uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Maha Elsabrouty
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Riemannian Geometry Based Blind Signal Separation Using Independent Component Analysis

TITRE DE LA THÈSE / TITLE OF THESIS

Martin Bouchard
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Tyseer Aboulnasr
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Miodrag Bolic

Paul Fortier

Rafik Goubran

Wail Gueaieb

Gary W. Slater
LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

RIEMANNIAN GEOMETRY BASED BLIND SIGNAL SEPARATION USING INDEPENDENT COMPONENT ANALYSIS

By

Maha Elsabrouty

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
UNIVERSITY OF OTTAWA
OTTAWA, ONTARIO
OCTOBER 2005

Your file Votre référence
ISBN: 978-0-494-15019-1
Our file Notre référence
ISBN: 978-0-494-15019-1

To My mother

Table of Contents

Table of Contents ii

List of Tables vi

List of Figures viii

Abstract xiii

Acknowledgements xiv

Abbreviations xv

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Contributions and the Organization of the Thesis . . . 2

2 Unsupervised Adaptive Filtering: Theory and Algorithms 6
- 2.1 Introduction . . . 6
- 2.2 Blind Source Separation (BSS): Problem Definition and Dimensions . . . 6
- 2.3 Approaches to Blind Signal Separation . . . 8
  - 2.3.1 Higher Order Statistics (HOS): A Quick Look . . . 9
- 2.4 Preprocessing of Data: Principal Component Analysis (PCA), Why? . . . 10
- 2.5 Independent Component Analysis(ICA): Identifiability and Key Concepts . . . 11
- 2.6 Classification of Contrast Functions and Optimization Algorithms for ICA . . . 13
- 2.7 Contrast Functions for ICA . . . 14
  - 2.7.1 Information Theoretic-Based Contrast Functions . . . 14
  - 2.7.2 Extensions of Second-Order Methods . . . 21
- 2.8 Noisy ICA . . . 25
  - 2.8.1 Noisy ICA-Models . . . 25
  - 2.8.2 Modified-Algorithms for Noisy ICA . . . 25
  - 2.8.3 Bias-Removal . . . 26
- 2.9 ICA with Over-Complete Bases . . . 26
  - 2.9.1 Estimation of the Independent Components . . . 27
  - 2.9.2 Estimation of the Mixing Matrix for Over-complete Basis BSS . . . 27

2.10 BSS for Nonlinear Mixing Systems . . . 28
2.10.1 Special Case: Post Nonlinear Mixtures . . . 28
2.10.2 Nonlinear BSS Algorithms: Brief Review . . . 28
2.11 Signal Separation Using Time-Structure . . . 30
2.12 Convolutive Blind Signal Separation . . . 30
2.12.1 Approaches to Multi-Channel Convolutive Blind Signal Separation . . 31
2.13 Conclusion . . . 33

3 Understanding the Background of the Riemannian Geometry 34
3.1 Introduction . . . 34
3.2 Motivation . . . 34
3.3 Identifying the Riemannian Optimization Problem . . . 35
3.4 Riemannian Manifold: Definition and Properties . . . 37
3.5 Tangent and Normal Spaces . . . 37
3.6 The Concept of a Connection . . . 38
3.7 Geodesic: Definition and Properties . . . 41
3.8 Lie Group and Lie Algebra . . . 43
3.9 From Tangent Space to Geodesics: the Exponential Mapping . . . 46
3.10 Differentiation on Riemannian Manifold . . . 47
3.11 Natural Gradient Derivation for Maximum-Likelihood Principle . . . 49
3.12 Optimization along Geodesics . . . 50
3.13 Optimization along a Stiefel Manifold . . . 51
3.14 Newton Method along the Stiefel Manifold [31] . . . 53
3.15 Conjugate-Gradient Method along Stiefel Manifold [31] . . . 54
3.16 Conclusion . . . 56

4 Improvements to Instantaneous Blind Signal Separation Algorithms: Pre-Whitened Case 58
4.1 Introduction . . . 58
4.2 Existing Maximum-Likelihood Based Algorithms . . . 59
4.3 Existing Negentropy Based Algorithms and Relationship to Maximum-likelihood Based Update . . . 60
4.4 Nonlinear PCA and its Relationship With Maximum-Likelihood Principle and Negentropy Principle . . . 62
4.5 Bussgang Based Cost Function and its Stiefel Manifold Gradient . . . 63
4.6 RLS1-BSE-Stiefel Algorithm Based on Nonlinear PCA Principle . . . 64
4.7 The RLS2-BSE-Stiefel Algorithm Based on Negentropy . . . 68
4.7.1 The RLS2-BSE-Stiefel Algorithm (1) . . . 68
4.7.2 The RLS2-BSE-Stiefel Algorithm (2) . . . 69
4.7.3 Simulation Results . . . 70
4.8 Modified RLS2-BSE Algorithm . . . 71
4.9 The RLS-BSS-Stiefel Algorithm Based on Maximum-Likelihood . . . 73
4.9.1 Simulation Results . . . 76

4.10 Conclusion 77

5 Improvements to Instantaneous Blind Signal Separation Algorithms: Non-Prewhitened Case 79
5.1 Introduction 79
5.2 Nonholonomic Natural Gradient Update 80
5.3 A Proposed More General Update for Maximum-Likelihood Cost-Function . 81
5.4 Review of Iterative Inversion Approach 84
5.5 New Generalized Iterative Inversion of Blind Signal Separation Problem . . . 86
5.5.1 First LMS- based Algorithm: LMS-1 86
5.5.2 Second LMS-based Algorithm: LMS-2 89
5.5.3 Third LMS-based Algorithm: LMS-RLS 90
5.6 RLS-Iterative Inversion Approach 91
5.7 Simulation and Results 92
5.8 Non-Linear RLS Update for Natural Gradient 94
5.9 Second-Order Riemannian Algorithms for Maximum-Likelihood Blind Signal Separation 97
5.10 Reduced-Hessian Online Algorithm 106
5.11 Diagonal-Hessian Online Algorithm 112
5.12 Simulation Results for Second Order Algorithms 116
5.13 Application of Hessian-Based Algorithms to The Pre-Whitened Case 117
5.14 Revisiting the Natural Gradient Form on the Riemannian Space 120
5.15 Conclusion 121

6 Verification of the Proposed Algorithms: Simulation Results and Analysis 125
6.1 Introduction 125
6.2 Performance Evaluation of Instantaneous BSS Methods 126
6.3 Simulation Environment 127
6.4 Simulation Data 129
6.5 Simulation Scenarios 130
6.6 Benchmark Algorithms 131
6.7 Simulation Results for Prewhitened Mixtures Case (Stiefel Manifold) 132
6.7.1 Simulation Results for RLS1-BSE Algorithm 133
6.7.2 Simulation Results for Modified RLS2-BSE Algorithm 136
6.7.3 Simulation Results for RLS-BSS Algorithm 144
6.8 Simulation Results for Non Prewhitened Mixtures Case 149
6.8.1 Simulation Results for Generalized Natural Gradient Iterative Inversion Algorithms 149
6.8.2 Simulation Results for Nonlinear-RLS Algorithm 154
6.8.3 Simulation Results for Hessian-based Algorithms 161
6.8.4 Simulation Results for Hessian-Stiefel Algorithms 168
6.9 Comparison Between Nonlinear-RLS and the Hessian-based Algorithms for the General Riemannian Manifold 172

6.10 Comparison Between Nonlinear-RLS and the Hessian-based Algorithms for the Stiefel Manifold 175
6.11 Conclusion 177

7 Conclusion and Future Work **178**
7.1 Conclusion 178
7.2 Future Work 180

Bibliography **181**

List of Tables

2.1 The Fast ICA algorithm for maximum likelihood estimation. . . . 18
2.2 The Fast ICA algorithm for finding one maximally nongaussian direction. [56] 20
2.3 Modifications to the algorithms of the basic ICA to incorporate noise 26

3.1 Newton method for minimizing J(W) on Stiefel manifold [31] 54
3.2 Conjugate-gradient method for minimizing J(W) on Stiefel manifold [31] 56

4.1 RLS1-BSE algorithm 65
4.2 RLS1-BSE algorithm : 2^{nd} implementation 66
4.3 Modified RLS2-BSE algorithm 72
4.4 RLS-BSS-Stiefel algorithm 75
4.5 Summary of algorithms developed for BSS on the Stiefel manifold 78

5.1 Summary of algorithms developed for BSS on the Riemannian manifold 122
5.2 Diagonal Hessian-Stiefel on-line algorithm for Stiefel manifold . . . 123
5.3 Reduced Hessian-Stiefel on-line algorithm for Stiefel manifold . . . 124

6.1 RLS1-BSE algorithm : 2^{nd} implementation 134
6.2 Parameters for RLS1-BSE simulations 134
6.3 Modified RLS2-BSE algorithm 137
6.4 Parameters for modified RLS2-BSE simulations 137
6.5 Parameters for RLS-BSS-Stiefel 144
6.6 Parameters of the Generalized Iterative Inversion family 150
6.7 Parameters of the General Riemannian Nonlinear-RLS algorithm . 155

6.8 **Parameters of Hessian-based algorithms** 161
6.9 **Parameters of Hessian-Stiefel algorithms** 168

List of Figures

2.1 Block diagram of unknown environment resulting in mixing the sources s(n) into the mixtures x(n) [44] 7
2.2 Block diagram of system configured to recover the original sources from the mixtures [44] 7
2.3 Block diagram of a multi-channel blind deconvolution system [22] . 8
2.4 Summary of the basic approaches to blind signal separation along with the principal algorithms in each approach 24
2.5 A real room source separation scenario [77] 30

3.1 Example of tangent and normal spaces at a point on a Riemannian manifold [31] 38
3.2 Parallel transposing a parallel vector [31] 39
3.3 Geodesic on a plane: every straight line is a geodesic [32] 42
3.4 Geodesic on a sphere: geodesics on spheres are great circles [32] . 42
3.5 Exponential mapping: from tangent to geodesic [15] 47
3.6 Illustrating conjugate gradients [32] 55
3.7 Conjugate gradients in a curved space. [31] 55

4.1 Average performance of the RLS1-BSE-Stiefel compared to the Natural Gradient on Stiefel Manifold, RLS-NPCA, Negentropy and FAST-ICA 67
4.2 Average Performance of RLS2-BSE-Stiefel compared to the Natural Gradient on Stiefel manifold, Negentropy and FAST-ICA 71
4.3 RLS2-BSE-Stiefel vs. modified RLS2-BSE-Stiefel 71
4.4 Average results for RLS-BSS-Stiefel algorithm: $N = 4$ 76
4.5 Average results for RLS-BSS-Stiefel algorithm: $N = 10$ 77

5.1 Generalized Iterative Inversion Natural Gradient Family 92
5.2 Results for a sample run of the Natural-gradient family algorithms 94
5.3 Results for a sample run of the General Riemannian Nonlinear-RLS Algorithm 97
5.4 Results for a sample run of the Hessian-based algorithms 117
5.5 Results for a sample run of the Hessian-based algorithms for Stiefel manifold 120

6.1 Block diagram of instantaneous mixing/demixing situation 127
6.2 Block diagram of instantaneous mixing/demixing situation with pre-whitening 128
6.3 RLS1-BSE simulation results for easy mixing scene 135
6.4 RLS1-BSE simulation results for difficult mixing scene 135
6.5 RLS1-BSE average results for 100 different mixing situations 136
6.6 Modified RLS2-BSE simulation results for easy mixing scene: 1^{st} benchmark data 138
6.7 Modified RLS2-BSE simulation results for difficult mixing scene: 1^{st} benchmark data 139
6.8 Modified RLS2-BSE average results for 100 different mixing situations: 1^{st} benchmark data 139
6.9 Modified RLS2-BSE simulation results for easy mixing scene: 2^{nd} benchmark data 140
6.10 Modified RLS2-BSE simulation results for difficult mixing scene: 2^{nd} benchmark data 141
6.11 Modified RLS2-BSE average results for 100 different mixing situations: 2^{nd} benchmark data 141
6.12 Modified RLS2-BSE simulation results for easy mixing scene: 3^{rd} benchmark data 142
6.13 Modified RLS2-BSE simulation results for difficult mixing scene: 3^{rd} benchmark data 143
6.14 Modified RLS2-BSE average results for 100 different mixing situations: 3^{rd} benchmark data 143

6.15 RLS-BSS-Stiefel simulation results for easy mixing scene: 1^{st} benchmark data . . . 145
6.16 RLS-BSS-Stiefel simulation results for difficult mixing scene: 1^{st} benchmark data . . . 145
6.17 RLS-BSS-Stiefel average results for 100 different mixing situations: 1^{st} benchmark data . . . 146
6.18 RLS-BSS-Stiefel simulation results for easy mixing scene: 2^{nd} benchmark data . . . 147
6.19 RLS-BSS-Stiefel simulation results for difficult mixing scene: 2^{nd} benchmark data . . . 147
6.20 RLS-BSS-Stiefel average results for 100 different mixing situations: 2^{nd} benchmark data . . . 148
6.21 RLS-BSS-Stiefel simulation results for difficult mixing scene: 3^{rd} benchmark data . . . 148
6.22 RLS-BSS-Stiefel average results for 100 different mixing situations: 3^{rd} benchmark data . . . 149
6.23 Results for Generalized Iterative Inversion family: easy mixing scene 151
6.24 Results for Generalized Iterative Inversion family: difficult mixing scene . . . 152
6.25 Generalized Iterative Inversion Family, simulation results for 100 random mixing scenes . . . 152
6.26 Generalized-Iterative Inversion Family simulation results for 100 random mixing scenes, $N < M$ . . . 153
6.27 General Riemannian Nonlinear-RLS simulation results for easy mixing situation: 1^{st} benchmark data . . . 155
6.28 General Riemannian Nonlinear RLS simulation results for difficult mixing situation: 1^{st} benchmark data . . . 156
6.29 General Riemannian Nonlinear RLS average simulation results for 100 different mixing situations: 1^{st} benchmark data . . . 156
6.30 General Riemannian Nonlinear RLS simulation results for easy mixing situation: 2^{nd} benchmark data . . . 157
6.31 General Riemannian Nonlinear RLS simulation results for difficult mixing situation: 2^{nd} benchmark data . . . 158

6.32 General Riemannian Nonlinear RLS average simulation results for 100 different mixing situations: 2^{nd} benchmark data 158
6.33 General Riemannian Nonlinear RLS simulation results for easy mixing situation: 3^{rd} benchmark data 159
6.34 General Riemannian Nonlinear RLS simulation results for difficult mixing situation: 3^{rd} benchmark data 160
6.35 General Riemannian Nonlinear RLS average simulation results for 100 different mixing situations: 3^{rd} benchmark data 160
6.36 Hessian-based algorithms simulation results for easy mixing situation: 1^{st} benchmark data 162
6.37 Hessian-based algorithms simulation results for difficult mixing situation: 1^{st} benchmark data 163
6.38 Hessian-based algorithms average simulation results for 100 different mixing situations: 1^{st} benchmark data 163
6.39 Hessian-based algorithms simulation results for easy mixing situation: 2^{nd} benchmark data 164
6.40 Hessian-based algorithms simulation results for difficult mixing situation: 2^{nd} benchmark data 165
6.41 Hessian-based algorithms average simulation results for 100 different mixing situations: 2^{nd} benchmark data 165
6.42 Hessian-based algorithms simulation results for easy mixing situation: 3^{rd} benchmark data 166
6.43 Hessian-based algorithms simulation results for difficult mixing situation: 3^{rd} benchmark data 166
6.44 Hessian-based algorithms average simulation results for 100 different mixing situations: 3^{rd} benchmark data 167
6.45 Hessian-Stiefel algorithms simulation results for easy mixing situation: 1^{st} benchmark data 169
6.46 Hessian-Stiefel algorithms simulation results for difficult mixing situation: 1^{st} benchmark 169
6.47 Hessian-Stiefel algorithms average simulation results for 100 different mixing situations: 1^{st} benchmark data 170

6.48 **Hessian-Stiefel algorithms simulation results for easy mixing situation: 2^{nd} benchmark data** 171
6.49 **Hessian-Stiefel algorithms simulation results for difficult mixing situation: 2^{nd} benchmark data** 171
6.50 **Hessian-Stiefel algorithms average simulation results for 100 different mixing situations: 2^{nd} benchmark data** 172
6.51 **Hessian-based algorithms vs. General Riemannian Nonlinear-RLS algorithm: 1^{st} benchmark data** 173
6.52 **Hessian-based algorithms vs. General Riemannian Nonlinear-RLS algorithm: 2^{nd} benchmark data** 174
6.53 **Hessian-based algorithms vs. General Riemannian Nonlinear-RLS algorithm: 3^{rd} benchmark data** 175
6.54 **Hessian-based algorithms vs. RLS-BSS algorithm for Stiefel manifold: 1^{st} benchmark data** 176
6.55 **Hessian-based algorithms vs. RLS-BSS algorithm for Stiefel manifold: 2^{nd} benchmark data** 176

Abstract

Blind Source Separation is one of the newest and most active research areas in adaptive filtering. It represents the solution for many real situations in the audio, speech processing and telecommunication fields. The word "blind" reflects the fact that neither the source nor the mixing channel is known. This is, clearly, a more difficult situation compared to conventional adaptive filtering problems. Algorithms developed for blind separation reflect this difficulty. They possess a higher degree of sophistication compared with algorithms in other adaptive filtering approaches. The cost functions employed for blind separation are mostly based, implicitly or explicitly, on higher order statistics, while some of the adaptive algorithms developed for blind signal separation witnessed the introduction of the powerful principle of differential geometry to modify the LMS-based algorithms to what is known as the *natural-gradient* update.

The aim of this work is to generalize differential geometry algorithms for different cost functions of blind signal separation and provide new faster converging RLS-based and Newton-based algorithms using the natural gradient update. The thesis starts by providing a thorough review of the existing solutions developed in the literature for blind separation. This is followed by a study of the mathematical basis of Riemannian geometry and providing an engineering insight into the intrinsic geometry of curved spaces and its relation to optimization in adaptive filtering. Several new update algorithms are then developed throughout the thesis. They are structured to perform at faster convergence rates even in difficult mixing situations. These algorithms provide significantly improved performance in comparison with existing algorithms and are suitable for on-line applications.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Dr. Tyseer Aboulnasr and Dr. Martin Bouchard. Their vast knowledge and stimulating discussion made this research one of the most enjoyable experiences in my life. I owe them a lot for their technical feedback and their continuous support.

I also wish to thank my family. Their love, encouragement and confidence in me gave me the motivation and the eager to seek the best in my research.

Ottawa, Ontario — Maha Elsabrouty

October 13, 2005

Abbreviations

BSE	Blind Signal Extraction
BSS	Blind Signal Separation
BSP	Blind Signal Processing
CCI	Co-Channel Interference
EASI	Equivariant Adaptive Separation via Independence.
EEG	Electroencephalograms
FIR	Finite Impulse Response
GSM	Global System for Mobile communications
GTM	Generative Topographic Maps
EVD	Eigen Value Decomposition
HOS	Higher Order Statistics
IC(s)	Independent Components
ICA	Independent Components Analysis
IIR	Infinite Impulse Response
Infomax	Information Maximization
ISI	Inter Symbol Interference
JADE	Joint Approximate Diagonalization of Eigenmatrices.
LMS	Least Mean Square
MAP	Maximum A Posteriori
MEG	Magnetoencephalograms
MIMO	Multi-Input Multi-Output
ML	Maximum-Likelihood
MLP	Multi-Layer Perceptron
MMSE	Minimum Mean Square Error
MRI	Magnetic Resonance Imaging

NAG	Natural Gradient
PCA	Principle Components Analysis
RLS	Recursive Least Squares
SOM	Self Organizing Maps
SOS	Second Order Statistics
STFT	Short Time Fourier Transform

Chapter 1

Introduction

1.1 Motivation

For a long time, the signal processing and communication fields have relied on the classical model of adaptive filtering. The data received from a channel is processed by filters whose coefficients adaptation depends on the received data and a presumably known sequence of desired output. However, this classical setup started to face more obstacles recently. In many real life situations, it is difficult to access the sequence of desired output needed for the adaptation. In recent years, a new field in adaptive filtering has emerged to deal with the double challenge of filter adaptation and the absence of *the supervision* of a reference output. This new field is known as "unsupervised adaptive filtering" or "Blind Signal Processing". "Blind" reflects the fact that neither the source nor the channel is known. This interesting and rather new branch in signal processing can be traced to the early work of Jutten in 1987 [62]. However, the prosperity epoch for the field of blind processing has been the 90's till these days.
Considerable attention has been focused on this new trend in signal processing. It has a variety of potential applications and can be a real-life candidate replacement for the classical filtering if more research focus finds more accurate and simpler ways to approach the blind algorithms. The possible application domains of blind signal processing cover biomedical engineering, speech separation and enhancement, mobile communications, etc. Some of the current applications are listed below:

1. **Brain Imaging**: One of the new trends in studying the human brain is to use computerized X-ray tomography and Magnetic Resonance Imaging (MRI) to study the structure of the brain without surgery. Electroencephalograms (EEG) and magnetoencephalograms (MEG) study the functionality of the human brain and the activation of neurons. Blind Signal Separation (BSS) is the most eligible method to separate the other signals interfering with the pure brain signals (like muscle movement and other activations) that are not produced by the brain. BSS can also help identifying and localizing the signals to tell which parts of the brain have produced them. This area of applications for BSS is considered one of the most important. Many research centers are completely dedicated to the study of brain imaging applications of BSS [75], [22],

[87].

2. **Biomedical applications**: The same procedure used for brain imaging can be extended to other fields of biomedical signal processing. Studying the functionality of the muscles of the arms and determining which muscle is triggered for a certain function is an example of possible application. Separating signals generated by the fetus and the mother for better processing of the fetus condition is also an application [23]. Generally, better de-noising of the heart signals received on the instruments from other non-important signals can be achieved by BSS. Many of the external-anatomy (anatomy without surgical procedure) and separation of different biological mixed signals are future applications for BSS [22].

3. **Image Processing applications**: Blind techniques, especially Independent Components Analysis (ICA), which uses higher order statistics, can be used to enhance images. Applications like de-noising of images can perform better if the independent features (most basic features) of the image are known. Better coding is also feasible when redundancy is removed and only the most basic unique features are the ones to be coded [56].

4. **Acoustic applications**: Many of the acoustic set-ups require or function better when using blind techniques. Consider the cocktail-party problem, when several talkers talk at the same time and different sounds existing on the auditory scene are mixed together. In such cases, to recover the original speech and sound signals produced independently we must use blind models in which both the inputs and the filter coefficients are unknown. Speech enhancement and echo-cancellation are examples of the acoustic problems that can be better resolved when using the more appropriate model to the real scene, namely blind model [77].

5. **Telecommunications**: Recently, blind and semi-blind techniques have received special attention in the telecommunication area. Many of the classical models existing for problems like Inter-symbol Interference (ISI) and Co-Channel Interference (CCI) are based on the assumption that the channel is known or can be identified. Blind Signal Processing (BSP) offers a new way of dealing with such problems without explicitly defining the channel. The target of optimally designing equalizers and filters seems to be theoretically closer to this blind procedure. Along with the new advances in BSP, the introduction of convolutive blind techniques opens a new horizon of faithfully representing more difficult situations like dispersive channels [35]. This is also true for highly fading channels commonly encountered in the cellular communication domain.

1.2 Contributions and the Organization of the Thesis

The thesis aims at providing improvements to the existing techniques in the BSS domain. To start, we consider the two most reliable cost functions in the domain, namely Negentropy and Maximum-likelihood. These two techniques are based on solid theoretical background and are quite successful in the area of BSS. The target is to improve the performance of the

existing methods by introducing faster converging on-line algorithms with acceptable steady state error and a good robustness to difficult mixing conditions. The scientific contributions of the thesis include:

- The Bussgang criterion is applied to the instantaneous mixing case to provide a new cost function. This cost function is then examined along with other different cost functions on the pre-whitened optimization space, i.e. Stiefel manifold. A new insight on the equivalence of the first order gradients on the Stiefel manifold for all the cost functions examined is established.

- The problem of on-line instantaneous blind source extraction is considered in Chapter 4. The first algorithm developed employs pre-whitening of the data and a RLS-like update to perform the extraction. The RLS1-BSE-Stiefel algorithm, can achieve improved performance along with smaller steady state error.

- Another blind signal extraction algorithm is proposed that is capable of working with speech and audio signals, which are nonstationary signals that form mixtures difficult to separate. The proposed on-line algorithm, named the RLS2-BSE-Stiefel, also employs pre-whitening. This contribution is presented in publication [33]. The RLS2-BSE-Stiefel algorithm can be developed as the solution to the constrained optimization problem of the Negentropy cost function. This optimization problem can be solved using two different approaches. The first approach is through the Lagrange multiplier and the second approach is using the geometry of the Stiefel manifold. Both approaches lead to the same algorithm as presented in Chapter 4.

- Another contribution is the RLS-BSS-Stiefel on-line algorithm. This algorithm, presented also in chapter 4, is the third algorithm that employs pre-whitening. However, this algorithm is developed for blind signal separation, i.e. separating all the signals at once. It has been tested and proven to provide substantial improvement when compared with other on-line algorithms.

- Chapter 5 included numerous contributions mainly developed for the general mixing case that does not require pre-whitening. First, the iterative inversion approach to blind signal separation is generalized to provide a more flexible set of updates. The first algorithm based on the generalized iterative inversion approach is given the name LMS-1. The LMS-1 is particularly valuable thanks to its improved performance and robustness in the under-determined mixing case, i.e. when the number of sources is less than the number of mixtures. Two other algorithms are developed based on the generalization of the iterative inversion BSS approach. They are the LMS-2 and LMS-RLS algorithms and they provide fast convergence speed at affordable complexity.

- Another contribution in Chapter 5 is the RLS-Iterative algorithm. This proposed algorithm provides considerably improved performance both in speed of convergence and smaller steady state error compared to the other existing on-line algorithms at a low computational cost.

- The idea of nonlinear-RLS is successfully implemented for the general Riemannian manifold separation. The "General-Riemannian nonlinear-RLS" algorithm provides improved performance when tested with speech/audio benchmarks in various mixing/demixing situations.

- Another contribution presented in Chapter 5 is the derivation of the second order derivative for the Riemannian manifold. The Hessian matrix over the Lie-algebra of the Riemannian space is calculated along with approximations that can help reduce the computational load of the Hessian matrix.

- Based on the derivation of the Hessian matrix and using realistic approximations, a new 2^{nd}-order based algorithm is implemented. The algorithm, given the name "Reduced-Hessian" algorithm, proves to be of improved performance and efficient complexity.

- Another approximation of the Hessian matrix for the Riemannian manifold yields a new, more computationally optimized algorithm named "Diagonal-Hessian" algorithm. The algorithm is developed and tested in different mixing scenarios.

- Extensions of both the "Diagonal- Hessian" algorithm and the "Reduced-Hessian" algorithm to the pre-whitened mixtures case have also been developed.

- Validation and comparison of the proposed algorithms in this dissertation are established through simulations with different benchmarks in different mixing situation.

- Along with the above developed algorithms, the thesis also includes a literature survey on the topic of blind signal separation in Chapter2.

- Another literature summary on Riemannian optimization is presented in Chapter 3. The presentation of this rather challenging mathematical domain is intended to provide a simple overview of the differential optimization.

The organization of the thesis is as follows:

Chapter 2: Unsupervised Adaptive Filtering: Basic Principles

This chapter presents the basic mixing-demixing model of instantaneous mixtures along with the necessary conditions required for successful separation, based upon those separation criteria (conditions). The chapter begins with a review of the basic and most important cost functions developed for BSS. Then for each of the cost functions developed, a review of the associated suitable optimization algorithms that have reported success in the literature is presented.

This review is then extended from the basic mixing linear-instantaneous noise free case to more difficult mixing situations. The cost functions and algorithms developed for noisy-ICA, nonlinear-ICA and blind signal separation based on the time-structure of the mixtures are briefly discussed. Some practical considerations, mainly about preprocessing of data, are also considered.

The basic problem of instantaneous blind signal separation is then extended to the challenging scenario of convolutive mixtures. The setup of the problem is formed along with solutions that have been developed so far in the available literature. A

discussion of the advantages and problems associated with the current approaches is then presented.

Chapter 3: Understanding the Riemannian Geometry

The difficult nature of the blind signal separation problem has inspired active research for more mathematically efficient methods of optimization, with the hope of delivering improved accuracy of adaptation to compensate for the limited information provided to solve the problem. Embedding the information about the geometry of the optimization space in the optimization procedure has proved to be of paramount importance in improving the performance of the blind signal separation algorithms. The third chapter serves as a concise presentation of the most important concepts of the Riemannian geometry. The chapter also presents a derivation of a first order Riemannian-based algorithm that has successfully been used in BSS and also of new second order algorithms.

Chapter 4: Improvements to BSS Algorithms on the Stiefel Manifold

This chapter includes the proposed algorithms that have been developed for the pre-whitened mixtures case. The main goal is to improve the performance of selected algorithms that already exhibit some favorable characteristics i.e. being reliable and immune to noise. The improvement intended is to be produced by using the underlying Stiefel-manifold Riemannian-geometry nature previously presented in Chapter 3.
The work developed is compared to the original algorithms, with the improved performance demonstrated via simulations.

Chapter 5: Improvements to BSS Algorithms on the Riemannian Manifold

This chapter includes the proposed algorithms that have been developed for the non pre-whitened mixtures case. The aim of the work in this chapter is to provide algorithms with improved performance based on the maximum-likelihood cost function. The algorithms are based on three different approaches, namely iterative inversion, nonlinear RLS and second order Hessian-based approach on the Riemannian manifold. The work developed is compared to the natural gradient algorithm, with the improved performance demonstrated via simulations.

Chapter 6: Verification of the Proposed Algorithms

This chapter includes a complete collection of the simulations carried out for each of the algorithms. It starts by illustrating the simulation environment, conditions, performance index and benchmark data used in the simulation. Results for each of the algorithms developed in the thesis are presented for different mixing scenarios. Discussions of the results and comparisons of the performances of different algorithms are also presented.

Chapter 7: Conclusion and Future Work

This chapter provides a summary of the work completed in the thesis. The chapter also includes a presentation of possible future extensions of the developed algorithms and ideas that can add enhancements to the blind separation model.

Chapter 2

Unsupervised Adaptive Filtering: Theory and Algorithms

2.1 Introduction

This chapter presents a constructive review of the Blind Source Separation (BSS) problem and its extension to Blind Signal Extraction (BSE) and Blind deconvolution. This fundamental problem has many applications in signal processing and telecommunications. It has received considerable attention and developed a well-established platform of eligible solutions. One of the most powerful mathematical approaches for BSS is Independent Component Analysis (ICA). It is very common in literature to use both terms interchangeably. Actually, ICA is the main focus of this chapter. The review is then extended from the basic model to the more extendable complicated ones.
Adaptive filtering is a signal processing operation that applies a *self-adjusting*, usually iterative, algorithm to a set of noisy data in order to extract a prescribed quantity of interest. Typically, adaptive filters use a training sequence to adjust its parameters to the desired response. The filter in this case is provided with both the input and the output and the task is to optimize the proper coefficients of the filter. In such a case the filter is referred to as *"Supervised Adaptive Filter"*. However, in signal processing, communications and control systems, there arise some applications that demand an adaptation algorithm that is capable of functioning without the need for knowledge of the input. The decision in such a scenario is based solely on the observable received data. This challenging adaptation structure is referred to in the literature as *"Unsupervised Adaptive Filter"*.

2.2 Blind Source Separation (BSS): Problem Definition and Dimensions

Blind Source Separation (BSS) is intended to separate or recover a set of unobservable sources from another set of mixtures without having access to the mixing matrix coefficients.
Assuming a Multi-Input Multi-Output (MIMO) system with sources denoted by s(n)= $[s_1(n)s_2(n)\dots s_M(n)]^T$. At the output end of the mixer is a set of sensors with a set of

observations $\mathbf{x}(n) = [x_1(n)x_2(n)\ldots x_N(n)]^T$. The requirement is to estimate a demixer that produces a set of outputs $\mathbf{y}(n) = [y_1(n)y_2(n)\ldots y_M(n)]^T$ that are equal to the input sources. In matrix form, this could be shown to be:

$$\mathbf{x}(n) = \mathbf{A}\mathbf{s}(n) \tag{2.1}$$

$$\mathbf{y}(n) = \mathbf{W}\mathbf{x}(n) = \mathbf{W}\mathbf{A}\mathbf{s}(n) \rightarrow \mathbf{D}\mathbf{P}\mathbf{s}(n) \tag{2.2}$$

where $\mathbf{A}$ is the $N \times M$ mixing matrix and $\mathbf{W}$ is the $M \times N$ weight matrix of the demixer. Acknowledging the fact that the sources can be recovered with inevitable ambiguity in both scale and order then the product ($\mathbf{A} \times \mathbf{W}$) may not be the identity matrix but rather $\mathbf{PD}$, where $\mathbf{D}$ is a diagonal matrix, and $\mathbf{P}$ is a permutation matrix. Figures 2.1 and 2.2 [44] show block diagrams of the mixer and the demixer.

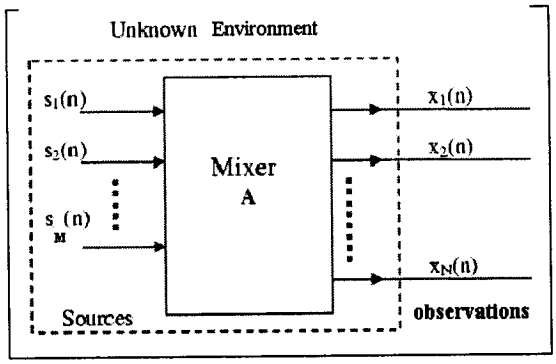

Figure 2.1: **Block diagram of unknown environment resulting in mixing the sources s(n) into the mixtures x(n) [44]**

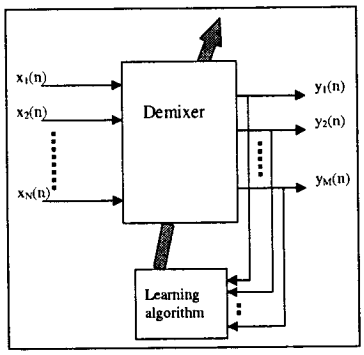

Figure 2.2: **Block diagram of system configured to recover the original sources from the mixtures [44]**

The design of the ultimate separator depends, principally, on the following criteria:

(i) whether the mixer is linear or nonlinear.

(ii) whether the mixer is stationary or time varying.

(iii) whether the sensors are noiseless or noisy.

(iv) whether the mixing operation is non-convolutive (memoryless) or convolutive.

(v) how the number of the sources, M, and the number of sensors, N, are related. (i.e. if $M \leq N$ or $M > N$)

Clearly, the first assumptions in the above list are the easier ones to deal with. Accordingly, the simplest scenario would be: a linear stationary memoryless mixture, received by noiseless sensors and $N \geq M$. Another important criterion for the separation is whether the input mixtures are pre-whitened. Pre-whitening and its importance in BSS will follow in discussion in the next sections.
Blind Signal Extraction (BSE) is a special case of BSS, where only one source or group of sources are of interest to the separation process. In this case, the separation is performed sequentially in a deflationary manner in an order decided upon by the cost function used. Blind deconvolution is an extension of BSS for the instantaneous case to the convolutive case where the entries of the mixing matrix are not scalar but rather are filters. This introduces components related to the past values of the original sources in the present mixtures formed at the output of the mixing matrix. The following figure [22] presents the idea of multi channel blind deconvolution. The mixing matrix in this case is a group of filters and is denoted in the z-domain as $\mathbf{H}(z)$.

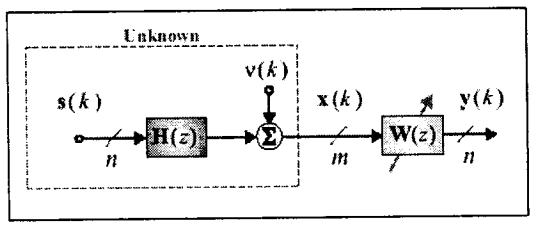

Figure 2.3: **Block diagram of a multi-channel blind deconvolution system [22]**

2.3 Approaches to Blind Signal Separation

Looking at the problem of signal separation it may seem impossible to determine the unknown inputs and the unknown channel or environment only from the output of the mixing process. It is clearly a "one estimates two" under-determined problem. However, as many of the signal processing approaches try to mimic the human behavior (Neural Networks for example), researchers tried to do the same when considering the blind separation solution. A classical example is the cocktail party problem. Clearly, the human brain manages to perform the "concentrate on one speaker" task. The key point is the ability of the brain to extract the most independent features of the oral scene. This is the same way the brain perceives the data from a visual scene, it tries to remove as many redundancy as possible and extract the most **independent** components.
Independence must be distinguished from uncorrelatedness. Indeed, if the y_i observations are independent then the nonlinear-cross correlation $E\{g_1(y_i)g_2(y_j)\}$ fulfills:

$$E\{g_1(y_i)g_2(y_j)\} - E\{g_1(y_i)\}E\{g_2(y_j)\} = 0, \text{for} \quad i \neq j. \tag{2.3}$$

for any function g_1 and g_2. This is clearly a stricter condition than the condition of uncorrelatedness which requires only that:

$$E\{y_i y_j\} - E\{y_i\}E\{y_j\} = 0, i \neq j. \tag{2.4}$$

Remark 2.3.1. The two conditions are equivalent for Gaussian data, which carries all its information in the 1^{st} and 2^{nd} order statistics. However, for other types of data, Higher Order Statistics (HOS) should be used. This gives rise to the first and most popular approach of BSS which is Independent Component Analysis(ICA).

2.3.1 Higher Order Statistics (HOS): A Quick Look

A natural question before jumping into the details of the ICA algorithm is: Why should a Higher-Order Statistics (HOS) based method like ICA perform better than a second order-based method like the Principle Component Analysis (PCA)? The basis for the second order-based PCA is to find the basic components $s_i, i = 1....N$ on the direction that maximizes the variance of the linear N- transform applied to input signals $x_i, i = 1....N$, i.e. the direction yielding least redundant new values. More details on PCA will follow in the next sections. For Gaussian data, all the information is contained in the covariance matrix. Thus, the second-order methods are sufficient. However, if the data is non-Gaussian, the PCA neglects other aspects such as clustering and independence (which is different from uncorrelatedness) as has been indicated earlier. In such a case, it is always useful to move the problem to the HOS domain where there exists more degrees of freedom. HOS can be used explicitly in the cost functions to provide more aspects about data (skewness, symmetry etc.) or implicitly as approximation to a certain measure of meaningful projection used for separation. The value of using HOS is inversely proportional to the gaussianity. Thus, it is important to evaluate the desired probability density distribution, in a solid form, as a cost function that can be calculated. For example, a natural choice to determine how different the distribution is from a Gaussian distribution is the differential entropy. The differential entropy $H(s)$ for a random variable s with pdf as $p(s)$ is given by:

$$H(s) = -\int p_s(\xi) \log p(\xi) d\xi \tag{2.5}$$

However, such information-theoretic quantities are difficult to calculate in practice. Implicit use of higher order statistics serves as approximation in this example. The probability density function $p(s)$ can be approximated using 4^{th} order statistics (cumulant and kurtosis). The exact form of the approximation will be presented later in the chapter along with its use in cost functions. In light of the use of the HOS in BSS, the following definition of ICA can be introduced.

Definition 2.3.1. (General definition) "ICA of the random vector $\mathbf{x}$ consists of finding a transform $\mathbf{y} = \mathbf{W}\mathbf{x}$ so that the components y_i are as independent as possible. This is achieved by some function $F(y_1, \ldots, y_M)$ that measures statistical independence, sparseness or non-Gaussianity [57]".

The ICA transform can be linear or nonlinear, but the scope of this thesis is limited to the linear case. From the above definition, it is clear that the sole assumption is statistical independence which is a feasible and logical condition. It employs higher order statistics either implicitly or explicitly in the cost function. The system allows the existence of no more than one Gaussian source, otherwise the additional Gaussian source will not be distinguished as the HOS does not provide any information for those types of sources. In real life situations, this non-gaussianity ambiguity does not pose a real problem as most sources in signal processing and communication system are nongaussian by nature, i.e super-gaussian or sub-gaussian[1].
In the case that the mixtures $\mathbf{x}$ are composed of time series, the autocorrelation of these mixtures can be used for separating them. In this case, a new category based on Second Order Statistics (SOS) can be used in the cost function [92], [78], [12]. However, these methods fail to separate sources with identical power spectral shape [22]. For non-stationary data, two other approaches have been exploited. A combination of SOS and variance nonstationarity has been presented in [74]. Another approach is to perform joint frequency-time analysis (STF).
The above discussion reveals that the ICA is the most general approach (requires only the statistical independence as a condition and is not based on special cases, i.e. being time series or non-stationary). Thus, it is the most popular approach in BSS and is chosen to be the main focus of the thesis. In fact, ICA can be extended to the other special cases of BSS. For example, a new algorithm combining both ICA and SOS for time series has been presented in [58].

2.4 Preprocessing of Data: Principal Component Analysis (PCA), Why?

Considering the mixing model:

$$\mathbf{x} = \mathbf{A} \times \mathbf{s} \tag{2.6}$$

An important feature used in ICA stems from the ambiguities of the separation process.

1. The independent components can be estimated with ambiguity in sign and multiplicative scalar.

2. The order of the independent components can not be recovered by the separation process.

Given that the exact amplitude of the independent components cannot be recovered, a natural choice employed in all relevant cost functions of ICA is to assume that the original sources $s_i, i = 1 \dots M$ are of zero mean and unit variance, i.e. $\|s_i\| = 1$ and the output of the separation process $y_i, i = 1 \dots M$ are also to have unit variance and zero-mean. It is also common before considering ICA to apply whitening (sphering) on the input mixtures.

[1]Signals with positive kurtosis are referred to as *supergaussian* or ("leptokurtic" in statistics). They have sharp peaky pdf and heavy tails (like Laplacian function). On the contrary, negative kurtosis signals are called *subgaussian* (platykurtic). Subgaussian signals have a more uniform distribution.

This operation exhausts the second-order information and thus speeds the convergence. The whitened version of the mixtures $\mathbf{x}$ is given by:

$$\mathbf{z} = \mathbf{V} \times \mathbf{x} \tag{2.7}$$

Such that $E\{\mathbf{z}\mathbf{z}^T\} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. This constrains the de-mixing matrix $\mathbf{W}$ to lie in the group of orthogonal matrices, an important step that usually helps to tighten the scope of search. Whitening can be done by many methods. The most commonly used is classical Principal Component Analysis (PCA).
Principal Component Analysis, and other concepts like Karhunen-Loéve or the Hotelling transform aims at reducing the redundancy of a certain group of measurements $\mathbf{x}$. Redundancy here is considered in the second order statistics sense. The target is to produce a smaller group of variables as uncorrelated as possible. A famous approach to PCA is the minimization of the mean square error. Define the direction of the first principal component as the vector $\mathbf{w}_1$, by [56]

$$\mathbf{w}_1 = \arg \max_{\|\mathbf{w}\|=1} E(\mathbf{w}^T\mathbf{x})^2 \tag{2.8}$$

where $\mathbf{w_1}$ is the weight vector $\mathbf{w}$ that maximizes the above expectation. Having determined the first k-1 principal components, the k-th principal component is determined as the principal component of the residual [56]:

$$\mathbf{w}_k = \arg \max_{\|\mathbf{w}\|=1} E\{[\mathbf{w}^T(\mathbf{x} - \sum_{i=1}^{k-1}(\mathbf{w}_i\mathbf{w}_i^T\mathbf{x})]^2\} \tag{2.9}$$

The principal components are then given by $y_i = \mathbf{w}_i^T\mathbf{x}$. In practice, the computation is much easier if we use the eigen value de-composition principle.

$$\mathbf{E}D\mathbf{E}^{-1} = \langle \mathbf{x}\mathbf{x}^T \rangle - \langle \mathbf{x} \rangle \langle \mathbf{x} \rangle^T \tag{2.10}$$

where $\mathbf{D}$ is a diagonal matrix with the eigen values on the diagonal, $\mathbf{E}$ is an orthogonal matrix whose columns are the eigen vectors. Then $\mathbf{W_p}$ (PCA decorrelation matrix) is given by:

$$\mathbf{W}_p = \mathbf{D}^{1/2}\mathbf{E}^T \tag{2.11}$$

$\mathbf{W}_p$ in such case is orthogonal.
The PCA operation is particularly important when the number of independent components n is smaller than the number of mixtures m. In such case the mixing and separating matrices are not square which may cause a problem. The main ICA algorithms were developed for square mixing/demixing models. Using PCA helps reducing the dimension of the mixtures N to the dimension of the sources M in case $N > M$.

2.5 Independent Component Analysis(ICA): Identifiability and Key Concepts

It is useful to have a quick look at the necessary condition for the identifiability of the ICA model. These conditions (beside the basic assumption that the sources are originally statistically independent) assure the identifiability for the noise-free ICA model.

1. The mixing matrix A must be of full rank.
2. The number of observed linear mixture Nmust be at least as large as the number of independent components M.
3. At most one source has Gaussian distribution.
4. Sources are stationary and of zero mean.

The first condition is easily deduced from the fact that $\mathbf{W} = \mathbf{A}^{-1}$. The second assumption $N \geq M$ is in fact not necessary for the identifiability of the mixing matrix A, it may be identifiable even in the case $N \leq M$. However, the algorithms developed in the later sections are no longer valid in such a case. This scenario is often treated with the so called *"ICA with over-complete bases"*. This method is discussed briefly later in this chapter. The third condition can be interpreted in the light of the fact that the uncorrelatedness condition for Gaussian sources implies independence and so any decorrelation matrix would be valid. This has been previously pointed out in 1.3.
Without loss of generality, the estimation of the data model of the ICA is usually done by formulating an objective/contrast function and then either maximizing or minimizing it. Based on the objective function, different mathematical approaches can be used for optimization. Thus the review is based on presenting the most important functions used along with the most popular optimization algorithms associated with each function. Independence implies that for the m-dimensional probability density function (*pdf*) of the sources $\mathbf{s}$, the joint probability density factorizes as:

$$p(\mathbf{s}) = \prod_{i=1}^{m} p(s_i(t)) \tag{2.12}$$

Yellen and Weinstein [99] introduced the use of this factorization as a necessary and sufficient condition for the independence and consequently separation of the outputs of BSS filters. In addition, Comon et al. [26] showed that if the mixing process is linear, then it is sufficient for independence of the output that they be pairwise mutually independent.
A good measure of the successful factorization of the *pdfs* is the Kullback-Leibler divergence [57] . The Kullback-Leibler divergence between two probability densities $p_x(x)$ and $q_x(x)$ is given by:

$$\mathcal{K}(p,q) = \int p_x(x) log(\frac{p_x(x)}{q_x(x)}) dx \tag{2.13}$$

The Kullback-Leibler divergence measures the divergence (difference) between the probability distribution $p_x(x)$ and $q_x(x)$. Maximizing such an expression gives a faithful minimum of the mutual information between the two sources and addresses the factorization goal discussed above. The *pdf*, clearly, plays a key role in the BSS algorithms. However, when formulating contrast function the log derivatives of the source densities appear more frequently. They are given the name, the *score* function or the "*squashing*" function $\phi(p_i)$:

$$\phi(p_i) = -\log(p_i)' \qquad \Rightarrow \qquad \phi(p_i) = -\frac{(p_i')}{(p_i)} \tag{2.14}$$

where $(.)'$ refers to the derivative and log(.) is the natural logarithm.
A lot of literature has focused on the proper choice of the score function. It was Girolami [40] who introduced the idea of using different score functions for super- and sub-Gaussian sources. Douglas et al. [29] introduced switching between nonlinearities by analyzing the statistics of the separated sources. The choice of the correct score function is crucial for the separation process. Inadequate choice can destroy the convergence of the whole separation process.

2.6 Classification of Contrast Functions and Optimization Algorithms for ICA

The choice of the suitable contrast function affects the statistical properties of the ICA model. Properties such as consistency, asymptotic variance and robustness are mainly controlled by the contrast function. On the other hand, the algorithmic properties (e.g., convergence speed, memory requirements, numerical stability) depend mainly on the optimization algorithm [57]. Mainly, contrast functions can be classified according to:
(A)The Theoretical basis:

Information Theoretic-based functions : The contrast functions in this group stem from information theory. They usually utilize the *pdf* distribution and entropy expressions. They give faithful and theoretically optimal global solutions and thus are usually robust and immune to outliers [56]. However, they are difficult to estimate in practice and their efficiency relies to a large extent on the approximation used to calculate the distributions used in the cost functions.

Extension of Second order cost functions : These functions were inspired by examining equation(2.3) versus equation (2.4). In these equations, we can capture the resemblance between the uncorrelatedness condition and its extension to the targeted independence condition. The contrast functions in this group are based on contrast functions used to minimize the joint correlation where the expectation of the second order is replaced by the expectation of a higher order function $g(.)$. This rather heuristic extension of Second Order (SO) methods has proven to work in practice. However, there is no theoretical guarantee that it provides sufficient conditions for independence. The reason is clear as independence implies that equation(2.4) applies for all functions $g(.)$ while in practice the contrast functions optimizes a chosen function and does not guarantee global optimization.

Another categorization of the cost functions can be based on:
(B) The number of sources estimated [57]:

Multi-unit contrast functions : Functions of this type estimate all the independent components, or the whole data model, at the same time.

One-unit contrast functions : These functions are intended mainly for blind signal extraction. They have the ability to sequentially extract the sources one by one and thus

estimate only as many sources as needed. The order of the estimated sources depends on the cost function criteria (i.e. if it targets minimizing the entropy then the sources with the furthest distributions from Gaussian are to come out first).

The optimization algorithms can also be classified into two main classes based on the actual processing as:

- **On-line or Iterative algorithms:** These algorithms employ update equations that can be modified from one time sample to another and thus are prefered in the case of changing (nonstationary) systems.
- **Batch (block) algorithms:** They provide the optimum solution for the case of stationary environments. Their convergence properties are usually better than adaptive algorithms and they are better match for data arriving in bursts, like the Global System for Mobile communications (GSM). Even in the presence of time-varying channels, they are considered invariant in their burst period [35]. This invariant model can even hold in the case of slow fading channels. However, in the case of fast fading the inherent non-recursive nature of the batch mode hinders its ability to track this obviously non-stationary case and thus there are always applications that require on-line mode of operation.

2.7 Contrast Functions for ICA

In the following section the major contrast functions for the ICA model are briefly presented. We follow the first categorization based on the theoretical approach. Each function is given a rating of multi-unit/one-unit functions along with its presentation. Generally, any one-unit contrast can be developed to produce a multi-unit result.
It should be noted that all the formulae presented here are based on the assumption that the number of mixtures and the number of original sources are equal i.e. square matrices for mixing and de-mixing.

2.7.1 Information Theoretic-Based Contrast Functions

1. Minimization of Mutual Information

Minimization of mutual information is one fundamental approach to separation based on the information theory. Mutual information can be seen as a measure of dependence between components. The mutual information between scalar random outputs $y_i, i = 1 \dots m$ is given by:

$$I(y_1, y_2, y_3, \dots, y_m) = \sum_{i=1}^{m} H(y_i) - H(\mathbf{y}) \tag{2.15}$$

where $H(\mathbf{y})$ is the differential entropy of the sources $\mathbf{y}$.

$$H(\mathbf{y}) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y} = H(\mathbf{x}) - \log(\det(\mathbf{W})) \tag{2.16}$$

The above mutual information expression can be re-written in terms of $\mathbf{x}$, the mixed inputs as:

$$I(y_1, y_2, y_3, \ldots, y_m) = \sum_{i=1}^{m} H(y_i) - H(\mathbf{x}) - \log(\det(\mathbf{W})) \tag{2.17}$$

where $\mathbf{W}$ is the demixing matrix to estimate.
The mutual information is thus always non-negative and zero if and only if the components are independent. Minimizing mutual information can thus achieve sufficient and necessary requirements for statistical independence.
However, mutual information depends on the entropy calculation and thus is difficult to estimate in practice. Approximations using higher order statistics have been used to provide an expression for the probability density function of a zero-mean unit-variance random variable ζ as following [64],[53]:

$$p(\zeta) \approx \varphi(\zeta)(1 + \kappa_3(\zeta)h_3/6 + \kappa_4(\zeta)h_4/24 + \ldots) \tag{2.18}$$

where $h_i(.), i = 3, 4$ are Hermite polynomials, κ_i are the cumulants of order i, and $\varphi(.)$ is the density function of a standardized Gaussian variable. The above polynomial approximation is used to calculate the entropy expression and as such the mutual information in equation (2.17) [2],[61], [53] is given by:

$$I(\mathbf{y}) \approx C + \frac{1}{48}\sum_{i=1}^{m}[4\kappa_3(y_i)^2 + \kappa_4(y_i)^2 + 7\kappa_4(y_i)^4 - 6\kappa_3(y_i)^2\kappa_4(y_i)] \tag{2.19}$$

where C is a constant.
However, the approximation in equation (2.18) works only if the data distribution is close to Gaussian distribution [53]. Better approximations have been developed in [51] based on a more accurate approximation of the probability densities. The presentation of these approximations for the probability density functions is postponed to be introduced with the closely related concept of Negentropy.
The concept of mutual information works as the basis for all the information theory based approaches. Research does not provide much of explicit algorithms for mutual information calculation. However, all other contrast functions can be considered to be simplified approximations targeting mutual information minimization of the approximated output signals. Generally, all the algorithms that will be developed in the following subsections are mutual information minimizing algorithms.

2. Likelihood and Information Maximization (Infomax)

Maximum-Likelihood (ML) estimation is a popular fundamental method in statistical estimation. It possesses desirable optimal properties that makes it appealing in the case of ICA especially when a large number of sources is required to be separated. It is a multi-unit approach. Meaning that it estimates both the complete separating matrix and the estimated sources all at once.
In the noise-free case of ICA, we consider the following model [56]:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{2.20}$$

where $\mathbf{x}$, $\mathbf{A}$ and $\mathbf{s}$ are as defined in section 2.2.1. Accordingly the probability densities of the mixtures $\mathbf{x}$ can be formulated as:

$$p_x(\mathbf{x}) = |\det(\mathbf{A}^{-1})| p_s(\mathbf{s}) = |\det(\mathbf{W})| p_s(\mathbf{s}) \tag{2.21}$$

Using the assumption of statistical independence in equation (2.12), the above equation can be re-written as:

$$p_x(\mathbf{x}) = |\det(\mathbf{W})| \prod_i p_i(s_i) = |\det(\mathbf{W})| \prod_i p_i(\mathbf{w}_i^T \mathbf{x}) \tag{2.22}$$

where $\mathbf{W} = \mathbf{A}^{-1} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$. Assuming a series of T observations, then the maximum-likelihood $L(\mathbf{W})$ expression is given as:

$$L(\mathbf{W}) = \prod_{t=1}^{T} \prod_{i=1}^{n} p_i(\mathbf{w}_i^T \mathbf{x}(t)) |\det(\mathbf{W})| \tag{2.23}$$

In the log sense, maximum-likelihood is given by:

$$\log(L(\mathbf{W})) = \sum_{t=1}^{T} \sum_{i=1}^{n} \log(p_i(\mathbf{w}_i^T \mathbf{x}(t))) + T \log|\det(\mathbf{W})| \tag{2.24}$$

$$\frac{1}{T} \log(L(\mathbf{W})) = E\{\sum_{i=1}^{n} \log(p_i(\mathbf{w}_i^T \mathbf{x})\} + \log|\det(\mathbf{W})| \tag{2.25}$$

An interesting case to point out here is when the input mixtures $\mathbf{x}$ are already pre-whitened [7]. In this case, based on the assumption that the outputs are of unit variance, the matrix $\mathbf{W}$ is orthogonal and thus has a unit determinant which means that equation (2.25) becomes:

$$\frac{1}{T} \log(L(\mathbf{W})) = E\{\sum_{i=1}^{n} p_i(\mathbf{w}_i^T \mathbf{x})\} \tag{2.26}$$

We can see that maximizing L depends on the densities of the independent components and on the matrix $\mathbf{W}$.

- **On-line Maximum Likelihood Algorithms**: The pioneer algorithm in this domain can be traced to Bell and Sejnowski paper [10]. They were the first to present the idea of maximum-likelihood and they developed an adaptive algorithm by differentiating the expression in equation (2.25):

$$\frac{1}{T} \frac{\partial \log L}{\partial \mathbf{W}} = [\mathbf{W}^T]^{-1} + E\{\mathbf{g}(\mathbf{W}\mathbf{x})\}\mathbf{x}^T \tag{2.27}$$

  It follows that:

$$\triangle \mathbf{W} \propto [\mathbf{W}^T]^{-1} + E\{\mathbf{g}(\mathbf{W}\mathbf{x})\}\mathbf{x}^T \tag{2.28}$$

Here $\mathbf{g(y)} = (g_1(y_1) \ldots, g_n(y_n))$, where $g_i(y_i)$ is the score function of the variable y_i defined following equation (2.5.3). A good choice for the score functions is:

- (i)Super-Gaussian data: $g^+(y_i) = -\tanh(y_i)$ corresponding to $G_i^+(y_i) = \log p_i(y_i) = -\log\cosh(y_i)$.
- (ii)Sub-Gaussian data: $g^-(y_i) = \tanh(y_i) - y_i$ corresponding to $G_i^-(y_i) = \log p_i(y_i) = -[\frac{y_i^2}{2} - \log\cosh(y_i)]$

However, the Bell-Sejwenski algorithm as described in equation (2.28) converges very slowly. An elegant way to improve the convergence speed of the algorithm came from exploring the geometric properties of the optimization domain and used the mathematical set of the Lie-group to follow the special nature of this geometry to adjust the gradient in the direction that points to the global minima. This is realized by both the separate works of Amari [6] as the natural gradient and Cardoso and Laheld as the relevant gradient which was deduced from a different route [19].
The complete derivation of the algorithm is postponed to the next chapter where the whole concept of geometrical approaches to adapt the gradient is presented. Briefly, in the current case the natural gradient modifies the gradient of the cost function by a right multiplication of $\mathbf{W}^T\mathbf{W}$. The natural gradient equation (2.28) is:

$$\triangle\mathbf{W} \propto [I + E\{\mathbf{g(Wx)}\}\mathbf{y}^T]\mathbf{W} \tag{2.29}$$

This algorithm converges much faster than the original one. It is actually the main algorithm in the on-line category while the original gradient algorithm is obsolete.

- **Batch Algorithms**: In [56], the Fast-ICA for maximum-likelihood batch algorithm has been presented. The algorithm requires the data to be pre-whitened. This algorithm is not provided by an elaborate mathematical derivation. Instead, it is considered as extension to earlier work of the author who has developed a successful batch algorithm for a different category of cost functions referred to as Negentropy (will follow in discussion). Table(2.1) summarizes the algorithm [56]

Table 2.1: **The Fast ICA algorithm for maximum likelihood estimation.**

1. Center the data to make its mean zero. Apply a whitening process to the data
2. Choose an initial (e.g. random) separating matrix $\mathbf{W}$.
3. Compute

$$\mathbf{y} = \mathbf{W}\,\mathbf{x} \tag{2.30a}$$

$$\beta_i = -E\{y_i g(y_i)\}, \quad for \quad i = 1, ..., N \tag{2.30b}$$

$$\alpha_i = -1/(\beta_i + E\{g(y_i)\}), \quad for \quad i = 1, ..., N \tag{2.30c}$$

   where $g(y_i$ is $g^+(y_i) = -\tanh(y_i)$ for supergaussian data and $g^-(y_i) = \tanh(y_i) - y_i$ for subgaussian data.

4. Update the separating matrix by:

$$\mathbf{W} \longleftarrow \mathbf{W} + \mathrm{Diag}(\alpha_i)[\mathrm{Diag}(\beta_i + E\{\mathbf{g}(\mathbf{y})\mathbf{y}^T\}\mathbf{W} \tag{2.31}$$

   where $\mathrm{Diag}(\alpha_i)$ forms an $N \times N$ diagonal matrix whose diagonal elements are the elements of the $N \times 1$ vector $\alpha_i, i = 1 \ldots N$.

5. If not converged, go back to step 3.

Now that the concept and the main algorithms for maximum-likelihood have been pointed out, it is possible to discuss the relationship between the maximum-likelihood concept and the other concepts in the information theory domain.

Relationship with Information maximization (Infomax):
A different view to the maximum-likelihood got its roots from neural networks. The idea is simply to try to maximize the amount of information delivered from the mixtures $\mathbf{x}$ to the outputs of the separating network $\mathbf{y}$. Following this rather different view leads to the same set of equations provided for maximum-likelihood adaptation.

Relationship with mutual information concept equation (2.17) reveals the symmetry between mutual information minimization and the maximum-likelihood maximization. In fact, approximating the densities of y_i and modifying the entropy accordingly will result in the following approximation for mutual information:

$$I(y_1, \ldots, y_n) = -\sum_i E\{G_i(y_i)\} - \log|\det(W)| - H(\mathbf{x}) \tag{2.32}$$

where, $G_i(y_i)$ is the approximation for $\log p_i(y_i)$. Choosing $G_i(y_i)$ to be of the same value used in maximum-likelihood section (with the choice for either super-gaussian or sub-gaussian) provides identical approximation for the expression of minimizing mutual information and maximizing maximum-likelihood (the difference in the signs of expressions is justified as one is maximized and the other is minimized while the entropy $H(\mathbf{x})$ is constant that has no influence on the derivative used in the adaptation).

3. Maximization of Negentropy

The legitimate justification for this approach comes from the fact that the ICA seeks non-Gaussian projections for the outputs. Principally, the goal is to achieve non-gaussianity. It has previously been mentioned that Gaussian variables have the largest entropy. Unfortunately, differential entropy is not invariant to scale transforms. Hence, modifications have to be made to obtain a linear-affine invariant version of entropy [2]. This is achieved by the quantity named *"Negentropy"*. Negentropy actually means the negative of entropy. It is well-known that for variables with the same covariance matrices the Gaussian distribution has the largest entropy. Negentropy of a certain variable is then defined as:

$$J(\mathbf{y}) = H(\nu) - H(\mathbf{y}) \tag{2.33}$$

where ν is a Gaussian random vector of the same variance as $\mathbf{y}$. This quantity is always positive and invariant to transforms. Maximizing Negentropy is then equivalent to maximizing non-gaussianity.
As most blind signal separation algorithms, Negentropy-based algorithms require the input mixtures to be whitened either before or during the separation process (on-line whitening) [45]. This information theoretic approach has the advantage of being a one-unit contrast function approach. Thus, it is able to extend its application to cover both blind signal separation and blind signal extraction. Another advantage of Negentropy is that it does not assume prior distribution of the sources. Unfortunately, as all statistical quantities depending on entropy in definition, Negentropy is difficult to estimate. One approximation is given by the *Edgeworth* expansion [54]:

$$J(\mathbf{y}) \approx \frac{1}{12}k_3(\mathbf{y}^2) + \frac{1}{48}k_4(\mathbf{y}^2) \tag{2.34}$$

If $\mathbf{y}$ has a skewed distribution then $J(\mathbf{y}) \approx \frac{1}{12}k_3(\mathbf{y}^2)$. If $\mathbf{y}$ has a symmetric disribution then $J(\mathbf{y}) \approx \frac{1}{48}kurt(\mathbf{y}^2)$. In this case, the maximization of Negentropy is more or less the maximization of the absolute value of the kurtosis. In particular, this approximation suffers from the same robustness problem. However, more sophisticated approximations also exist. If generalized to non-quadratic functions the *Edgeworth* approximation can be written as:

$$J(\mathbf{y}) \approx c_1(E\{G_1(\mathbf{y})\})^2 + c_2(E\{G_2(\mathbf{y})\})^2 - E\{G_2(\nu)\}^2 \tag{2.35}$$

where c_1 and c_2 are positive contrasts . In the case we use only one non-quadratic function, the approximation becomes:

$$J(\mathbf{y}) \propto [E\{G(\mathbf{y})\} - E\{G(\nu)\}]^2 \tag{2.36}$$

The following choices of G, the integral of the score function, have provided very efficient algorithms:

[2]The linear affine invariant quantities are invariant to scale transforms meaning that assumptions made about the sources $\mathbf{s}$ will not be affected by the linear transforms $\mathbf{A}$ and $\mathbf{W}$ respectively and thus are faithfully preserved for the recovered signals $\mathbf{y}$. Such invariant characteristics can thus be used as objective measures in the optimization algorithms.

