# ADVANCES IN SPARSE SIGNAL ANALYSIS WITH APPLICATIONS TO BLIND SOURCE SEPARATION AND EEG/MEG SIGNAL PROCESSING.

# ADVANCES IN SPARSE SIGNAL ANALYSIS WITH APPLICATIONS TO BLIND SOURCE SEPARATION AND EEG/MEG SIGNAL PROCESSING.

By

Nasser Mourad, B.Sc., M.Sc.

2009

A Thesis

Submitted to the Department of Electrical & Computer Engineering

and the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree of

Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2009) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario

| | |
|---|---|
| TITLE: | Advances in Sparse Signal Analysis with Applications to Blind Source Separation and EEG/MEG Signal Processing. |
| AUTHOR: | Nasser Mourad<br>B.Sc. (Communications and Electronics Engineering),<br>M.Sc. (Communications and Electronics Engineering),<br>Assiut University, Assiut, Egypt. |
| SUPERVISORS: | James P. Reilly, Professor |
| NUMBER OF PAGES: | xxiv, 208 |

**Dedications**

*To my parents,*

*my siblings,*

*my wife Sahar,*

*and my beloved daughters Maryam and Janna.*

# Abstract

The focus of this thesis is on the utilization of the *sparsity* concept in solving some challenging problems, e.g., finding a unique solution to the under–determined linear system of equations in which the number of equations is less than the number of unknowns. This concept is extended to the problem of solving the under–determined blind source separation (BSS) problem in which the number of source signals is greater than the number of sensors and both the mixing matrix and the source signals are unknowns. In this respect we study three problems:

1. Developing some algorithms for solving the under–determined linear system of equations for the case of a sparse solution vector.

   In this thesis we develop a new methodology for minimizing a class of non–convex (concave on the non–negative orthant) functions for solving the afore mentioned problem. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. For a certain selection of the convex objective function, the existing class of algorithms called Iterative Re–weighted Least Squares (IRLS) can be derived from the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods. In this thesis we also propose a convex objective function that produces an algorithm that can converge to a sparse solution vector in significantly fewer iterations than the IRLS

algorithms.

2. Solving the under–determined BSS problem by developing new clustering algorithms for estimating the mixing matrix.

   The under–determined BSS problem is usually solved by following a two–step approach, in which the mixing matrix is estimated in the first step, then the sources are estimated in the second step. For the case of sparse sources, the mixing matrix is usually estimated by clustering the columns of the observation matrix. In this thesis we develop three novel clustering algorithms that can efficiently estimate the mixing matrix, as well as the number of sources, which is usually unknown. Numerical simulations verify the efficiency of the proposed algorithms compared to some well known algorithms that are usually used for solving the same problem.

3. Extraction of a desired source signal from a linear mixture of hidden sources when prior information is available about the desired source signal.

   There are many situations in which one is interested in extracting a specific source signal. The a priori available information about the desired source signal could be temporal, spatial, or both. In this thesis we develop new algorithms for extracting a desired *sparse* source signal from a linear mixture of hidden sources. The information available about the desired source signal, as well as its sparsity, are incorporated in an optimization problem for extracting this source signal. Four different algorithms have been developed for solving this problem. Numerical simulations show that the proposed algorithms can be used successfully for removing different kind of artifacts from real electroencephalographic (EEG) data and for estimating the event related potential (ERP) signal from synthesized EEG data.

# Acknowledgements

My first gratitude is to God, "*Allah*", the Most Merciful, the Most Compassionate. Without the help of *Allah*, none of the work in this thesis would be done. I also thank *Allah* for his guidance to select McMaster University and to select Dr. Jim Reilly to be my supervisor.

I would like to express my gratitude to Dr. Jim Reilly for his excellent supervision during my Ph.D. program. Dr. Reilly has helped me a lot in developing myself as an academic, and encouraged me to reach further. As a supervisor, the results in this thesis would not have been possible without his direction and insightful discussions. He has taught me many things, not least of which is clear technical writing. As a person, he has always been a father to his students, provided them with advice, and shown understanding at hard times. There are still many things to learn from Dr. Reilly, but, unfortunately, my Ph.D. program is reaching its end so quickly.

I would like also to extend my thanks to my supervisory committee members, Dr. H. deBruin, Dr. Bernhard Ross, Dr. Laurel Trainor, and Dr. T. Kirubarajan for their beneficial discussions, suggestions and for their contributions in refining the ideas in this thesis. I owe special thanks to Dr. Laurel Trainor for offering me the chance to join her lab and to gain experience in dealing with EEG data.

I do not find suitable words to express my feelings towards my wife Sahar. She left her family and our warm country, Egypt, and joined me in Canada just to support and to provide me with help. During the last four years, she has always shown patience,

perseverance and continuous understanding. I would also like to extend my gratitude to my daughters Maryam and Janna, even though they may not be aware of it. They have been providing me with daily portions of happiness that have had the greatest impact on developing this thesis.

I would also like to sincerely thank my father, Mourad, and my mother, Sekena, for their continuous help, support, and encouragement. My thanks go also to my siblings Zeinab, Mohammad, Tarek, Azza, and Salwa for supporting me all the way before and after graduation, and for encouraging me to excel in my studies.

I would like to thank the ECE department of McMaster university for the research facilities and healthy environment it provides for its students. I would like to thank our administrative and technical staff. In particular, I would like to thank Mrs. C. Gies, Mr. T. Greenlay and Mr. C. Coroiu for keeping our systems up and running.

I would like to thank all my friends and colleagues in the department for the continuous discussions that have certainly enhanced my scope of knowledge.

I would like to thank Dr. Bernhard Ross for his valuable discussions and for providing us with samples of MEG data that we have used in the simulations. Also I would like to thank Dr. Gary Hasey and Dr. Duncan MacCrimmon for providing us with samples of EEG data that we used the simulations.

Finally, I would like to thank the Egyptian Ministry of High Education for financially supporting this work.

# List of Acronyms

| | |
|---|---|
| BSE | Blind Source Extraction. |
| BSS | Blind Source Separation. |
| BSSE | Blind Sparse Signal Extraction. |
| cICA | Constraint Independent Component Analysis. |
| CP | Concentration Parameter. |
| CS | Compressed Sensing. |
| cSCA | Constrained Sparse Component Analysis. |
| ERP | Event Related Potential. |
| EEG | Electroencephalography. |
| HC | Hierarchical clustering. |
| ICA | Independent Component Analysis. |
| *i.i.d.* | Independent and Identically Distributed. |
| IRLS | Iterative Re – weighted Least Squares. |
| MAH | Modified Angular Histogram. |
| MAHSCE | Modified Angular Histogram followed by Sequential Cluster Extraction. |
| MEG | Magnetoencephalography. |
| MSE | Mean Square Error. |
| NMSE | Normalized Mean Square Error. |
| PCA | Principal Component Analysis. |
| RIP | Restricted Isometry Property. |

| | |
|---|---|
| SCA | Sparse Component Analysis. |
| SCE | Sequential Cluster Extraction |
| SIR | Signal to Interference Ratio. |
| SNR | Signal to Noise Ratio. |
| scSCA | Spatial Constraint SCA. |
| sucSCA | Support Constraint SCA. |
| SVD | Singular Value Decomposition. |
| tcSCA | Temporal Constraint SCA. |

# List of Common Notations

| | |
|---|---|
| $a$ | Scalar |
| $\boldsymbol{a}$ | Column vector |
| $\bar{\boldsymbol{a}}$ | Row vector |
| $\boldsymbol{A}$ | Matrix |
| $\boldsymbol{A}^T$ | Matrix transpose |
| $\boldsymbol{A}^\dagger$ | Pseudo inverse of the matrix$\boldsymbol{A}$ |
| $a[i]$ | the $i$th element of the vector $\boldsymbol{a}$ |
| $\boldsymbol{a}_i$ | the $i$th column of the matrix $\boldsymbol{A}$ |
| $\boldsymbol{a}^i$ | the $i$th row of the matrix $\boldsymbol{A}$ |
| $\|\cdot\|_{\ell_i}$ | the $i$th norm of a vector |
| $\nabla$ | the gradient operator |
| $\nabla^2$ | the Laplacian operator |

# Contents

**Abstract** v

**Acknowledgements** vii

**List of Acronyms** ix

**List of Common Notations** xiii

**1 Introduction** 1

1.1 Some Applications of Compressed Sensing . . . . . 3

1.1.1 EEG Source Imaging . . . . . 3

1.1.2 Single-pixel compressive digital camera . . . . . 6

1.1.3 Blind separation of more sources than sensors . . . . . 8

1.2 Contributions and Thesis Organization . . . . . 11

1.2.1 Chapter 2 . . . . . 11

1.2.2 Chapter 3 . . . . . 11

1.2.3 Chapter 4 . . . . . 12

1.2.4 Chapter 5 . . . . . 14

**2 Fundamentals of compressed sensing (CS)** 17

2.1 Problem formulation . . . . . 18

2.1.1 Compressed sensing . . . . . 19
2.1.2 Sparsity . . . . . 20
2.1.3 Incoherence and restricted isometry properties . . . . . 21
2.1.3.1 Incoherence . . . . . 22
2.1.3.2 Restricted isometry property (RIP) . . . . . 24
2.2 Signal reconstruction algorithms . . . . . 26
2.2.1 Minimum $\ell_2$-norm reconstruction . . . . . 28
2.2.2 Minimum $\ell_0$-norm reconstruction . . . . . 28
2.2.3 Minimum $\ell_1$-norm reconstruction . . . . . 29
2.2.4 Minimum $\ell_q$-norm reconstruction for $0 < q < 1$ . . . . . 30
2.2.5 Weighted norm minimization . . . . . 32
2.2.6 Geometric Interpretation . . . . . 33
2.2.7 Sparse signal reconstruction from noisy measurements . . . . . 36

**3 Minimizing Nonconvex Functions for Sparse Signal Reconstruction 39**
3.1 Introduction . . . . . 39
3.2 Proposed Convex Function . . . . . 41
3.3 The MCCR Algorithms . . . . . 46
3.3.1 First approach: Select $\boldsymbol{B}_0$ such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2$ . . . . . 47
3.3.2 Second Approach: Select $\boldsymbol{B}_0$ such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s} - \theta\boldsymbol{s}_0)||_2^2$ 53
3.3.3 Proposed objective functions . . . . . 57
3.4 Performance Enhancement . . . . . 61
3.4.1 Inversion Operation Associated with (3.16) . . . . . 61
3.4.2 Non–convexity of $g(\boldsymbol{s})$ . . . . . 62
3.5 Simulation Results . . . . . 64
3.6 Conclusion . . . . . 81

**4 Blind Source Separation of an Unknown Number of Sources 83**

4.1 Introduction . . . . . 83

4.2 Problem Formulation . . . . . 88

4.2.1 Sparse representation . . . . . 89

4.2.2 Preparation steps . . . . . 91

4.2.3 Mixing matrix estimation . . . . . 98

4.3 Proposed Clustering Algorithms . . . . . 102

4.3.1 Hierarchical Clustering . . . . . 103

4.3.2 $T^2$–statistical Test . . . . . 106

4.3.3 First Clustering Algorithm: Sequential Cluster Extraction (SCE) 108

4.3.3.1 Estimating the mixing matrix and the number of sources 109

4.3.3.2 SCE for noisy measurements . . . . . 118

4.3.4 Second Clustering Algorithm: Modified Angular Histogram (MAH) 123

4.3.5 Third Clustering Algorithm: Modified Angular Histogram followed by Sequential Cluster Extraction (MAHSCE) . . . . . 129

4.4 Simulation Results . . . . . 133

4.5 Conclusion . . . . . 139

**5 Sequential Sparse Signal Extraction with Applications to EEG/MEG Signal Processing 141**

5.1 Introduction . . . . . 141

5.2 Blind Sparse Signal Extraction (BSSE) . . . . . 148

5.2.1 Extraction of a sparse source. . . . . 149

5.2.2 Removing the extracted source from the mixture. . . . . 152

5.3 Constrained Sparse Component Analysis (cSCA) . . . . . 154

5.3.1 Temporal constraint SCA (tcSCA) algorithm . . . . . 156

5.3.2 Support constraint SCA (sucSCA) algorithm . . . . . 161

5.3.3 Spatial constraint SCA (scSCA) algorithm . . . . . 166
5.4 Simulation results . . . . . 169
5.5 Conclusion . . . . . 186

**6 Conclusion and future work** **189**
6.1 Suggestions for future research . . . . . 190

# List of Figures


1.1 [34] EEG source imaging. (a) Boundary element model for EEG forward calculations, (b) Example of cortical patch activations: Three simultaneous Gaussian distributed sources were assumed in both hemispheres, (c) Estimated source distribution. . . . . 5

1.2 [49] (a) Single-pixel, compressive sensing camera. (b) Conventional digital camera image of a soccer ball. (c) $64 \times 64$ black-and-white image $\hat{\boldsymbol{y}}$ of the same ball ($n = 4,096$ pixels) recovered from $m = 1,600$ random measurements taken by the camera in (a). . . . . 7

1.3 [50] Blind source separation of 6 acoustic sources from 4 mixtures. . . . 9

2.1 A plot of $g_p(s)$ for some values of $0 \leq p \leq 2$ . . . . . 27

2.2 Geometric interpretation of the (a) failure of $\ell_2$–norm, (b) success of $\ell_1$–norm, (c) failure of $\ell_1$–norm, (d) success of $\ell_{w1}$–norm, (e) success of $\ell_{w2}$–norm, and (f) success of $\ell_q$–norm ($0 < q < 1$), in estimating sparse solution vectors. See text for more details. . . . . 34

3.1 A non-convex function $g(\boldsymbol{z})$ that has many local minima and a smooth convex function $f(\boldsymbol{z})$. . . . . 42

3.2 A non-convex function $g(s)$ and three different convex functions $f_i(s)$. Only $f_3(s)$ satisfies the conditions of Theorem 3.2. . . . . 44

3.3 Successive iterations of the proposed algorithm for minimizing the nonconvex function $g_c(s)$ (black curve). The hollow circles represent the starting points, while the solid ones represent the global minima of the locally–convex functions. . . . . 45

3.4 Successive iterations of the proposed algorithm for minimizing the nonconvex function $g_c(s[j])$ (black curve). The hollow circles represent the starting points, while the solid ones represent the global minima of the locally–convex functions. . . . . 51

3.5 Geometric interpretation of the solution vector obtained by: (a) $||\boldsymbol{s}||_2^2$ minimization, (b) $||\boldsymbol{W}_{k-1}^{-1}\boldsymbol{s}||_2^2$ minimization, and (c) $||\boldsymbol{W}_{k-1}^{-1}(\boldsymbol{s}-\theta\boldsymbol{s}_{k-1})||_2^2$ minimization . . . . . 52

3.6 $g_{atan}(s) = \text{atan}(|s|/\delta)$ for different values of $\delta$. . . . . 58

3.7 The effect of $\delta$ on the performance of MCCR when $g_c(s) = g_{atan}(s)$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$); (b) The average number of iterations required to get a sparse solution vector. . . . . 67

3.8 Effect of $\epsilon$ on the performance of MCCR for $g_c(s) = g_{log}(s)$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) The average number of iterations required to get a sparse solution vector. . . . . 69

3.9 Effect of $\epsilon$ on the performance of MCCR for $g_c(s) = g_{atan}(s)$ and $\delta = 1$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) The average number of iterations required to get a sparse solution vector. . . . . 70

3.10 Effect of $\epsilon$ on the performance of MCCR for $g_c(s) = g_{atan}(s)$ and $\delta = 0.1$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) The average number of iterations required to get a sparse solution vector. . . . . 71

3.11 Effect of $\theta$ on the performance of MCCR for three different objective functions for the condition $i = 30$, $n = 256$, and $m$ increases from 60 to 90. (a) the probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) the average number of iterations required to get a sparse solution vector, and (c) the average time in (sec.). . . . . 73

3.12 Effect of $\theta$ on the performance of MCCR for three different objective functions for the condition $i = 60$, $n = 512$, and $m$ increases from 120 to 160. (a) the probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) the average number of iterations required to get a sparse solution vector, and (c) the average time in (sec.). . . . . 74

3.13 Comparison between the five objective functions in Table 3.1. (a) The probability of exact reconstruction of the original sparse vector; (b) The average number of iterations required to get a sparse solution vector. . . 76

3.14 Comparison between MCCR and PMCCR. (a) The probability of exact reconstruction of the original sparse vector; (b) The average number of iterations required to get a sparse solution vector. The number of iterations of the PMCCR algorithm represents the value of $j$ in Table 3.3. . . . . 78

3.15 Comparison between MCCR and four different algorithms. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$); (b) The average number of iterations required to get a sparse solution vector. . . . . 80

4.1 (a) Scatter plot of a mixture of four sparse sources, (b) Scatter plot of the normalized mixtures. Four clusters corresponding to the four mixing columns are clearly visible. . . . . 92

4.2 (a) Scatter plot of a mixture of four sparse sources after removing the columns that have small norm values, (b) Scatter plot of the normalized mixtures on the surface of the unit hemisphere. (c) The proposed normalization technique shows four clear clusters corresponding to the four mixing columns. . . . . 96

4.3 (a) A data set consists of 3 clusters and 4 outlier points; (b) Clustering the data set presented in (a) using hierarchical clustering. . . . . 105

4.4 (a) A scatter plot of the measured data matrix, (b) The CP1 parameter of the estimated clusters, (c) The first 5 estimated clusters. . . . . 114

4.5 The distance from $\lambda_2$ to the virtual line connecting $\lambda_1$ and $\lambda_m$ can be used as a concentration parameter. . . . . 117

4.6 Effect of the trimming procedure on the identification of the number of sources from noisy measurements. (a) CP1 before the trimming step, (b) CP2 before the trimming step,(c) CP1 after the trimming step, (d) CP2 after the trimming step. . . . . 120

4.7 Comparison between the proposed algorithm and the $k$-means algorithm in terms of the average SIR$(\boldsymbol{A}, \hat{\boldsymbol{A}})$. The sparse source matrix $\boldsymbol{S}$ is constructed such that $\delta\%$ of its columns satisfy the disjoint orthogonality principal. (a) $\delta = 30$, (b) $\delta = 50$. . . . . 121

4.8 (a) A scatter plot of the measured data matrix, (b) The histogram plot of the calculated data. . . . . 125

4.9 (a) A scatter plot of the first three rows of the measured data matrix, (b) The CP parameter of the estimated clusters. . . . . 130

4.10 Results of Example 1. (a) Original photos, (b) Mixed photos, (c) Estimated photos using FastICA algorithm, (d) and (e) Estimated photos when the mixing matrix is estimated using $k$–means and MAHSCE, respectively. . . 135

4.11 Results of Example 2. (a) Original source signals, (b) Mixed signals, (c) and (d) Estimated sources when the mixing matrix is estimated using the MAHSCE and the k-means, respectively. . . . . 137

5.1 Comparison between the least squares solution and the solutions produced by the tcSCA algorithm. The signals from top to bottom are the desired signal, the estimated signal using the tcSCA algorithm with $\alpha = 0$, the estimated signal using the tcSCA algorithm with $\alpha = 1$, the estimated signal using the lest squares approach, and the reference signal, respectively. 160

5.2 An example of extracting a sparse signal using sucSCA algorithm in four different scenarios. . . . . 166

5.3 Results of Example 1. (a) Original source signals, (b) Mixed signals, (c) and (d) Estimated source signals using the BSSE algorithm and the FastICA algorithm, respectively. . . . . 170

5.4 The average NMSE between the estimated sources and the original ones. The $x$ axis represents the indices of the original sources, and the $y$ axis represents the average NMSE over 200 different runs corresponding to 200 different selections of the square mixing matrix. The figures (a)–(d) correspond to the cases $n = 5$, 10, 15, and 20, respectively. . . . . 171

5.5 Comparison between the tcSCA, scSCA, and sucSCA algorithms. (a)-(b) The average NMSE and the average convergence time, respectively, for the case of all sources have the same amplitude, (c)-(d) The average NMSE and the average convergence time, respectively, for the case when the amplitude of the desired source signal is 5 times larger than that of the other source signals. . . . . 173

5.6 Automatic removal of eye blink artifact from real EEG data using the tcSCA algorithm. (a) Original EEG data, (b) The signals from top to bottom correspond to the extracted eye blink artifact, the denoised eye blink artifact, and the reference signal, respectively, (c) Clean version of the EEG data shown in (a). . . . . 175

5.7 Automatic removal of eye blink artifact from real EEG data using the scSCA algorithm. (a) Reference topographic map, (b) The topographic map of the extracted artifact, (c) The upper signal correspond to the extracted eye blink, while the lower signal is the denoised eye blink artifact, (d) Clean version of the EEG data shown in Figure 5.6(a) . . . . . 176

5.8 (a) Simulated EEG data (b) The simulated EEG data contaminated with a 60 Hz line voltage interference, (c) Cleaned EEG data using the sucSCA algorithm. . . . . 179

5.9 (a) Template for the implanted ERP signal (dotted curve) and the averaged ERP signal at the first electrode of the generated EEG data (continuous curve), (b) The constructed EEG data for Example 3. . . . . 182

5.10 (a) Different ERP signals associated with the tcSCA algorithm. The signals top to bottom are the original ERP signal, the template signal, the denoised version of the estimated ERP signal, and the estimated ERP signal, respectively. (b) Different ERP signals associated with the scSCA algorithm. The signals from top to bottom are the original ERP signal, the denoised version of the estimated ERP signal, and the estimated ERP signal, respectively. 185

# Chapter 1

# Introduction

There are many situations in science and technology that seek solutions to underdetermined systems of equations, i.e. systems of linear equations with fewer equations than unknowns. Such problems are extremely common for a variety of reasons. For instance, the number of sensors may be small due to physical limitations as in breast cancer imaging, or the sensing process may be slow so that one can only measure the object a few times, as in MRI. Many other examples in inverse problems, array signal processing, and biomagnetic imaging all come to mind. However, it is known that a system of linear equations with fewer equations than unknowns has infinitely many solutions, and thus it is necessary to impose constraints on the candidate solution to identify which of these candidate solutions is the desired one.

On closer inspection, there are many applications which ask for "*sparse*" solutions of such systems, i.e. solutions with few nonzero elements; the interpretation being that we know a priori that most of the candidate sources, pixels or genes are zeros, while the locations of the nonzero elements are not known a priori. The problem of reconstruction of sparse signals from a limited number of linear measurements is known in the literature as "*compressed sensing*" or "*compressive sampling*" (CS).

Restricting the solution vector to be sparse converts the underlying problem from

being impossible into being a tractable, but nevertheless still difficult, problem. For an $(m \times n)$ system of linear equations, where $m$ is the number of measurements and $n > m$ is the length of the solution vector, a naive approach to find a *l-sparse* solution vector (i.e., a solution vector that has at most $l \ll n$ nonzero entries), is to find the solution vector to every $(m \times l)$ submatrix of the measuring matrix, and then selecting the set of indices that minimizes the residual error. Since there are $\binom{n}{l}$ submatrices of size $(m \times l)$, this combinatorial approach is prohibitive when $n$ and $l$ are large.

To overcome this difficulty, a number of computational strategies have been developed to find solutions with low computational complexity. Examples include greedy algorithms [1–5], the basis pursuit [6,7], iterative-thresholding algorithms [8,9], and iterative reweighted norm algorithms [10–18]. A brief description of some of these approaches is presented in Chapter 2.

All the techniques mentioned in the previous paragraph can solve the problem of linear underdetermined system of equations as long as the solution vector is sparse, sparsely represented in some basis (e.g., wavelet and Gabor dictionaries among others [19]), or piecewise constant, (hence its gradient is sparse [11], [20]). Accordingly, compressed sensing has a wide range of applications and it has been studied in the literature under many different names such as subset selection [21], sparse coding [22], sparse regression [23,24], sparse component analysis (SCA) [25–27] with application to blind source separation of more sources than sensors [28–33]. Other applications are biomagnetic inverse problems [10, 34–37], subband decomposition [38], sparse audio signal representation [39], dictionary learning [40–42], image acquisition [43], direction-of-arrival estimation [10], [44], channel equalization [45], echo cancelation [46], and image restoration [11, 47, 48]. See [18] and the references therein for a list of more applications. In all these applications, the underlying linear inverse problem is the same and can be stated as follows: Represent a signal of interest using the minimum number of "*atoms*" (vectors) from an overcomplete dictionary. In the next

section we introduce some practical applications that have been addressed in the literature.

## 1.1 Some Applications of Compressed Sensing

In this section we present some examples that reflect the wide range of applications of compressed sensing (CS) and its ability to solve some difficult problems. In the first example CS is used in localizing the brain sources that generate a measured event related potential (ERP) signal. This is an ill-posed problem because the number of candidate source locations exceeds many thousand, while the number of sensors is very small, e.g. 128. Although the number of source locations is very large, only few sources are responsible for the measured ERP signal; hence, the solution vector is sparse.

The second example we present is a practical application in which the theory of compressed sensing is utilized in building a *"single–pixel"* digital camera which can be utilized in capturing still images. The single–pixel camera takes a few measurements of the target scene, then a CS algorithm is used to restore the original image. In the third example CS is used in solving the problem of blind source separation (BSS) when the number of sensors is less than the number of sources. The difficulties associated with this example are: (i) both the sources and the mixing matrix are unknown, (ii) there are more sources than sensors, and (iii) the sources are, in general, not sparse. However, by utilizing the CS techniques, the sources can be perfectly estimated under ideal conditions. We now discuss these examples in more detail.

### 1.1.1 EEG Source Imaging

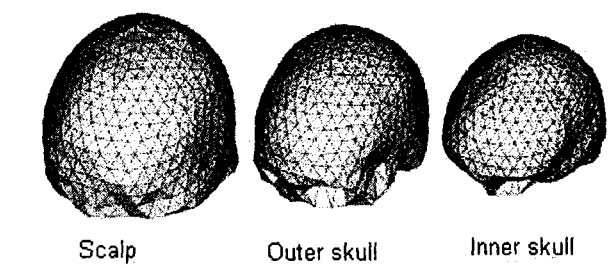
EEG source imaging is an inverse problem in which noninvasive measurements of electrical potentials at multichannel scalp electrodes are used for estimating neuronal

electrical sources distributed on a human brain cortex. The neuronal sources are usually modeled as dipole sources that are confined within thin ($\sim$ 4 mm) gray matter regions of the cerebral cortex. To determine the proper locations and orientations of the sources, the cortical surface is usually partitioned into a large number of triangular elements (Figure 1.1(a)). The current intensity of each source is related to the measured EEG data through the following linear model [34]

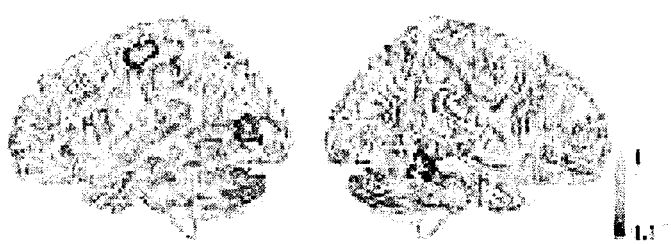
$$\boldsymbol{x} = \boldsymbol{L}\boldsymbol{i} + \boldsymbol{v}$$

where $\boldsymbol{L}$ is the $(m \times n)$ lead field matrix, $\boldsymbol{x}$ is a column vector representing the measured EEG signal at the $m$ electrodes at a given time instant, $\boldsymbol{i}$ is a column vector made of the $n$ corresponding current source intensities, and $\boldsymbol{v}$ is a noise vector.

Since exact source locations inside the real human brain cannot be estimated a priori, neuroelectromagnetic inverse problems are difficult to verify by *in vivo* experiments. Thus we present in this example a simulated experiment that was done in [34]. For forward calculations, a three-layer boundary element model shown in Figure 1.1(a) was used. This model consists of inner and outer skull boundaries and a scalp surface. To construct an artificial ERP signal, cortical patches were made from a set of dipoles with a Gaussian distributed current intensity profile, perpendicularly orientated to the partitioned cortical surface. Figure 1.1(b) shows an example of the assumed cortical patch activations. After calculating electrical potentials at the 128 electrodes assuming a 200-Hz sampling rate, real brain noise, which was obtained from a prestimulus period of a practical EEG experiment, was added. The original ERP signal without noise was scaled to produce a 7 dB signal to noise ratio (SNR). The inverse problem is solved using the FOCUSS algorithm [10], and the distribution of the estimated current intensities is presented in Figure 1.1(c). As shown in this figure, the distribution of the sources is perfectly estimated.




(a)



(b)

(c)

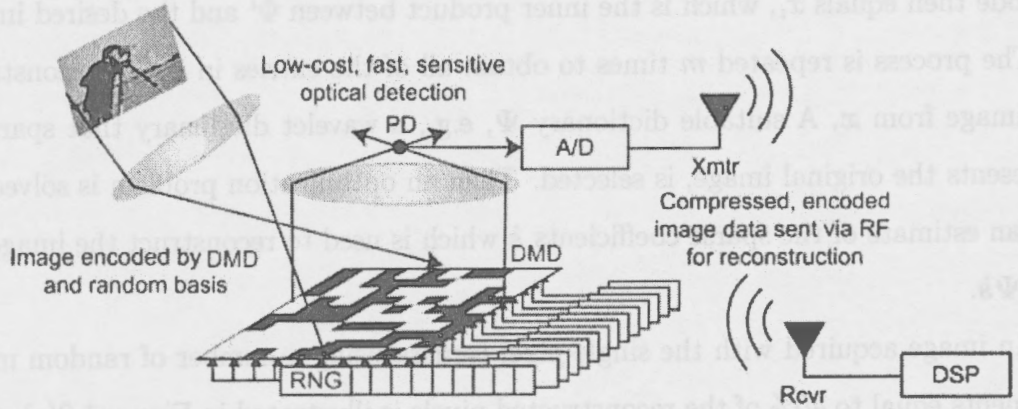
Figure 1.1: [34] EEG source imaging. (a) Boundary element model for EEG forward calculations, (b) Example of cortical patch activations: Three simultaneous Gaussian distributed sources were assumed in both hemispheres, (c) Estimated source distribution.

### 1.1.2 Single-pixel compressive digital camera

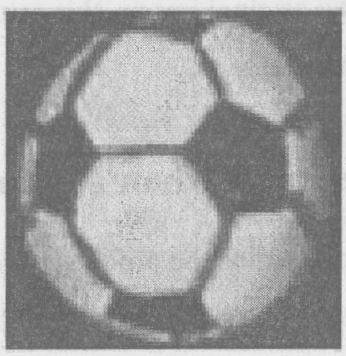
As will be described in the next chapter, the idea of compressed sensing is that, for a $l$-sparse vector $\boldsymbol{y}$ of length $n \gg l$ there is a measuring matrix $\Phi \in \mathbb{R}^{(m \times n)}$ such that, with high probability, $\boldsymbol{y}$ can be uniquely recovered from a low-dimensional vector $\boldsymbol{x} = \boldsymbol{\Phi y}$, where $\boldsymbol{x} \in \mathbb{R}^m$ and $l \leq m \ll n$. The idea is also applicable if $\boldsymbol{y}$ is not sparse but can be sparsely represented in terms of some basis, i.e. $\boldsymbol{y} = \boldsymbol{\Psi s}$, where the columns of $\boldsymbol{\Psi}$ represents the basis, and $\boldsymbol{s}$ is a sparse vector of weighting coefficients. In this case, $\boldsymbol{x}$ is expressed as $\boldsymbol{x} = \boldsymbol{\Phi y} = \boldsymbol{\Phi \Psi s} = \boldsymbol{As}$. Accordingly, $\boldsymbol{y}$ can be recovered from $\boldsymbol{x}$ by first estimating the sparse vector $\boldsymbol{s}$, then estimating $\boldsymbol{y}$ using $\boldsymbol{y} = \boldsymbol{\Psi \hat{s}}$, where $\hat{\boldsymbol{s}}$ is the estimated weighting coefficients.

Conventional imaging devices that use CMOS technology are limited essentially to the visible spectrum. The single–pixel camera that we are presenting in this section could significantly expand this capability (see [49] for more details). The idea of the single-pixel digital camera is based on the previously described model, where the target scene is represented by vector $\boldsymbol{y}$ and the $i$th pixel of the measured picture $\boldsymbol{x}$ is the inner product between the $i$th row of $\boldsymbol{\Phi}$ and the target scene $\boldsymbol{y}$. The implementation of the single-pixel digital camera is shown in Figure 1.2(a) [49]. As shown in this figure, the incident light-field corresponding to the desired image $\boldsymbol{y}$ is reflected off a digital micromirror device (DMD) consisting of an array of $n$ tiny mirrors each of which can be oriented in two different positions. The orientation of the DMD is controlled by the random number generator (RNG). The reflected light is then collected by a second lens and focused onto a single photodiode (the single pixel).

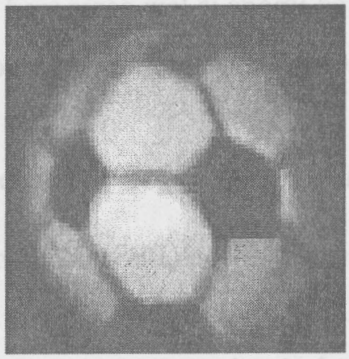
The entries of the measuring matrix $\boldsymbol{\Phi}$ are either 1 or 0 with equal probability, i.e., they can be modeled as independent and identically distributed (*i.i.d.*) random variables from a Bernoulli density function. This is implemented in Figure 1.2(a)




(a)



(b)



(c)

Figure 1.2: [49] (a) Single-pixel, compressive sensing camera. (b) Conventional digital camera image of a soccer ball. (c) $64 \times 64$ black-and-white image $\hat{y}$ of the same ball ($n = 4,096$ pixels) recovered from $m = 1,600$ random measurements taken by the camera in (a).

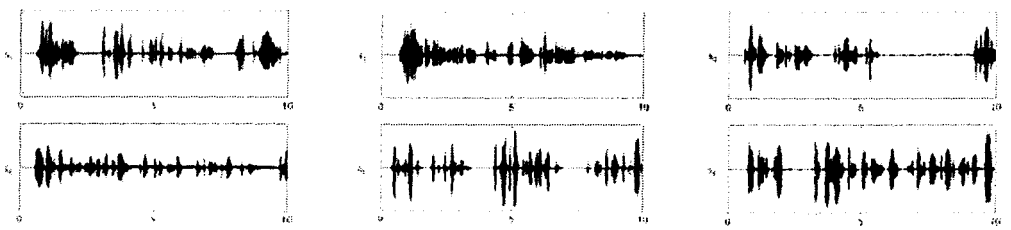
by the RNG which sets the mirror orientation either towards the photodiode (corresponding to 1) or away from the photodiode (corresponding to 0). The positions of the mirrors correspond to the entries of $\Phi^i$, the $i$th row of $\boldsymbol{\Phi}$. The voltage at the photodiode then equals $x_i$, which is the inner product between $\Phi^i$ and the desired image $\boldsymbol{y}$. The process is repeated $m$ times to obtain all of the entries in $\boldsymbol{x}$. To reconstruct the image from $\boldsymbol{x}$, A suitable dictionary $\boldsymbol{\Psi}$, e.g., a wavelet dictionary that sparsely represents the original image, is selected. Then an optimization problem is solved to find an estimate of the sparse coefficients $\hat{\boldsymbol{s}}$ which is used to reconstruct the image as $\hat{\boldsymbol{y}} = \boldsymbol{\Psi}\hat{\boldsymbol{s}}$.

An image acquired with the single-pixel camera using a number of random measurements equal to 40% of the reconstructed pixels is illustrated in Figure 1.2(c); this can be compared to the target image in Figure 1.2(b). Although the resolution of the reconstructed image is low compared to the original image, this resolution can be increased by increasing the number of measurements.

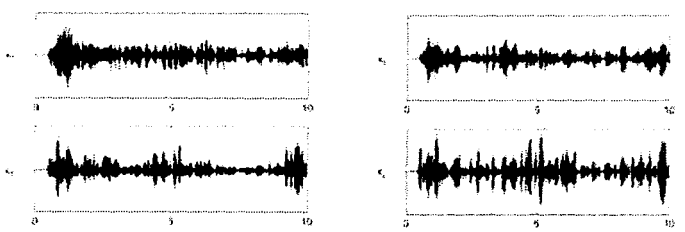
### 1.1.3 Blind separation of more sources than sensors

The blind source separation (BSS) problem is defined as the problem of reconstructing $n$ *unknown* source signals from $m$ linear measurements when the mixing matrix is *unknown*. The relation between the measured signals and the original source signals can be expressed mathematically as $\boldsymbol{X} = \boldsymbol{AS}$, where $\boldsymbol{X} \in \mathbb{R}^{m \times T}$ is a matrix of measured signals, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is an unknown mixing matrix, $\boldsymbol{S} \in \mathbb{R}^{n \times T}$ is a matrix of unknown source signals, $m$ is the number of observations, $n$ is the number of sources, and $T$ is the number of samples.

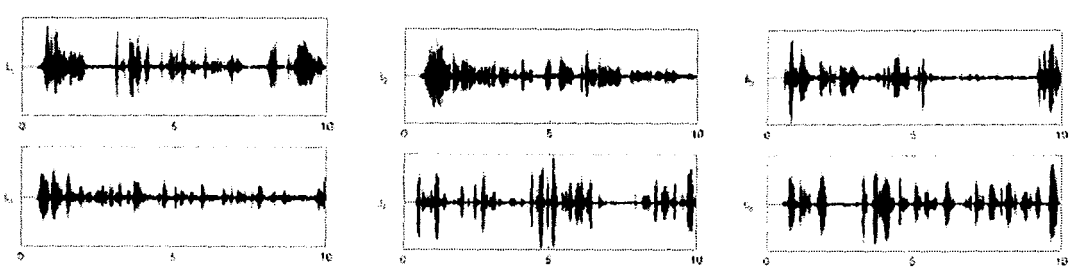
Over the last two decades, the BSS problem was extensively studied and many algorithms have been developed for estimating the hidden source signals. The most successful technique that has been developed in the literature for solving the BSS problem is called independent component analysis (ICA) [51, 52, 54, 55]. The BSS



(a) Six acoustic sources



(b) Four mixtures



(c) Estimated Sources

Figure 1.3: [50] Blind source separation of 6 acoustic sources from 4 mixtures.

problem can be solved using the ICA technique as long as the underlying BSS problem satisfies the following conditions: 1) the sources are mutually independent, 2) the number of sources is less than or equal the number of sensors, 3) at most one of the sources is Gaussian, and 4) the mixing matrix is full column rank. However, there are many applications for which one or more of these assumptions are violated. For example, in some applications the number of sources is generally unknown and could be greater than the number of sensors, while in other applications some of the sources could be correlated. In these applications, ICA does not produce satisfactory results, and an alternative technique must be utilized for estimating the hidden sources.

A recently developed technique, known as Sparse Component Analysis (SCA), has received a great deal of attention in recent years. SCA can solve the under–determined BSS problem, i.e. the case of more sources than sensors. The additional information required in compensating the limited number of sensors is the sparseness of the source matrix $\boldsymbol{S}$. Since non–sparse sources can often be sparsely represented under a suitable linear transformation, (e.g., the short time Fourier transform, the wavelet transform, the wavelet packets transform,... etc.), the SCA problem is quite general and also applicable to non-sparse sources. Accordingly, SCA can substitute for ICA in the cases when some of the assumptions associated with ICA are violated, e.g., when the number of sources is greater than the number of sensors. See Chapter 4 and Chapter 5 for more details about this technique.

Figure 1.3 presents an example presented in [50] for separating 6 acoustic source signals from 4 mixtures using the SCA analysis technique. The problem is solved in four steps. In the first step, a linear transformation is applied on the measured signals for sparse representation. The mixing matrix $\boldsymbol{A}$ is estimated in the second step via clustering the columns of the measured signals in the transform domain. The coefficients of the original source signals in the transform domain are estimated using a CS algorithm. Finally, the sources are reconstructed from their estimated coefficients

by performing the inverse transformation. The reconstructed signals are shown in Figure 1.3(c). Comparing the reconstructed signals with the original source signals shown in Figure 1.3(a), it is clear that the original signals are well reconstructed even though the number of measurements is less than the number of sources.

## 1.2 Contributions and Thesis Organization

In this section, the contributions of this thesis are summarized.

### 1.2.1 Chapter 2

Chapter 2 provides a review of the recently developed technique called compressed sensing (CS). The CS technique is developed to solve a linear under-determined system of equations, i.e., when the number of measurements is less than the number of unknowns. In Chapter 2 we review the fundamental principals of the CS technique and we discuss the conditions under which a unique solution of the linear under-determined system of equations can be obtained using the CS technique. Finally we provide a revision of the most popular algorithms that have been developed in the recent literature.

### 1.2.2 Chapter 3

In Chapter 3 we develop a novel methodology for minimizing a class of non–convex (concave on the non–negative orthant) functions for solving the linear under–determined system of equations $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{x}$ when the solution vector $\boldsymbol{s}$ is known a priori to be sparse. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. The resulting algorithm is

iterative and is absolutely converging to a fixed point of the original objective function. The importance of the proposed methodology is that, for a certain selection of the convex objective function, the class of algorithms called Iterative Reweighted Least Square (IRLS) [10–16] can be derived from the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods. In addition, we also propose a new class of algorithms with better convergence properties compared to the regular IRLS algorithms and, hence, can be considered as enhancements to these algorithms.

In this chapter we propose a straightforward technique for selecting a convex function such that, for any starting solution vector $\boldsymbol{s}_0$, the algorithm generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges to a fixed point of the original objective function. Since the original objective functions are non-convex, the proposed algorithm is susceptible to convergence to a local minimum. To alleviate this difficulty, we propose a random perturbation technique that enhances the performance of the proposed algorithm. The numerical results show that the proposed algorithms outperform some of the well known algorithms that are usually utilized for solving the same problem.

### 1.2.3 Chapter 4

In Chapter 4 we discuss the problem of finding a unique solution to the under–determined BSS problem $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$ using the SCA approach. The discussion presented in this chapter is restricted to the two–step approach, in which the mixing matrix $\boldsymbol{A}$ and the source matrix $\boldsymbol{S}$ are estimated separately. In the first part of this chapter, we discuss in detail the main steps associated with the two–step approach, and we provide a literature review about the popular clustering techniques that have been suggested in the literature for estimating the mixing matrix.

Most of the clustering algorithms that have been utilized for estimating the mixing matrix suffer from the presence of noisy and/or outlier measurements, and they also

require the number of clusters, which equals the number of sources, to be known in advance. In the second part of this chapter we propose three different clustering algorithms that can be utilized for estimating the mixing matrix, as well as the number of sources, even when the measurements are contaminated with additive noise.

The first clustering algorithm we propose is based on a clustering technique called *hierarchical clustering* (HC). Identification of the individual clusters constructed with HC is a fundamental difficulty associated with this clustering technique. Previous approaches for identifying the clusters are either manual or depend on some parameters which are hard to determine. In this chapter we propose a novel clustering algorithm which mitigates the previously mentioned difficulty by incorporating a statistical test with the HC algorithm. The proposed algorithm is based on sequentially extracting compact clusters that have been constructed by the HC algorithm, where the extraction decision is based on the statistical test. A cluster is extracted when its mean is significantly different from that of the closest cluster. For identifying the clusters that correspond to the columns of the mixing matrix, we developed a quantitative measure called the *concentration parameter* (CP).

The second clustering algorithm that we propose in Chapter 4 is a generalization of the angular histogram clustering algorithm. Previously suggested angular histogram clustering algorithms are restricted to the 2-D case. However, the proposed algorithm generalizes the previous algorithms to the $m \geq 2$ case. Under certain conditions, which will be described in Chapter 4, the proposed algorithm can efficiently estimate the mixing matrix and the number of sources.

The third clustering algorithm that we propose in this chapter is a combination of the first two algorithms. This algorithm combines the advantages of the first two clustering algorithms and avoids their limitations. The numerical results show that the proposed algorithms are more efficient than some of the existing algorithms.

### 1.2.4 Chapter 5

In Chapter 5 we extend two ICA techniques into the SCA domain. We refer to these as the sequential blind source extraction (BSE) and the constraint ICA (cICA) techniques. An ICA–based algorithm can extract a source signal by finding a separating vector that maximizes the *non–Gaussianity* of the extracted source signal. These algorithms are general purpose algorithms and are not designed specifically for extracting sparse sources. In this chapter we extend the previous work and propose four novel algorithms which are capable of extracting sparse source signals. The four algorithms are based on finding a separating vector that maximizes the *sparsity* of the extracted source signal. The first algorithm is an extension of the BSE algorithm to the case of extracting sparse source signals. The remaining three algorithms are based on incorporating prior information available about the desired source signal into the optimization problem which is designed for extracting the sources. These algorithms are called constrained SCA (cSCA).

The first algorithm in the cSCA class is based on utilizing a reference signal that conveys *temporal* information about the desired source signals. The proposed algorithm has two different versions, depending on the measure of closeness between the extracted source signal and the reference signal. The second algorithm in this class is based on utilizing a reference signal that conveys information about the *support* of the desired sparse source signal. This algorithm is useful when there is an ambiguity associated with the sign of each sample of the desired source signal. This algorithm was utilized successfully in removing the (50/60 Hz) line voltage interference from a simulated EEG data. The last algorithm that we propose in this chapter can extract the desired sparse source signal when prior information about its mixing column is available. Previous approaches for solving this problem cannot estimate a single source signal, and are based on the ICA technique, and hence does not enforce the sparsity of the desired source signal.

In the field of biomedical signal analysis it is common that some prior information is available about the temporal or spatial characteristics of the desired signal. This prior information can then be used by a cSCA algorithm to extract the desired source signal. Simulation results show that the three proposed cSCA algorithms can be successfully used for removing different kind of artifacts from real EEG data and for estimating the ERP signal from synthesized EEG data.

Finally, in Chapter 6 we conclude the thesis and propose some potential research problems for future work.

# Chapter 2

# Fundamentals of compressed sensing (CS)

Compressed sensing or compressive sampling (CS) is a simple and efficient signal acquisition technique that collects a few measurements about the signal of interest and later uses optimization techniques for reconstructing the original signal from what appears to be an incomplete set of measurements [56]. Accordingly, CS can be seen as a technique for sensing and compressing data simultaneously (thus the name). The CS technique relies on two fundamental principals: 1) *sparse* representation of the signal of interest in some basis, which is called the *representation* basis; and 2) *incoherence* between the sensing matrix and the representation basis. The terms sensing, sparsity, and incoherence will be defined in the next section.

The objective of this chapter is to answer the following questions:

1. Is it possible to design a sensing matrix with far fewer rows than columns to capture the main information in the measured signal?

2. What is the sufficient number of measurements (rows of the sensing matrix) such that the original signal can be reconstructed with high probability?

| List of symbols | |
|---|---|
| $\boldsymbol{\Phi}$ | Sensing matrix. |
| $\boldsymbol{\Psi}$ | Dictionary matrix. |
| $\mathcal{N}$ | Null space. |
| $\delta_l$ | The isometry constant. |
| $\boldsymbol{x}$ | The vector of measurements of length $m$. |
| $\boldsymbol{s}$ | Sparse solution vector of length $n$. |
| $\boldsymbol{s}^*$ | The true sparse vector. |
| $\hat{\boldsymbol{s}}$ | An estimate of $\boldsymbol{s}^*$. |
| $\boldsymbol{s}_l^*$ | The vector $\boldsymbol{s}^*$ with all but the $l$-largest entries set to zero. |
| $\boldsymbol{s}_k$ | An estimate of $\boldsymbol{s}^*$ at the $k$th iteration. |
| $\boldsymbol{W}_k$ | A diagonal weightng matrix at the $k$th iteration. |
| $m$ | The number of measurements (length of $\boldsymbol{x}$). |
| $n$ | Length of $\boldsymbol{s}$. |
| $l$ | The diversity of $\boldsymbol{s}^*$. |
| $\lvert\cdot\rvert$ | Cardinality of a vector. |

3. What are the available techniques that can solve the inverse problem, i.e., reconstructing the original signal from the few measurements?

## 2.1 Problem formulation

In this section we present the mathematical formulation of the problem of compressed sensing. The following subsections present the formal definitions of some terminologies that will be used in this chapter.

### 2.1.1 Compressed sensing

In this subsection we present a formal description of "*compressed sensing*". Sensing of the time domain signal $y(t)$ is defined as the process of collecting some measurements about $y(t)$ by correlating $y(t)$ with some *sensing waveforms* $\{\phi_j(t)\}$, i.e.,

$$x_j = \langle y, \phi_j \rangle, \quad j = 1, 2, \ldots, m. \tag{2.1}$$

Based on the sensing waveforms, the entries of the vector $\boldsymbol{x}$ have different interpretations. For example, if the sensing waveforms are sinusoids, then $\boldsymbol{x}$ is a vector of Fourier coefficients, and if the sensing waveforms are Dirac delta functions, then $\boldsymbol{x}$ is a vector of sampled values of $\boldsymbol{y}(t)$.

To simplify the presentation of the CS technique we will restrict our attention to discrete signals $\boldsymbol{y} \in \mathbb{R}^n$. Accordingly, equation 2.1 can be rewritten in matrix form as

$$\boldsymbol{x} = \boldsymbol{\Phi y}, \tag{2.2}$$

where the $j$th row of the *sensing matrix* $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$ is the discrete representation of the $j$th sensing function $\phi_j(t)$, and $\boldsymbol{y} \in \mathbb{R}^n$ is the discrete representation of $y(t)$. Based on this model, compressed sensing is defined as the sensing process for which the number $m$ of available measurements is much smaller than the dimension $n$ of the signal $\boldsymbol{y}$. The problem associated with compressed sensing is that we have to solve an under–determined system of equations to recover the original signal $\boldsymbol{y}$ from the measurement vector $\boldsymbol{x}$. However, since the number of equations is less than the number of unknowns, it is known that this system has infinitely many solutions , and thus it is necessary to impose constraints on the candidate solution to identify which of these candidate solutions is the desired one. A powerful constraint that can be used in this regard is the "*sparsity*" of the solution vector, which is defined in the next subsection.

### 2.1.2 Sparsity

Before explaining the importance of the sparsity constraint in solving under-determined systems of equations, we present the following definitions [41]:

- *Sparsity* $= |\{s[i] = 0\}|$ = number of zero entries in $\boldsymbol{s}$, where $|\cdot|$ denotes cardinality of a set.

- *Diversity* $= |\{s[i] \neq 0\}|$ = number of nonzero entries in $\boldsymbol{s}$.

- *l–sparse* vector: a *l–sparse* vector is defined as the vector that has at most $l$ nonzero entries.

- *Compressible* vector: The vector $\boldsymbol{s}$ is called compressible if its entries obey a power law $|s|_{(j)} \leq C_r j^{-r}$, where $|s|_{(j)}$ is the $j$th largest value of $\boldsymbol{s}$, i.e., $\left(|s|_{(1)} \geq |s|_{(2)} \geq \ldots \geq |s|_{(n)}\right), r > 1$, and $C_r$ is a constant which depends only on $r$ [57]. This means that most entries of a compressible vector are small while only few entries are large. Such a model is appropriate for the wavelet coefficients of a piecewise smooth signal, for example.

As stated in the previous subsection, an underdetermined system of linear equations has infinite candidate solutions of the form $\hat{\boldsymbol{y}} = \boldsymbol{y}_0 + \mathcal{N}$ where $\boldsymbol{y}_0$ is any vector that satisfies $\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{y}_0$, and $\mathcal{N} := \mathcal{N}(\boldsymbol{\Phi})$ is the null space of $\boldsymbol{\Phi}$. As will be shown later, if the candidate solution vector is known to be *l–sparse*, and under some conditions on the sensing matrix $\boldsymbol{\Phi}$, the solution vector can be uniquely determined using an optimization technique. Fortunately, this also applies to nonsparse vectors that can be sparsely represented in a suitably selected basis $\boldsymbol{\Psi}$, i.e.,

$$\boldsymbol{y} = \boldsymbol{\Psi}\boldsymbol{s}, \tag{2.3}$$

where the coefficient vector $\boldsymbol{s}$ is sparse.[1] Clearly $\boldsymbol{y}$ and $\boldsymbol{s}$ are equivalent representations of the signal, with $\boldsymbol{y}$ in the time or space domain and $\boldsymbol{s}$ in the $\boldsymbol{\Psi}$ domain. In some applications, it may be natural to choose $\boldsymbol{\Psi}$ as an orthonormal basis, while in others the signal $\boldsymbol{y}$ may only be sparsely represented when $\boldsymbol{\Psi}$ is a *redundant* dictionary; i.e., it has more columns than rows. A good example is provided by an audio signal which is often sparsely represented in an overcomplete dictionary with atoms (columns) that have the form of modulated Gaussian pulses, e.g., $\sigma^{-\frac{1}{2}} e^{-\frac{(t-t_0)^2}{2\sigma^2}} e^{i\omega t}$, where $t_0$, $\omega$, and $\sigma$ are the discrete shift, modulation and scale parameters, respectively [18].

Combining (2.2) and (2.3) and taking into consideration the case of noisy measurements, the sensing process can be written as

$$\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{s} + \boldsymbol{v} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{v}, \tag{2.4}$$

where $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi} \in \mathbb{R}^{m \times n}$, and $\boldsymbol{v} \in \mathbb{R}^n$ is a noise vector. Assuming that the coefficient vector $\boldsymbol{s}$ is *l–sparse*, then $\boldsymbol{s}$, and hence $\boldsymbol{y} = \boldsymbol{\Psi}\boldsymbol{s}$, can only be estimated from $\boldsymbol{x}$ if the matrices $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$, and $\boldsymbol{A}$ satisfy the properties described in the next subsection.

### 2.1.3 Incoherence and restricted isometry properties

The sparsity of the solution vector, or its representation in some basis, is a necessary but not sufficient condition for finding a unique solution to an underdetermined system of linear equations. In addition to the sparsity principle, CS relies on another principle which is the "*incoherence*" between the sensing matrix $\boldsymbol{\Phi}$ and the sparsity basis $\boldsymbol{\Psi}$. The incoherence principal is also related to an equivalent property, which is associated with $\boldsymbol{A}$, called restricted isometry property (RIP).

[1] In some papers a vector is called *l–sparse* if it is a linear combination of only $l$ basis vectors. However, in this thesis we use the definition presented in the text.

#### 2.1.3.1 Incoherence

To simplify the treatment, we assume that the sparsity basis matrix $\boldsymbol{\Psi}$ is orthonormal, and the sensing matrix $\boldsymbol{\Phi}$ consists of $m$ rows drawn randomly from an orthogonal basis $\hat{\boldsymbol{\Phi}} \in \mathbb{R}^{n \times n}$, which is normalized such that $\hat{\boldsymbol{\Phi}}^T \hat{\boldsymbol{\Phi}} = n\boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix. The operation of extracting the $m$ rows of $\boldsymbol{\Phi}$ from $\hat{\boldsymbol{\Phi}}$ is denoted as $\boldsymbol{\Phi} := \hat{\boldsymbol{\Phi}}^{\Omega}$, where $\Omega \subset \{1, \ldots, n\}$ is a subset of indices of size $|\Omega| = m$. Based on this notation, $\boldsymbol{A}$ can also be written as $\boldsymbol{A} := \hat{\boldsymbol{A}}^{\Omega}$, where $\hat{\boldsymbol{A}} = \hat{\boldsymbol{\Phi}}\boldsymbol{\Psi}$ is an orthogonal matrix with $\hat{\boldsymbol{A}}^T \hat{\boldsymbol{A}} = n\boldsymbol{I}$.

