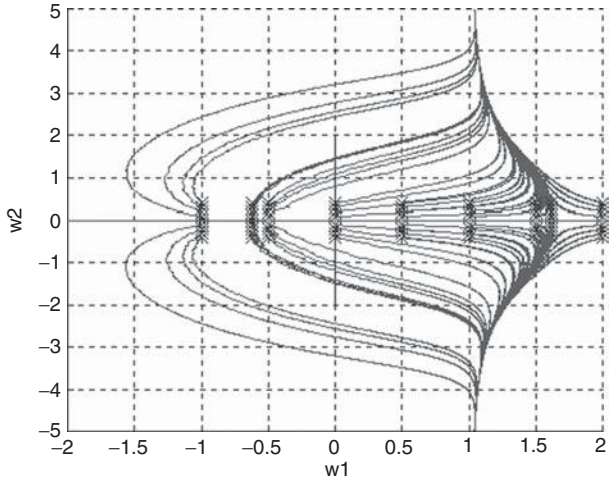


Figure 5.19 Evolution map of the sequential TLS EXIN for the second nongeneric TLS benchmark problem. The initial conditions for the converging trajectories are in red; the others are in black. The horizontal straight line passing through the red points contains the saddle/solution locus. The dark blue thick vertical line is the divergence line. (*See insert for color representation of the figure*.)

5.5.1 DLS Problem

From the point of view of the neural approaches and, more generally, of the gradient flows, the DLS problem ($\zeta = 1$) is the most difficult (for $b \neq 0$) for the following reasons:

- The domain of convergence is smaller, as shown in Section 5.4.
- Extending the definition of motor gap (Definition 38) to the height distance between a saddle i (height σ_i^2) and the corresponding immediately superior infinite (height $\sigma_{i-1}'^2 = \lambda_{i-1}$), the motor gap represents the gap of energy to descend to the inferior critical point, and in this case it is the smallest, just decelerating the gradient flow.
- All the infinite heights (different directions) are the lowest with respect to the other GeTLS problems.
- As shown in Proposition 105, the DLS solution is the farthest GeTLS solution from the origin.
- As shown in Remark 98, the DLS error cost is not defined for null initial conditions, and every gradient flow algorithm diverges for this choice of initial conditions. Hence, there is no universal choice that guarantees the convergence as for the other GeTLS EXIN neurons.

5.5.2 Direct Methods

Two direct methods are presented in [35]: the weighted TLS for $\alpha \to \infty$, introduced in Section 5.1.1, and the Householder transformation.

5.5.2.1 Householder Transformation This direct method has been proposed in [52] and is equivalent to the method in [51]. Compute a Householder transformation matrix Q such that $Q^T b = \left[\|b\|_2, \ldots, 0\right]^T$. Then the problem is reduced to solving

$$Q^T A x = \left[\|b\|_2, \ldots, 0\right]^T \tag{5.133}$$

Define $a_{r,1}$ as the first row of $Q^T A$. Let A' be the matrix consisting of the last $m-1$ rows of $Q^T A$. Find $\hat{x}$ such that

$$A'\hat{x} = 0 \quad \wedge \quad a_{r,1}\hat{x} = \|b\|_2 \tag{5.134}$$

When A is perturbed, all that can be said is $A'x \approx 0$. In this case, the closest lower rank approximation $\hat{A}'$ to A' (in the sense that $\left\|\hat{A}' - A'\right\|_F^2$ is minimized) is computed, together with its null vector, say v. This v is parallel to the right singular vector corresponding to the smallest singular value of A'. Thus, $\hat{x}$ is a scaled version of v such that $a_{r,1}\hat{x} = \|b\|_2$. If v is orthogonal to the first row of $Q^T A$ (nongeneric case), no solution exists. In this case, additional constraints must be added to this formulation.

5.5.3 DLS Scheduling

If GeTLS EXIN is used, the DLS problem requires a *finite* parameter ζ, but the corresponding cost function is not regular at the origin. In this chapter it has been proved that for every choice of ζ, the null initial conditions always guarantee the convergence. Hence, if no a priori information is given, the only way to assure the convergence to the DLS solution is the choice of a parameter very close but not equal to 1; this closeness influences the accuracy of the solution. To allow the null initial conditions without loss of accuracy, the DLS scheduling is proposed.

Note that the GeTLS EXIN weight vector follows the solution locus toward the DLS solution if its parameter ζ is made variable and increasing from 0 to 1 according to a predefined law (*scheduling*). The first iteration, for $\zeta = 0$, is in the direction of the OLS direction: therefore, in a case where both the null initial conditions and the values of the weights at the first iteration are in the domain of convergence. This new position is also in the DLS domain because it is in the direction of the solution locus. Changing slowly ζ makes the weight vector follow the hyperbola branch containing the solution locus for every x plane. Furthermore, the GeTLS neuron, for not too high an initial learning rate, begins with very low initial conditions from the first iteration. This accelerates the learning law. This idea is illustrated in Figure 5.20. Indeed, expressing the GeTLS cost error function (5.6) as

$$E_{\text{GeTLS EXIN}}(x, \zeta) = \frac{1}{2}\frac{(Ax-b)^T (Ax-b)}{y(\zeta)} \tag{5.135}$$

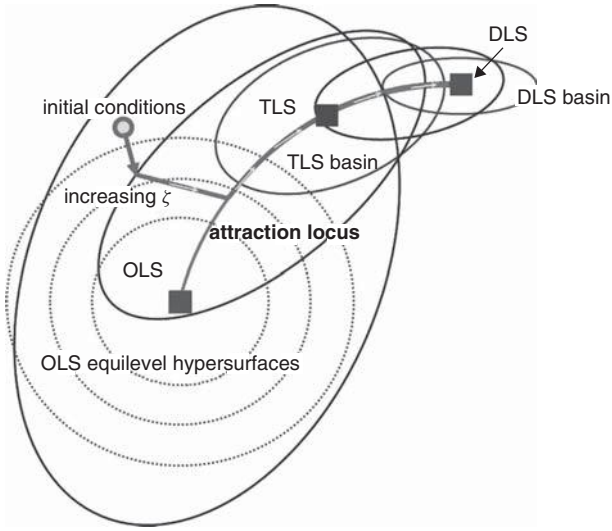


Figure 5.20 DLS scheduling. (*See insert for color representation of the figure*.

where $y(\zeta) = (1-\zeta) + \zeta x^T x = \left(\|x\|_2^2 - 1\right)\zeta + 1$, it is easy to verify the following properties:

1. If the weight vector x is internal to the unit hypersphere, then the GeTLS EXIN energy is directly proportional to ζ. In particular,

$$\|x\|_2 < 1 \Rightarrow E_{\text{DLS}} > E_{\text{TLS}} > E_{\text{OLS}} \tag{5.136}$$

2. If the weight vector x is on the unit hypersphere, then the GeTLS EXIN energy is invariant with respect to ζ. In particular:

$$\|x\|_2 = 1 \Rightarrow E_{\text{DLS}} = E_{\text{TLS}} = E_{\text{OLS}} \tag{5.137}$$

3. If the weight vector x is external to the unit hypersphere, then the GeTLS EXIN energy is inversely proportional to ζ. In particular,

$$\|x\|_2 > 1 \Rightarrow E_{\text{DLS}} < E_{\text{TLS}} < E_{\text{OLS}} \tag{5.138}$$

From these properties it is evident that it is necessary to have low weights after the first iteration and a scheduling for weight vectors into the unit hypersphere, because at a certain weight position x, the energy value rises for increasing ζ, thus accelerating the method.

Proposition 123 (DLS Scheduling) *A scheduling of the parameter ζ, defined as a continuous function from 0 to 1, combined with not too high a learning rate, accelerates the DLS EXIN neuron and guarantees its convergence in the absence*

of a priori knowledge about the initial conditions (*i.e., the null initial conditions are the universal choice for the convergence*).

Remark 124 (Energy Invariant) *The restriction of the energy function* $E_{GeTLS\ EXIN}$ *to the unit hypersphere is an invariant for every GeTLS problem* (*i.e., with respect to* ζ).

5.5.4 Examples

The first example [24] deals with the DLS benchmark problem in [195] considered in Section 2.10. A line model $a_1 x + a_2 y = 1$, where $a_1 = 0.25$ and $a_2 = 0.5$, is fitted to an observation data set by the estimation of its parameters. The set of equations is created by adding a Gaussian noise of zero mean and variance $\sigma^2 = 0.5$ to x and y. In comparison, the learning rate follows the same law as in Section 2.10: It is initially constant and equal to 0.01; then it is linearly reduced to 0.001 during the first 500 iterations and afterward is held constant. DLS EXIN and DLS scheduling EXIN are compared. The initial conditions $[0.1, 0.1]^T$ are chosen for the DLS EXIN neuron by using the a priori information that they are in the domain of convergence. Scheduling for the other neuron is

$$\zeta(j) = 1 - \frac{1}{j} \tag{5.139}$$

where $j \geq 1$ is the iteration number (hyperbolic scheduling). Figure 5.21 shows the transient: DLS scheduling EXIN reaches the value 0.5 600 iterations before DLS EXIN because of the possibility of working with very low weights. Figure 5.22 shows the dynamic behavior of the two neurons: They are similar, but DLS scheduling EXIN has smaller elongations. Concluding, DLS scheduling EXIN behaves slightly better than DLS EXIN, but it is always guaranteed to converge.

Remark 125 (Null Observation Vector) *The DLS scheduling cannot have null initial conditions when the observation vector* (b) *is null; indeed, the first step* ($OLS\ \zeta = 0$) *has* $x^T A^T A x$ *as an energy function, which is null for null initial conditions. The problem of the lack of universal initial conditions can be circumvented by inserting one or more OLS learning steps in the scheduling in order to enter the domain of convergence. If the initial conditions are very small, one OLS step is enough to assure the convergence.*

5.6 ACCELERATED MCA EXIN NEURON (MCA EXIN+)

As anticipated in Sections 2.5.3 and 2.11, the DLS scheduling EXIN can be used to improve the MCA EXIN neuron. Indeed, the DLS energy cost for $b = 0$ becomes [see eq. (5.21)]

$$E_{\mathrm{DLS}}(x) = \frac{1}{2} \frac{x^T A^T A x}{x^T x} = \frac{1}{2} r\left(x, A^T A\right) \qquad \forall x \in \Re^n - \{0\} \tag{5.140}$$

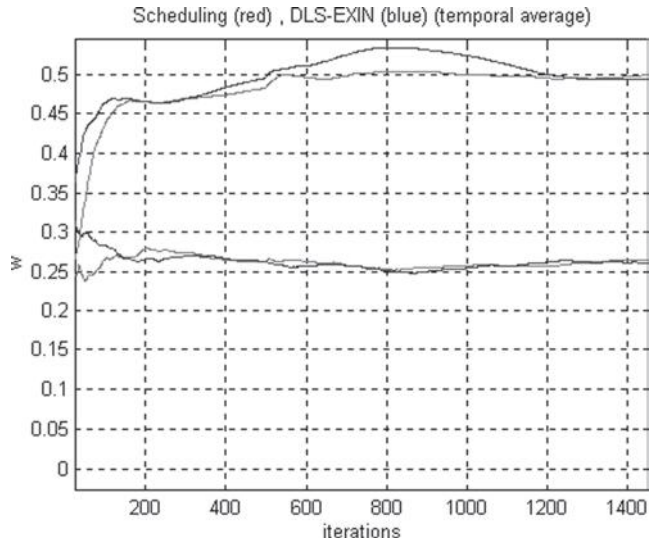


Figure 5.21 Line fitting without preprocessing for a noise variance of 0.5: transient analysis. The values are averaged using a temporal mask with width equal to the number of iterations up to a maximum of 500. (*See insert for color representation of the figure*.)

(i.e., the Rayleigh quotient of A^TA). Recalling that the neuron is fed with the rows of the matrix A and that the autocorrelation matrix R of the input data is equivalent to A^TA/m, it follows that the DLS neural problem is equivalent to an MCA neural problem with the same inputs. This equivalence is only possible for DLS EXIN and MCA EXIN because both neurons use the *exact* error gradient in their learning law.

Proposition 126 (DLS-MCA Equivalence) *The DLS EXIN neuron fed by the rows of a matrix A and with null target is equivalent to the MCA EXIN neuron with the same input. Both neurons find the minor component of A^TA (i.e., the right singular vector associated with the smallest singular value of A). The drawback of the equivalence is the fact that instead of MCA EXIN, which always converges, DLS EXIN is not guaranteed to converge.*

This equivalence is also easily proved by taking the DLS EXIN learning law and corresponding ODE [(5.10) and (5.12) for $\zeta = 0$] and setting $b = 0$; it yields the MCA EXIN learning law and corresponding ODE [eqs. (2.35) and (2.33). As a consequence of the equivalence, the DLS scheduling EXIN can be used to improve the MCA EXIN; for this purpose, it is called **MCA EXIN+**.

Definition 127 (MCA EXIN+) *The MCA EXIN+ is a linear neuron with a DLS scheduling EXIN learning law, the same inputs of the MCA EXIN, and a null target.*

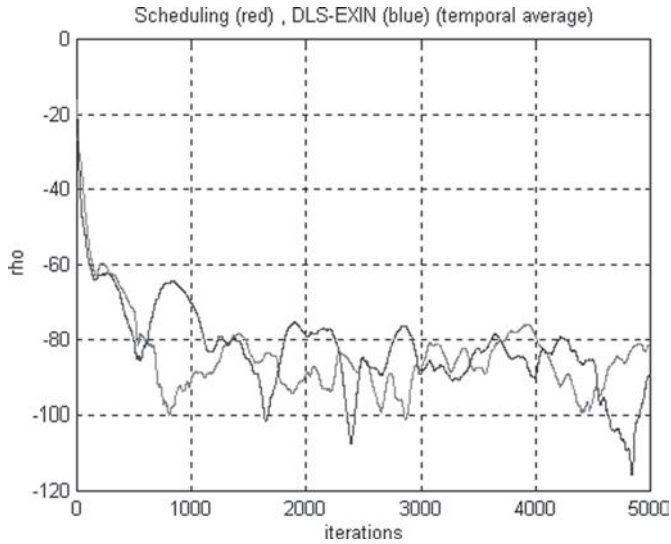


Figure 5.22 Line fitting without preprocessing for a noise variance of 0.5: plot of the index parameter (expressed in decibels). The values are averaged using a temporal mask with width equal to the number of iterations up to a maximum of 500. (*See insert for color representation of the figure*.)

As shown in Section 2.7, big fluctuations of the weights imply that the learning law increases the estimation error, and when this increase is too large, it will make the weight vector deviate drastically from normal learning, which may result in divergence or an increased learning time. This is a serious problem for MCA EXIN (see Second 74) when the initial conditions are infinitesimal. On the other hand, MCA EXIN converges faster for smaller initial conditions (Theorem 62). Recalling these observations and Remark 125, the MCA EXIN+ improvements with respect to the MCA EXIN are:

- *Smoother dynamics*, because the weight path in every plane $z_i z_j$ remains near the hyperbola branch containing the solution locus
- *Faster convergence*, because of the smaller DLS scheduling fluctuations, which reduce the settling time
- *Better accuracy*, because of the small deviations from the solution

The next simulations deal with the benchmark in [195] considered in Sections 2.10 and 5.5.3. MCA EXIN+ uses the same normalized training set of MCA neurons. The first simulation uses the learning rate $\alpha(t)$ such that $\alpha(0) = 0.01$; then it is linearly reduced to 0.001 after 500 iterations, and, afterward, kept constant. The initial conditions are $[0.1, 0.1]^T$. The additive noise is Gaussian with $\sigma^2 = 0.5$. The scheduling is linear from $\zeta(0) = 0$ to $\zeta(5000) = 1$. Figures 5.23 and 5.24 show, respectively, the plot of the index parameter (expressed in decibels) and

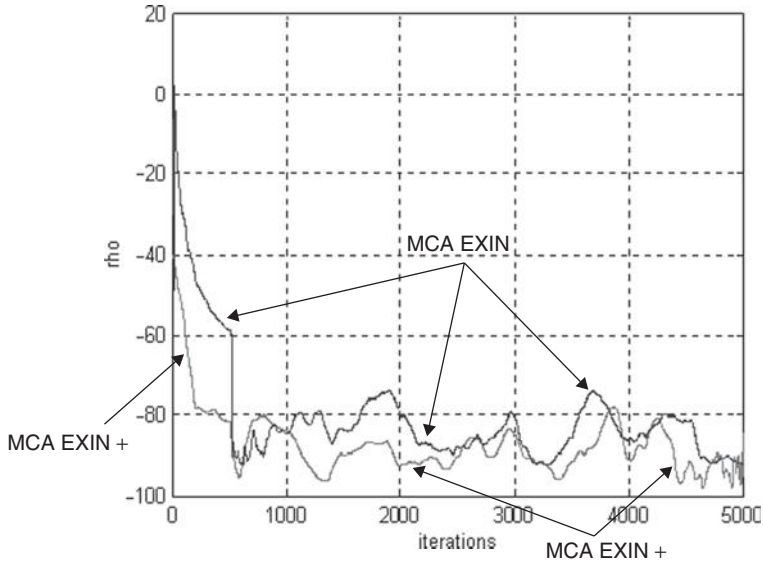


Figure 5.23 Line fitting for a noise variance of 0.5: plot of the index parameter (expressed in decibels) for MCA EXIN and MCA EXIN+. The values are averaged using a temporal mask with width equal to the number of iterations up to a maximum of 500. (*See insert for color representation of the figure*.)

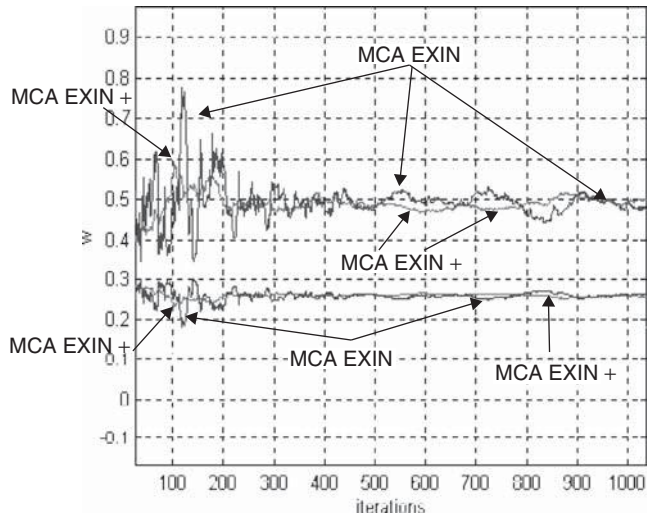


Figure 5.24 Line fitting for a noise variance of 0.5: transient for MCA EXIN and MCA EXIN+. (*See insert for color representation of the figure*.)

the transient for MCA EXIN and MCA EXIN+. The first figure evidentiates well the faster convergence and better accuracy of MCA EXIN+, the second its smoother dynamics.

Remark 128 (Learning Rate) *The smoother dynamics of MCA EXIN+ with respect to MCA EXIN allows the use of higher learning rates for accelerating the convergence. For too fast learning rates, on the contrary, MCA EXIN has a better transient and convergence because MCA EXIN+ loses its stable path.*

As pointed out in Section 2.5.3, MCA EXIN has substantial problems in dealing with very small initial conditions because of the amplifying effect on its learning law (MCA EXIN works well for small, but not too small, weights). MCA EXIN+ solves this problem, always yielding better properties than the other MCA neurons. The second simulation is analog to the first except for the initial conditions, which are now $\left[10^{-8}, 10^{-8}\right]^T$. MCA EXIN gives inconsistent results. As seen in Remark 125, one OLS step in the scheduling is enough to assure the convergence. It is also enough to improve the MCA EXIN convergence. The second simulation compares the MCA linear neurons with the MCA EXIN+ endowed with a linear scheduling from $\zeta(0) = 0$ to $\zeta(15) = 1$. Figure 5.25 shows the better accuracy of MCA EXIN+ (LUO has coincident inconsistent weights). Different types of scheduling are possible: The hyperbolic scheduling (5.139), typical of the DLS scheduling EXIN, has poorer accuracy than the linear scheduling of these last simulations. Indeed, the hyperbolic scheduling goes to 1 only at infinity. The transient behavior depends on the slope of the linear scheduling straight

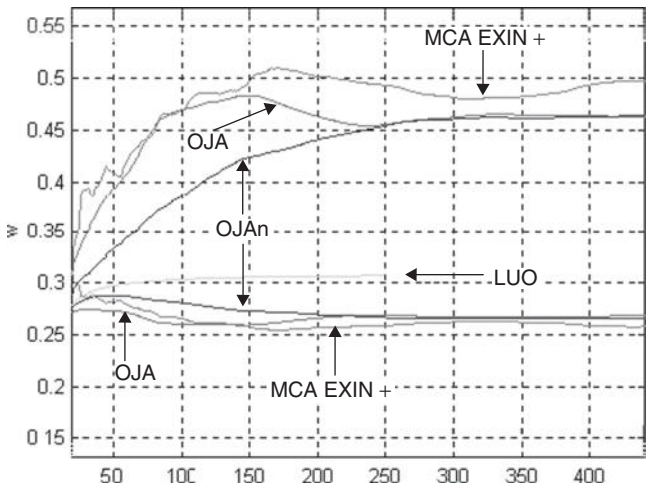


Figure 5.25 Line fitting for a noise variance of 0.5: transient for OJA, OJAn, LUO, and MCA EXIN+. The values are averaged using a temporal mask with width equal to the number of iterations up to a maximum of 500. (*See insert for color representation of the figure*.)