$G = \log\cosh(\mathbf{y})$
$G = -\exp(-\mathbf{y}^2/2)$
This approximation is fast to compute and has appealing statistical properties, especially robustness to outliers.

- **Iterative Negentropy update**[57]: Using equation (2.36) for Negentropy and choosing $G(\mathbf{y}) = \log(\cosh(\mathbf{y}))$ gives $g(\mathbf{y}) = G'(\mathbf{y}) = \tanh(\mathbf{y})$; and $g'(y_i) = 1 - \tanh^2(y_i)$. The tanh function is the most suitable to use among all non-quadratic functions and produces good results for both super and sub-gaussian sources. A stochastic on-line update of equation (2.36):

$$\mathbf{w}_+ = \mathbf{w} + \mu_r \gamma \mathbf{z} g(\mathbf{w}^T\mathbf{z}) \tag{2.37}$$

$$\mathbf{w}_{new} = \mathbf{w}_+ / \|\mathbf{w}_+\|$$

where $\mathbf{w}_{new}$ is the new updated version of $\mathbf{w}$. The choice of $\gamma = [E\{G(\mathbf{w}^T\mathbf{z})\}\} - E\{G(v)\}]$ can be set to $+1$ for subgaussian data and -1 for supergausian data to ensure stability. $\mathbf{z}$ is the vector of pre-whitened mixtures. However this simple update in equation (2.37) does not work well in case of highly correlated sources.

- **Batch Negentropy-based Algorithm**: This batch FAST-ICA algorithm depends on a Newton-type update of equation (2.36) and thus works in batch mode. The complete derivation of the algorithm is provided in [54]. Briefly, the algorithm depends on the following equation as an update role:

$$\mathbf{w}_+ = E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\} - E\{g'(\mathbf{w}^T\mathbf{z})\}\mathbf{w} \tag{2.38}$$
$$\mathbf{w}_{new} = \mathbf{w}_+ / \|\mathbf{w}_+\|$$

where $g(.)'$ is the derivative of the score function $g(.)$. The following table summarizes the main steps of the algorithm provided as matlab package in [46].

Table 2.2: **The Fast ICA algorithm for finding one maximally nongaussian direction. [56]**

1. Center the data to make its mean zero.
2. Whiten the data to give z.
3. Choose an initial (e.g. random) vector $\mathbf{w}$ of unit norm.
4. Let $\mathbf{w}_+ \longleftarrow E\{\mathbf{z}\mathbf{g}(\mathbf{w}^T\mathbf{z})\} - E\{\mathbf{g}'(\mathbf{w}^T\mathbf{z})\}\mathbf{w}$.
5. Normalize $\mathbf{w}_{new} \longleftarrow \mathbf{w}_+ / \| \mathbf{w}_+ \|$
6. If not converged, go back to step 4.

The Fast-ICA works efficiently in Blind Signal Extraction (BSE) and is considered by far the most successful algorithm based on Negentropy. However, the fact that

the algorithm only works off-line presents a disadvantage, especially in tracking non-stationary mixtures. Thus, it is valuable to present an on-line algorithm that combines the advantages of the Negentropy principle (especially the ability to extract as many sources as needed) with a fast convergent on-line operation that is comparable to other existing on-line algorithms in other categories of blind signal separation.

The relationship between Negentropy and the concept of mutual information minimization is clear. It has been pointed out that the mutual information $I(\mathbf{y})$ is closely tied to Negentropy $J(\mathbf{y})$. In fact, mutual information can be defined in terms of Negentropy as follows:

$$I(y_1, y_2, \ldots, y_n) = J(\mathbf{y}) - \sum_i J(y_i) \tag{2.39}$$

Thus, finding the maximum Negentropy directions, i.e. directions where the sum of the elements $J(y_i)$ is maximized, turns out to be equivalent to finding a representation where the mutual information is minimized.

2.7.2 Extensions of Second-Order Methods

1. Higher-order Tensor algorithms

Generally, tensors are higher order matrices. A 4^{th} order cumulant tensor is thus a linear operator or a 4-indices matrix and can be expressed by the function $cum(x_i, x_j, x_k, x_l)$ for any set of 4-inputs $\mathbf{x}$. Cumulant tensors are symmetric and thus ordering of data samples is irrelevant. A 4^{th} order cumulant contains all 4^{th} order information about the data. This is similar to the auto-correlation which contains all second order information about the data. In another analogy to autocorrelation, independence implies that all cross cumulants, i.e. cumulants with at least two different indices, are equal to zero. Just like ucorrelatedness implies that the cross-correlation terms vanish.
The goal of all the algorithms developed in this domain is forcing the cross-cumulants to zero. The algorithms developed under this concept are mostly multi-unit contrast functions aiming at separating all the data at once. Typically batch algorithms dominate this category of cost functions. The input mixtures are required to be pre-whitened to make the separating matrix orthogonal and thus:

$$\mathbf{W}\mathbf{W}^T = \mathbf{I}, \qquad \mathbf{A} = \mathbf{W}^T \tag{2.40}$$

In [18] the concept of Joint Approximate Diagonalization of Eigenmatrices (JADE) was presented. The JADE algorithm is considered the most prominent algorithm in this family. The contrast function of the JADE algorithm can be defined to be:

$$J_{JADE}(\mathbf{w}) = \sum_{ijkl \neq iikl} cum(y_i, y_j, y_k, y_l) \tag{2.41}$$

which is exactly forcing the cross cumulants to be zero. However, what the JADE adds is the idea of breaking the tensor down to small matrices and trying to diagonalize each instead of working on a big tensor.

Remark 2.7.1. Looking at the cost function of the JADE algorithms reveals the fact that they break the symmetry of the structure of the matrices (by working on sub-matrices individually). This limits the use of JADE algorithms. They also inherit the same problems of methods using an explicit tensor EVD. They cannot be used in high-dimensional space.

More details on the Higher-order tensor methods can be found in [79], [17]

2. Higher-order Cumulants

Another method that applies HOS explicitly uses higher-order cumulants. The function to optimize in this case is a one-unit contrast function that is effectively equivalent to the Negentropy cost function $J(\mathbf{y})$ when direct approximations of the nonlinearities performed with HOS polynomials are used for both cost functions as in equation (2.42):

$$J(\mathbf{y}) \approx \frac{1}{12} k_3(\mathbf{y}^2) + \frac{1}{48} k_4(\mathbf{y}^2) \tag{2.42}$$

The mathematical simplicity of determining the cumulants, especially the possibility of achieving global convergence, has contributed largely to the popularity of the high-order cumulants approach.
However, kurtosis provides a rather poor objective function. This is because of the two following reasons:

1. Higher-order cumulants measure mainly the tails of the distribution. They are largely unaffected by the structure in the middle.

2. Estimators of higher-order cumulants are highly sensitive to outliers (not robust)

So, even in the case of no noise, neither the independent components nor the mixing matrix can be computed accurately.

3. Nonlinear Cross-correlation

In the above subsection an example was presented on extending the second order statistics directly to higher order ones. This section presents a different approach to generalization from second order to higher order. Nonlinear cross correlation can be defined as:

$$E\{g_1(y_i)g_2(y_j)\}, \text{for} \quad i \neq j. \tag{2.43}$$

Clearly, it is a modification of the original equation of cross-correlation to the nonlinear case. It implicitly integrates higher order concepts in the definition of the function $g(.)$ used as nonlinearities. These nonlinearities (implicit HOS functions) replace the second order quantities in the cost functions originally developed for second-order algorithm (de-correlation algorithm)to generalize decorrelation to independence.
This principle of canceling nonlinear cross-correlation was first introduced by Jutten and Herault [63]. Justification can be seen from the definition of independence and uncorrelatedness. Recalling the equations, equation (2.3) and equation (2.4)

$$E\{g_1(y_i)g_2(y_j)\} - E\{g_1(y_i)\}E\{g_2(y_j)\} = 0, \quad \text{for} \quad i \neq j.$$

$$E\{y_i y_j\} - E\{y_i\}E\{y_j\} = 0, i \neq j.$$

It is clear that it is possible to generalize the concept of uncorrelatedness to any nonlinear transformation given by the set of function g_i. It was pointed out in [56] that a sufficient but not necessary condition for the nonlinear uncorrelatedness is that y_i, $1 \leq i \leq M$, are independent and the nonlinearities are odd functions. This makes it credible that nonlinear correlations could be useful as possible general criterion for independence. The choice of the nonlinearities depends on the pdf's of the independent components.
Generally speaking, the contrast functions in this domain serve as multi-unit ones. What is really appealing in them is that they are very suitable for on-line adaptation algorithms.

- **On-line Algorithms:** It is remarkable that although the principle of blind signal separation based on cancelation of nonlinear cross-correlation is rather heuristic, the algorithm developed turned out to be of the same form as natural gradient algorithms developed by the well-justified maximum-likelihood principle [9]. In the following some highlights are presented on the most prominent algorithms used.
  Jutten-Héralut Algorithm: This algorithm was inspired by neural networks. The principle is to cancel the nonlinear cross-correlation i.e. forcing the diagonal terms of the correlation matrix to be zeros. The complete derivation is presented in [45]. Briefly, a new off-diagonal matrix $\mathbf{M}$ is defined with updates according to:

$$\Delta \mathbf{M}_{ij} \propto g_1(y_i)g_2(y_j), for \quad i \neq j \tag{2.44}$$

  where g_1 and g_2 are some odd-symmetric nonlinearities. $\mathbf{y}$ is computed after each iteration according to:

$$\mathbf{y} = (\mathbf{I} + \mathbf{M})^{-1}\mathbf{x} = \mathbf{W}\mathbf{x} \tag{2.45}$$

  In the numerical computation of the matrix $\mathbf{W}$ the term $(\mathbf{I} + \mathbf{M})^{-1}$ must be updated at each step. This is computationally heavy, especially if the approach is extended to more than two sources. One way to circumvent this approach was provided by Cichoki and Unbehaun [24].

$$\mathbf{y} = (\mathbf{I} + \mathbf{M})^{-1}\mathbf{x} \approx (\mathbf{I} - \mathbf{M})\mathbf{x} \tag{2.46}$$

  This algorithm was the cornerstone for developing a more sophisticated algorithm employing feedforward network with higher dimensionality that lead to the update [24]:

$$\Delta \mathbf{W} \propto (\mathbf{I} - \mathbf{g}_1(\mathbf{y})\mathbf{g}_2(\mathbf{y}^T))\mathbf{W} \tag{2.47}$$

  which is similar to the natural gradient update. Another alternative was introduced as a separate algorithm referred to a Equivariant Adaptive Separation via Independence (EASI)[19]:

$$\Delta \mathbf{W} \propto (I - \mathbf{y}\mathbf{y}^T - \mathbf{g}(\mathbf{y})\mathbf{y}^T + \mathbf{y}\mathbf{g}(\mathbf{y}^T))\mathbf{W} \tag{2.48}$$

  This algorithm is identical to the natural-gradient with added on-line whitening.

4. Nonlinear PCA

The trend of generalization from second order to higher order or linear to nonlinear inspired several authors to apply the same concept to PCA. Following the same route as nonlinear

correlation cancelation, nonlinear PCA can be obtained by applying a nonlinearity $g(.)$ to the classical PCA formula:

$$\mathbf{w} = \arg \max_{\|\mathbf{w}\|=1} E\{g(\mathbf{w}^T\mathbf{x})^2\} \tag{2.49}$$

The algorithms in this category mainly depend on nonlinear extensions to the neural-network approach for the classical PCA. The hierarchical learning rules is:

$$\Delta w_i \propto g(y_i)\mathbf{x} - g(y_i)\sum_{j=1}^{i} g(y_i)\mathbf{w}_j \tag{2.50}$$

A simplification to this algorithm is the bi-gradient algorithm. The bi-gradient algorithm replaces the feedback term by a simpler one. The learning rule is given by:

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \mu(t)\mathbf{g}(\mathbf{W}(t)\mathbf{x}(t))\mathbf{x}^T(t) + \alpha(\mathbf{I} - \mathbf{W}(t)\mathbf{W}^T(t))\mathbf{W}(t) \tag{2.51}$$

where μ is the learning rate and α is a constant such that $0.5 \leq \alpha \leq 1$. Another Recursive Least Square (RLS)-based algorithm was also developed for the same concept in [84].

Summary of ICA Basic Methods

The previous sections presented a review of the basic techniques and contrast functions used in ICA. When considering real-life applications, the JADE algorithm complexity increases with the number of sources and thus its use is limited to small dimension problems. However, the most widely acclaimed techniques are those of the natural gradient and the Fast-ICA Negentropy algorithm that provides the option of extracting as many sources as needed. The domain of ICA keeps evolving and some interesting extensions to the natural gradient have been provided in [9], [39]. The above section presented only the most famous techniques.

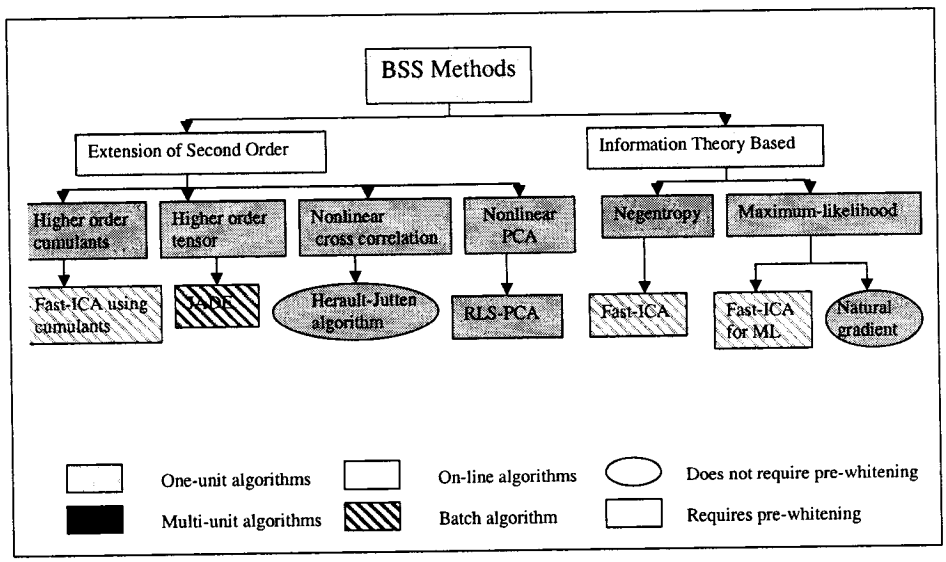

Figure 2.4: **Summary of the basic approaches to blind signal separation along with the principal algorithms in each approach**

2.8 Noisy ICA

In real life, noise always exists. This is an unfortunate situation because it introduces a new problem. The noisy model is not invertible and therefore the estimation of the noise-free components requires new methods.

2.8.1 Noisy ICA-Models

Here, the basic ICA model is extended to the situation where the noise is present. However, the new model depends on the type of the noise. *"Sensor"* noise is represented by the noise variables added separately to the mixtures $\mathbf{x}$.

$$\mathbf{x} = \mathbf{As} + \mathbf{n} \tag{2.52}$$

"Source" noise on the other hand represents the case when the noise vector is added to the independent components $\mathbf{s}$.

$$\mathbf{x} = \mathbf{A}(\mathbf{s} + \mathbf{n}) = \mathbf{As} + \mathbf{An} \tag{2.53}$$

The source noise model clearly reduces to the basic ICA model with extra independent components. The following realistic assumptions about the vector $\mathbf{n} = (n_1, ..., n_N)$ are made in order to facilitate the calculations:

1. The noise is additive.
2. The noise is independent from the independent components.
3. The noise is Gaussian.

2.8.2 Modified-Algorithms for Noisy ICA

The following table summarizes how to modify the above mentioned algorithms, if needed, to deal with the noisy environment. The key to noise treatment comes from the assumption that the noise is Gaussian.

Table 2.3: **Modifications to the algorithms of the basic ICA to incorporate noise**

Method	Modification
1. Whitening:	The ordinary whitening should be replaced by the operation: $$\tilde{\mathbf{x}} = (\mathbf{C} - \mathbf{\Sigma})^{-1/2}\mathbf{x} \quad (2.54)$$ where $\mathbf{C} = E\{\mathbf{x}\mathbf{x}^T\}$ and $\Sigma = \sigma^2\mathbf{I}$ is the covariance of the noise [57].
2. Kurtosis-based:	The kurtosis based algorithms are immune to Gaussian noise, their lack of robustness may be very problematic in a noisy environment though.
3. Higher-order cumulant:	The previously mentioned higher-order cumulants used both second order and fourth order cumulants. The second order cumulant is not immune to noise. Some modified algorithms introduced the sixth-order cumulant. However, 6^{th} order cumulant is very sensitive to outliers. Despite that, the modified pre-whitening in step 1 could be very useful in removing noise.
4. Maximum Likelihood:	Maximum likelihood can be modified in the noisy situation by approximating the densities of the independent components as Gaussian mixtures. This can be done by maximizing the marginal densities of the components and the covariance of the noise, if possible. However, the computational complexity grows exponentially with the dimension of the data.
5. General Nongaussianity measures:	For these general (non-kurtosis) measures of nongaussianity the best modification is introduced by bias removal techniques.

2.8.3 Bias-Removal

Bias removal is concerned with modifying the ordinary (noise-free) ICA methods so that the bias due to noise is removed or at least minimized. The modified whitening technique in Table 2.3 can be considered as a bias-removal. Several sophisticated approaches of this most prominent technique are presented in [56] for general non-gaussianity measures.

2.9 ICA with Over-Complete Bases

Another situation that is more challenging than the basic ICA model is when the number of independent components N is larger than the number of mixtures M. This missing data situation results in two complicated problems:

1. The mixing matrix is very difficult to estimate. Even if the matrix is known it is not

invertible.

2. How to estimate the independent components if the mixing matrix is not invertible.

It may be more useful in this situation to use the basic ICA model as an approximation to the over-complete bases model. At least, efficient and reliable algorithms have been developed for the ordinary case. Such algorithms can serve as a starting point for further modifications.

2.9.1 Estimation of the Independent Components

A natural choice to solve the problem of non-invertibility of the mixing matrix is to apply pseudo-inverse:

$$\widehat{\mathbf{s}} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{x} \tag{2.55}$$

In some situations this simple pseudo-inverse is sufficient. However, in many cases the need arises for a more sophisticated procedure. This is achieved through Maximum Likelihood (ML) in a manner similar to the Maximum A Posteriori (MAP) estimate.

$$\widehat{\mathbf{s}} = \arg\max_{\mathbf{x}=\mathbf{A}\mathbf{s}} \sum_i \log p_i(s_i) \tag{2.56}$$

where p_i are the prior probability densities of the independent components. At first glance the algorithm seems difficult to compute. However, it exhibits a very interesting sparse nature when the independent components are supergaussian. In this case, the ML estimator gives coefficients s_i of which only M (equal to the number of mixtures) are non-zero. Thus, the system can be invertible for those components.

2.9.2 Estimation of the Mixing Matrix for Over-complete Basis BSS

It is important to recognize the aspects of similarity between the over-complete bases problem and the noisy case. They both have the same problem in the difficulty to estimate the mixing matrix and the independent component. One is, thus, tempted to try to apply the mathematical tools used for the noisy model to the present problem. Maximum-likelihood is then a perfect candidate. This can be done by formulating the joint likelihood of the mixing matrix $\mathbf{A}$ and the realization of the components s_i. Then, maximizing this joint likelihood with respect to all these variables. Another approach is to view the problem through Bayesian estimation and to try to maximize the marginal probability of the mixing matrix. Several approaches in this framework have been developed:

1. **Laplacian approximation:** It has been successfully used in separation of image and audio sources with over-complete bases, but the computational complexity is high [68].

2. **Monte-Carlo methods:** It typically gives estimators with good statistical properties but is also computationally expensive [82].

3. **Competitive neural-learning:** It works with supergaussian data and assumes the extreme case of sparsity (one component is non-zero at a time). It is very computationally efficient but unrealistic and works with limited group of components [52].

The conclusion that can be drawn here is that, due to the difficulty of the problem, ML seems rather inefficient. New techniques based on modifying the basic ICA algorithm and finding quasi-decorrelating sparse decomposition were introduced in [56]. These techniques might be the most promising ones. They proved to be both very efficient in estimation and more computationally relaxed.

2.10 BSS for Nonlinear Mixing Systems

In some situations, the linear ICA model is too simple to describe the real mixing operation. It would be more realistic to introduce the following nonlinear model:

$$\mathbf{x} = f(\mathbf{s}) \tag{2.57}$$

where $\mathbf{x}$ is the N-dimensional observed vector and $f(.)$ is an unknown N-dimensional nonlinear mixing function. For simplicity, we will assume that the number of sources $M = N$. In such a case, the problem is to find a mapping function $h(.)$ such that:

$$\mathbf{y} = h(\mathbf{x}) = \mathbf{s} \tag{2.58}$$

However, this problem is completely different from the linear ICA model. In the nonlinear domain two components y_1 and y_2 could be statistically independent but still mixed. In this case the ICA problem is ill-posed. The solution is non-unique and extra information should be provided to the model. The ICA basic definition of making the components of the output vector y as independent as possible no longer holds.

2.10.1 Special Case: Post Nonlinear Mixtures

One important special case of the nonlinear mixtures is given by the following model:

$$x_i = f_i(\sum_{j=1}^{n} a_{ij} s_i), \quad i = 1, ..., M \tag{2.59}$$

The sources in this case are mixed linearly first then applied to a nonlinear function f. An intuitive solution to this model is to try to remove this nonlinearity and then apply the basic ICA algorithm to the roughly linear mixture. The sources can be separated or the independent components can be estimated up to scaling, permutation and sign ambiguities under weak conditions on the mixing matrix $\mathbf{A}$ (the same case with basic ICA). This model could be very useful in real life for nonlinear sensor distortion. However, it cannot be applied to other situations successfully.

2.10.2 Nonlinear BSS Algorithms: Brief Review

The algorithms developed for the nonlinear BSS problem can be generally categorized in the following two classes:

1. **Signal Transformation Approach**: The goal is to estimate the sources directly using inverse transformation.

2. **Generative Approach**: The goal is to model the nonlinear mapping. It is concerned with a more thorough understanding of how the mixture is formed.

Clearly, the generative models are expected to be more complicated yet more accurate than the signal transformation models.

1. Self-Organizing Maps

One example of the *signal transformation* algorithms is the Self Organizing- Maps (SOM) [66]. SOM depend on unsupervised learned mapping from a high dimensional to a two-dimensional rectangular grid. The objective is to conserve the data structure as well as possible while uniformly distributing it over the grid. Uniform distribution choice for the mapped data emerged from the following theoretical fact: "if the joint probability of two random variables is uniformly distributed inside a rectangle, then clearly the marginal densities along the sides of the rectangle are statistically independent" [57]. However, this implied uniform distribution is not suitable for some types of data (it implies a subgaussian nature and does not suit supergaussian data). Another problem is the high-computational complexity of the algorithm [83].

2. Generative Topographic Mapping (GTM)

An example of the *generative approaches* is the Generative Topographic Mapping (GTM). This generative approach starts with assuming a model for the latent variables (independent components in our case). The model is formed of mutually similar delta functions equally distributed over the rectangular grid. The unsupervised mapping from the mixture high-dimensional space to the grid is modeled by a combination of Gaussian functions. The coefficients (weights) of those functions are estimated by maximum-likelihood. A modification to this approach is to add weights to the delta functions modeling the components $\mathbf{s}$.

3. Ensemble Learning Approach

This is a new *generative* approach that uses the Multi-Layer Perceptron (MLP) network structure. Bayesian ensemble learning rule is used for the unsupervised learning of the algorithm. The idea is to offer a flexible algorithm that is capable of treating both mild and strong nonlinearity (overcome the problem of under- and over-fitting). The idea is to offer several models. Each of these models is given a weight based on the a posteriori pdf's of the sources [14]. Although this a posteriori pdf is difficult to estimate, *ensemble learning* also known as variational learning is capable of providing good approximation [50]. The cost-function in this algorithm penalizes the misfit between the exact a posteriori of the independent components $p(\mathbf{s}, \mathbf{theta}/\mathbf{x})$ and their parametric estimates $(q(\mathbf{s}, \theta/\mathbf{x}))$measured by the Kullback-Leiber (KL) divergence:

$$J_{KL} = \int q(\mathbf{s}, \theta/\mathbf{x}) \log \frac{q(\mathbf{s}, \theta/\mathbf{x})}{p(\mathbf{s}, \theta/\mathbf{x})} \quad d\theta \; d\mathbf{s} \tag{2.60}$$

2.11 Signal Separation Using Time-Structure

In all the previous models of ICA, the Independent Components (ICs) were assumed to be random variables. However, in many cases this is not the case. In many situations the ICs are time series, $s_i(t)$, $t = 1...T$, $i = 1....n$. The question now is: Is this extra information useful for the separation? The answer is definitely yes. It opens the door for a new set of approaches to the separation problem. These approaches are based on the time structure of the ICs rather than the nongaussianity concept [8]. This is useful in both providing simpler algorithms for the ordinary separable mixtures and providing possible solutions for the inseparable mixtures (for example the Gaussian ones). One of the most powerful tools in the time-structure based methods is the auto-covariance. The idea is to find a separating matrix $\mathbf{W}$ such that the instantaneous and lagged covariances between the different components in the output vector $\mathbf{y}$ is zero:

$$E\{y_i(t)y_j(t-\tau)\} = 0, \quad for\ all\ i,j,\tau \tag{2.61}$$

The motivation is that for the ICs s_i the lagged covariances are zero due to independence. Thus, auto-covariance can be used instead of the nongaussianity criterion. Several other proposals for the time-structure based algorithms depending on the non-stationarity of the covariance, autocorrelation and higher order cumulants were provided in the literature. However, they are beyond the scope of the thesis.

2.12 Convolutive Blind Signal Separation

In many real life situations, the mixing process does not only include the instantaneous mixture values but delayed and convolved samples. One good example of that is the intersymbol interference in fading communication channels. Another good example in acoustics is when considering the effect of the echo and reverberation on the audio scene. This is clear in the following figure [77].

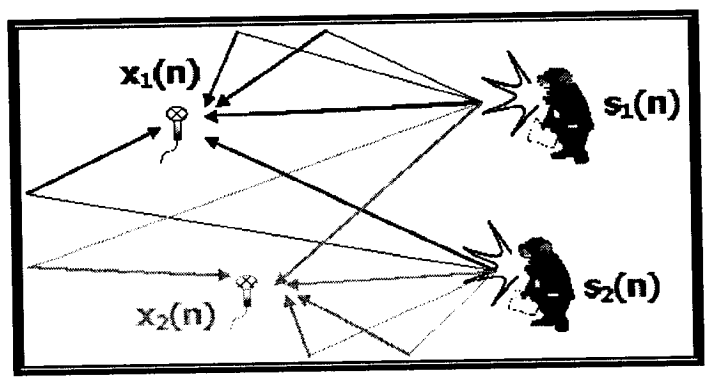

Figure 2.5: **A real room source separation scenario** [77]

Consider a multi-microphone system equipped to capture different audio mixtures in a room. Due to the echo and reverbation effect, each microphone will receive delayed versions of the intended signal and other fading delayed copies of the other signals. Simply, the mixing model does not hold as a simple matrix multiplication:

$$\mathbf{x}(n) = \mathbf{A} \times \mathbf{s}(n) \tag{2.62}$$

It is rather expressed as a convolution between the mixing matrix (filter $\mathbf{H}$) and the sources:

$$\mathbf{x}(n) = \mathbf{H} * \mathbf{s}(n) \tag{2.63}$$

where the elements of $\mathbf{H}$ are actually FIR filters. Thus, the algorithms already implemented for the simple instantaneous mixing case should be modified for the above case of Multi-Channel Convolution (MCC).

2.12.1 Approaches to Multi-Channel Convolutive Blind Signal Separation

In the convolutive mixtures case two main approaches have been considered, either solving the problem in the time domain or moving to the frequency domain. The main reason for which frequency domain approaches emerged is that by transforming the above mixing equation to the frequency domain it is possible to get rid of the convolution operation and transform the problem back to the instantaneous mixtures case.

$$\mathbf{x}(f) = \mathbf{H}(f) \times \mathbf{s}(f) \tag{2.64}$$

Clearly, by doing so all the previously mentioned adaptive algorithms in the preceding chapters can still hold with the modification that the mathematical operations are always considered for frequency bins instead of time samples, and that data are considered to be complex data not just real data. An example of how straightforward working in the frequency domain could be is the natural gradient algorithm in the frequency domain. In this case for every frequency bin f considered in the Short Time Fourier Transform (STFT) there is a demixing matrix $\mathbf{W}_f$ given by:

$$\mathbf{W}_f(k) = \mathbf{W}_f(k-1) + \mu(\mathbf{I} - g(\mathbf{u}(f) * \mathbf{u}^H)\mathbf{W}_f(\mathbf{k}-\mathbf{1}) \tag{2.65}$$

where k is the block index and $(.)^H$ refers to the Hermitian transpose. Clearly this equation is very similar to the basic natural gradient equation for instantaneous mixtures, with the main difference being the use of the Hermitian transpose instead of the simple transposition of the matrices. Another potential for the frequency-domain is that data are better represented statistically in this domain, meaning that sparse data (i.e. super-gaussian data) have more spiky distributions and are easier to distinguish from Gaussian distributions compared to their counterpart representation in the time domain [77]. However, one disadvantage is that we would be forced to work on frame by frame basis and the adaptation would be applied on the frequency bins. Solving the problem independently for each frequency bin introduces a permutation problem. The permutation problem stems from the fact that the de-mixing

process involves an unresolved order ambiguity, i.e. the sources are restored but the ordering might not match the ordering of the original sources. This has not been a problem for the time-domain instantaneous approaches. However, for the frequency case the order of the sources must remain the same along the frequency axis. Otherwise, separation might be achieved but, due to permutation, the mixtures cannot be restored when returning to the time-domain. Several approaches have been presented in the literature to provide frequency coupling between neighboring frequency bins and thus impose restrictions that result in the same order of de-mixing along the frequency axis. Other approaches are focused on the use of beam-forming to achieve the same objective.

Time domain approaches, however, try to deal with the new mixing model including the convolution operation. Several attempts can be traced to the pioneer work of Lambert [67]. He introduced a de-mixing model for Finite Impulse Response (FIR) filters that are minimum phase. Several other researchers have worked in the time domain motivated by the fact that although time domain can be more complicated due to the convolutive nature it possesses the advantage that no permutation problem exists. In [45], [22] Amari et al. have implemented a convolutive version of the natural gradient approach by modifying the gradient. If the mixing model is:

$$\mathbf{y}(k) = \sum_{p=-\infty}^{\infty} \mathbf{W}_p(k)\mathbf{x}(k-p) \tag{2.66}$$

where $\mathbf{y}(k) = [y_1(k) \dots y_N(k)]$ is the N-dimensional vector of outputs at time k and $\mathbf{x}(k) = [x_1(k) \dots x_N(k)]$ is the N-dimensional vector of input mixtures while $\mathbf{W}_p$ is a sequence of matrices of dimensions $N \times N$ containing the mixing coefficients at time k. For the above model the natural gradient algorithm has been developed to be [29]:

$$\mathbf{W}_p(k+1) = \mathbf{W}_p(k) + \mu[\mathbf{W}_p(k) - g(\mathbf{y}(k))\mathbf{u}_p^T(k)] \tag{2.67}$$

where $\mathbf{u}_p(k)$ is given by:

$$\mathbf{u}_p(k) = \sum_{q=-\infty}^{\infty} \mathbf{W}_q^T(k)\mathbf{y}(k-p+q) \tag{2.68}$$

For practical implementation, the doubly infinite filter model used in the de-mixing model is replaced by an FIR filter of L taps:

$$y(k) = \sum_{p=0}^{L} \mathbf{W}_p(k)\mathbf{x}(k-p) \tag{2.69}$$

The natural gradient update rule can then be modified to be:

$$\mathbf{W}_p(k+1) = \mathbf{W}_p(k) + \mu[\mathbf{W}_p(k) - g(\mathbf{y}(k-L))\mathbf{u}^T(k-p)] \tag{2.70}$$

where $\mathbf{u}$ in such case is expressed as:

$$\mathbf{u}(k) = \sum_{q=0}^{L} \mathbf{W}_{L-q}^T(k)\mathbf{y}(k-p+q) \tag{2.71}$$

This algorithm has been tested and proven to be able to provide reasonable source separation performance. However, a closer look at the update equation reveals that it actually uses an L delayed version of the nonlinearity $g(\mathbf{y}(k-L))$. The main assumption here is that the data is slowly changing so that the following approximation is valid:

$$g(\mathbf{y}(k-L)) \approx g(\mathbf{y}(k)) \tag{2.72}$$

In the case of using large order filter L, like the case of echo-cancelation, this approximation might not be a valid one. It might actually worsen the results compared to using less filter taps. This approximation thus might eliminate any potential improvement that the natural gradient modification presents. Another general problem related to the large number of filter taps is that time-domain algorithms (involving convolution) tend to perform poorly and costly as the number of taps increase even if the above approximation is not performed [44]. One approach to solve problems of long FIR filters is to use shorter Infinite Impulse Response (IIR) filters instead. However, the drawbacks of such approach are clear as IIR-filters have stability problems, especially if the system is non-minimum phase.

2.13 Conclusion

This chapter presents the problem of blind signal mixing/demixing along with the most successful algorithms developed in the literature to perform the separation. The problem of instantaneous blind source separation was considered first. It has been shown that several categories of cost functions exist to deal with this mixing situation, among which the information-theoretic based approaches, namely maximum-likelihood and Negentropy, are favored due to their robustness to outliers. Other approaches based on generalizing the second-order decorrelation solution and using higher order statistics were also discussed.
The review of the separation problem was afterwards extended to more difficult scenarios where noise exists and/or the mixing operation cannot be linear. The more advanced solutions for these situations were briefly reviewed. This review concluded with the discussion of the case where the sources mixed are time series. In such a case, one can make use of the time structure to form other types of cost functions that can address the special nature of the mixing situation.
Finally, the extension of the instantaneous mixtures case to the more realistic convolutive mixtures is introduced along with a review of the methods employed in the literature to target this problem. Mainly, frequency domain approaches are considered favorites when dealing with convolutive mixtures situation as they present a window to map the problem back to instantaneous filter multiplication between different frequency bins. By reviewing the advantages and disadvantages of the convolutive mixtures approaches, the chapter can be considered as a concise review of the work done in the blind signal separation domain.
It should be noted that the approaches to blind separation presented in this chapter (with the exception of the natural gradient algorithm) are based on optimization on the Euclidean space. The breakthrough that was achieved in the natural gradient principle was mainly the introduction of the idea of optimization on more general spaces, i.e. curved spaces that better represent the optimization problem at hand. In the following chapter, a presentation of the mathematical concepts of optimization on curved spaces is presented along with a geometrical derivation of the natural gradient principle and other related extensions.

Chapter 3

Understanding the Background of the Riemannian Geometry

3.1 Introduction

This chapter is intended to be an intuitive guide to the Riemannian geometry from an engineering point of view. This constructive understanding of the mathematical background is then extended to a logical derivation of the concept of natural gradient (first order derivative in Riemannian geometry) and the less famous second derivatives in Riemannian geometry methods such as Newton and the conjugate gradient.

3.2 Motivation

Optimization is the core task of many engineering problems. Most approaches to practical solutions depend on a mathematical formulation of the problem and determining the best values using the proper optimization technique. The question then arises: How to determine the appropriate optimization technique?
When reviewing the methods used in optimization and adaptive filtering, one can easily notice that they are almost all based on the assumption that the space in which the problem is posed is Euclidean. For years, this facilitated the derivation of most of the optimization methods. Methods like the steepest descent and the Newton algorithm that appeared a long time ago up to modern optimization techniques developed in the mid 20^{th} century, like the conjugate gradient by Stiefel and Hestenes [49] in 1952, were all based on this assumption. All these methods have been long enough on the optimization scene and have developed a solid background of properties and performance. However, it has long been admitted by the optimization community that the optimization procedure to be chosen depends on many factors, most important of which is the nature and geometry of the problem posed. Many problems arise with special constraints on the Euclidean manifold. The most common are the equality constraints, especially the orthogonality ones. These problems belong in fact to another manifold, the Riemannian manifold. Respecting the geometry and the natural curvature of such a manifold was proven to be important. It was Fletcher [36] who raised

taking the special nature of the manifolds into account. He proposed an ideal steepest descent on the constraint surface with steps along the geodesic [1]. However, the algorithm was not feasible and Fletcher followed the concept of constrained optimization instead.
All the classical constrained optimization methods (like projective methods, Lagrange multipliers methods, etc) are extrinsic methods that do not make use of the intrinsic geometry of the optimization domain. This appeared to penalize the performance of the developed algorithms based on constrained optimization. Fuhrmann and Liu [37] recognized that the improved performance of the second order methods are no longer guaranteed when the curvature terms are not taken into account in the optimization algorithms. They developed a conjugate gradient algorithm taking the curvature of the manifold into consideration and proved to be superior to the constrained optimization counterpart. It thus appeared very appealing to go a step further in the mathematics fundamentals, and explore intrinsic approaches to develop an unconstrained optimization to take over the classical constrained optimization and develop more efficient [2], more robust, and faster convergent algorithms.
Researchers in signal processing, BSS and independent component analysis were not far from these new developments in the optimization field. Amari et al. have proposed that taking the actual geometry of the optimization space into account will provide an improved performance [45], [5], [4]. The so-called natural gradient algorithm is a direct application of taking the Riemannian structure of the adaptation problem into account in the case of the maximum-likelihood cost function. This adaptive algorithm exhibits a dramatic superior performance to the stochastic gradient algorithm and yet it is simpler. Meanwhile, in [38], the proportionate LMS technique was linked to the Riemannian geometry concept, specifically to the natural gradient. Taking the natural gradient as a starting point to our improved performance algorithms, this motivates us to present this comprehensive review of the Riemannian geometry as an introduction to the new methods in the thesis.

3.3 Identifying the Riemannian Optimization Problem

After the previous section that pointed out the benefits of exploring the actual structure of the optimization domain, it would be interesting to point out which type of problems one should identify as Riemannian optimization problem.
Consider the problem of optimizing a certain cost function $J(\mathbf{W})$ of a matrix $\mathbf{W}$. The goal is to find the optimal value of $\mathbf{W}$: $\mathbf{W}_{opt}$. Conventionally, using the steepest descent method, the following update is developed:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu \nabla_{\mathbf{W}} J \tag{3.1}$$

and works well if the function J is quadratic. i.e. unimodal. The gradient will be linear and consequently isotropic [3] at any point in any direction away from the minimum.
However, if this is not the case, the steepest descent and consequently the gradient should be modified to provide a homogeneous isotropic update emanating from different points in the space. This will give the first case where Riemannian optimization should be recognized:

[1] Geodesics are curves of special nature on the Riemannian space that will be explained later
[2] It is worth noting that projective methods widely used in constrained optimization are costly algorithms
[3] Istropic tensors have the same components in all rotated coordinate systems [95]

(1) **Cost function that is not quadratically isotropic**.
The second obvious case is the case:
(2) **When constraints are imposed on the adaptation**: This is previously mentioned in the above section with an example of constraint as the orthogonality constraint.
Actually the two cases (1) and (2) are correlated to each other. If constraints are imposed on the adaptation of the parameter $\mathbf{W}$ then the gradient of the function $\nabla_{\mathbf{W}} J$ does not represent the actual update of the parameter as another step, i.e. orthogonalization or another gradient, Lagrange gradient, should be included. Overall the update step in the end will not be isotropic.
On the other hand, if it is the cost function that is not isotropic , i.e. case (1), then one can always find the constraints of the motion on the optimization space [4].
After identifying the optimization on the Riemannian space, a natural question is how to change the steepest descent or for that matter any other optimization technique to the new geometry.
There exist two main approaches which are actually correlated mathematically and lead to very similar update results. However, the explanation in the chapter is chosen to identify the two approaches separately to emphasize each of them.
The first approach comes as an immediate answer to the first case where the Riemannian space is identified, i.e. "**Cost function that is not quadratically isotropic**". The solution consists of identifying a mapping function $\mathbf{g}$ between the notion of the distance in the chosen Riemannian space and the conventional notion of distance in the Euclidean space. By defining this mapping function, the gradient of the function J can be modified to express the real distance in the Riemannian space. The new gradient $\tilde{\nabla}_{\mathbf{W}} J$ will be:

$$\tilde{\nabla}_{\mathbf{W}} J \equiv \mathbf{g}^{-1} \nabla_{\mathbf{W}} J \tag{3.2}$$

The new gradient will be isotropic from any point away from the minima.
Note that the mapping function $\mathbf{g}$ is called the Riemannian metric tensor. It will be explained in more details throughout the chapter.
The other approach stems from the second definition of the problem, i.e. "**When constraints are imposed on the adaptation**". Quite briefly, by defining these constraints one could identify an algebraic constrained formulation for the manifold. This algebraic formulation is denoted as "Lie Groups". Every Lie group has an associated derivative-like space called Lie-Algebra . The solution of the optimization problem will be to perform the conventional optimization technique on the Lie-algebra space and then mapping it back to the Lie group. Detailed explanation of this process will be presented later in the chapter along with definitions of Lie group and Lie algebra.
The important point to establish here is that Lie group is the intrinsic means to apply the constraint on the space. By constituting the space as Lie group and obtaining the gradient on a well defined Lie algebra, more control can be imposed on the coefficient of the parameter we want to optimize to "lock" it on the optimization space.
The above discussion can by no means be considered complete. One could just think of it as preparing the stage and having a global vision to justify the following more mathematically

[4] Admittedly this could be difficult in some cases. However, as the true manifold of optimization is Riemannian such constraints and conditions always exist

related sections. More understanding of the structure of the Riemannian manifold is needed along with the introduction of the terms like Lie- group, tangent space, etc. The following sections provide what is thought to be the required mathematical tools for Riemannian optimization. Later in the chapter, these tools are applied upon the problem of BSS using the maximum-likelihood principle.

3.4 Riemannian Manifold: Definition and Properties

Before talking specifically about Riemannian manifold, it would be useful to define the meaning of a "**Manifold**" in a simple way. A manifold is simply a topological space that is locally Euclidean. A good example of a manifold is earth. Earth is proven to be of spherical shape, however for long time it was considered flat (Euclidean). This discrepancy arises essentially from the fact that on the small scales that we see, the Earth does indeed look flat. An exact mathematical definition of a manifold is the following [28].

Definition 3.4.1. A topological space M is called a **topological manifold** of dimension m if each point $p \in M$ has a connected continuous neighborhood U that posses a *homeomorphic*[5] map x on the Euclidean space $\mathbb{R}^m$ of the same dimension m. This pair of the neighborhood and its map constitutes what is known as a **chart** (local co-ordinates).

A continuous transformation that preserves distance between points is particularly interesting and is given the name '**isometry**'. A Riemannian manifold is a manifold that posses a *metric tensor* **g** that is both symmetric and positive definite. This metric tensor is a function used to calculate the actual distance on the curved surface of the Riemannian manifold. Being symmetric means that the distance between two points (a, b) is the same no matter which one is the starting point of the displacement and having the metric to be positive definite implies that the distance is always positive. More highlights on the metric tensor will follow in the discussion.

3.5 Tangent and Normal Spaces

The rest of the Riemannian geometry explanation part draws on the main characteristic of this manifold, namely "*curvature*". In order to proceed with the equations governing this constrained manifold new definitions, concepts and equations revealing the special nature of motion over such curved or twisted manifolds should be considered. Most of these ideas are trivial or non-existing when considering the Euclidean flat space.
As a beginning, consider Figure 3.1 [31] showing a point on a Riemannian manifold. At each point p on the manifold we can find tangent vectors in all directions. This set of vectors form a ***tangent plane*** T_p, which is a space containing the set of tangent vectors at this point p. This tangent plane acts as a first order differentiation of the manifold at the point p.

[5]Homeomorphism: A one-to-one equivalence mapping between points from one topological space to another and vice versa (i.e. continuous in both directions) [95]

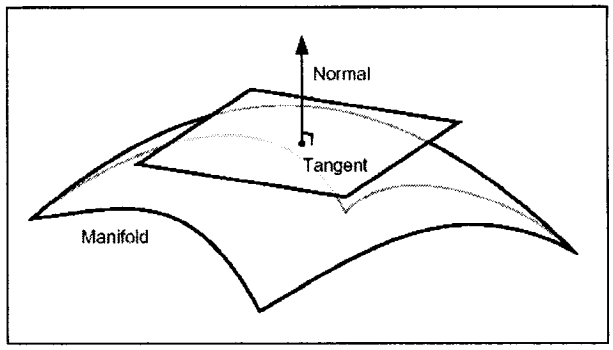

Figure 3.1: **Example of tangent and normal spaces at a point on a Riemannian manifold** [31]

In some sense the tangent plane is the Euclidean approximation of Riemannian manifold at the point. The collection of the tangent planes on the manifold is given the name tangent space T_M, and is of dimension M, the same dimension as the manifold. The normal space is the orthogonal complement of the tangent space. The tangent space should contain all the tangent vectors of all points along all possible curves of motion along the manifold. Clearly, due to curvature Tangent spaces are not naturally connected.
In order to collect all the tangent spaces together a certain "connection" function is needed to guarantee a smooth directed translation from one tangent plane (subspace) at one point to another tangent plane at another point. We can conclude from the above that the actual Riemannian first derivative is different from the form that results from direct differentiation in the Euclidean space. The next section presents a more in depth look at the connection connecting the tangent spaces and the translation by which the tangents are translated along a certain path.
Another quantity that is unique to the curvature property of the Riemannian manifold is the"***curvature vector***". It indicates the direction in which a certain curve under inspection is turning. In equation form, if we define a parameterized curve as γ with a tangent vector $\dot{\gamma}$ then the unit tangent is defined as:

$$\mathbf{T} = \frac{\dot{\gamma}}{|\dot{\gamma}|} \tag{3.3}$$

Then the unit curvature vector is:

$$\kappa = \frac{\partial \mathbf{T}}{\partial t} \tag{3.4}$$

It indicates the rate of change of the unit tangent to the curve studied.

3.6 The Concept of a Connection

Figure 3.1 helps visualizing the tangent plane T_p of a Manifold M at a certain point p. There exists an infinite number of tangential vectors connected to a manifold. There is, however, a fundamental difference between the tangent vectors at a certain point on a Riemannian manifold and those of Euclidean spaces. In Euclidean spaces, the tangents are *naturally isomorphic*, meaning that there is a natural isomorphism (connection) between the tangents.

Moving one tangent vector from a point to another is an operation that does not require more than the start and the end point along with the original vector.
However, on the Riemannian manifold, due to curvature such isomorphism should be imposed. There are several choices of relations to establish between tangent spaces at different points depending on the motion curve connecting them. However, when talking about general vector fields, defined pointwise in the manifold then the derivatives will generally have components in the normal space along with the desired component in the tangent space. The projection of the derivative on the normal space is referred to as "***the covariant derivative***" or what is referred to sometimes as a "***connection***". The covariant derivative of a certain vector can be re-phrased to be the rate of change of a vector along a certain path. If the rate of change of the vector is zero, then the vector is being "parallel transported" along the path. Thinking of the arc distance as displacement, the tangent vector would be the first order differentiation (velocity). Then, the rate of change of the vector is the acceleration. To have the acceleration equal to zero means to have it orthogonal to the manifold i.e. a vector in the orthogonal space.
Considering the parallel transpose of a tangent vector visually, the goal is to move the vector infinitesimally in parallel to its original orientation and then remove the normal component from the new vector to obtain the parallel transposed one. Every Riemannian manifold has

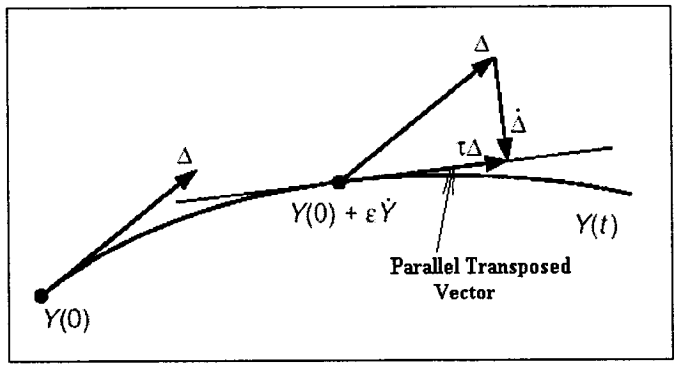

Figure 3.2: **Parallel transposing a parallel vector** [31]

a canonical connection, the Levi-Civita connection, that is used to determine the 'natural' parallel transport of vectors. This connection is intrinsic and metric, defined by the metric tensor coefficients g_{ij}. This connection is unique to the manifold. The fundamental theorem of the Riemannian manifold implies that for any $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ vector fields, the Levi-Civita connection satisfies :

(i) $\nabla_{\mathbf{x}}\mathbf{y} - \nabla_{\mathbf{y}}\mathbf{x} = [\mathbf{x}, \mathbf{y}]$. The connection is symmetric or torsion-free. The indication of torsion free is that this connection does not change unnecessarily the direction of the vectors transposed.

(ii) $\mathbf{x}\langle \mathbf{y}, \mathbf{z}\rangle = \langle \nabla_{\mathbf{x}}\mathbf{y}, \mathbf{z}\rangle + \langle \mathbf{y}, \nabla_{\mathbf{x}}\mathbf{z}\rangle$ the connection is metric (contains the information of the curvature of the manifold). This also implies that parallel translation is an isometry. Meaning that the vector length is preserved.

The above notation $\nabla_{\mathbf{x}}\mathbf{y}$ denotes a *directional derivative*, that is the rate of change of the vector $\mathbf{y}$ in the direction of the other vector $\mathbf{x}$. $[\mathbf{x}, \mathbf{y}]$ is the commutator:

$$[\mathbf{x}, \mathbf{y}] = \mathbf{xy} - \mathbf{yx} \tag{3.5}$$

Define the *Christoffel* coefficients as Γ_{ij}^{k}:

$$\Gamma_{ij}^{k} = \frac{1}{2}\sum_{l} g^{kl}\left(\frac{\partial g_{li}}{\partial x^{j}} - \frac{\partial g_{ij}}{\partial x^{l}} + \frac{\partial g_{jl}}{\partial x^{i}}\right) \tag{3.6}$$

where g^{ij} are the elements of the inverse of the tensor g. Two important observations can be noticed from the above equation. The first is that the *Christoffel* coefficients are defined solely by the metric tensor coefficients g_{ij} and that the coefficients Γ_{ij}^{k} span all the metric coefficients in the direction $\mathbf{x}^{i}$ and $\mathbf{x}^{j}$. Thus, the set of *Christoffel* coefficients are both intrinsic (defined uniquely by the geometry of the space) and complete. There exists a 1-to-1 relationship between the manifold and its Christoffel coefficients. The second observation that can be drawn is that each element in the above equation is formed by differentiating a coefficient $g_{\alpha\beta}$, where $\mathbf{x}^{\alpha}, \mathbf{x}^{\beta}$ are any two co-ordinates on the manifold, with respect to a third co-ordinate $\mathbf{x}^{\xi}$. These elements will solely have a value due to the curvature nature of the manifold. Imagining a flat local plane where $\frac{\partial g_{\alpha\beta}}{\partial x^{\xi}} = 0$ then the *Christoffel* coefficients will reduce to zero too. Such locally flat planes are called "Inertial frames".

Definition 3.6.1. (Existence of a Local Inertial Frame) If p is any point in the Riemannian manifold M, then there exists a local coordinate system x^{i} at p such that:
(a)$g_{ij}(p) = \begin{cases} 1, j = i \\ 0, j \neq i \end{cases}$
(b) $\frac{\partial g_{kj}}{\partial x^{j}} = 0$ for every k.
We call such a coordinate system a local inertial frame or a normal frame.

It follows that $\Gamma_{ij}^{k}(p) = 0$ in an inertial frame. Inside the inertial frames, certain curves exhibit a very nice nature of having zero curvature. These curves will seem as straight lines (locally) in the inertial frame. This type of curves are called geodesic. In terms of *Christoffel* coefficients, Γ_{ij}^{k}, one can define the Levi-Civita connection using the canonical basis $(\partial/\partial x^{1}), \ldots, (\partial/\partial x^{m})$ of the patch chart $(U, x^{1}, \ldots, x^{m})$ defined on the manifold M^{m} and $i, j, k \leq m$ as:

$$\nabla_{\partial/\partial x^{i}} \frac{\partial}{\partial x^{j}} = \sum_{i,j,k} \Gamma_{i,j}^{k} \frac{\partial}{\partial x^{k}} \tag{3.7}$$

The significance of such operation is the use of the whole set of *Christoffel* coefficients carrying all the information about the curvature of the manifold. Such coefficients optimize the path along which the connection works. Optimization comes in terms of minimizing its length along with preserving the relations distinguishing the Levi-Civita connection. In other words the optimized curve is carried along successive interial frames, being a straight line (locally) in each frame. It follows from the above discussion that optimization in Riemannian manifold is carried along "***geodesic***". The structure of geodesics is of paramount importance in the optimization on the Riemannian manifold and will be presented in more details in the following sections.

3.7 Geodesic: Definition and Properties

This section presents geodesics, their characteristics and points out their importance in Riemannian optimization. Thus, some other definitions and formulae are needed to set the stage for the different equations associated with geodesics.

Definition 3.7.1. Let M be a Riemannian manifold, then there exists a unique intrinsic tensor called the metric tensor $\mathbf{g}$. The tensor $\mathbf{g}$ is composed of m^2 functions, where m is the dimension of the manifold, defined as follows. For $i = 1 \dots m$ and $j = 1 \dots m$ are the span of the co-ordinate chart:

$$\mathbf{g} = \sum_{i,j} \mathbf{g}_{ij} dx^i \otimes dx^j \tag{3.8}$$

for the above equation the Kronecker multiplication $\otimes$ results in an element by element multiplication spanning all possible combination of i, j. The resulting tensor has the square dimension of the dimension of the manifold M and the above summation in equation (3.8) will reduce to a span of all elements i, j.

$$\mathbf{g} = SPAN_{i,j} \mathbf{g}_{ij} dx^i \otimes dx^j \tag{3.9}$$

As $\mathbf{g}_p$ is non-degenerate [6], then it will have the inverse with elements denoted by g^{ij}.
The metric tensor $\mathbf{g}$ satisfies the following properties for every point $p \in M$ and $\mathbf{x}, \mathbf{y}$ vector fields defined on the manifold, i.e.. $\mathbf{x}, \mathbf{y} \in T_p$:

$$\mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{g}(\mathbf{y}, \mathbf{x}) \tag{3.10}$$

$\mathbf{g}_p : T_p \times T_p \rightarrow \mathbf{R}$ is positive definite.

The bilinear symmetric form $\mathbf{g}_p$ defines the inner product on the tangent space T_p at a point p:

$$\mathbf{g}_p(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle \tag{3.11}$$

The brackets $\langle ., . \rangle_p$ define the metric, i.e. the inner product of two vectors defined on the tangent space at the point p.
Clearly, this metric is different on the Riemannian manifold than on the Euclidean space. In order to distinguish between the two spaces, the above inner product $\langle ., . \rangle_p$, which is actually defined on the Riemannian manifold, will be from now on given the subscript R as opposed to the inner product on the Euclidean space, which will be given the subscript E. In the same context the actual length of a vector $\mathbf{x} \in T_p$ is defined as the following:

$$\| \mathbf{x} \| = \mathbf{g}_p(\mathbf{x}, \mathbf{x})^{1/2} \tag{3.12}$$

Thus, the actual distance between two points on a Riemannian manifold, which is determined by the Riemannian metric, is calculated along the shortest arc connecting the two points. Let $t \rightarrow \gamma(t)$ be the curve connecting between a and b, then its length L is given by:

$$L = \int_{t_0}^{t_1} g_{\gamma(t)}(\dot{\gamma}, \dot{\gamma})^{1/2} dt \tag{3.13}$$

[6] Non-degenerate: a bilinear symmetric transform that is 1-1, i.e. completely and uniquley defines a manifold M. The object manifold M is both continuous and smooth. Thus the tensor defining the bilinear transform is invertible

where $\dot{\gamma}$ is the first order derivative of the curve γ. This shortest curve on a Riemannian manifold is referred to as a "**Riemannian metric distance**" and has to lie on a geodesic between the two points t_0, t_1. The shortest arc between any general two point is then an arc located at one of the geodesics of the Riemannian space. In other words, the old concept of "the shortest distance between two point is a straight line", which applies to Euclidean space is mapped into "the shortest distance between two points is an arc of a geodesic", for the Riemannian space. Although all the shortest arcs connected between points are geodesics the inverse is not true. There exists other geodesics that are not shortest distance. However, the ones the optimization algorithms are interested in are those that optimize the distances between points on the Riemannian manifold.
Recall the discussion in section 3.4 about the local inertial frames. The shortest distance geodesic is locally a geodesic that is a straight line in the subject inertial frame existing on the curved surface of the examined manifold. A set of examples of different geodesics on different spaces is presented in the following figures.

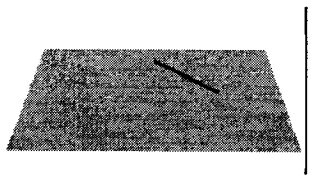

Figure 3.3: **Geodesic on a plane: every straight line is a geodesic [32]**

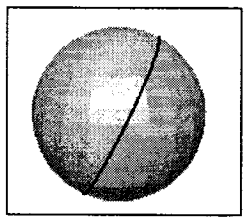

Figure 3.4: **Geodesic on a sphere: geodesics on spheres are great circles [32]**

One can actually see that the above simple definition of a geodesic as straightest curve on the manifold is the closest to imagination and the more general that covers all types of geodesic, and not only the shortest distance one. Looking at any of the curves assigned to be geodesics, one can see that they will indeed look as straight lines in a 2-dimension space. For example, in the case of the sphere, geodesics are great circles (like equator for earth). From the above explanation, the following interesting properties about geodesics can be stated:

1. **Geodesics have no tangential curvature**: Curvature definition has been discussed in the previous sections. Let us define the tangential curvature, or what is sometimes referred to as the "*geodesic curvature*" κ_g, as the covariant derivative of the unit tangent vector or in other words as the projection of the curvature κ on tangential space. Geodesics are the curvature where κ_g are equal to zero.

2. **Geodesic curvature depends entirely on the surface of the manifold**: The

geodesics contain the set of points satisfying the following non-linear equation:

$$\frac{d^2x^k}{dt^2} + \sum_{i,j} \frac{dx^i}{dt}\frac{dx^j}{dt}\Gamma^k_{ij} = 0 \tag{3.14}$$

Remark 3.7.1. It is clear from the above equation that the term containing the *Christoffel* coefficients clearly contains information of the normal space. This follows from the fact that the second derivative of $\mathbf{x}$ does not contain information in the tangent space.

3. **From each point on the manifold there is a geodesic in every direction**

4. **Geodesics are homogeneous, every sub segment on a geodesic carries all the geodesic characteristics** :
   Every point on the geodesic satisfies the geodesic equation.

5. **A Geodesic parallel transposes its own tangent vector to itself**:
   This follows directly from the fact that geodesics are the curves forming the Levi-Civita connection path. This is why the covariant derivative of the tangential vectors are zero. In fact some mathematical books and dissertations [90] use this property as a definition of geodesics. In mathematical notation:
   If $\gamma(t)$ is a geodesic with tangential vector $\mathbf{x}$ such that $\mathbf{x}(t) = \dot{\gamma}(t)$ then:

$$\nabla_{\mathbf{x}}\mathbf{x} = 0 \tag{3.15}$$

   Any other vector $\mathbf{y}(t)$ in the tangential space is said to be parallel-transposed if:

$$\nabla_{\mathbf{x}}\mathbf{y}(t) = 0 \tag{3.16}$$

   for all t. The connection $\nabla_{\mathbf{x}}$ is it self the Levi-Civita connection defined along the path $\gamma(t)$

Now, a preliminary idea why geodesics are important can be presented. When performing optimization along the Riemannian space, geodesics are the only curve that are able to carry the derivative vectors pointing to the minima on the curve without rotating their directions and through the shortest possible path. The logical question that follows is how to move along geodesics given the tangent vector at a certain point. This is the topic of the next section.

3.8 Lie Group and Lie Algebra

Here, a *basic* sufficient information on Lie groups and Lie algebra is presented. First, one might need to refresh the mathematical definition of a group.

Definition 3.8.1. A group $(G, \star)$ is a non-empty set, together with a binary operation, called the group operation, such as multiplication or addition, satisfying the following axioms:

1. Closure: The group G is Closed under the defined mathematical operation $\star$: $(G \star G \rightarrow G)$ i.e. for any two values $a, b \in G$ $(a \star b) = c \in G$.

2. Associativity: For all a, b and c in G, $(a \star b) \star c = a \star (b \star c)$.
3. Existence of identity element: There exists an element e in G such that for all a in G, $e \times a = a \times e = a$.
4. Existence of inverse element: For every a in G, there is an element b in G such that $a \times b = b \times a = e$, where e is the identity element from the previous axiom.

The branch of mathematics which studies groups is called group theory, which can be traced back to the works of Evariste Galois (1830). Previous to his work, groups were mainly studied concretely, in other branches of mathematics without being explicitly defined.
In 1870, Sophus Lie introduced a new type of groups called the Lie groups. Lie groups are groups where operations like multiplication and inversion are given by analytical mapping [7]. This actually means that these operations are smooth. Translating the algebraic definition of Lie groups into geometry by reviewing the definition of "Manifolds" reveals that Lie groups are actually the algebraic counterparts to the geometry representation of the manifold. They define the constrained spaces in a compact mathematical forms. This link will facilitate our study of the examples that follow.
It is the valuable property of closure that promotes Lie groups to play the role of intrinsically constrained manifold. Defining the right type of the Lie group and then moving along the group from one old point to another will guarantee the new point to lie on the manifold. The method by which moving from one point to another will be explained later in the section but first examples of different types of Lie groups are presented. Lie groups actually cover a wide range of existing groups. For example, the **General Linear** group $GL(n, R)$, which is the set of all real n-by-n invertible matrices. $GL(n)$ is easily verified to be a Lie group of dimension n^2. Another example of Lie groups is the **Orthogonal** group. This is a subgroup of $GL(n)$ given by:

$$O(n) = \{\Theta \in GL(n) : \Theta^T \Theta = I\}$$

$O(n)$ is a Lie group of dimension $n(n-1)/2$. What is meant by dimension here is the degrees of freedom found in the equations.
A special case of the orthogonal group is the identity component of $O(n)$, which is called the **Special Orthogonal** group $SO(n)$, which is defined by:

$$SO(n) = \{\Theta \in GL(n) : \Theta^T \Theta = I, \det \Theta = 1\}$$

Another important example of Lie groups is the **Stiefel** Manifold $V_{n,k}$. This is the set of all real matrices (n-by-k), $k \leq n$, with orthonormal columns, i.e.,

$$V_{n,k} = \{U \in R^{n \times k} : U^T U = I\}$$

Another new notation related to Lie Group is the *Lie algebra*. To every Lie-group, there exists a Lie algebra which completely captures the local structure of the group.