Let $\mu(\hat{\boldsymbol{A}})$ be the element with the largest magnitude among all entries of $\hat{\boldsymbol{A}}$, i.e.,

$$\mu(\hat{\boldsymbol{A}}) = \max_{k,j} |\hat{A}_{k,j}|. \tag{2.5}$$

Assume that the measurements are noise-free and the sparse solution vector $\boldsymbol{s}$ is *l-sparse* and is reconstructed using basis pursuit, i.e.,

$$\hat{\boldsymbol{s}} = arg \min_{\boldsymbol{z}} ||\boldsymbol{z}||_{\ell_1} \quad \text{subject to} \quad \hat{\boldsymbol{A}}^{\Omega} \boldsymbol{z} = \hat{\boldsymbol{A}}^{\Omega} \boldsymbol{s}, \tag{2.6}$$

then it was proved in [58] that $\hat{\boldsymbol{s}} = \boldsymbol{s}$ with overwhelming probability for all subsets $\Omega$ with size

$$|\Omega| \geq C.\mu^2(\hat{\boldsymbol{A}}).l.\log n. \tag{2.7}$$

for some positive constant $C$. Equation (2.7) indicates that, in addition to the size and the sparsity of the solution vector, the number of measurements depends on the largest magnitude among all entries of $\hat{\boldsymbol{A}}$. Since each row (or column) of $\hat{\boldsymbol{A}}$ necessarily has an $\ell_2$-norm equal to $\sqrt{n}$, $\mu(\hat{\boldsymbol{A}})$ will take a value between 1 and $\sqrt{n}$. When the magnitude of each entry of $\hat{\boldsymbol{A}}$ equals 1 as in the case when $\hat{\boldsymbol{A}}$ is the discrete Fourier transform, $\mu(\hat{\boldsymbol{A}}) = 1$ and the number of measurements in (2.7) is the smallest. On the other hand, if a row of $\hat{\boldsymbol{A}}$ is maximally concentrated–all the row entries but one vanish–then $\mu^2(\hat{\boldsymbol{A}}) = n$, and (2.7) indicates that there is no guarantee that the solution vector can be recovered from a limited number of samples.

Since $\hat{A}_{k,j} = \langle \hat{\Phi}_k, \Psi_j \rangle$, where $\hat{\Phi}_k$ is the $k$th row of $\hat{\mathbf{\Phi}}$ and $\Psi_j$ is the $j$th column of $\mathbf{\Psi}$, $\mu(\hat{\boldsymbol{A}})$ can be rewritten as

$$\mu(\hat{\mathbf{\Phi}}\mathbf{\Psi}) = \max_{k,j} |\langle \hat{\Phi}_k, \Psi_j \rangle|. \tag{2.8}$$

For $\mu(\hat{\mathbf{\Phi}}\mathbf{\Psi})$ to be close to its minimum value of 1, each of the sensing vectors (rows of $\hat{\mathbf{\Phi}}$) must have a dense representation in $\mathbf{\Psi}$. To emphasize this relationship, $\mu(\hat{\mathbf{\Phi}}\mathbf{\Psi})$ is often referred to as the "*mutual coherence*" [59], [60].

The bound (2.7) indicates that a *l-sparse* signal can be reconstructed from $\sim l \log n$ measurements using basis pursuit as long as the pair $(\hat{\mathbf{\Phi}}, \mathbf{\Psi})$ has very low mutual coherence parameter. Examples of such pairs are [56]:

**1- $\hat{\mathbf{\Phi}}$ is the spike basis and $\mathbf{\Psi}$ is the Fourier basis.**

In this case the $k$th row of $\hat{\mathbf{\Phi}}$ is expressed as $\hat{\phi}_k(t) = \delta(t-k)$ and the $j$th column of $\mathbf{\Psi}$ is expressed as $\psi_j(t) = n^{-1/2} e^{-i2\pi jt/n}$. Since $\hat{\mathbf{\Phi}}$ is the sensing matrix, this corresponds to the classical sampling scheme in the time or space domain. The time-frequency pair obeys $\mu(\hat{\mathbf{\Phi}}\mathbf{\Psi}) = 1$ and, therefore, we have maximal incoherence.

**2- $\hat{\mathbf{\Phi}}$ is the noiselet basis [62] and $\mathbf{\Psi}$ is the wavelet basis.**

The coherence between noiselets and Haar wavelets is $\sqrt{2}$, and that between noiselets and Daubechies $D4$ and $D8$ wavelets is respectively about 2.2 and 2.9 across a very wide range of sample sizes $n$ [56]. Noiselets are also maximally incoherent with spikes and incoherent with the Fourier basis.

**3- $\hat{\mathbf{\Phi}}$ is a random matrix and $\mathbf{\Psi}$ is any fixed basis.**

With high probability, the coherence between any orthobasis $\hat{\mathbf{\Phi}}$ selected at random and any fixed basis $\mathbf{\Psi}$ is about $\sqrt{2 \log n}$. This is also applicable when the entries of

$\hat{\mathbf{\Phi}}$ are samples of independent and identically distributed (*iid*) random variables from Gaussian or Bernoulli distributions [56].

#### 2.1.3.2 Restricted isometry property (RIP)

The restricted isometry property is a notion introduced in [63] and has proved to be very useful in studying the general robustness of CS. As will be shown later, RIP provides a very useful tool for determining sufficient conditions that guarantee exact reconstruction of a sparse solution vector for different reconstructing (decoding) algorithms. In contrast to (2.7), the conditions derived based on the RIP are deterministic, i.e. there is no probability of failure.

Consider the following definition.

**Definition 2.1** *[63]*

*For each integer $l = 1, 2, \ldots$, the isometry constant $\delta_l$ of a matrix $\boldsymbol{A}$ is defined as the smallest number such that*

$$(1 - \delta_l)||\boldsymbol{s}||_{\ell_2}^2 \leq ||\boldsymbol{A}\boldsymbol{s}||_{\ell_2}^2 \leq (1 + \delta_l)||\boldsymbol{s}||_{\ell_2}^2. \tag{2.9}$$

*holds for all $l$–sparse vectors $\boldsymbol{s}$.*

It will be loosely said that a matrix $\boldsymbol{A}$ obeys the *RIP* of order $l$ if $\delta_l$ is not too close to 1. When the RIP holds, the Euclidean length of *$l$–sparse* signals is approximately preserved by $\boldsymbol{A}$, which in turn implies that *$l$–sparse* vectors *cannot be* in the nullspace of $\boldsymbol{A}$. Clearly this is very important as otherwise there would be no hope of reconstructing these *$l$–sparse* vectors. The RIP can also be interpreted as all subsets of $l$ columns taken from $\boldsymbol{A}$ being *nearly* orthogonal (the columns of $\boldsymbol{A}$ cannot be exactly orthogonal since we have more columns than rows).

The following example [61] reflects the connection between the RIP and CS. Suppose that we wish to acquire *$l$–sparse* signals with $\boldsymbol{A}$. Assume first that $\delta_{2l} < 1$; then

it can be shown that one can recover a *l–sparse* vector $\boldsymbol{s}$ from the data $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{s}$. Indeed, $\boldsymbol{s}$ is the unique sparsest solution of the system $\boldsymbol{y} = \boldsymbol{A}\hat{\boldsymbol{s}}$., i.e. the one with the smallest number of nonzero entries. This can be shown as follows: consider any other solution of the form $\boldsymbol{s} + \boldsymbol{z}$ with $\boldsymbol{z} \in \mathcal{N}(\boldsymbol{A})$ and$\boldsymbol{z} \neq 0$. Then $\boldsymbol{A}\boldsymbol{z} = 0$ and therefore, $\boldsymbol{z}$ must have at least $2l + 1$ nonzero entries. It then follows that $\boldsymbol{s} + \boldsymbol{z}$ must have at least $l + 1$ nonzero entries. Conversely, assume that $\delta_{2l} = 1$. Then $2l$ columns of $\boldsymbol{A}$ could be linearly dependent in which case there is a $2l$*–sparse* vector $\boldsymbol{z}$ satisfying $\boldsymbol{A}\boldsymbol{z} = 0$. Then $\boldsymbol{z}$ can be decomposed as $\boldsymbol{z} = \boldsymbol{s} - \tilde{\boldsymbol{s}}$, where both $\boldsymbol{s}$ and $\tilde{\boldsymbol{s}}$ are *l–sparse*. Accordingly, we can write $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{A}\tilde{\boldsymbol{s}}$ which indicates that there are a pair of *l–sparse* vectors giving the same measurements. Clearly, one cannot reconstruct such sparse objects. Hence, to recover *l–sparse* signals, one would need to impose $\delta_{2l} < 1$.

What is remaining is to find some matrices that satisfy the RIP and to determine the relation between the number of measurements $m$ and the sparsity of the solution vector $l$. There are many matrices that satisfy the RIP, i.e. matrices with column vectors taken from arbitrary subsets being nearly orthogonal. Consider the following sensing matrices: [56]

1. Form $\boldsymbol{A}$ by sampling $n$ column vectors uniformly at random on the unit sphere of $\mathbb{R}^m$.

2. Form $\boldsymbol{A}$ by sampling *i.i.d.* entries from the normal distribution with mean zero and variance $1/m$.

3. Form $\boldsymbol{A}$ by sampling *i.i.d.* entries from a symmetric Bernoulli distribution $(P(A_{i,j} = \pm 1/\sqrt{m}) = \frac{1}{2})$ or other sub–Gaussian distribution.

Then with overwhelming probability, all these matrices obey the restricted isometry property provided that

$$m \geq C.k.\log(n/l); \tag{2.10}$$

where $C$ is some constant depending on each instance.

Note that, for a nonsparse signal that can be sparsely represented in an arbitrary orthobasis $\boldsymbol{\Psi}$, the RIP can also hold for sensing matrices $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$, where $\boldsymbol{\Phi}$ is an $m \times n$ measurement matrix drawn randomly from a suitable distribution. It was addressed in [56] that, for a given $\boldsymbol{\Psi}$, if $\boldsymbol{\Phi}$ is selected as one of the three previously mentioned cases, then with overwhelming probability, the matrix $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$ obeys the RIP provided that (2.10) is satisfied, where again $C$ is some constant depending on each instance. It has to be noted that these random measurement matrices $\boldsymbol{\Phi}$ are in a sense universal [64]; the sparsity basis need not even be known when designing the measurement system. To reconstruct the original signal from the $m$ measurements in the vector $\boldsymbol{y}$, an optimization technique must be used, and this is the topic of the next section.

## 2.2 Signal reconstruction algorithms

The signal reconstruction algorithm (some times called the decoder) must take the $m$ random measurements in the vector $\boldsymbol{x}$, the basis $\boldsymbol{\Psi}$, and the random measurement matrix $\boldsymbol{\Phi}$ (or the random seed that generated it) and reconstruct the signal $\boldsymbol{y} \in \mathbb{R}^n$ or, equivalently, its *l-sparse* coefficient vector $\boldsymbol{s}^*$. In this section we will consider solving the following linear underdetermined system of equations

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}^*, \tag{2.11}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{s}^* \in \mathbb{R}^n$ is a *l-sparse* vector, and $m < n$.

The goal of a sparse–signal recovery algorithm is to obtain an estimate of $\boldsymbol{s}^*$ given only $\boldsymbol{x}$ and $\boldsymbol{A}$. This problem is non-trivial since $\boldsymbol{A}$ is overcomplete, i.e., the number of equations is less than the number of unknowns. Accordingly there are infinitely many solutions to (2.11) of the form $\hat{\boldsymbol{s}} = \boldsymbol{s}_0 + \mathcal{N}$, where $\boldsymbol{s}_0$ is any vector that satisfies (2.11).

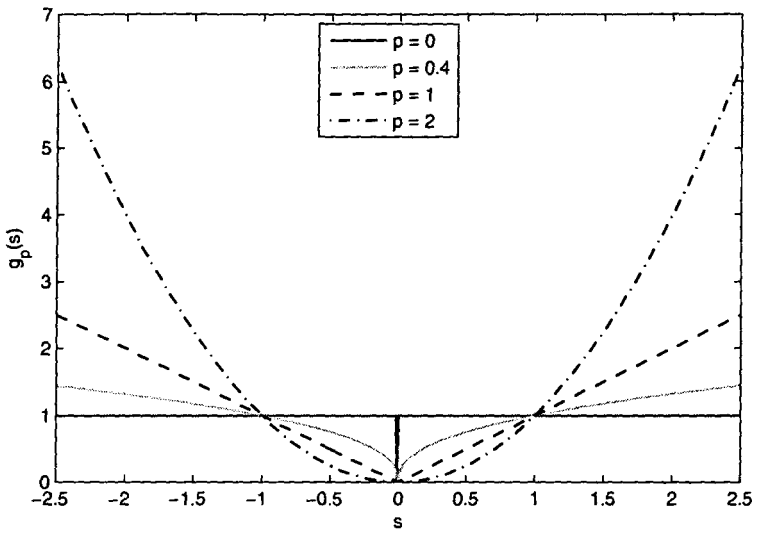



Figure 2.1: A plot of $g_p(s)$ for some values of $0 \leq p \leq 2$

Since the original vector $\boldsymbol{s}^*$ is sparse, the problem of finding the desired solution can be phrased as an optimization problem where the objective is to maximize (minimize) an appropriate measure of sparsity (diversity) while simultaneously satisfying the constraints defined by (2.11), respectively. This can be expressed mathematically as

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{2.12}$$

where $g(\cdot)$ is an objective function to be minimized that encourages sparsity in the solution. We consider this function to be of the form

$$g_p(\boldsymbol{s}) = ||\boldsymbol{s}||_p^p = \sum_i |s[i]|^p, \tag{2.13}$$

where $p \geq 0$, and $s[i]$ is the $i$-th element of $\boldsymbol{s}$. Eq. (2.13) expresses the $p$-th norm of $\mathbf{s}$ (although it is not strictly a valid norm for $0 \leq p < 1$). A plot of $g_p(s)$ for some values of $p$ is presented in Figure 2.1. We now briefly discuss issues related to solving (2.13) for various values of $p$.

### 2.2.1 Minimum $\ell_2$-norm reconstruction

The $\ell_2$-norm solution to (2.12) is the well–known least squares solution given by $\boldsymbol{s}_{Ls} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$. This is a closed form solution. With reference to Figure 2.1, the convexity of $g_2(s)$ implies a unique solution to (2.12). Since the penalty imposed by $g_2(s[i])$ on small nonzero coefficients of the solution vector is small, the least squares solution has a tendency to spread the energy among a large number of entries of $\boldsymbol{s}$, resulting in a non–sparse solution. Accordingly, $\ell_2$ minimization is not appropriate for finding a *l–sparse* solution.

### 2.2.2 Minimum $\ell_0$-norm reconstruction

Since the $\ell_2$–norm measures signal energy and not signal sparsity, consider the $\ell_0$–norm that counts the number of non-zero entries in $\boldsymbol{s}$. (Hence a *l-sparse* vector has $\ell_0$–norm equal to $l$.) The optimization problem (2.12) in this case can be written as

$$(P_0) \quad \min_{\boldsymbol{s}} ||\boldsymbol{s}||_{\ell_0} \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \tag{2.14}$$

Referring to Figure 2.1, we observe that $g_0(s)$ is flat over all values of $s$ except at $s = 0$, which implies that any gradient descent technique will fail to converge to the sparse solution. Since solving this problem is equivalent to selecting $l$ vectors of the measuring matrix $\boldsymbol{A}$ that best represented the measured vector $\boldsymbol{x}$, the solution vector to $(P_0)$ can be obtained by searching over the $\binom{n}{l}$ possible ways in which the basis sets can be chosen to find the best solution. In principle, this strategy is effective. For example, in the particular case of random measurements, where the entries of $\boldsymbol{A}$, or equivalently $\boldsymbol{\Phi}$, are drawn from a Gaussian distribution, and a signal $\boldsymbol{s}^*$ with $||\boldsymbol{s}^*||_{\ell_0} = l$, then with probability 1 the problem $(P_0)$ will have a unique solution $\hat{\boldsymbol{s}}$ that is exactly $\boldsymbol{s}^*$, as long as $m \geq 2l$ [65].

Unfortunately, the cost of such combinatorial searches is prohibitive, making finding an optimal solution using an exhaustive search infeasible. In addition to this

difficulty, it was shown that $(P_0)$ yields a solution which is not robust to noise [25].

These limitations motivated researchers to replace $g_0(\boldsymbol{s})$ by other functions that are robust to noise and can be solved efficiently (such as $g_2(\boldsymbol{s})$), but nevertheless offer sparse solutions (such as $g_0(\boldsymbol{s})$). A straightforward approach of achieving this goal is to minimize $g_p(\boldsymbol{s})$ for $0 \leq p \leq 2$.

### 2.2.3 Minimum $\ell_1$-norm reconstruction

For $p = 1$, (2.12) is usually called basis pursuit [6] and is expressed as

$$(P_1) \quad \min_{\boldsymbol{s}} ||\boldsymbol{s}||_{\ell_1} \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \tag{2.15}$$

Since $p = 1$ is the smallest value of $p$ for which $g_p(\boldsymbol{s})$ is convex, $\ell_1$-minimization has been utilized in the context of sparse solutions for many years. See [14], [18] and the references therein for the history of $\ell_1$-minimization and its applications. Because (2.15) is convex, it can be solved efficiently, a much better situation than that of (2.14). The improved sparsity of the $\ell_1$-norm relative to the least squares solution is partially due to the fact that the penalty imposed by $\ell_1$-norm on values of $0 \leq |s| < 1$ is greater than that imposed by $\ell_2$-norm, refer to Figure 2.1.

The equivalence between the solution vectors of $(P_1)$ and $(P_0)$ was extensively studied in the literature. As stated in the previous section, a remarkable result of Candes and Tao [66] for random, Gaussian measurements is that $(P_1)$ can recover with high probability any *l–sparse* vector $\boldsymbol{s}^*$ provided that the number of measurements satisfies (2.10) for some constant $C$, which depends on the desired probability of success. In any case, $C$ tends to one as $n \rightarrow \infty$. The cost of replacing $(P_0)$ by $(P_1)$ is that more measurements are required, depending logarithmically on $n$. Sharp reconstruction thresholds have been computed by Donoho and Tanner [67] so that for any choice of sparsity $l$ and signal size $n$, the required number of measurements $m$ for $(P_1)$ to recover $\boldsymbol{s}^*$ with high probability can be determined precisely. Their

results replace $\log(n/l)$ with $\log(n/m)$, i.e. $m \geq C.l.\log(n/m)$. However, $m$ appears in both sides of this inequality, and this can be adjusted to compute a threshold of the sparsity $l \leq \frac{m}{C\log(n/m)}$ for a given number of measurements $m$.

The following results are based on utilizing the RIP described in the previous section. It was shown in [66] that if the solution vector satisfies $||\boldsymbol{s}^*||_{\ell_0} = l$ and the sensing matrix $\boldsymbol{A}$ satisfies the relation $\delta_{3l} + 3\delta_{4l} < 2$, then $\hat{\boldsymbol{s}} = \boldsymbol{s}^*$ is the unique minimizer of $(P_1)$. Also it was shown in [68, 69] that all vectors $\boldsymbol{s}^*$ with $||\boldsymbol{s}^*||_{\ell_0} \leq l$ can be recovered exactly using $(P_1)$ as long as the measuring matrix $\boldsymbol{A}$ obeys $\delta_{2l} + 2\delta_{3l} < 1$. The following Theorem was stated in [70] regarding the reconstruction of a compressible vector $\boldsymbol{s}^*$, i.e. a vector with few large entries and many small ones.

**Theorem 2.1** *[70]*

*Assume that $\delta_{2l} < \sqrt{2} - 1$. Then the solution $\hat{\boldsymbol{s}}$ to (2.15) obeys*

$$||\hat{\boldsymbol{s}} - \boldsymbol{s}^*||_{\ell_1} \leq C_0||\boldsymbol{s}_l^* - \boldsymbol{s}^*||_{\ell_1} \tag{2.16}$$

*and*

$$||\hat{\boldsymbol{s}} - \boldsymbol{s}^*||_{\ell_2} \leq C_0 k^{-1/2}||\boldsymbol{s}_l^* - \boldsymbol{s}^*||_{\ell_1} \tag{2.17}$$

*where $\boldsymbol{s}_l^*$ is the vector $\boldsymbol{s}^*$ with all but the $l$-largest entries set to zero, and $C_0$ is a constant given explicitly in [70].*

In particular, if $\boldsymbol{s}^*$ is *$l$-sparse*, the recovery is exact.

### 2.2.4 Minimum $\ell_q$-norm reconstruction for $0 < q < 1$

In view of the above, and referring to Figure 2.1, it is natural to select $p = q$ where $0 < q < 1$. In this case (2.12) can be written as

$$(P_q) \quad \min_{\boldsymbol{s}} ||\boldsymbol{s}||_{\ell_q} \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \tag{2.18}$$

Recall that $||s||_p^p = \sum_i |s[i]|^p$. Accordingly, $||\cdot||_q$ is not a norm when $0 < q < 1$, though $||\cdot||_q^q$ satisfies the triangle inequality and induces a metric.

Although the function $g_q(s)$ is not convex, which means that it may have multiple local minima, it is more "democratic" than the $\ell_1$-norm. This behavior is shown in Figure 2.2. As we can see from this figure, $g_q(s)$ imposes a larger penalty on values of $0 \leq |s| < 1$ than $g_1(s)$, and the opposite is true for values of $|s| > 1$. It was shown in [71–73] that minimizing $\ell_q$–norm, for values of $0 < q < 1$ performs better than minimizing the $\ell_1$–norm in the sense that a smaller number of measurements are needed for exact reconstruction of the sparse solution vector. More precisely, it was shown in [73] that for the case of random Gaussian measurements, the above condition (2.10) of Candes and Tao generalizes to

$$m \geq C_1(q)l + qC_2(q)l\log(n/l),$$

where $C_1$, $C_2$ are determined explicitly, and are bounded in $q$. Thus, *ideally*, the dependence of the sufficient number of measurements $m$ on the signal size $n$ vanishes as $q \rightarrow 0$. However the simulations presented in [73] did not reflect this behavior.

The following theorem provides useful results that were derived based on the RIP.

**Theorem 2.2** *[71]*

*Let $s^* \in \mathbb{R}^n$ have sparsity $||s^*||_{\ell_0} = l$, $0 < q \leq 1$, $b > 1$, and $a = b^{q/(2-q)}$. Suppose that $\boldsymbol{A}$ satisfies $\delta_{al} + b\delta_{(a+1)l} < b - 1$. Then the unique minimizer of $(P_q)$ is exactly $s^*$.*

Note that, for example, when $q = 0.5$ and $a = 3$, the above theorem guarantees perfect reconstruction with $\ell_{0.5}$ minimization under the condition that $\delta_{3l} + 27\delta_{4l} < 26$, which is a less restrictive condition than the one needed to guarantee perfect reconstruction by $\ell_1$ minimization [74].

In [71] $(P_q)$ was minimized by alternating between gradient descent and projection into the constraint $\boldsymbol{A}\hat{\boldsymbol{s}} = \boldsymbol{x}$. This technique produced very promising results but

converged very slowly. Another approach for minimizing $(P_q)$ was suggested in [12] and is based on utilizing an affine scaling methodology. This approach led to an iterative algorithm similar to the FOCUSS algorithm derived in [10] and is presented in the next subsection.

### 2.2.5 Weighted norm minimization

In contrast to the least squares solution, (2.15) and (2.18) do not have closed form solutions and require optimization software. Accordingly several alternatives to (2.15) and (2.18) that combine the simplicity of the least squares solution and perform as well as, or even better than, the $\ell_1$-norm, have been proposed [10,12–14,75,76]. One of such alternatives is called Iterative Re-weighted Least Squares (IRLS) minimization. IRLS algorithms have the form

$$(P_{w\ell_2}) \quad \min_{\boldsymbol{s}} ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2 \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{2.19}$$

where $\boldsymbol{W}$ is a diagonal weighting matrix that reflects our prior knowledge about the solution vector $\boldsymbol{s}$. The resulting algorithm is iterative, and the estimated solution at the $k$th iteration can be expressed as

$$\boldsymbol{s}_k = \boldsymbol{W}_k(\boldsymbol{A}\boldsymbol{W}_k)^{\dagger}\boldsymbol{x}, \tag{2.20}$$

where $^{\dagger}$ is the Moore-Penrose inverse [77]. The difference between different IRLS algorithms resides in the way that the diagonal matrix is defined. In [10] $\boldsymbol{W}$ was selected as $\boldsymbol{W}_k = diag\,(|\boldsymbol{s}_{k-1}|)$ which was shown to be equivalent to minimizing $g(\boldsymbol{s}) = \sum_i \log(|s[i]|)$, while it was selected as $\boldsymbol{W}_k = diag(|s_{k-1}[i]|^{1-0.5q})$ in [12] for minimizing (2.18).

The motivation behind these weighted norm approaches can be explained as follows. Let $\boldsymbol{w}$ denote the main diagonal of the diagonal matrix $\boldsymbol{W}^{-1}$, then for (2.19) to be minimized, it is clear that the nonzero elements of $\boldsymbol{s}$ must be concentrated at

the indices where $w[i]$ has small values, while the values of $s[i]$ will converge to zero for those indices at which the $w[i]$ have large values. So starting from a point $\boldsymbol{s}_0$ close enough to a sparse solution $\boldsymbol{s}^*$, the IRLS algorithm (2.19) generates a sequence $\{\boldsymbol{s}_k\}_{k=0}^{\infty}$ which converges to $\boldsymbol{s}^*$. The local rate of convergence varies according to the expression for $\boldsymbol{W}$; for instance it was shown to be quadratic in [10], while in [14] it was either linear or super–linear for the algorithm approximating (2.15) or (2.18), respectively.

By examining the structure of the weighting matrices, we notice that any element in the solution vector that was estimated at any iteration to be zero will be kept at a value of zero at all successive iterations. This is the main drawback of this approach, because if any element of the solution vector is erroneously estimated as zero at any iteration, the algorithm will never converge to the exact solution. To overcome this difficulty and to improve the performance of the previously described algorithms a monotonically decreasing constant can be added to the diagonal elements of the weighting matrix [13, 14].

Another approach for weighted norm minimization is the one proposed in [18], where $\ell_1$-norm in (2.15) is replaced by $g_{w\ell_1}(\boldsymbol{s}) = ||\boldsymbol{W}_k^{-1}\boldsymbol{s}||_{\ell_1}$, where $\boldsymbol{W}_k = diag(|s_{k-1}[i]|)$ is also a diagonal weighting matrix. It was shown in [18] that this algorithm performs much better than the $\ell_1$-norm minimization and converges in few iterations. However each iteration is computationally expensive compared with an IRLS iteration.

### 2.2.6 Geometric Interpretation

In this section we present a geometric interpretation of the performance of the previously discussed objective functions, e.g. $\ell_p$-norm and weighted $\ell_p$-norm where $0 < p \leq 2$, in estimating sparse solutions. This geometric interpretation helps visualize why $\ell_2$-norm reconstruction fails to find the sparse solution that can be identified by $\ell_1$-norm and weighted-norm reconstruction.

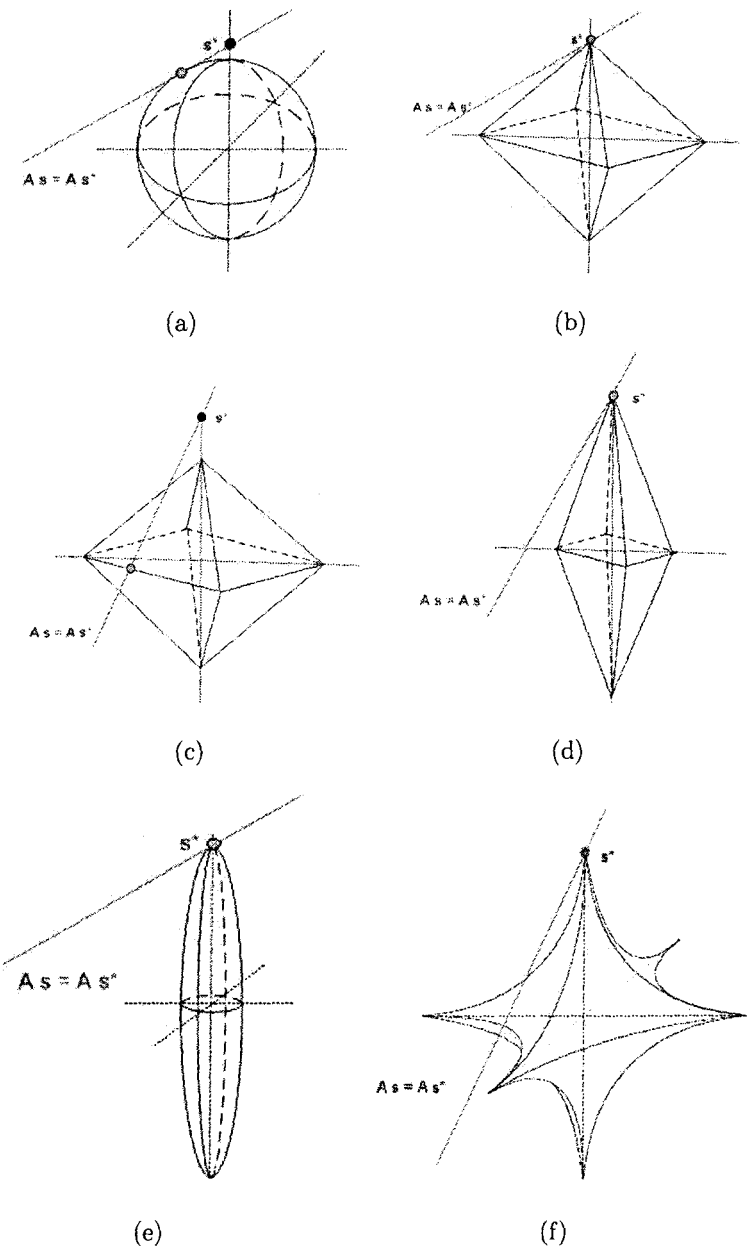



Figure 2.2: Geometric interpretation of the (a) failure of $\ell_2$–norm, (b) success of $\ell_1$–norm, (c) failure of $\ell_1$–norm, (d) success of $\ell_{w1}$–norm, (e) success of $\ell_{w2}$–norm, and (f) success of $\ell_q$–norm ($0 < q < 1$), in estimating sparse solution vectors. See text for more details.

For the sake of illustration, consider the simple 3-D example in Figure 2.2. The coordinate axes in this figure are $s_1$, $s_2$, and $s_3$. In this figure, the exact and the estimated solution vectors are represented by the solid (blue) circle at $\boldsymbol{s}^* = [0 \quad 1 \quad 0]^T$ and the gray (green) circle, respectively, while $\mathcal{H}$, the set of all points $\boldsymbol{s} \in \mathbb{R}^3$ obeying $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{A}\boldsymbol{s}^*$, is represented by the red line passing through $\boldsymbol{s}^*$.

The $\ell_2$ minimizer of (2.12) is the point on $\mathcal{H}$ closest to the origin. This point can be found by blowing up the $\ell_2$ ball, represented by the hypersphere in Figure 2.2(a), until it contacts $\mathcal{H}$. Due to the randomness of the entries of the sensing matrix $\boldsymbol{A}$, $\mathcal{H}$ is oriented at random angle. Accordingly, with high probability, the closest point $\hat{\boldsymbol{s}}$ will live away from the coordinate axes and hence will be neither sparse nor close to the correct answer $\boldsymbol{s}^*$ [78]. In contrast, the $\ell_1$ ball in Figure 2.2(b) has points aligned with the coordinate axes. Therefore, depending on the orientation of $\mathcal{H}$, there are two possible cases. In the first case, when the $\ell_1$ ball is blown up, it will contact $\mathcal{H}$ at a point near the coordinate axes, which is precisely where the sparse vector $\boldsymbol{s}^*$ is located as shown in Figure 2.2(b). In the second case, shown in Figure 2.2(c), the $\ell_1$ ball contacts $\mathcal{H}$ at a point far from the exact solution vector. Since both the RIP and the orientation of $\mathcal{H}$ depend on the entries of the measuring matrix $\boldsymbol{A}$, Theorem 2.2 can be interpreted geometrically as follows: for all measuring matrices with $\delta_{2k} < \sqrt{2} - 1$, $\mathcal{H}$ is oriented such that the $\ell_1$ ball contacts $\mathcal{H}$ at a point satisfying (2.16) and (2.17).

Geometrically, incorporating a diagonal weighting matrix into the $\ell_p$-norm, where $p = 1$ or 2, causes the $\ell_p$ ball to elongate along certain directions and no longer be symmetric. If the weighting matrix is properly selected, the $\ell_p$ ball will contact $\mathcal{H}$ at, or very close to, the exact solution vector $\boldsymbol{s}^*$ as shown in Figure 2.2(d)-(e) for $p = 1$ and 2, respectively. Note that the orientation of $\mathcal{H}$ in Figure 2.2(c),(d) is the same, i.e., the weighted $\ell_1$-norm minimization can find solutions to problems when the condition of Theorem 2.1 is violated. Also note that the orientation of $\mathcal{H}$ in Figure 2.2(e) is similar to that in Figure 2.2(a). However, by incorporating a

diagonal weighting matrix into the $\ell_2$-norm, the solution vector is estimated correctly as in Figure 2.2(e).

The failure of $\ell_p$-norm, where $p \geq 1$, in estimating a sparse vector, as in Figure 2.2(a) and (c), is due to the shape of the $\ell_p$ ball. As shown in Figure 2.2, the $\ell_p$ ball "bulges outward" for all $p > 1$, while it has a "diamond" shape for $p = 1$. This problem was partially alleviated in Figure 2.2(d) and (e) by incorporating a diagonal weighting matrix. Another way to overcome this difficulty is using $\ell_q$-norm, where $0 < q < 1$. For this range of $q$, the $\ell_q$ ball "bulges inward" as shown in Figure 2.2(f). Comparing Figure 2.2(c) and (f) we find that, in contrast to the $\ell_1$ ball, the $\ell_q$ ball does not have flat edges and hence it contacts $\mathcal{H}$ at the exact solution vector. This geometric analysis might explain why the $\ell_q$-norm outperforms the $\ell_1$-norm for all $0 < q < 1$.

### 2.2.7 Sparse signal reconstruction from noisy measurements

In any real application measured data could be corrupted by additive noise. Accordingly, CS should be able to deal with noisy measurements. In the presence of noise, the measured vector can be expressed as

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}^* + \boldsymbol{v}, \tag{2.21}$$

where $\boldsymbol{v} \in \mathbb{R}^m$ is a vector of additive noise with $||\boldsymbol{v}||_{\ell_2} \leq \epsilon$. If one seeks an estimate $\hat{\boldsymbol{s}}$ that leads to an exact reconstruction of $\boldsymbol{x}$, it will have generically at least $n$ nonzero components. To get a sparse representation one therefore has to allow for reconstruction errors. The best solution $\hat{\boldsymbol{s}}$ that one can expect is the one that has nonzero entries within the same support as the exact solution vector $\boldsymbol{s}^*$, with the same signs but of course slightly different values. The difference converges to zero as the variance of the noise diminishes.

To handle the presence of noise, the reconstruction algorithm (2.12) is modified

to

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad ||\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}||_{\ell_2}^2 \leq \epsilon, \tag{2.22}$$

where $\epsilon$ bounds the amount of noise in the measured data, and $g(\boldsymbol{s})$ is an objective function that encourages the sparsity of its argument, e.g. $||\boldsymbol{s}||_{\ell_1}$, $||\boldsymbol{W}^{-1}\boldsymbol{s}||_{\ell_1}$, $||\boldsymbol{s}||_{\ell_p}$, or $||\boldsymbol{W}^{-1}\boldsymbol{s}||_{\ell_2}$. Eq. (2.22) is equivalent to the following optimization problem

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} \frac{1}{2}||\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}||_{\ell_2}^2 + \lambda g(\boldsymbol{s}) \tag{2.23}$$

for an adequately chosen parameter $\lambda > 0$. Indeed if $\lambda$ is selected as the inverse of twice the Lagrangian multiplier of the constraint in (2.22), then both problems have the same optimum. If an estimate of the noise variance is available then the solution vector can be estimated using (2.22), otherwise, (2.23) has to be used. The regularization parameter $\lambda$ in this case can be estimated using L-curve method [79–81].

When $||\boldsymbol{s}||_{\ell_1}$ is used in (2.22) as the objective function $g(\boldsymbol{s})$ [56,57,70,82–84], the optimization problem becomes convex (a second order cone program) and the solution vector has the following property [56,57]:

**Theorem 2.3** *Assume that the measuring matrix $\boldsymbol{A}$ satisfies $\delta_{2l} < \sqrt{2} - 1$. Then the solution $\hat{\boldsymbol{s}}$ to (2.22), with $g(\boldsymbol{s}) = ||\boldsymbol{s}||_{\ell_1}$, obeys*

$$||\hat{\boldsymbol{s}} - \boldsymbol{s}^*||_{\ell_1} \leq \frac{C_0}{l}||\boldsymbol{s}_l^* - \boldsymbol{s}^*||_{\ell_1} + C_1\epsilon, \tag{2.24}$$

*where $\boldsymbol{s}_l^*$ is the vector $\boldsymbol{s}^*$ with all but the $l$-largest entries set to zero, and $C_0$ and $C_1$ are some constants.*

Theorem 2.3 states that the reconstruction error is bounded by the sum of two terms. The first is the error which would occur if one had noiseless data, see (2.16), and the second is proportional to the noise variance. The constants $C_0$ and $C_1$ are typically small. For example, with $\delta_{2l} = 1/4$, $C_0 \leq 5.5$ and $C_1 \leq 6$ [56].

When $||\boldsymbol{s}||_q^q$, with $0 < q < 1$, is used in (2.22) as the objective function $g(\boldsymbol{s})$, the solution vector will have the following property [74]:

**Theorem 2.4** *Assume that for some constants $a > 1$, and $al \in \mathbb{Z}^+$, the measuring matrix $\boldsymbol{A}$ satisfies $\delta_{al} + a^{\frac{2}{q}-1}\delta_{(a+1)l} < a^{\frac{2}{q}-1} - 1$, then the solution $\hat{\boldsymbol{s}}$ to (2.22), with $g(\boldsymbol{s}) = ||\boldsymbol{s}||_q^q$, obeys*

$$||\hat{\boldsymbol{s}} - \boldsymbol{s}^*||_{\ell_2}^q \leq C_1.\epsilon^q + \frac{C_2}{l^{1-q/2}}||\boldsymbol{s}_l^* - \boldsymbol{s}^*||_q^q, \tag{2.25}$$

*where $\boldsymbol{s}_l^*$ is the vector $\boldsymbol{s}^*$ with all but the $l$-largest entries set to zero, and the constants $C_1$ and $C_2$ are given explicitly in [74].*

Thus, as for the $\ell_1$-norm recovery, the reconstruction error (to the $q$th power) is bounded by the sum of two terms; the first term is proportional to the noise variance, while the second term is proportional to the best $l$-term approximation error of the exact solution vector.

When $||\boldsymbol{s}||_q^q$, with $0 < q < 1$, is used in (2.23) as an objective function, it was shown in [15,41] that (2.23) is the maximum a-posteriori estimate of the sparse vector $\boldsymbol{s}^*$ when a super Gaussian distribution is assumed as a prior distribution. Utilizing affine scaling methodology, the following iterative algorithm was derived in [15]

$$\boldsymbol{s}_{k+1} = \boldsymbol{W}_k\boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{W}_k^2\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{x}. \tag{2.26}$$

where $\boldsymbol{W}_k = diag\left(|s_k[i]|^{1-(q/2)}\right)$, and $\boldsymbol{I}$ is an identity matrix. Note that in the limit as $\lambda \rightarrow 0$, equations (2.26) and (2.20) are equivalent.

# Chapter 3

# Minimizing Nonconvex Functions for Sparse Signal Reconstruction

## 3.1 Introduction

The problem of finding a unique solution to a linear under–determined system of equations $\boldsymbol{As} = \boldsymbol{x}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $m < n$, is challenging indeed. As explained in Section 2.2, there are many algorithms developed for solving the above mentioned problem. In this chapter a novel methodology is developed and employed to minimize a class of non–convex (concave on the non–negative orthant) functions for solving the aforementioned problem. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. The resulting algorithm is iterative, and it will be shown that the penalty imposed by the convex objective function at any iteration depends on the gradient of the original objective function evaluated at the previous solution vector. Accordingly, if the objective function is carefully chosen, the penalty imposed by the convex function on the small entries of the solution vector can increase as the algorithm gets closer to a local minimum of the original objective function. As will be shown in this chapter,

| List of symbols | |
|---|---|
| $\mathcal{O}_1$ | The non-negative orthant. |
| $g(\boldsymbol{s})$ | An objective function for measuring the diversity of $\boldsymbol{s}$. |
| $f(\boldsymbol{s})$ | Convex approximation of $g(\boldsymbol{s})$. |
| $\boldsymbol{d}_0$ | The gradient of $g(\boldsymbol{s})$ at the starting point $\boldsymbol{s}_0$. |
| $\boldsymbol{s}^*$ | Global minimizer of $f(\boldsymbol{s})$. |
| $\boldsymbol{s}_k$ | The solution vector at the $k$th iteration. |

for a certain selection of the convex objective function, the class of algorithms called Iterative Re–weighted Least Squares (IRLS) [10–16] can be derived from the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods.

In this chapter we propose a straightforward technique for selecting a convex function such that, for any starting solution vector $\boldsymbol{s}_0$, the algorithm generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges to a fixed point of the original objective function. Since the original objective functions are non-convex, the proposed algorithm is susceptible to convergence to a local minimum. To alleviate this difficulty, we propose a random perturbation technique that enhances the performance of the proposed algorithm. The proposed algorithm is called MCCR, as an abbreviation of Minimizing a Concave function via a Convex function Replacement.

Our contributions in this chapter can be summarized as follows. (1) we propose a technique that provides a deeper understanding of the IRLS class of algorithms and provides a natural mechanism for deriving IRLS algorithms starting from suitably chosen diversity measures, (2) we propose a new class of algorithms with better convergence properties compared to the regular IRLS algorithms and, hence, can be considered as enhancements to these algorithms, and (3) we suggest some novel techniques for improving the performance of IRLS methods.

## 3.2 Proposed Convex Function

In absence of noise, a sparse solution vector of the under–determined system of equations, $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{x}$, can be obtained by solving the following optimization problem

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{3.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $m < n$, and $g(\cdot)$ is an objective function to be minimized that encourages sparsity in the solution. In this chapter we restrict our attention to the class of non–convex objective functions that have the following properties [40]:

(P1) $g(\boldsymbol{s})$ is separable, i.e., $g(\boldsymbol{s}) = \sum_i g_c(s[i])$, where $g_c(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is a scalar function, and $s[i]$ is the $i$th entry of $\boldsymbol{s}$. Therefore, $g(\boldsymbol{s})$ is also permutation invariant, i.e., $g(\boldsymbol{s}) = g(\boldsymbol{P}\boldsymbol{s})$ for any permutation matrix $\boldsymbol{P}$.

(P2) The function $g_c(s) : \mathbb{R} \rightarrow \mathbb{R}$ referred to in P1 is sign invariant and concave–and–monotonically increasing on the nonnegative orthant $\mathcal{O}_1$.

Examples of $g_c(s)$ that were extensively used in the literature are $g_q(s) = |s|^q$ [1] and $g_{log}(s) = \log(|s|)$ [12, 13, 71, 72]. In this chapter we also propose using the following three functions $g_{log_\delta}(s) = \log\left(\frac{|s|}{\delta} + 1\right)$, $g_{atan}(s) = \text{atan}\left(\frac{|s|}{\delta}\right)$, and $g_{s/s} = \frac{|s|}{|s|+\delta}$, where $\delta > 0$. A comparison between all these functions is presented in the simulation results.

Since $g(\boldsymbol{s})$ is sign invariant and concave on $\mathcal{O}_1$, it is concave on each of the other orthants $\mathcal{O}_k, 1 \leq k \leq 2^n$. However, this does not imply that $g(\boldsymbol{s})$ is concave on $\mathbb{R}^n$. Consider the following theorem.

**Theorem 3.1** ( **[40] Theorem 8**) *Let $g(\boldsymbol{s}) : \mathbb{R}^n \rightarrow \mathbb{R}$ be permutation invariant, sign invariant, and concave on the nonnegative orthant $\mathcal{O}_1$. Then the global minimum of the objective function (3.1), has at most $m$ nonzero entries.*

[1] unless explicitly stated, $0 < q < 1$.

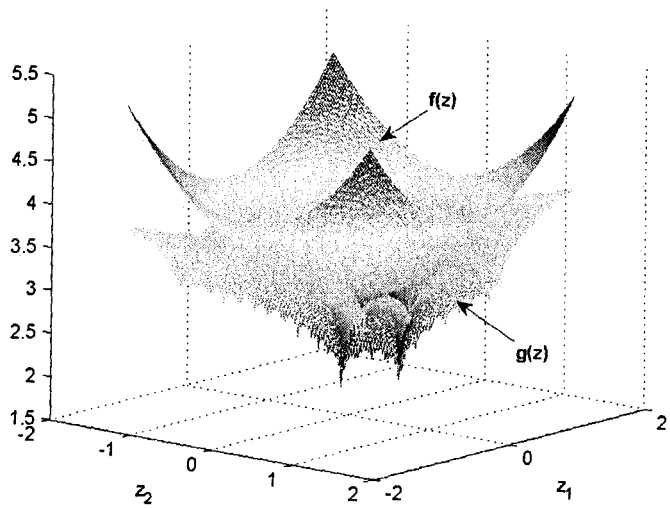



Figure 3.1: A non-convex function $g(\boldsymbol{z})$ that has many local minima and a smooth convex function $f(\boldsymbol{z})$.

***Proof***: See [40]. □

Theorem 3.1 implies that if the purpose of solving (3.1) is to achieve a sparse solution, then the permutation invariant and sign invariant concave functions can be used as diversity measure functions.

The objective of this chapter is to replace $g_c(s)$, and hence $g(\boldsymbol{s})$, by a properly chosen convex function. The motivation behind this approach can be described as follows. Let $\boldsymbol{s}_0$ be any solution vector to (3.1). Then any vector $\boldsymbol{s} = \boldsymbol{s}_0 + \boldsymbol{F}\boldsymbol{z}$; where $\boldsymbol{F}$ is a basis of the null space of $\boldsymbol{A}$, and $\boldsymbol{z} \in \mathbb{R}^{n-m}$ is a random vector, is also a solution vector. Accordingly, (3.1) can be converted into the following unconstrained minimization problem

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) := g(\boldsymbol{s}_0 + \boldsymbol{F}\boldsymbol{z}). \tag{3.2}$$

Note that (3.1) and (3.2) are equivalent, i.e., if $\hat{\boldsymbol{z}}$ is a fixed point to (3.2) then $\hat{\boldsymbol{s}} = \boldsymbol{s}_0 + \boldsymbol{F}\hat{\boldsymbol{z}}$ is also a fixed point to (3.1). A three dimensional plot of $g(\boldsymbol{z})$ is

shown in Figure 3.1. As shown in this figure, the surface of $g(\boldsymbol{z})$ is very rough and has many local minima and hence any minimization technique will fail to converge to a sparse solution. So, the idea presented in this chapter is to iteratively replace $g(\boldsymbol{z})$ by a convex smooth function $f(\boldsymbol{z})$, or equivalently $f(\boldsymbol{s})$, which can be easily minimized. An example of such a function is shown in Figure 3.1. The advantages of the proposed approach is that, by replacing the nonconvex function $g(\boldsymbol{s})$ by the convex function $f(\boldsymbol{s})$, the optimization problem becomes more tractable and can be solved in few iterations. However, since the original function $g(\boldsymbol{s})$ is nonconvex, the proposed algorithm may converge to a local minimum. This last situation could happen when the global minimizer of $f(\boldsymbol{s})$ at a certain iteration belongs to the basin of attraction of one of the local minima of $g(\boldsymbol{s})$.

For the proposed approach to be useful, the function $f(\boldsymbol{s})$ must be selected such that its global minimizer is guaranteed to reduce the original function $g(\boldsymbol{s})$. Towards that end, and to simplify the exposition, we will consider replacing the one dimensional function $g_c(s)$ by a convex function $f(s)$ first, then we generalize to the $n$ dimensional case. Consider the following theorem

**Theorem 3.2** *Given a differentiable one dimensional function $g_c(s) : \mathbb{R} \rightarrow \mathbb{R}$ that obeys P1-P2, a one dimensional convex function $f(s) : \mathbb{R} \rightarrow \mathbb{R}$ with a global minimizer $s^o$, and an initial point $s_0$, then sufficient conditions for $g_c(s)$ to be reduced at $s^o$, i.e., $g_c(s^o) \leq g_c(s_0)$, are: (1) $\text{sign}\,(f'(s_0)) = \text{sign}\,(g_c'(s_0))$, and (2) $|s^o| \leq |s_0|$, where $g'(s_0)$ is the gradient of $g(s)$ at $s = s_0$.*

***Proof***: Since the negative of the gradient of any objective function provides a valid direction for minimizing this objective function, the first condition in Theorem 3.2 is necessary to insure that the minimizing direction of $f(s)$ is a valid minimizing direction for the original function $g(s)$. On the other hand, from P2, it is known that $g(s)$ is sign invariant and monotonically increasing on the nonnegative orthant.

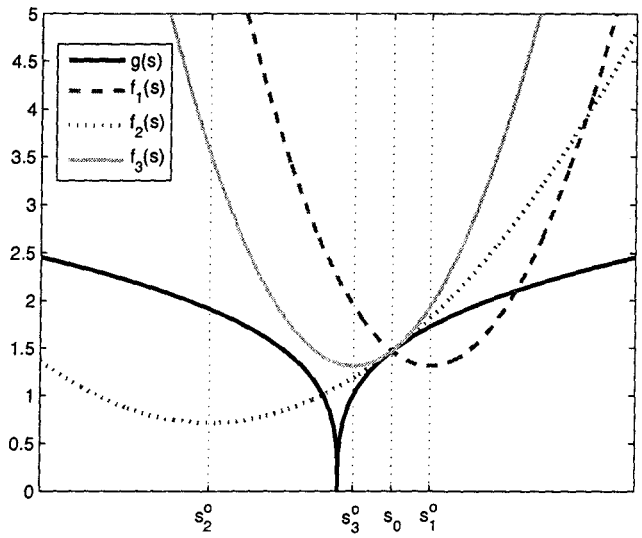



Figure 3.2: A non-convex function $g(s)$ and three different convex functions $f_i(s)$. Only $f_3(s)$ satisfies the conditions of Theorem 3.2.

Accordingly, the second condition in Theorem 3.2 is necessary to insure that $g(s^o) \leq g(s_0)$. □

Figure 3.2 presents three different convex functions that intersect with a nonconvex function $g(s)$ at $s_0$. The first function $f_1(s)$ violates the two conditions of Theorem 3.2, while the second function $f_2(s)$ violates the second condition. As shown in Figure 3.2; $g(s_i^o) > g(s_0)$, $i = 1, 2$; where $s_1^o$ and $s_2^o$ are the global minima of $f_1(s)$ and $f_2(s)$, respectively. On the other hand, $f_3(s)$ in Figure 3.2 is the only function among these three functions that satisfies the two conditions of Theorem 3.2, hence $g(s_3^o) < g(s_0)$

There are many ways of selecting convex objective functions that satisfy the two conditions of Theorem 3.2. However, in this chapter we propose a simple and straight forward technique for doing this task. The proposed function has the following form

$$f(s) = g_c(s_0) + g_c'(s_0)(s - s_0) + \beta_0(s - s_0)^2, \tag{3.3}$$

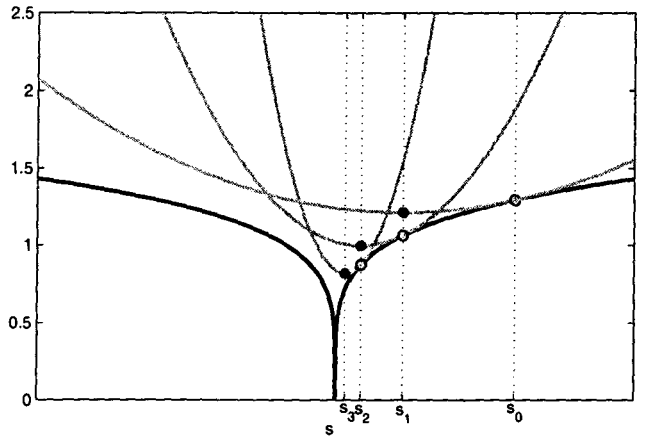



Figure 3.3: Successive iterations of the proposed algorithm for minimizing the nonconvex function $g_c(s)$ (black curve). The hollow circles represent the starting points, while the solid ones represent the global minima of the locally–convex functions.

where $g_c'(s_0)$ is the gradient of $g_c(s)$ at $s_0$, and $\beta_0 \geq 0$. Clearly the first condition of Theorem 3.2 is satisfied, since $f'(s_0) = g_c'(s_0)$, and the second condition can be met by properly selecting the value of $\beta_0$.

*Remark:* If $\beta_0$ in (3.3) is selected as $\beta_0 = -0.5g_c''(s_0)$, where $g_c''(s_0)$ is the Hessian of $g_c(s)$ at $s = s_0$, then the function $f(s)$ in (3.3) is equivalent to the second–order Taylor series expansion of $g_c(s)$, with the sign of the quadratic term reversed. This results in $f(s)$ being a convex function. However it does not guarantee that the second condition of Theorem 3.2 is satisfied.

Assuming that the value of $\beta_0$ was chosen such that $f(s)$, defined in (3.3), satisfies the second condition of Theorem 3.2, then the point $s_1 = \arg\min_s f(s)$ is a global minimizer to $f(s)$ but not to $g(s)$. Accordingly, to reach the global minima of $g(s)$, the procedure has to be repeated, i.e., to set $s_0 \leftarrow s_1$ and select $\beta_0$ such that (3.3) satisfies the conditions of Theorem 3.2. This procedure can be repeated many times till $g(s)$ is globally minimized. This procedure is shown in Figure 3.3.

Since $g(\boldsymbol{s})$ is separable, the one dimensional function (3.3) can be readily extended to the $n$ dimensional case by constructing the following separable convex function

$$\begin{aligned} f(\boldsymbol{s}) &= \sum_{i=1}^{n} f(s[i]) \\ &= \sum_{i=1}^{n} g_c(s_0[i]) + g_c'(s_0[i])(s[i] - s_0[i]) + \beta_0[i](s[i] - s_0[i])^2 \\ &= g(\boldsymbol{s}_0) + (\boldsymbol{s} - \boldsymbol{s}_0)^T \boldsymbol{d}_0 + (\boldsymbol{s} - \boldsymbol{s}_0)^T \boldsymbol{B}_0 (\boldsymbol{s} - \boldsymbol{s}_0), \end{aligned} \tag{3.4}$$

where $\boldsymbol{d}_0 = \nabla g(\boldsymbol{s}_0)$ is the gradient of $g(\boldsymbol{s})$ at $\boldsymbol{s} = \boldsymbol{s}_0$, and $\boldsymbol{B}_0$ is a diagonal positive semidefinite matrix. Instead of using (3.4) in deriving the proposed algorithms, the following simplified function will be used

$$\tilde{f}(\boldsymbol{s}) = \boldsymbol{s}^T (\boldsymbol{d}_0 - 2\boldsymbol{B}_0 \boldsymbol{s}_0) + \boldsymbol{s}^T \boldsymbol{B}_0 \boldsymbol{s}, \tag{3.5}$$

which is equivalent to $f(\boldsymbol{s})$ after dropping the constant terms. Although the constant term in (3.4) does not affect the value of the solution vector of $f(\boldsymbol{s})$, it is important in proving the convergence of the derived algorithms.