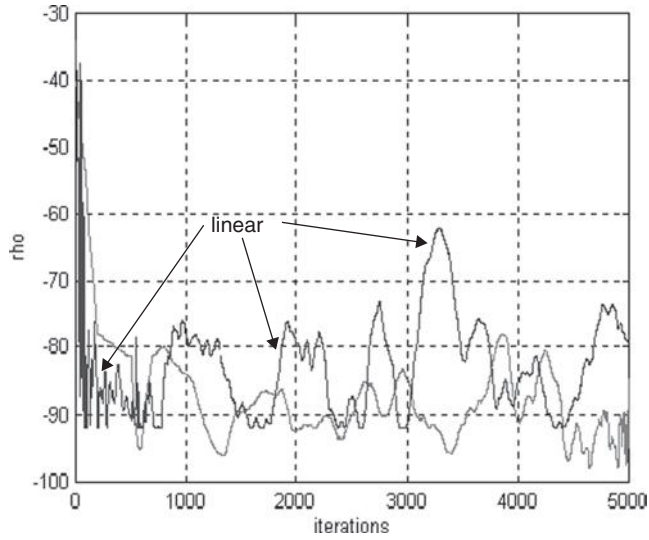


Figure 5.26 Line fitting for a noise variance of 0.5: plot of the index parameter (expressed in decibels) for MCA EXIN+ with linear scheduling (red) and MCA EXIN+ with hyperbolic scheduling (blue). The values are averaged using a temporal mask with width equal to the number of iterations up to a maximum of 500. (*See insert for color representation of the figure*.)

line with respect to the tangent to the hyperbolic scheduling curve at the first iteration. Figure 5.26 shows this comparison.

5.6.1 MCA EXIN + Flowchart

- *Goal:* to find the minimum eigenvector $x(t)$ of the matrix A.
- *Inputs:*
  1. $\eta(t)$: learning rate, decreasing to zero
  2. $x(0)$: initial conditions (better as small as possible, but not null)
  3. $\zeta(t)$: GeTLS parameter, increasing from 0 to 1
  4. $a(t)$: the row of A that is input at instant t
  5. ε: stop threshold
  6. $t_{\max}$: maximum number of iterations
- *Algorithm:*
  1. For each t
     (a) Compute:

$$x(t+1) = x(t) - \eta(t)\gamma(t)a(t) + \left[\zeta(t)\eta(t)\gamma^2(t)\right]a(t)$$

where

$$\gamma(t) = \frac{a^T(t)x(t)}{1-\zeta(t)+\zeta(t)x^T(t)x(t)}$$

(b) Compute:

$$E_{\text{GeTLS}}(t+1) = \frac{1}{2}\frac{(Ax(t))^T (Ax(t))}{1-\zeta(t)+\zeta(t)x^T(t)x(t)} = \frac{\gamma(t)}{2}\frac{(Ax(t))^T (Ax(t))}{a^T(t)\,x(t)}$$

(c) If

$$\|E_{\text{GeTLS}}(t+1) - E_{\text{GeTLS}}(t)\|_2 < \varepsilon$$

for a certain number of iterations (i.e., 100), then STOP.

(d) If $t > t_{\max}$, then STOP.

(e) Update $\eta(t)$ and $\zeta(t)$.

• *Note*: $\zeta(t)$ must be equal to 1 well before the algorithm is stopped.

5.7 FURTHER CONSIDERATIONS

5.7.1 Homogeneous Linear Systems

Given an approximated homogeneous linear system in the form

$$Uz \approx 0 \tag{5.141}$$

where $U \in \Re^m \times n$, the solution z (approximate linear relation between the columns of U) can be obtained solving the optimization problem called the *orthogonal L_2 approximation problem* [98, p. 45; 173,188].

Definition 129 (Orthogonal L_2 Approximation Problem) *Given the data matrix $U \in \Re^{m\times n}$, the orthogonal L_2 approximation problem seeks*

$$\min_{z\in\Re^n} \|Uz\|_2 \qquad \text{subject to} \quad z^T z = 1 \tag{5.142}$$

The constraint in z is needed to avoid the trivial solution $z = 0$. The solution to this problem is given by the eigenvector of $U^T U$ associated with the smallest eigenvalue (i.e., the right singular vector of U associated with the smallest singular value). Hence, in the framework of the theory presented in this chapter, the problem can be solved by using the following three neurons (with the usual assumptions on the additive noise in the coefficients of U):

1. *TLS EXIN*. Taking as A (data matrix) $n-1$ columns of U and as b (observation vector) the remaining column [recall the equivalence between the TLS problem and the homogeneous system (1.14) with the corresponding solution as the intersection of the "minimum" right singular vector of

$[A; b]$ with the TLS hyperplane, this neuron solves the problem; after the convergence, normalization of the weights is needed in order to respect the constraint (5.143).

2. *MCA EXIN*. Taking as input each row of U, picked up randomly, this neuron solves the problem by directly minimizing the Rayleigh quotient of $U^T U$. Recalling the divergence problem, normalization is needed after the stopping criterion is satisfied, in order to respect the constraint (5.143).
3. *MCA EXIN+*. Taking $A = U$ and $b = 0$, this neuron solves the problem, but needs further normalization, as in the other cases.

The choice of the neuron is problem dependent and is a consequence of its own properties.

5.7.2 Underdetermined Unidimensional Linear Systems

Given the underdetermined unidimensional system $Ax = b$, where $A \in \Re^m \times n$, $b \in \Re^m$, and $m < n$, using the usual notation it follows that

$$\text{rank}[A; b] = m \Rightarrow \sigma_m > \sigma_{m+1} = \cdots = \sigma_{n+1} = 0 \tag{5.143}$$

(i.e., the last $n - m + 1$ singular values are null). Recalling the interlacing property (1.16), it also follows that the last $n - m$ eigenvalues of $A^T A$ are null. The existence condition is $\sigma'_m > \sigma_{m+1}$, which implies that $\sigma_m \geq \sigma'_m > 0$. The TLS problem (1.9) still yields a solution, but it is not unique. Indeed, any linear combination of singular vectors $v_{m+1}, \ldots, v_{n+1}$ solves the TLS problem after normalization (intersection with the TLS hyperplane). For reasons of stability and minimal sensitivity, it is desirable to single out the solution with minimal L_2 norm. Recalling eqs. (1.30) and (1.7), the compatibility of the system, given by $\sigma_{n+1} = 0$, yields

$$x' = \hat{x} = (A^T A)^{-1} A^T b = A^+ b \tag{5.144}$$

as the minimal L_2 norm solution. This norm is

$$\left\| x' \right\|_2^2 = \| \hat{x} \|_2^2 = b^T \left(A A^T \right)^{-1} b \tag{5.145}$$

and the corresponding error cost is null [see, e.g., eq. (1.21)]. The following theorem extends this analysis.

Theorem 130 (Underdetermined GTLS Problems) *If the system of linear equations $Ax = b$ is underdetermined, all the minimal L_2 norm GeTLS solutions (i.e., $\forall \zeta$) are equal.*

Proof. Considering that $\text{rank}[A] = m < n$, the last $n - m$ eigenvalues of $A^T A$ are null. Recalling the theory in Section 5.3.1, the function $g(\gamma, \zeta)$ has $n - m$

asymptotes coincident with the $\gamma = 0$ axis. It implies that $\gamma_{\min} = 0 \ \ \forall\zeta$. Thus, eq. (5.51) yields the coincidence of all GeTLS solutions. The corresponding parameter $t_{\min} = 0 \ \ \forall\zeta$, and therefore this common solution is the minimal L_2 norm solution (see Proposition 105 and Theorem 108; the point at $t = 0$ is the origin of the solution locus and is nearest the origin of the reference system x). ■

Corollary 131 (Neural Solutions) *For the underdetermined unidimensional problem, the GeTLS EXIN neuron yields the same solution independent of its parameter* ζ.

The following simulation deals with the benchmark problem taken from [22, Ex. 1]:

$$\begin{bmatrix} 2 & -1 & 4 & 0 & 3 & 1 \\ 5 & 1 & -3 & 1 & 2 & 0 \\ 1 & -2 & 1 & -5 & -1 & 4 \end{bmatrix} x = \begin{bmatrix} 2 \\ 1 \\ -4 \end{bmatrix} \tag{5.146}$$

The minimal L_2 norm solution, obtained by using MATLAB, is

$$x' = \hat{x} = x'' = [0.088, 0.108, 0.273, 0.505, 0.383, -0.310]^T \tag{5.147}$$

The only comparable results in [22, Ex. 1] have a settling time of about 250 ns for the OLS problem (clock frequency = 100 MHz, which implies one iteration every 10 ns). Table 5.2 shows the results for GeTLS EXIN. The initial conditions are null; to avoid the divergence, the accelerated DLS neurons have $\zeta = 0.99$. Every batch is composed of the coefficients of the three equations. The learning rate for the nonaccelerated GeTLS EXIN is $\alpha(t) = \alpha_0/t^\gamma$. Note the acceleration of the SCG and, above all, of the BFGS methods (recall Remark

Table 5.2 Underdetermined linear system benchmark[a]

	ζ	α_0	γ	Time (ns)
GeTLS EXIN seq.	0	0.09	0.51	200
GeTLS EXIN seq.	0.5	0.09	0.51	750
GeTLS EXIN seq.	hyp.	0.09	0.51	1100
GeTLS EXIN batch	0	0.1	0.8	300
GeTLS EXIN batch	hyp.	0.1	0.8	1650
SCG GeTLS EXIN	0	—	—	270
SCG GeTLS EXIN	0.5	—	—	270
SCG GeTLS EXIN	0.99	—	—	330
BFGS GeTLS EXIN	0	—	—	200
BFGS GeTLS EXIN	0.5	—	—	200
BFGS GeTLS EXIN	0.99	—	—	300

[a] seq., sequential; hyp., hyperbolic scheduling.

40). Evidently, the DLS approach needs the largest number of iterations. All transients are far better than in [22, Ex. 1]; for the TLS and DLS approaches it is a consequence of the approximated learning law of these neurons, whereas in the OLS approach, it is a consequence of their particular presentation of the inputs.

5.7.3 Mixed OLS–TLS Problems

The mixed OLS–TLS problems introduced in Section mix are solved by the GeTLS EXIN linear neuron (MADALINE for the multidimensional case) by working on the training set. The procedure follows (compare with [98, Alg. 3.2]):

1. *Preprocessing of the training set:* column permutations, to have the first columns of the data matrix known exactly and QR factorization.
2. *Training of GeTLS EXIN for* $\zeta = 0.5$ (TLS): null initial conditions avoid the need of a numerical rank determination (see Section 5.4).
3. *Training of GeTLS EXIN for* $\zeta = 0$ (OLS).
4. *Postprocessing of the solution* (*multidimensional case*): inverse permutations.

The same idea of pre- and postprocessing can be applied to other problems, just like the GeTLS problem in [98, Sec. 10.3] (preprocess the training set by using error equilibration matrices) and the general TLS formulation in [74] (preprocess the training set by using the nonsingular diagonal matrices D and T).

5.7.4 Note on the Choice of the Numerical Stop Criteria

As pointed out in this chapter, it is impossible to use the usual stop criteria for the MCA and TLS learning law, because the error cost does not go to zero at convergence. Indeed, the minimum of the MCA error cost is equal to the smallest eigenvalue of the autocorrelation matrix, and the minimum of the GeTLS error cost depends on the incompatibility of the linear equations and on the validity of the linear modeling [for TLS the minimum is equal to σ_{n+1}^2, as shown in eq. (1.21)]. Hence, it is necessary to detect the flatness of the curves representing either the weight components or the error function. In all simulations the following numerical stop criteria have been chosen:

- *For TLS EXIN, TLS GAO, and all MCA linear neurons*: Define

$$\Delta w(k) = w(k) - w(k-1) \tag{5.148}$$

where $w \in \Re^n$ and k is the iteration. Then the learning law stops when

$$\|\Delta w(k)\|_\infty \equiv \max_i |\Delta w_i(k)| < \varepsilon \tag{5.149}$$

for T iterations, where the L_∞ or Chebyshev norm has been taken into account; T is in general equal to the number of vectors of the training set or to a predefined number in case of online learning and $\varepsilon = 10^{-10}$.

- *For SCG TLS EXIN*:The learning law stops if either of the following two criteria is satisfied:
  1. $\|Ax - b\|_2^2 < \varepsilon_m$, where ε_m is the machine error (valid only for compatible or underdetermined systems).
  2. Define $E(k)$, the error at iteration k,

$$|\Delta E(k)| = |E(k) - E(k-1)| < \varepsilon \tag{5.150}$$

  for T iterations, where $\varepsilon = 10^{-5}$ and $T = \max(3, n)$, n being the dimension of the weight vector.
- *For BFGS TLS EXIN* : By using (2.123), the learning law stops when

$$\|\Delta w(k)\|_1 \equiv \sum_{i=1}^{n} |\Delta w_i(k)| < n\varepsilon \tag{5.151}$$

for $T = n$ iterations (n is the dimension of the weight vector), where the L_1 or absolute value norm has been taken into account and $\varepsilon = 10^{-10}$.

5.8 SIMULATIONS FOR THE GeTLS EXIN NEURON

The first simulation deals with the benchmark problem in [195] considered in Sections 2.10 and 5.5.4. First, m points are taken from this line with $0 < x < 5$ by uniform sampling. Then, the noisy observation set is generated by adding Gaussian noise with zero mean and variance $\sigma^2 = 0.5$ to these points. Figure 5.27 shows the temporal evolution of the estimates for the inverse iteration (dashed) and scheduling (solid) methods for $m = 1000$. The inverse iteration method requires fewer iterations, but the cost per iteration is much larger. It is apparent in Table 5.3, where the problem is solved for several values of m. The scheduling technique works far better for large m: For $m \geq 400$, the computational cost remains nearly the same for a given accuracy. This phenomenon can easily be explained by the observation that the GeTLS error cost 5.6 derives from the (weighted) Rayleigh quotient of $C^T C$, and then its minimization operates on the column space of C, implying that it depends primarily on n and not on m [24]. This is a very important advantage for large and very large systems of linear equations [33,61].

The next simulations deal with the benchmark problem (5.57) introduced in Section 5.3.3 and considered subsequently in the analysis of the GeTLS stability. In this section we compare the GeTLS EXIN neuron with the other principal TLS neurons. The first simulations, whose results are shown in Table 5.4, deal

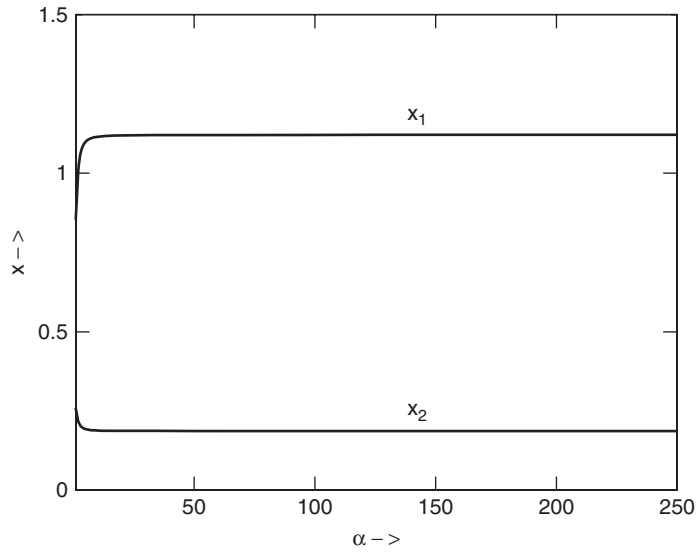


Figure 5.27 Weighted TLS method: plot of x_1 and x_2 vs. α.

Table 5.3 Total computational cost (flops) of the inverse iteration and scheduling methods for the benchmark problem

	m				
	50	100	250	400	1000
Inverse iteration	31,123	122,596	756,316	1,928,876	12,022,016
Scheduling	46,175	313,799	451,397	632,009	632,747

Table 5.4 Overdetermined linear system benchmark[a]

	ζ	α_0	γ	Time (ms)
GeTLS EXIN seq.	0	0.25	0.45	0.4
GeTLS EXIN seq.	hyp.	0.035	0.002	2
GeTLS EXIN batch	hyp.	0.0046	0	194
GeTLS EXIN batch	0	0.4	0.6845	4.2
GeTLS EXIN batch	0.5	0.03	0.0038	10
SCG GeTLS EXIN	0	—	—	0.1
SCG GeTLS EXIN	0.5	—	—	0.8
SCG GeTLS EXIN	0.99	—	—	1.6
BFGS GeTLS EXIN	0	—	—	0.1
BFGS GeTLS EXIN	0.5	—	—	0.1
BFGS GeTLS EXIN	0.99	—	—	0.1

[a] seq., sequential; hyp., hyperbolic scheduling.

with a comparison of the neurons presented in [22]. As in Section 5.7.2, the only comparable results in [22, Ex. 3] have a settling time of about 250 ns. The initial conditions are null (see Section 5.7.2 for the DLS case). Every batch is composed of the coefficients of the five equations. The learning rate for the nonaccelerated GeTLS EXIN is $\alpha(t) = \alpha_0/t^\gamma$. The discrete-time sequential (online) linear neuron in [22, Fig. 7] has a settling time of more than 40 ms for the OLS and TLS problems. The continuous-time batch linear neuron in [22, Fig. 6] reaches a good accuracy for the OLS and TLS problems after about 0.4 ms. The results for the GeTLS EXIN, shown in Table [5.4], corresponding to the same accuracy of the MATLAB results, are far better than the previous ones, especially for its accelerated versions (recall Remark 40). The same considerations of Section 5.7.2 can be repeated here. About the DLS estimation, the classical methods and the DLS scheduling EXIN are compared. The Householder method gives the solution $x = [1.1200, 0.1867]^T$. For the weighted TLS method, the parameter α has been varied between 1 and 250 in steps of 1. The variation of x_1 and x_2 vs. α is shown in Figure 5.27. For $\alpha \approx 200$ the solution is $x = [1.1200, 0.1867]^T$ and does not change for larger α. Then a not too large value of α is enough for a good accuracy of the DLS solution. Figure 5.28 shows the plot of the weights of the sequential GeTLS EXIN with hyperbolic scheduling. The initial conditions are null. The settling time is about 2.5 ms (1 itereration = 0.01 ms). The learning rate is held constant at the value 0.5 for the first 130 iterations and at the value 0.001 after iteration 300. The two constant values are connected by a third-degree polynomial. The transient and the accuracy are excellent. Figure 5.29

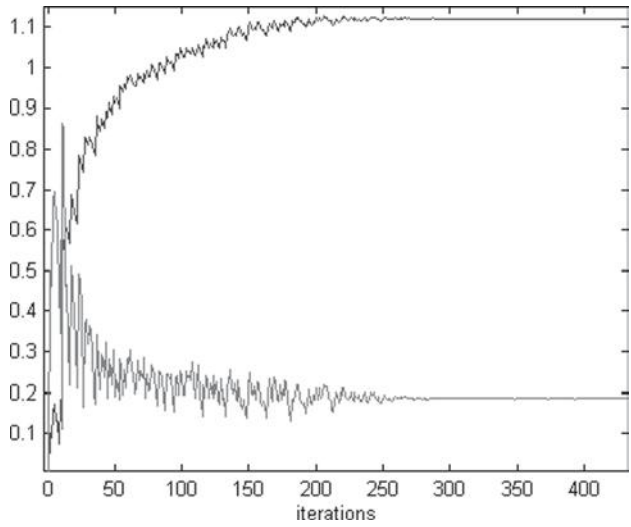


Figure 5.28 Plot of the weights of DLS scheduling EXIN for the benchmark problem (hyperbolic scheduling and null initial conditions). (*See insert for color representation of the figure*.)

