This is to certify that the

dissertation entitled

BLIND SEPARATION OF UNKNOWN SOURCES IN DYNAMIC ENVIRONMENTS: THEORETICAL FORMULATION AND MICRO-ELECTRONIC IMPLEMENTATION

presented by

Ammar B. Gharbi

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Electrical Engineering

Major professor

Date Dec 19, 1996

Blind Separation of Unknown Sources in Dynamic Environments: Theoretical Formulation and Micro-Electronic Implementation

By

Ammar B. Gharbi

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

1996

ABSTRACT

Blind Separation of Unknown Sources in Dynamic Environments: Theoretical Formulation and Micro-Electronic Implementation

By

Ammar B. Gharbi

Novel learning algorithms for the problem of blind separation of signals in static and dynamic environments are developed. The derived learning rules of these algorithms are based on the minimization of mutual information functionals. A novel learning rule is initially derived from the decorrelation of the output signals. This rule is then enhanced by including higher order terms to test for the independence of signals.

Optimization theory is utilized to derive a general framework to develop an update rule for the parameters of a linear dynamical network model. Higher order statistics are also explored to develop an approximate expression of the mutual information which depends on the *unknown* probability density functions of the output signals.

The modeling of the environment is considered as an important factor in the development of the update law. Keeping the analog implementation of the algorithms in mind, state space realizations of the network which minimize the number of pa-

rameters are considered. Such choice of realization would result in a reduction of the complexity of the network and also the corresponding circuit blocks.

Computer simulation are conducted to evaluate the performance of the developed algorithm. A circuit implementation of one of the developed algorithms for the dynamic case is described and its performance is verified using the PSPICE circuit simulator.

In summary, the main contributions of this thesis are the development of:

1. a novel update law for the static environment case based on the decorrelation of the output signals and its invariants;

2. a novel energy function that characterizes the statistical independence of signals using higher order statistics for both the feedforward and the feedback structures;

3. a novel framework to derive the update laws for the parameters of a dynamic network using optimization theory and the calculus of variations; and

4. a circuit realization of one of the developed algorithms for a dynamic feedback network.

To my wife Nancy, my parents and my family

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to professor Fathi Salam for his guidance, support, and inspiration throughout the course of this research. His valuable, kind consideration, and understanding have been an incentive for the completion of this dissertation.

Sincere gratitude is extended to the members of my guidance committee: Professor Hassan Khalil, Professor Don Reinhart and professor Charles Maccluer for their valuable assistance and service on my committee.

I would like to express my sincere thanks and gratitude to my wife Nancy for her patience, love, affection, support, encouragement and sacrifice during every step that I made towards the completion of this research.

Last, but not least, I would like to express my sincere thanks and appreciation to my parents, and every member of my family for their patience, support and sacrifice awaiting for this day.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The Blind Separation of Sources is a challenging signal processing problem with significant potential applications. The problem is informally described as follows: several unknown but independent temporal signals propagate through a mixing and/or filtering, natural or synthetic environment. By observing the outputs of this environment, a network (e.g., a system, a neural network, or a device) is configured to counteract the effect of the environment and adaptively recovers the original signals. For this processing, only the property of signal independence is assumed. No additional a priori knowledge of the original signals is required. This processing represents a form of self (or unsupervised) learning. The weak assumptions and the self-learning capability render such a network attractive from the viewpoint of real-world applications where training is not an option.

The blind separation approach has great advantages over the existing adaptive filtering algorithms. For example, when the mixture of other signals is labeled as noise in this approach, no specific a priori knowledge about any of the signals is assumed; only that the processed signals are independent. This is in contrast with the noise cancelation method proposed by Widrow *et al.* [1] which requires that a reference signal be correlated exclusively to the part of the waveform (i.e., noise) that needs to be filtered out. This latter requirement entails specific a priori knowledge

about the noise, as well as the signal(s).

The blind separation of sources is valuable in numerous and major applications in areas as diverse as telecommunication systems, sonar and radar systems, audio and acoustics, image/information processing, and biomedical engineering. Consider, e.g., the audio and sonar application where the original signals are sounds, and the mixed signals are the output of several microphones or sensors placed at different vantage points. A network would receive, via each microphone, a mixture of sounds that are usually delayed relative to one another and/or the original sounds. The network's role is then to dynamically reproduce the original signals, where each separated signal can be subsequently channeled for further processing or transmission. Similar application scenarios can be described in situations involving heart-rate measurements, communication in noisy environments, engine diagnostics, and uncorrupted cellular phone communications.

Adequate models of the environment that include time delays or filtering are very important when the sensors are viewed as part of the environment. In many applications where the sensors (e.g., microphones) include their own dynamics, the model of environment should include time-delays or filtering. Otherwise, assuming pure static models of the environment would result in highly sensitive and unrobust processing for any network. Indeed, in order to render the network operable in real-world applications, robust operations must be ensured to parameter variations, dynamic influences and signal delays that often result in asynchronous signal propagation. Examples of such effects include filters in the case of an audio system, or delay lines and/or echoes as in the case of a radar system. The environment should be modeled as a dynamic linear (or even, nonlinear) system [2, 3].

The challenges for this area reside in the development of a mathematical analysis and framework to the problem of blind separation of sources. This is the angle from which this research has been approached: a review of the current literature has been

conducted to study the various approaches to this problem, and a general framework to develop an update law for the network parameters based on optimization theory, the calculus of variations and higher order statistics has been proposed.

The thesis is divided into eight chapters. An introduction of this work is presented in Chapter 1. Also included is the definition of the problem and some applications which are given to illustrate the need for a solution to this problem. Chapter 2 presents an overview of the literature. The pioneering work of Herault and Jutten is presented first followed by the work of Cardoso, Amari and Sejnowski and their coworkers. This review presents the different approaches that have been attempted in finding a solution to this problem, including neuro-biologically based findings, algebraic method solutions and higher order statistics methods. This chapter summarizes a snapshot of the status of current approaches in the field. In Chapter 3, a solution to the problem in a static environment is developed based on the decorrelation condition of the output vector. However, decorrelation characterizes only second order statistics, which are not sufficient for solving this problem. Therefore, improved versions of the algorithm, built upon the works of Herault, Jutten and Amari, are developed, and computer simulations are given which validate the performance of these proposed algorithms. In Chapter 4, higher order statistics are introduced in order to develop an approximation of an energy function. The developed energy function approximation does not assume that the output signals should have unit variance. The developed energy function also uses a higher order approximation of the natural logarithmic function. Such an energy function will require more computations. However, it is shown that the adaptive laws based on such an energy function will perform well, where other existing algorithms fail to perform the same task. In Chapter 5, the environment and network models are represented by a dynamical system described by a matrix transfer function. The existence of a theoretical solution to the problem is presented. Then, a state space realization of the network is developed. Such a

realization is represented by the least number of parameters. This translates into efficient computations and would eventually results in reduced chip area in electronic implementation. The optimization theory and the calculus of variations are explored in this chapter to establish a general framework for updating the parameters of the dynamic network. Computer simulations show the performance and the limitations of the developed algorithms. In Chapter 6, a feedback structure of the dynamic network is considered. The derived update laws here are based on an extension of the static case. Computer simulations for the feedback structure are presented and discussed. In Chapter 7, the basic building blocks for the micro-electronic circuit implementation of the algorithm are developed. Consequently, the complete realization of a feedback neural network architecture with learning is presented and supported by PSPICE computer simulations. Finally, concluding remarks and directions for future work are presented in Chapter 8.

CHAPTER 2

Literature Review

In this chapter, a review of the present literature is presented in order to reveal the different approaches already considered in solving the problem of the blind separation of signals. These efforts have helped direct the path that the research has taken in tackling this problem. Some of these ideas herein will be explored as possible solutions and will be incorporated in the development of this work. A general overview is due, however, in order to give credit to some of the scientists who have contributed to the evolution of this exciting area of intelligent nonlinear signal processing. Yet, on the outset, we confess that our overview would not be comprehensive, and very likely, would overlook some portion of this revolutionary literature.

The blind separation of signals approach, motivated from neuro-biology, was first introduced by Herault and Jutten in the late 1980's. They developed an adaptive algorithm based on neuro-biological findings [4, 5, 6, 7]. Some theoretical analysis was later developed which provided some validation of the algorithm [8, 9, 10, 11]. This line of work, based on the neuromemitic approach, was further considered by Karhunen [12] and Chichocki and Moszcznski [13].

The studies on the blind separation of signals are based on the assumption that no a priori detailed knowledge of the unknown sources is available. However, the unknown sources are assumed to be (statistically) independent. Thus, to solve the

problem one has to render the components of the output of the network statistically independent. This hypothesis will be the basis for the development of various energy functions, or sometimes called *contrast* functions or independence criteria.

Despite the fact that decorrelation of the components of a signal vector is a weaker condition than independence, some researchers have considered such a necessary, but not sufficient, conditions to develop criteria and corresponding algorithms [14]. However, decorrelation describes only 2^{nd} order statistics of the output signal vector. Therefore, these criteria are valid only for a small set of input signals and higher order statistics need to be considered when developing more capable algorithms.

Since the cross cumulants of independent signals are zero, several contrast functions have been considered in solving the problem. For example, when the independence is measured in terms of the cancelation of fourth order cumulants of the outputs, cubic nonlinearity, similar to that defined in [4], will appear in the update of the algorithm [15, 16, 17]. Cardoso, on the other hand, focused on the algebraic properties of fourth order cumulants and considered the problem as a series of whitening and diagonalization processes [18, 19, 20, 21, 22].

These criteria are necessary, but not sufficient, as most of these conditions are constrained to some specified set of inputs. Therefore, the celebrated work of Comon [23] is considered an important framework. The mutual information is considered as an energy function. The mutual information is expressed in terms of the marginal probability density functions. However, knowledge of the densities is not accessible by the hypothesis of the problem. Therefore, an Edgeworth expansion was considered to approximate them. Amari et al [24] followed the same line of work by taking a Charlier-Gram expansion. Bell and Sejnowski [25, 26] had also considered the optimization of only one term of the mutual information, but had to resort to express it in terms of a nonlinear function of the output signals in order to capture the statistical information of the output.

It should be noted that an important technical difficulty faces the identifiability of the solution to the problem of the blind separation of signals. Due to lack of information, regarding such items as the signal power, the spectral content or the modulation scheme, the output of the separating network cannot be ordered corresponding to the order of sources signals. Thus, the signals can be identified up to an indetermination in terms of scale and order. This identifiabity problem was first treated by Giannakis et al [27] who used third order cumulants, and, it was further addressed by Tong et al [28, 29, 30, 31] who used fourth order cumulants.

Several algorithms have been proposed in the literature for separating signals based on the availability of prior spatial, temporal or statistical information. This direction of work defines another approach to blind separation because necessary and sufficient conditions of statistical independence are hard to satisfy. For example, the MUSIC [32] and ESPRIT [33] algorithms take advantage of the structural information of the channels based on the Vandermode matrix channel characterization to obtain an estimate of the parameters. Unfortunately, such structural information is not always available. Other algorithms that exploit the temporal structure of a communication channel, while assuming no priori spatial knowledge, have been proposed in the literature. These techniques consider the constant modulus property [34], discrete alphabet [35], self-coherence [36] and the finite alphabet property [37]. Other algorithms were developed based on a priori knowledge of the statistical information of the signals. When the sources have known probability densities, the maximum likelihood estimator is used to provide a solution to the problem [38, 39, 40].

So far, this review has focused on the case when a static modeling of the environment is considered. How about the case when the environment is modeled as a dynamic system? In this case, two lines of work can be described: digital versus continuous. Most of the studies in the literature have tackled this problem using FIR (finite impulse response) filters. For example, Moulines et al [41] considered the sub-

space method to decompose the signal-noise space from the noise space to recover the unknown sources. Gerven and Compernolle [42] used a criteria based on the second order statistics while Thi and Jutten [43] considered criteria based on the cancelation of fourth order cumulants. In these works, the goal was to determine the FIR coefficients that separate the signals. Bell and Sejnowski considered an information-theoretic approach [44, 45]. In all these works, FIR's were considered to model both the environment and the network. This thesis intends to address the problem by considering an environment and a network model that are described by a continuous linear dynamical system [46, 47], and to define a general framework for solving the problem.

Several analog implementations of the blind separation algorithms have been reported in the literature. Vittoz and Arreguit [48] and Cohen and Andreou [49] have considered the implementation of the static HJ algorithm. On the other hand, Gharbi and Salam have considered an analog implementation of the dynamic HJ algorithm [2, 50, 51, 52, 52, 47].

2.1 Neuromemitic Algorithm

Herault and Jutten have pioneered a new paradigm in the area of blind separation of in the Ph.D. work of Jutten under the supervision of his advisor, Herault [4]. The problem was labeled as the Independent Component Analyzer (ICA) [4, 6] because of its similarity to the Principal Component Analysis [53, 54]. In [8, 9, 10, 7, 5], Herault and Jutten proposed an algorithm for the separation of independent sources. It was assumed that the medium is linear and static. The inputs to the network were the measured signals $e_i(t)$, $1 \leq i \leq n$, which are linear combinations of the original signals, namely,

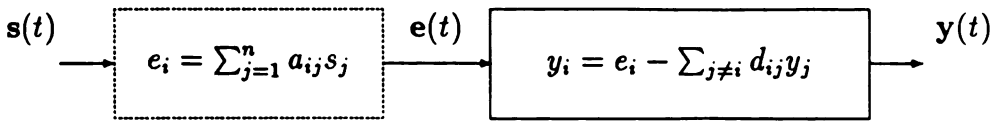


Figure 2.1. Herault and Jutten Architecture

$$e_i(t) = \sum_{i=1}^{n} a_{ij}\ s_j(t) \tag{2.1}$$

which, in vector form, can be expressed as

$$\mathbf{e}(t) = A\ \mathbf{s}(t) \tag{2.2}$$

A is an $n \times n$ matrix whose components a_{ij} are unknown. The matrix A models the mixing static environment and is assumed to be a nonsingular matrix. Furthermore, for normalized mixing, its diagonal entries are all ones, and each off- diagonal element is less than one in absolute value. Herault and Jutten used a recursive architecture made up of fully interconnected outputs. Each output, $y_i(t), 1 \leq i \leq n$, received the mixed signal, $e_i(t)$, and a weighted sum of all other outputs, $-\sum_{j \neq i} d_{ij}\ y_j(t)$. Thus,

$$y_i(t) \ = \ e_i(t) - \sum_{j \neq i} d_{ij}\ y_j(t) \quad 1 \leq i \leq n \tag{2.3}$$

which, in vector form, becomes

$$\mathbf{y}(t) = \mathbf{e}(t) - D\ \mathbf{y}(t) \tag{2.4}$$

D is an $n \times n$ weight matrix whose main diagonal is zero. Now, the problem of separation of signals translates to retrieving the original signals $\mathbf{s}(t)$. In the limit, it is thus desired to have:

$$\mathbf{y}(t) = P\ \mathbf{s}(t) \tag{2.5}$$

where P is a generalized permutation matrix; a matrix that is obtained from a non-singular diagonal matrix by row and/or column permutation.

Herault and Jutten were motivated by some neuro-biological evidence that information about the speed and position of body joints is mixed before being sent to the brain by two different types of nerves. The brain, however, is perfectly capable of separating (and recovering) speed and position signals. This biological and intuitive inspiration led Herault and Jutten to propose an update law, reminiscent of the Hebbian rule, that presumed the independence of the original signals

$$\dot{d}_{ij} = \eta_{ij}\ f(y_i)g(y_j) \qquad\qquad i \neq j \tag{2.6}$$

where $f(.)$ and $g(.)$ are two nonlinear odd functions and η_{ij} is a constant learning rate. Despite the fact that their original idea came from a neuro-biological inspiration, the authors supplied an initial mathematical reasoning for the update law.

2.1.1 Derivation of The H-J Update Law

In [10], the authors have introduced a justification of the developed update law defined by (2.6). The work of the authors and some issues that may not have been considered during the development of their algorithm will be presented.

This algorithm was developed by assuming that the network is near convergence. This means that the first $(n-1)$ outputs were assumed to have converged; $y_i = a_{ii}s_i,\ \forall i < n$. Then, the last output can be expressed as:

$$y_n = e_n - \sum_{j \neq n} d_{nj}\ y_j \tag{2.7}$$

$$= \sum_{j} a_{nj}\ s_j - \sum_{j=1}^{n-1} d_{nj}\ y_j \tag{2.8}$$

$$= \sum_{j=1}^{n-1} (a_{nj} - d_{nj}a_{jj})\ s_j + a_{nn}s_n \tag{2.9}$$

Since the sources s_i's are independent, the cross correlation between two different outputs is null. Therefore, the output power of y_n can be expressed as:

$$< y_n^2 >= \sum_{j=1}^{n-1} (a_{nj} - d_{nj}a_{jj}) \ < s_j^2 > + a_{nn} < s_n^2 > \tag{2.10}$$

Thus in order to minimize the power of y_n, it is required that

$$a_{nj} - d_{nj}a_{jj} = 0, \quad \forall j \neq n \tag{2.11}$$

In this case, the power of the n^{th} output $y_n(t)$ is proportional to that of $s_n(t)$. This above relationship is true only for the n^{th} output assuming that all other $(n-1)$ signals are at the desired solution.

The authors proposed to develop the algorithm based on the principle of minimizing the individual output powers using the gradient descent method which will be presented in the next section. However, minimizing the individual powers of the outputs does not lead, necessarily, to the minimization of the total energy function, which is the sum of all individual powers. They assumed that the only contribution from the total energy to a particular parameter d_{ij} comes only from the corresponding output signal y_i. Later, in this work, a different update law obtained from the sum of all individual output powers will be considered as an energy function for the problem. Due to the fact that the algorithm developed from this energy function is merely a decorrelation of two output signals, it fails to separate sources. However, including the nonlinearity functions f and g in (2.6) have provided some improvements.

2.1.2 Gradient Descent Method

As was discussed earlier, the authors in [10] proposed that the network parameters d_{ij} would be updated using the gradient descent method in order to minimize the

output power of the individual signals defined as

$$E_i = \frac{1}{2}\, y_i^2 \tag{2.12}$$

Thus the parameters were updated as follows:

$$\dot{d}_{ij} = -\eta\, \frac{\partial E_i}{\partial d_{ij}} = -\eta y_i\, \frac{\partial y_i}{\partial d_{ij}} \tag{2.13}$$

It is given that

$$\begin{aligned} y_m &= e_m - \sum_{k \neq m} d_{mk}\, y_k \\ &= e_m - \sum_{k} (1 - \delta_{mk}) d_{mk}\, y_k \end{aligned} \tag{2.14}$$

Now, differentiate both sides of the above equation with respect to d_{ij} in order to obtain the expression for $\partial y_m / \partial d_{ij}$:

$$\frac{\partial y_m}{\partial d_{ij}} = -\sum_k \delta_{im}\, \delta_{jk}\, y_k\, (1 - \delta_{km}) - \sum_k d_{mk}\, \frac{\partial y_k}{\partial d_{ij}}\, (1 - \delta_{km}) \tag{2.15}$$

Rearrange the above equation to obtain:

$$\sum_k (\delta_{mk} + d_{mk}\, (1 - \delta_{mk}))\, \frac{\partial y_k}{\partial d_{ij}} = \delta_{mi}\, (\delta_{mj} - 1) y_j = v_{jm}^{(i)} \tag{2.16}$$

Let

$$Q = (I + D)^{-1} \tag{2.17}$$

and the previous equation becomes

$$(I + D) \frac{\partial y}{\partial c_{ij}} = \mathbf{v}_j^{(i)} \tag{2.18}$$

Therefore,

$$\frac{\partial y_m}{\partial d_{ij}} = \left[\frac{\partial y}{\partial d_{ij}}\right]_m \tag{2.19}$$

$$= \left[Q\ \mathbf{v}_j^{(i)}\right]_m \tag{2.20}$$

$$= \sum_k q_{mk}\ v_{jk}^{(i)} \tag{2.21}$$

$$= \sum_k q_{mk} y_j\ \delta_{ki}\ (\delta_{kj} - 1) \tag{2.22}$$

$$= q_{mi}\ y_j\ (\delta_{ij} - 1) \tag{2.23}$$

Consequently

$$\dot{d}_{ij} = -\eta\ y_i\ \frac{\partial y_i}{\partial d_{ij}} \tag{2.24}$$

$$= -\eta q_{ii}\ (\delta_{ij} - 1)\ y_i\ y_j \tag{2.25}$$

$$= \eta\ q_{ii}\ y_i\ y_j\ (1 - \delta_{ij}) \tag{2.26}$$

2.1.3 Evaluation of the update law

The update law defined in (2.26) always converges to a symmetric solution matrix; thus limiting the structure of the environment to a symmetric matrix. However, the general environment cannot always be modeled by a symmetric matrix. This represents a drawback of the update law defined by (2.26). Also, when equation (2.26) is expressed on average as

$$< \dot{c}_{ij} >= \eta\ q_{ii}\ < y_i\ y_j >\ (\delta_{ij} - 1) \tag{2.27}$$

this rule tests only the decorrelation of the output signal $y(t)$. However, the goal is to develop one update law which pushes for the independence of the components of $y(t)$. To do so, the rule was modified in order to accommodate higher order moments by imposing a nonlinear function which produced various moments of the output, as

follows:

$$\dot{d}_{ij} = \eta \; q_{ii} \; f(y_i) \; g(y_j) \; (1 - \delta_{ij}) \tag{2.28}$$

where f and g are two odd functions. These two functions should be different. Otherwise, the matrix D becomes symmetric. The use of such two functions introduces higher- order moments. To illustrate this, consider the Taylor series expansion of these two odd functions as

$$f(x) = \sum_k \alpha_{2k+1} \; x^{2k+1} \quad \text{and} \quad g(x) = \sum_l \beta_{2l+1} \; x^{2l+1} \tag{2.29}$$

Thus, the equilibria of equation (2.28) on the average satisfy

$$< \dot{d}_{ij} >= \eta q_{ii} \sum_{k,l} \alpha_{2k+1} \beta_{2l+1} \; < y_i^{2k+1} \; y_j^{2l+1} > = 0 \tag{2.30}$$

Assuming α_{2k+1} and β_{2k+1} are not zero, this results in

$$< y_i^{2k+1} \; y_j^{2l+1} >= 0, \; \forall \; k, \; l \tag{2.31}$$

This condition means that all joint odd moments are zero. This condition is stronger than correlation. However, it is implied by independence of the components of the signal vector $\mathbf{y}(t)$, when these signals have even probability density functions.

As a last approximation, the authors used the assumption that the diagonal entries q_{ii}'s were very close to one, since the off-diagonal entries c_{ij}, $i \neq j$, were less than one. Thus, the proposed update law becomes as defined by equation (2.6).

2.1.4 Computer Simulations

Computer simulations were performed for the H-J algorithm defined by (2.6). The unknown sources are two sine waveforms with respective frequencies $1kHz$ and $2kHz$.

The environment mixing matrix is assumed to be

$$A = \begin{bmatrix} 1.0 & 0.6 \\ 0.4 & 1.0 \end{bmatrix}$$

The learning rate is taken to be $\eta = 100$ and the initial conditions are all zero. Figure 2.2 shows the performance of the algorithm when the odd functions f and g are respectively the cubic and the linear functions.

$$f(x) = x^3 \qquad \text{and} \qquad g(x) = x$$

One observes that the off-diagonal coefficients of the D matrix converge to the desired values. In Figure 2.2, the D matrix converges to

$$D_f = \begin{bmatrix} 0.0000 & 0.4004 \\ 0.6004 & 0.0000 \end{bmatrix}$$

In addition, Figure 2.2 displays the performance index, which is defined next.

Performance Index

In the problem of blind separation of signals, one desires to design a network such that its output is a replica of the unknown sources. Because of the lack of knowledge of the unknown sources and the mixing matrix, one does not expect to completely identify the unknown sources. Therefore, the order of the components of the output cannot be determined, nor is their corresponding magnitudes. Consequently, one can identify the unknown sources up to a permutation and a scaling. This is defined as the wave preserving property [28]:

Definition 1 (wave preserving property) *The signals y(t) and s(t) satisfy the wave-preserving property if and only if*

$$\mathbf{y}(t) = I_P \, \Gamma \, \mathbf{s}(t) = P\mathbf{s}(t) \tag{2.32}$$

where I_P is a permutation matrix and Γ is a diagonal matrix, and G is the generalized permutation matrix. The definitions of a permutation matrix and a generalized permutation matrix are given in Appendix A.

In the case of the H-J work, the network input-output relationship is described by equation (2.4). For the sake of generality, let's assume that such a relationship is represented by

$$\mathbf{y} = W\mathbf{e} \tag{2.33}$$

Thus, considering equations (2.32) and (2.33), one obtains

$$P = WA \tag{2.34}$$

So, at convergence, the gain matrix WA, which is the product of the matrices W and A, has to be a generalized matrix in order to claim that the network converged to a solution that separates the mixed signals.

The following mapping

$$\begin{aligned} I: \; R^{n\times n} &\longrightarrow R^{+} \\ P &\longmapsto I(P) = \sum_i \Big[\sum_j \frac{|p_{ij}|}{\max_k |p_{ik}|} - 1\Big] \\ &\qquad + \sum_j \Big[\sum_i \frac{|p_{ij}|}{\max_k |p_{kj}|} - 1\Big] \end{aligned} \tag{2.35}$$

is always positive and is zero if and only if the matrix P is a generalized permutation matrix. Therefore, this mapping will be considered as a performance index of any given algorithm for blind signal separation. Its plot, versus the time of evolution of the algorithm, represents a measure of the convergence of that algorithm.

Thus, one clearly observes that the performance index, shown in Figure 2.2, approaches zero and consequently the gain matrix $P = WA$ becomes

$$P = \begin{bmatrix} 1.0002 & -0.0006 \\ -0.0005 & 1.0004 \end{bmatrix}$$

Figure 2.3 displays the response of the network after convergence. The output signals of the network, $y_i(t)$, are approximately the unknown sources, $s_i(t)$. There is an error between the input and the output of few milli units.

Computer simulations were also performed using different types of odd functions. In this case,

$$f(x) = \sinh x \qquad \text{and} \qquad g(x) = \tanh x$$

Figure 2.4 and 2.5 show the performance of the algorithm defined (2.6) using these odd functions. The settings of this simulation are exactly the same as for the ones above, in terms of initial conditions, learning rate, mixing matrix and unknown sources. When, the training time is equal to $0.1sec$, it was observed that the network did not converge yet. Therefore, a longer training time, namely $0.2sec$, was allocated. In this case, the parameters converged to the near desired values as shown in Figure 2.4. Also, Figure 2.5 shows that the output represents a near replica of the unknown sources

When computer simulations where performed for the algorithm defined by equation (2.26), the network converged to a symmetric matrix as it was anticipated in the

discussion of the algorithm development. Figures 2.6 and 2.7 show the performance.

If one considers deriving the update law based on minimizing the total energy of

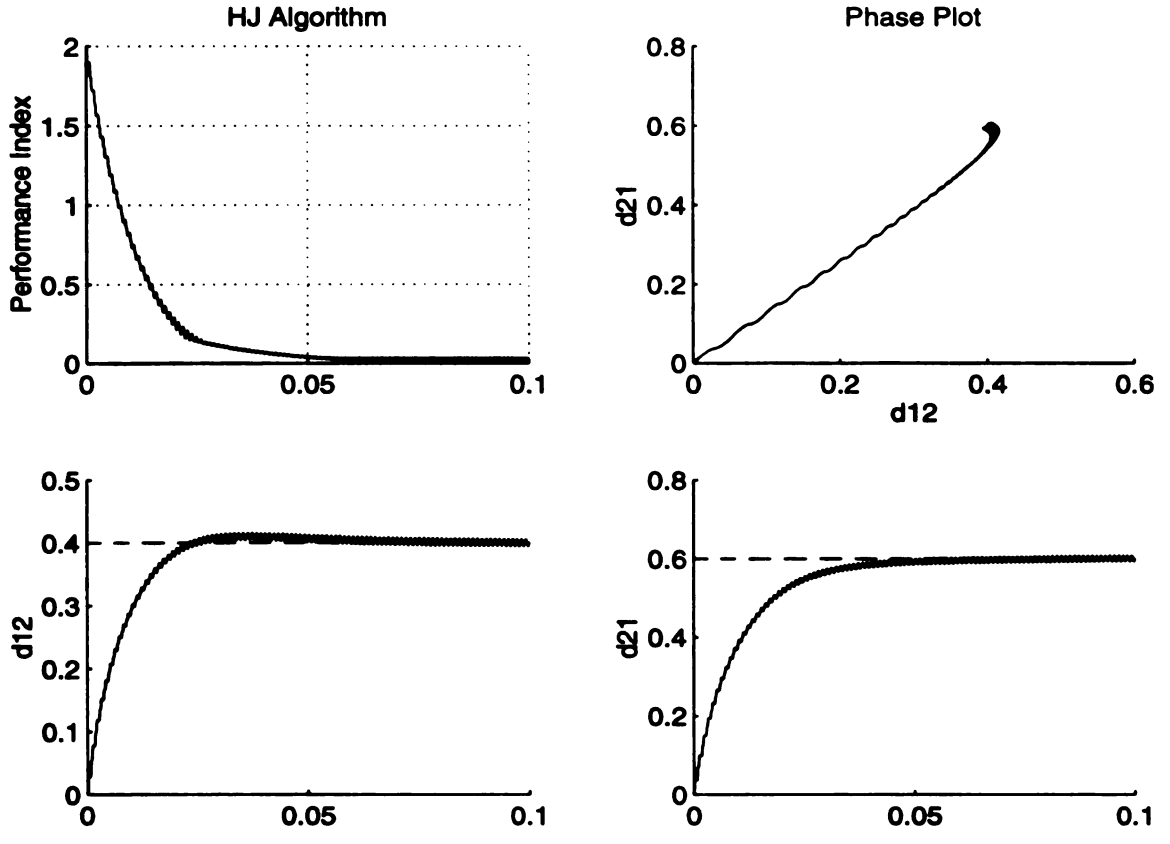


Figure 2.2. Performance of H-J Algorithm. The parameters are updated according to $\dot{d}_{ij} = \eta\ y_i^3\ y_j$

the signal or any other energy function of the form

$$\phi = \frac{1}{2}\sum_m \phi(y_i^2) \tag{2.36}$$

one could use equation (2.23) to obtain

$$\dot{d}_{ij} = -\eta \sum_m \phi'(y_m) y_m q_{mi}\ y_j\ (\delta_{ij} - 1) \tag{2.37}$$

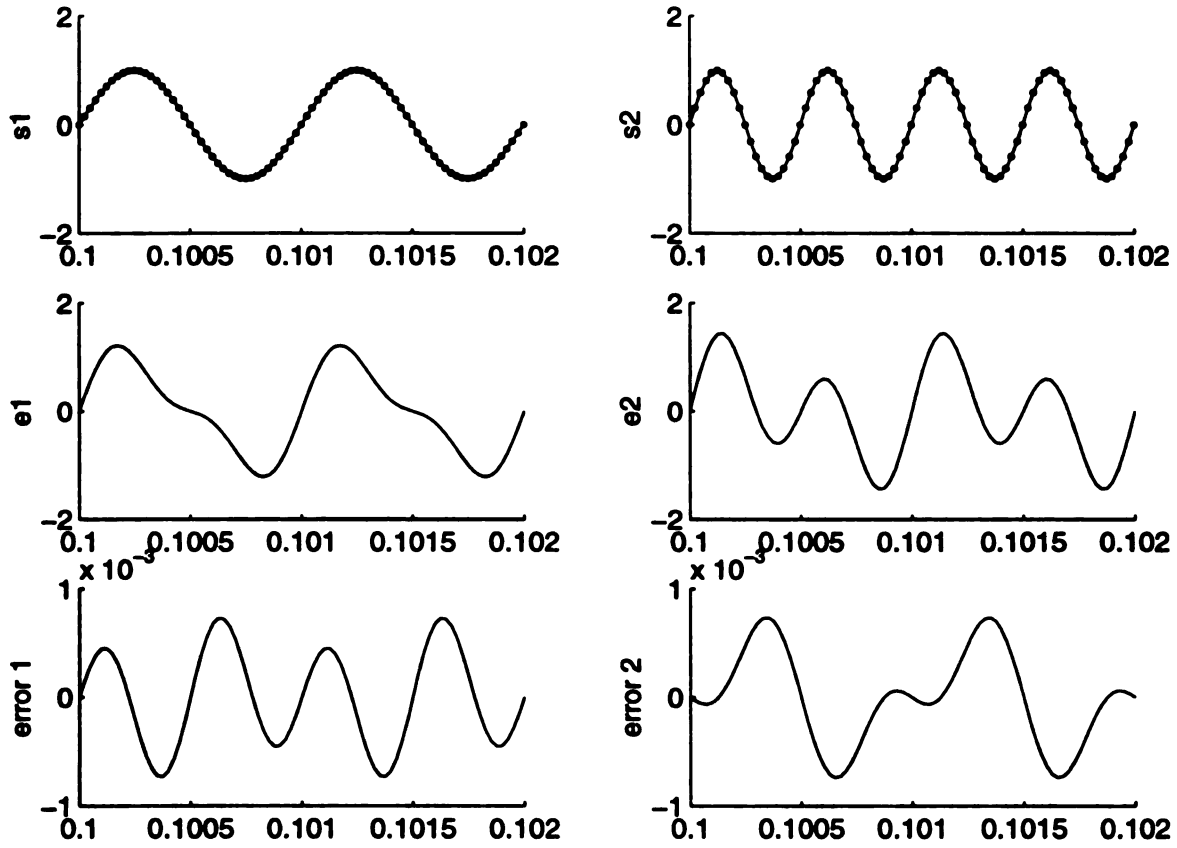


Figure 2.3. Performance of H-J Algorithm. The parameters are updated according to $\dot{d}_{ij} = \eta\ y_i^3\ y_j$. The solid curve is the unknown sources. The dotted curve, superimposed onto the solid curve, is the network output

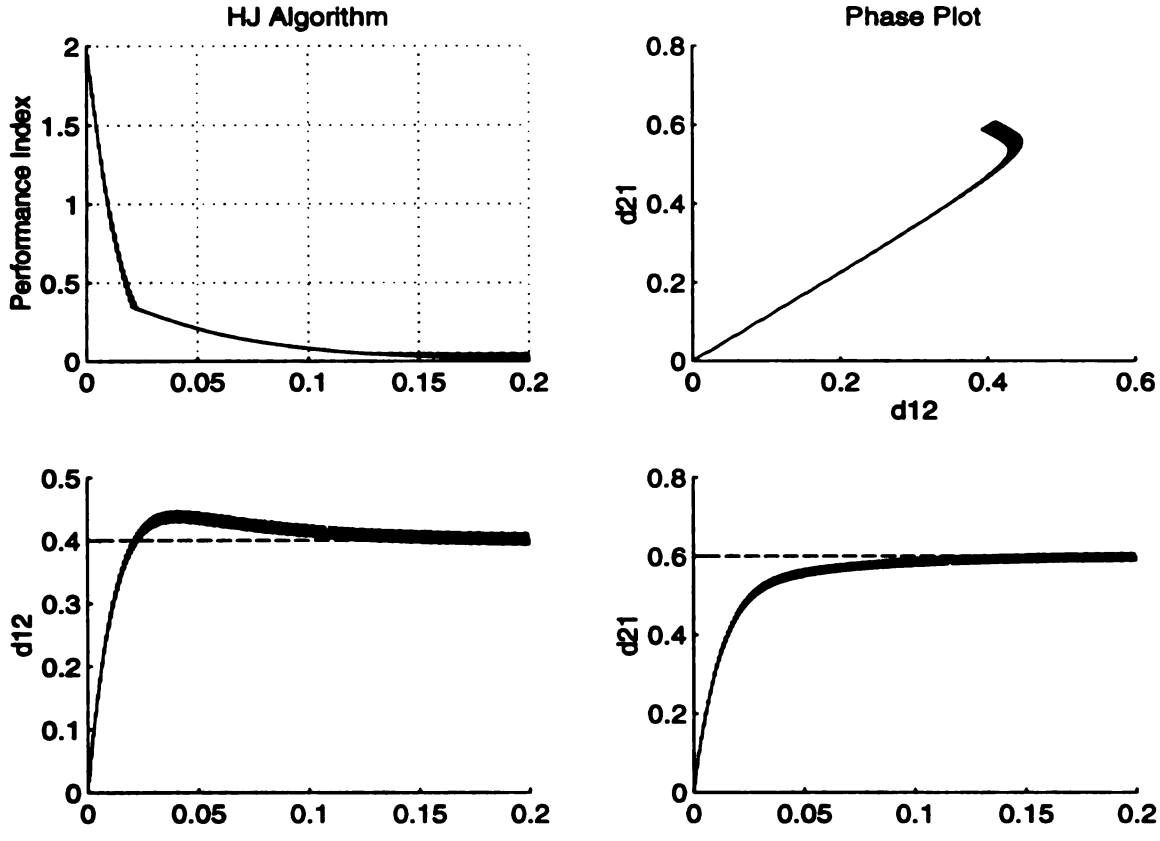


Figure 2.4. Performance of H-J Algorithm. The parameters are updated according to $\dot{d}_{ij} = \eta \sinh y_i \tanh y_j$