## 3.3 The MCCR Algorithms

In this section we derive some algorithms for finding a sparse solution vector of the noise–free linear underdetermined system of equations. The derived algorithms are based on replacing the nonconvex objective function $g(\boldsymbol{s})$ in (3.1) by the convex objective function $f(\boldsymbol{s})$ defined in (3.4), or equivalently $\tilde{f}(\boldsymbol{s})$ defined in (3.5). Since $f(\boldsymbol{s})$ satisfies the first condition of Theorem 3.2, the remaining part of this section considers the problem of selecting $\boldsymbol{B}_0$ such that, for a given starting point $\boldsymbol{s}_0$, the original objective function $g(\boldsymbol{s})$ satisfies $g(\boldsymbol{s}_1) \leq g(\boldsymbol{s}_0)$, where

$$\begin{aligned} \boldsymbol{s}_1 = \arg\min_{\boldsymbol{s}} \quad & \boldsymbol{s}^T (\boldsymbol{d}_0 - 2\boldsymbol{B}_0 \boldsymbol{s}_0) + \boldsymbol{s}^T \boldsymbol{B}_0 \boldsymbol{s} \\ \text{subject to} \quad & \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \end{aligned} \tag{3.6}$$

The optimization problem (3.6) can be solved by following the standard method of Lagrangian multipliers (see, e.g. [85,86]). We define the Lagrangian $L(\boldsymbol{s}, \boldsymbol{\lambda})$ as

$$L(\boldsymbol{s}, \boldsymbol{\lambda}) = \boldsymbol{s}^T (\boldsymbol{d}_0 - 2\boldsymbol{B}_0 \boldsymbol{s}_0) + \boldsymbol{s}^T \boldsymbol{B}_0 \boldsymbol{s} + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{s} - \boldsymbol{x}), \tag{3.7}$$

where $\boldsymbol{\lambda}$ is the $(m \times 1)$ vector of Lagrange multipliers. Since $f(\boldsymbol{s})$ is convex, a necessary and sufficient condition for the Lagrangian $L(\boldsymbol{s}, \boldsymbol{\lambda})$ to be minimized at $\boldsymbol{s}_1$ is that $(\boldsymbol{s}_1, \boldsymbol{\lambda}_*)$ be stationary points of the Lagrangian function, i.e.

$$\nabla_{\boldsymbol{s}} L(\boldsymbol{s}_1, \boldsymbol{\lambda}_*) = \boldsymbol{d}_0 + 2\boldsymbol{B}_0(\boldsymbol{s}_1 - \boldsymbol{s}_0) + \boldsymbol{A}^T \boldsymbol{\lambda}_* = 0, \tag{3.8}$$

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{s}_1, \boldsymbol{\lambda}_*) = \boldsymbol{A}\boldsymbol{s}_1 - \boldsymbol{x} = 0. \tag{3.9}$$

From (3.8) we have

$$\boldsymbol{s}_1 = \boldsymbol{s}_0 - 0.5\boldsymbol{B}_0^{-1}(\boldsymbol{d}_0 + \boldsymbol{A}^T \boldsymbol{\lambda}_*). \tag{3.10}$$

Substituting this equation into (3.9), and solving for $\boldsymbol{\lambda}_*$ we get

$$\begin{aligned} \boldsymbol{\lambda}_* &= -2(\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{A}^T)^{-1}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}_0 + 0.5\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{d}_0) \\ &= -(\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{d}_0, \end{aligned} \tag{3.11}$$

where the second equality holds because $\boldsymbol{s}_0$ is feasible, i.e. $\boldsymbol{A}\boldsymbol{s}_0 = \boldsymbol{x}$. Substituting (3.11) back into (3.10), we get the following general expression for $\boldsymbol{s}_1$:

$$\boldsymbol{s}_1 = \boldsymbol{s}_0 - 0.5\boldsymbol{B}_0^{-1}\boldsymbol{d}_0 + 0.5\boldsymbol{B}_0^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{d}_0. \tag{3.12}$$

Eq. (3.12) represents the general expression of the proposed MCCR algorithm. In the next subsections we propose two different choices for $\boldsymbol{B}_0$ that guarantee $g(\boldsymbol{s})$ is reduced at $\boldsymbol{s}_1$, i.e., $g(\boldsymbol{s}_1) \leq g(\boldsymbol{s}_0)$.

### 3.3.1 First approach: Select $\boldsymbol{B}_0$ such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2$

In this subsection we select $\boldsymbol{B}_0$ such that the global minima of $\tilde{f}(\boldsymbol{s})$, and hence $f(\boldsymbol{s})$, occurs at $\boldsymbol{s}^o = 0$.[2] From (3.4), or equivalently (3.5), the global minima of $f(\boldsymbol{s})$ occurs

[2]It should be noted that, even though the global minimizer of $\tilde{f}(\boldsymbol{s})$ is $\boldsymbol{s}^o = \boldsymbol{0}$, the solutions to the constrained problem (3.6) is given by (3.12) and is generally not $\boldsymbol{s}^o = \boldsymbol{0}$.

at $\boldsymbol{s}^o$, which can be calculated from the following equation

$$\nabla_{\boldsymbol{s}} f(\boldsymbol{s})|_{\boldsymbol{s}^o} = \boldsymbol{d}_0 + 2\boldsymbol{B}_0(\boldsymbol{s}^o - \boldsymbol{s}_0) = 0,$$

which leads to

$$\boldsymbol{s}^o = \boldsymbol{s}_0 - 0.5\boldsymbol{B}_0^{-1}\boldsymbol{d}_0. \tag{3.13}$$

Accordingly, setting $\boldsymbol{s}^o = 0$ is equivalent to

$$\boldsymbol{s}_0 = 0.5\boldsymbol{B}_0^{-1}\boldsymbol{d}_0. \tag{3.14}$$

Substituting (3.14) into (3.12), and utilizing the fact that $\boldsymbol{s}_0$ is feasible, i.e., $\boldsymbol{A}\boldsymbol{s}_0 = \boldsymbol{x}$, the expression of $\boldsymbol{s}_1$ is reduced to

$$\boldsymbol{s}_1 = \boldsymbol{B}_0^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_0^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}. \tag{3.15}$$

Note that $\boldsymbol{s}_1$ is the global minima of the convex problem (3.6) but not of the original problem (3.1). As will be shown later by Theorem 3.3, $g(\boldsymbol{s}_1) \leq g(\boldsymbol{s}_0)$. Therefore, a stable point of (3.1) can be obtained by iteratively repeating (3.15), i.e., for $k \geq 0$

$$\boldsymbol{s}_{k+1} = \boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}, \tag{3.16}$$

where $\boldsymbol{B}_k$ is a diagonal matrix which depends on $\boldsymbol{s}_k$ and $\boldsymbol{d}_k = \nabla g(\boldsymbol{s}_k)$, the gradient of $g(\boldsymbol{s})$ at $\boldsymbol{s}_k$. The diagonal entries of $\boldsymbol{B}_k$ can be easily calculated from (3.14) as follows

$$B_k[i,i] = \frac{d_k[i]}{2s_k[i]}, \quad i = 1, \ldots, n. \tag{3.17}$$

Note that $B_k[i,i]$ calculated from (3.17) is nonnegative, i.e., $\boldsymbol{B}_k$ is positive semidefinite, which is a necessary condition for $f(\boldsymbol{s})$ to be convex. This follows readily from the sign–invariant property of $g(\boldsymbol{s})$ and its monotonically increasing property on the nonnegative orthant $\mathcal{O}_1$. Note that, substituting (3.14) into (3.5), the expression of $\tilde{f}(\boldsymbol{s})$ is reduced to $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}\boldsymbol{s}||_2^2$ as desired, where $\boldsymbol{W} = \boldsymbol{B}_0^{-1/2}$.

Since the original objective function $g(\boldsymbol{s})$ is nonconvex, the convergence of the proposed algorithm (3.16) into the "global" minima of the original problem (3.1) is not guaranteed. However, the proposed algorithm always converges to a fixed point of (3.1) and does not have undesired properties such as divergence or oscillation. The proof of convergence of (3.16) to a fixed point of (3.1) is shown by the following theorem.

**Theorem 3.3** *Given a starting solution vector $\boldsymbol{s}_k \in \mathbb{R}^n$ and a separable function $g(\boldsymbol{s}) = \sum_i g_c(s[i]) : \mathbb{R}^n \rightarrow \mathbb{R}$, where $g_c(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ is differentiable and obeys P2. Let $\boldsymbol{d}_k$ denote the gradient of $g(\boldsymbol{s})$ at $\boldsymbol{s}_k$. If $\boldsymbol{B}_k$ in (3.4) is calculated using (3.17) and a new solution vector $\boldsymbol{s}_{k+1}$ is obtained using (3.16), then $g(\boldsymbol{s}_{k+1}) \leq g(\boldsymbol{s}_k)$ with the equality holds if and only if $\boldsymbol{s}_k$ is a fixed point of the original problem (3.1).*

***Proof***: Since the global minima of $f(\boldsymbol{s})$ occurs at $\boldsymbol{s}^o = 0$, it is straightforward to show that $f(\boldsymbol{s})$ is sign invariant, i.e., $f(\boldsymbol{s}) = f(|\boldsymbol{s}|)$. Since both $g(\boldsymbol{s})$ and $f(\boldsymbol{s})$ are sign invariant, it will be assumed without loss of generality that $\boldsymbol{s}_k \in \mathcal{O}_1$, the nonnegative orthant. From (3.4) it is clear that both $f(\boldsymbol{s})$ and $g(\boldsymbol{s})$ share a common tangent at $\boldsymbol{s}_k$, which is given by the first two terms of (3.4). From the convexity of $f(\boldsymbol{s})$ and the concavity of $g(\boldsymbol{s})$ on $\mathcal{O}_1$, a common tangent to $f(\boldsymbol{s})$ and $g(\boldsymbol{s})$ on $\mathcal{O}_1$ implies that $g(\boldsymbol{s})$ is upper bounded by $f(\boldsymbol{s})$ on $\mathcal{O}_1$. This is also applicable to the other $2^n - 1$ orthants. Therefore, $g(\boldsymbol{s}) \leq f(\boldsymbol{s})$, where the equality holds only at the $2^n$ tangent points, i.e., for a given starting point $\boldsymbol{s}_k \in \mathcal{O}_1$ we have $g(\boldsymbol{s}_k) = f(\boldsymbol{s}_k)$. However, from (3.6), and since $f(\boldsymbol{s})$ is a quadratic convex function, we have $f(\boldsymbol{s}_{k+1}) \leq f(\boldsymbol{s}_k)$ where the equality holds if and only if $\boldsymbol{s}_{k+1} = \boldsymbol{s}_k$, i.e., $\boldsymbol{s}_{k+1}$ can not be one of the other $2^n - 1$ tangent points. In conclusion, if $\boldsymbol{s}_{k+1}$ is different from $\boldsymbol{s}_k$, then $g(\boldsymbol{s}_{k+1}) < f(\boldsymbol{s}_{k+1}) < f(\boldsymbol{s}_k) = g(\boldsymbol{s}_k)$, while $g(\boldsymbol{s}_{k+1}) = f(\boldsymbol{s}_{k+1}) = f(\boldsymbol{s}_k) = g(\boldsymbol{s}_k)$ if and only if $\boldsymbol{s}_{k+1} = \boldsymbol{s}_k$, where the only if part follows from the fact that $\boldsymbol{s}_{k+1}$ is not one of the other tangent points. In this last case, $\boldsymbol{s}_k$ is a fixed point of (3.1). □

Thus, starting from any feasible solution vector $\boldsymbol{s}_0 \in \mathbb{R}^n$, the algorithm (3.16) generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges at least to a local minimum of the original problem (3.1). At this point, $\boldsymbol{s}$ does not vary from one iteration to the next.

Theorem 3.3 shows that any algorithm of the form (3.16) with $\boldsymbol{B}_k$ calculated using (3.17) is guaranteed to converge to a fixed point of the original optimization problem (3.1) as long as $g(\boldsymbol{s}) \in \mathcal{G}$, where $\mathcal{G}$ is the set of all objective functions that obey P1–P2. The exact expression of the objective function $g(\boldsymbol{s})$ affects only the rate of convergence of (3.16). The rate of convergence of some functions $g(\boldsymbol{s}) \in \mathcal{G}$ are derived in Section 3.3.3.

### *Relation with IRLS algorithms*

Define $\boldsymbol{W}_k = \boldsymbol{B}_k^{-\frac{1}{2}}$, then (3.16) can be written as

$$\boldsymbol{s}_{k+1} = \boldsymbol{W}_k(\boldsymbol{A}\boldsymbol{W}_k)^{\dagger}\boldsymbol{x}, \tag{3.18}$$

where $\dagger$ is the Moore-Penrose inverse [77]. Comparing (3.18) with (2.20) we readily find that the derived algorithm has the form of the IRLS algorithms. The convergence of an IRLS algorithm to a sparse solution vector depends on the relation between $\boldsymbol{W}_k$ and the previous solution vector $\boldsymbol{s}_k$. Some IRLS algorithms might not converge to a sparse solution vector, e.g., the algorithm derived in [12] for minimizing the Shannon entropy. However, the proposed technique provides a general methodology for deriving IRLS algorithms that converge to sparse solution vectors and enjoy fixed point convergence, i.e., convergence to a fixed point solution vector starting from any feasible solution vector.

The performance of the proposed algorithm in finding a sparse solution is shown in Figure 3.4. The difference between Figure 3.4 and Figure 3.3 is that the global minimum of each convex function in Figure 3.4 is at the origin. Figure 3.4 shows the successive iterations associated with reducing a zero entry of the solution vector $\boldsymbol{s}$

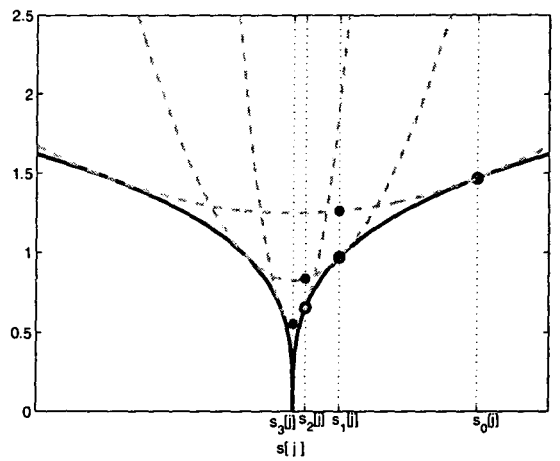



Figure 3.4: Successive iterations of the proposed algorithm for minimizing the nonconvex function $g_c(s[j])$ (black curve). The hollow circles represent the starting points, while the solid ones represent the global minima of the locally-convex functions.

towards its optimum value of zero. As shown in this figure, the algorithm takes large steps when the value of $s[j]$ is large, while the steps get smaller and smaller as $s_k[j]$ approaches zero. This performance is due to the weighting matrix $\boldsymbol{B}_k$ whose $i$th diagonal element is given by (3.17). From (3.17) we can see that as $s_k[i] \rightarrow 0$ the weights become large due to the fact that the function $g_c(s)$ is concave and monotonically increasing on $\mathcal{O}_1$, and hence its gradient increases as the value of $s$ decreases. Large weights associated with small values of $\boldsymbol{s}$ are required for a sparse solution. Note that the quadratic replacement function $f(\boldsymbol{s})$ on the other hand imposes small penalties on small values of $\boldsymbol{s}$, a fact which explains why the MCCR approach can not achieve the sparse solution in one step.

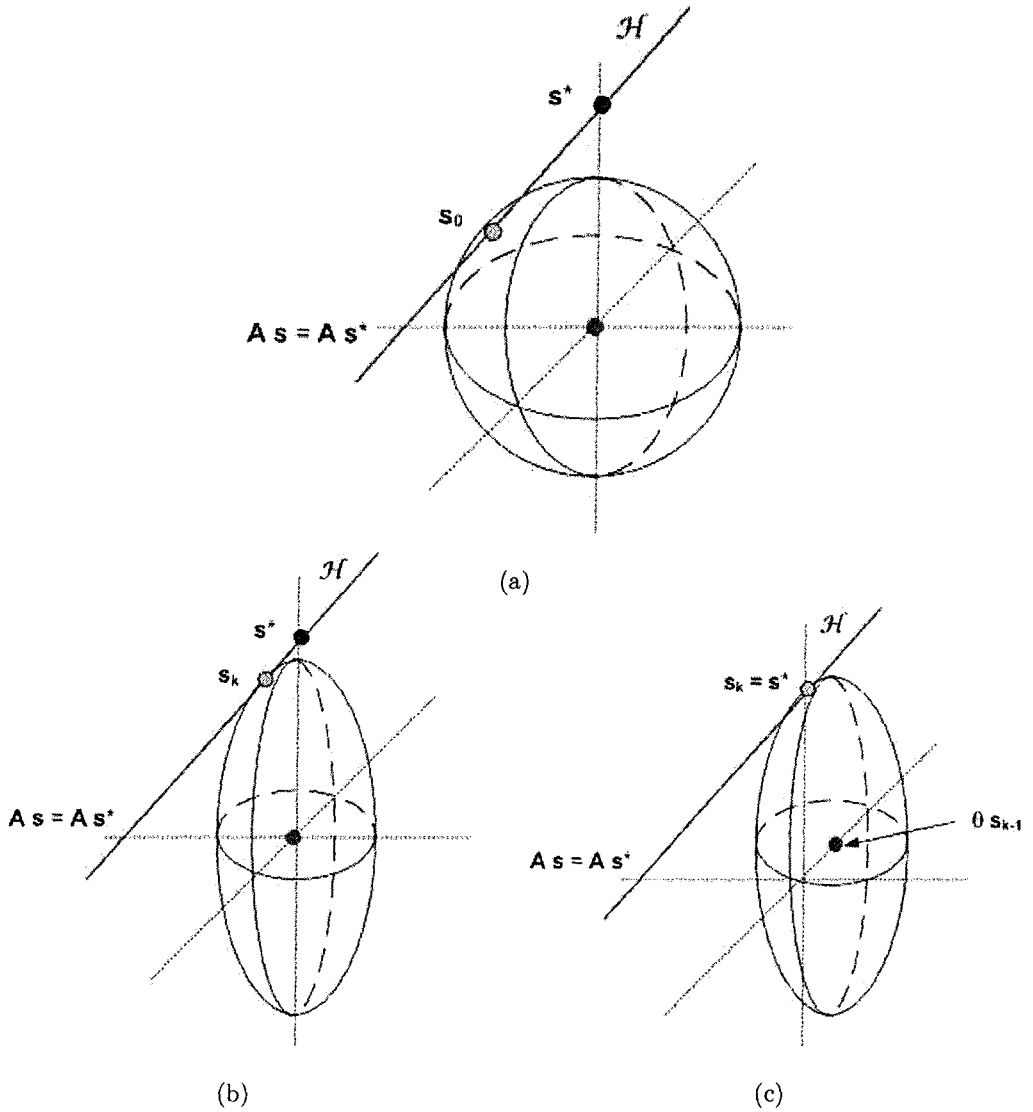



Figure 3.5: Geometric interpretation of the solution vector obtained by: (a) $||\boldsymbol{s}||_2^2$ minimization, (b) $||\boldsymbol{W}_{k-1}^{-1}\boldsymbol{s}||_2^2$ minimization, and (c) $||\boldsymbol{W}_{k-1}^{-1}(\boldsymbol{s}-\theta\boldsymbol{s}_{k-1})||_2^2$ minimization .

### 3.3.2 Second Approach: Select $\boldsymbol{B}_0$ such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s} - \theta \boldsymbol{s}_0)||_2^2$

In Section 3.3.1 we proved that selecting $\boldsymbol{B}_k$ as in (3.17) reduces (3.4) to $f(\boldsymbol{s}) = ||\boldsymbol{W}_k^{-1}\boldsymbol{s}||_2^2$, where $\boldsymbol{W}_k = \boldsymbol{B}_k^{-\frac{1}{2}}$. Consequently, the resulting algorithm (3.16), or equivalently (3.18), has the general form of the IRLS algorithms. Moreover, it was proved by Corollary 3.3 that, starting from any bounded solution vector, the proposed algorithm is guaranteed to converge to a fixed solution vector of the objective function $g(\boldsymbol{s})$. However, the rate of convergence of the proposed algorithm is limited by the nature of the weighted $\ell_2$–ball. In contrast to the $\ell_1$–ball, the $\ell_2$–ball, and hence $\ell_{w2}$–ball, the weighted $\ell_2$–ball, has rounded tips. As a result, the proposed algorithm, and all other IRLS algorithms, may require large number of iterations to converge to a solution vector, especially when $m$ and $n$ are large. This behavior is shown geometrically in Figure 3.5.

In this figure, the exact and the estimated solution vectors are represented by the blue circle at $\boldsymbol{s}^* = [0 \quad 1 \quad 0]^T$ and the green circle, respectively, while $\mathcal{H}$, the set of all points $\boldsymbol{s} \in \mathbb{R}^3$ obeying $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{A}\boldsymbol{s}^*$, is represented by the red line passing through $\boldsymbol{s}^*$. As described in Chapter 2, the $\ell_2$ minimizer of (3.1) is the point on $\mathcal{H}$ closest to the origin. This point can be found by blowing up the $\ell_2$–ball, represented by the hypersphere in Figure 3.5(a), until it contacts $\mathcal{H}$. Due to the randomness of the entries of $\boldsymbol{A}$, $\mathcal{H}$ is oriented at random angle. Accordingly, with high probability, the closest point $\boldsymbol{s}_0$ will lie away from the coordinate axes and hence will be neither sparse nor close to the correct answer $\boldsymbol{s}^*$ [78].

Geometrically, incorporating a diagonal weighting matrix into the $\ell_2$-norm causes the $\ell_2$–ball to elongate along a certain direction and no longer be symmetric. Assuming that the proposed algorithm (3.16) is converging to the exact solution vector $\boldsymbol{s}^*$, then at the $k$th iteration the $\ell_{w2}$–ball will contact $\mathcal{H}$ at $\boldsymbol{s}_k$, which is closer to the exact solution vector $\boldsymbol{s}^*$ as shown in Figure (b). At the next iteration, and due to the new weighting matrix, the $\ell_{w2}$–ball will be further squeezed and contact $\mathcal{H}$ at a

point $\boldsymbol{s}_{k+1}$ closer to $\boldsymbol{s}^*$ than the previous solution point $\boldsymbol{s}_k$. The procedure continues until the $\ell_{w2}$–ball contacts $\mathcal{H}$ at $\boldsymbol{s}^*$.

Since the sparse solution vector $\boldsymbol{s}^*$ is restricted to lie at the tip of the $\ell_{w2}$–ball, algorithm (3.16) may take a large number of iterations to converge to that vector, i.e., for the $\ell_{w2}$–ball to contact $\mathcal{H}$ at $\boldsymbol{s}^*$. However, the number of iterations can be significantly reduced if the sparse solution vector $\boldsymbol{s}^*$ is allowed to lie on the surface of the $\ell_{w2}$–ball. This can be achieved by changing the origin of the $\ell_{w2}$–ball in each iteration of (3.16), where the origin at the $k$th iteration depends on the previous solution vector. The proposed modification is shown in Figure 3.5(c), where the origin of the $\ell_{w2}$–ball is shifted to $\theta \boldsymbol{s}_{k-1}$, where $\theta$ is a parameter to be determined. As shown in this figure, although $\mathcal{H}$ contacts the $\ell_{w2}$–ball at a point away from its tip, this point coincides with the exact solution vector $\boldsymbol{s}^*$, which minimizes the original objective function $g(\boldsymbol{s})$.

Based on this geometric interpretation, we propose selecting $\boldsymbol{B}_0$ in (3.5) such that $\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s} - \theta \boldsymbol{s}_0)||_2^2$, where $\boldsymbol{W}$ is a diagonal weighting matrix and $\boldsymbol{s}_0$ is a given solution vector. Since $\boldsymbol{W}$ is diagonal, $\tilde{f}(\boldsymbol{s})$ can be written as

$$\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}^{-1}(\boldsymbol{s} - \theta \boldsymbol{s}_0)||_2^2 = -2\theta \boldsymbol{s}^T \boldsymbol{W}^{-2} \boldsymbol{s}_0 + \boldsymbol{s}^T \boldsymbol{W}^{-2} \boldsymbol{s} + C, \tag{3.19}$$

where $C$ is a constant term that does not depend on $\boldsymbol{s}$. Equating (3.5) and (3.19) we get

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{B}_0^{-\frac{1}{2}}, \\ B_0[i,i] &= \frac{d_0[i]}{2(1-\theta)s_0[i]}, \quad i = 1, \ldots, n, \end{aligned} \tag{3.20}$$

where the second equation results from equating the first terms in (3.5) and (3.19), and utilizing the fact that $\boldsymbol{B}_0$ is diagonal.

As described in Section 3.3.1, the quantity $d_0[i]/s_0[i]$ is always nonnegative. Accordingly, for $\boldsymbol{B}_0$ in (3.20) to be nonnegative definite, the value of $\theta$ must be less

than 1. As will be shown later, this is also a necessary condition for (3.19) to produce a sequence of solution vectors that minimize the original objective function, i.e., $g(\boldsymbol{s}_{k+1}) < g(\boldsymbol{s}_k)$, for all $k \geq 0$.

Based on this formulation, the optimization problem (3.6) is reduced to

$$\boldsymbol{s}_1 = \arg\min_{\boldsymbol{s}} \quad ||\boldsymbol{W}_0^{-1}(\boldsymbol{s} - \theta\boldsymbol{s}_0)||_2^2 \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}, \tag{3.21}$$

where $\boldsymbol{W}_0$ is a diagonal weighting matrix calculated from (3.20). Equation (3.21) is not in the standard form of a weighted minimum norm problem. Accordingly, for solving this problem, we follow the following steps. First, assuming that $\boldsymbol{W}_0$ is invertible, let $\boldsymbol{y}$ be defined as

$$\boldsymbol{y} \triangleq \boldsymbol{W}_0^{-1}(\boldsymbol{s} - \theta\boldsymbol{s}_0).$$

Then, $\boldsymbol{s}$ is expressed as

$$\boldsymbol{s} = \boldsymbol{W}_0\boldsymbol{y} + \theta\boldsymbol{s}_0. \tag{3.22}$$

Finally, substituting (3.22) into (3.21) we get

$$\tilde{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}} \quad ||\boldsymbol{y}||_2^2 \quad \text{subject to} \quad \bar{\boldsymbol{x}} = \bar{\boldsymbol{A}}\boldsymbol{y}, \tag{3.23}$$

where $\bar{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{W}_0$ and $\bar{\boldsymbol{x}} = \boldsymbol{x} - \theta\boldsymbol{A}\boldsymbol{s}_0 = (1-\theta)\boldsymbol{x}$, where in the second equality we used the fact that $\boldsymbol{s}_0$ is feasible, i.e., $\boldsymbol{A}\boldsymbol{s}_0 = \boldsymbol{x}$. Equation (3.23) is a minimum norm problem in the standard form and has the following solution

$$\tilde{\boldsymbol{y}} = (\bar{\boldsymbol{A}})^\dagger\bar{\boldsymbol{x}} = (1-\theta)(\boldsymbol{A}\boldsymbol{W}_0)^\dagger\boldsymbol{x}. \tag{3.24}$$

Substituting (3.24) into (3.22), the solution vector at the $(k+1)$th iteration is expressed as

$$\begin{aligned} \boldsymbol{s}_{k+1} &= \theta\boldsymbol{s}_k + (1-\theta)\boldsymbol{W}_k(\boldsymbol{A}\boldsymbol{W}_k)^\dagger\boldsymbol{x}. \\ &= \theta\boldsymbol{s}_k + (1-\theta)\boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}, \end{aligned} \tag{3.25}$$

where $\boldsymbol{B}_k$ is calculated using (3.20) and $\boldsymbol{s}_0$ and $\boldsymbol{d}_0$ have been replaced by $\boldsymbol{s}_k$ and $\boldsymbol{d}_k$, respectively, where $\boldsymbol{d}_k = \nabla g(\boldsymbol{s}_k)$. In the remaining part of this thesis, the acronyms MCCR and IRLS will be used to refer to (3.25)and (3.16), respectively .

***Selecting the value of $\theta$***

The performance of the MCCR algorithm (3.25) depends on the value of $\theta$. From (3.25) we note that multiplying $\boldsymbol{B}_k$ by any scaling parameter does not affect the value of $\boldsymbol{s}_{k+1}$. Accordingly, the constant term in the expression of $B_k[i,i]$ in (3.20) can be dropped without affecting the value of $\boldsymbol{s}_{k+1}$. As a result, the dependency of $\boldsymbol{B}_k$ on the unknown parameter $\theta$ can be dropped by calculating $B_k[i,i]$, or equivalently $B_k^{-1}[i,i]$, using the following equation

$$B_k^{-1}[i,i] = \frac{s_k[i]}{d_k[i]}, \quad i = 1, \ldots, n, \tag{3.26}$$

and consequently, (3.25) can be written as

$$\boldsymbol{s}_{k+1} = \theta \boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1}, \tag{3.27}$$

where $\tilde{\boldsymbol{s}}_{k+1} = \boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$ is the IRLS solution vector obtained using (3.16). Eq. (3.27) shows that the MCCR solution at the $(k+1)$th iteration is an *affine* combination of the solution vector at the $k$th iteration and the IRLS solution at the $(k+1)$th vector. Geometrically this means that the solution vector $\boldsymbol{s}_{k+1}$ exists at a point determined by the parameter $\theta$, somewhere along the line connecting $\boldsymbol{s}_k$ and $\tilde{\boldsymbol{s}}_{k+1}$. Note that $\boldsymbol{s}_{k+1}$ calculated using (3.27) is always feasible, i.e., $\boldsymbol{A}\boldsymbol{s}_{k+1} = \boldsymbol{x}$. Accordingly, we will refer to the set of all affine combinations of $\boldsymbol{s}_k$ and $\tilde{\boldsymbol{s}}_{k+1}$ for various values of $\theta$ as the *feasible*–line.

In Section 3.3.1 we proved that the IRLS algorithms derived in this chapter are fixed point converging, i.e., for a given feasible solution vector $\boldsymbol{s}_k$, the IRLS solution vector $\tilde{\boldsymbol{s}}_{k+1}$ in (3.27) satisfies $g(\tilde{\boldsymbol{s}}_{k+1}) \leq g(\boldsymbol{s}_k)$. This implies that the direction from $\boldsymbol{s}_k$

to $\tilde{\boldsymbol{s}}_{k+1}$ on the *feasible*–line provides a descending direction for the original objective function $g(\boldsymbol{s})$. This in turn implies that $\theta$ must satisfy the condition ($\theta \leq 1$) in order that $g(\boldsymbol{s}_{k+1}) \leq g(\boldsymbol{s}_k)$. The optimum value of $\theta$, denoted $\theta_o$, is the one that minimizes $g(\boldsymbol{s})$ for all $\boldsymbol{s}$ belonging to the *feasible*–line. Accordingly, $\theta_o$ can be determined by solving the following optimization problem

$$\theta_o = \arg\min_{\theta} \quad g(\theta \boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1}) \quad \text{subject to} \quad \theta_{min} < \theta < 1, \tag{3.28}$$

where $\theta_{min}$ is a lower bound on $\theta$. This formulation is readily implemented using, e.g., the Golden-section search method [87], and can be solved using the Matlab function "*fminbnd.m*". In the simulation results to be shown we empirically set $\theta_{min} = -2$.

Another advantage of the MCCR solution (3.25) over the IRLS solution (3.16) is that (3.25) can be easily adapted to solve the following problem

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}} g(\boldsymbol{s}) \quad \text{subject to} \quad \boldsymbol{s} \geq 0, \quad \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}. \tag{3.29}$$

This can be readily done by selecting a starting feasible solution vector and properly selecting the value of $\theta_{min} \leq 1$ in (3.28) to ensure that $\theta \boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1} \geq 0$ for all $\theta_{min} \leq \theta \leq 1$.

### 3.3.3 Proposed objective functions

The analysis presented so far is general and applicable to any objective function of the form $g(\boldsymbol{s}) = \sum_i g_c(s[i])$, where $g_c(s[i])$ obeys P1–P2. Examples of $g_c(s)$ that were extensively used in the literature are $g_{log}(s) = \log(|s|)$ and $g_q(s) = |s|^q$ with $0 < q < 1$ [12, 13, 71, 72]. In this chapter we propose a class of objective functions that depend on a parameter $\delta > 0$, which, if adjusted properly, can greatly influence the penalties imposed on the entries of the solution vector as the algorithm converges to a sparse solution vector. To be specific, consider the function $g_c(s) = g_{atan}(s) = \text{atan}(|s|/\delta)$.

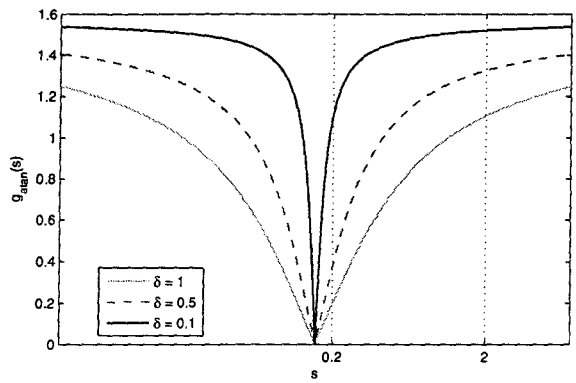



Figure 3.6: $g_{atan}(s) = \text{atan}(|s|/\delta)$ for different values of $\delta$.

This function is concave on $\mathcal{O}_1$ for all $\delta > 0$. However, the shape of $g_{atan}(s)$ depends greatly on the value of $\delta$ as shown in Figure 3.6.

Recall from Section 3.3.1 that selecting $\boldsymbol{B}_k$ as in (3.17) reduces $\tilde{f}(\boldsymbol{s})$ into the form

$$\tilde{f}(\boldsymbol{s}) = ||\boldsymbol{W}_k^{-1}\boldsymbol{s}||_2^2 = \sum_i \frac{d_k[i]}{2s_k[i]} s^2[i], \tag{3.30}$$

where we used the result that $\boldsymbol{W}_k^{-2} = \boldsymbol{B}_k$, and the expression of $B_k[i,i]$ is substituted from (3.17). As discussed before, the weights in (3.30) should be selected such that large weights are associated with small values of $\boldsymbol{s}$ while small weights are associated with large values. However (3.30) shows that the weight associated with $s[i]$ depends, not only on the value of the previous solution $s_k[i]$, but also on the gradient of $g_c(s)$ at $s_k[i]$. Accordingly, $g_c(s)$ should be chosen such that its gradient at any point $s$ varies inversely with the value of that point. This property is satisfied for any function obeying P1–P2. For example, for $g_c(s) = g_{atan}(s)$ the gradient at $s_k[i] \in \mathcal{O}_1$ is given by

$$d_k[i] = \frac{\delta}{\delta^2 + s_k^2[i]} \propto \begin{cases} 1/\delta, & s_k[i] \ll \delta \\ \delta, & s_k[i] \gg \delta \end{cases} \tag{3.31}$$

which varies inversely with $s_k[i]$.

For a given value of $s_k[i]$, (3.31) also shows the effect of the value of $\delta$ on the value of $d_k[i]$. This effect is shown in Figure 3.6. In this figure, the gradient of $g_{atan}(2)$ decreases as $\delta$ decreases from 1 to 0.1, while the opposite is true for $g_{atan}(0.2)$. Accordingly, to obtain the most appropriate weights in (3.30), the value of $\delta$ must be chosen as small as possible, and it is clear from (3.31) and Figure 3.6 that, as $\delta \rightarrow 0$, $\sum_i g_{atan}(s[i]) \rightarrow \zeta ||\boldsymbol{s}||_0$ for some $\zeta > 0$. However, it was empirically observed that the number of local minima of $g_{atan}(s)$ increases as $\delta$ decreases [3]. To overcome this difficulty, we suggest starting the algorithm with a reasonably large value of $\delta$, e.g., $\delta = 1$, and then decreasing its value as the algorithm progress and converges to a sparse solution vector. The strategy of reducing the value of $\delta$ is an open problem, and different reduction strategies can result in different performance. The following strategies can be used for reducing the value of $\delta$:

1. Linear reduction: $\delta_k = \delta_0 - \Delta_\delta k, \quad 0 < \Delta_\delta \ll 1, \quad k = 1, 2, \ldots$.

2. Exponential reduction: $\delta_k = a\delta_{k-1}, 0 < a < 1, \quad k = 1, 2, \ldots$.

3. Performance dependent: starting with $\delta = 1$ then, if the condition $\frac{||\boldsymbol{s}_k - \boldsymbol{s}_{k-1}||_2}{||\boldsymbol{s}_{k-1}||_2} < \frac{\sqrt{\delta}}{100}$ is satisfied, the value of $\delta$ is reduced as $\delta \leftarrow \delta/10$.

4. Solution dependent: $\delta_k = \mathcal{I}(\boldsymbol{s}_{k-1})$, where $\mathcal{I}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a function of the previous solution vector.

The first two choices reduce the value of $\delta$ in each iteration regardless of the convergence of the algorithm, while the value of $\delta$ in the last two choices depends on the convergence of the algorithm. The third choice reduces the value of $\delta$ in steps depending on the value of $\epsilon_k = ||\boldsymbol{s}_k - \boldsymbol{s}_{k-1}||_2 / ||\boldsymbol{s}_{k-1}||_2$, while the last choice selects the value of $\delta$ depending on the the previous solution vector $\boldsymbol{s}_{k-1}$ and the function

[3] This statement is not proved but observed by some numerical experiments, see e.g., Figure 3.7(a).

Table 3.1: Some objective functions that can be used with MCCR.

| Function | $d[i]$ | $B^{-1}[i,i]$ | Order of the local rate of convergence |
|---|---|---|---|
| $g(\boldsymbol{s}) = \sum_i \lvert s[i]\rvert^q,\ 0 < q < 1$ | $qs[i]\lvert s[i]\rvert^{q-2}$ | $\lvert s[i]\rvert^{2-q}$ | $2-q$ |
| $g(\boldsymbol{s}) = \sum_i \log(\lvert s[i]\rvert)$ | $\frac{s[i]}{\lvert s[i]\rvert^2}$ | $\lvert s[i]\rvert^2$ | 2 |
| $g(\boldsymbol{s}) = \sum_i \log(1+\lvert s[i]\rvert/\delta)$ | $\frac{s[i]}{\lvert s[i]\rvert(\delta+\lvert s[i]\rvert)}$ | $\lvert s[i]\rvert(\delta+\lvert s[i]\rvert)$ | 1, $(\delta \gg \epsilon)$<br>2, $(\delta \ll \epsilon)$ |
| $g(\boldsymbol{s}) = \sum_i \operatorname{atan}(\lvert s[i]\rvert/\delta)$ | $\frac{\delta s[i]}{\lvert s[i]\rvert(\delta^2+s^2[i])}$ | $\lvert s[i]\rvert(\delta^2+s^2[i])$ | 1, $(\delta^2 \gg \epsilon^2)$<br>3, $(\delta^2 \ll \epsilon^2)$ |
| $g(\boldsymbol{s}) = \sum_i \frac{\lvert s[i]\rvert}{\lvert s[i]\rvert+\delta}$ | $\frac{\delta s[i]}{\lvert s[i]\rvert(\delta+s[i])^2}$ | $\lvert s[i]\rvert(\delta+s[i])^2$ | 1, $(\delta \gg \epsilon)$<br>3, $(\delta \ll \epsilon)$ |

$\mathcal{I}(\cdot)$. If the solution vector is *l-sparse* and the value of $l$ is known, then the choice $\mathcal{I}(\boldsymbol{s}_{k-1}) = \left|s_{k-1}^{(l+1)}\right|$ is appropriate, where $\left|s_{k-1}^{(l+1)}\right|$ is the $(l+1)$th largest element of $|\boldsymbol{s}_{k-1}|$. This choice insures that large penalties are imposed on entries of the solution vectors that should be zero while small penalties are imposed on the nonzero entries of the solution vector. If the value of $J$ is unknown, then $\mathcal{I}(\cdot)$ can be chosen as $\mathcal{I}(\boldsymbol{s}_{k-1}) = \rho\ \text{mean}(|\boldsymbol{s}_{k-1}|)$, where $0 < \rho \leq 1$.

In addition to $g_{atan}(s)$ we propose the following functions $g_{log_\delta}(s) = \log\left(\frac{|s|}{\delta}+1\right)$, and $g_{s/s} = \frac{|s|}{|s|+\delta}$, where $\delta > 0$. The expressions for $B^{-1}[i,i]$ for five different objective functions that can be used with MCCR, are presented in Table 3.1. In Table 3.1 all the constant terms are dropped from the expression of $B^{-1}[i,i]$.

## Local rate of convergence

As shown in Section 3.3.1, the derived algorithm (3.12) is equivalent to the IRLS algorithm with $\boldsymbol{W}_k = \boldsymbol{B}_k^{-\frac{1}{2}}$. Accordingly, for each proposed objective function, the

local rate of convergence can be derived by following the procedure proposed in [10] and utilizing the corresponding expression of $\boldsymbol{W}_k$, or equivalently $\boldsymbol{B}_k^{-\frac{1}{2}}$, from Table 3.1. The calculated local rates of convergence for the proposed objective functions are listed in the right column in Table 3.1. The value of $\epsilon$ in the table is defined in (3.34).

As shown in this Table, the local rate of convergence of $\ell_q$-norm, $0 < q \leq 1$, depends on the value of $q$. The convergence is linear for $q = 1$, and the algorithm approaches quadratic convergence as $q \rightarrow 0$. On the other hand, the local rate of convergence of the last three objective functions listed in Table 3.1 depends on the relation between $\delta$ and $\epsilon$. The smaller the value of $\delta$ compared to the value of $\epsilon$, the faster the local convergence. This behavior is demonstrated in the simulation results presented in the next section.

## 3.4 Performance Enhancement

In this section the performance of the derived algorithms is discussed. Since the MCCR solution (3.25) is an affine combination of the previous solution and the IRLS solution (3.16), we will consider only the factors that affect the performance of the IRLS solution (3.16). The discussion presented in this section is general and is applicable to all of the objective functions in Table 3.1.

There are two issues that can affect the performance of the updating equation (3.16). These issues, as well as their mitigating interventions, are now discussed.

### 3.4.1 Inversion Operation Associated with (3.16)

The first issue is the inversion operation associated with (3.16). Define $\boldsymbol{C}_k = \boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T$. Then for the solution of (3.16) to exist, $\boldsymbol{C}_k$ must be invertible. Recall from (3.17)

that [4]

$$B_k^{-1}[i,i] = \frac{s_k[i]}{d_k[i]}, \quad i = 1, \ldots, n. \tag{3.32}$$

Accordingly, $\boldsymbol{C}_k$ can be expressed as

$$\boldsymbol{C}_k = \sum_i^n \frac{s_k[i]}{d_k[i]} \boldsymbol{a}_i \boldsymbol{a}_i^T, \tag{3.33}$$

where $\boldsymbol{a}_i$ is the $i$th column of $\boldsymbol{A}$. Thus $\boldsymbol{C}_k$ is a summation of $n$ rank-1 $(m \times m)$ matrices. Accordingly, for $\boldsymbol{C}_k$ to be invertible, at least $m$ elements of $\boldsymbol{s}_k$ must be significant from zero, which contradicts the assumption that $\boldsymbol{s}$ is sparse. To overcome this difficulty, we redefine $B_k^{-1}[i,i]$ as follows

$$B_k^{-1}[i,i] = \begin{cases} \frac{s_k[i]}{d_k[i]} & \text{if} \quad |s_k[i]| \geq \epsilon \\ \frac{s_k[i]}{d_k[i]}\Big|_{s_k[i] \leftarrow \epsilon}, & \text{otherwise,} \end{cases} \tag{3.34}$$

where $\epsilon$ is a small positive number. The second line in (3.34) means that, if the condition $|s_k[i]| \geq \epsilon$ is not satisfied, then $B_k^{-1}[i,i]$ is calculated by evaluating $s_k[i]/d_k[i]$ first then replacing each $s_k[i]$ by $\epsilon$. For example, for $g_c(s) = |s|^q$ we have $d = qs|s|^{q-2}$, and if $|s| < \epsilon$ then the corresponding entry of $\boldsymbol{B}^{-1}$ will equal $\epsilon^{2-q}/q$.

Unfortunately, as will be shown in the simulation results, the performance of MCCR depends on the value of $\epsilon$. To overcome this difficulty, we follow the procedure suggested in [13]. Here, $\epsilon$ is initiated to a relatively large value, e.g. $\epsilon = 1$. Its value is then reduced by a factor of 10 at iterations where the condition $||\boldsymbol{s}_k - \boldsymbol{s}_{k-1}||/||\boldsymbol{s}_k|| < \sqrt{\epsilon}/100$ is satisfied. The MCCR algorithm is summarized in Table 3.2.

### 3.4.2 Non–convexity of $g(s)$

The second issue that affects the performance of (3.16), and hence (3.25), is the non–convexity of the objective functions considered in this chapter. As a result, MCCR may converge to a local minima. This problem may be partially alleviated by the

[4] In (3.32) the constant associated with $B_k^{-1}[i,i]$ is neglected.

Table 3.2: The MCCR Algorithm

Algorithm 1: The MCCR Algorithm

Given an $(m \times n)$ matrix $\boldsymbol{A}$ of basis vectors, and a vector $\boldsymbol{x} \in \mathbb{R}^m$, select one of the objective functions in Table 3.1, a "large" value for $\epsilon$, e.g. $\epsilon = 1$ , an empirically–selected value for $\theta_{min}$, e.g. $\theta_{min} = -2$, a small threshold $\beta$, and an initial feasible point $\boldsymbol{s}_0$. This point can be selected as the least squares solution, i.e. $\boldsymbol{s}_0 = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$. Then set $k = 0$ and repeat the following steps:

1. Repeat until convergence:

   - Calculate $\boldsymbol{B}_k^{-1}$ using (3.34).
   - Calculate $\tilde{\boldsymbol{s}}_{k+1} = \boldsymbol{B}_k^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{B}_k^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{x}$.
   - Calculate $\theta$ using (3.28).
   - Set $\boldsymbol{s}_{k+1} = \theta\boldsymbol{s}_k + (1-\theta)\tilde{\boldsymbol{s}}_{k+1}$.
   - **if** $||\boldsymbol{s}_{k+1} - \boldsymbol{s}_k||_2/||\boldsymbol{s}_{k+1}||_2 < \sqrt{\epsilon}/100$, set $\epsilon = \epsilon/10$ end
   - Set $k = k+1$,

End

2. Output $\boldsymbol{s}_{k+1}$ as the solution.

*perturbed* MCCR algorithm (PMCCR) described in Table 3.3. At each iteration, the solution vector $\boldsymbol{s}_j^0$ is perturbed by a random noise vector $\boldsymbol{v}$, which is constrained to be in the null space of the mixing matrix; i.e. $\boldsymbol{v} = \boldsymbol{F}\boldsymbol{u}$ where $\boldsymbol{F}$ is any matrix whose range is the null space of $\boldsymbol{A}$ and $\boldsymbol{u} \in \mathbb{R}^{n-m}$ is a random noise vector. The columns of $\boldsymbol{F}$ can be chosen by first calculating the singular value decomposition (SVD) of $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, and then selecting the columns of $\boldsymbol{F}$ as the last $(n-m)$ columns of $\boldsymbol{V}$. In the simulation results, the elements of $\boldsymbol{u}$ are sampled from a zero mean uniformly distributed random variable between $\pm\alpha\bar{s}_{max}$, where $\bar{s}_{max}$ is the maximum absolute element in the MCCR solution vector $\bar{\boldsymbol{s}}$, and $\alpha$ is an empirically determined non–negative number in the range e.g., $1 \leq \alpha \leq 2$. Note that the perturbation noise is constrained to be in the null space of the mixing matrix to insure the feasibility of the new perturbed vector. A stopping criterion could be a maximum number of iterations, or a pre–specified value of the cardinality of the solution. Note that, in each step, the new solution vector is accepted only if its cardinality is less than the cardinality of the previous solution vector. Accordingly, by following this strategy, it is guaranteed that no performance degradation occurs.

## 3.5 Simulation Results

In this section, a set of examples are presented in order to examine the effect of the parameters $\delta$, $\epsilon$, and $\theta$ on the performance of the MCCR algorithm, and to provide a comparison between the MCCR algorithm and other well known algorithms. In the first Example, the effect of the parameter $\delta$ on the performance of MCCR is presented. In this example $g_{atan}(\boldsymbol{s})$ is used as an objective function. The second example demonstrates the effect of $\epsilon$ on the performance of MCCR. Since the objective functions proposed in Table 3.1 can be classified into two groups, according to their dependency on $\delta$, two different objective functions are used in this example, one for

Table 3.3: The PMCCR Algorithm

Algorithm 2: The Perturbed MCCR Algorithm (PMCCA)

*Initialization:* Select one of the objective functions in Table 3.1; select a feasible solution $\boldsymbol{s}_0$; Choose $\alpha \in [1, 2]$; set iteration index $j = 0$.

1. For the selected objective function, execute MCCR with initial value $\boldsymbol{s}_0$ to obtain $\bar{\boldsymbol{s}}_0^0$.

2. Repeat until convergence:

   - evaluate perturbed feasible solution $\boldsymbol{s}_j^p = \bar{\boldsymbol{s}}_j^0 + \boldsymbol{F}\boldsymbol{u}$. Refer to the text for the definitions of $\boldsymbol{F}$ and $\boldsymbol{u}$.
   - execute MCCR with initial value $\boldsymbol{s}_j^p$ to obtain $\bar{\boldsymbol{s}}_j^1$.
   - **if** $||\bar{\boldsymbol{s}}_j^1||_{\ell_0} \leq ||\bar{\boldsymbol{s}}_j^0||_{\ell_0}$, $\bar{\boldsymbol{s}}_{j+1}^0 = \bar{\boldsymbol{s}}_j^1$.

     **else** $\bar{\boldsymbol{s}}_{j+1}^0 = \bar{\boldsymbol{s}}_j^0$.
   - $j \leftarrow j + 1$

End

3. Output $\bar{\boldsymbol{s}}_{j+1}^0$ as the solution.

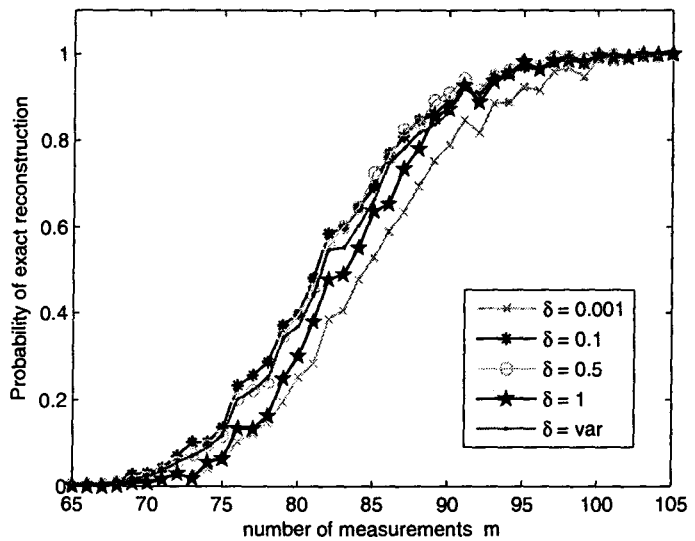
each group. The first objective function is $g_{atan}(s)$, which depends on $\delta$, and the second objective function is $g_{log}(s)$. A comparison between the proposed MCCR algorithms and the counterpart IRLS algorithms is presented in the third example. This example reflects the effect of the parameter $\theta$. The fourth example presents a comparison between the five objective functions in Table 3.1. Finally, the fifth example presents a comparison between the MCCR algorithm and some algorithms that are usually utilized for estimating sparse vectors.

In all these examples, two parameters are used as measures of performance. The first parameter is the probability of exact reconstruction (PER) of the solution vector. The PER is defined as the ratio between the number of runs at which the algorithm successfully estimates the sparse solution vector, to the total number of runs. The second parameter is the average number of iterations taken by each algorithm to converge to a solution vector. Each one of these two parameters is plotted as a function of the number of measurements ($m$).

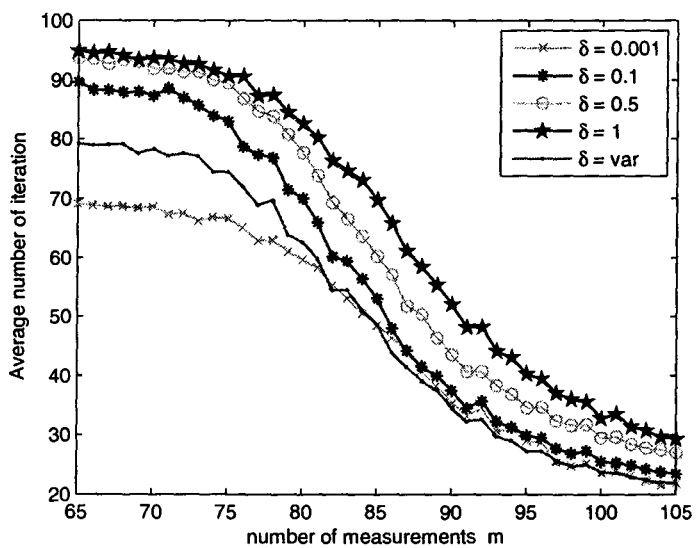
**Example 1: Effect of $\delta$.**

In this example, we examine the effect of $\delta$ on the performance of MCCR for the case $g_c(s) = g_{atan}(s)$. The results are also applicable to $g_{log_\delta}(s)$ and $g_{s/s}(s)$. In this example, $n = 256$, and the number of nonzero elements of $\boldsymbol{s}$ is $i = 40$, while $m$ increases from 65 to 105. For each value of $m$, a random $(m \times n)$ matrix $\boldsymbol{A}$ is created whose entries are each Gaussian random variables with zero mean and unit variance. A sparse vector $\boldsymbol{s}_s$ with $i$ nonzero entries is then created and the corresponding $\boldsymbol{x}$ is generated as $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}_s$. The indices of these $i$ entries of $\boldsymbol{s}_s$ are randomly selected, and their values are chosen randomly from a zero mean Gaussian random variable with variance = 4. The total number of runs in this example is 500.

The compared values of $\delta$ are: 0.001, 0.1, 0.5, 1, and a variable value that depends on the previous solution vector, i.e., $\delta_k = \mathcal{I}(\boldsymbol{s}_{k-1})$, with $\mathcal{I}(\boldsymbol{s}_{k-1}) = 0.5\text{mean}(|\boldsymbol{s}_{k-1}|)$.




(a)




(b)

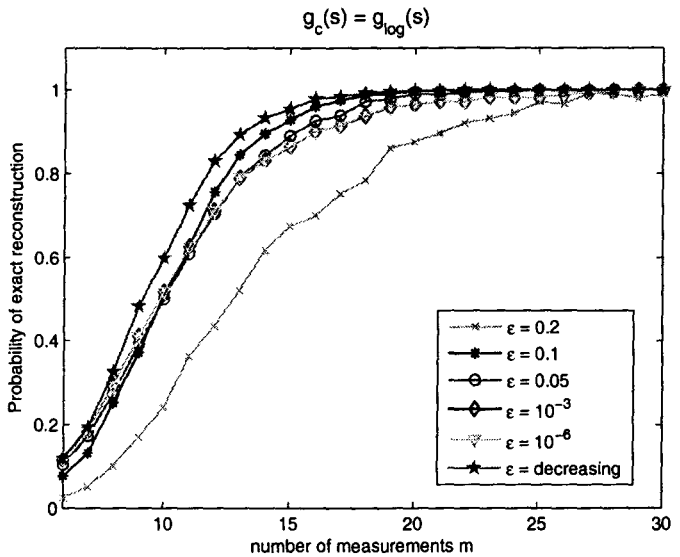
Figure 3.7: The effect of $\delta$ on the performance of MCCR when $g_c(s) = g_{atan}(s)$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$); (b) The average number of iterations required to get a sparse solution vector.

The results are shown in Figure. 3.7. As shown in Figure 3.7(b), the smaller the value of $\delta$, the faster the convergence of the MCCR algorithm. However, in terms of the PER presented in Figure 3.7(a), there is no direct relationship between the value of $\delta$ and the PER. For example, $\delta = 0.001$ produced the worst performance, while the best performance is obtained when $\delta = 0.5$, not $\delta = 1$. On the other hand, it is clear from Figure 3.7 that varying $\delta$ in the proposed manner provides a good tradeoff between the PER and the average number of iteration. However, selecting an optimal value for $\delta$ is an open problem, and other choices could provide better performance.

