RAFAEL ASSATO ANDO

BLIND SOURCE SEPARATION IN THE CONTEXT OF POLYNOMIAL MIXTURES

SEPARAÇÃO CEGA DE FONTES NO CONTEXTO DE MISTURAS POLINOMIAIS

CAMPINAS

2013

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

RAFAEL ASSATO ANDO

BLIND SOURCE SEPARATION IN THE CONTEXT OF POLYNOMIAL MIXTURES

SEPARAÇÃO CEGA DE FONTES NO CONTEXTO DE MISTURAS POLINOMIAIS

Orientador: Prof. Dr. Romis Ribeiro de Faissol Attux
Coorientador: Prof. Dr. Leonardo Tomazeli Duarte

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Engenharia Elétrica, na área de Engenharia de Computação.

Master dissertation presented to the Electrical Engineering Postgraduation Program of the School of Engineering Electrical of the University of Campinas to obtain the M.Sc. grade in Engineering Electrical, in field of Computer Engineering.

Este exemplar corresponde à versão final da dissertação
defendida pelo aluno RAFAEL ASSATO ANDO
e orientada pelo Prof. Dr. ROMIS RIBEIRO DE FAISSOL ATTUX

_____

CAMPINAS

2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

An24b

Ando, Rafael Assato, 1986-
Blind source separation in the context of polynomial mixtures / Rafael Assato Ando. – Campinas, SP : [s.n.], 2013.

Orientador: Romis Ribeiro de Faissol Attux.
Coorientador: Leonardo Tomazeli Duarte.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento de sinais. 2. Algoritmos evolutivos. 3. Análise de componentes independentes. 4. Sistemas não lineares. 5. Separação de fontes. I. Attux, Romis Ribeiro de Faissol,1978-. II. Duarte, Leonardo Tomazeli. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Separação cega de fontes no contexto de misturas polinomiais
Palavras-chave em inglês:
Signal processing
Evolutionary algorithms
Independent component analysis
Nonlinear systems
Source separation
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Romis Ribeiro de Faissol Attux [Orientador]
Ricardo Suyama
Diogo Coutinho Soriano
Data de defesa: 30-08-2013
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Rafael Assato Ando

Data da Defesa: 30 de agosto de 2013

Título da Tese: "Separação Cega de Fontes no Contexto de Misturas Polinomiais"

Prof. Dr. Romis Ribeiro de Faissol Attux (Presidente): _____

Prof. Dr. Ricardo Suyama: _____

Prof. Dr. Diogo Coutinho Soriano: _____

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude:

To my parents, Satoshi and Kio, and my sister, Tania, for their support, encouragement and comprehension.

To my supervisor, Prof. Romis Ribeiro de Faissol Attux, for his dedicated support for my research, his aptitude to motivate and for being a great teacher since graduation.

To my co-supervisor, Prof. Leonardo Tomazeli Duarte, for his help with the research, his well appreciated suggestions and for acting as a valuable liaison between the researchers from France and us.

To Prof. Diogo Coutinho Soriano, for his help on dynamic systems, analysis of local stability and chaotic theory.

To Profs. Yannick Deville and Christien Jutten, for contributing to my research with their comprehensive knowledge on the subject of blind source separation.

To my fellow researchers and friends from DSPCom, for welcoming me to a pleasant and inspiring work environment.

To my friends from Unicamp and École Centrale de Nantes, for their support and incentive.

To the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), and to the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), for the financial support.

To the members of the dissertation defense committee, Prof. Diogo Coutinho Soriano and Prof. Ricardo Suyama, for their encouraging words of appreciation, as well as their invaluable corrections and suggestions to the dissertation.

RESUMO

Neste trabalho, estudamos o problema de BSS no contexto de misturas polinomiais sob três perspectivas: uma teórica - voltada ao estudo de separabilidade estrutural -, uma ligada à proposta de novas abordagens - especialmente como extensões de metodologias baseadas em redes recorrentes – e uma relacionada ao tratamento de problemas práticos como redução do efeito show-through na digitalização de documentos.

A primeira dessas perspectivas levou à proposta de uma nova abordagem do problema de separação não-linear baseada numa formulação do problema instantâneo de inversão como uma tarefa de solução de um sistema de equações algébricas não-lineares. Essa abordagem levou à proposição de novos métodos para lidar com o problema LQ e também pode ser aplicada a outros modelos de mistura. A segunda perspectiva levou à construção de um arcabouço para tratamento do problema LQ baseado numa rede imunológica artificial, o qual trouxe uma menor demanda por informação a priori sobre o problema e provê maior robustez em termos de convergência global. Por fim, a aplicação do ferramental desenvolvido a problemas práticos de tratamento de imagens levou a um desempenho bastante satisfatório, encorajando a extensão futura para outros cenários de teste (como sensores químicos).

Palavras-chave: Processamento de sinais, Separação cega de fontes, Sistemas não-lineares, Análise de componentes independentes, Algoritmos evolutivos.

ABSTRACT

In this work, the BSS problem in the context of polynomial mixtures will be studied under three perspectives: a theoretical one, regarding the structural separability analysis; another related to the proposal of new methodologies – especially as extensions of algorithms based on recurrent networks – and finally, one regarding the solutions to real world problems, such as the reduction of the show-through effect produced by digitally scanning documents.

The first such perspectives led to the proposal of a new approach to the nonlinear BSS problem, based on a formulation to the instantaneous inversion problem as the solution of a nonlinear algebraic equation system. This approach led to the proposal of new methods to deal with the LQ problem, which may also be applied to other mixing models.

The second perspective led to the development of an algorithm based on artificial immune system (AIS) to solve the LQ model, requiring less *a priori* information about the problem and providing better robustness in terms of global convergence. Finally, the application of the proposed methods to the practical problem of image treatment presented a very satisfactory performance, encouraging the possible extension to other test scenarios in the future, such as chemical sensors.

Keywords: Signal processing, Blind Source Separation, Nonlinear systems, Independent component analysis, Evolutionary Algorithm.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

aiNet	Artificial immune network
AIS	Artificial Immune System
ATM	Additive target mixtures
BSS	Blind source separation
DH	Deville-Hosseini [network]
GGD	Generalized Gaussian distribution
ICA	Independent component analysis
LQ	Linear quadratic
MI	Mutual Information
ML	Maximum likelihood
MSE	Mean-squared error
opt-aiNet	Optimization artificial immune network
pdf	Probability density function
PNL	Post nonlinear
RMS	Root mean square
Unicamp	Universidade Estadual de Campinas

SUMMARY

CHAPTER 1

1 INTRODUCTION

The blind source separation (BSS) problem can be considered an important cornerstone of unsupervised signal processing theory. Although this problem has been traditionally studied under the assumption of linear mixing models, having to deal with applications in which such models are not sufficient led to the extension of the BSS theory to also include analysis on nonlinear functions.

While the general nonlinear problem can be considered theoretically unsolvable, it is possible to study specific models that can be solved and are of great practical importance, among which we can include polynomial models such as the linear quadratic (LQ) model and the post nonlinear (PNL) model.

In this dissertation, the study of the BSS problem in the context of polynomial mixtures will be analyzed according to the following outline. In chapter 2, we will briefly explain the basic ideas regarding the BSS problem. Initially, the main concepts for the case of linear mixtures will be presented, due to their historical importance and as the basis of the theory used to solve the more complex, nonlinear case. We will subsequently present the existing algorithms for nonlinear models, and more specifically, for the LQ case.

In chapter 3, a new perspective on the separation problem will be established, based on a reinterpretation of the problem as the solution of a nonlinear algebraic equation system. This allows us to use new methodologies to develop a different class of networks based on numerical root-finding algorithms.

By formulating the problem under the new approach, a specific algorithm based on the Newton-Raphson's method will be proposed, and a comparison to the original networks will show promising results. As will be explained, the proposed solution presents an improvement in terms of stability and convergence rate, as well as a simple strategy to generalize the proposed algorithms for different mixing models and number of sources.

In chapter 4, we will analyze the current methods presented in the literature to estimate the mixing parameters in the case of blind separation for the LQ model, and develop techniques

that might be able to improve them. One of the ideas is to use an evolutionary algorithm based on the artificial immune system (AIS), which is able to provide better robustness in terms of global convergence and does not require *a priori* knowledge about the source distribution.

We will also propose a different optimization strategy based on mutual information between the sources, which is able to mitigate some of the existing problems of the cost function based on maximum likelihood (ML), as originally proposed by Deville and Hosseini [14].

Moreover, we will show simulations in which the proposed networks were able to be tested on a real world application that can be modeled by LQ mixtures: the removal of the show-through effect on scanned images. The simulation results show satisfactory results, with the algorithm being able to successfully separate the images.

Finally, in chapter 5 we will briefly summarize all of the improvements proposed in this work, and conclude with some remarks on the nonlinear BSS problem.

CHAPTER 2

2 SOURCE SEPARATION MODELS

In this chapter, we will explain the basic concepts of blind source separation (BSS), as well as the fundamentals of some key algorithms for solving it. It will be seen that, for the linear case, it is possible to separate the sources - given the hypotheses that the sources are mutually independent and have a non-Gaussian distribution[1] - using a tool called *independent component analysis* (ICA). We will also explain why this tool cannot be used for a general nonlinear case and present some of the main scenarios and approaches to deal with specific nonlinear formulations.

2.1 BASIC CONCEPTS OF SOURCE SEPARATION

We will start by explaining the basic concepts of the BSS problem. Simply put, the problem can be defined in the following way: given a set of N sources $\boldsymbol{s}(n) = [s_1(n), \dots, s_N(n)]^T$, there exists a mixing model that somehow combines them, thereby engendering a set of M mixtures $\boldsymbol{x}(n) = [x_1(n), \dots, x_M(n)]^T = f[\boldsymbol{s}(n)]$, where $M \geq N$ and $f[.]$ is a general mixing function. The aim is then to obtain the source values from the mixtures and some sort of additional information regarding either the sources or the mixing model.

The above description is quite abstract, as the function that is responsible for mixing the sources can have several different natures. Since there is no solution for the most general case, the model has to be analyzed according to a case-by-case basis. A useful taxonomy can be built in terms of the following characteristics of the model:

[1] Actually, in strict terms, as will be seen in section 2.2.2, it is acceptable that a single source be Gaussian.

Linear or Nonlinear: The mixture is said to be *linear* if and only if:

$$f[\alpha_1 \boldsymbol{x_1}(n) + \alpha_2 \boldsymbol{x_2}(n)] = \alpha_1 f[\boldsymbol{x_1}(n)] + \alpha_2 f[\boldsymbol{x_2}(n)] \tag{2.1}$$

and it is said to be *nonlinear* otherwise. It should be mentioned that linear mixtures can be represented in terms of the following expression if the mixing model is instantaneous:

$$f[\boldsymbol{s}(n)] = \boldsymbol{A}\boldsymbol{s}(n) \tag{2.2}$$

where $\boldsymbol{A}$ is a *mixing matrix*.

Underdetermined, Determined or Overdetermined: When $M = N$, the BSS problem is said to be *determined*, and it is well-posed from the standpoint of information recovery. If $M > N$, the mixture is said to be *overdetermined*, providing additional information that can be used to better estimate the mixture parameters and the sources [41]. Finally, if $M < N$ the process can be referred to as *underdetermined* – in which case there is a lack of information caused by the inherently non-invertible character of the mixing model. This may cause the problem to become unsolvable unless additional information about the sources and/or the mixing process is available (the use of sparsity, discussed in [5], is a promising possibility in this sense).

Blind or Non-Blind: The source separation problem can be defined as *non-blind* if there is *a priori* knowledge about the parameters of the mixing model; it is said to be *blind* if this is not the case.

The study of non-blind mixing models, in spite of their more restrictive nature, is certainly important. Firstly, it is possible that, sometimes, in real world applications, at least some of the mixing model coefficients are known; secondly, even if that is not the case, it might be possible to create an external set of known source-mixture pairs that would allow this knowledge to be obtained in a deterministic fashion [6]. Finally, as will be seen later in this work, understanding the non-blind separation case can be an essential step towards solving the blind case.

Stochastic or Deterministic: The mixing function can be considered stochastic if different outputs can be obtained from the same input, in consonance with a certain probability model. Conversely, in the case of deterministic models, the same input will always produce the same output.

Time-invariant or Time-variant: The system can be defined as *time-invariant* if its input-output response is not modified by a time shift, and it is otherwise called *time-variant*. In mathematical terms, a time-invariant system must satisfy:

$$\text{If } f[\boldsymbol{x}(t)] = \boldsymbol{y}(t), \qquad \text{then } f[\boldsymbol{x}(t+\delta)] = \boldsymbol{y}(t+\delta) \quad \forall \delta \in \mathbb{R} \tag{2.3}$$

Memoryless or with Memory: As briefly mentioned earlier, the mixing system can be said to be *memoryless* (or *instantaneous*) if its output depends only on the current value of the input, and it is said to have *memory* if the output possesses some kind of dependence with respect to previous input values.

In this dissertation, the focus will be on mixing models that are nonlinear, determined, time-invariant, memoryless and deterministic. However, before the nonlinear case be addressed, it is important to discuss the linear case in view of its historical background and of its associated simple and elegant framework.

2.2 LINEAR SOURCE SEPARATION

The classical BSS scenario is defined by a linear, determined and instantaneous mixing model [9]. In that case, the mixing model can be, as anticipated in (2.2), described by:

$$\boldsymbol{x}(n) = \boldsymbol{A}\,\boldsymbol{s}(n) \tag{2.4}$$

where $\boldsymbol{A}$ is a full-rank N x N mixing matrix. If the mixing matrix is known *a priori*, solving the problem is straightforward – all one must do to recover the sources is to make use of its inverse:

$$\boldsymbol{s}(n) = \boldsymbol{A}^{-1}\boldsymbol{x}(n) \tag{2.5}$$

A more thorough approach to the problem will be necessary for the blind case, as the coefficients of $\boldsymbol{A}$ will be unknown and additional information will be essential to handle it. This leads us to the analysis of methods for effectively performing ICA in the linear and instantaneous case. The discussion will begin with the pioneering formulation by Hérault, Jutten and Ans – to be revisited later in terms of original results presented in this dissertation - and will, in the following, cover the standard approach to the problem, based on the notion of independent component analysis (ICA) [7, 9, 22].

2.2.1 The *Hérault-Jutten* Algorithm

A pioneering effort in solving the linear BSS problem is the *Hérault-Jutten* algorithm [9, 42], which based on a neurocomputing theoretical framework [6]. The separating structure for the two-source ($N = M = 2$) case can be seen in Fig. 2.1.

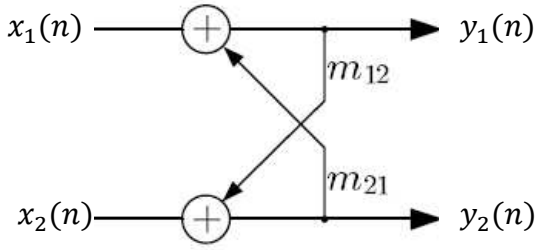

Figure 2.1 - Separating structure used in the Hérault-Jutten algorithm

In the structure presented in Fig. 2.1, $x_i(n)$, $i = 1, 2$, are the mixtures and $y_j(n)$, $j = 1, 2$, are the source estimates. If the time index is omitted for the sake of simplicity, the structure input-output relationship can be expressed as:

$$\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{M}\boldsymbol{y} \tag{2.6}$$

where $\boldsymbol{M}$ is the matrix containing the weights m_{ij}, and is such that its diagonal terms are zero. By isolating $\boldsymbol{y}$ in (2.6), one obtains:

$$\boldsymbol{y} = (\boldsymbol{I} + \boldsymbol{M})^{-1}\boldsymbol{x} \tag{2.7}$$

The goal of the algorithm is, then, to estimate the weight matrix $\boldsymbol{M}$. Hérault, Jutten and Ans used the following update term [9, 26]:

$$\Delta m_{ij} \propto g_1(y_i)g_2(y_j), \quad i \neq j \tag{2.8}$$

The rationale of (2.8) is based on the concept of *nonlinear cross-correlation*. If it is assumed that the sources are mutually independent, it can be shown that [9]:

$$E\{g_1(s_i)g_2(s_j)\} = 0 \qquad \forall i \neq j \tag{2.9}$$

where $g_1(.)$ and $g_2(.)$ are suitable odd nonlinear functions that satisfy $E\{g_1(s_i)\} = E\{g_2(s_i)\} = 0, \forall i \in \{1, \dots, N\}$. The Hérault-Jutten algorithm then estimates $E\{g_1(s_i)g_2(s_j)\}$ as $g_1(y_i)g_2(y_j)$, using this stochastic estimate in a gradient-like method to obtain $\boldsymbol{W}$, as shown in (2.8).

The nonlinear cross-correlation was selected as a computationally simple measure of independence between the variables. While $E\{s_i s_j\} = 0$ (uncorrelatedness) is a necessary condition for independence, it is not sufficient. As explained in [26], by choosing appropriate nonlinear functions $g_1(.)$ and $g_2(.)$, condition (2.9) implies not only uncorrelatedness, but also that higher order product-moments $E\{s_i^{n_1} s_j^{n_2}\}$ be zero, making a much stronger case for independence. The functions $g_1(.)$ and $g_2(.)$ must be odd and different, but otherwise a broad range of functions can be used. In their paper, successful simulations were performed using $g_1(x) = x^3$, and $g_2(x)$ as either x, $sign(x)$ or $\tan^{-1} x$.

Gaussian distributions can be tricky in the sense that their higher order product-moments can be zero, and yet the variables can still be independent. As a result, the algorithms should be used when at most one of the variables presents a Gaussian distribution.

When convergence is reached, each $y_i(n)$ should ideally correspond to a source. The algorithm converges only under certain circumstances [9, 22], but was, notwithstanding, an important step towards the creation of *independent component analysis* theory, which is, as already mentioned, the canonical method for solving the linear BSS problem. The adoption of recurrent networks to iteratively solve a BSS task was also of great importance to the development of a family of solutions to the nonlinear BSS case, as will be seen in chapter 3.

2.2.2 Independent Component Analysis

In simple terms, *independent component analysis* (ICA) is a statistical method to discover, in a certain set of samples of multivariate data, mutually independent components or factors that are representative of the underlying information content. In order to do so, it is necessary to initially define a way to quantify the degree of independence between signals and variables.

Let us now assume the validity of the linear and instantaneous model presented in (2.4), and consider that all sources are mutually independent. In view of the fact that all mixtures that form $\boldsymbol{x}(n)$ are linear combinations of the same set of sources, they, in general, will no longer be independent. Hence, it is natural to speculate on the possibility that, by recovering the independence condition that is inherent to the source set, one will also obtain a suitable separating matrix $\boldsymbol{W}$ that will play the role of an inverse to the mixing matrix $\boldsymbol{A}$. This ICA-based BSS approach is indeed sound, as shown by Comon [6], given the following assumptions:

Separability Conditions – Linear and Instantaneous Model - ICA:

1. The mixing model is invertible (i.e., the mixing matrix $\boldsymbol{A}$ is non-singular)
2. The sources are mutually independent.
3. Among the sources there is at most one source with a Gaussian distribution.

In other words, given the validity of these conditions, if a matrix $\boldsymbol{W}$ is obtained such that the signals $\boldsymbol{y}(n) = \boldsymbol{W}\boldsymbol{x}(n)$ are also mutually independent, this matrix will necessarily be of the form $\boldsymbol{W} = \boldsymbol{PDA}^{-1}$, where $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{D}$ is a diagonal matrix. The latter matrix accounts for the fact that the use of ICA leads to estimates of the sources in any order – as a permutation does not alter the independence between random variables – whereas the matrix $\boldsymbol{D}$ indicates that the sources can be recovered up to general scale factors.

In the context of linear and instantaneous BSS, two of the most widespread methods to measure independence are [7, 22]:

Non-Gaussianity: according to the central limit theorem [27], the limit of a sum of independent random variables tends towards a Gaussian variable. Intuitively, this reveals that a linear mixture of sources is "more Gaussian" than any of the sources – as a consequence, the more "Gaussian-like" a signal $y_i(n)$ is, the less it will correspond to the estimate of a signal of interest. Some ways to quantify non-Gaussianity includes the use of *kurtosis* [28] and *negentropy* [22].

Mutual Information: considering the set of source estimates $\boldsymbol{y}(n) = [y_1(n), \dots, y_N(n)]^T$, it is possible to define mutual information (MI) as:

$$I(\boldsymbol{y}(n)) = \sum_{i=1}^{N} h(y_i(n)) - h(\boldsymbol{y}(n)) \tag{2.10}$$

where $h(.)$ is Shannon's differential entropy [29, 30]:

$$h(X) = -\int_{-\infty}^{\infty} f(x) \log f(x)\, dx \tag{2.11}$$

and $f(x)$ is the pdf of the random variable X – or, in the case of a set of variables as $\boldsymbol{y}(n)$, the joint pdf distribution.