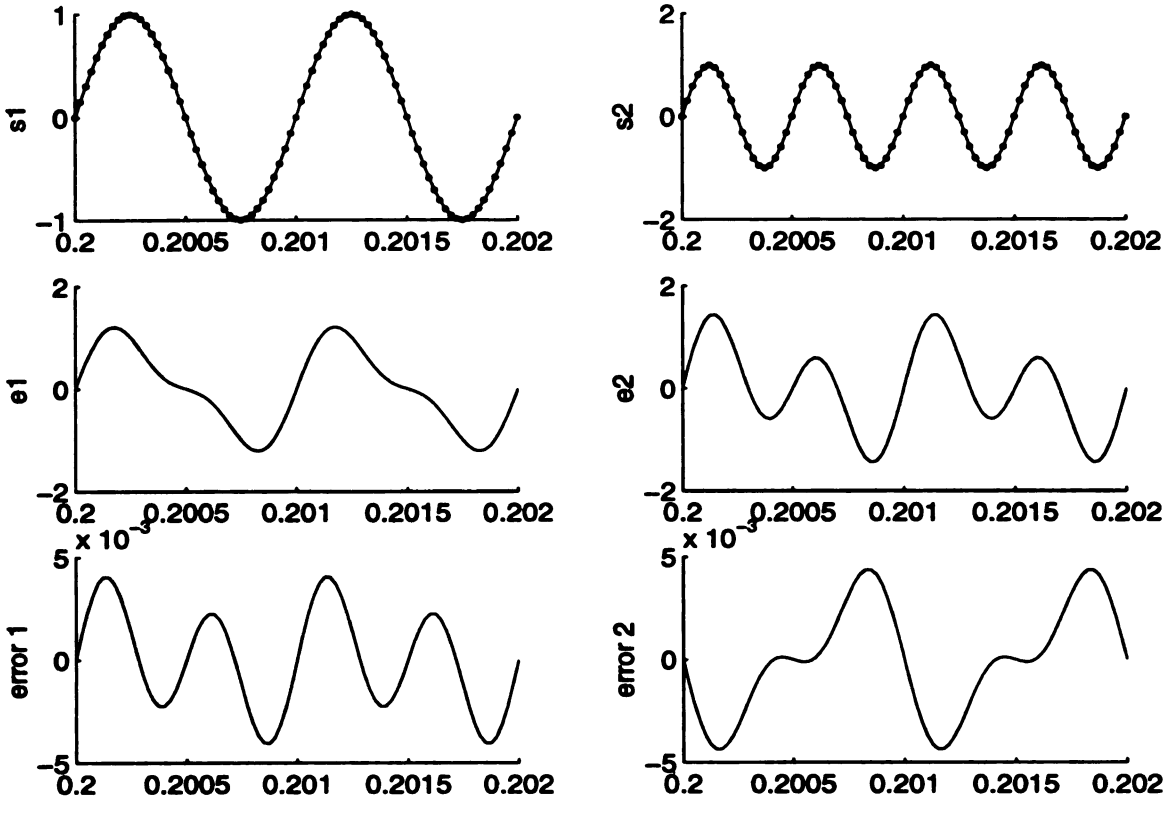


Figure 2.5. Performance of H-J Algorithm. The parameters are updated according to $\dot{d}_{ij} = \eta \sinh y_i \tanh y_j$. The solid curve is the unknown sources. The dotted curve, superimposed onto the solid curve, is the network output

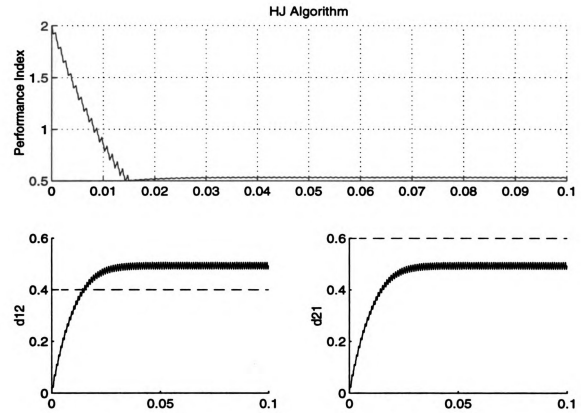


Figure 2.6. Performance of H-J Algorithm. The parameters are updated according to $\dot{d}_{ij} = \eta\ q_{ii}\ y_i\ y_j$

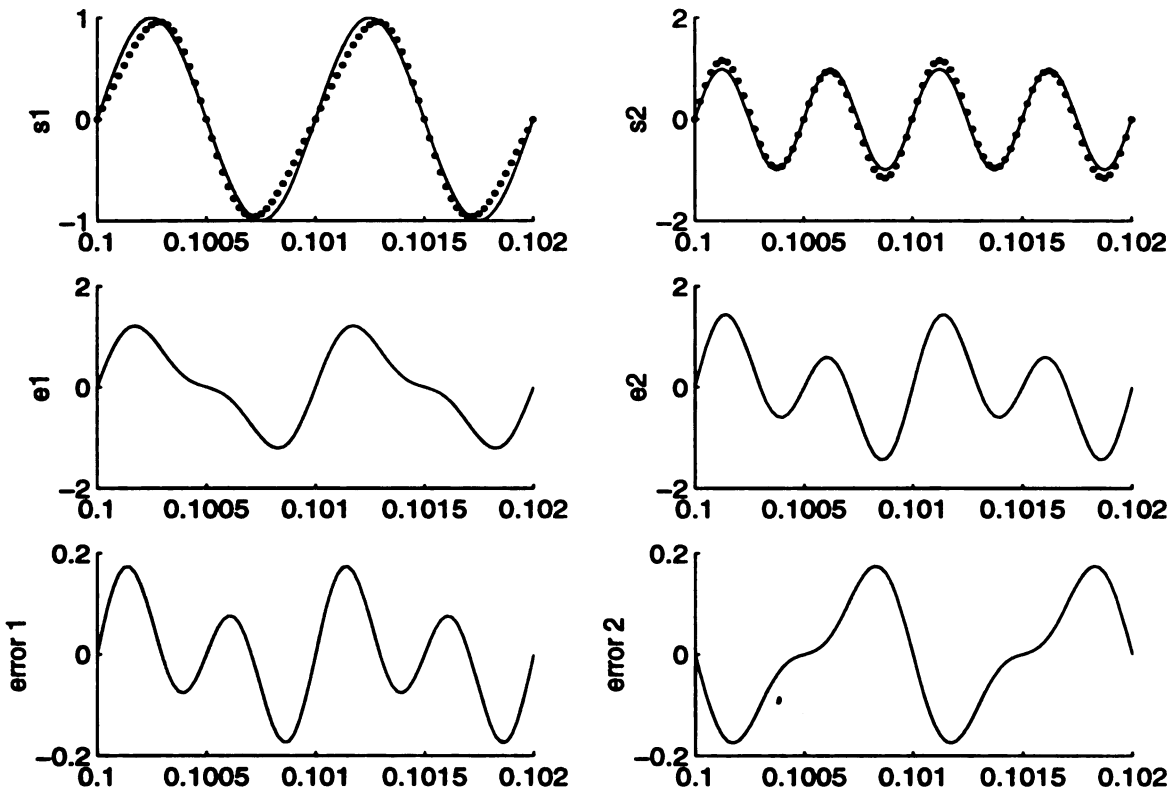


Figure 2.7. Performance of H-J Algorithm. The parameters are updated according to $\dot{d}_{ij} = \eta\ q_{ii}\ y_i\ y_j$. The solid curve is the unknown sources. The dotted curve, superimposed onto the solid curve, is the network output

2.2 Mutual Information Approach

In [24], Amari et al. considered the following feedforward architecture for the blind separation problem. They used the Independent Component Analyzer (ICA) frame-

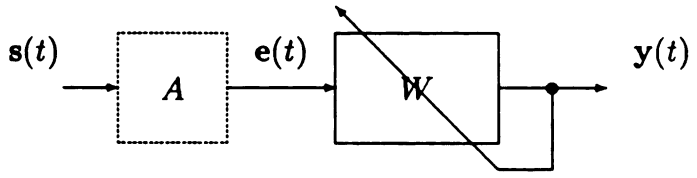


Figure 2.8. Feedforward Architecture

work formulated by Comon [23] which is based on minimizing the dependency among the output components. This dependency is measured by the Kullback-Leilber divergence between the joint and the marginal probability density functions:

$$I(\mathbf{y}) = \int f_{\mathbf{y}}(\mathbf{y}) \ln \frac{f_{\mathbf{y}}(\mathbf{y})}{\prod_i f_{y_i}(y_i)} dy \tag{2.38}$$

Equation (2.38) is minimum and is equal to zero only when all the components of the output vector,namely y_i, are statistically independent. The averaged mutual information can be expressed in terms of entropy as

$$I(\mathbf{y}) = -H(\mathbf{y}) + \sum_i H(y_i) \tag{2.39}$$

where $H(\mathbf{y})$ is the entropy of $\mathbf{y}$ which is a measure of uncertainty of the occurrence of the event produced by $\mathbf{y}$ is defined in Appendix A.

When using the mutual information as an independence criterion, partial knowledge of the statistical information of the output is needed, since the chosen independence criterion is a function of the probability density of the output. To surpass that constraint, Amari [24], as well as Comon [23], used a truncation of infinite series

expansion of the probability density functions. Unlike Comon, who used Edgeworth expansion, Amari completed a Gram-Charlier expansion to approximate the probability distribution of the output. The fourth-order Gram-Charlier expansion of the $f_{y_i}(y_i)$ is

$$f_{y_i}(y_i) \approx \tilde{f}_{y_i}(y_i) = \sigma(y_i)\Big[1 + \frac{\kappa_{3i}}{3!}H_3(y_i) + \frac{\kappa_{4i}}{4!}H_4(y_i)\Big] \tag{2.40}$$

where $\kappa_{3i} = m_{3i}$, $\kappa_{4i} = m_{4i} - 3$, $m_{ki} = E[y_i^k]$ is the k^{th} moment of y_i, $\sigma(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$, and $H_k(x)$ are the Chebyshev-Hermite polynomials defined by

$$(-1)^k \frac{d^k\ \sigma(x)}{dx^k} = H_k(x)\sigma(x) \tag{2.41}$$

It is assumed here that the variance term is unity, i.e.

$$m_{2i} = 1, \qquad \forall i \tag{2.42}$$

A second approximation of the natural logarithmic function was used in order to be able to compute the integrals

$$\ln(1 + x) \approx x - \frac{x^2}{2} \tag{2.43}$$

Assuming that the weighting matrix W was nonsingular, then the following would hold

$$f_{\mathbf{y}}(\mathbf{y}) = \frac{f_{\mathbf{x}}(\mathbf{x})}{|W|} \tag{2.44}$$

Therefore, $H(\mathbf{y})$ can be rewritten as

$$H(\mathbf{y}) = E[\ln|W|] - E[\ln f_{\mathbf{x}}(\mathbf{x})] = H(\mathbf{x}) + E[\ln|W|] \tag{2.45}$$

Thus,

$$I(\mathbf{y}) = \sum_i H(y_i) - E[\ln |W|] - H(\mathbf{x}) \tag{2.46}$$

Using the above approximations and the gradient descent method to minimize the mutual independence, the authors arrived at the following update law [24]:

$$\dot{W} = \eta[W^{-T} - f(\mathbf{y})\mathbf{x}^T] \tag{2.47}$$

where

$$f(y) = \frac{3}{4}y^{11} + \frac{25}{4}y^9 - \frac{14}{3}y^7 - \frac{47}{4}y^5 + \frac{29}{4}y^3 \tag{2.48}$$

This function is plotted in Figure 2.9 along with other functions that have been considered in the literature. However using the information theory perspective [55] and assuming the mixing matrix to be nonsingular, the above update law will be rewritten as

$$\dot{W} = \eta\left[I - f(\mathbf{y})\mathbf{y}^T\right]W \tag{2.49}$$

Computer simulations of this algorithm were performed. The unknown sources are assumed to be two sine waveforms with respective frequencies $1Hz$ and $2Hz$. The mixing matrix is chosen to be random

$$A = \begin{bmatrix} 0.7012 & 0.7622 \\ 0.9103 & 0.2625 \end{bmatrix}$$

The learning rate is $\eta = 0.1$ and the random initial condition

$$W_0 = \begin{bmatrix} 0.0475 & 0.3282 \\ 0.7361 & 0.6326 \end{bmatrix}$$

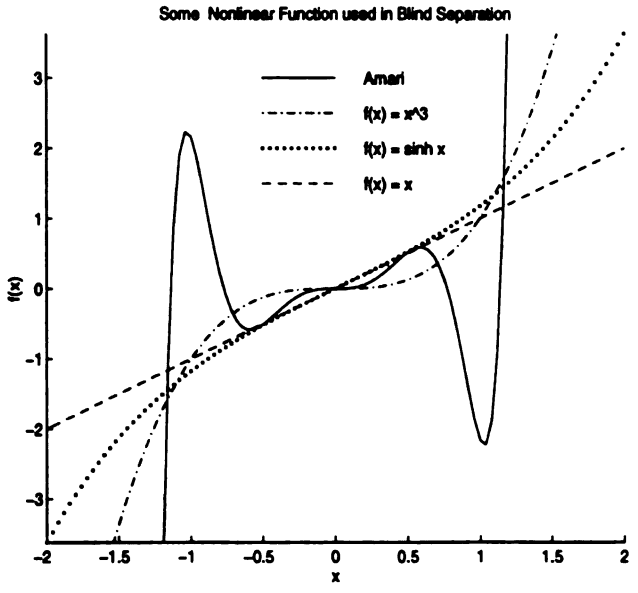


Figure 2.9. Nonlinear functions

According to such a setting, the initial gain matrix is

$$G_0 = W_0 A = \begin{bmatrix} 0.3321 & 0.1223 \\ 1.0920 & 0.7271 \end{bmatrix}$$

It is clear that the gain matrix, G_0, is not a generalized permutation matrix. So, the goal is that the network would update its parameters such that the gain matrix becomes a generalized permutation matrix. Figure 2.10 shows the performance of the algorithm using the settings described above. One can observe that all the parameters of the matrix W converged to

$$W_f = \begin{bmatrix} -0.6835 & 1.9155 \\ 2.1673 & -1.6387 \end{bmatrix}$$

which corresponds to the gain matrix

$$G_f = W_f A = \begin{bmatrix} \boxed{1.2644} & -0.0183 \\ 0.0280 & \boxed{1.2218} \end{bmatrix}$$

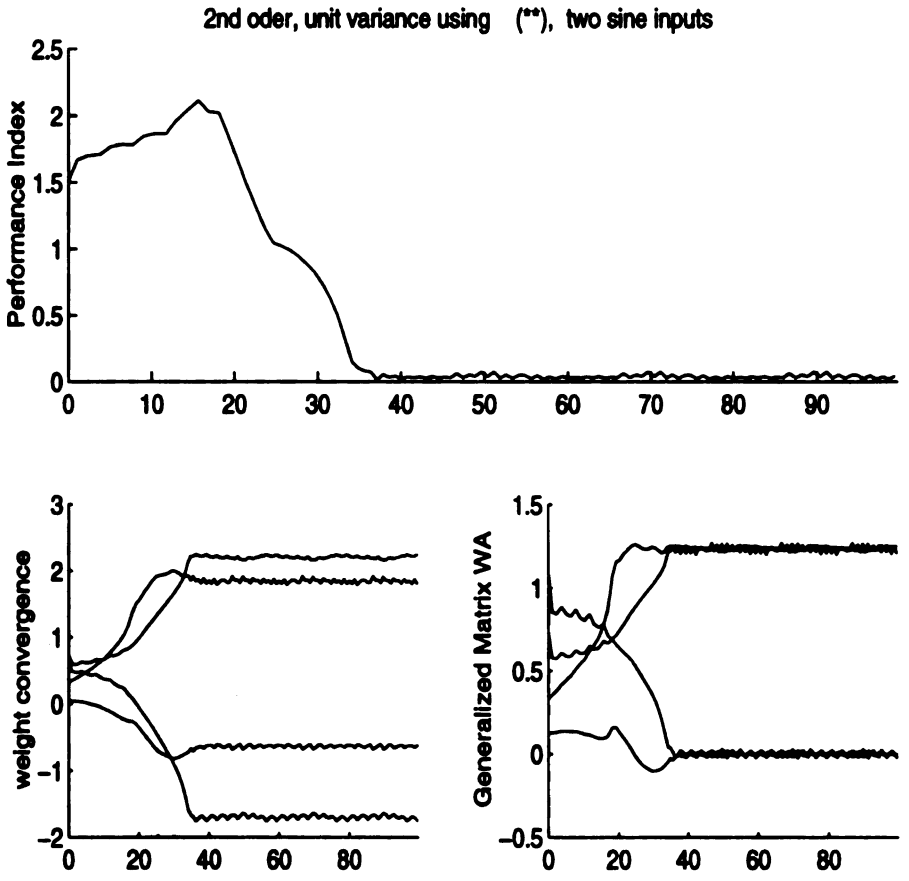


Figure 2.10. Parameter convergence of the algorithm defined by equation (2.49)

It is very important to note that the gain matrix converged to a matrix different from the identity, unlike the algorithm defined by H-J, which was discussed in the previous section. The reason is that Herault and Jutten had assumed a special

architecture of the environment by taking the diagonal of the A matrix to be one. Thus, when developing the update law, they did not consider any law for the diagonal elements of the network matrix. Therefore, the network converged to the exact inverse of the environment model. By doing so, the identifiability issue was eliminated. However, in the work of Amari et al, no such special structure is assumed. Therefore, the problem is not completely identifiable. Consequently, one should not expect to obtain the inverse of the mixing matrix as the solution to the problem.

The authors' main contribution is their analytical derivation to obtain a nonlinear function $f(.)$ which previously has been chosen in an ad hoc manner in the literature. However, only a 2^{nd} order approximation of the logarithmic function was considered in which the derivation and output signal were assumed to have unit variance (2.42). This certainly simplifies the computations. But, it does not generate the correct nonlinear function, since these output signal variances will contribute to the higher-order terms. Therefore, in Chapter 4, a new energy function that is based on 3^{rd} order approximation of the logarithmic function, with no unit variance assumption, will be derived. A justification of this derivation will become apparent in the chapter. Consequently, an update law based on the proposed energy function will perform certain separation tasks, whereas the update law defined by equation (2.47) and (2.49) will be shown to fail.

2.3 Information Theoretic Approach

The authors, Bell and Sejnowski [45, 45, 25], considered a feedforward neural network described by Figure 2.11.

$\mathbf{y}$ is the output of the neural network, $\mathbf{x}$ is its input vector, W is the parameter matrix and $\mathbf{b}$ is the bias vector.

In their work, the authors considered a criterion that maximized the information

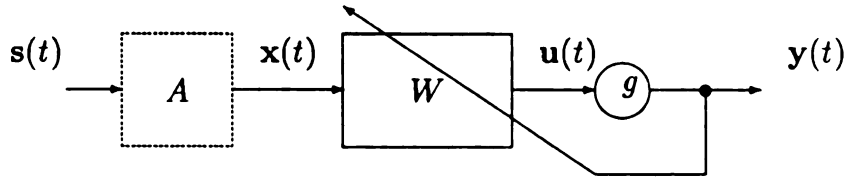


Figure 2.11. Bell and Sejnowski's Architecture

transfered through the network, which is defined by

$$I(\mathbf{x},\mathbf{y}) = E\left[\ln \frac{f_{\mathbf{xy}}(\mathbf{x},\mathbf{y})}{f_{\mathbf{x}}(\mathbf{x})f_{\mathbf{y}}(\mathbf{y})}\right] \tag{2.50}$$

Using the properties of the entropy given in appendix A, one can express equation (2.50) as

$$I(\mathbf{x},\mathbf{y}) = H(\mathbf{y}) + H(\mathbf{x}) - H(\mathbf{x},\mathbf{y}) \tag{2.51}$$

or

$$I(\mathbf{x},\mathbf{y}) = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) \tag{2.52}$$

$H(\mathbf{y})$ is the entropy of $\mathbf{y}$ and $H(\mathbf{y}|\mathbf{x})$ is the entropy of $\mathbf{y}$ not generated from the input $\mathbf{x}$. The entropy of $\mathbf{y}$ is defined as:

$$H(\mathbf{y}) = -E\left[\ln f_{\mathbf{y}}(\mathbf{y})\right] = -\int_{-\infty}^{+\infty} f_{\mathbf{y}}(\mathbf{y}) \ln f_{\mathbf{y}}(\mathbf{y})\, d\mathbf{y}$$

where $f_{\mathbf{y}}(\mathbf{y})$ is the probability density function (pdf) of $\mathbf{y}$. These author state that $H(\mathbf{y}|\mathbf{x})$ does not depend on W. Thus, maximizing the information transfered through the network was equivalent to maximizing the entropy of $\mathbf{y}$.

$$\frac{\partial I(\mathbf{x},\mathbf{y})}{\partial W} = \frac{\partial H(\mathbf{y})}{\partial W}$$

Assuming that the nonlinearity function g is invertible, then the following will hold

$$f_{\mathbf{y}}(\mathbf{y}) = \frac{f_{\mathbf{x}}(\mathbf{x})}{|J|}$$

where $|J|$ is the determinant of the Jacobian matrix and $J = \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$. So $H(\mathbf{y})$ can be rewritten as

$$H(\mathbf{y}) = E[\ln |J|] - E[\ln f_{\mathbf{x}}(\mathbf{x})] = H(\mathbf{x}) + E[\ln |J|]$$

Thus,

$$\frac{\partial H(y)}{\partial W} = E\Big[\frac{\partial \ln |J|}{\partial W}\Big]$$

Consequently, the parameters will be updated according to

$$\Delta W = \eta E\Big[\frac{\partial \ln |J|}{\partial W}\Big] \tag{2.53}$$

Bell and Sejnowski suggested an approximation by dropping the expected value:

$$\Delta W = \eta \frac{\partial \ln |J|}{\partial W} \tag{2.54}$$

To compute the gradient, a compact form for $\ln |J|$ must be found.

$$\Big[\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\Big]_{ij} = \frac{\partial y_i}{\partial x_j} = g'(u_i)\frac{d\, u_i}{d\, x_j} = g'(u_i)w_{ij} = \Big[\Lambda_{g'(\mathbf{u})}W\Big]_{ij}$$

Thus, $|J|$ can be expressed as

$$|J| = |\Lambda_{g'(\mathbf{u})}W| = |\Lambda_{g'(\mathbf{u})}|.|W| = |W|\prod_i g'(u_i) \tag{2.55}$$

where $\Lambda_{\mathbf{v}} = diag(v_1, \cdots, v_n)$.

$$\ln|J| = \ln|W| + \sum_m \ln g'(u_m) \tag{2.56}$$

Therefore

$$\begin{aligned}
\Delta w_{ij} &= \eta \frac{\partial \ln|J|}{\partial w_{ij}} && (2.57)\\
&= \eta\Big[\frac{1}{|W|}\frac{\partial |W|}{\partial w_{ij}} + \frac{1}{g'(u_i)}\frac{\partial g'(u_i)}{\partial w_{ij}}\Big] && (2.58)\\
&= \eta\Big[\frac{coef_{w_{ij}}}{|W|} + \frac{g''(u_i)}{g'(u_i)}x_j\Big] \qquad \text{since } |W| = \sum_j w_{ij} coef_{ij} && (2.59)\\
&= \eta\Big[W^{-T} + \frac{g''(\mathbf{u})}{g'(\mathbf{u})}\,\mathbf{x}^T\Big]_{ij} && (2.60)
\end{aligned}$$

In matrix form

$$\Delta W = \eta\Big[W^{-T} + \frac{g''(\mathbf{u})}{g'(\mathbf{u})}\,\mathbf{x}^T\Big] \tag{2.61}$$

In particular, if

$$g(x) = \frac{1}{1+e^{-x}} \tag{2.62}$$

Then

$$\Delta W = \eta\Big[W^{-T} + (\mathbf{1} - 2\mathbf{y})\,\mathbf{x}^T\Big] \tag{2.63}$$

Computer simulations were performed in order to study the update law defined by (2.63). Two prototype sine functions of respective frequencies $1kHz$ and $2kHz$ were used as unknown sources. Computer simulations of various mixing matrices, initial conditions and learning rates were performed. They all revealed that the algorithm failed to separate signals. Figure 2.12 shows one example of its performance. Observe that the performance index did not converge to zero and that the gain matrix $P = WA$

is not a generalized permutation matrix. One could justify the failure of the update law defined by (2.61) because it is an approximation of (2.53) in which the expected value was dropped.

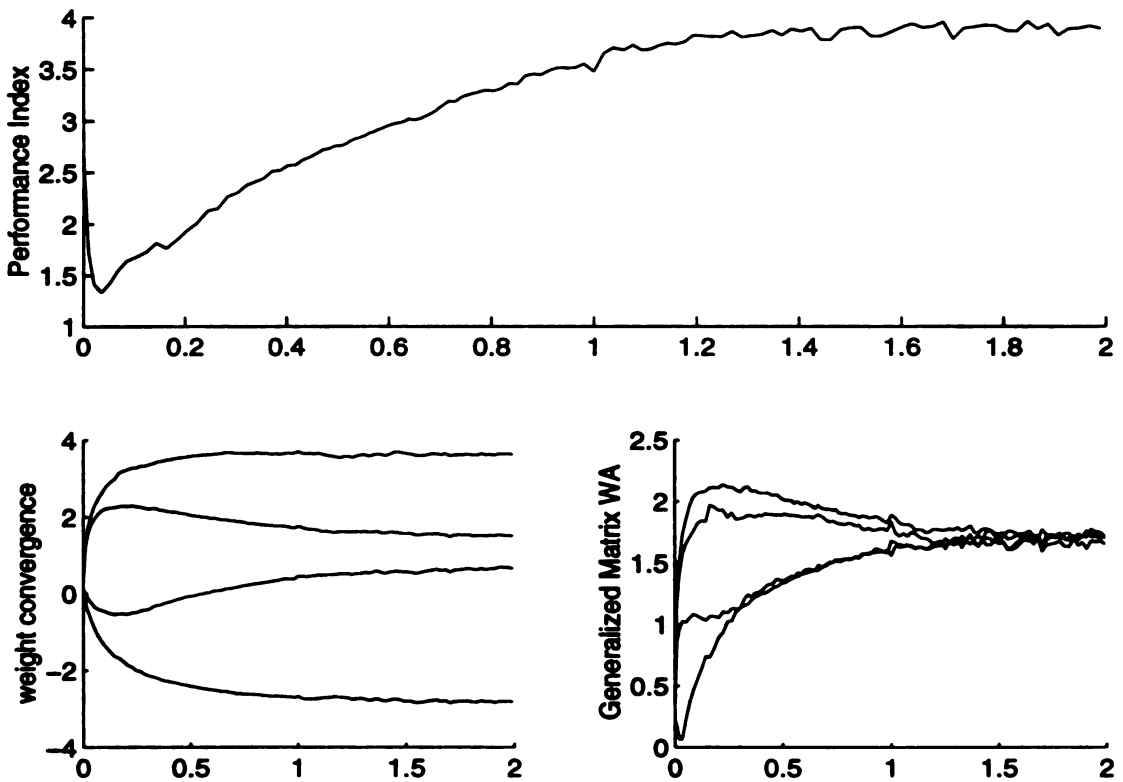


Figure 2.12. Performance of Bell and Sejnowski Algorithm

In their work [25, 45, 26, 44], the authors did not simulate the update law as it is defined in (2.61). Instead, they considered the following algorithm. Given n mixture defined in a time interval $[0, T]$, the authors first defined:

$$\delta(t) = (\mathbf{1} - 2\mathbf{y}(t))\mathbf{x}(t)^T$$

Table 2.1. Information Theoretic Algorithm

Step	Action
1	Pick a random time point t_i such that $t_i + \tau \leq T$.
2	Compute $\bar{\Delta} W_i = \frac{1}{\tau} \sum_{t_i}^{t_i+\tau} \delta(t)$
3	Evaluate W_i^{-T}
4	Compute $\Delta W_i = W_i^{-T} + \bar{\Delta} W_i$
5	Update the weights $W_{i+1} = W_i + \eta \Delta W_i$
6	Compute performance index P_i
7	While $P_i > P_{desired}$, go back to step 1

Then the weights were updated according to the steps given in Table 2.1.

Computer simulations of this algorithm were conducted using speech and prototype sine signals. The algorithm performed well when speech signals were used. It was able to separate mixtures of two, three and five speech segments of different speakers. Figure 2.13 shows one example of its performance. However, the algorithm failed to separate prototype sine signals. Observe Figure 2.14.

One undesirable feature of the algorithm developed by the authors is the choice of a random time t_i at every epoch. The authors claimed that such a choice of t_i would guarantee the input stationarity assumption. However, this choice makes the algorithm implementation in digital signal processing cumbersome since it requires the storage of all the data in memory. Consequently, the algorithm does not perform in real time. Most applications of the blind separation of signals require real time performance. Thus, the algorithm, as it is, is not attractive and needs to be modified.

One other undesirable feature of the algorithm is the computation of the inverse of the weight matrix W. By analyzing the steps of the algorithm closely, one concludes that the computation of the inverse occurs after all the weights were accumulated.

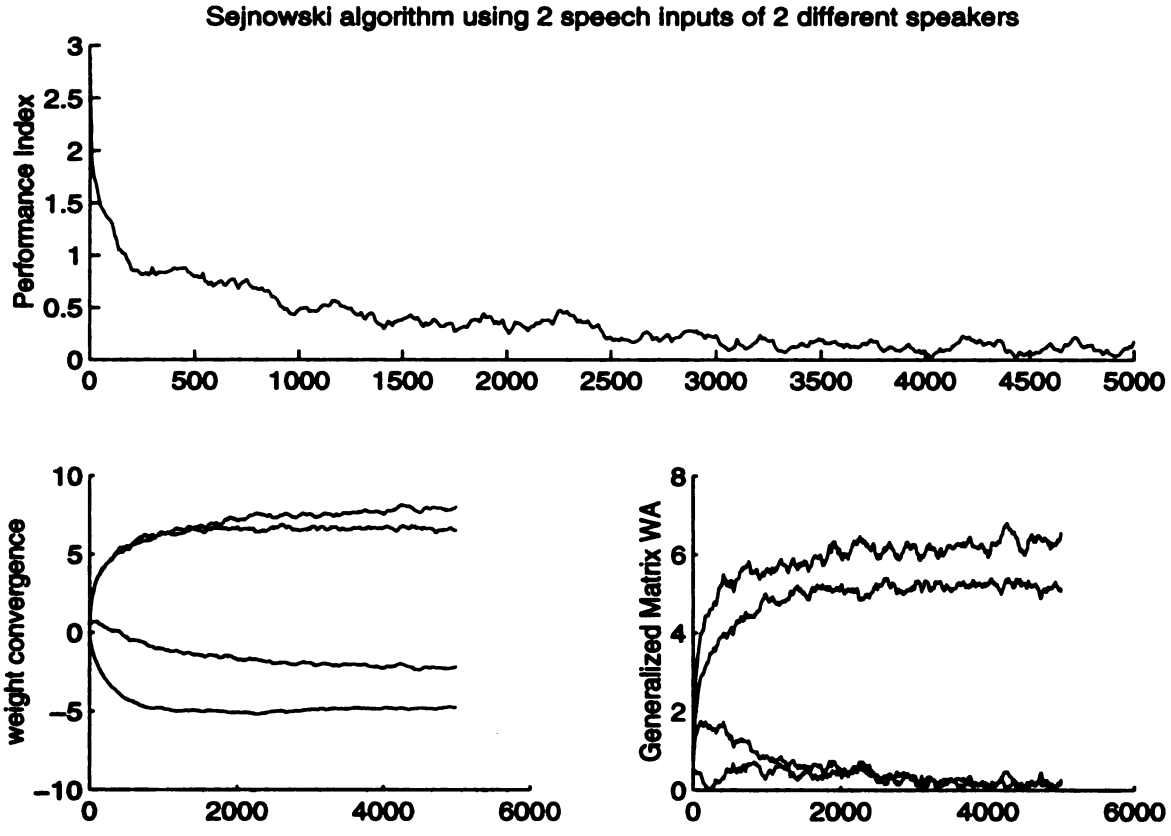


Figure 2.13. Performance of Bell and Sejnowski Algorithm Using Speech Signals

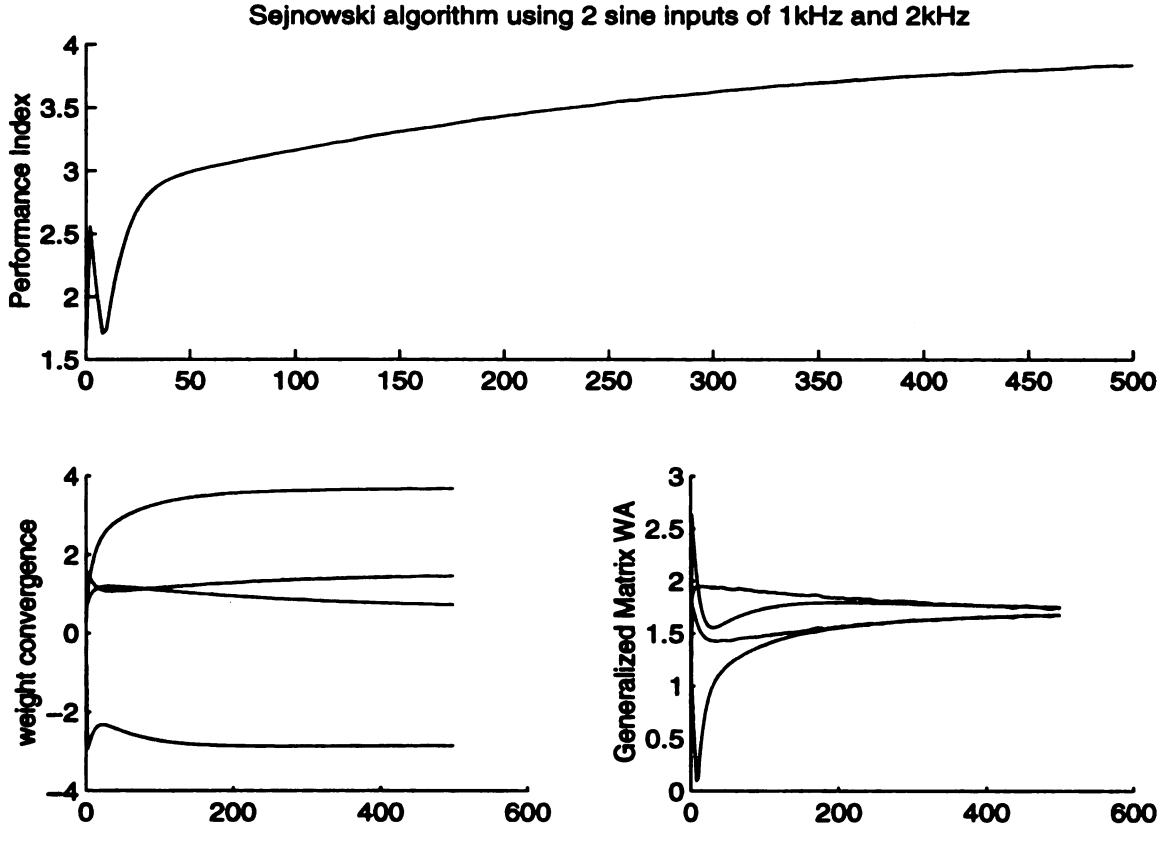


Figure 2.14. Performance of Bell and Sejnowski Algorithm Using Sine signals

This means that the input signals are frozen until such computation is completed. Thus, the algorithm is not suitable for real time performance as is. One should reconsider the way, how and/or when such an expression is computed in order to make the algorithm perform in real time. One solution would be to compute the inverse while accumulating the weights. This means that the operation of weight accumulation and inverse computation would be performed in parallel.