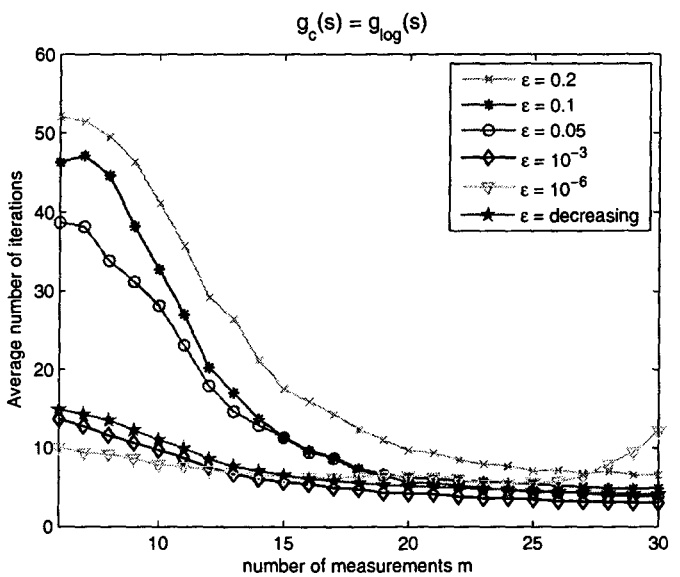
**Example 2: Effect of $\epsilon$.**

In this example, the effect of $\epsilon$ on the performance of MCCR is examined. In this example we select two different objective functions. The first one is $g_{atan}(s)$, which depends on $\delta$, and the other one is $g_{log}(\boldsymbol{s})$. The motivation behind choosing these two objective functions is to check whether the parameter $\delta$ in $g_{atan}(s)$ affects its sensitivity to the variation in the value of $\epsilon$ or not. Accordingly, two different values of $\delta$ are used. The large one is $\delta = 1$, while the small one is $\delta = 0.1$. In this example, $n = 40$, $i = 3$, and $m$ increases from 6 to 30. For each value of $m$, $\boldsymbol{A}$ and $\boldsymbol{s}_s$ are generated as in Example 1. The total number of runs in this example is 1000. The compared values of $\epsilon$ are: 0.2, 0.1, 0.01, $10^{-3}$, $10^{-6}$, and a decreasing value of $\epsilon$ as presented in Table 3.2. The results are shown in Figures 3.8–Figure 3.10.

For the two objective functions shown in these figures and generally speaking, the number of iterations decreases as the value of $\epsilon$ decreases, and the highest PER is obtained when $\epsilon$ decreases in a manner described in Table 3.2. In terms of the PER, it is clear from Figure 3.9–Figure 3.10 that the performance of $g_{atan}$ depends on the value of $\delta$. For the case when $\delta = 1$, it is clear from Figure 3.9(a) that the PER improves as the value of $\epsilon$ decreases from 0.2 to $10^{-3}$, while no improvement is observed by reducing $\epsilon$ beyond $10^{-3}$. On the other hand, when $\delta = 0.1$ the performance of

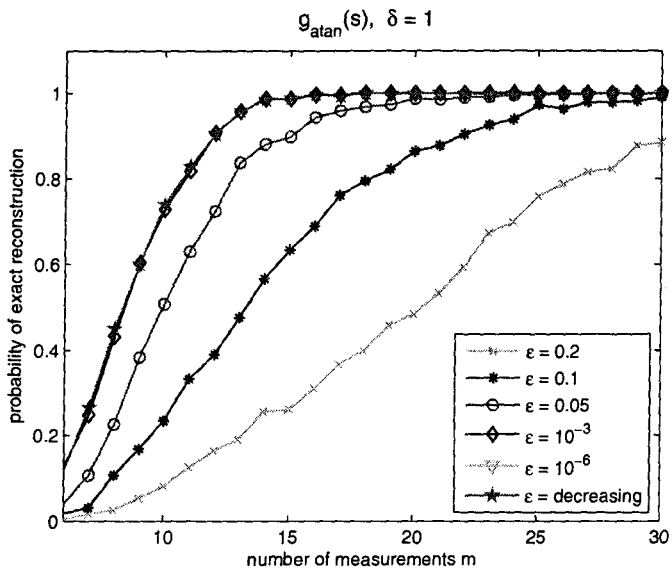



(a)

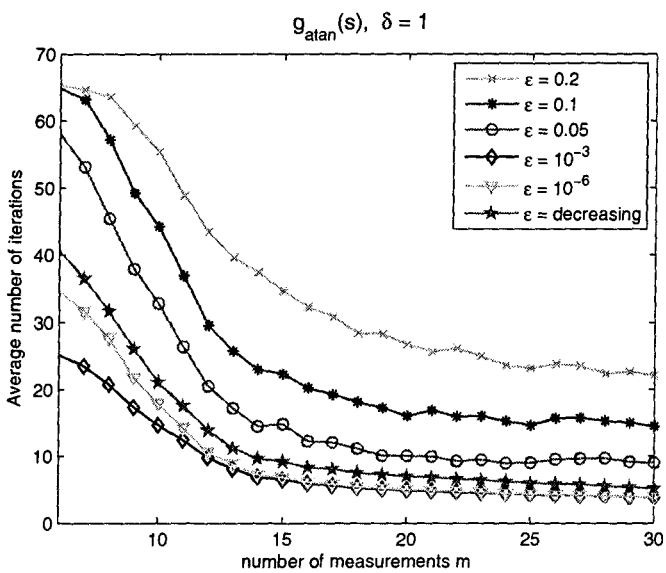



(b)

Figure 3.8: Effect of $\epsilon$ on the performance of MCCR for $g_c(s) = g_{log}(s)$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) The average number of iterations required to get a sparse solution vector.

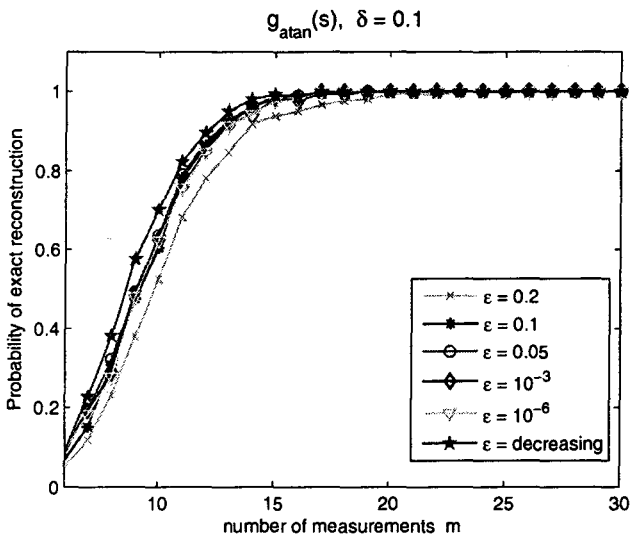



(a)

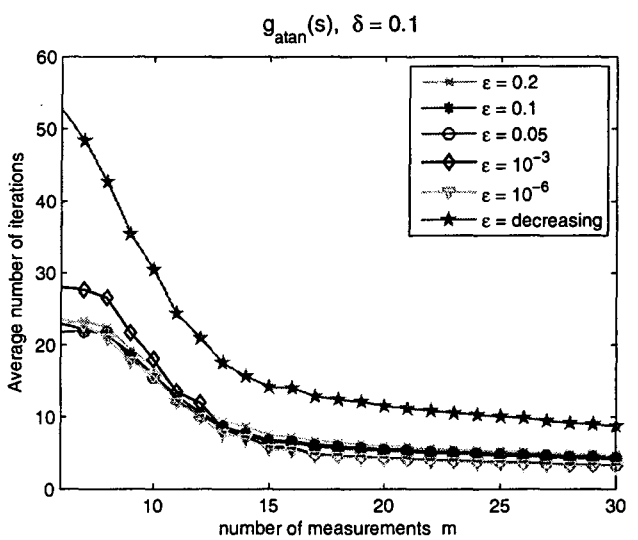



(b)

Figure 3.9: Effect of $\epsilon$ on the performance of MCCR for $g_c(s) = g_{atan}(s)$ and $\delta = 1$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) The average number of iterations required to get a sparse solution vector.




(a)




(b)

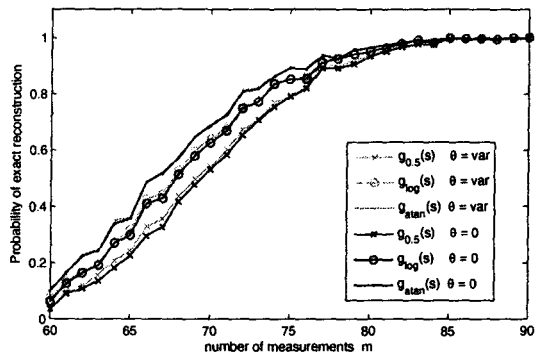
Figure 3.10: Effect of $\epsilon$ on the performance of MCCR for $g_c(s) = g_{atan}(s)$ and $\delta = 0.1$. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) The average number of iterations required to get a sparse solution vector.

$g_{atan}(s)$ is less sensitive to the variation in the value of $\epsilon$ as shown in Figure 3.10(a). Accordingly, the optimum choice of $\epsilon$ for $g_{atan}$ can be chosen as $\epsilon \leq 10^{-3}$. On the other hand, for $g_{log}(s)$ shown in Figure 3.8 none of the fixed values of $\epsilon$ provides a good tradeoff between the PER and the number of iterations. For example, $\epsilon = 0.1$ produced the highest PER but at the cost of a large number of iterations, and the opposite is true for $\epsilon = 10^{-6}$.

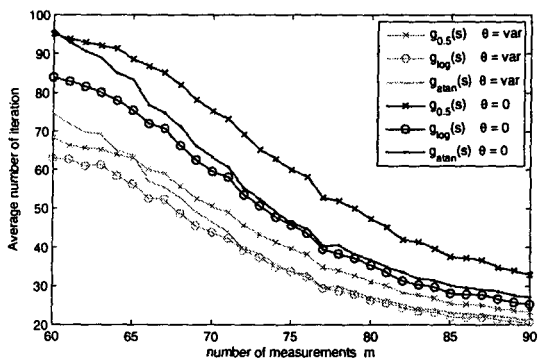
From this example it is clear that small $\epsilon$ produces good results for $g_{atan}(s)$ but not for $g_{log}(s)$, while a decreasing $\epsilon$ produces good results for both of them. Accordingly, since varying $\epsilon$ in the way described in Table 3.2 has a negligible computational cost, we recommend choosing a variable $\epsilon$ with all of the objective functions in Table 3.1.

**Example 3: Effect of $\theta$.**

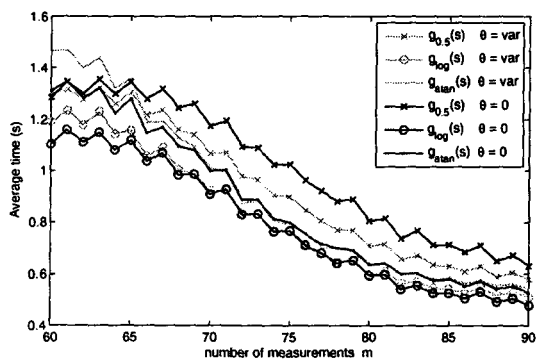
In this Example we compare the effect of $\theta$ on the performance of MCCR. Recall that the difference between the MCCR algorithm (3.25) and the IRLS algorithm (3.16) is the parameter $\theta$. The two algorithms are equivalent if $\theta = 0$, and the goal of this example is to investigate the impact of $\theta$ on the performance of MCCR. Three different objective functions are incorporated in this comparison. The functions are $g_{0.5}(s) = |s|^{0.5}$, $g_{log}(s)$, and $g_{atan}(s)$. Since the computation of $\theta$ requires solving a nonlinear optimization problem (3.28), it is important to compare not just the average number of iterations but also the average amount of time taken by each algorithm to converge to a solution vector. For doing this, the results are calculated for two different conditions. In the first condition, the low dimensional case, the parameters are $i = 30$, $n = 256$, and $m$ increases from 60 to 90, while for the second condition, the high dimensional case, the parameters are $i = 60$, $n = 512$, and $m$ increases from 120 to 160. The total number of runs for each case is 500, and the results are shown in Figure 3.11 and Figure 3.12. The results for MCCR are presented in faint brown while the results for IRLS ($\theta = 0$)are presented in blue.




(a)

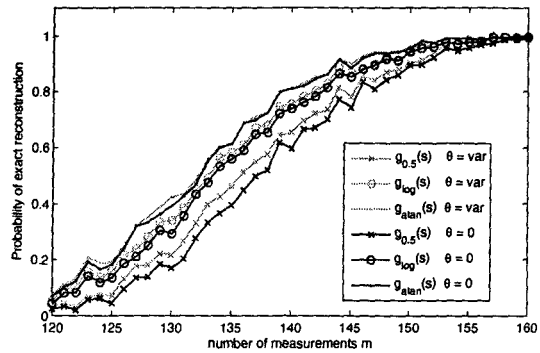



(b)

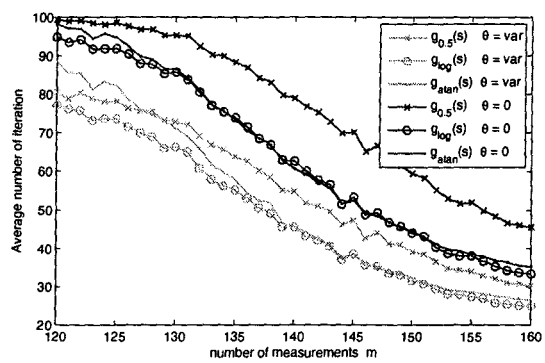



(c)

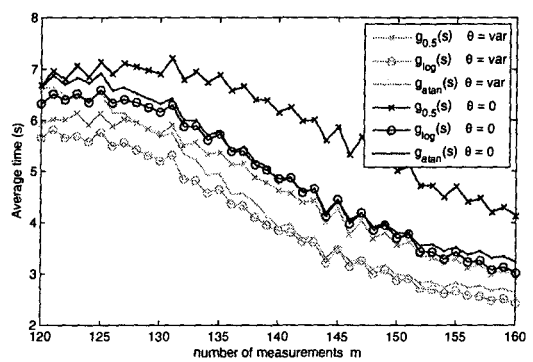
Figure 3.11: Effect of $\theta$ on the performance of MCCR for three different objective functions for the condition $i = 30$, $n = 256$, and $m$ increases from 60 to 90. (a) the probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) the average number of iterations required to get a sparse solution vector, and (c) the average time in (sec.).




(a)




(b)




(c)

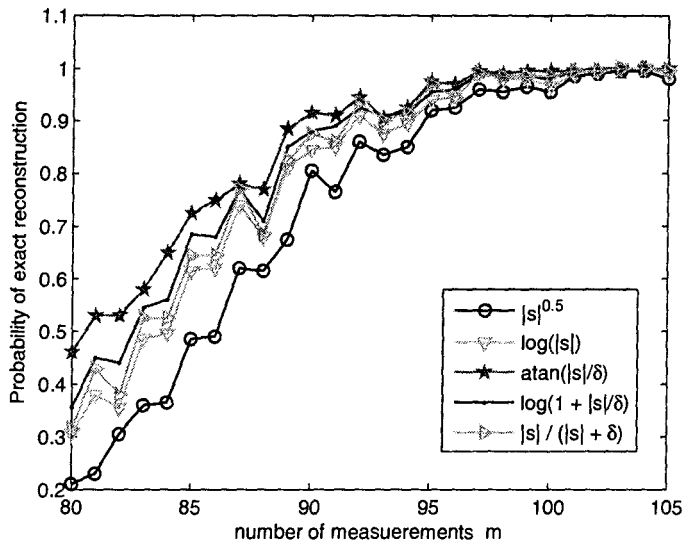
Figure 3.12: Effect of $\theta$ on the performance of MCCR for three different objective functions for the condition $i = 60$, $n = 512$, and $m$ increases from 120 to 160. (a) the probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$), (b) the average number of iterations required to get a sparse solution vector, and (c) the average time in (sec.).

As shown in Figure 3.11(a) and Figure 3.12(a) for the two conditions and for all the objective functions, MCCR and IRLS have almost similar PERs, meaning that they almost converge to the same solution vectors. However, comparing the average number of iterations taken by MCCR and IRLS algorithms in Figure 3.11(b) and Figure 3.12(b) it is clear that MCCR takes fewer iterations than IRLS to converge to the solution vectors. For example, in Figure 3.12(b), and for the objective function $g_{0.5}(s)$ and the case $m = 140$ we find that, on average, IRLS converges after 80 trials while MCCR converges after 54 trials only. However, since the computation of $\theta$ requires extra computational time, it is important to check the average time taken by each algorithm to converge to a solution vector. This comparison is shown in Figure 3.11(c) and Figure 3.12(c).

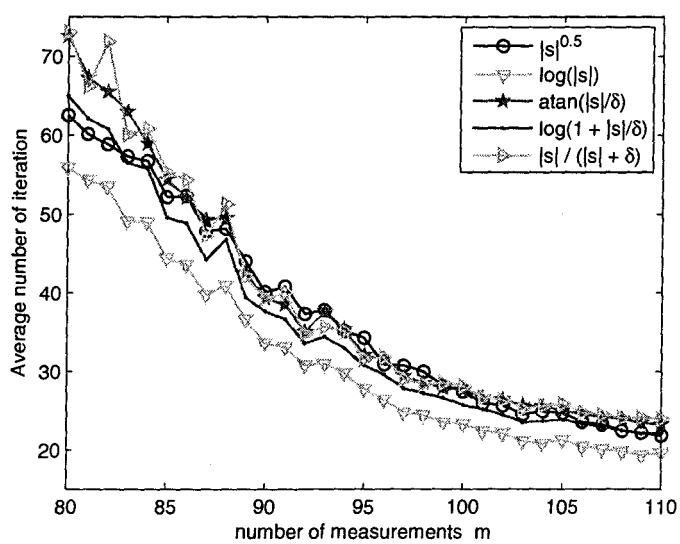
For the low dimensional case presented in Figure 3.11(c) it is clear that, except for $g_{0.5}(s)$, the two algorithms take on average the same amount of time to converge to a solution vector, which implies that there is no advantage of replacing the IRLS algorithm by the MCCR algorithm for low dimensional problems. However, the average computational time could be significantly reduced if a faster algorithm is used for calculating $\theta$, or if the computational cost of $\theta$ is low compared with the inversion operation in (3.25). This later case is presented in Figure 3.12(c). From this figure it is clear that MCCR in this case converges in a shorter time than IRLS. Accordingly, we conclude this example by recommending the MCCR algorithm to solve large scale problems.

**Example 4: Comparison between different objective functions.**

In this Example we examine the performance of MCCR as a function of the five objective functions in Table 3.1. As before, the performance is measured using the probability of exact reconstruction of the sparse vector, and the average number of iterations required for convergence to that solution vector, as functions of the number




(a)




(b)

Figure 3.13: Comparison between the five objective functions in Table 3.1. (a) The probability of exact reconstruction of the original sparse vector; (b) The average number of iterations required to get a sparse solution vector.

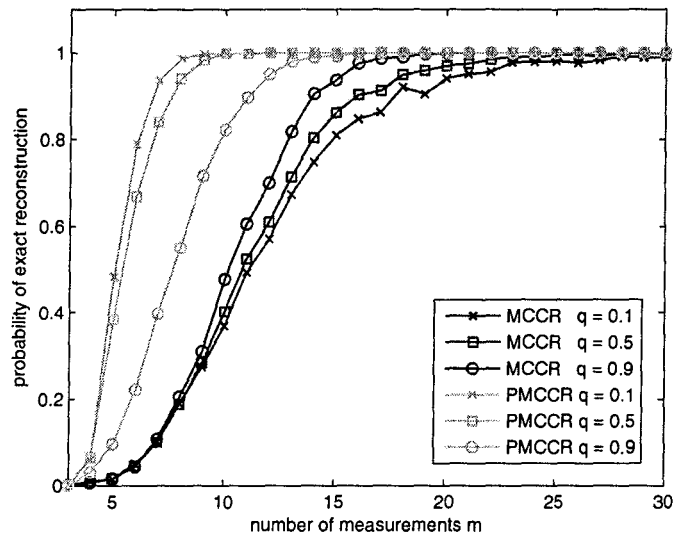
of observations $m$. In this example, $n = 256$, $i = 40$, and $m$ increases from 80 to 105. For each value of $m$, $\boldsymbol{A}$ and $\boldsymbol{s}_s$ are generated as in Example 1. The total number of runs in this example is 200, and the results are shown in Figure 3.13.

In terms of the probability of exact reconstruction (PER), it is clear from Figure 3.13(a) that the best performance of MCCR is obtained when $g_c(s) = \text{atan}(|s|/\delta)$ is used as an objective function, while the worst performance is obtained when $g_c(s) = |s|^{0.5}$. Also it can be shown that $\log(1+|s|/\delta)$ performs better than $\log(|s|)$. However, in terms of the average number of iterations shown in Figure 3.13(b), it is clear that the fastest convergence of MCCR is obtained when $g_c(s) = \log(|s|)$.

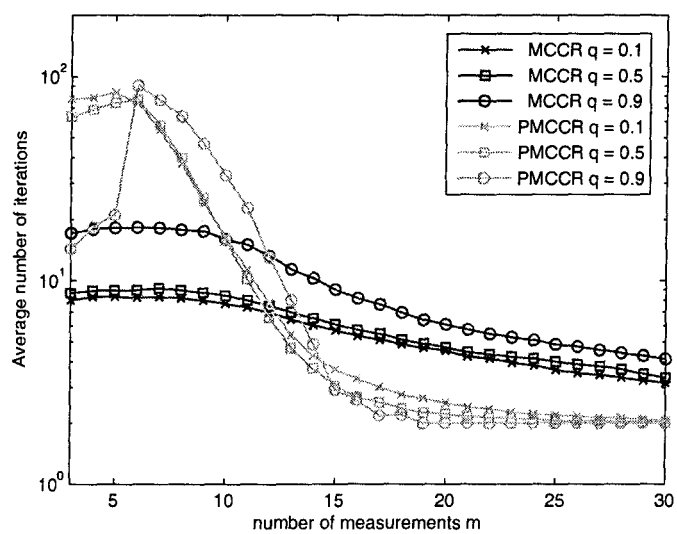
### Example 5: Comparison between MCCR and PMCCR.

In this example we examine the effect of the perturbation approach on the performance of the PMCCR. Specifically, we compare the performance of the MCCR and the PMCCR algorithm for the case $g(\boldsymbol{s}) = ||\boldsymbol{s}||_{\ell_q}^q$ for the following values of $q$; 0.1, 0.5, and 0.9. The comparison is made in terms of the probability of exact reconstruction of the sparse vector, and the average number of iterations required for convergence to that solution vector, as functions of the number of observations $m$, when both $n$ and $i$ are fixed at 40 and 3, respectively. For the two algorithms, the values of $\epsilon$ and $\theta$ are selected as in Table 3.2. The results are shown in Figure 3.14.

As shown in Figure 3.14(a) the probability of the exact reconstruction of the PMCCR algorithm is significantly better than that of the MCCR algorithm. For a low value of $m$, e.g., $m = 10$, the probability of exact reconstruction for PMCCR is 100% for $q = 0.1$ and 0.5. The number of iterations taken by each algorithm is presented in Figure 3.14(b). The number of iterations of the PMCCR algorithm presented in Figure 3.14(b) is the number of times the MCCR algorithm was called by the PMCCR algorithm, i.e., the final value of $j$ in Table 3.3. It is obvious that the PMCCR algorithm takes large number of iterations compared with the MCCR




(a)




(b)

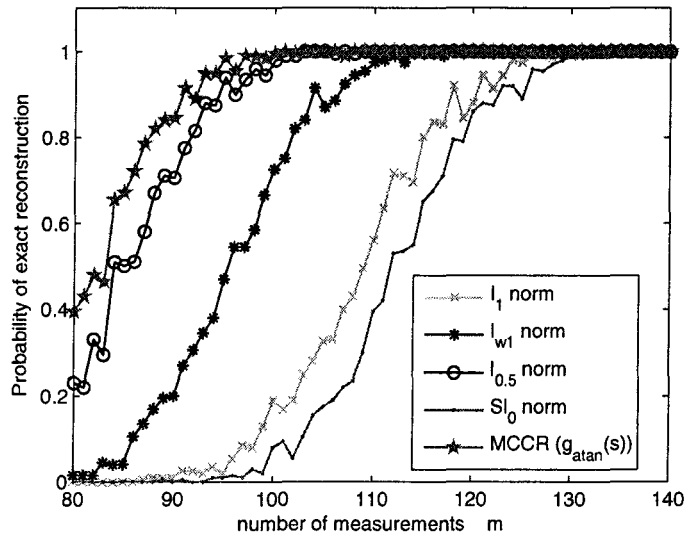
Figure 3.14: Comparison between MCCR and PMCCR. (a) The probability of exact reconstruction of the original sparse vector; (b) The average number of iterations required to get a sparse solution vector. The number of iterations of the PMCCR algorithm represents the value of $j$ in Table 3.3.

algorithm, especially when the number of measurements is small, e.g., $m < 10$ in this case. However, for larger $m$, e.g., $m \geq 15$, the performance of PMCCR in terms of successful reconstruction is 100%, while the number of outer iterations required is roughly less than or equal to 3. Therefore, PMCCR can be seen as a high-performance, more costly method for difficult cases (i.e., low values of $m$), whereas it is a higher-performance method at roughly the same cost, for cases which work well with other methods.

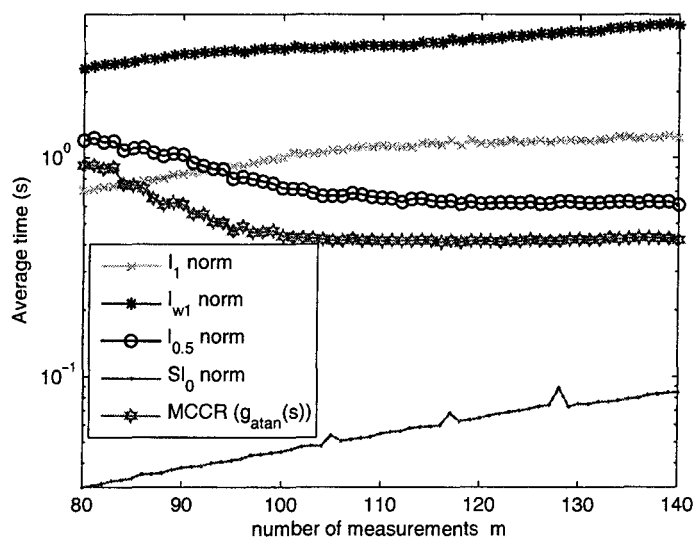
**Example 6: Comparison between MCCR and other algorithms.**

We conclude this chapter by a comparison between MCCR and four different algorithms. The four algorithms are: the $\ell_1$–norm; the weighted $\ell_1$–norm [18], denoted as $\ell_{w1}$–norm; the $\ell_{0.5}$–norm implemented using the algorithm presented in [13], which is equivalent to MCCR when $g_c(s) = |s|^{0.5}$ and $\theta = 0$; and the recently developed algorithm called smooth $\ell^0$–norm algorithm [88], denoted as $Sl_0$ in Figure 3.15. The objective function used with MCCR is $g_{atan}(s)$. In this example, $n = 256$, $i = 40$, and $m$ increases from 80 to 140. For each value of $m$, $\boldsymbol{A}$ and $\boldsymbol{s}_s$ are generated as in Example 1. The total number of runs in this example is 200, and the results are presented in Figure 3.15.

As shown in Figure 3.15(a) MCCR has the best performance in terms of the PER, while the smooth $\ell^0$–norm algorithm has the worst performance among the compared algorithms. Also it is clear from Figure 3.15(a) that the PER of $\ell_{0.5}$–norm algorithm is very close to that of MCCR and the $\ell_{w1}$–norm performs better than the regular $\ell_1$–norm. However, in terms of the average amount of time shown in Figure 3.15(b) it is clear that, aside from the smooth $\ell^0$–norm, the MCCR is the fastest algorithm, while the $\ell_{w1}$–norm is the most computationally expensive one. Although both MCCR and $\ell_{0.5}$–norm algorithms estimated all the solution vectors exactly for all $m > 100$, the MCCR converged in a significantly shorter time. For example, for the case when




(a)




(b)

Figure 3.15: Comparison between MCCR and four different algorithms. (a) The probability of exact reconstruction of the original sparse vector as a function of the number of measurements ($m$); (b) The average number of iterations required to get a sparse solution vector.

$m = 100$, MCCR converged in 0.43 seconds while the $\ell_{0.5}$–norm algorithm converged in 0.72 seconds.

## 3.6 Conclusion

In this chapter a novel methodology was developed and employed to minimize a class of non–convex (concave on the non–negative orthant) functions for solving an under–determined system of linear equations for the case of sparse solution vector. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. It was shown in this chapter, for a certain selection of the convex objective function, the class of algorithms called Iterative Re–weighted Least Squares (IRLS) can be derived from the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods. In this chapter we also proposed a convex objective function that produces an algorithm that can converge to the sparse solution vector in a significantly fewer number of iterations than the IRLS algorithms. Other selections of the convex objective function may produce algorithms with convergence properties better than the IRLS algorithms.

In this chapter we also proposed a straightforward technique for selecting a convex function such that, for any starting solution vector $\boldsymbol{s}_0$, the algorithm generates a sequence $\{\boldsymbol{s}_k\}_{k=1}^{\infty}$ that converges to a fixed point of the original objective function. Since the original objective functions are non-convex, the proposed algorithm is susceptible to convergence to a local minimum. To alleviate this difficulty, we proposed a random perturbation technique that enhances the performance of the proposed algorithm. Extensive simulation results were presented to examine the effect of various parameters on the performance of the proposed algorithm and to compare its performance with some well known algorithms that are usually utilized for solving the same

problem. The simulation results show that the proposed algorithm outperforms the existing algorithms in terms of the execution time and the accuracy of reconstructing a sparse solution vector.

# Chapter 4

# Blind Source Separation of an Unknown Number of Sources

## 4.1 Introduction

In Chapter 3 we discussed the problem of finding a unique solution to a linear under-determined system of equations $\boldsymbol{As} = \boldsymbol{x}$, when both $\boldsymbol{x}$ and $\boldsymbol{A}$ are known, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $m < n$. In this chapter we discuss a more challenging problem of finding a unique solution to the linear system $\boldsymbol{X} = \boldsymbol{AS}$ when both $\boldsymbol{A}$ and $\boldsymbol{S}$ are *unknowns*, where $\boldsymbol{X} \in \mathbb{R}^{m \times T}$ is a matrix of measured signals, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is an unknown mixing matrix, $\boldsymbol{S} \in \mathbb{R}^{n \times T}$ is a matrix of unknown sources, $m$ is the number of observations, $n$ is the number of sources, and $T$ is the number of samples.

This problem has been studied for nearly two decades under the name blind source separation (BSS) [25–33, 50–54, 89–104]. In BSS it is known a priori that the $m$ measured signals are linear combinations of $n$ hidden sources, and the task is to recover the sources as accurately as possible when the mixing coefficients are unknown. Based on the relation between the number of measurements $m$ and the number of

sources $n$; the BSS problem is usually classified as *complete* (even–determined), *over–determined*, or *under–determined* for the cases $m = n$, $m > n$, or $m < n$, respectively. Since the over–determined case can always be transformed into a complete case by using the principal component analysis (PCA) technique, the discussion presented in this section is restricted to the complete and the under–determined cases only.

The early years of BSS research concentrated on solutions for even-determined and over–determined mixing processes. The earliest approach traces back to Herault and Jutten [90] whose goal was to separate an instantaneous linear even-determined mixtures of non-Gaussian independent sources. Since then, many algorithms [51–55] have been developed for separating *independent sources* from their linear mixtures, the technique known in the literature as independent component analysis (ICA). See [55] for a revision of the most known ICA algorithms.

It is known [51–55] that the ICA technique is restricted to the cases for which the underlying problem satisfies the following conditions: 1) the sources are mutually independent, 2) the number of sources is less than or equal the number of sensors, 3) at most one of the sources is Gaussian, and 4) the mixing matrix is full column rank. However, there are many applications for which one or some of these assumptions are violated. For example, in some applications the number of sources is generally unknown and could be greater than the number of sensors, while in other applications some of the sources could be correlated. In these applications, ICA does not produce satisfactory results, and an alternative technique must be utilized for estimating the hidden sources.

A recently developed technique, known as Sparse Component Analysis (SCA), has received a great deal of attention in recent years. SCA can solve the under–determined BSS problem, i.e. the case of more sources than sensors. The additional information required in compensating the limited number of sensors is the sparseness of the source matrix $\boldsymbol{S}$. Since non–sparse sources can often be sparsely represented

under a suitable linear transformation, (e.g., the short time Fourier transform, the wavelet transform, the wavelet packets transform,... etc.), the SCA problem is quite general and also applicable to non-sparse sources. Accordingly, SCA can substitute for ICA in cases when some of the assumptions associated with ICA are violated, e.g., when the number of sources is greater than the number of sensors, or when some of the sources are correlated. In the sequel, we indicate a transformed quantity by a hat over the respective symbol, e.g., $\hat{\boldsymbol{S}}$ is the sparse version of $\boldsymbol{S}$, obtained under some suitable linear transformation.

There are two main approaches for solving the BSS problem via SCA. In the first approach, the mixing matrix and the sparse sources are estimated simultaneously via maximum likelihood or maximum a posteriori approaches [28–30]. However, these approaches converge to local minima and have poor convergence properties [25,28,91]. In the second approach [25–27,31–33,92–94,96–104], the mixing matrix and the sparse sources are estimated separately. When the source matrix $\boldsymbol{S}$ is sufficiently sparse in the original space of representation, the BSS problem is solved in two steps, where the mixing matrix is estimated in the first step via clustering the columns of the measured matrix $\boldsymbol{X}$, while the sparse source matrix $\boldsymbol{S}$ is estimated in the second step by solving the system of linear equations $\boldsymbol{X} = \boldsymbol{AS}$ under the constraint that $\boldsymbol{S}$ is sparse. This latter problem can be solved using one of the compressed sensing algorithms described in Chapter 2 and Chapter 3.

It is obvious that the accuracy of estimating the sparse sources in the second step of the two-step approach depends on how well the mixing matrix is estimated in the first step. Accordingly, the most critical component in the two-step approach is the first step, during which the mixing matrix is usually estimated via clustering some or all of the columns of the measured matrix $\boldsymbol{X}$. However, precise estimation of the mixing matrix remains a problem in the two-step approach. Most of the clustering algorithms that have been utilized for estimating the mixing matrix have

some limitations. For example, most SCA algorithms are based on estimating the mixing matrix via partitioning clustering algorithms, e.g. $k$–means [25,33], fuzzy c–means [50,95], or modified $k$–means [94,100,103]. However, and generally speaking, there are three main problems associated with most partitioning algorithms [105]:

- The number of clusters, which equals the number of sources, has to be known in advance; a condition which might not be available in some applications, e.g., BSS of EEG signals.

- Since each point (column of $\boldsymbol{X}$) must be assigned into a cluster, most partitioning clustering algorithms fail in the presence of noise and/or outlier points.

- All partitioning clustering algorithms are locally convergent and sensitive to the initial choice of the clusters' centroids.

A revision of some clustering algorithms, that were suggested in the literature for estimating the mixing matrix, is presented in the next section.

In this chapter we propose three novel clustering algorithms. The impact of the three limitations of the previous approaches are alleviated with the proposed algorithms. Moreover, the proposed algorithms can estimate the number of clusters directly from the data matrix. Accordingly, the proposed algorithms can handle the case where the number of clusters (sources) is unknown.

The remaining part of this chapter is organized as follows. In Section 4.2 we present in detail the various steps associated with solving the BSS problem via the two-step SCA approach. Three novel clustering algorithms, which can be used for estimating the mixing matrix and the number of hidden sources, are proposed in Section 4.3. Section 4.4 presents some computer simulations for assessing the performance of the proposed algorithms. Finally, conclusions are given in Section 4.5.

| List of symbols | |
|---|---|
| $\boldsymbol{\Phi}$ | The dictionary matrix. |
| $\boldsymbol{A}$ | The mixing matrix. |
| $\tilde{\boldsymbol{A}}$ | An estimate of the mixing matrix. |
| $\boldsymbol{D}$ | The dissimilarity matrix. |
| $\boldsymbol{S}$ | The source matrix. |
| $\hat{\boldsymbol{S}}$ | The source coefficient matrix in the transform domain. |
| $\boldsymbol{V}$ | The noise matrix. |
| $\hat{\boldsymbol{V}}$ | The noise coefficient matrix in the transform domain. |
| $\boldsymbol{X}$ | The measurement matrix. |
| $\hat{\boldsymbol{X}}$ | The measurement coefficient matrix in the transform domain. |
| $\boldsymbol{Y}$ | The feature matrix. |
| $\hat{\boldsymbol{Y}}$ | A sub-matrix of the feature matrix $\boldsymbol{Y}$. |
| $\boldsymbol{Z}$ | The normalized feature matrix. |
| $\boldsymbol{r}$ | A reference vector. |
| $m$ | The number of sensors (rows of $\boldsymbol{X}$). |
| $n$ | The number of sources (rows of $\boldsymbol{S}$). |
| $T$ | The number of samples (columns of $\boldsymbol{S}$). |
| $\hat{T}$ | The number of columns of $\boldsymbol{Z}$. |
| $d(\cdot,\cdot)$ | Distance between two vectors. |
| $\mu$ | The largest $\ell_2$-norm of the columns of $\hat{\boldsymbol{Y}}$. |
| $\mathcal{I}_l$ | The indices of the $i$th cluster. |
| $\boldsymbol{C}^l$ | A sub-matrix of $\hat{\boldsymbol{Y}}$ corresponding to the $l$th cluster. |
| $\beta_l$ | The first concentration parameter of the $l$th cluster. |
| $\gamma_l$ | The second concentration parameter of the $l$th cluster. |

## 4.2 Problem Formulation

In this section the problem of solving the under–determined blind source separation using SCA is presented. The BSS problem is defined as the problem of retrieving $n$ unknown source signals $\boldsymbol{s}(\xi) \in \mathbb{R}^n$ from $m$ linear measurements $\boldsymbol{x}(\xi) \in \mathbb{R}^m$ when the mixing matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is unknown, and the measurements are possibly corrupted by additive noise $\boldsymbol{v}(\xi) \in \mathbb{R}^m$:

$$\boldsymbol{x}(\xi) = \boldsymbol{A}\boldsymbol{s}(\xi) + \boldsymbol{v}(\xi), \quad \xi = 1, \ldots, T, \tag{4.1}$$

where $T$ is the number of samples. The parameter $\xi$ represents either time, spatial coordinates in the case of images, spatio–temporal parameter in the case of video sequences, or wavelength in the case of multispectral or other optical signals. Eq. (4.1) can be written in the following compact form

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S} + \boldsymbol{V}, \tag{4.2}$$

where $\boldsymbol{X} \in \mathbb{R}^{m \times T}$ is a matrix of observed signals, $\boldsymbol{S} \in \mathbb{R}^{n \times T}$ is a matrix of unknown sources, $\boldsymbol{V} \in \mathbb{R}^{m \times T}$ is a matrix of additive noise, $m$ is the number of observations, and $n$ is the number of sources.

The task of blind source separation is to accurately estimate the sources from the observed signals $\boldsymbol{X}$ when the mixing matrix is unknown. In this chapter we follow the two-step approach in solving the BSS problem via SCA. The two–step approach can be generally summarized in the following steps [26, 27, 91, 95, 97, 99, 103]:

1. **Sparse representation:** If the source matrix is not sparse, select a suitable linear transformation, (e.g., the short time Fourier transform or the wavelet transform), and calculate $\hat{\boldsymbol{X}}$, the coefficients of the measured matrix $\boldsymbol{X}$ in the transform domain. Otherwise, go to step 2.

2. **Form the feature matrix $\boldsymbol{Y}$** by selecting columns of the measured matrix $\boldsymbol{X}$ or (*a subset of*) their transformed coefficients $\hat{\boldsymbol{X}}$.

3. **Preparation steps:**

   - Normalize the feature vectors: $\boldsymbol{y}_k = \boldsymbol{y}_k / ||\boldsymbol{y}_k||_{\ell_2}$, in order to project data points onto the surface of a unit sphere, where $||\cdot||_{\ell_2}$ denotes the $\ell_2$ norm.

     Before normalization, it is reasonable to remove data points with a very small norm, since these points likely correspond to noise.

   - Move the data points to a half-hypersphere by multiplying each column of the feature vectors $\boldsymbol{y}_k$ by the sign of its first entry.

4. **Mixing matrix Estimation:** Estimate the columns of the mixing matrix by clustering the columns of the feature matrix $\boldsymbol{Y}$. The coordinates of the center of each cluster will form one column of the estimated mixing matrix $\tilde{\boldsymbol{A}}$.

5. **Sparse coefficient estimation:** Use the estimated mixing matrix $\tilde{\boldsymbol{A}}$ and the measured coefficient matrix $\hat{\boldsymbol{X}}$ and solve the following under–determined system of linear equations $\hat{\boldsymbol{X}} = \tilde{\boldsymbol{A}}\hat{\boldsymbol{S}}$ to find an estimate of $\hat{\boldsymbol{S}}$, the source coefficient matrix in the transform domain.

6. Stop if Step 1 is not utilized. Otherwise, perform the next step.

7. **Source estimation:** Apply the inverse transformation on $\hat{\boldsymbol{S}}$ to obtain an estimate of the source matrix $\boldsymbol{S}$

In the remaining part of this section we provide a detailed description of these steps and their effect on the overall solution of the BSS problem.

### 4.2.1 Sparse representation

Most natural source signals are non–sparse in their original space of representation. Although one can find examples of natural signals or images that are sparse in their

original space of representation, in most cases the source signals have a rather non-sparse nature. However, the sparsity of the source components plays a key role in solving the under–determined BSS problem using the SCA technique. The reason will be clear after the next subsection. Accordingly, the first step in solving the under–determined BSS problem using SCA is to find a suitable linear transformation such that the unknown source signals have sparse coefficients in the transform domain. To be specific, let $\{\boldsymbol{\phi}_j(\xi) \in \mathbb{R}^T\}_{j=1}^J$ denote a set of $J$ basis vectors, and $\hat{s}_{ij}$ be the decomposition coefficients of the $i$th source signal in $\boldsymbol{\phi}_j(\xi)$, i.e.,

$$\boldsymbol{s}^i(\xi) = \sum_{j=1}^{J} \hat{s}_{ij} \boldsymbol{\phi}_j^T(\xi) = \hat{\boldsymbol{s}}^i \boldsymbol{\Phi}^T(\xi), \quad i = 1, \ldots, n, \tag{4.3}$$

where $\boldsymbol{s}^i(\xi)$ is the $i$th row of $\boldsymbol{S}$, and $\boldsymbol{\Phi}(\xi) \in \mathbb{R}^{T \times J}$ is called the *basis* or *dictionary* matrix. Depending on the number of columns $J$, the dictionary is either complete ($J = T$) or overcomplete ($J > T$). Examples of such dictionaries that are usually used for sparse representation are Fourier basis, Gabor basis, various wavelet-related bases, etc. [28, 29].

The sparsity of the source coefficients $\{\hat{\boldsymbol{s}}^i\}_{i=1}^n$ depends on the utilized dictionary matrix. Unfortunately, there is no universal transformation that projects any given natural signal or image onto its optimal sparse representation. Different classes of signals require their specific, optimal (in some sense), sparsification transformations [91]. For example, speech signals can be sparsely represented in terms of a time-frequency Gabor dictionary that consists of a variety of sine waves modulated by Gaussian windows, with different locations and scales [18], while images can be sparsely represented using derivative operators [95].

The corresponding representation of the mixtures in terms of the same basis functions is given by

$$\boldsymbol{x}^k(\xi) = \sum_{j=1}^{J} \hat{x}_{kj} \boldsymbol{\phi}_j^T(\xi) = \hat{\boldsymbol{x}}^k \boldsymbol{\Phi}^T(\xi), \quad k = 1, \ldots, m, \tag{4.4}$$

where $\hat{x}_{kj}$ are the decomposition coefficients of the $k$th measured signal, i.e., the $k$th row of $\boldsymbol{X}$. Substituting (4.3) and (4.4) into (4.2) and assuming that the dictionary matrix is complete, i.e. $J = T$, we get

$$\hat{\boldsymbol{X}} = \boldsymbol{A}\hat{\boldsymbol{S}} + \hat{\boldsymbol{V}}, \tag{4.5}$$

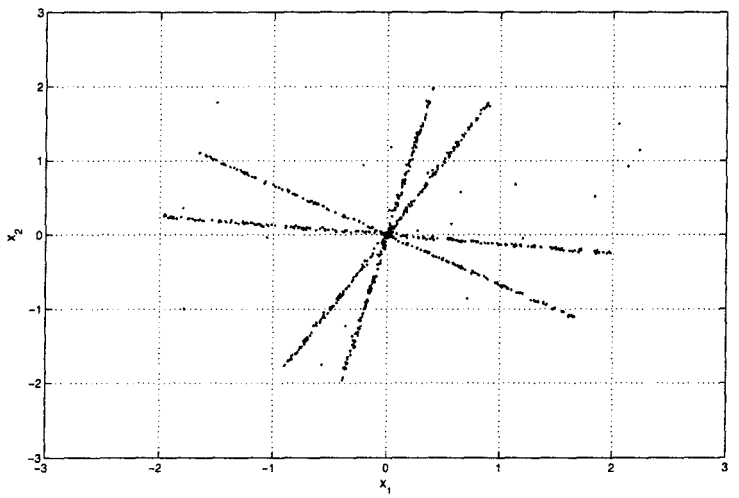
where $\boldsymbol{V} = \hat{\boldsymbol{V}}(\xi)\boldsymbol{\Phi}^T(\xi)$, and $\hat{\boldsymbol{V}}$ is the decomposition coefficients of the noise matrix $\boldsymbol{V}(\xi)$ in $\boldsymbol{\Phi}(\xi)$. Comparing (4.5) and (4.2) we find that the relation between the sources and the mixtures in the original space of representation is preserved in the transform domain. Accordingly, the estimation of the mixing matrix can be performed in the transform domain via clustering some or all of the columns of $\hat{\boldsymbol{X}}$.

### 4.2.2 Preparation steps

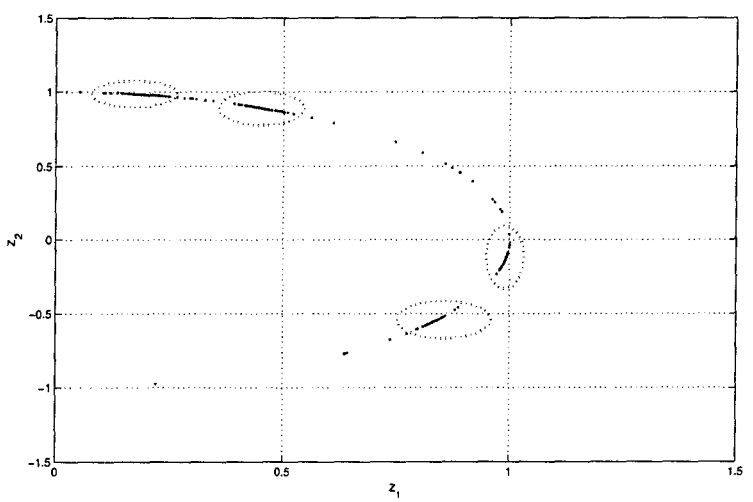
If the dictionary matrix is properly selected such that the coefficients of the sources are very sparse, then, with high probability, there will be many columns of the source coefficient matrix $\hat{\boldsymbol{S}}$ that have only one significantly nonzero entry. For each one of these columns, the corresponding column of $\hat{\boldsymbol{X}}$ is collinear with one column of the mixing matrix $\boldsymbol{A}$. Mathematically speaking, assume that the $i$th entry of the $k$th column of $\hat{\boldsymbol{S}}$ is nonzero, while the remaining $(n-1)$ entries are zeros. Then, from (4.5) and neglecting the effect of noise, the $k$th column of $\hat{\boldsymbol{X}}$ is given by

$$\hat{\boldsymbol{x}}_k = \boldsymbol{A}\hat{\boldsymbol{s}}_k = \boldsymbol{a}_1\hat{s}_k[1] + \ldots + \boldsymbol{a}_n\hat{s}_k[n] = \boldsymbol{a}_i\hat{s}_k[i]. \tag{4.6}$$

Therefore, in the $m$ dimensional space, the scatter plot of the column $\hat{\boldsymbol{x}}_k$ would lie on a hyperline passing through the origin with orientation corresponding to the $i$th column of the mixing matrix, $\boldsymbol{a}_i$. When the source coefficients are sufficiently sparse to satisfy the *disjoint orthogonality* condition [103], i.e., $\hat{s}_l[i] \cdot \hat{s}_l[j] = 0, \forall i \neq j$ (or $\hat{s}_l[i] \cdot \hat{s}_l[j] \approx 0$ in the noisy case), for a large number of columns of $\hat{\boldsymbol{S}}$, the scatter plot of the columns of the data coefficients matrix $\hat{\boldsymbol{X}}$ in the $m$ dimensional space would be




(a)




(b)

Figure 4.1: (a) Scatter plot of a mixture of four sparse sources, (b) Scatter plot of the normalized mixtures. Four clusters corresponding to the four mixing columns are clearly visible.

concentrated around $n$ hyperlines whose orientations correspond to the columns of the mixing matrix $\boldsymbol{A}$. To clarify this point, consider the following demonstrating example, in which a sparse source matrix $\boldsymbol{S} \in \mathbb{R}^{4\times 2000}$ is randomly generated. Each row of the sparse source matrix $\boldsymbol{S}$ is constructed such that it has exactly 500 nonzero entries. The indices of these 500 entries are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm 2$. The measured matrix $\boldsymbol{X} \in \mathbb{R}^{2\times 2000}$ is constructed by multiplying the source matrix $\boldsymbol{S}$ by the following $(2 \times 4)$ mixing matrix.

$$\boldsymbol{A} = \begin{bmatrix} 0.1 & -1.4 & 0.7 & 0.3 \\ 0.5 & 0.17 & 1.4 & -0.2 \end{bmatrix}$$

The scatter plot of the two mixtures is shown in Figure 4.1(a), where each point in this figure represents one column of the data matrix $\boldsymbol{X}$. As shown in this figure, and due to the sparsity of the source matrix, the columns of the data matrix are concentrated around four lines whose orientations correspond to the columns of the mixing matrix $\boldsymbol{A}$. Therefore, the essence of solving the BSS problem using SCA is the estimation of the $n$ columns of the mixing matrix via the identification of the orientations of the most concentrated $n$ hyperlines from the observed data using clustering algorithms.

It was shown in [50, 91] that, in some cases, especially when the wavelet packet transform is utilized for the sparse representation step, a good estimate of the mixing matrix is possible if only a fraction of the columns of the coefficient data matrix is used in the clustering process. The motivation behind this approach is that, in many cases, the sources have different sparsity properties at different nodes of a wavelet packet tree. Accordingly, only a subset of the nodes, the ones for which the sources are very sparse, can be used for estimating the mixing matrix. For this reason, a new matrix, called the *feature matrix* $\boldsymbol{Y}$, is constructed from selected columns of the data coefficient matrix $\hat{\boldsymbol{X}}$. The feature matrix $\boldsymbol{Y}$ will be used only for estimating the mixing matrix, while the data coefficient matrix $\hat{\boldsymbol{X}}$ is utilized for estimating the

source coefficient matrix $\hat{\boldsymbol{S}}$ [25, 33, 91, 93, 95].

From (4.6) it is clear that the identification of $\boldsymbol{a}_i$ from $\hat{\boldsymbol{x}}_k$, or equivalently $\boldsymbol{y}_k$, is associated with sign and scale indeterminacies produced by $\hat{s}_k[i]$. Accordingly, these two ambiguities must be removed before estimating the columns of the mixing matrix via clustering the columns of the feature matrix $\boldsymbol{Y}$.

**1 - Removing the scale ambiguity:**

Since multiplying the $i$th column of $\boldsymbol{A}$ by any constant $\alpha \neq 0$ and dividing the $i$th row of $\hat{\boldsymbol{S}}$ by the same constant does not affect the entries of the data coefficient matrix $\hat{\boldsymbol{X}}$, or equivalently $\boldsymbol{Y}$, it will be assumed without loss of generality that the $\ell_2$–norm of each column of $\boldsymbol{A}$ equals one. Accordingly, the scale ambiguity can be removed by projecting the columns of the feature matrix $\boldsymbol{Y}$ into the surface of the unit hyper–sphere. However, before applying this step it is reasonable to remove the columns of $\boldsymbol{Y}$ with a norm value less than a pre–specified small threshold $\theta$, since these columns very likely correspond to noise. After selecting the threshold $\theta$, a new data matrix is generated as

$$\hat{\boldsymbol{Y}} = \{\boldsymbol{y}_k : ||\boldsymbol{y}_k||_{\ell_2} > \theta\}, \tag{4.7}$$

and each column is projected into the surface of the unit sphere to construct a new data matrix $\boldsymbol{Z}$ whose $k$th column is given by

$$\boldsymbol{z}_k := \hat{\boldsymbol{y}}_k / ||\hat{\boldsymbol{y}}_k||_{\ell_2}. \tag{4.8}$$

**2 - Removing the sign ambiguity:**

After removing the scale ambiguity, it is reasonable to remove the sign ambiguity from each column of $\boldsymbol{Z}$, otherwise, each line in Figure 4.1(a) will produce two clusters on opposite sides of the unit hemisphere. This ambiguity can be removed by multiplying

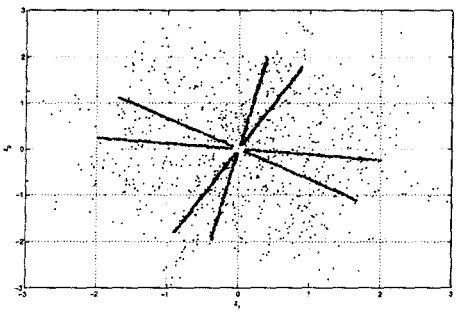
each column of the new data matrix $\boldsymbol{Z}$ by the sign of its first entry, i.e.,

$$\boldsymbol{z}_k \leftarrow \text{sign}(z_k[1]) \cdot \boldsymbol{z}_k. \tag{4.9}$$

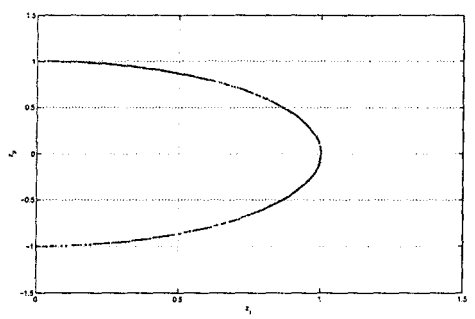
The result of applying these two steps on the data set shown in Figure 4.1(a) is shown in Figure 4.1(b). As shown in this figure, four different clusters emerge on the surface of the unit circle. Accordingly, the four clusters can be easily identified by applying a partitioning clustering algorithm, e.g. $k$–means, on this preproceed data set. The mean of the data points in each cluster can then be considered as an estimate of one column of the mixing matrix $\boldsymbol{A}$.

The example presented in Figure 4.1 shows that, when the source coefficient matrix $\hat{\boldsymbol{S}}$ is very sparse, the two preprocessing steps can project the columns of the feature matrix into separate clusters on the surface of the unit hemisphere. However, in practice, $\hat{\boldsymbol{S}}$ is not very sparse and many of its columns have more than one large entry. In this case, the scatter plot of the columns of the feature matrix $\boldsymbol{Y}$ will show a cloud surrounding the hyperlines (see Figure 4.2(a) for example). Therefore, after projecting the columns of the feature matrix into the surface of the unit hemisphere, the points in the scatter plot corresponding to the cloud will fill the gaps between the clusters, and no clear boundaries between the clusters will be visible. This situation is shown in Figure 4.2(b).