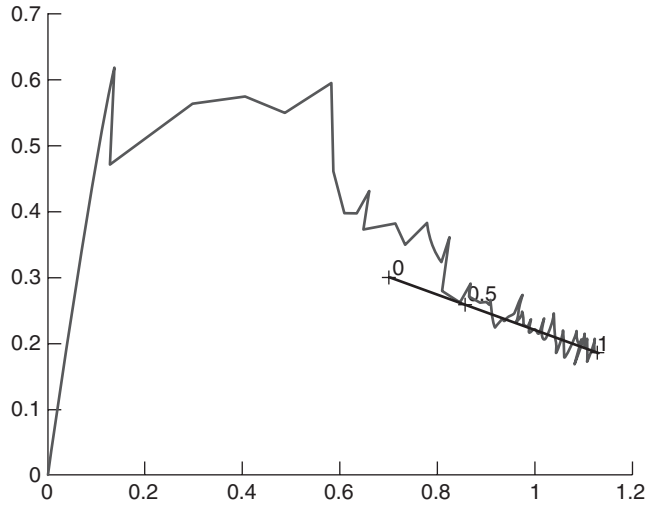


Figure 5.29 Phase diagram of the weights of DLS scheduling EXIN for the benchmark problem (hyperbolic scheduling and null initial conditions). The solution locus is indicated together with the OLS, TLS, and DLS solutions.

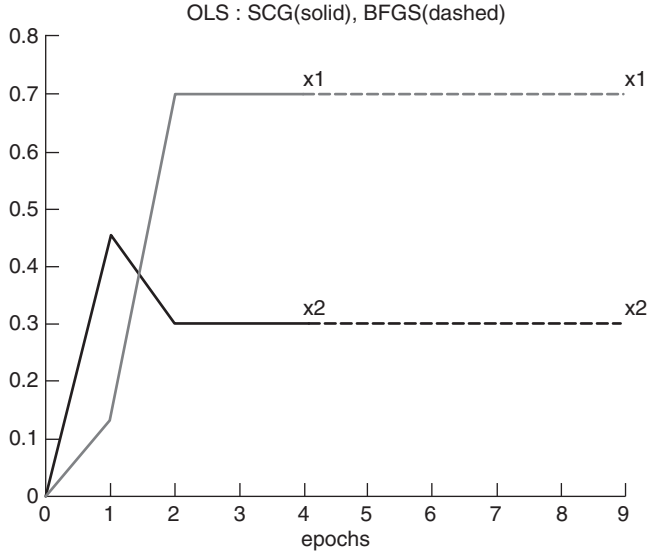


Figure 5.30 Plot of the weights of SCG and BFGS EXIN for the OLS benchmark problem. (*See insert for color representation of the figure*.)

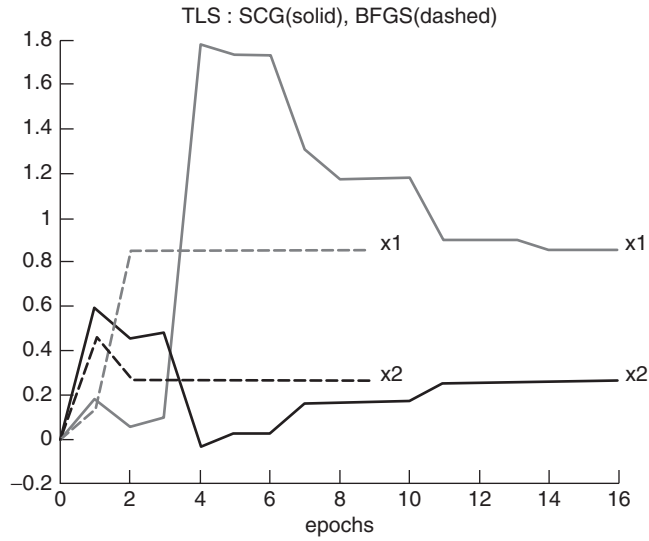


Figure 5.31 Plot of the weights of SCG and BFGS EXIN for the TLS benchmark problem. (*See insert for color representation of the figure*.)

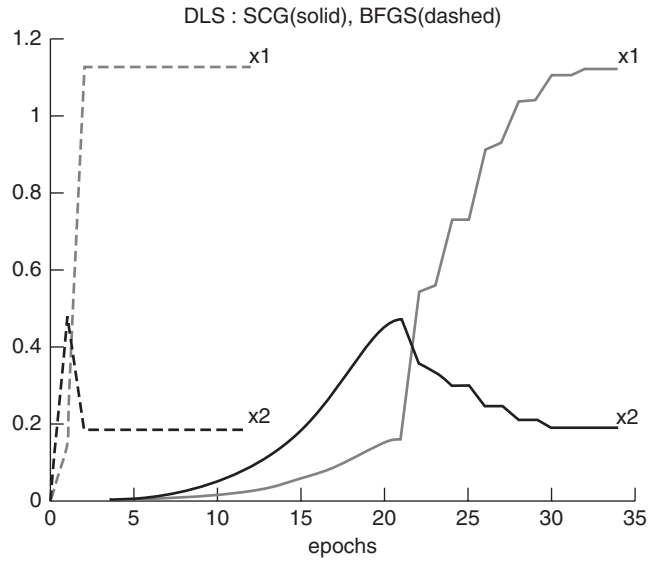


Figure 5.32 Plot of the weights of SCG and BFGS EXIN for the DLS benchmark problem. (*See insert for color representation of the figure*.)

Table 5.5 Comparison of TLS GAO with TLS EXIN for the TLS benchmark problem

	α_0	γ	Iterations
TLS EXIN	0.001	0	1718
TLS GAO	0.001	0	2132
TLS EXIN	0.01	0	367
TLS GAO	0.01	0	431

shows the weight phase diagram of this simulation. The weights approach the solution locus (parameterized in the figure by ζ) just before the TLS solution, because the hyperbolic scheduling yields only a single learning step for $\zeta = 0$. Note the *attractive* effect of the solution locus, which confirms the analysis in Section 5.5. Figures 5.30, 5.31 and 5.32 show the weight vector dynamics of the accelerated versions of GeTLS EXIN for, respectively, the OLS, TLS, and DLS problems. The initial conditions are null. The DLS is solved by using $\zeta = 0.9999$, thus allowing null initial conditions. Note the very good transients, especially for the BFGS approach, which also has a very fast convergence: 0.1 ms for OLS, TLS, and DLS (1 epoch = 5 iterations). The SCG and BFGS learning laws stop differently; see Section 5.7.4 for the stop criteria chosen.

Table 5.5 shows a comparison between TLS EXIN and TLS GAO for null initial conditions. TLS GAO has a better transient and a faster convergence. The differences are smaller for greater $\alpha(t) = \alpha_0/t^{\gamma}$ because it increases the randomness of the TLS EXIN law, thus giving an effect of the same type of linearization for the TLS GAO.

6

GeMCA EXIN THEORY

6.1 GeMCA APPROACH

The cost function (5.6) can be rewritten as a generalized Rayleigh quotient:

$$E_{\mathrm{GeTLS}}(x) = \frac{1}{2}\frac{(Ax-b)^T(Ax-b)}{(1-\zeta)+\zeta x^T x} = \frac{u^T R u}{u^T D u} \tag{6.1}$$

where $u = \left[x^T; -1\right]^T$, $R = [A; b]^T [A; b]$, and $D = \mathrm{diag}\left(\overbrace{2\zeta, \ldots, 2\zeta}^{n \text{ times}}, 2(1-\zeta)\right)$. For $0 < \zeta < 1$, the error (5.6) is the ratio of positive definite quadratic forms and represents the generalized Rayleigh quotient of the symmetric positive definite pencil (R, D) [178]. Its critical points solve the generalized eigenvalue problem $Ru = \gamma Du$:

$$\begin{bmatrix} A^T A & A^T b \\ b^T A & b^T b \end{bmatrix} u = 2\gamma \begin{bmatrix} \zeta I_n & 0 \\ 0^T & 1-\zeta \end{bmatrix} u \tag{6.2}$$

The last equation in (6.2) yields the GeTLS EXIN secular equation:

$$\begin{aligned} & b^T A x_{\mathrm{crit}} + 2\gamma(1-\zeta) - b^T b = 0 \\ & \quad \Longrightarrow q^T \left(\Lambda' - 2\gamma\zeta I_n\right)^{-1} q + 2\gamma(1-\zeta) - b^T b = 0 \end{aligned} \tag{6.3}$$

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

where $V'\Lambda'V'^T$ is the eigendecomposition of $A^TA, q \equiv V'^TA^Tb$, and x_{crit} is the value of x corresponding to γ, which is one of the critical values for $E_{\text{GeTLS}}(\zeta, x)$.

The first n equations in (6.2) yield the GeTLS EXIN critical value corresponding to γ:

$$x_{\text{crit}} = \left[A^TA - 2\gamma\zeta I_n\right]^{-1} A^Tb \tag{6.4}$$

Equations (6.3) and (6.4) are the basis of the GeTLS EXIN theory. In this theory the TLS hyperplane is defined as the locus whose equation is $u_{n+1} = -1$. It contains the GeTLS solutions for each value of ζ [24,31,35,156].

In conclusion, the GeTLS EXIN approach can be analyzed as a generalized eigenvalue problem because of the form of the error function (5.6). This interpretation is important from both a theoretical point of view (new theorems and inequalities have been deduced) and a numerical point of view (other algorithms can be used besides the GeTLS EXIN algorithm (neuron) [24,31,35], such as Wilkinson's algorithm [35]).

In the next sections we analyze this symmetric positive definite (S/PD) generalized eigenvalue problem in detail.

6.1.1 Particular Cases

6.1.1.1 Case $\zeta = 0$ (OLS) If $\zeta = 0$, it follows that

$$D = 2\begin{bmatrix} 0_n & 0 \\ 0^T & 1 \end{bmatrix} \tag{6.5}$$

Define as $u_\perp$ each vector u belonging to the n-dimensional hyperplane passing through the origin and whose normal is given by the vector $e_{n+1} \in \Re^{n+1}$, all of whose components are null except the last. It follows that

$$u_\perp \perp e_{n+1} \Rightarrow Du_\perp = 0 \Rightarrow \frac{1}{\gamma}Ru_\perp = 0 \Rightarrow \frac{1}{\gamma} = 0 \Rightarrow \gamma = \infty \tag{6.6}$$

because R is nonsingular. Hence, $(\infty, u_\perp)$ is an eigenpair with an infinite eigenvalue whose algebraic multiplicity is n and $u_\perp$ belongs to the subspace orthogonal to e_{n+1}. It means that this eigenspace is parallel to the TLS hyperplane. It implies that all critical points except one of the GeTLS EXIN cost functions disappear. The following analysis estimates the remaining eigenpair and confirms the foregoing considerations. From eq. (6.2), by defining $u = \left[u_1^T, u_{n+1}\right]^T$ with $u_1 \in \Re^n$ and $u_{n+1} \in \Re$, we can derive

$$\begin{bmatrix} A^TA & A^Tb \\ b^TA & b^Tb \end{bmatrix}\begin{bmatrix} u_1 \\ u_{n+1} \end{bmatrix} = 2\gamma\begin{bmatrix} 0_n & 0 \\ 0^T & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_{n+1} \end{bmatrix}$$

$$\Rightarrow \begin{cases} A^T A u_1 + A^T b u_{n+1} = 0 \\ b^T A u_1 + b^T b\, u_{n+1} = 2\gamma u_{n+1} \end{cases}$$

$$\Rightarrow \begin{cases} u_1 = -\left(A^T A\right)^{-1} A^T b\, u_{n+1} \\ -b^T A \left(A^T A\right)^{-1} A^T b\, u_{n+1} + b^T b\, u_{n+1} = 2\gamma u_{n+1} \end{cases}$$

$$\Rightarrow b^T \left[I - A \left(A^T A\right)^{-1} A^T \right] b\, u_{n+1} = 2\gamma u_{n+1} \tag{6.7}$$

Two solutions are possible. The first requires that $u_{n+1} = 0$ and is another proof of the existence of the $(\infty, u_\perp)$ eigenpairs. The second requires that $u_{n+1} \neq 0$ (i.e., u not parallel to $u_\perp$) and

$$\gamma = \frac{1}{2} b^T \left[I - A \left(A^T A\right)^{-1} A^T \right] b = \frac{1}{2} b^T P_A^\perp b = \frac{1}{2} \left\| P_A^\perp b \right\|_2^2 \tag{6.8}$$

where $P_A^\perp = \left(I_m - A \left(A^T A\right)^{-1} A^T \right)$ is the symmetric projection matrix ($P_A^{\perp 2} = P_A^\perp$) that projects into the orthogonal complement of the column space of A. From eq. (6.4) it follows that for $\zeta = 0$,

$$x_{\text{OLS}} = \left[A^T A\right]^{-1} A^T b = A^+ b \tag{6.9}$$

which is the well-known OLS solution. Recalling that γ corresponds to the value of the error function (5.6), we have

$$\begin{aligned} \gamma_{\text{OLS}} &= E_{\text{OLS}}\left(x_{\text{OLS}}\right) = \tfrac{1}{2} \left(A x_{\text{OLS}} - b\right)^T \left(A x_{\text{OLS}} - b\right) = \tfrac{1}{2} \left\| A x_{\text{OLS}} - b \right\|_2^2 \\ &= \tfrac{1}{2} \left\| A \left[A^T A\right]^{-1} A^T b - b \right\|_2^2 = \tfrac{1}{2} \left\| P_A^\perp b \right\|_2^2 \end{aligned} \tag{6.10}$$

which confirms the result (6.8).

Summarizing: There are n infinite eigenvalues whose corresponding eigenvectors do not intersect the TLS hyperplane and one eigenvalue, γ_{OLS}, corresponding to the eigenvector which is is the minimum of the GeTLS EXIN cost function and intersects the TLS hyperplane in the OLS solution (6.9).

6.1.1.2 Case $\zeta = 1$ (DLS)

If $\zeta = 1$, it follows that

$$D = 2 \begin{bmatrix} I_n & 0 \\ 0^T & 0 \end{bmatrix} \tag{6.11}$$

$$u = e_{n+1} \Rightarrow D e_{n+1} = 0 \Rightarrow \frac{1}{\gamma} R e_{n+1} = 0 \Rightarrow \frac{1}{\gamma} = 0 \Rightarrow \gamma = \infty \tag{6.12}$$

Hence, (∞, e_{n+1}) is an eigenpair with an infinite simple eigenvalue. The same type of analysis as before estimates the remaining eigenpairs. From eq. (6.2), it follows that

$$\begin{bmatrix} A^TA & A^Tb \\ b^TA & b^Tb \end{bmatrix} \begin{bmatrix} u_1 \\ u_{n+1} \end{bmatrix} = 2\gamma \begin{bmatrix} I_n & 0 \\ 0^T & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_{n+1} \end{bmatrix}$$

$$\Rightarrow \begin{cases} A^TAu_1 + A^Tbu_{n+1} = 2\gamma u_1 \\ b^TAu_1 + b^Tb\, u_{n+1} = 0 \end{cases}$$

$$\Rightarrow \begin{cases} A^TAu_1 - A^Tb\left(b^Tb\right)^{-1} b^TAu_1 = 2\gamma u_1 \\ u_{n+1} = -\left(b^Tb\right)^{-1} b^TAu_1 \end{cases}$$

$$\Rightarrow A^T\left(I_m - b\left(b^Tb\right)^{-1} b^T\right)Au_1 = 2\gamma u_1 \tag{6.13}$$

Equation (6.13) can be solved for $u_1 = 0$ (i.e., u parallel to e_{n+1}), as seen previously. To estimate the other eigenpairs, it can be derived from eq. (6.13) :

$$A^TP_b^{\perp}Au_1 = 2\gamma u_1$$

where $P_b^{\perp} = \left(I_m - b\left(b^Tb\right)^{-1} b^T\right)$ is the symmetric projection matrix ($P_b^{\perp 2} = P_b^{\perp}$) that projects the column space of A into the orthogonal complement of b. Hence,

$$A^TP_b^{\perp 2}Au_1 = 2\gamma u_1 \Rightarrow A^TP_b^{\perp T}P_b^{\perp}Au_1 = 2\gamma u_1$$
$$\Rightarrow \left(P_b^{\perp}A\right)^T\left(P_b^{\perp}A\right)u_1 = 2\gamma u_1 \tag{6.14}$$

It can be concluded that if the singular values and right singular vectors of matrix $P_b^{\perp}A$ are given by π_i and v_i, respectively, for $i = 1, \ldots, n$ (the singular values are sorted in decreasing order), the required eigenpairs of the DLS case are given by the n couples $\left(\frac{\pi_i}{2}, \begin{bmatrix} v_i \\ \rho_i \end{bmatrix}\right)$ where $\rho_i = -\left(b^Tb\right)^{-1} b^TAv_i$. It can be deduced that the first n components of the DLS eigenvectors are parallel to the v_i singular vectors. In particular, the right singular vector associated with the smallest singular value of $P_b^{\perp}A$ corresponds to the DLS solution (see [51]). Indeed, it corresponds to the last eigenpair, scaled by $\rho_{\min} = -1$.

6.1.2 General Case

In the case $0 < \zeta < 1$, R can be diagonalized by a D-orthogonal transformation [178]. Indeed,

$$D = Z^TZ \tag{6.15}$$

where

$$Z = \begin{bmatrix} \sqrt{2\zeta}\, I_n & 0 \\ 0^T & \sqrt{2\,(1-\zeta)} \end{bmatrix} \tag{6.16}$$

Define matrix K as

$$K = Z^{-T} R Z^{-1} = \frac{1}{2} \begin{bmatrix} \dfrac{A^T A}{\zeta} & \dfrac{A^T b}{\sqrt{\zeta\,(1-\zeta)}} \\ \dfrac{b^T A}{\sqrt{\zeta\,(1-\zeta)}} & \dfrac{b^T b}{1-\zeta} \end{bmatrix} = Z^{-T} \left[A;\, b\right]^T \left[A;\, b\right] Z^{-1}$$

$$= \left\| \left[A;\, b\right] Z^{-1} \right\|_2^2 = \left\| \left[\frac{A}{\sqrt{2\zeta}};\, \frac{b}{\sqrt{2\,(1-\zeta)}} \right] \right\|_2^2 \tag{6.17}$$

and the corresponding eigendecomposition is given by

$$V^T K V = \operatorname{diag}\left(\alpha_1, \ldots, \alpha_{n+1}\right) = D_\alpha \tag{6.18}$$

The eigenvector matrix Y is defined as

$$Y = \left[y_1, \ldots, y_{n+1}\right] = Z^{-1} V \tag{6.19}$$

and is D-orthogonal; that is,

$$Y^T D Y = I \tag{6.20}$$

Hence, (α_i, y_i) is an eigenpair of the symmetric positive definite pencil (R, D).

From a MCA (minor components analysis) point of view, the theory above can be reformulated as a Rayleigh quotient:

$$E_{\text{GeTLS}}(x) = \frac{1}{2} \frac{(Ax-b)^T (Ax-b)}{(1-\zeta) + \zeta x^T x} = \frac{u^T R u}{u^T D u} = \frac{u^T R u}{u^T Z^T Z u}$$

$$= \frac{v^T Z^{-T} R Z^{-1} v}{v^T v} = \frac{v^T K v}{v^T v} = E_{\text{MCA}}(v) \tag{6.21}$$

where

$$u = Z^{-1} v \tag{6.22}$$

The critical vectors v^c of E_{MCA} correspond to the columns of matrix V defined in eq. (6.18). Hence, eq. (6.22) corresponds to eq. (6.19). The associated eigenvalues α_i are the corresponding values of E_{GeTLS} From a theoretical point of view, this equivalence GeTLS MCA has very important implications; the MCA theory [30] can be used for explaining the GeTLS behavior for all values of the parameter ζ. From a numerical point of view, it means that like the MCA EXIN algorithm (neuron) [30] (basically, a gradient flow of the Rayleigh quotient),

an MCA algorithm can replace the GeTLS EXIN algorithm by using matrix K as autocorrelation matrix. The estimated minor component must be scaled by using eq. (6.22) and normalized by constraining the last component equal to -1. Choosing K as a autocorrelation matrix implies that the MCA EXIN neuron is fed by input vectors m_i, defined as

$$m_i = \left[\frac{a_i^T}{\sqrt{2\zeta}}, \frac{b_i}{\sqrt{2\,(1-\zeta)}} \right]^T \tag{6.23}$$

a_i being the column vector representing the ith row of matrix A. The MCA EXIN neuron whose input is preprocessed by means of eq. (6.23) is called the *GeMCA EXIN neuron*.