As a final remark, the algorithm defined by (2.63) is based on the maximization of the information transferred through the network, defined by equation (2.52), with appropriate choice of the neuron's nonlinear function. One can interpret this development based on the minimization of of the first term of the mutual information defined by equation (2.39), namely the entropy of the output vector $-H(\mathbf{y})$. This is equivalent to the maximization of $H(\mathbf{y})$.

2.4 Algebraic Approach

In [21, 56, 57, 22], Cardoso et al. proposed a series of two processes as shown in the Figure below to achieve blind signal separation. The mixing matrix W was factored out as the product of whitening matrix B and an orthogonalizing matrix U; $W = UB$. Algorithms for updating the matrices B and U were developed, then combined to obtain a *one-stage* update for the matrix W.

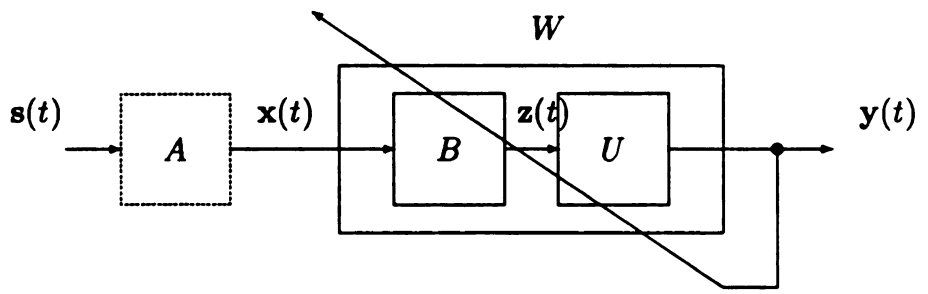


Figure 2.15. Cardoso's Architecture

The matrix B was updated such that the output vector $\mathbf{z}$ was white, *i.e.* $R_{\mathbf{z}} = I$ where $\mathbf{z} = B\mathbf{x}$. This was obtained by minimizing the distance between the matrices $R_{\mathbf{z}}$ and I. Thus, the appropriate error function was the Kullback-Leibler divergence [58] defined as

$$E_b(B) = Trace(R_{\mathbf{z}}) - \ln |R_{\mathbf{z}}| - n \tag{2.64}$$

The matrix B was then updated along its gradient descent:

$$\Delta B = -\eta \frac{\partial E_b}{\partial B} = -\eta[\mathbf{z}\mathbf{z}^T - I)B \tag{2.65}$$

The matrix U was updated such that $E_u(U) = E\ f(\mathbf{y}) = E\ f(U\mathbf{z})$ was minimized. Since this minimization was not constrained and may lead to a solution that did not preserve orthogonality, the matrix U will be updated according to

$$\Delta U = -\frac{\eta}{2}\Big[\frac{\partial E_u}{\partial U} - \frac{\partial E_u}{\partial U}^T\Big] = -\eta\Big[f'(\mathbf{y})\mathbf{y}^T - \mathbf{y}f'(\mathbf{y})\Big]U \tag{2.66}$$

By combining the whitening and the orthogonality stages, Cardoso et al obtained an overall update for the matrix B:

$$\Delta W = \eta\Big[I - \mathbf{y}\mathbf{y}^T - f'(\mathbf{y})\mathbf{y}^T + \mathbf{y}f'(\mathbf{y})\Big]W \tag{2.67}$$

Computer simulations were performed to study the update law defined by (2.67). When the network dimension was two, the algorithm was able to separate signals regardless of the initial conditions and the learning rate η. Figure 2.16 presents an example of performance.

When developing the algorithm described by (2.67), the authors considered a Euler approximation of the gradient of the considered energy (contrast) functions. The developed algorithm is suitable for digital implementation. Here, I will consider its

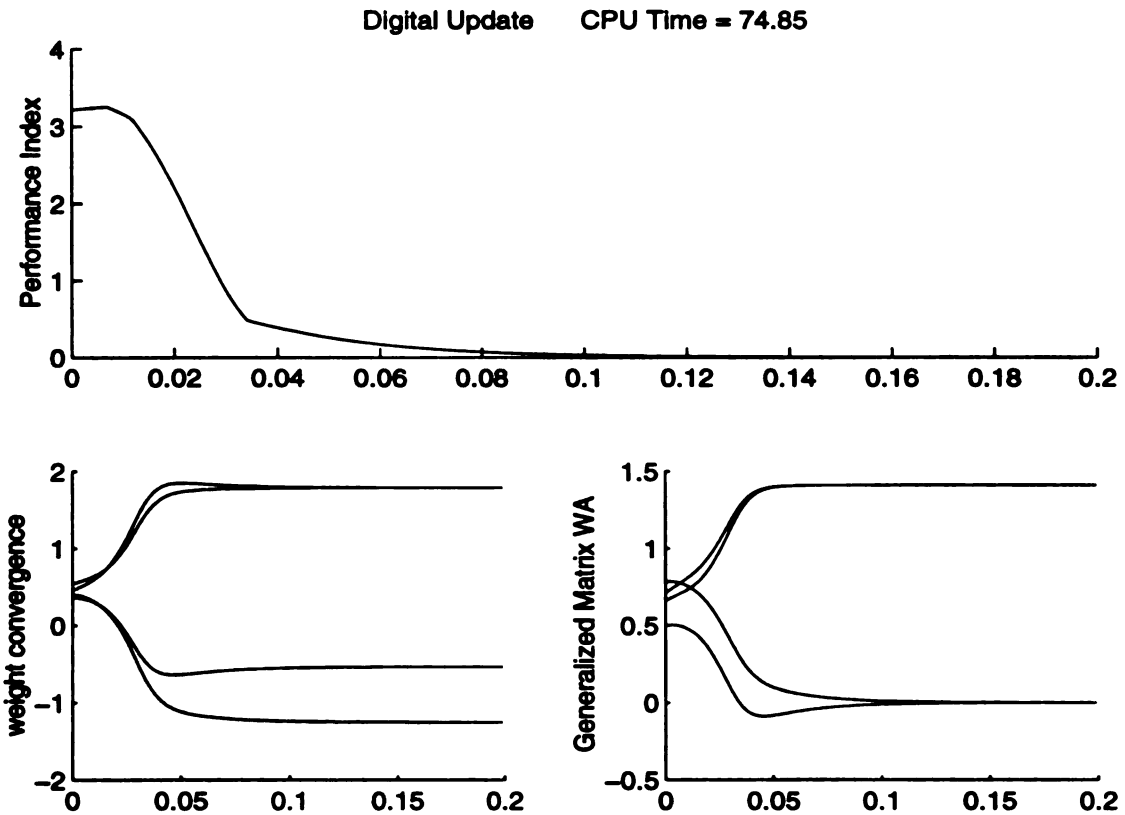


Figure 2.16. Cardoso Algorithm for n =2

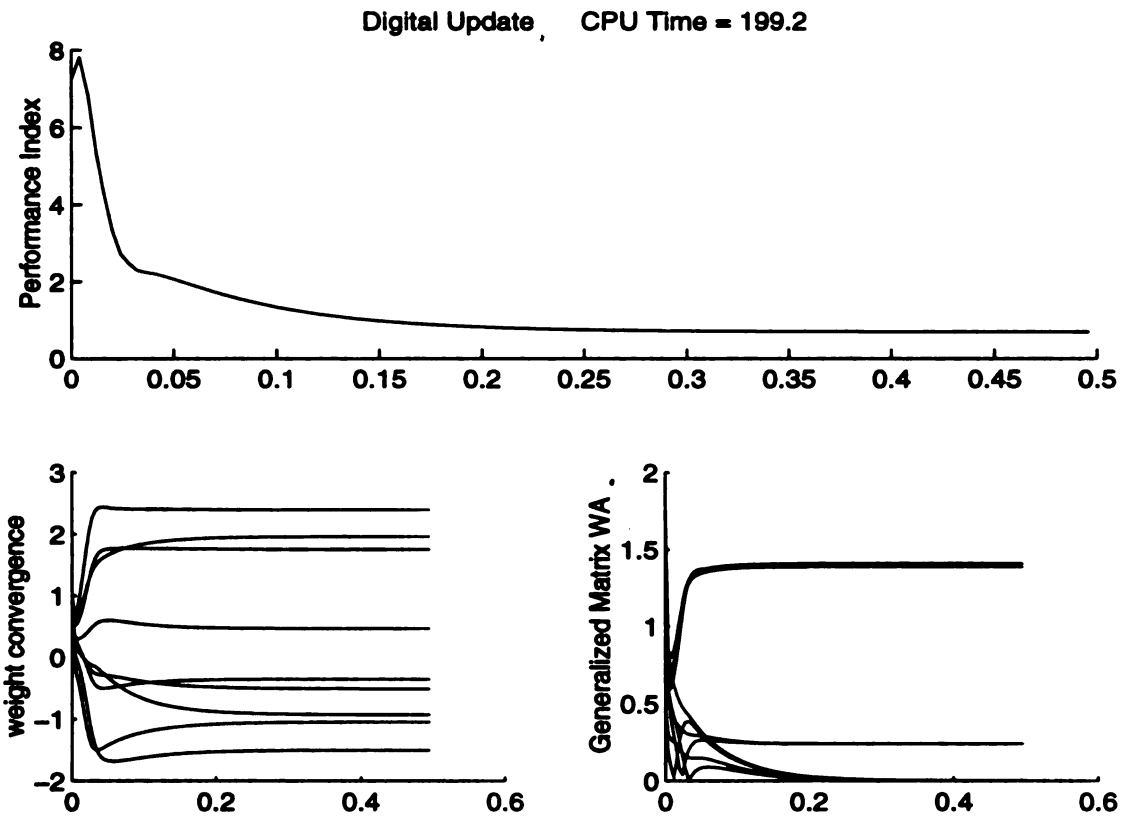


Figure 2.17. Cardoso Algorithm for n =3

continuous-time realization of the algorithm, investigate its performance and compare it to its discrete-time counter part as defined by (2.67). The continuous-time algorithm is now defined by the instantaneous update equation

$$\dot{W} = \eta\left[I - \mathbf{y}\mathbf{y}^T - f'(\mathbf{y})\mathbf{y}^T + \mathbf{y}f'(\mathbf{y})\right]W \tag{2.68}$$

Starting from the same conditions, computer simulations for both realizations were performed. A comparison of the cpu time taken by each algorithm is considered. It can be concluded that the discrete-time version of the algorithm took more time than its continuous-time counterpart when one compares the $74.85sec$ that the discrete-time algorithm took compared to the $50.93sec$ for the continuous-time algorithm. Examples of simulations for a two dimensional network are presented in Figures 2.16 and 2.18. However, when the network is of dimension 3, the algorithm converges, but not to an acceptable solution. Observe in Figures 2.17 and 2.19 how more than 3 entries of the generalized permutation matrix converge to non-zero values. Thus, the solution is not acceptable. One concludes that the algorithm developed by Cardoso may not work for networks of dimension higher than 2.

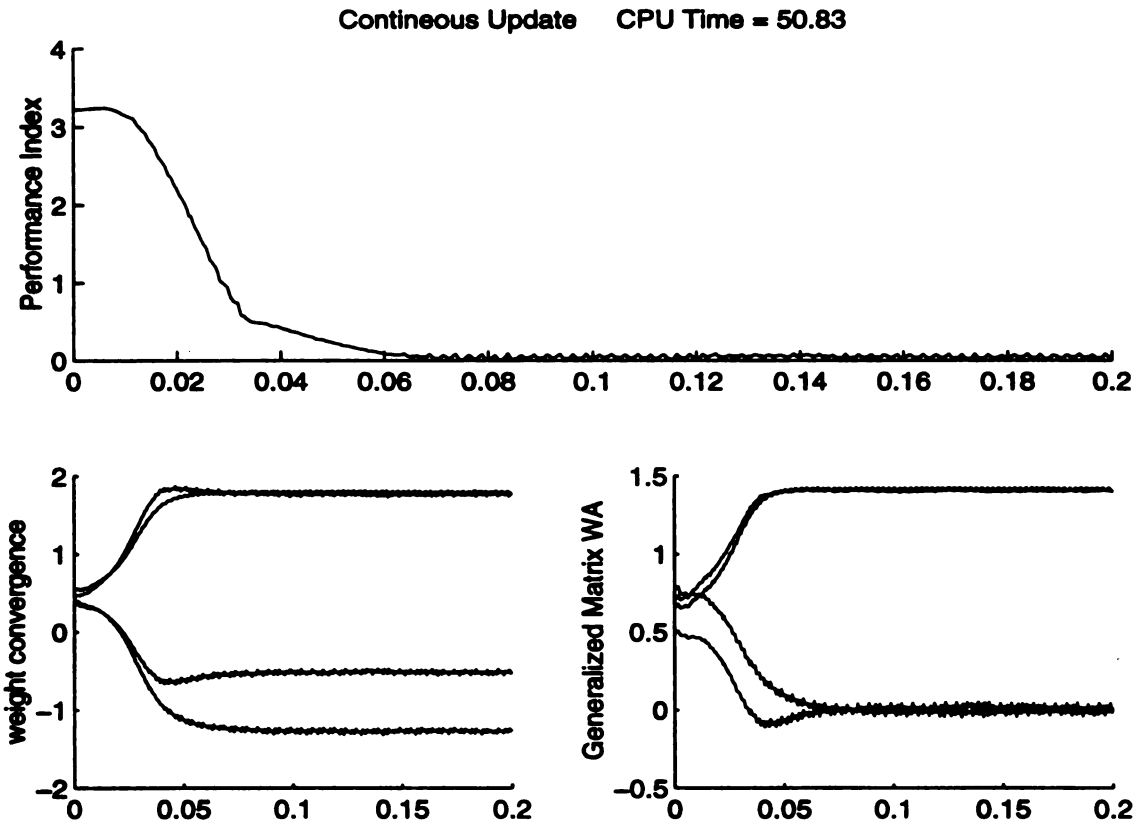


Figure 2.18. Continuous-time Algorithm for n =2

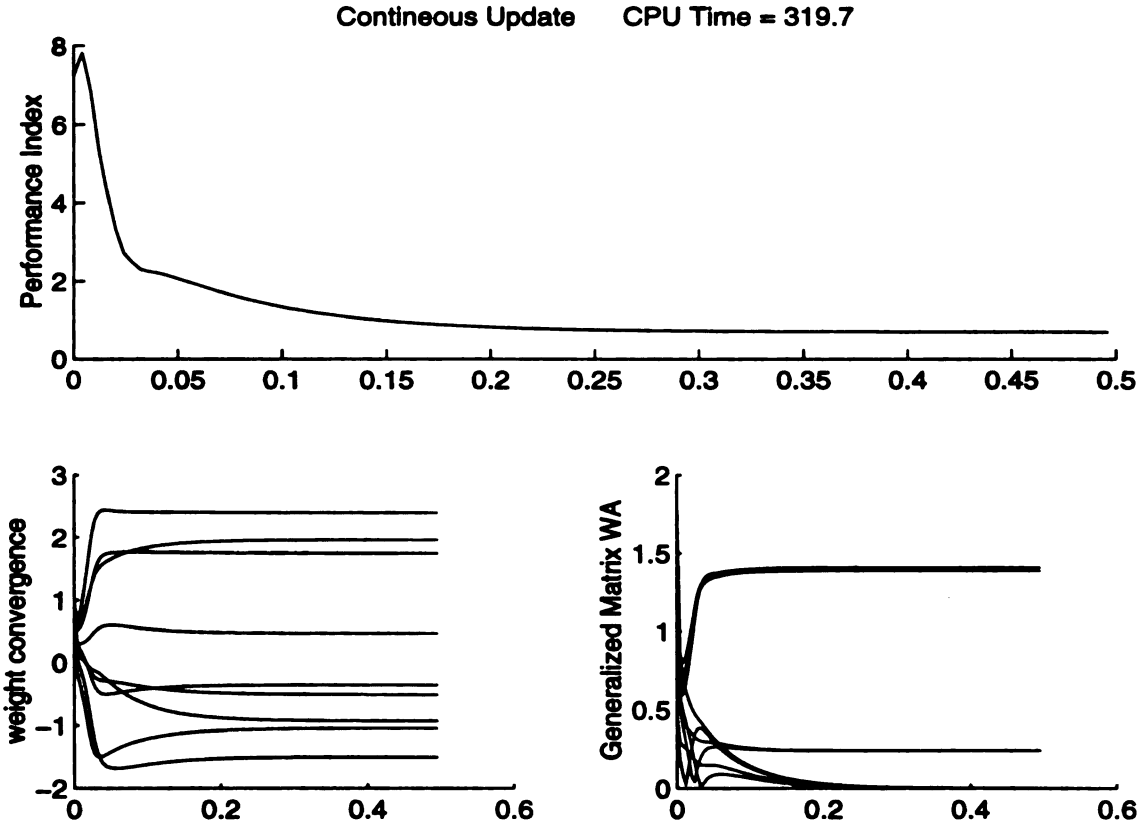


Figure 2.19. Continuous-time Algorithm for n =3

CHAPTER 3

Blind Separation in a Static Environment

In this chapter, a blind separation algorithm in static environment will be developed. The mathematical analysis to develop adaptive laws for the problem will be presented. The law will be derived based on the nonlinear decorrelation condition of the output. The resulting update laws will then b to test for independence of the output. Computer simulations will be presented to demonstrate the performance of these novel algorithms.

3.1 Problem Definition and Architecture

The problem can be posed as follows: given that some unknown sources are sent from unknown sources, these sources are mixed according to some unknown model that describes the medium through which these sources have traveled. The goal then, is to construct a system that recovers these unknown sources based only on the measurements of the mixtures of the original unknown sources. It is also assumed that the environment is a static model. It can, therefore, be represented by a static matrix to characterize its input-output relationship. Figure 3.1 describes the architecture of

the system. The unknown sources, the output of the environment and of the network are respectively labeled as $\mathbf{s}(t)$, $\mathbf{x}(t)$ and $\mathbf{y}(t)$.

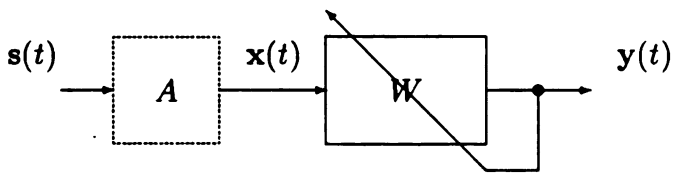


Figure 3.1. Static Environment Architecture

3.2 Theoretical Solution

Given the measured vector $\mathbf{x}(t)$, which is the only given data, the neural network should be constructed to adaptively change the parameter W so that its output $y(t)$ and the original signal $s(t)$ will satisfy the wave preserving property which was defined section 2.1. This property implies that one is not after an exact replica of the unknown, but rather a permutation and up to a constant of them.

Now, assume that the correct update law of the problem is developed and the system converged to a solution W^*. Then,

$$\mathbf{y}(t) = W^*\mathbf{x}(t) = W^*A\mathbf{s}(t) \tag{3.1}$$

Therefore, by combining equations (2.32) and (3.1), one obtains

$$W^*\, A = P\, \Gamma \tag{3.2}$$

Recall that P and Γ are respectively a permutation and nonsingular diagonal matrix.

Equation (3.2) can be rewritten as

$$W^* = P\ \Gamma\ A^{-1} \tag{3.3}$$

This equation shows that a theoretical solution to the problem does exist and that there are infinitely many possible solutions, thanks to the freedom that the matrices P and Γ provide. Also, equation (3.2) requires that the gain matrix $G = WA$ is a generalized permutation matrix.

Knowing that a theoretical solution exists, one can now investigate possible ways to construct a network with appropriate update laws that perform the task of separating signals. To do so, an appropriate energy function is defined next.

3.3 Energy Function

To develop an update law, an energy function that characterizes the problem is now defined. If it were a recognition or a classification problem, the energy function would have been chosen as an error function; a difference between the desired output vector $\mathbf{s}(t)$ and the observed output vector $\mathbf{y}(t)$ defined as:

$$\phi(\underline{y}, \underline{s}) = \frac{1}{2} \int \sum_i (s_i(t) - y_i(t))^2\ dt \tag{3.4}$$

This criterion will, therefore, require the knowledge of the target signal. However, by assumption, there is no a priori knowledge of the signal vector. In this problem, it is expected that the network will reproduce, regenerate, and recover these sources without any direct knowledge of them. Thus, such a traditional energy function will not be appropriate.

On the other hand, some kind of knowledge about the original signals is assumed: they are independent. This feature or criterion will be the basis for defining the

energy function analogous to the one defined by (3.4). In order to define the energy function that fits the problem at hand, one needs to rely on the definition of the independence of signals [59].

Definition 2 (Deterministic Independence) *The components of an n-dimensional time-varying signal vector* $\mathbf{y}(t) = [y_1(t) \cdots y_n(t)]^T$ *are linearly independent over a time interval* $[0, T]$ *if and only if their auto-correlation matrix is positive definite for all time* $t \in [0, T]$.

$$R_{\mathbf{y}}(t) = \int_0^t \mathbf{y}(\tau)\, \mathbf{y}(\tau)^T \, d\tau \qquad \textit{is positive definite for all } t \leq T$$

Based on this definition, one possible energy function could be based on identifying necessary and sufficient conditions for the autocorrelation matrix to be positive definite. Theorem 1 defines this characterization [60].

Theorem 1 (Positive Definite) *If a matrix* M *is positive definite, then*

$$|M| \leq \prod_{k=1}^{n} m_{kk} \tag{3.5}$$

with equality if M *is diagonal.*

The operator $|M|$ denotes the determinant in the case of a square matrix M. The proof of this theorem is arrived at by induction and shown in appendix B.1. Using Theorem 1, one can develop a criterion, or energy function, that is always positive, but is minimum when the matrix is diagonal. To develop such a function, some equivalent statements of Theorem 1 will be presented. Let M be a positive definite matrix. Then,

$$\begin{aligned} M \text{ is positive definite} \quad &\Rightarrow \quad \prod_{k=1}^{n} m_{kk} \geq |M| \\ &\Rightarrow \quad \ln \prod_{i=1}^{n} m_{kk} \geq \ln |M| \end{aligned}$$

$$\Rightarrow \sum_{k=1}^{n} \ln m_{kk} - \ln |M| \geq 0$$

with equality if and only if M is diagonal.

Define the mapping

$$\begin{aligned} J: \; R^{n\times n} &\longrightarrow R^{+} \\ M &\longmapsto J(M) = \frac{1}{2}\Big[\sum_{k=1}^{n} \ln m_{kk} - \ln |M|\Big] \end{aligned} \tag{3.6}$$

When the energy function defined by (3.6) is minimized, one would obtain a diagonal matrix. Thus, this energy function will be used in this problem by considering the autocorrelation matrix, since the decorrelation of the components of the output signal vector is obtained when the autocorrelation matrix of the output vector is diagonal. In the work presented in this chapter, an update law that pushes for the decorrelation condition only at the first step will be developed. Then, some techniques found in the literature [8, 24] to push for the independence condition will be used. Also, different numbers of measurements and sensors will be considered. Thus, the matrix W is an $n \times m$ where $m \neq n$. A special form of the update law for the different scenarios of the relationships between the number of sensors and measurements will be given.

The autocorrelation matrix of the output signal vector is defined as

$$R_{\mathbf{y}}(t) = \int_{t_0}^{t} \mathbf{y}(\tau)\mathbf{y}(\tau)^T \, d\tau = < \mathbf{y}\mathbf{y}^T >_t \tag{3.7}$$

Thus, using the mapping defined by equation (3.6), the energy function will be defined as

$$\Phi = J(R_{\mathbf{y}}(t)) = \frac{1}{2}\Big[\sum_{k=1}^{n} \ln < y_k^2 >_t - \ln | < \mathbf{y}\mathbf{y}^T >_t | \Big] \tag{3.8}$$

The matrix W will be updated along the gradient descent of Φ

$$\dot{W} = -\eta \frac{\partial \Phi}{\partial W} \tag{3.9}$$

$$= \eta \Big[\frac{1}{2} \frac{1}{|< \mathbf{y}\mathbf{y}^T >_t|} \frac{\partial |< \mathbf{y}\mathbf{y}^T >_t|}{\partial W} - \frac{1}{2} \sum_{k=1}^{n} \frac{1}{< y_k^2 >_t} \frac{\partial < y_k^2 >_t}{\partial W} \Big] \tag{3.10}$$

3.4 Update law derivation

First compute

$$\frac{\partial < y_k^2 >_t}{\partial w_{ij}} = 2 < y_k \frac{\partial y_k}{\partial w_{ij}} >_t$$

However,

$$y_k = \sum_l w_{kl} x_l$$

Therefore,

$$\frac{\partial y_k}{\partial w_{ij}} = \sum_l \frac{\partial w_{kl}}{\partial w_{ij}} x_l = \sum_l \delta_{ki} \delta_{kj} x_l = \delta_{ki} x_j$$

Consequently,

$$\frac{\partial < y_k^2 >_t}{\partial w_{ij}} = 2\delta_{ki} < y_k x_j >_t \tag{3.11}$$

So,

$$\begin{aligned} \frac{1}{2} \sum_{k=1}^{n} \frac{1}{< y_k^2 >_t} \frac{\partial < y_k^2 >_t}{\partial W} &= \frac{1}{2} \sum_{k=1}^{n} \frac{1}{< y_k^2 >_t} 2\delta_{ki} < y_k x_j >_t \\ &= \frac{< y_i x_j >_t}{< y_i^2 >_t} \\ &= \Big[(diag < \mathbf{y}\mathbf{y}^T >_t)^{-1} < \mathbf{y}\mathbf{x}^T >_t \Big]_{ij} \end{aligned} \tag{3.12}$$

The other term

$$\frac{\partial|<\mathbf{y}\mathbf{y}^T>_t|}{\partial W} = \frac{\partial|<W\mathbf{x}\mathbf{x}^T W^T>_t|}{\partial W} \tag{3.13}$$

In [60], it was derived that for any square matrix M and any matrix X such that XMX^T is nonsingular

$$\frac{\partial|XMX^T|}{\partial X} = |XMX^T|\left[(XMX^T)^{-1}XM + (XMX^T)^{-T}XM^T\right] \tag{3.14}$$

In addition, if M is symmetric, equation (3.14) becomes

$$\frac{\partial|XMX^T|}{\partial X} = 2|XMX^T|(XMX^T)^{-1}XM \tag{3.15}$$

Applying equation (3.15) to equation (3.13)

$$\begin{aligned}\frac{\partial|<\mathbf{y}\mathbf{y}^T>_t|}{\partial W} &= 2<|W\mathbf{x}\mathbf{x}^T W^T|(W\mathbf{x}\mathbf{x}^T W^T)^{-1}W\mathbf{x}\mathbf{x}^T>_t \\ &= 2<|\mathbf{y}\mathbf{y}^T|(\mathbf{y}\mathbf{y}^T)^{-1}\mathbf{y}\mathbf{x}^T>_t \end{aligned} \tag{3.16}$$

Plug in equations (3.12) and (3.16) into (3.10), and one obtains

$$\dot{W} = \eta\left[\frac{<|\mathbf{y}\mathbf{y}^T|(\mathbf{y}\mathbf{y}^T)^{-1}\mathbf{y}\mathbf{x}^T>_t}{<|\mathbf{y}\mathbf{y}^T|>_t} - (diag<\mathbf{y}\mathbf{y}^T>_t)^{-1}<\mathbf{y}\mathbf{x}^T>_t\right] \tag{3.17}$$

If W is a square nonsingular matrix, then (3.17) becomes

$$\dot{W} = \eta\left[W^{-T} - (diag<\mathbf{y}\mathbf{y}^T>_t)^{-1}<\mathbf{y}\mathbf{x}^T>_t\right] \tag{3.18}$$

$$= \eta\left[I - (diag<\mathbf{y}\mathbf{y}^T>_t)^{-1}<\mathbf{y}\mathbf{y}^T>_t\right]W^{-T} \tag{3.19}$$

$$= \eta\left[I - (diag R_{\mathbf{y}})^{-1}R_{\mathbf{y}}\right]W^{-T} \tag{3.20}$$

At steady state, equation (3.19) becomes

$$\begin{aligned} \dot{W} = 0 \;&\Rightarrow\; I = (diag < \mathbf{y}\mathbf{y}^T >_t)^{-1} < \mathbf{y}\mathbf{y}^T >_t \\ &\Rightarrow\; < \mathbf{y}\mathbf{y}^T >_t = diag < \mathbf{y}\mathbf{y}^T >_t \\ &\Rightarrow\; < y_i y_j >_t = 0, \forall i \neq j \\ &\Rightarrow\; \text{the components of the signal vector } \mathbf{y}(t) \text{ are decorrelated.} \end{aligned}$$

Thus, the algorithm defined by equation (3.19) tests only for decorrelation.

In order to compute the weight update according to equation (3.19), the time average $< \mathbf{y}\mathbf{y}^T >_t$ defined by equation (3.7) must be computed. To do so, a state matrix $Z(t)$ is defined as

$$\dot{Z}(t) = \mathbf{y}(t)\mathbf{y}(t)^T, \qquad \text{with } Z(t_0) = 0 \tag{3.21}$$

Thus,

$$Z(t) = \int_{t_0}^{t} \mathbf{y}(\tau)\mathbf{y}(\tau)^T \, d\tau = < \mathbf{y}\mathbf{y}^T >_t$$

Therefore, equation (3.19) is simplified to

$$\dot{W} \;=\; \eta\left[I - (diag\ Z)^{-1} Z\right] W^{-T} \tag{3.22}$$

By running equations (3.21) and (3.22) simultaneously, the algorithm defined by (3.19) will be successfully implemented.

3.5 Computer Simulations

Computer simulations were performed to study the derived algorithm. Starting with two sine waveforms as the unknown sources and mixing them by a matrix A, the

mixture signal vector will be obtained. This signal is then fed to the feed forward neural network. If the output of the network is a constant of any of the two unknown sources, then the algorithm would have succeeded in separating signals. The separation can also be achieved if the forward loop gain matrix $P = WA$ is a generalized permutation matrix, where a generalized permutation (GP) matrix is one that has only one nonzero entry in each row and column.

Numerous simulations for various initial conditions and learning rates were performed in order to study the algorithm as defined by (3.20). In all these simulations, the algorithm failed to separate the signals. An example of such simulations is shown in figure 3.2. The initial condition is

$$W_0 = \begin{bmatrix} 0.1352 & 0.4553 \\ 0.7832 & 0.3495 \end{bmatrix}$$

The network converged to

$$W = \begin{bmatrix} 0.2265 & 0.4187 \\ 0.8116 & 0.2758 \end{bmatrix}$$

which gives the over all gain matrix

$$WA = \begin{bmatrix} 0.1277 & 0.3293 \\ 0.9226 & -0.3700 \end{bmatrix}$$

Figure 3.2 shows that the performance index does not go to zero, implying that the gain matrix WA is not a generalized permutation matrix. This can also be observed by looking at the plots of its entries and noting that none of them go to zero. If the algorithm converged to a separating matrix W, then two entries of the gain matrix should go to zero. However, the figure shows that all the entries converged to nonzero values.

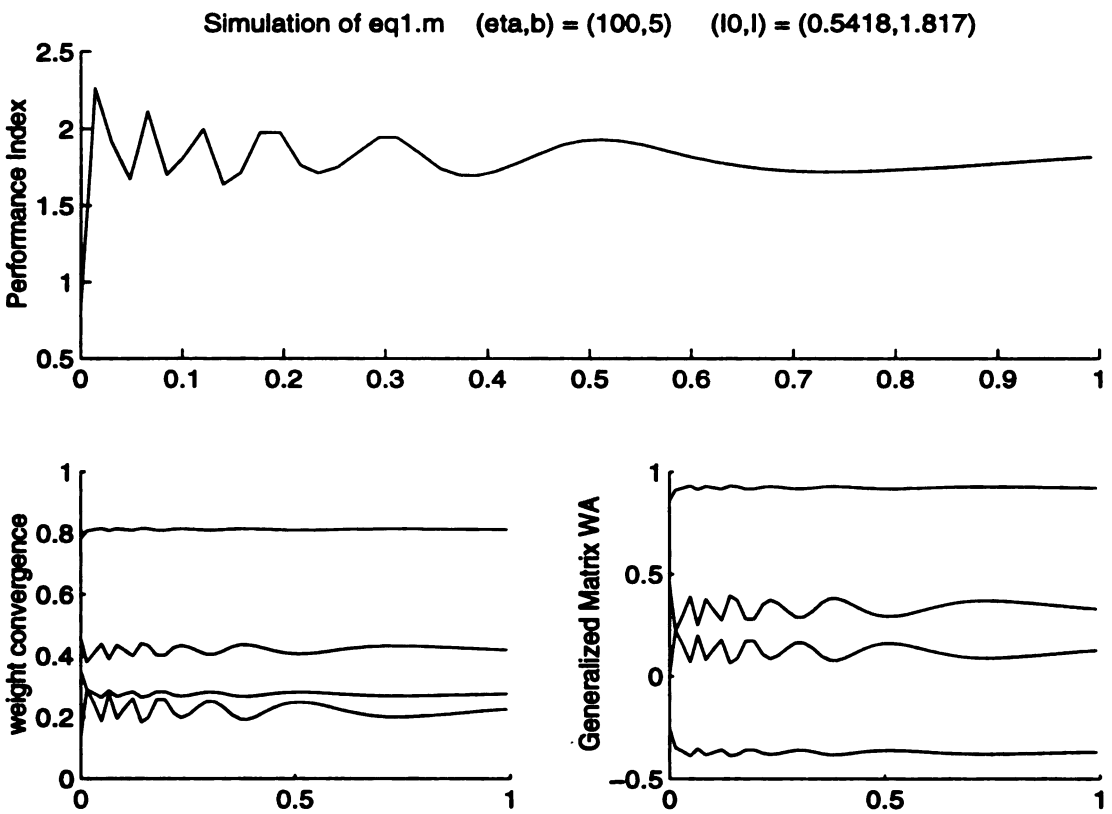


Figure 3.2. Performance of Algorithm defined by equation (3.19)

3.6 First Improvement

The algorithm defined in (3.19) failed to perform a separation of signals because it tested only for decorrelation of the output signal. In [8, 9, 10], Herault and Jutten also arrived initially at an algorithm that was similar to the one derived in (3.19). They claimed that independence would be satisfied if higher order moments of the output are generated within the update rule. Therefore, some odd nonlinear functions that would exhibit an infinitely many order of moments by expanding these functions in their Taylor series expressions was introduced.

Thus, by considering a similar approach [8], as was discussed in Chapter 2, (3.19) becomes

$$\dot{W} = \eta\Big[I - (diag < f(\mathbf{y})g(\mathbf{y})^T >_t)^{-1} < f(\mathbf{y})g(\mathbf{y})^T >_t\Big]W^{-T} \tag{3.23}$$

At steady state, (3.23) is

$$\begin{aligned}
\dot{W} = 0 \;&\Rightarrow\; I = (diag < f(\mathbf{y})g(\mathbf{y})^T >_t)^{-1} < f(\mathbf{y})g(\mathbf{y})^T >_t \\
&\Rightarrow\; < f(\mathbf{y})g(\mathbf{y})^T >_t = diag < f(\mathbf{y})g(\mathbf{y})^T >_t \\
&\Rightarrow\; < f(y_i)g(y_j) >_t = 0,\ \forall i \neq j \\
&\Rightarrow\; < \sum_{l,k\ odd} \alpha^l \beta^k y_i^l y_j^k >_t = 0,\ \forall i \neq j \\
&\Rightarrow\; < y_i^l y_j^k >_t = 0,\ \forall i \neq j, \forall l,\ \forall k
\end{aligned}$$

where the coefficients α_l and β_k are the Taylor series coefficients of the functions f and g, respectively. This proves that the solution of this algorithm provides independent output signals as a solution to the problem, which is the desired goal.