Definition 3.8.2. A *Lie algebra* $\mathbf{g}$ over R is a flat vector space over R with a closed bilinear operation [,]: $\mathbf{g} \times \mathbf{g} \rightarrow \mathbf{g}$ (called the *bracket*) such that for all $x, y, z \in \mathbf{g}$,

1. $[x, x] = 0$ (implies anticommutivity),

[7] An analytic function is a function that is locally given by a convergent power series

2. $[x,[y,z]]+[y,[z,x]]+[z,[x,y]]=0$ (Jacobi identity)

The mathematical expression, $(\mathbf{g}\times\mathbf{g}\rightarrow\mathbf{g})$ means that for any two values $a,b\in\mathbf{g}$ $[a,b]=c\in\mathbf{g}$.
A Lie algebra $\mathbf{g}$ is a logarithm of a Lie group, and a Lie group is an exponential of a Lie algebra. The Lie algebra must transform by exponentiation into a Lie group. It is important to note that the formula $\exp(A)\exp(B)=\exp(A+B)$ holds only for a commutative Lie algebra - $U(1)$
Optimization on Lie-groups then consists of performing the differentiation of the subject function $J(p)$ at the point p on the space of the Lie-algebra to obtain the directional derivative and then mapping the directional derivative back to the Lie-group via exponential mapping. The derivative calculation will follow in more details in next section.

Theorem 3.8.1. *[90] For every element* $\mathbf{x}$ *in* $\mathbf{g}$ *there is a unique map* $R\rightarrow G$ *called the* one-parameter subgroup *of* G, *such that* $\dot{\phi}(0)=\mathbf{x}$.
The exponential mapping $\mathbf{g}\rightarrow G$ *is controlled by a single parameter* $t\in R$ *such that the one parameter subgroup, defined as* $t\rightarrow\phi(t)$, *generated by* $\mathbf{x}$ *is denoted by* $t\rightarrow exp(t\mathbf{x})$. *This one parameter subgroup emanates from the identity point on the Lie group* $\phi(0)=e$.

Now, as a practical application on the concept of Lie- algebra and its connection to one parameter subgroups, consider $\mathbf{SO}(n)$, the special orthogonal group. The unique one parameter subgroup in $\mathbf{O}(n)$ emanating from the identity $\mathbf{I}$ in the direction $\mathbf{X}\in so(n)$ (where $so(n)$ is the corresponding lie algebra of the Special orthogonal group $SO(n)$) [8] is given by the formula:

$$t\mapsto e^{\mathbf{X}t}$$

The matrices $\mathbf{X}$ are characterized by a skew-symmetric structure. We thus can say that both the geodesics in $\mathbf{O}(n)$ and parallel translations along geodesics in $\mathbf{O}(n)$ may be computed via matrix exponentiation of skew-symmetric matrices [91].
The orthogonal group which can be considered as a special case of a Stiefel manifold is of paramount importance in applications such as blind signal separation. As previously discussed in the preceding chapter, data are preprocessed by whitening. This operation transforms the mixing and demixing matrices to lie in the orthogonal group. It follows that the geometry and optimization techniques developed for the Stiefel manifold can thus eliminate the need for projective and Lagrange multiplier methods and improve the performance of the separation. More details on the optimization techniques and their application to blind signal separation will follow in the next section.
Another sort of invariance on Lie groups is that moving the inner product from one point in the space to another point (by mapping the elements) does not change the value of the output. This is very much the same as the invariance principle under the change of basis vectors on which the matrix acts. As a matter of fact the domain of Lie group and Lie algebra is quite broad. What is presented here is the minimum necessary information related to our problem at hand.

[8] Having the Lie algebra as a group of matrices instead of vectors does not violate the assumption of the Lie algebra as vector space as each matrix can be considered as a collection of vectors

3.9 From Tangent Space to Geodesics: the Exponential Mapping

This section presents the mapping between tangent spaces and geodesics. This mapping as expected is directly connected to the concepts of Lie groups and Lie algebra. The target here is to perform this mapping along the geodesic we are interested in, namely the shortest distance geodesic.

Theorem 3.9.1. *[42] Let M be a connected differential manifold, then there exists a single geodesic giving the shortest distance between any two points on the manifold.*

Interestingly, the one parameter subgroup emanating for the identity element e and developed by exponential mapping of the Lie algebra to the Lie group is actually this shortest distance geodesic on the corresponding manifold. This is proven using the following theorem.

Theorem 3.9.2. *[42] Let M be a connected differential manifold, then there exists around any point p on the manifold a set of local co-ordinates $(x_1, \ldots, x_m)$ called "***normal neighborhood***" (i.e. inertial frames).*
A neighborhood N_p around a point p is a normal neighborhood if and only if $N_p = N_0$, where N_0 is the neighborhood around the identity element or the origin in the tangent space at the origin.

The above theorem indicates that inertial frames travel along the manifold by exponential mapping of the inertial frame at the origin. Recalling the important property of inertial frames of having zero *Christoffel coefficients*, then exponential mapping guarantees better tracking of the derivatives and ensures that the search direction (when done on Lie algebra space) will lead to points on the surface of the intended manifold. The importance of the above two theorems ensure that the exponential moves the gradient along the shortest distance geodesic on the Riemannian manifold (both unique and traveling along the inertial frames).
To summarize the process of exponential mapping, one can say that exponential mapping Exp_p maps the tangent space T_p to M. It sends a vector $\mathbf{v}$ on T_p to a point q on the manifold at a distance $|\mathbf{v}|$ along the geodesic in the direction of $\mathbf{v}$. The exponent of a matrix X can be defined as:

$$Exp(\mathbf{X}) = \mathbf{I} + \mathbf{X} + \frac{\mathbf{X}^2}{2!} + \ldots\ldots \tag{3.17}$$

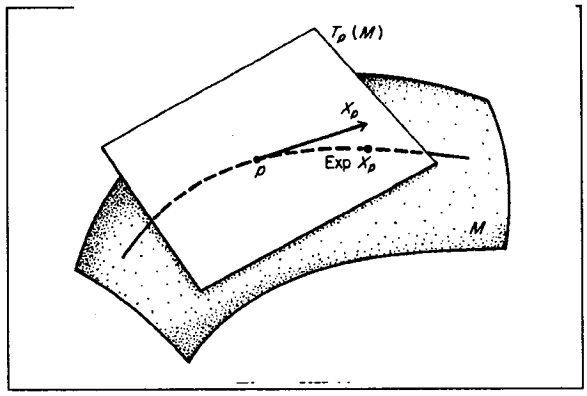

Figure 3.5: **Exponential mapping: from tangent to geodesic [15]**

3.10 Differentiation on Riemannian Manifold

The discussion in section 3.2 concluded that the Euclidean gradient of a function J with respect to a variable $\mathbf{w}$ does not represent the actual gradient for the Riemannian manifold. To calculate the natural gradient one could follow the following derivation [6]:
Consider:

$$d\mathbf{w} = \varepsilon \mathbf{a} \tag{3.18}$$

where ε is a small value, then search for a that minimizes:

$$J(\mathbf{w} + d\mathbf{w}) = J(\mathbf{w}) + \varepsilon \nabla_{\mathbf{w}} J(\mathbf{w}) \cdot \mathbf{a} \tag{3.19}$$

under the constraint of having $\mathbf{a}$ with a unit norm i.e.

$$\|a\|^2 = \sum_{ij} g_{ij} a_i a_j = 1 \tag{3.20}$$

This constrained optimization can be re-written in the Lagrange multiplier form as:

$$\frac{1}{a_i}\{\nabla_{\mathbf{w}} J(\mathbf{w})^T \mathbf{a} - \lambda \mathbf{a}^T \mathbf{g} \mathbf{a}\} = 0 \tag{3.21}$$

The solution is given as:

$$\mathbf{a} = \frac{1}{2\lambda} \mathbf{g}^{-1} \nabla_{\mathbf{w}} J(\mathbf{w}) \tag{3.22}$$

where λ is determined from the constraint. From the above equation we can actually re-define the natural gradient on the Riemannian space as in the following Theorem.

Theorem 3.10.1. *The steepest descent direction of a function $J(\mathbf{w})$ on the Riemannian manifold is given by the following modified natural gradient:*

$$\tilde{\nabla}_{\mathbf{w}}(J(\mathbf{w})) = \mathbf{g}^{-1} \nabla_{\mathbf{w}}(J(\mathbf{w})) = \sum_{ij} g^{ij} \nabla_{\mathbf{w}ij}(J(\mathbf{w})) \tag{3.23}$$

However, when performing the optimization on the Riemannian manifold using Lie group methods, the gradient that should be calculated is the gradient along the Lie algebra space, i.e. directional derivative. Assuming the Lie algebra space to be $\mathbf{h}$, one can write this Lie-algebra derivative as follows:

Definition 3.10.1. [90] For a Riemannian manifold M with Riemannian structure $\mathbf{g}$ and a given function J, the gradient of J at a certain point, denoted as $\nabla_{\mathbf{h}} J|_p$ where $\mathbf{h}$ is Lie algebra, is the unique vector in the tangent space such that:

$$dJ|_p/d\mathbf{x} = \langle \nabla_{\mathbf{h}} J|_p, \mathbf{x} \rangle \tag{3.24}$$

for all $\mathbf{x} \in T_p$.

The above definition means that the derivative of the function J in the direction a tangent vector $\mathbf{x}$ in the tangent space is equal to the inner product of the $\nabla_{\mathbf{h}} J|_p$ and that tangent vector.
This directional derivative is directly related to the derivative with respect to the original variable $\mathbf{w}$ on the corresponding Lie-group as:

$$\mathbf{w}(t) = \exp(\mathbf{h})\mathbf{w}_0 \tag{3.25}$$

then taking the derivative with respect to t yields:

$$\frac{d\mathbf{w}(t)}{dt} = \frac{d\mathbf{h}(t)}{dt} \exp(\mathbf{h})\mathbf{w}_0 = \frac{d\mathbf{h}(t)}{dt}\mathbf{w}(t) \tag{3.26}$$

From the discussion in section 3.2:

$$d\mathbf{w}/dt = -\mu \tilde{\nabla}_{\mathbf{w}} J \tag{3.27}$$

Similarly, on the flat Lie-algebra space

$$d\mathbf{h}/dt = -\mu \nabla_{\mathbf{h}} J \tag{3.28}$$

Applying this equation to equation (3.26):

$$\tilde{\nabla}_{\mathbf{w}} J = \nabla_{\mathbf{h}} J \times \mathbf{w} \tag{3.29}$$

Moving along the Riemannian space using the above gradient as the search direction will guarantee moving along geodesics in the plane. In most cases, the motion will be globally along several geodesics, depending on the inertial frame shortest distance geodesic.
Having provided all the basic information needed to work on Riemannian space, the following sections will present a practical application of these concept to the problem the thesis is dedicated for, namely blind signal separation. The following is a derivation of the natural gradient and other higher order optimization algorithms for BSS based on the maximum-likelihood principle.

3.11 Natural Gradient Derivation for Maximum-Likelihood Principle

In [6], [97], the natural gradient for maximums-likelihood blind signal separation was defined by the Riemannian metric. The natural gradient is a first order gradient calculated along the *actual* steepest descent direction of the Riemannian manifold and thus follows the equation:

$$\tilde{\nabla}_{\mathbf{w}}(J(\mathbf{w})) = \mathbf{g}^{-1}\nabla_{\mathbf{w}}(J(\mathbf{w})) = \sum_{ij} g^{ij}\nabla_{\mathbf{w}ij}(J(\mathbf{w})) \tag{3.30}$$

The main contribution of Amari, was to calculate the metric $\mathbf{g}$ or as referred to in his publications G on the most general space of lie groups , the general linear group $\mathbf{GL}(n)$. The reason for choosing this most general Lie group is that it contains all $n \times n$ real matrices with the property of being invertible, which is the only constraint that can be imposed on the general blind signal separation problem.

Remark 3.11.1. The target of the blind signal separation is to find the de-mixing matrix $\mathbf{W}$ which is the inverse of the matrix $\mathbf{A}$ and thus the target matrix $\mathbf{W}$ is always invertible [9] from the definition of the problem.

Before beginning the derivation, one must note that the space on which the optimization is carried out consists of different points p which are in fact matrices $\mathbf{W}_k, k = 0....$ not just general points as in the previous discussion of the Riemannian mathematics. Thus, the metric tensor $\mathbf{g}$ will have double the co-ordinates that were previously assumed for vectors in the previous general derivation. The vector fields constituting the tangent space will be matrices and the same goes for the vector fields of the Lie algebra.
In the derivation, the following two fact are considered:

1. The Lie group invariance implies that the metric is the same on all the points of the manifold:
$$\langle d\mathbf{W}, d\mathbf{W}\rangle_{\mathbf{W}} = \langle d\mathbf{W}Y, d\mathbf{W}Y\rangle_{\mathbf{W}Y}$$
2. The identity matrix $\mathbf{I}$ on the manifold is the 0-diffeomorphic point (identity point) at which the Riemannian manifold holds the normal co-ordinates and is equal to the Euclidean. i.e. the metric tensor $\mathbf{g} = \mathbf{I}$.

Let $d\mathbf{W}$ be a small deviation of the demixing matrix from $\mathbf{W}$ to $\mathbf{W} + d\mathbf{W}$ then the tangent space $T_{\mathbf{W}}$ (the Lie algebra of $\mathbf{GL}(n)$) contains all such small deviations $d\mathbf{W}_{ij}$.
Then $g_W(d\mathbf{W}, d\mathbf{W})$ to be the metric at the point $\mathbf{W}$ is:

$$\mathbf{g}_{\mathbf{W}}(d\mathbf{W}, d\mathbf{W}) = \langle d\mathbf{W}, d\mathbf{W}\rangle = \|d\mathbf{W}\|^2 \tag{3.31}$$

Now mapping the matrix $\mathbf{W}$ to the identity matrix $\mathbf{I}$ requires to right multiply $\mathbf{W}$ by $\mathbf{W}^{-1}$ then the tangent $d\mathbf{X} = d\mathbf{W}\mathbf{W}^{-1}$ at the identity corresponds to $d\mathbf{W}$ at $\mathbf{W}$ and has the same length (refer to the first fact listed above).

$$\langle d\mathbf{W}, d\mathbf{W}\rangle_{\mathbf{W}} = \langle d\mathbf{X}, d\mathbf{X}\rangle_{\mathbf{I}}$$

[9] Assuming that the number of sources is equal to the number of sensors and that thus the mixing matrix $\mathbf{A}$ is not singular

$$\langle d\mathbf{W}, d\mathbf{W}\rangle_{\mathbf{W}} = \langle d\mathbf{W}\mathbf{W}^{-1}, d\mathbf{W}\mathbf{W}^{-1}\rangle_{\mathbf{I}} \tag{3.32}$$

Using the second fact, the metric of $d\mathbf{X}$ as in the Euclidean space can simply be calculated as:

$$\langle d\mathbf{X}, d\mathbf{X}\rangle_{\mathbf{I}} = \sum_{i,j}(d\mathbf{X}_{ij})^2 = tr(d\mathbf{X}^T d\mathbf{X}) \tag{3.33}$$

Thus,the Riemannian metric structure at $\mathbf{W}$ is:

$$\langle d\mathbf{W}, d\mathbf{W}\rangle_{\mathbf{W}} = tr(\mathbf{W}^{-1})^T d\mathbf{W}^T d\mathbf{W}\mathbf{W}^{-1} \tag{3.34}$$

Then, one can extract the metric $\mathbf{g}_{ij,kl}$ from the above equation:

$$\|d\mathbf{W}\|^2 = \sum \mathbf{g}_{ij,kl} d\mathbf{W}_{ij} d\mathbf{W}_{kl} \tag{3.35}$$

$\mathbf{g}_{ij,kl}$ are thus given by the following formula;

$$\mathbf{g}_{ij,kl} = \sum_{m} \delta_{ik} \mathbf{W}^{-1}_{jm} \mathbf{W}^{-1}_{lm} \tag{3.36}$$

It is actually the inverse of $\mathbf{g}$ that is needed to calculate the derivative on the Riemannian manifold. This inverse is go which is given in closed form as $\mathbf{W}^T\mathbf{W}$.
The true gradient on the Riemannian manifold (known as the natural gradient) is given as:

$$\tilde{\nabla}_{\mathbf{W}} J = \nabla J \mathbf{W}^T \mathbf{W} \tag{3.37}$$

where $\tilde{\nabla}(.)$ denotes the natural gradient.

3.12 Optimization along Geodesics

The previous sections presented the main basic items needed in the Riemannian geometry understanding and how the natural gradient relates to the differential geometry of the Riemannian manifold. However, it is still not clear what the relation of geodesics and parallel translations is, and what a more in depth use of the Riemannian manifold could be. The question still arises: How to make use of such geometric properties in our optimization?
The derivation of the natural gradient presented a first step in the optimization in the Riemannian manifolds. It provided an accurate steepest descent flow in the Riemannian space. However, it did not perform the search along the geodesic.
While the gradient flow methods have only linear convergence rate, one can make use of the search direction obtained with generalized Newton methods or Conjugate gradient algorithms.
From this point, our derivation goes one step further to calculate the gradient on the Lie-algebra space $\tilde{\nabla}_{\mathbf{H}} J(\mathbf{W})$, i.e. the optimal search direction along geodesic. Equation 3.38 presented a direct link between the Lie algebra gradient $\tilde{\nabla}_{\mathbf{H}} J(\mathbf{W})$ and the gradient of $\tilde{\nabla}_{\mathbf{W}} J(\mathbf{W})$, where $\mathbf{H}$ is the Lie algebra associated with the Lie group. For the general maximum-likelihood this is the general linear algebra $\mathbf{H} \in gl$.

$$\tilde{\nabla}_{\mathbf{w}} J = \tilde{\nabla}_{\mathbf{H}} J \times \mathbf{w} \tag{3.38}$$

Applying the above equation tn the expression (3.37), then :

$$\tilde{\nabla}_{\mathbf{H}} J = \nabla J \mathbf{W}^T \tag{3.39}$$

the new value of the matrix $\mathbf{W}$ is given by:

$$\mathbf{W}_{k+1} = \exp(\mu \tilde{\nabla}_{\mathbf{H}} J)\mathbf{W}_k \tag{3.40}$$

where k is the iteration number. It is easily verified that the above expression can be approximated to the original expression of the natural gradient update using $\tilde{\nabla}_{\mathbf{W}} J$:

$$\mathbf{W}_{k+1} = \exp(\mu \tilde{\nabla}_{\mathbf{H}} J)\mathbf{W}_k = (I + \mu \tilde{\nabla}_{\mathbf{H}} + (\mu \tilde{\nabla}_{\mathbf{H}})^2/2 + ...) \tag{3.41}$$

By ignoring the higher order terms the expression reduces to the steepest descent update using $\tilde{\nabla}_{\mathbf{W}} J$. This indicates that this expression is the generalized update and also suggests that if the search is carried out using higher order methods, where the possibility of divergence increases, the search should be carried out along the Lie algebra, i.e. along the geodesic in the corresponding Lie group to have a better opportunity to track the surface of the optimization manifold.
The parallel translation can also be calculated along the tangent vector space, i.e. Lie algebra. Recall that the main property of the parallel transpose is to have the new vector field parallel to the new point on the manifold and to maintain the length. This can be done by the following equation for a general vector field $\mathbf{X}(k)$ in the tangent space of $\mathbf{W}(k)$:

$$\mathbf{X}(k+1) = \mathbf{X}(k)\mathbf{W}^{-1}(k)\mathbf{W}(k+1) \tag{3.42}$$

These sets of derivatives and operations will be particularly useful when performing line search for certain methods like Newton-based or Conjugate gradient methods.

3.13 Optimization along a Stiefel Manifold

One of the constrained manifold that exhibits special importance for blind signal separation is the Stiefel manifold. The reason for this is that after the whitening operation the de-mixing matrix $\mathbf{W}$ follows the following equation:

$$\mathbf{W}\mathbf{W}^T = \mathbf{I} \tag{3.43}$$

Along with the above orthogonality constraint, for any cost function involving pre-whitened mixtures, $J(\mathbf{W})$:

$$J(\mathbf{W}) \neq J(\mathbf{Q}\mathbf{W}) \tag{3.44}$$

for any Hermitian matrix $\mathbf{Q}$. This extra property places the de-mixing matrix $\mathbf{W}$ in a sub-manifold of the orthogonal group known as the Stiefel manifold.
In order to derive the first order derivative of the Stiefel-manifold, the equations previously derived for the natural gradient of the GL-group are needed. These equations still apply for the Stiefel manifold as it is a subset in to the GL-group. What should be added to these

equations is the use of the new constraint of orthogonality.
Recalling that the natural gradient is given by:

$$\tilde{\nabla}_{\mathbf{H}} J = \nabla_{\mathbf{W}} J \times \mathbf{W}^T = \frac{d\mathbf{W}}{dt}\mathbf{W}^T \tag{3.45}$$

and differentiating equation 5.179, yields:

$$\frac{d\mathbf{W}}{dt}\mathbf{W}^T + \mathbf{W}(\frac{d\mathbf{W}}{dt})^T = 0 \tag{3.46}$$

which can be written as:

$$\tilde{\nabla}_{\mathbf{H}} J + \tilde{\nabla}^T_{\mathbf{H}} J = 0 \tag{3.47}$$

meaning that the Lie algebra is skew-symmetric. This has been mentioned for the examples of *so* Lie-algebra.
Thus, the Lie-algebra $\mathbf{H}$ can be written as:

$$\tilde{\nabla}_{\mathbf{H}} J = \text{skew}(\tilde{\nabla}_{\mathbf{H}} J) = 1/2(\tilde{\nabla}_{\mathbf{H}} J + \tilde{\nabla}^T_{\mathbf{H}} J) \tag{3.48}$$

Meaning that:

$$\tilde{\nabla}_{\mathbf{W}} J = (\tilde{\nabla}_{\mathbf{W}} J)\mathbf{W}^T\mathbf{W} + \mathbf{W}(\tilde{\nabla}^T_{\mathbf{W}} J)\mathbf{W} \tag{3.49}$$

This is the form of the natural gradient for pre-whitened blind signal separation algorithms. In [91] and [31], second order methods including Newton's method and conjugate gradient algorithms have been developed. These methods rely on exploring the special geometry of the Stiefel manifold and optimizing the search direction along the geodesic.
After a rather lengthy derivation [31], the geodesic on the Stiefel manifold in the special case of orthogonal group (i.e. the matrix $\mathbf{W}$ is of square of dimension N-by-N) is given by the following equation , define :

$$\mathbf{W}(t) = \mathbf{W}\mathbf{M}(t) + \mathbf{Q}\mathbf{N}(t) \tag{3.50}$$

where $\mathbf{W}(t)$ is the new location along the geodesic at time t and $\mathbf{W}$ is the original location $\mathbf{W}(0)$. For $\mathbf{H}$ being the direction of the derivative at $\mathbf{W}(0)$ and matrix $\mathbf{A}$ such that: $\mathbf{A} = \mathbf{W}^T\mathbf{H}$ is skew-symmetric then using the compact QR-decomposition:

$$\mathbf{QR} : \mathbf{K} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{H} \tag{3.51}$$

where both $\mathbf{Q}$ and $\mathbf{R}$ are N-by N matrices. The matrices $\mathbf{M}(t)$ and $\mathbf{N}(t)$ used in equation (3.50) are given by:

$$\begin{pmatrix} \mathbf{M}(t) \\ \mathbf{N}(t) \end{pmatrix} = \exp t \begin{pmatrix} \mathbf{A} & -\mathbf{R}^T \\ \mathbf{R} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \tag{3.52}$$

where $\mathbf{I}$ is the identity matrix of size N-by N.
Now, having defined the geodesic equation and the first order derivative over the Stiefel manifold, it is possible to move forward to the main two second order update algorithms on the Stiefel manifold. The next two sections present the Newton-algorithm and the conjugate gradient algorithm developed in [31]. The presentations of such algorithms are rather brief. More details about the derivation of the algorithms and justifications of the update equations are presented in [31].

3.14 Newton Method along the Stiefel Manifold [31]

In the Newton method, the aim is focused on finding the update Δ which satisfies:

$$\Delta = -\text{Hess}^{-1}\tilde{\nabla}_{\mathbf{H}}J \tag{3.53}$$

where the Hessian, Hess, is defined by the quadratic form:

$$\text{Hess}J(\Delta,\Delta) = \frac{d^2}{dt^2}|_{t=0}J(\mathbf{W}(t)) \tag{3.54}$$

where $\mathbf{W}(t)$ is the geodesic and Δ is the tangent at time $t = 0$.
A key equation to calculating the Hessian is the second covariant tensor equation for the Riemannian manifold [90]. For the function $J : M \rightarrow R$ and the co-ordinate chart $(U, x^1, \ldots, x^n)$, the tensor takes the form [91]

$$Hess(J) = \sum_{i,j}((\frac{\partial^2 J}{\partial x^i \partial x^j}) - \sum_k \Gamma^k_{ji}(\frac{\partial J}{\partial x^k}))dx^i \otimes dx^j \tag{3.55}$$

This same equation is more conveniently presented in [31] in terms of the tangents Δs and the geodesic curve instead of abstract chart components as:

$$\text{Hess}J(\Delta_1,\Delta_2) = \nabla_{\mathbf{WW}}J(\mathbf{W})(\Delta_1,\Delta_2) - \text{tr}\nabla^T_{\mathbf{W}}J(\mathbf{W})\Gamma(\Delta_1,\Delta_2) \tag{3.56}$$

where $\nabla_{\mathbf{WW}}J(\mathbf{W})(\Delta_1,\Delta_2)$ represents the tensor whose scalar components are $(\nabla_{\mathbf{WW}}J)_{ij,kl}(\Delta_1)_{ij}(\Delta_2)_{kl}$:

$$(\nabla_{\mathbf{WW}}J(\mathbf{W}))_{ij,kl} = \frac{\partial^2 J}{\partial \mathbf{W}_{ij}\partial \mathbf{W}_{kl}} \tag{3.57}$$

Following the derivation in [31] the solution of the equation (3.56) along with the geodesic equation (3.14) to get Δ reduces to solving the linear equation:

$$\nabla_{\mathbf{WW}}J(\Delta) - \mathbf{W}\text{skew}(\nabla_{\mathbf{W}}J(\mathbf{W})^T\Delta) - \text{skew}(\Delta\nabla_{\mathbf{W}}J(\mathbf{W})^T)\mathbf{W} - \frac{1}{2}(\mathbf{I} - \mathbf{W}\mathbf{W}^T)\Delta\mathbf{W}^T\nabla_{\mathbf{W}}J(\mathbf{W}) = -G \tag{3.58}$$

where:

$$G = \tilde{\nabla}_{\mathbf{W}}J(\mathbf{W}) \tag{3.59}$$

The following table summarizes the Newton method on the Stiefel manifold:

Table 3.1: **Newton method for minimizing J(W) on Stiefel manifold [31]**

1. Compute $\mathbf{G} = \tilde{\nabla}_W J = \nabla_{\mathbf{W}} J(\mathbf{W}) - \mathbf{W}\nabla_{\mathbf{W}}^T J(\mathbf{W})\mathbf{W}$

2. Compute $\Delta = -\text{Hess}^{-1}\tilde{\nabla}_W J(\mathbf{W})$ such that $\mathbf{W}^T\Delta$ = skew symmetric

$$\nabla_{\mathbf{WW}} J(\Delta) - \mathbf{W}\text{skew}(\nabla_{\mathbf{W}} J(\mathbf{W})^T\Delta) - \text{skew}(\Delta\nabla_{\mathbf{W}} J(\mathbf{W})^T)\mathbf{W} - \tag{3.60}$$

$$\frac{1}{2}(\mathbf{I} - \mathbf{W}\mathbf{W}^T)\Delta\mathbf{W}^T\nabla_{\mathbf{W}} J(\mathbf{W}) = -\tilde{\nabla}_{\mathbf{W}} J(\mathbf{W})$$

3. Update along the geodesic $\mathbf{W}(t)$ in the direction Δ using the formula:

$$\mathbf{W}(t) = \mathbf{W}\mathbf{M}(t) + \mathbf{Q}\mathbf{N}(t) \tag{3.61}$$

where:

$$\begin{pmatrix} \mathbf{M}(t) \\ \mathbf{N}(t) \end{pmatrix} = \exp t \begin{pmatrix} \mathbf{A} & -\mathbf{R}^T \\ \mathbf{R} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \tag{3.61}$$

and:

$$\mathbf{A} = \mathbf{W}^T\mathbf{H}, \qquad \text{is skew-symmetric}$$
$$\mathbf{QR} : \mathbf{K} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{H}$$

3.15 Conjugate-Gradient Method along Stiefel Manifold [31]

Another popular optimization algorithm is the conjugate gradient. The conjugate gradient belongs to a family of optimization algorithms collectively known as "*Conjugate-direction methods*". This family of optimization techniques was developed to achieve better than linear convergence without requiring the calculation and the inversion of the Hessian matrix. It would be useful to begin with a review of the basic idea behind the conjugate gradient in its basic form, (i.e. linear conjugate gradient on Euclidean flat space). The conjugate gradient tries to improve over the steepest descent by modifying the search direction. Optimizing a cost function $J(x)$, steepest descent method naturally begins with calculating the update from the initial point p_0 in the direction of the first gradient $\nabla J(p_0)$ at the initial point p_0. The idea with the conjugate gradient is that when calculating the update is performed not in the gradient direction but in a direction that is perpendicular to the previous gradient $\nabla J_{(p_0)}$. The following figure illustrates the idea [32].
In Figure(3.6), the direction of steepest descent at point p_0 is $g_0 = \nabla J_{p0}$ and h_1 is the direction of the gradient conjugate to g_0 at the point p_1. However, the perpendicular direction to the gradient cannot be developed immediately [43], [71]. Alternatively, a new search direction is formed as a combination of the old search direction and the new gradient. Naming

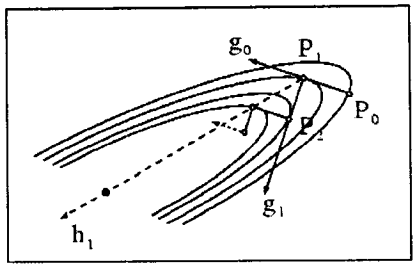

Figure 3.6: **Illustrating conjugate gradients [32]**

the search direction at step k as H_k the search update equation translates to:

$$H_k = \nabla J_k + \gamma_k H_{k-1} \tag{3.60}$$

One crucial enhancement to the computational efficiency of the conjugate gradient method is to relate the step size γ_k to the Hessian. The trick is to approximate the Hessian as the finite difference between the gradients. This leads to different sub-algorithms in the conjugate gradient method. The exact formula for conjugacy, known as Polak-Ribière update equation can be written as:

$$\gamma_k = \frac{\langle \tilde{\nabla}_{\mathbf{W}} J_k - \tilde{\nabla}_{\mathbf{W}} J_{k-1}, \tilde{\nabla}_{\mathbf{W}} J_k \rangle}{\langle \tilde{\nabla}_{\mathbf{W}} J_k, \tilde{\nabla}_{\mathbf{W}} J_k \rangle} \tag{3.61}$$

However, if the cost function $J(\mathbf{W})$ is well approximated in the quadratic form, the higher order term in the above equation can be ignored and this results in the most widely used Fletcher-Reeves update:

$$\gamma_k = \frac{\langle \nabla J_k, \nabla J_k \rangle}{\langle \nabla J_{k-1}, \nabla J_{k-1} \rangle} \tag{3.62}$$

The above conventional conjugate gradient, used for unconstrained optimization, can be carried over to the Stiefel manifold. The main changes include updating along geodesics and the definitions of inner products. Figure (3.7) demonstrates the principle of conjugate gradient algorithm on a curved surface.

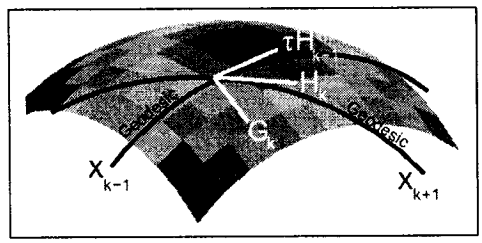

Figure 3.7: **Conjugate gradients in a curved space. [31]**

The first observation from the above figure is that the old search direction H_{k-1} in the update equation on the flat space equation(3.60) should be replaced by a parallel translated vector τH_{k-1}. The update equation can thus be:

$$H_k = \tilde{\nabla}_{\mathbf{W}} J_k + \gamma_k \tau H_{k-1} \tag{3.63}$$

However, the calculation of τH_{k-1} is not complicated. Essentially the parallel translation is performed along the geodesic of the space, which has already been identified. Following the same guide-lines as for the general conjugate-gradient (Fletcher-Reeves), the "***Stiefel-constrained***" conjugate-gradient algorithm can be written as in Table 3.2 [31]:

Table 3.2: **Conjugate-gradient method for minimizing J(W) on Stiefel manifold [31]**

1. Compute $\mathbf{G}_0 = \tilde{\nabla}_{\mathbf{W}} J_0 = \nabla_{\mathbf{W}_0} J(\mathbf{W}) - \mathbf{W}_0 \nabla_{\mathbf{W}_0} J(\mathbf{W})^T \mathbf{W}_0$ and set $\mathbf{H}_0 = -\mathbf{G}_0$
2. For $k = 0, 1, \ldots$
   - Minimize $J(\mathbf{W}_k(t))$ over t where $\mathbf{W}(t)$ is the geodesic described by the equations:

$$\mathbf{W}(t) = \mathbf{W}\mathbf{M}(t) + \mathbf{Q}\mathbf{N}(t) \tag{3.64}$$

   where:

$$\begin{pmatrix} \mathbf{M}(t) \\ \mathbf{N}(t) \end{pmatrix} = \exp t \begin{pmatrix} \mathbf{A} & -\mathbf{R}^T \\ \mathbf{R} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \end{pmatrix} \tag{3.65}$$

   and:

$$\mathbf{A} = \mathbf{W}^T\mathbf{H}, \qquad \text{is skew-symmetric}$$
$$\mathbf{QR} : \mathbf{K} = (\mathbf{I} - \mathbf{W}\mathbf{W}^T)\mathbf{H}$$

   - Compute $\mathbf{G}_{k+1} = \nabla_{\mathbf{W}} J_{k+1} - \mathbf{W}_k(t)(\nabla_{\mathbf{W}} J_{k+1})^T \mathbf{W}_k(t)$
   - Parallel transpose $\mathbf{H}_k$ to the point W_{k+1}.
   - Compute the new search direction:
     $\mathbf{H}_k = \mathbf{G}_k + \gamma_k \tau \mathbf{H}_{k-1}$ where:
     $\gamma_k = \frac{\langle \mathbf{G}_{k+1}, \mathbf{G}_k \rangle}{\langle \mathbf{G}_k, \mathbf{G}_k \rangle}$ and the inner product is defined as:
     $\langle \Delta_1, \Delta_2 \rangle = \mathrm{tr} \Delta_1^T (\mathbf{I} - \frac{1}{2}\mathbf{W}\mathbf{W}^T)\Delta_2$

3.16 Conclusion

This chapter includes the mathematical background for the Riemannian manifold that will be used along the thesis. The chapter begins with explaining the need for studying the Riemannian geometry in order to develop a more efficient optimization algorithm for several mathematical problems including the blind signal separation problem. The geometry review started with explaining the concept of a manifold, tangent and normal spaces. These definitions were used to introduce the concept of curvature which is the main property of the Riemannian manifold. The parallel transposition was then introduced followed by the

definition of geodesics which is pivotal in the motion along Riemannian spaces. The mathematical representation of manifolds as Lie groups was introduced along with examples of different manifolds like the general linear group and the Stiefel manifold. In the later sections, the application of Riemannian geometry to the maximum-likelihood blind source separation was introduced, along with the introduction of the idea of using higher order optimization algorithms for Riemannian space. The chapter concluded with presenting two of these algorithms, namely the Newton and Conjugate-gradient, which were already developed in the literature for the case of the Stiefel-manifold.

Chapter 4

Improvements to Instantaneous Blind Signal Separation Algorithms: Pre-Whitened Case

4.1 Introduction

This chapter explores blind signal separation algorithms on the Stiefel manifold. The Stiefel manifold, which is a special sub-manifold on the Riemannian space, is the geometrically optimum optimization space for the orthogonal separating matrices of the pre-whitened mixtures.

As has been presented in Chapter 2, major attention in the literature has been focused on different criteria to be used as objective functions for BSS. However, most of the existing on-line methods can be categorized as stochastic gradient or LMS-based algorithms. The need for second-order based algorithms for BSS can be easily revealed. The fast convergence rate of the RLS-based family can be advantageous for many applications. Fast converging on-line algorithms are especially convenient when dealing with non-stationary mixture cases. Among the many existing objective functions for blind signal separation, Maximum Likelihood and Negentropy stand as strong criteria well justified as minimizers of the mutual information between the original independent components [57]. They are especially suitable for heavy tailed distribution signals, e.g. audio data [65] because they are less sensitive to outliers.

In Chapter 3, it was shown that applying the constraints of the Stiefel manifold to the optimization leads to a modified gradient rather than the ordinary gradient of the objective function. The new constrained gradient is referred to as the natural gradient on the Stiefel manifold. A natural question would be how to extend and combine the performance of the natural gradient with an RLS-like or Newton-like algorithms. The work presented in this chapter targets this question and proposes a collection of algorithms that combine the fast converging RLS-family and the use of the natural gradient.

In the following sections a review of the natural gradient for the maximum-likelihood blind source separation and its form in the Stiefel manifold is presented. This is followed by a review of the Negentropy principle that actually works only in the case of Pre-whitened mixtures (i.e. Stiefel manifold condition). Another cost function considered is the nonlinear

PCA cost function. In addition, a new cost function is developed for the purpose of blind signal separation.

4.2 Existing Maximum-Likelihood Based Algorithms

Let $s_i(t)$, $i = 1, 2, ..., N$ be scalar inputs to the blind signal separation model at time t. For simplicity, we assume that the mixing is linear and that the mixing matrix is square, i.e. the number of inputs N is equal to the number of mixtures x_i, $i = 1, 2, ..., N$. Therefore, the mixing matrix A is a square matrix of size $N \times N$. The mixing model can be expressed as:

$$\mathbf{x}(t) = \mathbf{A} \times \mathbf{s}(t) \tag{4.1}$$

The mixture $\mathbf{x}$ is then applied to the whitening matrix $\mathbf{V}$. The resulting whitened mixtures' vector $\mathbf{z}$ is expressed as:

$$\mathbf{z}(t) = \mathbf{V} \times \mathbf{x}(t) = \mathbf{V} \times \mathbf{A} \times \mathbf{s}(t) = \mathbf{B} \times \mathbf{s}(t) \tag{4.2}$$

where $\mathbf{B}$ is the resulting mixing matrix after the whitening stage. The purpose of the blind signal separation algorithms is to estimate a matrix $\mathbf{W}$ such that $\mathbf{W} \times \mathbf{B} = \mathbf{I}_{N \times N}$, where $\mathbf{I}$ is the identity matrix [1] .

Maximum Likelihood targets separation via increasing the likelihood between the outputs $y_i(n)$, $i = 1, 2, ..., N$ and the inputs $z_i(n)$, $i = 1, 2, ..., N$. The concept has been originally introduced in [22]. The cost function of the log-likelihood $L(\mathbf{W})$ of the demixing matrix $\mathbf{W}$ can be expressed as:

$$L(\mathbf{W}) = \log|\det(\mathbf{W})| + E\{\sum_{i=1}^{N} \log p_i(\mathbf{w}_i \mathbf{z})\} \tag{4.3}$$

where $E\{.\}$ refers to the expected value, $\mathbf{w}_i$ is the i^{th} row of the matrix $\mathbf{W}$ and $p_i(.)$ is the probability density function. In the case of pre-whitened inputs, $\det(\mathbf{W})$ is constant and thus $\log|\det(\mathbf{W})|$ is constant and will not have an impact on the derivative. Therefore, the above cost function can be reduced to:

$$L(\mathbf{W}) = E\{\sum_{i=1}^{N} \log p_i(\mathbf{w}_i \mathbf{z})\} \tag{4.4}$$

which has the gradient $\nabla L(\mathbf{W})$ as:

$$\begin{aligned} \nabla L(\mathbf{W}) &= -E\{\mathbf{g}(\mathbf{W}\mathbf{z})\mathbf{z}^T\} \\ &= -E\{\mathbf{g}(\mathbf{y})\mathbf{z}^T\} \end{aligned} \tag{4.5}$$

where $g(y_i) = -\frac{p_i'}{p_i}$ and is usually set to $\tanh(y_i)$ for super-gaussian data.

Pre-whitening also constrains the matrix $\mathbf{W}$ to be an orthogonal matrix, meaning that

[1] Permutation identity matrix is also accepted as a separation result. However, for simplicity the equation is written for the identity matrix

$\mathbf{W}\mathbf{W}^T = \alpha\mathbf{I}_{N\times N}$. This constraint places the optimization of the cost function on the Stiefel manifold, where we can employ the knowledge of the differential geometry to adjust the original gradient $\nabla L(\mathbf{W})$ to the natural gradient $\tilde{\nabla} L(\mathbf{W})$ that follows the geometry of the manifold as following [30],[31]:

$$\tilde{\nabla} L(\mathbf{W}) = \nabla L(\mathbf{W}) - \mathbf{W}\nabla L(\mathbf{W})^T\mathbf{W} \tag{4.6}$$

$$= -(\mathbf{g}(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{g}(\mathbf{y})^T)\mathbf{W}$$

The natural gradient LMS algorithm is based on taking the instantaneous value of the update of the above gradient so the update equation would be:

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \mu(\mathbf{y}\mathbf{g}(\mathbf{y})^T - \mathbf{g}(\mathbf{y})\mathbf{y}^T)\mathbf{W}_{t-1} \tag{4.7}$$

where μ is the step size that should be small enough to prevent divergence.

4.3 Existing Negentropy Based Algorithms and Relationship to Maximum-likelihood Based Update

Negentropy is one of the principal techniques for independent component analysis. As most blind signal separation algorithms, Negentropy based algorithms require the input mixtures to be whitened either before or during the separation process (on-line whitening). Using the same notation for the mixing/demixing operation as the previous section and noting that the Negentropy requires the data to be whitened as a condition for proper operation, an output y_i of the separation can be expressed as :

$$y_i = (\mathbf{w}_i)^T \times \mathbf{z} = (\mathbf{q}_i)^T \times \mathbf{s} \tag{4.8}$$

where $\mathbf{w}_i$ is the de-mixing vector used to estimate the output y_i from the mixtures set and $\mathbf{q}_i = \mathbf{A}^T\mathbf{V}^T\mathbf{w}_i$. Here, it is important to point out that the separation using Negentropy principle is capable of restoring the sources sequentially and thus is suitable for blind extraction of desired signals while being able to discard the rest of the mixtures. Recalling that for any separation procedure, the separated sources can be recovered with inevitable ambiguity in the sign and magnitude. Thus, a natural choice then is to assume that the input sources s_i are of unit variance. Thus the output y_i will also have unit variance meaning that $\| \mathbf{q}_i \|^2 = 1$: $\| \mathbf{q}_i \|^2 = (\mathbf{w}_i^T \times \mathbf{V} \times \mathbf{A}) \times (\mathbf{A}^T \times \mathbf{V}^T \times \mathbf{w}_i) = \| \mathbf{w}_i \|^2 = 1$.
Hence, the separation algorithm tries to find the vector $\mathbf{w}_i$ such that $\mathbf{w}_i^T$ is one of the rows of the inverse of $\mathbf{V} \times \mathbf{A}$ and has unit norm. As mentioned in Chapter 2, the Negentropy function is difficult to calculate in practice. Several approximations are thus used to provide accurate estimation. Kurtosis, a Higher Order Statistic (HOS), has been extensively used to provide a rather simple approximation. However, it is not very accurate and has several robustness problems. Another approximation is a generalized HOS using other non-quadratic functions to approximate Negentropy. The most convenient approximation of Negentropy is to consider one non-quadratic function to build a cost function of form [57] :

$$J(\mathbf{w}^T\mathbf{z}) = [E\{G(\mathbf{w}^T\mathbf{z})\} - E\{G(v)\}]^2 \tag{4.9}$$

where v is a standardized-Gaussian variable [57]. Choosing $G(y_i) = \log\cosh(y_i)$ gives $g(y_i) = G'(y_i) = \tanh(y_i)$; and $g'(y_i) = 1 - \tanh^2(y_i)$. The tanh function is the most suitable to use among all non-quadratic functions and gives the best results for both super- and sub-Gaussian sources [57],[6], [29].
Differentiating the above equation (4.9) results in the following expression for $\Delta\mathbf{w}$:

$$\Delta\mathbf{w} \propto \frac{\partial[E\{G(\mathbf{w}^T\mathbf{z})\} - E\{G(v)\}]^2}{\partial\mathbf{w}} = 2[E\{G(\mathbf{w}^T\mathbf{z})\} - E\{G(v)\}] \times \mathbf{z} \times g(\mathbf{w}^T\mathbf{z}) \quad (4.10)$$

A stochastic on-line version of equation 4.10 is (as previously presented):

$$\mathbf{w}_+ = \mathbf{w}_{t-1} + \mu_r\gamma\mathbf{z}_t g(\mathbf{w}_{t-1}^T\mathbf{z}_t) \quad (4.11)$$

$$\mathbf{w}_t = \mathbf{w}_+/\|\mathbf{w}_+\|$$

where $\gamma = [E\{G(\mathbf{w}^T\mathbf{z})\}\} - E\{G(v)\}]$ can be set to -1, which is very suitable for super-gaussian sources like speech [57]. However this gradient algorithm does not work well in practice and another algorithm was derived in [6] that depends on a Newton-type update of equation (4.10) and thus works in batch mode. This algorithm, referred to in the literature as the Fast-ICA, has been summarized in Table 2.2. The algorithm depends on the following main equation as an update :

$$\mathbf{w}_+ = E\{\mathbf{z}g(\mathbf{w}_{t-1}^T\mathbf{z})\} - E\{g'(\mathbf{w}_{t-1}^T\mathbf{z})\}\mathbf{w}_{t-1} \quad (4.12)$$

$$\mathbf{w}_t = \mathbf{w}_+/\|\mathbf{w}_+\|$$

The Fast-ICA works efficiently in Blind Signal Extraction (BSE) and is considered by far the most successful algorithm based on Negentropy. However, the fact that the algorithm only works off-line presents a disadvantage, especially in tracking non-stationary mixtures. Thus, it would be valuable to present an on-line algorithm that combines the advantages of the Negentropy principle (especially the ability to extract as many sources as needed) with a fast convergent on-line operation that should be comparable to other existing on-line algorithms in other categories of blind signal separation. One way to extend the Negentropy algorithm to perform blind signal separation in one step (not sequential) is to modify the same cost function (4.9) for the vector of all outputs $\mathbf{y}$ as:

$$F(\mathbf{y}) = [E\{G(\mathbf{y})\} - E\{G(\mathbf{v})\}]^2 \quad (4.13)$$

$$F(\mathbf{y}) = [E\{G(\mathbf{W}\mathbf{z})\} - E\{G(\mathbf{v})\}]^2$$

where $\mathbf{W}$ is the demixing matrix applied to all the mixtures. The gradient of the above cost function can be simplified to be:

$$\nabla F(\mathbf{y}) = \gamma E\{\mathbf{g}(\mathbf{W}\mathbf{z})\mathbf{z}^T\} = \gamma E\{\mathbf{g}(\mathbf{y})\mathbf{z}^T\} \quad (4.14)$$

Comparing the gradient form of equation (4.5) to the one in equation (4.14), we can see that they are identical. Thus the natural gradient for the Negentropy algorithm when we estimate the whole de-mixing matrix is given by the following equation:

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \mu(\mathbf{y}\mathbf{g}(\mathbf{y})^T - \mathbf{g}(\mathbf{y})\mathbf{y}^T)\mathbf{W}_t \quad (4.15)$$

while the update equation for the natural gradient in the case of blind signal extraction , i.e. estimating each source separately, can be given by:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) + \mu(y_i(t)g(y_i(t))\mathbf{w}_i(t-1) - g(y_i(t))\mathbf{z}(t)) \tag{4.16}$$

where $\mathbf{w}_i$ is the transpose of the i^{th} row of the de-mixing matrix $\mathbf{W}$. Negentropy and Maximum-likelihood have been both separately linked to mutual information [56]. However, laying their direct equivalence, in the pre-whitened case, in this identical equation form illuminates the strong relation between these two main information- theoretic based criteria and puts emphasis on the resulting gradient in the two cases as a theoretically justified separation rule driven from two different roads of Blind Signal Separation.

4.4 Nonlinear PCA and its Relationship With Maximum-Likelihood Principle and Negentropy Principle

In Chapter 2, nonlinear PCA has been introduced as a nonlinear extension to the linear PCA criterion. The nonlinear PCA is defined as the optimum least squares problem:

$$F(\mathbf{W}_t) = E\{\|\mathbf{z}_t - \mathbf{W}_t^T f(\mathbf{y}_t)\|^2\} \tag{4.17}$$

where $f(\mathbf{y})$ is the tanh function for the subgaussian data and the cubic $f(\mathbf{y}) = \mathbf{y}^3$ for the supergaussian data. In [100] a Stiefel-manifold LMS-based algorithm was introduced for the above nonlinear PCA cost function with the form:

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu[f(\mathbf{y}_t)\mathbf{y}_t^T - \mathbf{y}_t f(\mathbf{y}_t)^T]\mathbf{W}_t \tag{4.18}$$

The symmetry between this algorithm and (4.15) is clear. They are identical except for the difference in sign and the nonlinear function in the update. However, a more thorough look through the stability analysis indicates that these two differences cancel each other. According to the stability analysis [89], [22], [30], and defining $g_i'(y_i)$ as the derivative of the score function of the i^{th} element y_i, the update equation (4.15) is stable if:

$$E\{g_i'(y_i)\} - E\{g_i(y_i)y_i\} > 0 \tag{4.19}$$

which is achieved by the choice of the $g(\mathbf{y}) = \tanh(\mathbf{y})$ for supergaussian data and $g(\mathbf{y}) = \mathbf{y}^3$ for the subgaussian data. For the above nonlinear PCA-algorithm the stability condition is given by:

$$E\{f_i(y_i)y_i\} - E\{f_i'(y_i)\} > 0 \tag{4.20}$$

which is achieved by the forth mentioned reversed choices of the score function relative to (4.15), i.e. $f(\mathbf{y}) = \tanh(\mathbf{y})$ for subgaussian data and $f(\mathbf{y}) = \mathbf{y}^3$ for the supergaussian data. Thus, the two algorithms are equivalent. This gives more credibility to equation (4.15) as the generalized on-line update rule for blind signal separation in the case of pre-whitened mixtures while equation (4.16) can be considered as the generalized on-line update rule for blind signal extraction.

4.5 Bussgang Based Cost Function and its Stiefel Manifold Gradient

Another cost function that can be formulated to give the same Stiefel manifold gradient is based on the Bussgang-property [67]. This cost function proposed mainly for blind equalization, and usually used for blind-deconvolution, can be manipulated to be a penalty for the instantaneous mixing scenario. According to the Bussgang-property, the independence condition of the separated output implies that the auto-correlation of the separated sources $\mathbf{y}$ must be equal to the cross-correlation between $\mathbf{y}$ and its nonlinear version $\hat{\mathbf{y}}$,

$$E\{\hat{\mathbf{y}}\mathbf{y}^T\} = E\{\mathbf{y}\mathbf{y}^T\} \tag{4.21}$$

where:

$$\hat{\mathbf{y}} = g(\mathbf{y}) \tag{4.22}$$

and $g(.)$ still follows the same stability conditions as in the previous sections, meaning that $g(.) = \tanh(.)$ for super-gaussian data and $g(.) = (.)^3$ for sub-gaussian data. The Bussgang-cost function can thus be written with a Minimum-Mean Square Error (MMSE)-form:

$$C(\mathbf{W}) = E\{\|\mathbf{y} - g(\mathbf{y})\|^2\} \tag{4.23}$$

At convergence, $\mathbf{s} \approx \hat{\mathbf{y}} \approx g(\mathbf{y})$. Thus, ther term $g(y)$ acts as the refrence signal and is not taken into account in the derivative of the above cost function. The derivative can be approximated to be:

$$\frac{\partial C(\mathbf{W})}{\partial \mathbf{W}} \approx E\{(g(\mathbf{y}) - \mathbf{y})\mathbf{x}^T\} \tag{4.24}$$

The LMS-update for the above equation can be given as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{y}(t) - g(\mathbf{y}(t)))\mathbf{x}^T \tag{4.25}$$

This LMS-form can be easily translated into the natural-gradient form on the GL-group by post-multiplying by $\mathbf{W}\mathbf{W}^T$:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{y}(t) - g(\mathbf{y}(t)))\mathbf{y}^T\mathbf{W}(t-1) \tag{4.26}$$

which is a very similar update equation to Amari's natural gradient based on maximum-likelihood. However, when applying the natural gradient formula for the Stiefel manifold, namely:

$$\widetilde{\Delta}\mathbf{W} = \Delta\mathbf{W}\mathbf{W}^T\mathbf{W} - \mathbf{W}\Delta\mathbf{W}^T\mathbf{W}$$

the equation (4.25), will yield the following update:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{y}(t)g(\mathbf{y}(t)) - g(\mathbf{y}(t))\mathbf{y}^T)\mathbf{W}(t-1) \tag{4.27}$$

which is identical to the forms deduced for the maximum-likelihood cost function, the Negentropy cost function and the Nonlinear PCA cost functions.
It is worth noting that prior equivalence between these cost function has been presented in paper [69]. In this publication [69] the concepts of ML, Negentropy were linked to the

Information Theory cost function and re-defined through the Kullback-Leibler divergence, previously introduced in Chapter 2. However, the equivalence relationship developed in this chapter for the derivatives of the different cost functions on the Stiefel manifold was based directly on the original cost functions for each of the maximum-likelihood as defined by [6], Negentropy as defined in [57], NPCA as defined in [86] and the newly extended Bussgang cost function. Clearly, these cost function yield different update in the general Riemannian manifold. The aim of the discussion in the previous sections, section 4.2 to 4.5 is to state that these different cost functions all have the same update on the Stiefel manifold. The proposed algorithms to be introduced in the following sections can thus be considered as applicable RLS-extensions for any of the three concepts: Maximum-likelihood, Nonlinear PCA, Bussgang-nonlinearity cost function and Negentropy. They are thus named with a prefix Stiefel to highlight the fact that they stem solely from optimizing the cost functions on the Stiefel manifold.

4.6 RLS1-BSE-Stiefel Algorithm Based on Nonlinear PCA Principle

As has been previously mentioned in this chapter, providing fast converging on-line algorithms can be advantageous in many situations. In the following discussion, a new algorithm is developed for blind signal extraction. The algorithm is RLS-based and builds on the Stiefel-manifold update form of (4.16) and thus makes full use of the orthogonality constraints. The proven equivalence between Maximum-likelihood, Negentropy and NPCA under orthogonality condition places the RLS1-BSE-Stiefel algorithm as a solution for pre-whitened mixtures extraction. In order to derive the algorithm, we begin with the natural gradient BSE equation (4.16):

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) + \mu(y_i(t)g(y_i(t))\mathbf{w}_i(t-1) - g(y_i(t))\mathbf{z}(t)) \tag{4.28}$$

or equivalently in terms of $f(y_i)$:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - \mu(y_i(t)f(y_i(t))\mathbf{w}_i(t-1) - f(y_i(t))\mathbf{z}(t)) \tag{4.29}$$

where the definitions of $g(y_i)$ and $f(y_i)$ are the same as in the preceding section. $g(y_i) = \tanh(y_i)$ for the supergaussian data and $g(y_i) = y_i^3$ for the subgaussian data. On the other hand, $f(y_i) = y_i^3$ for the supergaussian data and $f(y_i) = \tanh(y_i)$ for the subgaussian data. The natural cost function $J_i(\mathbf{w}_i)$ can actually be re-written in the least-squares sense as:

$$J_i(\mathbf{w}_i) = E\{\|\mathbf{z}(t) - f(\mathbf{y}(t))\mathbf{w}_i(t)\|^2\} \tag{4.30}$$

A modified version of the above cost function can be obtained by replacing the mean-square error by an exponentially weighted sum of errors as [84]:

$$J_i(\mathbf{w}_i) = \sum_{m=1}^{t} \beta^{t-m}\|\mathbf{z}(m) - f(\mathbf{y}(m))\mathbf{w}_i(t)\|^2 \tag{4.31}$$

where the forgetting factor $\beta < 1$ is close to 1. The gradient $\nabla J(\mathbf{w}_i) = \frac{\partial J(\mathbf{w}_i)}{\partial \mathbf{w}_i}$ is given by the equation:

$$\nabla J(\mathbf{w}_i) = \sum_{m=1}^{t} \beta^{t-m}(-f(y_i(m))\mathbf{z}_i(m) + f(y_i(m)^2)\mathbf{w}_i(t)) \tag{4.32}$$

The natural gradient $\widetilde{\nabla} J(\mathbf{w}_i)$ on the Stiefel manifold is related to the regular gradient $\nabla J(\mathbf{w}_i)$ by the relation [31]:

$$\widetilde{\nabla} J(\mathbf{w}_i) = \nabla J(\mathbf{w}_i) - \mathbf{w}_i \nabla J(\mathbf{w}_i)^T \mathbf{w}_i \tag{4.33}$$

$$\widetilde{\nabla} J(\mathbf{w}_i) = \sum_{m=1}^{t} \beta^{t-m}(-f(y_i(m))\mathbf{z}(m) - y_i(m) f(y_i(m))\mathbf{w}_i(m)) \tag{4.34}$$

At the optimum weight vector $\mathbf{w}_{i,opt}$ the gradient is equal to zero, thus:

$$\mathbf{w}_{i,opt}(t) = \frac{\sum_{m=1}^{t} \beta^{t-m} f(y_i(m))\mathbf{z}(m)}{\sum_{m=1}^{t} \beta^{t-m} y_i(m) f(y_i(m))} = \frac{\mathbf{C}_t}{R_t} \tag{4.35}$$

where $R(t) = \sum_{m=1}^{t} \beta^{t-m} y_i(m) f(y_i(m))$ and $\mathbf{C}(t) = \sum_{m=1}^{t} \beta^{t-m} f(y_i(m))\mathbf{z}(m)$. Given that $R(t) = \beta R(t-1) + y_i(t) f(y_i(t))$ and using the matrix inversion lemma [22], the following set of equations can be developed to describe the RLS1-BSE-Stiefel system for $P_i(t) = R^{-1}(t)$:

Table 4.1: **RLS1-BSE algorithm**

$$y_i(t) = \mathbf{w}_i(t-1)^T \mathbf{z}(t) \tag{4.36}$$

$$Q_i(t) = \frac{P_i(t-1)}{\beta + f(y_i(t)) P_i(t-1) y_i(t)} \tag{4.37}$$

$$P_i(t) = \frac{1}{\beta}[P_i(t-1) - Q_i(t) y_i(t) f(y_i(t)) P_i(t-1)] \tag{4.38}$$

$$\mathbf{w}_i(t) = \mathbf{C}_i(t) P_i(t) = \mathbf{w}_i(t-1) + [P_i(t) f(y_i(t))\mathbf{z}(t) - Q_i(t) y_i(t) f(y_i)\mathbf{w}_i(t-1)] \tag{4.39}$$

for each $\mathbf{w}_i, i = 1 \ldots N$. Each row $\mathbf{w}_i^T$ is filtered separately from other rows.
As such, re-orthogonalization is needed to be performed to ensure the independence of each pair of rows $\mathbf{w}_i, \mathbf{w}_j$. This can be done via projection followed by dividing each vector by its norm.
The RLS1-BSE-Stiefel algorithm can be re-written in terms of $g(\mathbf{y})$ instead of $f(\mathbf{y})$. However in such a case, the update equation corresponding to equation (4.39) will be of the from:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - [P_i(t) g(y_i(t))\mathbf{z}_i(t) - Q_i(t) y_i(t) g(y_i(t))\mathbf{w}_i(t-1)] \tag{4.40}$$

instead of:

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) + [P_i(t) f(y_i(t))\mathbf{z}_i(t) - Q_i(t) y_i(t) f(y_i(t))\mathbf{w}_i(t-1)]$$

Table 4.2: **RLS1-BSE algorithm : 2^{nd} implementation**

$$y_i(t) = \mathbf{w}_i(t-1)^T\mathbf{z}(t) \tag{4.41}$$

$$Q_i(t) = \frac{P_i(t-1)}{\beta + g(y_i(t))P_i(t-1)y_i(t)} \tag{4.42}$$

$$P_i(t) = \frac{1}{\beta}[P_i(t-1) - Q_i(t)y_i(t)g(y_i(t))P_i(t-1)] \tag{4.43}$$

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - [P_i(t)g(y_i(t))\mathbf{z}(t) - Q_i(t)y_i(t)g(y_i(t))\mathbf{w}_i(t-1)] \tag{4.44}$$

The above set of equations (i.e. the second implementation) has been tested with a set of 4 subgaussian sources [98], [97]:

$$s_t = [\text{sign}(\cos(2\pi 155t), \sin(2\pi 800t), \sin(2\pi 300t + 6\cos(2\pi 60t)), \sin(2\pi 90t)]^T$$

The mixed signals are sampled at the rate of 10 kHz. In [100] a similar approach was introduced to achieve faster convergence than conventional nonlinear PCA algorithms for the purpose of blind signal separation. This algorithm is included in the comparison under the name "RLS-NPCA" . Thus, the RLS1-BSE algorithm has been compared to:

1. **Conventional Negentropy stochastic gradient descent algorithm** [57]: As described by equation (4.11).

2. **Natural gradient based on Nonlinear PCA** (NAG) [97], [5]:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\mathbf{y}(t) \times g(\mathbf{y}(t))^T - g(\mathbf{y}(t)) \times \mathbf{y}^T(t)]\mathbf{W}(t-1) \tag{4.45}$$

3. **RLS algorithm for Blind Separation (BSS) based on the NPCA** (RLS-NPCA) [100].

As has been pointed out in the previous sections, the natural gradient update (NAG) is the same natural gradient update for Maximum-Likelihood and Negentropy principles on the Stiefel manifold.
The performance index used is [97], [98]:

$$\eta = \sum_{i=1}^{N}(\sum_{j=1}^{N}\frac{|\theta_{ij}|}{\max_k|\theta_{ik}|} - 1) + \sum_{j=1}^{N}(\sum_{i=1}^{N}\frac{|\theta_{ij}|}{\max_k|\theta_{kj}|} - 1) \tag{4.46}$$

where θ_{ij} is one element of the matrix $\mathbf{\Theta} = \mathbf{WVA}$ which is supposed to be a permutation of the identity matrix at $\mathbf{W}_{opt}$. The permutation comes as a result of the ambiguity of the order. The scaling ambiguity is solved by the division by the maximum element in each row in the performance index equation. The above cost function thus penalizes the deviation of $\mathbf{\Theta}$ from the permutation structure. More explaination on the significance of the performance index is presented in section 6.2. Figure (4.1) shows the average for the results of 100 different randomly generated mixing situations. The figure shows that the new algorithm

converges much faster than the natural gradient update. The RLS1-BSE algorithm is also not far behind the improved performance of its BSS counterpart (RLS-NPCA). More simulation results are presented in Chapter 6.
However, one advantage with the RLS1-BSE-Stiefel algorithm is that it can perform both blind signal extraction or sequential blind signal separation. Blind extraction also provides more stable separation when the number of sources is not correctly estimated. If the number of sources is larger than the number of sensors, Blind Signal Extraction (BSE) algorithms usually extract the most dominant sources sequentially and thus provides a better separation in such a case.