Two properties of the MI will be very important for its use in ICA: a) it is always nonnegative and b) it is null if, and only if, the random variables are statistically independent [30]. These properties justify the use of this metric to quantify independence: the closer to zero $I(\boldsymbol{y}(n))$ is, the "more independent" the associated variables are.

In summary, these two ICA formulations can be employed to build a criterion to determine the separating matrix $\boldsymbol{W}$ such that $\boldsymbol{y}(n) = \boldsymbol{W}\boldsymbol{x}(n)$ contains maximally-independent components. The underlying cost function can, in this case, be described as a *contrast function*. A contrast function $\Psi(.)$ is any non-linear function which is invariant to permutation and scaling matrices, and achieves its global minimum when the components of its input are mutually independent.

The process of optimizing this function is typically iterative, generating a class of ICA-based BSS algorithms like the FastICA [28] and the Infomax algorithms [31], only to mention a few.

2.3 THE NONLINEAR CASE

Despite the facts that the linear model is undoubtedly useful and has been extensively studied over the last years, there are situations in which nonlinear models are imperative to properly represent the necessary physical quantities and relationships. In contrast with the well-posed linear case, in a nonlinear scenario, even for non-blind source separation problems, the inversion of the mixing function can be difficult to achieve and sometimes even analytically impossible, similarly to the problem presented by underdetermined linear models.

It should be noted that, for the linear case, even when the coefficients of the mixing matrix are unknown, if the presented separability conditions are respected, ICA methods can ensure

proper source recovery, up to permutation and scaling ambiguities [9]. For nonlinear problems, though, it can be shown that these conditions are not sufficient – it is possible to find examples where, for a certain set of mixtures, two or more different solutions with independent components can be found [10]. For that reason, in the case of a generic nonlinear mixture, if we want to guarantee the uniqueness of ideal solutions, more information (about the sources and/or the mixture) will be needed.

Because nonlinear models are too generic, there is no algorithm that can be used to solve the problem regardless of the mixture. Instead, the BSS task must be dealt with on a case-by-case basis.

2.3.1 Post Nonlinear Models

An emblematic framework for performing nonlinear BSS is that arising from the use of *post nonlinear* (PNL) mixing models [11]. A general PNL mixing model can be illustrated with the aid of the diagram presented in Fig. 2.2:

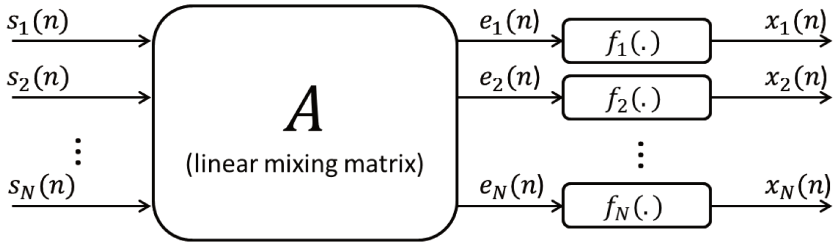

Figure 2.2 - PNL model scheme

Mathematically, the diagram corresponds to the following input-output relationship:

$$\boldsymbol{x}(n) = \boldsymbol{f}[\boldsymbol{A}\,\boldsymbol{s}(n)] \tag{2.12}$$

where $\boldsymbol{f}[\boldsymbol{e}] = [f_1(e_1), f_2(e_2), \ldots, f_N(e_N)]^{\mathrm{T}}$ is a nonlinear vector field whose components are invertible scalar functions [11]. In other words, the sources are linearly mixed according to a full-rank matrix and then each mixture is independently applied to an invertible nonlinear function. It is important to notice that the nonlinearity only affects the intermediate mixtures individually – it

does not "mix the signals further" (that is, $f_1(e_1)$ does not depend on e_2, etc.). This formulation is interesting to model, for example, the nonlinear effects of amplifying stages and sensors over linear signal superposition [11].

A natural way to invert a PNL model is to cancel off the nonlinear functions and, then, to perform standard linear BSS. Hence, a separating system can be as shown in equation (2.13):

$$\boldsymbol{y}(n) = \boldsymbol{W}\boldsymbol{g}[\boldsymbol{x}(n)] \tag{2.13}$$

where $\boldsymbol{g}[\boldsymbol{x}(n)] = [g_1(x_1(n)), g_2(x_2(n)), \ldots, g_N(x_N(n))]^{\mathrm{T}}$ should be such that the components of $\boldsymbol{g}[\boldsymbol{x}(n)]$ be a linear mixture of the original sources. The PNL model has been studied along the last decade, and the separability conditions for this model, assuming the use of ICA, have been found to be [11, 23]:

Separability Conditions - PNL Model - ICA:

1. The mixing matrix $\boldsymbol{A}$ is invertible and effectively mixes the sources – in other words, each row and column of this matrix contains at least two non-zero elements.
2. The functions $\boldsymbol{f}[.]$ and $\boldsymbol{g}[.]$ are monotonic (and, as a result, so is $(\boldsymbol{g}\ o\ \boldsymbol{f})[.]$).
3. At most one of the sources has a Gaussian distribution.
4. The joint pdf of the sources is of class C^1 – that is, it is differentiable and its derivative is continuous.

Much like in the case of linear BSS, methods for performing ICA-based separation for PNL models are generally based on the direct use of mutual information, as the adoption of a limited set of moments can be misleading [20, 21]. Algorithms based on nonlinear optimization methods and also on bio-inspired algorithms are reported in the literature, and different strategies can be selected to handle the aspects of modeling of $\boldsymbol{g}[.]$ and of probability density function estimation. The interested reader is referred to [11, 20, 21, 23] for more details on the subject.

2.3.2 The Linear Quadratic Model

Another nonlinear mixing model that can be considered of particular importance both in theoretical and practical terms [12] is the *linear quadratic* (LQ) model. A general mixture associated with this model can be described as follows:

$$x_i(n) = \sum_{j=1}^{N} a_{ij} s_j(n) + \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} b_{ijk} s_j(n) s_k(n) \tag{2.14}$$

What equation (2.14) basically means is that each mixture is composed of a linear part and a quadratic part based on cross products of different sources. For the sake of clarity, we will initially study the simpler case of LQ models with two sources and two mixtures (*N = M = 2*):

$$\begin{aligned} x_1(n) &= a_{11} s_1(n) + a_{12} s_2(n) + b_1 s_1(n) s_2(n) \\ x_2(n) &= a_{21} s_1(n) + a_{22} s_2(n) + b_2 s_1(n) s_2(n) \end{aligned} \tag{2.15}$$

Since we consider solutions that are different only up to permutation and scaling factors as equally valid, it is possible to assume that the sources to be recovered are $s'_1(n) = a_{11} s_1(n)$ and $s'_2(n) = a_{22} s_2(n)$, and, instead, write the model as:

$$\begin{aligned} x_1(n) &= s'_1(n) - L_1 s'_2(n) - Q_1 s'_1(n) s'_2(n) \\ x_2(n) &= s'_2(n) - L_2 s'_1(n) - Q_2 s'_1(n) s'_2(n) \end{aligned} \tag{2.16}$$

where $L_i = -a_{ij}/a_{ii},\ i \neq j$, and $Q_i = -b_i/(a_{11} a_{22})$. For the sake of simplicity, we will henceforth refer to these new values of the sources as $s_1(n)$ and $s_2(n)$ again.

Equations (2.15) and (2.16) ultimately represent the same mixing process (in the sense that the solutions are equivalent up to scaling factors), but the latter is defined in terms of 4 parameters (instead of 6), which will be very attractive from the standpoint of model identification in the context of the BSS algorithms discussed later [13].

An interesting point to notice is that for the linear case, modifying the sources by offsets and scaling factors merely cause the mixtures to be offset or scaled respectively. Since these modified sources effectively represent the same solution, we can scale and offset the mixtures as we see fit, before applying BSS algorithms to it. This initial operation is called *preprocessing*.

For nonlinear models as a whole, preprocessing cannot be done. Sources that differ by scaling factors and offsets still represent the same solution, but they cannot be easily translated to a similar change in the mixtures due to the nonlinearities present in the model.

2.3.3 Invertibility of the LQ Model

As an important remark, it should be noticed that the conditions for the separability of the LQ model are still a subject of active research. While it is known that independence is not enough for coping with general nonlinear mixtures [10], experimental results suggest that, for the LQ model, it might be sound [10, 13].

Similarly to the case of linear mixtures, the invertibility of the mixing model has to be a part of the separability prerequisites that have to be satisfied. Since the mixing model is no longer straightforwardly invertible from a purely analytical standpoint, though, the actual meaning of this invertibility needs to be reinterpreted.

Knowing that the LQ model represents a second-order polynomial system, it is natural for it to have two solutions. As will be shown in section 2.4.1, these two solutions are actually equivalent – in the sense that they will differ only by order, scaling factor and a constant offset. The solutions can be written in the form of (2.17):

$$(s_1^*, s_2^*) = \left(\frac{-b_1 \pm J}{2a_1}, \frac{-b_2 \pm J}{2a_2}\right) \tag{2.17}$$

where J is the determinant of the Jacobian of the system, and a_i, b_i are constants that will be defined the next section.

When an algorithm to separate the sources is applied, either of the solutions is considered to be acceptable. However, it is important to always choose the same sign, in order to avoid mixing these two equivalent, but different solutions when estimating the source as a whole.

One problem that will become apparent in section 2.4.1, though, is that we will only be able to estimate J^2 from the parameters and the mixtures, from which we can estimate $|J|$, but not J. Hence, the invertibility condition that has to be satisfied is [13, 14]:

Invertibility of the LQ model:

The Jacobian of the mixing system does not change sign throughout all points on the data set:

$$J(n) > 0 \ \ \forall n, \ \text{ or } \ J(n) < 0 \ \forall n, \qquad n \in \{1,2,\dots,N\} \tag{2.18}$$

where $J(n) = 1 - L_1 L_2 - (Q_2 + Q_1 L_2)s_1(n) - (Q_1 + Q_2 L_1)s_2(n)$.

2.4 SOLUTIONS FOR LINEAR QUADRATIC MODELS

In this section, we will present the methods for solving non-blind LQ source separation problem presented in the literature and also discuss some of their features and drawbacks. Assuming *a priori* knowledge about the mixing model coefficients, or if we are able to estimate them from a set of known source-mixtures pairs, the source separation problem will amount, essentially, to non-analytical system inversion. In the context of BSS, the problem will be tackled based on the following general method:

General Procedure for Performing BSS in the Context of a Nonlinear Model:

1. Start with random mixing parameters.
2. Using the current mixing parameters estimates, solve the problem as if it were non-blind.
3. Using the source estimates obtained on Step 2, use an algorithm to estimate a better set of values for the mixing parameters.
4. Repeat steps 2 and 3 until convergence (if the procedure does not converge, start again from step 1).

For the linear case, step 2 is trivial, in which case the success of the whole algorithm depends on step 3, as we have seen in section 2.2. When dealing with a nonlinear mixture, however, separating the sources can be difficult even when the coefficients are available – hence the need for recurrent network-based approach to step 2. In this dissertation, chapter 3 will propose solutions to the problem in step 2, while chapter 4 will address the problem in the next step.

In sections 2.4.1 to 2.4.3 we will present the canonical methods for solving the non-blind problem under LQ mixtures, while section 2.4.4 will introduce an algorithm for estimating the mixing parameters of the model (step 3) in the non-blind case.

2.4.1 Analytical Solution for Two Sources

In the simple case of two sources, as shown in (2.16), we can actually obtain an analytical solution for inversion in the non-blind case [13]. To do so, we initially rewrite equation (2.16) as:

$$x_1(n) - s_1(n) + L_1 s_2(n) + Q_1 s_1(n) s_2(n) = 0 \tag{2.19}$$

$$x_2(n) - s_2(n) + L_2 s_1(n) + Q_2 s_1(n) s_2(n) = 0 \tag{2.20}$$

By calculating $Q_2 \times (2.19) - Q_1 \times (2.20)$, we can cancel out the cross product terms and obtain a relation between $s_1(n)$ and $s_2(n)$:

$$s_2(n) = \frac{(Q_2 + Q_1 L_2)s_1(n) - \left(Q_2 x_1(n) - Q_1 x_2(n)\right)}{Q_1 - Q_2 L_1} \tag{2.21}$$

We can then use (2.21) in (2.19), and obtain:

$$\begin{aligned}(Q_1 L_2 + Q_2)s_1^2(n) + (Q_1 x_2(n) - Q_2 x_1(n) + L_1 L_2 - 1)s_1(n) \\ + (x_1(n) + L_1 x_2(n)) = 0\end{aligned} \tag{2.22}$$

The same thing can be done by isolating and replacing $s_1(n)$, to obtain:

$$\begin{aligned}(Q_2 L_1 + Q_1)s_2^2(n) + (Q_2 x_1(n) - Q_1 x_2(n) + L_1 L_2 - 1)s_2(n) \\ + (x_2(n) + L_2 x_1(n)) = 0\end{aligned} \tag{2.23}$$

By solving both quadratic equations we can obtain an analytical solution to the problem:

$$(s_1^*, s_2^*) = \left(\frac{-b_1 \pm \sqrt{\Delta_1}}{2a_1}, \frac{-b_2 \pm \sqrt{\Delta_2}}{2a_2}\right) \tag{2.24}$$

where:

$$a_1 = Q_1 L_2 + Q_2 \qquad a_2 = Q_2 L_1 + Q_1$$
$$b_1 = Q_1 x_2(n) - Q_2 x_1(n) + L_1 L_2 - 1 \qquad b_2 = Q_2 x_1(n) - Q_1 x_2(n) + L_1 L_2 - 1,$$
$$c_1 = x_1(n) + L_1 x_2(n) \qquad c_2 = L_2 x_1(n) + x_2(n)$$
$$\Delta_1 = b_1^2 - 4a_1 c_1 \qquad \Delta_2 = b_2^2 - 4a_2 c_2$$

We can also check that $\Delta_1 = \Delta_2 = J(n)^2$, where $J(n)$ is the system Jacobian given in (2.17). As was discussed in section 2.3.3 the system will always have two different solutions, depending on the sign selected in (2.24). One of the solutions is $(s_1^*, s_2^*) = (s_1, s_2)$; while the other is

$$(s_1^*, s_2^*) = \left(\frac{-a_2 s_2 - L_1 L_2 + 1}{a_1}, \frac{-a_1 s_1 - L_1 L_2 + 1}{a_2}\right) \tag{2.25}$$

The solution presented in (2.25) is equivalent to the first solution, except for a permutation, a scaling factor, and an offset [13, 14]. Because both solutions are equivalent, either of them is acceptable, as long as points from different sets are not mixed.

As explained in section 2.3.3, if the sign of the Jacobian remains constant throughout the data set of the sources, it is possible to ensure that points from the same solution set are always chosen [14]. If the sign of the Jacobian changes, though, in order to obtain a consistent solution we would sometimes need to apply the positive sign on (2.23), and sometimes the negative sign, depending on the Jacobian sign, which unfortunately cannot be estimated.

2.4.2 The Basic Deville-Hosseini (DH) Recurrent Network

The analytical solution is certainly effective and easy to calculate, but it only works for two sources. So, a different kind of algorithm is required to hand the more general case. The difficulty for obtaining an analytical solution suggested to Hosseini and Deville [13] the adoption of a recurrent network that iteratively converges to the desired solution.

The first algorithm of this kind was proposed in [13], and it presented a significant similarity with respect to the Hérault-Jutten structure presented in section 2.2.1. However, it was applied to a class of nonlinear mixtures termed *additive target-mixtures* (ATMs) [13]. ATMs form a class of nonlinear mixtures that satisfy equation (2.26) below, and the LQ model is a member of this class.

$$x_i(n) = T_i[\boldsymbol{s}(n)] - I_i[\boldsymbol{s}(n)], \quad \forall i \in \{1, \dots, P\} \tag{2.26}$$

where $T_i[.]$ and $I_i[.]$ are defined as the *target* and *interfering* terms; or, in other words, they are the mixture components that would be interesting to keep and remove, respectively, from the outputs of the separating system. The same separating network (which will be referred to as Deville-Hosseini or DH network) can be used for dealing with any ATM model - it is illustrated in Fig. 2.3:

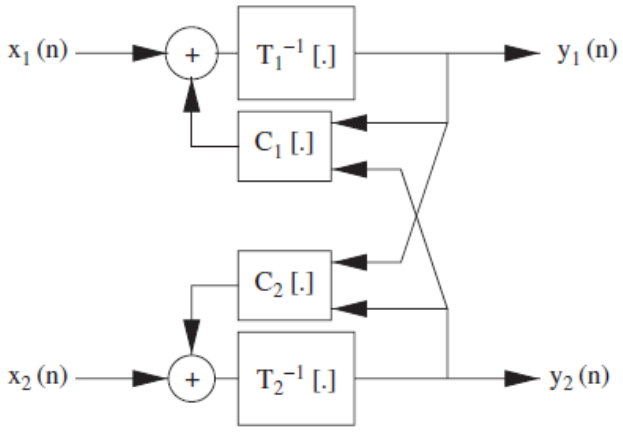

Figure 2.3 - Recurrent network for general additive-target mixtures

A comparison between Fig. 2.3 and Fig. 2.1 stresses the similarity between the DH network is similar to that associated with the Hérault-Jutten algorithm, with the addition of a cross-multiplied component required to cancel the nonlinear terms. When specifically applied to the LQ model [13], we obtain:

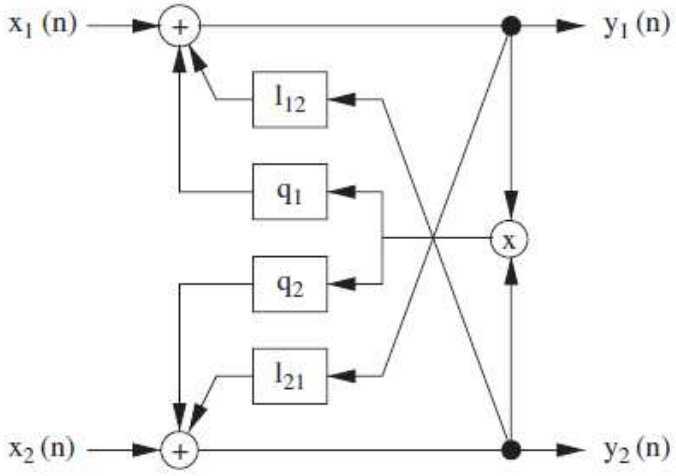

Figure 2.4 - Recurrent network for LQ model

which can be represented, in mathematical terms, in the form:

$$\begin{aligned} y_1(m+1) &= x_1(n) + L_1 y_2(m) + Q_1 y_1(m) y_2(m) \\ y_2(m+1) &= x_2(n) + L_2 y_1(m) + Q_2 y_1(m) y_2(m) \end{aligned} \tag{2.27}$$

In (2.27), the DH algorithm iterates through m while keeping n constant. This can be done because since the model is considered to be instantaneous, we can solve one point at a time independently. The variables (y_1, y_2) represent estimates of the sources, and ideally, they should converge to either (s_1, s_2), or to the alternative proposed in (2.25) The iteration presented by (2.27) uses only two sources, but it is possible to generalize it to any amount [13].

The basic algorithm, though, only converges for some values of source amplitudes and mixing parameters, as can be shown by a theoretical analysis on the local stability, as well as by simulations presented in chapter 3.

2.4.3 The Stabilized Deville-Hosseini Network

To help solve the stability problem, Hosseini and Deville proposed an extended version of their network, which includes a self-feedback loop that stabilizes the algorithm [13], resulting in the following network:

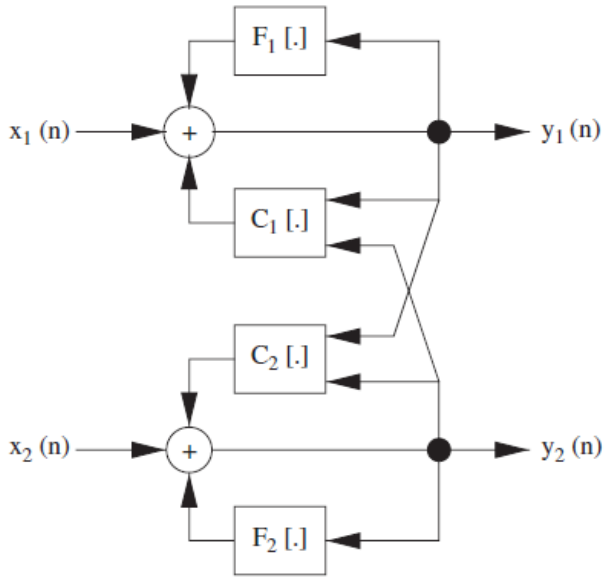

Figure 2.5 - Recurrent network for LQ model, extended version

which can also be described by the iterations:

$$\begin{aligned} y_1(m+1) &= x_1(n) + \ell_{11}y_1(m) + \ell_{12}y_2(m) + q_1y_1(m)y_2(m) \\ y_2(m+1) &= x_2(n) + \ell_{22}y_2(m) + \ell_{21}y_1(m) + q_2y_1(m)y_2(m) \end{aligned} \tag{2.28}$$

where the ℓ_{ii} terms are associated with the aforementioned self-feedback strategy. Unlike the basic network, the terms ℓ_{ij} and q_i presented in (2.28) are not the mixture coefficients – instead, they are different constants based on the mixture parameters and the self-feedback parameters according to the following rules [13]:

$$\begin{aligned} \ell'_{11} &= 1 - \ell_{11} \\ \ell'_{22} &= 1 - \ell_{22} \\ \ell_{12} &= \ell'_{22}L_1 \\ \ell_{21} &= \ell'_{11}L_2 \\ q_1 &= \ell'_{11}\ell'_{22}Q_1 \\ q_2 &= \ell'_{11}\ell'_{22}Q_2 \end{aligned} \tag{2.29}$$

As a drawback, the stabilized DH network is also limited to two sources; or, more precisely, in the way it was proposed in [13], it is difficult to generalize the implementation to encompass more than two sources.