3.6.1 Computer Simulations

To evaluate the update of the weight according to (3.23), the computation of $< f(\mathbf{y})g(\mathbf{y})^T >_t$ must be performed. Thus, the state matrix

$$\dot{Z}(t) = f(\mathbf{y}(t))g(\mathbf{y}(t))^T, \qquad \text{with } Z(t_0) = 0 \tag{3.24}$$

is defined. Consequently, (3.23) becomes

$$\dot{W} = \eta\left[I - (diag\ Z)^{-1}Z\right]W^{-T} \tag{3.25}$$

By running (3.24) and (3.25) simultaneously, the algorithm defined by (3.23) is realized.

Computer simulations for the algorithm defined by (3.23) were performed using different nonlinear functions. Therefore, different cases are considered.

Case 1: $f(x) = x^3 \qquad g(x) = x$

Numerous simulations were performed using different initial conditions and learning rates. The following set of simulations was obtained by using the same initial conditions

$$W_0 = \begin{bmatrix} 0.4523 & 0.9317 \\ 0.8089 & 0.6516 \end{bmatrix}$$

but, different learning rates. Figures 3.3 through 3.5 show the performance of algorithm (3.23) when the learning rate is equal to $\eta(t) = 100,\ 10,\ 100e^{-5t}\ \forall t$. In figure 3.3, One can observe that the algorithm converges. However, it exhibits some oscillations and never settles to a constant level of zero, though it oscillates around it. This

behavior can be eliminated if a smaller learning rate is considered. Figure 3.4 shows the simulations under the same conditions when $\eta = 10$. It can be observed that a longer time is needed for the algorithm to converge. Therefore, a way of combining the observed behaviors in figures 3.3 and 3.4 is to consider a time-varying learning rate. Thus, when $\eta = 100e^{-5t}$, one observes a smooth convergence of the algorithm as shown in figure 3.5.

Table 3.1 shows the performance of the algorithm for 20 different random initial conditions, but all having the same time-varying learning rate $\eta = 100e^{-5t}$. It can be observed that the algorithm converges in all cases.

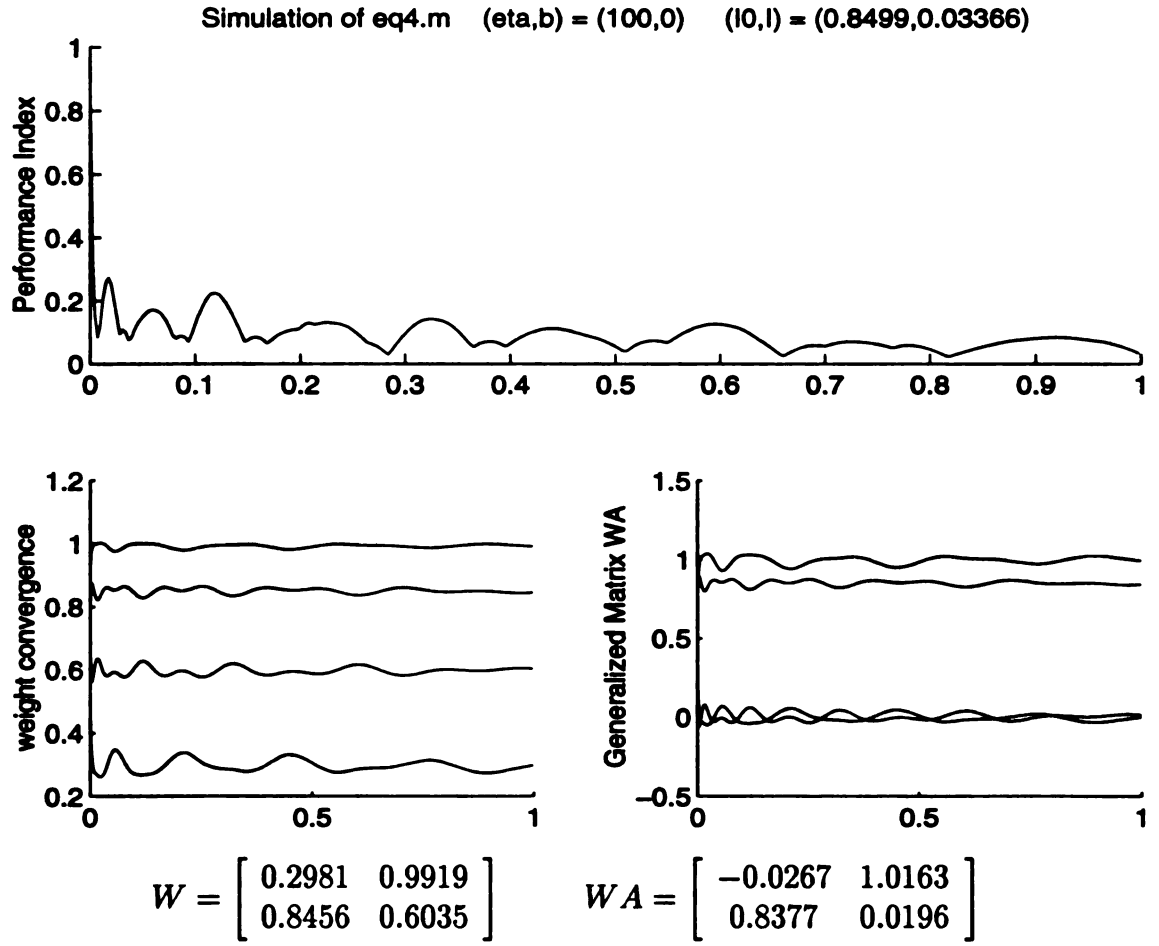
Case 2 $f(x) = \sinh x \qquad g(x) = \tanh x$

Numerous simulations were performed using different initial conditions and learning rates. The following set of simulations was obtained by using the same initial conditions

$$W_0 = \begin{bmatrix} 0.1352 & 0.4553 \\ 0.7832 & 0.3495 \end{bmatrix}$$

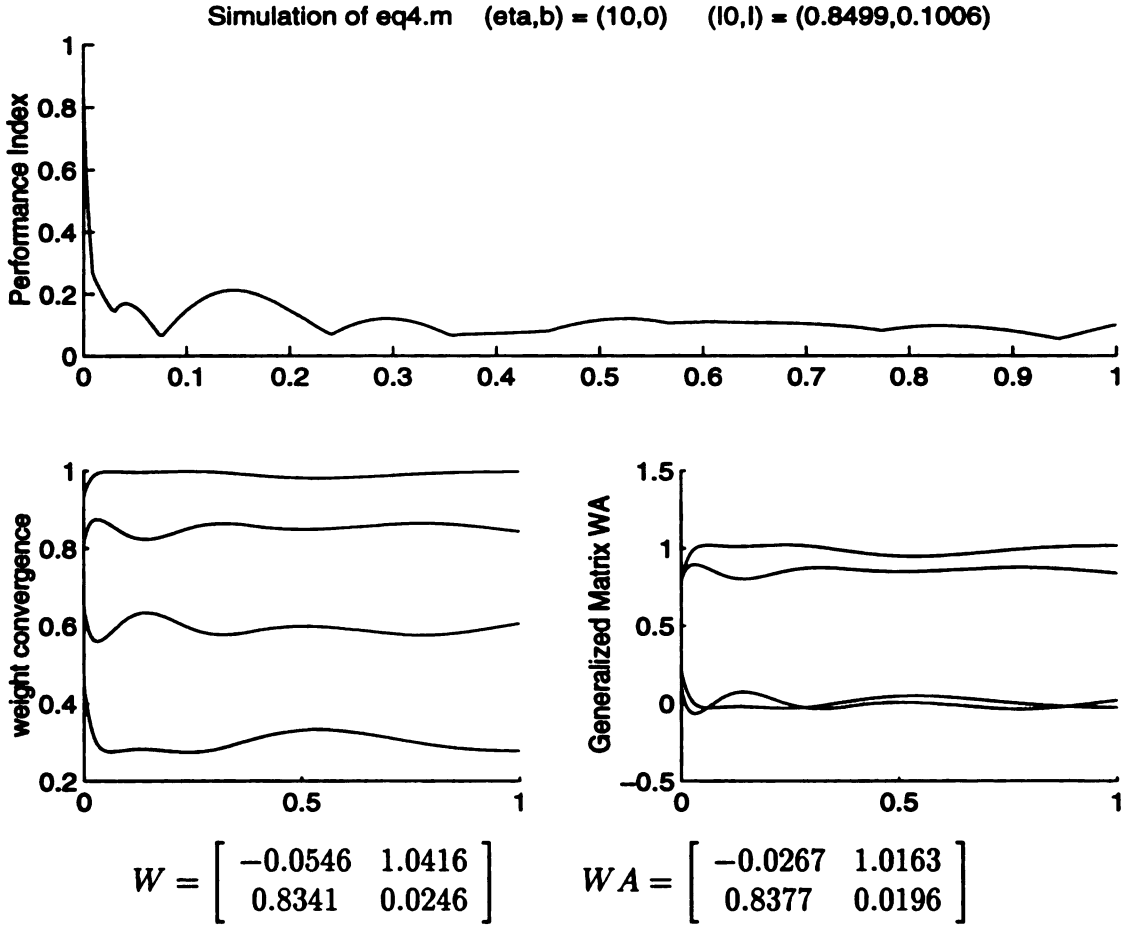
and variable learning rates. Figures 3.6 through 3.8 show the performance of algorithm (3.23) when the learning rate is $\eta(t) = 100,\ 10,\ 100e^{-5t}\ \ \forall t$

Observe that a learning rate η of 10 was too small to obtain any concluding results within the time span for training. When η is 100, one is able to observe that the network is converging to a generalized permutation matrix. However, it can be observed that the parameters exhibit some oscillations around a constant level. These oscillations were eliminated by considering a learning rate that decays with time. The appropriate function is $\eta(t) = 100e^{-5t}$. A set of 20 simulations of the algorithm with different initial conditions was performed. Table 3.2 represents the performance of the algorithm in all these simulations. The initial conditions were chosen randomly. The



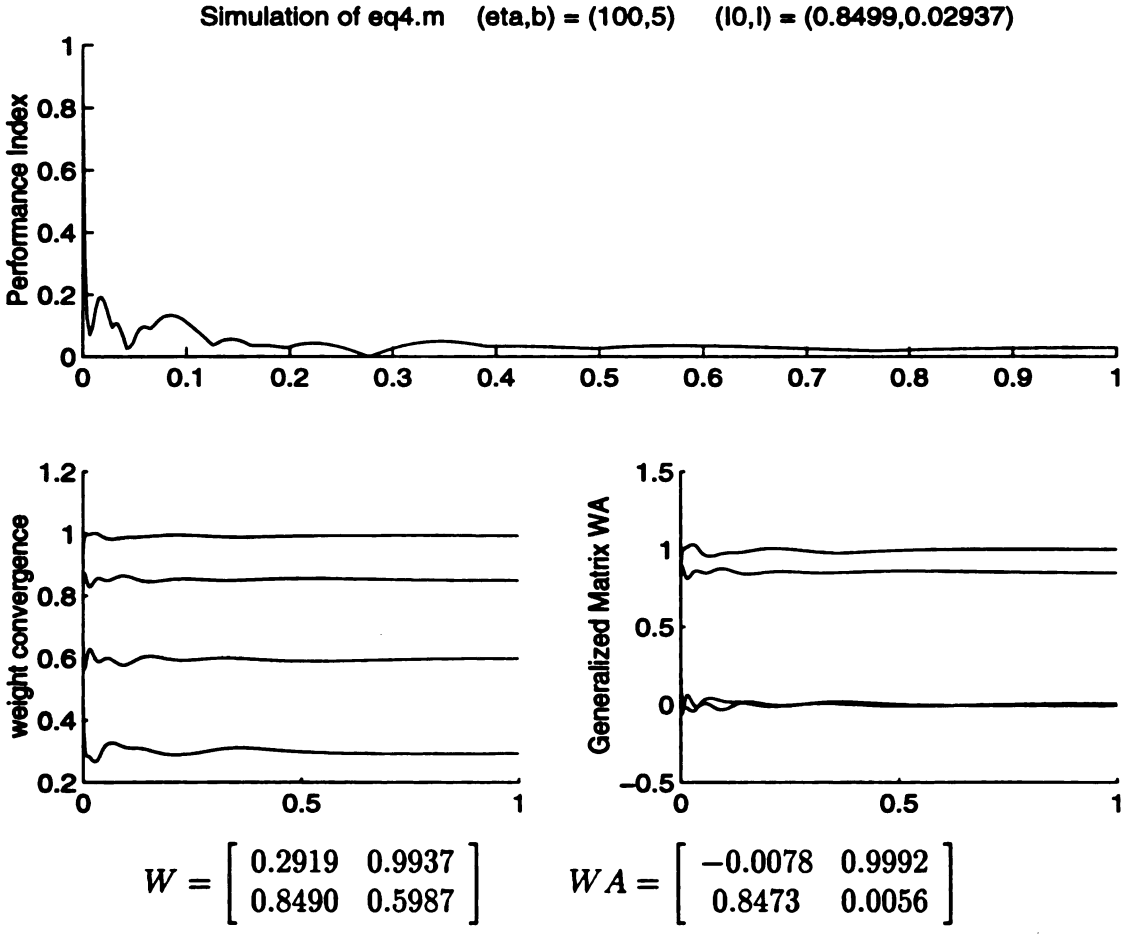
$$W = \begin{bmatrix} 0.2981 & 0.9919 \\ 0.8456 & 0.6035 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0267 & 1.0163 \\ 0.8377 & 0.0196 \end{bmatrix}$$

Figure 3.3. Performance of Algorithm defined by equation (3.23) for $\eta(t) = 100$. $f(x) = x^3$ and $g(x) = x$



$$W = \begin{bmatrix} -0.0546 & 1.0416 \\ 0.8341 & 0.0246 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0267 & 1.0163 \\ 0.8377 & 0.0196 \end{bmatrix}$$

Figure 3.4. Performance of Algorithm defined by equation (3.23) for $\eta(t) = 10$. $f(x) = x^3$ and $g(x) = x$

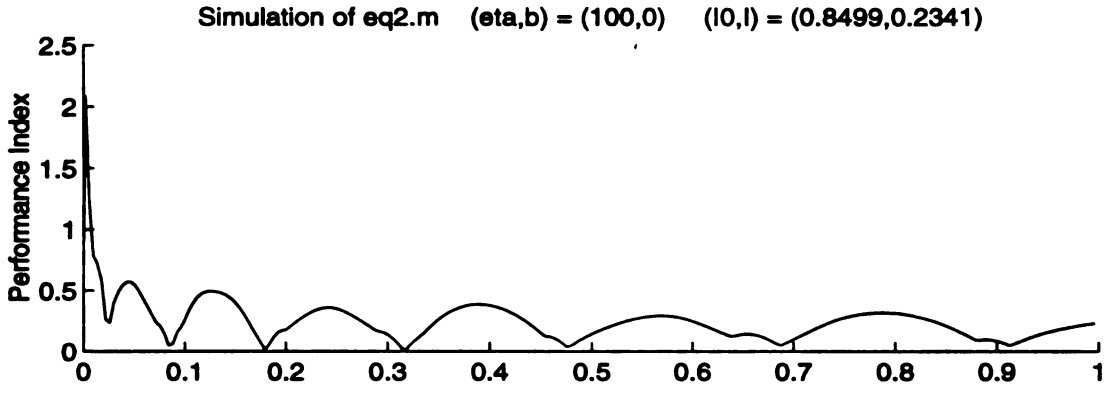


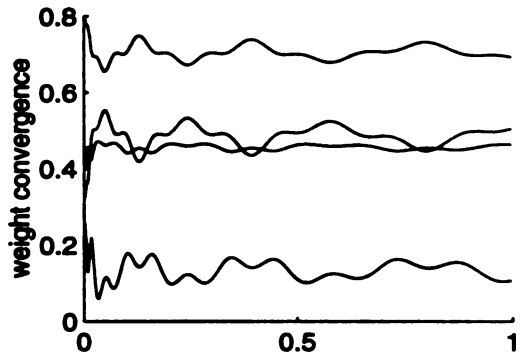
$$W = \begin{bmatrix} 0.2919 & 0.9937 \\ 0.8490 & 0.5987 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0078 & 0.9992 \\ 0.8473 & 0.0056 \end{bmatrix}$$

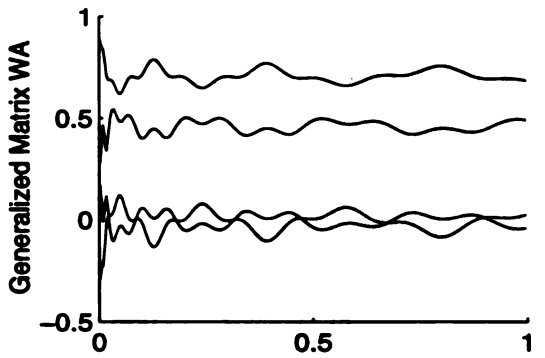
Figure 3.5. Performance of Algorithm defined by equation (3.23) for $\eta(t) = 100e^{-5t}$. $f(x) = x^3$ and $g(x) = x$

Table 3.1. Simulation Results Algorithm defined by equation (3.23). $f(x) = x^3$ and $g(x) = x$

Simulation #	Final Performance Index
1	0.1490
2	0.0125
3	0.0132
4	0.1692
5	0.0134
6	0.1036
7	0.0924
8	0.0157
9	0.0587
10	0.2035
11	0.0560
12	0.1302
13	0.0164
14	0.0266
15	0.7045
16	0.0377
17	0.0209
18	0.0244
19	0.0417
20	0.0526

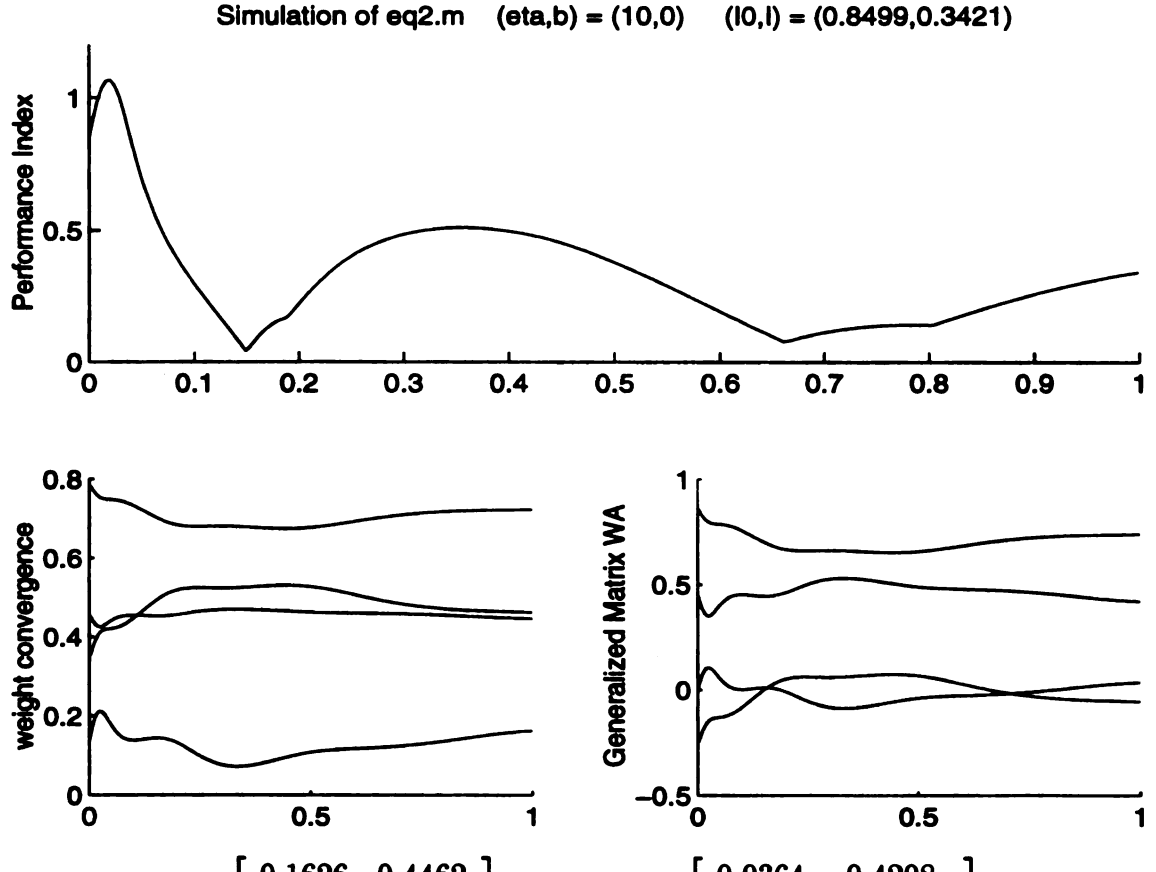






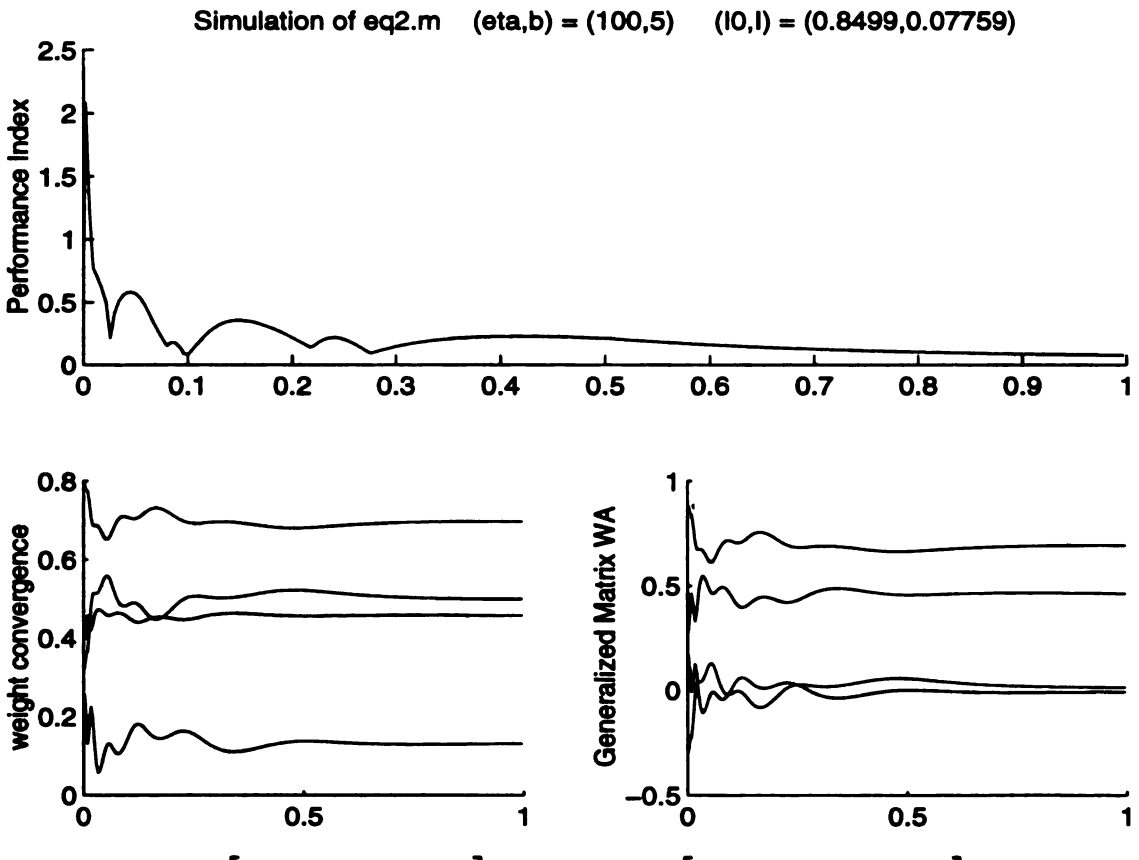
$$W = \begin{bmatrix} 0.1076 & 0.4627 \\ 0.6920 & 0.5060 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0395 & 0.4903 \\ 0.6838 & 0.0273 \end{bmatrix}$$

Figure 3.6. Performance of Algorithm defined by equation (3.23) for $\eta(t) = 100$. $f(x) = \sinh x$ and $g(x) = \tanh x$



$$W = \begin{bmatrix} 0.1626 & 0.4462 \\ 0.7229 & 0.4623 \end{bmatrix} \qquad WA = \begin{bmatrix} 0.0364 & 0.4208 \\ 0.7395 & -0.0554 \end{bmatrix}$$

Figure 3.7. Performance of Algorithm defined by equation (3.23) for $\eta(t) = 10$. $f(x) = \sinh x$ and $g(x) = \tanh x$



$$W = \begin{bmatrix} 0.1313 & 0.4567 \\ 0.6970 & 0.4991 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0073 & 0.4618 \\ 0.6927 & 0.0142 \end{bmatrix}$$

Figure 3.8. Performance of Algorithm defined by equation (3.23) for $\eta(t) = 100e^{-5t}$

Table 3.2. Simulation Results Algorithm defined by equation (3.23). $f(x) = \sinh x$ and $g(x) = \tanh x$

Simulation #	Final Performance Index
1	0.0066
2	0.1168
3	0.0517
4	0.1163
5	0.0543
6	0.0462
7	0.1836
8	0.2338
9	0.0062
10	0.0192
11	0.0084
12	0.0620
13	0.0347
14	0.0410
15	0.0038
16	0.2772
17	0.0165
18	0.0180
19	0.0510
20	0.0863

results show that the convergence was obtained regardless of the initial conditions.

3.7 Second Improvement

In this section, the focus will change to an implementation view point in order to modify the derived algorithm defined by (3.23). Recall that such equation is

$$\dot{W} = \eta\left[I - (diag < f(\mathbf{y})g(\mathbf{y})^T >_t)^{-1} < f(\mathbf{y})g(\mathbf{y})^T >_t\right] W^{-T}$$

This algorithm requires the computation of the inverse of the matrix W. This computation is complex and time consuming. It will be of great advantage if such a term could be eliminated from the update law. In [55], the author suggests that using the information geometry perspective, the update equation of W defined by (3.9) can become

$$\dot{W} = -\eta \frac{\partial \Phi}{\partial W} W^T W \tag{3.26}$$

since the matrix A is assumed to be nonsingular. Therefore, equation (3.23) can be modified to the following algorithm:

$$\dot{W} = \eta \Big[I - (diag < f(\mathbf{y}) g(\mathbf{y})^T >_t)^{-1} < f(\mathbf{y}) g(\mathbf{y})^T >_t \Big] W \tag{3.27}$$

3.7.1 Computer Simulations

By running (3.24) and

$$\dot{W} = \eta \Big[I - (diag\ Z)^{-1} Z \Big] W \tag{3.28}$$

simultaneously, the algorithm defined by (3.27) was implemented successfully. Computer simulations for the algorithm were performed using different types of nonlinearities, as was done for the previous algorithm.

Case 1: $f(x) = x^3$ **and** $g(x) = x$

Numerous simulations were performed using different initial conditions and learning rates. The following set of simulations was obtained by using the same initial conditions used to test the previous algorithm, in order to provide some comparison

between the two. The initial condition is

$$W_0 = \begin{bmatrix} 0.4523 & 0.9317 \\ 0.8089 & 0.6516 \end{bmatrix}$$

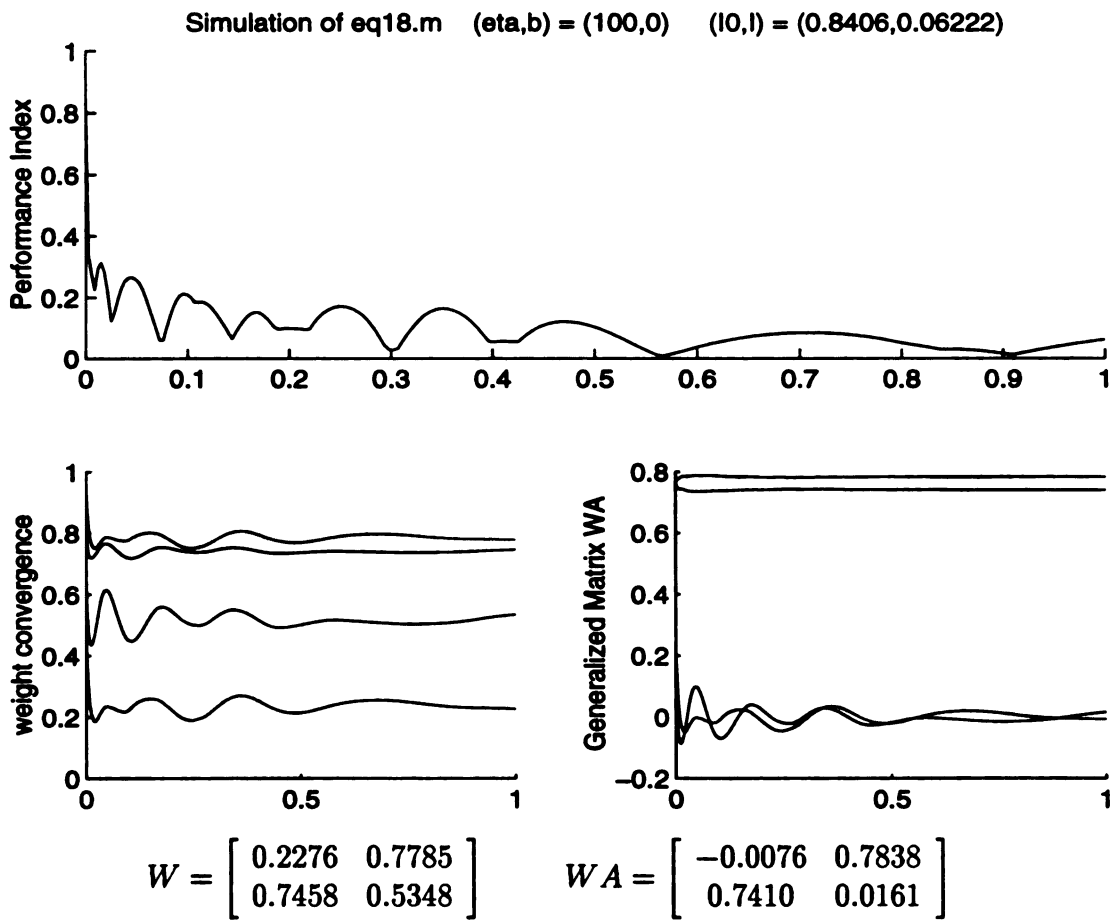
Figures 3.9 through 3.11 show the performance of algorithm (3.27) when the learning rates are $\eta(t) = 100,\ 10,\ 100e^{-5t}\ \ \forall t$, respectively. One can observe in Figure 3.9 that the network parameters oscillate around the the desired values when $\eta = 100$. However, by considering a time-varying learning rate $\eta = 100e^{-5t}$, such oscillation were eliminated and the parameters will settle to a constant value, as shown in Figure 3.11. Figure 3.10, however, demonstrates that a learning rate $\eta = 10$ is too small in order to obtain any concluding results.

Table 3.3 shows the results of twenty simulations of the algorithm defined by equation (3.27) with different initial conditions. It should be observed that 4 out of the 20 simulations failed to converge to a separating network since the corresponding performance indices are not near zero. Simulations that have a final performance index of more than $n/4$, in this case 0.5 since $n = 2$, are considered as failures. These simulations are labeled by an asterisk* in Table 3.3.