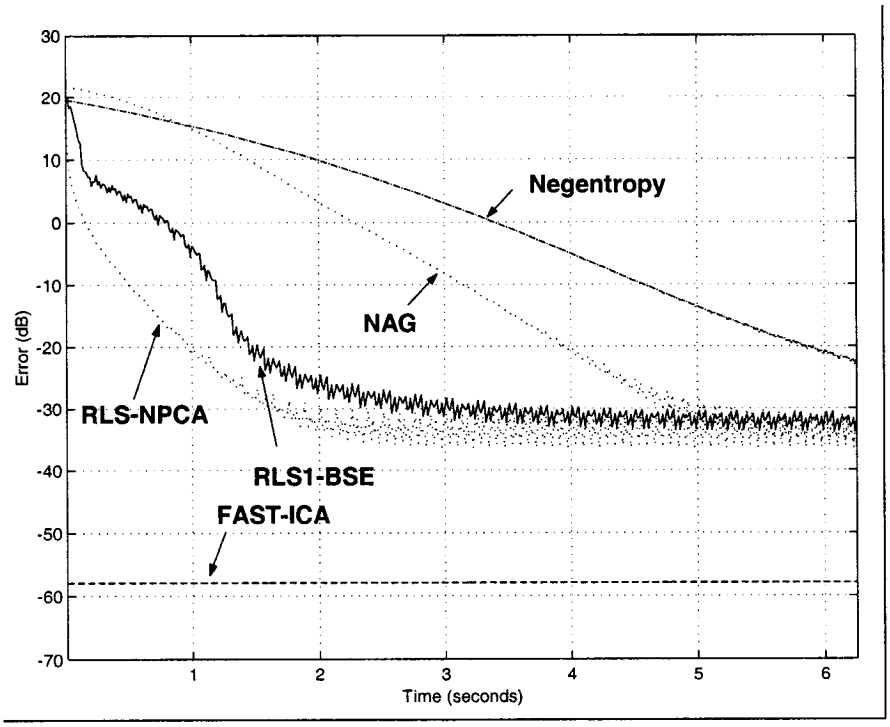

Figure 4.1: **Average performance of the RLS1-BSE-Stiefel compared to the Natural Gradient on Stiefel Manifold, RLS-NPCA, Negentropy and FAST-ICA**

Another advantage of the RLS1-BSE-Stiefel algorithm over The RLS-NPCA developed in [100] is its low computational load. Reviewing the summary of the RLS1-BSE-Stiefel algorithm, either in Table 4.1 or Table 4.2, it can be seen that the main difference from its natural gradient counterpart (4.16) is the division steps in equations (4.37) and (4.38). However, these divisions are scalar divisions as opposed to matrix inversions as in the BSS algorithm in [100]. Scalar division are in the same complexity order as additions and multiplications [85], and usually require 10 times the computations of a regular add or multiply operations [60]. These two scalar division steps will add roughly 20 additional operations to each extraction step. If all the N sources are extracted sequentially, the total complexity will still be in the range of N^2, which is the same range as the BSS natural gradient algorithm. On the other hand, the algorithm in [100], which involves matrix inversions will require at least the order of N^3 operations to perform the matrix inversions [85].

While the RLS1-BSE-Stiefel algorithm exhibits an improved performance, one problem with both the RLS1-BSE-Stiefel and the RLS-NPCA in [100] is that they do not produce good results when applied to highly correlated nonstationary speech like mixtures. The next section presents another RLS algorithm for this type of data. This algorithm can be considered as a BSE algorithm for either of the three cost function ML, Negentropy and nonlinear PCA.

4.7 The RLS2-BSE-Stiefel Algorithm Based on Negentropy

This section presents a new Negentropy based algorithm, that is capable of handling speech signals. The key to the derivation of the algorithm is an update that incorporates a step size instead of a direct RLS-evaluation of the rows of the de-mixing matrix $\mathbf{W}$. The algorithm is presented in publication [33]. The following are two different routes to derive the algorithm. The first is directly from the stochastic gradient and the second is done via exploring Stiefel-manifold optimization.

4.7.1 The RLS2-BSE-Stiefel Algorithm (1)

The derivation begins with modifying the cost function in equation (4.9) to include a penalty term due to the normalization of $\|\mathbf{w}\|^2 = 1$ [56] to be:

$$J(\mathbf{w}^T\mathbf{z}) = [E\{G(\mathbf{w}^T\mathbf{z})\} - E\{G(v)\}]^2 - \beta_1(\|\mathbf{w}\|^2) \tag{4.47}$$

The penalty term can actually be evaluated by finding the maxima of the above cost function through solving the equation [56] :

$$2\gamma E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\} - 2\beta_1\mathbf{w} = 0 \tag{4.48}$$

where $\gamma = [E\{G(\mathbf{w}^T\mathbf{z})\} - E\{G(v)\}]$ and can be set to -1 for supergaussian data. Setting $\beta = \beta_1/\gamma$, then the above equation is satisfied by $\beta = E\{\mathbf{w}_{opt}^T\mathbf{z}g(\mathbf{w}_{opt}^T\mathbf{z})\}$.
The optimal value $\mathbf{w}_{opt}$ can be approximated by the current value $\mathbf{w}$, thus: $\beta = E\{\mathbf{w}^T\mathbf{z}g(\mathbf{w}^T\mathbf{z})\}$.
From the above set of equations the following rate of change of $\mathbf{w}$ can be calculated :

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{w}} &= 2\gamma E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\} - 2\beta_1\mathbf{w} \\ &= 2\gamma E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\} - 2\gamma\beta\mathbf{w}\end{aligned} \tag{4.49}$$

The update equations can then be set as:

$$\begin{aligned}\mathbf{w}_+(t) &= \mathbf{w}(t-1) - \mu[E\{\mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t))\} - \beta\mathbf{w}(t-1)] \\ \mathbf{w}(t) &= \mathbf{w}_+/\|\mathbf{w}_+\|\end{aligned} \tag{4.50}$$

As long as $\mathbf{w}(t)$ is normalized, one can actually manipulate the expression in (4.50) as follows:

$$\begin{aligned}\mathbf{w}_+(t) &= (1+\beta)\mathbf{w}(t-1) - \mu E\{\mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t))\} \\ \mathbf{w}_+(t)/(1+\beta) &= \mathbf{w}(t-1) - \mu E\{\mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t))\}/(1+\beta) \\ \mathbf{w}_+(t) &= \mathbf{w}(t-1) - \mu E\{\mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t))\}/(1+\beta) \\ &\approx \mathbf{w}(t-1) - \mu\ E\{\mathbf{z}(t)\ g(\mathbf{w}(t-1)^T\mathbf{z}(t))\}/\beta\end{aligned}$$

where the division $\mathbf{w}_+(t)/(1+\beta)$ is made equal to $\mathbf{w}_+(t)$ because of the normalization step $\mathbf{w}(t) = \mathbf{w}_+/\|\mathbf{w}_+\|$ that should follow.
From the above derivation the following new update rule is used for our proposed algorithm:

$$\mathbf{w}_+(t) = \mathbf{w}(t-1) - E\{\mathbf{z}g(\mathbf{w}(t-1)^T\mathbf{z}(t))\}/\beta \tag{4.51}$$
$$\mathbf{w}(t) = \mathbf{w}_+/\|\mathbf{w}_+\|$$

A stochastic gradient version of the algorithm is:

$$\mathbf{w}_+(t) = \mathbf{w}(t-1) - \mu_n \times \mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t))/\beta \tag{4.52}$$
$$\mathbf{w}(t) = \mathbf{w}_+/\|\mathbf{w}_+\|$$

while β is kept as $E\{\mathbf{w}(t-1)^T\mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t))\}$. This expected value of β is replaced by an exponentially weighted window estimate given as:

$$\beta(t) = \lambda \times \beta(t-1) + \mathbf{w}(t-1)^T\mathbf{z}(t)g(\mathbf{w}(t-1)^T\mathbf{z}(t)) \tag{4.53}$$

A look at the structure of the update equation (4.60) given an indication why the algorithm was actually considered as RLS type algorithm. The term:

$$\beta = E\{\mathbf{w}^T\mathbf{z}g(\mathbf{w}^T\mathbf{z})\} = E\{\mathbf{y}(t)g(\mathbf{y}(t))\} \tag{4.54}$$

can be considered as a correlation term between the signal $\mathbf{y}$ and its nonlinear counterpart $g(\mathbf{y})$. At the optimal separation point, no non-linear cross-correlation exists among the output signals $\mathbf{y}$.

4.7.2 The RLS2-BSE-Stiefel Algorithm (2)

The above algorithm can be derived using a totally different approach and yet yield the same results. The fact that the data are pre-whitened prior to the application of Negentropy-algorithms moves the constrained problem of estimating the demixing matrix $\mathbf{W}$ to the Stiefel manifold.
Starting with the Negentropy cost function:

$$J(\mathbf{w}) = [E\{G(\mathbf{w}^T\mathbf{z})\} - E\{G(v)\}]^2 - \beta_1(\|\mathbf{w}\|^2)$$

and recalling that the equation of the natural gradient in the Stiefel manifold is:

$$\tilde{\nabla}J(\mathbf{w}) = \nabla J(\mathbf{w}) - \mathbf{w}\nabla J(\mathbf{w})^T\mathbf{w} \tag{4.55}$$

The Stiefel-gradient of the Negentropy algorithm was derived in section 4.3, and is given as:

$$\tilde{\nabla}J(\mathbf{w}) \propto E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\} - \mathbf{w}E\{(\mathbf{w}^T\mathbf{z})g(\mathbf{w}^T\mathbf{z})\} \tag{4.56}$$

At the optimal separating vector $\mathbf{w}_{opt}$:

$$\tilde{\nabla}J(\mathbf{w}) = 0 \tag{4.57}$$

Applying equation (4.57) to (4.56) gives:

$$\mathbf{w}_{opt} = E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\}/(E\{(\mathbf{w}^T\mathbf{z})g(\mathbf{w}^T\mathbf{z})\}) \tag{4.58}$$

which is the same update obtained by the Lagrange multiplier method in the previous derivation.

To obtain an on-line version of the above equation we can simply replace:

- $E\{\mathbf{z}g(\mathbf{w}^T\mathbf{z})\}$ by its instantaneous value $\mathbf{z}g(\mathbf{w}(t-1)^T\mathbf{z}(t))$.
- $E\{(\mathbf{w}^T\mathbf{z})g(\mathbf{w}^T\mathbf{z})\}$ by an exponential window estimate with a forgetting factor λ.

This will yield the same equations (4.51) to (4.53), which describe the new algorithm developed in the previous section without using the Stiefel-manifold principle. Having derived the algorithm by two different routes, it might be noticed that both the unconstrained Lagrange multiplier method, i.e. the first derivation, and the second Stiefel-based derivation lead to the same optimal solution.
Another note is that because the above algorithm uses a step size and has the same structure like LMS-based algorithm, it is possible to develop a natural gradient version of the RLS2-BSE algorithm by re-projecting the update on the Stiefel manifold. This modification will be presented in the next section following the simulation results.

4.7.3 Simulation Results

The proposed algorithm is compared with the following algorithms:

1. **Conventional Negentropy stochastic gradient descent algorithm**.
2. **Fast-ICA**.
3. **Natural gradient based on the Negentropy principle for Stiefel manifold**

The simulated test was performed on audio data files obtained from [46] which are 4 sources: two males, one female and soft music. These files are sampled at 8 kHz and are of duration 6.5 sec The step size of the algorithm μ_n is set to 0.025. The forgetting factor λ can be either set to a default value of 0.99 or made variable from 0.98 at sample $n = 0$ to 0.99 at sample $n = 10000$ which gives a smoother convergence rate. The performance of the algorithms is judged based on the same cross-talk error used in the preceding section. The following figure shows the result of the algorithm.
As can be seen in the Figure (4.2), the RLS-BSE-Stiefel algorithm works well and converges faster than the natural gradient algorithm. It can, on average, reach the same steady state error achieved by the FAST-ICA off-line algorithm.

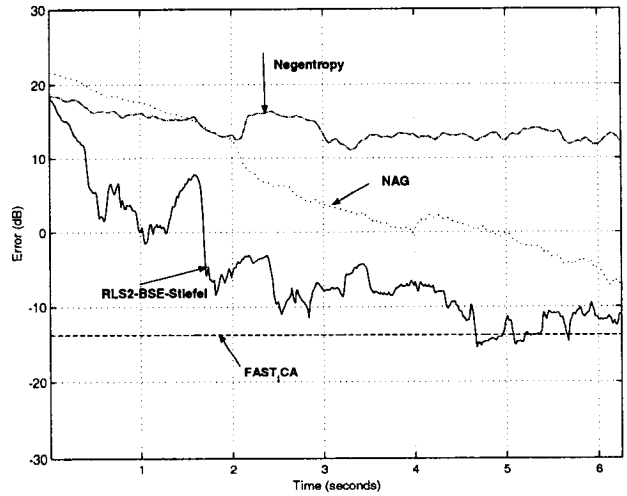

Figure 4.2: **Average Performance of RLS2-BSE-Stiefel compared to the Natural Gradient on Stiefel manifold, Negentropy and FAST-ICA**

4.8 Modified RLS2-BSE Algorithm

The previous algorithm (RLS2-BSE) can be further enhanced by projecting the update $\Delta = \mu_n \times \mathbf{z}g(\mathbf{w}^T\mathbf{z})/\beta$ on the Stiefel manifold using the Stiefel-gradient equation (4.55) to be:

$$\widetilde{\Delta} = \Delta - \mathbf{w} \times \Delta^T \times \mathbf{w} \tag{4.59}$$

This modification improves the performance as can be seen in the following figure.

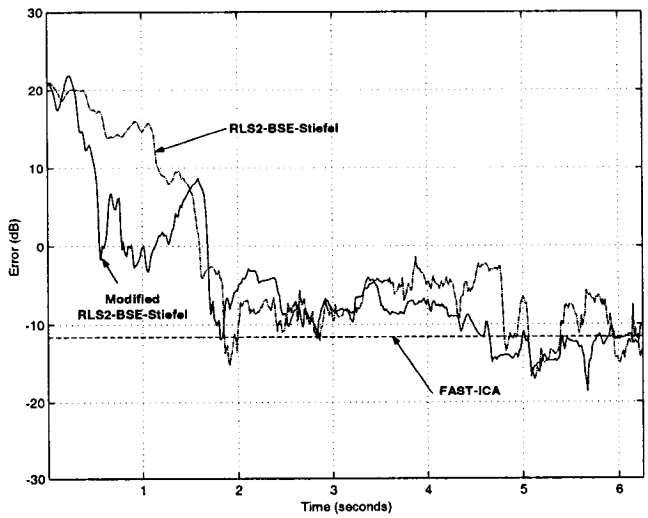

Figure 4.3: **RLS2-BSE-Stiefel vs. modified RLS2-BSE-Stiefel**

More results of this modified algorithm are presented in Chapter 6. Another advantage of

this projection is that it helps further stabilizing the algorithm so that the normalization step:

$$\mathbf{w}(t) = \mathbf{w}_+ / \|\mathbf{w}_+\|$$

can be done less frequently. The projection is performed every 1000 samples. This helps reducing the complexity of the algorithm. The following table summarizes the "modified RLS2-BSE" algorithm.

Table 4.3: **Modified RLS2-BSE algorithm**

$$y_i(t) = \mathbf{w}_i(t-1)^T \mathbf{z}(t) \tag{4.60}$$

$$\beta_i(t) = \lambda \beta_i(t-1) + g(y_i(t)) y_i(t) \tag{4.61}$$

$$\Delta = \mu g(y_i(t)) \mathbf{z}(t) / \beta \tag{4.62}$$

$$\mathbf{w}_+ = \mathbf{w}_i(t-1) - (\Delta - \mathbf{w}(t-1) \times \Delta^T \times \mathbf{w}(t-1)) \tag{4.63}$$

- if $t/1000 = n, n = 1, 2, \cdots$:

$$\mathbf{w}_i(t) = \mathbf{w}_+ / \|\mathbf{w}_+\| \tag{4.64}$$

- else:

$$\mathbf{w}_i(t) = \mathbf{w}_+ \tag{4.65}$$

The complexity of this algorithm can be calculated following the rules used to calculate the complexity of the RLS1-BSE algorithm. The main differences in complexity, compared to conventional Negentropy or natural gradient BSE on-line algorithms, come from 3 sources:

1. The division by the scalar β, which can be thought of as requiring roughly 10 operations for each $\mathbf{w}_i, i = 1 \cdots N$ and the exponential window moving average used to calculate β, which also require an order of N operations for each $\mathbf{w}_i, i = 1 \cdots N$.

2. The projection step implemented in equation (4.59), which involves the multiplication of vectors of size N. This projection roughly requires an order of N^2 for each $\mathbf{w}_i, i = 1 \cdots N$.

3. The normalization step, which contributes another N-order operations for each $\mathbf{w}_i, i = 1 \cdots N$. However, implementing the normalization less frequently in the case of the "modified RLS2-BSE" algorithm helps to reduce the complexity of this step.

The algorithm can thus be estimated to be roughly in the order of N^2 operations for each $\mathbf{w}_i, i = 1 \cdots N$ as compared to N-order operations for each $\mathbf{w}_i, i = 1 \cdots N$ in the case of on-line Negentropy. If all the N sources are extracted sequentially then the overall complexity of the algorithm becomes in the order of N^3 as opposed to the less complex natural gradient algorithm or on-line Negentropy BSE-algorithm (when all the N-sources are extracted), both estimated in the order of N^2 operations.

4.9 The RLS-BSS-Stiefel Algorithm Based on Maximum-Likelihood

The RLS-BSS-Stiefel algorithm to be proposed is developed for the purpose of blind signal separation, i.e. estimating all the signals at once, as opposed to the new algorithms previously introduced, which were structured to perform blind extraction. Although this might seem as a limitation, it is widely recognized that if sequential extraction of the signals is not required, simultaneous blind separation is favored to blind extraction, which permits accumulation of errors from one signal extraction step to the next signal extraction. Thus, in general BSS algorithms converge faster with smaller errors than BSE- algorithms [56]. One way to implement an RLS-type algorithm for the above situation is served from the adaptive filter theory. In [16], a non-linear RLS-like implementation was presented. It depends on using the chain rule to define an "equivalent error" signal and an "equivalent input" signal in the derivative of a certain non-linear cost function $J(\mathbf{W})$, as an analogy to the classical stochastic gradient update equation for linear adaptive filters:

$$J(\mathbf{W}) = f_1(\phi), \phi = f_2(\psi), \psi = f_3(\mathbf{y}), \mathbf{y} = f_4(\mathbf{W})$$

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{\partial J}{\partial \phi} \times \frac{\partial \phi}{\partial \psi} \times \frac{\partial \psi}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{W}} \tag{4.66}$$

The update rule for $\mathbf{W}$ can be written as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) - \mu \frac{\partial J}{\partial \mathbf{W}}$$

$$\mathbf{W}(t) = \mathbf{W}(t-1) - \mu \frac{\partial J}{\partial \mathbf{W}} = \mathbf{W}(t-1) - \mu \frac{\partial J}{\partial \phi} \times \frac{\partial \phi}{\partial \psi} \times \frac{\partial \psi}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{W}}$$

$$\mathbf{W}(t) = \mathbf{W}(t-1) - \mu \text{ equivalent error} \times \text{equivalent input}$$

The terms equivalent error and equivalent input are analogous to the classical adaptive filtering where $\frac{\partial J}{\partial \mathbf{y}}$ is the negative of error signal and $\frac{\partial \mathbf{y}}{\partial \mathbf{W}}$ is the input signal $\mathbf{x}$. After forming the gradient in this chain formula, one can decide on taking as many terms as suitable to form an "equivalent input signal" and then multiply this input by the inverse of a correlation matrix as in classical RLS filtering. The above equation can be seen as:

$$\frac{\partial J}{\partial \mathbf{W}} = \underbrace{\frac{\partial J}{\partial \phi}}_{equivalent-error} \times \underbrace{\frac{\partial \phi}{\partial \psi} \times \frac{\partial \psi}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{W}}}_{equivalent-input} \tag{4.67}$$

Since is no clear distinction between the "input" and the "error" signal it was chosen to take the gradient of the cost function as a whole as the equivalent input and to apply the RLS algorithm to speed the convergence. Defining $\mathbf{R}$ as:

$$\mathbf{R} = E\{\tilde{\nabla} J(\mathbf{W})_{vec}\} E\{\tilde{\nabla} J(\mathbf{W})_{vec}^T\} \tag{4.68}$$

where $\tilde{\nabla} J(\mathbf{W})_{vec}$ is a re-arrangement of the $N \times N$ matrix $\tilde{\nabla} J(\mathbf{W})$ as a vector of size $N^2 \times 1$ via column stacking. The resulting $\mathbf{R}$ is thus a matrix of size $N^2 \times N^2$.

Another interpretation for this choice can be brought up from a different perspective. Near the optimal solution the Hessian matrix Hess can be approximated as [13], [27]:

$$\text{Hess} \approx 2 \times \tilde{\nabla} J(\tilde{\nabla} J)^T \tag{4.69}$$

However, the approach of the non-linear RLS will prove relevant when considering later the non-prewhitened natural gradient algorithm as the choice of the "non-linear" input will be optimized to a slightly different expression than the whole gradient. It is only in the case of pre-whitened data that the algorithm, proposed in this section, can be viewed as either a non-linear RLS algorithm or a Gauss-Newton algorithm. It can be called non-linear RLS if we follow the steps of [16] and consider the gradient as the nonlinear input signal to be multiplied by the inverse correlation matrix. It can also be called relative-Newton if we consider $\mathbf{R}$ to be an approximation to the Hessian matrix as given by (4.69). Back to the derivation of the algorithm one can replace the expected value by the following exponentially weighted, on line update:

$$\mathbf{R}(t) = \lambda \mathbf{R}(t-1) + \tilde{\nabla} J(\mathbf{W})_{vec} \tilde{\nabla} J(\mathbf{W})_{vec}^T$$

where λ is the forgetting factor close to 1. From the above estimate of $\mathbf{R}(t)$, the following update algorithm can be developed:

1. Start at time $t = 0$ with an initial matrix $\mathbf{W}_t = \mathbf{I}_{N \times N}$ and $\mathbf{R} = \mathbf{I}_{N^2 \times N^2}$.
2. Calculate the estimate output $\mathbf{y}_t$:$\mathbf{y}_t = \mathbf{W}_t \mathbf{z}_t$
3. Calculate the gradient $\tilde{\nabla} L(\mathbf{W})_t$:$\tilde{\nabla} L(\mathbf{W})_t = (\mathbf{g}(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{g}(\mathbf{y})^T)\mathbf{W}$
4. Vectorize the gradient $\tilde{\nabla} L(\mathbf{W}_t)$:$\tilde{\nabla} L(\mathbf{W}_t)_{vec} = Vec(\tilde{\nabla} L(\mathbf{W}_t))$
5. Compute the estimate:$\mathbf{R}_t = \lambda \mathbf{R}_{t-1} + \tilde{\nabla} L(\mathbf{W})_{vec} \tilde{\nabla} L(\mathbf{W})_{vec}^T$
6. Compute the current update, $\triangle \mathbf{W}_{vec}$: $\triangle \mathbf{W}_{vec} = (\mathbf{R}_t)^{-1} \tilde{\nabla} L(\mathbf{W}_t)_{vec}$
7. Transform the update $\triangle \mathbf{W}_{vec}$ to the matrix form of size $N \times N$ by rearranging the columns: $\triangle \mathbf{W} = mat(\triangle \mathbf{W}_{vec})$
8. Update the current estimate of the matrix $\mathbf{W}$ as :$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_{RLS} \triangle \mathbf{W}$

where $mat(.)$ transforms a vector of size $N^2 \times 1$ into a matrix of size $N \times N$ via column stacking.

The above algorithm proved to have a fast initial convergence. However, it diverges with time. The reason for that is that the new step update $\triangle \mathbf{W}$ diverges from the constraint of the Stiefel manifold. To troubleshot this problem, a projection of the new update $\triangle \mathbf{W}$ on the Stiefel manifold should be added between step (7) and (8). The resulting projected step is:

$$\triangle \tilde{\mathbf{W}} = \mathbf{W}\mathbf{W}^T \triangle \mathbf{W} - \mathbf{W} \triangle \mathbf{W}^T \mathbf{W} \tag{4.70}$$

This added step does not affect the initial fast convergence speed of the algorithm while it provides the required robustness against divergence. It is also performed in the $N \times N$ matrix domain and thus it is not computationally demanding. The inverse of the matrix

$\mathbf{R}$ in step (6) can be computed using the matrix inversion lemma, referring to $\mathbf{R}$ as $\mathbf{P}$ and $\tilde{\nabla}L(\mathbf{W}_t)_{vec}$ as $\mathbf{m}$:

$$\mathbf{P}(t)^{-1} = \frac{1}{\lambda}[\mathbf{P}(t-1)^{-1} - \frac{\mathbf{P}(t-1)^{-1}\mathbf{m}_t\mathbf{m}_t^T\mathbf{P}(t-1)^{-1}}{\lambda + \mathbf{m}_t^T\mathbf{P}(t-1)^{-1}\mathbf{m}_t}] \tag{4.71}$$

Therefore, the complete **RLS-BSS-Stiefel** algorithm can be stated as in Table 4.4.:

Table 4.4: **RLS-BSS-Stiefel algorithm**

1. Start at time $t = 0$ with an initial matrix $\mathbf{W}_n$ equal to the identity matrix $\mathbf{I}_{N\times N}$ and $\mathbf{R} = \mathbf{I}_{N^2\times N^2}$.
2. Calculate the estimate output $\mathbf{y}_t$
$$\mathbf{y}_t = \mathbf{W}_t\mathbf{z}_t$$
3. Calculate the gradient $\tilde{\nabla}L(\mathbf{W})_t$:
$$\tilde{\nabla}L(\mathbf{W})_t = (\mathbf{g}(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{g}(\mathbf{y})^T)\mathbf{W}$$
4. Vectorize the gradient $\tilde{\nabla}L(\mathbf{W}_t)$:
$$\tilde{\nabla}L(\mathbf{W}_t)_{vec} = Vec(\tilde{\nabla}L(\mathbf{W}_t))$$
5. Compute the estimate:
$$\mathbf{R}_t = \lambda\mathbf{R}_{t-1} + \tilde{\nabla}L(\mathbf{W})_{vec}\tilde{\nabla}L(\mathbf{W})_{vec}^T$$
6. Compute the inverse of $\mathbf{R}$ as in equation (4.71).
7. Compute the current update:
$$\triangle\mathbf{W}_{vec} = (\mathbf{R}_t)^{-1}\tilde{\nabla}L(\mathbf{W}_t)_{vec}$$
8. Transform the update $\triangle\mathbf{W}_{vec}$ to the matrix form of size $N \times N$ by rearranging the columns $\triangle\mathbf{W} = mat(\triangle\mathbf{W}_{vec})$
9. Calculate the projection of $\triangle\mathbf{W}$ on the Stiefel manifold as:
$$\tilde{\triangle}\mathbf{W} = \mathbf{W}\mathbf{W}^T\triangle\mathbf{W} - \mathbf{W}\triangle\mathbf{W}^T\mathbf{W}$$
10. Update the current estimate of the matrix $\mathbf{W}$ as :
$$\mathbf{W}_{t+1} = \mathbf{W}_t + \mu_{RLS}\tilde{\triangle}\mathbf{W}$$

The complexity of the "RLS-BSS-Stiefel" algorithm is mainly governed by the inversion step implemented in step (6). This inversion involves a correlation matrix $\mathbf{R}$ of size $N^2 \times N^2$. The inversion of a matrix of size $N \times N$ can be optimized to N^3 operations [85]. The inversion of $\mathbf{R}$ will thus be roughly in the order of N^6. This complexity is much larger than the complexity required for the blind extraction algorithms. However, the algorithm works with both faster convergence and smaller steady state error than BSE algorithms. In the following subsection, an example of results of the RLS-BSS-Stiefel algorithm are presented. More results are provided in Chapter 6.

4.9.1 Simulation Results

Using the cross talking error measure provided by equation (4.72)

$$\eta = \sum_{i=1}^{N}\left(\sum_{j=1}^{N}\frac{|\theta_{ij}|}{\max_k |\theta_{ik}|} - 1\right) + \sum_{j=1}^{N}\left(\sum_{i=1}^{N}\frac{|\theta_{ij}|}{\max_k |\theta_{kj}|} - 1\right) \tag{4.72}$$

The new RLS-algorithm has been compared to the natural gradient on the Stiefel-manifold [7], given by the equation (5.11).
The step size of the natural gradient algorithm is adequately set to 0.0005. The data sets used are presented in more details in Chapter 6. They are namely:

(a) **4 sources** two male, one female and soft music.

(b) **10 sources** two males, two females, two orchestra music, one opera , one soft music, a street noise and a siren.

Figures (4.4) and (4.5) present the results of the average of 100 randomly generated mixing situations.

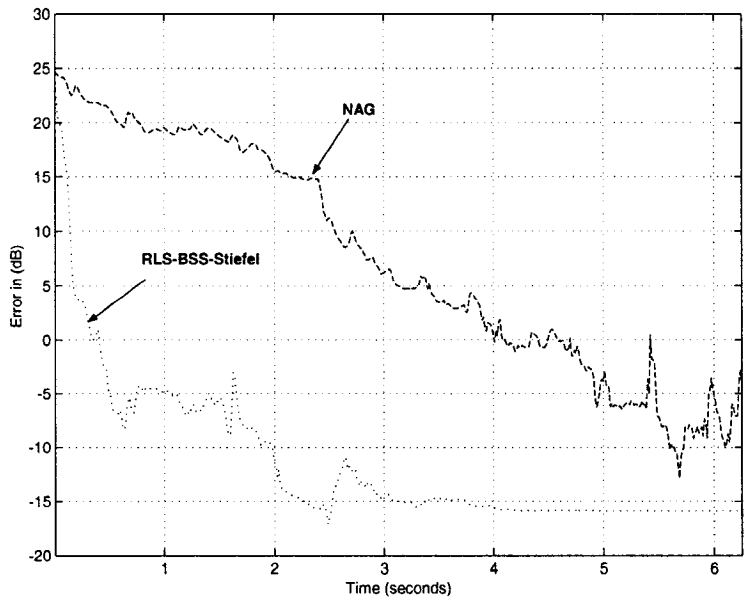

Figure 4.4: **Average results for RLS-BSS-Stiefel algorithm:** $N = 4$

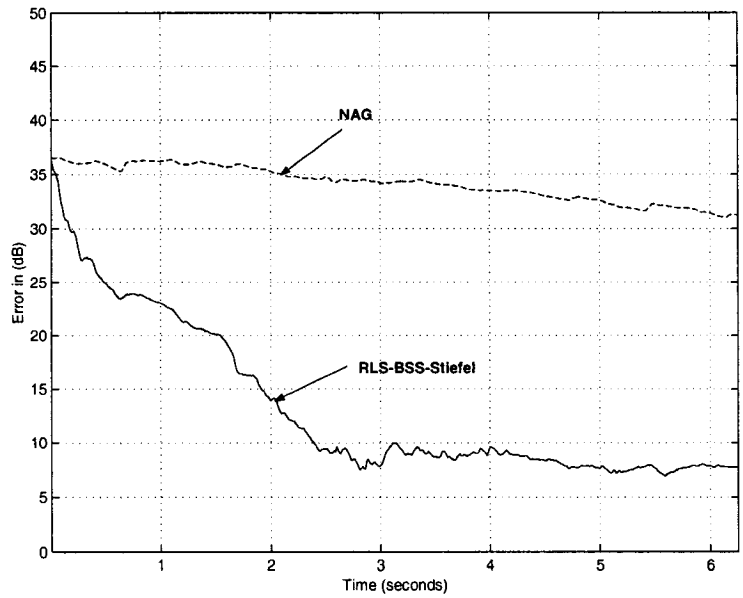

Figure 4.5: **Average results for RLS-BSS-Stiefel algorithm:** $N = 10$

From the above figures, one can see clearly the excellent convergence speed of the proposed algorithm when compared to the natural gradient algorithm. Clearly, the algorithm can robustly handle difficult mixing situations where several types of signals exist, i.e. speech and music and even other types of coloured noise like street noise.

4.10 Conclusion

This chapter presented a collection of new algorithms. The algorithms were developed for the important case where pre-whitening is applied to the data or the case where the data is assumed to be white mixtures. Before the algorithms were introduced, the equivalence of three of the main cost functions, namely Maximum-likelihood, Negentropy, nonlinear PCA and a newly developed Bussgang cost function on the Stiefel manifold, is discussed. Although these functions have different first gradients, these gradients reduce to the same natural gradient on the Stiefel manifold. This provides a sense of generality for any algorithm that builds on the natural gradient-concept of orthogonal matrix group. Afterwards, the discussion in the chapter focused on the new proposed algorithms. The first algorithm (RLS1-BSE-Stiefel) was built especially for the blind signal extraction purpose. It was developed by constructing the least squares version of the Negentropy blind signal extraction cost function and evaluating the value of the optimal de-mixing matrix in an RLS-manner. The algorithm proved very efficient and superior for most of the test data. However, the RLS1-BSE-Stiefel algorithm did not match this improved performance when applied to highly correlated nonstationary signals like speech. To combat this problem, another algorithm "RLS2-BSE-Stiefel" was presented to work well with highly correlated mixtures as well as

with other type of mixtures. The algorithm is again Stiefel based. It is derived in two different ways. The first derivation uses the Lagrange multiplier method, while the second one is obtained from the first order derivative on the Stiefel manifold.
The last algorithm presented was the "RLS-BSS-Stiefel" algorithm. The developed algorithm can be viewed as non-linear RLS-based algorithm. The performance of the algorithm is verified to be of improved convergence speed via simulations. In the following table, a summary of the algorithms developed in this chapter is presented highlighting their basic properties and their estimated approximate complexities.

Table 4.5: **Summary of algorithms developed for BSS on the Stiefel manifold**

Algorithm	Type	Application Data	Complexity
RLS1-BSE-Stiefel	BSE	slow varying data	$O(N)$ for each source
modified RLS2-BSE-Stiefel	BSE	audio data	$O(N^2)$ for each source
RLS-BSS-Stiefel	BSS	audio data	$O(N^6)$

Chapter 5

Improvements to Instantaneous Blind Signal Separation Algorithms: Non-Prewhitened Case

5.1 Introduction

Chapter 4 presented a collection of algorithms to provide improved convergence to blind signal separation cost functions on the Stiefel-manifold (i.e. the constrained orthogonal de-mixing matrices space of optimization when mixtures are pre-whitened). As has been presented in Chapter 2, the pre-whitening process must be performed for some of the algorithms, including the information-theoretic based Negentropy algorithm. Pre-whitening is also still recommended for the maximum-likelihood cost function. It helps to get rid of the noise and constrains the algorithm to the Stiefel-manifold where the natural gradient form spans a smaller set of matrices and thus produces faster convergence, even compared to the natural gradient on the general-linear GL-group, which is the equivalent general Riemannian manifold of invertible matrices. However, the orthogonality of the demixing matrices is not a necessary condition of operation for the maximum-likelihood cost function.

It would thus be useful to extend the idea of developing algorithms with improved performance for the general Riemannian case, i.e. GL-group, to account for systems which may not be able to afford performing pre-whitening prior to separation, while requiring fast convergence rates.

This chapter begins with a more broad look at the structure of the natural gradient update equation. The nonholonomic natural gradient is introduced as a variant form of the natural gradient. The introduction of the nonholonomic natural gradient update stimulates a more general view of the maximum-likelihood cost function and its derivative in the case where the output separated sources are not assumed to be of unit variance. Not assuming the sources to be of unit variance provides a new degree of freedom, that is of paramount use when combined with another promising algorithm of BSS, namely the "Iterative Inversion" approach.

This promising approach for maximum-likelihood was presented in [1] based on iterative inversion (II) of the mixing matrix $\mathbf{A}$. Interestingly, the closed form of the algorithms emanating from this approach has the same form of the natural gradient update and were linked

to the work done by Cardoso [20] and Oja [86]. A new proposed family of algorithms is introduced, combining the magnitude-relieved ML-cost function, i.e. ML-update that does not require the sources to be of unit variance, and the iterative inversion approach. This proposed family of natural-gradient updates provides a collection of algorithms with different convergence rates and steady state errors. The particular value of this family of algorithms is their flexibility and the ability to deduce various forms of already existing algorithms by substituting different choices for the parameters developed in each algorithm. Among these algorithms is a promising RLS-based algorithm that proves to be very efficient in the sense of both fast convergence and lower steady state error.
The first half of the chapter concludes with the extension of the BSS-RLS-Stiefel algorithm, developed in Chapter 4, to the general mixing scenario where mixtures are not pre-whitened. The goal is to produce algorithms that are capable of working under more difficult situations e.g. highly correlated nonstationary audio mixtures. The idea of nonlinear RLS implemented through this algorithm for the GL-manifold produces improved performance and showed suitability with the maximum-likelihood cost function.
This chapter, then, concentrates on a different route for developing new algorithms, namely, the Riemannian structure of the maximum-likelihood BSS problem. The geometric properties of the separation problem are investigated to produce new algorithms with well-justified better performance, combining second-order techniques well known fast convergence and geometric (Riemannian) optimization with assurance of better stability and better directed gradients. The Hessian matrix, is then calculated and a new on-line algorithm is developed. Another form of reduced order second-derivative is calculated and the update is developed in an RLS-like update with improved convergence behavior. This reduced algorithm is then incorporated with the concept of median-gradient, developed in [41], to develop an even faster convergence and smaller error.
The last section in the chapter proposes an application of some second-order algorithms to the Stiefel-manifold, i.e. the pre-whitened mixtures case, to develop higher order faster converging algorithms at affordable computational cost.

5.2 Nonholonomic Natural Gradient Update

In this section, a review of an important variant of the natural gradient, namely, the nonholonomic natural gradient is presented. The importance of the nonholonomic version of the natural gradient is that it lifts the unity constraint $E\{y_i^2\} = 1$ from the maximum-likelihood cost function. This removal of the unnecessary and sometimes untrue constraint [1] gives more insight in how the maximum-likelihood cost function update is formed and is valuable in deriving new more general updates of natural gradient, which, when combined with the concept of iterative inversion, provide a new spectrum or family of natural-gradient-based updates. The nonholonomic update is based on the special structure of the demixing matrix $\mathbf{W}$. Defining a mathematical group $C_{\mathbf{W}}$ of all diagonally scaled versions of $\mathbf{W}$:

$$C_{\mathbf{W}} = \Lambda\mathbf{W} \tag{5.1}$$

[1] This unnecessary limitation of the amplitude of the outputs causes problems for sources that have nonstationary nature and whose amplitude could go suddenly to zero. Constraining the output to have unit variance would cause the de-mixing matrix to diverge [9]

where Λ is a general diagonal matrix:

$$\Lambda = \begin{pmatrix} \gamma_1 & & \\ & \ddots & \\ & & \gamma_N \end{pmatrix} \tag{5.2}$$

One can easily see that the effect of the diagonal matrix is adding a scale factor to each row of the matrix $\mathbf{W}$, transforming it to:

$$\begin{pmatrix} \gamma_1 \begin{pmatrix} W_{11} & \cdots & W_{1N} \end{pmatrix} \\ \vdots \\ \gamma_N \begin{pmatrix} W_{N1} & \cdots & W_{NN} \end{pmatrix} \end{pmatrix} \tag{5.3}$$

which translates into a scaling factor in the output y_i formed by the i^{th} row of the de-mixing matrix $\mathbf{W}$. The group $C_{\mathbf{W}}$ for a certain matrix $\mathbf{W}$ is given the name the "***equivalence class***". The equivalence class can be incorporated in the natural gradient update equation to produce the form [9]:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\Lambda - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{5.4}$$

The nonholonomic natural gradient assigns a specific value for the diagonal matrix Λ. Λ is chosen to be equivalent to the diagonal elements of the term $g(\mathbf{y})\mathbf{y}^T$ such that the matrix $[\Lambda - g(\mathbf{y})\mathbf{y}^T]$ will have no diagonal elements. This provides automatic scaling for $\mathbf{y}$. Reference [9] provides more details about the reason for such a choice and how it is suitable in the case where the number of sensors M is larger than the number of sources N. However, what is important here is the ability or more exactly the flexibility of the matrix Λ and the introduction of the idea that it might be useful to lift the unitary constraint condition on the output, which is classically imposed by researchers working in the BSS domain.
In the following section, more insight is provided on how this more general update, i.e. equation (5.4), is formed using the stability conditions of the ML-cost function derivative and what is its link to the autocorrelation of the output signal $\mathbf{y}$. This understanding will help in choosing alternative suitable values for the diagonal matrix Λ and eventually, when combined with the concept of iterative inversion, will provide a very interesting form of update that is very similar to the Wiener filter update.

5.3 A Proposed More General Update for Maximum-Likelihood Cost-Function

Let $s_i(n)$, $i = 1, 2, ..., N$ be scalar inputs to the blind signal separation model at a time n. For simplicity, it is assumed that the mixing is linear and that the mixing matrix is square, i.e. the number of inputs N is equal to the number of mixtures x_i, $i = 1, 2, ..., N$. Therefore, the mixing matrix $\mathbf{A}$ is a square matrix of size $N \times N$. The mixing model can be expressed as:

$$\mathbf{x}(n) = \mathbf{A} \times \mathbf{s}(n) \tag{5.5}$$

The purpose of the blind signal separation algorithms is to estimate a matrix $\mathbf{W}$ such that $\mathbf{W} \times \mathbf{A} = \mathbf{I}_{N\times N}$, where $\mathbf{I}$ is the identity matrix. If this condition is met then the output of the separation process referred to as $y_i(n)$, $i = 1, 2, ..., N$ is identical to the independent sources $s_i(n)$, $i = 1, 2, ..., N$ except for order and scale.
Maximum Likelihood targets separation via increasing the likelihood between the outputs $y_i(n)$, $i = 1, 2, ..., N$ and the inputs $x_i(n)$, $i = 1, 2, ..., N$. The concept has been originally introduced in [10]. The cost function of the log-likelihood $L(\mathbf{W})$ of the demixing matrix $\mathbf{W}$ can be expressed as:

$$L(\mathbf{W}) = \log|\det(\mathbf{W})| + E\{\sum_{i=1}^{N} \log p_i(\mathbf{w}_i\mathbf{x})\} \tag{5.6}$$

where $E\{.\}$ refers to the expected value, $\mathbf{w}_i$ is the i^{th} row of the matrix $\mathbf{W}$ and $p_i(.)$ is the probability density function.
$L(\mathbf{W})$ has the gradient $\nabla L(\mathbf{W})$ as:

$$\begin{aligned}\nabla L(\mathbf{W}) &= (\mathbf{W}^{-1})^T - E\{\mathbf{g}(\mathbf{W}\mathbf{x})\mathbf{x}^T\} \\ &= (\mathbf{W}^{-1})^T - E\{\mathbf{g}(\mathbf{y})\mathbf{x}^T\}\end{aligned} \tag{5.7}$$

where $g(y_i) = -\frac{p'(y_i)}{p(y_i)}$.
Up to this point in the derivation of the cost function, no special assumptions or constraints are imposed on the update. However, researchers who investigated the suitable choices of the nonlinear functions $g(.)$ to ensure stability provided the choices based on requiring the output $\mathbf{y}$ to be zero mean and of unit variance [56].

Theorem 5.3.1. *[56], [1] "Denote by p_i the assumed densities of the independent components*

$$g(y_i) = -\log(p(y_i))' \qquad \Rightarrow g(y_i) = -\frac{p'(y_i)}{p(y_i)} \tag{5.8}$$

and constrain the estimate of the independent components y_i to be uncorrelated and to have unit variance. Then the ML estimator is locally consistent, if the assumed densities p_i fulfill

$$E\{\tilde{s}_i g_i(\tilde{s}_i) - g'(\tilde{s}_i)\} > 0 \tag{5.9}$$

where $\tilde{s}_i$ is the i^{th} source assumed to have unit variance for all $i = 1 \cdots N$ "

Applying the condition in the above theorem yields the already known choices for $g(.)$ as used in this thesis , namely, $g(.) = \tanh(.)$ for supergaussian data and $g(.) = (.)^3$ for subgaussian data.
The gradient in this case is chosen to follow the constrained update and yields the well-known natural gradient $\tilde{\nabla} L(\mathbf{W})$ that follows the geometry of the Riemannian manifold as follows:

$$\begin{aligned}\tilde{\nabla} L(\mathbf{W}) &= \nabla L(\mathbf{W})\mathbf{W}^T\mathbf{W} \\ &= (\mathbf{I}_{N\times N} - \mathbf{g}(\mathbf{y})\mathbf{y}^T)\mathbf{W}\end{aligned} \tag{5.10}$$

The natural gradient LMS algorithm is based on taking the instantaneous value of the update of the above gradient [56] so the update equation would be:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I}_{N\times N} - \mathbf{g}(\mathbf{y})\mathbf{y}^T)\mathbf{W}(t-1) \tag{5.11}$$

However, one may desire to implement a more general update that does not impose unnecessary and sometimes unrealistic constraints of having unit outputs y_i while keeping the stability condition intact. This can be achieved by the following proposed update to compensate for lifting the unit variance constraint.
Denote the source signals that are not required to be of unit variance as s_i then the relationship between s_i and $\tilde{s}_i$, its unit variance counterpart, is:

$$\tilde{\mathbf{s}} = \frac{1}{N}\mathbf{R}_{ss}^{-1}\mathbf{s} \tag{5.12}$$

where $\mathbf{R}_{ss}$ is the covariance matrix of the sources $\mathbf{s}$, which is equal to the autocorrelation matrix in this case as the sources are still assumed to be of zero mean.
The outputs $\mathbf{y}$ will consequently follow the same route and are no longer constrained to have unit variance. To have the nonlinearity $g(\mathbf{y})$ following the same stability conditions in the above theorem and maintain the same choices for super- and sub-gaussian data, the nonlinearities can be replaced by $\mathbf{R}_{ss}^{-1}g(\mathbf{y})$, where $\mathbf{y}$ now are not constrained to unit variance. This variant form can be easily proven mathematically. Considering the sources $\mathbf{s}$ to be the real unconstrained output and $\mathbf{y}$ to be its constrained counterpart, then the relationship between $\mathbf{y}$ and $\mathbf{s}$ can be written in the same manner as equation (5.12):

$$\mathbf{y} = \mathbf{R}_{ss}^{-1}\mathbf{s} \tag{5.13}$$

and consequently:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{s}} = \mathbf{R}_{ss}^{-1} \tag{5.14}$$

$$g(\mathbf{s}) = -\log(p(\mathbf{s}))' = -\log(p(\mathbf{y}))'\frac{\partial \mathbf{y}}{\partial \mathbf{s}} = -\mathbf{R}_{ss}^{-1}\log(p(\mathbf{y}))' = \mathbf{R}_{ss}^{-1}g(\mathbf{y}) \tag{5.15}$$

Using the above unconstrained modifications, a new form of update for the ML-function is:

$$\nabla L(\mathbf{W}) = (\mathbf{W}^{-1})^T - E\{\mathbf{R}_{ss}^{-1}\mathbf{g}(\mathbf{y})\mathbf{x}^T\} \tag{5.16}$$

The instantaneous natural gradient update form of the above Euclidean gradient is:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I}_{N\times N} - \mathbf{R}_{ss}^{-1}\mathbf{g}(\mathbf{y})\mathbf{y}^T)\mathbf{W}(t-1) \tag{5.17}$$

The above form can be considered a generalized form of natural gradient update. It easily reduces back to the form in equation 5.11 if the constraint of unit variance is re-imposed.
The discussion in this section along with the above generalized equation serve as a first step in the direction of a generalized form of the natural gradient algorithm. Other forms will follow in the next few sections. However, another concept used to develop the natural gradient, i.e. "Iterative inversion", which was developed in [1] needs to be revisited first. The following section is dedicated to present a summary of this concept and the algorithms developed in [1].

5.4 Review of Iterative Inversion Approach

In [1], an iterative inversion approach was implemented to provide a robust de-mixing system that led to a unified interpretation of various natural gradient approaches without explicitly using the Riemannian geometry.
The derivation starts by the maximum-likelihood cost function:

$$L(\mathbf{W}) = \log|\det(\mathbf{W})| + E\{\sum_{i=1}^{N} \log p_i(\mathbf{w}_i\mathbf{x})\} \tag{5.18}$$

At the optimal value $\mathbf{W}_{opt}$, the necessary condition for a correct estimate of the separation is:

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}}|_{\mathbf{W}=\mathbf{W}_{opt}} = (\mathbf{R}_{\psi\mathbf{x}} - \mathbf{W}^{-T})|_{\mathbf{W}=\mathbf{W}_{opt}} = 0 \tag{5.19}$$

where $\mathbf{R}_{\psi\mathbf{x}} = E[\psi(\mathbf{y})\mathbf{x}^T]$ and $\psi(.) = -\log(p(y_i))'$, which is substituted, as in the above section and other researchers' work, as the nonlinearity $g(\mathbf{y})$. However, the derivation in [1] chose to place the nonlinearity as a general $\psi(.)$ which can be given various different values. The above condition (5.19) can be re-written by taking the conjugate of both sides as:

$$\mathbf{W}_* = \mathbf{R}_{\mathbf{x}\psi}^{-1}|_{\mathbf{W}_*} \tag{5.20}$$

Or in terms of the mixing matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{R}_{\mathbf{x}\psi}|_{\mathbf{W}_*} \tag{5.21}$$

The above equation was then used to implement a recursive estimate of $\mathbf{W}$ as follows.
An iterative correction of the current estimate of the mixing system $\mathbf{A}$ in the direction of the robust estimate $\mathbf{R}_{\mathbf{x}\psi}|_{\mathbf{W}_*}$ can be implemented as:

$$\mathbf{A}(t) = \mathbf{A}(t-1) + \eta(\mathbf{R}_{\mathbf{x}\psi} - \mathbf{A}(t)) \tag{5.22}$$

or in terms of $\mathbf{W}$:

$$\mathbf{W}^{-1}(t) = \mathbf{W}^{-1}(t-1) + \eta(\mathbf{R}_{\mathbf{x}\psi} - \mathbf{W}^{-1}(t)) \tag{5.23}$$

Defining $\mu = \eta/(1+\eta)$, the above equation can be re-written as:

$$\mathbf{W}^{-1}(t) = (1-\mu)\mathbf{W}^{-1}(t-1) + \mu\mathbf{R}_{\mathbf{x}\psi} \tag{5.24}$$

$$\mathbf{W}^{-1}(t) = (1-\mu)\mathbf{W}^{-1}(t-1) + \mu\mathbf{W}^{-1}(t-1)\mathbf{W}(t-1)\mathbf{R}_{\mathbf{x}\psi} \tag{5.25}$$

$$\mathbf{W}^{-1}(t) = \mathbf{W}^{-1}(t-1)(\mathbf{I} + \mu(\mathbf{W}(t-1)\mathbf{R}_{\mathbf{x}\psi} - \mathbf{I})) \tag{5.26}$$

Then the above system can be inverted:

$$\mathbf{W}(t) = (\mathbf{I} + \mu(\mathbf{W}(t-1)\mathbf{R}_{\mathbf{x}\psi} - \mathbf{I}))^{-1}\mathbf{W}(t-1) \tag{5.27}$$

The term $\mathbf{W}(t-1)\mathbf{R}_{\mathbf{x}\psi} = \mathbf{R}_{\mathbf{y}\psi}$ is expressed in [1] as $\mathbf{R}_{\mathbf{gf}}$ where $f(\mathbf{y})$ is a second nonlinear function of $\mathbf{y}$[2].

$$\mathbf{W}(t) = (\mathbf{I} + \mu(\mathbf{R}_{\mathbf{gf}} - \mathbf{I}))^{-1}\mathbf{W}(t-1) \tag{5.28}$$

[2]This definition helps in providing a general solution that was afterwards compared to several existing blind signal separation algorithms, which were then presented as special cases of the final solution provided by the iterative inversion.

The first form of inversion of equation (5.27) can be obtained by the approximation:

$$(\mathbf{I} + \mu(\mathbf{R_{gf}} - \mathbf{I}))^{-1} \approx \mathbf{I} - \mu(\mathbf{R_{gf}} - \mathbf{I}) \tag{5.29}$$

For small μ, the update equation can thus be expressed as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - \mathbf{R_{gf}})\mathbf{W}(t-1) \tag{5.30}$$

The stability analysis of the above update can be obtained under the following conditions:

1. The step size μ is small enough.
2. The source signals $\mathbf{s}$ as well as the vector of separated outputs $\mathbf{y}$ are real zero mean signals.
3. The functions $f(.)$ and $g(.)$ are element-wise differentiable.

If the above three conditions are met, then the stability conditions can be summarized as [1]:

$$E[g_i']E[s_j f_j] + E[g_j']E[s_i f_i] > 0 \tag{5.31}$$
$$E[g_i' s_i f_i] + E[g_i s_i f_i'] > 0 \tag{5.32}$$
$$E[g_i']E[g_j']E[s_j f_j]E[s_i f_i] > E[f_i']E[f_j']E[s_j g_j]E[s_i g_i] \tag{5.33}$$

for all $i, j|_{i \neq j}$. In the case where $f(\mathbf{y}) = \mathbf{y}$, the stability conditions reduce to:

$$E[g_i'] > 0 \tag{5.34}$$
$$E[g_i' s_i^2] > 0 \tag{5.35}$$
$$E[g_i']E[g_j']E[s_i^2]E[s_j^2] > 1 \tag{5.36}$$

It can be proven that these reduced conditions can be met for the choice of $g(\mathbf{y}) = \tanh(\mathbf{y})$ for the supergaussian data and $g(\mathbf{y}) = \mathbf{y}^3$ for subgaussian data.
Substituting $\mathbf{R_{gy}} = g(\mathbf{y})\mathbf{y}^T$, equation (5.30) reduces to:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{W}(t-1) \tag{5.37}$$

which is the natural gradient algorithm. However, if the following inequality holds, then an interchange between $f(\mathbf{y})$ and $g(\mathbf{y})$ insures stability:

$$E[g_i']E[g_j']E[s_j f_j]E[s_i f_i] < E[f_i']E[f_j']E[s_j g_j]E[s_i g_i] \tag{5.38}$$

Thus, an alternative choice for the nonlinearity $f(\mathbf{y})$ is the tanh function for the subgaussian data and the cubic function for the supergaussian data. This leads to $g(\mathbf{y}) = \mathbf{y}$. These choices will guarantee the stability condition in (5.31).
After pointing out the stability conditions, another form of iterative inversion can be obtained by applying the Sherman-Morrison inversion formula [22] to $(\mathbf{I} + \mu(\mathbf{R_{gf}} - \mathbf{I}))^{-1}$ in equation

(5.28).
The Sherman -Morrison inversion formula states:

$$(\mathbf{I}+\mu(\mathbf{R_{gf}}-\mathbf{I}))^{-1}=((1-\mu)\mathbf{I}+\mu g(\mathbf{y})f^T(\mathbf{y}))^{-1} \tag{5.39}$$

$$((1-\mu)\mathbf{I}+\mu g(\mathbf{y})f^T(\mathbf{y}))^{-1}=\frac{1}{(1-\mu)}(\mathbf{I}-\frac{\mu}{(1-\mu)}\frac{g(\mathbf{y})f^T(\mathbf{y})}{1+\frac{\mu}{(1-\mu)}f^T(\mathbf{y}g(\mathbf{y}))}) \tag{5.40}$$

Substituting $\eta=\frac{\mu}{(1-\mu)}$, the above equation can be used along with equation (5.28) to produce the following update [1]:

$$\mathbf{W}(t)=(1+\eta)(\mathbf{I}-\eta\frac{g(\mathbf{y})f^T(\mathbf{y})}{1+\eta f^T(\mathbf{y})g(\mathbf{y})})\mathbf{W}(t-1) \tag{5.41}$$

which can be approximated, for $\eta \ll 1$, to produce:

$$\mathbf{W}(t)=\mathbf{W}(t-1)-\eta(\frac{\mathbf{I}-g(\mathbf{y})f^T(\mathbf{y})}{1+\eta f^T(\mathbf{y})g(\mathbf{y})})\mathbf{W}(t-1) \tag{5.42}$$

which is the normalized Equivariant algorithm presented by Cardoso in [19].

5.5 New Generalized Iterative Inversion of Blind Signal Separation Problem

In this section, the iterative inversion approach is extended to the general case where the variance of the sources is not constrained to the unit matrix.
This formulation will allow providing a general framework from which several algorithms follow as special cases. This framework also leads to new algorithms with new convergence and stability behaviors.

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}}|_{\mathbf{W}=\mathbf{W}_{opt}}=(\mathbf{R_{ss}}^{-1}\mathbf{R}_{\psi\mathbf{x}}-\mathbf{W}^{-T})|_{\mathbf{W}=\mathbf{W}_{opt}}=0 \tag{5.43}$$

In terms of the mixing system $\mathbf{A}$:

$$\mathbf{A}_*=\mathbf{R_{xs}}R_{\mathbf{ss}}^{-1} \tag{5.44}$$

The above equation can also be considered as the Wiener solution for the mixing filter $\mathbf{A}$. Considering $\mathbf{s}$ as the input to the Wiener filter, which is the matrix $\mathbf{A}$ in this case, and having $\mathbf{x}$ as the output, the cross-correlation $\mathbf{R_{sx}}=E\{\mathbf{s}\mathbf{x}^T\}$ and the autocorrelation $\mathbf{R_{ss}}=E\{\mathbf{s}\mathbf{s}^T\}$, then the Wiener filter will give an identical form to the above equation.
With the choice of $\mathbf{R_{xs}}=\mathbf{R}_{\mathbf{x}\psi}$ and $\mathbf{R_{ss}}=\mathbf{I}$, the Wiener solution reduces immediately to the iterative forms developed in the previous sections.
In the following development of new algorithms $\mathbf{R_{ss}}$ is kept explicit for generalization. Different iterative algorithms will result from different choices of this correlation matrix.

5.5.1 First LMS- based Algorithm: LMS-1

To calculate the update we can use the iterative equation [93]:

$$\mathbf{A}(t)=\mathbf{A}(t-1)(I-\mu\mathbf{R_{ss}})+\mu\mathbf{R}_{\mathbf{x}\psi} \tag{5.45}$$

It is easy to check that the above iterative update will converge to the optimal solution [93]. After k iterations:

$$\mathbf{A}(t) = \mu \mathbf{R}_{\mathbf{x}\psi}[\sum_{m=1}^{k-1}(I - \mu \mathbf{R}_{\mathbf{ss}})] \tag{5.46}$$

The power series for a scalar constant α that has a magnitude less than 1 , $|\alpha| < 1$ is:

$$\lim_{k\to\infty} \sum_{m=0}^{k-1} |\alpha|^m = \frac{1}{1-|\alpha|} \tag{5.47}$$

Similarly, the matrix $(I - \mu \mathbf{R}_{\mathbf{ss}})$ is less than the identity matrix such that a similar summation can be implemented and applied to equation (5.45). At iteration k, the summation will be of the following form:

$$\mathbf{A}(k) = \mu \mathbf{R}_{\mathbf{x}\psi}\{[\mathbf{I} - (\mathbf{I} - \mu \mathbf{R}_{\mathbf{ss}})^k].[\mathbf{I} - (\mathbf{I} - \mu \mathbf{R}_{\mathbf{ss}})]^{-1}\} \tag{5.48}$$

with the assumption:

$$\lim_{k\to\infty} (\mathbf{I} - \mu \mathbf{R}_{\mathbf{ss}})^k \to \underline{\mathbf{0}}_{N\times N} \tag{5.49}$$

where $\underline{\mathbf{0}}_{N\times N}$ is the zero matrix of the same size as the identity matrix $\mathbf{I}$. Applying the above assumption to the summation equation in (5.48), then the matrix $\mathbf{A}$ will indeed converge eventually to the optimal solution:

$$\lim_{k\to\infty} \mathbf{A}(k) = \mathbf{R}_{\mathbf{xs}} R_{\mathbf{ss}}^{-1} \tag{5.50}$$

More details about this update form is presented in [93]. After proving the convergence of the update form (5.45), the update equation can now be re-written in terms of $\mathbf{W}$ as:

$$\mathbf{W}^{-1}(t) = \mathbf{W}^{-1}(t-1)(\mathbf{I} - \mu \mathbf{R}_{\mathbf{ss}}) + \mu \mathbf{W}^{-1}(t-1)\mathbf{R}_{\mathbf{gf}} \tag{5.51}$$

$$\mathbf{W}(t) = (I + \mu(\mathbf{R}_{\mathbf{gf}} - \mathbf{R}_{\mathbf{ss}}))^{-1}\mathbf{W}(t-1) \tag{5.52}$$

using the approximation: $(\mathbf{I} + \mu(\mathbf{R}_{\mathbf{fg}} - \mathbf{R}_{\mathbf{ss}}))^{-1} \approx (I - \mu(\mathbf{R}_{\mathbf{gf}} - \mathbf{R}_{\mathbf{ss}}))$, the above equation becomes:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{R}_{\mathbf{ss}} - \mathbf{R}_{\mathbf{gf}})\mathbf{W}(t-1) \tag{5.53}$$

choosing $f(\mathbf{y}) = \mathbf{y}$:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{R}_{\mathbf{ss}} - \mathbf{R}_{\mathbf{gy}})\mathbf{W}(t-1) \tag{5.54}$$

Several choices can be made for the value of the explicit matrix $\mathbf{R}_{\mathbf{ss}}$.

1. For $\mathbf{R}_{\mathbf{ss}} = \mathbf{I}$, the above equation reduces to the natural gradient algorithm:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - \mathbf{R}_{\mathbf{gy}}))\mathbf{W}(t-1) \tag{5.55}$$

2. Another valid choice is to choose to replace $\mathbf{s}$ by its estimated value $\mathbf{y}$. Then, $\mathbf{R}_{\mathbf{ss}} = E\{\mathbf{y}\mathbf{y}^T\}$, which can be approximated on-line at a given time instant t as

$$\mathbf{R}_{\mathbf{ss}}(t) = \lambda \mathbf{R}_{\mathbf{ss}}(t-1) + (1-\lambda)\mathbf{y}\mathbf{y}^T \tag{5.56}$$

which can be implemented with the forgetting factor $\lambda < 1$ chosen close to 1 and the initial matrix $\mathbf{R}_{\mathbf{ss}}(0) = \mathbf{I}$.

3. Same as the above choice 2, $\mathbf{R_{ss}}$ can still be assumed to be $\mathbf{R_{yy}}$. However, one can choose to replace the expectation $\mathbf{R_{yy}} = E\{\mathbf{y}\mathbf{y}^T\}$ by the instantaneous value $\mathbf{y}\mathbf{y}^T$. The update in this case can be estimated using the approximation $(\mathbf{I} + \mu(\mathbf{R_{fg}} - \mathbf{R_{ss}}))^{-1} = (\mathbf{I} + \mu(g(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{y}^T))^{-1} = (\mathbf{I} - \mu(g(\mathbf{y}) - \mathbf{y})\mathbf{y}^T)$.

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{y}\mathbf{y}^T - g(\mathbf{y})\mathbf{y}^T))\mathbf{W}(t-1) \tag{5.57}$$

This update form is exactly the same update form obtained from the newly developed Bussgang cost function in Chapter 4. The equations (5.58) and (4.27) are identical, which gives this generalized form of iterative inversion extra credibility as a family of separation algorithms.

4. Another choice comes from a closer look at the autocorrelation matrix $\mathbf{R_{ss}}$. The sources must be independent to be considered for separation. Independent sources are also uncorrelated by definition. Thus, the autocorrelation matrix can be assumed diagonal. When the unknown sources $\mathbf{s}$ are replaced by their estimates $\mathbf{y}$, the above two assumptions for the value of the autocorrelation matrix eventually lead to a diagonal matrix $\mathbf{R_{yy}}$. However, another valid choice is to substitute for $\mathbf{R_{ss}} = \mathbf{\Lambda}$ where $\mathbf{\Lambda}$ is a diagonal matrix without constraining its elements to be one, providing the generalized natural gradient algorithm [6]:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{\Lambda} - \mathbf{R_{gy}}))\mathbf{W}(t-1) \tag{5.58}$$

This diagonal matrix can be chosen to be of diagonal elements $\mathrm{diag}(\mathbf{R_{ss}})$ [3], meaning that the update constrains the choice of the sources to be always de-correlated. The above diagonal form for $\mathbf{R_{ss}}$ was validated through simulations and they proved in most cases to work with better convergence and exhibit more robustness than the original natural gradient algorithm, where the autocorrelation of the sources is fixed to unity.