2.4.4 Estimation of the Mixing Parameters

For the blind case, Hosseini and Deville used a *maximum likelihood* (ML) approach to generate a cost function that can be used to estimate the mixing parameters [14]. The ML principle is a general method for estimating parameters in a statistical model, in which a *likelihood* value can be assigned for each set of parameters estimate, depending on how likely it would be for the parameters in question to generate the observed data set. The likelihood function can be expressed as:

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = f(\boldsymbol{x}|\boldsymbol{\theta}) \tag{2.30}$$

that is, as the probability (or, in the continuous case, as the pdf value) of obtaining the observed data, given that the parameters of the model are $\boldsymbol{\theta}$.

The application of the likelihood function to the particular model we are studying, as well as the actual calculation the cost function obtained using this method can be found in chapter 4 and in their paper [14]. The expression of the cost function obtained is:

$$C = -E_K\left[\sum_{i=1}^{N} \log f_{s_i}(\hat{s}_i)\right] + E_K\left[\log\left|\hat{J}\right|\right] \tag{2.31}$$

where $E_K[.]$ is the time average operator, $f_{s_i}(.)$ is the pdf of source s_i, and $\hat{\mathrm{J}}$ is the estimate of the Jacobian of the equation system. The algorithm proposed by Hosseini and Deville then uses the gradient method to minimize the cost function in (2.31), which in turn demands knowledge of the derivatives of the cost function [14]. For the simplest case of two sources, the gradient can be calculated as [14]:

$$\frac{dC}{d\widehat{\boldsymbol{\theta}}} = E_K\left[\frac{D_1}{|\hat{J}|}, \frac{D_2}{|\hat{J}|}, \frac{D_3}{|\hat{J}|}, \frac{D_4}{|\hat{J}|}\right] \tag{2.32}$$

where $\psi_{s_i}(s_i) = -\,d\,log\,f_{s_i}(s_i)\,/\,ds_i$ is the score function, $\widehat{\boldsymbol{\theta}}$ is the estimate of the parameter vector, and:

$$\begin{aligned} D_1 = {} & \psi_{s_1}(\hat{s}_1)\left(1 - \hat{Q}_2\hat{s}_1\right)\hat{s}_2 + \left(\hat{L}_2 + \hat{Q}_2\hat{s}_2\right)\left(\psi_{s_2}(\hat{s}_2)\hat{s}_2 - 1\right) \\ & - \left[\left(\hat{Q}_2 + \hat{L}_2\hat{Q}_1\right)\left(1 - \hat{Q}_2\hat{s}_1\right) + \left(\hat{Q}_1 + \hat{L}_1\hat{Q}_2\right)\left(\hat{L}_2 + \hat{Q}_2\hat{s}_2\right)\right]\hat{s}_2/|\hat{J}| \end{aligned} \tag{2.33}$$

$$\begin{aligned} D_2 = {} & \psi_{s_2}(\hat{s}_2)\left(1 - \hat{Q}_1\hat{s}_2\right)\hat{s}_1 + \left(\hat{L}_1 + \hat{Q}_1\hat{s}_1\right)\left(\psi_{s_1}(\hat{s}_1)\hat{s}_1 - 1\right) \\ & - \left[\left(\hat{Q}_1 + \hat{L}_1\hat{Q}_2\right)\left(1 - \hat{Q}_1\hat{s}_2\right) + \left(\hat{Q}_2 + \hat{L}_2\hat{Q}_1\right)\left(\hat{L}_1 + \hat{Q}_1\hat{s}_1\right)\right]\hat{s}_1/|\hat{J}| \end{aligned} \tag{2.34}$$

$$\begin{aligned} D_3 = {} & \left[\psi_{s_1}(\hat{s}_1)\left(1 - \hat{Q}_2\hat{s}_1\right) + \psi_{s_2}(\hat{s}_2)\left(\hat{L}_2 + \hat{Q}_2\hat{s}_2\right)\right]\hat{s}_1\hat{s}_2 - \hat{L}_2\hat{s}_1 - \hat{s}_2 \\ & - \left[\left(\hat{Q}_2 + \hat{L}_2\hat{Q}_1\right)\left(1 - \hat{Q}_2\hat{s}_1\right) + \left(\hat{Q}_1 + \hat{L}_1\hat{Q}_2\right)\left(\hat{L}_2 + \hat{Q}_2\hat{s}_2\right)\right]\hat{s}_1\hat{s}_2/|\hat{J}| \end{aligned} \tag{2.35}$$

$$\begin{aligned} D_4 = {} & \left[\psi_{s_2}(\hat{s}_2)\left(1 - \hat{Q}_1\hat{s}_2\right) + \psi_{s_1}(\hat{s}_1)\left(\hat{L}_1 + \hat{Q}_1\hat{s}_1\right)\right]\hat{s}_1\hat{s}_2 - \hat{L}_1\hat{s}_2 - \hat{s}_1 \\ & - \left[\left(\hat{Q}_1 + \hat{L}_1\hat{Q}_2\right)\left(1 - \hat{Q}_1\hat{s}_2\right) + \left(\hat{Q}_2 + \hat{L}_2\hat{Q}_1\right)\left(\hat{L}_1 + \hat{Q}_1\hat{s}_1\right)\right]\hat{s}_1\hat{s}_2/|\hat{J}| \end{aligned} \tag{2.36}$$

A potential problem of the method is that it requires either *a priori* knowledge of the distribution of the sources or a reliable way to estimate their derivatives from the data. While methods for estimating such derivatives exist [17], they are not always accurate enough. Additionally, if the sources have distributions that cannot be differentiated in all points (for example, a uniform distribution), there will be potential difficulties related to gradient calculation.

Another problem is that, unlike in equation (2.31), where we can see the cost function C has a general expression that is valid for any number of sources (and even for different mixing model functions), the gradient would have to be explicitly calculated on a case-by-case basis, depending on the number of sources and the mixing model. The resulting expression would have to be hardcoded on the algorithm, making it impractical for general use. The gradient expression can also be seen to be increasingly more complex for a greater number of sources, which can quickly render the method intractable.

As we will see in chapter 4, we will be able to circumvent this requirement by using a bio-inspired algorithm that will not require gradient estimation, as well as a different cost function that requires less prior information on the sources.

CHAPTER 3

3 ON THE STRUCTURE OF THE SOURCE SEPARATION PROBLEM

In this chapter, we will present a first set of contributions associated with this work, which are related to structural aspects of nonlinear source separation. Throughout this exposition, it will be assumed that the problem to be dealt with is non-blind, with the discussion of proposals and applications related to the blind case left to chapter 4. This division is useful from the standpoint of clarity of exposition, but it is also relevant because, as outlined in section 2.4, the non-blind separation task will be a stage of the analyzed blind methods.

3.1 A NEW APPROACH TO THE PROBLEM OF SYSTEM INVERSION IN BSS

As discussed in chapter 2, the canonical strategies to solve the non-blind part of the LQ source separation [14] problem can be understood in terms of neural networks with an interesting conceptual similarity with respect to the emblematic structure proposed by Hérault, Jutten and Ans [26]. In other words, the use of a recurrent network like that shown in Fig. 3.1 has been part of the *modus operandi* of the separating algorithms, and weights, connections and feedback loops have to be adapted to suit each kind of mixing model and number of sources, in such a way that the recurrent algorithm be able to yield the separated sources upon convergence.

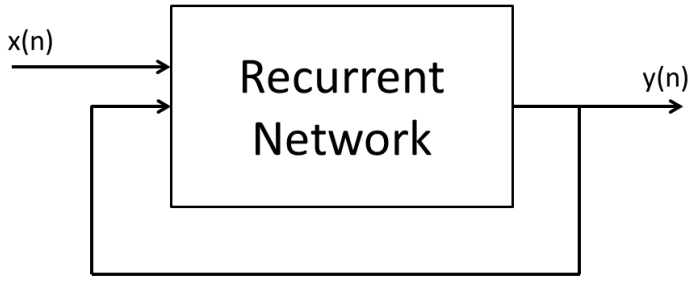

Figure 3.1 - Generic recurrent network structure

In order to understand this state of things from a mathematical point of view, let us remember the basic definition of the BSS problem. Our goal is to obtain the set of sources $\boldsymbol{s}(n)$, given a nonlinear function $\mathcal{F}[.]$ and the set of mixtures $\boldsymbol{x}(n)$ satisfying:

$$\boldsymbol{x}(n) = \mathcal{F}[\boldsymbol{s}(n)] \tag{3.1}$$

Consider that the mixing function $\mathcal{F}[.]$ is complex enough that we cannot analytically invert it (an inverse exists, but cannot be found in closed form and may or may not be unique). The standard approach that has been used is to create an algorithm that effectively behaves as an estimate of the function $\mathcal{W}[.]$ such that:

$$\boldsymbol{s}(n) = \mathcal{W}[\boldsymbol{x}(n)] \tag{3.2}$$

Essentially, the canonical methods are based on $\mathcal{W}[.]$, which is an actual inverse of $\mathcal{F}[.]$ (if there is more than one, any is acceptable). For that to be possible, we need to know *a priori* how the general structure of the inverse should be, because the structure of the recurrent network is based on this function. Even for simple cases like the LQ model, for a higher number of sources, knowing the structure of the inverse can already be difficult.

In this dissertation, we will propose a different approach to the problem, which is conceptually simple and very general in its scope, both with respect to the mixing models and to the number of sources. As will be seen later, the new approach will also give rise to recurrent networks, but they will not arise as *ad hoc* solutions.

Instead of designing a specific structure whose equilibrium points are equivalent to inverse solutions to a predetermined system, we will rewrite equation (3.1) as:

$$\mathcal{G}[\boldsymbol{s}(n)] = \mathcal{F}[\boldsymbol{s}(n)] - \boldsymbol{x}(n) = 0 \tag{3.3}$$

and then directly solve the new equation system $\mathcal{G}[.]$. In other words, the process of inverting the mixing system will be treated simply as an iterative method for solving a system of nonlinear algebraic equations. This perspective has the advantage of simplifying the problem formulation –

it is reduced to the much more broadly studied problem of solving a nonlinear system - and raises the possibility of using several well-established iterative general-purpose algorithms [16, 33].

It is also important to notice that, in spite of the fact that most of the results presented in the rest of the dissertation focus on the LQ model, this new approach is not restricted to this particular model: any known mixing model and with any amount of sources can be formulated from this perspective, and can be numerically solved via standard root-finding algorithms.

3.2 APPLICATION TO THE LINEAR-QUADRATIC CASE

Applying equation (2.14) to (3.3), it is possible to obtain a representation of $\mathcal{G}[.]$ (the system whose roots we want to find) for the specific case of LQ models. For the case of two sources ($N = M = 2$), we obtain:

$$\mathcal{G}[\boldsymbol{s}(n)] = \begin{bmatrix} s_1(n) - L_1 s_2(n) - Q_1 s_1(n) s_2(n) - x_1(n) \\ s_2(n) - L_2 s_1(n) - Q_2 s_1(n) s_2(n) - x_2(n) \end{bmatrix} \tag{3.4}$$

Firstly, we will demonstrate that the algorithms presented in chapter 2 can be derived as particular cases of first-order root-finding algorithms applied to the LQ model. In this context, the expression *first-order algorithm* is used to indicate a method that converges with linear speed [24], that is:

$$0 < \lim_{k\to\infty} \frac{|y(k+1) - L|}{|y(k) - L|} < 1 \tag{3.5}$$

where $L = \lim_{k\to\infty} y(k)$.

Likewise, we can define a higher-order method as having rate of convergence q if the method satisfies:

$$\lim_{k\to\infty} \frac{|y(k+1) - L|}{|y(k) - L|} = 1 \text{ and } \lim_{k\to\infty} \frac{|y(k+1) - L|}{|y(k) - L|^q} > 0 \tag{3.6}$$

3.2.1 Reinterpretation of the Basic Deville-Hosseini Algorithm

If we revisit the Deville-Hosseini (DH) algorithm, discussed in section 2.4.2, it is possible to see that (2.24) can also be written as [15]:

$$\boldsymbol{y}(m+1) = \boldsymbol{y}(m) - \mathcal{G}[\boldsymbol{y}(m)] \tag{3.7}$$

where $\boldsymbol{y}(m)$ is the estimate of $\boldsymbol{s}(n)$. If the algorithm converges, then $\boldsymbol{y}(m+1) = \boldsymbol{y}(m)$ and $\mathcal{G}[.] = 0$, as desired. This can be viewed as a simple fixed-point method to find the roots of $\mathcal{G}[.]$, but it suffers from the concrete menace of instability. Another way of interpreting it would be as a simplified version of the *gradient descent* method [16] – a first-order optimization algorithm – to minimize the cost function given by:

$$C[\boldsymbol{y}(m)] = \frac{|\mathcal{G}[\boldsymbol{y}(m)]|^2}{2} = \frac{\mathcal{G}_1^2(\boldsymbol{y}(m)) + \mathcal{G}_2^2(\boldsymbol{y}(m))}{2} \tag{3.8}$$

The basic gradient descent method has the following form:

$$\boldsymbol{y}(m+1) = \boldsymbol{y}(m) - \alpha \nabla C[\boldsymbol{y}(m)], \qquad \alpha > 0 \tag{3.9}$$

where $\alpha > 0$, as the aim is to minimize the cost function. Since $\nabla C[\boldsymbol{y}(m)] = \mathcal{G}[\boldsymbol{y}(m)]$, equation (3.7) can be seen as a simplified version of (3.9), where $\alpha = 1$. It is known that the *gradient descent* method converges linearly, so the basic DH network can be classified as a first-order algorithm according to the above definition.

The simplest improvement that can be suggested with respect to this algorithm is to treat α in (3.9) as a time-variant value by allowing it to be potentially modified on an iterative basis, being the choice guided by the idea of obtaining minimizing steps from the standpoint of $C[\boldsymbol{y}(m)]$ - equation (3.8). Notice how, under the new interpretation to the problem, the idea of implementing this improvement arises naturally as a stabilizing factor and convergence acceleration factor, whereas, in the original interpretation of a DH network, its purpose is not *prima facie*

so straightforward. The idea of employing a variable step size will be further discussed in the next section.

3.3 ON THE USE OF A VARIABLE STEP-SIZE FACTOR

The implementation of a variable step size is an extension to the basic DH network that arises in a natural way from the use of a gradient descent method. According to the proposed framework, the inclusion of this variable gives rise to the following update expression:

$$\boldsymbol{y}(m+1) = \boldsymbol{y}(m) - \alpha(m)\mathcal{G}[\boldsymbol{y}(m)] \tag{3.10}$$

For the case with $N = M = 2$, this leads to the following equations:

$$\begin{aligned} y_1(m+1) &= (1-\alpha(m))y_1(m) + \alpha(m)\big(x_1(n) + L_1 y_2(m) + Q_1 y_1(m) y_2(m)\big) \\ y_2(m+1) &= (1-\alpha(m))y_2(m) + \alpha(m)\big(x_2(n) + L_2 y_1(m) + Q_2 y_1(m) y_2(m)\big) \end{aligned} \tag{3.11}$$

Notice that, on each iteration, $y_i(m+1)$ is a weighted average between $y_i(m)$, the current value of y_i, and the value that would be assumed by it in the original DH recurrent network, discussed in section 2.4.2. The performance of this strategy will be now analyzed.

3.3.1 Selecting the Value of the α Variable

One way to implement the use of the variable step size algorithm is to select, for each iteration, the value of α that leads to the "most minimizing" step with respect to the cost function. This establishes a *line search* task [16] that can be computationally demanding when real-time applications are considered, hence the algorithm that will be discussed in the following is a simpler version that, instead of seeking the optimal solution, accepts a value that is, according to a predetermined threshold, close enough. While the usage of suboptimal values of α might lead to more iterations until the estimate of the sources converge, this is compensated by the fact that

each iteration will be simpler, due to the less expensive version of the line-search algorithm implemented.

The idea can be summarized in terms of the algorithm described below:

Algorithm for Selecting the Value of $\alpha(m)$:

1. Start with $\alpha(m) = 1$ (a "large value").
2. Calculate the cost function $C[\boldsymbol{y}(m)]$.
3. Slightly reduce the value of $\alpha(m)$ (for example, by multiplying it by 0.99, or some other rate).
4. Recalculate $C[\boldsymbol{y}(m)]$.
5. Repeat steps 3 and 4 until the cost either stabilizes, or starts increasing.

After the above algorithm concludes its execution, a value of $\alpha(m)$ will have been obtained that, while not necessarily optimal, can be found relatively easily and should have a satisfactory performance.

3.3.2 Local Instability / Chaotic Behavior

In order to verify whether a variable step-size really improves the local stability of the DH algorithm, we selected a scenario with mixing parameters and source values that, in the literature, were known to cause the basic network to be unstable [13]. The mixing parameters in this simulation were $L_1 = L_2 = 0$, $Q_1 = Q_2 = 0.5$, and the source values were $s_1 = -1, s_2 = -2$. In the algorithm, the initial estimate was $y_1(0) = y_2(0) = 0$, and the value of α was kept constant on all iterations, but varied on different instances of the same simulation, ranging from 0.01 to 1.2.

The obtained results can be seen in Fig. 3.2. In the upper graph, we can see the value obtained for y_1 – the estimate of s_1 –, depending on the value of α. The values of y_1 have only started to be recorded after 50 iterations in order to avoid the transient associated with the initial iterations. In the lower graph, the value of the largest Lyapunov exponent of the system [25, 35] is represented. The largest Lyapunov exponent characterizes the rate of separation of infinitesi-

mally close trajectories in a dynamical system in such a way that $|\delta \boldsymbol{Z}(t)| \approx e^{\lambda t}|\delta \boldsymbol{Z}_0|$, or, more precisely:

$$\lambda = \lim_{t\to\infty} \lim_{\delta \boldsymbol{Z}_0 \to 0} \frac{1}{t} \ln \frac{|\delta \boldsymbol{Z}(t)|}{|\delta \boldsymbol{Z}_0|} \tag{3.12}$$

For fixed point iterations described by $x_{n+1} = f(x_n)$, equation (3.12) becomes:

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln|f'(x_i)| \tag{3.13}$$

The value of this exponent is very important to characterize the dynamic behavior of the iterative method. When it has a negative value, the system steady-state behavior will be of periodic character – convergence towards an equilibrium point or a limit cycle [34]. On the other hand, a positive value, in the absence of divergence towards infinity, is a sound indicative of the existence of chaos. Additionally, the greater the magnitude of the Lyapunov exponent, the faster the dynamic system will converge (in case of negative exponent) or diverge (for positive exponent). In simple terms, if the focused system operates in a chaotic regime, it will show a pattern characterized by aperiodicity and sensitive dependence with respect to the initial conditions [35].

The upper graph, which is termed a *bifurcation diagram*, is a very useful tool to study modifications in the stability of limit sets [35]. It shows that the proposed separating system will have a proper operation (recovering the correct source sample) for all values of α in the interval between 0 and 0.8. For values of α slightly above 0.8, the system starts to present periodic oscillations, and a cascade of period-doubling bifurcations takes place when α is further increased. After this cascade, the system presents chaotic behavior (visually identifiable in the zones with an expressive point density) with periodicity windows. The behavior, which is duly confirmed by the analysis of the largest Lyapunov exponent, appears often in scenario for discrete-time dynamical systems (also called *maps*), and is known as a Feigenbaum bifurcation diagram [25, 34, 35, 36].

In our example, we can notice that the fastest convergence happens at $\alpha \approx 0.9$, at which point the Lyapunov exponent in minimal. However, as seen in the bifurcation diagram, at this point the system does not converge to the correct solution – instead, it converges to a limit cycle. Therefore, the ideal value of α in our case would be approximately 0.57, when we can see the minimum value of the Lyapunov exponent in the interval that converges to the correct solution.