Case 2: $f(x) = \sinh x$ **and** $g(x) = \tanh x$

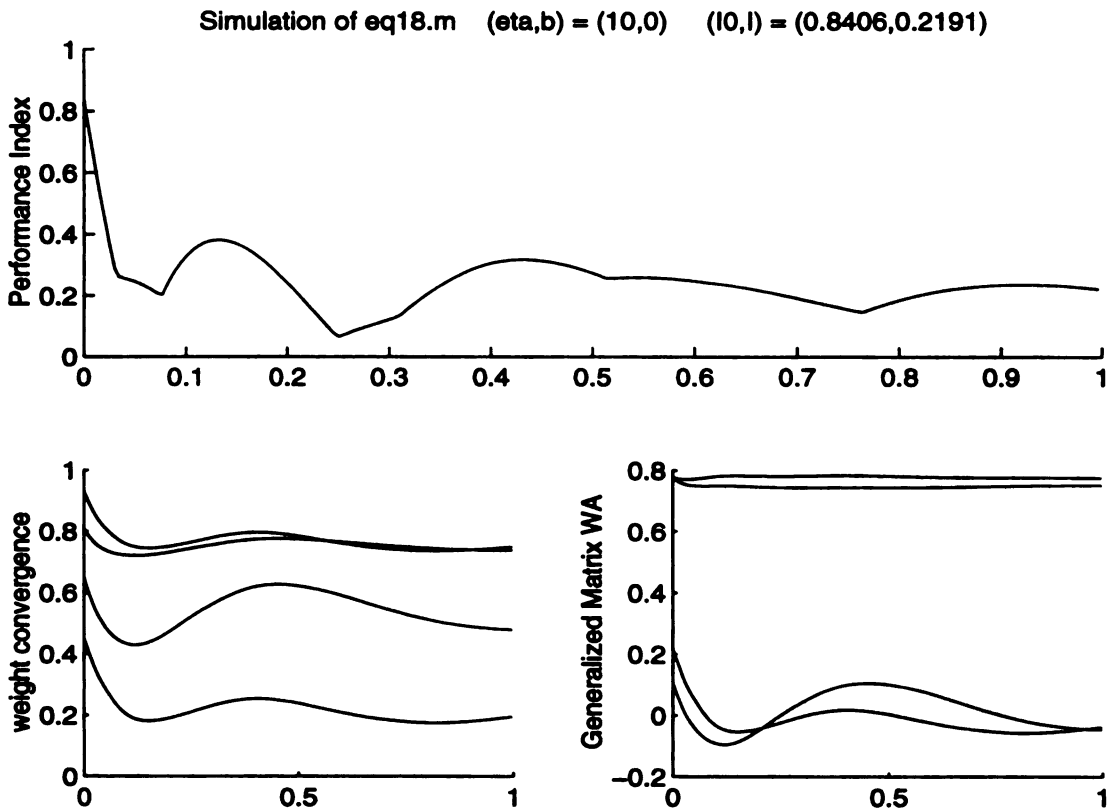
Numerous simulations were performed using different initial conditions and learning rates. The following set of simulations was obtained by using the same initial conditions used to test the previous algorithm in order to provide some comparison between the two. The initial weight is

$$W_0 = \begin{bmatrix} 0.0475 & 0.3282 \\ 0.7361 & 0.6326 \end{bmatrix}$$



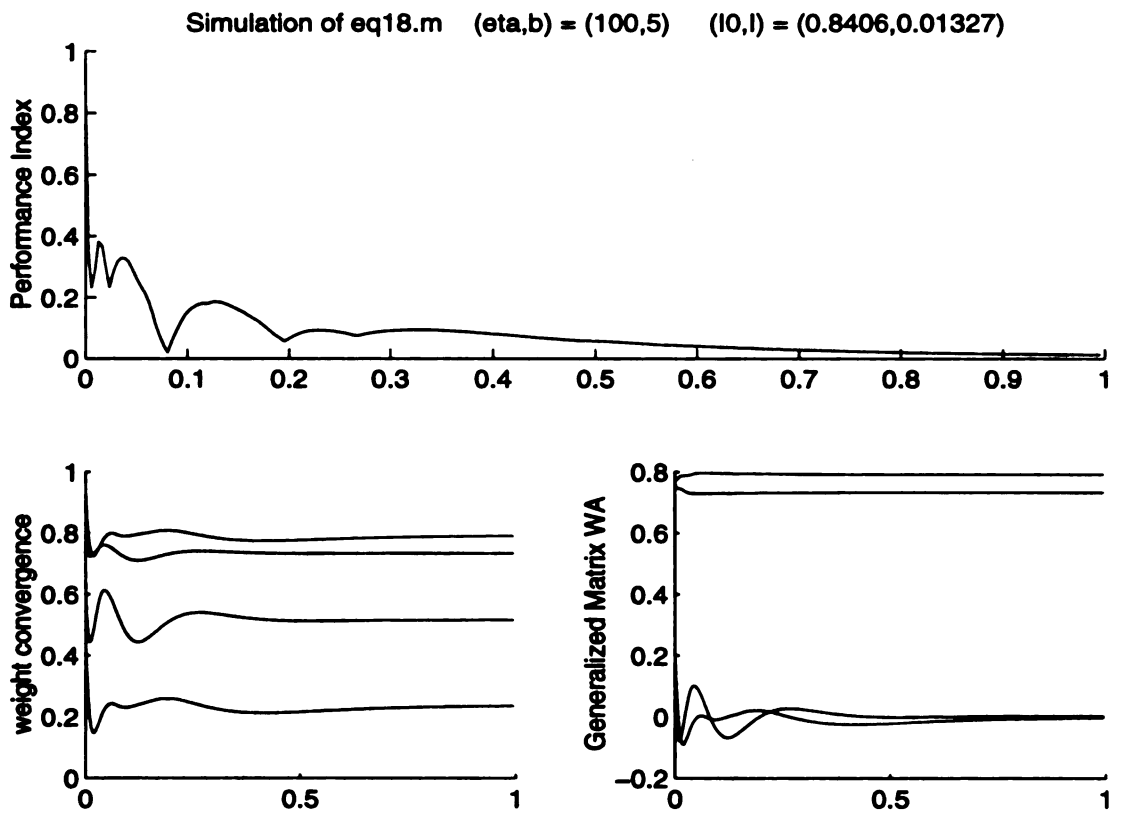
$$W = \begin{bmatrix} 0.2276 & 0.7785 \\ 0.7458 & 0.5348 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0076 & 0.7838 \\ 0.7410 & 0.0161 \end{bmatrix}$$

Figure 3.9. Performance of Algorithm defined by equation (3.27) for $\eta(t) = 100$. $f(x) = x^3$ and $g(x) = x$



$$W = \begin{bmatrix} 0.1960 & 0.7498 \\ 0.7373 & 0.4790 \end{bmatrix} WA = \begin{bmatrix} -0.0366 & 0.7754 \\ 0.7514 & -0.0470 \end{bmatrix}$$

Figure 3.10. Performance of Algorithm defined by equation (3.27) for $\eta(t) = 10$. $f(x) = x^3$ and $g(x) = x$



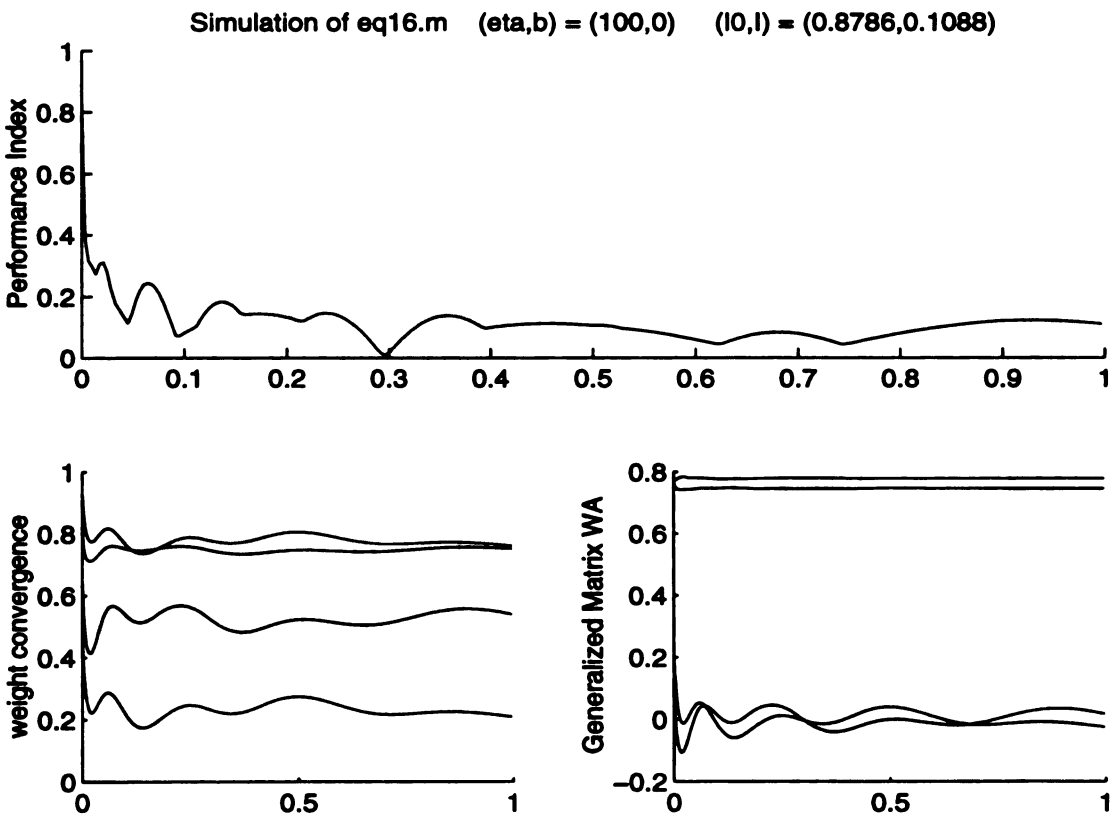
$$W = \begin{bmatrix} 0.2357 & 0.7910 \\ 0.7341 & 0.5163 \end{bmatrix} WA = \begin{bmatrix} -0.0020 & 0.7924 \\ 0.7332 & 0.0031 \end{bmatrix}$$

Figure 3.11. Performance of Algorithm defined by equation (3.27) for $\eta(t) = 10$. $f(x) = x^3$ and $g(x) = x$

Table 3.3. Simulation Results Algorithm defined by equation (3.27). $f(x) = x^3$ and $g(x) = x$

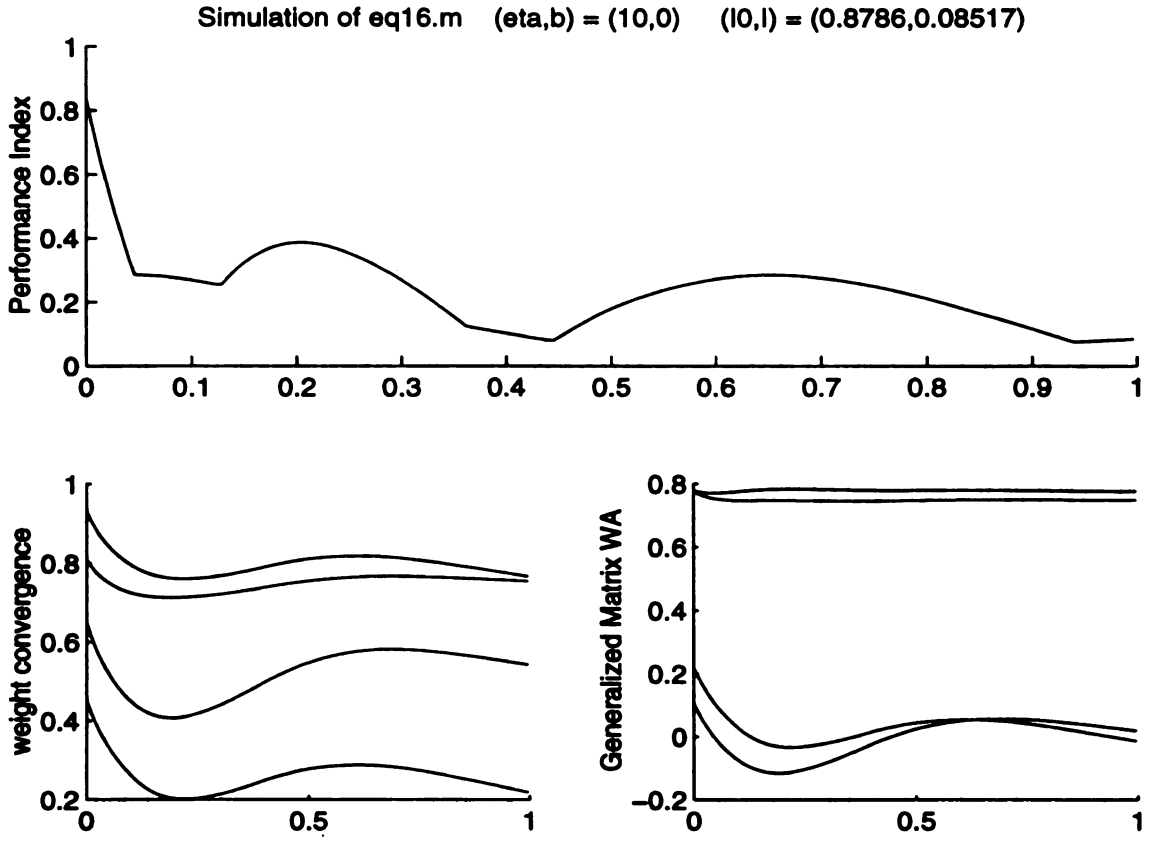
Simulation #	Final Performance Index
1	0.0211
2	0.0733
3	0.3574
4	1.5079*
5	0.0138
6	0.0298
7	0.1022
8	0.2058
9	1.0605*
10	0.0478
11	0.0578
12	0.0228
13	0.0544
14	1.3249*
15	1.4127*
16	0.0254
17	0.0332
18	0.0636
19	0.0415
20	0.0467

while the learning rates varied. Figures 3.12 through 3.14 show the performance of algorithm (3.27) when the learning rates are $\eta(t) = 100,\ 10,\ 100e^{-5t}\ \forall t$, respectively. It can also be noted that, in the set of simulations, a time-varying learning rate $\eta = 100e^{-5t}\ \forall t$, had to be introduced in order to eliminate oscillations, as observed in Figure 3.12, when the learning rate is of a constant value equal 100. Figure 3.14 shows the results for such a time-varying learning rate. On the other hand, when the $eta = 10$, such a learning is to too small to enable the network to arrive at the desired solutions.



$$W = \begin{bmatrix} 0.2085 & 0.7607 \\ 0.7508 & 0.5386 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0250 & 0.7782 \\ 0.7459 & 0.0165 \end{bmatrix}$$

Figure 3.12. Performance of Algorithm defined by equation (3.27) for $\eta(t) = 100$. $f(x) = \sinh x$ and $g(x) = \tanh x$

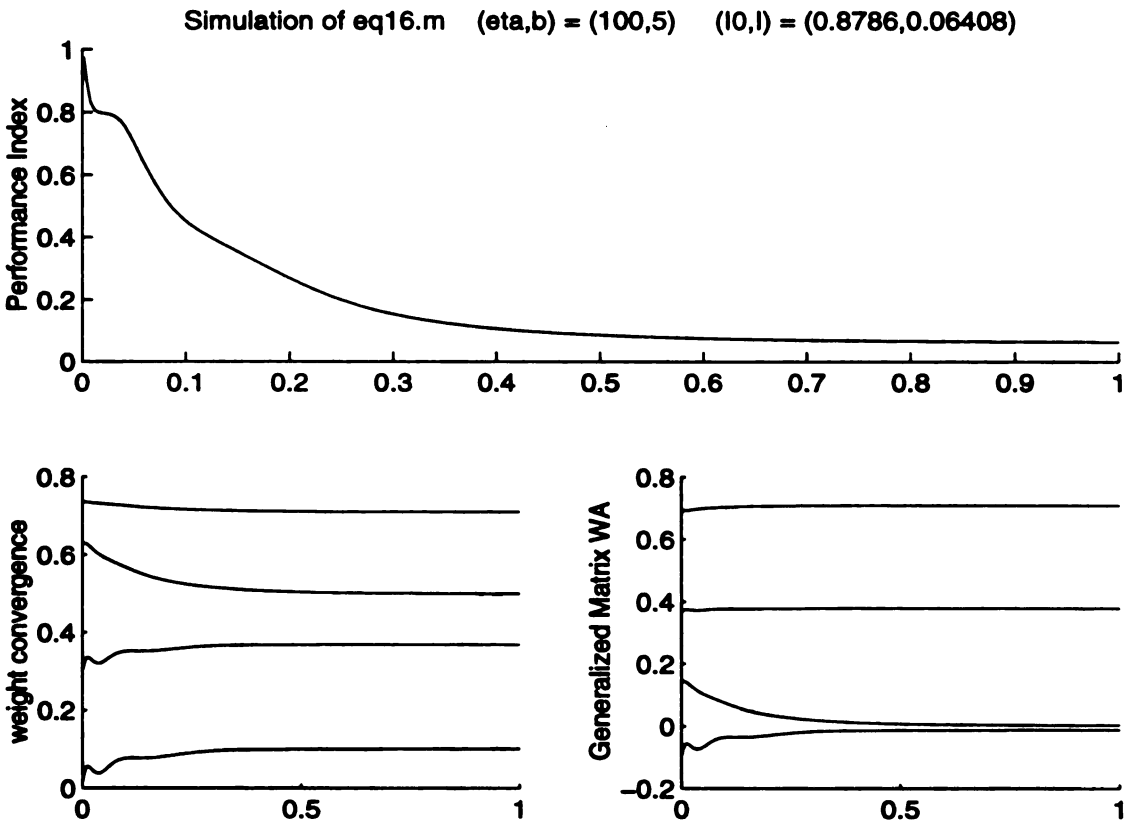


$$W = \begin{bmatrix} 0.2183 & 0.7657 \\ 0.7540 & 0.5420 \end{bmatrix} \qquad WA = \begin{bmatrix} -0.0144 & 0.7757 \\ 0.7485 & 0.0181 \end{bmatrix}$$

Figure 3.13. Performance of Algorithm defined by equation (3.27) for $\eta(t) = 10$. $f(x) = \sinh x$ and $g(x) = \tanh x$

Table 3.4 shows the results of twenty simulations of the algorithm defined by (3.27) with different initial conditions. It should be observed that four simulations out of these twenty did not succeed in separating the signals. They are marked with a star in Table 3.4.

Table 3.5 shows the results of computer simulations between algorithms defined by (3.23) and (3.27). One may conclude that both have similar performance.



$$W = \begin{bmatrix} 0.2402 & 0.7836 \\ 0.7457 & 0.5223 \end{bmatrix} \qquad WA = \begin{bmatrix} 0.0065 & 0.7791 \\ 0.7456 & 0.0004 \end{bmatrix}$$

Figure 3.14. Performance of Algorithm defined by equation (3.27) for $\eta(t) = 100e^{-5t}$. $f(x) = \sinh x$ and $g(x) = \tanh x$

Table 3.4. Simulation Results Algorithm defined by equation (3.27). $f(x) = \sinh x$ and $g(x) = \tanh x$

Simulation #	Final Performance Index
1	0.2664
2	0.2652
3	0.0806
4	0.8735*
5	0.0906
6	0.0832
7	0.1283
8	0.7715*
9	0.1956
10	0.2490
11	0.1415
12	0.0141
13	1.7091*
14	1.0756*
15	0.0776
16	0.3482
17	0.1142
18	0.0558
19	0.3650
20	0.0484

Table 3.5. Results of Simulation Comparison between Algorithms defined by equations (3.23) and (3.27)

Initial Conditions W_0	Final Results Of (3.23) WA	Final Results Of (3.27) WA
$\begin{bmatrix} 0.9347 & 0.5194 \\ 0.3835 & 0.8310 \end{bmatrix}$	$\begin{bmatrix} 0.8805 & -0.0079 \\ 0.0082 & 0.8692 \end{bmatrix}$ $I_f = 0.0366$	$\begin{bmatrix} 0.9906 & 0.0055 \\ -0.0127 & 0.7380 \end{bmatrix}$ $I_f = 0.0433$
$\begin{bmatrix} 0.0475 & 0.3282 \\ 0.7361 & 0.6326 \end{bmatrix}$	$\begin{bmatrix} -0.0054 & 0.3223 \\ 0.8063 & -0.0169 \end{bmatrix}$ $I_f = 0.0966$	$\begin{bmatrix} 0.0229 & 0.3824 \\ 0.7008 & 0.0089 \end{bmatrix}$ $I_f = 0.1283$
$\begin{bmatrix} 0.1351 & 0.4553 \\ 0.7832 & 0.3495 \end{bmatrix}$	$\begin{bmatrix} -0.0091 & 0.4634 \\ 0.6913 & 0.0163 \end{bmatrix}$ $I_f = 0.0916$	$\begin{bmatrix} 0.0069 & 0.4423 \\ 0.8857 & 0.0091 \end{bmatrix}$ $I_f = 0.0545$
$\begin{bmatrix} 0.1351 & 0.4553 \\ 0.7832 & 0.3495 \end{bmatrix}$	$\begin{bmatrix} -0.0091 & 0.4634 \\ 0.6913 & 0.0163 \end{bmatrix}$ $I_f = 0.0916$	$\begin{bmatrix} 0.0069 & 0.4423 \\ 0.8857 & 0.0091 \end{bmatrix}$ $I_f = 0.0545$

3.8 Observation and Remarks

In this chapter, an algorithm based on the decorrelation condition between the components of output of the signal vector was developed. The resulting algorithm was improved in order to test for the independence between the components of the output vector. It was shown that the algorithm defined by the first modification always converged to a separating network regardless of the initial conditions. The algorithm of the second modification is more attractive since it eliminates the computation of the inverse of a matrix. However, such elimination resulted in an algorithm that has a 20% chance of failure rate.

CHAPTER 4

Higher Order Statistics

It was proved in the literature [8, 23, 61] that second order statistics are not sufficient to solve the problem of the blind separation of sources. One will also recall that the static algorithm, developed in Chapter 3, was initially based on the second order statistics, since the correlation defined such characteristics as the statistical description of a signal. Some techniques widely used in the literature were used to improve the algorithm performance to test for higher order statistics by generating infinitely many orders of moments through the injection of odd nonlinear functions f and g in the algorithm. It will be shown how these functions can be determined by defining an independence criterion that defines the problem. It was proved in the literature, that **cumulants** up to the fourth order are sufficient to approximate the probability density functions of a random variable using the fourth order **Edgeworth approximation** [62]. Consequently, higher order statistics have enabled several researchers to analyze the problem to some extent, Amari et al [24, 63, 64, 65], Comon et al [23, 66, 22], Tong et al [29, 28]. However, one should keep in mind that these algorithms were developed for a static environment or discrete dynamic Finite Impulse Response (FIR) models. However, none of these approaches addressed a general dynamical environment as it will be defined in this work. Such a dynamic system has memory represented by its dynamical states. Through the framework of opti-

mal control theory, or the calculus of variations, and through the use of higher order statistics , this research will focus on developing new algorithms. In this chapter, an independence criterion will be developed. It will be used to derive an update law for the parameters of a static feedforward and feedback network models. The developed independence criterion will be also used to derive update laws for the parameters of a dynamic feedforward and feedback network models the subsequent two chapters.

To introduce higher order statistics, one must first give the definition of statistical independence.

Definition 3 (Statistical Independence) *The components of a random signal vector* $\mathbf{y}$ *are statistically independent if and only if their joint probability density (pdf)* $p_{\mathbf{y}}(\mathbf{y})$ *is the product of all individual marginal probability density functions* $p_{y_i}(y_i)$. *Symbolically,*

$$p_{\mathbf{y}}(\mathbf{y}) = \prod_i p_{y_i}(y_i) \tag{4.1}$$

4.1 Mutual information

One way to measure the independence of the components of a random vector is to measure the distance between the right and the left hand sides of equation (4.1). When such a distance is zero, then the random vector's components are statistically independent. A well known distance in the literature is the Kullback divergence functional [58]

$$I(\mathbf{y}) = \int f_{\mathbf{y}}(\mathbf{u}) \ln \frac{f_{\mathbf{y}}(\mathbf{u})}{\prod_i f_{y_i}(u_i)} d\mathbf{u} \tag{4.2}$$

where $f_{\mathbf{y}}(\mathbf{y})$ is the pdf of a random vector $\mathbf{y}$. The functional $I(\mathbf{y})$ is always positive and is zero if the components of the random vector $\mathbf{y}$ are statistically independent. It defines the level of dependence between the components of the signal. Therefore,

it represents a good functional for characterizing statistical independence. $I(\mathbf{y})$ can be expressed in terms of entropy as defined in Appendix A.

$$I(\mathbf{y}) = -H(\mathbf{y}) + \sum_i H(y_i) \tag{4.3}$$

Some properties of entropy are presented in Appendix A.

Consider the vector $\mathbf{y} \in \mathcal{R}^m$ to be the output of a system described by the linear mapping $W = [W_1\ W_2]$ due to the input signal vector $\mathbf{x} \in \mathcal{R}^n$ as shown in Figure 4.1.

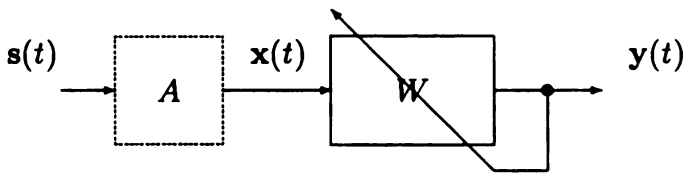


Figure 4.1. Feedforward Architecture

Define the vector

$$\mathbf{z} = [y_1 \cdots y_m, x_{m+1} \cdots x_n] \tag{4.4}$$

Then,

$$\mathbf{z} = \tilde{W}\mathbf{x} \tag{4.5}$$

where

$$\tilde{W} = \begin{bmatrix} W_1 & W_2 \\ 0 & I_{n-m} \end{bmatrix} \tag{4.6}$$

$\tilde{W}$ is a nonsingular square matrix with the assumption that W_1 is also nonsingular.

Using equation (4.5), a relationship between the probability density functions of the random vectors $\mathbf{z}$ and $\mathbf{x}$ can be obtained

$$f_{\mathbf{z}} = \frac{f_{\mathbf{x}}}{|\tilde{W}|} \tag{4.7}$$

Since $\mathbf{z} = [y_1 \cdots y_m, x_{m+1} \cdots x_n]$, one can express the pdf of $\mathbf{y}$ in terms of that of $\tilde{\mathbf{x}} = [x_1 \cdots x_m]$

$$\begin{aligned} f_{\mathbf{y}} &= \int_{-\infty}^{\infty} f_{\mathbf{z}}(\mathbf{z}) dx_{m+1} \cdots dx_n & (4.8) \\ &= \frac{\int_{-\infty}^{\infty} f_{\mathbf{x}}(\mathbf{x}) dx_{m+1} \cdots dx_n}{|\tilde{W}|} & (4.9) \\ &= \frac{f_{\tilde{\mathbf{x}}}}{|\tilde{W}|} & (4.10) \end{aligned}$$

Finally, using the entropy definition and the equation above

$$H(\mathbf{y}) = -E[\ln f_{\mathbf{y}}] = \ln |\tilde{W}| + H(\tilde{\mathbf{x}})$$

Therefore, equation (4.3) becomes

$$I(\mathbf{y}) = -H(\tilde{\mathbf{x}}) - \ln |\tilde{W}| + \sum_i H(y_i) \tag{4.11}$$

In equation (4.11), one may compute the first two terms of the expression. However, the summation term is unknown since we have no knowledge of the probality densities f_{y_i}.

$$H(y_i) = -\int f_{y_i} \ln f_{y_i} dy_i \tag{4.12}$$

Thus, one must approximate the probability density f_{y_i} and also its logarithmic $\ln f_{y_i}$ in order to obtain some approximate expression of the entropy as defined by equation (4.12). To compute such an approximation, one would first need to introduce moments

and cumulants. Then, an Edgeworth expansion will be used to approximate the probality density f_{y_i} and also its logarithmic $\ln f_{y_i}$.

4.1.1 Cumulants and Moments

Given a random variable x, the n^{th} order moment is defined as

$$\mu_n = E[x^n]$$

Moments are also often defined as the formal power series expansion of the moment generation function defined as

$$M(\beta) = E[e^{\beta x}] = E\Big[\sum_k \frac{(\beta x)^k}{k!}\Big] = \sum_k \frac{\beta^k}{k!} E[x^k] = \sum_k \mu_k \frac{\beta^k}{k!}$$

The cumulant generation function is defined as

$$K(\beta) = \ln M(\beta) = \ln E[e^{\beta x}] = \sum_k \kappa_k \frac{\beta^k}{k!}$$

where κ_k is the cumulant of the random variable x. One can express the moments in terms of the cumulants, and vice versa, by solving the equation

$$\sum_i \kappa_i \frac{\beta^i}{i!} = \ln \sum_i \mu_i \frac{\beta^i}{i!}$$

Table 4.1 shows the relationship between them for the first few orders of moments and cumulants.

Cumulants proved to be computationally efficient compared to moments despite the fact that the two quantities are equivalent. Some of the most important features of cumulants , for statistically independent signals [67], are

- the cumulant of the sum is the sum of the cumulants,

Table 4.1. Conversion of moments and cumulants

Cumulants in terms of moments	Moments in terms of cumulants
$\kappa_1 = \mu_1$	$\mu_1 = \kappa_1$
$\kappa_2 = \mu_2 - \mu_1^2$	$\mu_2 = \kappa_2 + \kappa_1^2$
$\kappa_3 = \mu_3 - 3\mu_1\mu_2 + 2\mu_1^3$	$\mu_3 = \kappa_3 + 3\kappa_1\kappa_2 + \kappa_1^3$
$\kappa_4 = \mu_4 - 4\mu_1\mu_3 - 3\mu_2^2 + 12\mu_2\mu_1^2 - 6\mu_1^4$	$\mu_4 = \kappa_4 + 4\kappa_1\kappa_3 + 3\kappa_2^2 + 6\kappa_2\kappa_1^2 + \kappa_1^4$

- the cross cumulants are zero,
- the Edgeworth expansion is most conveniently expressed in terms of cumulants, and
- most pdfs of practical signals can be approximated by a finite number of cumulants.

4.1.2 Edgeworth Expansion

The Edgeworth expansion of a given distribution density function $f(x)$ is formally defined as a density function having cumulants κ_i that are constructed from a modification of a baseline density function $f_0(x)$ having cumulants ν_i [67],

$$f(x) = f_0(x) \sum_{k=0}^{\infty} h_k(x) \frac{\mu_k^*}{k!} \tag{4.13}$$

where $h_k(x)$ defines a family of orthogonal functions known as the Hermite polynomial functions as seen below

$$h_k(x) = (-1)^k \frac{f_0^{(k)}(x)}{f_0(x)} \tag{4.14}$$

Here the μ_k^*'s are the pseudo-moment that satisfy the equation

$$\sum_{k=0}^{\infty}(\kappa_i - \nu_i)\frac{\beta^k}{k!} = \ln\sum_{k=0}^{\infty}\mu_k^*\frac{\beta^k}{k!} \tag{4.15}$$

The choice of the baseline depends on the the statistical properties of the variable that is to be approximated. However, if no such information is available, it is most convenient to use the normal density function

$$f_0(x) = \sigma(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} \tag{4.16}$$

as a baseline. When $f_0(x) = \sigma(x)$ and the equation (4.13) is truncated at some finite number of terms, the resulting density function approximation is called a Gram-Charlier series [68]. Expansions based on other than normal distributions are rare. There are, however, some that are based on the χ^2 distribution density function [62].

Normal Baseline Expansion

It is intended here to determine an expression for the Edgeworth expansion when the baseline is the normal distribution $\sigma(x)$ as defined by (4.16). The cumulants of $\sigma(x)$ can be extracted from its cumulant generating function

$$\ln E[\exp\beta x] = \frac{\beta^2}{2} = \sum_k \nu_k\frac{\beta^k}{k!}$$

Therefore, all the cumulants of $\sigma(x)$ are zero except $\nu_2 = 1$. Thus, in this case, the first two pseudo-moments μ_k^* are zero and the rest are the unmodified corresponding cumulants of the distribution density function $f(x)$, namely κ_k. Consequently, the Edgeworth expansion of a density function with respect to a normal baseline is

$$f(x) = \sigma(x)\Big[1 + \frac{\kappa_3}{3!}h_3(x) + \frac{\kappa_4}{4!}h_4(x) + \cdots\Big] \tag{4.17}$$

Table 4.2. Hermite Polynomials derived from a normal baseline density function $\sigma(x)$

$h_1(x)$	=	x
$h_2(x)$	=	$x^2 - 1$
$h_3(x)$	=	$x^3 - 3x$
$h_4(x)$	=	$x^4 - 6x^2 + 3$

Equation (4.14) defines the Hermite polynomials. In the case of a normal baseline, they are easy to compute. Table 4.2 presents few orders of these polynomials.

4.1.3 Entropy Approximation

The tools to approximate the entropies are now developed. For ease of notation, a random variable x having a density function $f(x)$ will be considered. Its fourth order Gram-Charlier approximation will be as follows:

$$f(x) \approx \sigma(x)\left[1 + \frac{\kappa_3}{3!}h_3(x) + \frac{\kappa_4}{4!}h_4(x)\right] = \sigma(x)p(x) \tag{4.18}$$

Therefore,

$$\begin{aligned} H(x) &= -\int f(x)\ln f(x)\,dx \\ &\approx -\int f(x)\ln \sigma(x)\,dx - \int \sigma(x)p(x)\ln p(x)\,dx \end{aligned} \tag{4.19}$$

However,

$$-\ln\sigma(x) = -\ln\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} = \frac{1}{2}\ln 2\pi + \frac{x^2}{2} \tag{4.20}$$

Therefore,

$$\begin{aligned} -\int f(x)\ln\sigma(x)\,dx &= \frac{1}{2}\ln 2\pi\int f(x)\,dx+\frac{1}{2}\int x^2 f(x)\,dx \\ &= \frac{1}{2}\ln 2\pi+\frac{\mu_2}{2} \end{aligned}$$

In order to compute the second term in equation (4.19), one must approximate the logarithmic function as

$$\ln(1+x)=x-\frac{x^2}{2}+\frac{x^3}{3}+\mathcal{O}(x^4) \tag{4.21}$$

Therefore,

$$\begin{aligned} \ln p(x) &= [p(x)-1]-\frac{1}{2}[p(x)-1]^2+\frac{1}{3}[p(x)-1]^3 \\ &= \frac{\kappa_3}{3!}h_3+\frac{\kappa_4}{4!}h_4-\frac{1}{2}\Big[\frac{\kappa_3}{3!}h_3+\frac{\kappa_4}{4!}h_4\Big]^2 \\ &\quad+\frac{1}{3}\Big[\frac{\kappa_3}{3!}h_3+\frac{\kappa_4}{4!}h_4\Big]^3 \\ &= \frac{\kappa_3}{3!}h_3+\frac{\kappa_4}{4!}h_4-\frac{\kappa_3^2}{2\cdot 3!^2}h_3^2-\frac{\kappa_4^2}{2\cdot 4!^2}h_4^2-\frac{\kappa_3\kappa_4}{3!4!}h_3h_4 \\ &\quad+\frac{\kappa_3^3}{3\cdot 3!^3}h_3^3+\frac{\kappa_4^3}{3\cdot 4!^3}h_4^3+\frac{\kappa_3^2\kappa_4}{3!^2 4!}h_3^2h_4+\frac{\kappa_3\kappa_4^2}{3!4!^2}h_3h_4^2 \end{aligned}$$

Consequently,

$$\begin{aligned} p(x)\ln p(x) &= \ln p(x)+\frac{\kappa_3}{3!}h_3\ln p(x)+\frac{\kappa_4}{4!}h_4\ln p(x) \\ &= \frac{\kappa_3}{3!}h_3+\frac{\kappa_4}{4!}h_4-\frac{\kappa_3^2}{2\cdot 3!^2}h_3^2-\frac{\kappa_4^2}{2\cdot 4!^2}h_4^2-\frac{\kappa_3\kappa_4}{3!4!}h_3h_4 \\ &\quad+\frac{\kappa_3^3}{3\cdot 3!^3}h_3^3+\frac{\kappa_4^3}{3\cdot 4!^3}h_4^3+\frac{\kappa_3^2\kappa_4}{3!^2 4!}h_3^2h_4+\frac{\kappa_3\kappa_4^2}{3!4!^2}h_3h_4^2 \\ &\quad+\frac{\kappa_3}{3!}\Big[\frac{\kappa_3}{3!}h_3^2+\frac{\kappa_4}{4!}h_3h_4-\frac{\kappa_3^2}{2\cdot 3!^2}h_3^3-\frac{\kappa_4^2}{2\cdot 4!^2}h_3h_4^2-\frac{\kappa_3\kappa_4}{3!4!}h_3^2h_4 \\ &\quad+\frac{\kappa_3^3}{3\cdot 3!^3}h_3^4+\frac{\kappa_4^3}{3\cdot 4!^3}h_3h_4^3+\frac{\kappa_3^2\kappa_4}{3!^2 4!}h_3^3h_4+\frac{\kappa_3\kappa_4^2}{3!4!^2}h_3^2h_4^2\Big] \\ &\quad+\frac{\kappa_4}{4!}\Big[\frac{\kappa_3}{3!}h_3h_4+\frac{\kappa_4}{4!}h_4^2-\frac{\kappa_3^2}{2\cdot 3!^2}h_3^2h_4-\frac{\kappa_4^2}{2\cdot 4!^2}h_4^3-\frac{\kappa_3\kappa_4}{3!4!}h_3h_4^2 \\ &\quad+\frac{\kappa_3^3}{3\cdot 3!^3}h_3^3h_4+\frac{\kappa_4^3}{3\cdot 4!^3}h_4^4+\frac{\kappa_3^2\kappa_4}{3!^2 4!}h_3^2h_4^2+\frac{\kappa_3\kappa_4^2}{3!4!^2}h_3h_4^3\Big] \end{aligned}$$

In order to complete the integral in equation (4.19), one uses the following properties of the Hermite polynomials

$$\int h_i(x)h_j(x)\sigma(x)\,dx = i!\,\delta_{ij} \tag{4.22}$$

The following formulae for the moments of the normal density function $\sigma(x)$ are also needed

$$\mu_{2k} = \int x^{2k}\sigma(x)\,dx = 1\cdot 3\cdots(2k-1) \tag{4.23}$$

$$\mu_{2k+1} = \int x^{2k+1}\sigma(x)\,dx = 0 \tag{4.24}$$

Consequently,

$$\int h_4(x)\sigma(x)\,dx = \mu_4 - 6\mu_2 + 3 = 0$$

$$\int h_3^2(x)\sigma(x)\,dx = \mu_6 - 6\mu_4 + 9\mu_2 = 6$$

$$\int h_4^2(x)\sigma(x)\,dx = \mu_8 - 12\mu_6 + 42\mu_4 - 36\mu_2 + 9 = 24$$

$$\int h_3^2(x)h_4(x)\sigma(x)\,dx = \mu_{10} - 12\mu_8 + 48\mu_6 - 72\mu_4 + 27\mu_2 = 216$$

$$\int h_4^3(x)\sigma(x)\,dx = \mu_{12} - 18\mu_{10} + 117\mu_8 - 324\mu_6 + 351\mu_4 - 162\mu_2 + 27 = 1728$$

$$\int h_3^4(x)\sigma(x)\,dx = \mu_{12} - 12\mu_{10} + 54\mu_8 - 108\mu_6 + 81\mu_4 = 3348$$

$$\int h_3^2(x)h_4^2(x)\sigma(x)\,dx = \mu_{14} - 18\mu_{12} + 123\mu_{10} - 396\mu_8$$

$$+603\mu_6 - 378\mu_4 + 81\mu_2 = 30672$$

$$\begin{aligned}\int h_4^4(x)\sigma(x)\,dx &= \mu_{16} - 24\mu_{14} + 228\mu_{12} - 1080\mu_{10} \\ &\quad +2646\mu_8 - 3240\mu_6 + 2052\mu_4 - 648\mu_2 + 81 = 368064\end{aligned}$$