Another advantage of this LMS-1 form for natural gradient is that it is very suitable for separation when the number of sources is not correctly estimated, i.e. where the number of sensors M is larger than the number of sources N. This case leads to ill-conditioned matrices. The nonholonomic natural gradient [9] is developed especially for such cases. However, the nonholonomic natural gradient is it performs poorly in the normal mixing case, i.e. when $N = M$ [76].
This is not the case with the LMS-1 algorithm. It performs well in the conventional mixing case, i.e. $N = M$ and also is stable when the number of sources is unknown. This important property of the algorithm is proved via simulations in Chapter 6.
Considering the computational complexity of this algorithm, one can see that it is the same as the calculation of the natural gradient algorithm except for the term $\mathbf{R_{ss}}$ which replaces the fixed identity matrix in the natural gradient algorithm. For the different cases of $\mathbf{R_{ss}}$, different complexities will result. Considering that a multiplication, an addition or a multiplication and addition require the same computational load, which is true for many processors:

[3] diag(.) is a dual function. When applied to a matrix it produces a vector containing the diagonal elements of the matrix. While when applied to a vector it produces a diagonal matrix with diagonal elements equal to the elements in the vector.

1. If a correlation-like matrix, i.e. $\mathbf{R_{yy}}$ or $\mathbf{R_{yg(y)}}$ is used, an additional computational load of the order N^2 is added.

2. If the identity matrix is replaced with a diagonal matrix, then a mere addition of order N calculations is required. This is negligible considering that the natural gradient update, in general, requires computations in the order of N^2.

5.5.2 Second LMS-based Algorithm: LMS-2

Another way to implement recursion is to use recursive corrections of the direction of the mixing matrix $\mathbf{A}$ as a different starting point, i.e. update equation. The LMS-2 algorithm uses the update equation (5.22), which is a different update equation than (5.45) used to develop LMS-1. This will result in a different form for a different algorithm. The algorithm starts with the update equations (5.22) and (5.23), with the only difference being the replacement of the robust estimate $\mathbf{R_{x\psi}}$ in (5.22) by $\mathbf{R_{x\psi}R_{ss}^{-1}}$. Here, the choice of the function ψ is limited to the nonlinearity $g(\mathbf{y})$ with the stability condition derived in section 5.4. Thus,

$$\mathbf{A}(t) = \mathbf{A}(t-1) + \eta(\mathbf{R_{gx}}R_{ss}^{-1} - \mathbf{A}(t)) \tag{5.59}$$

In terms of $\mathbf{W}$ the above equation can be re-written as:

$$\mathbf{W}^{-1}(t) = \mathbf{W}^{-1}(t-1) + \eta(\mathbf{R_{gx}}R_{ss}^{-1} - \mathbf{W}(t)^{-1}) \tag{5.60}$$

Defining $\mu = \eta/(1+\eta)$, the above equation can be re-written as:

$$\mathbf{W}^{-1}(t) = (1-\mu)\mathbf{W}^{-1}(t-1) + \mu\mathbf{R_{gx}R_{ss}^{-1}} \tag{5.61}$$

Inverting the above system:

$$\mathbf{W}(t) = (\mathbf{I} + \mu(\mathbf{W}(t-1)\mathbf{R_{gx}R_{ss}^{-1}} - \mathbf{I}))^{-1}\mathbf{W}(t-1) \tag{5.62}$$

The inversion $(\mathbf{I}+\mu(\mathbf{W}(t-1)\mathbf{R_{gx}}R_{ss}^{-1}-\mathbf{I}))^{-1}$ can be approximated as $(\mathbf{I}-\mu(\mathbf{W}(t-1)\mathbf{R_{gx}}R_{ss}^{-1}-\mathbf{I}))$. This approximation yields the form LMS-2:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - \mathbf{R_{gy}R_{ss}^{-1}}))\mathbf{W}(t-1) \tag{5.63}$$

Again as in the previous section, i.e. LMS-1 section, several choices can be introduced for the matrix $\mathbf{R_{ss}} = \mathbf{I}$, leading to variant forms of the LMS-2 algorithm.

1. For the choice of $\mathbf{R_{ss}} = \mathbf{I}$, the algorithm reduces to the natural gradient algorithm.

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - \mathbf{R_{gy}})\mathbf{W}(t-1) \tag{5.64}$$

2. Choosing $\mathbf{R_{ss}} = \mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is a diagonal matrix, leads to a different set of normalized algorithms. Choosing $\mathbf{\Lambda} = \text{diag}(E\{g(\mathbf{y})\mathbf{y}^T\})$ or $\mathbf{\Lambda} = \text{diag}(E\{\mathbf{y}\mathbf{y}^T\})$ leads to a form that is very similar to Hyvärinen is fixed point (off-line) update implemented for the maximum-likelihood [56].

$$\mathbf{W} = \mathbf{W} + \text{diag}(\alpha)[\text{diag}(\beta) - E\{g(y)y^T\}]\mathbf{W} \tag{5.65}$$

Here the use of the sample by sample update based on t was omitted because the algorithms works mainly in batch-mode. β is a vector containing the diagonal elements of $\mathbf{y}g(\mathbf{y})^T$:

$$\beta = \text{diag}(\mathbf{y}g(\mathbf{y})^T) \tag{5.66}$$

On the other hand, the diagonal matrix diag(α) contains the $\alpha_i, i = 1 \ldots N$:

$$\alpha_i = 1/(\beta_i - E\{g'(y_i)\}) \tag{5.67}$$

Here if the second term in α_i containing the gradient of $g(y_i)$, i.e. if $E\{g'(y_i)\}$ is small enough then the update (5.65) can be approximated to be:

$$\mathbf{W} = \mathbf{W} + E\{[\mathbf{I} - E\{g(\mathbf{y})\mathbf{y}^T\}(\text{diag}(\beta))^{-1}]\}\mathbf{W} \tag{5.68}$$

which can be implemented on-line by choosing to incorporate a step size $\mu < 1$ as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\mathbf{I} - E\{g(\mathbf{y})\mathbf{y}^T\}(\text{diag}(\beta))^{-1}]\mathbf{W}(t-1) \tag{5.69}$$

which is a similar form of update to the LMS-2 introduced in equation (5.63)

3. Another valid choice is to set $\mathbf{R_{ss}} = E\{\mathbf{y}\mathbf{y}^T\}$. Again, the expected value $E\{.\}$ can be calculated on-line as in (5.56)

$$\mathbf{R_{ss}}(t) = \lambda\mathbf{R_{ss}}(t-1) + (1-\lambda)\mathbf{y}\mathbf{y}^T \tag{5.70}$$

The results, to be shown in the simulation section, show that the algorithm is valid and that separation can be reached. The algorithm performance is discussed along with a comparison with other developed algorithms.

When considering the computations required for such an update, it can be seen that the main complexity addition will stem from the calculation of the inverse matrix $\mathbf{R_{ss}^{-1}}$. The inverse of the matrix can be calculated by the matrix inversion lemma to reduce the complexity. However, the computation of the matrix inverse by the lemma will still require computations at least in the order N^3 [85]. However, if the matrix $\mathbf{R_{ss}}$ is substituted as a diagonal matrix, then the calculation of the inverse will be in the order of N, usually $10N$ [60].

5.5.3 Third LMS-based Algorithm: LMS-RLS

Another form of update can be obtained by re-writing equation (5.62) as:

$$\mathbf{W}(t) = (\mathbf{I} + \mu(\mathbf{R_{gy}} - \mathbf{R_{ss}})\mathbf{R_{ss}^{-1}})^{-1}\mathbf{W}(t-1) \tag{5.71}$$

Using the approximation $(\mathbf{I} + \mu(\mathbf{R_{gy}} - \mathbf{R_{ss}})\mathbf{R_{ss}^{-1}})^{-1} \approx (\mathbf{I} - \mu(\mathbf{R_{gy}} - \mathbf{R_{ss}})\mathbf{R_{ss}^{-1}})$ leads to the following update:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{R_{ss}} - \mathbf{R_{gy}})\mathbf{R_{ss}^{-1}}\mathbf{W}(t-1) \tag{5.72}$$

This update is similar in form to the classical filtering RLS-update and thus is given the name "LMS-RLS". Although being a minor modification of the "LMS-2" algorithm, with the choice of alternative windowing update for the correlation term $\mathbf{R_{ss}}$, the algorithm results in different convergence behavior.

1. Again, the choice of $\mathbf{R_{ss}} = \mathbf{I}$ reduces the update to the natural gradient update.
2. Another valid choice is $\mathbf{R_{ss}} = \mathbf{R_{yy}} = E\{\mathbf{y}\mathbf{y}^T\}$.
3. The choice of $\mathbf{R_{ss}} = \text{diag}(g(\mathbf{y})\mathbf{y}^T)$ leads to an update that is computationally simple and can produce good results as will be seen in the simulations.

The computations required for such an RLS-update is still governed by the calculation of the inverse matrix $\mathbf{R_{ss}^{-1}}$. In the same manner used to calculate the additional complexity of the LMS-2 algorithm, this algorithm should also require N^3 order computations for the choice $\mathbf{R_{ss}} = \mathbf{R_{yy}}$ and the order of N for the choice $\mathbf{R_{ss}} = \text{diag}(g(\mathbf{y})\mathbf{y}^T)$.

5.6 RLS-Iterative Inversion Approach

In this section, a new RLS approach to blind signal separation that does not require pre-whitening is developed. The algorithm is based on iterative inversion and the use of the matrix inversion lemma [22].
For the following derivation, the choice of the autocorrelation matrix $\mathbf{R_{ss}}$ is limited to the identity matrix $\mathbf{I}$. Thus, the solution for the optimal value of $\mathbf{W}$ is:

$$\mathbf{W}_* = \mathbf{R_{xg}^{-1}}|_{\mathbf{W}_*} \tag{5.73}$$

or as can be written in an iterative formula:

$$\mathbf{W}(t) = \mathbf{R_{gy}^{-1}}\mathbf{W}(t-1) \tag{5.74}$$

$\mathbf{W}(t)$ can be updated iteratively as follows:
Consider:

$$\mathbf{A}(t) = \lambda\mathbf{A}(t-1) + (1-\lambda)\mathbf{A}(t-1)\mathbf{R_{gy}} \tag{5.75}$$

Substituting for $\mathbf{A}$ by $\mathbf{W}^{-1}$, the above equation can be re-written as:

$$\mathbf{W}^{-1}(t) = \lambda\mathbf{W}^{-1}(t-1) + (1-\lambda)\mathbf{W}^{-1}(t-1)\mathbf{R_{gy}} \tag{5.76}$$

In terms of $\mathbf{W}$, the above equation can be re-written as:

$$\mathbf{W}(t) = [\lambda\mathbf{I} + (1-\lambda)\mathbf{R_{gy}}]^{-1}\mathbf{W}(t-1) \tag{5.77}$$

Replacing $\mathbf{R_{gy}}$ by its instantaneous value, $g(\mathbf{y})\mathbf{y}^T$, the above equation will be:

$$\mathbf{W}(t) = [\lambda\mathbf{I} + (1-\lambda)g(\mathbf{y})\mathbf{y}^T]^{-1}\mathbf{W}(t-1) = \frac{1}{\lambda}[\,\mathbf{I} + \frac{(1-\lambda)}{\lambda}g(\mathbf{y})\mathbf{y}^T]^{-1}\mathbf{W}(t-1) \tag{5.78}$$

Using the Sherman-Morrisson inversion lemma will give the following recursive update:

$$\mathbf{W}(t) = \frac{1}{\lambda}[\mathbf{I} - \frac{g(\mathbf{y}(t))\mathbf{y}(t)^T}{\alpha + g(\mathbf{y}(t))^T\mathbf{y}(t)}]\mathbf{W}(t-1) \tag{5.79}$$

where $\alpha = \frac{\lambda}{(1+\lambda)}$. The last equation represents the RLS-update for the algorithm. The algorithm works very well and exhibits a good stability and a small steady state error.

Other forms of this RLS-update can be developed based on the choice of $\mathbf{R_{ss}}$ (excluding $\mathbf{R_{ss}} = \mathbf{I}$, which was covered in this section).
Moving to the additional computations required for this algorithm, it is easily seen that the main addition is in the division step, i.e. dividing each element in the $N \times N$ matrix $g(\mathbf{y}(t))\mathbf{y}(t)^T$ by $(\alpha + g(\mathbf{y}(t))^T\mathbf{y}(t))$ which will add the order of N additional computations to each of the N^2 elements in the dividend matrix. The computational cost will be thus in the order of N^3.
The following is a diagram summarizing the "Generalized-Iterative inversion family".

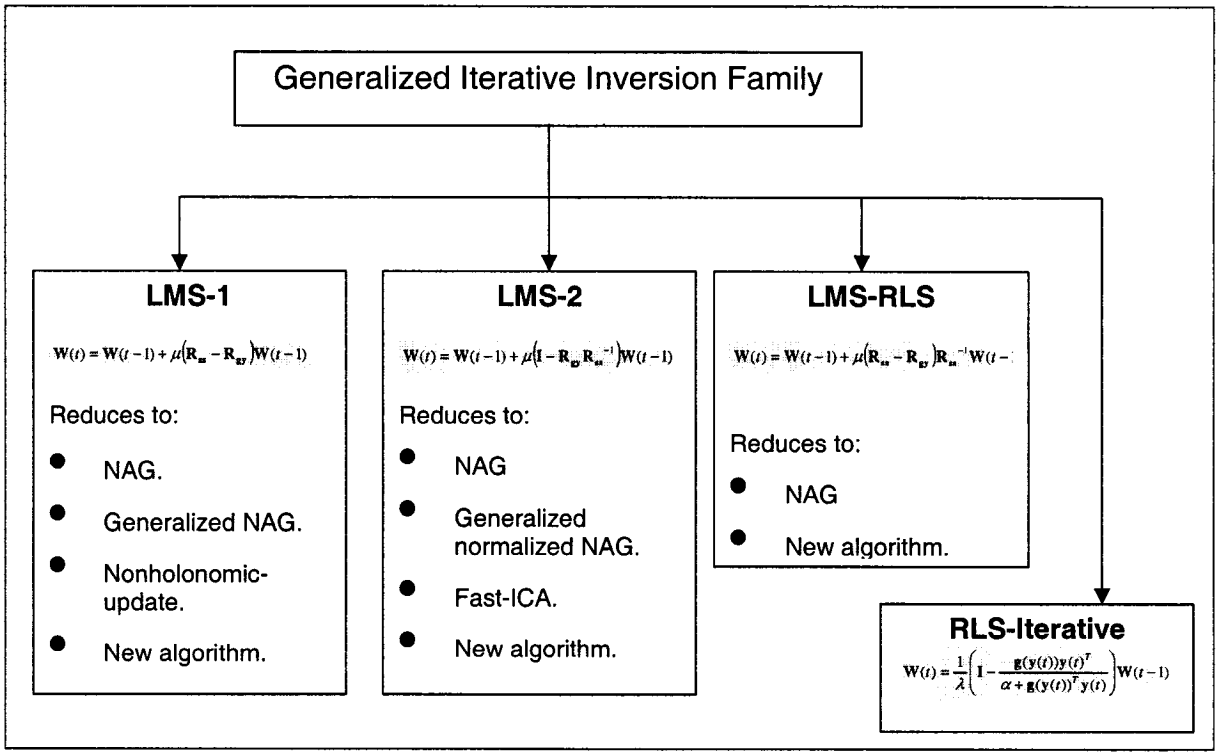

Figure 5.1: **Generalized Iterative Inversion Natural Gradient Family**

5.7 Simulation and Results

The above algorithms were compared by considering the separation of the following four signals. These signals have been proposed by Amari. et al. and they have been used in the following journal papers: [2], [97], [100] and [76]. More detailed simulation results are presented in Chapter 6 :

$$s_t = [\text{sign}(\cos(2\pi 155t), \sin(2\pi 800t), \sin(2\pi 300t + 6\cos(2\pi 60t)), \sin(2\pi 90t)]^T$$

with the sampling rate of 10 kHz. The algorithms compared are:

- **LMS-1** with the update equation:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{LMS-1}(\mathbf{R_{ss}} - \mathbf{R_{gy}}))\mathbf{W}(t-1) \tag{5.80}$$

μ_{LMS-1} is chosen to be 0.005, and $\mathbf{R_{ss}} = \text{diag}(\mathbf{yy}^T)$. The diag(.) function transforms a square matrix to diagonal matrix by setting the off-diagonal values to zeros.

- **LMS-2** This algorithm has the update equation as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{LMS-2}(\mathbf{I} - \mathbf{R_{gy}}\mathbf{R_{ss}^{-1}}))\mathbf{W}(t-1) \tag{5.81}$$

μ_{LMS-2} is chosen to be 0.0005.
$\mathbf{R_{ss}}(t) = E\{\text{diag}(\mathbf{yy}^T)\} = \lambda\mathbf{R_{ss}}(t-1) + (1-\lambda)\text{diag}(\mathbf{yy}^T)(t)$, and $\lambda = 0.99$.

- **LMS-RLS** This algorithm is represented by the update equation:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{LMS-RLS}(\mathbf{R_{ss}} - \mathbf{R_{gy}})\mathbf{R_{ss}^{-1}}\mathbf{W}(t-1) \tag{5.82}$$

$\mu_{LMS-RLS}$ is chosen to be 0.01.
$\mathbf{R_{ss}}(t) = E\{\text{diag}(\mathbf{yy}^T)\} = \lambda\mathbf{R_{ss}}(t-1) + \text{diag}(\mathbf{yy}^T)(t)$, and $\lambda = 0.99$.

- **RLS-Iterative algorithm**: This algorithm introduced in section 5.6, is considered the best performing algorithm presented in the family of NAG-based algorithms. It has an RLS-update following the equation 5.79:

$$\mathbf{W}(t) = \frac{1}{\lambda}[\mathbf{I} - \frac{g(\mathbf{y}(t))\mathbf{y}(t)^T}{\alpha + g(\mathbf{y}(t))^T\mathbf{y}(t)}]\mathbf{W}(t-1)$$

The forgetting factor λ can be set to two different values with a different steady state error resulting for each case . In "RLS-Iterative 1" the forgetting factor starts at 0.99 at $t = 0$ and increases to 0.9995 at $t = 6000$. In the second version "RLS-Iterative 2", λ starts at 0.99 at $t = 0$ and it increases to 0.99995 at $t = 6000$. The use of a larger forgetting factor permitted the algorithm to converge to smaller steady state error.

For the first three above algorithms $\mathbf{R_{ss}} = \text{diag}(\mathbf{yy}^T)$. The algorithms are compared to:
NAG: the natural gradient algorithm which can be seen as a special case of any of the above algorithms with the choice of $\mathbf{R_{ss}}$ to be equal to the identity matrix. The choice for the nonlinear function $g(\mathbf{y})$ is chosen according to [98], [97] as:

$$g(y_i) = y_i^3 \tag{5.83}$$

The following is a figure presenting a sample run of the above algorithms for the arbitrary mixing matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 0.5620 & 0.3302 & 0.9791 & 0.8699 \\ 0.8628 & 0.5317 & 0.6344 & 0.6985 \\ 0.2021 & 0.5995 & 0.3742 & 0.6742 \\ 0.5013 & 0.3323 & 0.7566 & 0.0418 \end{pmatrix}$$

As can be seen in the figure, the RLS-Iterative inversion algorithm provides the best performance, i.e. the fastest convergence speed and the smallest steady state error. All the other algorithms converge to the same steady state error with different convergence speed. The second best algorithm is the LMS-RLS algorithm followed by the LMS-2 algorithm. The LMS-1 algorithm converges faster than the natural gradient algorithm.

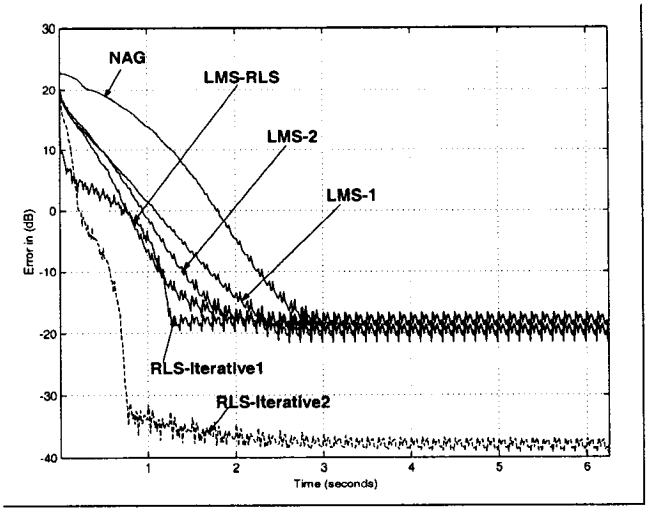

Figure 5.2: **Results for a sample run of the Natural-gradient family algorithms**

This sample result can be considered a faithful representation of the average performance of the algorithms. More simulations are presented in Chapter 6. The LMS-1 algorithm might seem inferior to the other proposed algorithms. However, the LMS-1 exhibits a very robust performance in the case of under-determined BSS, i.e. when the number of sources N is less than the number of sensors M. Simulations on this specific case are also presented in Chapter 6.

Despite having excellent results when tested with the above set of data, these algorithms did not perform very well when applied to speech signals. This type of behavior and difficulty when considering speech signals has been encountered before in the last chapter. Several new algorithms were developed in Chapter 4 and specifically tested with speech signals. This stimulates the idea of trying to extend the work done in the last chapter, more specifically the idea of non-linear RLS, which successfully managed to separate pre-whitened speech signals, to the general case of natural gradient, where pre-whitening is not applied. In the following section, such an extension is considered and proves to be successful.

5.8 Non-Linear RLS Update for Natural Gradient

This section presents another fast converging algorithm based on the concept of nonlinear-RLS update. One way to implement an RLS-type algorithm for the above situation can be achieved in a similar manner to the RLS-BSS-Stiefel update, a non-linear RLS implementation that was presented in Chapter 4, section 4.9. It is possible to apply the chain rule to define an "equivalent-error" signal and an "equivalent-input" signal in the derivative of any non-linear cost function L:
$L = f_1(\phi), \phi = f_2(\psi), \psi = f_3(\mathbf{y}), \mathbf{y} = f(\mathbf{W})$ as follows:

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \phi} \times \frac{\partial \phi}{\partial \psi} \times \frac{\partial \psi}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{W}} \tag{5.84}$$

The update rule for $\mathbf{W}$ can then be written as:

$$\mathbf{W}_{new} = \mathbf{W}_{old} - \mu \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W}_{new} = \mathbf{W}_{old} - \mu \frac{\partial L}{\partial \mathbf{W}} = \mathbf{W}_{old} + \mu \frac{\partial L}{\partial \phi} \times \frac{\partial \phi}{\partial \psi} \times \frac{\partial \psi}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{W}}$$

$$\mathbf{W}_{new} = \mathbf{W}_{old} + \mu \text{ equivalent error} \times \text{equivalent input}$$

The terms equivalent error and equivalent input are analogous to the classical adaptive filtering where $\frac{\partial J}{\partial \mathbf{y}}$ is the negative of the error signal and $\frac{\partial \mathbf{y}}{\partial \mathbf{W}}$ is the input signal $\mathbf{x}$. After forming the gradient in this chain formula, one can split it into an "equivalent input signal" and an "equivalent error signal". Then, it is possible to multiply this input by the inverse of a correlation matrix as in classical RLS filtering. For example the above equation can be seen as:

$$\frac{\partial L}{\partial \mathbf{W}} = \underbrace{\frac{\partial L}{\partial \phi}}_{equivalent-error} \times \underbrace{\frac{\partial \phi}{\partial \psi} \times \frac{\partial \psi}{\partial y} \times \frac{\partial \mathbf{y}}{\partial \mathbf{W}}}_{equivalent-input} \tag{5.85}$$

For the maximum-likelihood cost function $L(\mathbf{W})$, since there is no clear distinction between the "input" and the "error" signal, it was chosen to take the gradient of the cost function with respect to the Lie-algebra matrix, $\mathbf{X}$, as a whole and try to apply the RLS algorithm to speed its convergence.

The algorithm is based on a variant of the natural gradient, namely, the nonholonomic-natural gradient:

$$\mathbf{\Lambda} = \text{diag}(g(\mathbf{y})\mathbf{y}^T) \tag{5.86}$$

Denoting the update matrix on the lie algebra space by $\Delta\mathbf{X}$:

$$\Delta\mathbf{X} = \Delta\mathbf{W}\mathbf{W}^{-1} \tag{5.87}$$

Then the case of the nonhlonomic natural gradient, $\Delta\mathbf{X}$ has the form:

$$\Delta\mathbf{X} = \mathbf{\Lambda} - g(\mathbf{y})\mathbf{y}^T = \text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T \tag{5.88}$$

Defining $\mathbf{R}$ as:

$$\mathbf{R} = E\{\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T\}_{vec} E\{\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T\}_{vec}^T \tag{5.89}$$

where $(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec}$ is a re-arrangement of the $N \times N$ matrix $(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)$ as a vector of size $N^2 \times 1$ via column stacking. The resulting $\mathbf{R}$ is thus a matrix of size $N^2 \times N^2$.

The update equation of the nonlinear RLS-algorithm can then be developed by the following series of equations:

$$\mathbf{R} = E\{\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T\}_{vec} E\{\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T\}_{vec}^T \tag{5.90}$$

$$\mathbf{R}(t) = \mathbf{R}(t-1) + (\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec}(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec}^T \tag{5.91}$$

$$(\Delta\mathbf{X})_{vec} = \mathbf{R}^{-1}(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec} \tag{5.92}$$

$$\Delta\mathbf{X} = mat(\Delta\mathbf{X})_{vec} \tag{5.93}$$

$$\mathbf{W} = \exp(\mu\Delta\mathbf{X})\mathbf{W} \approx \mathbf{W} + \mu\Delta\mathbf{X}\mathbf{W} \tag{5.94}$$

where the function mat(.) is the same as defined in section 4.9. mat(.) transforms a vector of size $N^2 \times 1$ into a matrix of size $N \times N$ via column stacking. The following figure represents a sample simulation run of the algorithm with comparison to the natural gradient algorithm:

$$\mathbf{W} = \mathbf{W} + \mu_{NAG}(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{W} \tag{5.95}$$

The test data are 4 audio files sampled at 8 kHz and of duration 6.5 sec.:

1. 1 female: known as female-1 in Chapter 6 audio benchmarks. This file is obtained from [48]. The speaker is a female reading a sentence in English.

2. 2 males: known as Male-1 and Male-2 in Chapter 6 audio benchmarks. Male-1 is obtained from [48]. This file is a male speaker reading a sentence in English. Male-2 is obtained from [73]. It is also for a male speaker with similar voice properties to the above speaker and reading a different sentence in English, with similar style and speed of reading.

3. one soft music: known as music-1 in the same benchmarks group.

The above choice of audio data is to test the algorithm with general mixing cases, where the audio scene consists of various audio data of different types. This general mixing situation is the most likely scenario in practice and ensures that the compared algorithms are not biased towards a certain type of audio or speech data. More information about the benchmark data is presented in Chapter 6 along with more extensive simulations. The mixing matrix is randomly generated. For the following results the matrix $\mathbf{A}$ has the value:

$$\mathbf{A} = \begin{pmatrix} 0.9 & 0.9 & 0.8 & 0.5 \\ 0.9 & 1.2 & 1.3 & 0.99 \\ 0.6 & 0.8 & 0.8 & 0.84 \\ 0.25 & 0.32 & 0.5 & 0.41 \end{pmatrix}$$

The step size for the natural gradient algorithm μ_{NAG} is set to 0.0005 and the step size for the nonlinear RLS-nonholonomic is set to 1 to produce a comparable steady state error. The following figure shows the results for the above mixing scenario.
From the figure, it can be seen that the idea of nonlinear-RLS update can be successfully extended to the General Riemannian case (i.e. non-prewhitened case). The proposed algorithm converges slightly faster than the conventional natural gradient algorithm while achieving better steady state error. Clearly, the additional computational load for this algorithm is governed by the calculation of the inverse matrix. Although the matrix has a rather sparse nature, due to the fact that $(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)$ has zero diagonal elements, the update should still requiring $(N^2)^3 = N^6$-order of computations.
In all of the algorithms developed so far, the special geometric nature of the Riemannian space was limited to the calculation of the first-order gradient, i.e. natural gradient. As has been discussed in Chapter 3, extensive use of the geometry of the optimization space can be crucial in achieving improved performance without compromising simplicity or being obliged to massively invest in computations. In the rest of this chapter, a comprehensive consideration of second order-updates for the maximum-likelihood cost function is considered using the differential geometry.

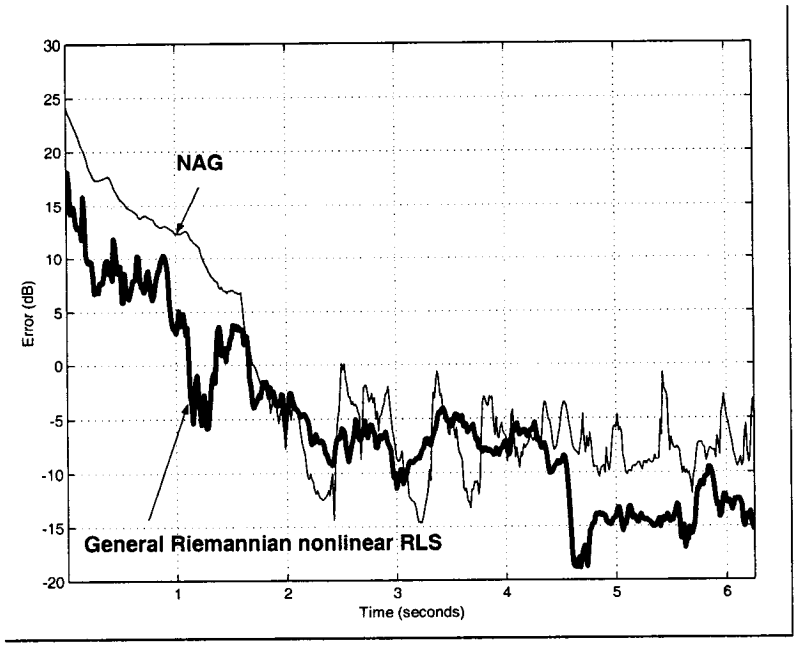

Figure 5.3: **Results for a sample run of the General Riemannian Nonlinear-RLS Algorithm**

5.9 Second-Order Riemannian Algorithms for Maximum-Likelihood Blind Signal Separation

In this section a group of algorithms that use the concept of second order optimization are presented. This second order optimization is carried with consideration to the Riemannian nature of the optimization space of the blind signal separation and thus is different from regular Newton-based algorithms which have been proven by Amari et al. to be inferior to the first order natural gradient algorithm [5].

In order to derive the second order derivative on the Riemannian manifold for invertible matrices (general linear (GL- group)), it is necessary to determine the actual gradient used for update on this specific manifold.

Through the study of Riemannian geometry [90], it was shown that the Riemannian gradient should be calculated on the Lie algebra flat space and then the update can be calculated through exponential mapping to the corresponding new point on the Lie group as explained in Chapter 3. The new point $\mathbf{W}_{new}$ should then be represented in the form:

$$\mathbf{W}_{new} = \exp^{(\mu\Delta\mathbf{X})}\mathbf{W}_{old} \tag{5.96}$$

where $\Delta\mathbf{X}$ is the gradient calculated on the Lie-algebra space. Recall the form of the natural gradient presented in [16]:

$$\mathbf{W}_{new} = \mathbf{W}_{old} + \mu\Delta\mathbf{W} \tag{5.97}$$

where $\Delta\mathbf{W} = \frac{\partial L}{\partial \mathbf{W}}\mathbf{W}^T\mathbf{W}$ is what is referred to as the natural gradient and is calculated directly on the Riemannian manifold using the cost function $L(\mathbf{W})$ which is the maximum-likelihood cost function in this case.

A comparison of the above form and the form of the natural gradient in (5.96) can be done by recalling that the exponential of any general matrix $\mathbf{A}$ is given by the form:

$$\exp^{\mathbf{A}} = \mathbf{I} + \mathbf{A} + \mathbf{A}^2/2! \ldots . \tag{5.98}$$

for $\det|\mathbf{A}| < 1$. Thus:

$$\mathbf{W}_{new} = \exp^{(\mu\Delta\mathbf{X})}\mathbf{W}_{old} \approx \mathbf{W}_{old} + \mu\Delta\mathbf{X}\mathbf{W}_{old} \tag{5.99}$$

which is valid as long as $\mu \ll 1$. This approximation would provide a relationship between $\Delta\mathbf{W}$ and $\Delta\mathbf{X}$.

$$\Delta\mathbf{X} = \Delta\mathbf{W}\mathbf{W}^{-1} = (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T) \tag{5.100}$$

It should be mentioned here that equation (5.96) is considered the general form of the natural gradient update on the Riemannian manifold (geodesic search update). Thus, using more sophisticated approximations to calculate the matrix exponentiation can yield more accurate updates that are less prone to divergence updates compared to the update in equation (5.97). A relationship between $d\mathbf{X}$ and $d\mathbf{y}$ can be constructed as well:

$$d\mathbf{y} = d\mathbf{W}\mathbf{x} = d\mathbf{W}\mathbf{W}^{-1}\mathbf{y} = d\mathbf{X}\mathbf{y} \tag{5.101}$$

and thus:

$$\frac{d\mathbf{y}}{d\mathbf{X}} = \begin{pmatrix} \frac{dy_1}{d\mathbf{X}} \\ \frac{dy_2}{d\mathbf{X}} \\ \vdots \\ \frac{dy_N}{d\mathbf{X}} \end{pmatrix} \tag{5.102}$$

can be calculated as follows:

- Each element in the differentiation $\frac{dy_i}{d\mathbf{X}}, i = 1 \ldots N$ is a matrix of size $N \times N$ and has the following structure:

$$\frac{dy_i}{d\mathbf{X}} = \begin{pmatrix} \frac{dy_i}{dX_{11}} & \frac{dy_i}{dX_{12}} & \cdots & \frac{dy_i}{dX_{1N}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{dy_i}{dX_{i1}} & \frac{dy_i}{dX_{i2}} & \cdots & \frac{dy_i}{dX_{iN}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{dy_i}{dX_{N1}} & \frac{dy_i}{dX_{N2}} & \cdots & \frac{dy_i}{dX_{NN}} \end{pmatrix} \tag{5.103}$$

  Keeping in mind that the output y_i is given by the following equation:

$$y_i = \mathbf{X}_i^T\mathbf{y} = \begin{pmatrix} X_{i1} & X_{i2} & \ldots & X_{iN} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \tag{5.104}$$

Then all the rows in the differentiation matrix will vanish except for the row i which contains a row equal to $\mathbf{y}^T$. Equation 5.103 reduces to:

$$\frac{dy_i}{d\mathbf{X}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ y_1 & y_2 & \cdots & y_N \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \underline{0} \\ \vdots \\ \mathbf{y}^T \\ \underline{0} \\ \vdots \end{pmatrix} \tag{5.105}$$

This sparse matrix of size $N \times N$ can be re-arranged in a vector form by stacking the rows to form a vector of size $N^2 \times 1$. For example:

$$\text{vec}(\frac{d\mathbf{y}_1}{d\mathbf{X}}) = \begin{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}_{N\times 1} \\ \begin{pmatrix} \underline{0} \end{pmatrix}_{(N^2-N)\times 1} \end{pmatrix} \tag{5.106}$$

and for a general y_i the row-stacked differentiation has the following form:

$$\text{vec}(\frac{d\mathbf{y}_i}{d\mathbf{X}}) = \begin{pmatrix} \begin{pmatrix} \underline{0} \end{pmatrix}_{(i-1)N\times 1} \\ \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}_{N\times 1} \\ \begin{pmatrix} \underline{0} \end{pmatrix}_{(N-i)N\times 1} \end{pmatrix} \tag{5.107}$$

where the function vec(.) transforms a square matrix into a vector by row-stacking. This vector form of the gradient is identical to differentiating the terms $y_i, i = 1 \cdots N$ with respect to the vector vec($\mathbf{X}$) or equivalently $\mathbf{X}_{vec}$, which is a re-arrangement of the matrix $\mathbf{X}$ in a vector by row stacking such that:

$$\mathbf{X}_{vec} = \begin{pmatrix} X_{11} \\ X_{12} \\ \vdots \\ X_{1N} \\ \vdots \\ X_{N1} \\ X_{N2} \\ \vdots \\ X_{NN} \end{pmatrix} \tag{5.108}$$

This vector-form is much easier to use and is considered as a standard mathematical version for second-order differentiation given that for those higher order differentiation, the resulting matrix size grows exponentially.

- Another quantity of interest is the differentiation, $\frac{dy_i}{d\mathbf{X}_{vec}^T}$, which can be equivalently calculated using the fact that:

$$\frac{dy_i}{d\mathbf{X}_{vec}^T} = \begin{pmatrix} \frac{dy_i}{d\mathbf{X}_{11}} & \frac{dy_i}{d\mathbf{X}_{12}} & \cdots & \frac{dy_i}{d\mathbf{X}_{1N}} & \frac{dy_i}{d\mathbf{X}_{21}} & \cdots \end{pmatrix} \tag{5.109}$$

with:

$$\mathbf{X}_{vec}^T = \begin{pmatrix} \mathbf{X}_1^T & \mathbf{X}_2^T & \cdots & \mathbf{X}_N^T \end{pmatrix} \tag{5.110}$$

which is a row of size $1 \times N^2$ as $\mathbf{X}_i$ is the column vector transpose of the i^{th} row in the matrix $\mathbf{X}$.
The quantity $\frac{dy_i}{d\mathbf{X}_{vec}^T}$ thus has the following form:

$$\frac{dy_i}{d\mathbf{X}_{vec}^T} = \begin{pmatrix} \begin{pmatrix} \underline{0} \end{pmatrix}_{1\times(i-1)N} \begin{pmatrix} \mathbf{y}^T \end{pmatrix}_{1\times N} \begin{pmatrix} \underline{0} \end{pmatrix}_{1\times(N-i)N} \end{pmatrix} \tag{5.111}$$

which is a row of size $1 \times N^2$.

- At the steady state point (when separation is reached), the following two conditions hold:

  1. The output signals $\mathbf{y}$ are independent and thus are uncorrelated, meaning:

$$E\{\mathbf{y}\mathbf{y}^T\} = \mathbf{I} \tag{5.112}$$

  If one decides to relax the unit variance condition imposed on the outputs, the above autocorrelation matrix can be more accurately considered as:

$$E\{\mathbf{y}\mathbf{y}^T\} = \begin{pmatrix} E\{y_1\}^2 & 0 & \cdots & 0 \\ 0 & E\{y_2\}^2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & E\{y_N\}^2 \end{pmatrix} \tag{5.113}$$

  2. The decorrelation also means that $E\{\mathbf{y}g(\mathbf{y})^T\}$ is also a diagonal matrix:

$$E\{\mathbf{y}g(\mathbf{y})^T\} = \begin{pmatrix} E\{y_1 g(y_1)^T\} & 0 & \cdots & 0 \\ 0 & E\{y_2 g(y_2)^T\} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & E\{y_N g(y_N)^T\} \end{pmatrix} \tag{5.114}$$

  or approximately (when assuming the outputs are of unit variance):

$$E\{\mathbf{y}g(\mathbf{y})^T\} = \mathbf{I} \tag{5.115}$$

These two conditions will have special importance in simplifying the gradients of the cost function $L(\mathbf{W})$.

- Knowing that the first derivative $\frac{\partial L(\mathbf{W})}{\partial \mathbf{X}} = \nabla L(\mathbf{W}) = (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)$ the second derivative can be estimated by differentiating this expression with respect to $\mathbf{X}^T$.
- The differentiation will involve the second term $g(\mathbf{y})\mathbf{y}^T$. Before proceeding with the derivation of the second derivative, it would be useful to verify how this first differential, $g(\mathbf{y})\mathbf{y}^T$, is constructed.

The penalty term in the cost function $L(\mathbf{W})$ responsible for this term is $\sum_{i=1}^{N} E\{\log|p_i(y_i)|\}$, where $L(\mathbf{W})$ is again:

$$L(\mathbf{W}) = \log|\det \mathbf{W}| + \underbrace{\sum_{i=1}^{N} E\{\log|p_i(y_i)|\}} \tag{5.116}$$

The differentiation of $\sum_{i=1}^{N} E\{\log|p_i(y_i)|\}$ has the following form:

$$-\frac{\partial \sum_{i=1}^{N} E\{\log|p_i(y_i)|\}}{\partial \mathbf{X}} = -\frac{\partial E\{\log|p_1(y_1)|\}}{\partial \mathbf{X}} - \frac{\partial E\{\log|p_2(y_2)|\}}{\partial \mathbf{X}} \cdots$$

$$= E\{g(y_1)\frac{\partial y_1}{\partial \mathbf{X}} + g(y_2)\frac{\partial y_2}{\partial \mathbf{X}} + \cdots + g(y_N)\frac{\partial y_N}{\partial \mathbf{X}}\}$$

$$= E\{g(y_1)\begin{pmatrix} y_1 & \cdots & y_N \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} + g(y_2)\begin{pmatrix} 0 & \cdots & 0 \\ y_1 & \cdots & y_N \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} + \cdots + g(y_N)\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \\ y_1 & \cdots & y_N \end{pmatrix}\}$$

$$= E\{g(\mathbf{y})\mathbf{y}^T\}$$

The above differentiation is easier when considering the second differentiation applied to each element in the matrix $g(\mathbf{y})\mathbf{y}^T$. It is important to note the form of this first gradient at the optimal value of separation. Recalling the conditions of perfect separation, i.e. equations (5.113), (5.115), equation (5.9) at the optimal value of $\mathbf{W}$ reduces to the following form:

$$-\frac{\partial \sum_{i=1}^{N} E\{\log|p_i(y_i)|\}}{\partial \mathbf{X}}|_{\mathbf{W}_{opt}} = E\{\begin{pmatrix} g(y_1)y_1 & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & \cdots & 0 \\ 0 & g(y_2)y_2 & \vdots \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} \cdots + \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \\ 0 & \cdots & g(y_N)y_N \end{pmatrix}\} \tag{5.117}$$

$$= E\{\begin{pmatrix} g(y_1)y_1 & 0 & \cdots \\ 0 & \cdots & 0 \\ \vdots & g(y_i)y_i & \vdots \\ 0 & \cdots & g(y_N)y_N \end{pmatrix}\} \tag{5.118}$$

which is a diagonal matrix.

The meaning of this reduced form at the optimal value is that the update along the Lie algebra should be a diagonal matrix. Diagonal updates reduce to scalar multiplication when exponential mapping is considered. For example, for a 2×2 mixing matrix $\mathbf{W}$, at the optimal point of separation the update becomes:

$$\mathbf{W}(t+1) = \exp(\Delta\mathbf{X})\mathbf{W}(t) = \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix}\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} = \begin{pmatrix} \alpha_1\begin{pmatrix} W_{11} & W_{12} \end{pmatrix} \\ \alpha_2\begin{pmatrix} W_{21} & W_{22} \end{pmatrix} \end{pmatrix} \tag{5.119}$$

which obviously reduces to a scaling factor when calculating the output $\mathbf{y} = \mathbf{W}\mathbf{x}$.

The above diagonal form can also be viewed in terms of the relationship between the outputs $\mathbf{y}$ and the Lie-algebra matrix $\mathbf{X}$. The differentiation $\frac{\partial y_i}{\partial \mathbf{X}}$ at the optimal separation point follows the form:

$$\frac{\partial y_i}{\partial \mathbf{X}}|\mathbf{w}_{opt} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & y_i & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{5.120}$$

or in vector form:

$$\frac{\partial y_i}{\partial \mathbf{X}_{vec}}|\mathbf{w}_{opt} = \begin{pmatrix} 0 \\ \vdots \\ y_i \\ 0 \\ \vdots \end{pmatrix} \tag{5.121}$$

or when considered with respect to $\mathbf{X}_{vec}^T$:

$$\frac{\partial y_i}{\partial \mathbf{X}_{vec}^T}|\mathbf{w}_{opt} = \begin{pmatrix} 0 & \cdots & y_i & 0 & \cdots \end{pmatrix} \tag{5.122}$$

implies that the output y_i at the optimal separation point is totally uncorrelated from the other output signals and depends solely on y_i. Equations (5.120), (5.121) will help in simplifying the Hessian of the maximum-likelihood cost function as will follow in the discussion.

After obtaining the form of the gradient on the Riemannian space for the $GL-$ group, i.e. $\Delta\mathbf{X}$, its relationship with $\Delta\mathbf{W}$ and $\Delta\mathbf{y}$ and the condition of optimal separation, it is now

possible to extend this first-order optimization method to second order optimization by estimating the second gradient (Hessian) along the Lie-algebra space, i.e. $\frac{\partial^2 L(\mathbf{W})}{\partial \mathbf{X} \partial \mathbf{X}^T} = \nabla^2 L(\mathbf{W})$ as follows:
The differentiation will result in two terms:

1.
$$\frac{\partial g(\mathbf{y})}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{X}^T} \mathbf{y}^T \tag{5.123}$$
which will be referred to as "Hessian-1".

2.
$$\frac{\partial \mathbf{y}}{\partial \mathbf{X}^T} g(\mathbf{y})^T \tag{5.124}$$

which will be given the name "Hessian-2".

- The first term in the differentiation, i.e. "Hessian-1" has the following form:

$$\text{Hessian-1} = E\{ \begin{pmatrix} \frac{\partial g(y_1)}{\partial y_1} \frac{\partial y_1}{\partial \mathbf{X}_{vec}^T} y_1 \\ \frac{\partial g(y_1)}{\partial y_1} \frac{\partial y_1}{\partial \mathbf{X}_{vec}^T} y_2 \\ \vdots \\ \frac{\partial g(y_1)}{\partial y_1} \frac{\partial y_1}{\partial \mathbf{X}_{vec}^T} y_N \\ \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} \\ \underline{\mathbf{0}}_{(N^2-N)\times N^2} \end{pmatrix} + \begin{pmatrix} \begin{pmatrix} \\ \\ \\ \\ \end{pmatrix} \\ \mathbf{0}_{N\times N^2} \\ \frac{\partial g(y_2)}{\partial y_2} \frac{\partial y_2}{\partial \mathbf{X}_{vec}^T} y_1 \\ \frac{\partial g(y_2)}{\partial y_2} \frac{\partial y_2}{\partial \mathbf{X}_{vec}^T} y_2 \\ \vdots \\ \frac{\partial g(y_2)}{\partial y_2} \frac{\partial y_2}{\partial \mathbf{X}_{vec}^T} y_N \\ \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} \\ \mathbf{0}_{(N^2-2N)\times N^2} \end{pmatrix} + \cdots + \begin{pmatrix} \begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} \\ \mathbf{0}_{(N^2-N)\times N^2} \\ \frac{\partial g(y_N)}{\partial y_N} \frac{\partial y_N}{\partial \mathbf{X}_{vec}^T} y_1 \\ \vdots \\ \frac{\partial g(y_N)}{\partial y_N} \frac{\partial y_N}{\partial \mathbf{X}_{vec}^T} y_N \end{pmatrix} \} \tag{5.125}$$

substituting $\frac{\partial y_i}{\partial \mathbf{X}_{vec}^T}, i = 1 \cdots N$ from equation (5.111) and denoting $\frac{\partial g(y_i)}{\partial y_i}$ as $g'(y_i)$, the above equation turns out to have a special structure that is block diagonal with diagonal blocks $i = 1..N$ of size $N \times N$:

$$g'(y_i)\mathbf{y}\mathbf{y}^T \tag{5.126}$$

$$\text{Hessian-1} = E\{\begin{pmatrix} g'(y_1)\mathbf{y}\mathbf{y}^T & \underline{\mathbf{0}}_{N\times N} & \cdots & \cdots \\ \underline{\mathbf{0}}_{N\times N} & g'(y_2)\mathbf{y}\mathbf{y}^T & \underline{\mathbf{0}}_{N\times N} & \cdots \\ \vdots & \underline{\mathbf{0}}_{N\times N} & \ddots & \vdots \\ \vdots & \vdots & \vdots & g'(y_N)\mathbf{y}\mathbf{y}^T \end{pmatrix}\} \tag{5.127}$$

- The second term in the differentiation, i.e. "Hessian-2" has the following form:

$$\text{Hessian-2} = E\{\begin{pmatrix} g(y_1)\frac{\partial y_1}{\partial \mathbf{X}_{vec}^T} \\ g(y_1)\frac{\partial y_2}{\partial \mathbf{X}_{vec}^T} \\ \vdots \\ g(y_1)\frac{\partial y_N}{\partial \mathbf{X}_{vec}^T} \end{pmatrix}\begin{pmatrix} \\ \\ \\ \\ \\ \underline{\mathbf{0}}_{(N^2-N)\times N^2} \end{pmatrix} + \begin{pmatrix} \begin{pmatrix} \\ \\ \underline{\mathbf{0}}_{N\times N^2} \end{pmatrix} \\ g(y_2)\frac{\partial y_1}{\partial \mathbf{X}_{vec}^T} \\ g(y_2)\frac{\partial y_2}{\partial \mathbf{X}_{vec}^T} \\ \vdots \\ g(y_2)\frac{\partial y_N}{\partial \mathbf{X}_{vec}^T} \end{pmatrix}\begin{pmatrix} \\ \\ \\ \\ \underline{\mathbf{0}}_{(N^2-2N)\times N^2} \end{pmatrix} + \cdots + \begin{pmatrix} \begin{pmatrix} \\ \\ \\ \\ \underline{\mathbf{0}}_{(N^2-N)\times N^2} \end{pmatrix} \\ g(y_N)\frac{\partial y_1}{\partial \mathbf{X}_{vec}^T} \\ \vdots \\ g(y_N)\frac{\partial y_N}{\partial \mathbf{X}_{vec}^T} \end{pmatrix}\} \tag{5.128}$$

substituting $\frac{\partial y_i}{\partial \mathbf{X}_{vec}^T}, i = 1 \cdots N$ from equation (5.111), the above equation turns out to

have a very sparse structure to add to Hessian-1:

$$\text{Hessian-2} = E\{\begin{pmatrix} g(y_1)\begin{pmatrix}\mathbf{y}^T & & \underline{\mathbf{0}}_{1\times(N^2-N)}\end{pmatrix} \\ g(y_1)\begin{pmatrix}\underline{\mathbf{0}}_{1\times N} & \mathbf{y}^T & 0_{1\times(N^2-2N)}\end{pmatrix} \\ \vdots \\ g(y_1)\begin{pmatrix} & \underline{\mathbf{0}}_{1\times(N^2-N)} & \mathbf{y}^T\end{pmatrix} \\ g(y_2)\begin{pmatrix}\mathbf{y}^T & & \underline{\mathbf{0}}_{1\times(N^2-N)}\end{pmatrix} \\ g(y_2)\begin{pmatrix}\underline{\mathbf{0}}_{1\times N} & \mathbf{y}^T & 0_{1\times(N^2-2N)}\end{pmatrix} \\ \vdots \\ g(y_2)\begin{pmatrix} & \underline{\mathbf{0}}_{1\times(N^2-N)} & \mathbf{y}^T\end{pmatrix} \\ \vdots \\ g(y_N)\begin{pmatrix}\mathbf{y}^T & & \underline{\mathbf{0}}_{1\times(N^2-N)}\end{pmatrix} \\ \vdots \\ g(y_N)\begin{pmatrix} & \underline{\mathbf{0}}_{1\times(N^2-N)} & \mathbf{y}^T\end{pmatrix}\end{pmatrix}\} \tag{5.129}$$

Thus:

$$\text{Hessian} = \text{Hessian-1} + \text{Hessian-2} \tag{5.130}$$

This second order gradient, i.e. Hessian, can be used to implement a Newton-like update. However, the computational cost will be high. Roughly, the computational cost for calculating a Newton update step can be estimated as follows for a general $N \times N$ de-mixing system:

1. The term Hessian-1 will require N^3-order operations, i.e. multiplications. This computational load is estimated as follows:
   - The repeated diagonal blocks will require N^2 operations.
   - The multiplication of each of these blocks by $g'(y_i)$ will require another N operations for each block.
   - The overall computation will thus be $N \times N^2 = N^3$
2. Equivalently, the term Hessian-2 computational load can be calculated as follows:
   - For the $N \times N$ matrix $g(\mathbf{y})\mathbf{y}^T$ the required computation is N^2 computation, which is basically multiplying each element in $g(\mathbf{y})$ by an $1 \times N$ row $\mathbf{y}^T$.
   - for the term Hessian-2, however, each element in $g(\mathbf{y})$ is multiplied by $\mathbf{y}^T$, repeatedly N-times. This will make the computational load N^3.
   - However if the repeated operations are calculated once and then repeated in their respective locations in the Hessian-2 matrix, the operations will reduce to N^2. However, memory consumption will still be higher for the Hessian-2 matrix rather than the calculation of $g(\mathbf{y})\mathbf{y}^T$.

3. The main source of computational load is to calculate the inverse of the Hessian matrix. The least expensive approach is to make use of the sparse nature of the Hessian matrix and to apply one of the Cholessky-family sparse decomposition techniques [70]. However, the inverse estimation will still be in the order of N^6 for an $N^2 \times N^2$ matrix [85].

Thus the overall Newton step will be in the order or higher than than N^6 operations. This is a high computational load considering that the first gradient requires only N^2-order operations. One way to reduce the computational load is to apply realistic simplifications to the Hessian expression using equations (5.117), (5.118), (5.120), (5.121). In the following sections two types of reductions are applied, resulting in significant reduction in the computational cost while maintaining the performance that is noticeably better compared to the natural gradient algorithm.

5.10 Reduced-Hessian Online Algorithm

The first approximate algorithm that can be implemented instead of the Hessian-based algorithm replaces the first gradient $g(\mathbf{y})\mathbf{y}^T$ or equivalently $\text{vec}(g(\mathbf{y})\mathbf{y}^T)$ by its value at the optimal separation point, which is a diagonal matrix with diagonal elements equivalent to $\text{diag}(g(\mathbf{y})\mathbf{y}^T)$. The new approximate second gradient, i.e. Hessian, will be:

$$\text{Hessian}_{approx} = \frac{\partial \begin{pmatrix} g(y_1)y_1 \\ 0 \\ \vdots \\ g(y_2)y_2 \\ 0 \\ \vdots \\ g(y_i)y_i \\ 0 \\ \vdots \\ g(y_N)y_N \end{pmatrix}}{\partial \mathbf{X}_{vec}^T} \tag{5.131}$$

This approximate form of the Hessian can also be decomposed into two terms, $\text{Hessian}_{approx} - 1$ and $\text{Hessian}_{approx} - 2$ in analogy to Hessian-1 and Hessian-2 in the last section, where:

$$\text{Hessian}_{approx} - 1 = \sum_{i=1}^{N} \frac{\partial g(y_i)}{\partial \mathbf{X}_{vec}^T} y_i \tag{5.132}$$

meaning, the differentiation is applied to the nonlinearity term. The second term $\text{Hessian}_{approx} - 2$ is basically the result of differentiating y_i:

$$\text{Hessian}_{approx} - 2 = \sum_{i=1}^{N} g(y_i) \frac{\partial y_i}{\partial \mathbf{X}_{vec}^T} \tag{5.133}$$

The above two terms can be calculated as Hessian-1 and Hessian-2 in the previous sections. The final form of each will reduce to the same structure of Hessian-1 and Hessian-2 with the cancelation of terms that are already known to decay to zero in Hessian-1 and Hessian-2 as the update moves closer and closer to the separation point. $\text{Hessian}_{approx}-1$ matrix is also a block diagonal matrix. However, it will become very sparse with only one row in each block to have a non-zero value. Comparing to the form (5.123), $\text{Hessian}_{approx}-1$ reduces to:

$$\text{Hessian}_{approx}-1 = \begin{pmatrix} \begin{pmatrix} \frac{\partial g(y_1)}{\partial y_1}\frac{\partial y_1}{\partial \mathbf{X}_{vec}^T}y_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ \begin{pmatrix} \\ \\ \underline{0}_{(N^2-N)\times N^2} \\ \\ \\ \end{pmatrix} \end{pmatrix} + \begin{pmatrix} \begin{pmatrix} \underline{0}_{N\times N^2} \\ \\ \end{pmatrix} \\ 0 \\ \frac{\partial g(y_2)}{\partial y_2}\frac{\partial y_2}{\partial \mathbf{X}_{vec}^T}y_2 \\ 0 \\ \vdots \\ \begin{pmatrix} \\ \underline{0}_{(N^2-2N)\times N^2} \\ \\ \end{pmatrix} \end{pmatrix} + \cdots + \begin{pmatrix} \begin{pmatrix} \\ \\ \underline{0}_{(N^2-N)\times N^2} \\ \\ \\ \end{pmatrix} \\ 0 \\ \vdots \\ \frac{\partial g(y_N)}{\partial y_N}\frac{\partial y_N}{\partial \mathbf{X}_{vec}^T}y_N \end{pmatrix} \tag{5.134}$$

The diagonal blocks will thus be of the form:

$$g'(y_i)y_i\mathbf{y}^T \tag{5.135}$$

or equivalently:

$$\text{Hessian}_{approx}-1 = E\{\begin{pmatrix} g'(y_1)y_1\begin{pmatrix}\mathbf{y}^T\\ \underline{0}_{1\times N}\\ \vdots \\ \underline{0}_{1\times N}\end{pmatrix} & \underline{0}_{N\times N} & \cdots & \cdots \\ \underline{0}_{N\times N} & g'(y_2)y_2\begin{pmatrix}\underline{0}_{1\times N}\\ \mathbf{y}^T\\ \underline{0}_{N\times N}\\ \vdots\end{pmatrix} & \underline{0}_{1\times N} & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \underline{0}_{N\times N} & \cdots & \cdots & g'(y_N)y_N\begin{pmatrix}\underline{0}_{1\times N}\\ \vdots \\ \underline{0}_{1\times N}\\ \mathbf{y}^T\end{pmatrix} \end{pmatrix}\} \tag{5.136}$$

The second differentiation term, i.e. $\text{Hessian}_{approx}-2$ will also result in a reduced form of the same structure as Hessian-2. However, the important conclusion here is that the new rows in $\text{Hessian}_{approx}-2$ are all aligned along the diagonal, which is the case in Hessian-2, and fill the same positions filled by the rows of the corresponding $\text{Hessian}_{approx}-1$, such that the final Hessian_{approx} will have the same structure as $\text{Hessian}_{approx}-2$ and $\text{Hessian}_{approx}-1$, which

is rows aligned along the diagonal of the $N^2 \times N^2$ matrix. The form of $\text{Hessian}_{approx} - 2$ is as follows:

$$\text{Hessian}_{approx} - 2 = E\{\begin{pmatrix} g(y_1)\begin{pmatrix} \mathbf{y}^T \\ \underline{0}_{1\times N} \\ \vdots \\ \underline{0}_{1\times N} \end{pmatrix} & \underline{0}_{N\times N} & \cdots & \cdots \\ \underline{0}_{N\times N} & g(y_2)\begin{pmatrix} \underline{0}_{1\times N} \\ \mathbf{y}^T \\ \underline{0}_{1\times N} \\ \vdots \end{pmatrix} & \underline{0}_{N\times N} & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \underline{0}_{N\times N} & \cdots & \cdots & g(y_N)\begin{pmatrix} \underline{0}_{1\times N} \\ \vdots \\ \underline{0}_{1\times N} \\ \mathbf{y}^T \end{pmatrix} \end{pmatrix}\} \quad (5.137)$$

Such that:

$$\text{Hessian}_{approx} = \text{Hessian}_{approx} - 1 + \text{Hessian}_{approx} - 2 \quad (5.138)$$

and has the following structure:

$$\text{Hessian}_{approx} : \begin{pmatrix} \begin{pmatrix} & \end{pmatrix}_{1\times N} & 0 & \cdots \\ 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \begin{pmatrix} & \end{pmatrix}_{1\times N} & \cdots \\ 0 & \cdots & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & \cdots & \begin{pmatrix} & \end{pmatrix}_{1\times N} \end{pmatrix} \quad (5.139)$$

It can be seen that the matrix Hessian_{approx} is very sparse. It can actually be re-arranged in a compact $N \times N$ matrix without any loss of information as it contains only N^2 elements and the non-zero elements in $\text{Hessian}_{approx} - 1$ and $\text{Hessian}_{approx} - 2$ are aligned in the same locations. In the same way a vector $N^2 \times 1$ is formed from an $N \times N$ matrix, the non-zero rows of Hessian_{approx} can be re-arranged along the columns of a new $N \times N$ matrix. The new reduced size matrix will be called Hess_{red}, which is composed of the two terms $\text{Hess}_{red} - 1$ and $\text{Hess}_{red} - 2$. The term $\text{Hess}_{red} - 1$ is the reduced matrix $N \times N$ of the term $\text{Hessian}_{approx} - 1$ written in compact form as:

$$\text{Hess}_{red} - 1 = E\{\mathbf{y}\mathbf{y}^T \text{diag}(g'(\mathbf{y}))\} \quad (5.140)$$

where $\text{diag}(g'(\mathbf{y}))$ is the diagonal matrix of size $N \times N$ whose diagonal elements are $g'(y_i), i =$

$1 \cdots N$

$$\text{diag}(g'(\mathbf{y})) = \begin{pmatrix} g'(y_1) & 0 & \cdots & \cdots \\ 0 & g'(y_2) & 0 & \cdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & g'(y_N) \end{pmatrix} \tag{5.141}$$

while the term $\text{Hess}_{red} - 2$ is the reduced matrix $N \times N$ of the term $\text{Hessian}_{approx} - 2$ written in compact form as:

$$\text{Hess}_{red} - 2 = E\{\mathbf{y}g(\mathbf{y})^T\} \tag{5.142}$$

Thus, the final form for Hess_{red} is an $N \times N$ matrix following the expression:

$$\text{Hess}_{red} = E\{\mathbf{y}\mathbf{y}^T \text{diag}(g'(\mathbf{y}))\} + E\{\mathbf{y}g(\mathbf{y})^T\} \tag{5.143}$$

At the optimal separation point $\mathbf{W}_{opt}$ or equivalently $\mathbf{X}_{opt}$, reviewing equation (5.113), it is clear that the expected value $E\{\mathbf{y}g(\mathbf{y})^T = I\}|_{\mathbf{W}_{opt}}$ and thus the first term $\text{Hess}_{red} - 1$ will reduce to the diagonal term $g'(\mathbf{y})$. Similarly, for the second-term, equation (5.114) can be used to reduce $\text{Hess}_{red} - 2$ to another diagonal matrix of the form:

$$\text{Hess}_{red} - 2|_{\mathbf{W}_{opt}} = \begin{pmatrix} E\{g(y_1)\}E\{y_1\} & 0 & \cdots & 0 \\ 0 & E\{g(y_2)\}E\{y_2\} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & E\{g(y_N)\}E\{y_N\} \end{pmatrix} \tag{5.144}$$

Thus, the matrix Hess_{red} will reduce to a diagonal matrix which is necessary so that the optimal de-mixing point does not change at convergence as has been previously discussed. The above reduced form is then used to provide the following update:

$$\Delta\mathbf{X} = Hess_{red}^{-1} E\{(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\} \tag{5.145}$$

$$\mathbf{W}_{new} = \exp(\Delta\mathbf{X})\mathbf{W}_{old} \tag{5.146}$$

The above update can be performed on-line by replacing the expected value of the first gradient $E\{(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\}$ by its instantaneous value replacing the expected value of the reduced Hessian, Hess_{red}, by a running estimate of the expected value updated from time instant $(t-1)$ to time instant t using an exponential window with a forgetting factor λ as:

$$\text{Hess}_{red}(t) = \lambda \text{Hess}_{red}(t-1) + \mathbf{y}\mathbf{y}^T \text{diag}(g'(\mathbf{y})) + \mathbf{y}g(\mathbf{y})^T \tag{5.147}$$

An extra simplification can be achieved by applying the matrix inversion lemma to the above sequential update of the expected value of the Hessian to calculate the inverse of this expected value at each time instant based on the previous inverse calculated at the past instant.