6.2 ANALYSIS OF MATRIX *K*

Matrix K can be decomposed [9, Prop. 2.8.3, p. 43] as

$$\begin{aligned} 2K &= \begin{bmatrix} I_n & 0 \\ \sqrt{\dfrac{\zeta}{1-\zeta}}\left(A^+ b\right)^T & 1 \end{bmatrix} \begin{bmatrix} \dfrac{A^T A}{\zeta} & 0 \\ 0^T & \dfrac{\|b_\perp\|_2^2}{1-\zeta} \end{bmatrix} \begin{bmatrix} I_n & \sqrt{\dfrac{\zeta}{1-\zeta}} A^+ b \\ 0^T & 1 \end{bmatrix} \\ &= \begin{bmatrix} I_n & 0 \\ \sqrt{\dfrac{\zeta}{1-\zeta}} x_{\mathrm{OLS}}^T & 1 \end{bmatrix} \begin{bmatrix} \dfrac{A^T A}{\zeta} & 0 \\ 0^T & \dfrac{\|b_\perp\|_2^2}{1-\zeta} \end{bmatrix} \begin{bmatrix} I_n & \sqrt{\dfrac{\zeta}{1-\zeta}} x_{\mathrm{OLS}} \\ 0^T & 1 \end{bmatrix} \\ &= S^T K_1 S \end{aligned} \tag{6.24}$$

where

$$b_\perp = P_A^\perp b = \left(I - A\left(A^T A\right)^{-1} A^T \right) b \tag{6.25}$$

represents the component of b orthogonal to the column space of A. Matrix $P_A^\perp$ is the corresponding orthogonal projection matrix. Then , as seen before, $\|b_\perp\|_2^2$ is the sum of squares of the OLS residuals. It can be deduced that matrix K is congruent to matrix K_1 and then, for Sylvester's theorem, it inherits the same inertia [i.e., K is positive semidefinite, which is also evident from eq. (6.17)]. This analysis also yields the value of the determinant of matrix K:

$$\det K = \frac{\|b_\perp\|_2^2}{2\,(1-\zeta)} \det\left(\frac{A^T A}{2\zeta}\right) = \frac{\|b_\perp\|_2^2}{2^{n+1}\zeta^n\,(1-\zeta)} \det\left(A^T A\right) \tag{6.26}$$

By looking at matrix K [see eq. (6.17)] and identifying the principal submatrix $A^T A/2\zeta$ by means of the interlacing theorem, it follows $\forall\ i = 1, \ldots, n$ that

$$\alpha_{i+1} \leq \frac{\sigma_i'^2}{2\zeta} \tag{6.27}$$

$$\alpha_1 \geq \frac{\sigma_1'^2}{2\zeta} \tag{6.28}$$

where α_i is the ith eigenvalue of matrix K [see eq.(6.18)] and $\sigma_i'^2$ is the ith eigenvalue of matrix $A^T A$ (eigenvalues are sorted in decreasing order). Hence,

$$\prod_{i=1}^{n} \alpha_{i+1} \leq \prod_{i=1}^{n} \frac{\sigma_i'^2}{2\zeta} = \frac{1}{2^n \zeta^n} \prod_{i=1}^{n} \sigma_i'^2 = \frac{\det\left(A^T A\right)}{2^n \zeta^n} \tag{6.29}$$

Hence,

$$\det(K) \leq \frac{\det\left(A^T A\right)}{2^n \zeta^n} \alpha_1 \tag{6.30}$$

By using eq. (6.26) it follows that

$$\frac{\|b_\perp\|_2^2}{2^{n+1} \zeta^n (1-\zeta)} \det\left(A^T A\right) \leq \frac{\det\left(A^T A\right)}{2^n \zeta^n} \alpha_1 \tag{6.31}$$

that is [recalling eq. (6.17)],

$$\alpha_1 \geq \frac{\|b_\perp\|_2^2}{2(1-\zeta)} = \frac{\gamma_{\text{OLS}}}{1-\zeta} \tag{6.32}$$

which is a lower bound for the eigenvector associated with the largest eigenvalue of K . The inequality (6.27) shows that all eigenvalues of K but the largest are bounded by $\sigma_1'^2/2\zeta$, which varies from infinity (OLS), just allowing the n infinite eigenvalues, as seen before, to $\sigma_1'^2/2$ for $\zeta = 1$ (DLS). In the latter case, only one eigenvalue of K is allowed to increase toward infinity (i.e., α_1). This is also apparent from the lower bound (6.32), which tends to infinity as ζ tends to 1 [the upper bound (6.28) tells nothing about the infinity for α_1]. This analysis confirms what has been said in Section 6.1.1.2.

Another manipulation of the formulas above yields a novel inequality. Inequalities (6.27) and (6.28) can be rewritten as

$$\alpha_i \geq \frac{\sigma_i'^2}{2\zeta} \qquad \forall\, i = 1, \ldots, n \tag{6.33}$$

which imply that

$$\prod_{i=1}^{n} \alpha_i \geq \prod_{i=1}^{n} \frac{\sigma_i'^2}{2\zeta} = \frac{1}{2^n \zeta^n} \prod_{i=1}^{n} \sigma_i'^2 = \frac{\det\left(A^T A\right)}{2^n \zeta^n} \tag{6.34}$$

The remaining procedure is the same as before:

$$\det(K) \geq \frac{\det\left(A^T A\right)}{2^n \zeta^n} \alpha_{n+1} \tag{6.35}$$

$$\frac{\|b_\perp\|_2^2}{2^{n+1}\zeta^n (1-\zeta)} \det\left(A^T A\right) \geq \frac{\det\left(A^T A\right)}{2^n \zeta^n} \alpha_{n+1} \tag{6.36}$$

$$\alpha_{n+1} \leq \frac{\|b_\perp\|_2^2}{2(1-\zeta)} = \frac{\gamma_{\text{OLS}}}{1-\zeta} \tag{6.37}$$

Considering also that

$$\frac{\sigma_i'^2}{2\zeta} \leq \alpha_i \leq \frac{\sigma_{i-1}'^2}{2\zeta} \qquad \forall\, i = 2, \ldots, n \tag{6.38}$$

it can be deduced that for ζ tending to zero (OLS), all the eigenvalues of matrix K but the smallest go to infinity. Inequality (6.37) shows that the smallest eigenvalue in the OLS case cannot exceed the finite upper bound γ_{OLS}. This study confirms the analysis in Section 6.1.1.1. Also, if $\|b_\perp\|_2 = 0$, which means that $b \in \text{range}(A)$, the upper bound (6.37) implies that $\alpha_{n+1} = 0$ (a compatible system of equations).

By looking again at the matrix K [see eq. (6.17)] and identifying the principal (scalar) submatrix $b^T b/2(1-\zeta)$, by means of a consequence of the interlacing theorem [199, Prob. 2, p. 225], we obtain

$$\alpha_{n+1} \leq \frac{b^T b}{2(1-\zeta)} \leq \alpha_1 \tag{6.39}$$

Hence, defining

$$\gamma_{\text{OLS}}^0 = E_{\text{OLS}}(0) = \tfrac{1}{2} b^T b$$

it follows that

$$\alpha_{n+1} \leq \frac{\gamma_{\text{OLS}}}{1-\zeta} \leq \frac{\gamma_{\text{OLS}}^0}{1-\zeta} \leq \alpha_1 \tag{6.40}$$

It can be concluded that $\gamma_{\text{OLS}}^0/(1-\zeta)$ is a better lower bound for α_1, while $\gamma_{\text{OLS}}/(1-\zeta)$ is a better upper bound for α_{n+1}.

6.2.1 Note on the Taylor Approximation of the Bounds of the Eigenvalues of *K*

The bounds for the eigenvalues of matrix K expressed by (6.38) can be approximated for $\zeta \in [0, 1]$ by a Taylor expansion of the first order:

$$\frac{\sigma_i'^2}{2\zeta} = \frac{\sigma_i'^2}{2}\left[1 + (1-\zeta) + o\left((1-\zeta)^2\right)\right] \tag{6.41}$$

which is a good approximation in a neighborhood of $\zeta = 1$ (DLS). For $\zeta = 1$ these bounds have the lowest values with respect to ζ (i.e., $\sigma_i'^2/2 \forall i$). Considering the change of variable $\mu = 1 - \zeta$, these bounds can be analyzed for increasing μ (from zero on); that is, for problems with decreasing ζ but still close to DLS:

$$\frac{\sigma_i'^2}{2}(1+\mu) \leq \alpha_i \leq \frac{\sigma_{i-1}'^2}{2}(1+\mu) \qquad \forall\, i = 2, \ldots, n \tag{6.42}$$

All bounds increase linearly with μ, which implies that all the eigenvalues of K but the smallest and largest ones[1] increase linearly with μ. However, the increase rate is not the same. Obviously, larger eigenvalues increase more than do smaller eigenvalues.

About the TLS case $\zeta = 0.5$, the bounds can be approximated as

$$\frac{\sigma_i'^2}{2\zeta} = \sigma_i'^2\left[1 - 2(\zeta - 0.5) + \mathrm{o}\left((\zeta - 0.5)^2\right)\right] \tag{6.43}$$

By using the substitution $\mu = 0.5 - \zeta$, the approximation around the TLS case becomes

$$\sigma_i'^2(1+2\mu) \leq \alpha_i \leq \sigma_{i-1}'^2(1+2\mu) \qquad \forall\, i = 2, \ldots, n \tag{6.44}$$

and the same analysis as for DLS can be repeated here in a neighborhood of $\zeta = 0.5$ (i.e., around $\mu = 0$).

6.3 ANALYSIS OF THE DERIVATIVE OF THE EIGENSYSTEM OF GeTLS EXIN

6.3.1 Eigenvalue Derivative

From eq. (6.2), repeated here:

$$Ru = \alpha Du \tag{6.45}$$

it follows, by deriving both sides with respect to ζ:

$$R\frac{\mathrm{d}u}{\mathrm{d}\zeta} = \frac{\mathrm{d}\alpha}{\mathrm{d}\zeta}Du + \alpha\frac{\mathrm{d}D}{\mathrm{d}\zeta}u + \alpha D\frac{\mathrm{d}u}{\mathrm{d}\zeta}$$

[1]This Taylor approximation analysis cannot take these two extreme cases into account because only one bound is given for each case.

$$
\begin{aligned}
u^T R \frac{\mathrm{d}u}{\mathrm{d}\zeta} &= u^T \frac{\mathrm{d}\alpha}{\mathrm{d}\zeta} Du + \alpha u^T \frac{\mathrm{d}D}{\mathrm{d}\zeta} u + \alpha u^T D \frac{\mathrm{d}u}{\mathrm{d}\zeta} \\
\left(u^T R - \alpha u^T D\right) \frac{\mathrm{d}u}{\mathrm{d}\zeta} &= u^T \frac{\mathrm{d}\alpha}{\mathrm{d}\zeta} Du + \alpha u^T \frac{\mathrm{d}D}{\mathrm{d}\zeta} u \\
0 &= u^T \frac{\mathrm{d}\alpha}{\mathrm{d}\zeta} Du + \alpha u^T \frac{\mathrm{d}D}{\mathrm{d}\zeta} u
\end{aligned}
\tag{6.46}
$$

Hence,

$$
\frac{\mathrm{d}\alpha}{\mathrm{d}\zeta} = -\alpha \frac{u^T (\mathrm{d}Du/\mathrm{d}\zeta) u}{u^T Du}
$$

Considering that

$$
\frac{\mathrm{d}D}{\mathrm{d}\zeta} = 2 \begin{bmatrix} I_n & 0 \\ 0^T & -1 \end{bmatrix} = 2J \tag{6.47}
$$

and

$$
u^T Du = 2(1-\zeta) + 2\zeta x^T x
$$

then

$$
\frac{\mathrm{d}\alpha}{\mathrm{d}\zeta} = -\alpha \frac{x^T x - 1}{(1-\zeta) + \zeta x^T x} \tag{6.48}
$$

This equation can be rewritten as

$$
\frac{\mathrm{d}\alpha}{\mathrm{d}\zeta} = -\alpha \frac{\Psi}{\zeta \Psi + 1} = -\alpha \phi(\Psi, \zeta)
$$

6.3.2 Eigenvector Derivative

From eq. (6.46) it follows for the ith eigenvector u_i associate with the eigenvalue α_i:

$$
(R - \alpha_i D) \frac{\mathrm{d}u_i}{\mathrm{d}\zeta} = \frac{\mathrm{d}\alpha_i}{\mathrm{d}\zeta} Du_i + \alpha_i \frac{\mathrm{d}D}{\mathrm{d}\zeta} u_i \tag{6.49}
$$

and

$$
\left(D^{-1} R - \alpha_i I_{n+1}\right) \frac{\mathrm{d}u_i}{\mathrm{d}\zeta} = \frac{\mathrm{d}\alpha_i}{\mathrm{d}\zeta} u_i + \alpha_i D^{-1} \frac{\mathrm{d}D}{\mathrm{d}\zeta} u_i
$$

Define

$$
H = D^{-1} R - \alpha_i I_{n+1}
$$

Then it follows that

$$\frac{\mathrm{d}u_i}{\mathrm{d}\zeta} = H^+ \left(\frac{\mathrm{d}\alpha_i}{\mathrm{d}\zeta} + \alpha_i D^{-1} \frac{\mathrm{d}D}{\mathrm{d}\zeta} \right) u_i \tag{6.50}$$

$$\begin{aligned} \frac{\mathrm{d}u_i}{\mathrm{d}\zeta} &= H^+ \alpha_i \left(\underbrace{-\frac{\overbrace{u_i^T \frac{\mathrm{d}D}{\mathrm{d}\zeta} u_i}^{r}}{u_i^T D u_i} I_{n+1} + D^{-1} \frac{\mathrm{d}D}{\mathrm{d}\zeta}}_{\rho} \right) u_i \\ &= H^+ \alpha_i \left(-r I_{n+1} + D^{-1} \frac{\mathrm{d}D}{\mathrm{d}\zeta} \right) u_i = H^+ \alpha_i \, \rho \, u_i \end{aligned} \tag{6.51}$$

Assuming that the eigenvalue decomposition of $D^{-1}R$ is equal to $UD_\alpha U^T$, it can be deduced that

$$\begin{aligned} H^+ &= U D_\alpha^+ U^T \\ &= U \ \mathrm{diag} \left(\frac{1}{\alpha_1 - \alpha_i}, \ldots, \underbrace{0}_{i\text{th position}}, \ldots, \frac{1}{\alpha_{n+1} - \alpha_i} \right) U^T \end{aligned}$$

Equation (6.47) yields

$$D^{-1} \frac{\mathrm{d}D}{\mathrm{d}\zeta} = 2D^{-1} J = \mathrm{diag} \left(\frac{I_n}{\zeta}, -\frac{1}{1-\zeta} \right)$$

Hence,

$$\rho = -r I_{n+1} + \mathrm{diag} \left(\frac{I_n}{\zeta}, -\frac{1}{1-\zeta} \right) = \mathrm{diag} \left(\left(\frac{1}{\zeta} - r \right) I_n, -\frac{1}{1-\zeta} - r \right)$$

Then eq. (6.50) becomes

$$\frac{\mathrm{d}u_i}{\mathrm{d}\zeta} = \alpha_i U D_\alpha^+ U^T \mathrm{diag} \left(\left(\frac{1}{\zeta} - r \right) I_n, -\frac{1}{1-\zeta} - r \right) u_i = \alpha_i U D_\alpha^+ F \tag{6.52}$$

To simplify eq. (6.52), the normalization of the eigenvectors must be taken into account. Indeed, the constraint of belonging to the TLS hyperplane determines

the last component, $u_i = [\,\widehat{u_i}^T \; -1\,]$, where $\widehat{u_i} \in \Re^n$. Hence,

$$
\begin{aligned}
F &= \begin{bmatrix} u_1^T \\ \vdots \\ u_{n+1}^T \end{bmatrix} \begin{bmatrix} \left(\frac{1}{\zeta} - r\right)\widehat{u_i} \\ \frac{1}{1-\zeta} + r \end{bmatrix} \\
&= \begin{bmatrix} -\left(\frac{1}{\zeta} - r\right) - \left(\frac{1}{1-\zeta} + r\right) \\ \vdots \\ \left(\frac{1}{\zeta} - r\right)\left(\|u_i\|_2^2 - 1\right) - \left(\frac{1}{1-\zeta} + r\right) \\ \vdots \\ -\left(\frac{1}{\zeta} - r\right) - \left(\frac{1}{1-\zeta} + r\right) \end{bmatrix} \\
&= \begin{bmatrix} -\frac{1}{\zeta} - \frac{1}{1-\zeta} \\ \vdots \\ \left(\frac{1}{\zeta} - r\right)\|u_i\|_2^2 - \frac{1}{\zeta} - \frac{1}{1-\zeta} \\ \vdots \\ -\frac{1}{\zeta} - \frac{1}{1-\zeta} \end{bmatrix} \\
&= \begin{bmatrix} -\eta \\ \vdots \\ \left(\frac{1}{\zeta} - r\right)\|u_i\|_2^2 - \eta \\ \vdots \\ -\eta \end{bmatrix}
\end{aligned}
$$

where

$$\eta = \frac{1}{\zeta} + \frac{1}{1-\zeta} = \frac{1}{\zeta\,(1-\zeta)}$$

Equation (6.52) then becomes

$$\frac{\mathrm{d}u_i}{\mathrm{d}\zeta} = U\,\alpha_i D_\alpha^+ F$$

$$= U \operatorname{diag}\left(\frac{\alpha_i}{\alpha_1 - \alpha_i}, \ldots, \underbrace{0}_{i\text{th position}}, \ldots, \frac{\alpha_i}{\alpha_{n+1} - \alpha_i}\right) \begin{bmatrix} -\eta \\ \vdots \\ \left(\frac{1}{\zeta} - r\right) \|u_i\|_2^2 - \eta \\ \vdots \\ -\eta \end{bmatrix}$$

$$= \begin{bmatrix} u_1 & \cdots & u_{n+1} \end{bmatrix} \begin{bmatrix} \eta \frac{\alpha_i}{\alpha_i - \alpha_1} \\ \vdots \\ 0 \\ \vdots \\ \eta \frac{\alpha_i}{\alpha_i - \alpha_{n+1}} \end{bmatrix}$$

So the following result has been proved:

$$\begin{aligned} \frac{\mathrm{d}u_i}{\mathrm{d}\zeta} &= \eta \left(\frac{\alpha_i}{\alpha_i - \alpha_1} u_1 + \cdots + \frac{\alpha_i}{\alpha_i - \alpha_{i-1}} u_{i-1} + \frac{\alpha_i}{\alpha_i - \alpha_{i+1}} u_{i+1} + \cdots \right. \\ &\quad \left. + \frac{\alpha_i}{\alpha_i - \alpha_{n+1}} u_{n+1} \right) \\ &= \frac{1}{\zeta\,(1-\zeta)} \sum_{j \neq i} \frac{\alpha_i}{\alpha_i - \alpha_j} u_j \end{aligned} \tag{6.53}$$

Defining $\tau_{f,e}$ as

$$\tau_{f,g} = \begin{cases} \dfrac{\alpha_g}{\alpha_g - \alpha_f} & \text{for } g \neq f \\ 0 & \text{otherwise} \end{cases} \tag{6.54}$$

eq. (6.53) can be written in a more compact form as

$$\frac{\mathrm{d}u_i}{\mathrm{d}\zeta} = \eta U \begin{bmatrix} \tau_{1,i} \\ \vdots \\ 0 \\ \vdots \\ \tau_{n+1,i} \end{bmatrix}$$

Using the following notation:

$$\frac{\mathrm{d}U}{\mathrm{d}\zeta} = \begin{bmatrix} \frac{\mathrm{d}u_1}{\mathrm{d}\zeta} & \cdots & \frac{\mathrm{d}u_{n+1}}{\mathrm{d}\zeta} \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & \tau_{1,2} & \cdots & \tau_{i,n+1} \\ \tau_{2,1} & 0 & \cdots & \tau_{2,n+1} \\ \vdots & \ddots & 0 & \vdots \\ \tau_{n+1,1} & \tau_{n+1,2} & \cdots & 0 \end{bmatrix}$$

the following main result is obtained:

$$\frac{\mathrm{d}U}{\mathrm{d}\zeta} = \eta UT \tag{6.55}$$

6.3.2.1 Analysis of Matrix T From definition (6.54) it is deduced that

$$\tau_{f,g} = -\frac{\alpha_g}{\alpha_f}\tau_{g,f} \tag{6.56}$$

6.4 RANK ONE ANALYSIS AROUND THE TLS SOLUTION

Consider the SVD of matrix $[A; b]$ given by USW^T and the corresponding decomposition of matrix R given by $W\Gamma W^T$, where $\Gamma = \mathrm{diag}\left(\sigma_1^2, \ldots, \sigma_{n+1}^2\right)$. It follows that

$$D^{-1}Ru = \gamma u \Rightarrow D^{-1}W\Gamma W^T u = \gamma u \tag{6.57}$$

Defining $n = W^T u$, we obtain

$$W^T D^{-1} W \Gamma n = \gamma n \tag{6.58}$$

that is, $\Xi n = \gamma n$, where

$$\Xi = W^T D^{-1} W \Gamma = W^T D^{-1} R W \tag{6.59}$$

Decomposing the orthonormal matrix W as $\begin{bmatrix} W_1 \\ w_{n+1}^T \end{bmatrix}$, where w_{n+1}^T is its last row, after some simple mathematics we obtain

$$\Xi = \frac{\Gamma}{2\zeta}\left(I_{n+1} - \frac{1-2\zeta}{1-\zeta}xy^T\right) \tag{6.60}$$

where

$$\begin{aligned} x &= \Gamma^{-1} w_{n+1} \\ y &= \Gamma\, w_{n+1} \end{aligned} \tag{6.61}$$

with the property $\mathrm{tr}\left(xy^T\right) = 1$, where tr represents "trace". Assuming that the matrix $\Gamma/2\zeta$ represents the eigenvalue matrix of the TLS problem [98] if $\zeta = 0.5$, eq. (4.13) represents how the eigenvalues of the GeTLS/GeMCA problem are changed with regard to TLS when ζ differs from 0.5. It changes in a relative way as a rank one matrix proportional to a function of ζ. In particular, using a first-order Taylor approximation around $\zeta = 0.5$, it follows that

$$\Xi \sim \frac{\Gamma}{2\zeta}\left(I_{n+1} - 4\,(\zeta - 0.5)\,xy^T\right) \tag{6.62}$$

which is in agreement with the analysis in Section 6.2.1.