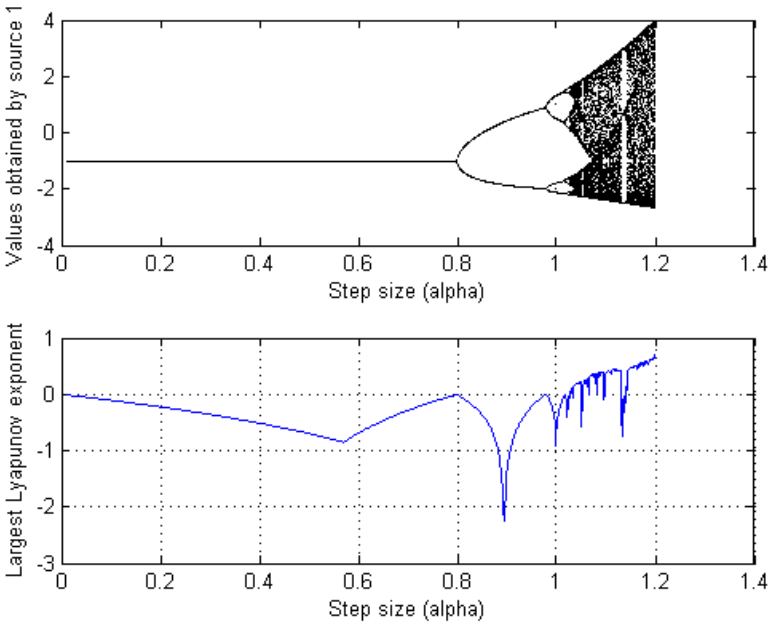

Figure 3.2 - Bifurcation diagram and Lyapunov exponent graphs

This analysis, albeit preliminary, clearly reveals the complexity of possible behaviors arising from recurrent networks with a DH structure and also indicates the importance of using control strategies like the adoption of a variable step-size, or, alternatively, the self-feedback extension presented in 2.4.3.

3.3.3 Reinterpretation of the Stabilized Deville-Hosseini Method

In this section, we will show that the stabilized DH algorithm is, in fact, equivalent to the basic network with the implementation of a general (albeit fixed) step-size. The stabilized network given at section 2.4 can be described in terms of the following expressions:

$$\begin{aligned} y_1(m+1) &= x_1(n) + \ell_{11} y_1(m) + \ell_{12} y_2(m) + q_1 y_1(m) y_2(m) \\ y_2(m+1) &= x_2(n) + \ell_{22} y_2(m) + \ell_{21} y_1(m) + q_2 y_1(m) y_2(m) \end{aligned} \tag{3.14}$$

By considering $\alpha_1 = \ell'_{11}$ and $\alpha_2 = \ell'_{22}$, we can apply (2.27) to (3.14) to obtain the equivalent form:

$$\begin{aligned} y_1(m+1) &= (1-\alpha_1) y_1(m) + x_1(n) + L_1 \alpha_2 y_2(m) + Q_1 \alpha_1 \alpha_2 y_1(m) y_2(m) \\ y_2(m+1) &= (1-\alpha_2) y_2(m) + x_2(n) + L_2 \alpha_1 y_1(m) + Q_2 \alpha_1 \alpha_2 y_1(m) y_2(m) \end{aligned} \tag{3.15}$$

Multiplying both sides of the first and second equations by α_1 and α_2, respectively, and by making $y'_1 = \alpha_1 y_1$, and $y'_2 = \alpha_2 y_2$, we can rewrite (3.13) as:

$$\begin{aligned} y'_1(m+1) &= (1-\alpha_1) y'_1(m) + \alpha_1 \big(x_1(n) + L_1 y'_2(m) + Q_1 y'_1(m) y'_2(m) \big) \\ y'_2(m+1) &= (1-\alpha_2) y'_2(m) + \alpha_2 \big(x_2(n) + L_2 y'_1(m) + Q_2 y'_1(m) y'_2(m) \big) \end{aligned} \tag{3.16}$$

It should be noted that the solution (y'_1, y'_2) is equivalent to (y_1, y_2), since it only differs by scaling factors. Equations (3.16) are very similar to (3.13), which are the equations for the basic DH algorithm, with the addition of a general step-size. The difference is that, instead of using a single parameter α, the extended DH network has two parameters for the step-size, one for each source.

Another way of writing the expression of interest is:

$$\boldsymbol{y}(m+1) = \boldsymbol{y}(m) - \begin{bmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{bmatrix} \mathcal{G}[\boldsymbol{y}(m)] \tag{3.17}$$

which, again, is similar to (3.10). Therefore, we can see that the stabilized DH network is, according to the new interpretation to the problem, similar to the root-finding algorithm presented in section 3.2.

If $\alpha_1 \neq \alpha_2$, then the step will not follow the direction given by the gradient. Since this vector indicates the local direction with the greatest rate of variation of the cost function $C[.]$ to be minimized, to use $\alpha_1 = \alpha_2$ seems justifiable. In fact, in their paper [13], Deville and Hosseini

themselves made this choice in their own simulations. In that case, the extended DH network would be exactly the same as the gradient descent method described in the beginning of the section, under the constraint that $\alpha(m)$ be fixed.

Another interesting fact regarding this new interpretation is that, while it was not evident in the literature how to extend the stabilized DH recurrent network to accommodate more than two sources [13], using the gradient method makes the extension quite direct. It should also be remarked that it would also be direct to generalize the model for different mixing models.

3.3.4 Local Stability of First-Order Algorithms

In this section, we will analyze the local stability of the first-order algorithm presented in (3.10). A classical way to perform this analysis is to study the behavior of a linearized version of the system of interest in the vicinity of an equilibrium point, which leads to the criterion described below:

Verification of Local Stability:

1. Write the iterative system in the form:

$$\boldsymbol{x}(n+1) = \mathcal{H}[\boldsymbol{x}(n)] \tag{3.18}$$

2. Calculate $J_{\mathcal{H}}[\boldsymbol{x}_{eq}]$, the Jacobian of the $\mathcal{H}[.]$ applied to the equilibrium point $\boldsymbol{x}_{eq}$. The equilibrium point is such that $\boldsymbol{x}_{eq} = \mathcal{H}[\boldsymbol{x}_{eq}]$.

3. Calculate the eigenvalues of $J_{\mathcal{H}}[\boldsymbol{x}_{eq}]$.

4. The iterative system is said to be locally stable around $\boldsymbol{x}_{eq}$ if, and only if, all of the eigenvalues satisfy $|\lambda_i| < 1$.

Since the gradient method encompasses the stabilized DH network, when $\ell'_{11} = \ell'_{22}$, we can use the stability analysis already done by Deville and Hosseini in [13], at least for the case of

two sources. In their paper, they show that, in order to have local stability, the following conditions have to be satisfied:

$$\begin{aligned} |\ell'_{11}\ell'_{22}|\sqrt{\delta_{y_1}} - 2A\ell'_{11} - 2B\ell'_{22} + 4 > 0 \\ |\ell'_{11}\ell'_{22}|\sqrt{\delta_{y_1}} - A\ell'_{11} - B\ell'_{22} < 0 \end{aligned} \tag{3.19}$$

where:

$$\begin{aligned} A &= \frac{Q_1}{2\beta}\left[-Q_2x_1 + Q_1x_2 + \gamma - sgn(\ell'_{11}\ell'_{22})\sqrt{\delta_{y_1}}\right] + 1 \\ B &= \frac{Q_2}{2\alpha}\left[-Q_2x_1 + Q_1x_2 - \gamma + sgn(\ell'_{11}\ell'_{22})\sqrt{\delta_{y_1}}\right] + 1 \\ \delta_{y_1} &= [Q_2x_1 - Q_1x_2 + \gamma]^2 - 4\alpha(x_1 + x_2L_1) \\ \alpha &= Q_2 + Q_1L_2 \\ \beta &= -Q_1 - Q_2L_1 \\ \gamma &= 1 - L_1L_2 \end{aligned} \tag{3.20}$$

Note that, as long as $sgn(\ell'_{11}\ell'_{22})$ is constant, all the variables at (3.20) are actually constant, depending only on the mixing parameters and the mixtures. We are interested in studying the stability particularly under the interpretation of the algorithm as a form of gradient-descent method, so we will assume $\ell'_{11} = \ell'_{22} = \mu > 0$ (we shall use μ, since α was already used by Deville and Hosseini). We can then rewrite the conditions given at (3.19):

$$\mu^2\sqrt{\delta_{y_1}} - 2\mu(A + B) + 4 > 0 \tag{3.21}$$

$$\mu^2\sqrt{\delta_{y_1}} - \mu(A + B) < 0 \tag{3.22}$$

Equation (3.22) can be solved to give:

$$\mu < M = \frac{A + B}{\sqrt{\delta_{y_1}}} \tag{3.23}$$

And (3.21) then becomes:

$$\mu^2 - 2M\mu + \frac{4}{\sqrt{\delta_{y_1}}} > 0 \tag{3.24}$$

which can also be written as:

$$(M - \mu)^2 > M^2 - \frac{4}{\sqrt{\delta_{y_1}}} \tag{3.25}$$

If $M^2 < 4/\sqrt{\delta_{y_1}}$, then (3.25) is always satisfied, and the only condition that has to be satisfied is (3.23). Otherwise, we obtain:

$$\mu < M - \sqrt{M^2 - \frac{4}{\sqrt{\delta_{y_1}}}} \tag{3.26}$$

Equations (3.23) and (3.26) yield:

$$\mu < \mu_{\max} = M - \sqrt{\left(M^2 - \frac{4}{\sqrt{\delta_{y_1}}}\right)^+} \tag{3.27}$$

where x^+ is the positive part of x; that is, $x^+ = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$.

If $\mu_{\max} < 1$, then the basic DH network would have failed to converge. Additionally, as long as $\mu_{\max} > 0$, there always exists a value of μ that satisfies (3.27), so the algorithm is at least

guaranteed to locally converge by including the variable step size (or equivalently, the stabilized DH network). If $\mu_{\max} \leq 0$, even the stabilized network would have failed to converge.

For the example analyzed in section 3.3.2, where the mixing parameters were $L_1 = L_2 = 0$ and $Q_1 = Q_2 = 0.5$, we can apply equations (3.20) to obtain $\alpha = 0.5$, $\beta = -0.5$ and $\gamma = 1$. The mixture values are $x_1 = -2$ and $x_2 = -3$, from which we can then calculate $\sqrt{\delta_{y_1}} = 2.5$, $A = 2$, $B = 1.5$ and $M = 1.4$, from equations (3.20). $M^2 > 4/\sqrt{\delta_{y_1}}$, therefore, $\mu_{\max}$ should be calculated by expression (3.26) and gives $\mu_{\max} = 0.8$, as observed by the simulation.

3.4 SECOND-ORDER RECURRENT NETWORK

After discussing a new perspective on nonlinear BSS problems and showing how the canonical methods can be interpreted as *gradient descent* methods, the next step will be to investigate the use of methods that have a better rate of convergence than that of gradient descent approaches. A promising possibility, both for its simplicity and quadratic rate of convergence, is Newton's method [16]. When applied to an equation system of the form $\mathcal{G}[\boldsymbol{s}(n)] = 0$, the method can be described as:

$$\boldsymbol{y}(m+1) = \boldsymbol{y}(m) - \mu(m)\, J_{\mathcal{G}}^{-1}[\boldsymbol{y}(m)]\, \mathcal{G}[\boldsymbol{y}(m)] \tag{3.28}$$

where $J_{\mathcal{G}}[.]$ is the Jacobian of $\mathcal{G}[.]$, and $\boldsymbol{y}(m)$ is the estimate that, ideally, converges to $\boldsymbol{s}(n)$. Since the algorithm requires the use of $J_{\mathcal{G}}^{-1}[.]$, it can only be used if the Jacobian is non-singular – which is in consonance with the invertibility conditions discussed in section 2.3.3.

Aside from the use of a step-size variable μ, a way to improve the performance of the algorithm is to avoid the explicit calculation of the inverse of the Jacobian in (3.28) by solving the following linear system:

$$J_{\mathcal{G}}[\boldsymbol{y}(m)]\Delta\boldsymbol{y}(m) = -\mu(m)\, \mathcal{G}[\boldsymbol{y}(m)] \tag{3.29}$$

The structure of the proposed method can be seen below:

Algorithm Based on Netwon's Method:

1. Initialize $\boldsymbol{y}(0)$ randomly and set $\mu(0) = 1$.

2. Use the same method described in section 3.3 to select the value of $\mu(m)$. For each $\mu(m)$, when solving the system, use (3.29).

3. Having obtained $\Delta\boldsymbol{y}(m)$, calculate $\boldsymbol{y}(m+1) = \boldsymbol{y}(m) + \Delta\boldsymbol{y}(m)$.

4. Repeat steps 2 and 3 until $\boldsymbol{y}(m)$ converges.

3.4.1 Local Stability of the Algorithm Based on Newton's Method / Existence of Chaos

In order to analyze the local stability of the second-order algorithm presented in this section, we will resort again to local stability analysis explained in section 3.3.4. In this section, the analysis will be carried out exclusively for the case of two sources. The iterative system $\mathcal{H}[.]$ is:

$$\boldsymbol{y}(m+1) = \mathcal{H}[\boldsymbol{y}(m)] = \boldsymbol{y}(m) - \mu\, J_{\mathcal{G}}^{-1}[\boldsymbol{y}(m)]\, \mathcal{G}[\boldsymbol{y}(m)] \tag{3.30}$$

Let $\mathcal{G}[\boldsymbol{y}(m)] = \begin{bmatrix} G_1(\boldsymbol{y}(m)) \\ G_2(\boldsymbol{y}(m)) \end{bmatrix}$. The Jacobian of $\mathcal{G}[.]$ can be then calculated as:

$$J_{\mathcal{G}}[y(m)] = \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \end{bmatrix} \tag{3.31}$$

where $j_{ik} = \frac{\partial G_i}{\partial y_k}$. Therefore:

$$J_{\mathcal{G}}^{-1} = \frac{1}{D}\begin{bmatrix} j_{22} & -j_{12} \\ -j_{21} & j_{11} \end{bmatrix} \tag{3.32}$$

where $D = j_{11}j_{22} - j_{12}j_{21}$ is the determinant of the Jacobian. Replacing (3.32) in (3.30) yields:

$$\mathcal{H}[\boldsymbol{y}] = \begin{bmatrix} H_1(\boldsymbol{y}) \\ H_2(\boldsymbol{y}) \end{bmatrix} = \begin{bmatrix} y_1 - \frac{\mu}{D}(j_{22}G_1 - j_{12}G_2) \\ y_2 - \frac{\mu}{D}(j_{11}G_2 - j_{21}G_1) \end{bmatrix} \tag{3.33}$$

The Jacobian of the fixed point system $\mathcal{H}[\boldsymbol{y}]$ is then:

$$J_{\mathcal{H}}[\boldsymbol{y}] = \begin{bmatrix} \frac{\partial H_1}{\partial y_1} & \frac{\partial H_1}{\partial y_2} \\ \frac{\partial H_2}{\partial y_1} & \frac{\partial H_2}{\partial y_2} \end{bmatrix} \tag{3.34}$$

Let us calculate each of the four partial derivatives in (3.34) individually. For the first one, we obtain:

$$\frac{\partial H_1}{\partial y_1} = 1 - \frac{\mu}{D^2}\left(\frac{\partial(j_{22}G_1 - j_{12}G_2)}{\partial y_1}D - (j_{22}G_1 - j_{12}G_2)\frac{\partial D}{\partial y_1}\right) \tag{3.35}$$

Since we want to calculate the eigenvalues of $J_{\mathcal{H}}[y_E]$ when applied to the equilibrium point y_E, we have $G_1[y_E] = G_2[y_E] = 0$. Thus, equation (3.35) can be simplified to:

$$\begin{aligned} \frac{\partial H_1}{\partial y_1} &= 1 - \frac{\mu}{D}\left(\frac{\partial(j_{22}G_1 - j_{12}G_2)}{\partial y_1}\right) \\ &= 1 - \frac{\mu}{D}\left(\frac{\partial j_{22}}{\partial y_1}G_1 + j_{22}\frac{\partial G_1}{\partial y_1} - \frac{\partial j_{12}}{\partial y_1}G_2 - j_{12}\frac{\partial G_2}{\partial y_1}\right) \\ &= 1 - \frac{\mu}{D}(j_{22}j_{11} - j_{12}j_{21}) \\ &= 1 - \mu \end{aligned} \tag{3.36}$$

The second derivative can be calculated as follows:

$$\begin{aligned}\frac{\partial H_1}{\partial y_2} &= -\frac{\mu}{D^2}\left(\frac{\partial(j_{22}G_1 - j_{12}G_2)}{\partial y_2}D - (j_{22}G_1 - j_{12}G_2)\frac{\partial D}{\partial y_2}\right) \\ &= -\frac{\mu}{D}\left(\frac{\partial(j_{22}G_1 - j_{12}G_2)}{\partial y_2}\right) \\ &= -\frac{\mu}{D}(j_{22}j_{12} - j_{12}j_{22}) \\ &= 0\end{aligned} \tag{3.37}$$

Likewise, we can calculate the remaining two derivatives to obtain $\partial H_2/\partial y_1 = 0$ and $\partial H_2/\partial y_2 = 1 - \mu$. Using these derivatives in (3.34) yields:

$$J_{\mathcal{H}}[y_E] = \begin{bmatrix} 1-\mu & 0 \\ 0 & 1-\mu \end{bmatrix} \tag{3.38}$$

The eigenvalues of $J_{\mathcal{H}}[y_E]$ are then $\lambda_1 = \lambda_2 = 1 - \mu$, and, in order to obtain $|\lambda_i| < 1$, we need to satisfy $0 < \mu < 2$.

Unlike the first-order algorithm analyzed in the previous section, we can see that the condition that has to be satisfied so that the algorithm can be considered locally stable does not depend on the values of the mixtures or the sources, or even on the mixing parameters. In fact, it doesn't even depend on the mixing model, which are represented by the functions G_1 and G_2 – with a different model we would achieve the same conclusion, as long as the functions are differentiable.

Another interesting fact is that, when using $\mu = 1$, the system is always locally convergent. Therefore, the use of the variable step size strategy is not required to ensure stability when using an algorithm based on the Newton's Method.

When applying the Newton-based algorithm to the same simulation scenario described in section 3.3.4, we obtain different patterns for the associated bifurcation diagram and the largest Lyapunov exponent, as shown in Fig. 3.3.

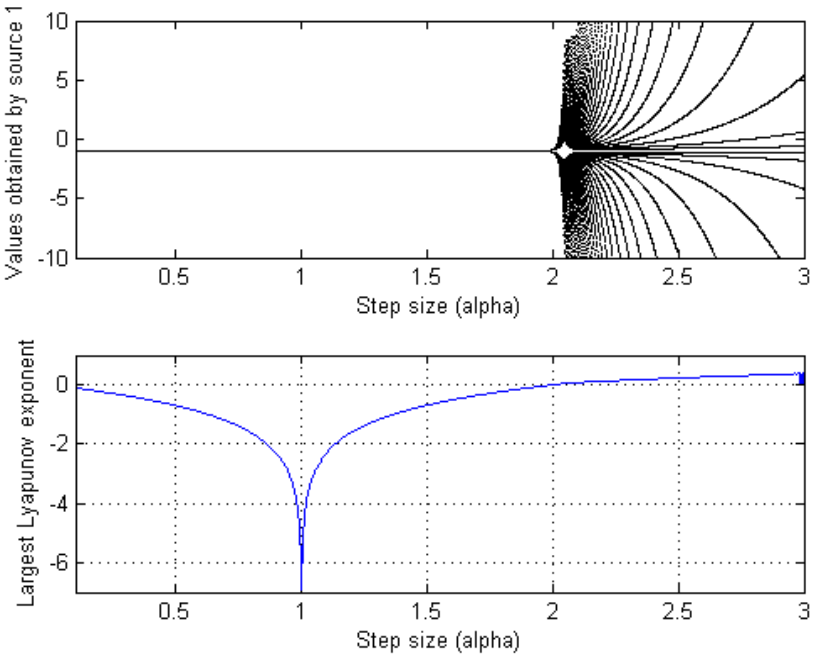

Figure 3.3 - Bifurcation diagram and largest Lyapunov exponent for Newton based algorithm

The bifurcation diagram, shown in the upper graph of Fig. 3.3, reveals that, for $0 < \mu < 2$, the algorithm converges to the correct solution and for $\mu > 2$ we see the chaotic behavior. This can be verified by the lower graph as well, showing that the largest Lyapunov exponent is positive for $\mu > 2$.

As expected, the algorithm converges for $\mu = 1$. Not only that, but the convergence rate for that step size is maximal – which is represented by the minimum value of the Lyapunov exponent. This suggests that, for the second-order algorithm based on the Newton's method, the use of a variable step-size strategy would not be needed, as shown by the theoretical analysis.

3.5 SIMULATIONS AND RESULTS

In this section, we will show the results of simulations to gauge the accuracy and efficiency of the algorithms presented in this chapter. All the simulations were done in on a Dual Core 2.26 GHz computer running Windows 7, on MATLAB R2010a.

3.5.1 Simulations on Variable Step Size Strategy

This simulation intends to show a comparison between the basic DH network and the first-order algorithm using the variable step size strategy.