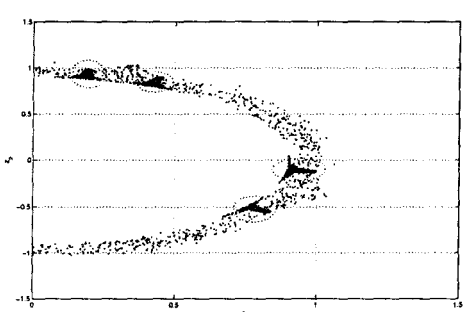
As will be shown later, the first proposed clustering algorithm in Section 4.3 is based on combining a statistical test with a *hierarchical clustering* technique to sequentially extract compact clusters. The statistical test is utilized to test whether the two closest clusters are significantly different or not. For this technique to succeed, the clusters must be clearly disjoint. However, as shown in Figure 4.2(b), the regular normalization technique may cause the statistical test to fail to distinguish the clusters resulting from the hyperlines in Figure 4.2(a). This is because the points in the cloud in Figure 4.2(a) fill the gap between the desired clusters.



(a)



(b)



(c)

Figure 4.2: (a) Scatter plot of a mixture of four sparse sources after removing the columns that have small norm values, (b) Scatter plot of the normalized mixtures on the surface of the unit hemisphere. (c) The proposed normalization technique shows four clear clusters corresponding to the four mixing columns.

To overcome this difficulty, we propose a method that maps points in the hyperlines in Figure 4.2(a) into discrete clusters, while ensuring that the points in the cloud are dispersed. To this end, we propose projecting the columns of the feature matrix $\boldsymbol{Y}$ into the narrow strip confined between two hyperspheres of radii $(1-\epsilon)$ and $(1+\epsilon)$, respectively, where $0 \leq \epsilon \leq 1$. Selecting $\epsilon = 0$ is equivalent to projecting the columns of the feature matrix on the surface of the unit sphere, while selecting $\epsilon = 1$ is equivalent to scaling the columns of the feature matrix such that its largest column has an $\ell_2$–norm of 2.

Here we consider a mapping from space $\mathcal{A}$, in which the points cluster into hyperlines as in Figure 4.2(a), into a space $\mathcal{B}$ where the points are represented as compact clusters. The proposed mapping $\mathcal{F} : \mathcal{A} \rightarrow \mathcal{B}$ is linear, i.e., for any $\boldsymbol{q} \in \mathcal{A}$, the corresponding point in $\mathcal{B}$ is given by $\boldsymbol{p} = \lambda \boldsymbol{q}$. The mapping parameter $\lambda$ is chosen such that, for all $\boldsymbol{q} \in \mathcal{A}$ with $0 < ||\boldsymbol{q}||_{\ell_2} \leq \mu$, where $\mu$ is the upper bound of the available data, the corresponding point in $\mathcal{B}$ is bounded as $(1-\epsilon) < ||\boldsymbol{p}||_{\ell_2} \leq (1+\epsilon)$. It is readily shown that $\lambda$ is then given as

$$\lambda = \frac{1-\epsilon}{||\boldsymbol{q}||_{\ell_2}} + \frac{2\epsilon}{\mu}. \tag{4.10}$$

Accordingly, following the proposed mapping technique, the normalization step (4.8) is replaced by the following mapping step

$$\boldsymbol{z}_k = \left( \frac{1-\epsilon}{||\hat{\boldsymbol{y}}_k||_{\ell_2}} + \frac{2\epsilon}{\mu} \right) \hat{\boldsymbol{y}}_k. \tag{4.11}$$

The value of $\mu$ can be calculated as the largest $\ell_2$–norm of the columns of the matrix $\hat{\boldsymbol{Y}}$ calculated in (4.7), and $\epsilon$ is a user-defined parameter. The result of applying this mapping technique on the data set shown in Figure 4.2(a) is shown in Figure 4.2(c). As shown in this figure, the data points in the fuzzy cloud in Figure 4.2(a) constitute a cloud in the narrow strip, while the four hyperlines constitute four concentrated clusters. In Figure 4.2(c), and in the remaining examples in this chapter, we set $\epsilon = 0.1$.

### 4.2.3 Mixing matrix estimation

There are several approaches that have been proposed in the literature for estimating the mixing matrix $\boldsymbol{A}$ via clustering the columns of the feature matrix $\boldsymbol{Y}$. The first approach [25, 33, 91, 93, 94, 101, 103] is based on utilizing a partitioning clustering algorithm, e.g., $k$–means and the modified $k$–means, which is based on the expectation maximization (EM) procedure. Table 4.1 presents a summary of the main steps associated with the class of partitioning clustering algorithms. As shown in Table 4.1, the algorithm accepts the feature matrix $\boldsymbol{Y}$ and the number of clusters $n$ as inputs, and produces the clusters' representative vectors as estimates of the columns of the mixing matrix. The difference between different partitioning clustering algorithms resides in the way that $\boldsymbol{c}_i$ and $d(\cdot,\cdot)$ are defined. In the conventional $k$–means algorithm, $d(\cdot,\cdot)$ is defined as the Euclidian distance between its arguments, i.e., $d(\boldsymbol{x},\boldsymbol{y}) = ||\boldsymbol{x}-\boldsymbol{y}||_{\ell_2}$, while the representative vector of the $i$th cluster is selected as the mean vector of all the vectors assigned to the $i$th cluster, i.e., $\boldsymbol{c}_i = \text{mean}(\{\boldsymbol{z}_k : \boldsymbol{z}_k \in \Omega_i\})$, where $\Omega_i$ is defined in Table 4.1. [25, 33, 93]. In the modified $k$–means approach [94, 101, 103] the representative vector of the $i$th cluster is computed as the *principal eigenvector* of the correlation matrix calculated using all the vectors assigned to the $i$th cluster, where the principal eigenvector is the eigenvector corresponding to the largest eigenvalue. Different expressions for the distance measure $d(\cdot,\cdot)$ were proposed in [94, 101, 103].

It it obvious that for a clustering algorithm to produce a good estimate of the mixing matrix, this algorithm must be robust to the presence of outlier points, and the number of clusters must be estimated from the data set. However, as described in Section 4.1, most partitioning clustering algorithms are sensitive to outlier points and need the number of clusters to be provided.

In contrast to the partitioning clustering algorithms, there are some clustering algorithms that can estimate the number of clusters from the data matrix $\boldsymbol{X}$, or equivalently $\boldsymbol{Y}$. One of these approaches is based on constructing a histogram of

Table 4.1: Steps of a partitioning clustering algorithm.

$[\tilde{\boldsymbol{A}}]$ = Partitioning clustering($\boldsymbol{Y}$, $n$)

1. Initialization: Initialize the clusters' representative vectors $[\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n]$, and construct the new data matrix $\boldsymbol{Z}$ from the data matrix $\boldsymbol{Y}$ using the two steps presented in Section 4.2.2.

2. Repeat until convergence:

E-step: Assign the $k$th column of the data matrix $\boldsymbol{Z}$ into the closest cluster $\Omega_i$.

$\Omega_i = \{\boldsymbol{z}_k : d(\boldsymbol{z}_k, \boldsymbol{c}_i) < d(\boldsymbol{z}_k, \boldsymbol{c}_j) \quad \forall i \neq j\}$, where $d(\boldsymbol{z}_k, \boldsymbol{c}_i)$ is the distance between the $k$th column of $\boldsymbol{Z}$ and the representative point of the $i$th cluster.

M-step: Update the cluster representative vector $\boldsymbol{c}_i$ of the $i$th cluster $\Omega_i, i = 1, \ldots, n$.

End.

3. Output $[\tilde{\boldsymbol{A}}] = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n]$.

the angles corresponding to the orientations of the columns of the feature matrix $\boldsymbol{Y}$. Motivated by the 2-D scatter plot, the idea of this approach is based on estimating the orientation of each data point $\boldsymbol{y}_k \in \mathbb{R}^2$ using the equation

$$\alpha_k = \tan^{-1}\left(\frac{y_k[2]}{y_k[1]}\right), \tag{4.12}$$

and then constructing a histogram plot for the vector of the calculated angles $\boldsymbol{\alpha}$. The number of peaks in the histogram corresponds to the number of clusters, and the corresponding angle bins correspond to the orientations of the columns of the mixing matrix [95, 96]. The accuracy of the angular histogram approach for estimating the columns of the mixing matrix is determined by the bin width that is assumed in evaluating the histogram. To enhance the accuracy, Parzen windowing was utilized

in [92] to estimate the probability density of the angles, and then the $n$ angles that produce the $n$ largest peaks of this distribution are used as the estimates for the directions of the $n$ columns of the mixing matrix $\boldsymbol{A}$. Although the angular histogram approach is efficient for the $m = 2$ case, its complexity grows exponentially with $m$. In the next section we propose a modification to that approach such that the new algorithm is applicable to the general case $m \geq 2$.

Since the number of sources is generally unknown, several approaches have been proposed in the literature for estimating the mixing matrix without prior knowledge about the number of clusters (sources). In [31] the authors used the clustering algorithm proposed in [106] which can extract an unknown number of clusters from a given data set. The estimated clusters are refined by first removing the outlier points from each cluster and then removing the clusters that have a number of points less than a pre–specified number. The disadvantage of this method is that its performance is very sensitive to the values of the two parameters that identify the outlier points and the cluster size, and no rigorous method was provided for selecting these two parameters.

Another approach for estimating the number of clusters from the data set was proposed in [103]. This approach is based on applying a modified $k$–means clustering algorithm with the number of clusters being over estimated, e.g., if the expected number of sources is $n$ then the number of clusters $K$ is selected as $K > 10n$. After estimating the $K$ clusters, a correlation matrix is calculated for the points in each cluster and the largest eigenvalue of the correlation matrix is calculated. The $K$ eigenvalues, one for each cluster, are then arranged in a descending order and plotted. If the value of $K$ is chosen properly and the noise level is small, the eigenvalues corresponding to the hyperline clusters will be larger than the other ones, and the plotted curve will show a transition gap at the index indicating the number of clusters. The clusters corresponding to the largest eigenvalues are then extracted and the

principal eigenvector of each of the extracted clusters is considered as an estimate of one column of the mixing matrix. Although this approach produced promising results, good results can be obtained only when the value of $K$ is selected to be much larger than the number of sources, which is unknown. For example, it was shown in [103] that, for the case of 5 sources, selecting $K = 20$ is not sufficient to produce acceptable results, while good results were obtained only for values of $K > 50$. Since this algorithm is a partitioning clustering algorithm, its performance is very sensitive to the initial estimates of the clusters' representative vectors. To overcome this difficulty, the authors of [103] proposed a sophisticated and complex initialization step.

In the next section we propose three novel clustering algorithms that can estimate both the number of sources and the mixing matrix. In contrast to the partitioning clustering algorithms, the proposed algorithms can estimate the number of sources from the data set and do not suffer from the problem of outlier points nor initialization.

The first clustering algorithm we propose is based on a clustering technique called *hierarchical clustering* (HC). Hierarchical clustering algorithms start with every column of $\boldsymbol{Z}$ assigned into a separate cluster. Then, in each successive step, the two closest[1] clusters are merged into a single cluster. The process continues until all columns are assigned into a single cluster. The main difficulty associated with HC is the identification of the individual clusters. Previous approaches for identifying the clusters are either manual or depend on some parameters which are hard to determine. See Section 4.3.1 for more details. In this chapter we propose a novel clustering algorithm in which a statistical test is utilized for identifying the individual clusters. The statistical test is applied at each step of the hierarchical process to test whether the two closest clusters are significantly different. The two closest clusters are merged

[1]The method to measure closeness of clusters is defined later.

into a single cluster if they are not significantly different, and the process continues. Otherwise, the largest (i.e., the one with largest number of objects) of them is extracted and removed from the data matrix. The process is repeated until all clusters are extracted. For identifying the clusters that correspond to the columns of the mixing matrix, we developed a quantitative measure called the *concentration parameter* (CP). If the disjoint orthogonality condition is satisfied for a reasonably large number of columns of the source coefficient matrix, then the number of sources can also be estimated using this parameter.

The idea of the second proposed clustering algorithm is inspired by the angular histogram clustering algorithm [92, 95, 96]. The previously suggested angular histogram clustering algorithms are difficult to extend beyond the 2-D case. However, in this chapter we propose a new algorithm that generalizes the previous algorithms to the $m \geq 2$ case. Under certain conditions, which will be described in the next section, the proposed algorithm can efficiently estimate the mixing matrix even when the disjoint orthogonality condition is satisfied for a small percentage of the columns of the source coefficient matrix.

The third clustering algorithm that we propose in this chapter is a combination of the first two algorithms. This algorithm combines the advantages of the first two clustering algorithms and avoids their limitations. As will be shown in the simulation results, the third algorithm is very efficient in estimating the mixing matrix and the number of sources.

## 4.3 Proposed Clustering Algorithms

As described in Section 4.2.3, most of the clustering algorithms previously utilized for estimating the mixing matrix, in the two–step approach, belong to the partitioning clustering class, which suffer from the three limitations described in Section 4.1. In

this section we propose three clustering algorithms that are based on different classes of clustering techniques. The first algorithm is based on the hierarchical clustering technique, while the second algorithm is based on the angular histogram technique. The third algorithm is a combination of these two algorithms. Before explaining the proposed clustering algorithms, we provide brief description of the hierarchical clustering technique and the statistical test utilized in the first proposed algorithm.

### 4.3.1 Hierarchical Clustering

Table 4.2: Steps of a HC clustering algorithm.

1. Initialization: Start with $\hat{T}$ singleton clusters.
2. Calculate the dissimilarity matrix $\boldsymbol{D} \in \mathbb{R}^{\hat{T}\times\hat{T}}$, whose $(i,j)$th coefficient is given by $d_{ij} = ||\boldsymbol{z}_i - \boldsymbol{z}_j||_{\ell_2}, \quad i,j = 1,2,\ldots,\hat{T}$.
3. For $k = 1,\ldots,\hat{T}$, repeat the following two steps:
   - Search the minimal distance between clusters
     $$D(C_u, C_v) = \min_{1\le i<j\le \hat{T}_k} D(C_i, C_j)$$
     where $D(*,*)$ is a distance function between two clusters, to be defined, and $\hat{T}_k = \hat{T} - k + 1$ is the number of clusters at the $k-th$ iteration.
   - Merge clusters $C_u$ and $C_v$ into a single cluster, and update the dissimilarity matrix.

Hierarchical clustering (HC) follows a clustering strategy which is quite distinct

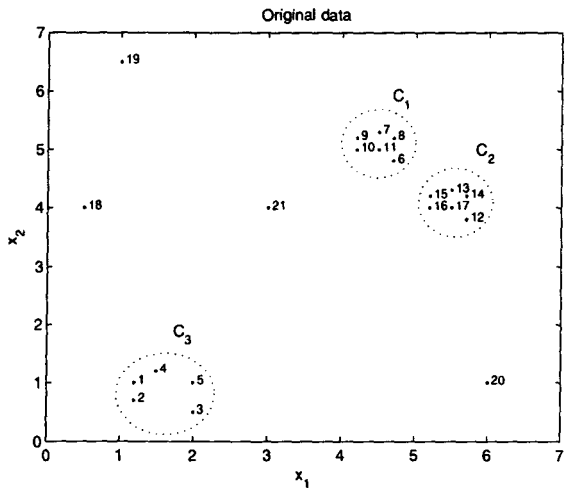
from that of the partitioning clustering algorithms. Hierarchical clustering algorithms start with every object (columns of $\boldsymbol{Z}$ in our case) assigned into a separate cluster. Then, in each successive step, the two closest clusters are merged into a single cluster. The process continues until all objects are assigned into a single cluster. The general steps of clustering the columns of the data matrix $\boldsymbol{Z} \in \mathbb{R}^{m \times \hat{T}}$ using a HC algorithm are summarized in Table 4.2 [105, 107], where $m$ is the number of variables and $\hat{T} \leq T$ is the number of columns of the feature matrix. The difference between different HC algorithms resides in the definition of the distance $D(*,*)$ between two clusters. The most popular distances used with the HC algorithms are the single–linkage, the complete–linkage and the average–linkage, see [105,107] for the definition and comparison between these distances. In the first proposed algorithm we use the average–linkage distance to measure the distance between two clusters. The average–distance between any two clusters $R$ and $Q$ is defined as [107]

$$D(R,Q) = \frac{1}{|R||Q|} \sum_{i \in R, j \in Q} d_{ij} \tag{4.13}$$

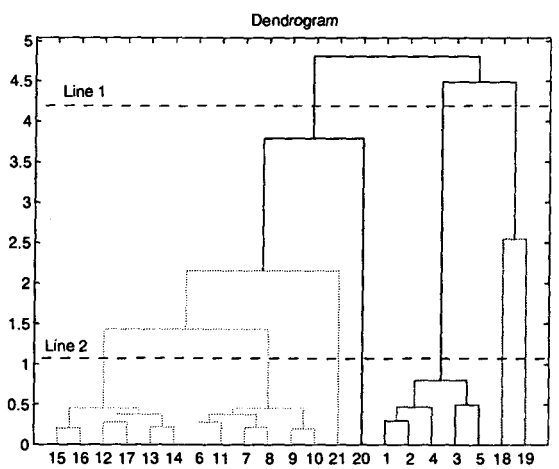
where $|R|$ and $|Q|$ denote the number of objects in clusters $R$ and $Q$, respectively, and $d_{ij}$ is defined in Table 4.2.

It is convenient to depict the result of an HC algorithm using a dendrogram plot. The dendrogram plot resulting from applying a HC algorithm on the simple data set shown in Figure 4.3(a) is shown in Figure 4.3(b). In this plot, the root node of the dendrogram represents the whole data set and each leaf node is regarded as a data object. An internal node represents the union of all objects in its subtree, and the height of an internal node represents the distance between its two child nodes [107]. Note that the labels of the root nodes are exactly those of the clustered data points in Figure 4.3(a).

A useful property associated with the HC algorithms [108], which is also depicted in Figure 4.3(b), is that closer objects are combined in early stages while distant




(a)




(b)

Figure 4.3: (a) A data set consists of 3 clusters and 4 outlier points; (b) Clustering the data set presented in (a) using hierarchical clustering.

objects (outliers) are combined in later stages. However, the main difficulty associated with HC is identifying the individual clusters. As shown in Figure 4.3(b), the clusters are not explicit and have to be determined in some way from the dendrogram.

Previous approaches for identifying the clusters from the dendrogram include: 1) converting the dendrogram into a reachability plot [108, 109], 2) manually selecting individual clusters from the dendrogram, 3) selecting a horizontal line, denoted as Line 1 and Line 2 in Figure 4.3(b), that cuts the dendrogram at the level at which the number of nodes equals the desired number of clusters. In this case the combination of all children connected to each resulting node is considered as a cluster. Unfortunately, each one of these approaches has its own limitations. For instance, the first approach requires an input parameter which is difficult to determine, while the other two approaches are user dependent and cannot be automated. For example, automatic selection of a horizontal line in Figure 4.3(b) such that the number of clusters equals 3 will result in Line 1. It is clear from Figure 4.3(b) that this is not a good choice; since both $C_1$ and $C_2$ are combined in a single cluster and two outlier objects, $obj_{18}$ and $obj_{19}$, constitute one of the estimated clusters. On the other hand, a manual selection of the horizontal line will select Line 2 because in this case each of the original three true clusters $C_1$–$C_3$ is well defined by an estimated cluster.

In Section 4.3.3 we propose utilizing the $T^2$–statistical test to identify the clusters from the dendrogram. The $T^2$–statistical test is used for testing the equality of the means of two multivariate populations. A brief revision of the $T^2$–statistical test is presented in the next subsection.

### 4.3.2 $T^2$–statistical Test

Consider the following two clusters $R = [\boldsymbol{r}_1\boldsymbol{r}_2 \dots \boldsymbol{r}_{T_1}]$ and $Q = [\boldsymbol{q}_1\boldsymbol{q}_2 \dots \boldsymbol{q}_{T_2}]$, where both $\boldsymbol{r}_i$ and $\boldsymbol{q}_j \in \mathbb{R}^{m\times 1}$. It is assumed that: 1) The sample set $\boldsymbol{r}_1\boldsymbol{r}_2 \dots \boldsymbol{r}_{T_1}$ is a random sample of size $T_1$ from an $m$–variate population with mean vector $\boldsymbol{\mu}_1$ and

covariance matrix $\Sigma_1$, 2) The sample set $\boldsymbol{q}_1\boldsymbol{q}_2\ldots\boldsymbol{q}_{T_2}$ is a random sample of size $T_2$ from an $m$–variate population with mean vector $\boldsymbol{\mu}_2$ and covariance matrix $\Sigma_2$, 3) Also, $\boldsymbol{r}_1\boldsymbol{r}_2\ldots\boldsymbol{r}_{T_1}$ are independent of $\boldsymbol{q}_1\boldsymbol{q}_2\ldots\boldsymbol{q}_{T_2}$. Our goal is to test whether the null hypothesis

$$H_0 : \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 = 0$$

is true or not.

$T^2$–statistic test [110] is utilized for testing the validity of $H_0$. The three assumptions mentioned before are sufficient for running the $T^2$–statistic test as long as the sample sizes $T_1$ and $T_2$ are large enough. However when the sample sizes are small, the following two assumptions are needed: 4) Both populations are multivariate normal, 5) $\Sigma_1 = \Sigma_2$.

The $T^2$–statistic test can be summarized as follows [110]:

1. For each cluster calculate the unbiased estimate of both the mean vector and covariance matrix as follows

$$\boldsymbol{\mu}_r = \frac{1}{T_1}\sum_{j=1}^{T_1}\boldsymbol{r}_j \qquad \boldsymbol{S}_r = \frac{1}{T_1-1}\sum_{j=1}^{T_1}(\boldsymbol{r}_j-\boldsymbol{\mu}_r)(\boldsymbol{r}_j-\boldsymbol{\mu}_r)^T$$

$$\boldsymbol{\mu}_q = \frac{1}{T_2}\sum_{j=1}^{T_2}\boldsymbol{q}_j \qquad \boldsymbol{S}_q = \frac{1}{T_2-1}\sum_{j=1}^{T_2}(\boldsymbol{q}_j-\boldsymbol{\mu}_q)(\boldsymbol{q}_j-\boldsymbol{\mu}_q)^T$$

2. Calculate the pooled covariance matrix

$$\boldsymbol{S}_{po} = \frac{(T_1-1)\boldsymbol{S}_r + (T_2-1)\boldsymbol{S}_q}{T_1+T_2-2}$$

3. Calculate the quantity

$$T^2 = \frac{T_1T_2}{T_1+T_2}(\boldsymbol{\mu}_r-\boldsymbol{\mu}_q)^T\boldsymbol{S}_{po}^{-1}(\boldsymbol{\mu}_r-\boldsymbol{\mu}_q)$$

which is distributed as $\frac{(T_1+T_2-2)m}{(T_1+T_2-m-1)}$ times $F_{m,T_1+T_2-m-1}$, where $F_{a,b}$ is an $F$-distribution with $a$ and $b$ degrees of freedom $(d.f.)$

4. Select a value for $\alpha$, e.g. $\alpha = 0.05$, then reject $H_0$ if $T^2 > \frac{(T_1+T_2-2)m}{(T_1+T_2-m-1)} F_{m,T_1+T_2-m-1}(\alpha)$, where $F_{m,T_1+T_2-m-1}(\alpha)$ is the upper $(100\alpha) - th$ percentile of an $F$-distribution with $m$ and $(T_1 + T_2 - m - 1)$ $d.f.$

### 4.3.3 First Clustering Algorithm: Sequential Cluster Extraction (SCE)

As described in Section 4.3.1, a hierarchical clustering algorithm starts by assigning each column of the data matrix $\boldsymbol{Z}$ into a separate cluster, i.e., the initial guess of the number of clusters is the number of available data points. Therefore, in contrast to the partitioning clustering algorithms, hierarchical clustering algorithms do not suffer from the initialization problem. Then the clustering process proceeds in steps where in each step the closest two clusters are merged into a single cluster. Accordingly, the effect of the outlier points on the performance of the HC algorithms can be greatly alleviated, since they are combined into clusters at the last stages. This behavior is depicted by the dendrogram in Figure 4.3(b) in which the four outlier objects $obj_{18}$–$obj_{21}$ are combined into clusters after the main three clusters are constructed. Accordingly, the effects of the outlier points can be alleviated, or even eliminated, if the hierarchical process stopped before combining these outlier points.

The main difficulty associated with the hierarchical clustering algorithms is the identification of the clusters. For solving this problem we suggest a novel approach which is based on utilizing the $T^2$–statistical test at each merging step to decide whether the means of the two closest clusters are significantly different or not. If they are not significantly different, the two clusters are merged and the process continues. Otherwise, the cluster with the largest number of objects is extracted and removed from the data set. The mean of the extracted cluster is considered as an estimate of one column of the mixing matrix. The process can then be repeated until all the

clusters are extracted. Since the closest clusters are merged in the early stages, the early extracted clusters most likely correspond to the columns of the mixing matrix, while the extracted clusters at later stages are most likely due to outlier points, i.e., the points due to the linear combination of more than one column of the mixing matrix. The proposed algorithm is summarized in Table 4.3.

The algorithm has two inputs and two outputs. The first input is the data matrix $\boldsymbol{Z}_l$ which equals the data matrix $\boldsymbol{Z}$ after removing the columns corresponding to the previously extracted $(l-1)$ clusters. Accordingly, $\boldsymbol{Z}_1 = \boldsymbol{Z}$. Recall that the data matrix $\boldsymbol{Z}$ is constructed from the coefficients data matrix $\boldsymbol{X}$ using the preprocessing steps described in Section 4.2.2 with the normalization step performed using (4.11). The second input is the value of $\alpha$ required for applying the statistical test. On the other hand, the first output $\mathcal{I}_l$ contains the indices of the $l$th extracted cluster, while the second output is the input data matrix after removing the columns corresponding to the extracted cluster. Since the proposed algorithm extracts only one cluster, it can be repeated many times until the number of the remaining columns in the data matrix is less than or equal to 1.

#### 4.3.3.1 Estimating the mixing matrix and the number of sources

Let $L$ denote the total number of estimated clusters from the data matrix $\boldsymbol{Z}$ using Algorithm 1. To identify the clusters corresponding to the columns of the mixing matrix, we make use of the assumption that the disjoint orthogonality property is satisfied for a sufficiently large number of columns of the source coefficient matrix $\boldsymbol{S}$. Accordingly, for the columns of the source coefficient matrix $\boldsymbol{S}$ that contain a single nonzero entry, the corresponding columns of $\boldsymbol{X}$, or equivalently $\hat{\boldsymbol{Y}}$ defined in (4.7), constitute hyperlines in the $m$ dimensional space, where the orientations of the hyperlines correspond to the columns of the mixing matrix $\boldsymbol{A}$. Therefore, to identify the number of sources and the clusters corresponding to the columns of the mixing

Table 4.3: Sequential Cluster Extraction Algorithm

Algorithm 1: $[\mathcal{I}_l,\ \boldsymbol{Z}_{l+1}] = \mathrm{SCE}(\boldsymbol{Z}_l,\ \alpha)$

1. Start with $T^l$ singleton clusters, where $T^l$ is the number of columns of $\boldsymbol{Z}_l$.

2. Calculate the dissimilarity matrix $\boldsymbol{D} \in \mathbb{R}^{T^l \times T^l}$, whose $(i,j)$th coefficient is given by $d_{ij} = ||\boldsymbol{z}_i^l - \boldsymbol{z}_j^l||_{\ell_2}, \quad i,j = 1,2,\ldots T^l$, where $\boldsymbol{z}_i^l$ is the $i$th column of $\boldsymbol{Z}_l$.

3. Search the minimal distance between clusters

$$D(C_u, C_v) = \min_{1 \le i < j \le T_k} D(C_i, C_j),$$

where $D(*,*)$ is a distance function between two clusters, defined in (4.13), and $T_k$ is the number of clusters at the $k-th$ iteration.

4. For the given value of $\alpha$, apply the $T^2$–statistical test, presented in Section 4.3.2, to test whether the centroids (means) of the two clusters $C_u$ and $C_v$ are significantly different or not.

5. if the two means are not significantly different, merge clusters $C_u$ and $C_v$ in a single cluster, and update the dissimilarity matrix. Go to step 3.

   else

   if $|C_v| \le |C_u|$, set $\mathcal{I}_l = [i_{u1}, \ldots, i_{uT_u}]$;

   else set $\mathcal{I}_l = [i_{v1}, \ldots, i_{vT_v}]$;

   break

   end

   end

6. $\boldsymbol{Z}_{l+1} = \boldsymbol{Z}_l / \boldsymbol{Z}_l(:\ , \mathcal{I}_l)$.

matrix, we propose calculating a parameter called the "*concentration parameter*" (CP) from the data matrix $\hat{\boldsymbol{Y}}$. The CP parameter measures how well the points of each cluster are concentrated around a hyperline in the $m$ dimensional space. There are many ways for defining a CP that measures the concentration of the columns of a given data matrix around a hyperline in the $m$ dimensional space. In this subsection we propose two different concentration parameters.

Recall that the columns of the normalized data matrix $\boldsymbol{Z}$ are used for estimating the indices of the estimated clusters using Algorithm 1. Since there is a one–to–one correspondence between the columns of $\boldsymbol{Z}$ and $\hat{\boldsymbol{Y}}$, each extracted cluster from $\boldsymbol{Z}$ corresponds to a cluster in $\hat{\boldsymbol{Y}}$ with the same indices. Accordingly, for each extracted cluster, the value of the CP parameter is calculated from the corresponding cluster in the data matrix $\hat{\boldsymbol{Y}}$. To be specific, let $\mathcal{I}_l$ denote the set of indices of the $l$th cluster estimated from $\boldsymbol{Z}_l$ using Algorithm 1, and let $\boldsymbol{C}^l = \hat{\boldsymbol{Y}}_l(:, \mathcal{I}_l)$ denote the matrix constructed from the columns of the data matrix $\hat{\boldsymbol{Y}}_l$ with indices $\mathcal{I}_l$, where $\hat{\boldsymbol{Y}}_l$ consists of the data matrix $\hat{\boldsymbol{Y}}$ after removing the columns corresponding to the previously extracted $(l-1)$ clusters. Let $\boldsymbol{R}^l$ be defined as

$$\boldsymbol{R}^l = \boldsymbol{C}^l \boldsymbol{C}^{l^T}. \tag{4.14}$$

If the $l$th cluster constitutes a hyperline in the $m$ dimensional space, the orientation of the hyperline will coincide with that of the principal eigenvector of $\boldsymbol{R}^l$. Therefore, the principal eigenvector of $\boldsymbol{R}^l$ will be used as the representative vector of the $l$th cluster. Let $\boldsymbol{b}_l$ denote the representative vector of the $l$th cluster, then the matrix $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_L]$ can be considered as an initial estimate of the mixing matrix $\boldsymbol{A}$. If $L > n$, where $n$ is the number of sources, then a better estimate of $\boldsymbol{A}$ can be obtained by selecting the $n$ columns of $\boldsymbol{B}$ that are very close to the columns of the mixing matrix. However, since the columns of the mixing matrix as well as the number of sources are unknowns, they must be estimated. For accomplishing that

goal, we propose the following two parameters, which measure the concentration of the points of a given cluster around the representative point of this cluster.

### First concentration parameter

It is known that all the points constructing a hyperline cluster in the $m$ dimensional space are concentrated around a hyperline oriented in the direction of the principal eigenvector of that cluster. Therefore, a cluster corresponding to one column of the mixing matrix can be identified by calculating the average distance between the points in that cluster and its principal eigenvector. The first concentration parameter (CP1) of the $l$th cluster, $\beta_l$, is defined as

$$\beta_l = \exp(-\hat{\beta}_l / T_l). \tag{4.15}$$

where $T_l$ is the total number of points in the $l$th cluster, i.e., the number of columns of $\boldsymbol{C}^l$, and $\hat{\beta}_l$ is the average distance between the columns of $\boldsymbol{C}^l$ and $\boldsymbol{b}_l$, the principal eigenvector of $\boldsymbol{R}^l$ defined in (4.14). Let $\boldsymbol{C}^l = \left[\boldsymbol{c}_1^l, \ldots, \boldsymbol{c}_{T_l}^l\right]$ denote the $l$th cluster, then the distance from $\boldsymbol{c}_i^l$ to the hyperline represented by $\boldsymbol{b}_l$, is given by [100]

$$d_i^l = \sqrt{||\boldsymbol{c}_i^l||_{\ell_2}^2 - \frac{\left(\boldsymbol{b}_l{}^T \boldsymbol{c}_i{}^l\right)^2}{||\boldsymbol{b}_l||_{\ell_2}^2}}. \tag{4.16}$$

and $\hat{\beta}_l$ is then calculated as

$$\hat{\beta}_l = \frac{1}{T_l} \sum_{i}^{T_l} d_i^l. \tag{4.17}$$

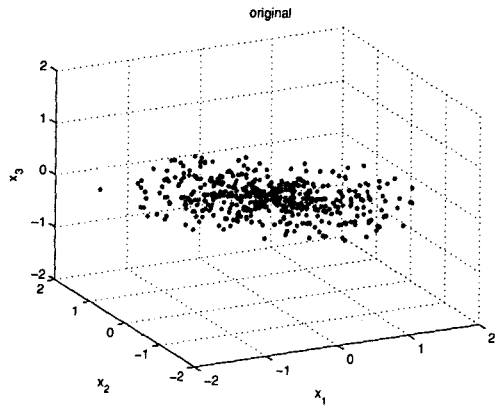
Note that if the $l$th and $k$th clusters have different sizes and the same average distance, i.e., $\hat{\beta}_l = \hat{\beta}_k, l \neq k$, and $T_l > T_k$ say, then the $l$th cluster is considered more concentrated than the $k$th cluster. This point was taken into consideration by dividing the value of $\hat{\beta}_l$ by the cluster's size $T_l$ in (4.15).

If the columns of the $l$th cluster $\boldsymbol{C}^l$ are concentrated around the hyperline represented by $\boldsymbol{b}_l$, then the value of $\hat{\beta}_l$ will be very small and the corresponding value

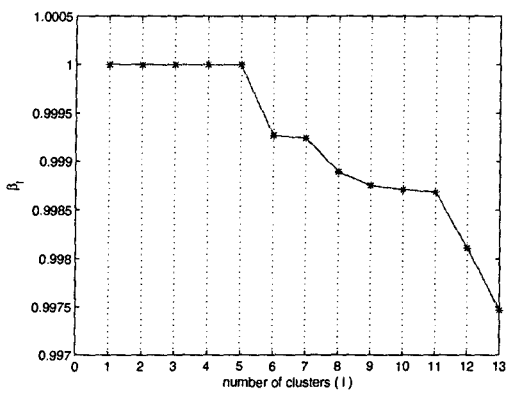
of $\beta_l$ will be very close to 1. Accordingly, $\boldsymbol{b}_l$ can be considered as an estimate of one column of the mixing matrix $\boldsymbol{A}$. On the other hand, if the columns of the $l$th cluster are not concentrated around $\boldsymbol{b}_l$, the value of $\hat{\beta}_l$ will be large resulting in a small value of $\beta_l$. In this case the $l$th cluster consists of some points in the cloud in Figure 4.2. Accordingly, the value of CP1 can be used to identify the number of sources and the clusters corresponding to the columns of the mixing matrix. This can be accomplished by plotting the concentration parameters $\beta_l, l = 1, \ldots, L$ after arranging them in descending order. If the disjoint orthogonality property is satisfied for a reasonably large number of columns of the source matrix such that the hyperlines are well defined in the $m$ dimensional space, the plot of the arranged values of CP1 will have a value close to 1 for the clusters corresponding to the columns of the mixing matrix. The value of CP1 drops rapidly for the clusters corresponding to the cloud points. This behavior is depicted in Figure 4.4(b).

The data set shown in Figure 4.4(a) was constructed by multiplying a sparse matrix $\boldsymbol{S} \in \mathbb{R}^{5\times1000}$ by a mixing matrix $\boldsymbol{A} \in \mathbb{R}^{3\times5}$ randomly generated from a white normal distribution with zero mean and unit variance. The sparse matrix is constructed such that 30% of its columns satisfy the disjoint orthogonality principal, i.e., each source is uniquely represented by 60 $(= 0.3 * 1000/5)$ columns. The indices of the nonzero entries of each row of $\boldsymbol{S}$ are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm1$.

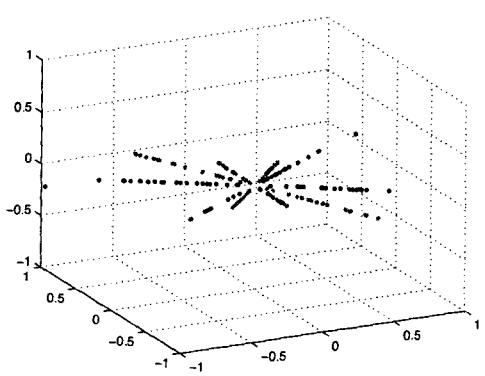
Since the number of columns of the mixing matrix $\boldsymbol{A}$ is greater than the number of rows, it is expected that some columns of the mixing matrix could be very close to each other, and hence they could be combined into the same cluster. The closeness between the columns of $\boldsymbol{A}$ can be measured by calculating the correlation coefficients




(a)




(b)



(c)

Figure 4.4: (a) A scatter plot of the measured data matrix, (b) The CP1 parameter of the estimated clusters, (c) The first 5 estimated clusters.

between these columns, which is given by

$$\boldsymbol{R}_{AA} = \begin{bmatrix} 1.0000 & 0.1829 & 0.2603 & 0.5900 & -0.5278 \\ 0.1829 & 1.0000 & \mathbf{0.9969} & 0.9017 & 0.7384 \\ 0.2603 & \mathbf{0.9969} & 1.0000 & \mathbf{0.9332} & 0.6827 \\ 0.5900 & 0.9017 & \mathbf{0.9332} & 1.0000 & 0.3744 \\ -0.5278 & 0.7384 & 0.6827 & 0.3744 & 1.0000 \end{bmatrix}$$

It is clear that the second and third columns of $\boldsymbol{A}$ are very close to each other, and hence they could be combined into a single cluster.

After running the proposed clustering algorithm, and as shown in Figure 4.4(b), the total number of the extracted clusters is 13. However, the value of CP1 is approximately one for five clusters only, i.e., these corresponding to the columns of the mixing matrix. The five clusters with largest values of CP1 are shown in Figure 4.4(c). It is worth mentioning that these are the clusters that are extracted first by the proposed algorithm. As shown in Figure 4.4(c), each cluster constitutes a hyperline in the 3 dimensional space.

As a measure of closeness between the true mixing matrix and the estimated mixing matrix, the cross-correlation matrix between these two matrices is calculated (after correcting for the sign and permutation ambiguities) and is given by

$$\boldsymbol{R}_{A\tilde{A}} = \begin{bmatrix} \mathbf{1.0000} & 0.1829 & 0.2561 & 0.5946 & -0.5278 \\ 0.1829 & \mathbf{1.0000} & 0.9972 & 0.8992 & 0.7384 \\ 0.2603 & 0.9969 & \mathbf{1.0000} & 0.9311 & 0.6827 \\ 0.5900 & 0.9017 & 0.9316 & \mathbf{1.0000} & 0.3744 \\ -0.5278 & 0.7384 & 0.6858 & 0.3690 & \mathbf{1.0000} \end{bmatrix}$$

It is clear that the absolute correlation coefficients between the estimated columns and the columns of the mixing matrix are ones, which is an indication of perfect estimation. On the other hand, the cross-correlation matrix between the true mixing

matrix and the mixing matrix estimated using $k$-means is given by

$$\boldsymbol{R}_{AA_{km}} = \begin{bmatrix} \mathbf{0.9473} & 0.1207 & 0.6198 & 0.8761 & -0.7440 \\ -0.1417 & \mathbf{0.9980} & 0.8849 & -0.3136 & 0.5208 \\ 0.3002 & 0.8727 & \mathbf{0.9993} & 0.1276 & 0.1006 \\ -0.0628 & 0.9899 & 0.9190 & -0.2374 & 0.4515 \\ -0.7721 & 0.7795 & 0.3393 & -0.8719 & \mathbf{0.9602} \end{bmatrix}$$

which shows that the fourth estimated cluster does not correspond to any column of the mixing matrix. Furthermore, the signal to interference ratio (SIR) (defined in (4.20)) between the true mixing matrix and the mixing matrix estimated using the proposed algorithm is 51.6992 dB, while the SIR between the true mixing matrix and the mixing matrix estimated using $k$-means is only 8.1376 dB.

### Second concentration parameter

The second CP proposed in this section uses the values of the eigenvalues of $\boldsymbol{R}^l$, defined in (4.14), of the $l$th cluster to investigate whether this cluster is a concentrated cluster or not. To be specific, let $\boldsymbol{C}^l \in \mathbb{R}^{m \times T_l}$, where $T_l$ is the number of objects in the $l$th cluster, denote the matrix constructed from the columns of the data matrix $\hat{\boldsymbol{Y}}_l$ with indices corresponding to the $l$th extracted cluster, and let $\boldsymbol{R}^l \in \mathbb{R}^{m \times m}$ be defined as in (4.14). The eigenvalue decomposition of $\boldsymbol{R}^l$ can be expressed as

$$\boldsymbol{R}^l = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T,$$

where the columns of $\boldsymbol{U}$ are the eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues sorted in the main diagonal. It is assumed without loss of generality that the eigenvalues are arranged in a descending order, i.e. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$.

If the columns of $\boldsymbol{C}^l$ are concentrated around a hyperline in the $m$ dimensional space, then the value of $\lambda_1$, the first eigenvalue of $\boldsymbol{R}^l$, will be much larger than the value of $\lambda_2$, which in turn will be very close to the value of $\lambda_m$. In this case, the plot

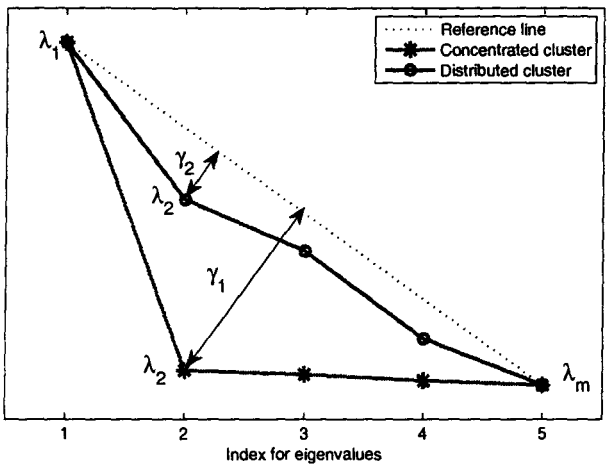



Figure 4.5: The distance from $\lambda_2$ to the virtual line connecting $\lambda_1$ and $\lambda_m$ can be used as a concentration parameter.

of the eigenvalues will show an L–shape. This behavior is depicted in Figure 4.5 by the red curve. On the other hand, if the columns of $\boldsymbol{C}^l$ are not concentrated around a hyperline in the $m$ dimensional space, the plot of the eigenvalues will not show an L–shape, and the transition from one eigenvalue to the other will be smooth. This behavior is depicted in Figure 4.5 by the blue curve.

From this geometric interpretation it is clear that there is a direct relation between the concentration of a cluster and the shape of the plot of its eigenvalues. What remains is to calculate a parameter whose value reflects this relation. There are many possible ways for calculating this parameter. For instance, this parameter could be calculated as the relative difference between $\lambda_1$ and $\lambda_2$, i.e, $CP = \frac{\lambda_1 - \lambda_2}{\lambda_1 - \lambda_m}$. Empirically, we found that a plot of this parameter does not show a sharp transition gap at the index corresponding to the number of sources.

In this chapter we propose using the distance from $\lambda_2$ to a *virtual* line connecting $\lambda_1$ and $\lambda_m$ as the second concentration parameter (CP2). The virtual line connecting

$\lambda_1$ and $\lambda_m$ is shown by the dotted line in Figure 4.5, and the distance is depicted by the double arrow. As shown in this figure, $\gamma_1 \gg \gamma_2$, where $\gamma_1$ and $\gamma_2$ are the distances corresponding to the concentrated cluster and the distributed cluster, respectively. Accordingly, the value of CP2 can be used to differentiate between concentrated and distributed clusters.

It is not hard to prove that the distance from the point $(2, \lambda_2)$ to the line connecting the two points $(1, \lambda_1)$ and $(m, \lambda_m)$ is given by

$$\gamma = \frac{(m-1)(\lambda_1 - \lambda_2) - (\lambda_1 - \lambda_m)}{\sqrt{(m-1)^2 + (\lambda_1 - \lambda_m)^2}}. \tag{4.18}$$

The value of $\gamma$ can be used for estimating the number of sources and the clusters corresponding to the columns of the mixing matrix. This can be accomplished by first calculating $\{\gamma_l, l = 1, \ldots, L\}$, where $\gamma_l$ is the value of $\gamma$ for the $l$th cluster, and $L$ is the total number of clusters. Then plot $\{\gamma_l\}$ after arranging them into descending order. If the disjoint orthogonality property is satisfied for a reasonably large number of columns of the source matrix, the plot of the arranged values of CP2 will have large value for the clusters corresponding to the columns of the mixing matrix. The value of CP2 is expected to drop rapidly for the clusters corresponding to the cloud points.

#### 4.3.3.2 SCE for noisy measurements

When the measurements are contaminated with additive noise, it may become difficult to identify a transition gap in the plot of the concentration parameters. To clarify this point, consider the following example in which a sparse source matrix $\boldsymbol{S} \in \mathbb{R}^{7\times 800}$ is randomly generated and multiplied by a mixing matrix $\boldsymbol{A} \in \mathbb{R}^{5\times 7}$ randomly generated from a white normal distribution with zero mean and unit variance. The sparse matrix is constructed such that 50% of its columns satisfy the disjoint orthogonality principal, i.e., each source is uniquely represented by 58 ($= \lceil 0.5*800/7 \rceil$) columns.The indices of

the nonzero entries of each row of $\boldsymbol{S}$ are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm 1$. A zero mean white Gaussian noise matrix $\boldsymbol{V} \in \mathbb{R}^{5\times 800}$ is randomly generated. The amplitude of the noise is adjusted to produce a $30dB$ SNR. After extracting the clusters using the SCE algorithm, the two concentration parameters are calculated and arranged in a descending order. The arranged CP1 and CP2 are shown in Figure 4.6(a)–(b), respectively. As shown in these figures, the CP1 shows a sharp transition at a number of clusters equal to 8 rather than 7, while CP2 show a small transition at index 7 and large transition at index 9.

To overcome this problem we propose performing a *trimming* step after estimating each cluster. The trimming step simply removes all the points that are farther than a certain threshold from the representative point of the cluster. To be specific, let $\boldsymbol{C}^l = [\boldsymbol{c}_1^l, \ldots, \boldsymbol{c}_{T_l}^l]$ denote the $l$th cluster, and let $\boldsymbol{u} \in \mathbb{R}^m$ be the representative point of this cluster. The representative point could be chosen as the principal eigenvector of $\boldsymbol{R}^l = \boldsymbol{C}^l \boldsymbol{C}^{l^T}$ for instance. Then the trimming step can be expressed mathematically as

$$\boldsymbol{C}^l \leftarrow \left\{ \boldsymbol{c}_i^l : \frac{\boldsymbol{u}^T \boldsymbol{c}_i^l}{||\boldsymbol{c}_i^l||_{\ell_2}} \geq \rho \right\}, \tag{4.19}$$

where $0 \ll \rho < 1$ is a pre-specified threshold. For all the examples presented in this chapter, the value of $\rho$ is selected empirically as $\rho = 0.97$. In (4.19) it was assumed that $||\boldsymbol{u}||_{\ell_2} = 1$. The trimming step is repeated until no further points are removed from the cluster. Note that a new representative point $\boldsymbol{u}$ must be calculated after each trimming step. The trimming step may be viewed as removing points which are not likely to belong to a specified cluster.

The trimming step has the following two advantages: 1) it removes any outlier point that might be assigned into a compact cluster, and hence produces a better estimate of the columns of the mixing matrix, 2) it reduces the effect of distributed clusters on identifying the number of sources from the plot of the CP parameters. The

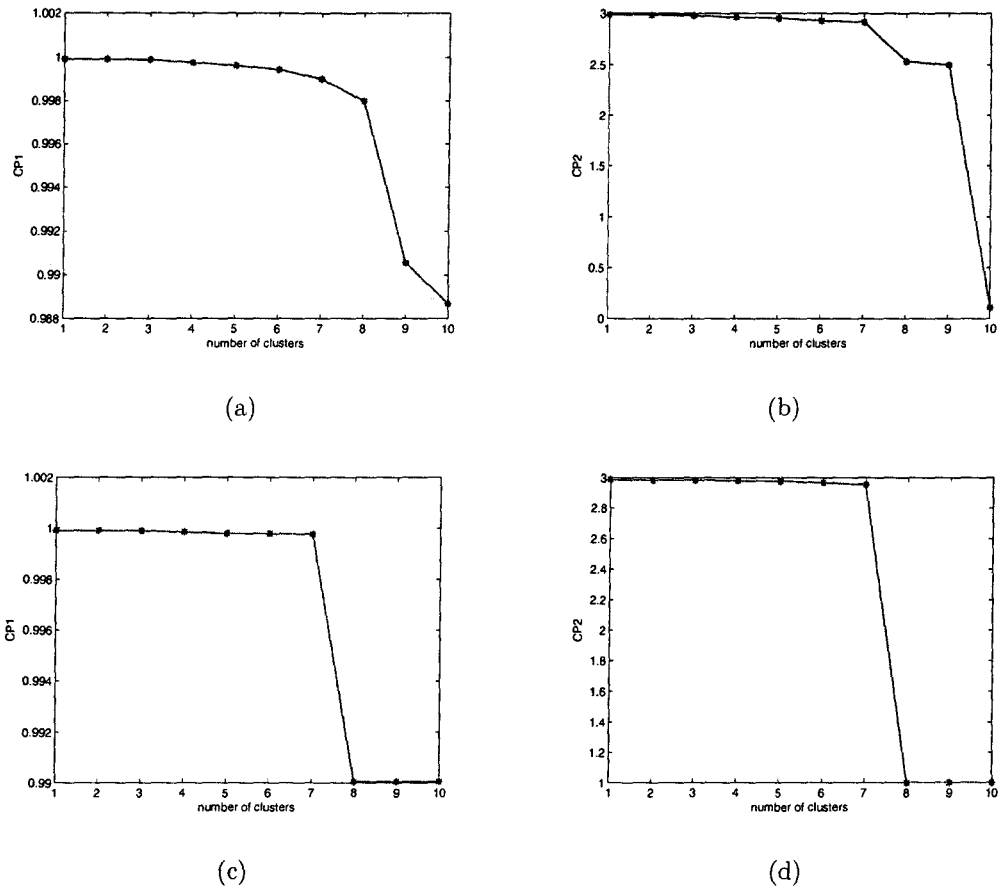
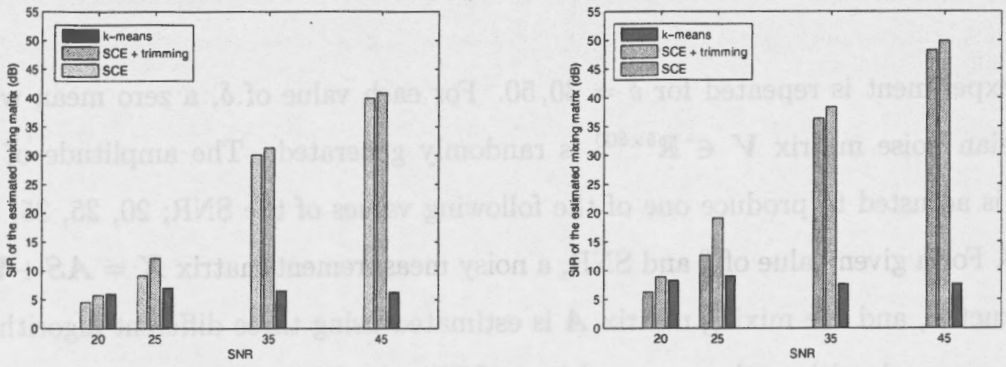



Figure 4.6: Effect of the trimming procedure on the identification of the number of sources from noisy measurements. (a) CP1 before the trimming step, (b) CP2 before the trimming step,(c) CP1 after the trimming step, (d) CP2 after the trimming step.

second point can be explained in light of (4.15). For distributed clusters, the average distance $\hat{\beta}_l$ is usually larger than that for the case of concentrated clusters. However, if the number of points in a distributed cluster is large compared to the number of points in a concentrated cluster, then the value of CP1 defined in (4.15) could be small and comparable to that for a concentrated cluster. As a result, the CP plots may not show a clear transition gap at the index corresponding to the actual number of sources. This behavior is depicted in Figure 4.6(a) and (b) for CP1 and CP2, respectively. The plots of CP1 and CP2 after the trimming step are shown in Figure




(a) $\delta = 30$ (b) $\delta = 50$

Figure 4.7: Comparison between the proposed algorithm and the $k$-means algorithm in terms of the average $\text{SIR}(\boldsymbol{A}, \hat{\boldsymbol{A}})$. The sparse source matrix $\boldsymbol{S}$ is constructed such that $\delta\%$ of its columns satisfy the disjoint orthogonality principal. (a) $\delta = 30$, (b) $\delta = 50$.

4.6(c) and (d), respectively. As shown in these figures, there is a clear transition gap at the number corresponding to the actual number of sources.

**Example:** In this example we compare the performance of the proposed SCE algorithm and the $k$-means algorithm when the measurements are contaminated with additive Gaussian noise. The comparison is made in terms of the signal to interference ratio (SIR) between the true mixing matrix $\boldsymbol{A}$ and the estimated mixing matrix $\hat{\boldsymbol{A}}$, where the SIR is defined as [103]

$$\text{SIR}(\boldsymbol{A}, \hat{\boldsymbol{A}}) = -20 \log_{10} \left( \frac{1}{n} \sum_{i=1}^{n} \min_{j=1,\ldots,n} \frac{\min \{ ||\boldsymbol{a}_i - \hat{\boldsymbol{a}}_j||_{\ell_2}, ||\boldsymbol{a}_i + \hat{\boldsymbol{a}}_j||_{\ell_2} \}}{||\boldsymbol{a}_i||_{\ell_2}} \right) \text{(dB)}. \tag{4.20}$$

The measurements are constructed as follows. A sparse source matrix $\boldsymbol{S} \in \mathbb{R}^{7 \times 800}$ is randomly generated and multiplied by a mixing matrix $\boldsymbol{A} \in \mathbb{R}^{5 \times 7}$ randomly generated from a white normal distribution with zero mean and unit variance. The sparse source matrix is constructed such that $\delta\%$ of its columns satisfy the disjoint orthogonality principal. The indices of the nonzero entries of each row of $\boldsymbol{S}$ are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm 1$.

The experiment is repeated for $\delta = 30, 50$. For each value of $\delta$, a zero mean white Gaussian noise matrix $\boldsymbol{V} \in \mathbb{R}^{5\times 800}$ is randomly generated. The amplitude of the noise is adjusted to produce one of the following values of the SNR; 20, 25, 35, and 45 dB. For a given value of $\delta$ and SNR, a noisy measurement matrix $\boldsymbol{X} = \boldsymbol{AS} + \boldsymbol{V}$ is constructed, and the mixing matrix $\boldsymbol{A}$ is estimated using three different algorithms; the $k$-means algorithm, the proposed basic SCE, and SCE with the trimming step, respectively. The concentration parameter CP2 is used for sorting the clusters estimated using the SCE algorithm. For the three algorithms, the number of clusters was assumed to be known a priori, i.e., the first 7 arranged clusters obtained from the SCE algorithm are considered as an estimate of the mixing matrix. This is because we are comparing to the $k$-means algorithm, which requires the number of sources to be known. After estimating the mixing matrix, the SIR parameter is calculated using (4.20). For each value of $\delta$ and the SNR, the experiment is repeated 100 different times and the average SIR is calculated and plotted in Figure 4.7.