Consequently,

$$\begin{aligned}\int \sigma(x)p(x)\ln p(x)\,dx &= \frac{1}{12}\kappa_3^2 + \frac{1}{48}\kappa_4^2 - \frac{1}{48}\kappa_3^2\kappa_4 - \frac{1}{48}\kappa_4^3 + \frac{71}{24}\kappa_3^2\kappa_4^2 \\ &\quad +\frac{31}{36}\kappa_3^4 + \frac{71}{192}\kappa_4^4\end{aligned}$$

Finally,

$$\begin{aligned}H(x) &\approx \frac{1}{2}\ln 2\pi + \frac{1}{2}\kappa_2 - \frac{1}{12}\kappa_3^2 - \frac{1}{48}\kappa_4^2 + \frac{1}{48}\kappa_3^2\kappa_4 + \frac{1}{48}\kappa_4^3 - \frac{71}{24}\kappa_3^2\kappa_4^2 \qquad (4.25)\\ &\quad -\frac{31}{36}\kappa_3^4 - \frac{71}{192}\kappa_4^4 \qquad (4.26)\\ &= \mathcal{H}(\kappa_2, \kappa_3, \kappa_4) \qquad (4.27)\end{aligned}$$

4.2 Independence Criteria

Mutual information is a good measure of the statistical independence of the components of a random vector. Recall that a better expression for it as described by equation (4.11) has been developed. In the previous two sections, it was explained how the marginal entropies are approximated by

$$H(y_i) = \mathcal{H}(\kappa_{2i}, \kappa_{3i}, \kappa_{4i}) \qquad (4.28)$$

where κ_{ki} and k^{th} order cumulants of the random variable y_i. Thus, the mutual information functional defined by equation (4.11) can be approximated by

$$I(\mathbf{y}) \approx -H(\tilde{\mathbf{x}}) - \ln |\tilde{W}| + \sum_i \mathcal{H}_i \tag{4.29}$$

Note, however, that the term $-H(\tilde{\mathbf{x}})$ is not a function of $\mathbf{y}$, nor it is a function of $\tilde{W}$. Therefore such a term can be eliminated from the independence criterion. Thus, the independence criterion is

$$\phi(\mathbf{y}) = -\ln |\tilde{W}| + \sum_i \mathcal{H}_i \tag{4.30}$$

4.3 Static Case: Feedforward Network Structure

Consider the static environment case defined defined by

$$\mathbf{y} = W\mathbf{x} \tag{4.31}$$

The derivation will be made for the case when the number of outputs is equal to the number of sensors. This implies that W is a square matrix. Since the environment matrix is assumed to be nonsingular, so is W. Also, because W is a square matrix, one has $\tilde{W} = W$. The parameters of the network will be updated according to the gradient descent method along the energy function defined by (4.30). Thus, the update law is

$$\dot{W} = -\eta \frac{\partial \phi}{\partial W} \tag{4.32}$$

where η is the learning rate.

$$\frac{\partial \phi}{\partial W} = -\frac{\partial \ln |W|}{\partial W} + \sum_m \frac{\partial \mathcal{H}_m}{\partial W} \tag{4.33}$$

$$= -W^{-T} + \sum_m \frac{\partial \mathcal{H}_m}{\partial W} \tag{4.34}$$

However,

$$\frac{\partial \phi}{\partial w_{ij}} = -\left[W^{-T}\right]_{ij} + \sum_{lm} \frac{\partial \mathcal{H}_m}{\partial \kappa_{lm}} \frac{\partial \kappa_{lm}}{\partial w_{ij}} \tag{4.35}$$

$$= -\left[W^{-T}\right]_{ij} + \sum_{lm} \frac{\partial \mathcal{H}_m}{\partial \kappa_{lm}} \frac{\partial \kappa_{lm}}{\partial y_m} \frac{\partial y_m}{\partial w_{ij}} \tag{4.36}$$

and

$$\frac{y_m}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_r w_{mr} x_r \tag{4.37}$$

$$= \sum_r \delta_{mi} \delta_{jr} x_r \tag{4.38}$$

$$= \delta_{mi} x_j \tag{4.39}$$

Therefore,

$$\frac{\partial \phi}{\partial w_{ij}} = -\left[W^{-T}\right]_{ij} + \sum_{lm} \frac{\partial \mathcal{H}_m}{\partial \kappa_{lm}} \frac{\partial \kappa_{lm}}{\partial y_m} \delta_{mi} x_j \tag{4.40}$$

$$= -\left[W^{-T}\right]_{ij} + \sum_{l} \frac{\partial \mathcal{H}_i}{\kappa_{li}} \frac{\partial \kappa_{li}}{\partial y_i} x_j \tag{4.41}$$

The computation of the second term may be cumbersome. What one needs to do is to first compute $\frac{\partial \mathcal{H}_i}{\kappa_{li}}$ in terms of cumulants, use the convolution of polynomials to compute the product of the cumulants by expressing the cumulants in terms of moments as shown in Table 4.1 and finally convolve the results with $\frac{\partial \kappa_{li}}{\partial y_i}$. This process is repeated l times. While evaluating the cumulants, one can now consider any additional constraint on the statistical information of the output signal. For example, one could consider normalizing the output by assuming that the output signals have unit variance. A MALTAB code is provided in Appendix C to perform

such computations accurately. When all the computations are complete, one obtains

$$\frac{\partial \phi}{\partial w_{ij}} = \left[W^{-T}\right]_{ij} - f(y_i)x_j \tag{4.42}$$

If no constraints are assumed, then the nonlinear function is computed to be

$$f_a(y) = \frac{71}{12}y^{15} - \frac{355}{12}y^{13} + \frac{190}{3}y^{11} - \frac{4033}{24}y^9 + \frac{941}{3}y^7 + \frac{47}{8}y^5 + y^3 + y \tag{4.43}$$

However, if the output signals are assumed to have unit variance, then

$$f_b(y) = \frac{71}{12}y^{15} + \frac{497}{12}y^{13} - \frac{259}{6}y^{11} - \frac{4265}{24}y^9 + \frac{1937}{12}y^7 + \frac{1285}{8}y^5 - \frac{325}{2}y^3 \tag{4.44}$$

Therefore, the update law in the general case is defined as

$$\dot{W} = \eta\left[W^{-T} - f(\mathbf{y})\mathbf{x}^T\right] \tag{4.45}$$

In [24], the assumption that the variance of the signal is unity was considered from the start of the derivation. In the work presented here, however, no such assumption was considered. Instead, the signals are assumed to have unconstrained unknown variance. Any constraint on the variance is considered only at the last step of the algorithm derivation. Such considerations are supported in [67], where the author makes the observation that one should not consider simplification of the expression at the last step of the development. Because such last step simplification would give rise to some other terms that are essential to describe the function. We note that, in [24], the constraint that the variance is unity was assumed before taking the gradient of the performance index. Therefore, the approximate expression of the entropy in this case is

$$H(x) \approx \frac{1}{2}\ln(2\pi e) - \frac{\kappa_3^2}{23!} - \frac{\kappa_4^2}{24!} + \frac{5}{8}\kappa_3^2\kappa_4 + \frac{1}{16}\kappa_3^3 \tag{4.46}$$

and the corresponding the nonlinear function is

$$f_d(y) = \frac{3}{4}y^{11} + \frac{25}{4}y^9 - \frac{14}{3}y^7 - \frac{47}{4}y^5 + \frac{29}{4}y^3 \tag{4.47}$$

If the constraint of unit variance on output is assumed at the end of the derivation, then the nonlinear function becomes expressed differently in approximation

$$f_c(y) = \frac{3}{4}y^{11} - \frac{1}{2}y^9 - \frac{29}{12}y^7 - \frac{17}{2}y^5 - y^3 + y \tag{4.48}$$

The graphs of the four different nonlinear functions described above are shown together in Figure 4.2.

In addition, a third order approximation of the logarithmic function is considered. This would provide a better approximation of the entropies of the output signals. Figure 4.3 shows the plot of the logarithmic function and its first, second and third order approximations.

4.3.1 Computer Simulations

Using the feedforward structure as shown in Figure 4.1, computer simulations of the algorithm defined by equation (4.45) are conducted by considering the developed nonlinear function f_a and the nonlinear function f_d developed in [24]. The simulations are performed using a uniformity distributed random signal over the interval $[-1, 1]$ and a sine waveform of frequency $1Hz$. The mixing matrix A is a random matrix defined by

$$A = \begin{bmatrix} 0.4385 & 0.2209 \\ 0.0421 & 0.7463 \end{bmatrix}$$

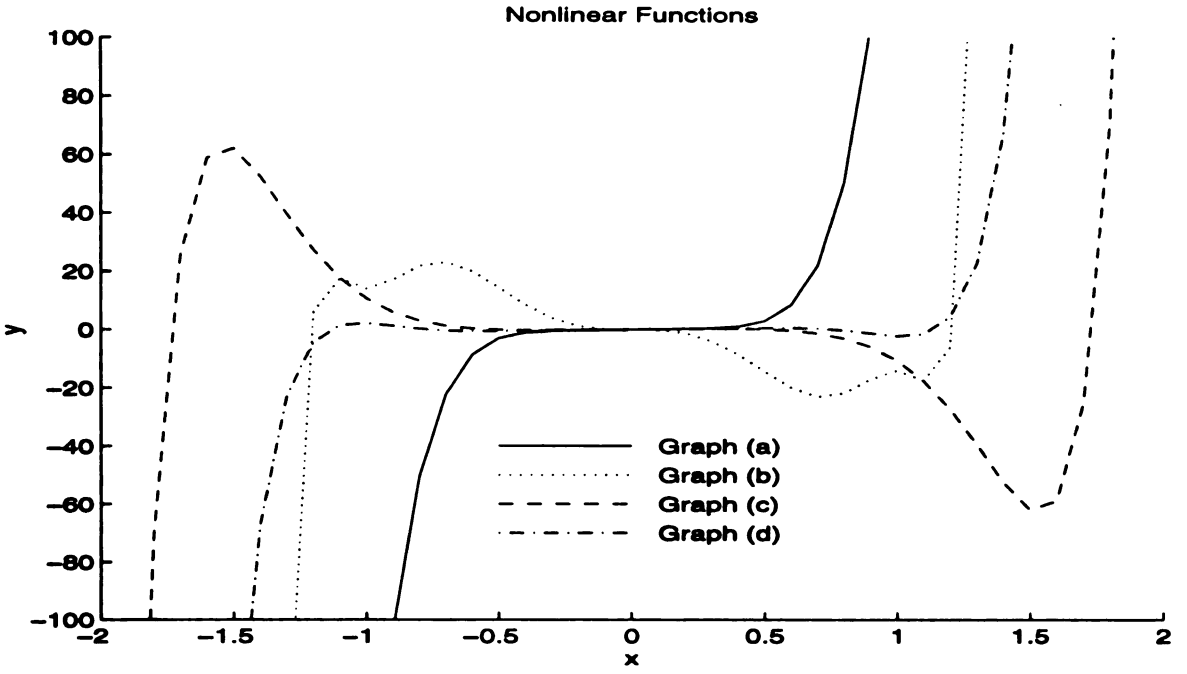


Figure 4.2. Nonlinear function. Graphs (a)-(d) show the plots of the functions $f_a(y)$ through $f_d(y)$

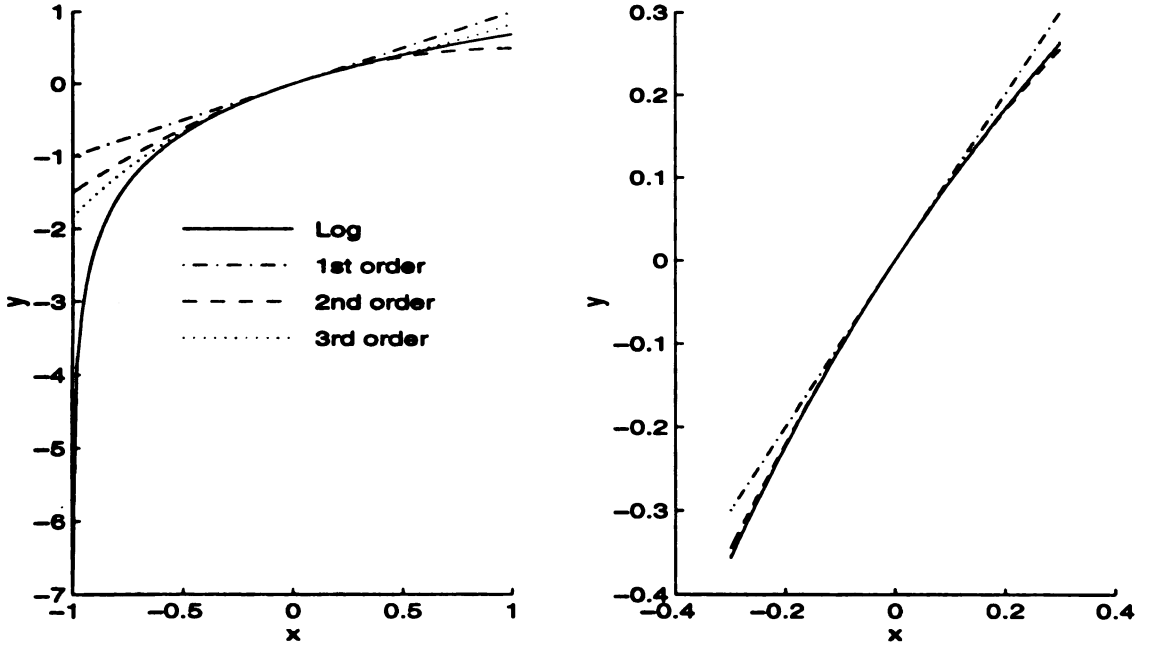


Figure 4.3. Natural Logarithmic function and few orders of approximations

The initial weights are also randomly chosen and are defined by the matrix

$$W_0 = \begin{bmatrix} 0.5277 & 0.4755 \\ 0.7625 & -0.5303 \end{bmatrix}$$

Thus, the initial overall gain matrix is

$$G_0 = W_0 A = \begin{bmatrix} 0.3356 & 0.5572 \\ -0.0647 & -0.1067 \end{bmatrix}$$

Clearly, such a matrix is not a generalized permutation matrix. Thus, it is desired that the network would update its parameters such that the gain matrix $W_f A$, where W_f is the final weight matrix, is a generalized permutation matrix.

Using the algorithm defined by equation (4.45) and the nonlinear function f_a defined by equation (4.43) and considering a learning rate $\eta = 0.005$, the network converges to

$$W_f = \begin{bmatrix} -0.0703 & 0.7625 \\ 1.8137 & -0.5303 \end{bmatrix}$$

Thus,

$$W_f A = \begin{bmatrix} 0.0013 & 0.5535 \\ 0.7729 & 0.0049 \end{bmatrix}$$

is a generalized matrix. Consequently, the algorithm converges to a separating solution. Figure 4.4 shows the results of such a simulations.

Using the same mixing matrix A and initial weight matrix W_0, and considering a learning rate $\eta = 0.1$, the algorithm defined by equation (4.45) and the nonlinear

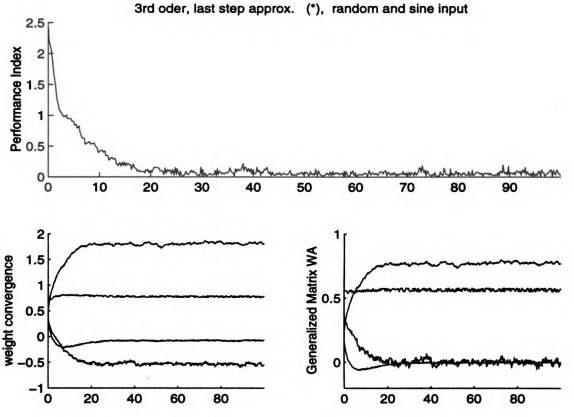


Figure 4.4. Computer Simulation of Feedforward Structure Using f_a

function f_d defined by equation (4.47) converges to the final weight matrix

$$W_f = \begin{bmatrix} -1.4506 & 3.3833 \\ 2.2009 & -0.2815 \end{bmatrix}$$

Thus, final overall gain matrix

$$G_f = W_f A = \begin{bmatrix} 0.0127 & 1.2280 \\ 1.6472 & -0.0006 \end{bmatrix}$$

is also a generalized permutation matrix. Figure 4.5 shows the results of the simulations. These two simulations are a sample of numerous simulations that were conducted to analyze the performance of the algorithms for various initial conditions. In all cases, the algorithm always converges to a desired solution.

4.4 Static Case: Feedback Network Structure

Recall that the mutual information is the algebraic sum of the joint entropy and the sum of all marginal entropies, as described in equation (4.3). Because of the opposite signs that appear next to each of these two terms, one can deduce that the minimization of the mutual information is somehow, but not exactly, equivalent to the minimization of the sum of marginal entropies and/or the maximization of the joint entropy. In the work of Bell and Sejnowski [45], the authors approached the problem by maximizing the joint entropy in order to develop an update law for the network. On the other hand, I propose an update law based on the minimization of the sum of marginal entropies. Thus, the theorem below is stated.

Theorem 2 *Under the assumption that $q_{mi} = 0 \; \forall m \neq i$, the update law*

$$\dot{d}_{ij} = \eta f_\alpha(y_i) y_j \tag{4.49}$$

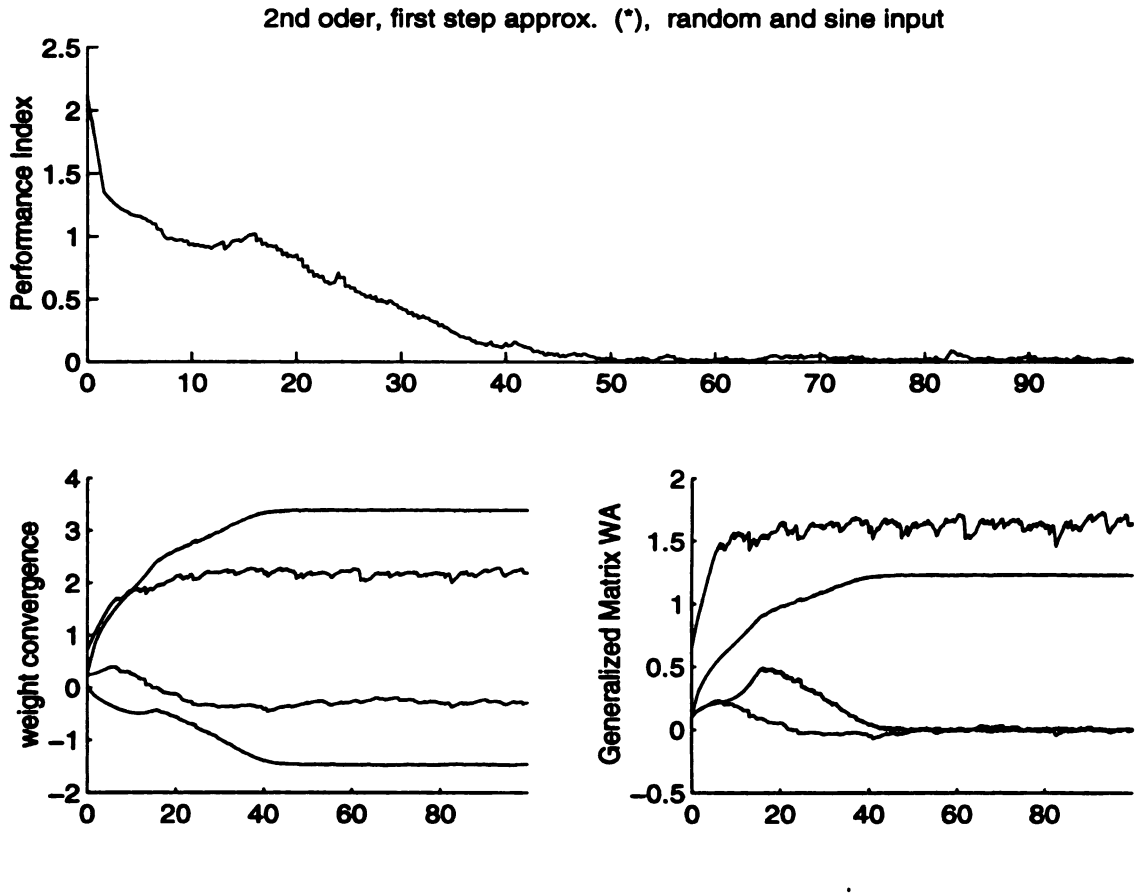


Figure 4.5. Computer Simulation of Feedforward Structure Using f_d

is derived by minimizing the functional

$$\phi = \sum_i H(y_i) \tag{4.50}$$

Proof:

The parameters of the network are updated using the gradient descent method

$$\dot{d}_{ij} = -\eta \frac{\partial \phi}{\partial d_{ij}} \tag{4.51}$$

However, using equation (2.23), the gradient of the function is expressed as

$$\frac{\partial \phi}{\partial d_{ij}} = \sum_m \frac{\partial \phi}{\partial y_m} \frac{\partial y_m}{\partial d_{ij}} \tag{4.52}$$

$$= -\sum_m \frac{\partial \phi}{\partial y_m} q_{mi} y_j \tag{4.53}$$

By considering the assumption that $q_{mi} = 0 \; \forall m \neq i$, then

$$\frac{\partial \phi}{\partial d_{ij}} = -q_{ii} \frac{\partial \phi}{\partial y_i} y_j \tag{4.54}$$

$$= -q_{ii} f_\alpha(y_i) y_j \tag{4.55}$$

Therefore , the parameters of the D matrix are updated according to

$$\dot{d}_{ij} = \eta f_\alpha(y_i) y_j \tag{4.56}$$

□

The algorithm described by equation (4.56) provides a justification of specialized view of the algorithm developed by Herault and Jutten based on neuromimetic approach [8], since Equation (4.56) is analogous to equation (2.6) where $f(x) = f_\alpha(x)$ and $g(x) = x$.

4.4.1 Computer Simulations

Using the feedback structure as defined by Figure 2.1, the performance of the algorithm defined by the nonlinear function f_a, as described by equation (4.43), will be compared to that defined by the nonlinear function f_d which is developed in [24] and is described by equation (4.47). Using this structure, the update law for the parameters is as follows:

$$\dot{w}_{ij} = \eta_\alpha f_\alpha(y_i) y_j \quad i \neq j, \alpha = a, d \tag{4.57}$$

Computer simulations using 2 sine functions of respective frequencies $1Hz$ and $2Hz$ are first conducted. The mixing matrix is

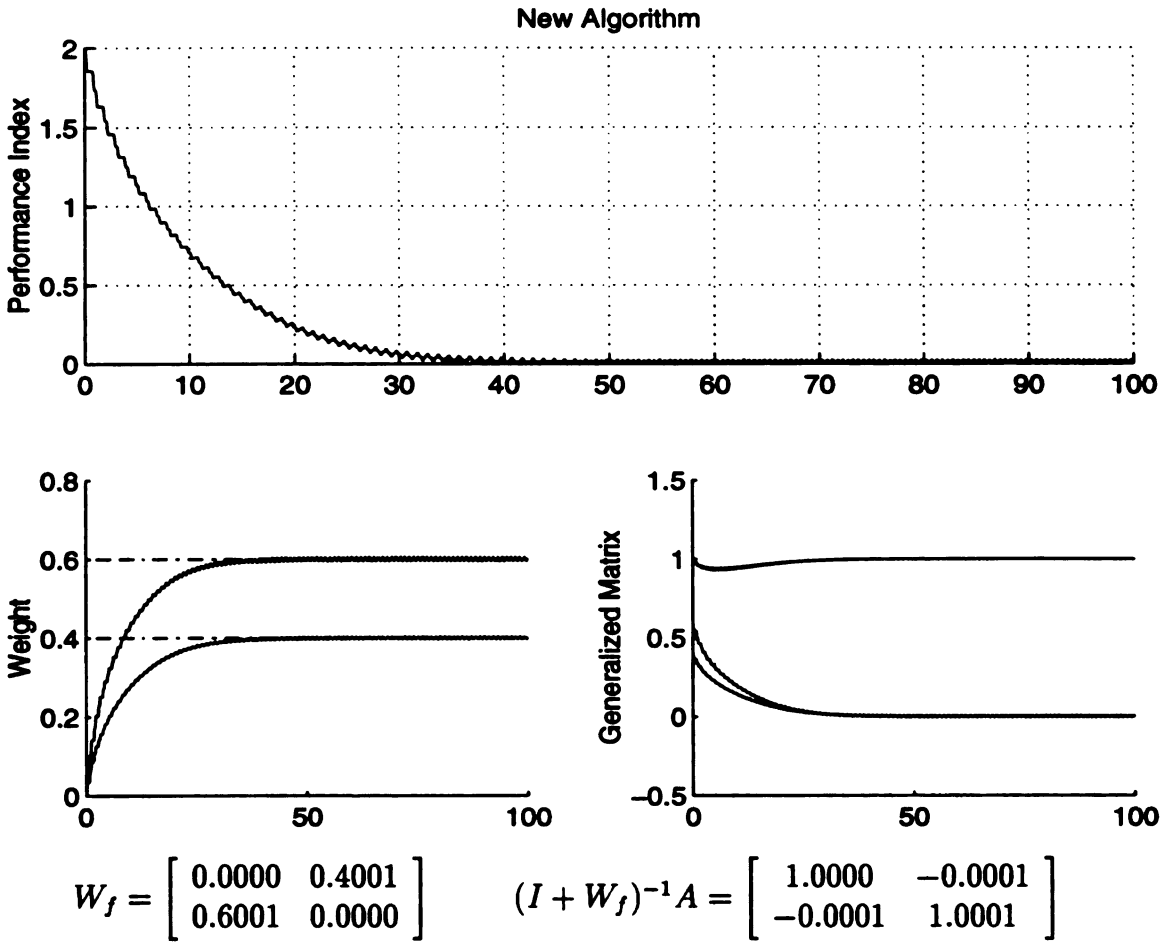
$$A = \begin{bmatrix} 1.0 & 0.4 \\ 0.6 & 1.0 \end{bmatrix} \tag{4.58}$$

The initial weight matrix is zero and the learning rates are the following

$$\eta_a = 0.005 \qquad \eta_d = 0.1 \tag{4.59}$$

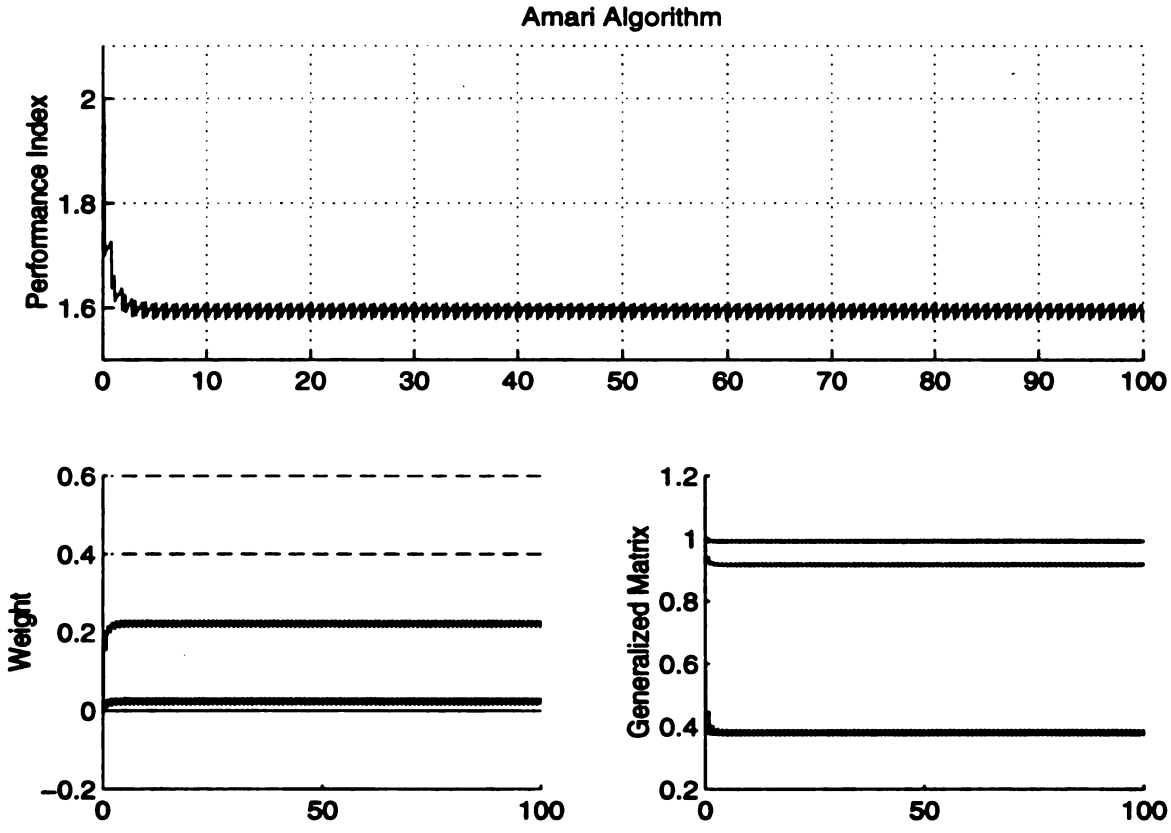
Figures 4.6 and 4.7 show the performance of both algorithms. One can observe that the algorithm defined by f_a succeeded in separating the original sources, whereas the algorithm defined by f_d fails to perform separation.

Computer simulations are also performed using a Gaussian random signal and a sine function of frequency $2Hz$. Also, other simulations using a Gaussian random signal and a square function are completed. In each case, the algorithm defined by f_d fails to separate the signals. On the other hand, the algorithm defined by f_a performs the separation task successfully. Figures 4.8and 4.9 show the performance of both algorithms in the case of a Gaussian random signal and sine function were considered as the unknown sources. Figure 4.10 shows the original sources, the mixtures and the



$$W_f = \begin{bmatrix} 0.0000 & 0.4001 \\ 0.6001 & 0.0000 \end{bmatrix} \qquad (I + W_f)^{-1}A = \begin{bmatrix} 1.0000 & -0.0001 \\ -0.0001 & 1.0001 \end{bmatrix}$$

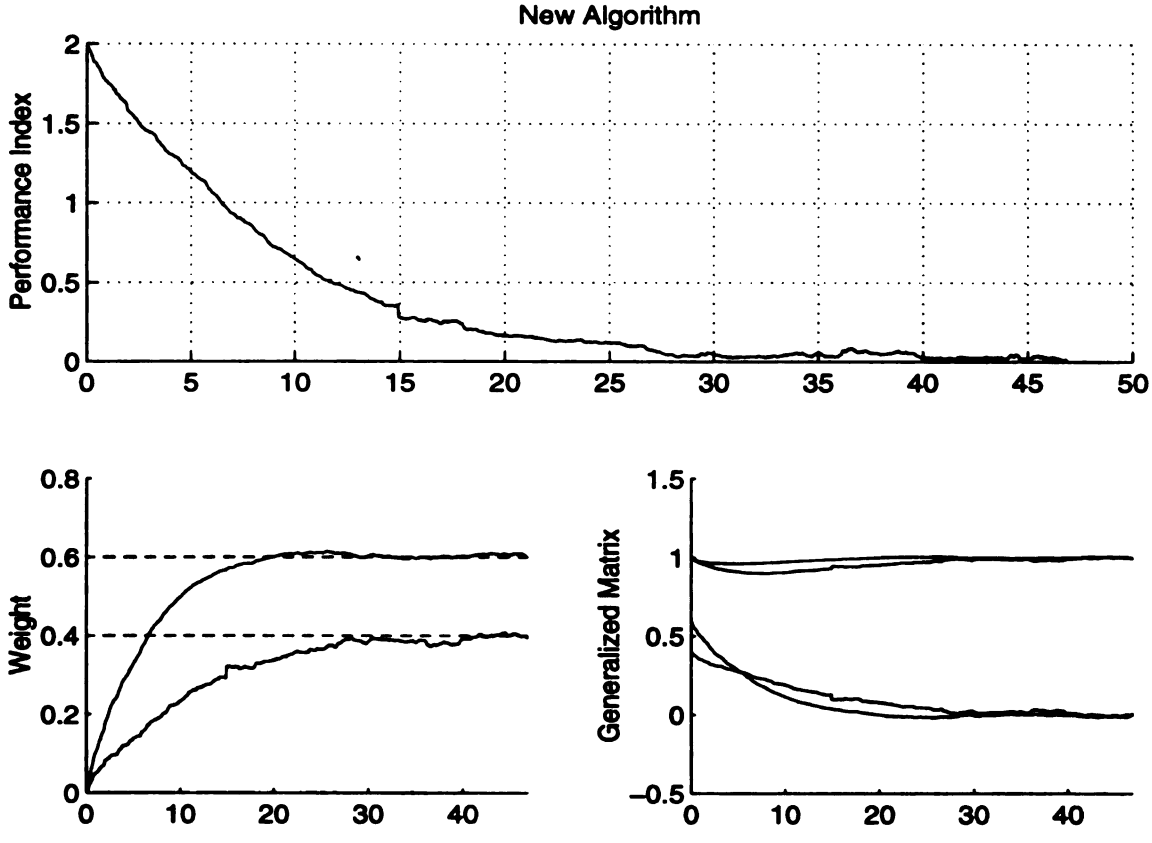
Figure 4.6. Performance of the Algorithm defined by the nonlinear function f_a



$$W_f = \begin{bmatrix} 0.0000 & 0.0222 \\ 0.2229 & 0.0000 \end{bmatrix} \qquad (I + W_f)^{-1}A = \begin{bmatrix} 0.9916 & 0.3797 \\ 0.3789 & 0.9154 \end{bmatrix}$$

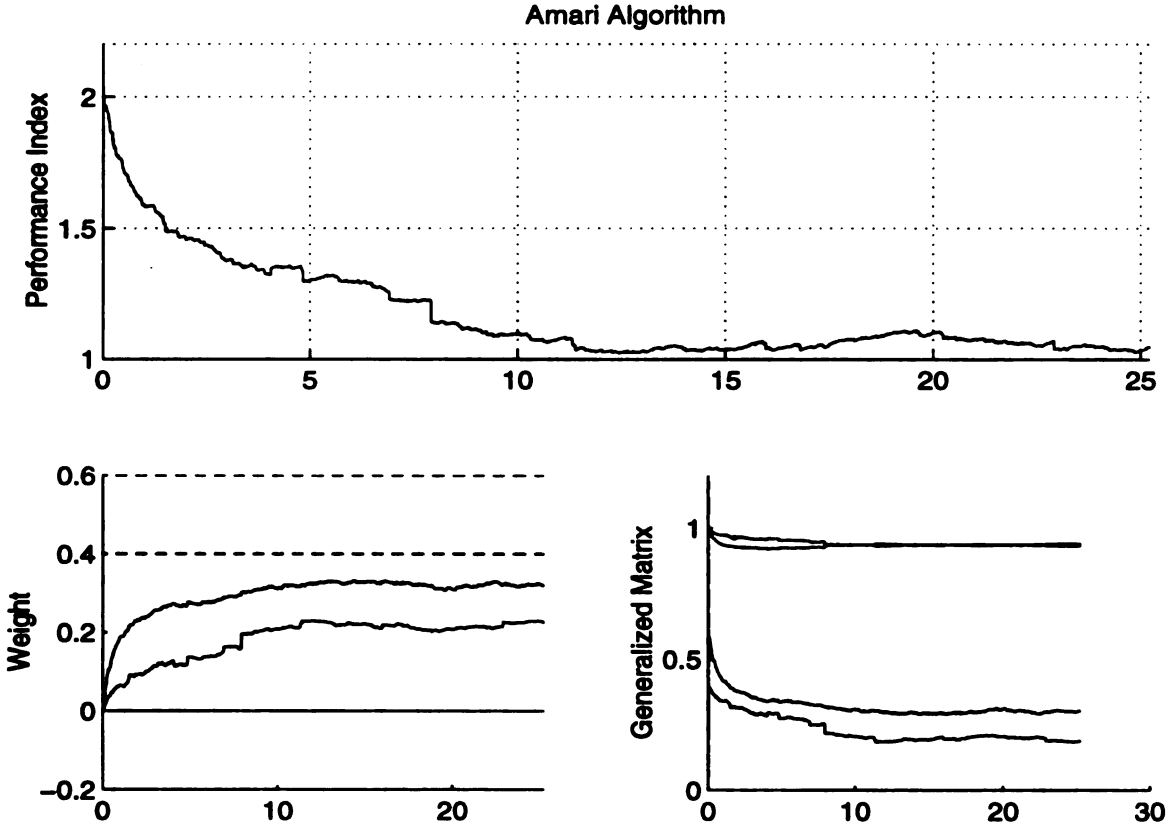
Figure 4.7. Performance of the Algorithm defined by the nonlinear function f_d

output of the network at convergence.