In our first simulation, we used the following mixing parameters: $L_1 = Q_1 = -0.8$, $L_2 = Q_2 = -0.9$, and the sources are uniformly distributed in the $[-1; 1]$ interval. As suggested in [13], these parameters are such that the basic DH network does not always converge. The results are shown in Fig. 3.4.

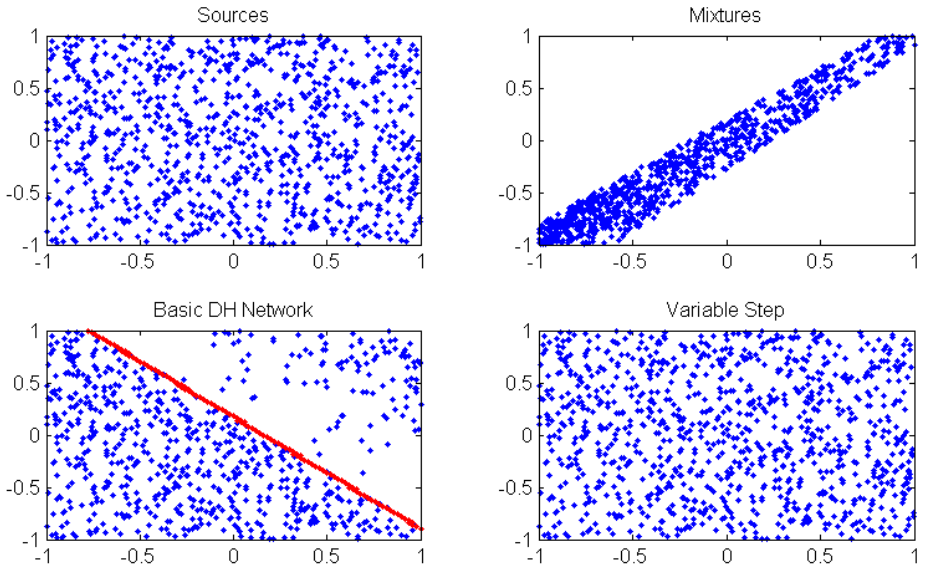

Figure 3.4 - Results of simulation for basic DH network and variable step size

In Fig. 3.4, we can see the original joint source distribution, where the data is plotted on a (s_1, s_2) graph, as well as the mixture distribution (x_1, x_2). The graph on the lower left, representing the source estimates obtained by the basic DH network, shows that, for many points, the algorithm diverged, as expected. In the figure, we can see the *locus* of the points where $\mu_{\max} = 1$, from equation (3.27), after which we expect the algorithm will diverge.

Finally, on the last graph, we plot the source estimates obtained by the variable step size strategy, and the result can be seen to be more accurate. For the basic DH network, we obtained a *mean-squared error* (MSE) of 2.67×10^{-2} for source 1, and 2.65×10^{-2} for source 2, consider-

ing only the points that converged. Additionally, we have also verified that 67.4% of the points converged. After implementing the variable step size, we obtain 100% convergence rate, and the MSE is reduced to 2.58×10^{-4} and 1.21×10^{-5} for sources 1 and 2 respectively.

In a subsequent simulation, we used $L_1 = 0.5$, $L_2 = 0.8$, $Q_1 = -0.5$ and $Q_2 = 0.8$. The sources had a uniform distribution in the interval $[-2,2]$. The obtained results can be seen in Fig. 3.5.

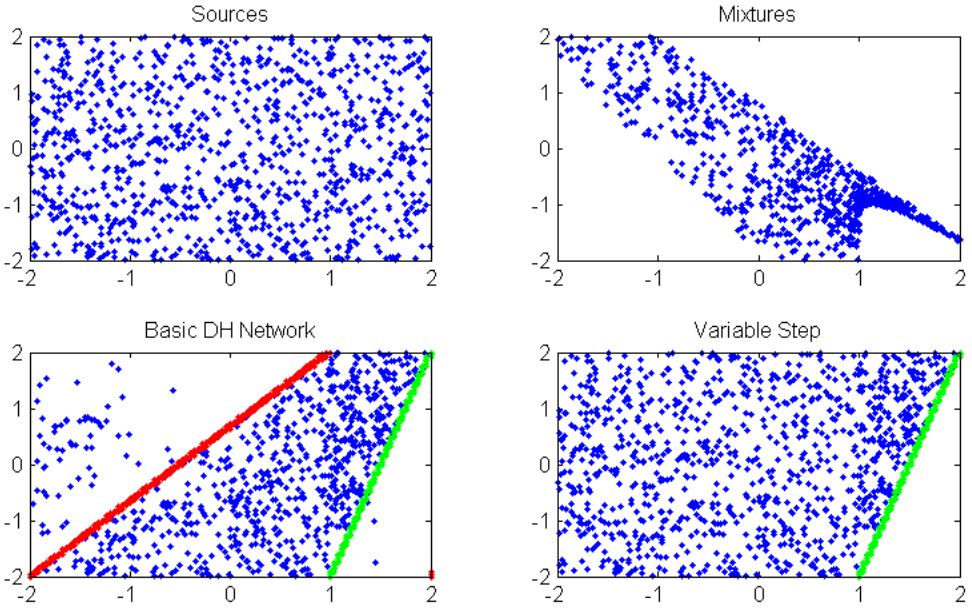

Figure 3.5 - Second simulation for basic DH network and variable step size

In this simulation, we can see that neither of the algorithms could fully separate the sources, but we can again see that the method with a variable step size obtained a better result. The basic network can fail to converge for two reasons – either the algorithm is not locally stable (which is represented by the red line); or because the LQ model does not satisfy the invertibility condition (represented by the green line). As can be seen, the variable step network is able to converge in the first case, similarly to the results presented in the previous simulation.

Table 3.1 summarizes the results:

Table 3.1 - Comparison between basic DH and the Variable step size networks

	MSE (s_1, s_2)	Convergence rate
Basic DH network	$(0.24, 0.64)$	68.1%
Variable Step Size network	$(0.11, 0.50)$	87.0%

In both simulations, the variable step size clearly proved to be a successful strategy to increase the stability of the algorithm, as expected from section 3.3.4.

3.5.2 Simulations on Newton-based Algorithm

Our next simulation intends to compare the second-order network based on the Newton's method with the variable step size network – which, as shown in 3.3.3, is equivalent to the extended DH network.

The mixing parameters used were $L_1 = -L_2 = 0.8$ and $Q_1 = -Q_2 = 0.2$. The sources were uniformly distributed in the $[-5; 5]$ interval. The results obtained can be seen in Fig. 3.6.

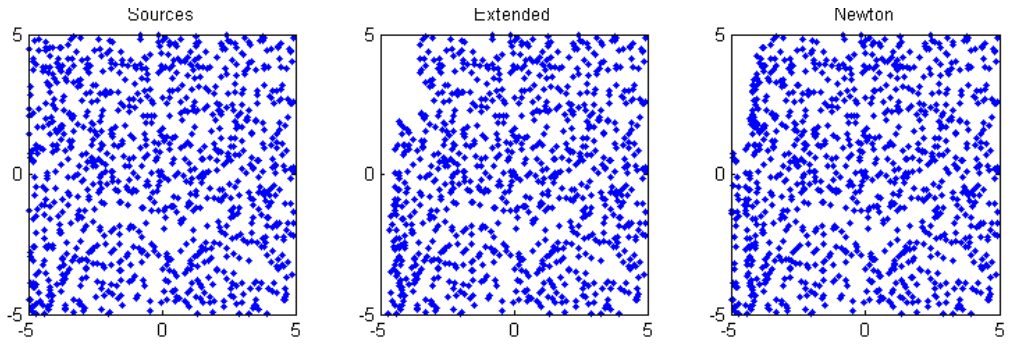

Figure 3.6 - Simulation results comparing the extended DH and Newton-based networks

As we can see, both methods present a good result. A more thorough analysis can be seen in Table 3.2.

Table 3.2 - Comparison between extended DH and Newton-based networks

	MSE (s_1, s_2)	Convergence rate	Elapsed time (ms)
Extended DH network	$(0.655, 0.849)$	100%	46
Newton-based network	$(0.006, 0.501)$	100%	282

As shown by the MSE values, the second-order algorithm is more accurate, but at the cost of additional elapsed time. This was expected, since the algorithm is computationally more expensive. If the application requires real-time source separation, the Newton-based network might not be suitable – however, for other applications where the response time is not essential, the proposed algorithm should obtain better results.

While the simulation shows a convergence rate of 100%, we can see that some points on the upper left portion of the graph could not be reached. We can conclude, thus, that the mixtures corresponding to these source points converged, but not to the correct solution. This happened because one of the separability conditions given in 2.3.3 was violated – namely, the condition about the Jacobian sign invariance. In fact, if we plot the line at which the Jacobian is zero (and hence change signs) on the same graph, we obtain Fig. 3.7, where this effect can be clearly seen.

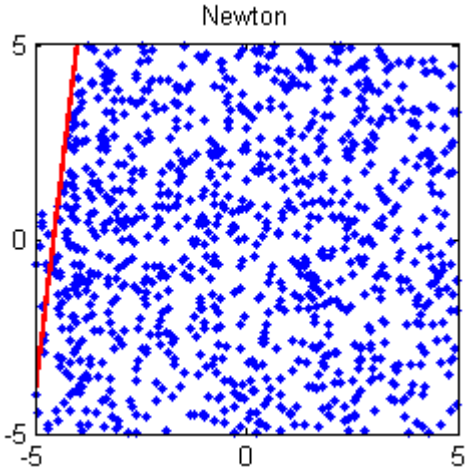

Figure 3.7 - Source estimates along with the frontier where the Jacobian changes sign (in red)

Another interesting test is to verify the effect of an increase in the amplitude range of the sources – if the sources are uniformly distributed in the interval [-10; 10], the obtained results are summarized in Fig. 3.8.

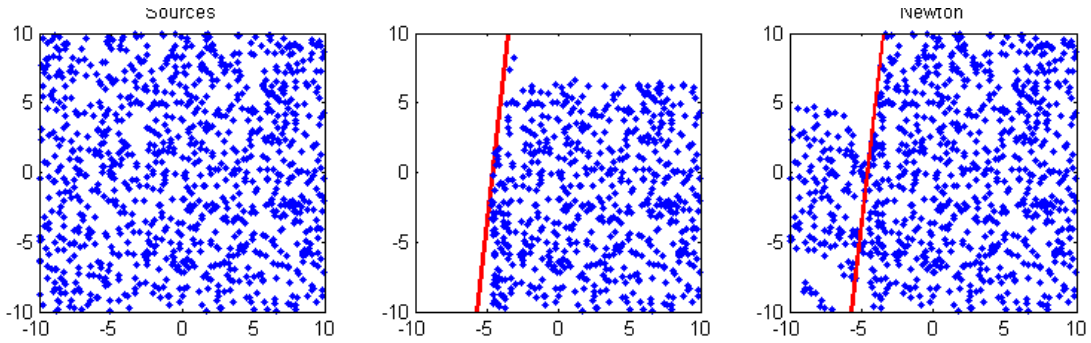

Figure 3.8 - Simulation with source amplitude 10

Table 3.3 - Comparison between methods for source amplitude 10

	MSE (s_1, s_2)	Convergence rate	Elapsed time (ms)
Extended DH network	$(11.5, 199.0)$	100%	71
Newton-based network	$(1.4, 115.8)$	100%	487

In this simulation, we can see that even on the right side of the $J_G = 0$ line, the first-order network does not necessarily converge to the correct solution, as can be seen on the upper portion of the graph. Neither of them performs well on the left side of the line, but that was expected, since it violates the separability conditions.

The MSE for both methods were very high, though that could be expected with amplitudes as high as 10 and a significant portion of the set converging to the wrong solution.

3.5.3 Simulation – Signals Represented over Time

In the next simulation, we decided to represent the signal over time. The first source, plotted on Fig. 3.9, is a sinusoidal wave, while the second one will be random noise. The mixing parameters were $L_1 = -Q_1 = 0.5$, $L_2 = -Q_2 = 0.8$, the same parameters for the unstable mixture from simulation 3.5.1.

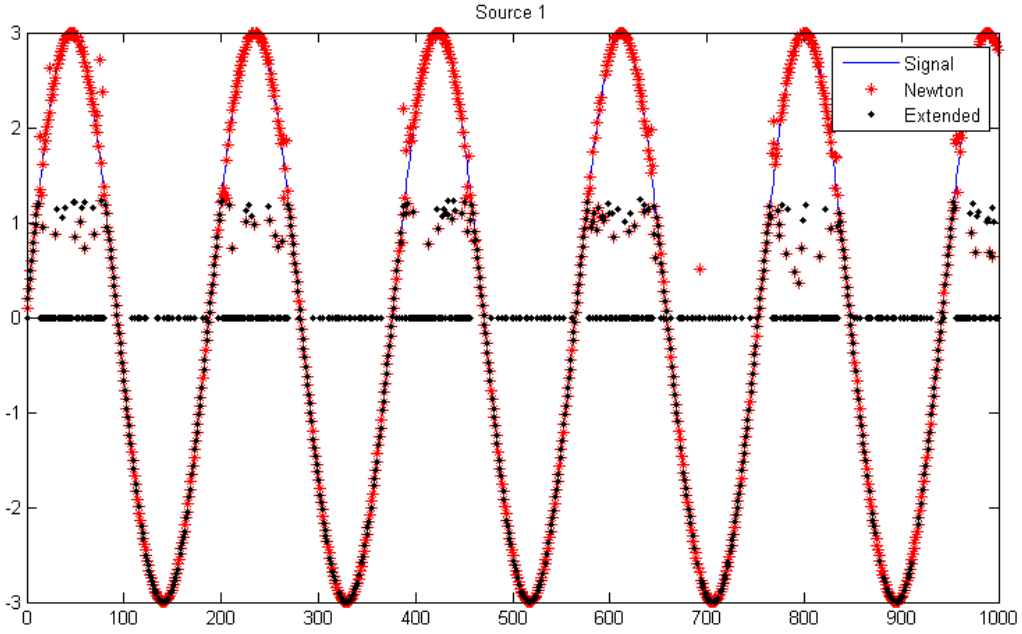

Figure 3.9: Simulation representing sinusoidal wave over time

In the simulation, we can clearly see that the Newton-based algorithm obtained a much better estimate than the one obtained by the extended network, especially when the signal value was above $s_1 = 1$, where presumably the network is no longer locally stable. The MSE obtained for source 1 was 0.25 for the Newton approach, and 1.28 for the extended DH network.

3.5.4 Simulation – Newton-Based Algorithm with More than Two Sources

In this section of the simulations, we will check how the Newton-based algorithm performs when separating three sources. The sources were uniformly distributed in the interval $[-0.75,0.75]$, and the mixing coefficients were randomly selected in $[-1,1]$. The simulation was run several times, and the algorithm was always able to obtain good source estimates. One of the performed simulations can be seen in Fig. 3.10.

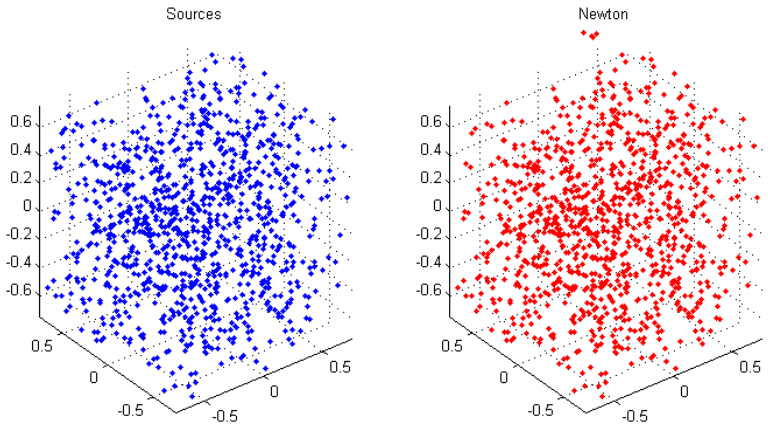

Figure 3.10 - Newton-based method for 3 sources

In the above simulation, the MSE was 0.008 and the elapsed time was 539 ms.

Since it is difficult to visualize more than two sources when plotted in the source space, we can plot each source individually over time. In the next simulation, we used 9 sources and random mixing parameters selected from the interval $[-1,1]$. One of the sources was a random noise, and the remaining 8 sources were random sinusoidal waves, with amplitudes, periods and phases uniformly distributed in the ranges $[-1;1]$, $[0,50]$ and $[0,2\pi]$ respectively. The results can be seen in Figs. 3.11 to 3.13.

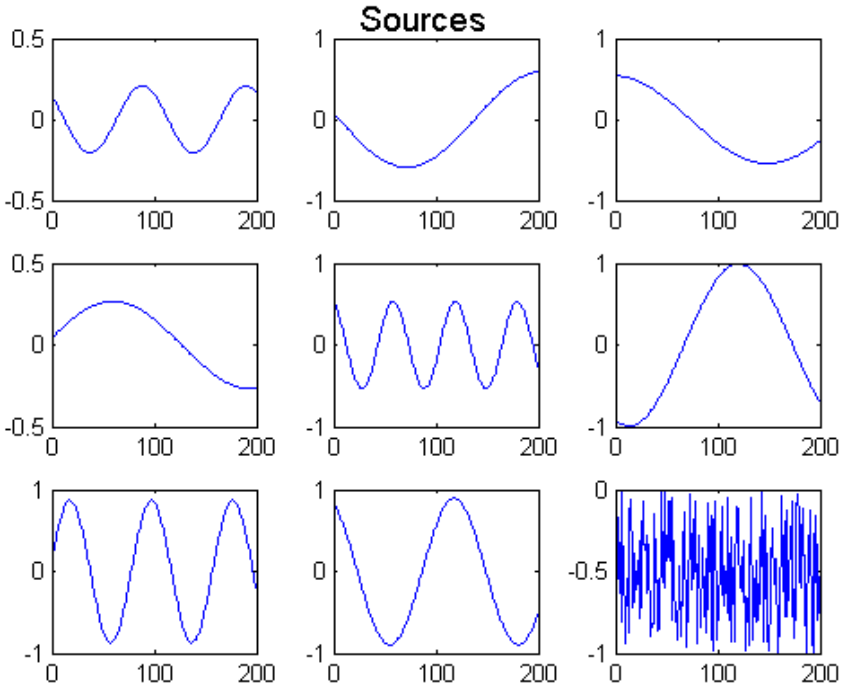

Figure 3.11 – Original sources for the simulation with n=9

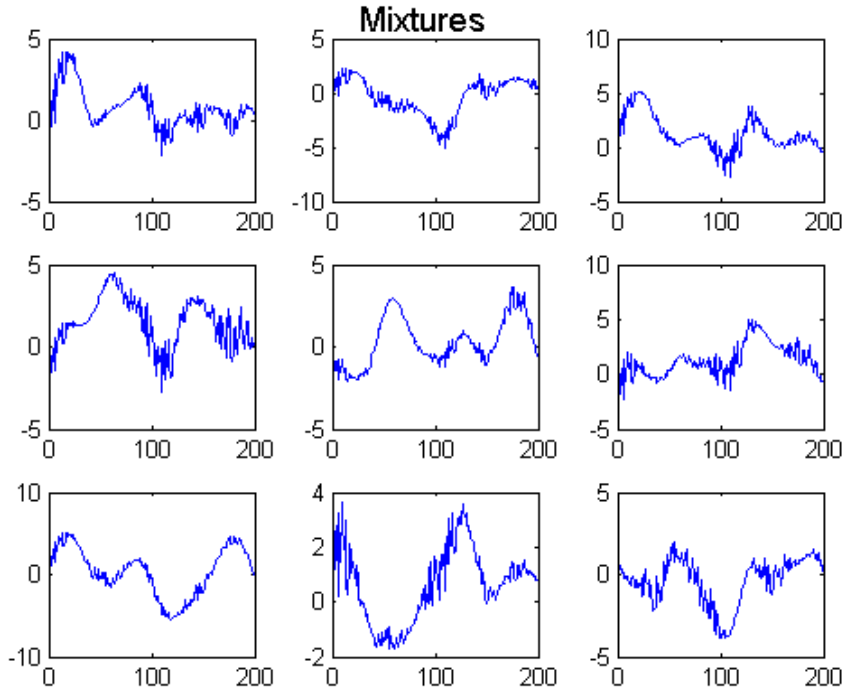

Figure 3.12 – Mixtures obtained for simulation with n=9.