After obtaining the inverse of the reduced Hessian an update of the demixing matrix can be carried out as follows:

$$\Delta\mathbf{X} = \text{Hess}_{red}^{-1} (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T) \tag{5.148}$$

$$\mathbf{W} = \exp(\mu\Delta\mathbf{X})\mathbf{W} \approx \quad \mathbf{W} + \mu\Delta\mathbf{X}\mathbf{W} \tag{5.149}$$

Thus the update can be written as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu \mathrm{Hess}_{red}^{-1}(t)(\mathbf{I} - g(\mathbf{y}(t))\mathbf{y}^T(t))\mathbf{W}(t-1) \tag{5.150}$$

This reduced Hessian algorithm works very well in practice and provides the best convergence speed when applied to test data. It can successfully handle the difficult situation of acoustic mixtures characterized with heavy tail distributions and inherent high autocorrelation (i.e. speech and music data). This will be confirmed by simulations shown afterwards in this Chapter and in Chapter 6. Several forms of the above algorithm can be obtained by applying the ideas of the iterative inversion Wiener approach developed in section 5.4.3 . These iterative approaches yielded variant algorithms that still belong in form to the natural-gradient family.
Some of these algorithms have the form:

$$\mathbf{W} = \mathbf{W} + \mu(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{R}^{-1}\mathbf{W} \tag{5.151}$$

Several forms for the matrix $\mathbf{R}$ have been presented and the simulation results presented along with the algorithms proved that this form of algorithms exhibit RLS-like, i.e. second order, fast convergence speed.
[39] provided ideas about various forms of natural gradients. Among those concepts the following theorem mathematically proves that the above form of the natural gradient converges to the same solution as the original natural gradient.

Theorem 5.10.1. *Consider a dynamic system described by the following differential equation:*

$$\frac{d\mathbf{W}}{dt} = -\eta(\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}})F(\mathbf{W})^T F(\mathbf{W}) \tag{5.152}$$

where $\mathbf{W} \in \mathbf{R}^{N\times N}$ *is a square matrix function of* t*, defined on the space* $\mathbf{R}^{N\times N}$*,* $L : \mathbf{R}^{N\times N} \rightarrow \mathbf{R}$ *is a differentiable function with matrix argument,* $F : \mathbf{R}^{N\times N} \rightarrow \mathbf{R}^{N\times N}$ *is an arbitrary matrix-valued function with matrix arguments and nonsingular values, and* $\eta \in \mathbf{R}^{N\times N}$ *is a symmetric positive definite matrix (possibly a function of* t*). Then* L *is a Lyapunov function* [4] *(in a wide sense) for the dynamic system.*

What the above theorem means is that a more broad family of natural gradients can be generalized by replacing the term $\mathbf{W}^T\mathbf{W}$ by a general function $\mathbf{F}^T(\mathbf{W})\mathbf{F}(\mathbf{W})$. This function can be another linear transformation applied on $\mathbf{W}$, like a multiplication by another matrix $\mathbf{D}^{1/2}$, which is positive definite, the form of the resulting natural-gradient family is then:

$$\mathbf{W}(t) = \exp(\mu(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{D})\mathbf{W}(t-1) \quad \text{or} \quad \mathbf{W} = \mathbf{W} + \mu(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{D}\mathbf{W} \tag{5.153}$$

[4] *A Lyapunov function is a scalar function* $V(y)$ *defined on a region* D *that is continuous, positive definite, and has continuous first-order partial derivatives at every point of* D*. Defining a cost function to be Lyapunov guarantees the stability of the zero solution ,i.e optimum solution obtained by adapting the search direction based on the derivative of the cost function.*

The above form, equation (5.153), will be called from henceforth as the "Generalized Median Gradient" (GMG) algorithm. The choice to call the above type of gradient "*median*" is based on the location of the transform function $\mathbf{D}$ which is placed between $\mathbf{W}$ and $\mathbf{W}^T$ while "General" is chosen to indicate that this algorithm is employing a general positive definite matrix $\mathbf{D}$ with a general form compared to a more specific median gradient algorithm presented in [41] under the name "Stochastic Median Gradient" (SMG), where the transform matrix $\mathbf{D}$ is a diagonal matrix of the form:

$$\mathbf{D} = \text{diag}(|y_1|^{-1}, |y_2|^{-1}, \ldots, |y_N|^{-1}) \tag{5.154}$$

which transforms the natural update rule to:

$$\mathbf{W} = \mathbf{W} + \mu(\mathbf{D} - g(\mathbf{y})\mathbf{W}\text{sign}(\mathbf{y})^T) \tag{5.155}$$

This SMG update rule is reported to have a more robust performance compared to the natural gradient in the presence of noise [41], [39].
The proposed general median gradient algorithm can thus provide a bigger family of natural gradient algorithms that extends beyond the SMG algorithm and can provide a new variant to the reduced-Hessian on-line update rule by choosing Hess_{red}^{-T} as the transform matrix $\mathbf{D}$. The resulting algorithm is referred to as the Median-Relative Hessian algorithm and is identical to the Relative Hessian algorithm except for the final update equation applied on the de-mixing matrix $\mathbf{W}$, i.e. equation (5.150) which takes the form:

$$\Delta\mathbf{X} = (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\text{Hess}_{red}^{-T} \tag{5.156}$$

$$\mathbf{W} = \exp(\mu\Delta\mathbf{X})\mathbf{W} \approx \mathbf{W} + \mu\Delta\mathbf{X}\mathbf{W} \approx \mathbf{W} + \mu(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\text{Hess}_{red}^{-T}\mathbf{W} \tag{5.157}$$

The median version of the Reduced Hessian algorithm provides a slightly faster convergence and more robust performance than the Reduced-Hessian algorithm.
Considering the computational load, it is easy to see that the reduction of the size of the Hessian form used has a significant effect on the computational load. Most notably in the calculation of the inverse, where the required computation will be of the order N^3 instead of N^6. This change alone will make the whole algorithm significantly simpler. Roughly the computations required are :

1. For the first term $\text{Hess}_{red} - 1$, the calculation $\mathbf{y}\mathbf{y}^T$ will be of $N^2/2$ operations due to symmetry and the multiplication of the diagonal matrix $diag(g'(\mathbf{y}))$ will add another N operations.

2. For the first term $\text{Hess}_{red} - 1$, the calculation $\mathbf{y}\mathbf{y}^T$ will be of $N^2/2$ operations.

3. The addition of the two terms is in the order of N^2 operations, assuming that the addition operation has the same load as the multiplication, which is a reasonable assumption for recent processors.

4. The inversion process is in the order of N^3 operations, [85].

Mainly, the inversion of the Hessian matrix controls the computational order of the second-order based algorithm. Thus, the whole algorithm is in the order of N^3 operation, which is a significant improvement over the use of the actual Hessian N^6-based algorithm.
In the following section, more simplifications of the Hessian matrix are considered to develop an even simpler algorithm than the approximate-Hessian developed in this section. The key to the simplicity of the following algorithm, the "Diagonal-Hessian", is that it does not require matrix inversion and it employs fewer terms.

5.11 Diagonal-Hessian Online Algorithm

Consider the form of the first term in the original Hessian matrix, Hessian-1:

$$\text{Hessian}-1=\begin{pmatrix} g'(y_1)\mathbf{y}\mathbf{y}^T & \underline{0}_{N\times N} & \cdots & \cdots \\ \underline{0}_{N\times N} & g'(y_2)\mathbf{y}\mathbf{y}^T & \underline{0}_{N\times N} & \cdots \\ \vdots & \underline{0}_{N\times N} & \ddots & \vdots \\ \vdots & \vdots & \vdots & g'(y_N)\mathbf{y}\mathbf{y}^T \end{pmatrix} \tag{5.158}$$

In order to produce an algorithm that is capable of working on-line with good stability, a valid approximation that can be applied to the above form is applying equations (5.113) or (5.114), with assumptions which are valid at the convergence point:

$$\mathbf{y}\mathbf{y}^T=\begin{pmatrix} y_1^2 & \cdots & \cdots & 0 \\ 0 & y_2^2 & \cdots & 0 \\ \vdots & 0 & y_i^2 & \vdots \\ 0 & 0 & \cdots & y_N^2 \end{pmatrix} \tag{5.159}$$

Substituting equation(5.159) in the first term of the Hessian matrix given by equation (5.126) results in:

$$\begin{pmatrix} g'(y_1)\mathbf{y}\mathbf{y}^T & \cdots & \underline{0} \\ \vdots & \ddots & \vdots \\ \underline{0} & \cdots & g'(y_N)\mathbf{y}\mathbf{y}^T \end{pmatrix} \approx \begin{pmatrix} g'(y_1)\begin{pmatrix} y_1^2 & \cdots & \cdots & 0 \\ 0 & y_2^2 & \cdots & 0 \\ \vdots & 0 & y_i^2 & \vdots \\ 0 & 0 & \cdots & y_N^2 \end{pmatrix} & \underline{0} & \cdots & \\ & \ddots & & \\ \underline{0} & \cdots & g'(y_N)\begin{pmatrix} y_1^2 & \cdots & \cdots & 0 \\ 0 & y_2^2 & \cdots & 0 \\ \vdots & 0 & y_i^2 & \vdots \\ 0 & 0 & \cdots & y_N^2 \end{pmatrix} \end{pmatrix} \tag{5.160}$$

The above approximation clearly reduces the first term in the Hessian from a block diagonal matrix to a diagonal matrix, which is given the name Hess-diag1 to emphasize the fact that the approximation is a diagonal matrix.

The second term, Hessian-2, can also be approximated at the separation point in two steps.

1. First, the term $\frac{\partial y_i}{\partial \mathbf{X}_{vec}^T}$ can be substituted with its value at the optimal point of separation as expressed in equation (5.122):

$$\frac{\partial y_i}{\partial \mathbf{X}_{vec}^T} = \begin{pmatrix} 0 & \cdots & y_i & 0 & \cdots \end{pmatrix} \tag{5.161}$$

Thus the term Hessian-2 will reduce to:

$$\text{Hessian-2} \approx \begin{pmatrix} g(y_1)\begin{pmatrix} y_1 & 0 & \cdots & \cdots \end{pmatrix} \\ g(y_1)\begin{pmatrix} 0 & y_2 & 0 & \cdots \end{pmatrix} \\ \vdots \\ g(y_1)\begin{pmatrix} 0 & \cdots & \cdots & y_N \end{pmatrix} \\ g(y_2)\begin{pmatrix} y_1 & 0 & \cdots & \cdots \end{pmatrix} \\ g(y_2)\begin{pmatrix} 0 & y_2 & 0 & \cdots \end{pmatrix} \\ \vdots \\ g(y_2)\begin{pmatrix} 0 & \cdots & \cdots & y_N \end{pmatrix} \\ \vdots \\ g(y_N)\begin{pmatrix} y_1 & 0 & \cdots & \cdots \end{pmatrix} \\ \vdots \\ g(y_N)\begin{pmatrix} 0 & \cdots & \cdots & y_N \end{pmatrix} \end{pmatrix} \tag{5.162}$$

which is a very sparse matrix with only one non-zero element in each row. However, these elements are not aligned along the diagonal of the matrix.

2. Another equation that will help further reduce the term Hessian-2 is the fact that at the separation only $E\{g(y_i)y_i\}$ exists while all $E\{g(y_i)y_j\}, i \neq j$ decay to zero. Applying this fact to the above equation, the matrix becomes even more sparse containing only N elements aligned along the diagonal:

$$\text{Hessian-2-diag} = \begin{pmatrix} \begin{pmatrix} g(y_1)y_1 & 0 & \cdots \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} & \underline{0} & & \cdots \\ & \ddots & & \\ & & & \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ \vdots & \cdots & g(y_N)y_N \end{pmatrix} \\ & \underline{0} & \cdots & \end{pmatrix} \tag{5.163}$$

which will be called Hess-diag2 in analogy to the first term Hess-diag1. The above approximation, which reduces the second term in the Hessian matrix from a mere sparse matrix to a sparse diagonal matrix is a crucial simplification, especially knowing that the first term reduces to diagonal form as well. This means that the whole Hessian matrix is reduced to a sparse diagonal of $N^2 \times N^2$ which opens the door for a variety of simplified algorithms for the update.
The diagonal Hessian term, named Hess-diag can thus be written as:

$$\text{Hess-diag} = \text{Hess-diag1} + \text{Hess-diag2} \tag{5.164}$$

The next step is to make use of the diagonal nature of the approximation Hess-diag to implement a simpler inversion and update.
It is well known that the inverse of any general diagonal matrix $\mathbf{D}_{N\times N}$ is given by the following equation:

$$D^{-1} = \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \vdots & 0 \\ 0 & 0 & \dots & d_N \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{d_1} & 0 & \dots & 0 \\ 0 & \frac{1}{d_2} & \dots & 0 \\ \vdots & \vdots & \vdots & 0 \\ 0 & 0 & \dots & \frac{1}{d_N} \end{pmatrix} \tag{5.165}$$

This property facilitates the calculation of the inverse of the matrix. This inverse of the Hess-diag matrix is then multiplied by the vector $(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)_{vec}$ of size $N^2 \times 1$. Thus the basic Newton step can be described by the following set of equations:

$$\Delta\mathbf{X} = (\text{Hess-diag})^{-1}(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)_{vec} \tag{5.166}$$

$$\Delta\mathbf{X} = \begin{pmatrix} \frac{1}{Hess-diag_{11}} & 0 & \dots & 0 \\ 0 & \frac{1}{Hess-diag_{22}} & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & \dots & \frac{1}{Hess-diag_{N^2N^2}} \end{pmatrix} (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)_{vec} \tag{5.167}$$

$$\mathbf{W} = \exp(\mu\Delta\mathbf{X})\mathbf{W} \approx \mathbf{W} + \mu\Delta\mathbf{X}\mathbf{W} \tag{5.168}$$

Another simplification to the above algorithm is to re-write the inverse of the $N^2 \times N^2$ sparse diagonal matrix Hess-diag in a more compact form by row stacking the diagonal elements to an $N \times N$ matrix $\mathbf{H}$. The new update $\Delta\mathbf{X}$ can then be calculated by element-by-element multiplication between the matrix $\mathbf{H}$ and the first gradient $(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)$ using the operator $\odot$:

$$\begin{pmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NN} \end{pmatrix} \odot \begin{pmatrix} b_{11} & \dots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{N1} & \dots & b_{NN} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & \dots & a_{1N}b_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1}b_{N1} & \dots & a_{N1}b_{NN} \end{pmatrix} \tag{5.169}$$

This reduction in the size of Hess-diag^{-1} of size $N^2 \times N^2 \rightarrow \mathbf{H}$ of size $N \times N$ will allow the algorithm to get rid of the *vec* operator applied to the first gradient $(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)$ and work in

the domain $N \times N$ of the separating matrix $\mathbf{W}$. The resulting Newton-step based algorithm that uses this reduced space is given the name "Diagonal Hessian Online" algorithm and can be summarized in the following steps:

1. Calculate the diagonal elements of the matrix Hess-diag as given by equations (5.160), (5.163), (5.164).

2. The inverse of the diagonal matrix is given by the inverse of each element along the diagonal:

$$\begin{pmatrix} \frac{1}{Hess_{11}} & 0 & \dots & 0 \\ 0 & \frac{1}{Hess_{22}} & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & \dots & \frac{1}{Hess_{N^2N^2}} \end{pmatrix} \tag{5.170}$$

3. The inverse of the Hessian is stacked row-wise in a matrix $\mathbf{H}$ of size $N \times N$:

$$\mathbf{H} = \begin{pmatrix} \frac{1}{Hess_{11}} & \frac{1}{Hess_{22}} & \dots \\ \vdots & \ddots & \ddots \\ \dots & \dots & \frac{1}{Hess_{N^2N^2}} \end{pmatrix} \tag{5.171}$$

4. The new update $\Delta\mathbf{X}$ is calculated as:

$$\Delta\mathbf{X} = \mathbf{H} \odot (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T) \tag{5.172}$$

5. The new value of the de-mixing matrix can be evaluated by projecting the update $\Delta\mathbf{X}$ to the space of $\mathbf{W}$ as follows:

$$\mathbf{W} \quad = \quad \exp(\mu\Delta\mathbf{X})\mathbf{W} \qquad \approx \quad \mathbf{W} + \mu\Delta\mathbf{X}\mathbf{W} \tag{5.173}$$

The algorithm works very well in practice and is capable of separating highly correlated speech signals with a faster convergence speed and a smaller steady state error compared to the first order-based natural gradient.
Another advantage of the algorithm is that it requires only the inversion of the elements of the Hess-diag matrix, which are N^2 elements and thus the computational load of the matrix inversion is much less than the inversion of the actual Hessian of size $N^2 \times N^2$ or even the approximate Hessian, Hess_{red}, used in the previous section to implement the approximate-Hessian update algorithm.
One other simplification is that the multiplication of the inverse of the Hessian matrix by the first gradient is reduced to element by element multiplication. This reduces the order of the computational load of the update step to the order of N^2, which is the same order of computations required to calculate the natural gradient.

5.12 Simulation Results for Second Order Algorithms

This section presents a sample result of the second-order algorithms developed in sections 5.10 and 5.11, namely:

1. **Diagonal Hessian algorithm**: Employs a reduced $N \times N$ $\mathbf{H}$ matrix representing the inverse of an approximated diagonal $N^2 \times N^2$ Hessian matrix $Hess$. The matrix $\mathbf{H}$, formed by row stacking of the diagonal elements of Hess, presents a compact, easy to apply inverse that can help estimate the Newton step with reduced memory and cost by following the update equations, (5.172),(5.173):

$$\Delta \mathbf{X} = \mathbf{H} \odot (\mathbf{I} - g(\mathbf{y})\mathbf{y}^T) \tag{5.174}$$

$$\mathbf{W} = \exp(\mu \Delta \mathbf{X})\mathbf{W} \approx \mathbf{W} + \mu \Delta \mathbf{X}\mathbf{W} \tag{5.175}$$

2. **Reduced Hessian algorithm**: The approximated Hessian Hess_{red} is a reduced $N \times N$ expression of the Hessian matrix. The Newton step in this algorithm is estimated by the multiplication of $Hess_{red}^{-1}$ by the first gradient as in equations (5.150):

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu \text{Hess}_{red}^{-1}(t)(\mathbf{I} - g(\mathbf{y}(t))\mathbf{y}^T(t))\mathbf{W}(t-1) \tag{5.176}$$

3. The **Median-Reduced Hessian algorithm**: This algorithm employs both the Relative Hessian update rule and the General Median Gradient rule, which is theoretically valid, to provide a variant algorithm with slightly different performance than the Relative gradient. The calculations and procedure of the algorithm are identical to that of the Relative Hessian except for the update step, which is represented as:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - g(\mathbf{y}(t))\mathbf{y}^T(t))\text{Hess}_{red}^{-T}(t)\mathbf{W}(t-1) \tag{5.177}$$

The performances of the above algorithms are compared to the reference natural gradient algorithm given by the equation:

$$\mathbf{W} = \mathbf{W} + \mu_{NAG}(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{W} \tag{5.178}$$

The test data are the same 4 audio files sampled at 8 kHz and of duration 6.5 sec, that have been used in the simulations of the nonlinear RLS algorithm in section 5.8 and are presented in more details in Chapter 6:

1. Male-1:[48]. This file is a male speaker reading a sentence in English.
2. Male-2: [73]. It is also for a male speaker with similar voice properties to the above speaker.
3. Female-1: This file is obtained from [48]. However, the speaker is a female reading a third sentence in English.
4. Music-1: This file contains symphony music. It is obtained from [48].

The mixing matrix for the following figure is:

$$\mathbf{A} = \begin{pmatrix} 0.9 & 0.9 & 0.8 & 0.5 \\ 0.9 & 1.2 & 1.3 & 0.99 \\ 0.6 & 0.8 & 0.8 & 0.84 \\ 0.25 & 0.32 & 0.5 & 0.41 \end{pmatrix}$$

More simulations are presented in Chapter 6. However, one can consider this sample simula-

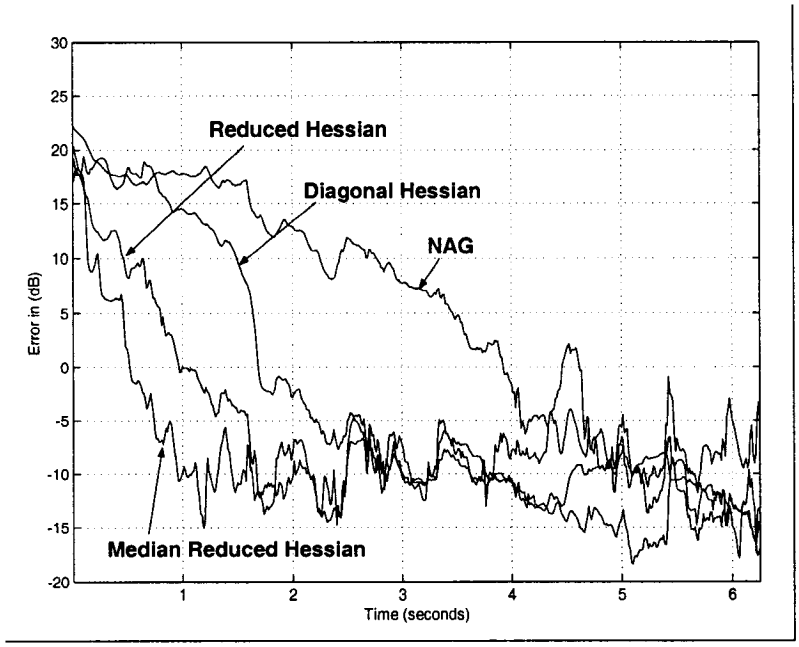

Figure 5.4: **Results for a sample run of the Hessian-based algorithms**

tion as a valid example reflecting the majority of the simulations in the sense of comparing the three algorithms. The Median-Reduced Hessian algorithm can be considered the fastest converging algorithm. The Median-Reduced Hessian algorithm achieves this convergence with a complexity comparable to the Reduced-Hessian algorithm, which is not very far behind when convergence speed is considered. Having the Diagonal-Hessian algorithm convergence speed slightly slower than the two previous algorithms is compensated by a smaller computational load. On the other hand, the natural gradient algorithm is shown to provide a significantly slower convergence speed to reach the same (slightly worse) steady state error.

5.13 Application of Hessian-Based Algorithms to The Pre-Whitened Case

As can be seen in the above sections, moving to the second order-based algorithms and combining them with the use of Riemannian geometry payed off in developing faster converging algorithms, that, with some reasonable approximations, were not computationally

expensive to implement. Simulation results showed that these Hessian-based algorithms work well even with the nonstationary data, i.e. audio data like speech, that constituted a challenging mixing scenario to separate. For all of the above reasons, it would thus be both logical and interesting to investigate the possibility of applying these algorithms to the pre-whitened case. In section 5.8, the idea of nonlinear decorrelation was extended from the Stiefel-manifold pre-whitened mixtures case to the general Riemannian case, where pre-whitening is not considered. Here, the approach is quite the opposite, the extension is from the Riemannian space to the Stiefel-manifold space.
It would be interesting if one could do such an application without requiring the calculation of the Hessian of the Stiefel-manifold case, i.e investing most of the effort in establishing a more broad general relationship between the Riemannian-update and the Stiefel-manifold update that might even work for other update equations, not only the Newton-based. In the following, an investigation of such a relationship is considered.
It has been shown and proved in the literature [97], that in the case where pre-whitening is applied to the mixtures prior to the de-mixing process, the de-mixing matrix is optimized on a reduced subspace of the general linear group, i.e. Stiefel manifold.
In other words, the de-mixing matrix is constrained to be an orthogonal matrix such that [31]:

$$\mathbf{W} = \exp(\mu \Delta \mathbf{X}_{so})\mathbf{W} = \exp(\mu(\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T))\mathbf{W} \tag{5.179}$$

Thus the derivative $\Delta \mathbf{X}_{so}$ along the Lie-algebra space *so* of the special orthogonal group *SO* [5] is given as a skew symmetric matrix. This property of the update $\Delta \mathbf{X}_{so}$ is the key for extending the two algorithms:

- Reduced Hessian algorithm
- Diagonal Hessian algorithm

from the formula for the natural gradient in the general case (GL-group):

$$\mathbf{W} = \exp(\mu \Delta \mathbf{X}_{GL})\mathbf{W} = \exp(\mu(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T)\mathbf{W} \tag{5.180}$$

It is easily seen that the two updates $\Delta \mathbf{X}_{so}$ and $\Delta \mathbf{X}_{GL}$ are related by the relation:

$$\Delta \mathbf{X}_{so} = \Delta \mathbf{X}_{NAG} - \Delta \mathbf{X}_{NAG}^T \tag{5.181}$$

This relationship is the basis for the extension of the above mentioned algorithms to the pre-whitened case, basically the extension is composed of the two steps:

1. Replace the first gradient in the update equation of the subject algorithm:

$$(\mathbf{I} - g(\mathbf{y})\mathbf{y}^T) \rightarrow (\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T) \tag{5.182}$$

2. Project the update step $\Delta \mathbf{X}$ calculated as in equation (5.172)(in the case of the Reduced Hessian algorithm) or equation (5.150)(in the case of the Relative Hessian algorithm) on the lie-algebra of the Stiefel manifold as follows:

$$\Delta \mathbf{X}_{so} = \Delta \mathbf{X} - \Delta \mathbf{X}^T \tag{5.183}$$

[5] a special case of the Stiefel manifold, i.e. when the orthogonal vectors are matrices of sizes $N \times N$.

Simulations applying the above two intuitive but rather heuristic steps proved these step to be successful in extending the fast converging algorithms to the pre-whitened mixtures case. Tables (5.2) and (5.3) provide summaries of the Stiefel extended algorithms. This section presents a sample result of the second-order algorithms extended to the pre-whitened case in the last section, namely:

1. **Diagonal Hessian-Stiefel Algorithm**: Which employs an $N \times N$ $\mathbf{H}$ matrix representing the inverse of an approximated diagonal $N^2 \times N^2$ Hessian matrix Hess according to the set of equations in Table 5.2.

2. **Reduced Hessian-Stiefel Algorithm**: The approximated Hessian Hess_{red} is a reduced $N \times N$ expression attaining the properties of the Hessian matrix. The algorithm is summarized in Table 5.3.

The algorithms are compared to the reference natural gradient algorithm on the Stiefel manifold given by the equation:

$$\mathbf{W} = \mathbf{W} + \mu_{NAG}(\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T)\mathbf{W} \tag{5.184}$$

The test data are the same 4 audio files sampled at 8 kHz and of duration 6.5 sec. that have been used in section 5.10 and 5.12:

1. Male-1: Obtained from [48]. This file is a male speaker reading a sentence in English.

2. Male-2: Obtained from [73]. It is for a different male speaker with voice properties similar to the 1^{st} speaker.

3. Female-1: This file is obtained from [48], for a female reading a third sentence in English.

4. Music-1: This file contains symphony music. It is obtained from [48].

For the results shown in the following figure, the mixing matrix is:

$$\mathbf{A} = \begin{pmatrix} 0.9 & 0.9 & 0.8 & 0.5 \\ 0.9 & 1.2 & 1.3 & 0.99 \\ 0.6 & 0.8 & 0.8 & 0.84 \\ 0.25 & 0.32 & 0.5 & 0.41 \end{pmatrix}$$

More simulations using this benchmark data and other benchmarks are presented in Chapter 6. However, figure (5.5) reflects the success of the application of the two algorithms to the pre-whitened case. Despite being rather intuitive, the newly developed algorithms provide, in most cases, noticeable improved convergence speed.
On the other hand, when reviewing the developed algorithms in Tables 5.1 and 5.2, it is easily seen that the approximated Hessian-matrices used in the algorithms are still the same ones previously developed for their respective general Riemannian counterpart. These Hessian matrices are the main computational load. Thus, it can be stated that these algorithms are of the same computational order than their general Riemannian counterparts, which are:

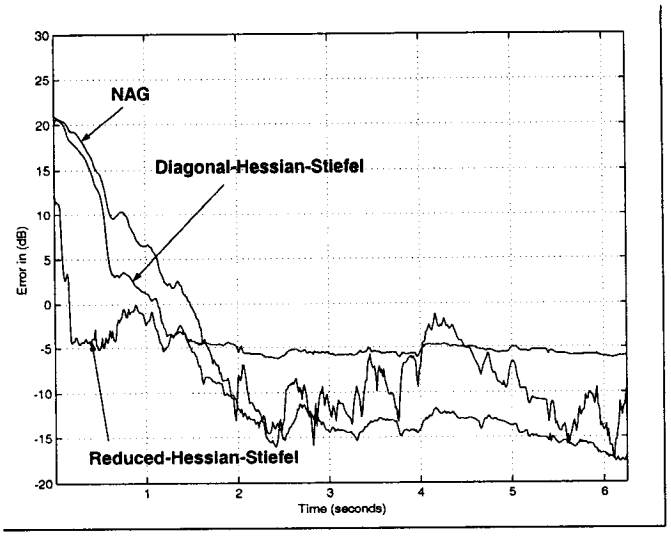

Figure 5.5: **Results for a sample run of the Hessian-based algorithms for Stiefel manifold**

- **Relative Hessian Stiefel**: is of order N^3
- **Diagonal Hessian Stiefel**: is of order N^2

5.14 Revisiting the Natural Gradient Form on the Riemannian Space

Throughout this chapter several variant forms were presented for natural-gradient like algorithms. Some of the variant forms were collectively presented in the iterative inversion family. Another variant form came from the Lyapunov theorem. The Nonholonomic form also provides another variant for natural gradient update equations.
It would be useful to present a global form for all variant forms with each variant presented as a special case of equation (5.152):

$$\frac{d\mathbf{W}}{dt} = -\eta(\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}})F(\mathbf{W})^T F(\mathbf{W}) \tag{5.185}$$

The above equation is the most general update form for LMS- Riemannian , i.e. natural gradient. The stability is insured by requiring F(.) to be nonsingular. Considering the following general form of update:

$$\mathbf{W} = \exp(\mu(\Theta - g(\mathbf{y})\mathbf{y}^T)\mathbf{D})\mathbf{W} \quad \text{or} \quad \mathbf{W} = \mathbf{W} + \mu(\Theta - g(\mathbf{y})\mathbf{y}^T)\mathbf{D}\mathbf{W} \tag{5.186}$$

Depending on the values of $\mathbf{D}$ and Θ a lot of the algorithms presented in this presentation can be related:

1. When $\mathbf{D} = \Theta = \mathbf{I}$, the resulting algorithm is the natural gradient:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - \mathbf{R}_{\mathbf{gy}}))\mathbf{W}(t-1) \tag{5.187}$$

2. When $\mathbf{D} = \mathbf{I}$ and $\mathbf{\Theta} = \mathbf{R_{ss}}$, the update produces the LMS-1 natural gradient update form:

$$\mathbf{W}(t) = \mathbf{W}(t) + \mu[\mathbf{R}_{ss} - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{5.188}$$

Depending on the choice of $\mathbf{R_{ss}}$, many update equations can be developed for the LMS-1 algorithm.

3. A special case of the LMS-1 algorithm is when $\mathbf{D} = \mathbf{I}$ and $\mathbf{\Theta} = \text{diag}(g(\mathbf{y})\mathbf{y}^T)$, the update produces the nonholonomic- natural gradient:

$$\mathbf{W}(t) = \mathbf{W}(t) + \mu[\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{5.189}$$

4. When $\mathbf{D} = \mathbf{R_{ss}}^{-1}$ and $\mathbf{\Theta} = \mathbf{R_{ss}}$ then the LMS-RLS algorithm is obtained:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{R_{ss}} - \mathbf{R_{gy}})\mathbf{R_{ss}}^{-1}\mathbf{W}(t-1) \tag{5.190}$$

Many update forms can be developed under the LMS-RLS form depending on the choice of $\mathbf{R_{ss}}$, as has been presented in section 5.5.3

5. The same conditions used in the previous LMS-RLS form, i.e. $\mathbf{D} = \mathbf{R_{ss}}^{-1}$ and $\mathbf{\Theta} = \mathbf{R_{ss}}$ can apply to the LMS-2 update:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu(\mathbf{I} - \mathbf{R_{gy}}\mathbf{R_{ss}}^{-1})\mathbf{W}(t-1) \tag{5.191}$$

Many special cases can be developed under the LMS-2 form depending on the choice of $\mathbf{R_{ss}}$ as has been presented in section 5.5.2.

6. When $\mathbf{\Theta} = \mathbf{I}$ and $\mathbf{D} = \text{Hess}_{red}$ then the Median-Relative Hessian is obtained:

7. When $\mathbf{\Theta} = \mathbf{I}$ and $\mathbf{D} = \text{diag}(|y_1|^{-1}, |y_2|^{-1}, \ldots, |y_N|^{-1})$ then the update is transformed to Stochastic Median-Gradient (SMG) :

$$\mathbf{W} = \mathbf{W} + \mu(\mathbf{D} - g(\mathbf{y})\text{sign}(\mathbf{y})^T) \tag{5.192}$$

5.15 Conclusion

This chapter presents a collection of new algorithms dedicated for the general mixing problem, which may be applicable if the separation system cannot afford or chooses not to perform pre-whitening. The cost function selected for developing these algorithms is the information-theoretic based maximum-likelihood, as the other information-theoretic based cost function, namely Negentropy, is inherently based on the assumption that the mixtures are pre-whitened.

The research in the chapter begins with investigating a more general formula for natural gradient. This general formula is then proven to be of direct relation to the autocorrelation of the estimated sources. Using this observation and combining it with the recently introduced concept of iterative inversion, the first group of algorithms is a family of extensions to the natural gradient approach, based on iterative inversion. The basis for this family has a very

similar form to the Wiener filtering. This family of algorithms can be viewed as a generalization of the natural gradient algorithm and can branch into many special cases of natural gradient, like the nonholonomic natural gradient update for example. These algorithms are developed without explicit use of Riemannian geometry, yet they provide the same structure of natural gradient update. Among these algorithm, the RLS-Iterative inversion algorithm exhibits improved performance both in the convergence speed and in the steady state error. Another algorithm is developed based on extending the successful concept of nonlinear-RLS, previously used on the Stiefel manifold in Chapter 4, to the general Riemannian space. This algorithm was able to deal with speech sources and successfully separate them with a relatively faster convergence than the natural gradient and thus proving the validity of the extension. However, the algorithm is high in complexity. This fact stimulated the search for improved algorithms in another direction, i.e. optimization over Riemannian geometry.
The second category of new algorithms is thus fundamentally based on the use of Riemannian geometry. These second-order Hessian based algorithms are based mainly on the identification of the first-order and second-order gradients along the Lie-algebra space of the general-linear group (the Riemannian manifold associated with invertible matrices). The identified Hessian is then simplified by using approximations to yield a diagonal matrix form that is easy to invert. The inverse of the Hessian is used to form a Newton-like update step that is capable, with adequate step size, of performing on-line updates. The resulting algorithm is named as "Reduced Hessian algorithm". Another variant of the algorithm is calculated and called "Diagonal Hessian algorithm". The improved performance of these two algorithms was validated through simulation.
These two algorithms were further extended to the Stiefel manifold. The extension was based on linking the type of update on the general linear group to the update on its constrained orthogonal subgroup (the Stiefel manifold group).
In the following table, a summary of the algorithms developed in this chapter is presented, highlighting their basic properties and their estimated approximate complexities.

Table 5.1: **Summary of algorithms developed for BSS on the Riemannian manifold**

Algorithm	**Classification**	**Application Data**	**Complexity**
LMS-1	Iterative Inversion	slow varying data	$O(N^2)$
LMS-2	Iterative Inversion	slow varying data	$O(N^2)/O(N^3)$
LMS-RLS	Iterative Inversion	slow varying data	$O(N^2)/O(N^3)$
RLS-Iterative	Iterative Inversion	slow varying data	$O(N^3)$
Nonlinear-RLS-Riemannian	Nonlinear-RLS	Audio data	$O(N^6)$
Reduced-Hessian	2^{nd}-order Riemannian optimization	Audio data	$O(N^3)$
Median-Reduced-Hessian	2^{nd}-order Riemannian optimization	Audio data	$O(N^3)$
Diagonal-Hessian	2^{nd}-order Riemannian optimization	Audio data	$O(N^2)$

Table 5.2: **Diagonal Hessian-Stiefel on-line algorithm for Stiefel manifold**

1. Compute the first gradient of the ML- function on the Stiefel manifold [90]:

$$\mathbf{G} = (\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T) \tag{5.193}$$

2. Calculate the Hess-diag matrix given by the two terms:

$$\text{Hess-diag1} = E\{\begin{pmatrix} g'(y_1)\begin{pmatrix} y_1^2 & 0 & \dots \\ & \ddots & \\ \vdots & \dots & y_N^2 \end{pmatrix} & \dots & \underline{0} \\ \vdots & \ddots & \vdots \\ \underline{0} & \dots & g'(y_N)\begin{pmatrix} y_1^2 & 0 & \dots \\ & \ddots & \\ \vdots & \dots & y_N^2 \end{pmatrix} \end{pmatrix}\} \tag{5.194}$$

   $E\{.\}$ can be replaced by a time-average, weighted sum through the use of the exponential window, with $\lambda < 1$ chosen close to 1

3.

$$\text{Hess-diag2} \approx \begin{pmatrix} \begin{pmatrix} 1 & 0 & \dots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \end{pmatrix} & \dots & \underline{0} \\ \vdots & \ddots & \vdots \\ \underline{0} & \dots & \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \dots & \vdots \\ 0 & \dots & 1 \end{pmatrix} \end{pmatrix} \tag{5.195}$$

$$\text{Hess-diag} = \text{Hess-diag1} + \text{Hess-diag2} \tag{5.196}$$

4. Re-arrange the inverse of the Hessian row-wise in a matrix $\mathbf{H}$ of size $N \times N$:

$$\mathbf{H} = \begin{pmatrix} \frac{1}{Hess_{11}} & \frac{1}{Hess_{22}} & \dots \\ \vdots & \ddots & \ddots \\ \dots & \dots & \frac{1}{Hess_{N^2N^2}} \end{pmatrix} \tag{5.197}$$

5. The new update $\Delta\mathbf{X}$ is calculated as:

$$\Delta\mathbf{X} = \mathbf{H} \odot (\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T) \tag{5.198}$$

6. The update $\Delta\mathbf{X}$ is projected to the lie algebra $so(n)$ by projecting the update $\Delta\mathbf{X}$ to the space of $\mathbf{W}$ as follows:

$$\Delta\mathbf{X}_{so} = \Delta\mathbf{X} - \Delta\mathbf{X}^T \tag{5.199}$$

7. The above update is then applied to the matrix $\mathbf{W}$ via exponential mapping to obtain the new $\mathbf{W}$

$$\mathbf{W} \quad = \quad \exp(\mu\Delta\mathbf{X}_{so})\mathbf{W} \qquad \approx \quad \mathbf{W} + \mu\Delta\mathbf{X}_{so}\mathbf{W} \tag{5.200}$$

Table 5.3: **Reduced Hessian-Stiefel on-line algorithm for Stiefel manifold**

1. Compute the first gradient of the maximum-likelihood on the Stiefel manifold:

$$\mathbf{G} = (\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T) \tag{5.201}$$

2. Calculate the Reduced Hessian Hess_{red} matrix of size $N \times N$,which may also be decomposed into two terms $\text{Hess}_{red} - 1$ and $\text{Hess}_{red} - 2$ as:

   - $$Hess_{red} - 1 = E\{\mathbf{y}\mathbf{y}^T \text{diag}(g(\mathbf{y})')\} \tag{5.202}$$

     where diag(.) transforms a vector of size $N \times 1$ to a diagonal matrix of size $N \times N$ by stacking the elements of the vector along the diagonal of the output matrix.

   - $$\text{Hess2}_{red} = E\{\mathbf{y}g(\mathbf{y})^T\} \tag{5.203}$$

   - The reduced Hessian Hess_{red} is thus approximated by the $N \times N$ superposition matrix:

     $$\text{Hess}_{red} = \text{Hess}_{red} - 1 + \text{Hess}_{red} - 2 \tag{5.204}$$

3. In order to perform the algorithm on-line the update step:

$$\Delta\mathbf{X} = (Hess_{red})^{-1} E\{(\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T)\} \tag{5.205}$$

   can be performed on-line by replacing the expected value of the first gradient $E\{(\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T)\}$ by its instantaneous value and by replacing the expected value of the reduced Hessian by a running estimate of the expected value updated from instant $(t-1)$ to time instant t using an exponential window with a forgetting factor λ as:

$$\text{Hess}_{red}(t) = \lambda \times \text{Hess}_{red}(t-1) + \text{Hess}_{red} \tag{5.206}$$

   An extra simplification can be achieved by applying the matrix inversion lemma on the above sequential update of the expected value of the Hessian to calculate the inverse of this expected value at each time instant based on the previous inverse.

4. After obtaining the inverse of the reduced Hessian, an update of the demixing matrix can be carried out as follows:

$$\Delta\mathbf{X} = (\text{Hess}_{red})^{-1}(\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T)\mathbf{y}^T) \tag{5.207}$$

$$\Delta\mathbf{X}_{so} = \Delta\mathbf{X} - \Delta\mathbf{X}^T \tag{5.208}$$

$$\mathbf{W} = \exp(\mu\Delta\mathbf{X}_{so})\mathbf{W} \approx \mathbf{W} + \mu\Delta\mathbf{X}_{so}\mathbf{W} \tag{5.209}$$

Chapter 6

Verification of the Proposed Algorithms: Simulation Results and Analysis

6.1 Introduction

This chapter presents a more complete collection of simulations for each of the new algorithms presented in the preceding chapters. The chapter starts with discussing the performance measure used to evaluate the simulation results. The second section presents the simulation environment and the assumptions made about the mixing/dimixing process. These assumptions may differ depending on the type of mixing category as previously discussed in Chapter 2. The basic mixing environment is noise-free, reverbation-free, which is the basic instantaneous mixing scenario. Other more difficult situations would require essentially more special treatment, i.e. more sophisticated extensions and additional processing like bias-removal for noise. All the extensions of the basic mixing/demixing model have been summarized in Chapter 2. Section 6.4 introduces the set of data, i.e. the sources used in the simulation tests. These data are typically standard data that have been in use for BSS-algorithms and were introduced by the pioneer researchers in this domain. Each set of data is listed with its type and a brief comment on its statistical properties. The data is all referenced to the location where it was obtained from along with the publications where it was used as test data to produce simulation results.
Following the above set up for the simulation results, each of the following sections presents a collection of results for each of the algorithms developed in this dissertation. The simulation sections include different mixing situations of varying separation difficulty. The proposed algorithms are compared to standard algorithms. The results are accompanied with comments on the performance.

6.2 Performance Evaluation of Instantaneous BSS Methods

In order to evaluate the performance of the instantaneous BSS methods one needs to recall that the purpose of separation is to estimate, blindly, a separation matrix $\mathbf{W}$ that is able to undo the mixing effect of the mixing environment, i.e. matrix $\mathbf{A}$. In mathematical form the optimal separation is guaranteed when:

$$\mathbf{W} = \mathbf{A}^{-1} \tag{6.1}$$

$$\text{or} \qquad \mathbf{AW} = \mathbf{P} \tag{6.2}$$

where $\mathbf{P}$ is a permuted scaled version of the identity matrix. This matrix can be the key to measure the performance of the separation. It is given the name "*Performance matrix*". The performance index or measure of separation depends primarily on the ability to identify the permutation nature of the matrix $\mathbf{P}$. This can be explained by the following example. Assume 4 sources and 4 sensors (the number of 4 is completely arbitrary) and a mixing matrix $\mathbf{A}$ given by:

$$\mathbf{A} = \begin{pmatrix} 0.21 & 1 & 0.46 & 1.1 \\ 1.4 & 1.4 & 1.3 & 1.4 \\ 0.8 & 0.57 & 1.1 & 0.18 \\ 1.1 & 0.92 & 1.2 & 0.41 \end{pmatrix} \tag{6.3}$$

During the simulation of any successful instantaneous BSS algorithm at a specific run of the algorithm, assume the resulting demixing matrix $\mathbf{W}$ to be of the following form:

$$\mathbf{W} = \begin{pmatrix} -1 & 1.75 & -2.3 & 2 \\ 3 & -2.3 & -6.9 & 3.66 \\ -0.5 & -0.18 & 4.2 & -4.1 \\ -2.5 & 0.6 & 5.2 & -1.93 \end{pmatrix} \tag{6.4}$$

The performance matrix $\mathbf{P}$ in this case is equal to:

$$\mathbf{P} = \begin{pmatrix} 0.21 & 1 & 0.46 & 1.1 \\ 1.4 & 1.4 & 1.3 & 1.4 \\ 0.8 & 0.57 & 1.1 & 0.18 \\ 1.1 & 0.92 & 1.2 & 0.41 \end{pmatrix} \begin{pmatrix} -1 & 1.75 & -2.3 & 2 \\ 3 & -2.3 & -6.9 & 3.66 \\ -0.5 & -0.18 & 4.2 & -4.1 \\ -2.5 & 0.6 & 5.2 & -1.93 \end{pmatrix} = \begin{pmatrix} -0.19 & \underline{-1.35} & 0.2 & 0.07 \\ \underline{-1.35} & -0.16 & -0.14 & -0.1 \\ -0.09 & 0 & -0.2 & \underline{-1.2} \\ 0.04 & -0.16 & \underline{-1.7} & -0.14 \end{pmatrix} \tag{6.5}$$

Clearly, the matrix $\mathbf{P}$ is not an identity matrix but a scaled permutation of the identity matrix with small values in place of zeros. The scaling is due to the ambiguity of scale and sign of the sources while the permutation reveals that the order of the output signals is not necessarily the same as the original independent sources. There is a dominant value in every column corresponding to the separated independent component signal, the position of this value corresponds to the original source it estimates. Optimally, the rest of the values in the same row or column in the matrix should decay to zero. However, if the separation is not perfect there remains cross-talking, i.e residual components from other signals contributing to the dominant output.

The performance index measures the difference of the final performance matrix $\mathbf{P}$ from a

scaled permutation of the identity matrix. This implies that for an $N \times N$ performance matrix, only N elements are not zeros and do not overlap in a row or a column. Secondly, the other elements should be zero. The cross-talking performance index is given by the following equation:

$$\eta = 20\log(\sum_{i=1}^{N}(\sum_{j=1}^{N}\frac{|\mathbf{P}_{ij}|}{\max_k|\mathbf{P}_{ik}|}-1)+\sum_{j=1}^{N}(\sum_{i=1}^{N}\frac{|\mathbf{P}_{ij}|}{\max_k|\mathbf{P}_{kj}|}-1)) \tag{6.6}$$

6.3 Simulation Environment

The scope of the thesis concentrates mainly on the instantaneous BSS algorithms. All the algorithms developed or compared with are instantaneous algorithms covering the basic concepts of separation. The main intention throughout the dissertation is to provide improved and enhanced algorithms based on the basic cost functions, upon which researchers have built the wide domain of BSS. The mixing environment is thus the basic BSS model, previously discussed in Chapter 2. It is reproduced in the following figure. In this model the sources are mixed linearly by a scalar square matrix $\mathbf{A}$ to produce a mixture $\mathbf{x}$, this mixture is then applied to the demixer, which is basically the unit where the intended BSS algorithms is applied. When separation is achieved, the resulting output $\mathbf{y}$ is assumed to be equal to the independent components $\mathbf{s}$ except for sign, scalar and order ambiguity.

The assumptions made about the mixing environment can thus be summarized as:

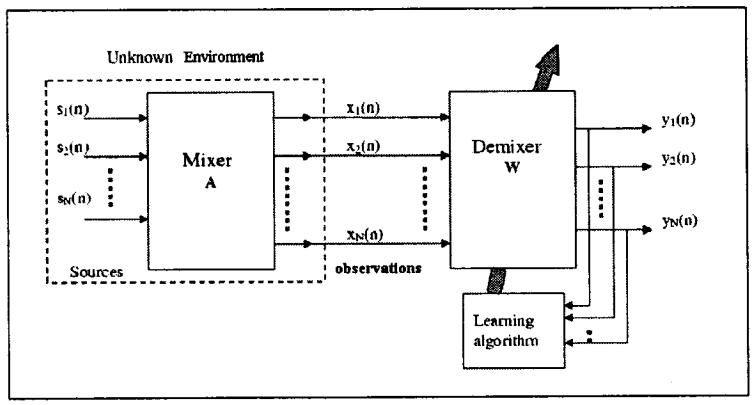

Figure 6.1: **Block diagram of instantaneous mixing/demixing situation**

1. The sources $\mathbf{s}$ are nongaussian.
2. The sources $\mathbf{s}$ are independent from each other.
3. The mixing matrix $\mathbf{A}$ is a scalar matrix: the mixing process is not time-dependent, i.e. the environment is echo-free or reverbation free in case of acoustic signals.
4. The number of sensors M is equal or greater than the number of sources N, i.e. $M \geq N$.
5. The mixing matrix $\mathbf{A}$ is linear.
6. The environment is noise-free.

Following these assumptions, the task at hand is dedicated to developing new algorithms without complicating the situation with other problems that are not fundamentally related to the separation target and are not related to the core scope of the thesis. Admittedly, the consideration of noise and reverbation would be useful in real-life applications. However, the solutions considered in the literature, as reviewed in Chapter 2, provide extensions that do not invalidate the developed algorithms but are more likely to extend them. For example adding bias-removal in the case of noise or transforming them to work in complex-domain for convolutive mixtures separation (i.e. the case where reverbation or echo exist). It would definitely be a good future work point to consider these cases.
Another important case to consider is the case where pre-whitening is applied, which is basically the environment of simulation for the algorithms developed in Chapter 4, i.e. Stiefel-manifold algorithms. The mixing conditions are the same as the above general instantaneous BSS scenario, the only addition is that the separation block is preceded by a whitening stage. The purpose of this stage is to produce a new mixture $\mathbf{z}$ that is zero mean, unit variance and uncorrelated. The visualization of the cascade whitening/demixing is presented in the next figure.

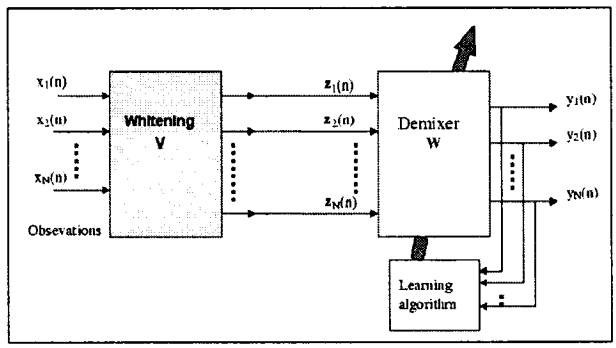

Figure 6.2: **Block diagram of instantaneous mixing/demixing situation with pre-whitening**

Here, one may define a new matrix $\mathbf{A}_{new}$ to represent the new mixing relationship between the sources $\mathbf{s}$ and the whitened mixtures $\mathbf{z}$.

$$\mathbf{A}_{new} = \mathbf{VA} \tag{6.7}$$

The separation block acts on the system after whitening, i.e. employs the mixtures $\mathbf{z}$ in the separation. Thus, the new mixing matrix $\mathbf{A}_{new}$ is the matrix that the separation block tries to invert in the BSS process and thus:

$$\mathbf{P} = \mathbf{WA}_{new} = \mathbf{WVA} \tag{6.8}$$

This is the new performance matrix. The actual implementation of the whitening uses Principle Components Analysis (PCA) as in all the algorithms developed for Negentropy [57], [56], [53], [51], [55], [54].

6.4 Simulation Data

In the beginning of our research related to the BSS problem, several web sites dedicated to the BSS problem were consulted, [47], [21], [72]. Among the data provided at these web-sites, the following are the suitable test data for instantaneous mixing tests. The data are categorized according to their type , i.e. super-gaussian or subgaussian.

1. **Subgaussian data**
   These signals have been proposed by Amari. et al. and they have been used in the following journal papers: [2], [97], [100] and [76]:

   (a) s_1: Sign signal
   $$s_1(t) = \text{sign}(\cos(2\pi 155t)) \tag{6.9}$$

   (b) s_2: High-frequency sinusoid signal:
   $$s_2(t) = \sin(2\pi 800t) \tag{6.10}$$

   (c) s_3: Low-frequency sinusoid signal:
   $$s_3(t) = \sin(2\pi 90t) \tag{6.11}$$

   (d) s_4: Amplitude-modulated signal
   $$s_4(t) = \sin(2\pi 9t)\sin(2\pi 300t) \tag{6.12}$$

   (e) s_5: Phase-modulated signal
   $$s_5(t) = \sin(2\pi 300t + 6\cos(2\pi 60t)) \tag{6.13}$$

   (f) s_6: Noise signal: $n(t)$ is uniformly distributed in [-1,+1],
   $$s_6(t) = n(t) \tag{6.14}$$

2. **Super-Gaussian data**The following speech and audio files where obtained from [73] and [48]. They are sampled at 8 kHz and have a duration of 6.5 sec. They have the following description:

   (a) Male-1 [48]: This file is a male speaker reading a sentence in English.

   (b) Male-2 [73]: This file has the name "mike" in the reference web site. It is also for a male speaker with similar voice properties to the above speaker and reading a different sentence in English, with similar style and speed of reading.

   (c) Female-1 [48]: This file has a female speaker reading a third sentence in English.

   (d) Male-3 [48]: This file is a male speaker reading a sentence in Finnish.

   (e) Male-4 [48]: This file is the same male speaker "male-3". However, in "male-4" file, he reads a sentence in English.

(f) Siren [48]: This file contains police siren as an example for audio data with fast changing amplitude.

(g) Music-1 [48]: This file contains symphony music.

(h) Music-2 [73]: Another symphony music. However, this file is called "beet.wav" at the reference site.

(i) Music-3 [73]: A third symphony music.

(j) Opera-1: Opera group vocal accompanied with orchestra music. This file has the original name "beet9.wav" at the reference [73].

(k) Opera-2 [48]: A soprano leading an opera piece with quiet music in the back.

(l) Street noise [73]: A recording of regular noise in outdoors environment.

3. **Correlated Speech**[59]: **10halo.mat**:
This matlab workspace file contains 10 speech signals that are highly correlated (all 10 speakers say the same word in Japanese). However, they are of very short duration, i.e. 0.6 sec.

6.5 Simulation Scenarios

The mixing scenarios are categorized in the following situations:

1 **Simple mixing situation**: In these cases the mixing matrix is close to a scaled permutation of the identity matrix. This means that a mixture signal is constructed with a high percentage of (mostly consisting of) one source with a residual of the other sources. An example of a 4×4 mixing matrix is:

$$\mathbf{A} = \begin{pmatrix} 0.9 & 0.09 & 0.18 & 0.25 \\ 0.29 & 1.2 & 0.3 & 0.29 \\ 0.46 & 0.28 & 0.98 & 0.05 \\ 0.15 & 0.32 & 0.15 & 0.81 \end{pmatrix} \tag{6.15}$$

The mixtures have the form:

$$\mathbf{x}_1 = 0.9\mathbf{s}_1 + 0.09\mathbf{s}_2 + 0.18\mathbf{s}_3 + 0.25\mathbf{s}_4 \tag{6.16}$$
$$\mathbf{x}_2 = 0.29\mathbf{s}_1 + 1.2\mathbf{s}_2 + 0.3\mathbf{s}_3 + 0.29\mathbf{s}_4 \tag{6.17}$$
$$\mathbf{x}_3 = 0.46\mathbf{s}_1 + 0.28\mathbf{s}_2 + 0.98\mathbf{s}_3 + 0.05\mathbf{s}_4 \tag{6.18}$$
$$\mathbf{x}_4 = 0.15\mathbf{s}_1 + 0.32\mathbf{s}_2 + 0.15\mathbf{s}_3 + 0.81\mathbf{s}_4 \tag{6.19}$$
$$\tag{6.20}$$

in which each mixture is dominated by one source.

2 **Difficult mixing situation**: In such a case the mixing matrix row coefficients are not biased towards a specific source. i.e. are quite similar in magnitude. The mixing matrix $\mathbf{A}$ in such a case is chosen to have the following value:

$$\mathbf{A} = \begin{pmatrix} 0.9 & 0.9 & 0.8 & 0.5 \\ 0.9 & 1.2 & 1.3 & 0.99 \\ 0.6 & 0.8 & 0.8 & 0.84 \\ 0.25 & 0.32 & 0.5 & 0.41 \end{pmatrix} \tag{6.21}$$

3 **Average mixing**: In this type of tests, the mixing matrix is generated randomly for each run, the mixing and de-mixing is performed by the algorithms at hand, and then the performance of these multiple runs are averaged. The parameters of the algorithms, (i.e. step size or the nonlinear score function used) are kept the same from one run to the other. The randomness of this test provides a true measure of the algorithm in practice. An ideal algorithm will perform well under any mixing situation and is not limited to a specific scenario or special case application.

6.6 Benchmark Algorithms

In Chapter 2, different algorithms developed in the literature were reviewed. In order to choose benchmark algorithms to compare the newly developed algorithms with, one has to consider the nature of the separation process, pre-whitened or non-prewhitened. Another consideration is that the algorithms developed are sample by sample based algorithms which can be implemented on-line. In [56], a very good discussion and comparison between most existing algorithms revealed that both the FAST-ICA algorithm, as summarized in Table (2.2), and the natural gradient provide the best performance. The best performance is based on immunity to outliers and convergence speed. Several other researchers concluded the same superiority of these two algorithms [88]. These two algorithms are thus used as the benchmark to compare the algorithms developed in the thesis.
However, it is important to note that the FAST-ICA algorithm requires that the input mixtures' $x_i, i = 1 \cdots N$ are pre-whitened prior to the separation step. Thus, the benchmark comparison is divided into 2 subcategories based on whether pre-whitening is applied or not:

- **Pre-whitened mixtures algorithms**: Basically the algorithms developed in Chapter 4. These algorithms work on the assumption that the mixtures are spatially pre-whitened and consequently the mixing matrix $\mathbf{A}$ and the de-mixing matrix $\mathbf{W}$ are orthogonal or orthonormal [1] matrices :

$$\mathbf{W}\mathbf{W}^T = \mathbf{I} \tag{6.22}$$

where $\mathbf{I}$ is the identity matrix.
These algorithms are, mainly, compared to:

[1] Blind separating matrices can be assumed to be orthonormal as the separated sources variances are unknown and can be assumed to be unity

1. The natural gradient (NAG) algorithm on the Stiefel manifold: This algorithms works on-line and has the following update equation:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.23}$$

where $g(y)$ is the element by element nonlinearity defined throughout the thesis to be:

$$g(y_i) = \begin{cases} y_i^3, & \text{for subgaussian } y_i \\ \tanh(y_i), & \text{for supergaussian } y_i \end{cases} \tag{6.24}$$

and $i = 1 \cdots N$, where N is the number of the recovered sources.

2. The second benchmark is the FAST-ICA algorithm: The algorithm works off-line (in batch mode) and requires several iterations (runs) to reach a convergence point. It is used in the simulations as an indication of the ultimate performance that the algorithms can reach. The point of the comparison with the FAST-ICA is to test how far the on-line single run algorithms performance is from the FAST-ICA performance.

- **Non-whitened mixtures algorithms**: These are basically the algorithms developed in Chapter 5. These algorithms work on the sole assumption that the mixing matrix $\mathbf{A}$ is real and invertible, i.e $\mathbf{A} \in GL$, where the General linear group $GL - group$ is the general group in the Riemannian manifold. The algorithms developed under this category are all based on the maximum-likelihood cost function and build on the natural gradient update equation. Consequently, the benchmark they are compared to is the natural gradient algorithm for the general Riemannian space, i.e. $GL - group$, given by the update equation:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\mathbf{I} - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.25}$$

The step size μ is usually set to 0.0005 which is a good choice to achieve stability, small steady state error and acceptable convergence speed for all the cases considered.
The FAST-ICA algorithm is not applicable in this case as it requires pre-whitening as a necessary condition for implementation.

The following sections present a more complete collection of results of the algorithms developed in the thesis. The results are displayed according to the order of the presentation of the algorithms in the thesis. The results are presented for the pre-whitened mixtures first, i.e. Stiefel-manifold BSS. The algorithms included in the results are those developed in Chapter 4. Afterwards, the algorithms developed in Chapter 5 are tested and presented along with their results collectively as "Non-prewhitend mixtures BSS" algorithms.

6.7 Simulation Results for Prewhitened Mixtures Case (Stiefel Manifold)

The algorithms developed in Chapter 4 to deal with the Stiefel manifold BSS problem are based on four cost functions, namely, maximum-likelihood, Negentropy, Nonlinear Principle

Component analysis (NPCA) and the newly developed Bussgang cost-function. The algorithms developed in this chapter can be categorized into two main categories based on the number of sources estimated.

- **One-unit algorithms**: These algorithms extract sources sequentially. Blind Signal Extraction (BSE) algorithms provide the system with the flexibility to extract the most dominant sources instead of having to separate all sources at once. BSE-algorithms are inherently more stable than their BSS counterparts when the number of sources is not known prior to the separation process. One-unit algorithms are also less computationally demanding and are suitable choices for de-noising and signal enhancement applications. The list of algorithms developed for blind extraction in Chapter 4 include:

  1. RLS1-BSE: This algorithm is developed based on the NPCA cost function and exhibits fast convergence speed and suitability for blind extraction of sources that are not highly non-stationary.
  2. RLS2-BSE: This algorithm, presented in [33] is developed based on the Negentropy cost function and exhibits fast convergence speed and suitability for non-stationary signals like speech signals.
  3. modified RLS2-BSE: This algorithm adds a projection step to the RLS2-BSE. This projection enhances both the robustness and the convergence speed of the algorithm.

  The following section presents the results for the algorithms "RLS1-BSE" and "modified RLS2-BSE". The performance of "modified RLS2-BSE" is an enhanced version of "RLS2-BSE", which is presented in Chapter 4 along with simulation results averaged for different number of sources N.

- **Multi-unit algorithms**: The algorithm developed in this domain is intended to separate all sources at once, i.e. BSS algorithm. The RLS-BSS algorithm is based on the maximum-likelihood cost function. It is presented along with its results following the BSE algorithms.

6.7.1 Simulation Results for RLS1-BSE Algorithm

The RLS1-BSE algorithm has been presented in section 4.6 along with a sample test of a randomly generated mixing situation. The following figures present simulation results for the RLS1-BSE with an instantaneous mixing situation, where the number of sources N is equal to the number of sensors M and both are equal to 4. The algorithm is summarized in Tables 4.1 and 4.2. The simulations presented in this chapter are following Table 4.2, which is reproduced in the following table:

Table 6.1: **RLS1-BSE algorithm : 2^{nd} implementation**

$$y_i(t) = \mathbf{w}_i(t-1)^T\mathbf{z}(t) \tag{6.26}$$

$$Q_i(t) = \frac{P_i(t-1)}{\beta + g(y_i(t))P_i(t-1)y_i(t)} \tag{6.27}$$

$$P_i(t) = \frac{1}{\beta}[P_i(t-1) - Q_i(t)y_i(t)g(y_i(t))P_i(t-1)] \tag{6.28}$$

$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - [P_i(t)g(y_i(t))\mathbf{z}(t) - \mathbf{Q}_i(t)y_i(t)g(y_i(t))\mathbf{w}_i(t-1)] \tag{6.29}$$

The algorithm is compared to:

1. Negentropy on-line algorithm: As presented by equation (4.11).
2. Natural gradient based on Nonlinear PCA (NAG) [97], [5]:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\mathbf{y}(t) \times g(\mathbf{y}(t))^T - g(\mathbf{y}(t)) \times \mathbf{y}^T(t)]\mathbf{W}(t-1) \tag{6.30}$$

3. **RLS algorithm for Blind Separation (BSS) based on the NPCA** (RLS-NPCA) [100]: This BSS counterpart of the "RLS1-BSE" algorithm has the inherent advantage that the error does not accumulate from one source to the source as in BSE algorithms.
4. FAST-ICA batch algorithm: The steps of the FAST-ICA algorithm are summarized in Table 2.1.