6.5 GeMCA SPECTRA

After adding some other inequalities, we continue with the bounds for the GeMCA critical values for every ζ. From eq. (6.2):

$$Ru = \gamma Du \Rightarrow D^{-1}Ru = \gamma u \tag{6.63}$$

Because the matrix $D^{-1}R$ is the product of two positive semidefinite matrices for $0 < \zeta < 1$, it can be derived [199, Th. 7.10, p. 227] that

$$\begin{aligned} &\frac{\sigma_i^2}{2\,(1-\zeta)} \le \alpha_i \le \frac{\sigma_i^2}{2\zeta} && \text{for } \zeta \le 0.5 \\ &\frac{\sigma_i^2}{2\zeta} \le \alpha_i \le \frac{\sigma_i^2}{2\,(1-\zeta)} && \text{for } \zeta \ge 0.5 \end{aligned} \tag{6.64}$$

σ_i^2 being the ith eigenvalue of matrix R (eigenvalues are sorted in decreasing order). Another possible inequality can be deduced by eq. (6.2):

$$\sigma_{n+1}^2 \le b^T b \le \sigma_1^2 \tag{6.65}$$

because $b^T b$ is a principal submatrix of R. Using all the bounds seen previously, it is possible to consider all the relative positions of the GeMCA critical values α_i for every possible value of the parameter ζ. Recall [30] that α_1 and α_{n+1} represent the maximum and the minimum of E_{GeTLS}, respectively. The other eigenvalues correspond to saddles.

6.5.1 Analysis of the Saddles

For each i such that $1 < i < n + 1$, define the *gaps*

$$g_i^- = \frac{\sigma_i^2}{\sigma_i'^2}$$

$$g_i^+ = \frac{\sigma_{i-1}'^2}{\sigma_i^2}$$

which are greater than or equal to 1. By assuming that σ_i^2 is an eigenvalue of R, the gaps depend on $b^T b$. For instance, $g_i^+ \longrightarrow 0$ for $b \longrightarrow 0$.

For $\zeta \leq 0.5$, the following *strictest* inequalities derived from the previous theory are true for the intermediate eigenvalue α_i $(1 < i < n + 1)$ of matrix K:

$$\begin{aligned} &\underbrace{\frac{\sigma_i^2}{2\,(1-\zeta)}}_{l1} \leq \alpha_i \leq \underbrace{\frac{\sigma_i^2}{2\zeta}}_{u1} && \left(\zeta^- \leq \zeta \leq 0.5\right) \\ &\underbrace{\frac{\sigma_i'^2}{2\zeta}}_{l2} \leq \alpha_i \leq \underbrace{\frac{\sigma_i^2}{2\zeta}}_{u1} && \left(0 \leq \zeta \leq \zeta^-\right) \end{aligned} \tag{6.66}$$

The value ζ^- is defined here as the *left crossover* and is given by

$$\zeta^- = \frac{1}{1 + g_i^-} \tag{6.67}$$

It is positive and less than or equal to 0.5.

For $\zeta \to 0$ (OLS), the bounds $l2$ and $u1$ tend to infinity; as seen before, all these critical values are infinite.

For $\zeta \geq 0.5$, the *strictest* inequalities are given by

$$\begin{aligned} &\underbrace{\frac{\sigma_i^2}{2\zeta}}_{l3} \leq \alpha_i \leq \underbrace{\frac{\sigma_i^2}{2\,(1-\zeta)}}_{u2} && \left(0.5 \leq \zeta \leq \zeta^+\right) \\ &\underbrace{\frac{\sigma_i^2}{2\zeta}}_{l3} \leq \alpha_i \leq \underbrace{\frac{\sigma_{i-1}'^2}{2\zeta}}_{u3} && \left(\zeta^+ \leq \zeta \leq 1\right) \end{aligned} \tag{6.68}$$

The value ζ^+ is here defined as the *right crossover* and is given by

$$\zeta^+ = \frac{g_i^+}{1 + g_i^+} \tag{6.69}$$

It is positive and greater than or equal to 0.5. It is not greater than 1.

For $\zeta \to 1$ (DLS) the lower bound $l3$ and the upper bound $u3$ tend to finite values. It follows that for $\zeta = 1$,

$$\frac{\sigma_i^2}{2} \leq \alpha_i \leq \frac{\sigma_{i-i}'^2}{2} \tag{6.70}$$

which is a bound for the DLS saddle critical values.

6.5.1.1 Analysis of the Strictest Inequalities for Saddles In this section the interval between the bounds of the previous strictest inequalities is studied for $\zeta^- \leq \zeta \leq \zeta^+$ and, above all, in a neighborhood of $\zeta = 0.5$ (TLS).

Define ε such that

$$\zeta = 0.5 - \frac{\varepsilon}{2} \tag{6.71}$$

The distance Ψ between the bounds $l1$ and $u1$ ($\zeta^- \leq \zeta \leq 0.5$) is given by

$$\Psi = \frac{\sigma_i^2}{2\zeta} - \frac{\sigma_i^2}{2(1-\zeta)} = \frac{\sigma_i^2}{1-\varepsilon} - \frac{\sigma_i^2}{1+\varepsilon} = \Psi(\varepsilon) \tag{6.72}$$

It can be seen that $\Psi(0) = 0$. Indeed, for TLS, $\alpha_i = \sigma_i^2$. As a consequence, Ψ represents the absolute error in estimating α_i as σ_i^2. It follows that

$$\Psi = \frac{2\varepsilon}{1-\varepsilon^2}\sigma_i^2 \tag{6.73}$$

The same reasoning can be repeated for the distance Ψ between the bounds $l3$ and $u2$ (in this case $\varepsilon \leq 0$). Hence, in the general case $\zeta^- \leq \zeta \leq \zeta^+$ it follows that

$$\Psi = \frac{2|\varepsilon|}{1-\varepsilon^2}\sigma_i^2 \tag{6.74}$$

Around $\zeta = 0.5$ (TLS), eq. (6.74) is approximated by

$$\Psi = 2\,|\varepsilon|\,\sigma_i^2\left(1+\varepsilon^2+o\left(\varepsilon^4\right)\right) \tag{6.75}$$

Hence, as a first approximation, the distance Ψ increases linearly with ζ. Also, Ψ increases proportionally to σ_i^2. It means that the increase is larger for larger saddles (i.e., saddles close to the maximum eigenvalue). As a consequence, the strictest inequalities are the inequalities concerning saddles near the minimum.

Now consider the derivative of Ψ with regard to ε:

$$\frac{\mathrm{d}\Psi}{\mathrm{d}\varepsilon} = 2\frac{1+\varepsilon^2}{\left(1-\varepsilon^2\right)^2}\sigma_i^2 \tag{6.76}$$

This derivative is an odd function and is always positive. It means that the growth of the distance with respect to ε is the same on both the left and right of $\zeta = 0.5$. In particular, for $\zeta = 0.5$ (TLS),

$$\left.\frac{\mathrm{d}\Psi}{\mathrm{d}\varepsilon}\right|_{\mathrm{TLS}} = 2\sigma_i^2 \tag{6.77}$$

6.5.2 Analysis of the Maximum α_1

The analysis of the maximum α_1 is the same as in the case of saddles. However, some crossovers miss.

For $\zeta \leq 0.5$, the following *strictest* inequalities derived from the previous theory are true:

$$\begin{aligned} \frac{\gamma_{\mathrm{OLS}}^0}{1-\zeta} = \underbrace{\frac{b^T b}{2\,(1-\zeta)}}_{l3} \leq \underbrace{\frac{\sigma_1^2}{2\,(1-\zeta)}}_{l2} \leq \alpha_1 \leq \underbrace{\frac{\sigma_1^2}{2\zeta}}_{u1} \qquad & \left(\zeta^- \leq \zeta \leq 0.5\right) \\ \underbrace{\frac{\sigma_1'^2}{2\zeta}}_{l4} \leq \alpha_i \leq \underbrace{\frac{\sigma_1^2}{2\zeta}}_{u1} \qquad & \left(0 \leq \zeta \leq \zeta^-\right) \end{aligned} \tag{6.78}$$

For $\zeta \to 0$ (OLS) the lower bound $l4$ and the upper bound $u1$ tend to infinity; as seen before, $\alpha_1 \to \infty$. For $\zeta \geq 0.5$, there is no right crossover and the *strictest* inequality is given by

$$\underbrace{\frac{\sigma_1^2}{2\zeta}}_{l3} \leq \alpha_1 \leq \underbrace{\frac{\sigma_1^2}{2\,(1-\zeta)}}_{u2} \tag{6.79}$$

However, from the inequality (6.40) it follows that

$$\underbrace{\frac{\gamma_{\mathrm{OLS}}^0}{1-\zeta}}_{l} \leq \alpha_1 \tag{6.80}$$

For $\zeta \to 1$ (DLS), this lower bound tends to infinity, as seen previously. The same is true for the upper bound $u2$. Indeed, the maximum eigenvalue for DLS is infinite.

6.5.3 Analysis of the Minimum α_{n+1}

Also in the analysis of the minimum α_{n+1}, previous considerations for the maximum can be repeated. For $\zeta \leq 0.5$, there is no left crossover and the *strictest*

inequality is given by

$$\underbrace{\frac{\sigma_{n+1}^2}{2\,(1-\zeta)}}_{l1} \leq \alpha_{n+1} \leq \underbrace{\frac{\sigma_{n+1}^2}{2\zeta}}_{u1} \leq \underbrace{\frac{\sigma_n'^2}{2\zeta}}_{u2} \tag{6.81}$$

From inequality (6.40) it follows that

$$\alpha_{n+1} \leq \underbrace{\frac{\|b_\perp\|_2^2}{2\,(1-\zeta)}}_{u3} = \frac{\gamma_{\mathrm{OLS}}}{1-\zeta} \tag{6.82}$$

For $\zeta \to 0$ (OLS), the bounds $l1$ and $u3$ tend to finite values, while $u1$ and $u2$ tend to infinity. It follows that

$$\alpha_{n+1} \leq \frac{\|b_\perp\|_2^2}{2\,(1-\zeta)} \to \frac{\|b_\perp\|_2^2}{2} = \gamma_{\mathrm{OLS}} \tag{6.83}$$

From (6.81) we also derive

$$\frac{\sigma_{n+1}^2}{2} \leq \gamma_{\mathrm{OLS}} \tag{6.84}$$

For $\zeta \geq 0.5$, the following *strictest* inequalities derived from the previous theory are true:

$$\begin{aligned} &\underbrace{\frac{\sigma_{n+1}^2}{2\zeta}}_{l3} \leq \alpha_{n+1} \leq \underbrace{\frac{\sigma_{n+1}^2}{2\,(1-\zeta)}}_{u2} && \left(0.5 \leq \zeta \leq \zeta^+\right) \\ &\underbrace{\frac{\sigma_{n+1}^2}{2\zeta}}_{l3} \leq \alpha_{n+1} \leq \underbrace{\frac{\sigma_n'^2}{2\zeta}}_{u3} && \left(\zeta^+ \leq \zeta \leq 1\right) \end{aligned} \tag{6.85}$$

For $\zeta \to 1$ (DLS) the bounds $l3$ and $u3$ tend to finite values. Hence, the bound for the DLS smallest eigenvalue is given by

$$\frac{\sigma_{n+1}^2}{2} \leq \alpha_{n+1} \leq \frac{\sigma_n'^2}{2} \tag{6.86}$$

which shows that the bound (2.117) is also valid for $i = n + 1$. In other words, the bound (2.117) is valid for each finite eigenvalue of K.

6.6 QUALITATIVE ANALYSIS OF THE CRITICAL POINTS OF THE GeMCA EXIN ERROR FUNCTION

Figure 6.1 summarizes the previous analysis. For visualization purposes, the GeTLS problem is two-dimensional ($n = 2$). The components of vector u are u_1, u_2, and u_3. The plane normal to u_3 has been drawn, together with the TLS hyperplane, whose equation is $u_3 = -1$, which contains the axes x_1 and x_2 for representation of the GeTLS solution x. All vectors point to the TLS hyperplane (the black dots), except the u axes and the dashed directions. All critical directions have been estimated by using eq. (6.21) and then scaled by means of eq. (6.22). The results for the three particular cases OLS, TLS, and DLS are shown. All minima (i.e., the GeTLS solutions) are shown. For DLS, the maximum is parallel to e_3 and corresponds to an infinite eigenvalue. For decreasing (decr.) ζ, the critical vector corresponding to a saddle or to a maximum tends to move toward a direction parallel to the TLS hyperplane (front plane in the figure) approaching the OLS case. For OLS only the minimum vector intersects the TLS hyperplane and so yields the corresponding OLS solution. The other critical directions correspond to infinite eigenvalues.

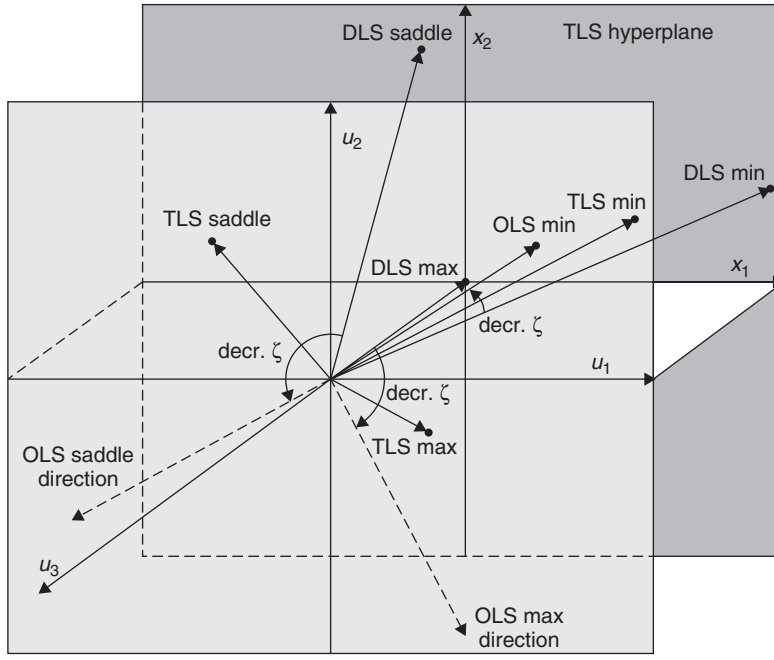


Figure 6.1 GeMCA EXIN and GeTLS EXIN critical directions and points.

6.7 CONCLUSIONS

In this chapter we have recalled the most important features of the GeTLS approach for linear parameter estimation when data are noisy. In particular, the noise variance estimation can be used for setting the corresponding parameter ζ. Hence, not only are OLS, TLS, and DLS important, but there is a need to analyze the behavior of the approach for intermediate values of ζ.

The most important contributions of this approach are the study of the error cost of GeTLS (5.6) as a generalized Rayleigh quotient and the consequent equivalence with a Rayleigh quotient (by means of matrix K), which allows the use of all the MCA theory (see chapter 2).

We included some novel inequalities that add to our understanding of how the critical values evolve by changing the parameter. They also confirm the analysis of the OLS, TLS, and DLS cases. In the last section we show qualitatively the previous theoretical results.

In summary: Unless the existing generalizations of the linear regression problems are valid only for the three cases OLS, TLS, and DLS, GeTLS endowed by GeMCA theory also yields meaningful results for intermediate values of the parameter ζ. This has very important implications when inferring statistical information from experimental data.

REFERENCES

1. T.J. Abatzoglou and J.M. Mendel. Constrained total least squares. *Proc. ICASSP*, pp. 1485–1488, 1987.
2. T.J. Abatzoglou, J.M. Mendel, and G.A. Harada. The constrained total least squares technique and its application to harmonic superresolution. *IEEE Trans. Signal Process*., 39(5):1070–1087, May 1991.
3. R.J. Adcock. A problem in least squares. *Analyst*, 5:53–54, 1878.
4. P. Baldi and K. Hornik. Neural networks and principal component analysis: learning from examples without local minima. *Neural Netw*., 2:52–58, 1989.
5. S. Barbarossa, E. Daddio, and G. Galati. Comparison of optimum and linear prediction technique for clutter cancellation. *IEE Proc. F*, 134:277–282, 1987.
6. E. Barnard. Optimization for training neural nets. *IEEE Trans. Neural Netw*., 3(2):232–240, 1992.
7. R. Battiti. Accelerated backpropagation learning: two optimization methods. *Complex Syst*., 3:331–342, 1989.
8. S. Beghelli, R.P. Guidorzi, and U. Soverini. Structural identification of multivariable systems by the eigenvector method. *Int. J. Control*, 46:671–678, 1987.
9. D.S. Bernstein. *Matrix Mathematics*. Princeton University Press, Princeton, NJ, 2005.
10. C.M. Bishop. *Neural Network for Pattern Recognition*. Clarendon Press, Oxford, UK, 1995.
11. N.K. Bose, H.C. Kim, and H.M. Valenzuela. Recursive total least squares algorithm for image reconstruction. *Multidimens. Syst. Signal Process*., 4:253–268, 1993.
12. R. Branham. Multivariate orthogonal regression in astronomy. *Celestial Mech. Dyn. Astron*., 61(3):239–251, Sept. 1995.
13. R.H. Byrd and J. Nocedal. A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM J. Numer. Anal*., 26(3):727–739, June 1989.
14. F. Chatelin. *Eigenvalues of Matrices*. Wiley, New York, 1993.
15. Y. Chauvin. Principal component analysis by gradient descent on a constrained linear Hebbian cell. *Proceedings of the IEEE International Conference on Neural Networks*, Washington, DC, Vol. I, pp. 373–380. IEEE Press, New York, 1989.