$$W_f = \begin{bmatrix} 0.0000 & 0.3940 \\ 0.6003 & 0.0000 \end{bmatrix} \qquad (I + W_f)^{-1}A = \begin{bmatrix} 1.0002 & 0.0078 \\ -0.0004 & 0.9953 \end{bmatrix}$$

Figure 4.8. Performance of the Algorithm defined by the nonlinear function f_a using random Gaussian and a sine function



$$W_f = \begin{bmatrix} 0.0000 & 0.2250 \\ 0.3199 & 0.0000 \end{bmatrix} \qquad (I + W_f)^{-1} A = \begin{bmatrix} 0.4949 & 0.1591 \\ 0.2323 & 0.4908 \end{bmatrix}$$

Figure 4.9. Performance of the Algorithm defined by the nonlinear function f_d using a random Gaussian and a sine function

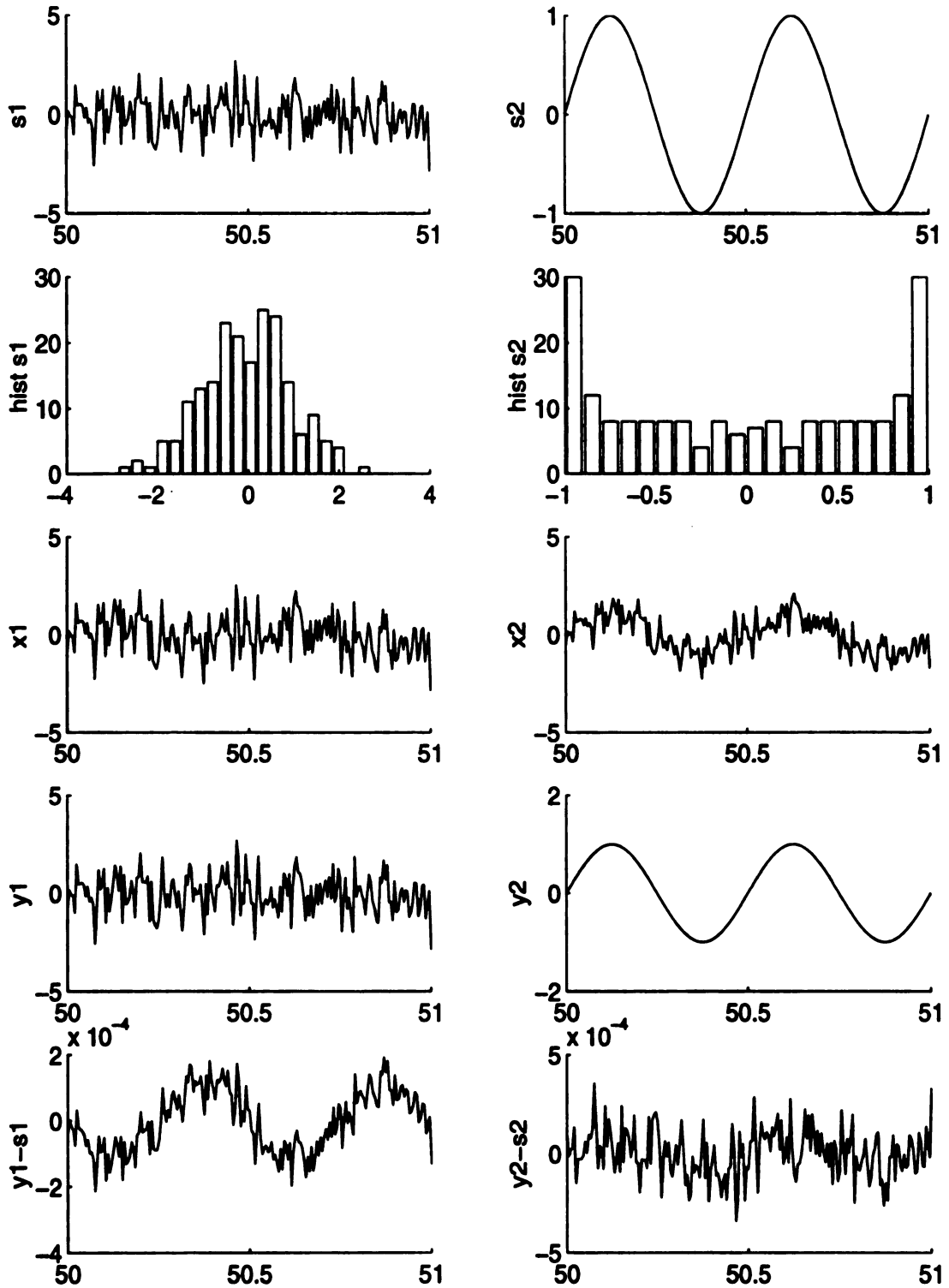


Figure 4.10. Output of the Algorithm defined by the nonlinear function f_a using random Gaussian and a sine function

CHAPTER 5

Blind Separation in a Dynamic Environment: FeedForward Structure

5.1 Problem Definition

In Chapter 3, static modeling for both the environment and the network was considered. However, media cannot always be modeled as such. Therefore, one must consider more realistic environments, define their models and develop an update law to recover the original signals. Such an environment will be modeled as a linear dynamical system. Consequently, the network will also be modeled as a linear dynamical system. Figure 5.1 depicts the architecture of this situation. The realiza-

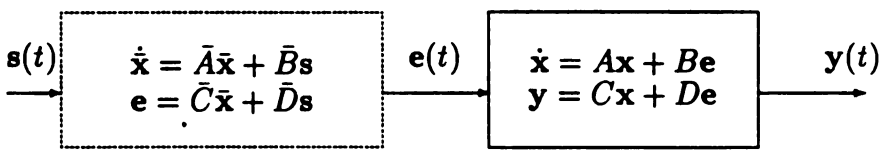


Figure 5.1. Feedforward Architecture

tion $\bar{\mathcal{D}} = (\bar{A}, \bar{B}, \bar{C}, \bar{D})$ represents the parameters of the environment, whereas $\mathcal{D} = (A, B, C, D)$ is that of the network. These parameters dictate the dynamics of both the environment and the network systems. While the realization $\bar{\mathcal{D}}$ is constant and models the behavior of the environment, the realization $\mathcal{D}$ of the network are yet to be modeled or defined. The goal here is to find an adaptive law (algorithm), in such a way, that when these parameters converge to some stable solution, the network output signals are replicas or similar to the original unknown sources. Or, by using the definition of wave-preserving, it is desired that the output vector preserves the waveforms of the components of the original signals. So, the question at hand is how should one update these parameters to arrive at such values? and, do such values exist in the first place? In this chapter, first, the existence of a theoretical solution to the problem will be shown. Then, Optimization Theory will be utilized to develop an adaptive rule based on a criterion that defines the independence of the output signals. A state representation of the network model that renders its implementation in VLSI attractive, and which also minimizes the number of parameters that characterize the network, will be developed. The algorithm will be tested via computer simulations. Limitations and some possible improvements of the algorithm will also be discussed.

5.2 Existence of a Theoretical Solution

Suppose that an update necessary to recover the original signals is developed and that the parameters of the neural networks have converged to the realization $\mathcal{D}^* = (A^*, B^*, C^*, D^*)$. Then, because of the wave preserving property, there exists a permutation matrix P and a positive definite diagonal matrix Γ such that

$$\mathbf{y}(t) = P\,\Gamma\,\mathbf{s}(t)$$

There also exists a nonsingular matrix T such that $\mathbf{x} = T\bar{\mathbf{x}}$. Thus, differentiating both sides with respect to time:

$$\begin{aligned}
\dot{\mathbf{x}} = T\dot{\bar{\mathbf{x}}} \;&\Rightarrow\; A^*\mathbf{x} + B^*\mathbf{e} = T\bar{A}\bar{\mathbf{x}} + T\bar{B}\mathbf{s} \\
&\Rightarrow\; A^*\mathbf{x} + B^*(\bar{C}\bar{\mathbf{x}} + \bar{D}\mathbf{s}) = T\bar{A}T^{-1}\mathbf{x} + T\bar{B}\mathbf{s} \\
&\Rightarrow\; A^*\mathbf{x} + B^*\bar{D}\mathbf{s} = (T\bar{A}T^{-1} - B^*\bar{C}T^{-1})\mathbf{x} + T\bar{B}\mathbf{s} \qquad (5.1)
\end{aligned}$$

Also,

$$\begin{aligned}
\mathbf{y} = P\Gamma\mathbf{s} \;&\Rightarrow\; P\Gamma\mathbf{s} = C^*\mathbf{x} + D^*\mathbf{e} \\
&\Rightarrow\; P\Gamma\mathbf{s} = C^*\mathbf{x} + D^*(\bar{C}\bar{\mathbf{x}} + \bar{D}\mathbf{s}) \\
&\Rightarrow\; P\Gamma\mathbf{s} = (C^* + D^*\bar{C}T^{-1})\mathbf{x} + D^*\bar{D}\mathbf{s} \qquad (5.2)
\end{aligned}$$

Therefore,

$$\begin{aligned}
A^* &= T\bar{A}T^{-1} - B^*\bar{C}T^{-1} && (5.3) \\
B^*\bar{D} &= T\bar{B} && (5.4) \\
C^* + D^*\bar{C}T^{-1} &= 0 && (5.5) \\
D^*\bar{D} &= I && (5.6)
\end{aligned}$$

Consequently,

$$\begin{aligned}
A^* &= T\left(\bar{A} + \bar{B}\bar{D}^{-1}\bar{C}\right)T^{-1} && (5.7) \\
B^* &= T\bar{B}\bar{D}^{-1} && (5.8) \\
C^* &= -P\Gamma\bar{D}^{-1}\bar{C}T^{-1} && (5.9) \\
D^* &= P\Gamma\bar{D}^{-1} && (5.10)
\end{aligned}$$

Particular Case

Suppose that $T = P\Gamma = I$ where I is the identity square matrix, then

$$A^* = \bar{A} - \bar{B}\bar{D}^{-1}\bar{C} \tag{5.11}$$

$$B^* = \bar{B}\bar{D}^{-1} \tag{5.12}$$

$$C^* = \bar{D}^{-1}\bar{C} \tag{5.13}$$

$$D^* = \bar{D}^{-1} \tag{5.14}$$

Thus, it is shown that the theoretical solution to the problem exists, and that if the appropriate energy function of the independence criterion is defined for the problem, then the parameters of the system will converge to one of the solutions defined by equations (5.7)-(5.10). However, before going into defining such energy functions, possible realizations of the system will be discussed.

5.3 State Realization

In this section, different network structures and their suitability to the problem of the separation of signals will be introduced. The network is assumed to be a multi-input multi-output system described by

$$Y(s) = H(s)U(s) \tag{5.15}$$

$H(s)$ is the $m \times r$ system's transfer matrix where r and m are, respectively, the number of input and output signals. The goal is to find a dynamical system $\mathcal{D} = (A, B, C, D)$ whose transfer function is the $m \times r$ matrix $H(s)$ where

$$H(s) = C(sI - A)^{-1}B + D \tag{5.16}$$

However, before starting the development of the structure, the following assumptions on the transfer function matrix $H(s)$ will be considered.

(A1) $H(s)$ is stable and minimum phase

(A2) Each entry of the matrix $H(s)$ is in its irreducible form

(A3) entries of $H(s)$ have no common poles

The rationale behind these assumptions will be explained in the development of the structure.

5.3.1 Controllable/Observable Canonical Form

The literature provides one with various realizations of $H(s)$, [59]. Of these, a canonical controllable realization will be considered. The procedure to develop the proposed structure is simple and straight forward. A main feature of the structure is that there is no direct coupling between the input signals. This realization has very nice features, in terms of its perfect fitness to the problem of blind separation, and also in terms of circuit implementation. It presents the least number of parameters of the state representation of the transfer matrix $H(s)$. Such features will enable one to define simple update laws for the parameters and to analyze their performance based on the independence criterion of the output signals. The structure generates a bank of filters that are jointly decoupled. The analysis provided below will display such characteristics of the developed realization. Some figures are also presented to illustrate this point.

Each output is described as

$$Y_i(s) = \sum_{j=1}^{r} H_{ij}(s)U_j(s) = \sum_{j=1}^{r} Y_{ij}(s) \tag{5.17}$$

According to assumption (A2), it is assumed that each $H_{ij}(s)$ is irreducible. Thus, we choose to represent $H_{ij}(s)$ by the canonical controllable realization $\mathcal{D}_{ij} = (A_{ij}, \mathbf{b}_{ij}, \mathbf{c}_{ij}^T, d_{ij})$.

$$\begin{bmatrix} \dot{x}_{ij}^{(1)} \\ \dot{x}_{ij}^{(2)} \\ \vdots \\ \dot{x}_{ij}^{(\mu_{ij}-1)} \\ \dot{x}_{ij}^{(\mu_{ij})} \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ -\alpha_{ij}^{(\mu_{ij})} & \cdots & \cdots & \cdots & -\alpha_{ij}^{(1)} \end{bmatrix}}^{A_{ij} \in \mathcal{R}^{\mu_{ij} \times \mu_{ij}}} \begin{bmatrix} x_{ij}^{(1)} \\ x_{ij}^{(2)} \\ \vdots \\ x_{ij}^{(\mu_{ij}-1)} \\ x_{ij}^{(\mu_{ij})} \end{bmatrix} + \overbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}^{\mathbf{b}_{ij} \in \mathcal{R}^{\mu_{ij}}} u_j$$

$$y_{ij} = \underbrace{\begin{bmatrix} c_{ij}^{(1)} & c_{ij}^{(2)} & \cdots & c_{ij}^{(\mu_{ij}-1)} & c_{ij}^{(\mu_{ij})} \end{bmatrix}}_{\mathbf{c}_{ij} \in \mathcal{R}^{\mu_{ij}}} \mathbf{x}_{ij} + d_{ij} u_j$$

where $\alpha_{ij}^{(k)}$ are the coefficients of the characteristic polynomial of the matrix A_{ij} defined by

$$\det(\lambda I - A_{ij}) = \lambda^{\mu_{ij}} + \alpha_{ij}^{(1)} \lambda^{\mu_{ij}-1} + \cdots + \alpha_{ij}^{(\mu_{ij}-1)} \lambda + \alpha_{ij}^{(\mu_{ij})} \tag{5.18}$$

The sub-realization $\mathcal{D}_{ij} = (A_{ij}, \mathbf{b}_{ij}, \mathbf{c}_{ij}^T, d_{ij})$ provides us with a set of different filtered versions of the j^{th} input u_j that affect only the i^{th} output y_i. Figure 5.2(a) presents the diagram of the state representation the sub-realization $\mathcal{D}_{ij}$, whereas Figure 5.2(b) shows the combination of the state and the output representation of the sub-realization $\mathcal{D}_{ij} = (A_{ij}, \mathbf{b}_{ij}, \mathbf{c}_{ij}^T, d_{ij})$. The combination of these realizations $\mathcal{D}_{ij}$, $j = 1, \cdots, r$, will provide us with the i^{th} output y_i. Figure 5.2(c) shows such a structure. But, let's proceed to see how it is obtained.

To do so, one must define the vector $\mathbf{x}_i = \begin{bmatrix} \mathbf{x}_{i1}^T & \cdots & \mathbf{x}_{ir}^T \end{bmatrix}^T$ to be a vector in $\mathcal{R}^{\mu_i}$

where $\mu_i = \sum_{j=1}^{r} \mu_{ij}$. Then,

$$\begin{bmatrix} \dot{\mathbf{x}}_{i1} \\ \vdots \\ \dot{\mathbf{x}}_{ij} \\ \vdots \\ \dot{\mathbf{x}}_{ir} \end{bmatrix} = \overbrace{\begin{bmatrix} A_{i1} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \cdots & \vdots \\ 0 & \cdots & A_{ij} & \cdots & 0 \\ \vdots & \cdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & A_{ir} \end{bmatrix}}^{A_i \in \mathcal{R}^{\mu_i \times \mu_i}} \mathbf{x}_i + \overbrace{\begin{bmatrix} \mathbf{b}_{i1} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \cdots & \vdots \\ 0 & \cdots & \mathbf{b}_{ij} & \cdots & 0 \\ \vdots & \cdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \mathbf{b}_{ir} \end{bmatrix}}^{B_i \in \mathcal{R}^{\mu_i \times r}} \mathbf{u}$$

$$y_i = \underbrace{\begin{bmatrix} \mathbf{c}_{i1}^T & \cdots & \mathbf{c}_{ij}^T & \cdots & \mathbf{c}_{ir}^T \end{bmatrix}}_{\mathbf{c}_i \in \mathcal{R}^{\mu_i}} \mathbf{x}_i + \underbrace{\begin{bmatrix} d_{i1} & \cdots & d_{ij} & \cdots & d_{ir} \end{bmatrix}}_{\mathbf{d}_i \in \mathcal{R}^r} \mathbf{u}$$

Therefore,

$$\dot{\mathbf{x}}_i = A_i \mathbf{x}_i + B_i \mathbf{u} \tag{5.19}$$

$$y_i = \mathbf{c}_i^T \mathbf{x}_i + \mathbf{d}_i^T \mathbf{u} \tag{5.20}$$

An important feature of this realization is the inherent parallel structure that it possesses, as shown in Figure 5.2(c). This structure clearly decouples the inputs from each other. This will be used as an advantage when an update law that pushes for the output independence is to be developed.

At the final stage, all the filters will be combined and the structure below will be developed. Thus, the vector $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T & \cdots & \mathbf{x}_m^T \end{bmatrix}^T$ in R^n is defined, where $n =$

$\sum_{i=1}^{m} \mu_i$. Then,

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \vdots \\ \dot{\mathbf{x}}_i \\ \vdots \\ \dot{\mathbf{x}}_m \end{bmatrix} = \overbrace{\begin{bmatrix} A_1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \cdots & \vdots \\ 0 & \cdots & A_i & \cdots & 0 \\ \vdots & \cdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & A_m \end{bmatrix}}^{A \in \mathcal{R}^{n \times n}} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_m \end{bmatrix} + \overbrace{\begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_i \\ \vdots \\ \mathbf{B}_m \end{bmatrix}}^{B \in \mathcal{R}^{n \times r}} \mathbf{u}$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_m \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{c}_1^T & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \cdots & \vdots \\ 0 & \cdots & \mathbf{c}_i^T & \cdots & 0 \\ \vdots & \cdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \mathbf{c}_m^T \end{bmatrix}}_{C \in \mathcal{R}^{m \times n}} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_m \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{d}_1^T \\ \vdots \\ \mathbf{d}_i^T \\ \vdots \\ \mathbf{d}_m^T \end{bmatrix}}_{D \in \mathcal{R}^{m \times r}} \mathbf{u}$$

Thus,

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{5.21}$$

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u} \tag{5.22}$$

where A, B, C and D are defined above. The realization $\mathcal{D} = (A, B, C, D)$ is called the controllable canonical realization of $H(s)$.

In [59], it is discussed that $\mathcal{D}_{ij}^T = (A_{ij}^T, \mathbf{b}_{ij}^T, \mathbf{c}ij^T, d_{ij})$ is called the observable canonical realization of $H_{ij}(s)$. Therefore, by following the same procedure, as discussed above, to develop a controllable canonical realization of $H(s)$, one can demonstrate that $\mathcal{D}^T = (A^T, B^T, C^T, D^T)$ is an observable canonical realization of $H(s)$.

By assumption (A2) and (A3), this structure is irreducible. If these two assumptions are not verified, then a more compact realization can be found [59]. But, for the

sake of generality, it is safe to have such assumptions. Assumption (A1) is introduced because, in real life applications and in computer implementation, such a feature is needed in order to guarantee stability.

5.3.2 Features of the realization

In the course of development of the realization, some features of the proposed realization have been mentioned. These include the decoupling between inputs, the inherent parallel structure of the realization, and of most importance, the reduced number of parameters by which the system can be defined. This last feature is very important in terms of circuit implementation because, it implies reduction in the circuit micro-electronic implemtation area.

Starting with a transfer matrix whose entries have no common poles, the state representation of such a transfer matrix would require as many poles as the transfer matrix has. This number is equal to the size of the state vector $\mathbf{x}$, namely n. Despite the fact that the matrix A is an $n \times n$ matrix, it is being represented by only $n = \sum_{i,j} \mu_{ij}$, since each matrix A_{ij} is represented only by the coefficients of is characteristic polynomial, and the remaining entries are all ones and zeros. This represents a reduction from n^2 to just n parameters to represent the matrix A.

The matrix C is an $m \times n$ matrix. However, only n parameters of the matrix are nonzero. The other $n(m-1)$ are all zeros. This again represents a reduction in the number of parameters from nm to just n.

The matrix B is formed by only zeros and ones. Thus, there is no parameter for this matrix. Since B is an $n \times r$ matrix, this again represents a reduction of nr parameters that must be updated. Finally the D matrix is represented by $m \times r$ nonzero parameters.

Table 5.3.2 shows the number of parameters that would represent a general $m \times r$ transfer matrix with no common poles between its entries in one column. The other

Table 5.1. Number of parameters for 2 realizations

Realization	General	Proposed
A	n^2	n
B	nr	0
C	nm	n
D	mr	mr
Total	$n^2 + nr + nm + mr$	$2n + mr$

column shows the number of parameters for the controllable canonical realization that has been defined above. One observes that the order of the number of parameters that define the system has been reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

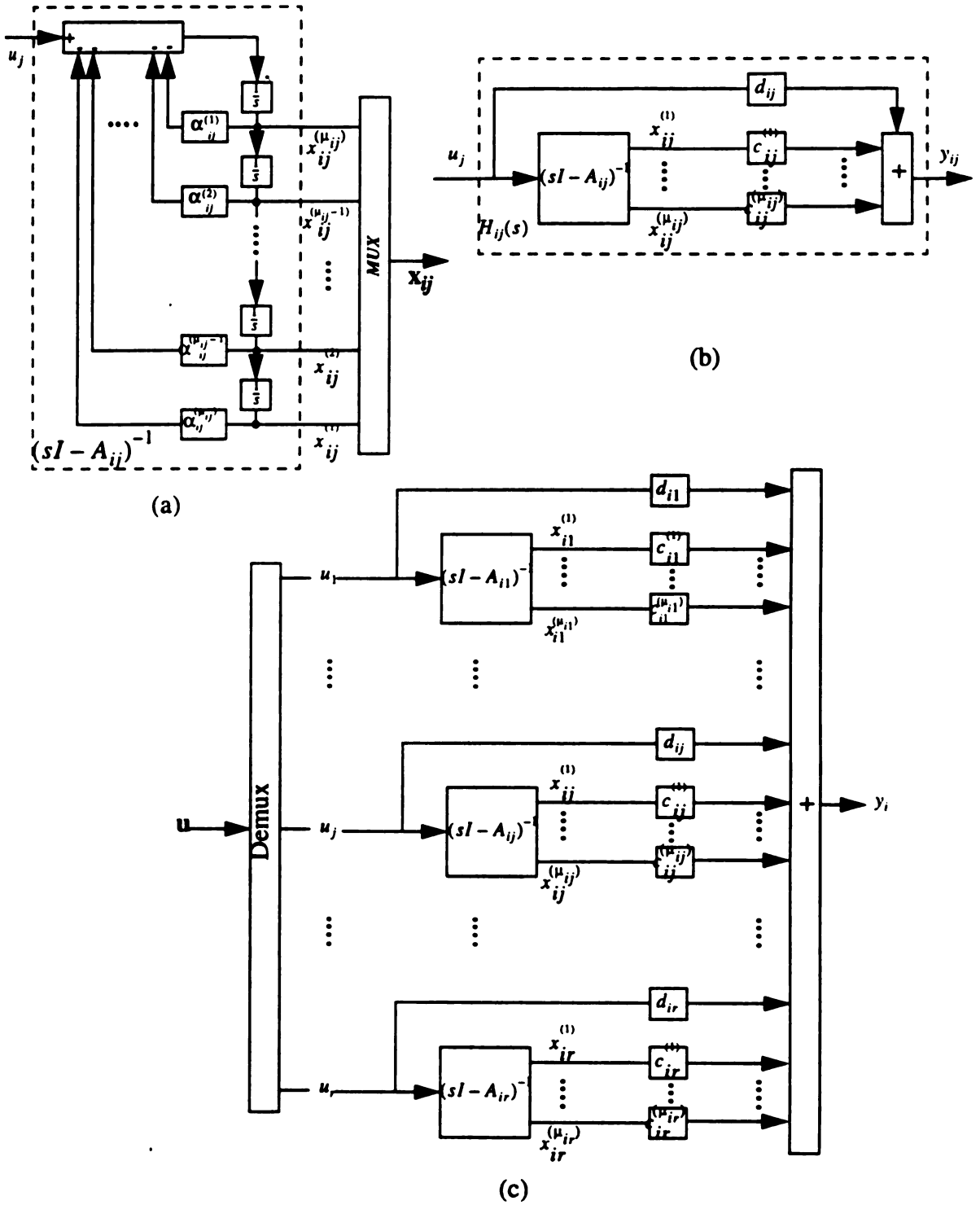


Figure 5.2. (a) State Equation $\dot{\mathbf{x}}_{ij} = A_{ij}\mathbf{x}_{ij} + \mathbf{b}_{ij}u_j$, (b) Output Equation $y_{ij} = \mathbf{c}_{ij}^T\mathbf{x}_{ij} + d_{ij}u_j$ and (c) Output Equation $y_i = \mathbf{c}_i^T\mathbf{x}_i + \mathbf{d}_i^T\mathbf{u}$

5.4 Adaptive Optimal Control Theory

Optimal adaptive control theory will be utilized to derive an update law for the blind separation problem in a dynamical environment. The derivation will be valid for both deterministic and stochastic cases. The update law will be derived for a general nonlinear dynamical case. Then, the derivation of the linear dynamical case will be straight forward.

Suppose that the independence of the output is characterized by

$$c(\mathbf{y}, \mathbf{w}) = \phi(t, \mathbf{x}, \mathbf{w}) \tag{5.23}$$

The goal is then to optimize the functional defined by (5.23) subject to the dynamics of the following system

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{w}_1), \qquad \text{with } \mathbf{x}(t_0) = \mathbf{x}_0 \tag{5.24}$$

$$\mathbf{y} = g(t, \mathbf{x}, \mathbf{w}_2) \tag{5.25}$$

where f and g are two twice-differentiable continuous functions, and $\mathbf{w}_1$ and $\mathbf{w}_2$ are the parameters of the network model. To accomplish the task of optimization, the following performance index is introduced

$$J(\mathbf{x}, \mathbf{w}) = \int_0^T \mathcal{L}(t, \mathbf{x}, \dot{\mathbf{x}}, \lambda, \mathbf{w}_1) dt$$

where $\mathbf{w} = [\mathbf{w}_1,\ \mathbf{w}_2]$ and $\mathcal{L}(.)$ is the Lagrangian function defined as

$$\mathcal{L}(t, \mathbf{x}, \dot{\mathbf{x}}, \lambda, \mathbf{w}) = \phi(t, \mathbf{x}, \mathbf{w}_2) + \lambda^T \Big(\dot{\mathbf{x}} - f(t, \mathbf{x}, \mathbf{w}_1) \Big)$$

By introducing the Lagrangian parameters $\lambda(t)$, the functional $J(\mathbf{x}, \mathbf{w})$ will be optimized by considering the different variables $\mathbf{w}(t)$, $\mathbf{x}(t)$ and $\lambda(t)$ as independent variables. There is no constraint on the final time condition of the system. Thus,

the optimization of the functional J is called a *free-end* variational problem. The variation of the integral leads to the following well-known Euler equations for the variables $\mathbf{x}(t)$ and $\lambda(t)$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} - \frac{d}{dt}\left\{\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{x}}}\right\} = 0 \tag{5.26}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} - \frac{d}{dt}\left\{\frac{\partial \mathcal{L}}{\partial \dot{\lambda}}\right\} = 0 \tag{5.27}$$

with boundary condition

$$\left.\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{x}}}\right|_{t=T} = 0 \tag{5.28}$$

Equation (5.26) is equivalent to

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{w}_1), \qquad \text{with } \mathbf{x}(0) = \mathbf{x}_0 \tag{5.29}$$

while equations (5.27) and (5.28) are equivalent to

$$\dot{\lambda} = -\left(\frac{\partial f}{\partial \mathbf{x}}\right)^T \lambda + \frac{\partial \phi}{\partial \mathbf{x}}, \qquad \text{with } \lambda(T) = 0 \tag{5.30}$$

The parameters will be updated using the gradient of the performance index

$$\dot{\mathbf{w}}_i = \eta \frac{\partial J}{\partial \mathbf{w}_i} = \eta \int_0^t \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} d\alpha, \quad i = 1, 2 \tag{5.31}$$

where η is the learning rate and is positive (negative) if the goal is to maximize (minimize) the performance index. Note here that λ is obtained by performing a back propagation through time starting at time $t = T$ with zero initial value.

Equation (5.31) can be simplified to

$$\dot{\mathbf{w}}_1 = -\eta \int_0^t \frac{\partial \lambda^T f}{\partial \mathbf{w}_1} \, d\alpha \tag{5.32}$$

$$\dot{\mathbf{w}}_2 = \eta \int_0^t \frac{\partial \phi}{\partial \mathbf{w}_2} \, d\alpha \tag{5.33}$$

The parameters could also be updated using an instantaneous update

$$\dot{\mathbf{w}}_i = \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} \quad i = 1, 2 \tag{5.34}$$

and would result in the following update laws for $\mathbf{w}_1$ and $\mathbf{w}_2$:

$$\dot{\mathbf{w}}_1 = -\eta \frac{\partial \lambda^T f}{\partial \mathbf{w}_1} \tag{5.35}$$

$$\dot{\mathbf{w}}_2 = \eta \frac{\partial \phi}{\partial \mathbf{w}_2} \tag{5.36}$$

Remark: one can observe that the update law for the parameter matrix $\mathbf{w}_2$ depends on the independence criterion function ϕ, whereas that of $\mathbf{w}_1$ depends on the choice of both the nonlinear function f and the independence criterion ϕ since the dynamics of λ depends on its direction along the state $\mathbf{x}$.

5.4.1 Computer Implementation

The computer simulation of the algorithm will proceed as follows:

1. Perform a forward integration in the time interval [0, T] in order to obtain the dynamic state equation and output equation defined by (5.24) and (5.25)

2. Perform a backward integration of the system's adjoint equation defined by the dynamics of equation (5.30)

3. Update the network parameters using either set of equations defined by (5.32-5.33) or (5.35-5.36)

4. While parameters are not convergent, go back to step 1.

This research has focused on the update of the parameters of $\mathbf{w}_2$ while keeping the parameters of $\mathbf{w}_1$ at their nominal solutions. In this case, one observes that the update of $\mathbf{w}_2$ does not depend on the adjoint state. Therefore, step 2 described above can be eliminated while we focus on the update of the $\mathbf{w}_2$ parameter.

5.5 Update Law Derivation

5.5.1 Nonlinear Dynamic Modeling

In this section, the output of the network is modeled as a nonlinear function of the weighted sum of the input vector **e** and state vector **x**.

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, w_1) = f(A\mathbf{x} + B\mathbf{e}) = f(\mathbf{u}) \tag{5.37}$$

$$\mathbf{y} = g(t, \mathbf{x}, w_2) = g(C\mathbf{x} + D\mathbf{e}) = g(\mathbf{v}) \tag{5.38}$$

Then, the update law for the parameters, as well as the state equation of the adjoint system, will be derived.

Update law Derivation for A and B

Let $\alpha = a,\ b$. Then,

$$\begin{aligned}
\dot{\alpha}_{ij} &= -\eta \frac{\partial(\lambda^T f)}{\partial \alpha_{ij}} \\
&= -\eta \sum_m \lambda_m \frac{\partial f_m(u_m)}{\partial \alpha_{ij}} \\
&= -\eta \sum_m \lambda_m f'_m(u_m) \frac{\partial u_m}{\partial \alpha_{ij}}
\end{aligned}$$

Thus, one needs to compute $\frac{\partial u_m}{\partial \alpha_{ij}}$:

$$\frac{\partial u_m}{\partial a_{ij}} = \sum_p \frac{\partial a_{mp}}{\partial a_{ij}} x_p = \sum_p \delta_{mi}\ \delta_{pj} x_p = \delta_{mi} x_j$$
$$\frac{\partial u_m}{\partial b_{ij}} = \sum_p \frac{\partial b_{mp}}{\partial b_{ij}} e_p = \sum_p \delta_{mi}\ \delta_{pj} e_p = \delta_{mi} e_j$$

where δ is the Kronecker-delta defined as

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Therefore,

$$\dot{a}_{ij} = -\eta \sum_m \lambda_m f'_m(u_m) \delta_{mi} x_j = -\eta f'_i(u_i) \lambda_i x_j$$
$$\dot{b}_{ij} = -\eta \sum_m \lambda_m f'_m(u_m) \delta_{mi} e_j = -\eta f'_i(u_i) \lambda_i e_j$$

In matrix form,

$$\dot{A} = -\eta \Lambda_{f'(\mathbf{u})}\ \lambda\ \mathbf{x}^T$$
$$\dot{B} = -\eta \Lambda_{f'(\mathbf{u})}\ \lambda\ \mathbf{e}^T$$

where $\Lambda_{\mathbf{z}} = diag(z_1, z_2, \cdots, z_n)$

Table 5.5.1 presents some nonlinear functions and their corresponding Anti-hebbian like update rules for the matrices A and B.

Update Law Derivation for C and D

The independence criterion ϕ is a function of the output $\mathbf{y}$, $\phi = \phi(\mathbf{y})$

Let $\alpha = c$, d again. Then,

$$\dot{\alpha}_{ij} = \eta \frac{\partial \phi(y)}{\partial \alpha_{ij}}$$

Table 5.2. Different Nonlinearity functions and their derivatives

Function $f(\mathbf{u})$	Derivative $f'(\mathbf{u})$	Anti-Hebb for A	Anti-Hebb for B
$\mathbf{u}$	1	$\lambda\mathbf{x}^T$	$\lambda\mathbf{e}^T$
$\frac{1}{1+e^{-\mathbf{u}}}$	$f(\mathbf{u}) - f(\mathbf{u})^2$	$\Lambda_{f(\mathbf{u})-f(\mathbf{u})^2}\lambda\mathbf{x}^T$	$\Lambda_{f(\mathbf{u})-f(\mathbf{u})^2}\lambda\mathbf{e}^T$
$tanh(\mathbf{u})$	$1 - f(\mathbf{u})^2$	$\Lambda_{1-f(\mathbf{u})^2}\lambda\mathbf{x}^T$	$\Lambda_{1-f(\mathbf{u})^2}\lambda\mathbf{e}^T$
$arctan(\mathbf{u})$	$\frac{1}{1+\mathbf{u}^2}$	$\Lambda_{\frac{1}{1+\mathbf{u}^2}}\lambda\mathbf{x}^T$	$\Lambda_{\frac{1}{1+\mathbf{u}^2}}\lambda\mathbf{e}^T$
$e^{-\mathbf{u}^2}$	$-2\mathbf{u}f(\mathbf{u})$	$\Lambda_{-2\mathbf{u}f(\mathbf{u})}\lambda\mathbf{x}^T$	$\Lambda_{-2\mathbf{u}f(\mathbf{u})}\lambda\mathbf{e}^T$

$$
\begin{aligned}
&= \eta \sum_m \frac{\partial \phi(y)}{\partial y_m} \frac{\partial y_m}{\partial \alpha_{ij}} \\
&= \eta \sum_m \frac{\partial \phi(y)}{\partial y_m} \frac{\partial g_m(v_m)}{\partial \alpha_{ij}} \\
&= \eta \sum_m \frac{\partial \phi}{\partial y_m} g'_m(v_m) \frac{\partial v_m}{\partial \alpha_{ij}}
\end{aligned}
$$

Thus, we need to compute $\frac{\partial v_m}{\partial \alpha_{ij}}$:

$$
\frac{\partial v_m}{\partial c_{ij}} = \sum_p \frac{\partial c_{mp}}{\partial c_{ij}} x_p = \sum_p \delta_{mi}\, \delta_{pj} x_p = \delta_{mi} x_j
$$

$$
\frac{\partial v_m}{\partial d_{ij}} = \sum_p \frac{\partial d_{mp}}{\partial d_{ij}} e_p = \sum_p \delta_{mi}\, \delta_{pj} e_p = \delta_{mi} e_j
$$

Therefore,

$$
\begin{aligned}
\dot{c}_{ij} &= \eta \sum_m \frac{\partial \phi(y)}{\partial y_m} g'_m(v_m) \delta_{mi} x_j = \eta g'_i(v_i) \frac{\partial \phi}{\partial y_i} x_j \\
\dot{d}_{ij} &= \eta \sum_m \frac{\partial \phi(y)}{\partial y_m} g'_m(u_m) \delta_{mi} e_j = \eta g'_i(v_i) \frac{\partial \phi}{\partial y_i} e_j
\end{aligned}
$$

In matrix form,

$$
\dot{C} = \eta \Lambda_{g'(\mathbf{v})} \frac{\partial \phi}{\partial \mathbf{y}} \mathbf{x}^T
$$

$$\dot{D} = \eta \Lambda_{g'(\mathbf{v})} \frac{\partial \phi}{\partial \mathbf{y}} \mathbf{e}^T$$

Below, two simple independence criteria are listed.