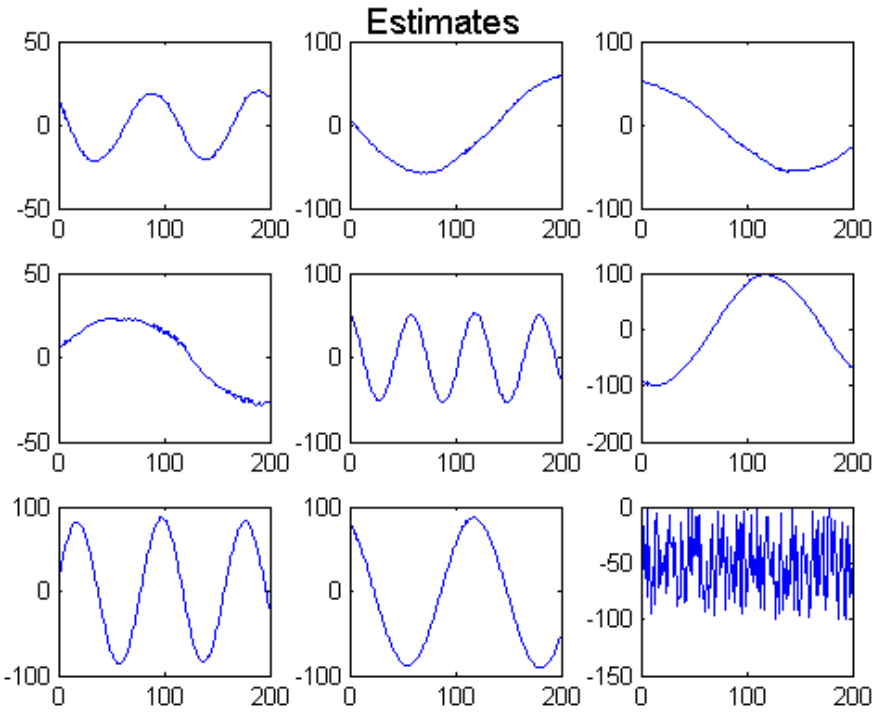

Figure 3.13 – Estimates obtained by Newton-based algorithm

As we can see, the source estimates obtained by the proposed algorithm presents very accurate results. The waveforms obtained have different amplitudes, which is acceptable in the paradigm of source separation problems. After appropriately normalizing the estimates to be the same as that of the sources, the RMS obtained ranged from 0.017 to 0.052, showing the results were very precise.

CHAPTER 4

4 ON THE ESTIMATION OF THE MIXING PARAMETERS

In chapter 3, we analyzed how we can use recurrent networks to solve the non-blind part of the source separation problem. In this chapter, we will focus on the other part of the problem – the estimation of the mixing parameters. As mentioned in chapter 2, both strategies complement each other when solving the blind source separation problem.

4.1 MAXIMUM LIKELIHOOD ESTIMATION

We will start by presenting in more detail the original algorithm proposed by Hosseini and Deville [14], where they applied a maximum likelihood (ML) principle in order to devise an algorithm capable of estimating the mixing parameters. In their algorithm, they assumed the following:

1. The mixture is determined – that is, the number of sources and mixtures are equal.
2. Each source s_i is independent and identically distributed (i.i.d.) with pdf $f_{s_i}(.)$. As the sources are statistically independent, the joint pdf satisfies $f_{\boldsymbol{s}}(.) = \prod_{i=1}^{N} f_{s_i}(.)$.

Let us denote the mixing parameters vector by $\boldsymbol{\theta}$. If the source pdfs $f_{s_i}(.)$ are known, then the mixture distribution depends only on $\boldsymbol{\theta}$, and we can denote it by $p_\theta(x_1(n), \dots, x_N(n))$. Considering that the source samples are obtained independently, the *likelihood function* can be defined as:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}\big(x_1(n), \dots, x_N(n)\big) \tag{4.1}$$

If we assume the mixing model to be invertible, the sources can be obtained by the inverse of the mixing system:

$$\boldsymbol{s}(n) = \mathcal{F}^{-1}[\boldsymbol{x}(n)] \tag{4.2}$$

which leads to:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \prod_{n=1}^{K} p_{\theta}\big(x_1(n), \dots, x_N(n)\big) \\ &= \prod_{n=1}^{K} \frac{f_{\boldsymbol{s}}\big(\hat{s}_1(n), \dots, \hat{s}_N(n)\big)}{|\hat{J}|} \\ &= \prod_{n=1}^{K} \frac{\prod_{i=1}^{N} f_{s_i}\big(\hat{s}_i(n)\big)}{|\hat{J}|} \end{aligned} \tag{4.3}$$

where $\hat{s}_i$ are the estimates of the sources, and $\hat{J}$ is the estimate of the Jacobian of the mixing system $\mathcal{F}[.]$. The basic principle of the ML strategy is to find the parameters $\boldsymbol{\theta}$ that maximize the likelihood of the mixture set, as given by equation (4.3).

Maximizing the likelihood function $L(\boldsymbol{\theta})$ is equivalent to minimizing the function $C(\boldsymbol{\theta}) = -\frac{1}{K}\log L\,(\boldsymbol{\theta})$, which can also be written in the form:

$$C(\boldsymbol{\theta}) = -E_K\left[\sum_{i=1}^{N} \log f_{s_i}(\hat{s}_i)\right] + E_K\big[\log|\hat{J}|\big] \tag{4.4}$$

where $E_K[.]$ is the time average operator.

Let us define the *score function* as:

$$\psi_{s_i}(s_i) = -\frac{d \log f_{s_i}(s_i)}{ds_i} \tag{4.5}$$

In order to obtain the mixing parameters $\boldsymbol{\theta}$ that minimize the cost function given in (4.4), Hosseini and Deville used the gradient method [14], which ideally yields a parameter vector $\widehat{\boldsymbol{\theta}}$ that makes the derivative of C vanish:

$$\frac{dC}{d\widehat{\boldsymbol{\theta}}} = \left(\sum_{i=1}^{N} E_K \left[\psi_{s_i}(\hat{s}_i) \frac{\partial \hat{s}_i}{\partial \widehat{\boldsymbol{\theta}}} \right] \right) + E_K \left[\frac{1}{|\hat{J}|} \frac{\partial |\hat{J}|}{\partial \widehat{\boldsymbol{\theta}}} \right] \tag{4.6}$$

When applied specifically to the LQ model for two sources, $\boldsymbol{\theta} = [L_1, L_2, Q_1, Q_2]^T$ and (4.6) can be rewritten in terms of equations (4.7) to (4.11):

$$\frac{dC}{d\widehat{\boldsymbol{\theta}}} = E_K \left[\frac{D_1}{|\hat{J}|}, \frac{D_2}{|\hat{J}|}, \frac{D_3}{|\hat{J}|}, \frac{D_4}{|\hat{J}|} \right]^T \tag{4.7}$$

where:

$$\begin{aligned} D_1 = {} & \psi_{s_1}(\hat{s}_1)\left(1 - \hat{Q}_2 \hat{s}_1\right)\hat{s}_2 + \left(\hat{L}_2 + \hat{Q}_2 \hat{s}_2\right)\left(\psi_{s_2}(\hat{s}_2)\hat{s}_2 - 1\right) \\ & - \left[\left(\hat{Q}_2 + \hat{L}_2 \hat{Q}_1\right)\left(1 - \hat{Q}_2 \hat{s}_1\right) + \left(\hat{Q}_1 + \hat{L}_1 \hat{Q}_2\right)\left(\hat{L}_2 + \hat{Q}_2 \hat{s}_2\right)\right]\hat{s}_2 / |\hat{J}| \end{aligned} \tag{4.8}$$

$$\begin{aligned} D_2 = {} & \psi_{s_2}(\hat{s}_2)\left(1 - \hat{Q}_1 \hat{s}_2\right)\hat{s}_1 + \left(\hat{L}_1 + \hat{Q}_1 \hat{s}_1\right)\left(\psi_{s_1}(\hat{s}_1)\hat{s}_1 - 1\right) \\ & - \left[\left(\hat{Q}_1 + \hat{L}_1 \hat{Q}_2\right)\left(1 - \hat{Q}_1 \hat{s}_2\right) + \left(\hat{Q}_2 + \hat{L}_2 \hat{Q}_1\right)\left(\hat{L}_1 + \hat{Q}_1 \hat{s}_1\right)\right]\hat{s}_1 / |\hat{J}| \end{aligned} \tag{4.9}$$

$$\begin{aligned} D_3 = {} & \left[\psi_{s_1}(\hat{s}_1)\left(1 - \hat{Q}_2 \hat{s}_1\right) + \psi_{s_2}(\hat{s}_2)\left(\hat{L}_2 + \hat{Q}_2 \hat{s}_2\right)\right]\hat{s}_1 \hat{s}_2 - \hat{L}_2 \hat{s}_1 - \hat{s}_2 \\ & - \left[\left(\hat{Q}_2 + \hat{L}_2 \hat{Q}_1\right)\left(1 - \hat{Q}_2 \hat{s}_1\right) + \left(\hat{Q}_1 + \hat{L}_1 \hat{Q}_2\right)\left(\hat{L}_2 + \hat{Q}_2 \hat{s}_2\right)\right]\hat{s}_1 \hat{s}_2 / |\hat{J}| \end{aligned} \tag{4.10}$$

$$D_4 = \left[\psi_{s_2}(\hat{s}_2)\left(1-\hat{Q}_1\hat{s}_2\right)+\psi_{s_1}(\hat{s}_1)\left(\hat{L}_1+\hat{Q}_1\hat{s}_1\right)\right]\hat{s}_1\hat{s}_2-\hat{L}_1\hat{s}_2-\hat{s}_1 \\ -\left[\left(\hat{Q}_1+\hat{L}_1\hat{Q}_2\right)\left(1-\hat{Q}_1\hat{s}_2\right)+\left(\hat{Q}_2+\hat{L}_2\hat{Q}_1\right)\left(\hat{L}_1+\hat{Q}_1\hat{s}_1\right)\right]\hat{s}_1\hat{s}_2/|\hat{J}| \quad (4.11)$$

The actual calculation for obtaining equations 4.7 to 4.11 is quite lengthy, being the interested reader referred to [14].

4.1.1 Problems with the ML Approach

One of the limitations of the ML approach described in section 4.1 is that it requires the derivatives of the pdf of the sources to be known. When that is not the case, the pdfs can be estimated by histograms [33], or by more advanced methods (for example, [17, 37]); however, estimating the derivative is much less reliable [17].

Another drawback of the method described on section 4.1 is that the cost function $C(\boldsymbol{\theta})$ can have multiple local optima, which means that the gradient method might not converge to the globally best solution. These points will be addressed in the following.

4.2 EVOLUTIONARY ALGORITHMS

In an attempt to mitigate the problems described in section 4.1.1, we propose the use of an evolutionary optimization algorithm, which, unlike the gradient method, does not require knowledge of the derivatives of the source pdfs. Furthermore, the algorithm presents a global search potential that should allow the likelihood cost function to be more thoroughly explored, thereby increasing the possibility of convergence to the ideal parameter configuration [2].

Evolutionary algorithms can be defined, in simple terms, as populational metaheuristics that apply the biological principles of evolution (e.g. natural selection, random mutations and strategies of reproduction) as means of obtaining optimal "individuals" (i.e. solutions) according to a specific *fitness* function [2]. From a pragmatic standpoint, the main aspects of the application of evolutionary algorithms are [2]:

1. **Individual**: in an evolutionary algorithm, each individual is a vector containing all the relevant parameters to the problem. A set of individuals is defined as a *population*, and the population existing at a given iteration is called a *generation.*

2. ***Fitness*** **function**: the fitness function quantifies how well the individual responds to the optimization problem to be solved. Similarly to what happens in evolutionary biology, an individual with higher fitness has a better chance of surviving and reproducing.

3. **Reproduction**: an individual may reproduce sexually and/or asexually, depending on the evolutionary algorithm being used. Asexual reproduction essentially amounts to cloning the individual, while sexual reproduction creates a new offspring with combined characteristics of its parent individuals. The methods on how to combine the traits also depend on the algorithm.

4. **Mutation**: any individual may randomly undergo a mutation from one generation to the next. When that happens, one or more of its traits are altered, in a manner that depends on the implementation of the algorithm. A mutation does not necessarily lead to fitness improvement.

5. **Natural selection**: from one generation to the next, the population changes as individuals reproduce, mutate or die. It might also include the addition of completely new individuals to increase variability. Because the strategies of reproduction and survival are based on the fitness of each individual, we can expect that, over time, the individuals of each population become closer and closer to the optimal solution.

A general evolutionary algorithm can be summarized as follows [2]:

Pseudo-code for general evolutionary algorithm:

1. Randomly generate the individuals of the initial population.
2. Evaluate the fitness of all individuals.
3. Perform a reproduction step. The actual reproduction strategy depends on each algorithm implementation, and it can be such that individuals with higher fitness have a better chance of reproducing.
4. Individuals may randomly experience mutations.
5. If necessary, remove some individuals from the population. Individuals with lower fitness should be more likely to be removed.
6. Repeat steps 2 to 5 until the stopping criteria are met.

4.2.1 The opt-aiNet Algorithm

For our particular problem, we decided to use a specific evolutionary algorithm called opt-aiNet [3]. It is an optimization version of an important immune-inspired algorithm designed for pattern recognition, called aiNet [4], which is based on the immune network theory and on the *clonal selection principle*. One of the advantages of the opt-aiNet is its good performance for optimization of multi-modal functions [3], which can be very useful for problems with nonlinear / multivariate functions like ours.

In the context of this algorithm, each individual will be referred to as a *network cell* or simply as a *cell*. The Euclidean distance between two cells will be defined as the *affinity* between the cells. A pseudo-code for the opt-aiNet can be described as follows [3]:

Pseudo-code for the opt-aiNet:

1. Randomly generate the individuals of first population.

2. While the stopping criteria are not met, do:

2.1 Evaluate the fitness of all cells, calculate the average and normalize the fitness vector.

2.2 Generate N_C clones of each cell.

2.3 Mutate each clone according to the fitness of the parent, and keep the parent cell. The mutation should follow equation (4.12):

$$\boldsymbol{c}' = \boldsymbol{c} + \alpha\, \boldsymbol{N}(0,1) \tag{4.12}$$

where $\boldsymbol{c}'$ is the mutated cell, $\boldsymbol{c}$ is the parent cell, $\alpha = (1/\beta)\exp(-f^*)$, β is a constant, f^* is the normalized fitness, and $\boldsymbol{N}(0,1)$ is a vector of the same dimension as $\boldsymbol{c}$, and where each element is independently chosen according to a Gaussian random variable with $\mu = 0$ and $\sigma = 1$.

2.4 Determine the fitness of all individuals.

2.5 For each parent cell, select the clone with the highest fitness (including the parent itself) and calculate the average fitness of the newly obtained population.

2.6 Repeat steps 2.1 to 2.5 until the average fitness can be considered, according to a predefined threshold, to be stable.

2.7 Determine the affinities of all cell pairs in the network. If an affinity is below the suppression threshold σ_s, delete the least fit cell of the pair.

2.8 Introduce random cells in the population. The amount of added cells is a percentage $d\%$ of the current population size.

3. End While (step 2).

4. The fittest cell of the population is the estimate of the solution to the optimization problem.

4.2.2 Defining a Fitness Function

One of the questions that might be posed when implementing the opt-aiNet algorithm is how to define a suitable fitness function. The *modus operandi* of the algorithm is such that a global optimum of the fitness function will ideally be found, and, since the algorithm is being used to minimize a cost function $cost(.)$, it is necessary to use a function that maps the maximum value of the fitness $fitness(.)$ to the minimum value of $cost(.)$.

Because the mutation on the cloned cells is proportional to $\exp(-f^*)$, we decided to avoid negative values for the fitness, hence preventing excessively large mutations from happening. Since the cost might reach values close to 0, it is also important to avoid functions of the form of $1/cost(.)$. Some of the possibilities we tested were:

1. $fitness(.) = 1/\big(1 + cost(.)\big)$
2. $fitness(.) = \exp\big(-cost(.)\big)$
3. $fitness(.) = c_{\max} - cost(.)$

As will be shown in section 4.5, in the simulations, the actual choice of the fitness function did not significantly affect the performance of the algorithm i.e. all choices proved themselves to be viable.

4.3 MUTUAL INFORMATION

When the maximum likelihood principle was applied in section 4.1, we adopted the hypothesis that the sources are independent to be able to separate $f_{\boldsymbol{s}}(s_1, s_2, \dots, s_N)$ into $\prod_{i=1}^{N} f_{s_i}(s_i)$. However, instead of trying to find the parameters that maximize the likelihood function, we could use *mutual information* (MI) to search for the parameters that maximize the degree of independence between sources, resorting more straightforwardly to the notion of ICA.

The primary advantage of using an MI based cost function instead of the one proposed in the literature is that we will require less *a priori* information on the sources distribution, as will be explained in the following section.

Mutual information is also a more accurate way to measure the degree of dependence between two variables – that is, $I(X;Y) = 0$ if and only if X and Y are independent. The mutual information is also nonnegative, so, in order to obtain sources that are as independent as possible, it is necessary to minimize their mutual information. The mutual information between two random variables can be described by the following equation [30]:

$$I(X;Y) = \int_X \int_Y p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dy\, dx \tag{4.13}$$

Equation (4.13) essentially measures the *Kullback-Leibler divergence* [30] between the joint pdf and the product of the marginal ones, which can be denoted by:

$$I(X;Y) = D_{KL}(p(x,y) \| p(x)p(y)) \tag{4.14}$$

Another way of calculating the mutual information is using the signal entropies [30]:

$$I(X;Y) = h(X) - h(X|Y) \tag{4.15}$$

where $h(X) = \int_X p(x) \log p(x)\ dx$ is the differential entropy [30]. The extension to multiple random variables follows the definition given in equation (2.10).

4.4 ESTIMATION OF THE SOURCE PDF

The strategy based on mutual information has essentially the same problem as the one utilizing the ML – it requires the *a priori* knowledge of the source pdfs (but not the derivatives). The cost function can be defined either by equation (4.4), using the maximum likelihood princi-

ple, or using mutual information. Regardless of which method we use to define a cost function, though, the pseudo-code for solving our BSS problem can be described as follows:

Pseudo-code for BSS using opt-aiNet:

1. Start with a randomly generated parameter vector $\boldsymbol{\theta}$.
2. Solve the problem as if it were non-blind (as described in chapters 2 and 3).
3. Calculate the current value of the cost function.
4. Estimate a better parameter vector using the opt-aiNet algorithm (as described in section 4.2).
5. Repeat steps 2 to 4 until a stopping criterion is met (e.g.: cost function stabilizes, maximum number of iterations reached, parameter vector stabilizes, etc.).

In the algorithms requiring the mutual information, it can be estimated by histograms [33] (if the sample size is big enough) or by an adaptive partitioning method [37].

For the original algorithm presented by Deville and Hérault [14], the iterations use the pdf derivatives, which is more difficult to accurately estimate than the distribution. In their paper, they used sources with *generalized Gaussian* distributions on the simulations. A generalized Gaussian random variable has the following pdf [38]:

$$f(x) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left(-\left(\frac{|x-\mu|}{\alpha}\right)^{\beta}\right) \tag{4.16}$$

where $\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t}\, dt$ is the Gamma function. Using the generalized Gaussian distribution, we obtain a score function of the form:

$$\psi_{s_i}(s_i) = \frac{\beta}{\alpha}\, |s_i - \mu|^{\beta-1}\, .sgn(s_i - \mu) \tag{4.17}$$

The generalized Gaussian is a very flexible distribution that can, depending on the values of the parameter, be *leptokurtic* (super Gaussian) - as the Laplace distribution - or *platykurtic* (sub Gaussian), like quasi-uniform distributions. In our simulations, we generated the sources as having generalized Gaussian distributions, and, for the original algorithm presented in [14], we assumed that the parameters were known.

4.5 SIMULATIONS AND RESULTS

In this section, we will present simulations for the BSS problem with two sources, using the evolutionary algorithm to optimize the parameter search. We also compared it with the original algorithm presented by Deville and Hosseini in [14] – which, however, required the source pdfs to be known. Because the original algorithm benefitted from "more *a priori* information" about the sources, the proposed solutions were not expected to perform better in the simulations.

Since these simulations will only measure the performance of the parameter estimation stage, we used the analytical solution for the non-blind portion of the algorithm. In a real-life application with more than two sources, or with nonlinear models other than the LQ, that would not be possible – however, the algorithm for solving a general BSS problem is very modular, and the module pertaining to the non-blind solution was already discussed in chapters 2 and 3.

In our simulations, different values of the parameters were tested, and the values that were selected to obtain a balance between performance and computational cost can be seen in Table 4.1.

Table 4.1 - opt-aiNet parameters used in the simulations

Parameter	Value	Description
N	20	Initial number of cells
N_c	10	Number of clones created per cell on each iteration
σ_s	0.5	Suppression threshold
d	15%	Percentage of newcomers added on each iteration
β	20	Mutation constant
N_{gen}	500	Maximum number of iterations

4.5.1 Comparing Different Fitness Functions

In this first test, we are only interested in choosing which model of fitness function we will use, among those presented in section 4.2.2. Therefore, we assumed that the source pdfs were known to minimize the interference from errors on the estimation (which shall be discussed in section 4.5.2).

We used a quasi-uniform distribution (generalized Gaussian distribution (GGD) with parameters $\{\mu, \alpha, \beta\} = \{0,1,15\}$), mixing parameters $L_1 = -L_2 = 0.2$, $Q_2 = -Q_1 = 0.8$, and source amplitude further normalized to 0.5 after being generated (since the sources are post-normalized, the α parameter of the GGD actually is not relevant). The post-normalization of the sources was done to guarantee the invertibility of the model.