As shown in this figure, and for the two values of $\delta$, the proposed algorithm performs much better than the $k$-means algorithm. Moreover, it is clear from the figure that the trimming step enhances the performance of the proposed algorithm, especially at the moderate SNR value of 25 dB. Also it is obvious from Figure 4.7 that the performance of the proposed algorithm depends on the value of $\delta$. The larger the value of $\delta$, the better the performance of the SCE algorithm.

Although the presented example shows that Algorithm 1 is capable of estimating the mixing matrix with a relatively high precision compared to the $k$-means algorithm, there are two limitations associated with the proposed algorithm. The first limitation is the number of columns of the data matrix $\boldsymbol{Z}$. Recall from the second step in Table 4.3 that the size of the dissimilarity matrix $\boldsymbol{D}$ is $\hat{T} \times \hat{T}$, where $\hat{T}$ is the number of columns of the data matrix $\boldsymbol{Z}$. Since only the upper triangle of $\boldsymbol{D}$ is used for estimating the two closest clusters, the size of $\boldsymbol{D}$ is approximately $\hat{T}^2/2$, i.e., the

size of $\boldsymbol{D}$ grows quadratically with the number of columns of the data matrix $\boldsymbol{Z}$. Accordingly, Algorithm 1 is suitable for clustering data matrices with only a relatively small number of columns. This limitation does not imply that the number of columns of the original data matrix $\boldsymbol{X}$ must be small. This limitation is applied only on the feature matrix $\boldsymbol{Y}$, or equivalently $\boldsymbol{Z}$, which could be constructed from only a few columns of the data matrix $\boldsymbol{X}$.

The second limitation of the proposed algorithm is that the CP parameter may not be sufficiently sensitive when the degree of sparsity is not high enough. If the source coefficient matrix is not sparse enough, then the plot of the CP parameters might not show a clear transition gap at the number corresponding to the true number of sources. However, if the number of clusters $n$ is known a priori, then, with high probability[2], the representative points of the first $n$ sorted clusters correspond to the columns of the mixing matrix.

### 4.3.4 Second Clustering Algorithm: Modified Angular Histogram (MAH)

In this subsection we propose a new algorithm that, under certain conditions, can correctly estimate the mixing matrix and the number of sources. The proposed algorithm is inspired by the angular histogram algorithm [92,95,96] described in Section 4.2.3. The previous algorithm is restricted to 2–D space, and its idea is based on constructing a histogram of the angles corresponding to the orientations of the columns of the data coefficient matrix $\hat{\boldsymbol{X}}$, or equivalently $\boldsymbol{Z}$, in 2–D space. In the proposed algorithm the calculated angles represent the relative angles between the columns of the data matrix $\boldsymbol{Z}$ and a reference vector $\boldsymbol{r}$, i.e.,

$$\theta_k = \cos^{-1}\left(\frac{\boldsymbol{r}^T \boldsymbol{z}_k}{||\boldsymbol{r}||_{\ell_2}||\boldsymbol{z}_k||_{\ell_2}}\right), k = 1, \ldots, \hat{T}. \tag{4.21}$$

[2]This statement is based on empirical observation.

Table 4.4: Modified Angular Histogram Algorithm

Algorithm 2: $[\tilde{\boldsymbol{A}}] = \text{MAH}(\boldsymbol{Z}, \boldsymbol{r}, \Delta\theta)$

1. Calculate the angles between the reference vector $\boldsymbol{r}$ and the columns of $\boldsymbol{Z}$ and save the result in the vector of angles $\boldsymbol{\theta} \in \mathbb{R}^{\hat{T}}$, where $\hat{T}$ is the number of columns of $\boldsymbol{Z}$.

2. Construct the vector $\boldsymbol{\phi} = [0, \Delta\theta, \ldots, 180^o]$, where $\Delta\theta$ is a histogram bin width.

3. Construct the vector $\tilde{\boldsymbol{\theta}}$ by quantizing the values of $\boldsymbol{\theta}$ into the appropriate bin specified in $\boldsymbol{\phi}$. This can be done using the following equation:

$$\tilde{\boldsymbol{\theta}} = \Delta\theta \ \text{round}(\boldsymbol{\theta} / \ \Delta\theta).$$

   where the round$(\cdot)$ function rounds its argument into the nearest integer.

4. For $i = 1, 2, \ldots$, calculate $h[i]$, the frequency of occurrence of $\phi[i]$ in $\tilde{\boldsymbol{\theta}}$.

5. Plot $\boldsymbol{h}$ against the indices corresponding to $\boldsymbol{\phi}$ and count the number of peaks $n_p$ and save the corresponding angles in the vector $\boldsymbol{\theta}_p = [\theta_{p_1}, \ldots, \theta_{p_{np}}]$;

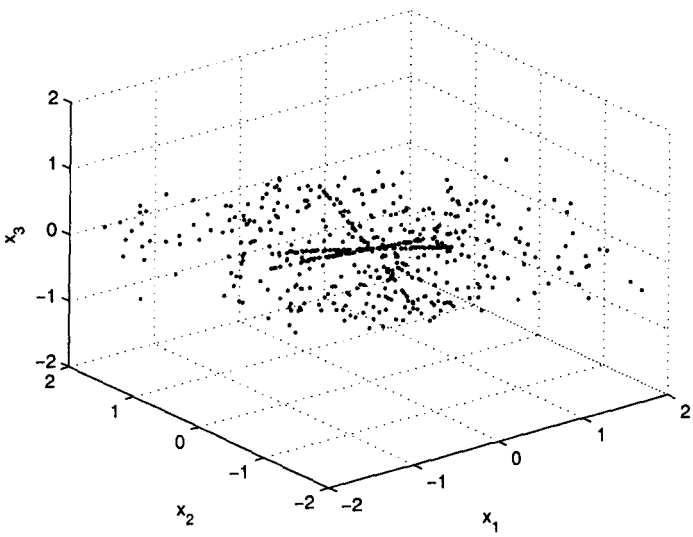
6. Estimate the columns of the mixing matrix using the following steps.

   for $j = 1 : n_p$

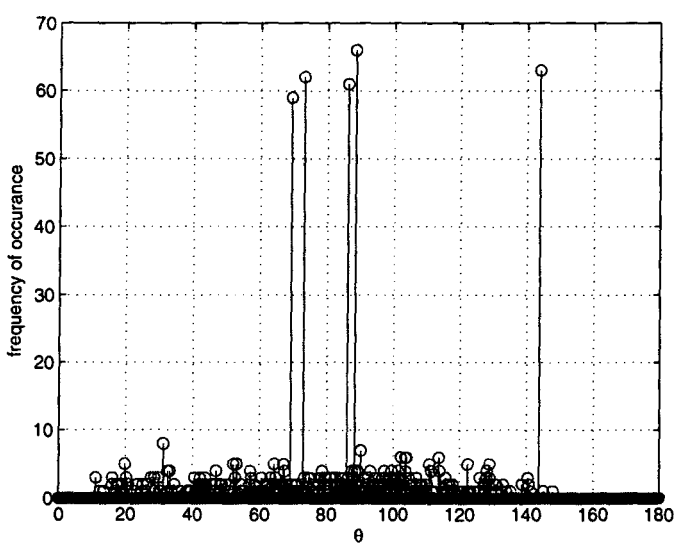
   $\mathcal{I}_j = \text{find}(\tilde{\boldsymbol{\theta}} = \theta_{p_j})$;

   $\tilde{\boldsymbol{a}}_j = \text{mean}(\boldsymbol{Z}(:, \mathcal{I}_j))$;

   end




(a)




(b)

Figure 4.8: (a) A scatter plot of the measured data matrix, (b) The histogram plot of the calculated data.

where $\hat{T}$ is the number of columns of $\boldsymbol{Z}$, and the reference vector $\boldsymbol{r}$ may be chosen arbitrarily.

The proposed algorithm is summarized in Table 4.4 and is described by the following example. Let $\boldsymbol{S} \in \mathbb{R}^{5\times 1000}$ be a sparse source matrix. Assume that the source matrix $\boldsymbol{S}$ is constructed such that 30% of its columns satisfy the disjoint orthogonality property, i.e., each source is uniquely represented by 60 $(= 0.3 * 1000/5)$ columns of the source matrix, while the remaining 70% of the columns are non–sparse and randomly generated from a Gaussian distribution with zero mean and unit variance. Let $\boldsymbol{A} \in \mathbb{R}^{3\times 5}$ denote a mixing matrix randomly generated from a white normal distribution with zero mean and unit variance. For estimating the mixing matrix and the number of columns, a new data matrix $\boldsymbol{Z}$ is constructed from the observation matrix $\boldsymbol{X} = \boldsymbol{AS}$ by following the preparation steps described in Section 4.2.2, with the normalization step performed using (4.8). The reference vector $\boldsymbol{r} \in \mathbb{R}^{3\times 1}$ in this example is selected as the principal eigenvector of the matrix $\boldsymbol{R}_z = \boldsymbol{ZZ}^T$.

The angle between the reference vector $\boldsymbol{r}$ and the $k$th column of $\boldsymbol{Z}$ can be calculated using (4.21). For the columns of the source matrix $\boldsymbol{S}$ that contain only one nonzero element, the corresponding columns of $\boldsymbol{Z}$ are normalized versions of the columns of the mixing matrix $\boldsymbol{A}$. Accordingly, if $\boldsymbol{z}_k = \boldsymbol{a}_i/||\boldsymbol{a}_i||_{\ell_2}$, then $\theta_k$ in (4.21) represents the angle between the reference vector $\boldsymbol{r}$ and $\boldsymbol{a}_i$, the $i$th column of the mixing matrix. Since, in the presented example, each source is uniquely represented by 60 columns in the source matrix $\boldsymbol{S}$, there will be 5 angles, corresponding to the number of sources, that are repeated 60 times in the vector of angles $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{\hat{T}}]$. On the other hand, for the remaining 70% of the columns of the source matrix that have random entries, the corresponding columns of $\boldsymbol{Z}$ are a random linear combination of the columns of $\boldsymbol{A}$. Consequently, the corresponding entries in the vector of angles $\boldsymbol{\theta}$ are random. Accordingly, by constructing a histogram plot for the vector of angles $\boldsymbol{\theta}$, the resulting figure will show 5 peaks corresponding to the 5 angles that are

repeated 60 times. The histogram plot of the presented example is shown in Figure 4.8.

As shown in Figure 4.8(b) there are five peaks in the histogram plot. The height of each peak is approximately 60, which agrees with the number of columns for which each source is uniquely represented in $\boldsymbol{S}$. For constructing the histogram shown in Figure 4.8(b), and for estimating the columns of the mixing matrix, as will be shown later, the vector of angles $\boldsymbol{\theta}$ must be *quantized* into a finite number of values. The value of the quantization level $\Delta\theta$ depends on the noise level. For the noise–free case shown in Figure 4.8(a), all the points corresponding to one hyperline in the $m$ dimensional space have the same orientation. Accordingly, a small value of $\Delta\theta$ should be selected. On the other hand, if the measured signals are contaminated with random noise, the points corresponding to one hyperline in the $m$ dimensional space will distribute around a hyperline, which is oriented in the same direction as one of the columns of the mixing matrix. In this case the value of $\Delta\theta$ must be chosen large enough such that the difference between the angles of the points surrounding one hyperline is less than $\Delta\theta$.

After constructing the vector $\tilde{\boldsymbol{\theta}}$, the quantized version of $\boldsymbol{\theta}$, a histogram plot of $\tilde{\boldsymbol{\theta}}$ is constructed and the number of peaks is counted. If the reference vector $\boldsymbol{r}$ was chosen properly, the number of peaks in the histogram will equal the number of columns of the mixing matrix. Since there is a one–to–one correspondence between the angles in $\tilde{\boldsymbol{\theta}}$ and the columns of $\boldsymbol{Z}$, the columns of the mixing matrix can be estimated by clustering the columns of $\boldsymbol{Z}$ corresponding to the angles that have peaks in the histogram. For example, the first peak in Figure 4.8(b) corresponds to the angle $69^o$. Let $\mathcal{I}_1$ denote the set of indices of the entries in the vector $\tilde{\boldsymbol{\theta}}$ that have that value. Then an estimate of one column of the mixing matrix can be calculated as $\tilde{\boldsymbol{a}}_i = \text{mean}(\boldsymbol{Z}(:, \mathcal{I}_1))$. The process can be repeated for the remaining peaks in the histogram until all the columns of the mixing matrix are estimated. Following this

procedure for the example presented in Figure 4.8, the correlation coefficient between the true mixing matrix and the estimated mixing matrix is given by

$$\boldsymbol{R}_{A\tilde{A}} = \begin{bmatrix} \mathbf{-1.0000} & -0.7046 & 0.5603 & -0.1760 & 0.8120 \\ 0.1760 & -0.5745 & -0.9140 & \mathbf{1.0000} & 0.4316 \\ -0.5603 & 0.1931 & \mathbf{1.0000} & -0.9140 & -0.0285 \\ -0.8135 & -0.9859 & -0.0260 & 0.4294 & \mathbf{1.0000} \\ -0.7046 & \mathbf{-1.0000} & -0.1931 & 0.5745 & 0.9863 \end{bmatrix}$$

Clearly all the columns of the mixing matrix are estimated correctly. Further, the SIR between the actual mixing matrix and the estimated mixing matrix is 73.9128 dB, while the SIR between the actual mixing matrix and the mixing matrix estimated using $k$-means is only 9.4741 dB.

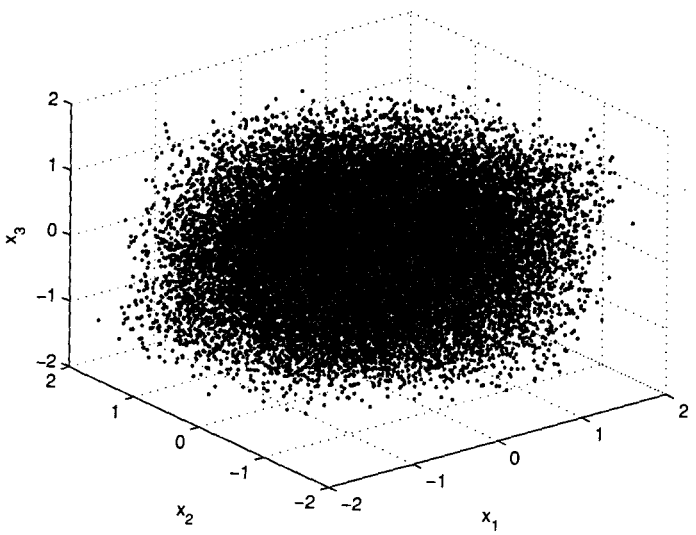
The main difficulty associated with the proposed algorithm is choosing the reference vector $\boldsymbol{r}$. Although the presented example shows that the proposed algorithm can estimate the mixing matrix correctly, this is not always the case. For example, if the reference vector is orthogonal to a hyperplane containing more than one column of the mixing matrix, the estimated angles for these columns will be the same and equal to $90^0$. Accordingly, the number of peaks in the histogram will be less than the number of columns of the mixing matrix, and the estimated column corresponding to the $90^0$ angle will be a linear combination of the vectors that belong to the same hyperplane. Further, if more than one column forms the same angle with respect to the refernce vector $\boldsymbol{r}$, those columns will map into the same histogram bin and the algorithm will fail. The third algorithm presented in the next subsection overcomes this difficulty.

### 4.3.5 Third Clustering Algorithm: Modified Angular Histogram followed by Sequential Cluster Extraction (MAH-SCE)

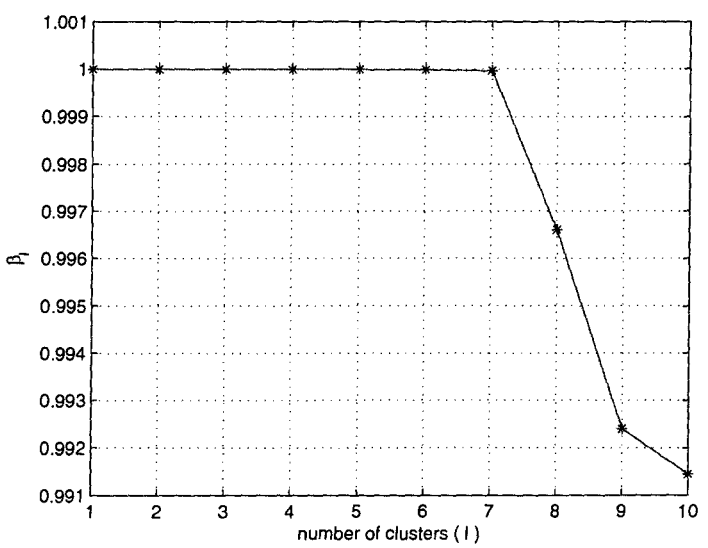
As shown in the previous two subsections, the proposed two algorithms can accurately estimate, under certain conditions, the number of sources and the mixing matrix. However, each one of these two algorithms has its own limitation. For instance, the first algorithm has an undesirable quadratic computational complexity, while the performance of the second algorithm depends greatly on the choice of the reference vector $\boldsymbol{r}$, i.e., the second algorithm may fail distinguishing between distinct multiple columns of $\boldsymbol{A}$.

In this subsection we propose combining the two algorithms into a single robust algorithm that alleviates the difficulties associated with each approach. This can be done by executing the modified angular histogram (MAH) algorithm $i_{max}$ times, where in each execution a new reference vector is used and an estimate of the mixing matrix $\tilde{\boldsymbol{A}}$ is obtained. There are many approaches for selecting the reference vector. For instance, the reference vector can be selected as one of the eigenvectors of $\boldsymbol{R}_z = \boldsymbol{Z}\boldsymbol{Z}^T$. This approach is obvious, since the eigenvectors of $\boldsymbol{R}_z$ capture the main directions of the columns of $\boldsymbol{Z}$. If $i_{max}$ is chosen to be greater than $m$, the number of eigenvectors of $\boldsymbol{R}_z$, then the remaining $(i_{max} - m)$ reference vectors can be selected randomly from, e.g., a Gaussian distribution with zero mean and unit variance.

The estimated columns of the mixing matrix produced by the MAH algorithm may contain some columns which are linear combinations of some columns of the true mixing matrix $\boldsymbol{A}$. This can occur when $\boldsymbol{r}$ has the same orientation with more than one column of $\boldsymbol{A}$. To alleviate this problem, we execute the SCE algorithm, where the input feature matrix $\boldsymbol{Y}$ is constructed by concatenating all the $\tilde{\boldsymbol{A}}$–matrices estimated using the MAH algorithm. The advantage of this approach is that the size




(a)




(b)

Figure 4.9: (a) A scatter plot of the first three rows of the measured data matrix, (b) The CP parameter of the estimated clusters.

of $\boldsymbol{Y}$ can be controlled to a reasonable level by appropriate choice of the parameter $i_{max}$. The steps of the third algorithm is presented in Table 4.5.

The following example demonstrates the ability of the MAHSCE algorithm in estimating the columns of the mixing matrix $\boldsymbol{A}$ in a very difficult situation. The number of sources $n$ in this example is 7 while the number of observations $m$ is 5. The number of columns of the source matrix $\boldsymbol{S}$ is 50,000 and only 2% of them satisfy the disjoint orthogonality principal. The matrix $\boldsymbol{X}$ of observed signals is generated as $\boldsymbol{X} = \boldsymbol{AS}$, where $\boldsymbol{A} \in \mathbb{R}^{5\times 7}$ is randomly generated from a white normal distribution with zero mean and unit variance. A scatter plot of the first 3 measured signals is shown in Figure 4.9(a). As shown in this figure, no clear clusters are visible. However, after running the MAH algorithm 50 different times and constructing a feature matrix for the SCE algorithm, the SCE algorithm estimates the columns of the mixing matrix and the number of sources correctly as shown in Figure 4.9(b). The correlation coefficient between the original mixing matrix and the estimated mixing matrix is given by

$$\boldsymbol{R}_{A\tilde{A}} = \begin{bmatrix} -0.8326 & 0.5500 & 0.0685 & -0.5131 & \mathbf{0.9999} & -0.2768 & 0.8573 \\ -0.6619 & 0.4817 & -0.1630 & \mathbf{-0.9999} & 0.5102 & -0.1253 & 0.1713 \\ 0.1837 & 0.5456 & -0.7397 & -0.1374 & 0.2692 & \mathbf{-1.0000} & -0.0052 \\ \mathbf{0.9999} & -0.4860 & -0.4648 & 0.6557 & -0.8313 & -0.2002 & -0.8077 \\ 0.8084 & -0.5177 & -0.5050 & 0.1765 & -0.8644 & 0.0044 & \mathbf{-0.9999} \\ 0.4573 & -0.0377 & \mathbf{-1.0000} & -0.1644 & -0.0771 & -0.7441 & -0.5128 \\ -0.4885 & \mathbf{0.9999} & 0.0297 & -0.4854 & 0.5458 & -0.5528 & 0.5073 \end{bmatrix}$$

Clearly the mixing matrix is well estimated using the proposed algorithm. The SIR between the actual mixing matrix and the estimated mixing matrix is 38.4434 dB. On the other hand, the correlation matrix between the mixing matrix and the matrix estimated using $k$–means is given by

Table 4.5: Modified Angular Histogram followed by Sequential Cluster Extraction

Algorithm 3: $[\tilde{\boldsymbol{A}}] = \text{MAHSCE}(\boldsymbol{Z}, i_{max}, \Delta\theta)$

1. Compute the eigenvalue decomposition of the matrix $\boldsymbol{R}_z = \boldsymbol{Z}\boldsymbol{Z}^T$; $[\boldsymbol{U}, \boldsymbol{\Sigma}] = \text{eig}(\boldsymbol{R}_z)$.
2. Generate an $(m \times (i_{max} - m))$ random matrix, $\hat{\boldsymbol{R}}$.
3. Generate the matrix of all reference vectors by concatenating the columns of $\boldsymbol{U}$ and$\hat{\boldsymbol{R}}$, i.e., $\boldsymbol{R} = [\boldsymbol{U}\hat{\boldsymbol{R}}]$.
4. Initialize an empty feature matrix $\boldsymbol{Y} = [\,]$.
5. Fill the feature matrix by running the MAH algorithm $i_{max}$ times as follows:

   for $i = 1 : i_{max}$

   $\boldsymbol{r}_i = \boldsymbol{R}(:, i)$;

   $\boldsymbol{B}_i = \text{MAH}(\boldsymbol{Z}, \boldsymbol{r}_i, \Delta\theta)$;

   $\boldsymbol{Y} = [\boldsymbol{Y}\boldsymbol{B}_i]$;

6. end
7. Estimate the mixing matrix $\tilde{\boldsymbol{A}}$ using Algorithm 1, with the constructed matrix $\boldsymbol{Y}$ as the input feature matrix.

$$\boldsymbol{R}_{A\tilde{A}_{km}} = \begin{bmatrix} -0.2028 & -0.6658 & 0.5471 & 0.0263 & -0.1851 & \mathbf{0.9524} & -0.0445 \\ 0.7315 & -0.6687 & -0.4542 & 0.1498 & -0.8208 & 0.1566 & \mathbf{-0.9710} \\ \mathbf{-0.9700} & 0.4783 & 0.6514 & -0.5142 & 0.7398 & 0.3195 & 0.8386 \\ -0.6054 & 0.2139 & -0.2033 & \mathbf{-0.9551} & 0.4465 & 0.0177 & 0.0720 \\ 0.7178 & -0.7953 & 0.1838 & 0.7412 & -0.8699 & 0.5393 & -0.6104 \\ 0.0096 & 0.1623 & 0.4621 & -0.1547 & -0.4937 & 0.4400 & -0.2873 \\ -0.2973 & -0.2270 & \mathbf{0.9316} & 0.4225 & 0.0338 & 0.7633 & 0.4846 \end{bmatrix}$$

which shows that only 5 columns are estimated with moderate accuracy, while the remaining ones are poorly estimated. The SIR between the actual mixing matrix and the estimated mixing matrix is only 5.0299 dB. This example reflects the advantage of the proposed algorithm over the partitioning clustering algorithms, which are represented by the $k$–means algorithm in this example. Although the number of clusters was provided to the $k$–means algorithm, the algorithm failed in estimating the columns of the mixing matrix, while the MAHSCE algorithm estimated the number of sources and the mixing matrix with very high accuracy. The reason behind this performance is that the MAHSCE algorithm is selective in the sense that only the points that show concentration around a hyperline are taken into consideration, while the $k$–means algorithm treats all the points equally.

## 4.4 Simulation Results

In this section we present two examples of separating real signals. In the first example the original signals are 5 different images, while in the second example the original signals are 4 different sounds. In both cases, the wavelet transform is applied on the measured signals. The mixing matrix and the wavelet coefficients of the sources are estimated in the wavelet domain. Then the inverse wavelet transform is applied on the wavelet coefficients of the sources to restore the original sources to their original

space of representation, i.e., the time domain for the sound signals and the spatial domain for the images.

### Example 1

In this example, the performance of the MAHSCE algorithm is compared with two different algorithms, the $k$–means clustering algorithm and the FastICA algorithm [52]. The ICA algorithm is incorporated in this comparison to demonstrate the ability of the SCA technique in solving some problems for which the ICA technique fails. In this example we present a case for which some of the original source signals are correlated. Accordingly, the sources violate the fundamental independence assumption, and consequently it is expected that the FastICA algorithm will fail in this case. Since the ICA algorithms assume that the number of sensors is at least equal to the number of sources, a square mixing matrix is used in this example.

The original sources in this example are the five photos shown in Figure 4.10(a). The correlation matrix of the five pictures is given by

$$\boldsymbol{R}_{SS} = \begin{bmatrix} 1.0000 & -0.1944 & 0.1956 & -0.3230 & -0.4664 \\ -0.1944 & 1.0000 & -0.1715 & 0.3154 & 0.0285 \\ 0.1956 & -0.1715 & 1.0000 & -0.4231 & 0.1963 \\ -0.3230 & 0.3154 & -0.4231 & 1.0000 & -0.0374 \\ -0.4664 & 0.0285 & 0.1963 & -0.0374 & 1.0000 \end{bmatrix}$$

which shows that some of the sources are correlated.

The five mixtures are shown in Figure 4.10(b), and the estimated sources using the FastICA algorithm are shown in Figure 4.10(c). It is clear from this figure that the FastICA algorithm failed in estimating the sources correctly, and none of the estimated sources correspond to the third original source.

On the other hand, estimating the sources using the SCA technique is performed by estimating the mixing matrix using two different clustering algorithms; the MAHSCE and the $k$–means algorithms. The 2–D "sym4" discrete wavelet transform is

(a)

(b)

(c)

(d)

(e)

Figure 4.10: Results of Example 1. (a) Original photos, (b) Mixed photos, (c) Estimated photos using FastICA algorithm, (d) and (e) Estimated photos when the mixing matrix is estimated using $k$–means and MAHSCE, respectively.

applied on the mixtures and each algorithm is applied separately on the resulting wavelet coefficients to estimate the mixing matrix. For each of the estimated mixing matrices, the corresponding wavelet coefficients of the sources are obtained by pre–multiplying the matrix of the mixtures' wavelet coefficients by the inverse of the estimated mixing matrix. The pictures are then reconstructed by applying the 2–D inverse wavelet transform on the resulting coefficients. Figure 4.10(d)–(e) show the reconstructed pictures when the mixing matrix is estimated using $k$–means and MAHSCE, respectively. As shown in these two figures, the sources are estimated correctly only when the mixing matrix is estimated using the MAHSCE algorithm.

**Example 2**

In this example we consider the case of separating 4 different sound sources from *only* 3 mixtures. Since the number of mixtures is less than the number of sources, the sources can not be estimated using ICA. In this example, the performance of the MAHSCE algorithm is compared with that of the $k$–means. The observed signals are sparsely represented using the sym8 discrete wavelet transform, and the decomposition coefficients are used for estimating the mixing matrix. After estimating the mixing matrix, the coefficients of the sources are estimated using the MCCR algorithm presented in Chapter 3. The objective function used with the MCCR algorithm is the $g_{log}(s)$. The sources are reconstructed from their coefficients by applying the inverse wavelet transform. The results of this example are shown in Figure 4.11.

As shown in Figure 4.11(c), the original sources are estimated correctly when the mixing matrix is estimated using the MAHSCE algorithm. The correlation matrix

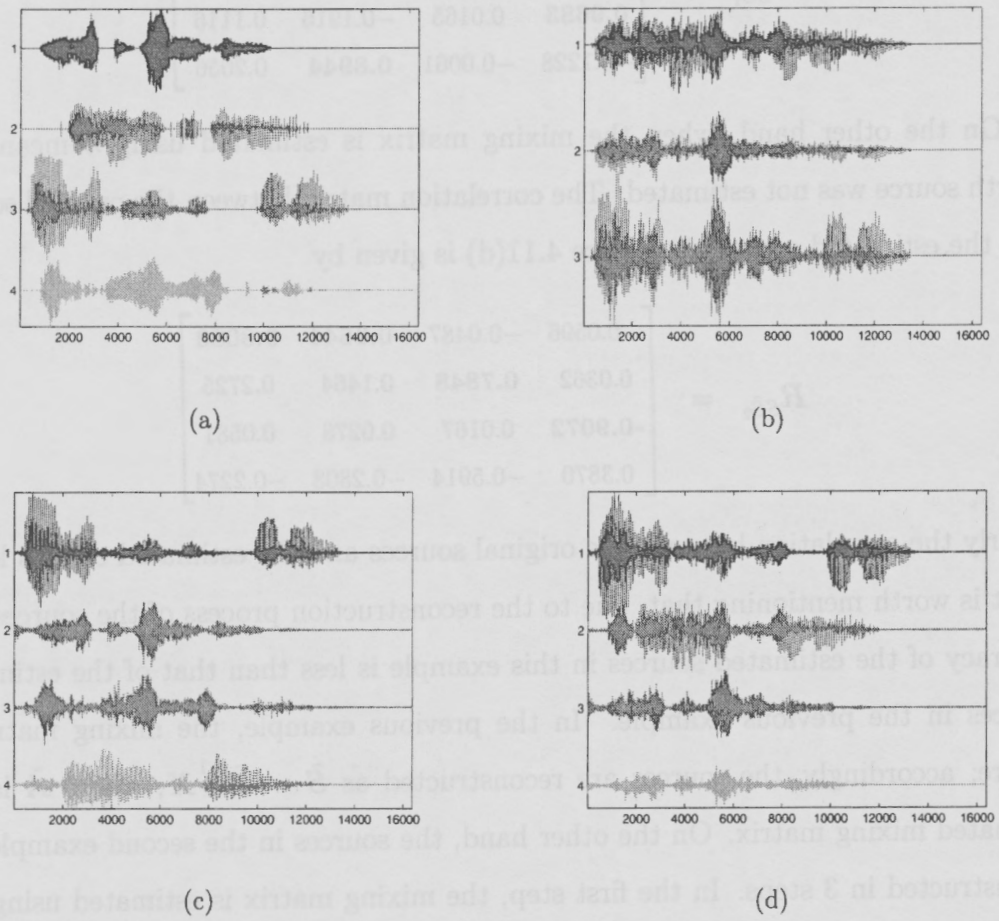



Figure 4.11: Results of Example 2. (a) Original source signals, (b) Mixed signals, (c) and (d) Estimated sources when the mixing matrix is estimated using the MAHSCE and the k-means, respectively. .

between the original sources and the estimated sources in Figure 4.11(c) is given by

$$\boldsymbol{R}_{S\tilde{S}_1} = \begin{bmatrix} 0.0077 & \mathbf{0.9346} & -0.0040 & -0.0216 \\ -0.0619 & 0.0254 & -0.2048 & \mathbf{-0.9291} \\ \mathbf{0.9683} & 0.0165 & -0.1916 & 0.1116 \\ -0.1228 & -0.0061 & \mathbf{0.8944} & 0.2056 \end{bmatrix}$$

On the other hand, when the mixing matrix is estimated using $k$–means, the fourth source was not estimated. The correlation matrix between the original sources and the estimated sources in Figure 4.11(d) is given by

$$\boldsymbol{R}_{S\tilde{S}_2} = \begin{bmatrix} -0.0596 & -0.0487 & \mathbf{-0.8644} & \mathbf{0.8080} \\ 0.0362 & \mathbf{0.7848} & 0.1464 & 0.2725 \\ \mathbf{-0.9072} & 0.0167 & 0.0278 & 0.0581 \\ 0.3870 & -0.5914 & -0.2808 & -0.2274 \end{bmatrix}$$

Clearly the correlation between the original sources and the estimated ones is low.

It is worth mentioning that, due to the reconstruction process of the sources, the accuracy of the estimated sources in this example is less than that of the estimated sources in the previous example. In the previous example, the mixing matrix is square; accordingly, the sources are reconstructed as $\tilde{\boldsymbol{S}} = \tilde{\boldsymbol{A}}^{-1}\boldsymbol{X}$, where $\tilde{\boldsymbol{A}}$ is the estimated mixing matrix. On the other hand, the sources in the second example are reconstructed in 3 steps. In the first step, the mixing matrix is estimated using the proposed MAHSCE clustering algorithm. In the second step, the estimated mixing matrix and the wavelet coefficients of the measured sounds are utilized for estimating the wavelet coefficient of the sources using the MCCR algorithm proposed in Chapter 3. Finally, the sources are estimated by performing the inverse wavelet transform on the estimated wavelet coefficients of the sources. Recall from Chapter 3 that there is a certain threshold for the diversity (number of nonzero entries) of the source vector below which the original source vector can be perfectly estimated using the MCCR algorithm. Therefore, even if the mixing matrix can be estimated correctly using the

MAHSCE algorithm, the wavelet coefficients of the sources may not be estimated correctly using the MCCR algorithm. This can happen when many columns of the sources coefficient matrix are not sparse enough for the MCCR algorithm, or any other algorithm, to estimate them correctly. Therefore, although the SCA technique is very promising for solving the under–determined BSS problem, the most critical step in the two–step technique is finding a proper transformation that projects the original sources into its optimal sparse representation. This is the topic of the future research.

## 4.5 Conclusion

In this chapter we proposed three different clustering algorithms that can be used for estimating the mixing matrix for solving the blind source separation (BSS) problem via the SCA technique. In contrast to most of the previously utilized algorithms, the proposed algorithms can estimate the number of sources from the observed signals. The first clustering algorithm is based on a clustering technique called hierarchical clustering. Identification of individual clusters constructed using hierarchical clustering is a difficult task. Previous approaches for identifying individual clusters are either manual or depend on some parameters that are difficult to adjust. In the first clustering algorithm we proposed a new technique for identifying these clusters. This technique is based on incorporating a statistical test with the hierarchical clustering algorithm. In addition, we proposed two parameters, called concentration parameters, that can identify from the extracted clusters the clusters that correspond to the columns of mixing matrix. Accordingly, the proposed algorithm can estimate the mixing matrix as well as the number of sources.

The idea of the second proposed clustering algorithm is inspired by the angular histogram clustering algorithm [92, 95, 96]. Previously suggested angular histogram

clustering algorithms are difficult to extend beyond the 2-D case. However, in this chapter we proposed a new algorithm that generalizes the previous algorithms to the $m \geq 2$ case. It was shown that, under certain conditions, the proposed algorithm can efficiently estimate the mixing matrix even when the disjoint orthogonality condition is satisfied for a small percentage of the columns of the source coefficient matrix.

The third clustering algorithm is a combination of the first two algorithms. This algorithm combines the advantages of the first two clustering algorithms and avoids their limitations. Two real examples demonstrated the ability of the proposed algorithms in estimating the mixing matrix and the number of sources.

# Chapter 5

# Sequential Sparse Signal Extraction with Applications to EEG/MEG Signal Processing

## 5.1 Introduction

In Chapter 4 we discussed the problem of finding a unique solution to the instantaneous blind source separation (BSS) problem defined as $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$ when both $\boldsymbol{A}$ and $\boldsymbol{S}$ are *unknowns*, where $\boldsymbol{X} \in \mathbb{R}^{m \times T}$ is a matrix of measured signals, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is an unknown mixing matrix, $\boldsymbol{S} \in \mathbb{R}^{n \times T}$ is a matrix of unknown sources, $m$ is the number of observations, $n$ is the number of sources, and $T$ is the number of samples. The algorithms derived in Chapter 4 are more suitable for solving the underdetermined BSS in which the number of sources $n$ is greater than the number of sensors $m$. In this chapter we develop new algorithms for solving the BSS problem when $m \geq n$.

As described in Chapter 4, the independent component analysis (ICA) technique can be utilized for solving the BSS problem as long as the underlying problem satisfies

the four conditions mentioned in Section 4.1. Since the over–determined case, in which $m > n$ can always be transformed into a complete case by using the principal component analysis (PCA) technique, it is assumed in this chapter, without loss of generality, that $m = n$, i.e., the number of sensors equals the number of sources.

Generally speaking, there are three different approaches that have been suggested in the literature for solving the BSS problem using the ICA technique. The first approach is to separate all the sources simultaneously. This can be accomplished by finding a separating matrix $\boldsymbol{B}$ such that the rows of $\boldsymbol{Y} = \boldsymbol{BX} = \boldsymbol{BAS}$ are mutually independent. The ideal value of $\boldsymbol{B}$ is $\boldsymbol{B} = \boldsymbol{A}^{-1}$. However, the solution of the BSS problem is always associated with permutation and scale indeterminacies. Therefore, the best solution that one can expect from any algorithm that solves the BSS problem is the one that satisfies $\boldsymbol{BA} = \boldsymbol{PF}$, where $\boldsymbol{F}$ is a diagonal scaling matrix and $\boldsymbol{P}$ is a permutation matrix. This approach has been utilized successfully for removing different kinds of artifacts from EEG/MEG data [111–113]. However, when the number of sensors is large, as in MEG measurements, the process of identifying the source of interest, e.g. the one corresponding to the artifact, from the separated sources is generally not trivial and needs some experience. Accordingly, in the case when only a very few source signals are the subject of interest, the computational cost as well as the post processing effort required for identifying the desired sources can be greatly reduced by extracting only the sources that have certain characteristics. This is the second approach for solving the BSS problem, and it is known in the literature as blind signal extraction (BSE).

BSE has become one of the promising techniques for solving the BSS problem, especially in the cases when only few sources with specific stochastic properties or features need to be extracted. The idea of existing BSE methods is to find a separating vector $\boldsymbol{b}$ that maximizes the non–Gaussianity of the separated signal $\bar{\boldsymbol{y}} = \boldsymbol{b}^T\boldsymbol{X}$, where $\bar{\boldsymbol{y}}$ is a *row* vector corresponding to one of the desired sources [52,53,55]. The extracted

source depends on the objective function that measures the non-Gaussianity of the separated source. For example, if the kurtosis is used as a measure of non-Gaussianity, the separated source could be either super–Gaussian or sub–Gaussian depending on whether the kurtosis is maximized or minimized, respectively. The extracted source is then removed from the mixture, and the process can be repeated until all the desired sources are extracted.

As stated in [55], the BSE approach has the following advantages over the simultaneous BSS approach; 1) signals can be extracted in a specific order according to some features of the source signals, 2) the approach is very flexible in the sense that various criteria can be used in each stage depending on the features of the source to be extracted, 3) only interesting source signals need to be extracted, and 4) extensive computing time and resources can be saved.

If one were to maximize the kurtosis, for instance, for extracting a single source, the algorithm would (ideally) converge to the single source having the maximum kurtosis of all the underlying sources. However, when one desires a *specific* source, then the BSE approach is of little use, unless the source carries the maximum kurtosis. Furthermore, due to random initialization of the algorithm, the algorithm is not guaranteed to converge to the global maximum [114]. To overcome this difficulty, prior knowledge about the desired source can be incorporated as additional constraints into the algorithm that solve the BSE problem. Following this technique, a third approach, known as constrained ICA (cICA), is developed in the literature for solving the BSS problem [114–119].

The cICA technique was first proposed in [115], and it was based on using a temporal constraint to extract a source signal which is statistically independent from the other sources and is closest to some reference signal $r(t)$. The technique is then developed in [116] to handle the case of utilizing multiple source signals for simultaneous extraction of multiple independent components. Further development was

then suggested in [118, 120, 121] for the case of a spatial constraint. In all these cases, the reference signal need not to be a perfect match (indeed, if it were, there would be no reason to perform the analysis) but it should have enough information to point the algorithm in the direction of the desired signal. For the case of imposing a temporal constraint [115, 116], the desired source signal is estimated by solving an optimization problem in which the negentropy was used as an objective function for measuring the non–Gaussianity of the extracted source, and the measure of closeness between the desired source signal and the reference signal was used as a constraint. The constrained optimization problem is then solved using a Newton-like learning procedure. Using the prior information provided by the reference signals, the cICA algorithm will then be able to extract only the useful components, thus eliminating the need for post-processing of the separated components, and reducing the amount of computational resources and costs.

In the field of biomedical signal analysis it is common that some prior information is available about the temporal or spatial characteristics of the desired signal. The prior information can then be used for constructing a reference signal, which in turn can be used as a temporal constraint in the cICA algorithm. For example, for the case of rejecting ocular artifacts from an EEG/MEG data, the ocular artifact most likely contaminates the frontal electrodes. Accordingly, a reference signal in this case can be easily derived from one of the frontal electrodes as the time samples that exceed a certain threshold [114, 117, 122]. Reference signals for other artifacts can be obtained by using the information available about these artifacts. For instance, the frequency of the noise introduced by the main supply (50/60 Hz) is known, and a reference signal for the electrocardiogram (ECG) contamination can be obtained by using a co-recording from a separate electrode placed on the chest. In each of these cases, a reference signal can quite readily be automatically derived to serve as a temporal constraint in the cICA algorithm [114]. Another approach for generating a reference

signal for extracting periodic signals, e.g., the event related potential (ERP) signal, is to use a sequence of pulses which have the same periodic frequency as the desired signal. This approach was utilized in [116] for estimating the different components of the ERP signal from real EEG data.

From this discussion it is clear that sequential source extraction techniques, especially cICA, are more useful than the simultaneous BSS technique in the field of biomedical signal processing, where only a few source signals are considered interesting. In this chapter we propose four novel algorithms for solving the BSE problem when the source signals are *sparse*. Since non–sparse sources can often be sparsely represented under a suitable linear transformation, (e.g., the short time Fourier transform, the wavelet transform, the wavelet packets transform,... etc.), it will be assumed in this chapter, without loss of generality, that all the desired sources are sparse. Instead of maximizing the non–Gaussianity of the extracted source signal, as in the ICA approach, the objective function used in these algorithms measures the sparsity of the extracted source signal, and a separating vector is estimated as the one that maximizes the sparsity of the extracted source signal.

In the first part of this chapter we propose a novel algorithm for solving the problem of sequential blind source extraction when the sources of interest are *sparse*. The ICA based algorithms that can extract sparse sources, e.g., the FastICA algorithm [52], are based on modeling the sparse source signal as a super–Gaussian signal. Hence, a separating vector is estimated by *maximizing* the kurtosis of the extracted source signal. However, in the proposed algorithm we suggest using an objective function that explicitly measures the *diversity* (antisparsity) of the extracted source signal. The simulation results presented in Section 5.4 show that the proposed algorithm can extract sparse source signals with relatively smaller residual error than the FastICA algorithm. The proposed algorithm can extract only a single sparse source signal, and it is blind due to the lack of information available about the desired source. After

running the algorithm, the algorithm would (ideally) converge to the sparsest source signal. Then the extracted source is removed from the mixture, and the process is repeated until all the desired sources are extracted. The algorithm is referred to as *blind sparse signal extraction* (BSSE).

In the second part of this chapter we propose three different novel algorithms that explicitly use the information available about the source of interest. The available information is used as a constraint in the optimization problem, and the resulting algorithms are referred to as *constrained sparse component analysis* (cSCA). The first algorithm in this class is based on utilizing the temporal information available about the source signal. A previous approach for solving this problem was proposed in [123] and is based on minimizing an objective function, which measures the sparsity of the extracted source signal, under the constraint that the correlation coefficient between the extracted source signal and a reference signal equal to one. The objective function used in [123] is a smooth approximation of the $\ell_1$–norm. However, in Chapter 2 and Chapter 3 it was shown that nonconvex objective functions perform better than the $\ell_1$–norm in estimating sparse signals. Therefore, the proposed algorithm is based on measuring the sparsity of the extracted source signal using one of the objective functions proposed in Chapter 3. The proposed algorithm has two different versions, depending on the measure of closeness between the extracted source signal and the reference signal.

The second algorithm is also based on utilizing temporal information about the desired source signal. However, in contrast to the first algorithm, only the support of the desired sparse source signal is available, i.e., the case where only the indices of the nonzero samples of the desired source signal are known while no information is available about their signs and/or values. Finally, the third algorithm is based on utilizing spatial information, i.e., information about the mixing column, of the source of interest. Previous approaches for solving this problem were proposed in [118,120,121].

These approaches are based on modifying an ICA based algorithm in a way that some of the columns of the mixing matrix are initialized by the reference vectors, which are either kept fixed or allowed to change slightly during the optimization process. In these algorithms, all components must be estimated first, then the desired signals are identified as the components associated with the reference mixing columns. In contrast to these algorithms, the proposed algorithm can extract only one *sparse* source signal which corresponds to the available spatial reference vector. Simulation results show that the three algorithms can be successfully used for removing different kinds of artifacts from real EEG data and for estimating the ERP signal from synthesized EEG data.

| **List of symbols** | |
|---|---|
| $\boldsymbol{A}$ | The mixing matrix. |
| $\boldsymbol{B}$ | The separating matrix. |
| $\boldsymbol{S}$ | The source matrix. |
| $\boldsymbol{X}$ | The measurement matrix. |
| $\boldsymbol{Y}$ | The estimated source matrix. |
| $\boldsymbol{W}_0$ | A diagonal weighting matrix. |
| $\boldsymbol{h}$ | An estimate of a mixing column. |
| $\boldsymbol{r}$ | A reference vector. |
| $\bar{\boldsymbol{r}}_t$ | A temporal reference (row) vector. |
| $\bar{\boldsymbol{r}}_{su}$ | A support reference (row) vector. |
| $\boldsymbol{r}_s$ | A spatial reference (column) vector. |
| $\text{eig}_{min}(\boldsymbol{R})$ | The eigenvector of $\boldsymbol{R}$ corresponding to the minimum eigenvalue. |
| $\mathcal{J}_r\left(\bar{\boldsymbol{b}};\boldsymbol{r}\right)$ | The regularization objective function. |
| $\gamma$ | The regularization parameter. |

## 5.2 Blind Sparse Signal Extraction (BSSE)

As described in Chapter 4, the BSS problem is defined as the problem of retrieving $n$ unknown source signals $\boldsymbol{s}(t) \in \mathbb{R}^n$ from $m$ linear measurements $\boldsymbol{x}(t) \in \mathbb{R}^m$ when the mixing matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is unknown:

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t), \quad t = 1, \ldots T. \tag{5.1}$$

here $m$ is the number of observations, $n$ is the number of sources, and $T$ is the number of samples. Eq. (5.1) can be written in the following compact form

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}, \tag{5.2}$$

where $\boldsymbol{X} \in \mathbb{R}^{m \times T}$ is a matrix of observed signals, $\boldsymbol{S} \in \mathbb{R}^{n \times T}$ is a matrix of unknown sources.

In Chapter 4 we considered the case of solving an under–determined BSS problem in which the number of sources $n$ is generally unknown and greater than the number of sensors $m$, and all the sources are estimated simultaneously. In this chapter we consider the even–determined BSS problem in which the number of sources equals the number of sensors, and we solve this problem via sequential extraction of the sources.

In the case of the even–determined BSS problem, estimates of the sources are obtained by finding a separating matrix $\boldsymbol{B}$ such that the estimated sources

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{X} = \boldsymbol{B}\boldsymbol{A}\boldsymbol{S} \tag{5.3}$$

satisfy some predefined statistical properties. For example, in the ICA approach, the separating matrix is estimated such that the rows of $\boldsymbol{Y}$ are mutually independent, while in the PCA approach, the separating matrix is estimated such that the rows of $\boldsymbol{Y}$ are uncorrelated. Since in the underlying model we assume that the sources are sparse, the separating matrix in this chapter is estimated such that the rows of $\boldsymbol{Y}$ are

as sparse as possible. Regardless of the utilized approach, the sources are considered to be correctly estimated if the overall separating–mixing system satisfies

$$\boldsymbol{C} = \boldsymbol{BA} = \boldsymbol{PF}, \tag{5.4}$$

where $\boldsymbol{F}$ is a diagonal scaling matrix, and $\boldsymbol{P}$ is a permutation matrix, i.e., the estimated sources are usually associated with scale and permutation indeterminacies.

### 5.2.1 Extraction of a sparse source.

In this section we consider a novel solution to the problem of sequential estimation of the rows of the separating matrix $\boldsymbol{B}$. Each row of the separating matrix $\boldsymbol{B}$ is estimated by maximizing the sparsity of the associated extracted source. From (5.3), the $i$th row $\boldsymbol{y}^i \in \mathbb{R}^T$ of $\boldsymbol{Y}$, corresponding to the $i$th separated source, is expressed as

$$\boldsymbol{y}^i = \boldsymbol{b}^i\boldsymbol{X} = \boldsymbol{c}^i\boldsymbol{S} = \sum_{j=1}^{n} c^i[j]\boldsymbol{s}^j, \tag{5.5}$$

where $\boldsymbol{b}^i$ is the $i$th row of $\boldsymbol{B}$, $\boldsymbol{c}^i = \boldsymbol{b}^i\boldsymbol{A}$ is the $i$th row of $\boldsymbol{C}$, and $\boldsymbol{s}^j$ is the $j$th source signal, i.e., the $j$th row of $\boldsymbol{S}$. Since the order of the estimated source is irrelevant, the superscript $^i$ will be dropped from the sequel equations, and we use the notation $(\bar{})$ to refer to a row vector. Accordingly (5.5) can be rewritten as

$$\bar{\boldsymbol{y}} = \bar{\boldsymbol{b}}\boldsymbol{X} = \bar{\boldsymbol{c}}\boldsymbol{S} = \sum_{j=1}^{n} c[j]\boldsymbol{s}^j, \tag{5.6}$$

which shows that the estimated source signal is a linear combination of the original source signals.

Assume that the support of the original source signals are independent. The support of the signal refers to the indices of the nonzero samples. Then, with high probability, the *diversity* (the number of nonzero samples) of the summation of two source signals is greater than the diversity of any one of the two sources. Accordingly,

the sparsest estimated signal $\bar{\boldsymbol{y}}$ can be obtained if $\bar{\boldsymbol{c}}$ has only one nonzero entry. Since $\bar{\boldsymbol{c}} = \bar{\boldsymbol{b}}\boldsymbol{A}$, this implies that $\bar{\boldsymbol{b}}$ corresponds to a scaled version of one row of $\boldsymbol{A}^{-1}$, the inverse of the mixing matrix. Accordingly, one sparse source signal can be obtained by finding a separating vector $\bar{\boldsymbol{b}}$ that minimizes the diversity of the estimated source vector $\bar{\boldsymbol{y}} = \bar{\boldsymbol{b}}\boldsymbol{X}$.

### Analytic Solution

Consider the case where the set of indices of the zero entries of the desired source, say the $i$th source, is known. Let $\mathcal{I}_i$ refer to these indices, i.e., $\boldsymbol{s}^i(\mathcal{I}_i) = 0$. Due to the assumption that the support of the original source signals are independent, then, with high probability, there are some indices for which $\boldsymbol{s}^j(\mathcal{I}_i) \neq 0, \forall j \neq i$, i.e., $||\boldsymbol{s}^j(\mathcal{I}_i)||_{\ell_2} > 0$. Therefore, if $\boldsymbol{B}$ is a separating matrix, and neglecting the permutation indeterminacy, the entries of the $i$th row of the matrix

$$\boldsymbol{Y}_{\mathcal{I}_i} = \boldsymbol{B}\boldsymbol{X}_{\mathcal{I}_i} \tag{5.7}$$

are all zeros, where $\boldsymbol{X}_{\mathcal{I}_i}$ is a sub–matrix of $\boldsymbol{X}$ with columns corresponding to the indices in $\mathcal{I}_i$. Accordingly, the $i$th row of the separating matrix $\boldsymbol{B}$ can be obtained by solving the following optimization problem

$$\tilde{\boldsymbol{b}}^i = \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_{\bar{b}} \triangleq ||\bar{\boldsymbol{b}}\boldsymbol{X}_{\mathcal{I}_i}||_{\ell_2}^2 \quad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1, \tag{5.8}$$

where the constraint $||\bar{\boldsymbol{b}}||_{\ell_2} = 1$ is utilized to prevent the trivial solution $\bar{\boldsymbol{b}} = 0$.

Following the standard method of Lagrangian multipliers (see, e.g. [85, 86]), the optimization problem (5.8) can be readily solved with a solution vector satisfying

$$\boldsymbol{X}_{\mathcal{I}_i}\boldsymbol{X}_{\mathcal{I}_i}^T\bar{\boldsymbol{b}}^T = \lambda\bar{\boldsymbol{b}}^T, \tag{5.9}$$

where $\lambda$ is the Lagrangian multiplier. Equation (5.9) shows that $\lambda$ and $\bar{\boldsymbol{b}}^T$ are eigenvalue and eigenvector of the matrix $\boldsymbol{X}_{\mathcal{I}_i}\boldsymbol{X}_{\mathcal{I}_i}^T$, respectively. Moreover, substituting

(5.9) into the expression of the objective function in (5.8) we get

$$\mathcal{J}_{\bar{b}} = ||\bar{\boldsymbol{b}}\boldsymbol{X}_{\mathcal{I}_i}||_{\ell_2}^2 = \bar{\boldsymbol{b}}\boldsymbol{X}_{\mathcal{I}_i}\boldsymbol{X}_{\mathcal{I}_i}^T\bar{\boldsymbol{b}}^T = \lambda\bar{\boldsymbol{b}}\bar{\boldsymbol{b}}^T = \lambda, \tag{5.10}$$

where in the last equality we used the fact that $||\bar{\boldsymbol{b}}||_{\ell_2} = 1$. Therefore, for minimizing the cost function $\mathcal{J}_{\bar{b}}$, $\lambda$ must be minimum. Accordingly, the solution of the optimization problem (5.8) is given by

$$\left(\tilde{\boldsymbol{b}}^i\right)^T = \text{eig}_{min}(\boldsymbol{X}_{\mathcal{I}_i}\boldsymbol{X}_{\mathcal{I}_i}^T), \tag{5.11}$$

where $\text{eig}_{min}(\boldsymbol{R})$ is the eigenvector of $\boldsymbol{R}$ corresponding to the minimum eigenvalue.

Equation (5.11) states that the separating vector of the $i$th source can be readily estimated from the measured signals once the indices at which the $i$th source equals zero are known. However, this is of little use, since these indices are generally unknown. We now propose an iterative algorithm that converges in its limit to (5.11).

### Proposed iterative algorithm

Since the prior information available about the desired source signal is its sparsity, the separating vector $\bar{\boldsymbol{b}}$ can be estimated by minimizing the following objective function

$$\tilde{\bar{\boldsymbol{b}}} = \arg\min_{\bar{\boldsymbol{b}}} g\left(\bar{\boldsymbol{y}}\right) = g(\bar{\boldsymbol{b}}\boldsymbol{X}) \quad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1, \tag{5.12}$$

where $g(\bar{\boldsymbol{y}})$ is a function that measures the diversity (antisparsity) of the separated vector $\bar{\boldsymbol{y}}$. In this chapter we consider the functions presented in Chapter 3, which have the general form

$$g(\bar{\boldsymbol{y}}) = \sum_{t=1}^{T} g_c(y[t]), \tag{5.13}$$

where $g_c(\cdot)$ is a symmetric and monotonically increasing concave function on the nonnegative orthant $\mathcal{O}_1$. See Table 3.1 for a list of objective functions that satisfy these conditions.