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

16. H. Chen and R.W. Liu. Adaptive distributed orthogonalization processing for principal component analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1992.
17. H. Chen and R.W. Liu. An on-line unsupervised learning machine for adaptive extraction. *IEEE Trans. Circuits and Syst. II*, 41:87–98, 1994.
18. T. Chen, S. Amari, and Q. Lin. A unified algorithm for principal and minor components extraction. *Neural Netw.*, 11:385–390, 1998.
19. W. Chen, J. Zhang, J. Ma, and B. Lu. Application of the total least squares method to the determination of preliminary orbit. *Chin. Astron. Astrophys.*, 30(4):431–436, Oct.–Dec. 2006.
20. C.L. Cheng and J.W. Van Ness. *Robust Errors-in-Variables Regression*. Technical Report 179. Programs in Mathematical Sciences. University of Texas, Dallas, TX, 1987.
21. A. Cichocki and R. Unbehauen. *Neural Networks for Optimization and Signal Processing*. Wiley, New York, 1993.
22. A. Cichocki and R. Unbehauen. Simplified neural networks for solving linear least squares and total least squares problems in real time. *IEEE Trans. Neural Netw.*, 5(6):910–923, Nov. 1994.
23. J.M. Cioffi. Limited-precision effects in adaptive filtering. *IEEE Trans. Circuits Syst.*, 34(7):821–833, July 1987.
24. G. Cirrincione. *A Neural Approach to the Structure from Motion Problem*. Ph.D. dissertation, LIS INPG Grenoble, Dec. 1998.
25. G. Cirrincione. TLS and constrained TLS neural networks in computer vision. *3rd International Workshop on TLS and Errors-in-Variables Modeling*, Leuven/Heverlee, Belgium, Aug. 27–29, 2001.
26. G. Cirrincione. Constrained total least squares for the essential/fundamental matrix estimation, a neural approach. *Int. J. Comput. Vision* (submitted in 2010).
27. G. Cirrincione and M. Cirrincione. Linear system identification using the TLS EXIN neuron. *NEURAP 98*, Marseille, France, Mar. 1998.
28. G. Cirrincione and M. Cirrincione. Robust total least squares by the nonlinear MCA EXIN neuron. *Proceedings of the IEEE International Joint Symposia on Intelligence and Systems*, Rockville, MD, pp. 295–300, May 1998.
29. G. Cirrincione and M. Cirrincione. A robust neural approach for the estimation of the essential parameters in computer vision. *Int. J. Artif. Intell. Tools*, 8(3), Sept. 1999.
30. G. Cirrincione, M. Cirrincione, J. Hérault, and S. Van Huffel. The MCA EXIN neuron for the minor component analysis. *IEEE Trans. Neural Netw.*, 13(1):160–187, Jan. 2002.
31. G. Cirrincione, M. Cirrincione, and S. Van Huffel. The GeTLS EXIN neuron for linear regression. *IJCNN*, July 2000.
32. G. Cirrincione, M. Cirrincione, and S. Van Huffel. The essential/fundamental matrix estimation as a constrained total least squares problem: Theory. *CVPRIP 2000*, Atlantic City, NJ, Jan.–Feb. 2000.
33. G. Cirrincione, M. Cirrincione, and S. Van Huffel. The GETLS EXIN neuron for linear regression. *IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference*, July 24–27, 2000. *Neural Netw.*, 6:285–289.

34. G. Cirrincione, M. Cirrincione, and S. Van Huffel. A robust neural approach against outliers for the structure from motion problem in computer vision. *Conference on Statistics with Deficient Data*, Munich, Germany, 2000.
35. G. Cirrincione, G. Ganesan, K.V.S. Hari, and S. Van Huffel. Direct and neural techniques for the data least squares problem. *MTNS 2000*, June 2000.
36. G. Cirrincione and M. Cirrincione. Linear system identification by using the TLS EXIN neuron. *Neurocomputing*, 28(1–3):53–74, Oct. 1999.
37. M. Cirrincione, M. Pucci, and G. Cirrincione. Estimation of the electrical parameters of an induction motor with the TLS EXIN neuron. *Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems*, Apr. 17–19, 2002, pp. I034–1 to I034-8.
38. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. A new TLS based MRAS speed estimation with adaptive integration for high performance induction machine drives. *Conference Record of the Industry Applications Conference, 38th IAS Annual Meeting*, Vol. 1, pp. 140–151, Oct. 12–16, 2003.
39. M. Cirrincione, M. Pucci, G. Cirrincione, and G.A. Capolino. An adaptive speed observer based on a new total least-squares neuron for induction machine drives. *Conference Record of the 2004 IEEE Industry Applications Conference, 2004. 39th IAS Annual Meeting*, Vol. 2, pp. 1350–1361, Oct. 3–7, 2004.
40. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. An enhanced neural MRAS sensorless technique based on minor-component-analysis for induction motor drives. *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005*, Vol. 4, pp. 1659–1666, June 20–23, 2005.
41. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. Sensorless control of induction motor drives by new linear neural techniques. *12th International Power Electronics and Motion Control Conference*, pp. 1820–1829, Aug. 2006.
42. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. A new experimental application of least-squares techniques for the estimation of the induction motor parameters. In *Conference Record of the Industry Applications Conference, 2002, 37th IAS Annual Meeting*, Vol. 2, pp. 1171–1180, Oct. 13–18, 2002.
43. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. A new experimental application of least-squares techniques for the estimation of the induction motor parameters. *IEEE Trans. Ind. Appl*., 39(5):1247–1256, Sept.–Oct. 2003.
44. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. A new TLS-based MRAS speed estimation with adaptive integration for high performance induction motor drives. *IEEE Trans. Ind. Appl*., 40(4):1116–1137, June–Aug. 2004.
45. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. An adaptive speed observer based on a new total least-squares neuron for induction machine drives. *IEEE Trans. Ind. Appl*., 42(1):89–104, Jan.–Feb. 2006.
46. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. Sensorless control of induction machines by a new neural algorithm: the TLS EXIN neuron. *IEEE Trans. Ind. Electron*., 54(1):127–149, Feb. 2007.
47. M. Cirrincione, M. Pucci, G. Cirrincione, and G.-A. Capolino. Sensorless control of induction motors by reduced order observer with MCA EXIN+ based adaptive speed estimation. *IEEE Trans. Ind. Electron*., 54(1):150–166, Feb. 2007.
48. P. Comon and G.H. Golub. Tracking a few extreme singular values and vectors in signal processing. *Proc. IEEE*, 78:1327–1343, 1990.

49. Y. Le Cun, P.Y. Simard, and B. Pearlmutter. Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors. In S.J. Hanson, J.D. Cowan, and C.L. Giles, eds., *Advances in Neural Information Processing Systems*, Vol. 5, pp. 156–163. Morgan Kaufmann, San Mateo, CA, 1993.
50. C.E. Davila. An efficient recursive total least squares algorithm for FIR adaptive filtering. *IEEE Trans. Signal Process*., 42:268–280, 1994.
51. R.D. Degroat and E. Dowling. The data least squares problem and channel equalization. *IEEE Trans. Signal Process*., 41(1):407–411, Jan. 1993.
52. A.J. Van der Veen and A. Paulraj. An analytical constant modulus algorithm. *IEEE Trans. Signal Process*., pp. 1136–1155, May 1996.
53. K.I. Diammantaras and S.Y. Kung. An unsupervised neural model for oriented principal component. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1049–1052, 1991.
54. G. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrica*, 1:211–218, 1936.
55. S.E. Fahlman. Faster-learning variations on back propagation: an empirical study. In D. Touretzky, G.E. Hinton, and T.J. Sejnowski, eds., *Proceedings of the 1988 Connectionist Models Summer School*, pp. 38–51. Morgan Kaufmann, San Mateo, CA, 1988.
56. D.Z. Feng, Z. Bao, and L.C. Jiao. Total least mean squares algorithm. *IEEE Trans. Signal Process*., 46(8):2122–2130, Aug. 1998.
57. K.V. Fernando and H. Nicholson. Identification of linear systems with input and output noise: the Koopmans–Levin method. *IEE Proc. D*, 132:30–36, 1985.
58. G.W. Fisher. Matrix analysis of metamorphic mineral assemblages and reactions. *Contrib. Mineral. Petrol*., 102:69–77, 1989.
59. R. Fletcher. *Practical Methods of Optimization*, 2nd ed. Wiley, New York, 1987.
60. W.A. Fuller. *Error Measurement Models*. Wiley, New York, 1987.
61. G.G. Cirrincione, S. Van Huffel, A. Premoli, and M.L. Rastello. An iteratively re-weighted total least-squares algorithm for different variances in observations and data. In V.P. Ciarlini, M.G. Cox, E. Filipe, F. Pavese, and D. Richter, eds., *Advanced Mathematical and Computational Tools in Metrology*, pp. 78–86. World Scientific, Hackensack, NJ, 2001.
62. P.P. Gallo. Consistency of regression estimates when some variables are subject to error. *Commun. Stat. Theory Methods*, 11:973–983, 1982.
63. K. Gao, M.O. Ahmad, and M.N. Swamy. Learning algorithm for total least-squares adaptive signal processing. *Electron. Lett*., 28(4):430–432, Feb. 1992.
64. K. Gao, M.O. Ahmad, and M.N. Swamy. A constrained anti-Hebbian learning algorithm for total least-squares estimation with applications to adaptive FIR and IIR filtering. *IEEE Trans. Circuits Syst.-II*, 41(11):718–729, Nov. 1994.
65. C.L. Giles and T. Maxwell. Learning, invariance, and generalization in high order neural networks. *Appl. Opt*., 26:4972–4978, 1987.
66. P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, New York, 1980.
67. R.D. Gitlin, J.E. Mazo, and M.G. Taylor. On the design of gradient algorithms for digitally implemented adaptive filters. *IEEE Trans. Circuits Syst*., 20, Mar. 1973.

68. R.D. Gitlin, H.C. Meadors, and S.B. Weinstein. The tap-leakage algorithm: an algorithm for the stable operation of a digitally implemented, fractionally adaptive spaced equalizer. *Bell Syst. Tech. J*., 61(8):1817–1839, Oct. 1982.
69. L.J. Gleser. Calculation and simulation in errors-in-variables regression problems. Mimeograph Ser. 78-5. Statistics Department, Purdue University, West Lafayette, IN, 1978.
70. L.J. Gleser. Estimation in a multivariate "errors in variables" regression model: large sample results. *Ann. Stat*., 9:24–44, 1981.
71. G.H. Golub. Some modified eigenvalue problems. *SIAM Rev*., 15:318–344, 1973.
72. G.H. Golub, A. Hoffman, and G.W. Stewart. A generalization of the Eckart–Young–Mirsky matrix approximation theorem. *Linear Algebra Appl*., 88–89: 322–327, 1987.
73. G.H. Golub and R.J. Leveque. Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems. *Proceedings of the 1979 Army Numerical Analysis and Computers Conference*, White Sands Missile Range, White Sands, NM, pp. 1–12, 1979.
74. G.H. Golub and C.F. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal*., pp. 883–893, 1980.
75. G.H. Golub and C.F. Van Loan. *Matrix Computations*, 2nd ed. Johns Hopkins University Press, Baltimore, 1989.
76. G.H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math*., 14:403–420, 1970.
77. J.W. Griffiths. Adaptive array processing, a tutorial. *IEE Proc. F*, 130:3–10, 1983.
78. R. Haimi-Cohen and A. Cohen. Gradient-type algorithms for partial singular value decomposition. *IEEE Trans. Pattern Anal. Machine Intell*., 9(1):137–142, Jan. 1987.
79. H.R. Hall and R.J. Bernhard. Total least squares solutions to acoustic boundary element models. In R.J. Bernhard and R.F. Keltie, eds., *Proceedings of the International Symposium on Numerical Techniques in Acoustic Radiation, ASME Winter Annual Meeting*, San Francisco, Vol. 6, pp. 145–152. American Society of Mechanical Engineers, New York, 1989.
80. F.R. Hampel, E.N. Ronchetti, P. Rousser, and W.A. Stachel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, New York, 1987.
81. M. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, MA, 1995.
82. S. Haykin. *Adaptive Filter Theory*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ, 1991.
83. S. Haykin. *Neural Networks: A Comprehensive Foundation*. IEEE Press, Piscataway, NJ, 1994.
84. U. Helmke and J.B. Moore. *Optimization and Dynamical Systems*. Communications and Control Engineering. Springer-Verlag, London, 1994.
85. M. Hestenes. *Conjugate Direction Methods in Optimization*. Springer-Verlag, New York, 1980.
86. M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand*., 49(6):409–436, 1952.
87. K. Hirakawa and T.W. Parks. Image denoising using total least squares. *IEEE Trans. Image Process*., 15:2730–2742, 2006.

88. R.R. Hocking. The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49, 1976.

89. R.R. Hocking. Developments in linear regression methodology 1959–1982. *Technometrics*, 25:219–230, 1983.

90. S.D. Hodges and P.G. Moore. Data uncertainties and least squares regression. *Appl. Stat*., 21:185–195, 1972.

91. P. Huber. *Robust Statistics*. Wiley, New York, 1981.

92. S. Van Huffel. *Analysis of the Total Least Squares Problem and Its Use in Parameter Estimation*. Ph.D. dissertation. Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1987.

93. S. Van Huffel. On the significance of nongeneric total least squares problems. *SIAM J. Matrix Anal. Appl*., 13(1):20–35, 1992.

94. S. Van Huffel. TLS applications in biomedical signal processing. In S. Van Huffel, ed., *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling*. SIAM Proceedings Series. SIAM, Philadelphia, 1997.

95. S. Van Huffel and J. Vandewalle. Subset selection using the total least squares approach in collinearity problems with errors in the variables. *Linear Algebra Appl*., 88–89:695–714, 1987.

96. S. Van Huffel and J. Vandewalle. The partial total least squares algorithm. *J. Comput. Appl. Math*., 21:333–341, 1988.

97. S. Van Huffel and J. Vandewalle. Analysis and properties of the generalized total least squares problem $ax = b$ when some or all columns of a are subject to errors. *SIAM J. Matrix Anal. Appl*., 10:294–315, 1989.

98. S. Van Huffel and J. Vandewalle. *The Total Least Squares Problems: Computational Aspects and Analysis*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 1991.

99. S. Van Huffel, J. Vandewalle, and A. Haegemans. An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values. *J. Comput. Appl. Math*., 19:313–330, 1987.

100. S. Van Huffel, J. Vandewalle, M.C. De Roo, and J.L. Willems. Reliable and efficient deconvolution technique based on total linear least squares for calculating the renal retention function. *Med. Biol. Eng. Comput*., 25:26–33, 1987.

101. R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Netw*., 1(4):295–307, 1988.

102. C.J. Tsai, N.P. Galatsanos, and A.K. Katsaggelos. Total least squares estimation of stereo optical flow. *Proceedings of the IEEE International Conference on Image Processing*, Vol. II, pp. 622–626, 1998.

103. E.M. Johannson, F.U. Dowla, and D.M. Goodman. Backpropagation learning for multilayer feedforward neural networks using the conjugate gradient method. *Int. J. Neural Syst*., 2(4):291–301, 1992.

104. J.H. Justice and A.A. Vassiliou. Diffraction tomography for geophysical monitoring of hydrocarbon reservoirs. *Proc. IEEE*, 78:711–722, 1990.

105. G. Kelly. The influence function in the errors in variables problem. *Ann. Stat*., 12:87–100, 1984.

106. R.H. Ketellapper. On estimating parameters in a simple linear errors-in-variables model. *Technometrics*, 25:43–47, 1983.

107. R. Klemm. Adaptive airborne MTI: an auxiliary channel approach. *IEE Proc. F*, 134:269–276, 1987.
108. T.C. Koopmans. *Linear Regression Analysis of Economic Time Series*. De Erver F. Bohn, Haarlem, The Netherlands, 1937.
109. A.H. Kramer and A. Sangiovanni-Vincentelli. Efficient parallel learning algorithms for neural networks. In D.S. Touretzky, ed., *Advances in Neural Information Processing Systems*, Vol. 1, pp. 40–48, Morgan Kaufmann, San Mateo, CA, 1989.
110. T.P. Krasulina. Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices. *Automat. Remote Control*, 2:215–221, 1970.
111. S.Y. Kung and K.I. Diammantaras. A neural network learning algorithm for adaptive principal component extraction. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 861–864, 1990.
112. S.Y. Kung, K.I. Diammantaras, and J.S. Taur. Adaptive principal component extraction and application. *IEEE Trans. Signal Process*., 42(5):1202–1217, 1994.
113. H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag New York, 1978.
114. C. Lawson and R. Hanson. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, NJ, 1974.
115. E. Linsker. From basic network principles to neural architecture: emergence of spatial opponent cells. *Proc. Natl. Acad. Sci. USA*, 83:7508–7512, 1986.
116. E. Linsker. From basic network principles to neural architecture: emergence of orientation selective cells. *Proc. Natl. Acad. Sci. USA*, 83:8390–8394, 1986.
117. E. Linsker. From basic network principles to neural architecture: emergence of orientation columns. *Proc. Natl. Acad. Sci. USA*, 83:8779–8783, 1986.
118. L. Ljung. Analysis of recursive stochastic algorithms. *IEEE Trans. Autom. Control*, 22:551–575, 1977.
119. D.G. Luenberger. *Linear and Nonlinear Programming*, 2nd ed. Addison-Wesley, Reading, MA, 1984.
120. F. Luo, Y. Li, and C. He. Neural network approach to the TLS linear prediction frequency estimation problem. *Neurocomputing*, 11:31–42, 1996.
121. F. Luo and R. Unbehauen. *Applied Neural Networks for Signal Processing*. Cambridge University Press, New York, 1997.
122. F. Luo and R. Unbehauen. A minor subspace analysis algorithm. *IEEE Trans. Neural Netw*., 8(5):1149–1155, Sept. 1997.
123. F. Luo and R. Unbehauen. Comments on: A unified algorithm for principal and minor components extraction. Letter to the Editor. *Neural Netw*., 12:393, 1999.
124. F. Luo, R. Unbehauen, and A. Cichocki. A minor component analysis algorithm. *Neural Netw*., 10(2):291–297, 1997.
125. F.L. Luo, R. Unbehauen, and Y.D. Li. A principal component analysis algorithm with invariant norm. *Neurocomputing*, 8(2):213–221, 1995.
126. A. Madansky. The fitting of straight lines when both variables are subject to error. *J. Am. Stat. Assoc*., 54:173–205, 1959.
127. S. Makram-Ebeid, J.A. Sirat, and J.R. Viala. A rationalized backpropagation learning algorithm. *Proceedings of the International Joint Conference on Neural Networks*, Vol. 2, pp. 373–380. IEEE Press, Piscataway, NJ, 1989.

128. G. Mathew and V. Reddy. Development and analysis of a neural network approach to Pisarenko's harmonic retrieval method. *IEEE Trans. Signal Process*., 42:663–667, 1994.

129. G. Mathew and V. Reddy. Orthogonal eigensubspace estimation using neural networks. *IEEE Trans. Signal Process*., 42:1803–1811, 1994.

130. V.Z. Mesarovic, N.P. Galatsanos, and A.K. Katsaggelos. Regularized constrained total least squares image restoration. *IEEE Trans. Image Process*., 48:1096–1108, Aug. 1995.

131. L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Q. J. Math. Oxford*, 11:50–59, 1960.

132. R.R. Mohler. *Nonlinear Systems*, Vol. I, *Dynamics and Control*. Prentice Hall, Englewood Cliffs, NJ, 1991.

133. M.F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw*., 6:525–533, 1993.

134. M. Moonen, B. De Moor, L. Vanderberghe, and J. Vandewalle. On- and off-line identification of linear state-space models. *Int. J. Control*, 49:219–232, 1989.

135. B. De Moor. *Mathematical Concepts and Techniques for Modelling Static and Dynamic Systems*. Ph.D. dissertation, Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1988.

136. B. De Moor and J. Vandewalle. An adaptive singular value decomposition algorithm based on generalized Chebyshev recursion. In T.S. Durrani, J.B. Abbiss, J.E. Hudson, R.W. Madan, J.G. McWhirter, and T.A. Moore, eds., *Proceedings of the Conference on Mathematics in Signal Processing*, pp. 607–635. Clarendon Press, Oxford, UK, 1987.

137. M. Muhlich and R. Mester. The role of total least squares in motion analysis. In H. Burkhardt, ed., *Proceedings of the European Conference on Computer Vision*, pp. 305–321. Lecture Notes on Computer Science. Springer-Verlag, New York, June 1998.

138. E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol*., 16:267–273, 1982.

139. E. Oja. *Subspace Methods of Pattern Recognition*. Research Studies Press/Wiley, Letchworth, UK, 1983.

140. E. Oja. Neural networks, principal components and subspace. *Int. J. Neural Syst*., 1:61–68, 1989.

141. E. Oja. Principal components, minor components and linear neural networks. *Neural Netw*., 5:927–935, 1992.

142. E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *J. Math. Anal. Appl*., 106:69–84, 1985.

143. E. Oja, H. Ogawa, and J. Wangviwattana. Learning in nonlinear constrained Hebbian networks. *Proceedings of the International Conference on ANNs*, Espoo, Finland, pp. 737–745, 1991.