The numerical choices for parameters of the algorithms are presented in the following table. The initial values for each of the vectors $\mathbf{w}_i, i = 1...N$ is given by :

Table 6.2: **Parameters for RLS1-BSE simulations**

Parameter	Value
$g(\mathbf{y})$	$\mathbf{y}^3$
μ_{NAG}	0.0001
$\mu_{Negentropy}$	0.0005
β	0.999

$$\mathbf{w}_i = \begin{pmatrix} \mathbf{0}_{i-1} \\ 1 \\ \mathbf{0}_{N-i} \end{pmatrix} \tag{6.31}$$

The test data is the subgaussian benchmark data presented in 6.4, which is the same data set used to produce the results in section 4.6.
$\mathbf{s}(t) = [\text{sign}(\cos(2\pi 155t)), \sin(2\pi 800t), sin(2\pi 90t), \sin(2\pi 9t)\sin(2\pi 300t)]^T$. The following are results for different mixing/demixing scenario.

- **Easy Mixing Scene**: Using the same easy mixing matrix presented in section 6.5, the following figure shows the results for the easy mixing scenario.

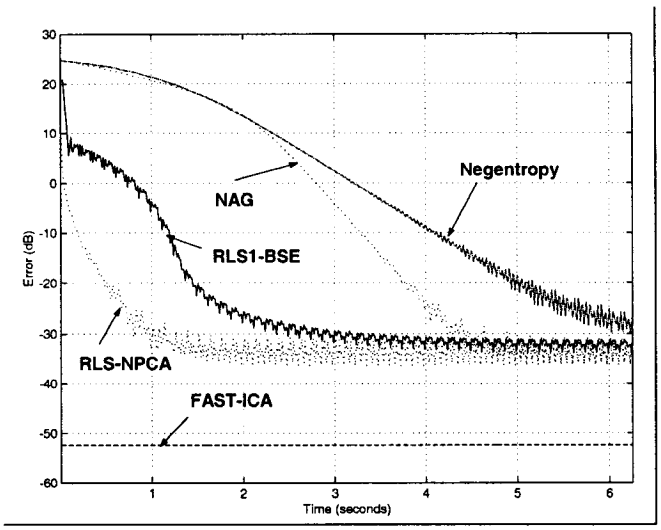

Figure 6.3: **RLS1-BSE simulation results for easy mixing scene**

- **Difficult Mixing Scene**: Using the same difficult mixing matrix presented in section 6.5, the following figure shows the results for the difficult mixing scenario.

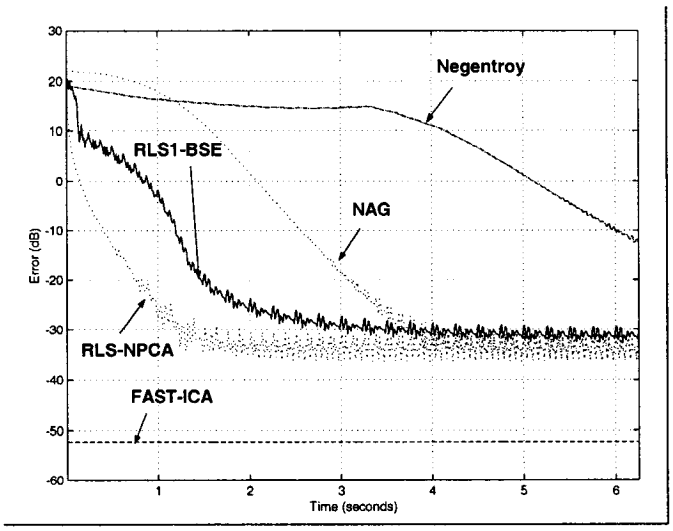

Figure 6.4: **RLS1-BSE simulation results for difficult mixing scene**

- **Average of 100 Random Mixing Situations**:

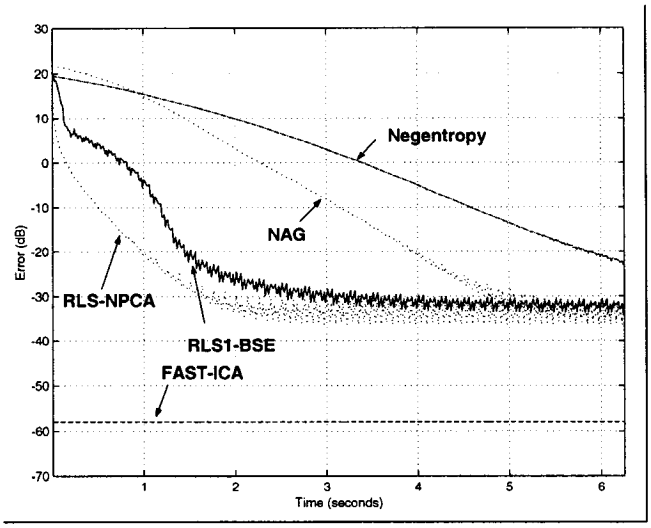

Figure 6.5: **RLS1-BSE average results for 100 different mixing situations**

One can see from the above figures that the RLS-NPCA is consistently the better on-line algorithm. However, the RLS1-BSE algorithm still provides comparable performance and is noticeably better than the natural gradient algorithm. The RLS1-BSE performs well in the different mixing/demixing scenarios considered. This is despite the fact that RLS1-BSE is an extraction algorithm, which means that error accumulates from one source to the source that follows in the extraction. However, the proposed algorithm works well and is shown to produce both improved performance, i.e. small steady state error and faster converging separation than the natural gradient algorithm.
Despite working well with the above set of benchmarks, the RLS1-BSE algorithm did not produce the same improvement when tested with non-stationary speech benchmarks. This remark, stated in Chapter 4, stimulated the development of the RLS2-BSE algorithm, whose results are presented in the next section.

6.7.2 Simulation Results for Modified RLS2-BSE Algorithm

The modified RLS2-BSE algorithm has been presented in section 4.8 along with a sample test of a randomly generated mixing situation to illustrate its improved performance over the RLS2-BSE. The following figures present simulation results for the RLS2-BSE algorithm with instantaneous mixing situation, where the number of sources N is equal to the number of sensors M and both are equal to 4. The algorithm is summarized in Table 4.3, which is reproduced in the following table:

Table 6.3: **Modified RLS2-BSE algorithm**

$$y_i(t) = \mathbf{w}_i(t-1)^T \mathbf{z}(t) \tag{6.32}$$

$$\beta_i(t) = \lambda\beta_i(t-1) + g(y_i(t))y_i(t) \tag{6.33}$$

$$\Delta = \mu g(y_i(t))\mathbf{z}(t)/\beta \tag{6.34}$$

$$\mathbf{w}_+ = \mathbf{w}_i(t-1) - (\Delta - \mathbf{w}(t-1) \times \Delta^T \times \mathbf{w}(t-1)) \tag{6.35}$$

- if $t/1000 = n, n = 1, 2, \cdots$:

$$\mathbf{w}_i(t) = \mathbf{w}_+/\|\mathbf{w}_+\| \tag{6.36}$$

- else:

$$\mathbf{w}_i(t) = \mathbf{w}_+ \tag{6.37}$$

Similar to RLS1-BSE algorithm, the RLS2-BSE algorithm is compared to:

1. Negentropy on-line algorithm: As presented by equation (4.11).
2. Natural gradient based on Nonlinear PCA (NAG) [97], [5]:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu[\mathbf{y}(t) \times g(\mathbf{y}(t))^T - g(\mathbf{y}(t)) \times \mathbf{y}^T(t)]\mathbf{W}(t-1) \tag{6.38}$$

3. FAST-ICA batch algorithm: this off-line algorithm presents a reference performance for Negentropy based algorithms. The steps of the FAST-ICA algorithm are summarized in Table 2.1.

The numerical choices for the parameters of the algorithms are the following.

Table 6.4: **Parameters for modified RLS2-BSE simulations**

Parameter	Value
$g(\mathbf{y})$	$\tanh(\mathbf{y})$
μ_{NAG}	0.001
$\mu_{Negentropy}$	0.0005
β	0.999
$\mu_{modifedRLS2-BSE}$	0.015

The initial values for each of the vectors $\mathbf{w}_i, i = 1...N$ is given by :

$$\mathbf{w}_i = \begin{pmatrix} \mathbf{0}_{i-1} \\ 1 \\ \mathbf{0}_{N-i} \end{pmatrix} \tag{6.39}$$

The test data is the supergaussian benchmark data presented in 6.4. The test has been performed on three different data sets.

- The first set of benchmarks consists of different acoustic signals, showing a general, more realistic situation of mixing scenarios where various sound signals are mixed:

  1. Male-1.
  2. Male-2.
  3. Female-1.
  4. Music-1.

The following are the results of this first group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**: The following figure shows the results for the easy mixing scenario.

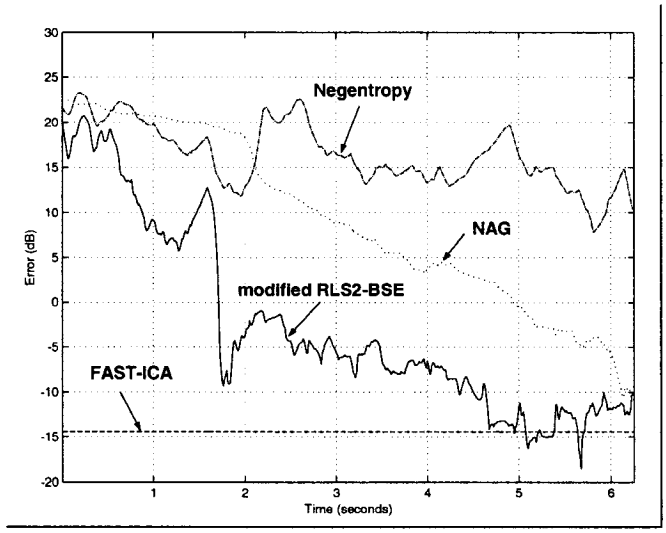

Figure 6.6: **Modified RLS2-BSE simulation results for easy mixing scene: 1^{st} benchmark data**

- **Difficult Mixing Scene**: The following figure shows the results for the above difficult mixing scenario.

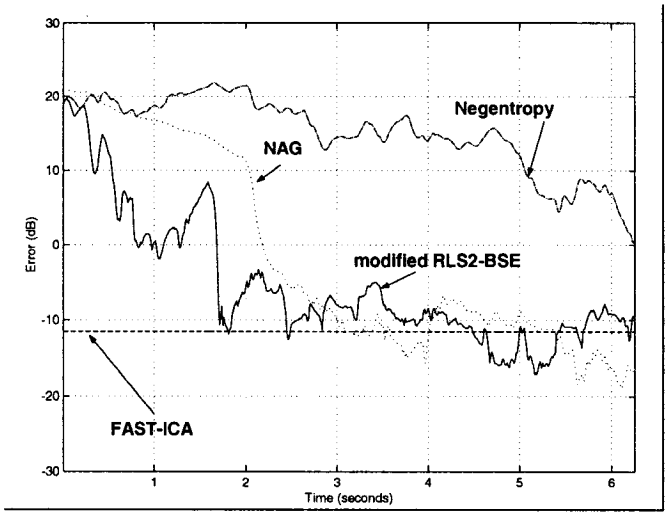

Figure 6.7: **Modified RLS2-BSE simulation results for difficult mixing scene: 1^{st} benchmark data**

- **Average of 100 Random Mixing Situations:**

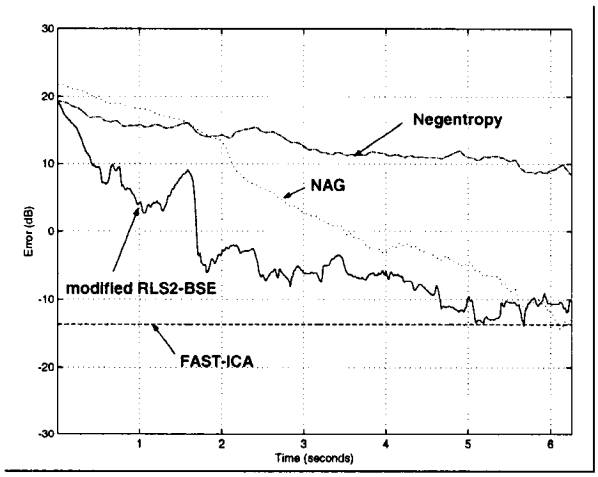

Figure 6.8: **Modified RLS2-BSE average results for 100 different mixing situations: 1^{st} benchmark data**

One can easily see from the above figures that the modified RLS2-BSE is consistently the better on-line algorithm. It performs well in the different mixing/demixing scenarios considered. This is despite the fact that the modified RLS2-BSE is a BSE algorithm and is inherently prone to more error accumulation than BSS algorithms. It works well with nonstationary signals like speech signals, while other Negentropy-based algorithms degrade in such case. A quick look at the above figures shows that the performance of the on-line Negentropy algorithm is significantly worse than its acceptable performance, previously seen with the subgaussian-data test performed along with the RLS1-BSE algorithm. The FAST-ICA algorithm also degrades in steady state error and requires more iterations to converge when speech and other acoustic signals are considered.

- A second set of benchmarks consists of speech signals that are very similar in characteristics. All of the signals are male speech. Two of the four male speakers are the same speaker saying two different sentences.

  1. Male-1.
  2. Male-2.
  3. Male-3.
  4. Male-4.

The following are the results of this group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**: The following figure shows the results for the easy mixing scenario.

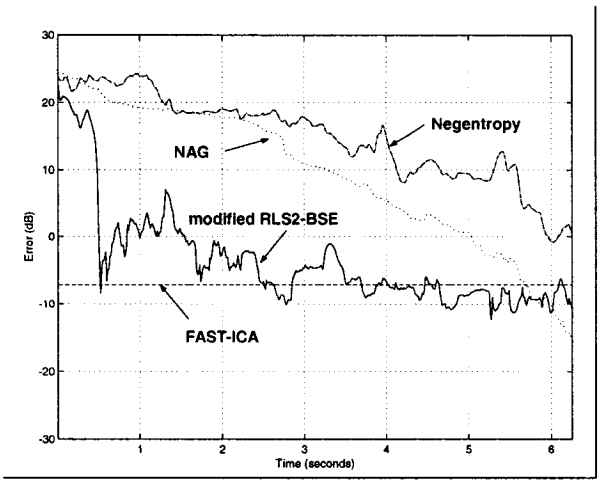

Figure 6.9: **Modified RLS2-BSE simulation results for easy mixing scene: 2^{nd} benchmark data**

- **Difficult Mixing Scene**:

The following figure shows the results for the above difficult mixing scenario.

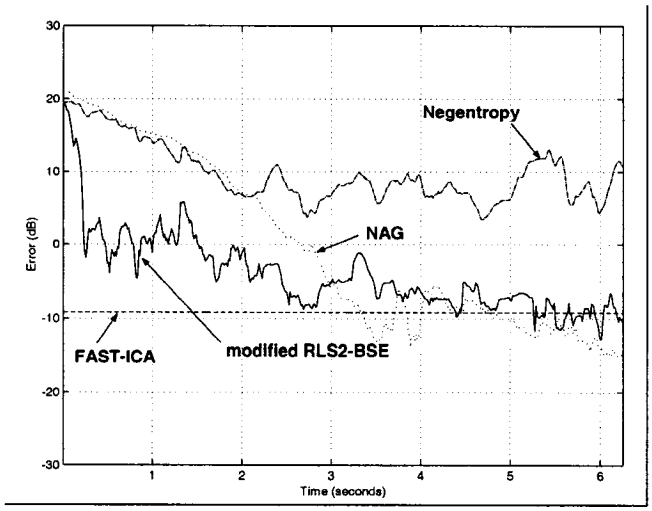

Figure 6.10: **Modified RLS2-BSE simulation results for difficult mixing scene: 2^{nd} benchmark data**

- **Average of 100 Random Mixing Situations:**

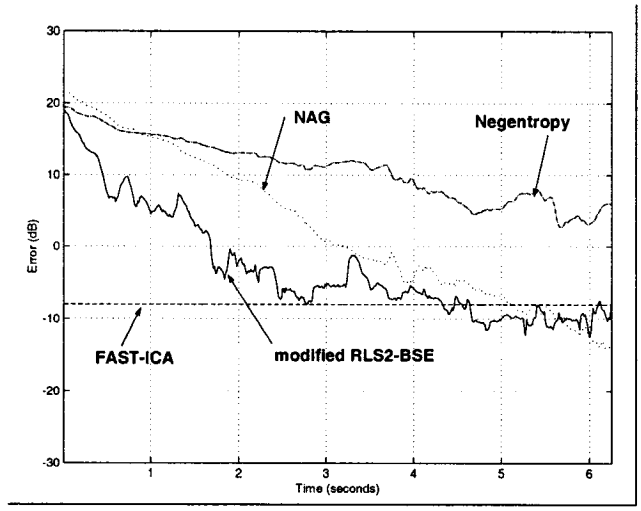

Figure 6.11: **Modified RLS2-BSE average results for 100 different mixing situations: 2^{nd} benchmark data**

Again, with the second group of benchmark files, the modified RLS2-BSE algorithm provides better convergence speed and can reach the steady state error achieved by the off-line FAST-ICA while working on a sample by sample basis.

- The last test is performed using four signals of the benchmark package **10halo.mat**, which contains 10 speech signals that are highly correlated (all 10 speakers say the

same sentence). However, they are of very short duration and thus were repeated 14 times.

Again the same mixing/demixing scenarios were considered. The following are the results of the first four signals of this correlated group of benchmarks:

- **Easy Mixing Scene**: The following figure shows the results for the easy mixing scenario.

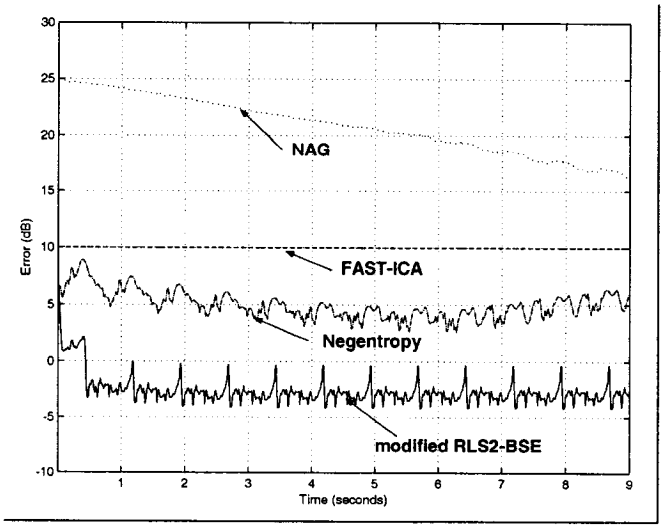

Figure 6.12: **Modified RLS2-BSE simulation results for easy mixing scene:** 3^{rd} **benchmark data**

- **Difficult Mixing Scene**: The following figure shows the results for the difficult mixing scenario.

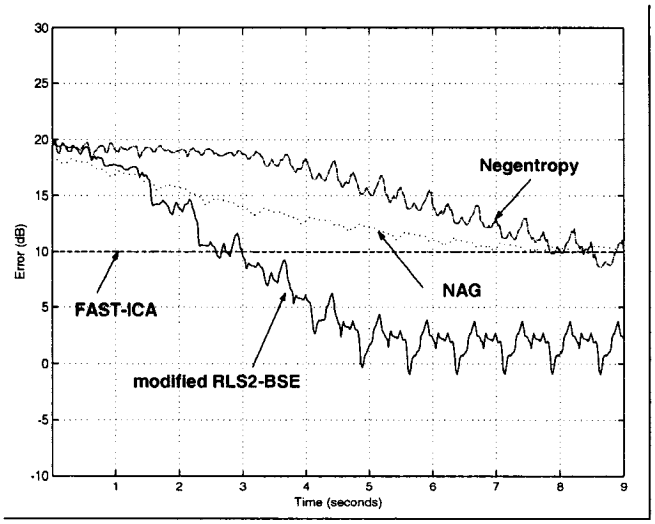

Figure 6.13: **Modified RLS2-BSE simulation results for difficult mixing scene: 3^{rd} benchmark data**

- **Average of 100 Random Mixing Situations:**

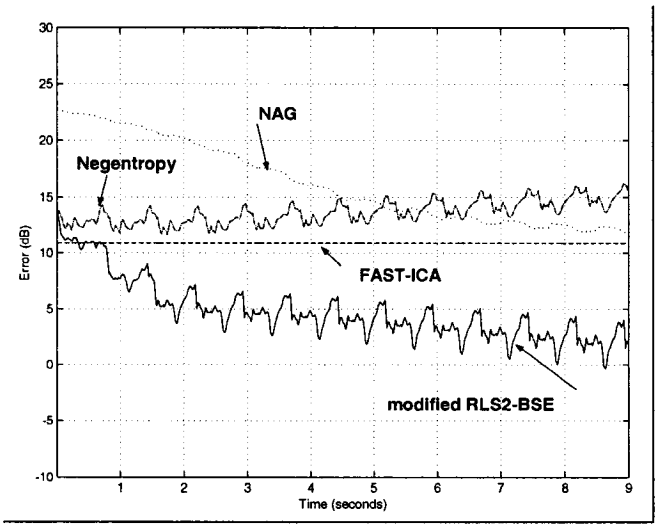

Figure 6.14: **Modified RLS2-BSE average results for 100 different mixing situations: 3^{rd} benchmark data**

Here one could recognize a significant deterioration in the performance of the FAST-ICA algorithm. It cannot achieve less than 10 dB. Another algorithm that is prone to deterioration is the on-line Negentropy algorithm, which, on average, has difficulty to converge. On the other hand, the proposed algorithm, "modified RLS2-BSE" is steadily proving good performance even under difficult mixing conditions.

6.7.3 Simulation Results for RLS-BSS Algorithm

The RLS-BSS algorithm is the blind signal separation algorithm presented in Chapter 4. It is based on the maximum-likelihood natural gradient and implements an RLS-like update using the matrix inversion lemma. In order to assess the performance of the RLS-BSS algorithm it is compared to the following algorithms:

1. Natural gradient on the Stiefel manifold as described by the update equation:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{NAG}[\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.40}$$

2. FAST-ICA batch algorithm as summarized in Table 2.1.

3. modified RLS2-BSE algorithm: this extraction algorithm presented in the thesis has proven to produce good results. It has been summarized in Table 4.3 and reproduced in Table 6.3.
   The modified RLS2-BSE is expected to be inferior to the performance provided by the RLS-BSS-Stiefel algorithm due to the fact that it is an extraction algorithm and the error accumulates from one source to the following. It is shown here because it is the fastest converging algorithm that produces the same or a smaller steady state error when compared to existing on-line separation algorithms, namely, the natural gradient and Negentropy.

The numerical choices for the parameters of the algorithms are the following. The test data

Table 6.5: **Parameters for RLS-BSS-Stiefel**

Parameter	Value
$g(\mathbf{y})$	$\tanh(\mathbf{y})$
μ_{NAG}	0.0005
$\mu_{Negentropy}$	0.0005
β	0.99
$\mu_{modifedRLS2-BSE}$	0.015
λ	0.9994
$\mu_{RLS-BSS}$	0.12

is the supergaussian benchmark data presented in 6.6. The test has been performed using the three different acoustic data sets. These are the same benchmarks used for the modified RLS2-BSE algorithm presented in the previous section.

- The first set of benchmarks consists of various sound signals:
  1. Male-1.
  2. Male-2.
  3. Female-1.
  4. Music-1.

Using the same mixing/demixing scenarios, the following are the results of this first group of benchmarks:

- **Easy Mixing Scene**:
  The following figure shows the results for the easy mixing scenario.

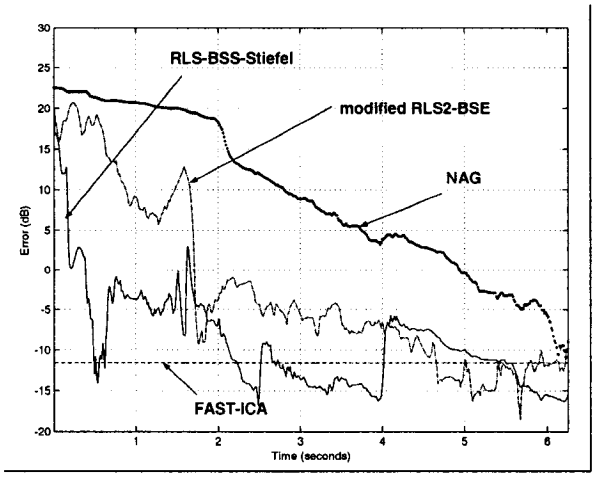

Figure 6.15: **RLS-BSS-Stiefel simulation results for easy mixing scene: 1^{st} benchmark data**

- **Difficult Mixing Scene**:
  The following figure shows the results for the difficult mixing scenario.

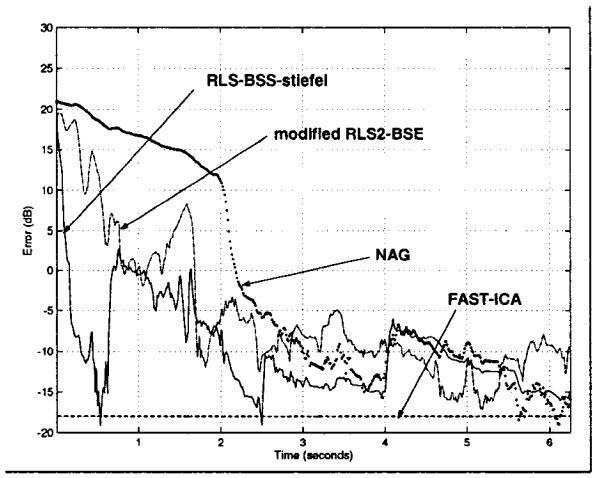

Figure 6.16: **RLS-BSS-Stiefel simulation results for difficult mixing scene: 1^{st} benchmark data**

- **Average of 100 Random Mixing Situations**:

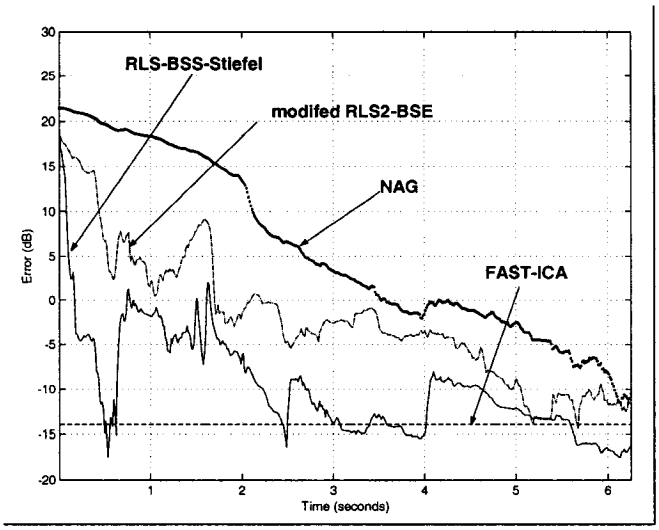

Figure 6.17: **RLS-BSS-Stiefel average results for 100 different mixing situations: 1^{st} benchmark data**

One can easily see from the above figures that the modified RLS-BSS is consistently the better on-line algorithm. It performs well in the different mixing/demixing scenarios considered.

- The second set of benchmarks consists of four male speech signals:

  1. Male-1.
  2. Male-2.
  3. Male-3.
  4. Male-4.

The following are the results of this group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**:

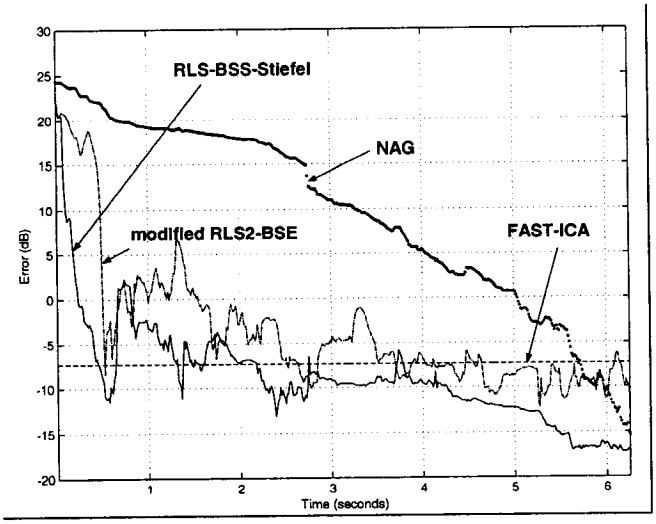

Figure 6.18: **RLS-BSS-Stiefel simulation results for easy mixing scene: 2^{nd} benchmark data**

- **Difficult Mixing Scene:**
  The following figure shows the results for the difficult mixing scenario.

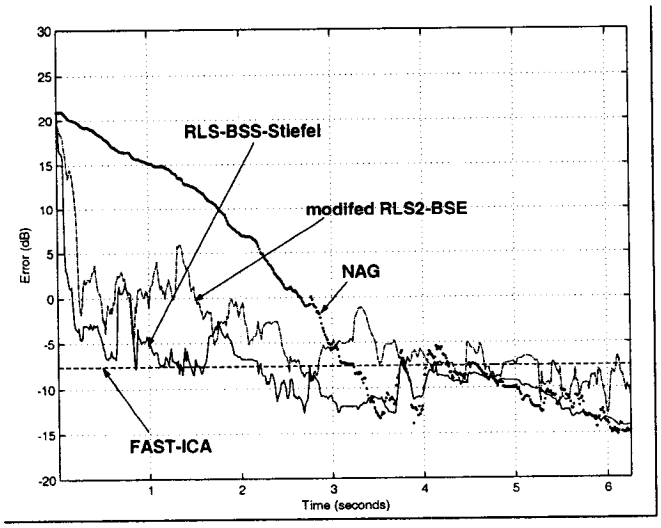

Figure 6.19: **RLS-BSS-Stiefel simulation results for difficult mixing scene: 2^{nd} benchmark data**

- **Average of 100 Random Mixing Situations:**

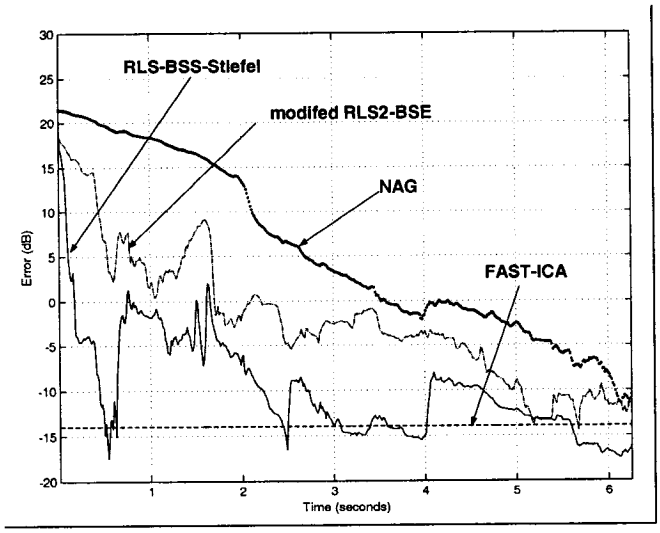

Figure 6.20: **RLS-BSS-Stiefel average results for 100 different mixing situations: 2^{nd} benchmark data**

- The last test is performed using four signals of the benchmark package **10halo.mat**
- **Difficult Mixing Scene:**
  The following figure shows the results for the difficult mixing scenario.

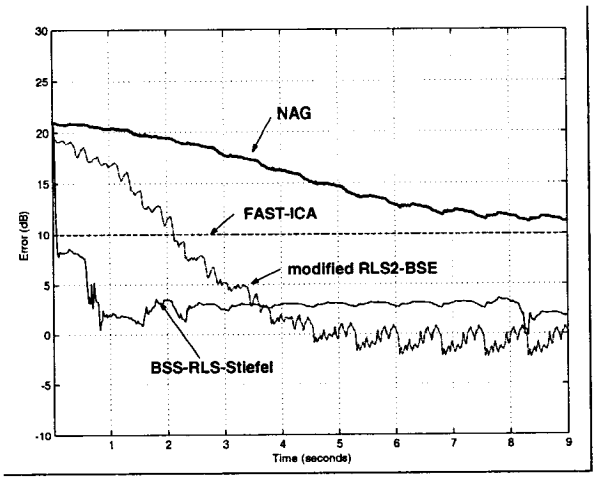

Figure 6.21: **RLS-BSS-Stiefel simulation results for difficult mixing scene: 3^{rd} benchmark data**

- **Average of 100 Random Mixing Situations:**
  The following figure shows the results for the average of the 100 different mixing scenarios.

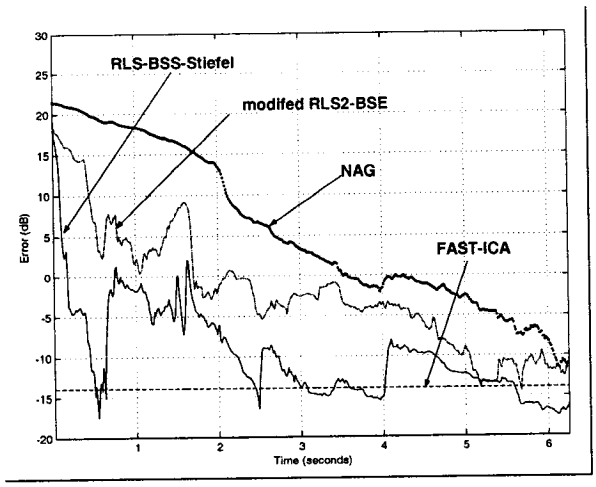

Figure 6.22: **RLS-BSS-Stiefel average results for 100 different mixing situations: 3^{rd} benchmark data**

As can be seen from all the above figures, the RLS-BSS-Stiefel algorithm provides the fastest convergence compared to all the other on-line algorithms. It consistently provides a robust and fast convergence in all mixing scenarios and all the benchmarks considered, followed in performance by the extraction algorithm, i.e. modified RLS2-BSE algorithm.

6.8 Simulation Results for Non Prewhitened Mixtures Case

The algorithms developed in Chapter 5 to deal with the general Riemannian manifold BSS problem are based on the maximum-likelihood cost function. These general Riemannian domain algorithms can be categorized into three categories based on the approach of derivation.

- **Generalized Natural Gradient based on Iterative Inversion Approach**
- **Generalized Riemannian Nonlinear-RLS Approach**
- **Hessian-based Algorithms**

6.8.1 Simulation Results for Generalized Natural Gradient Iterative Inversion Algorithms

The algorithms developed in this category were developed in details in section 5.5. The list of algorithms developed under this family includes:

1. LMS-1: This algorithm has the following update equation (5.58):

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{LMS-1}(\mathbf{R_{ss}} - \mathbf{R_{gy}})\mathbf{W}(t-1) \tag{6.41}$$

2. LMS-2: This algorithm has the following update equation (5.63):

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{LMS-2}(\mathbf{I} - \mathbf{R_{gy}}\mathbf{R_{ss}^{-1}}))\mathbf{W}(t-1) \tag{6.42}$$

3. LMS-RLS: This algorithm has the following update equation (5.71):

$$\mathbf{W}(t) = (\mathbf{I} + \mu_{LMS-RLS}(\mathbf{R_{gy}} - \mathbf{R_{ss}})\mathbf{R_{ss}^{-1}})^{-1}\mathbf{W}(t-1) \tag{6.43}$$

4. RLS-Iterative algorithm: This algorithm has the following update equation:

$$\mathbf{W}(t) = \frac{1}{\lambda}[I - \frac{\mathbf{g}(\mathbf{y}(t))\mathbf{y}(t)^T}{\alpha + \mathbf{g}(\mathbf{y}(t))^T\mathbf{y}(t)}]\mathbf{W}(t-1) \tag{6.44}$$

with $\mathbf{R_{gy}} = g(\mathbf{y})\mathbf{y}^T$. For all the above algorithms, the choice for $\mathbf{R_{ss}}$ is limited to a diagonal matrix, which is the least computationally expensive choice.

$$\mathbf{R_{ss}} = \text{diag}(\mathbf{y}\mathbf{y}^T) \tag{6.45}$$

These algorithms work well with data whose statistics do not change very fast. The following section presents the collective results for the above mentioned algorithms. The test data is the subgaussian benchmark data presented in 6.4, which is the same data set used to produce the results in section 4.6. The benchmark is:
$\mathbf{s}(t) = [\text{sign}(\cos(2\pi 155t)), \sin(2\pi 800t), \sin(2\pi 90t), \sin(2\pi 9t)\sin(2\pi 300t)]^T$.
The algorithms are compared to the natural gradient on the General Riemannian manifold:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{NAG}[\mathbf{I} - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.46}$$

The numerical choices for parameters of the algorithms are as follows.

Table 6.6: **Parameters of the Generalized Iterative Inversion family**

Parameter	Value
$g(\mathbf{y})$	$\mathbf{y}^3$
μ_{NAG}	0.0005
μ_{LMS-1}	0.005
μ_{LMS-2}	0.0005
$\mu_{LMS-RLS}$	0.01
β	starts at value 0.99 at $t = 0$ and increases to 0.99995 at $t = 6000$

Using the same mixing/demixing scenarios developed in section 6.5, the following are the simulation results:

- **Easy Mixing Scene**: The following figure presents the results for this situation.

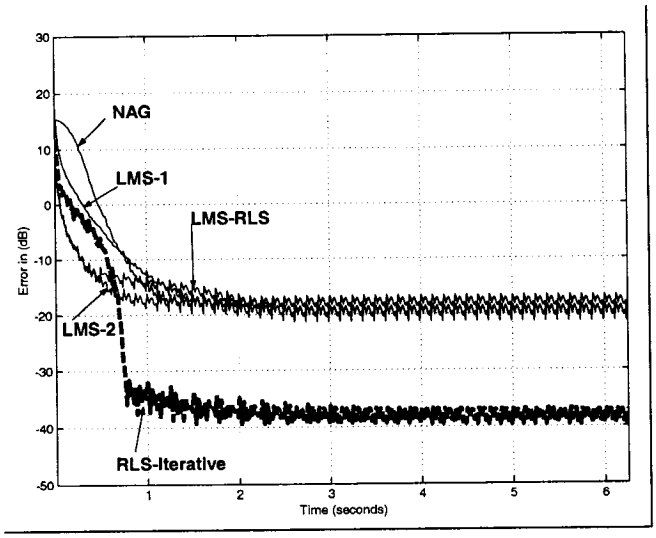

Figure 6.23: **Results for Generalized Iterative Inversion family: easy mixing scene**

In the above easy mixing situation the LMS-2 and LMS-RLS algorithms provide the fastest initial convergence speed. All the LMS-based algorithms, i.e. the natural gradient algorithm, the LMS-1 algorithm, the LMS-2 algorithm and the LMS-RLS algorithm converge to the same steady state error, while the RLS-iterative algorithm converges to a much smaller steady state error.
The conclusion that can be drawn from the above situation is that the LMS-2 and the RLS-iterative algorithm perform well in easy mixing situation. On the other hand, the LMS-1 algorithm can be considered slightly better than the natural gradient algorithm as it converges faster to the same steady state error. The RLS-iterative algorithm produces a fast convergence and a smaller steady state error than other algorithms. The average of the 100 random runs that will follow will provide a more decisive ranking of the performance of the algorithms. Such a ranking will be based on an average of different runs and will not be affected by a special case scenario.

- **Difficult Mixing Scene**:
The following figure presents the results for the difficult mixing case.
For the difficult mixing situations in Figure (6.24), the RLS-Iterative algorithm provides the best performance among all algorithms for this difficult mixing case, both in convergence speed and for steady state error. The LMS-RLS algorithm comes second in performance, while both the LMS-1 and LMS-2 algorithms provide a similar performance that is slightly better than the natural gradient algorithm.

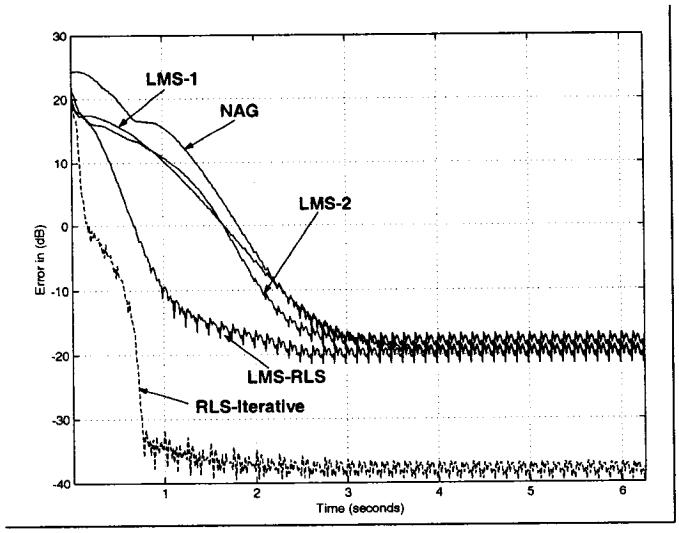

Figure 6.24: **Results for Generalized Iterative Inversion family: difficult mixing scene**

- **Average of 100 Random Mixing Situations:**

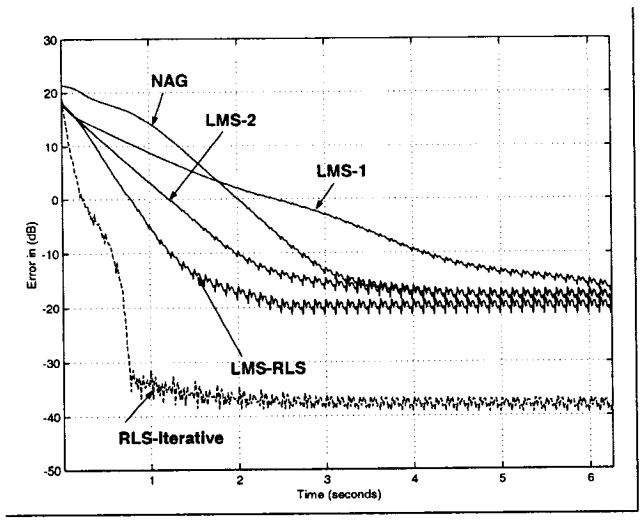

Figure 6.25: **Generalized Iterative Inversion Family, simulation results for 100 random mixing scenes**

It can be easily seen that the RLS-Iterative algorithm produces a consistently better performance than the whole family of the LMS-algorithms and the natural gradient algorithm. The RLS-Iterative inversion algorithm produces both a faster convergence and a smaller steady state error than the rest of the algorithms considered in the comparison. It has the same degree of complexity as the LMS-RLS or the LMS-2. Another advantage of the RLS algorithm over the LMS-RLS or the LMS-2 algorithm will become clear when considering the over-determined mixing situation $N < M$ in the next section.
On the other hand, the above figures show that the LMS-RLS algorithm is consistently the better on-line algorithm in the LMS- family. It performs well in the different mixing/demixing scenarios considered. The LMS-2 algorithm, which has the same complexity requirement as the LMS-RLS algorithm comes next in performance, achieving a slower convergence speed and a slightly higher steady state error. The natural gradient algorithm and the LMS-1 algorithm are comparable as the LMS-1 algorithm converges faster initially and then slows down. Both, the natural gradient algorithm and the LMS-1 achieve the same steady state error as the LMS-2, which is slightly higher than the LMS-RLS algorithm. However, the natural gradient and the LMS-1 algorithm are simpler in implementation compared to both the LMS-RLS and LMS-2. Along with the above $N = M$ conventional mixing case another special scenario is also considered. This special case is when the number of sources is not estimated correctly, i.e. the number of sources N is less than the number of sensors M. In such a case the mixing matrix $\mathbf{A}$ is ill-conditioned. Results of the average mixing scenario are presented in Figure (6.26).
In this test the mixing matrix $\mathbf{A}$ is of size 4×3 meaning that the number of sources is $N = 3$ and the number of mixtures is $M = 4$. The mixing matrix $\mathbf{A}$ is generated 100 different times, randomly, to produce different mixing situation. The 100 different results of each performances for each algorithm are averaged to provide a general idea of how well each algorithm performs.

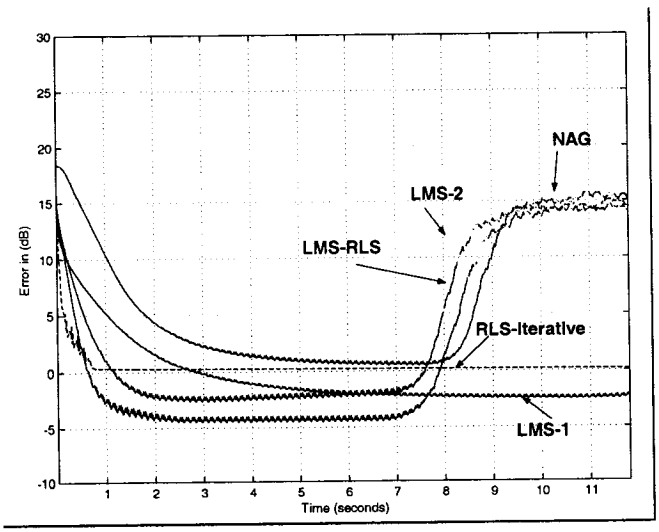

Figure 6.26: **Generalized-Iterative Inversion Family simulation results for 100 random mixing scenes, $N < M$**

As can be seen, the LMS-1 algorithm exhibits the most robust performance and does not

diverge as in the case for the rest of the LMS-based algorithms including the natural gradient algorithm. This robust performance can be justified by the special structure of the update equation of the LMS-1 algorithm as has been discussed in Chapter 5. Another algorithm that exhibits an acceptable performance is the RLS-Iterative inversion algorithm. However, the RLS-Iterative steady state performance deteriorates when compared to its remarkably superior performance in the case $N = M$. The steady state error of the RLS-iterative inversion algorithm is larger than the steady state error produced by the LMS-1 algorithm which requires less computations.
Collectively, the iterative inversion family of algorithms performs well with the subgaussian test data. However, the algorithms do not provide dramatically noticeable improvements over the natural gradient algorithm when tested with speech benchmarks. The following section provides the results for the Nonlinear RLS- algorithm, which can achieve better results with audio type data.

6.8.2 Simulation Results for Nonlinear-RLS Algorithm

The algorithm developed in section 5.6 can be considered as an extension of the idea of nonlinear RLS previously developed in Chapter 4 for the algorithm RLS-BSS-Stiefel. In this case the concept of nonlinear RLS is applied to the non prewhitened mixtures case using the following update, which was discussed in more details in section 5.6:

$$\mathbf{R} = E\{\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T\}_{vec} E\{\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T\}_{vec}^T \tag{6.47}$$

$$\mathbf{R}(t) = \mathbf{R}(t-1) + (\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec}(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec}^T \tag{6.48}$$

$$(\Delta\mathbf{X})_{vec} = \mathbf{R}^{-1}(\text{diag}(g(\mathbf{y})\mathbf{y}^T) - g(\mathbf{y})\mathbf{y}^T)_{vec} \tag{6.49}$$

$$\Delta\mathbf{X} = \text{mat}(\Delta\mathbf{X})_{vec} \tag{6.50}$$

$$\mathbf{W} = \exp(\mu\Delta\mathbf{X})\mathbf{W} \approx \mathbf{W} + \mu\Delta\mathbf{X}\mathbf{W} \tag{6.51}$$

This algorithm works well with nonstationary speech signals and thus it was tested with the speech and music benchmarks. The algorithm is compared to the natural gradient on the General Riemannian manifold:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{NAG}[\mathbf{I} - \phi(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.52}$$

The numerical choices for the parameters of the algorithms are presented in the following table.
where:

$$\text{sign}(\mathbf{y}) = \begin{cases} +1, y_i & \text{is positive} \\ -1, y_i & \text{is negative} \end{cases} \tag{6.53}$$

The test data is the supergaussian benchmark data presented in 6.6. The test has been performed on three different data sets.

- The first set of benchmarks consists of different acoustic signals:

  1. Male-1.

Table 6.7: **Parameters of the General Riemannian Nonlinear-RLS algorithm**

Parameter	Value
$\phi(\mathbf{y})$	$\tanh(\mathbf{y})$
$g(\mathbf{y})$	$\text{sign}(\mathbf{y})$
μ_{NAG}	0.0005
β	0.99
$\mu_{nonlinear RLS}$	1

2. Male-2.
3. Female-1.
4. Music-1.

The following are the results of this first group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**: The following figure shows the simulation results for the above easy mixing situation.

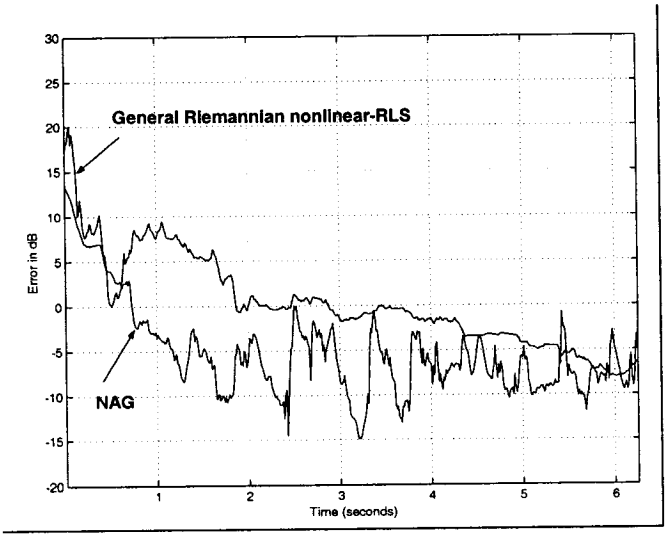

Figure 6.27: **General Riemannian Nonlinear-RLS simulation results for easy mixing situation: 1^{st} benchmark data**

In this easy mixing situation the natural gradient algorithm works better than the nonlinear RLS algorithm.

- **Difficult Mixing Scene**:
The following figure shows the results for the difficult mixing scenario. In the difficult mixing case, the general Riemannian nonlinear RLS algorithm provides better steady state error and converges initially faster compared to the natural gradient algorithm.

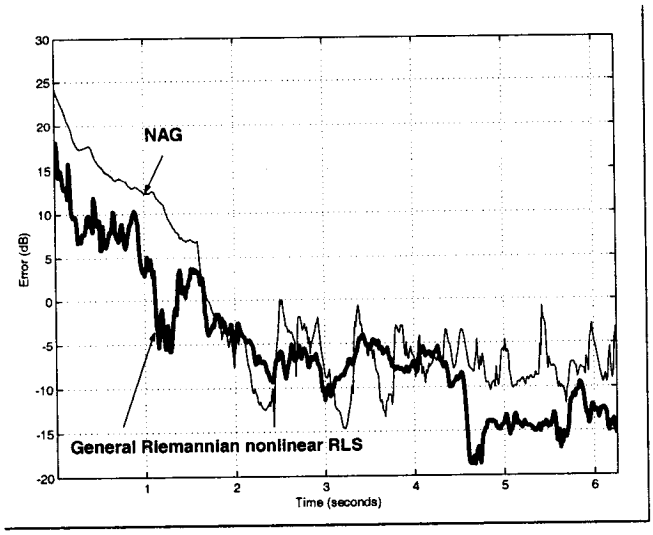

Figure 6.28: **General Riemannian Nonlinear RLS simulation results for difficult mixing situation: 1^{st} benchmark data**

- **Average of 100 Random Mixing Situations:**
  The following figure shows the average of the results for 100 different random mixing scenarios.

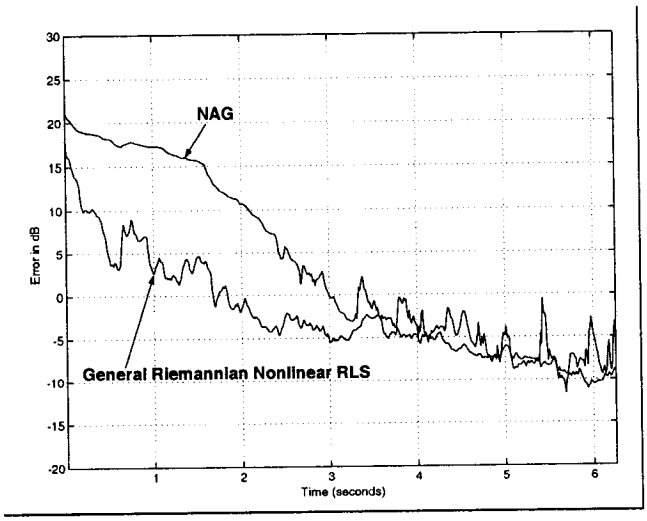

Figure 6.29: **General Riemannian Nonlinear RLS average simulation results for 100 different mixing situations: 1^{st} benchmark data**

On average, the general Riemannian nonlinear-RLS algorithm provides faster convergence compared to the natural gradient algorithm while achieving the same steady state error.

- The second set of benchmarks consists of speech signals that are very similar in characteristics.

    1. Male-1.
    2. Male-2.
    3. Male-3.
    4. Male-4.

The following are the results of this first group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**:
  The following figure presents the results for this mixing situation.

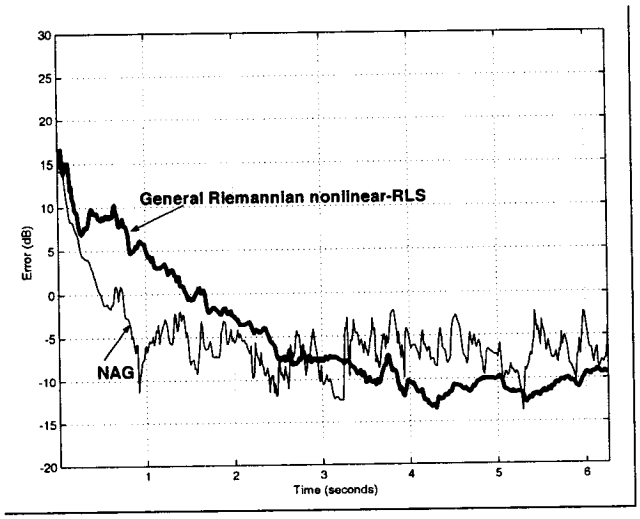

Figure 6.30: **General Riemannian Nonlinear RLS simulation results for easy mixing situation: 2^{nd} benchmark data**

Again, in an easy mixing scenario, the natural gradient algorithm performs slightly better than the General-Riemannian nonlinear RLS algorithm.

- **Difficult Mixing Scene**:
  The following figure shows the results for the difficult mixing scenario. In this difficult mixing case, the general Riemannian nonlinear RLS performs slightly better than the natural gradient algorithm.

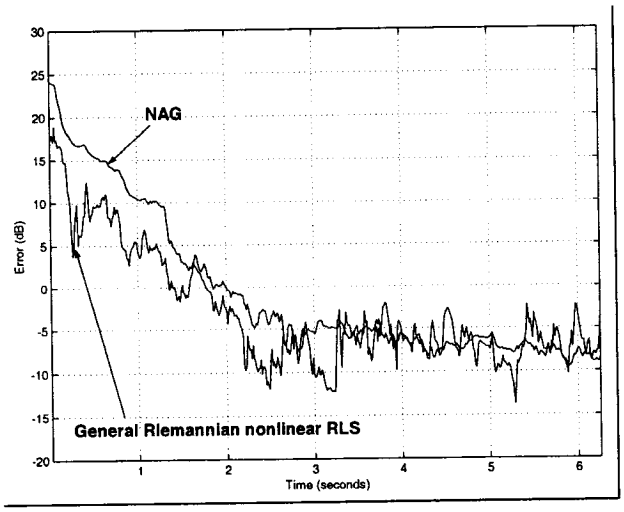

Figure 6.31: **General Riemannian Nonlinear RLS simulation results for difficult mixing situation: 2^{nd} benchmark data**

- **Average of 100 Random Mixing Situations**:
  The following figure shows the average of the results for 100 different randomly generated mixing scenarios.

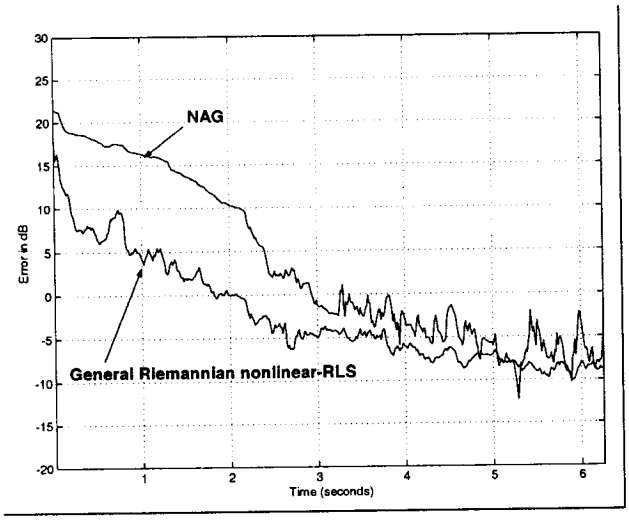

Figure 6.32: **General Riemannian Nonlinear RLS average simulation results for 100 different mixing situations: 2^{nd} benchmark data**

On average, the general Riemannian nonlinear RLS algorithm provides a faster convergence compared to the natural gradient algorithm while achieving the same steady state error.

- The last test is performed using four signals of the benchmark package **10halo.mat**, which contains 10 speech signals that are highly correlated.

The following are the results of this third group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**: The following figure shows the average of the above easy mixing scene.

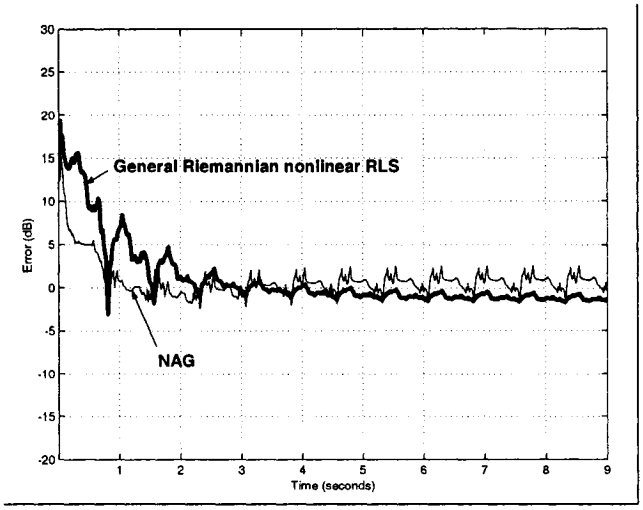

Figure 6.33: **General Riemannian Nonlinear RLS simulation results for easy mixing situation: 3^{rd} benchmark data**

Again, as with the other easy mixing scenarios of the two previous benchmarks, the natural gradient tends to produce a slightly faster convergence. However, the general Riemannian nonlinear RLS provides a slightly better steady state error.

- **Difficult Mixing Scene**:
The following figure shows the results for the difficult mixing scenario.

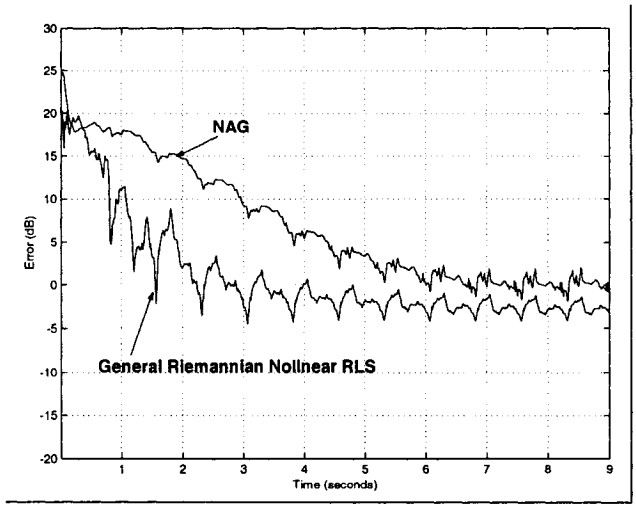

Figure 6.34: **General Riemannian Nonlinear RLS simulation results for difficult mixing situation: 3^{rd} benchmark data**

Clearly, the general Riemannian nonlinear RLS algorithm performs better than the natural gradient algorithm in the above difficult mixing situation.

- **Average of 100 Random Mixing Situations:**

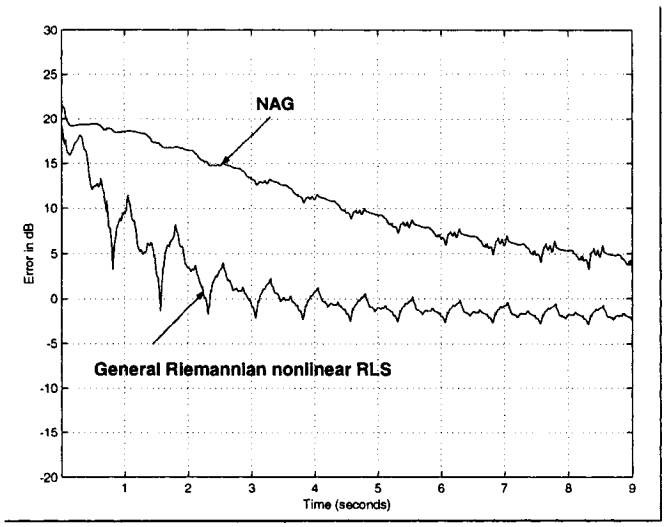

Figure 6.35: **General Riemannian Nonlinear RLS average simulation results for 100 different mixing situations: 3^{rd} benchmark data**

As can be seen in the above figure, the developed nonlinear RLS algorithm works well in the audio/speech mixing environment. It provides a better average performance than the natural gradient algorithm. However, one disadvantage of this algorithm is its computational requirement, which is considerably large, in the order of N^6 computations, where N is the

number of sources.
The following section provides the results for 2^{nd}-order Riemannian algorithms, which have been developed in Chapter 5 along with effective simplifications that lead to considerable reductions in the computational load.

6.8.3 Simulation Results for Hessian-based Algorithms

These algorithms have the unique advantage of using simplified and approximated updates of the Hessian matrix for the maximum-likelihood cost function on the Riemannian manifold. Due to the simplification implemented, the developed algorithms in this section are not computationally expensive yet they provide the improved performance expected from algorithms using second order derivatives. The list of algorithms developed in this category includes:

- Reduced Hessian Algorithm: This algorithm is presented in section 5.10.
- Median Reduced Hessian Algorithm: This algorithm is a variant of the Reduced Hessian algorithm. It is presented in details in section 5.11.
- Diagonal Hessian Algorithm: The most computationally optimized algorithm in this collection of algorithms is derived in details in section 5.12.

These algorithms provide very good performance both in convergence speed and stability. They have been tested with supergaussian data. The three algorithms developed in this family are compared to:

- The natural gradient algorithm on the General Riemannian manifold:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{NAG}[\mathbf{I} - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.54}$$

  The numerical choices for parameters of the algorithms are presented in the following table.