Given an estimate of the separating vector $\bar{\boldsymbol{b}}_k$ at the $k$th iteration, it was proved in Chapter 3 that the function $g(\bar{\boldsymbol{y}})$ is upper bounded by the convex function $f(\bar{\boldsymbol{y}}) = ||\bar{\boldsymbol{y}}\boldsymbol{W}_k||_{\ell_2}^2$, where $\boldsymbol{W}_k$ is a diagonal matrix, whose $i$th diagonal element is calculated from $\bar{\boldsymbol{y}}_k = \bar{\boldsymbol{b}}_k\boldsymbol{X}$ using the following equation

$$W_k[t,t] = \sqrt{\frac{d_k[t]}{2y_k[t]}}, \quad t = 1,\ldots,T, \tag{5.14}$$

where $d_k[t] = \frac{\partial g_c(y)}{\partial y}|_{y=y_k[t]}$. The proposed algorithm is based on minimizing the objective function $g(\bar{\boldsymbol{y}})$ by iteratively minimizing its upper bound $f(\bar{\boldsymbol{y}})$. Therefore, given an initial estimate of the separating vector $\bar{\boldsymbol{b}}_k$, a new estimate can be obtained by solving the following optimization problem

$$\bar{\boldsymbol{b}}_{k+1} = \arg\min_{\bar{\boldsymbol{b}}} ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_k||_{\ell_2}^2 \quad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1. \tag{5.15}$$

Comparing (5.15) with (5.8) we readily find that

$$\bar{\boldsymbol{b}}_{k+1}^T = \text{eig}_{min}\left(\boldsymbol{X}_k\boldsymbol{X}_k^T\right). \tag{5.16}$$

where $\boldsymbol{X}_k = \boldsymbol{X}\boldsymbol{W}_k$. Note that the $t$-th column of $\boldsymbol{X}_k$ is a scaled version of the $t$-th column of $\boldsymbol{X}$, with a scaling factor given by $W_k[t,t]$ defined in (5.14). Since $W_k[t,t]$ is inversely proportional to $y_k[t]$, multiplying the data matrix $\boldsymbol{X}$ by the diagonal matrix $\boldsymbol{W}_k$ has the effect of selecting the columns of the data matrix with indices corresponding to the indices of the small entries in the estimated source vector $\bar{\boldsymbol{y}}_k$. As the algorithm converges to the desired sparse signal, the diagonal matrix becomes more selective and $\boldsymbol{X}_k$ converges to $\boldsymbol{X}_{\mathcal{I}_i}$, defined in (5.7) as $k \rightarrow \infty$. The proposed algorithm is summarized in Table 5.1

### 5.2.2 Removing the extracted source from the mixture.

After estimating the source signal $\bar{\boldsymbol{y}}$ using the BSSE algorithm presented in Table 5.1, the estimated source must be removed from the mixture before extracting a new

Table 5.1: Extraction of a sparse source

$[\bar{\boldsymbol{y}}, \bar{\boldsymbol{b}}] = \text{BSSE}(\boldsymbol{X}, \bar{\boldsymbol{b}}_0)$

Select a small threshold parameter $\epsilon$, and calculate $\bar{\boldsymbol{y}}_0 = \bar{\boldsymbol{b}}_0 \boldsymbol{X}$.

1. For $k = 0, 1, \ldots,$ repeat until convergence:
   - Calculate $W_k[t,t] = \sqrt{\frac{d_k[t]}{2y_k[t]}}, \quad t = 1, \ldots, T.$
   - Update the separating vector: $\bar{\boldsymbol{b}}_{k+1}^T = \text{eig}_{min}\left(\boldsymbol{X}\boldsymbol{W}_k^2\boldsymbol{X}^T\right).$
   - Get a new estimate of the separated signal: $\bar{\boldsymbol{y}}_{k+1} = \bar{\boldsymbol{b}}_{k+1}\boldsymbol{X}.$
   - if $||\bar{\boldsymbol{y}}_{k+1} - \bar{\boldsymbol{y}}_k||_{\ell_2} / ||\bar{\boldsymbol{y}}_{k+1}||_{\ell_2} < \epsilon$, stop.
2. Output $\bar{\boldsymbol{y}}_{k+1}$ and $\bar{\boldsymbol{b}}_{k+1}$ as the solutions.

End

source. This can be done by finding a vector $\boldsymbol{h} \in \mathbb{R}^n$ that minimizes the following objective function [55]

$$\mathcal{J}(\boldsymbol{h}) = \sum_{t=1}^{T} \boldsymbol{x}_{jt}^T \boldsymbol{x}_{jt}, \tag{5.17}$$

where $\boldsymbol{x}_{jt}$ is the $t$-th column of the matrix $\boldsymbol{X}^j = \boldsymbol{X}^{j-1} - \boldsymbol{h}\bar{\boldsymbol{y}}$, where $j$ is an index corresponding to the number of the previously extracted source signals. Therefore, $\boldsymbol{X}^0 = \boldsymbol{X}$. Substituting the expression for $\boldsymbol{x}_{jt}$ into (5.17) we get

$$\begin{aligned} \mathcal{J}(\boldsymbol{h}) &= \sum_{t=1}^{T} \left(\boldsymbol{x}_{(j-1)t} - \boldsymbol{h}y[t]\right)^T \left(\boldsymbol{x}_{(j-1)t} - \boldsymbol{h}y[t]\right) \\ &= \sum_{t=1}^{T} \boldsymbol{x}_{(j-1)t}^T \boldsymbol{x}_{(j-1)t} - 2\boldsymbol{h}^T \sum_{t=1}^{T} \boldsymbol{x}_{(j-1)t} y[t] + \boldsymbol{h}^T\boldsymbol{h} \sum_{t=1}^{T} y^2[t], \end{aligned} \tag{5.18}$$

where $\boldsymbol{x}_{(j-1)t}$ is the $t$-th column of $\boldsymbol{X}^{j-1}$. Setting the gradient of $\mathcal{J}(\boldsymbol{h})$ with respect to $\boldsymbol{h}$ equal to zero, we get

$$\boldsymbol{h} = \frac{\boldsymbol{X}^{j-1}\bar{\boldsymbol{y}}^T}{||\bar{\boldsymbol{y}}||_{\ell_2}^2} = \frac{\boldsymbol{X}^{j-1}\left(\boldsymbol{X}^{j-1}\right)^T\bar{\boldsymbol{b}}^T}{||\bar{\boldsymbol{y}}||_{\ell_2}^2}, \tag{5.19}$$

where $\bar{\boldsymbol{b}}$ is the separating vector obtained from the BSSE algorithm in Table 5.1. Note that, $\boldsymbol{h}$ is, in fact, an estimate of the mixing column associated with the extracted source $\bar{\boldsymbol{y}}$. The new matrix $\boldsymbol{X}^j$ can then be used as an input to the BSSE algorithm for extracting the $(j+1)$–st source signal, which in turn is removed from $\boldsymbol{X}^j$ before extracting a new source signal. This process can be repeated until all the desired source signals are extracted. Note that, after removing $j \geq 1$ source signals, and for estimating the separating vector of the $(j+1)$–st source signal, the second step in Table 5.1 must be changed into

$$\bar{\boldsymbol{b}}_{k+1}^T = \text{eig}_{j+1}\left(\boldsymbol{X}^j\boldsymbol{W}_k^2\left(\boldsymbol{X}^j\right)^T\right),$$

where $\text{eig}_{j+1}(\boldsymbol{R})$ is the eigenvector corresponding to the $(j+1)$–st smallest eigenvalue of the correlation matrix $\boldsymbol{R}$. This is because, ideally, the smallest $j$ eigenvalues of the argument above are zero.

## 5.3 Constrained Sparse Component Analysis (cSCA)

The algorithm proposed in the previous section is useful for sequential extraction of sparse sources. Therefore, if one is interested in extracting a single source signal that has certain properties, then (ideally) the desired source will be the first extracted source signal as long as it is the sparsest source among all other sources. However, if the desired source is not the sparsest one, or if all the sources have the same sparsity, then there is no guarantee that the first extracted source is the desired one. One solution to this problem is to keep extracting sources one after the other until the

desired source is extracted. Clearly, this is not a good approach, especially when the number of sources is large. A more reasonable approach to overcome this difficulty is to explicitly incorporate all (or some of) the information available about the desired source signal into the optimization problem designed for extracting the sources.

In the field of biomedical signal processing there is always some information available about the desired signal. This information could be a temporal pattern or the topographic map of the desired source. For example, it is known that eye–blink artifact contaminates the frontal electrodes (spatial information), and its amplitude is much higher than the amplitude of the brain signal. Accordingly, temporal information about the eye–blink artifact can be obtained as the time samples at which one of the frontal electrodes exceeds a certain threshold. On the other hand, in the application of estimating the ERP signal from an EEG/MEG data, it is known *a priori* that there is a certain latency between the stimulus onset and the generation of the ERP signal; e.g., on average, the peak of the P300 occurs around 300 ms after the stimulus onset. This information can be incorporated as a temporal constraint for extracting the ERP signal.

In this section we propose three new algorithms that explicitly incorporate either temporal, support, or spatial information about the desired source signal into the optimization problem (5.15). In the three algorithms, a separating vector $\bar{\boldsymbol{b}}$ is iteratively obtained by solving the following optimization problem

$$\bar{\boldsymbol{b}}_1 = \arg\min_{\bar{\boldsymbol{b}}} ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_0||_{\ell_2}^2 + \gamma \mathcal{J}_r(\bar{\boldsymbol{b}}; \boldsymbol{r}) \quad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1, \tag{5.20}$$

where $\boldsymbol{r} \in \mathbb{R}^T$ is a reference vector that provides the information available about the desired signal, and $\gamma \geq 0$ is a regularization parameter. The regularizing function $\mathcal{J}_r(\bar{\boldsymbol{b}}; \boldsymbol{r})$ depends on whether the reference vector $\boldsymbol{r}$ conveys temporal or spatial information. Methods for choosing the reference vector $\boldsymbol{r}$ are discussed later in this chapter. Note that the constraint $||\bar{\boldsymbol{b}}||_{\ell_2} = 1$ is introduced to prevent the entries of

the solution vector $\bar{\boldsymbol{b}}$ from growing without limit, producing a minimum equal to $-\infty$.

### 5.3.1 Temporal constraint SCA (tcSCA) algorithm

In this subsection we consider the case of having a reference signal $\bar{r}_t$ that conveys temporal information about the desired source signal. The reference signal must be selected carefully to have enough information to guide the algorithm towards the desired signal. However, this does not mean that the reference signal is a replica of the desired signal.

The least squares approach is the conventional method for estimating a signal that is close to a given reference signal. In this method, a separating vector $\bar{\boldsymbol{b}}_{ls} \in \mathbb{R}^n$ is estimated by minimizing the following objective function

$$\bar{\boldsymbol{b}}_{ls} = \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_{ls}\left(\bar{\boldsymbol{b}}\right) \triangleq ||\bar{r}_t - \bar{\boldsymbol{b}}\boldsymbol{X}||_{\ell_2}^2 \tag{5.21}$$

for which the following closed–form solution is

$$\bar{\boldsymbol{b}}_{ls} = \bar{r}_t \boldsymbol{X}^T \left(\boldsymbol{X}\boldsymbol{X}^T\right)^{-1}, \tag{5.22}$$

and the corresponding extracted signal is given by

$$\bar{\boldsymbol{y}}_{ls} = \bar{\boldsymbol{b}}_{ls}\boldsymbol{X} = \bar{r}_t \boldsymbol{X}^T \left(\boldsymbol{X}\boldsymbol{X}^T\right)^{-1} \boldsymbol{X}. \tag{5.23}$$

It is clear that the least squares solution does not utilize all the information available about the desired source signal, since it neglects the sparsity of the desired source signal. Accordingly, it is expected that the extracted source signal using the least squares approach is, in general, non–sparse. To enforce the sparsity of the extracted source signal, both the sparsity and the closeness to the reference vector $\bar{r}_t$ must be incorporated into the objective function. There are many forms for measuring the closeness between two vectors. In this subsection we consider only two of them, the correlation coefficient and the mean–square error (MSE) between the two vectors.

### First approach: the correlation coefficient as a measure of closeness

In the first approach, the correlation coefficient between the desired signal and the reference signal is used as a measure of closeness between these two signals. Accordingly, in this case, $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r})$ in (5.20) has the form $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r}) = -\bar{\boldsymbol{y}}\bar{\boldsymbol{r}}_t^T = -\bar{\boldsymbol{b}}\boldsymbol{X}\bar{\boldsymbol{r}}_t^T$. Substituting this expression into (5.20) we get

$$\begin{aligned} \bar{\boldsymbol{b}}_{t1} &= \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_{t1}\left(\bar{\boldsymbol{b}}\right) \triangleq ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_0||_{\ell_2}^2 - \gamma\bar{\boldsymbol{b}}\boldsymbol{X}\bar{\boldsymbol{r}}_t^T \\ &\quad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1. \end{aligned} \tag{5.24}$$

This optimization problem can be solved by minimizing $\mathcal{J}_{t1}\left(\bar{\boldsymbol{b}}\right)$ with respect to $\bar{\boldsymbol{b}}$, then projecting the solution vector onto the surface of the unit sphere. Following these two steps, the expression of $\bar{\boldsymbol{b}}_{t1}$ is given by

$$\bar{\boldsymbol{b}}_{t1}^{+} = \gamma\bar{\boldsymbol{r}}_t\boldsymbol{X}^T\left(\boldsymbol{X}\boldsymbol{W}_0^2\boldsymbol{X}^T\right)^{-1}, \tag{5.25}$$

$$\bar{\boldsymbol{b}}_{t1} = \frac{\bar{\boldsymbol{b}}_{t1}^{+}}{||\bar{\boldsymbol{b}}_{t1}^{+}||_{\ell_2}}. \tag{5.26}$$

Note that, due to the normalization step (5.26), the value of $\bar{\boldsymbol{b}}_{t1}$ in this case does not depend on the value of the regularization parameter $\gamma$.

### Second approach: MSE as a measure of closeness

In the second approach, the MSE between the desired source signal and the reference signal is used as a measure of closeness between the two signals. Accordingly, in this case, $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r})$ in (5.20) has the form $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r}) = ||\bar{\boldsymbol{y}} - \bar{\boldsymbol{r}}_t||_{\ell_2}^2 = ||\bar{\boldsymbol{b}}\boldsymbol{X} - \bar{\boldsymbol{r}}_t||_{\ell_2}^2$. Substituting this expression into (5.20) we get

$$\bar{\boldsymbol{b}}_{t2} = \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_{t2}\left(\bar{\boldsymbol{b}}\right) \triangleq ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_k||_{\ell_2}^2 + \gamma||\bar{\boldsymbol{b}}\boldsymbol{X} - \bar{\boldsymbol{r}}_t||_{\ell_2}^2 \tag{5.27}$$

$$\text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1. \tag{5.28}$$

Table 5.2: Temporally Constrained SCA

$[\bar{y}, \bar{b}] = \text{tcSCA}(\boldsymbol{X}, \bar{r}_t, \alpha)$

Select a small threshold parameter $\epsilon$, and calculate $\bar{y}_0 = \bar{r}_t$.

1. For $k = 0, 1, \ldots$, repeat until convergence:
   - Calculate $W_k[t,t] = \sqrt{\frac{d_k[t]}{2y_k[t]}}, \quad t = 1, \ldots, T.$
   - if $\alpha = 1$, then calculate the value of $\gamma$ using the L–curve method (see text for more details).
   - Calculate the separating vector:

     $\bar{b}_{k+1}^{+} = \bar{r}_t \boldsymbol{X}^T \left( \boldsymbol{X} \left( \boldsymbol{W}_k^2 + \alpha\gamma \boldsymbol{I} \right) \boldsymbol{X}^T \right)^{-1}$

     $\bar{b}_{k+1} = \frac{\bar{b}_{k+1}^{+}}{||\bar{b}_{k+1}^{+}||_{\ell_2}}.$
   - Get a new estimate of the desired signal: $\bar{y}_{k+1} = \bar{b}_{k+1} \boldsymbol{X}$.
   - if $||\bar{y}_{k+1} - \bar{y}_k||_{\ell_2} / ||\bar{y}_{k+1}||_{\ell_2} < \epsilon$, stop.
2. Output $\bar{y}_{k+1}$ and $\bar{b}_{k+1}$ as the solutions.

End

This optimization problem can also be solved by minimizing $\mathcal{J}_{t2}\left(\bar{b}\right)$ with respect to $\bar{b}$, then projecting the solution vector onto the surface of the unit sphere. Following these two steps, the expression of $\bar{b}_{t1}$ is given by

$$\bar{b}_{t2}^{+} = \gamma \bar{r}_t \boldsymbol{X}^T \left( \boldsymbol{X} \left( \boldsymbol{W}_k^2 + \gamma \boldsymbol{I} \right) \boldsymbol{X}^T \right)^{-1}, \tag{5.29}$$

$$\bar{b}_{t2} = \frac{\bar{b}_{t2}^{+}}{||\bar{b}_{t2}^{+}||_{\ell_2}}. \tag{5.30}$$

where $\boldsymbol{I}$ is the identity matrix. A method for calculating the optimum value of $\gamma$ is presented shortly in this subsection. In both approaches, a new estimate of the

desired signal is calculated and the diagonal elements of $\boldsymbol{W}_k$ are updated using (5.14). The process is repeated until convergence.

### Calculating the value of the regularization parameter $\gamma$

The regularization parameter $\gamma$ can be calculated by first converting the objective function $\mathcal{J}_{t2}\left(\bar{\boldsymbol{b}}\right)$ defined in (5.28) into the standard form of Tikhonov regularization, then calculating the regularization parameter using one of the standard techniques, e.g., the L–curve method [79–81]. The objective function $\mathcal{J}_{t2}\left(\bar{\boldsymbol{b}}\right)$ can be converted to the standard form of Tikhonov regularization by defining $\bar{\boldsymbol{d}} = \bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_k$ and $\boldsymbol{C}_k = \boldsymbol{W}_k^{-1}\boldsymbol{X}^T\left(\boldsymbol{X}\boldsymbol{X}^T\right)^{-1}\boldsymbol{X}$, then expressing $\mathcal{J}_{t2}\left(\bar{\boldsymbol{b}}\right)$ as

$$\mathcal{J}_{t2}\left(\bar{\boldsymbol{d}}\right) = ||\bar{\boldsymbol{d}}||_{\ell_2}^2 + \gamma||\bar{\boldsymbol{d}}\boldsymbol{C}_k - \bar{\boldsymbol{r}}_t||_{\ell_2}^2. \tag{5.31}$$

The optimum solution vector $\bar{\boldsymbol{d}}$ to (5.31) is the one that has minimum norm value and minimizes the residual error $||\bar{\boldsymbol{d}}\boldsymbol{C}_k - \bar{\boldsymbol{r}}_t||_{\ell_2}^2$. The solution vector $\bar{\boldsymbol{d}}$ depends on the value of $\gamma$. A large value of $\gamma$ reduces the residual error while increasing the size of $\bar{\boldsymbol{d}}$, and vice versa. The optimum choice of $\gamma$ is the one that keeps these two values small. There are many techniques that have been proposed in the literature for finding such a value for $\gamma$, e.g., the L–curve and the generalized cross validation methods. See [79–81] and the references therein for more details. In this chapter the L–curve method, which is implemented in the Regularization Toolbox [1], is used for calculating the regularization parameter.

The proposed algorithm is summarized in Table 5.2. The input parameter $\alpha$ is a flag indicating which method is used as a measure of closeness, i.e., $\alpha = 1$ for the MSE case, while $\alpha = 0$ for the correlation coefficient case.

**Example:** In this example we provide a quick comparison between the original least squares solution and the proposed algorithms. The data for this example were

[1] This is a free software available at http://www2.imm.dtu.dk/~pch/Regutools/

constructed as follows. A sparse source matrix $\boldsymbol{S} \in \mathbb{R}^{10\times150}$ is generated such that each row has exactly 14 nonzero entries. The indices of these 14 entries are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm1$. The measured data matrix $\boldsymbol{X} \in \mathbb{R}^{10\times150}$ is constructed by multiplying the source matrix $\boldsymbol{S}$ by a square mixing matrix $\boldsymbol{A} \in \mathbb{R}^{10\times10}$ randomly generated from a white normal distribution with zero mean and unit variance. The reference signal is constructed by adding a zero mean white Gaussian random noise to the desired source signal. The amplitude of the noise is adjusted to produce a 2dB SNR. The results are shown in Figure 5.1

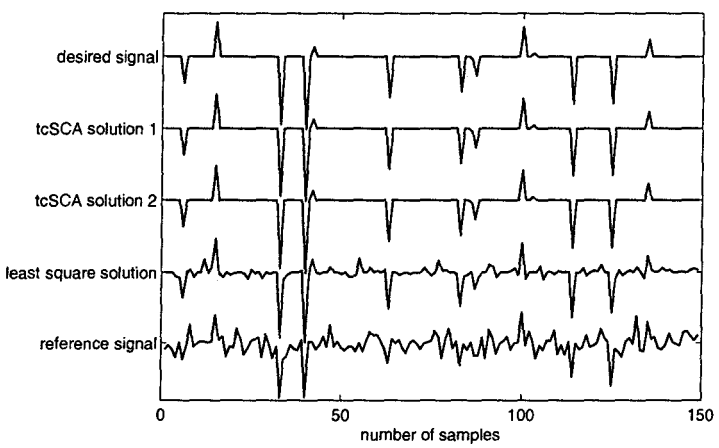



Figure 5.1: Comparison between the least squares solution and the solutions produced by the tcSCA algorithm. The signals from top to bottom are the desired signal, the estimated signal using the tcSCA algorithm with $\alpha = 0$, the estimated signal using the tcSCA algorithm with $\alpha = 1$, the estimated signal using the lest squares approach, and the reference signal, respectively.

As shown in this figure, and for the two cases $\alpha = 0$ and $\alpha = 1$, the desired signal is correctly estimated using the tcSCA algorithm, while the least–squares approach produced a non–sparse signal. As a measure of performance, the average normalized

MSE (NMSE) between the desired signal and the extracted signal is calculated for each approach, where the NMSE between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as

$$NMSE = \left|\left| \frac{\boldsymbol{x}}{||\boldsymbol{x}||_{\ell_2}} - \frac{\boldsymbol{y}}{||\boldsymbol{y}||_{\ell_2}} \right|\right|_2 . \tag{5.32}$$

The average NMSE is calculated over 1000 different runs, where in each run new source and mixing matrices are generated. The value of the average NMSE for the least–squares approach, the tcSCA with $\alpha = 0$, and the tcSCA with $\alpha = 1$ are 0.2016, $3.67 \times 10^{-4}$, and $3.66 \times 10^{-4}$, respectively. Clearly the value of the NMSE for the tcSCA approach is much less than that of the least–square approach.

### 5.3.2 Support constraint SCA (sucSCA) algorithm

In this subsection we propose an algorithm that can extract a sparse source signal when only the *support* of that source signal is known, where the support of a signal is defined as the indices of its nonzero samples. This algorithm can be useful when the signs and the magnitudes of the nonzero samples of the desired source signal are unknowns, while only the indices of these samples are known.

One application of this algorithm in the field of biomedical signal processing is removing the line voltage interference from the measured EEG/MEG data. Usually a notch filter is used for suppressing the 50/60 Hz noise from the measured EEG/MEG data. However, if an experiment is designed such that the desired response signal is expected to occupy the frequency band around the 50/60 Hz, then the notch filter will remove the desired response signal as well. Since the line voltage interference is very sparse in the frequency domain (it consists of a single pulse at 50/60 Hz), and its support (50/60 Hz) is known, this signal can be readily extracted using the proposed algorithm. See Example 3 in Section 5.4.

The function $\mathcal{J}_r(\bar{\boldsymbol{b}}; \boldsymbol{r})$ in the proposed sucSCA algorithm measures the closeness between a nonnegative reference signal $\bar{\boldsymbol{r}}_{su} \geq 0$ and the square of the desired signal.

We can use the correlation coefficient between these two signals as such a measure. Since there is no information available about the absolute values of the nonzero entries of the desired signal, a reasonable choice of the reference signal $\bar{\boldsymbol{r}}_{su}$ is the sign of the absolute value of the desired signal, i.e., it is one within the support of the desired signal and zero elsewhere. However, other choices, e.g., random positive entries within the support of the desired signal, can also be used.

The function $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r})$ in the proposed sucSCA algorithm has the following expression $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r}) = -\left(\bar{\boldsymbol{y}} \odot \bar{\boldsymbol{y}}\right)\bar{\boldsymbol{r}}_{su}^T$, where $\odot$ denotes the Hadamard product; i.e., element-by-element multiplication. The quantity $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r})$ can be expressed in terms of $\bar{\boldsymbol{b}}$ as follows

$$\begin{aligned}
\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r}) &= -\left(\bar{\boldsymbol{y}} \odot \bar{\boldsymbol{y}}\right)\bar{\boldsymbol{r}}_{su}^T = -\sum_{t=1}^{T} r_{su}[t]y^2[t] = \sum_{t=1}^{T} r_{su}[t]\left(\bar{\boldsymbol{b}}\boldsymbol{x}_t\right)^2 \\
&= -\sum_{t=1}^{T} r_{su}[t]\left(\bar{\boldsymbol{b}}\boldsymbol{x}_t\boldsymbol{x}_t^T\bar{\boldsymbol{b}}^T\right) = -\bar{\boldsymbol{b}}\left(\sum_{t=1}^{T} r_{su}[t]\boldsymbol{x}_t\boldsymbol{x}_t^T\right)\bar{\boldsymbol{b}}^T \\
&= -\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{D}_{r_{su}}\boldsymbol{X}^T\bar{\boldsymbol{b}}^T. \qquad (5.33)
\end{aligned}$$

where $\boldsymbol{x}_t$ is the $t$-th column of $\boldsymbol{X}$, and $\boldsymbol{D}_{r_{su}} = \mathrm{diag}\left(\boldsymbol{r}_{su}\right)$ is a diagonal matrix whose main diagonal equals the reference signal $\boldsymbol{r}_{su}$.

Substituting $\mathcal{J}_r(\bar{\boldsymbol{b}};\boldsymbol{r})$ into (5.20) we get

$$\begin{aligned}
\bar{\boldsymbol{b}}_{su} &= \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right) \triangleq ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_0||_{\ell_2}^2 - \gamma\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{D}_{r_{su}}\boldsymbol{X}^T\bar{\boldsymbol{b}}^T \\
&\qquad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1. \qquad (5.34)
\end{aligned}$$

Eq. (5.34) can be written in the following form

$$\begin{aligned}
\bar{\boldsymbol{b}}_{su} &= \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right) \triangleq \bar{\boldsymbol{b}}\boldsymbol{X}\left(\boldsymbol{W}_0^2 - \gamma\boldsymbol{D}_{r_{su}}\right)\boldsymbol{X}^T\bar{\boldsymbol{b}}^T \\
&\qquad \text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1. \qquad (5.35)
\end{aligned}$$

which has the solution

$$\bar{\boldsymbol{b}}_{su} = \mathrm{eig}_{min}\left(\boldsymbol{X}\left(\boldsymbol{W}_0^2 - \gamma\boldsymbol{D}_{r_{su}}\right)\boldsymbol{X}^T\right). \qquad (5.36)$$

Table 5.3: Support constraint SCA

$[\bar{\boldsymbol{y}}, \bar{\boldsymbol{b}}] = \text{sucSCA}(\boldsymbol{X}, \bar{\boldsymbol{r}}_{su})$

Select a small threshold parameter $\epsilon$. Calculate $\boldsymbol{D}_{r_{su}} = \text{diag}\,(\bar{\boldsymbol{r}}_{su})$, and set $\bar{\boldsymbol{y}}_0 = \bar{\boldsymbol{r}}_{su} \odot \boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{R}^T$ is a random noise vector, and $\odot$ indicates the Hadamard product.

1. For $k = 0, 1, \ldots$, repeat until convergence:

   - Calculate $\bar{\boldsymbol{w}}_k$, where $w_k[t] = \frac{d_k[t]}{2y_k[t]}, \quad t = 1, \ldots, T.$
   - Set $\boldsymbol{W}_k^2 = \text{diag}\,(\bar{\boldsymbol{w}}_k)$.
   - Calculate the value of $\gamma_k = \min\left(\frac{\bar{\boldsymbol{w}}_k \bar{\boldsymbol{r}}_{su}^T}{\bar{\boldsymbol{r}}_{su} \bar{\boldsymbol{r}}_{su}^T}, \min\left(\bar{\boldsymbol{w}}_k ./ \bar{\boldsymbol{r}}_{su}\right)\right)$.
   - Calculate the separating vector:

     $\bar{\boldsymbol{b}}_{k+1} = \text{eig}_{min}\left(\boldsymbol{X}\left(\boldsymbol{W}_k^2 - \gamma_k \boldsymbol{D}_{r_{su}}\right)\boldsymbol{X}^T\right)$
   - Get a new estimate of the desired signal: $\bar{\boldsymbol{y}}_{k+1} = \bar{\boldsymbol{b}}_{k+1}\boldsymbol{X}$.
   - **Optional step:** Set $\bar{\boldsymbol{y}}_{k+1}(\mathcal{I}_{r_0}) = 0$, where $\mathcal{I}_{r_0}$ are the indices of the zero entries of the reference signal $\bar{\boldsymbol{r}}_{su}$.
   - if $||\bar{\boldsymbol{y}}_{k+1} - \bar{\boldsymbol{y}}_k||_{\ell_2} / ||\bar{\boldsymbol{y}}_{k+1}||_{\ell_2} < \epsilon$, stop.

2. Output $\bar{\boldsymbol{y}}_{k+1}$ and $\bar{\boldsymbol{b}}_{k+1}$ as the solutions.

End

The value of $\gamma$ must be selected to minimize the value of $\mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right)$ in (5.35) and to insure the convexity of $\mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right)$. The convexity of $\mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right)$ is satisfied as long as $\left(\boldsymbol{W}_0^2 - \gamma \boldsymbol{D}_{r_{su}}\right)$ is nonnegative definite. Since both $\boldsymbol{W}_0$ and $\boldsymbol{D}_{r_{su}}$ are diagonal matrices, the value of $\gamma$ can be easily selected to satisfy this condition. Towards that end, let $\bar{w}_0 \in \mathbb{R}^T$ denote the main diagonal of $\boldsymbol{W}_0^2$, then the value of $\gamma$ that satisfies the convexity of $\mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right)$ must be selected such that

$$0 \leq \gamma \leq \min\left(\bar{w}_0 ./ \bar{r}_{su}\right), \tag{5.37}$$

where the division operator is applied element wise.

On the other hand, the value of $\gamma$ that minimizes the value of $\mathcal{J}_{su}\left(\bar{\boldsymbol{b}}\right)$ in (5.35) could be selected as the value that minimizes the difference $\left(\boldsymbol{W}_0^2 - \gamma \boldsymbol{D}_{r_{su}}\right)$, or equivalently $||\bar{w}_0 - \gamma \bar{r}_{su}||_{\ell_2}^2$, which is readily given by

$$\gamma = \frac{\bar{w}_0 \bar{r}_{su}^T}{\bar{r}_{su} \bar{r}_{su}^T}. \tag{5.38}$$

Combining (5.37) and (5.38), the value of $\gamma$ can be calculated as

$$0 \leq \gamma \leq \min\left(\frac{\bar{w}_0 \bar{r}_{su}^T}{\bar{r}_{su} \bar{r}_{su}^T}, \min\left(\bar{w}_0 ./ \bar{r}_{su}\right)\right). \tag{5.39}$$

The sucSCA algorithm is summarized in Table 5.3. The optional step in this table restricts the nonzero entries of the estimated signal to the support of the reference signal, which is also the support of the desired signal. This optional step is useful in a case when the measured signals are noisy.

**Example:** This example demonstrates the ability of the sucSCA algorithm to extract a sparse source signal when only the support of this signal is known. In this example two different observation matrices and two different reference signals are constructed. The two observation matrices are constructed as $\boldsymbol{X}_1 = \boldsymbol{A}\boldsymbol{S}$ and $\boldsymbol{X}_2 = \boldsymbol{X}_1 + \boldsymbol{V}$, where $\boldsymbol{S} \in \mathbb{R}^{10\times 150}$ is the sparse source matrix, $\boldsymbol{A} \in \mathbb{R}^{10\times 10}$ is a square mixing matrix, and $\boldsymbol{V} \in \mathbb{R}^{10\times 150}$ is a noise matrix which is randomly generated from

a white normal distribution with zero mean and unit variance. The entries of the noise matrix are adjusted such that the SNR = 10 dB.

The sparse source matrix $\boldsymbol{S}$ is generated such that each row has exactly 15 nonzero entries. The indices of these 15 entries are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm 1$. The square mixing matrix $\boldsymbol{A}$ is randomly generated from a white normal distribution with zero mean and unit variance. The two reference vectors are set to have the same support as the first original source signal. The nonzero entries of the first reference vector $\bar{\boldsymbol{r}}_1$ are all ones, while for the second reference vector, $\bar{\boldsymbol{r}}_2$, they are randomly generated from a random variable which is uniformly distributed between zero and one. According to the four different combinations of $\boldsymbol{X}_i$ and $\bar{\boldsymbol{r}}_i$, $i = 1, 2$, we have four different cases. Each combination of these four different combinations is used as an input to the sucSCA algorithm, and an estimate of the desired signal is obtained. The optional step in Table 5.3 is used with the case of noisy measurements. The four estimated signals using the sucSCA algorithm, as well as the desired signal and the two reference signals are shown in Figure 5.2.

As shown in this figure, the desired signal is correctly estimated in the four different scenarios. This experiment is repeated 1000 different times, and in each time new $\boldsymbol{S}$, $\boldsymbol{A}$, and $\boldsymbol{V}$ are generated, and the NMSE between the desired signal and the extracted signal is calculated for each one of the four different cases. The average NMSE between the estimated signal and the desired signal for the four cases $(\boldsymbol{X}_1; \bar{\boldsymbol{r}}_1)$, $(\boldsymbol{X}_2; \bar{\boldsymbol{r}}_1)$, $(\boldsymbol{X}_1; \bar{\boldsymbol{r}}_2)$, and $(\boldsymbol{X}_2; \bar{\boldsymbol{r}}_2)$ are 0.0933, 0.2498, 0.1167, and 0.2566, respectively. As expected, the best performance is obtained when $\bar{\boldsymbol{r}}_1$ is used with noise–free measurements, while the worst performance is obtained when $\bar{\boldsymbol{r}}_2$ is used with noisy measurements. In terms of the reference vector, it is clear that, for a given measurement matrix, the NMSE is small when $\bar{\boldsymbol{r}}_1$ is used as a reference vector. Accordingly, on an empirical basis, we recommend using $\bar{\boldsymbol{r}}_1$ as a reference vector with

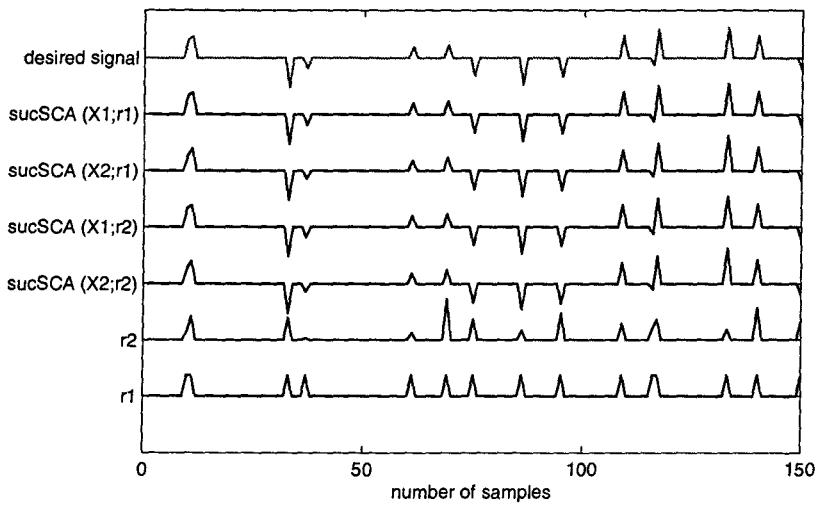



Figure 5.2: An example of extracting a sparse signal using sucSCA algorithm in four different scenarios.

the sucSCA algorithm.

### 5.3.3 Spatial constraint SCA (scSCA) algorithm

Recall from (5.1) that the measured signal at the $t$-th time instant can be expressed as

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t) = \sum_{i=1}^{n} s_i(t)\boldsymbol{a}_i, \quad t = 1, \ldots, T. \tag{5.40}$$

In the previous two subsections we proposed two new algorithms for extracting the desired signal $s_j(t)$, when some information about its temporal characteristics are available. In this subsection we propose a new algorithm for estimating the desired signal when some information is available about its spatial distribution, i.e., its mixing vector $\boldsymbol{a}_j$. The mixing vector $\boldsymbol{a}_j$ represents the contribution of the desired signal at the measuring sensors. In the field of EEG/MEG signal processing, the vector $\boldsymbol{a}_j$ represents the *topographic map* of the desired signal.

The topographic maps of some of the desired signals are known *a priori*, or can be estimated using visual inspection. In these cases, a reference vector of each one of these maps can be readily constructed. For example, it is known that the eye blink artifact contaminates the frontal electrodes. Accordingly a reference spatial vector for this source can be constructed by generating a vector of size $n$ (the number of sensors), and filling this vector by zeros and ones, where the indices of the nonzero entries correspond to those of the frontal electrodes.

Given some reference vectors for some of the mixing columns, previous approaches for estimating the corresponding sources were proposed in [118, 120, 121]. These approaches are based on modifying an ICA based algorithm in a way that some of the columns of the mixing matrix are initialized by the reference vectors and kept fixed during the optimization process. This approach is known as a *hard* spatial constraint. This type of hard spatial constraint is only advisable when the reference vectors are very close to the mixing columns of the desired sources [122]. Another approach, called *soft* spatial constraint [118], allows the columns of the mixing matrix that were initialized by the reference vectors to change slightly during the optimization process. In both cases, all components must be estimated first, then the desired signals are identified as the components associated with the reference mixing columns.

In this subsection we propose a new algorithm that can extract a sparse signal when prior information about its mixing column $\boldsymbol{h}$ is provided by a reference vector $\boldsymbol{r}_s$. The prior information could be the sign of the individual entries of the mixing column, or rough estimate of the contribution of the desired source at different electrodes. In the proposed scSCA algorithm, the correlation coefficient between the reference vector $\boldsymbol{r}_s$ and the mixing column $\boldsymbol{h}$ will be used as a measure of closeness between the two vectors. Therefore, in this case, the objective function $\mathcal{J}_r(\bar{\boldsymbol{b}}; \boldsymbol{r})$ in (5.20) can be expressed as $\mathcal{J}_r(\bar{\boldsymbol{b}}; \boldsymbol{r}) = -\boldsymbol{h}^T \boldsymbol{r}_s$. To express this objective function in terms of the separating vector $\bar{\boldsymbol{b}}$, rather than the mixing vector $\boldsymbol{h}$, recall from (5.19) that, the

Table 5.4: Spatial constraint SCA

$[\bar{\boldsymbol{y}}, \bar{\boldsymbol{b}}] = \text{scSCA}(\boldsymbol{X}, \boldsymbol{r}_s)$

Select a small threshold parameter $\epsilon$, and calculate $\boldsymbol{R}_x = \boldsymbol{X}\boldsymbol{X}^T$. Generate a random vector $\bar{\boldsymbol{y}}_0^+$, then set the initial estimate of the desired source as $\bar{\boldsymbol{y}}_0 = \frac{\bar{\boldsymbol{y}}_0^+}{||\bar{\boldsymbol{y}}_0^+||_{\ell_2}}$

1. For $k = 0, 1, \ldots$, repeat until convergence:
   - Calculate $W_k[t,t] = \sqrt{\frac{d_k[t]}{2y_k[t]}}, \quad t = 1, \ldots, T.$
   - Calculate the separating vector: $\bar{\boldsymbol{b}}_{k+1} = \left(\boldsymbol{X}\boldsymbol{W}_k^2\boldsymbol{X}^T\right)^{-1}\boldsymbol{R}_x\boldsymbol{r}_s$
   - Get a new estimate of the desired signal: $\bar{\boldsymbol{y}}_{k+1} = \frac{\bar{\boldsymbol{b}}_{k+1}\boldsymbol{X}}{||\bar{\boldsymbol{b}}_{k+1}\boldsymbol{X}||_{\ell_2}}$.
   - if $||\bar{\boldsymbol{y}}_{k+1} - \bar{\boldsymbol{y}}_k||_{\ell_2} < \epsilon$, stop.
2. Output $\bar{\boldsymbol{y}}_{k+1}$ and $\bar{\boldsymbol{b}}_{k+1}$ as the solutions.

End

mixing column of an estimated source $\bar{\boldsymbol{y}} = \bar{\boldsymbol{b}}\boldsymbol{X}$ is given by $\boldsymbol{h} = \frac{\boldsymbol{X}\boldsymbol{X}^T\bar{\boldsymbol{b}}^T}{||\bar{\boldsymbol{y}}||_{\ell_2}^2}$, where $\bar{\boldsymbol{b}}$ is the corresponding separating vector. Substituting this expression into the expression of $\mathcal{J}_r(\bar{\boldsymbol{b}}; \boldsymbol{r})$, the optimization problem (5.20) in this case is expressed as

$$\begin{aligned} \bar{\boldsymbol{b}}_s &= \arg\min_{\bar{\boldsymbol{b}}} ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_k||_{\ell_2}^2 - \gamma\frac{\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{r}_s^T}{||\bar{\boldsymbol{y}}||_{\ell_2}^2} \\ &\text{subject to} \quad ||\bar{\boldsymbol{b}}||_{\ell_2} = 1. \end{aligned} \tag{5.41}$$

Recall that the constraint $||\bar{\boldsymbol{b}}||_{\ell_2} = 1$ was introduced into the optimization problem (5.20) to prevent the entries of the solution $\bar{\boldsymbol{b}}$ from growing without limit, and hence producing a minimum equal to $-\infty$. However, this goal can also be accomplished by constraining the norm of the extracted signal, i.e., by setting $||\bar{\boldsymbol{y}}||_{\ell_2} = 1$. Accordingly,

by replacing the constraint $||\bar{\boldsymbol{b}}||_{\ell_2} = 1$ in (5.41) by the constraint $||\bar{\boldsymbol{y}}||_{\ell_2} = 1$, the optimization problem (5.41) becomes more tractable and is reduced to the following expression

$$\begin{aligned} \bar{\boldsymbol{b}}_s &= \arg\min_{\bar{\boldsymbol{b}}} \mathcal{J}_s(\bar{\boldsymbol{b}};\boldsymbol{r}) \triangleq ||\bar{\boldsymbol{b}}\boldsymbol{X}\boldsymbol{W}_k||_{\ell_2}^2 - \gamma \bar{\boldsymbol{b}}\boldsymbol{R}_x\boldsymbol{r}_s^T \\ &\text{subject to} \quad ||\bar{\boldsymbol{y}}||_{\ell_2}^2 = \bar{\boldsymbol{b}}\boldsymbol{R}_x\bar{\boldsymbol{b}}^T = 1, \end{aligned} \tag{5.42}$$

where $\boldsymbol{R}_x = \boldsymbol{X}\boldsymbol{X}^T$.

The optimization problem (5.42) can be solved in two steps, where in the first step an estimate of the separating vector $\bar{\boldsymbol{b}}_s$ is obtained by minimizing the objective function $\mathcal{J}_s(\bar{\boldsymbol{b}};\boldsymbol{r})$, then in the second step the constraint $||\bar{\boldsymbol{y}}||_{\ell_2}^2 = 1$ is satisfied by projecting the vector $\bar{\boldsymbol{y}} = \bar{\boldsymbol{b}}_s\boldsymbol{X}$ onto the surface of the unit circle. A new value of the diagonal matrix $\boldsymbol{W}_k$ is then calculated using $\bar{\boldsymbol{y}}$, and the procedure is repeated until convergence. The overall algorithm is summarized in Table 5.4. Note that the value of the solution vector does not depend on the value of the regularization parameter $\gamma$ due to the normalization step. Examples for the scSCA approach are given in the next section.

## 5.4 Simulation results

In this section we present four different examples that demonstrate the ability of the proposed algorithms in estimating sparse sources. In the first example we provide a comparison between the first algorithm summarized in Table 5.1 and the FastICA algorithm [52]. The remaining examples demonstrate the ability of the cSCA algorithms in extracting some desired signals from EEG data when different prior information about the desired signals is available.

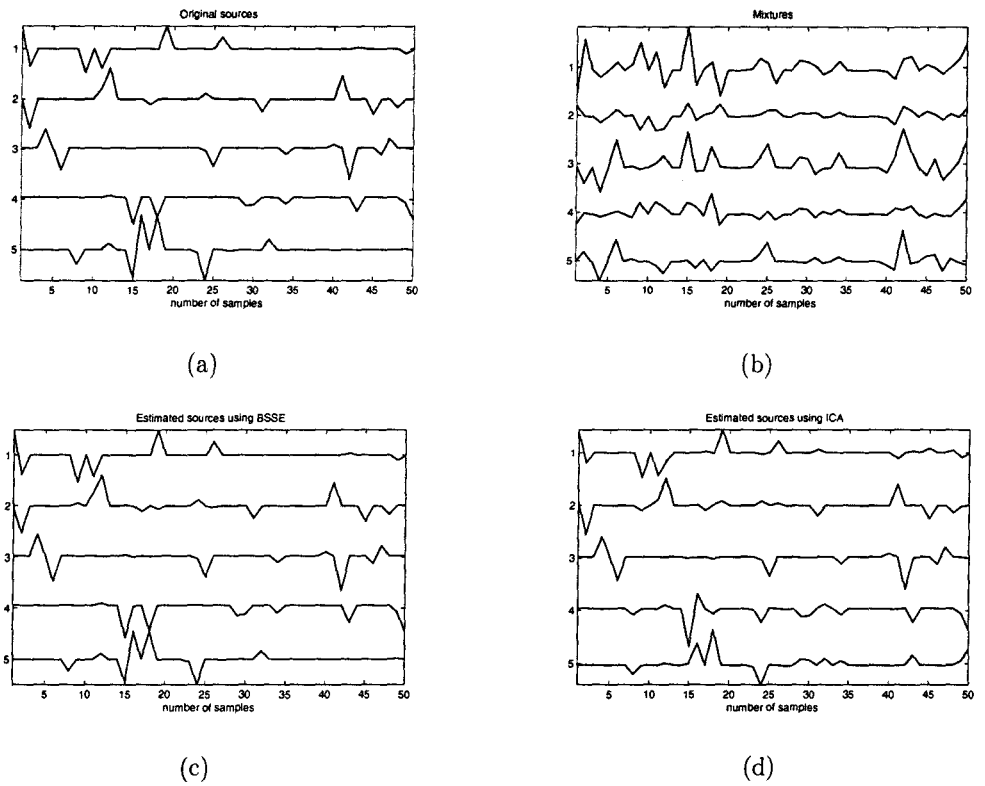



Figure 5.3: Results of Example 1. (a) Original source signals, (b) Mixed signals, (c) and (d) Estimated source signals using the BSSE algorithm and the FastICA algorithm, respectively.

### Example 1: Comparison between BSSE and ICA

In this example we provide a comparison between the BSSE algorithm, summarized in Table 5.1, and the FastICA algorithm in solving the BSS problem when the hidden sources are sparse. The measured signals are generated by multiplying a $(5 \times 50)$ randomly generated sparse source matrix by a square mixing matrix $\boldsymbol{A} \in \mathbb{R}^{5\times 5}$ randomly generated from a white normal distribution with zero mean and unit variance. Each row of the sparse source matrix $\boldsymbol{S}$ is constructed such that it has exactly 8 nonzero entries. The indices of these 8 entries are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm 1$. The sources and the measurements are shown in Figure 5.3(a) and (b), respectively. The estimated sources using

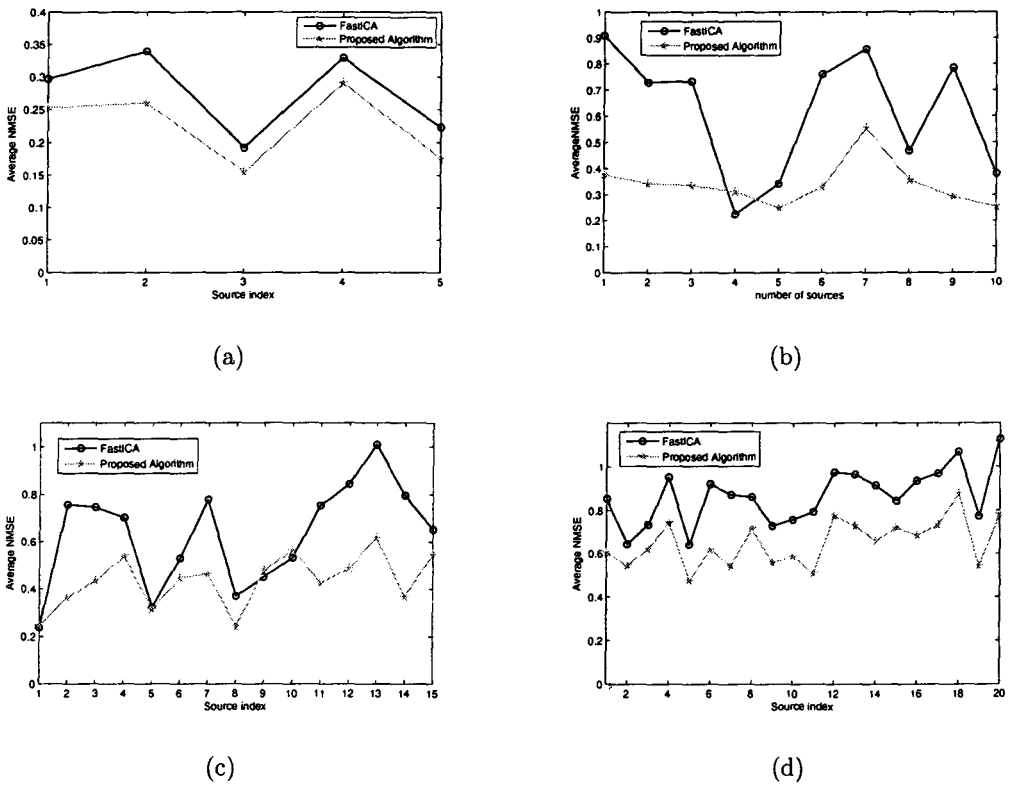



Figure 5.4: The average NMSE between the estimated sources and the original ones. The $x$ axis represents the indices of the original sources, and the $y$ axis represents the average NMSE over 200 different runs corresponding to 200 different selections of the square mixing matrix. The figures (a)–(d) correspond to the cases $n = 5$, 10, 15, and 20, respectively.

the proposed algorithm and the FastICA algorithm are shown in Figure 5.3(c) and Figure 5.3(d), respectively. As shown in these two figures, the original sources are estimated correctly using the proposed algorithm, while the estimated sources using FastICA are not sparse enough.

Figure 5.4 provides a comparison between the proposed BSSE algorithm and the FastICA algorithm in terms of the average NMSE and the size of the BSS problem. In this figure, we consider the following four different values of the number of sources, $n = 5, 10, 15$, and 20, respectively. For each value of $n$, a sparse $(n \times 100)$ source matrix is generated, and 200 different square mixing matrices are generated. For

each one of the mixing matrices, the BSS problem is solved using the proposed BSSE algorithm and the FastICA algorithm. The NMSE between each source and the corresponding estimate is calculated, and the average over the 200 different trials is plotted in Figure 5.4. As shown in this figure, and for each source in the four different cases, the average NMSE for the proposed algorithm is smaller than that of the FastICA algorithm. Accordingly, the proposed algorithm can estimate sparse sources more accurately than the FastICA algorithm.

### Example 2: Comparison between the tcSCA, scSCA, and sucSCA algorithms

In this example we compare the performance of the three proposed cSCA algorithms in estimating a desired sparse source signal. The comparison is done in terms of the average NMSE (between the desired and the estimated source signals) and the average convergence time. The experiment is performed for the cases $n = 4, 6, \ldots, 20$. For each value of $n$, a sparse source matrix $\boldsymbol{S} \in \mathbb{R}^{n \times 1000}$ is generated such that 30% of its columns satisfy the disjoint orthogonality property. For each row of $\boldsymbol{S}$, the indices of the nonzero entries are randomly selected, and their amplitudes are chosen from a uniform distribution between $\pm 1$. The measurement matrix $\boldsymbol{X} \in \mathbb{R}^{n \times 1000}$ is constructed by multiplying the source matrix $\boldsymbol{S}$ by a square mixing matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ randomly generated from a white normal distribution with zero mean and unit variance. The first source signal $\boldsymbol{s}^1$, the first row of $\boldsymbol{S}$, is selected as the desired source signal.

The reference vector $\bar{\boldsymbol{r}}_t$ for the tcSCA algorithm is constructed as $\bar{\boldsymbol{r}}_t = \boldsymbol{s}^1 + \bar{\boldsymbol{v}}$, where $\bar{\boldsymbol{v}}$ is a zero mean Gaussian random noise. The amplitude of $\bar{\boldsymbol{v}}$ is adjusted to produce 0 dB SNR. The reference vector $\bar{\boldsymbol{r}}_{su}$ for the sucSCA algorithm is selected as $\bar{\boldsymbol{r}}_{su} = \text{sign}\left(|\boldsymbol{s}^1|\right)$. The reference vector $\boldsymbol{r}_s$ for the scSCA algorithm is constructed as $\boldsymbol{r}_s = \text{sign}\left(\boldsymbol{a}_1\right)$, where $\boldsymbol{a}_1$ is the mixing column associated with the desired source

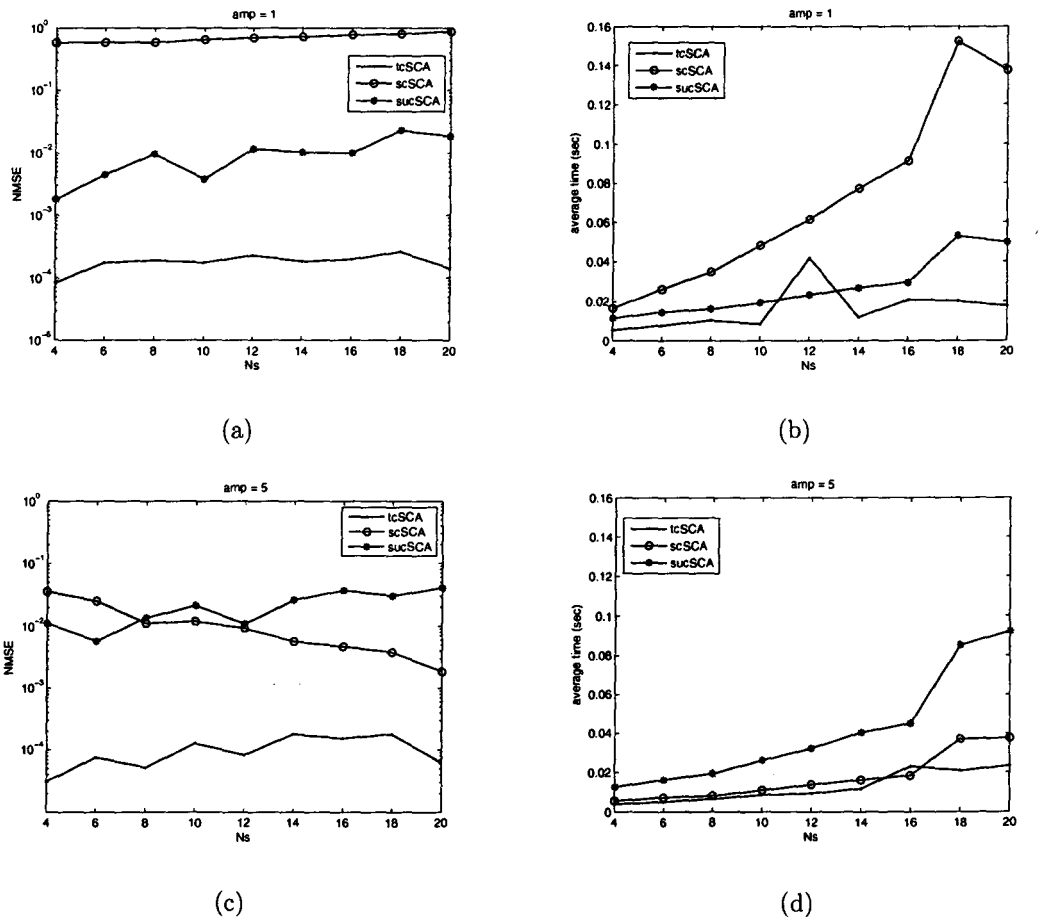
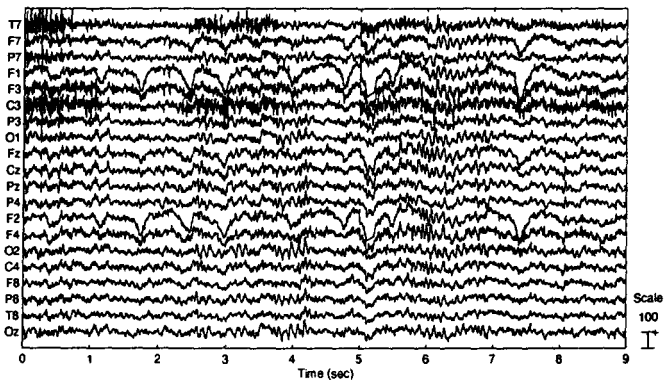



Figure 5.5: Comparison between the tcSCA, scSCA, and sucSCA algorithms. (a)-(b) The average NMSE and the average convergence time, respectively, for the case of all sources have the same amplitude, (c)-(d) The average NMSE and the average convergence time, respectively, for the case when the amplitude of the desired source signal is 5 times larger than that of the other source signals.