144. E. Oja and L. Wang. Neural fitting: robustness by anti-Hebbian learning. *Neurocomputing*, 12:155–170, 1996.

145. E. Oja and L. Wang. Robust fitting by nonlinear neural units. *Neural Netw*., 9(3):435–444, 1996.

146. C.C. Paige and Z. Strakos. Unifying least squares, total least squares and data least squares problems. In S. Van Huffel and P. Lemmerling, eds., *Proceedings of the Third International Workshop on TLS and Errors-in-Variables Modelling*, pp. 35–44. Kluwer Academic, Dordrecht, The Netherlands. 2001.

147. C.C. Paige and Z. Strakos. Scaled total least squares fundamentals. *Numer. Math*., 91:117–146, 2002.

148. Y. Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, Reading, MA, 1989.

149. B.N. Parlett. The Rayleigh quotient iteration and some generalizations for nonormal matrices. *Math. Comput*., 28:679–693, 1974.

150. B.N. Parlett. *The Symmetric Eigenvalue Problem*. Englewood Cliffs, NJ, Prentice Hall, 1980.

151. K. Pearson. On lines and planes of closest fit to points in space. *Philos. Mag*., 2:559–572, 1901.

152. V.F. Pisarenko. The retrieval of harmonics from a covariance function. *Geophys. J. R. Astron. Soc*., 33:347–366, 1973.

153. D. Plaut, S. Nowlan, and G.E. Hinton. *Experiments on Learning by Backpropagation*. Technical report T.R. CMU-CS-86-126. Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1986.

154. E. Polak. *Computational Methods in Optimization: A Unified Approach*. Academic Press, New York, 1971.

155. M.J.D. Powell. Restart procedures for the conjugate gradient method. *Math. Programming*, 12:241–254, 1977.

156. A. Premoli, M.L. Rastello, and G. Cirrincione. A new approach to total least-squares techniques for metrological applications. In P. Ciarlini, M.G. Cox, F. Pavese, and D. Richter, eds., *Advanced Mathematical Tools in Metrology II*. Series on Advances in Mathematics for Applied Sciences, Vol. 40, pp. 206–215. World Scientific, Hackensack, NJ, 1996.

157. W.H. Press, S.A. Teukolsky, W.T. Wetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, New York, 1992.

158. M.A. Rahman and K.B. Yu. Total least squares approach for frequency estimation using linear prediction. *IEEE Trans. Acoust. Speech Signal Process*., 35:1440–1454, 1987.

159. J.A. Ramos. Applications of TLS and related methods in the environmental sciences. *Comput. Stat. Data Anal*., 52(2):1234–1267, 2007.

160. B.D. Rao. Unified treatment of LS, TLS and truncated SVD methods using a weighted TLS framework. In S. Van Huffel, ed., *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling*, pp. 11–20. SIAM, Philadelphia, 1997.

161. H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat*., 22:400–407, 1951.

162. R. Rost and J. Leuridan. A comparison of least squares and total least squares for multiple input estimation of frequency response functions. Paper 85-DET-105. *ASME Design Engineering Division Conference and Exhibit on Mechanical Vibration and Noise*, Cincinnati, OH, 1985.

163. R. Roy and T. Kailath. Total least-squares ESPRIT. *Proceedings of the 21st Annual Asilomar Conference on Signals, Systems and Computers*, pp. 297–301, 1987.

164. Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comput*., 42:567–588, 1984.

165. S. Sagara and K. Wada. On-line modified least-squares parameter estimation of linear discrete dynamic systems. *Int. J. Control*, 25(3):329–343, 1977.

166. T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Netw*., 2:459–473, 1989.

167. R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Trans. Antennas Propag*., 34:276–280, 1986.

168. H. Schneeweiss. Consistent estimation of a regression with errors in the variables. *Metrika*, 23:101–115, 1976.

169. D. Shafer. Gradient vectorfields near degenerate singularities. In Z. Nitecki and C. Robinson, eds., *Global Theory of Dynamical Systems*, pp. 410–417. Lecture Notes in Mathematics 819. Springer-Verlag, Berlin, 1980.

170. D.F. Shanno. Conjugate gradient methods with inexact searches. *Math. Oper. Res*., 3(3):244–256, 1978.

171. M.T. Silvia and E.C. Tacker. Regularization of Marchenko's integral equation by total least squares. *J. Acoust. Soc. Am*., 72:1202–1207, 1982.

172. J.J. Sllynk. Adaptive IIR filtering. *IEEE ASSP Mag*., 2:4–21, Apr. 1989.

173. H. Spath. On discrete linear orthogonal LP-approximation. *Z. Angew. Math. Mech*., 62:354–355, 1982.

174. H. Spath. Orthogonal least squares fitting with linear manifolds. *Numer. Math*., 48:441–445, 1986.

175. P. Sprent. *Models in Regression and Related Topics*. Methuen, London, 1969.

176. J. Staar. *Concepts for Reliable Modelling of Linear Systems with Application to On-Line Identification of Multivariable State Space Descriptions*. Ph.D. dissertation, Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1982.

177. G.W. Stewart. *Sensitivity Coefficients for the Effects of Errors in the Independent Variables in a Line Regression*. Technical Report TR-571. Department of Computer Science, University of Maryland, College Park, MD, 1977.

178. G.W. Stewart. *Matrix Algorithms*, Vol. II, *Eigensystems*. SIAM, Philadelphia, 2001.

179. A. Stoian, P. Stoica, and S. Van Huffel. *Comparative Performance Study of Least-Squares and Total Least-Squares Yule–Walker Estimates of Autoregressive Parameters*. Internal Report. Department of Automatic Control, Polytechnic Institute of Bucharest, Bucharest, Roumania, 1990.

180. W.J. Szajowski. The generation of correlated Weibull clutter for signal detection problem. *IEEE Trans. Aerosp. Electron. Syst*., 13(5):536–540, 1977.

181. A. Taleb and G. Cirrincione. Against the convergence of the minor component analysis neurons. *IEEE Trans. Neural Netw*., 10(1):207–210, Jan. 1999.

182. R.C. Thompson. Principal submatrices: IX. Interlacing inequalities for singular values of submatrices. *Linear Algebra Appl*., 5:1–12, 1972.

183. J. Treichler. *The Spectral Line Enhancer*. Ph.D. dissertation, Stanford University, Stanford, CA, May 1977.

184. G. Ungerboeck. Fractional tap-spacing equalizerand consequences for clock recovery in data modems. *IEEE Trans. Commun*., 24:856–864, Aug. 1976.

185. T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, and D.L. Alkon. Accelerating the convergence of the back-propagation method. *Biol. Cybernet*., 59:257–263, 1988.

186. L. Wang and J. Karhunen. A unified neural bigradient algorithm for robust PCA and MCA. *Int. J. Neural Syst*., 7:53–67, 1996.

187. R.L. Watrous. Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization. *Proceedings of the IEEE First International Conference on Neural Networks*, Vol. 2, pp. 619–627. IEEE, San Diego, CA, 1987.

188. G.A. Watson. Numerical methods for linear orthogonal LP approximation. *IMA J. Numer. Anal*., 2:275–287, 1982.

189. A.R. Webb, D. Lowe, and M.D. Bedworth. *A Comparison of Non-linear Optimisation Strategies for Feed-Forward Adaptive Layered Networks*. RSRE Memorandum 4157. Royal Signals and Radar Establishment, Malvern, UK, 1988.

190. J.T. Webster, R.F. Gunst, and R.L. Mason. Latent root regression analysis. *Technometrics*, 16:513–522, 1974.

191. B. Widrow and M.A. Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proc. IEEE*, 78(9):1415–1442, Sept. 1990.

192. B. Widrow and S.D. Stearns. *Adaptive Signal Processing*. Prentice Hall, Englewood Cliffs, NJ, 1985.

193. J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, UK, 1965.

194. L. Xu, A. Krzyzak, and E. Oja. Neural nets for dual subspace pattern recognition method. *Int. J. Neural Syst*., 2(3):169–184, 1991.

195. L. Xu, E. Oja, and C. Suen. Modified Hebbian learning for curve and surface fitting. *Neural Netw*., 5:441–457, 1992.

196. X. Yang, T.K. Sarkar, and E. Arvas. A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix. *IEEE Trans. Acoust., Speech Signal Process*., 37(10):1550–1556, Oct. 1989.

197. D. York. Least squares fitting of a straight line. *Can. J. Phys*., 44:1079–1086, 1966.

198. C.L. Zahm. Applications of adaptive arrays to suppress strong jammers in the presence of weak signals. *IEEE Trans. Aerosp. Electron. Syst*., 9:260–271, Mar. 1973.

199. F. Zhang. *Matrix Theory*. Springer-Verlag, New York, 1999.

200. W. Zhu, Y. Wang, Y. Yao, J. Chang, L. Graber, and R. Barbour. Iterative total least-squares image reconstruction algorithm for optical tomography by the conjugate gradient method. *J. Opt. Soc. Am. A*, 14(4):799–807, Apr. 1997.

201. M.D. Zoltowski. Signal processing applications of the method of total least squares. *Proceedings of the 21st Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, pp. 290–296, 1987.

202. M.D. Zoltowski and D. Stavrinides. Sensor array signal processing via a Procrustes rotations based eigenanalysis of the ESPRIT data pencil. *IEEE Trans. Acoust. Speech Signal Process*., 37:832–861, 1989.

INDEX

Neural-Based Orthogonal Data Fitting: The EXIN Neural Networks,
By Giansalvo Cirrincione and Maurizio Cirrincione

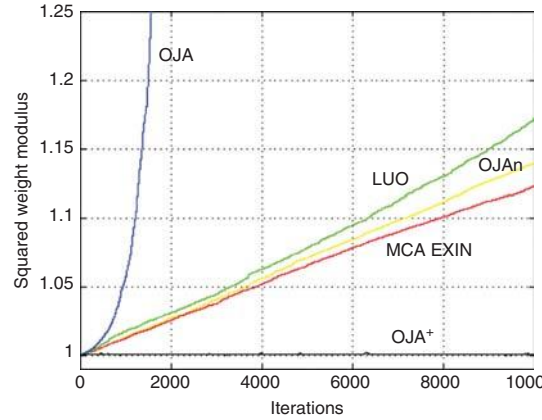


Figure 2.7 See page 60 for full caption.

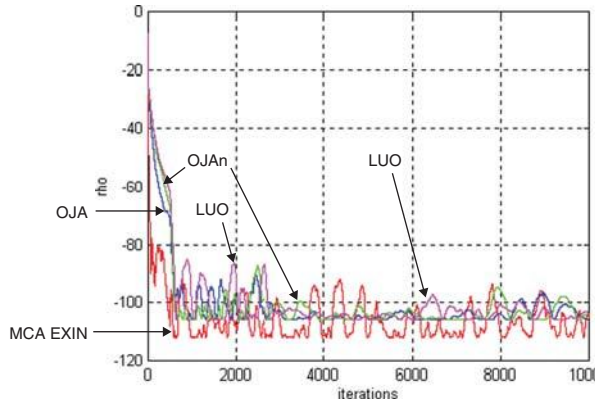


Figure 2.15 See page 79 for full caption.

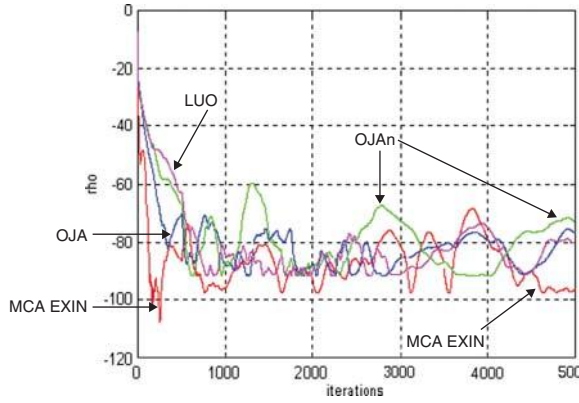


Figure 2.16 See page 80 for full caption.

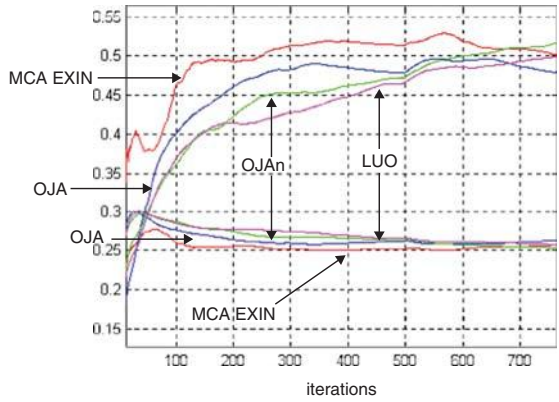


Figure 2.17 See page 81 for full caption.

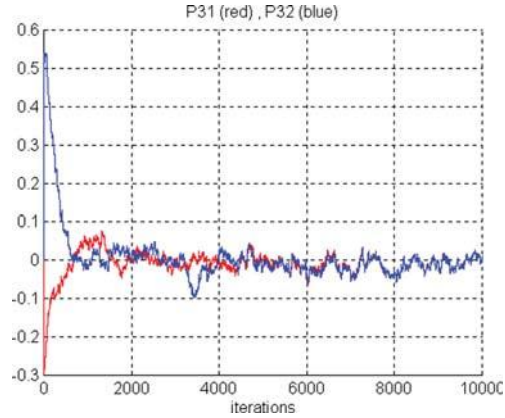


Figure 3.7 First example for MSA EXIN: P_{31} and P_{32}.

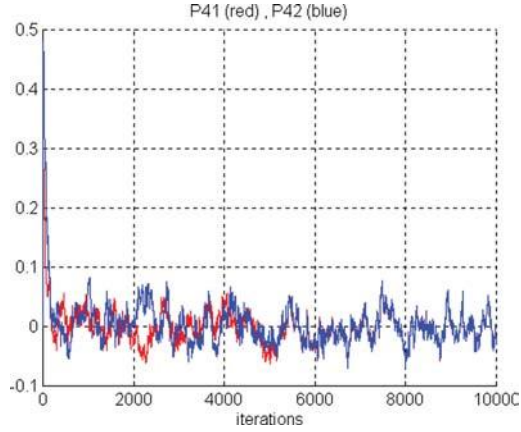


Figure 3.8 First example for MSA EXIN: P_{41} and P_{42}.

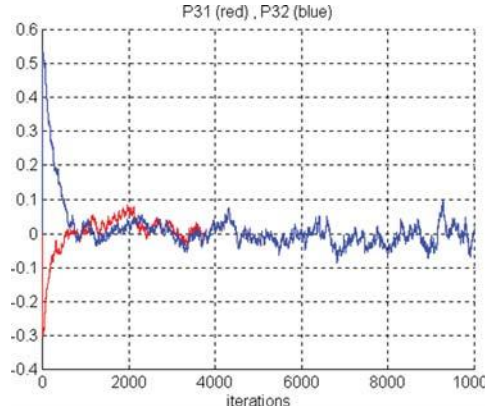


Figure 3.12 First example for MSA LUO: P_{31} and P_{32}.

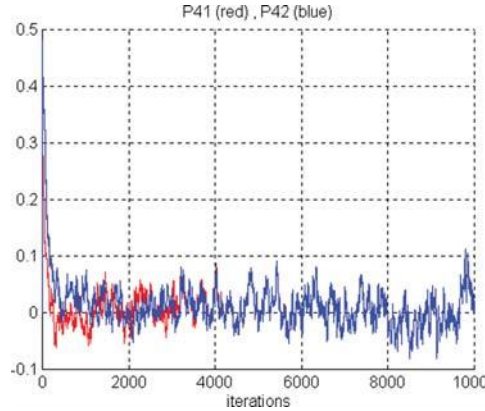


Figure 3.13 First example for MSA LUO: P_{41} and P_{42}.

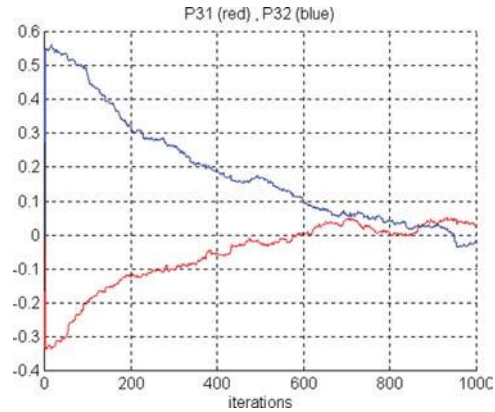


Figure 3.18 Second example for MSA EXIN: P_{31} and P_{32}.

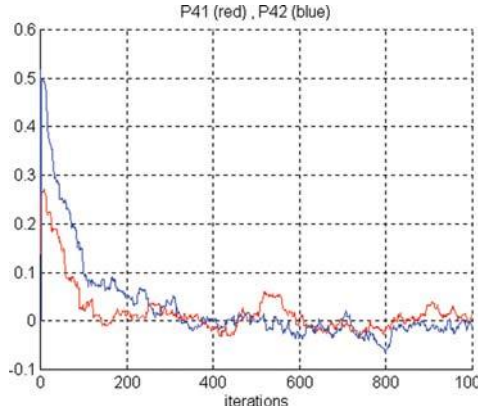


Figure 3.19 Second example for MSA EXIN: P_{41} and P_{42}.

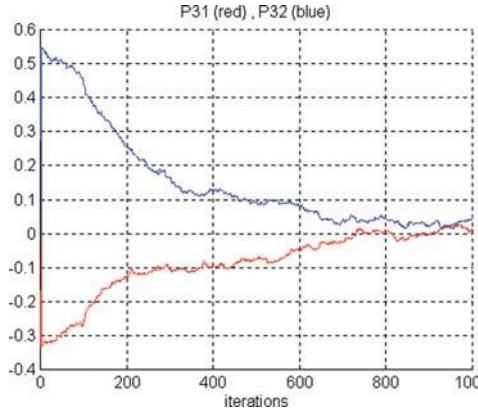


Figure 3.21 Second example for MSA LUO: P_{31} and P_{32}.

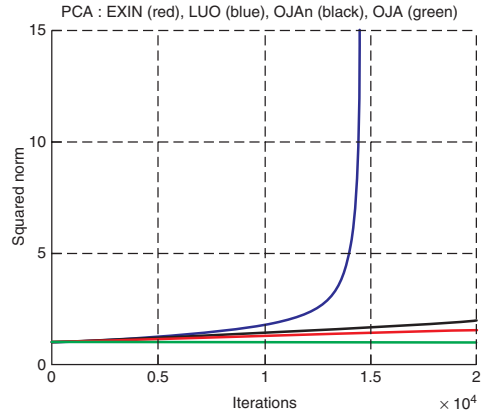


Figure 3.22 See page 114 for full caption.

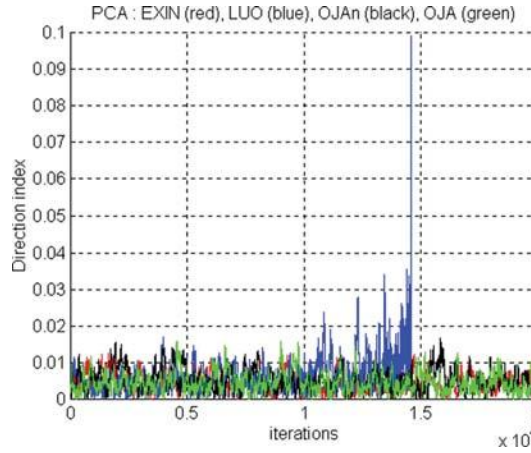


Figure 3.23 See page 115 for full caption.

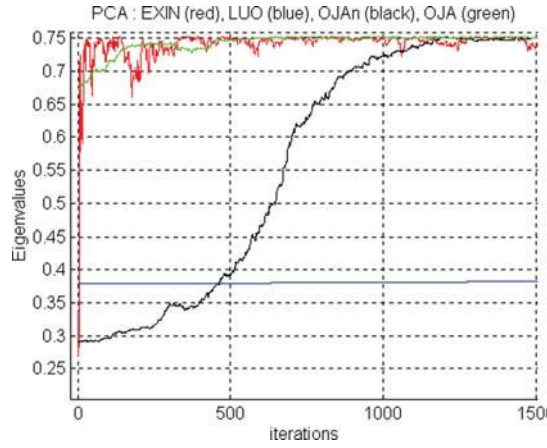


Figure 3.24 Computation of the maximum eigenvalue using PCA neurons.