The simulation results were satisfactory for all three versions of the fitness function mapping, and, in all cases, the results were similar to those shown in Fig. 4.1.

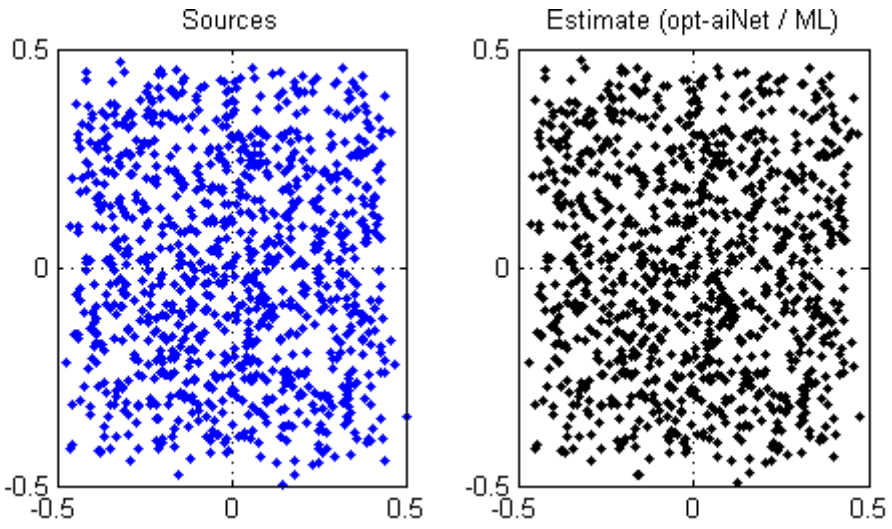

Figure 4.1 - Estimates obtained using opt-aiNet + ML-based cost

Because the result was good in all cases, not much can be concluded from the figures themselves, so we will use the *root mean square* (RMS) error and the elapsed time on each case to establish grounds for comparison. These values can be seen in Table 4.2.

Table 4.2 - Comparison between fitness functions

fitness	RMS ($\times 10^{-3}$)	Elapsed time (s)
$f = \exp(-cost(.))$	5.7	9.8
$f = \frac{1}{1 + cost(.)}$	7.4	9.4
$f = c_{\max} - cost(.)$	16.4	7.6

It can be seen that the last model, by being computationally simpler, converges faster, but ends up with higher RMS. If the running time is the most important aspect, then this should clearly be the choice. On the other hand, if time is not essential, the first two models offer similar times but lower RMS, with the exponential as the better choice. For our simulations, we decided to implement $f = \exp(-cost(.))$.

4.5.2 Estimating the Mutual Information

In order to use the MI-based cost function, we decided to compare two different MI-estimators. The simplest one is to estimate the source pdfs by their histogram, by dividing the set of points into several bins and assuming the distribution is discrete. We can then estimate the mutual information using the equation:

$$I(X;Y) = \sum_X \sum_Y p(x,y) \log \frac{p(x)p(y)}{p(x,y)} \tag{4.18}$$

A second method would be to directly estimate the mutual information without resorting to the source pdfs, by using an adaptive partitioning of the observation space [37].

We tested the algorithm with both methods of mutual information estimate and compared the errors and elapsed time to see which was better. In our tests, we had 1000 points, and the sources were distributed according to a quasi-uniform distribution. The results obtained can be seen in the Table 4.3.

Table 4.3 - Comparison between methods of mutual information estimation

Method	**RMS**	**Elapsed time (s)**
Histogram	4.7×10^{-3}	6.4
Adaptive Partitioning	4.1×10^{-2}	34.7

As we can see, the simpler, histogram-based method produced a more accurate and faster result. It is possible that the histogram-based estimate be more suited for uniform-like distributions, so we decided to carry out a second test with a Laplacian distribution. The obtained results are in Table 4.4.

Table 4.4 - Comparison for a Laplacian distribution

Method	**RMS**	**Elapsed time (s)**
Histogram	8.4×10^{-3}	6.0
Adaptive Partitioning	4.5×10^{-3}	35.0

In this case, we can see that a histogram-based estimate is still significantly faster, but for a Laplacian distribution it is less accurate. The plots obtained for this case can be seen in Fig. 4.2.

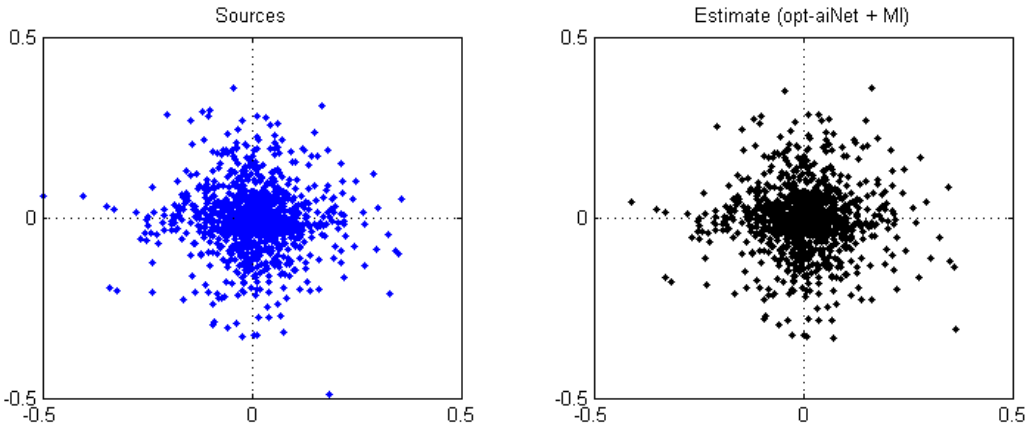

Figure 4.2 - Original sources and estimates for a Laplacian distribution

One possible explanation is that, for the histogram estimate, the source domain is divided into identical bins; however, because the Laplace distribution generates points that are too unevenly distributed, the bins farther from the center give too little information, while the bins closer to the center could give a more detailed estimate of the distribution if they were further separated into multiple, smaller sized, bins. In essence, the idea of using differently sized intervals in order to optimize the estimate is the main concept behind the adaptive partitioning algorithm [37], which might be why the method obtains more accurate estimates for unevenly distributed sources.

For our simulations, since most of our tests were done with evenly distributed sources, we decided to use the histogram-based distribution.

4.5.3 Simulation with Artificial Data: Comparison Between Methods

In this section, we will compare the three algorithms presented in this chapter: the one based on the gradient method, the opt-aiNet using an ML-based cost function and the opt-aiNet using an MI-based cost function. For the first two algorithms (based on the ML principle), the source pdfs were assumed to be known, and, for the MI-based method, the pdfs will be estimated via histograms. Even though the comparison is “unfair”, in the sense that the MI-based algorithm

has less *a priori* information about the sources than the ML ones, the comparison will be able to show that the MI algorithm can still provide competitive, and in some cases better, results.

4.5.3.1 Quasi-Uniform Distribution

Our first simulation was made using a quasi-uniform distribution ($GGD(0,1,15)$), Note that we cannot use a truly uniform distribution because its pdf is discontinuous and, as a result, its derivatives would not exist in all points.

The source estimate graphs for all three methods can be seen in Fig. 4.3, and the output data can be found on Tables 4.5 and 4.6.

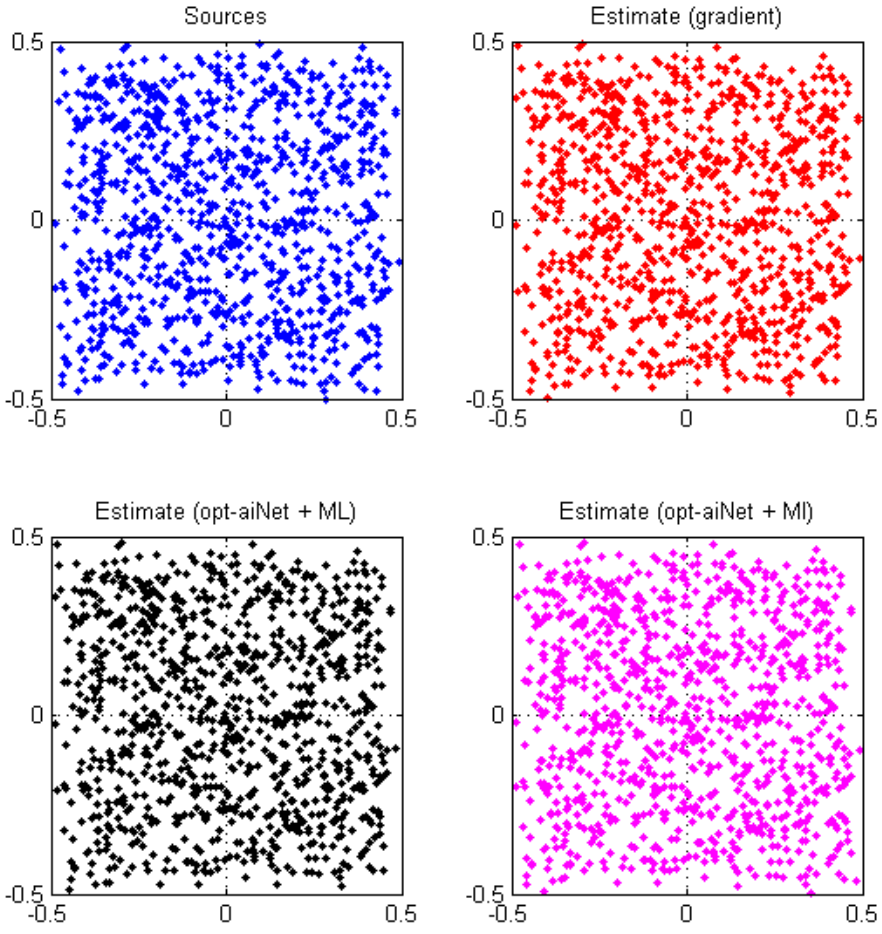

Figure 4.3 - Source estimates for different methods; quasi-uniform distribution

Table 4.5 - Error, time and cost for each method; quasi-uniform distribution

Algorithm	RMS	Elapsed time (s)	Estimated cost
Gradient + ML	0.0094	0.43	0.149 / 0.147
opt-aiNet + ML	0.0170	5.12	0.151 / 0.147
opt-aiNet + MI	0.0144	4.49	1.6×10^{-3}

Table 4.6 - Mixing parameters obtained for each method; quasi-uniform distribution

Algorithm	Obtained parameters
Real Parameters	$(-0.20, 0.20, -0.80, 0.80)$
Gradient + ML	$(-0.23, 0.20, -0.79, 0.70)$
opt-aiNet + ML	$(-0.26, 0.23, -0.85, 0.70)$
opt-aiNet + MI	$(-0.26, 0.22, -0.88, 0.75)$

In the simulation, we can see that both ML-based methods converge to an estimated cost close to the real sources ML cost, which is theoretically the minimum of the cost function. In the "estitmated cost" column of Table 4.5, the ML cost obtained for the real sources can be seen listed on the right. For the MI-based cost function, the theoretical minimum always equals to zero.

4.5.3.2 Quasi-Uniform with Random Mixing Parameters

In the next simulation, the mixing parameters were randomly chosen from the interval $[-1; 1]$. The source amplitudes were further adjusted so that the invertibility of the model is guaranteed, according to the following equation:

$$A_{\max} = \frac{|1 - L_1 L_2|}{|Q_1 + L_1 Q_2| + |Q_2 + L_2 Q_1|} \tag{4.19}$$

If the source amplitude does not exceed $A_{\max}$, it is easy to verify that the Jacobian $J = 1 - L_1 L_2 - (Q_1 + L_1 Q_2)s_2 - (Q_2 + L_2 Q_1)s_1$ never changes sign, which is one of the conditions for invertibility of the model for the case of two sources. The results obtained in the simulation can be seen on Fig. 4.4 and Tables 4.7 and 4.8.

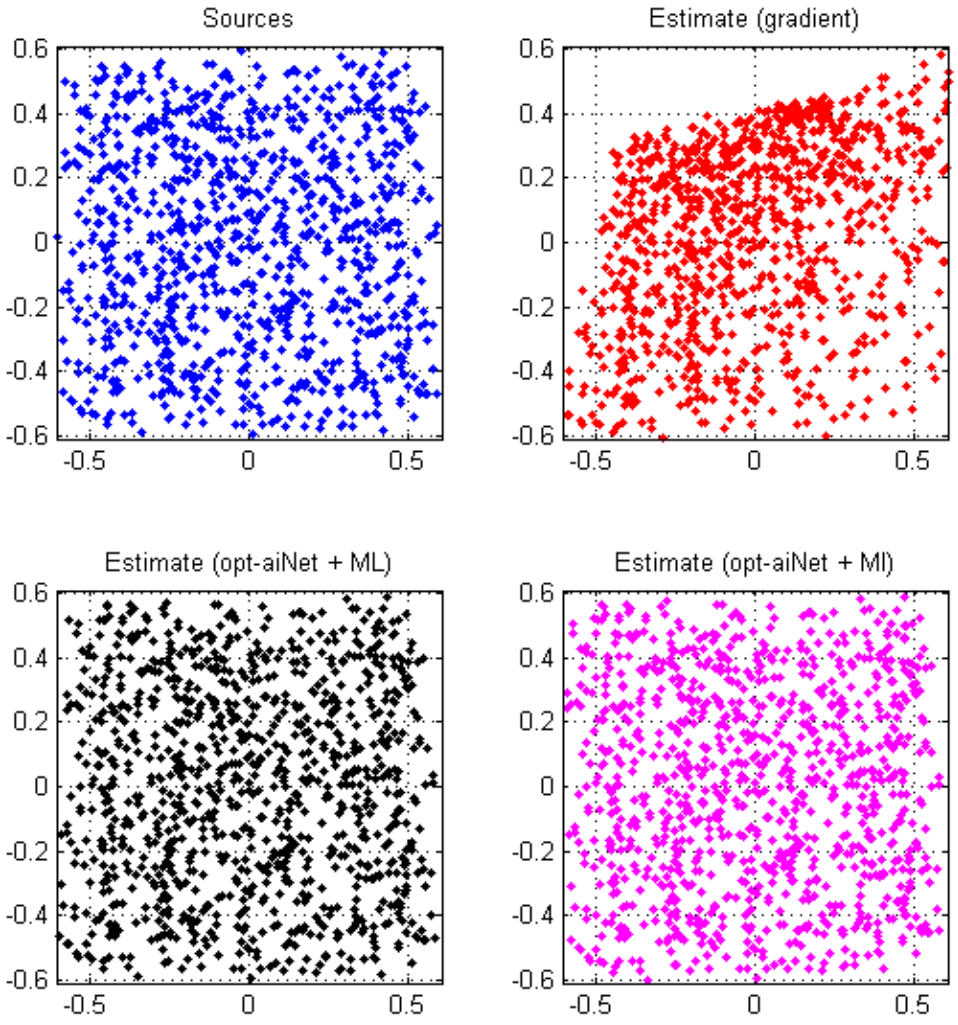

Figure 4.4 - Simulation results for random mixing parameters

Table 4.7 - Error, time and cost for each method; random mixing parameters

Algorithm	RMS	Elapsed time (s)	Estimated cost
Gradient + ML	0.1885	0.74	0.350 / 0.051
opt-aiNet + ML	0.0080	5.51	0.052 / 0.051
opt-aiNet + MI	0.0105	7.08	0.014

Table 4.8 - Mixing parameters obtained for each method

Algorithm	Obtained parameters
Real Parameters	(0.01,0.29,0.80,0.60)
Gradient + ML	(−0.34,0.86,0.03,1.48)
opt-aiNet + ML	(0.03,0.27,0.79,0.59)
opt-aiNet + MI	(0.03,0.27,0.86,0.51)

We can see in this simulation that the gradient-based method did not converge to the correct solution, and the estimated cost was significantly far from the desired value (0.35 instead of 0.05). The most likely explanation is that it has converged to a local minimum. As already mentioned, due to the nature of the gradient method, once the algorithm finds any local minimum it will converge to it. In contrast, evolutionary algorithms can avoid it by keeping multiple individuals on each iteration, and evolving them in different paths in parallel using a stochastic mutation operator.

4.5.3.3 Laplacian Distribution

In the next simulation, we utilized a Laplacian source distribution. A similar test with a Gaussian distribution was performed, and the results obtained were similar. The graphs and output data are shown in Fig. 4.5 and Tables 4.9 and 4.10.

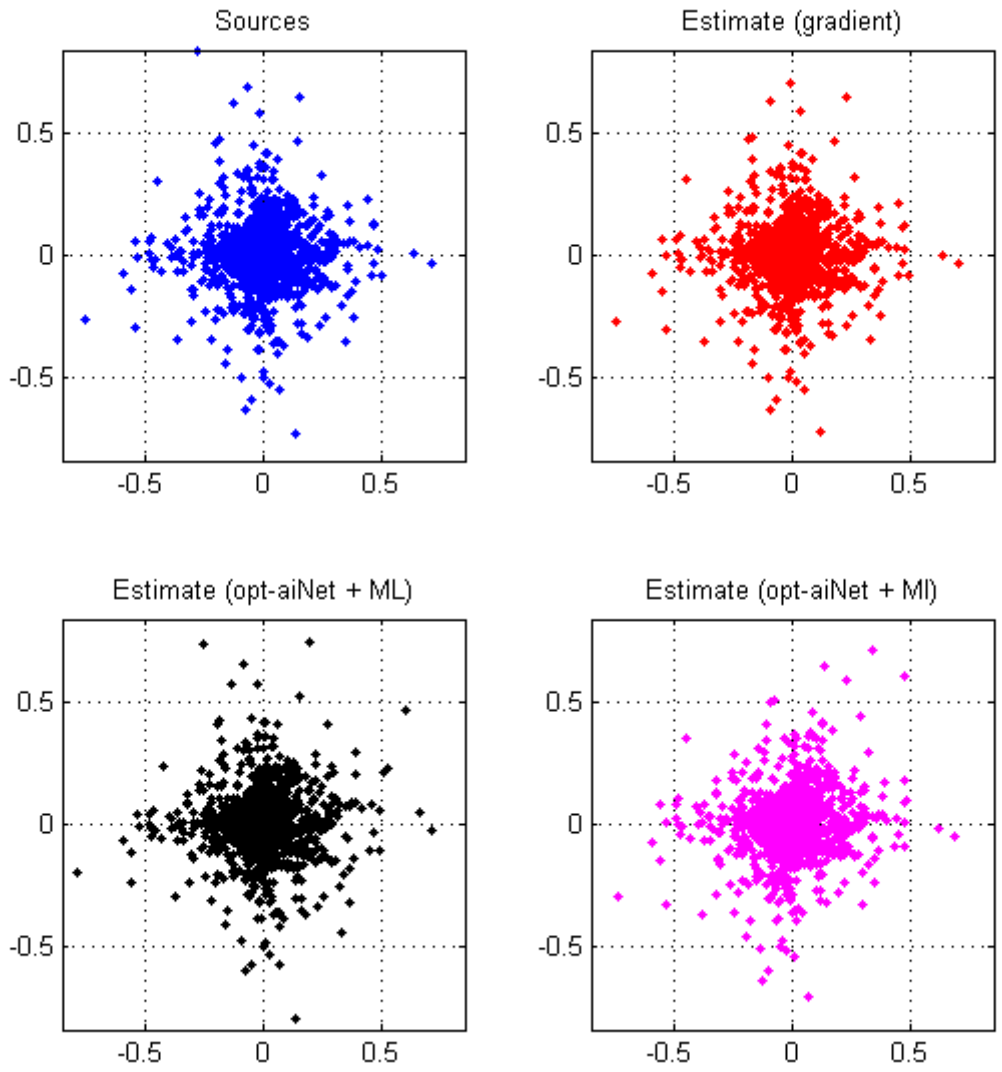

Figure 4.5 - Estimates for simulation with Laplacian distribution

Table 4.9 - Error, time and cost for each method; Laplacian distribution

Algorithm	RMS	Elapsed time (s)	Estimated cost
Gradient + ML	0.0045	0.27	1.931 / 1.931
opt-aiNet + ML	0.0078	2.85	1.933 / 1.931
opt-aiNet + MI	0.0231	5.07	0.023

Table 4.10 - Mixing parameters obtained for each method; Laplacian distribution

Algorithm	Obtained parameters
Real Parameters	$(0.05,0.04,0.83,0.30)$
Gradient + ML	$(0.09,0.03,0.91,0.20)$
opt-aiNet + ML	$(0.04,0.06,0.84,0.89)$
opt-aiNet + MI	$(0.20,0.04,0.83,-0.01)$

As we can see, the opt-aiNet algorithm using the MI-based cost function presents a slightly higher error, and also a higher elapsed time than both ML-based methods. However, it must be kept in mind that the MI algorithm does not require *a priori* knowledge of the source pdfs.