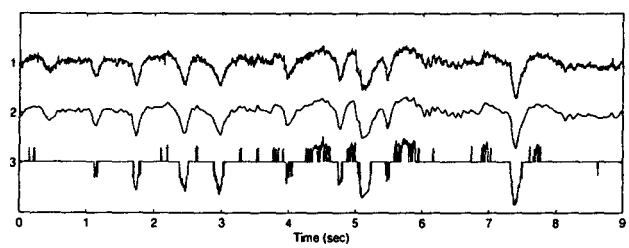
signal.

For each value of $n$, the average NMSE and the average convergence time are calculated as the average over 1000 different runs, where in each run a new mixing matrix and a new sparse source matrix are generated. The results are shown in Figure 5.5(a)-(b). As shown in these two figures, the tcSCA algorithm has the best performance in terms of the average NMSE and the average convergence time, whereas the scSCA algorithm has the worst performance. This result is expected since the reference vector used with the tcSCA algorithm contains more information about the desired source signal than the other two algorithms. On the other hand, the poorer performance of the scSCA algorithm is due to the following factors: 1) the scSCA algorithm does not have any temporal information about the desired source signal, 2) all the source signals are sparse and have the same sparsity, and 3) there could be more than one column of the mixing matrix that has the same sign pattern as the reference vector $\boldsymbol{r}_s$. Therefore, for the scSCA algorithm to produce satisfactory results, the desired source signal must be distinct (in terms of sparsity, amplitude, or sign pattern of the associated mixing column) from the other source signals.

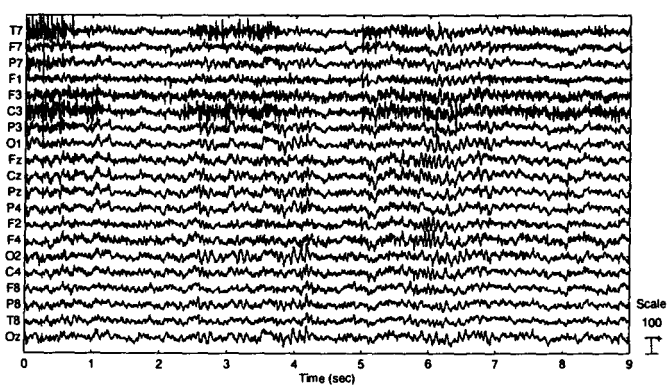
To check the effect of the amplitude of the desired source signal on the performance of the scSCA algorithm, the experiment is repeated with the amplitude of the desired source signal 5 times larger than that of the other source signals. The results of this new experiment are shown in Figure 5.5(c)-(d). As shown in these figures, the performance of the scSCA algorithm is remarkably enhanced. Similar performance enhancement is expected if the desired source signal is the sparsest source signal, or if the sign pattern of the mixing column associated with the desired source signal is distinct from that of the other mixing columns.




(a)

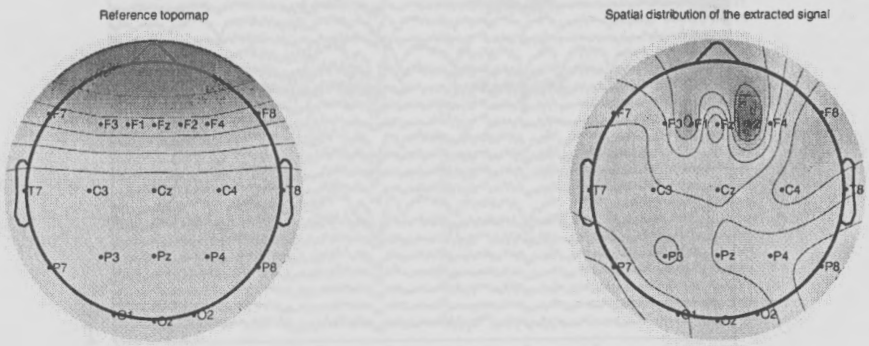



(b)

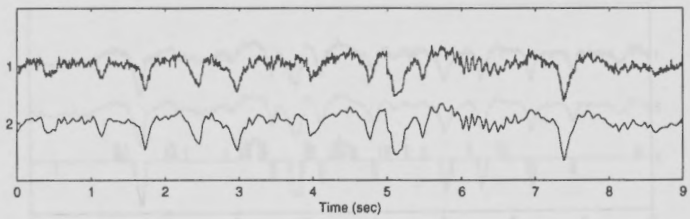



(c)

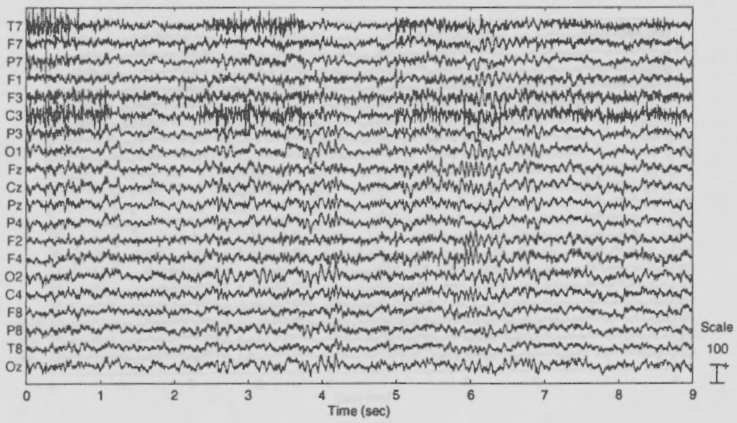
Figure 5.6: Automatic removal of eye blink artifact from real EEG data using the tcSCA algorithm. (a) Original EEG data, (b) The signals from top to bottom correspond to the extracted eye blink artifact, the denoised eye blink artifact, and the reference signal, respectively, (c) Clean version of the EEG data shown in (a).




(a) (b)




(c)




(d)

Figure 5.7: Automatic removal of eye blink artifact from real EEG data using the scSCA algorithm. (a) Reference topographic map, (b) The topographic map of the extracted artifact, (c) The upper signal correspond to the extracted eye blink, while the lower signal is the denoised eye blink artifact, (d) Clean version of the EEG data shown in Figure 5.6(a)

### Example 3: Rejection of eye blink artifact from real EEG data

In this example we demonstrate the ability of the tcSCA algorithm and the scSCA algorithm in extracting the eye blink artifact contaminating real EEG data shown in Figure 5.6(a). Removing the eye blink artifact from a recorded EEG data using ICA is usually done in three steps. In the first step, called the *processing* step, the ICA algorithm is applied on the EEG data and an estimate of the independent components and the associated mixing columns is obtained. In the second step, which is called the *investigation* step, both the estimated mixing matrix and the estimated components are carefully investigated to identify the components corresponding to the eye blink artifact. The decision is usually made based on the waveform of the suspected component and its mixing column which is usually depicted by a topographic map that represents the contribution of the eye blink artifact at each electrode. If the topographic map and the waveform of one of the components correspond to those of the eye blink artifact, this component is marked as the desired artifact. Finally, in the third step the component corresponding to the eye blink artifact is set to zero and the all the remaining components are recombined to reconstruct the clean EEG data.

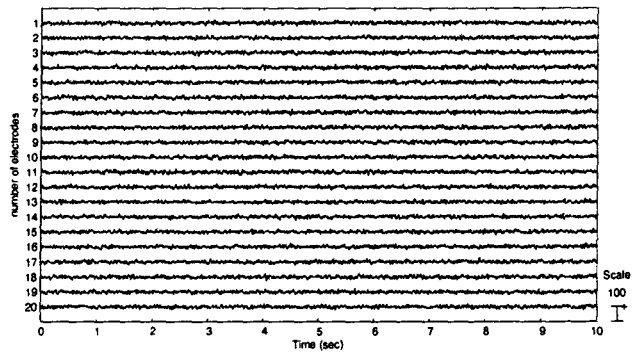
Clearly the most tedious step in the aforementioned procedure is the investigation step, during which all the sources and their associated maps are checked carefully to identify the artifact. This process is difficult to automate. In this example we show that the second step can be completely eliminated by using either the tcSCA algorithm or the scSCA algorithm for extracting only the desired source signal. In the tcSCA algorithm we use a reference signal which conveys temporal information about the eye blink artifact. This reference signal is extracted from one of the frontal electrodes, say F1, by setting to zero all samples with absolute value less than a threshold of 50 $\mu v$. This reference signal is shown by the third waveform in Figure 5.6(b). On the other hand, the reference vector for the scSCA algorithm is an $(n \times 1)$

vector of zeros except for those indices corresponding to the frontal electrodes, for which the reference vector equals one. The topographic map of this reference vector is shown in Figure 5.7(a).

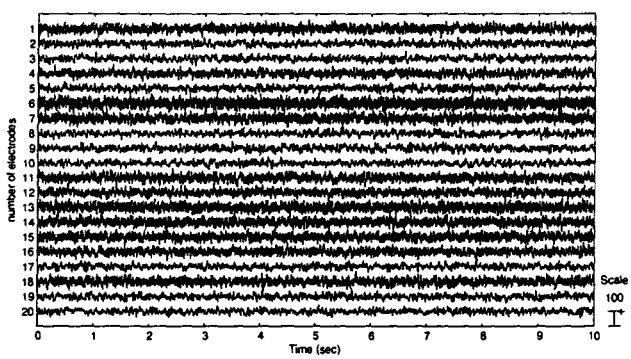
After running the tcSCA algorithm and the scSCA algorithm on the EEG data shown in Figure 5.6(a), each with its own reference vector, both algorithms converge successfully to the desired artifact signal. The extracted signals are shown by the first waveform in Figure 5.6(b) and Figure 5.7(c), respectively. The second waveform in each one of these figures represents a denoised version of the estimated artifact, where the denoising step is performed using wavelet analysis. Figure 5.7(b) shows the topographic map of the extracted source using scSCA algorithm. Clearly this topographic map corresponds to the expected topographic map of the eye blink artifact, for which F1 and F2 are the most contaminated electrodes. Finally, the clean EEG data is obtained by removing the denoised signals from the original EEG data using the technique mentioned in Section 5.2.2. The cleaned EEG data obtained after running the tcSCA algorithm and the scSCA algorithm are shown in Figure 5.6(c) and Figure 5.7(d), respectively. Clearly the eye blink artifact is removed successfully in both cases.

### Example 4: Elimination of a line voltage interference from a recorded EEG data using the sucSCA algorithm

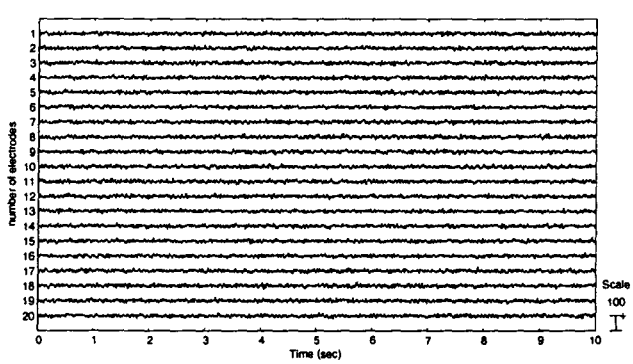
In this example we present an application for which the sucSCA algorithm is appropriate. In the field of EEG signal processing, the measured signal is usually contaminated by 50/60 Hz line voltage. This line voltage interference is usually removed from the measured EEG data using a notch filter with the correct center frequency. However, this filtering approach is not appropriate if a useful brain signal occupies a frequency band around 50/60 Hz. To overcome this difficulty, we propose utilizing the sucSCA algorithm for extracting the line voltage interference in the frequency domain. Note




(a)




(b)




(c)

Figure 5.8: (a) Simulated EEG data (b) The simulated EEG data contaminated with a 60 Hz line voltage interference, (c) Cleaned EEG data using the sucSCA algorithm.

that, the sucSCA algorithm, rather than the tcSCA algorithm, is used in this example for removing the line voltage interference because the phase of the interference sinusoidal signal is generally unknown.

The EEG data utilized in this example was generated as follows. First, a synthesized EEG data matrix $\tilde{\boldsymbol{X}}$, which corresponds to the background brain activity, and a 60 Hz sinusoidal signal $\bar{\boldsymbol{s}}$ were generated. The phase of the generated sinusoidal signal was chosen randomly. Then the overall EEG data matrix $\boldsymbol{X}$ was generated as $\boldsymbol{X} = \boldsymbol{a}\bar{\boldsymbol{s}} + \tilde{\boldsymbol{X}}$, where $\boldsymbol{a}$ is a mixing column randomly generated from a white normal distribution with zero mean and unit variance. The simulated EEG data $\tilde{\boldsymbol{X}}$ and the contaminated data $\boldsymbol{X}$ are shown in Figure 5.8(a) and (b), respectively. As shown in Figure 5.8(b), the EEG data is heavily contaminated by the 60 Hz line voltage interference.

Since the 60 Hz interference signal is not sparse in the time domain but very sparse in the frequency domain, all the analysis is performed in the frequency domain. Let $\boldsymbol{X}(f)$ denote the discrete time Fourier transform of the original EEG data matrix $\boldsymbol{X}$. The input data matrix to the sucSCA algorithm is constructed by concatenating the real and the imaginary parts of $\boldsymbol{X}(f)$. The reference signal to the sucSCA algorithm is constructed in the following steps. First a time domain reference signal $r(t) = \sin(2\pi 60t)$ is generated. Then this signal is transformed into the frequency domain using the fast Fourier transform, i.e., $\bar{\boldsymbol{r}}(f) = \text{fft}\,(r(t))$. Finally, the reference vector for the sucSCA algorithm is constructed as $\bar{\boldsymbol{r}}_{su} = \text{abs}\,([\text{real}(\bar{\boldsymbol{r}}(f)) \;\; \text{imag}(\bar{\boldsymbol{r}}(f))])$. After extracting the interference signal using the sucSCA algorithm, the extracted signal is removed from the EEG data matrix using the technique presented in Section 5.2.2. The cleaned EEG data is shown in Figure 5.8(c). Comparing the cleaned EEG data with the original EEG data shown in Figure 5.8(a), it is clear that the 60 Hz line voltage interference is successfully removed from the EEG data using the sucSCA algorithm. On the other hand, the signal to interference ratio (SIR) between the

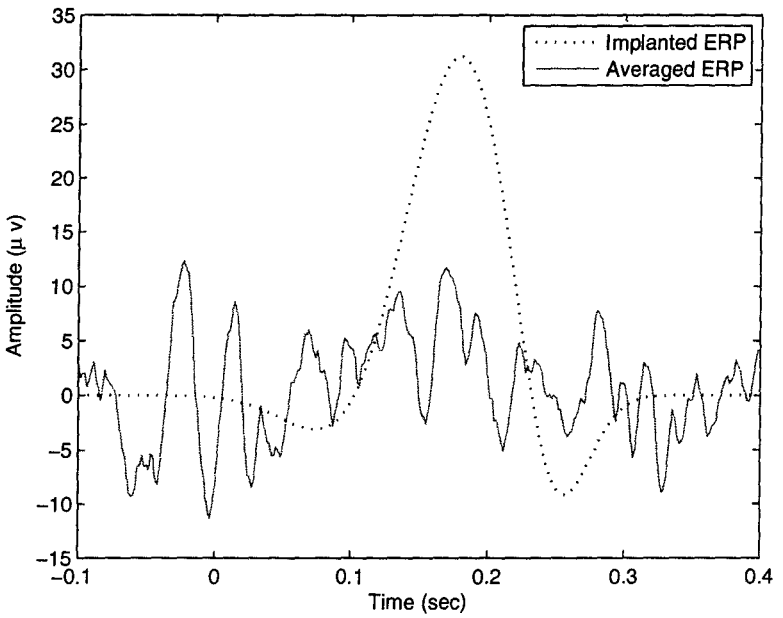
original EEG data and the contaminated EEG data is -6.3461 dB, while the SIR between the original EEG data and the cleaned EEG data is 12.7489 dB, i.e., the improvement in the SIR is 19.1 dB.

### Example 5: Estimation of the ERP signal from a limited number of trials

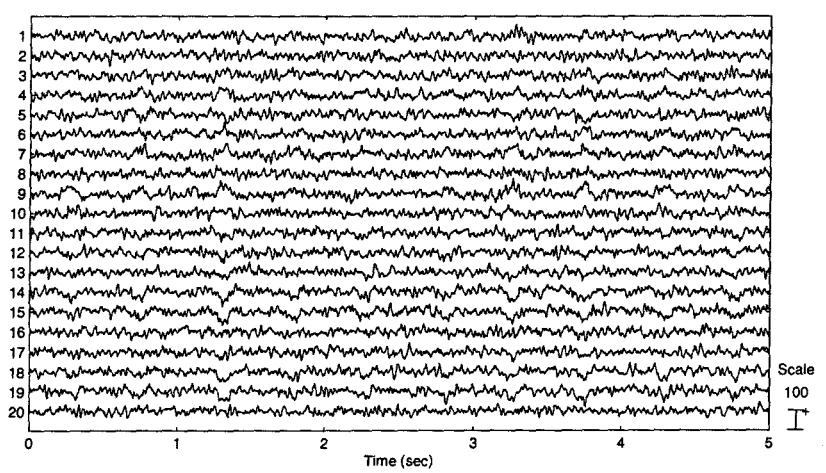
The *event related potential* (ERP) signal is defined as the brain activity in response to an external stimulus such as a sound. The ERP signal is phase–locked to the stimulus onset, and its signal to noise ratio (SNR) relative to the background brain activity is low. The conventional approach for estimating the ERP signal is to average over a large number of trials of the measured EEG data. This approach is motivated by the fact that the background brain activity is not phase locked to the stimulus onset, and that the ERP waveform is independent of the background brain activity. Hence, if the number of trials is sufficiently large, the background EEG will be suppressed in the averaged signal. This averaging approach depends also on the assumption that the amplitude and the latency of the ERP signal do not change from trial to trial. However, there is evidence to suggest that the ERP signal is associated with a random amplitude and a random delay at each trial [124–127]. Accordingly, by following the averaging procedure, the amplitude and latency variations of the ERP signal between different trials will distort the estimated ERP signal.

Accurate estimation of the amplitude and the latency parameters of the ERP signal in each individual trial is particularly important for clinical applications such as the diagnosis of possible brain injury or disorders in the central nervous system [124] because these measures indicate how the brain processes the presented stimuli [128]. Furthermore, with current techniques that largely involve averaging across trials, it is not possible to study fast learning phenomena, which could involve changes from trial to trial in both latency and amplitude.

Another drawback associated with the averaging approach is that the accuracy of




(a)




(b)

Figure 5.9: (a) Template for the implanted ERP signal (dotted curve) and the averaged ERP signal at the first electrode of the generated EEG data (continuous curve), (b) The constructed EEG data for Example 3.

the estimated ERP signal depends on the number of the averaged trials. The larger the number of trials is, the better the estimation of the ERP signal. However, there are many situations at which the number of available clean trails is very small. For example, for the case of infant's EEG data, the recording time is usually short, and the recorded data is usually contaminated with very large movement artifacts, which limits the number of clean trials.

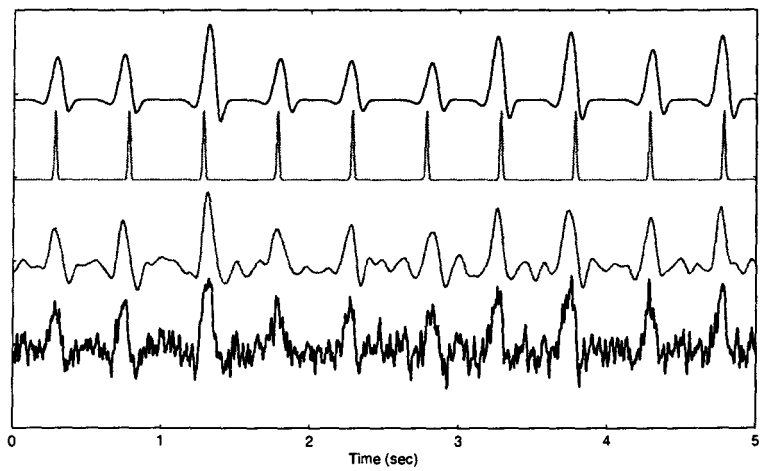
Based on this introduction, it is clear that there is a need to track the variation of the ERP signal from trial to trial, and also to estimate the ERP signal from a limited number of trials. In the remaining part of this example we will show that both the tcSCA and the scSCA algorithms can estimate the waveform including the amplitude and latency variations of the ERP signal at each individual trial, even when the number of trials is limited. This enables a more accurate estimation of the ERP signal to be made.

Towards that end, a synthesized EEG data matrix $\tilde{\boldsymbol{X}}$, which corresponds to the background brain activity, is generated. The number of rows of this EEG data matrix is 20, and the number of columns is 3100, which corresponds to 10 trials each of which has a duration of 0.5 seconds. For constructing a varying ERP signal, a single trial ERP signal is generated and used as a template for the other trials. This template signal is shown as the dotted signal in Figure 5.9(a). The overall ERP signal $\bar{\boldsymbol{s}}$ is then constructed by concatenating 10 different templates of the ERP signal, where each of them is associated with random amplitude and random delay. The waveform of this ERP signal is shown by the upper waveform in Figure 5.10(a) or Figure 5.10(b). The overall data matrix $\boldsymbol{X}$, which corresponds to the measured EEG data matrix, is generated as $\boldsymbol{X} = \boldsymbol{a}\bar{\boldsymbol{s}} + \tilde{\boldsymbol{X}}$, where $\boldsymbol{a}$ is a mixing column with random entries that are uniformly distributed between $\pm 0.9$. The mixing vector $\boldsymbol{a}$ corresponds to the topographic map of the ERP signal. The constructed EEG data is shown in Figure 5.9(b). As shown in this figure, the implanted ERP signal is invisible in any channel.

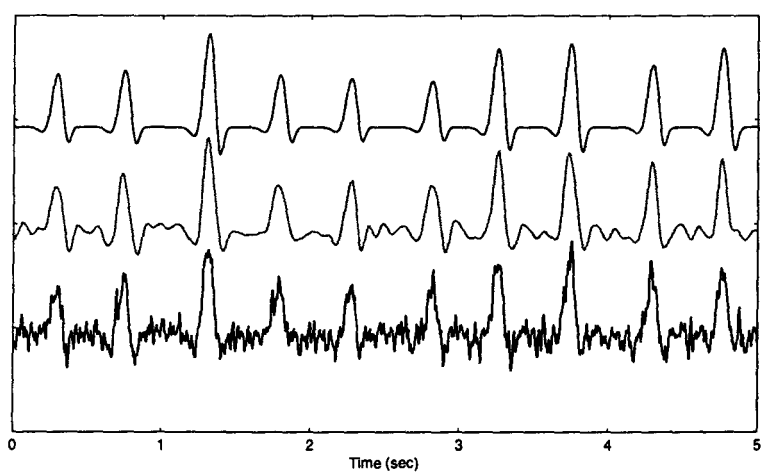
For the ERP signal $\bar{\boldsymbol{s}}$ to be estimated from the measured EEG data matrix $\boldsymbol{X}$ using either the tcSCA algorithm or the scSCA algorithm, an appropriate reference vector must be generated. For the tcSCA algorithm, the reference vector must convey information about the waveform and the timing of the ERP signal in each trial. Assume that the average latency of the peak of the ERP signal in each trial is known, which is usually the case (in the presented example, this time is 0.18 seconds after the stimulus onset). Then, for the presented example, a reference signal for the ERP signal $\bar{\boldsymbol{s}}$ can be generated by generating a train of pulses each of which has a *fixed* amplitude and a *fixed* latency equal to 0.18 seconds after the stimulus onset. The value of the amplitude parameter of this reference signal is not critical, however, its latency (or position in time) is very important. The waveform of the reference signal is presented by the second waveform in Figure 5.10(a).

On the other hand, for estimating the ERP signal using the scSCA algorithm, a reference vector for the mixing vector $\boldsymbol{a}$ must be generated. This vector can be constructed by incorporating prior information about the topographic map of the ERP signal. For example, if the amplitude parameter of the ERP signal is known a priori to be positive over the frontal electrodes and negative over the occipital electrodes, the entries of the reference vector in this case can be selected as the sign of the amplitude parameter of the ERP signal at each electrode. We followed this procedure in the presented example, i.e, we selected $\boldsymbol{r}_s = \text{sign}(\boldsymbol{a})$. The results of the tcSCA algorithm and the scSCA algorithm are shown in Figure 5.10(a) and (b), respectively. The estimated ERP signal in each case is presented by the lower waveform in the corresponding figure. A denoised version of the estimated ERP signal is obtained by performing a wavelet analysis and is shown in the third and second waveforms in Figure 5.10(a) and (b), respectively.

As shown in these two figures, the ERP signal is estimated correctly, and the variations of the amplitude and the delay parameters are preserved in the estimated




(a)




(b)

Figure 5.10: (a) Different ERP signals associated with the tcSCA algorithm. The signals top to bottom are the original ERP signal, the template signal, the denoised version of the estimated ERP signal, and the estimated ERP signal, respectively. (b) Different ERP signals associated with the scSCA algorithm. The signals from top to bottom are the original ERP signal, the denoised version of the estimated ERP signal, and the estimated ERP signal, respectively.

signals. On the other hand, the continuous waveform shown in Figure 5.9(a) presents the ERP signal obtained by averaging trials over the first row in the data matrix $\boldsymbol{X}$. As shown in this figure, the SNR of the averaged ERP signal is very low, and the averaged signal is, unlike those obtained by the proposed method, severely distorted. This is a direct result of the limited number of trials and the random variation in the amplitude and the delay parameters of the ERP signal in each trial.

## 5.5 Conclusion

In this chapter we proposed four novel algorithms for solving the problem of BSE when the desired source signals are *sparse*. In all these algorithms, a nonconvex objective function is utilized for measuring the sparsity of the extracted source signal. The first algorithm is based on sequentially extracting sparse source signals, where the sparsity of the desired source signals is the only prior information utilized for estimating the separating vectors. Simulation results show that the proposed algorithm can extract sparse source signals with relatively smaller residual error than the FastICA algorithm.

On the other hand, the other proposed algorithms are based on incorporating prior information about the desired signal, in addition to its sparsity, into the optimization problem designed for extracting the desired source signal. This class of algorithms was called constrained SCA (cSCA). The first algorithm in this class is based on utilizing a reference signal that conveys *temporal* information about the desired source signals. The proposed algorithm has two different versions, depending on the measure of closeness between the extracted source signal and the reference signal. The second algorithm in this class is based on utilizing a reference signal that conveys information about the *support* of the desired sparse source signal. This algorithm is useful when there is an ambiguity associated with the sign of each sample of the desired source

signal. This algorithm was utilized successfully in removing the (50/60 Hz) line voltage interference from a simulated EEG data. The last algorithm that we proposed in this chapter can extract the desired sparse source signal when prior information about its mixing column is available. Previous approaches for solving this problem can not estimate a single source signal, and are based on the ICA technique, hence does not enforce the sparsity of the desired source signal. Simulation results show that the three algorithms can be successfully used for removing different kind of artifacts from real EEG data and for estimating the ERP signal from synthesized EEG data.

# Chapter 6

# Conclusion and future work

In this thesis we discussed the utilization of the sparsity concept in solving three challenging problems. The first problem that we have discussed is finding a unique solution to an under–determined linear system of equations in which the number of equations is less than the number of unknowns. We showed that a unique solution vector can be obtained if it is sparse or can be sparsely represented under a suitable linear transformation. In this regard, we developed a new methodology for minimizing a class of nonconvex objective functions that measure the diversity (anti–sparsity) of the solution vector. The proposed technique is based on locally replacing the original objective function by a quadratic convex function which is easily minimized. We showed that, for a certain selection of the convex objective function, the class of algorithms called Iterative Re–weighted Least Squares (IRLS) can be derived from the proposed methodology. Thus the proposed algorithms are a generalization and unification of the previous methods. In this thesis we also proposed a convex objective function that produced an algorithm that can converge to the sparse solution vector in a significantly fewer number of iterations than the IRLS algorithms.

The second problem that we have discussed in this thesis is finding a unique solution to the under–determined BSS problem, in which the number of sources is greater

than the number of sensors. The difficulty associated with this problem is that both the mixing matrix and the source signals are unknowns. Moreover, in this thesis, we considered the more practical case in which the number of sources is also *unknown*. The under–determined BSS problem is usually solved by following the two–step SCA approach, in which the mixing matrix is estimated first via clustering the columns of the measurements matrix, then the sources are estimated using a compressed sensing technique, e.g., an IRLS algorithm. In this regard, we developed three novel different clustering algorithms that are robust to the presence of noisy and/or outlier measurements. We have also suggested two parameters, called *concentration parameters* (CP), for identifying the number of sources from the estimated clusters. Combining the clustering algorithms with the CPs, both the mixing matrix and the number of sources have been estimated with high accuracy compared with some clustering algorithms that are usually used for solving the same problem.

The third problem that we have discussed in this thesis is extracting a sparse source signal from a linear mixture of unknown source signals. We also considered the case in which some prior information is available about the desired source signal in addition to its sparsity. In this regard we developed four novel algorithms. The first algorithm is based on sequentially extracting sparse source signals, while the remaining three algorithms utilize a priori available information to extract a desired source signal. Numerical simulations show that the proposed algorithms can be successfully used for removing different kind of artifacts from real electroencephalographic (EEG) data and for estimating the event related potential (ERP) signal from synthesized EEG data.

## 6.1 Suggestions for future research

The following research points can be considered in the near future:

1. Developing the MCCR algorithm to the case of noisy measurements.

2. Developing the MCCR algorithm for the case when the sparse solution vector has a certain structure. This has a practical application in the field of brain source localization. For brain source localization, it is known that the brain source that generates the measured ERP signal is confined in a certain region in the brain, which could be unknown. Accordingly, by constraining the solution vector to be *block-sparse*, i.e., the nonzero entries are confined in separate blocks, better estimate of the brain source that generates the ERP signal can be obtained.

3. The PMCCR algorithm produced very promising result in estimating a sparse solution vector from few number of measurements. However, the computational cost of the PMCCR algorithm inhibits it from a practical use, especially for large scale problems. Adjusting the perturbation process in the PMCCR algorithm in a controlled manner, rather than random perturbation, is a topic of future research.

4. Developing a new algorithm for building dictionary matrices that produce better sparse representation of certain class of signals, e.g., sound signals or EEG signals, than the existing general purpose dictionaries, e.g., the vavelet related dictionaries.

5. Generalizing the BSSE algorithm, which is developed in Chapter 5, to the underdetermined BSS problem.

# Bibliography

[1] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," IEEE Trans. Signal Process., vol. 41, no. 12, pp. 33973415, Dec. 1993.

[2] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation, Part I: Greedy pursuit," Signal Process., vol. 86, no. 3, pp. 572588, 2006.

[3] D. Studer, U. Hoffmann, and T. Koenig, "From EEG dependency multichannel matching pursuit to sparse topographic EEG decomposition," J. Neuroscience Methods, vol. 153, pp. 261-275, 2006.

[4] K. Skretting and J. H. Husy, "Partial search vector selection for sparse signal representation," in: NORSIG-03, Bergen, Norway, October 2003, available at: http://www.ux.his.no/~karlsk/

[5] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in Proc. 27th Annu. Asilomar Conf. Signals Syst. Comput., Nov. 13, 1993, vol. 1, pp. 4044.

[6] S. Chen and D. Donoho, "Basis pursuit," in Proc. Twenty-Eighth Asilomar Conf. Signals, Syst. Comput., Monterey, CA, Nov. 1994, vol. I, pp. 41-44.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunder, "Atomic decomposition by basis pursuit," SIAM J. Sci. Comput., vol. 20, no. 1, pp. 3361, 1998.

[8] K. K. Herrity,A. C. Gilbert, and J. A. Tropp, Sparse approximation via iterative thresholding, in Proc. IEEE Int.Conf. Acoust. Speech Signal Process., Toulouse, France, May 1419, 2006, vol. III, pp. 624627.

[9] T. Blumensath, M. E. Davies, "Iterative Thresholding for Sparse Approximations," The Journal of Fourier Analysis and Applications, vol.14, no 5, pp. 629-654, Dec. 2008

[10] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstructions from limited data using FOCUSS: A re-weighted minimum norm algorithm," IEEE Trans. Signal Processing, vol. 45, pp. 600-616, Mar. 1997.

[11] P. Rodríguez and B. Wohlberg, "An iterative reweighted norm algorithm for total variation regularization," IEEE Signal Processing Letters, vol. 14, no. 12, pp. 948–951, Dec 2007.

[12] B. D. Rao and K. Kreutz-Delgado, "An affine scaling methodology for best basis selection," IEEE Trans. Signal Process., vol. 47, no. 1, Jan. 1999.

[13] R. Chartrand and W. Yin, "Iteratively Reweighted Algorithms for Compressive Sensing," in Proc. Acoustics, Speech and Signal Processing, ICASSP 2008, pp. 3869-3872.

[14] I. Daubechies, R. DeVore, M. Fornasier, and S. Gunturk, " Iteratively reweighted least squares minimization for sparse recovery," to appear in Commun. Pure Appl. Math. 2009.

[15] B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado, "Subset selection in noise based on diversity measure minimization," IEEE Trans. Signal Process., vol. 51, no. 3, pp. 760-770, Mar. 2003.

[16] B. D. Rao and K. Kreutz-Delgado, "Basis selection in the presence of noise," in Proc. Signals, Systems & Computers, 1998, vol. 1, pp. 752-756

[17] S. F. Cotter, B. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," IEEE Trans. Signal Process., vol. 53, no. 7, pp. 24772488, Jul. 2005.

[18] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted $\ell_1$ minimization," Journal of Fourier Analysis and Applications, vol. 14, no. 5, pp. 877-905, special issue on sparsity, December 2008.

[19] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," IEEE Trans. Acoust., Speech, Signal Processing, vol. 41, pp. 33973415, Dec. 1993.

[20] D. Malioutov, M. Çetin, and A. Willsky, "Optimal sparse representations in general overcomplete bases," IEEE Int. Conf. Acoustics, Speech, and Signal Processing, vol. 2, pp. II.793.796, May 2004.

[21] G. H. Golub and C. F. Van Loan, "Matrix Computations." Baltimore: The Johns Hopkins University Press, 1989.

[22] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, "Learning unions of orthonormal bases with thresholded singular value decomposition," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Philadelphia, PA, Mar. 1823, 2005, vol. 5, pp. v/293v/296.

[23] R. Tibshirani, "Regression shrinkage and selection via the lasso," J. Roy. Statist. Soc. B., vol. 58, pp. 267288, 1996.

[24] T. Simila, "Majorize-minimize algorithm for multiresponse sparse regression," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process, Honolulu, HI, 2007, vol. II, pp. II-553II-556.

[25] Y. Li, A. Cichocki, and S. Amari, "Sparse Component Analysis for Blind Source Separation With Less Sensors Than Sources," in Proc. 4th Int. Symp. Independent Component Analysis Blind Source Separation (ICA BSS), 2003, pp. 89-94.

[26] P. Georgiev, F.Theis, and A. Cichocki, "Blind Source Separation and Sparse Component Analysis of Overcomplete Mixtures," in Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP2004), Vol. V, (Montreal, Canada), May 2004, pp. 493-496.

[27] P. Georgiev, F. Theis, A. Cichocki, H. Bakardjian, "Sparse Component Analysis: A New Tools for Data Mining", Data Mining in Biomedicine, Springer, New York, USA, 2005

[28] M. Zibulevsky, and B.A. Pealmutter, "Blind Source Separation by Sparse Decomposition in a Signal Dictionary," in Neural Comp., Vol. 13, pp. 863-882, 2001.

[29] M. Zibulevsky, and B.A. Pealmutter, "Blind Source Separation by Sparse Decomposition," in Independent Component Analysis: Principles and Practice, S.J. Roberts and R.M. Everson, Eds. Cambridge, U.K.: Cambridge Univ., Press 2001.

[30] T.W. Lee, M.S. Lewicki, and T.J. Sejnowski, "Blind Source Separation of More Sources Than Mixtures Using Overcomplete Representations," IEEE Signal Process. Let., Vol. 6, No. 4, pp. 87-90, 1999.

[31] Q. Lv, and X. Zhang, "A Unified Method for Blind Separation of Sparse Sources With Unknown Source Number," IEEE Signal Process. Let., Vol. 13, No. 1, pp. 49-51, Jan. 2006.

[32] Y. Li, A. Cichocki, and S. Amari, "Blind Estimation of Channel Parameters and Source Components for EEG Signals: A Sparse Factorization Approach," IEEE Trans. Neurral Net., Vol. 17, No. 2, pp. 419-431, Mar. 2006.

[33] Y. Li, A. Cichocki, S. Amari, S. Shishkin, J. Cao, and F. Gu, "Sparse Representation and Its Applications in Blind Source Separation," in 7th Annual Conference on Neural Information Processing Systems (NIPS-2003), (Vancouver), Dec. 2003.

[34] C. Im, H. Jung, K. Jung, and S. Y. Lee, "Reconstruction of continuous and focalized brain functional source images from electroencephalogaphy," IEEE Trans. Magnetics, vol. 43, no. 4, pp. 1709-1712, Apr. 2007.

[35] J. W. Phillips, R. M. Leahy, and J. C. Mosher, "Meg-based imaging of focal neuronal current sources," IEEE Trans. Med. Imag., vol. 16, no. 3, pp. 338348, Jun. 1997.

[36] J. Han and K. S. Park, "Regularized FOCUSS algorithm for EEG/MEG source imaging," in Proc. 26th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc., San Francisco, CA, 1993, vol. 1, pp. 122124.

[37] P. Xu, Y. Tian, H. Chen, and D. Yao, "LP norm iterative sparse solution for EEG source localization," IEEE Trans. Biomed. Eng., vol. 54, no. 3, pp. 400409, Mar. 2007.

[38] M. E. Davies and L. Daudet, "Sparsifying subband decompositions," in Proc. IEEE Workshop Appl. Signal Process. Audio Acoust., 2003, pp. 107110.

[39] M. E. Davies and L. Daudet, "Sparse audio representations using the MCLT," Signal Process., vol. 86, no. 3, pp. 457470, 2006.

[40] K. Kreutz-Delgado and B. D. Rao, "FOCUSS-based dictionary learning algorithms," in Proc. Wavelet Appl. Signal Image Process., Bellingham, WA, Jul.-Aug. 2000, vol. 4119, pp. 459473.

[41] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T. Lee, and T. Sejnowski, "Dictionary learning algorithms for sparse representation," Neural Comput., vol. 15, pp. 349396, 2003.

[42] M. Aharon, M. Elad, and A. Bruckstein, "The K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," IEEE Trans. Signal Process., vol. 54, no. 11, pp. 43114322, Nov. 2006.

[43] Takhar D, Laska J N, Wakin M B, Duarte M F, Baron D, Sarvotham S, Kelly K F and Baraniuk R G 2006 A new compressive imaging camera architecture using optical-domain compression Computational Imaging IV - Proceedings of SPIE-IS and T Electronic Imaging vol 6065

[44] B. D. Jeffs, "Sparse inverse solution methods for signal and image processing applications," in Proc. ICASSP, vol. III, Seattle,WA, May 1998, pp. 18851888.

[45] I. J. Fevrier, S. B. Gelfand, and M. P. Fitz, "Reduced complexity decision feedback equalization for multipath channels with large delay spreads," IEEE Trans. Commun., vol. 47, no. 6, pp. 927937, Jun. 1999.

[46] D. L. Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancelers," IEEE Trans. Speech Audio Process., vol. 8, no. 5, pp. 508518, Sep. 2000.

[47] B. Jeffs and M. Gunsay, "Restoration of blurred star field images by maximally sparse optimization," IEEE Trans. Image Process., vol. 2, no. 2, pp. 202211, Mar. 1993.

[48] T. Adeyemi and M. Davies, "Sparse representation of images using overcomplete complex wavelets," in Proc. Statistical Signal Processing, pp. 865-870, Jul. 2005.

[49] D. Takhar, V. Bansal, M. Wakin, M. Duarte, D. Baron, J. Laska, K.F. Kelly, and R.G. Baraniuk, "A compressed sensing camera: New theory and an implementation using digital micromirrors," in Proc. Comput. Imaging IV SPIE Electronic Imaging, San Jose, Jan. 2006.

[50] P. D. O'Grady, B. A. Pearlmutter, and S. T. Rickard, "Survey of sparse and non-sparse methods in source separation," International Journal of Imaging Systems and Technology (IJIST), vol. 15, pp. 18-33, 2005.

[51] A.J. Bell, and T.J. Sejnowski, " An information-maximization approach to blind source separation and blind deconvolution," Neural Comp., vol. 7, no. 6, pp. 1129-1159, 1995.

[52] A. Hyvrinen, "Fast and robust fixed-point algorithms for independent component analyis," IEEE Trans. Neural Net., vol. 10, no. 3, pp. 626-634, 1999.

[53] A. Hyvrinen and E. Oja, "Independent component analyis: algorithms and applications" Neural Networks, vol. 13, no. (4-5), pp. 411-430, 2000.

[54] A. Belouchrani and J.-F. Cardoso, Maximum likelihood source separation for discrete sources, In Proc EUSIPCO, 1994, pp. 768771.

[55] A. Cichocki and S. Amari, "Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications," John Wiley & Sons, Ltd.

[56] E. J. Cands and M. B. Wakin, "An introduction to compressive sampling," IEEE Trans. Signal Proc., vol. 25, no. 2, pp. 21-30, 2008.

[57] E. Cands, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," Comm. Pure Appl. Math., vol. 59, no. 8, pp. 12071223, Aug. 2006.

[58] E. Cands and J. Romberg, "Sparsity and incoherence in compressive sampling," Inverse Prob., vol. 23, no. 3, pp. 969985, 2007.

[59] D. L. Donoho and M. Elad, "Optimally sparse representation in general (non-orthogonal) dictionaries via $\ell_1$ minimization," Proc. Natl Acad. Sci. USA, vol. 100, pp. 2197202, 2003.

[60] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," IEEE Trans. Inf. Theory, vol. 47, pp. 28452862, 2001.

[61] E. J. Cands and M. B. Wakin, "An introduction to compressive sampling," available at http://www.acm.caltech.edu/ emmanuel/papers/spm-robustcs-v05.pdf

[62] R. Coifman, F. Geshwind, and Y. Meyer, "Noiselets," Appl. Comput. Harmon. Anal., vol. 10, pp. 2744, 2001.

[63] E. Candes and T. Tao, "Decoding by linear programming," IEEE Trans. Inf. Theory, vol. 51, no. 12, pp. 4203-4215, Dec. 2005.

[64] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," Constr Approx DOI 10.1007/s00365-007-9003-x.

[65] D. Baron,M. F. Duarte,M. B. Wakin,S. Sarvotham, and R. G. Baraniuk, "Distributed Compressive Sensing," arXiv:0901.3403v1, Jan. 2009.

[66] E. Candes and T. Tao, "Near optimal signal recovery from random projections: universal, encoding strategies?" IEEE Trans. Inf. Theory, vol. 52, pp. 5406-5425, 2006.

[67] D. L. Donoho and J. Tanner, "Thresholds for the recovery of sparse solutions via L1 minimization," 40th Annual Conference on Information Sciences and Systems, 2006, pp 202-206.

[68] E. J. Candes, J. Romberg, and T. Tao, "Signal recovery from incomplete and inaccurate measurements," Comm. Pure Appl. Math., vol. 59 (8), pp. 12071223, 2005.

[69] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections and universal encoding strategies," IEEE Trans. Inform. Theory, vol. 52, pp. 54065425, 2006.

[70] E. J. Cands, "The restricted isometry property and its implications for compressed sensing," Compte Rendus de l'Academie des Sciences, Paris, Serie I, pp. 589-592.

[71] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," IEEE Signal Process. Lett., vol. 14, no. 10, Oct. 2007.

[72] N. Mourad and J. Reilly, "$\ell_p$ Minimization for Sparse Vector Reconstruction", *ICASSP2009*, Taipei, Taiwan, Apr. 19-24, 2009.

[73] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing", Inverse Problems, vol. 24, no. 035020, pp. 1–14, 2008.

[74] R. Saab, R. Chartrand and zgr Yilmaz, "Stable sparse approximations via nonconvex optimization", in 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008.

[75] Y. Li, "A globally convergent method for $\ell_p$ problems," SIAM J. Optim. vol. 3, no. 3, pp. 609629, 1993.

[76] D. L. Donoho and Y. Tsaig, "Fast solution of $\ell_1$-norm minimization problems when the solution may be sparse," preprint (2006).

[77] S. L. Campbell and C. D. Meyer, "Generalized inverse of linear transformations," London, U. K.: Pitman, 1979.

[78] R. G. Baraniuk, "Compressive Sensing," IEEE Trans. Signal Proc., vol. 24, no. 4, pp. 118 - 121, July 2007.

[79] P. C. Hansen, "Analysis of discrete ill-posed problems by means of the L-curve," SIAM Rev., vol. 34, pp. 56180, Dec. 1992.

[80] P. C. Hansen and D. P. OLeary, The use of the L-curve in the regularization of discrete ill-posed problems, SIAM J. Scientific Comput., vol. 14, pp. 14871503, Nov. 1993.

[81] T. Reginska, "A regularization parameter in discrete ill–posed problems," SIAM J. ScI. COMPUT., vol. 17, no. 3, pp. 740-749, May 1996.

[82] J. J. Fuchs, "Recovery of exact sparse representations in the presence of noise," Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2004, vol. 2, pp. II - 533-6.

[83] J. J. Fuchs, "Recovery of exact sparse representations in the presence of bounded noise," IEEE Trans. Inform. Theory, vol. 51, no. 10, 2005.

[84] J. A. Tropp, "Just relax: convex programming methods for identifying sparse signal in noise," IEEE Trans. Inform. Theory, vol. 52, no. 3, 2006.

[85] S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press. Available online at www.stanford.edu/ boyd/cvxbook/bv_cvxbook.pdf.

[86] E. de Klerk, C. Roos, and T. Terlaky, "Nonlinear Optimization", Waterloo, August 26, 2004.

[87] S. C. Chapra, R.P. Canale, "Numerical Methods for Engineers with Programming and Software Applications." Third Edition. USA: McGraw-Hill, 1998.

[88] G. H. Mohimani, M. Babaie-Zadah, and C. Jutten, "A Fast approach for overcomplete sparse decomposition based on smoothed $ell^0$ norm," IEEE Trans. Signal Processing, vol. 57, no. 1, pp. 289-301, Jan. 2009.

[89] Y. Chen, "Blind separation using convex functions," IEEE Trans. Signal Process., Vol. 53, No. 6; pp. 2027-2035, June 2005.

[90] J. Herault and C. Jutten, Space or time adaptive signal processing by neural models, In Proc AIP Conf on Neural Networks for Computing, American Institute of Physics, 1986, pp. 206211.

[91] P. Kisilev, M. Zibulevsky, and Y. Y. Zeevi, " A multiscale framework for blind separation of linearly mixed signals," Journal of Machine Learning Research, vol. 4, pp. 1339–1364, 2003.

[92] D. Erdogmus, L. Vielva, J. C. Principe, "Nonparametric estimation and tracking of the mixing matrix for underdetermined blind source separation," in Proc. ICA'01, San Diago, CA, Dec. 2001, pp. 189–194.

[93] Y. Li, A. Cichocki, and S. Amari, "Sparse Component Analysis for blind source separation with less sensors than sources," 4th International Symposium on Independent Component Analysis and Blind Source Separation, April 2003, Nara, Japan, pp. 89–94.

[94] P.D. OGrady and B.A. Pearlmutter, "Soft-LOST: EM on a mixture of oriented lines." In Fifth Int Conf on Independent Component Analysis, Granada, Spain, Sept. 2224, 2004b, Lecture Notes In Computer Science, Vol. 3195, Springer-Verlag, pp. 430436

[95] A. M. Bronstein, M. M. Bronstein, M. Zibulevsky, and Y. Y. Zeevi, "Sparse ICA for blind separation of transmitted and reflected images," International Journal of Imaging Science and Technology (IJIST), vol. 15/1, pp. 84-91, 2005.

[96] D. Luengo, I. Santamaria, and L. Vielva, "A general solution to blind inverse problems for sparse input signals," Neurocomputing, vol. 69, pp. 198–215, 2005.

[97] P. Georgiev, F. Theis, and A. Cichocki, "Sparse component analysis and blind source separation of underdetermined mixtures," IEEE Trans. Neural Net., vol. 16, no. 4, pp. 992–996, 2005.

[98] M. B. Zadeh, C. Jutten, A. Mansour, "Sparse ICA via cluster-wise PCA," Neurocomputing, vol. 69, pp. 1458–1466, 2006.

[99] Y. Q. Li, A. Cichocki, and S. Amari, "Blind estimation of channel parameters and source components for EEG signals: a sparse factorization approach," IEEE Trans. Neural Net., vol. 17, no. 2, pp. 419431, 2006.

[100] Z. S. He and A. Cichocki, "K-EVD clustering and its applications to sparse component analysis," in Proc. ICA2006,Charleston SC, USA, 2006, pp.9097.

[101] Z. He, A. Cichocki, and S. Xie, "K-hyperplanes clustering and its application to sparse component analysis," ICONIP, part I, pp. 1038-1047, 2006.

[102] F. M. Naini, G. H. Mohimani, M. B. Zadeh, and C. Jutten, "Estimating the mexing matrix in sparse component analysis (SCA) based on partial k-dimentional subspace clustering," Neurocomputing, vol. 71, pp. 2330–2343, 2008.

[103] Z. He, A. Cichocki, Y. Li, S. Xie, and S. Sanei, "K-hyperline clustering learning for sparse component analysis," Signal Processing, vol. 89, no. 6, pp. 1011-1022, 2009.

[104] G. Zhou, Z. Yang, X. Liao, and J. Zhang, "Estimating the mixing matrix by using less sparsity," Progress in Natural Science, vol. 19, pp. 1027–1031, 2009.

[105] R. Xu, and D. Wunsch, "Survey of Clustering Algorithms," IEEE Trans. Neural Net. Vol. 16, No. 3, May 2005.

[106] H. Frigui and R. Krishnapuram, "A robust algorithm for automatic extraction of an unknown number of clusters from noisy data," Pattern Recognit. Lett., vol. 17, pp. 12231232, 1996.

[107] L. Kaufman and P. J. Rousseeuw, "Finding Groups in Data : an Introduction to Cluster Analysis," New York, Wiley, c 1990

[108] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky, "Automatic Extraction of Clusters from Hierarchical Clustering Representations," Lecture Notes in Computer Science, Jan. 2003, Vol. 2637, pp. 75-87.

[109] M. Ankerst, M.M. Breunig, H. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," Proc. ACM SIGMOD'99 Int. Conf. on Manegment of Data, Philadelphia, 1999.

[110] A. C. Rencher, "Multivariate Statistical Inference and Applications," New York, Wiley, c 1998.

[111] T.-P. Jung, C. Humphries, T.-W. Lee, S. Makeig, M. J. McKeown, V. Iragui, and T. Sejnowski, Extended ICA removes artifacts from electroencephalographic recordings, presented at the Neural Information Processing Systems 10 (Proc. NIPS97), 1998.

[112] R. Vigrio, Extraction of ocular artifacts from EEG using independent component analysis, Electroenceph. Clin. Neurophysiol., vol. 103, pp. 395404, 1997.

[113] R. Vigrio, V. Jousmki, M. Hmlinen, R. Hari, and E. Oja, Independent component analysis for identification of artifacts in magnetoencephalographic recordings, in Neural Information Processing Systems 10 (Proc. NIPS97),M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds. Cambridge, MA: MIT Press, 1998.

[114] C. J. James, and O. J. Gibson, " Temporally consstrained ICA: an application to artifact rejection in electromagnetic brain signal analysis," IEEE Transaction on Biomedical Engineering, vol. 50, no. 9, pp. 1108–1116, 2003.

[115] W. Lu and J. C. Rajapakse, "ICA with reference" Proc. 3rd Int. Conf. on Independent Component Analysis and Blind Signal Separation: ICA2001 pp 1201255.

[116] L. K. L. Joshua and J. C. Rajapackse, "Extraction of event-related potentials from EEG signals using ICA with reference," Proceedings of International Joint

Conference on Neural Networks, Montreal, Canada, July 31 - August 4, 2005, pp. 2525–2531.

[117] C. J. James and C. W. Hesse, "Semi-blind source separation in EM brain signal processing," Medical Applications of Signal Processing, 2005. The 3rd IEE International Seminar on (Ref. No. 2005-1119).

[118] C. W. Hesse and C. J. James "The FastICA Algorithm With Spatial Constraints," IEEE SIGNAL PROCESSING LETTERS, vol. 12, NO. 11, pp. 792–795, 2005.

[119] Z. L. Zhang and Z. Yi, "Robust extraction of specific signals with temporal structure," Neurocomputing letters, vol. 69 pp. 888893, 2006.

[120] N. Ille, R. Beucker, and M. Scherg, Spatially constrained independent component analysis for artifact correction in EEG and MEG, Neuroimage, vol. 13, p. S159, 2001.

[121] N. Ille, P. Berg, and M. Scherg, Artifact correction of the ongoing EEG uisng spatial filters based on artifact and brain signal topographies, J. Clin. Neurophysiol., vol. 19, no. 2, pp. 113124, 2002.

[122] C. J. James and C. W. Hesse "Independent component analysis for biomedical signals," Physiol. Meas. vol. 26 (2005) R15R39.

[123] M. Zibulevsky and Y. Y. Zeevi, "Extraction of a source from multichannel data using sparse decomposition," Neurocomputing, vol. 49, pp. 163–173, 2002.

[124] X. Kong and N. V. Thakor, "Adaptive Estimation of Latency Changes in Evoked Potentials", IEEE Trans. on Biomed. Eng., Vol. 43, No. 2, pp.189 - 197, Feb. 1996.

[125] P. Jaskowski and R. Verleger, "An Evaluation of Methods for Single-Trial Estimation of P3 Latency", Psychophysiology, Vol. 37, pp. 153 - 162, 2000.

[126] H. Gibbons and J.Stahl, "Response-time Corrected Averaging of Event-Related Potentials", Clinical Neurophysiology, Vol. 118, 2007, pp. 197 - 208.

[127] P. Jaśkowski and R. Verleger , "Amplitudes and Latencies of single-trial ERP's Estimated by a Maximum-Likelihood Method", IEEE Trans. On Biomedical Engineering, Vol. 46, No. 8, Aug. 1999, pp. 987 - 993.

[128] C. Borgmann, B. RoB, R. Draganova, and C. Pantiv, " Human Auditory Middle Latency Responses: Influence of Stimulus Type and Intensity." Hearing Research, Vol. 158, pp. 57-64, 2001.