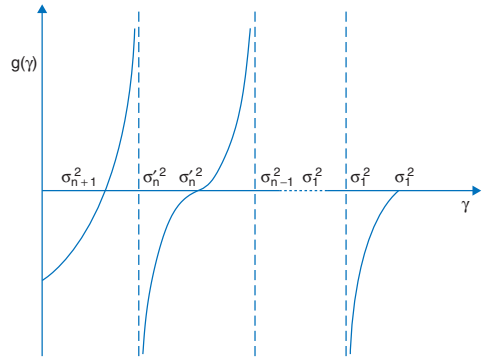


Figure 4.15 Plot of g as a function of the E_{TLS} level.

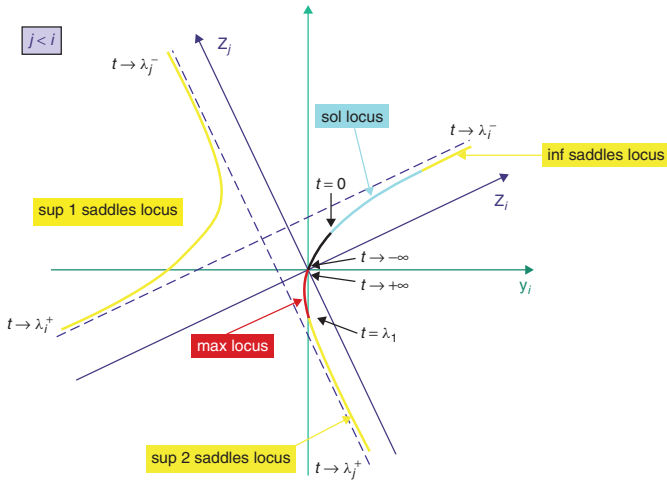


Figure 5.4 Critical loci in the plane $z_i z_j$.

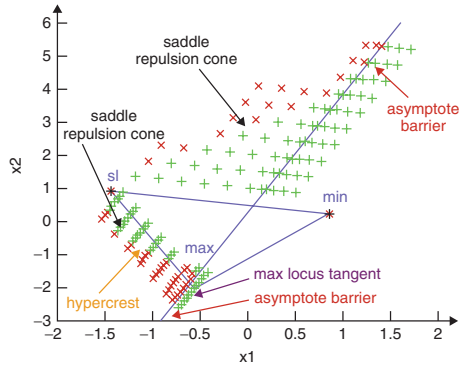


Figure 5.10 See page 174 for full caption.

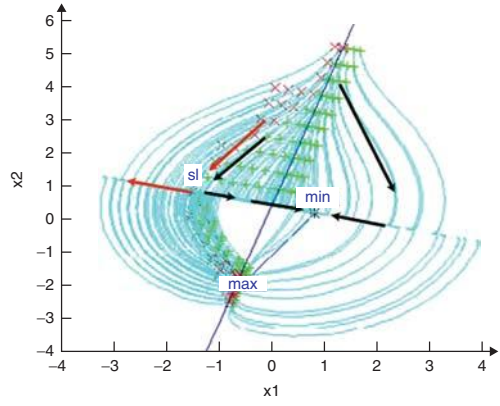


Figure 5.11 See page 175 for full caption.

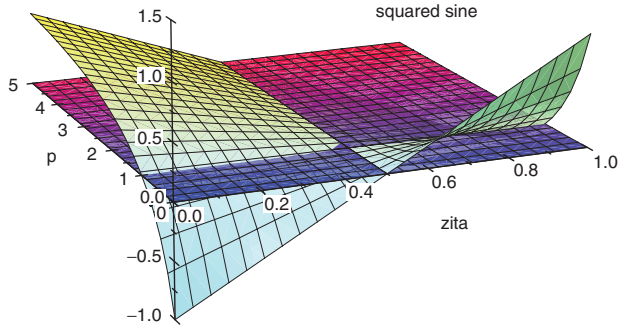


Figure 5.13 See page 177 for full caption.

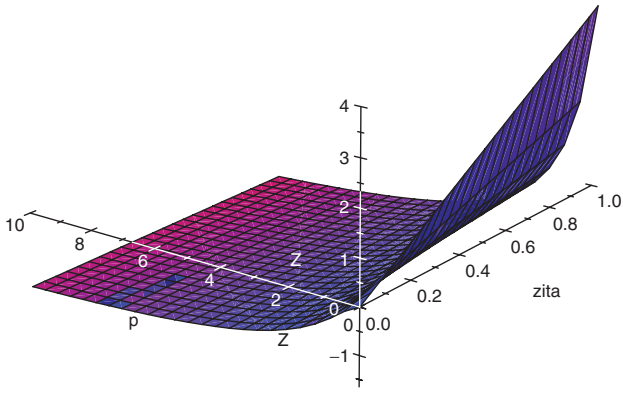


Figure 5.14 Plot of $\sin 2\alpha$.

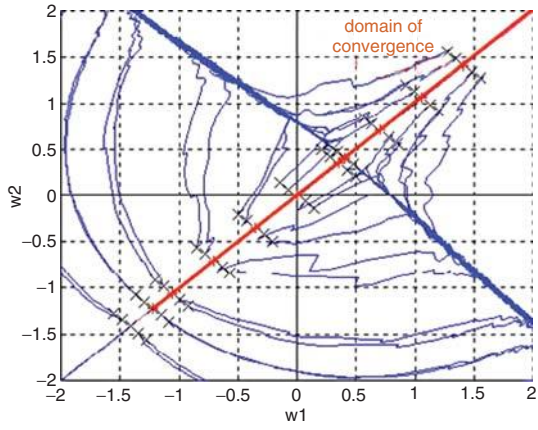


Figure 5.16 See page 182 for full caption.

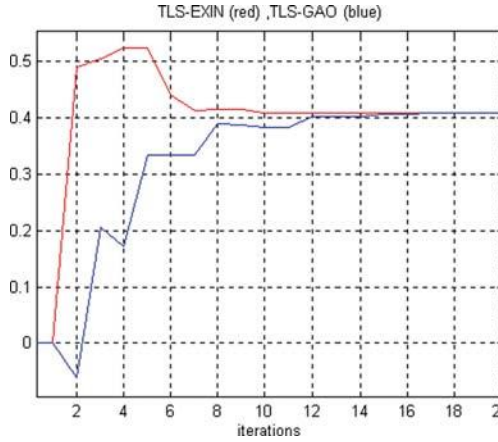


Figure 5.17 See page 182 for full caption.

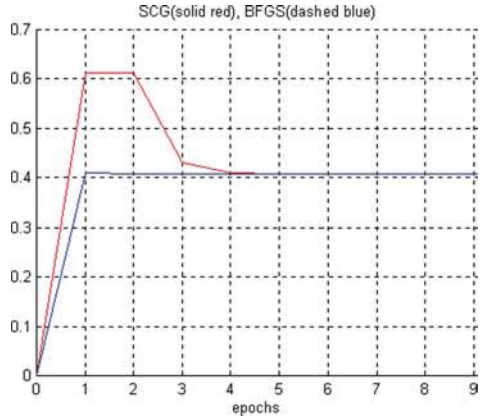


Figure 5.18 See page 183 for full caption.

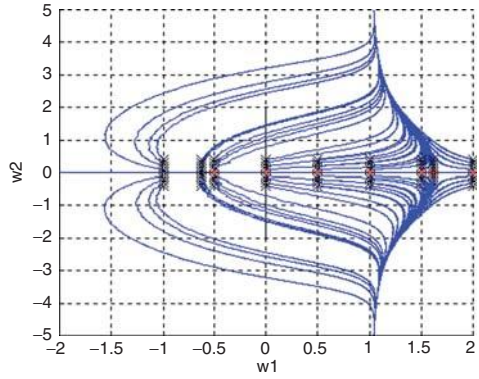


Figure 5.19 See page 185 for full caption.

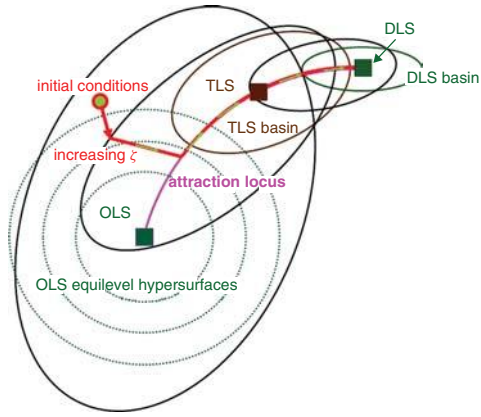


Figure 5.20 DLS scheduling.

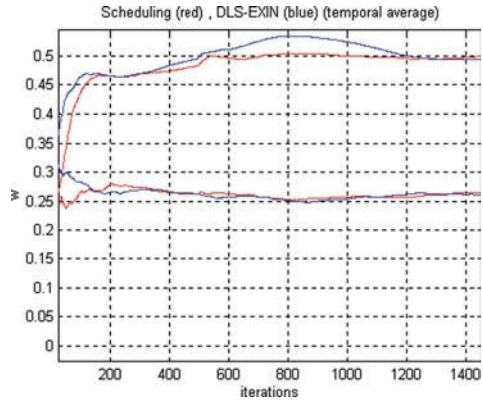


Figure 5.21 See page 189 for full caption.

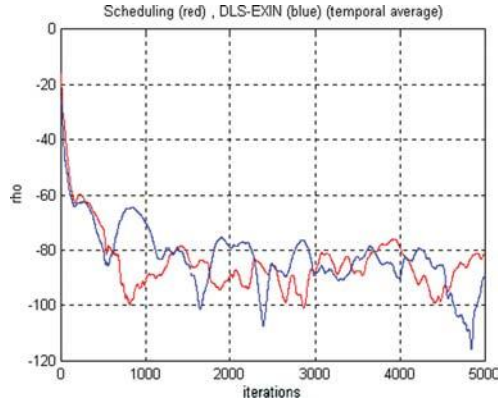


Figure 5.22 See page 190 for full caption.

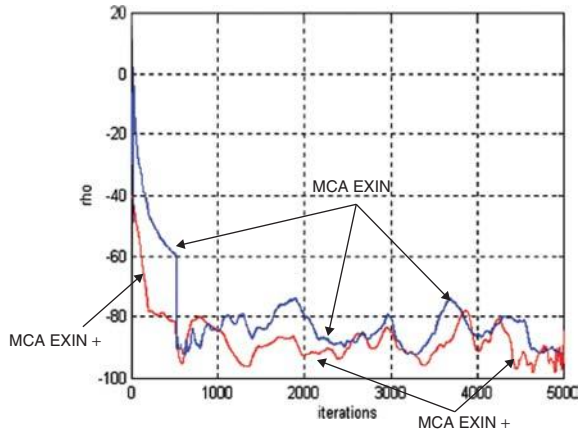


Figure 5.23 See page 191 for full caption.

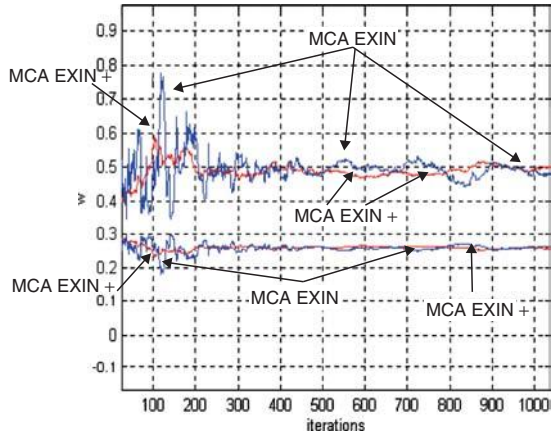


Figure 5.24 Line fitting for a noise variance of 0.5: transient for MCA EXIN and MCA EXIN+.

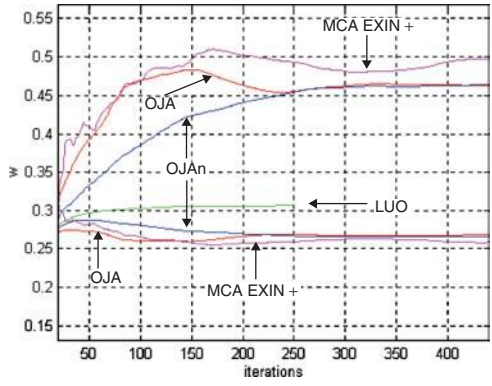


Figure 5.25 See page 192 for full caption.

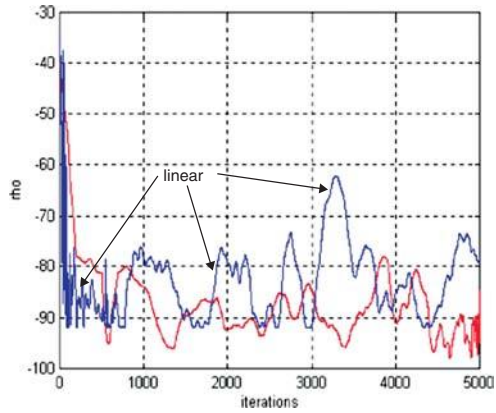


Figure 5.26 See page 193 for full caption.

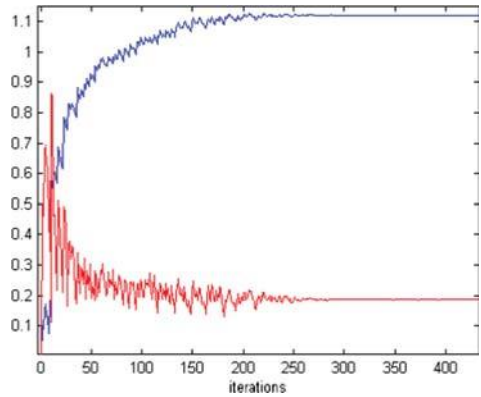


Figure 5.28 See page 200 for full caption.

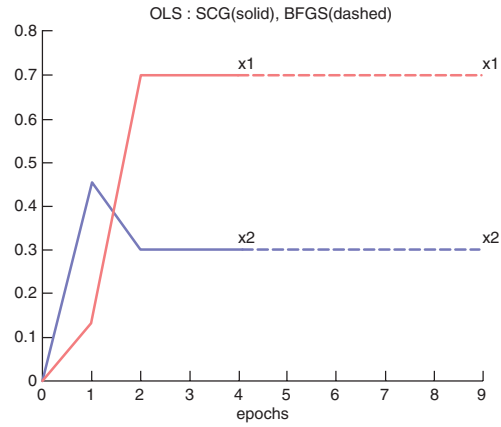


Figure 5.30 Plot of the weights of SCG and BFGS EXIN for the OLS benchmark problem.

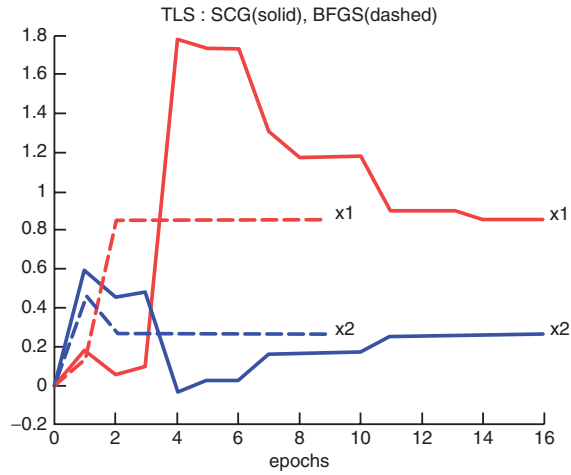


Figure 5.31 Plot of the weights of SCG and BFGS EXIN for the TLS benchmark problem.

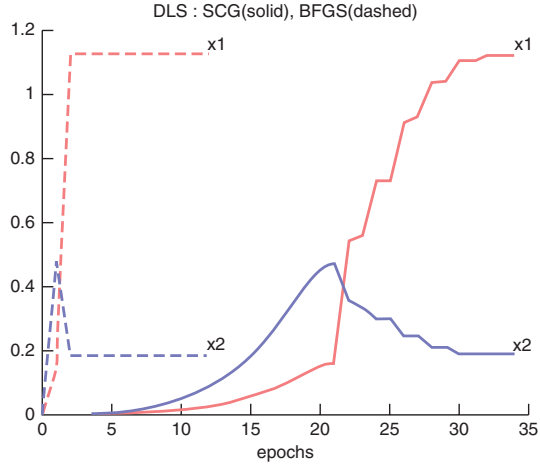


Figure 5.32 Plot of the weights of SCG and BFGS EXIN for the DLS benchmark problem.

Adaptive and Learning Systems for Signal Processing, Communication, and Control

Editor: Simon Haykin

Adali and Haykin / ADAPTIVE SIGNAL PROCESSING: Next Generation Solutions

Beckerman / ADAPTIVE COOPERATIVE SYSTEMS

Candy / BAYESIAN SIGNAL PROCESSING: CLASSICAL, MODERN, AND PARTICLE FILTERING METHODS

Candy / MODEL-BASED SIGNAL PROCESSING

Chen and Gu / CONTROL-ORIENTED SYSTEM IDENTIFICATION: An $\mathcal{H}_\infty$ Approach

Chen, Haykin, Eggermont, and Becker / CORRELATIVE LEARNING: A Basis for Brain and Adaptive Systems

Cherkassky and Mulier / LEARNING FROM DATA: Concepts, Theory, and Methods

Cirrincione and Cirrincione / NEURAL-BASED ORTHOGONAL DATA FITTING: The EXIN Neural Networks

Costa and Haykin / MULTIPLE-INPUT MULTIPLE-OUTPUT CHANNEL MODELS: Theory and Practice

Diamantaras and Kung / PRINCIPAL COMPONENT NEURAL NETWORKS: Theory and Applications

Farrell and Polycarpou / ADAPTIVE APPROXIMATION BASED CONTROL: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches

Gini and Rangaswamy / KNOWLEDGE-BASED RADAR DETECTION: Tracking and Classification

Hänsler and Schmidt / ACOUSTIC ECHO AND NOISE CONTROL: A Practical Approach

Haykin / UNSUPERVISED ADAPTIVE FILTERING: Blind Source Separation

Haykin / UNSUPERVISED ADAPTIVE FILTERING: Blind Deconvolution

Haykin and Puthussarypady / CHAOTIC DYNAMICS OF SEA CLUTTER

Haykin and Widrow / LEAST-MEAN-SQUARE ADAPTIVE FILTERS

Hrycej / NEUROCONTROL: Towards an Industrial Control Methodology

Hyvärinen, Karhunen, and Oja / INDEPENDENT COMPONENT ANALYSIS

Kristić, Kanellakopoulos, and Kokotović / NONLINEAR AND ADAPTIVE CONTROL DESIGN

Liu, Príncipe, and Haykin / KERNAL ADAPTIVE FILTERING

Mann / INTELLIGENT IMAGE PROCESSING

Nikias and Shao / SIGNAL PROCESSING WITH ALPHA-STABLE DISTRIBUTIONS AND APPLICATIONS

Passino and Burgess / STABILITY ANALYSIS OF DISCRETE EVENT SYSTEMS

Sánchez-Peña and Sznaier / ROBUST SYSTEMS THEORY AND APPLICATIONS

Sandberg, Lo, Fancourt, Principe, Katagiri, and Haykin / NONLINEAR DYNAMICAL SYSTEMS: Feedforward Neural Network Perspectives

Sellathurai and Haykin / SPACE-TIME LAYERED INFORMATION PROCESSING FOR WIRELESS COMMUNICATIONS

Spooner, Maggiore, Ordóñez, and Passino / STABLE ADAPTIVE CONTROL AND ESTIMATION FOR NONLINEAR SYSTEMS: Neural and Fuzzy Approximator Techniques

Tao / ADAPTIVE CONTROL DESIGN AND ANALYSIS

Tao and Kokotović / ADAPTIVE CONTROL OF SYSTEMS WITH ACTUATOR AND SENSOR NONLINEARITIES

Tsoukalas and Uhrig / FUZZY AND NEURAL APPROACHES IN ENGINEERING

Van Hulle / FAITHFUL REPRESENTATIONS AND TOPOGRAPHIC MAPS: From Distortion- to Information-Based Self-Organization

Vapnik / STATISTICAL LEARNING THEORY

Werbos / THE ROOTS OF BACKPROPAGATION: From Ordered Derivatives to Neural Networks and Political Forecasting

Yee and Haykin / REGULARIZED RADIAL BIAS FUNCTION NETWORKS: Theory and Applications