4.5.3.4 Nonzero Mean Quasi-Uniform Distribution

For our final simulation, we used sources that had a distribution with nonzero mean. For the linear mixing model, an offset to the sources mean would only cause an offset to the mixture mean. Depending on the application, constant offsets might not be important, and, as a result, it might be easier to simply ignore offsets and pre-treat the data, centering the offset at the origin [9].

However, for general nonlinear models, an offset to the sources mean result in a non-constant change to the mixtures, so its effect cannot be ignored. Fig.4.6 and Tables 4.11 and 4.12 present the results for this simulation:

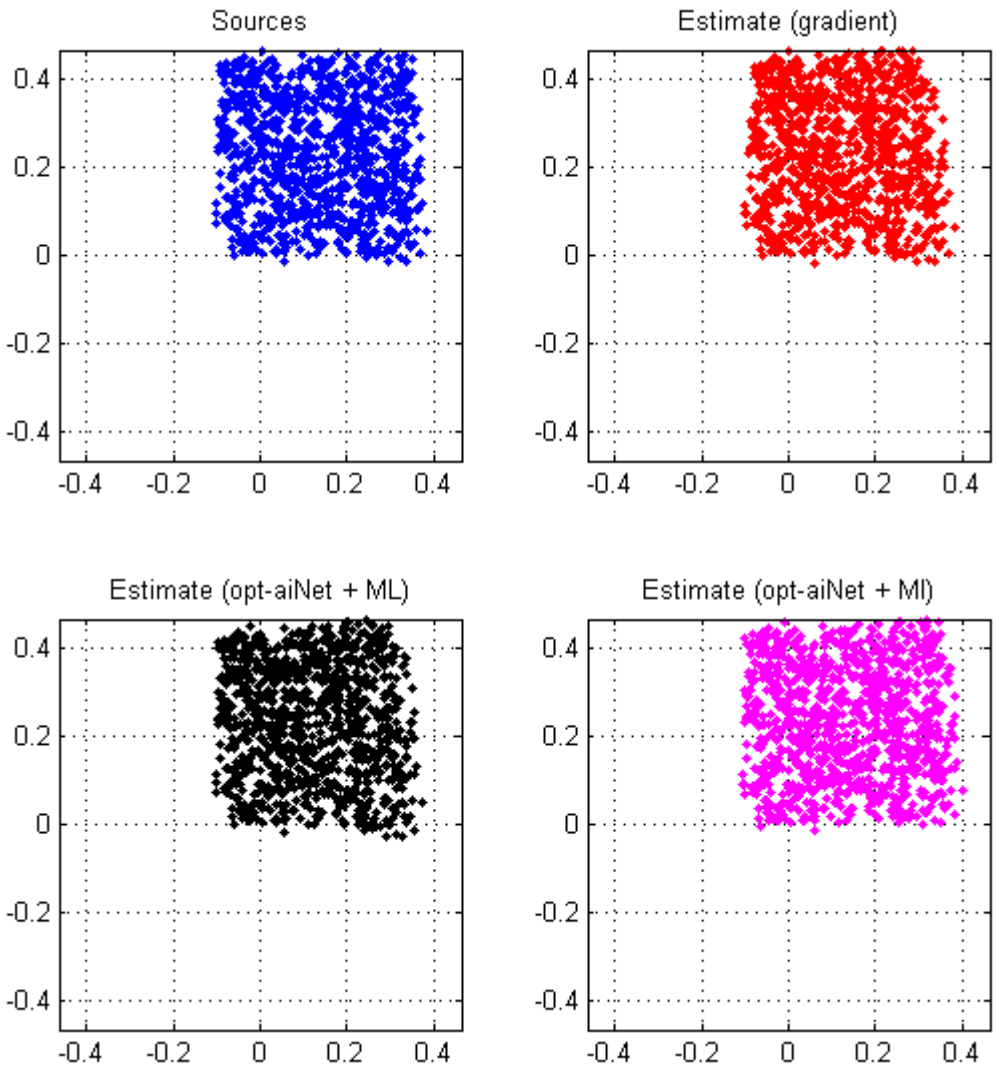

Figure 4.6 - Estimates for simulation with nonzero mean source distribution

Table 4.11 - Error, time and cost for each method; nonzero mean source distribution

Algorithm	RMS	Elapsed time (s)	Estimated cost
Gradient + ML	0.015	1.79	-0.61 / -0.65
opt-aiNet + ML	0.020	5.89	-0.60 / -0.65
opt-aiNet + MI	0.009	13.07	2.4×10^{-3}

Table 4.12 - Mixing parameters obtained for each method; nonzero mean source distribution

Algorithm	Obtained parameters
Real Parameters	(0.53,0.30,0.82,0.49)
Gradient + ML	(0.53,0.30,0.42,0.86)
opt-aiNet + ML	(0.51,0.28,0.44.0.95)
opt-aiNet + MI	(0.52,0.33,0.76,0.51)

As we can see, for the nonzero mean simulation the method relying on mutual information achieved a better performance. This result, moreover, was achieved without knowledge of the source pdfs.

4.5.4 Simulation with Real Data: Show-Through Image

For the simulation with real data, we used images subject to the so-called show-through effect [39]. In these images, a thin paper is digitally scanned on both sides, and, for each scan, there is an interference originated from the image on the other side. One of the images was mirrored to make it such that the two scans overlap the same images.

If we define the original images on each side as the sources, then the scans are the mixtures and they can be modeled by an LQ model [39]. The reason why this mixture is not linear is because lighter pixels can be thought of as being “more vulnerable” to interference from the reverse image than darker ones. This nonlinear interference can be modeled by a cross-product of both pixels’ brightness [39], which results in a LQ modeling of the problem.

In our first simulation, we used images of a handwritten text on each side, as can be seen on Fig. 4.7. The images can be found in [40].

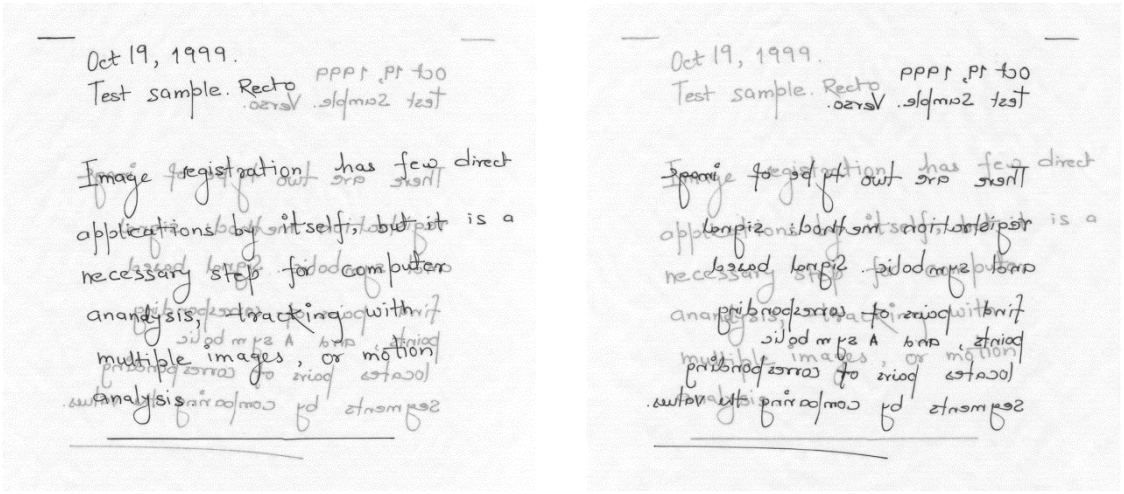

Figure 4.7 - Scanned images for handwritten text for show-through simulation

In order to solve the BSS problem in this case, we initially transformed each image into a vector of pixels on the interval [0,1]. Because the source pdfs are unknown, we used the opt-aiNet algorithm with cost function based on MI.

After separating the sources and converting them to images again, we obtain Fig. 4.8.

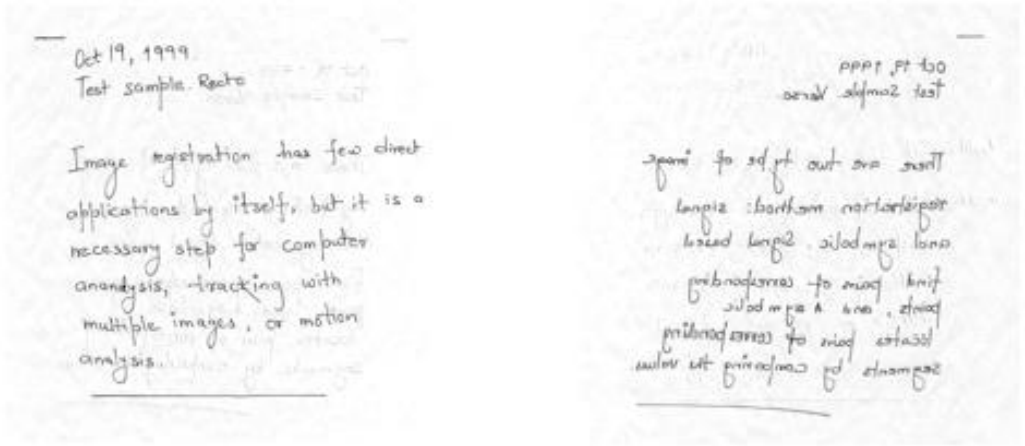

Figure 4.8 - Separated sources for show-trough simulation

The images were 200×200 pixels, resulting in a vector with 40000 points. When observing the images, we can still see some traces of the image on the back, but it is fair to consider that the algorithm successfully separated the images. After appropriately processing the estimates to obtain the same mean and amplitude that the original sources, we can obtain an RMS of 0.055.

The total elapsed time for the separation process was 133s, much longer than the times measured for the simulations with artificial data previously realized. The longer time can be explained by the fact that the images were vectors with 40000 points, instead of the usual data with 1000 points on the simulation.

In our second simulation, we used scanned images of real photographs, which can be seen at Fig. 4.9. These images were obtained at [39]. Unlike the scans of the handwritten text, which were mostly sparse, these images show a dense distribution of points and might end up being more difficult to separate. The image size was 440×321 pixels.

Figure 4.9 - Scanned images for show-through simulation

After running the separation algorithm we obtain the estimates shown at Fig. 4.10:

Figure 4.10 - Separated images from show-through simulation

Similarly to the result obtained on the last simulation, the separation can be said to have successfully separated the images. The original images can be seen in Fig. 4.11.

Figure 4.11 - Original images used on the show-through simulation

When comparing the original images to the ones obtained by the separation algorithm, we can see that the images have become significantly darker as a whole, but other than that, they seem to be fairly well separated. It can be assumed that an overall change in the image brightness might be represented by a scalar multiplier to the whole image. An example of the first image, after having its brightness lowered by multiplying by a constant factor (0.7, in this case) can be seen in Fig. 4.12.

Figure 4.12 - Image with brightness lowered, multiplying by 70%

We can see that Fig. 4.12, now with its brightness lowered, is very similar to the one obtained by the algorithm. As explained in chapter 2, two signals that only differ by a scalar factor can essentially be considered as equivalent, which shows that the algorithm is able to successfully separate mixtures generated by the show-through problem. After appropriately scaling and offsetting the mean to comply with the original images, we obtain an RMS of 0.192

The total elapsed time for this separation process was 758s – which is not suitable for real-time applications, but might be useful for offline applications where time is not an essential constraint.

CHAPTER 5

5 CONCLUSION

In this dissertation, we started by introducing the subject of blind source separation and explaining the basic algorithms that can be used to solve the problem. We initially studied source separation problems with linear mixing models, but eventually moved to nonlinear models – and particularly, the linear quadratic model.

We first analyzed the structure of the recurrent networks used to separate the sources under the LQ model, and tried to improve upon it by studying the problem under a new interpretation of the process. Instead of trying to create a recurrent network based on the inverse of the function, we approached the problem from a new perspective and tried to solve the problem directly, without having to find the inverse of the function.

Under this new strategy, we are able to use standard root-finding algorithms to develop viable recurrent networks that are able to solve the source separation problem. Using the Newton-Raphson method as the basis for our algorithm, we proposed a new structure for solving the problem. After comparing the proposed algorithm with the previously existing ones, we were able to assert that the Newton-based algorithm is more accurate and stable. It can also be generalized to solve the separation problem for any amount of sources, or even different mixing models.

The previously existing algorithm, proposed by Deville and Hosseini [13], was specifically designed for the LQ model, and could not easily be extended to accommodate for more than two sources; as a result, the Newton-based algorithm can be extremely useful. Due to its more computationally expensive iterations, this second-order algorithm is slower than the DH network, but, for applications where the elapsed time is not essential, its advantages can be of paramount importance.

Another aspect that we analyzed in this dissertation was the algorithm to estimate the mixing parameters of the model, in order to be able to solve the blind aspect of the problem. In the original algorithm proposed by Deville and Hosseini [14], a cost function was generated based on the ML principle, and this function is then optimized through gradient-descent method.

One possible improvement of the method would be to optimize the cost function not through a gradient-descent method, but using an evolutionary algorithm. The idea is that, while bio-inspired algorithms are computationally more expensive, they are able to successfully optimize multimodal functions, unlike the gradient-based networks that can become trapped in local minima. Additionally, evolutionary algorithms are able to optimize a function without resorting to its derivative, which can be useful when optimizing cost functions that are not continuous, or whose derivatives are not easy to be estimated.

In case of the ML-based cost function, the main strategy is to find the parameters that best fit the sources pdfs, assuming they are known *a priori*. For many applications, it could be unrealistic to suppose that the sources pdfs are known, so, in order to avoid that requirement, we were able to establish a different cost function, based on the mutual information of the sources. Using this MI-based strategy, we were able to successfully separate sources without *a priori* knowledge of the pdfs.

It is also important to mention that the structure of the ML-based cost function is strongly related to the mixing model and the number of sources – so, while the same strategy might be used for different mixing models and amount of sources, a completely different cost function would have to be developed for each case. Conversely, for the MI based cost function, only the interaction between the source estimates is taken into account, so the same cost function can be used for any mixing model and number of sources.

One real-world application where our proposed algorithm was tested was the problem of removing show-through images that appear on scanned images, and except for the significantly longer time required for the network to converge, the results were very satisfactory. The longer elapsed time was expected, since all of the aforementioned improvements increased the computational cost of the algorithm, in exchange for better stability, accuracy and reducing the requirements on the information to solve the problem. For offline applications where a real-time response is not necessary, the improvements are certainly promising.

Finally, as long as the minimum separability prerequisites of the mixture are satisfied, all of the proposed improvements – the reinterpretation of the BSS structure, the use of bio-inspired algorithms and the use of an optimization strategy based on mutual information – can be easily generalized to any kind of mixing model and for any amount of sources and mixtures. Together with the fact that the techniques presented can increase the robustness of the method, and require

less prior information on the sources, the techniques and algorithms proposed in this dissertation can be an invaluable contribution to the study of nonlinear BSS problems.

REFERENCES

6 REFERENCES

[1] Duarte, L. T., Suyama, R., Attux, R., Deville, Y., Romano, J. M., & Jutten, C. (2010). Blind source separation of overdetermined linear-quadratic mixtures. In Latent Variable Analysis and Signal Separation (pp. 263-270). Springer Berlin Heidelberg.

[2] Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*. IOP Publishing Ltd.

[3] De Castro, L. N., & Timmis, J. (2002, May). An artificial immune network for multimodal function optimization. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (Vol. 1, pp. 699-704). IEEE.

[4] De Castro, L. N., & Von Zuben, F. J. (2001). aiNet: an artificial immune network for data analysis. *Data mining: a heuristic approach*, *1*, 231-259.

[5] Bofill, P., & Zibulevsky, M. (2001). Underdetermined blind source separation using sparse representations. *Signal processing*, *81*(11), 2353-2362.

[6] Comon, P. (1989). Séparation de mélanges de signaux. In 12° Colloque sur le traitement du signal et des images, FRA, 1989. GRETSI, Groupe d'Etudes du Traitement du Signal et des Images.

[7] Comon, P. (1992). Independent component analysis. Higher-Order Statistics, 29-38.

[8] Hosseini, S., & Deville, Y. (2004). Blind maximum likelihood separation of a linear-quadratic mixture. In *Independent Component Analysis and Blind Signal Separation* (pp. 694-701). Springer Berlin Heidelberg.

[9] Comon, P., & Jutten, C. (2010). Handbook of Blind Source Separation: Independent component analysis and applications. Academic press.

[10] Jutten, C., & Karhunen, J. (2003, April). Advances in nonlinear blind source separation. In *Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)* (pp. 245-256).

[11] Taleb, A., & Jutten, C. (1999). Source separation in post-nonlinear mixtures. Signal Processing, IEEE Transactions on, 47(10), 2807-2820.

[12] Hosseini, S., & Deville, Y. (2003). Blind separation of linear-quadratic mixtures of real sources using a recurrent structure. In Artificial Neural Nets Problem Solving Methods (pp. 241-248). Springer Berlin Heidelberg.

[13] Deville, Y., & Hosseini, S. (2009). Recurrent networks for separating extractable-target nonlinear mixtures. Part I: Non-blind configurations. Signal Processing, 89(4), 378-393.

[14] Hosseini, S., & Deville, Y. (2012). Recurrent networks for separating extractable-target nonlinear mixtures. Part II: Blind configurations. Signal Processing.

[15] Ando, R. A., Duarte, L. T., Soriano, D. C., Attux, R., Suyama, R., Deville, Y., & Jutten, C. (2012) Recurrent Source Separation Structures as Iterative Methods for Solving Nonlinear Equation Systems. XXX Simpósio Brasileiro de Telecomunicações.

[16] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Chapter 9. Root Finding and Nonlinear Sets of Equations Importance Sampling". Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press.

[17] Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, *33*(3), 1065-1076.

[18] De Castro, L. N., & Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. Evolutionary Computation, IEEE Transactions on, 6(3), 239-251.

[19] Merrikh-Bayat, F., Babaie-Zadeh, M., & Jutten, C. (2011). Linear-quadratic blind source separating structure for removing show-through in scanned documents.*International Journal on Document Analysis and Recognition (IJDAR)*, *14*(4), 319-333.

[20] Suyama, R. (2007). Proposta de métodos de separação cega de fontes para misturas convolutivas e não-lineares.

[21] Duarte, L. T. (2006). Um Estudo sobre Separação Cega de Fontes e Contribuições ao Caso de Misturas Não-lineares.

[22] Hyvärinen, A., Karhunen, J. & Oja, E. (2001). Independent Component Analysis. Wiley.

[23] S. Achard and C. Jutten. Identifiability of post-nonlinear mixtures. IEEE Signal Processing Letters, 12(5):423–426, May 2005

[24] Schatzman, M. (2002). Numerical Analysis: a Mathematical Introduction. Clarendon Press, Oxford.

[25] Fiedler-Ferrara, N., Prado, C.P.C.: Caos Uma Introdução. Edgard Blücher (1994)

[26] Jutten, C., Hérault, J. (1991). Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, *24*(1), 1-10.

[27] Billingsley, Patrick (1995), *Probability and Measure* (3rd ed.), John Wiley & sons

[28] Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, *13*(4), 411-430.

[29] Hazewinkel, Michiel, ed. (2001), "Differential entropy", Encyclopedia of Mathematics, Springer.

[30] Cover, T. M., & Thomas, J. A. (2012). *Elements of information theory*. Wiley-interscience.

[31] Cardoso, J. F. (1997). Infomax and maximum likelihood for blind source separation. *Signal Processing Letters, IEEE*, *4*(4), 112-114.

[32] Watanabe, S. (1960). Information theoretical analysis of multivariate correlation.*IBM Journal of research and development*, *4*(1), 66-82.

[33] Scott, D. W. (2009). *Multivariate density estimation: theory, practice, and visualization* (Vol. 383). Wiley. com.

[34] Fiedler, B., & Gedeon, T. (1998). A class of convergent neural network dynamics. *Physica D: Nonlinear Phenomena*, *111*(1), 288-294.

[35] Monteiro, L. H. A. (2006). *Sistemas dinâmicos*. Editora Livraria da Física.

[36] Feigenbaum, M. J. "The Metric Universal Properties of Period Doubling Bifurcations and the Spectrum for a Route to Turbulence." *Ann. New York. Acad. Sci.* 357, 330-336, 1980.

[37] Darbellay, G. A., & Vajda, I. (1999). Estimation of the information by an adaptive partitioning of the observation space. *Information Theory, IEEE Transactions on*, *45*(4), 1315-1321.

[38] Nadarajah, S. (2005). A generalized normal distribution. *Journal of Applied Statistics*, *32*(7), 685-694.

[39] http://www.site.uottawa.ca/~edubois/documents/

[40] Almeida, L. B. (2005). Separating a real-life nonlinear image mixture. *arXiv preprint cs/0505044*.

[41] (overdetermined)

[42] Hérault, J., Jutten, C., & Ans, B. (1985). Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervi-

sé. In *10° Colloque sur le traitement du signal et des images, FRA, 1985*. GRETSI, Groupe d'Etudes du Traitement du Signal et des Images.