Table 6.8: **Parameters of Hessian-based algorithms**

Parameter	Value
$g(\mathbf{y})$	$\tanh(\mathbf{y})$
μ_{NAG}	0.0005
$\mu_{Median-ReducedHessian}$	0.09
$\mu_{Reduced-Hessian}$	0.1
$\mu_{diagonal-Hessian}$	0.1
$\beta_{Median-ReducedHessian}$	0.997
$\beta_{Reduced-Hessian}$	starts at value 0.992 at $t = 0$ and increases to 0.995 at $t = 25000$
$\beta_{Diagonal-Hessian}$	starts at value 0.994 at $t = 0$ and increases to 0.995 at $t = 25000$

The test data is the supergaussian benchmark data presented in 6.6. The test has been performed using the three different acoustic data sets. These are the same benchmarks used for the modified RLS2-BSE, RLS-BSS-Stiefel and the General Riemannian nonlinear-RLS algorithms presented in the previous sections.

- The first set of benchmarks consists of various sound signals:

  1. Male-1.
  2. Male-2.
  3. Female-1.
  4. Music-1.

The following are the results of this first group of benchmarks for different mixing/demixing scenario:

- **Easy Mixing Scene**:
  The following figure shows the results for the above easy mixing scenario.

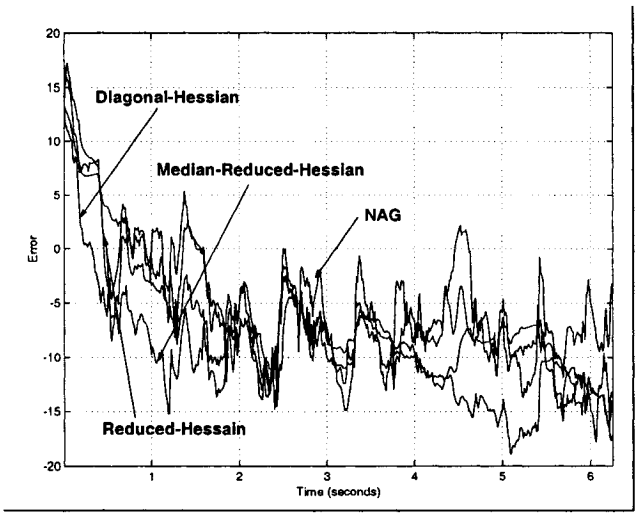

Figure 6.36: **Hessian-based algorithms simulation results for easy mixing situation: 1^{st} benchmark data**

In the above case, the Diagonal-Hessian algorithm performs slightly better than the rest of the on-line algorithms. However, it can be easily seen that all the algorithms, including the natural gradient, perform well in this case.

- **Difficult Mixing Scene**:
  The following figure shows the results for the difficult mixing scenario.
  In this case, the Median-Reduced-Hessian algorithm performs better than the rest of the on-line algorithms followed by the Reduced-Hessian. The Diagonal-Hessian algorithm comes third in convergence speed and steady state error achieved.

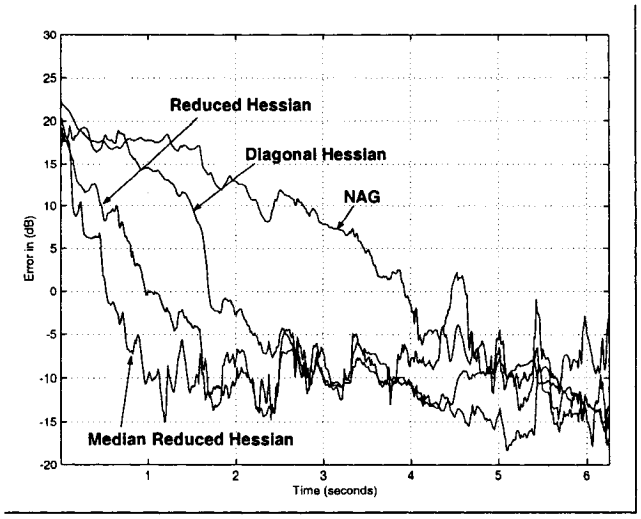

Figure 6.37: **Hessian-based algorithms simulation results for difficult mixing situation: 1^{st} benchmark data**

- **Average of 100 Random Mixing Situations**:
  The following figure shows the average of results for 100 different mixing scenarios.

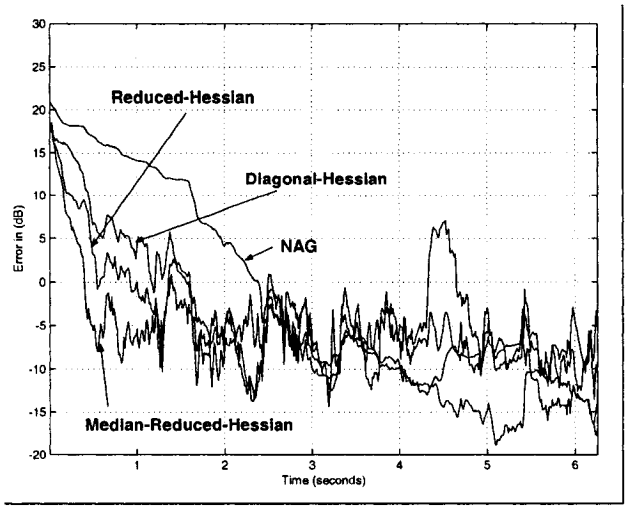

Figure 6.38: **Hessian-based algorithms average simulation results for 100 different mixing situations: 1^{st} benchmark data**

One can easily see from the above figures that the Hessian based algorithms are consistently the better on-line algorithms. They perform well in the different mixing/demixing scenarios considered.

- The second set of benchmarks consists of speech signals that are very similar in characteristics. All of the signals are male speech:

  1. Male-1.

2. Male-2.
3. Male-3.
4. Male-4.

The following are the results for this group of benchmarks

- **Easy Mixing Scene:**
  The following figure shows the results for the above easy mixing scenario.

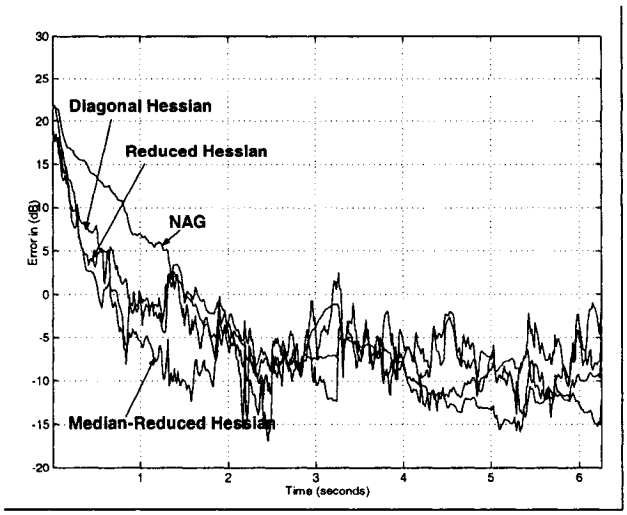

Figure 6.39: **Hessian-based algorithms simulation results for easy mixing situation: 2^{nd} benchmark data**

As can be seen in the figure, all algorithms perform well in the above easy mixing case. However, the Median-Reduced Hessian has a slightly better performance followed by the Reduced Hessian. The Diagonal Hessian comes third in convergence speed. All the Hessian based algorithms converge faster and with a smaller steady state error than the natural gradient algorithm.

- **Difficult Mixing Scene:**
  The following figure shows the results for the difficult mixing scenario.
  The figure shows that the performance of the natural gradient deteriorates in the difficult mixing case. On the other hand the Hessian-based algorithms produce consistently improved results. These improved convergence speeds for each of the algorithms are in accordance with their complexity meaning that the most complex algorithm, the Median-Reduced Hessian, produces the best convergence speed followed by the Reduced Hessian, while the Diagonal-Hessian algorithm produces a slower convergence while being very simple, computation wise.

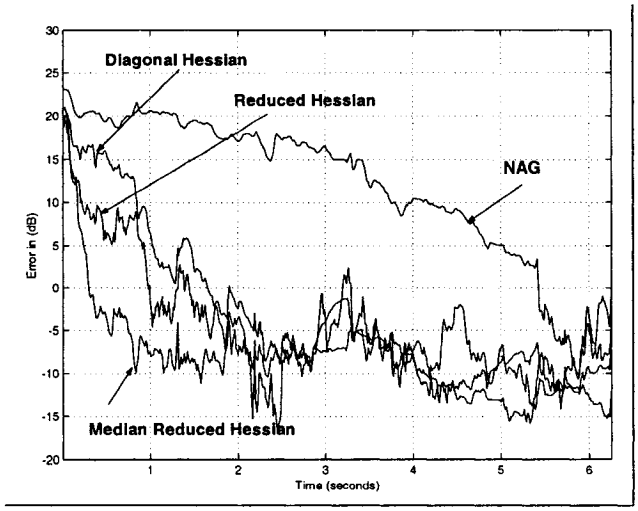

Figure 6.40: **Hessian-based algorithms simulation results for difficult mixing situation: 2^{nd} benchmark data**

- **Average of 100 Random Mixing Situations**:
  The same procedure of random test setup is performed in this case. 100 different mixing situations are evaluated and averaged to present an idea of how well each of the algorithms under inspection perform. The following figure shows the results.

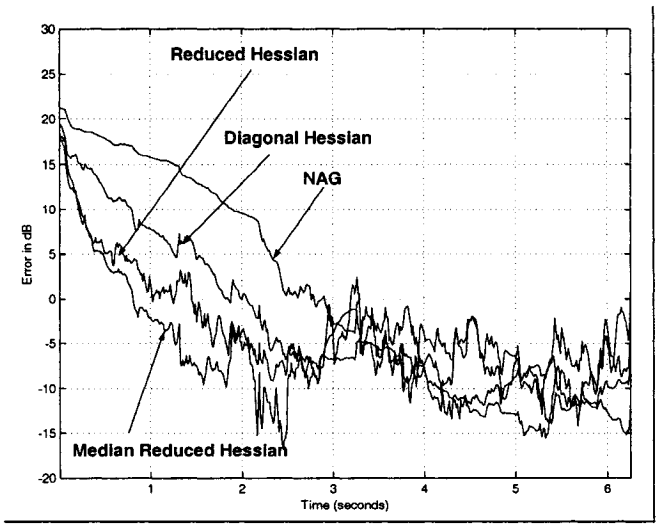

Figure 6.41: **Hessian-based algorithms average simulation results for 100 different mixing situations: 2^{nd} benchmark data**

Again, with the second group of benchmark files, the proposed algorithms provide a better convergence speed and can reach the same steady state error as the natural gradient.

- The last test is performed using four signals of the benchmark package **10halo.mat**. Again the same mixing/demixing scenarios were considered.

- **Easy Mixing Scene:**
  The following figure shows the results for the above easy mixing scenario. In this

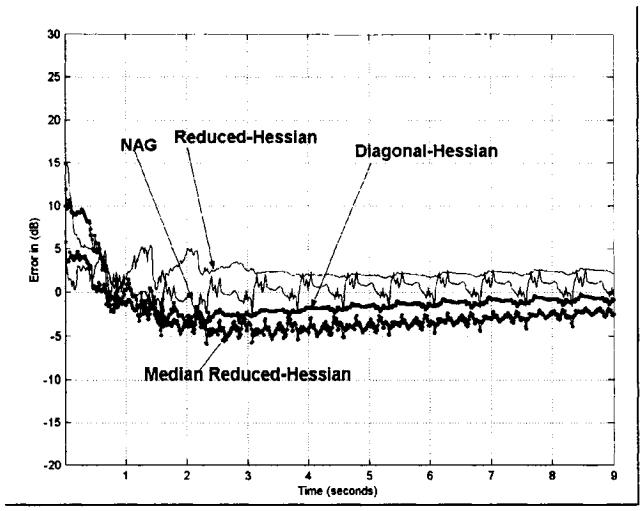

Figure 6.42: **Hessian-based algorithms simulation results for easy mixing situation: 3^{rd} benchmark data**

situation the Reduced-Hessian algorithm converges the fastest among all the on-line algorithms inspected. However, the steady state error of the reduced-Hessian algorithm is the highest. The Median-Reduced Hessian converges faster than both the Diagonal-Hessian and the natural gradient while providing the smallest steady state error.

- **Difficult Mixing Scene:**
  The following figure shows the results for the difficult mixing scenario.

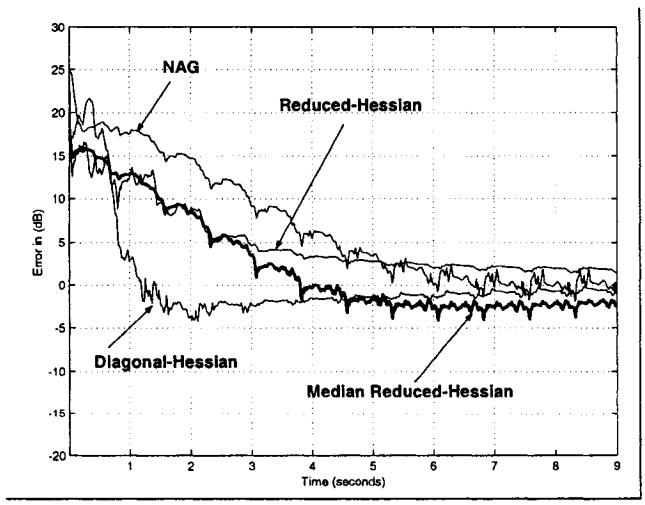

Figure 6.43: **Hessian-based algorithms simulation results for difficult mixing situation: 3^{rd} benchmark data**

In the above difficult mixing situation, the Diagonal-Hessian algorithm provides the

fastest convergence rate with a small steady state error. The Median Reduced Hessian algorithm provides the smallest steady state error among all the algorithms inspected while converging at a similar rate to the Reduced-Hessian algorithm. The following test provides a more decisive ranking of the algorithms based on the average of 100 different mixing scenarios.

- **Average of 100 Random Mixing Situations**:
  The following figure shows the average of the results for 100 different mixing scenarios, randomly generated.

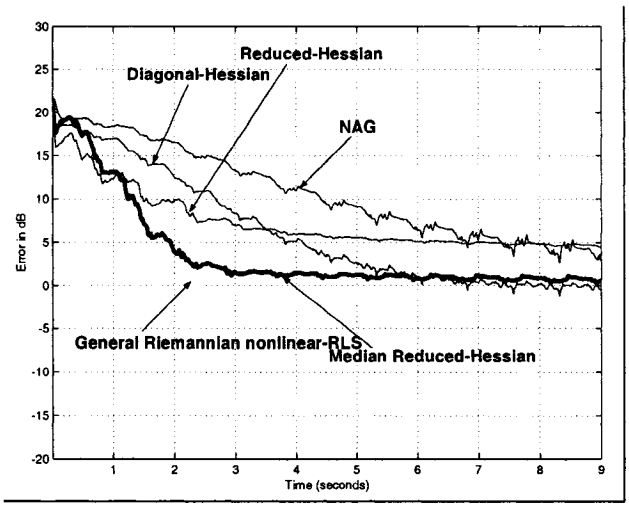

Figure 6.44: **Hessian-based algorithms average simulation results for 100 different mixing situations: 3^{rd} benchmark data**

As can be seen from the above figure, the performance of the Hessian-based algorithms in the case of highly correlated benchmark data is consistently better than the natural gradient. The Median-Reduced Hessian algorithm still provides the best convergence speed along with the smallest steady state error. With this type of test data the Diagonal-Hessian algorithm performs very well, better than the Reduced Hessian algorithm, providing a smaller steady state error.

As can be seen from all the above cases and benchmarks, in general, the Median-Reduced-Hessian algorithm provide the fastest convergence compared to all other on-line algorithms. This improved performance is slightly better than the Reduced-Hessian algorithm, which comes as second in performance for the Hessian-based algorithms. Both the Median-Reduced Hessian and the Reduced Hessian are very similar in implementation and have the same order of computational complexity. The Diagonal-Hessian algorithm performs slightly worse compared to to these two algorithms. However, the Diagonal-Hessian algorithm has the advantage of having a very simple computational load and is in the same order of the natural gradient.

6.8.4 Simulation Results for Hessian-Stiefel Algorithms

The concept of Hessian-based algorithms was applied to the pre-whitened mixtures case, i.e. the Stiefel manifold in section 5.13. This rather heuristic method helped develop algorithms that are not computationally expensive, yet providing noticeably improved performance. The list of algorithms developed in this category includes:

- **Reduced Hessian-Stiefel Algorithm**: This algorithm is presented in section 5.10.
- **Diagonal Hessian-Stiefel Algorithm**: The most computationally optimized algorithm in this collection of algorithms is derived in details in section 5.12.

The algorithms have been tested with supergaussian data. Sample results of the simulation tests were presented in Chapter 5. The two algorithms developed in this family are compared to the natural gradient algorithm on the Stiefel manifold:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \mu_{NAG}[\mathbf{y}g(\mathbf{y})^T - g(\mathbf{y})\mathbf{y}^T]\mathbf{W}(t-1) \tag{6.55}$$

The numerical choices for parameters of the algorithms are presented in the following table. The test data is the supergaussian benchmark data presented in 6.4. The test has been

Table 6.9: **Parameters of Hessian-Stiefel algorithms**

Parameter	Value
$g(\mathbf{y})$	$\tanh(\mathbf{y})$
μ_{NAG}	0.001
$\mu_{Reduced-Hessian}$	0.08
$\mu_{diagonal-Hessian}$	0.06
$\beta_{Reduced-Hessian}$	starts at value 0.992 at $t = 0$ and increases to 0.999 at $t = 15000$
$\beta_{Diagonal-Hessian}$	starts at value 0.995 at $t = 0$ and increases to 0.999 at $t = 25000$

performed using the three different acoustic data sets.

- The first set of benchmarks consists of various sound signals:
  1. Male-1.
  2. Male-2.
  3. Female-1.
  4. Music-1.

The following are the results for this first group of benchmarks and different mixing/demixing scenarios:

- **Easy Mixing Scene**:
  The following figure shows the results for the above easy mixing scenario.

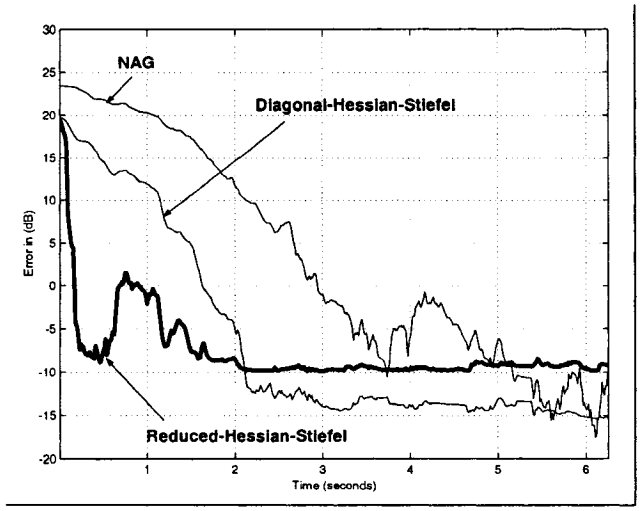

Figure 6.45: **Hessian-Stiefel algorithms simulation results for easy mixing situation: 1^{st} benchmark data**

In the above easy mixing situation the Reduced-Hessian-Stiefel algorithm provides a faster convergence than both the Diagonal-Hessian-Stiefel algorithm and the natural gradient algorithm. On the other hand, the Diagonal-Hessian-Stiefel algorithm provides the smallest steady state error.

- **Difficult Mixing Scene**:
  The following figure shows the results for the difficult mixing scenario.

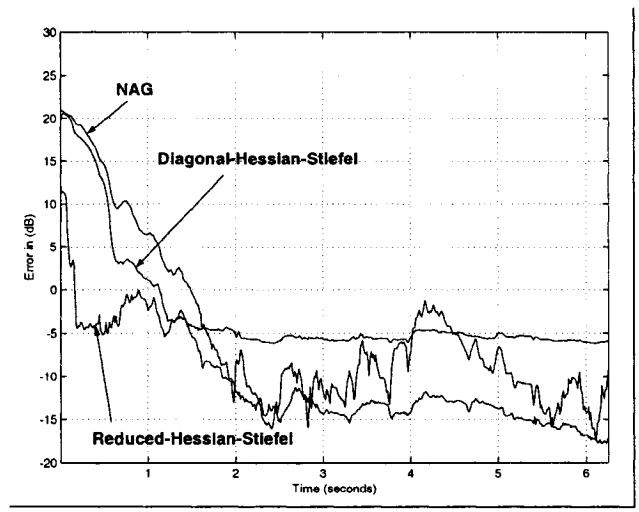

Figure 6.46: **Hessian-Stiefel algorithms simulation results for difficult mixing situation: 1^{st} benchmark**

In the difficult case, the Reduced Hessian-Stiefel algorithm still converges faster than the two other algorithms. However, it provides a higher steady state error. The Diagonal-Hessian-Stiefel algorithm converges slightly better than the natural gradient to the same steady state error.

- **Average of 100 Random Mixing Situations**:
  The following figure shows the average of the results for 100 random mixing scenarios. On average, the Reduced Hessian-Stiefel algorithm can achieve a faster convergence speed than the rest of the algorithms while converging to similar steady state error. The Diagonal-Hessian-Stiefel algorithm comes second in convergence speed but still much faster to converge compared to the natural gradient algorithm.

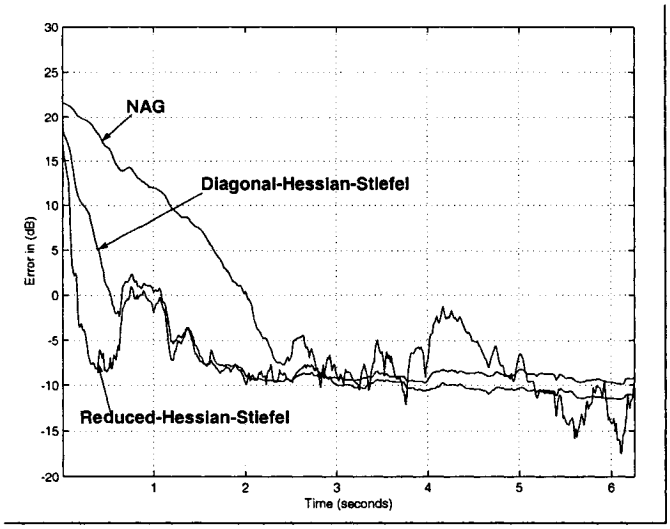

Figure 6.47: **Hessian-Stiefel algorithms average simulation results for 100 different mixing situations: 1^{st} benchmark data**

One can easily see from the above figures that the Hessian based algorithms on the Stiefel manifold are consistently the better on-line algorithms. They perform well in the different mixing/demixing scenarios considered.

- The second set of benchmarks consists of the male speech signals:

  1. Male-1.
  2. Male-2.
  3. Male-3.
  4. Male-4.

The following are the results of this group of benchmarks.

- **Easy Mixing Scene**:
  The following figure shows the results for the above easy mixing scenario.

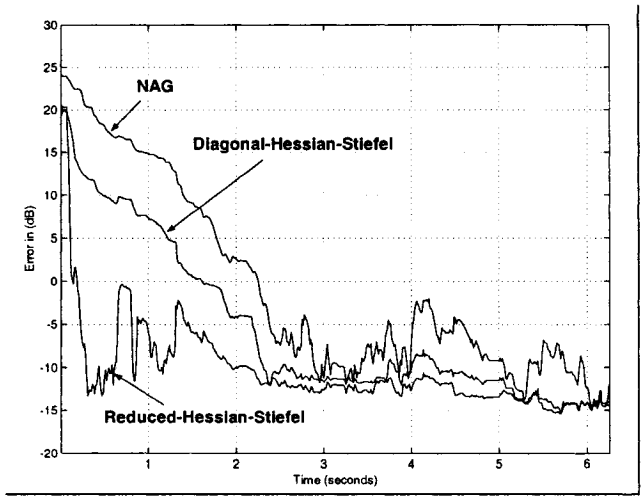

Figure 6.48: **Hessian-Stiefel algorithms simulation results for easy mixing situation: 2^{nd} benchmark data**

In the above easy mixing situation, the Reduced-Hessian Stiefel algorithm works well providing both the fastest convergence speed and a slightly smaller steady state error than the natural gradient. The Diagonal-Hessian Stiefel algorithm converges slower than the Reduced-Hessian Stiefel algorithm, with the same steady state error.

- **Difficult Mixing Scene**:
  The following figure shows the results for the difficult mixing scenario.

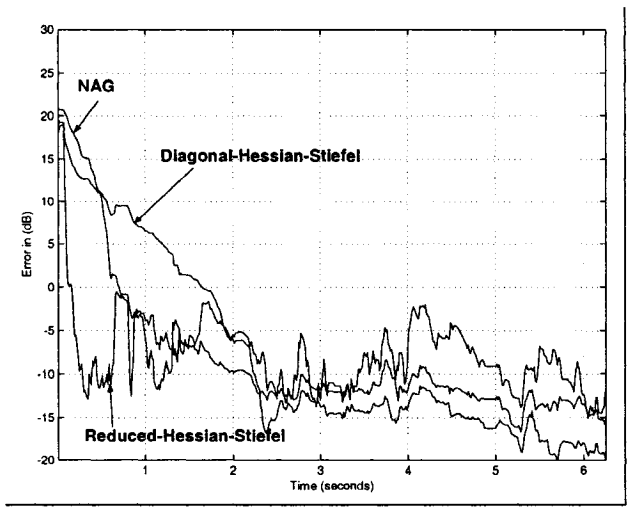

Figure 6.49: **Hessian-Stiefel algorithms simulation results for difficult mixing situation: 2^{nd} benchmark data**

In the above difficult mixing situation, the Diagonal-Hessian Stiefel algorithm converges slower than the natural gradient algorithm, to the same steady state error. On the other hand, the Reduced-Hessian Stiefel algorithm works well, providing both a fast convergence speed and the same steady state error as the natural gradient.

- **Average of 100 Random Mixing Situations**:
  The following figure shows the average results for the 100 different mixing scenarios.

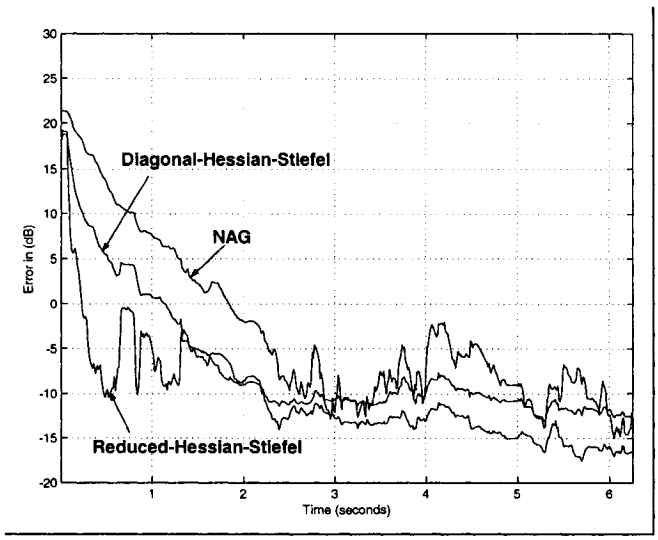

Figure 6.50: **Hessian-Stiefel algorithms average simulation results for 100 different mixing situations: 2^{nd} benchmark data**

Again, with the second group of benchmark files, the proposed algorithms provide, on average, better convergence speed and can reach the steady state error achieved by the natural gradient algorithm.

6.9 Comparison Between Nonlinear-RLS and the Hessian-based Algorithms for the General Riemannian Manifold

In the previous section (with the exception of Subsection 6.8.4) the simulation results of several algorithms for the non-prewhitened mixtures BSS, i.e. general Riemannian manifold, were presented. Among those algorithms two different approaches that worked successfully with speech/audio data have been presented, namely the general Riemannian nonlinear RLS algorithm and the Hessian-based algorithms. It would be useful to provide a practical comparison between the performance of the two approaches. In the following, both algorithms are tested with the three benchmarks considered for the speech/audio data. The test performed is the average of 100 different runs with 100 different mixing matrices $\mathbf{A}$ that were generated randomly for each of the 100 runs.

- The first set of benchmarks consists of various sound signals:
  1. Male-1.
  2. Male-2.

3. Female-1.
4. Female-2.

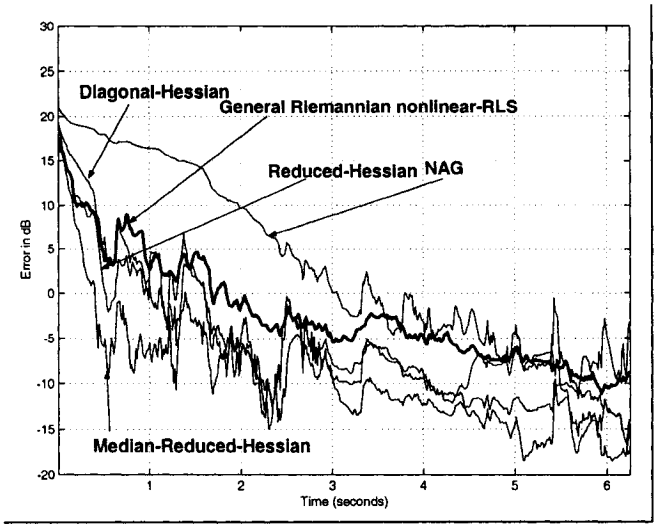

Figure 6.51: **Hessian-based algorithms vs. General Riemannian Nonlinear-RLS algorithm: 1^{st} benchmark data**

For the first benchmark data the nonlinear-RLS algorithm provides a slower convergence speed compared to the Median-Reduced Hessian algorithm. Adding this inferior convergence behavior to the fact that the nonlinear-RLS algorithm requires more computations, it can be easily seen that Hessian-based algorithms are more suitable for this type of benchmarks.

- The second set of benchmarks consists of speech signals that are very similar in characteristics. All of the signals are male speech:

  1. Male-1.
  2. Male-2.
  3. Male-3.
  4. Male-4.

The following figure shows the average result for 100 different random runs.

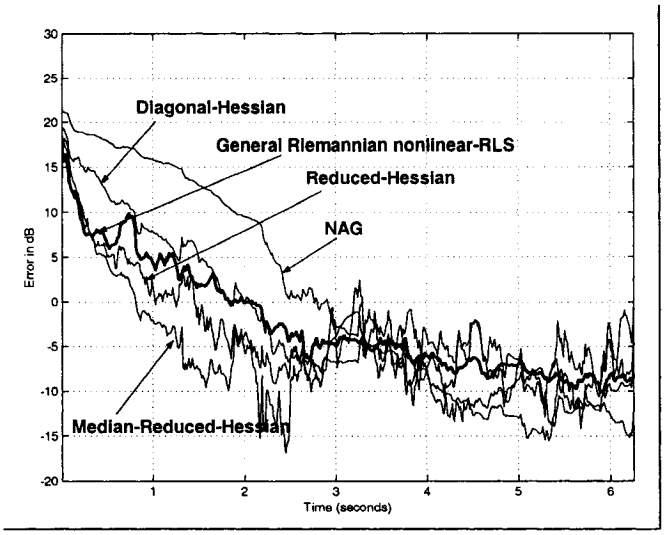

Figure 6.52: **Hessian-based algorithms vs. General Riemannian Nonlinear-RLS algorithm: 2^{nd} benchmark data**

As can be seen in the above figure, representing the average mixing scenario, the Median-Reduced Hessian algorithm produces the best performance both in convergence speed and steady state error. The general-Riemannian nonlinear RLS algorithm converges fast initially, at faster speed than the Median-Reduced Hessian algorithm, however it slows down and comes third in overall performance, following the Reduced Hessian algorithm.

- The last test is performed using four signals of the benchmark package **10halo.mat**.

The following figure shows the average result for 100 different random runs.
For the third benchmark data the nonlinear-RLS algorithm provides a similar convergence speed as the Median-Reduced Hessian algorithm. The nonlinear-RLS algorithm converges to a sliglty smaller steady state error than the Median-Reduced Hessian algrithm. However, this slightly improved peroformance comes at the cost of a considerably higher computational load.
One can easily see from the figures that the Hessian based algorithms provide similar or better convergence speed compared to the Nonlinear-RLS algorithm. The improved performance of the Hessian-based algorithms combined with their smaller complexity requirements compared with the Nonlinear-RLS algorithm gives the Hessian-based family of algorithms the advantage over the Nonlinear-RLS algorithm. However, both families stem from two different concepts that are not correlated and thus it is possible that in a particular mixing situation or for another type of data, the nonlinear-RLS algorithm might perform better than the Hessian-based algorithm.

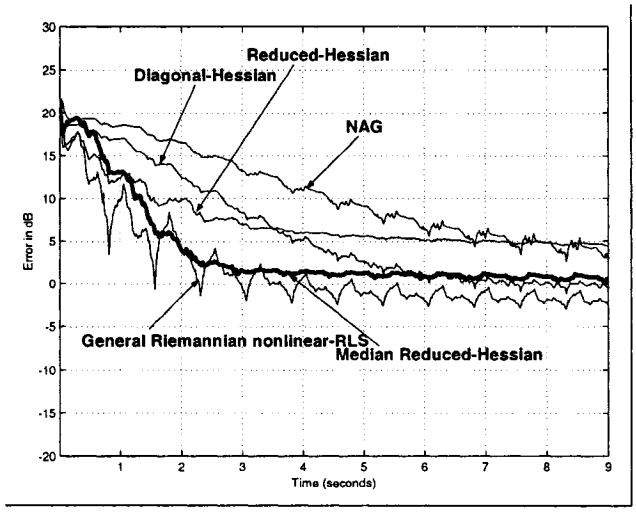

Figure 6.53: **Hessian-based algorithms vs. General Riemannian Nonlinear-RLS algorithm: 3^{rd} benchmark data**

6.10 Comparison Between Nonlinear-RLS and the Hessian-based Algorithms for the Stiefel Manifold

Section 6.7 and 6.8.4 present the simulation results of several algorithms for prewhitened mixtures BSS, i.e. Stiefel manifold. Among those algortihms two different approaches that worked successfully with speech/audio data have been presented, namely the RLS-BSS-Stiefel algorithm, with results presented in section 6.7.3 and the Hessian-based algorithms presented in section 6.8.4. It would be useful to provide a practical comparison between the performance of the two approaches. In the following, both families of algorithms are tested with two benchmarks considered for the speech/audio data. The test performed is the average of 100 different runs with 100 different mixing matrices $\mathbf{A}$ that are generated randomly for each of the 100 runs. The paramters used for each algorithm are the same values used to simulate the results in section 6.7.3 and 6.8.4.

- The first set of benchmarks consists of various sound signals:
  1. Male-1.
  2. Male-2.
  3. Female-1.
  4. Music-1.

The following figure shows the average of the results for the 100 different random mixing scenarios.

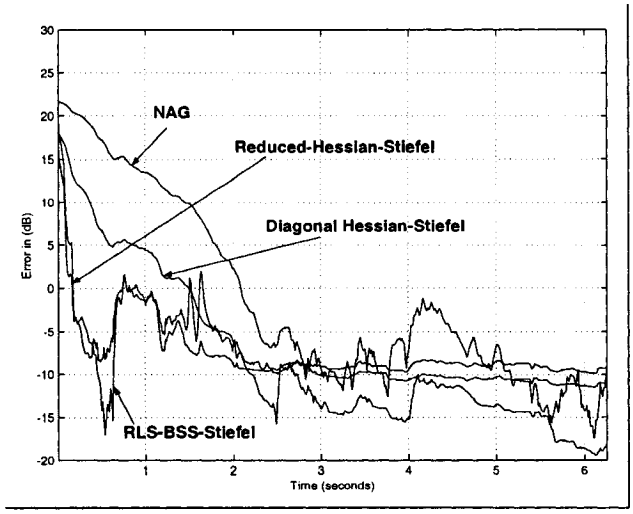

Figure 6.54: **Hessian-based algorithms vs. RLS-BSS algorithm for Stiefel manifold: 1^{st} benchmark data**

- The second test is performed using the male speech benchmark.
  1. Male-1.
  2. Male-2.
  3. Male-3.
  4. Male-4.

The following figure shows the average of the results for the 100 different random mixing scenarios.

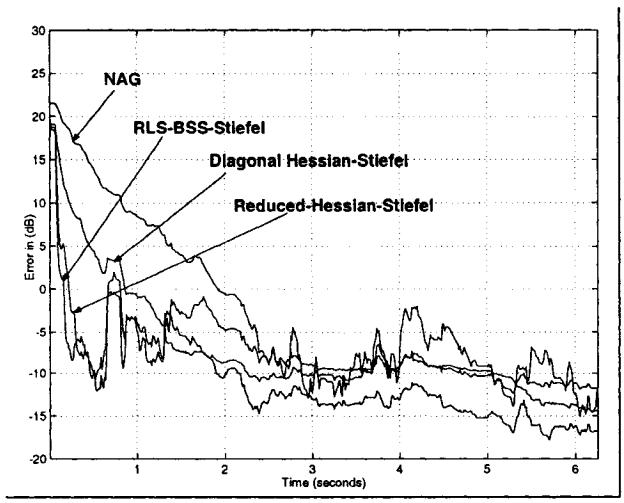

Figure 6.55: **Hessian-based algorithms vs. RLS-BSS algorithm for Stiefel manifold: 2^{nd} benchmark data**

One can see from Figures (6.54) and (6.55) that the nonlinear RLS based algorithm for the Stiefel manifold, i.e. RLS-BSS-Stiefel, always produces the smallest steady state error while maintaining the fastest convergence speed. However, its slight improved performance over the Hessian-based algorithms may not justify its considerably larger computational load. It should also be noted that the RLS-BSS-Stiefel algorithm provides a consistent behavior over different mixing situations and is not biased towards certain numerical values of the mixing matrices or the type of data involved in the mixing situation.

6.11 Conclusion

This chapter presented a more complete set of simulation results for the algorithms developed in the thesis. The chapter started with presenting the test environment, the required assumptions about the mixing process and the mixing/demixing models for both the pre-whitened and the non pre-whitened mixtures cases. The benchmark data sets used in the simulations were also presented and categorized into sub-gaussian data and super-gaussian data. Each of the algorithms was tested in three different mixing scenarios, easy mixing, difficult mixing and the average of 100 different randomly generated mixing situations.
Results for the algorithms developed for the Stiefel manifold were first presented. These RLS-like algorithms have been presented in Chapter 4. They consists of two algorithms for blind extraction, the RLS1-BSE and the modified RLS2-BSE. The third algorithm in this category is the RLS-BSS-Stiefel, which is a blind signal separation algorithm developed for extracting all the sources at once.
The results of the algorithms developed for the general Riemannian manifold were then presented. These algorithms were developed in Chapter 5 and are based on three different approaches. First, the natural gradient-iterative inversion family are the algorithms developed based on the iterative inversion approach and consist of the LMS-1, LMS-2, LMS-RLS algorithms and the RLS-iterative inversion algorithm. The LMS-1 algorithm and the RLS-iterative algorithms have shown to be working well in the case of ill conditioned mixing matrices $\mathbf{A}$, when the number of sources N is less than the number of sensors M. The other two approaches developed for the general Riemannian manifold BSS work well with speech/audio highly nonstationary data, unlike the iterative inversion family of algorithms. The general Riemannian nonlinear-RLS algorithm extends the concept of nonlinear-RLS developed in the prewhitened case, i.e. the RLS-BSS-Stiefel algorithm, to the general Riemannian case. The general Riemannian nonlinear-RLS algorithm has been tested with different benchmarks and proved to be converging fast and producing a small steady state error. However, the algorithm is costly in computations. The last family of algorithms tested in this chapter are the Hessian-based algorithms. This family of algorithms require a smaller computational load than the nonlinear-RLS algorithm and provide a fast convergence speed. This family of algorithms has also been applied successfully to the Stiefel-manifold. The results of those algorithms have shown that they can provide a good performance at smaller computational costs compared to the RLS-BSS-Stiefel algorithm.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis explored the problem of blind signal separation in general. It started by the instantaneous mixture separation where several algorithms were presented and a general conclusion was that the information-theoretic based algorithms are the most reliable approaches to achieve separation. Chapter 2 briefly reviewed some more complex mixing situations like the noisy-ICA and the nonlinear mixing model and special mixing situations when the data are of special nature (time-series).
Chapter 2 then explored the case of convolved mixtures, where the auditory scene is captured in a real room and the signals get mixed with other instantaneous signals, scaled delayed versions of themselves along with scaled and delayed versions of other signals. The discussion then moved to the two approaches to solve the convolutive mixtures problem, namely time-domain and frequency-domain. The review concluded that frequency domain is the more suitable framework based on the possibility of extending instantaneous-algorithms to work in the frequency domain.
Chapter 3 then explored one of the most powerful mathematical tools employed in blind signal separation, namely, optimization on the Riemannian manifold. The idea of intrinsic optimization has helped Amari et al. [6] to modify the gradient descent algorithm developed for blind separation based on maximum-likelihood to the dramatically more efficient natural gradient algorithm. Through our journey with the Riemannian manifold, the notions of geodesics, parallel translations and tangential spaces were explored along with a justification of the improved performance of the natural gradient, as it faithfully represents the steepest-descent direction better than the regular stochastic gradient. One of the most important cases of the Riemannian manifold was also investigated, namely the Stiefel manifold. The Stiefel-manifold represents the actual manifold for optimization in the case where pre-whitening has been applied to the mixtures.
After presenting the mathematical fundamentals for Riemannian space optimization, Chapter 4 addressed the next target which was to implement algorithms with improved performance based on the idea of intrinsic optimization (optimization along the actual manifold, Riemannian or Stiefel, instead of the Euclidean optimization). Next, a collection of algorithms was proposed to speed up the framework, with good results, both in separation quality

and convergence speed. A new cost function based on the Bussgang criterion is developed. An equivalence relationship between the gradient on the Stiefel manifold of most important on-line algorithms, namely Maximum-likelihood, Negentropy and nonlinear PCA along with the new cost function was derived under the orthogonality constraints. This equivalence relationship insured that whatever algorithm developed or to be developed using the natural gradient on the Stiefel-manifold will be a general solution to any of the previously mentioned cost functions.

Starting in section 4.5, the first algorithm was proposed to perform sequential separation of mixtures, i.e. blind extraction, producing both a fast convergence speed and a small steady state error. Simulation results showed the potential of this algorithm. The second algorithm, presented in section 4.6, is intended also for BSE but is implemented with a different structure that permits the use of a step size parameter and thus it handles the situation of nonstationary mixtures better. This algorithm was derived in two different ways, the first path to the derivation used an extrensic orthogonalization condition and evaluated the matrix at the optimal value, while the second way to derive the algorithm was structured based on the intrinsic properties of the Stiefel-manifold, yet it lead, at the optimal point, to the same algorithm. The algorithm and its successful results are summarized in the publication [33] by the author.

Another algorithm, derived in section 4.7, was presented. It can be described as a non-linear RLS extension to the natural gradient in the pre-whitened case. The performance of this algorithm is clearly superior to the original natural gradient algorithm. The algorithm was presented in publication [34] by the author.

Chapter 5 presents a collection of approaches to implement algorithms with improved performances for the general mixing situation, i.e. the general Riemannian manifold. Sections 5.3, 5.4 and 5.5 were devoted to the first approach based on generalizing the concept of iterative inversion successfully used for BSS in [1]. A family of algorithms was derived based on this generalization with the flexibility to adjust the choices of the autocorrelation matrix $\mathbf{R}_{ss}$ to either match already existing algorithms or provide new updates with advantageous qualities such as the LMS-1 algorithm, which provides robust performance in the case of under-determined mixing situations.

Another approach considered in Chapter 5 is the nonlinear RLS algorithm. The algorithm is successfully implemented in section 5.7 under the name "General Riemannian nonlinear RLS" algorithm. It is shown to provide faster convergence when tested with speech and audio signals.

The rest of Chapter 5 was devoted to second order algorithms on Riemannian manifolds. First, the form of the Hessian matrix on the Lie algebra space of the general Riemannian manifold was calculated. Realistic approximations were then applied to the Hessian matrix to develop two different approximated formulas for the Hessian matrix with different computational loads. Section 5.10 presented the "Reduced Hessian" and the "Median Reduced Hessian" algorithms. Section 5.11 presented the "Diagonal Hessian matrix" algorithm. Sample results showing the improved performance of the algorithm were presented in section 5.12. Section 5.13 introduced a rather heuristic extension of the "Diagonal Hessian" and the "Reduced Hessian" algorithms to the Stiefel manifold along with sample results showing, empirically, the success of this extension.

Lastly, Chapter 6 was dedicated to the presentation of a complete set of results for the

algorithms developed in the thesis. The chapter started with laying out the test setup environment, conditions and the benchmark data used. Results for each algorithm were then presented in different mixing situations along with comparisons with other algorithms.

7.2 Future Work

This study in the field of blind signal separation and Riemannian-based optimization has stimulated a diversity of new ideas that will potentially provide substantial additions to the research done in this thesis. In the following, a collection of possible objectives are considered for future work by other researchers.

1. One objective that might be considered, or be of interest to other researchers, is to investigate the extension of the new algorithms already developed to the convolutive multi-channel blind signal separation.
   A brief review of the convolutive mixtures has been presented in Chapter 2 where blind signal separation in the frequency domain faces the challenges of permutation along with separately adapting a de-mixing matrix $\mathbf{W}_f$ for different frequency bins, which may result in different convergence speed for different bands. Although this extension is possible, it requires a more in depth study of circulant matrices and projective methods of circular convolution into the frequency domain. Special attention should be paid to the filter sizes of the inputs, the demixing matrices and the output. A powerful permutation algorithm should also be selected with extensive care of its effect on the quality of the resulting mixtures (in the case of acoustic mixtures). These problems would require some additional study that is not directly related to the blind signal separation domain.

2. It would be interesting to explore the possible benefits of performing the de-mixing of instantaneous or convolutive mixtures in another domain such as the perceptual Bark-domain or combining the frequency with perceptual loudness and masking effects instead of the pure time or frequency domain. One advantage could be to eliminate unnecessary frequency bins and focusing on the frequency bands that affect the perceptual evaluation the most.

3. Speech enhancement or de-noising schemes could also be an idea to explore. They could be performed as a pre-processing step in order to remove noise from the mixtures. This could possibly speed up or help the de-mixing operation.
   Speech enhancement could also be considered as a post-processing step to eliminate the cross-talk of other audio sources in each of the recovered signals, while the separation algorithm has not yet reached full convergence. This post-processing step may be challenging as both the coloured noise to suppress and the source signals to recover would be speech signals that may have similar statistics. This would require a powerful speech enhancement algorithm that can successfully remove the cross talk without deteriorating the resulting quality of the source signal.

Bibliography

[1] S. Cruces-Alvarez, A. Cichocki, and L. Castedo-Ribas," An iterative inversion approach to blind source separation", *IEEE Transactions on Neural Networks*, Volume: 11 , Issue: 6 , Nov. 2000 Pages: 1423 - 1437.

[2] S. Amari, A. Cichocki, and H. H. Yang. "A new learning algorithm for blind source separation". In *Advances in Neural Information Processing 8*, pages 757-763. MIT Press, Cambridge, MA, 1996.

[3] S. Amari and J.-F. Cardoso, "Blind Source Separation - Semiparametric Statistical Approach", *IEEE Transaction on Signal Processing*, Vol. 45, No. 11, pp. 2692-2700, Dec. 1997.

[4] S. Amari, "Super-efficiency in blind source separation", *IEEE Proc. on Signal Processing*, Vol. 47, pp. 936-944, April 1997.

[5] S. Amari and S. C. Douglas. "Why natural gradient", In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Vol. II, pages: 1213-1216, Seatle, WA, May 1998.

[6] S. Amari, "Natural gradient works efficiently in learning", *Neural Computation*, Vol. 10, pp. 251-276, Jan. 1998.

[7] S. Amari, "Natural-gradient learning for over- and under-complete bases in ICA", *Neural Computation*,11 (8):1875-1883, November 1999.

[8] S. Amari, "Estimating Functions of Independent Component Analysis for Temporally Correlated Signals", *Neural Computation.* Vol. 12, pp. 2083-2107, 2000.

[9] S. Amari, T.-P. Chen and A. Cichocki. "Non-holonomic constraints in learning algorithms for blind source separation", *Neural computation*, Vol. 12, pp. 1463-1484, 2000.

[10] A. Bell, and T. Sejnowski, "An information-maximization approach for Blind Separation and Blind Deconvolution", *Neural Computation*, Vol.7, 1995, pp. 1129-1159.

[11] S. Bellini, Bussgang techniques for blind deconvolution and equalization. In S. Haykin, editor, *Blind Deconvolution*, pages 8-59, Prentice Hall, 1994.

[12] A. Belouchrani, K.K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "Second-order blind separation of correlated sources", In *Conference on Digital Sig. Processing*, pages 346-351, Cyprus 1993.

[13] D. Bertesekas, *Nonlinear Programming*, Second Ed., Athena Scientific, 1999.

[14] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, 1995.

[15] W. Boothby, *An Introduction to Differential Manifolds and Riemannian Geometry*, Revised Second Ed., Academic Press, 2002.

[16] M. Bouchard, "New recursive-least-squares algorithms for nonlinear active control of sound and vibration using neural networks ", *IEEE Transactions on Neural Networks*, Vol. 12, 2001, pp. 135 - 147.

[17] J.-F. Cardoso, "Eigen Structure of the fourth-order cumulant tensor with application to the blind source separation problem", In *Proc. IEEE Int. Conf on Acoustics, Speech and Signal Processing (ICASSP'90)*, pages 2655-2658, Albuquerque, New Mexico, 1990.

[18] J.-F Cardoso and A. Souloumiac, "Blind Beamforming for non-gaussian signals", *IEEE Proceedings- Part F*, Dec. 1993, Vol. 140, No. 6, pp. 362-370.

[19] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation", *IEEE Trans. on Signal Processing*, 44(12):3017-3030, 1996.

[20] J.-F. Cardoso, "Infomax and maximum likelihood for blind separation", *IEEE Signal Processing Lett*, Vol. 4, April 1997.

[21] J.-F. Cardoso, École Nationale Supérieure des Télécommunications, Paris, France, "ICA Central", *http://www.tsi.enst.fr/icacentral/*

[22] A. Cichoki and S. Amari. *Adaptive Blind Signal and Image Processing*, John Wiley & Sons Inc., New York, 2002.

[23] S. Choi, A. Cichocki, and S. Amari. "Fetal electrocardiogram data analysis via flexible independent component analysis", *In The* 4^{th} *Asia- Pacific Conference on Medical and Biological Engineering (ABMCPE'99)*, Seoul, Korea, 1999.

[24] A. Cichocki, R. Unbehauen, L. Moszczynski, and E. Rummert: "A new on-line adaptive learning algorithm for blind separation of source signals" , *Int. Symposium on Artificial Neural Networks, ISANN-94, Dec. 1994, Tainan, Taiwan*, pp. 406-411.

[25] P. Comon, "Independent Component Analysis, A New Concept?", *Signal Processing, Elsevier*, April 1994, Vol. 36, No. 3, pp. 287-314.

[26] P. Comon, C. Jutten and J. Hérnault, "Blind Separation of Sources, Part II: Problem Statement", *Signal Processing*, 13(4):883-898, 2001.

[27] S. Cruces and L. Castedo ,"Blind separation of convolutive mixtures: a Gauss-Newton algorithm", *Proceeding of IEEE workshop on Higher-Order Statistics*, 21-23 July 1997, pp.326 - 330.

[28] M. De Carmo, *Differential Geometry of Curves and Surfaces*, Second Ed., Prentice Hall, Upper Saddle River, 1976.

[29] S. C. Douglas, A. Cichocki, S.-I. Amari. "Multichannel blind separation and deconvolution of sources with arbitrary distribution", *IEEE Workshop on Neural Networks for Signal Processing*, pages 436-445, 1997.

[30] S. C. Douglas, "Self-stabilized gradient algorithms for blind source separation with orthogonality constraints", *IEEE Transactions on Neural Networks*, Vol. 11, 2000, pp. 1490-1497.

[31] A. Edelman et al., "The Geometry of Algorithms With Orthogonality Constraints", *SIAM Journal on Matrix Analysis and Applications*, Vol. 20, 1998, pp. 303-353.

[32] Elements Research home page , "Geodesic Computation", *http://www.elesoft.com/*.

[33] M. Elsabrouty, M. Bouchard, and T. Aboulnasr, "A New On-Line Negentropy-Based Algorithm for Blind Source Separation", *Proceedings of 46th IEEE International Midwest Symposium on Circuits and Systems* (paper 447D on CD-Rom), Cairo, Egypt, December 2003.

[34] M. Elsabrouty, T. Aboulnasr and M. Bouchard, "A Recursive Least-Squares Extension of the Natural Gradient Algorithm for Blind Signal Separation of Audio Mixtures", *Journal of the Canadian Acoustical Association (Proceedings of Acoustics Week in Canada 2004)*, Vol. 32, No. 3, pp. 136-137, Ottawa, Ontario, Oct. 2004.

[35] M. Ensecu, *Adaptive Methods for Blind Equalization and Signal Separation in MIMO Systems*, P.h.D thesis, Helsiniki University of Technology, Departement of Electrical and Communication Engineering, 2002.

[36] R. Fletcher, *Practical Methods of Optimization*, 2^{nd} Ed. John Wiley & Sons Inc., New York, 1987.

[37] D. R. Fuhrmann and B. Liu. "An iterative algorithm for locating the minimal eigenvector of a symmetric matrix". In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 45.8.1-4, 1984.

[38] S. L Gay and S. C. Douglas, "Normalized natural gradient adaptive filtering for sparse and nonsparse systems ", In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Vol. 2, pages:1405 - 1408, 2002.

[39] P. Georgiev, A. Cichocki, and S.-I. Amari. "On some extensions of the natural gradient algorithm". In *Proc. Third International Conference on Independent Component analysis(ICA 2001)*, pages:581-585, San Diego, USA, 2001.

[40] M. Girolami, "An alternative perspective on adaptive Independent Component Analysis algorithms", *Neural Computation*, 10(8), 1998.

[41] F. J. Gonzalez-Serrano, "Improving stability in blind source separation with stochastic median gradient", *Electronics Letters*, Volume 36, Issue 19, 14 Sept. 2000 Page(s):1662 - 1664.

[42] S. Gudmundsson, *An Introduction to Riemannian Geometry.* Lecture notes, version.1.235, Lund University, Sweden, 2004.

[43] S. Haykin et al., *Neural Networks: A Comprehensive Foundation*, second Ed., Prentice Hall, Upper Saddle River, 1999.

[44] S. Haykin, *Adaptive Filters Theory*, Fourth Ed., Prentice Hall, Upper Saddle River, 2000.

[45] S. Haykin et al., *Unsupervised Adaptive Filtering*, vol.1 *"Blind signal Separation"*, John Wiley & Sons Inc., New York, 2000.

[46] Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Network Research Center, "The Fast-ICA Package for MATLAB" *http://www.cis.hut.fi/projects/ica/fastica/*

[47] Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Network Research Center, "ICA link collection" *http://www.cis.hut.fi/projects/ica/book/links.html*

[48] Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Network Research Center, "ICA Sources and Demonstration" *http://www.cis.hut.fi/projects/ica/cocktail/cocktailen.cgi*

[49] M.R. Hestenes and E. Stiefel. "Methods of conjugate gradients for solving linear systems", *Jour. Res. Nat. Bur. Stand.*. 49:409-436, 1952.

[50] G. Hinton and D. Van Camp. "Keeping Neural Networks simple by minimizing the description length of the weights". In *Proc. of 6th annual ACM Conf. on Computational Learning Theory,* pages 5-13, Santa Cruz, CA, 1993.

[51] A. Hyvärinen, "New Approximation of differential entropy for independent component analysis and projection pursuit". In *Advances in Neural Information Processing 10*, pages 273-279. MIT Press, Cambridge, MA, 1998.

[52] A. Hyvärinen. "Indpendent component analysis in the presence of gaussian noise by maximizig joint likelihood", *Neurocomputing*, Vol. 22, pages 49-67, 1998.

[53] A. Hyvärinen, "Survey on Independent Component Analysis", *Neural Computing* Surveys 2:94-128, 1999.

[54] A. Hyvärinen, " Fast and Robust Fixed-Point Algorithms for Independent Component Analysis", *IEEE Transactions on Neural Networks* 10(3):626-634, 1999.

[55] A. Hyvärinen. "An alternative approach to infomax and independent component analysis", In *Neurocomputing*, 44-46(C):1089-1097, 2001.

[56] A. Hyvärinen et. al., *Independent Component Analysis*, John Wiley & Sons Inc., New York, 2001.

[57] A. Hyvärinen, "*Survey on Independent Component Analysis*, Helsinki University of Technology. http://www.cis.hut.fi/aapo/papers/NCS99web/.

[58] A. Hyvärinen, "Complexity Pursuit: Separating interesting components from time-series". *Neural Computation*, 13(4):883-898, 2001.

[59] "ICALAB for Signal Processing", Riken Institute for Brain Studies, Japan *http://www.bsp.brain.riken.go.jp/ICALAB/ICALABSignalProc/benchmarks/data/*.

[60] ITU-T G.711 Annex A: "A High Quality Low complexity Algorithm for Packet Loss Concealment with G.711", *International Telecommunication Union*, November 2000.

[61] M. C. Jones and R. Sibson, "What is projection pursuit?", *J. of the Royal Statistics Society, ser. A*, 150: 1-36, 1987.

[62] C. Jutten. *Calcul neuromimétique et traitement du signal, analyse en composantes indépendentes*, P.h.D thesis, INPG, Univ. Grenoble, 1987, (in French).

[63] C. Jutten and J. Hérault, "Blind Separation of Sources, part I: An Adaptive Algorithm Based on Neuromimetic Architecture", *Signal Processing,* 24:1-10, 1991.

[64] M. Kendall and A. Stuart. *The Advanced Theory of Statistics*, Charles Griffen & Company, 1958.

[65] P. Kidmose, *Blind Separation of Heavy Tail Signals*, Ph.D. thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Konges Lengby, 2001.

[66] T. Kohonen, *Self Organizing Maps*, Springer, 1995.

[67] R. H. Lambert and C. L. Nikias, Blind deconvolution of multipath mixtures, In S. Haykin, editor, *Unsupervised Adaptive Filtering , Volume I: Blind Source Separation*, pages 377-436. John Wiley & Sons, 2000.

[68] T.-W. Lee et al. "Blind source separation of more sources than mixtures using overcomplete representations", *IEEE Signal Processing Letters*, 4(5), 1999.

[69] T.-W. Lee et al. "A Unifying Information-Theoretic Framework for Independent Component Analysis", *International Journal of Computer and Mathematics with Applications*, 4(12), 1999.

[70] J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Englewood Cliffs NJ: Prentice-Hall, 1981.

[71] D. G. Luenberger, *Linear and Nonlinear Programing*, Addison-Weasly, Mass., 1994.

[72] Massachusetts Institute of Technology, MIT Media Laboratory, "ICA Benchmark" *http://www.media.mit.edu/*

[73] Massachusetts Institute of Technology, MIT Media Laboratory, "ICA Benchmark" *http://sound.media.mit.edu/ica-bench/sources/*

[74] K. Matsuoka and M. Kawamoto, "A neural net for blind separation of nonstationary signals". *Neural Networks*, 8 (3):411-419, 1995.

[75] M. McKeown et al., "Spatially independent activity patterns in functional magnetic resonance imaging data during the stroop color-naming task", *Proceedings of the National Academy of Science*, 95-803-810, 1998.

[76] Y. Ji-Min et al., "Adaptive Blind Separation with an Unknown Number of Sources", *Neural Computation*, vol. 16, pp. 16411660, Dec. 2004.

[77] N. Mitianoudis, *Audio Source Separation using Independent Component Analysis*, Ph.D. thesis, Department of Electronic Engineering, Queen Mary, University of London, London, England,2004.

[78] L. Molgedey and S. C. Schuster, "Separation of a mixture of independent sources using time-delayed correlation", *Physical Review Letters*, 72(23):3634-3637, 1994.

[79] E. Moreau, "A generalization of joint-diagonalization criteria for source separation", *IEEE Transactions on Signal Processing*, Volume: 49 , Issue: 3 , March 2001 Pages:530 - 541.

[80] F. Morgan, *Riemannian Geometry: A Beginner's Guide*, Ed. Jones & Bartlett Publishers Int., London, England, 1993.

[81] Y. Nishimori, "Learning algorithm for independent component analysis by geodesic flows on orthogonal group ", *IEEE International Joint Conference on Neural Networks*, 10-16 July 1999, vol.2, pp.933-938.

[82] B. A. Olshausen and K. J. Millman, Learning sparse codes with a mixture-of-gaussians prior. In *Advances in Neural Information Processing Systems*, volume 12, pages 841-847. MIT Press, 2000.

[83] P. Pajunen, A. Hyvärinen, and J. Karhunen. Nonlinear blind source separation by self-organzing maps. *In Proc. Int. Conf. on Neural Information Processing*, pages 1207-1210, Hong Kong, 1996.

[84] P. Pajunen and J. Karhunen. "Least-squares methods for blind source separation based on nonlinear PCA". *Int. J. of Neural Systems*, 8(5-6):601-612, 1998.

[85] W. H. Press et al. "Numerical Recipes in C++ The Art of Scientific Computing". Second Ed., Cambridge University Press, 2002.

[86] E. Oja, "The nonlinear PCA learning rule and signal separation mathematical analysis ", Lab. Comput. Inform. Sci., Helsinki Univ. Technol., Espoo, Finland, Tech. Rep. A26, Aug. 1995.

[87] RIKEN Brain Science Institute. Lab for Mathematical Neuroscience, *http://www.brain.riken.jp/labs/mns/*.

[88] D. Schobben et al., "Evaluation of Blind Signal Separation Methods" , *Proceedings Int. Workshop Independent Component Analysis and Blind Signal Separation*, Aussois, France, January 11-15, 1999.

[89] D. Schobben, *Real-Time Adaptive Concepts in Acoustics: Blind Signal Separation and Multichannel Echo Cancellation*, Kluwer Academic Publishers, 2001.

[90] S. Smith, *Geometric Optimization for Adaptive filtering*, Ph.D. thesis, Harvard University, Cambridge, MA, 1993.

[91] S. Smith, *Optimization Techniques on Riemannian Manifolds*, in Fields Institute Communications, Vol. 3, AMS, Providence, RI, 1994, pp.113-146.

[92] L. Tong, V.C. Soon, Y. Inouye, Y. Huang and R. Liu, "Wavefrom preserving blind estimation of multiple sources", In *Proc. IEEE Conference on Decision and Control,* pages 2388-2393, Vol.3, Brighton, UK, 1991.

[93] J. R. Treichler et al. *Theory and Design of Adaptive Filters*, Prentice Hall Inc., New Jersey, 2001.

[94] University of Vienna, "Introduction to Computational Physics", *http://www.ap.univie.ac.at/users/ves/cp0102/dx/dx.html.*

[95] E. Weisstein's world of mathematics, "Mathworld" *http://mathworld.wolfram.com/.*

[96] B. Widrow, S. D. Stearns, *Adaptive Signal Processing* , First Ed., Prentice Hall, Upper Saddle River, 1985.

[97] H. H. Yang and S. Amari. "Adaptive on-line learning algorithms for blind separation: maximum entropy and minimum mutual information", *Neural Computation*, 9(7):1457-1482, Oct. 1997.

[98] H. H. Yang, "Serial updating for blind separation derived from the method of scoring", *IEEE Trans. Signal Processing*, Vol. 47, pp. 22792285, Aug. 1999.

[99] D. Yellin, E. Weinstein, "Multi-channel signal separation based on cross bispectra", In *IEEE Signal Processing Workshop on Higher-Order Statistics*, pages: 270-274, 1993.

[100] X.-L. Zhu and X.-D. Zhang , "Adaptive RLS Algorithm for Blind Source Separation Using Natural Gradient", *IEEE Signal Processing Letters*, Vol. 9, pp. 432435, Dec. 2002.