1. $\phi(\mathbf{y}) = \frac{1}{2}\sum_i y_i^2 = \frac{1}{2}\mathbf{y}^T\mathbf{y}$. Therefore,

$$\dot{c}_{ij} = \eta g_i'(v_i) y_i x_j \tag{5.39}$$

$$\dot{d}_{ij} = \eta g_i'(v_i) y_i e_j \tag{5.40}$$

2. $\phi(y) = \frac{1}{4}\sum_i y_i^4$. Therefore,

$$\dot{c}_{ij} = \eta g_i'(v_i) y_i^3 x_j$$

$$\dot{d}_{ij} = \eta g_i'(v_i) y_i^3 e_j$$

Adjoint State Equation Derivation

Now, the adjoint state equation will be computed.

$$\begin{aligned}
\left[\left(\frac{\partial f}{\partial \mathbf{x}}\right)^T\right]_{ij} &= \frac{\partial f_j(u_j)}{\partial x_i} \\
&= f_j'(u_j)\frac{\partial u_j}{\partial x_i} \\
&= f_j'(u_j)\sum_m a_{jm}\frac{\partial x_m}{\partial x_i} \\
&= f_j'(u_j)\sum_m a_{jm}\delta_{mi} \\
&= f_j'(u_j)a_{ji} \\
&= \left[A^T \Lambda_{f'(\mathbf{u})}\right]_{ij}
\end{aligned}$$

$$\begin{aligned}
\left[\frac{\partial \phi(y)}{\partial \mathbf{x}}\right]_i &= \frac{\partial \phi(y)}{\partial x_i} \\
&= \sum_j \frac{\partial \phi}{\partial y_j}\frac{\partial y_j}{\partial x_i}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_j \frac{\partial \phi}{\partial y_j} g_j'(v_j) \frac{\partial v_j}{\partial x_i} \\
&= \sum_j \frac{\partial \phi}{\partial y_j} g_j'(v_j) c_{ji} \\
&= \left[C^T \Lambda_{g'(\mathbf{v})} \frac{\partial \phi}{\partial \mathbf{y}} \right]_i
\end{aligned}
$$

Thus, in compact form:

$$\dot{\lambda} = -A^T \Lambda_{f'(\mathbf{u})} \lambda + C^T \Lambda_{g'(\mathbf{v})} \frac{\partial \phi}{\partial \mathbf{y}}$$

In conclusion, given a network whose states follow the dynamics described by

$$\dot{\mathbf{x}} = f(A\mathbf{x} + B\mathbf{e}), \qquad \mathbf{x}_0 = \mathbf{x}(t_0) \tag{5.41}$$

and whose output is

$$\mathbf{y} = g(C\mathbf{x} + D\mathbf{e}) \tag{5.42}$$

The dynamics of the adjoint system are described by

$$\dot{\lambda} = -A^T \Lambda_{f'(\mathbf{u})} \lambda + C^T \Lambda_{g'(\mathbf{v})} \frac{\partial \phi}{\partial \mathbf{y}} \tag{5.43}$$

The parameters are updated according to:

$$\dot{A} = -\eta \Lambda_{f'(u)} \lambda \, x^T \tag{5.44}$$

$$\dot{B} = -\eta \Lambda_{f'(u)} \lambda \, e^T \tag{5.45}$$

$$\dot{C} = \eta \Lambda_{g'(v)} \frac{\partial \phi}{\partial y} x^T \tag{5.46}$$

$$\dot{D} = \eta \Lambda_{g'(v)} \frac{\partial \phi}{\partial y} e^T \tag{5.47}$$

5.5.2 Linear Dynamical Case

The derivation of the update laws for this model is a special case of the nonlinear dynamic model discussed in the previous section. In this case, the functions f and g in equations (5.41) and (5.42) are the identity functions. Thus, their derivatives are the identity matrices. Consequently, equation (5.43) is simplified to:

$$\dot{\lambda} = -A^T\lambda + C^T\frac{\partial\phi}{\partial \mathbf{y}} \tag{5.48}$$

and (5.44)-(5.47) are simply:

$$\dot{A} = -\eta\lambda\, x^T \tag{5.49}$$

$$\dot{B} = -\eta\, \lambda\, e^T \tag{5.50}$$

$$\dot{C} = \eta\frac{\partial\phi}{\partial y}x^T \tag{5.51}$$

$$\dot{D} = \eta\frac{\partial\phi}{\partial y}e^T \tag{5.52}$$

In order to compute the term $\frac{\partial\phi}{\partial \mathbf{y}}$, one can make direct computation as was pursued in Chapter 4 for the static case. One, however, can define new variables and transform the problem into one that is similar to the static case. This will be accomplished by considering the output equation of the linear dynamical system

$$\mathbf{y} = C\mathbf{x} + D\mathbf{e} \tag{5.53}$$

Then, define the vectors $\tilde{y}$ and $\tilde{x}$, and the matrix $\tilde{W}$ as

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} \qquad \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{e} \\ \mathbf{x} \end{bmatrix} \qquad \tilde{\mathbf{W}} = \begin{bmatrix} D & C \\ 0 & I \end{bmatrix} \tag{5.54}$$

However,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} D & C \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{x} \end{bmatrix} \tag{5.55}$$

Therefore, equation (5.53) can be written as

$$\tilde{\mathbf{y}} = \tilde{W}\tilde{\mathbf{x}} \tag{5.56}$$

Consequently, the update law for the parameters of the matrix $\tilde{W}$ is

$$\dot{\tilde{W}} = \eta\left[\tilde{W}^{-T} - f_\alpha(\tilde{\mathbf{y}})\tilde{\mathbf{x}}^T\right] \qquad \alpha \in \{a,\ b,\ c,\ d\} \tag{5.57}$$

First, one needs to find $\tilde{W}^{-T}$

$$\tilde{\mathbf{W}}^{-T} = \begin{bmatrix} D(D^TD)^{-1} & 0 \\ -C^TD(D^TD)^{-1} & I \end{bmatrix} \tag{5.58}$$

since

$$\begin{bmatrix} D^T & 0 \\ C^T & I \end{bmatrix} \begin{bmatrix} D(D^TD)^{-1} & 0 \\ -C^TD(D^TD)^{-1} & I \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{5.59}$$

Consequently, the update law for the matrix C

$$\dot{C} = -\eta f_\alpha(\mathbf{y})\mathbf{x}^T \tag{5.60}$$

and that of matrix D is

$$\dot{D} = \gamma\left[D(D^TD)^{-1} - f_\alpha(\mathbf{y})\mathbf{e}^T\right] \tag{5.61}$$

If D is a square nonsingular matrix, then equation (5.61) becomes

$$\dot{D} = \gamma\left[D^{-T} - f_\alpha(\mathbf{y})\mathbf{e}^T\right] \cdot \tag{5.62}$$

$$= \gamma\left[I - f_\alpha(\mathbf{y})\mathbf{y}^T\right]D^{-T} \tag{5.63}$$

5.6 Computer Simulations

Computer simulations of the update laws of the parameters of the matrices C and D, as defined by equation (5.60) and (5.62), were performed. The considered unknown sources are two sine waveforms with respective frequencies $1Hz$ and $2Hz$. The environment model is the dynamical system represented by the following transfer matrix

$$\bar{H}(s) = \begin{bmatrix} 1.0\frac{s^2+8s+15}{s^2+3s+2} & 0.6\frac{s+1}{s+3} \\ 0.2\frac{s^2+2s+3}{s^2+9s+20} & 1.0\frac{s^2+6s+8}{s^2+13s+42} \end{bmatrix}^{-1} = H^*(s)^{-1} \tag{5.64}$$

One theoretical solution to the problem is the transfer function $H^*(s)$ whose canonical state representation, $\mathcal{D}^* = (A^*,\ B^*,\ C^*,\ D^*)$, is defined by

$$A^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -20 & -9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -42 & -13 \end{bmatrix} \qquad B^* = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C^* = \begin{bmatrix} 13 & 5 & -1.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.6 & -1.2 & -34 & -7.0 \end{bmatrix} \qquad D^* = \begin{bmatrix} 1.0 & 0.6 \\ 0.2 & 1.0 \end{bmatrix}$$

The bode plots of the environment, defined by $\bar{H}(s)$, and the desired realization of the network model, defined by $H^*(s)$, are shown respectively in figures 5.3 and 5.4.

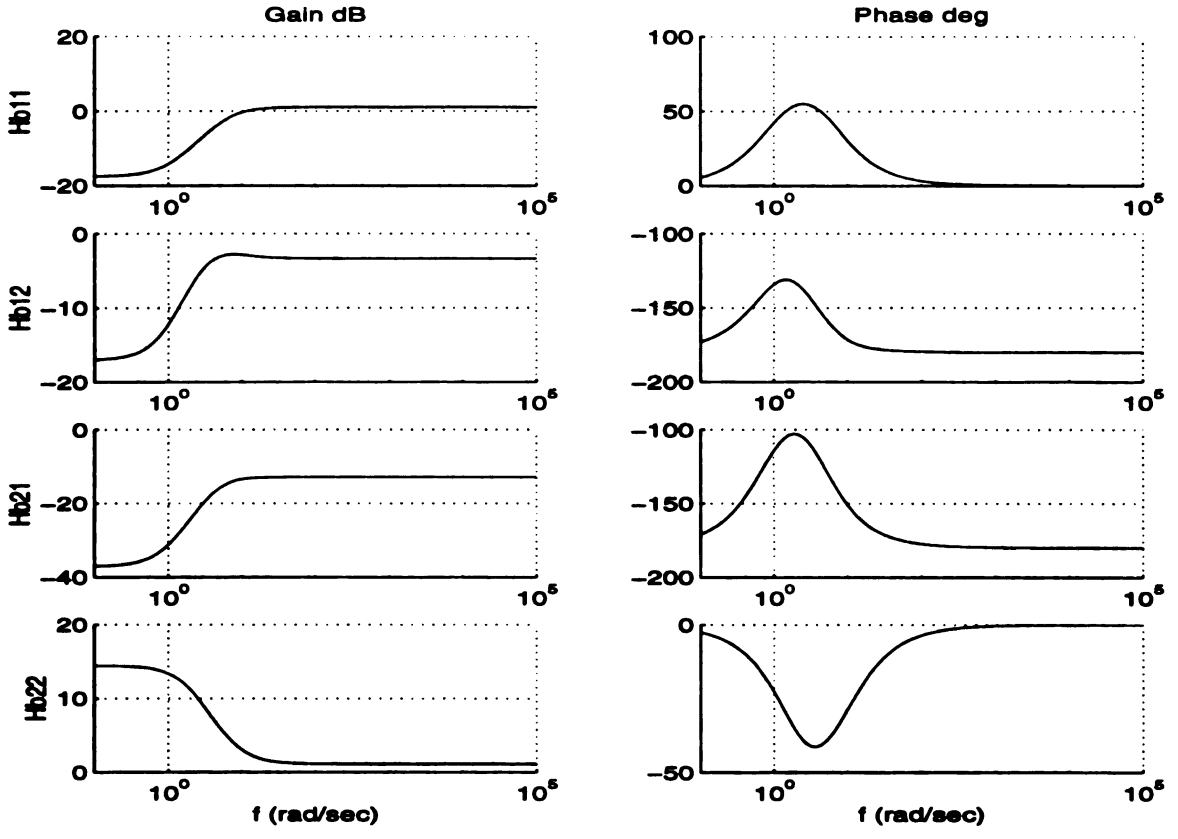


Figure 5.3. Bode plot of the environment model defined by $\bar{H}(s)$

First, computer simulations were performed by updating only the parameters of one matrix while holding the parameters of the other matrix at their nominal values, as defined by $\mathcal{D}^*$. In either experiments, the algorithm failed to converge to the desired solution. Figures 5.5 and 5.6 show the the algorithm fails to converge to the desired solution. Figure 5.5 shows the performance of the algorithm by updating the parameters of the D matrix while those of the C matrix are held at their nominal values. On the other hand, Figure 5.6 shows the performance of the algorithm by

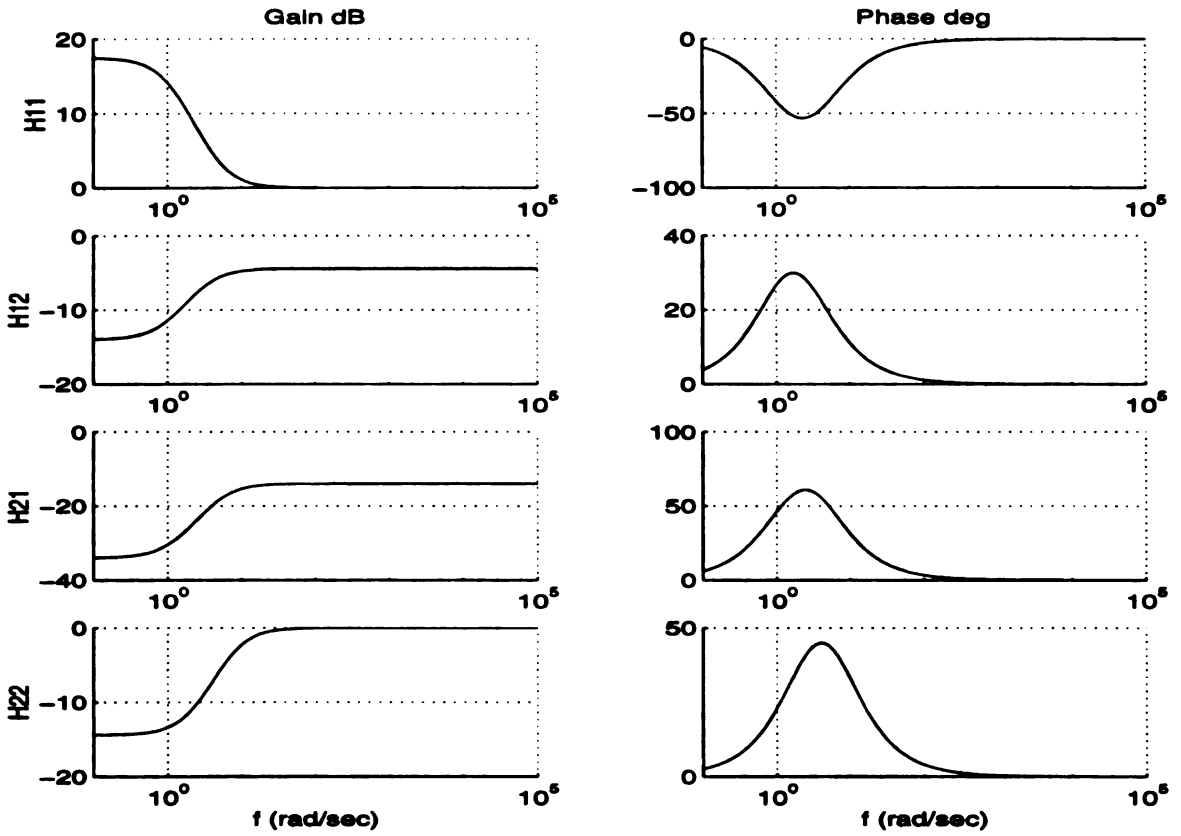


Figure 5.4. Bode plot of the network model defined by $H^*(s)$

updating only the C matrix and fixing the parameters of the D matrix to their nominal values. Observe that the algorithm fails to converge in both simulations. The algorithm also fails to accomplish the convergence by considering the nonlinear functions f_b, f_c, f_d as well as the cubic and the sine hyperbolic ones. Figure 5.7 shows a case of simulations when the nonlinear function is f_d. The reason for the failure of the algorithm is the fact that the problem is not completely identifiable. Recall that from equations (5.7)-(5.10), the system has infinitely many solutions that it could converge to. The solution defined by $\mathcal{D}^* = (A^*,\ B^*,\ C^*,\ D^*)$ is one particular solution. The realization $\mathcal{D}_P^* = (A^*,\ B^*,\ PC^*,\ PD^*)$ is also another possible solution where P is generalized permutation matrix. Thus, by keeping one matrix constant at its nominal value, one is limiting the set of all possible solutions to one particular point in the set of equilibrium points.

Next, computer simulations were performed by considering the simultaneous update of both parameters of the matrices C and D. Figure 5.8 shows the respective performance of when two sine functions are assumed to be the unknown sources. One observes that the all the parameters converge, but not to the desired values.

On the other hand, if one considers an update law for the matrix D defined by (4.56)

$$\dot{D} = -\gamma f_\alpha(\mathbf{y})\mathbf{y}^T, \qquad \alpha \in \{a,\ b,\ c,\ d\} \tag{5.65}$$

the algorithm demonstrated its capability to converge only when the nonlinear function is defined by f_a. Figure 5.9 and 5.10 are samples of such simulations. One can observe the convergence of the matrix D in Figure 5.9 as all its parameters become equal to the desired values in a finite time. On the other hand, the parameters of the D matrix do converge in Figure 5.10, but not to the desired values. Figure 5.11 shows the test simulation of the algorithm defined by equation (5.65) using the nonlinear

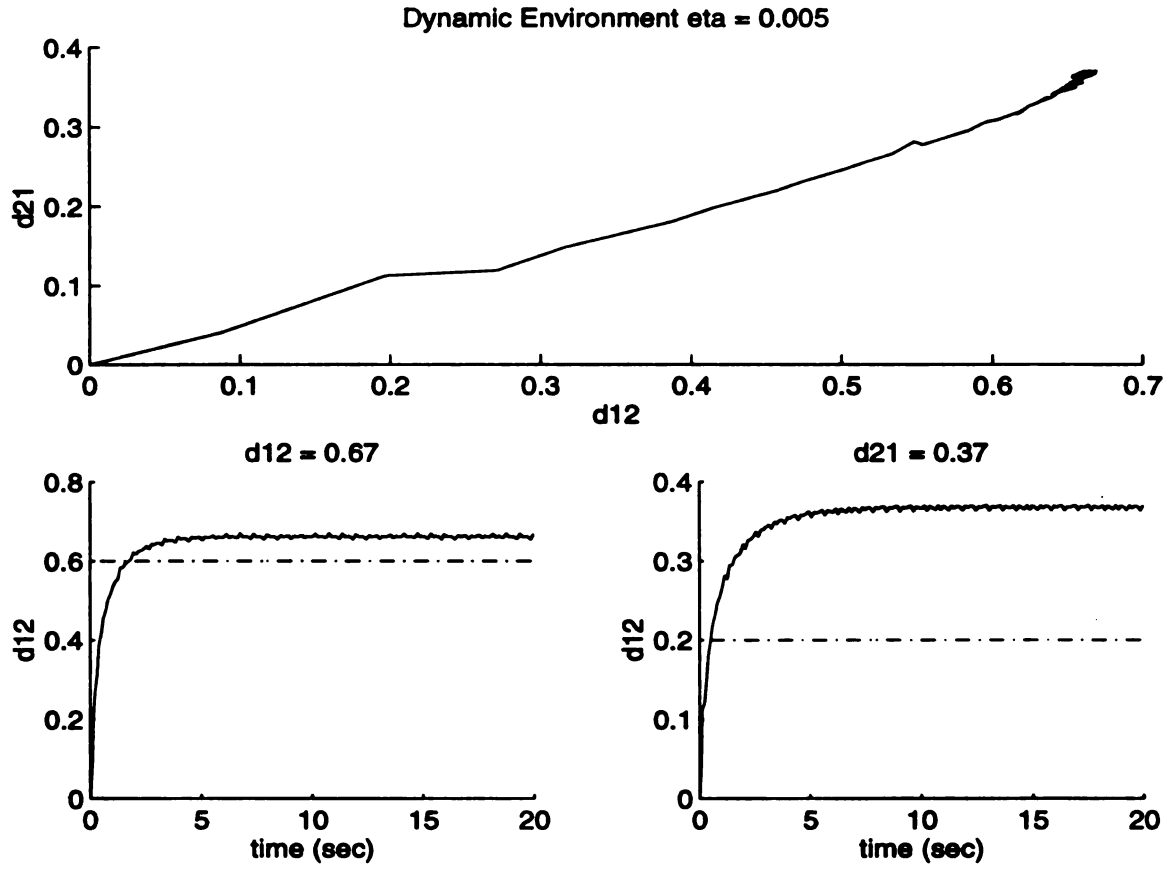


Figure 5.5. Dynamic Forward Structure. Update only the D matrix according to $\dot{D} = \gamma\left[D^{-T} - f_a(\mathbf{y})\mathbf{e}^T\right]$ using two sine waveforms, $\gamma = 0.005$ and $D_0 = 0$

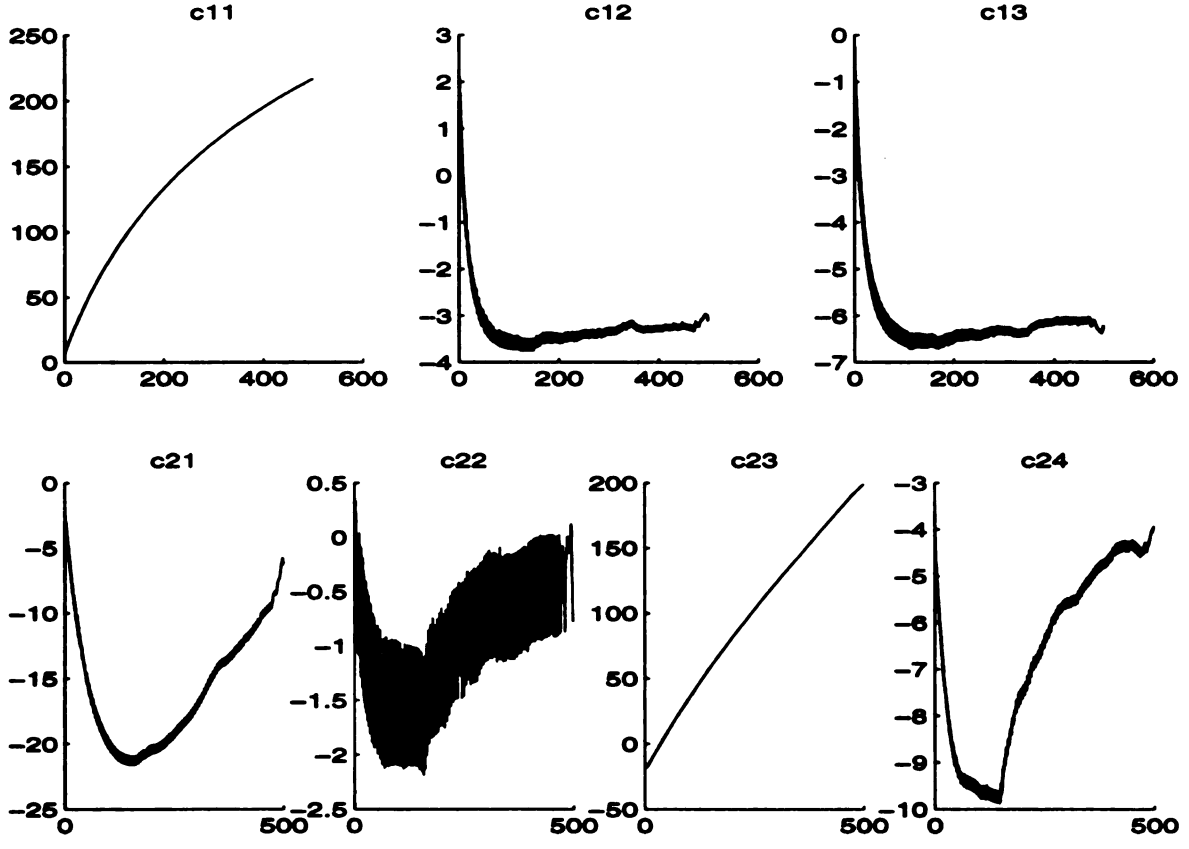


Figure 5.6. Dynamic Forward Structure. Update only the C matrix according to $\dot{C} = -\eta f_a(\mathbf{y})\mathbf{x}^T$ using two sine waveforms, η = [1000 100 1 100 1 100 10] and $C_0 = 0.8C^*$

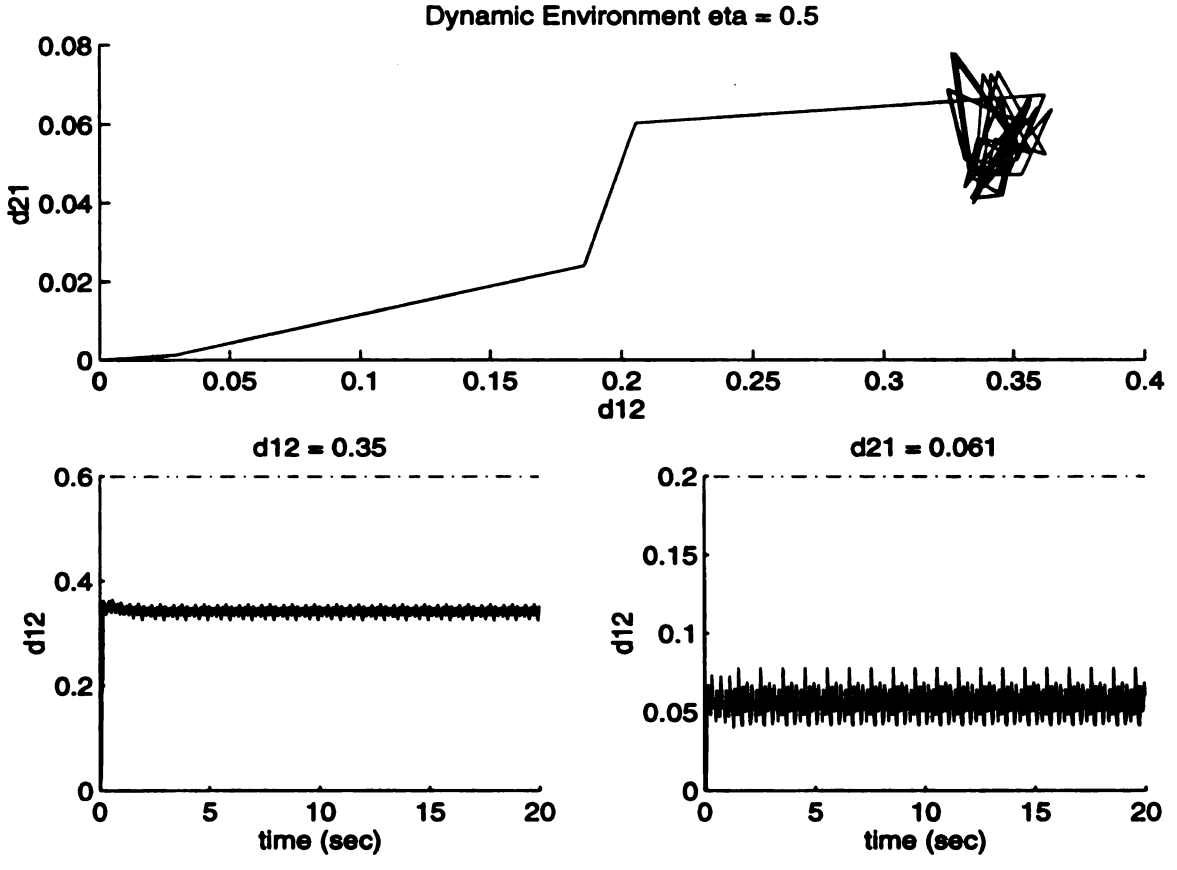


Figure 5.7. Dynamic Forward Structure. Update only the D matrix according to $\dot{D} = \gamma\left[D^{-T} - f_d(\mathbf{y})\mathbf{e}^T\right]$ using two sine waveforms, $\gamma = 0.5$ and $D_0 = 0$

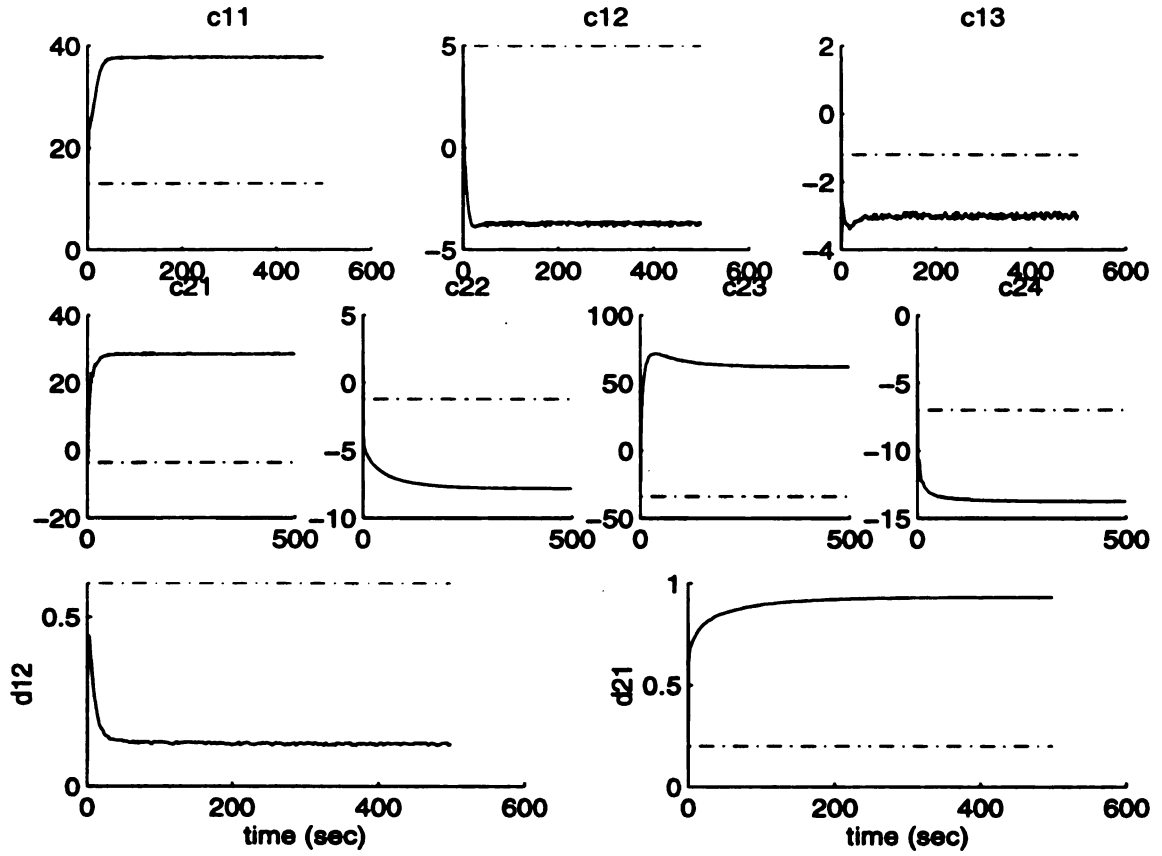


Figure 5.8. Dynamic Forward Structure. Update both the C and D matrices according to $\dot{D} = \gamma\left[D^{-T} - f_a(\mathbf{y})\mathbf{e}^T\right]$ and $\dot{C} = -\eta f_a(\mathbf{y})\mathbf{x}^T$ using two sine waveforms, $\gamma = 0.005$, $\eta = [100\ 10\ 1\ 1000\ 5\ 2000\ 10]$. All initial conditions are zero

function f_a at convergence. Obeserve that the output of the network is almost an exact replica of the unknown sources.

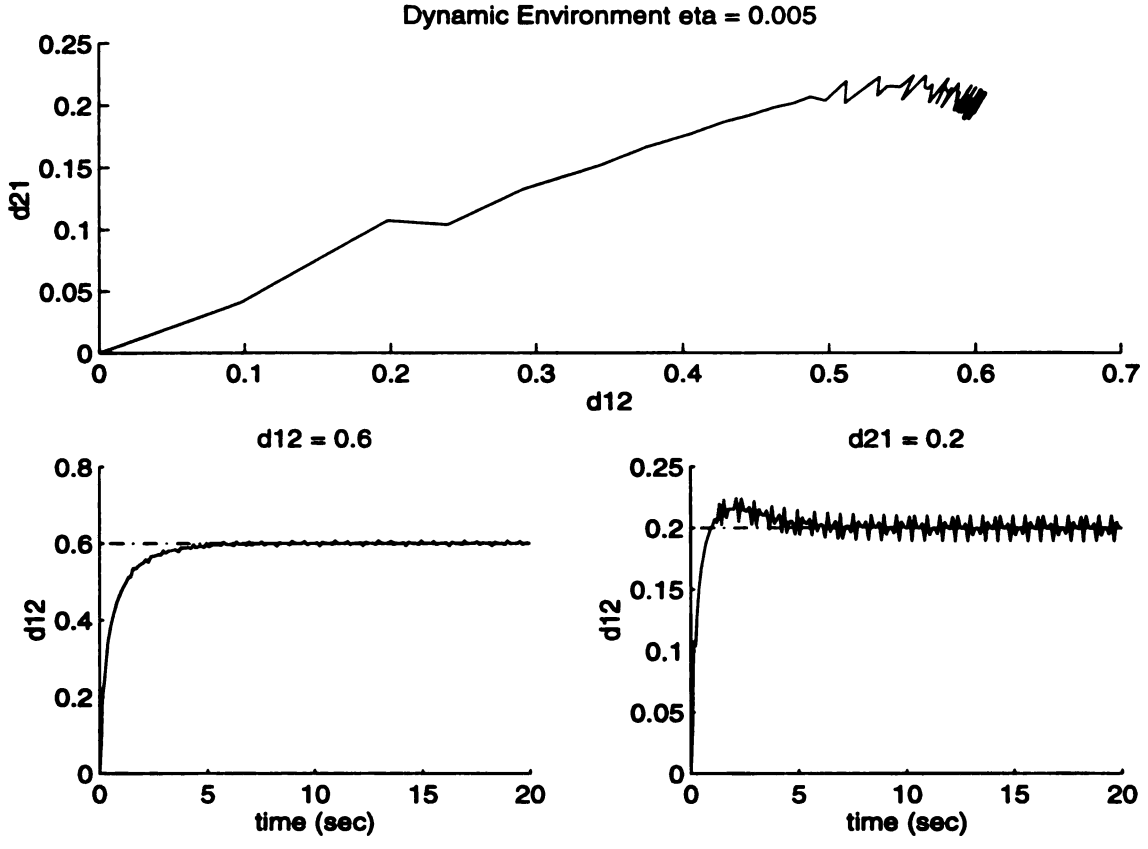


Figure 5.9. Dynamic Forward Structure. Update only the D matrix according to $\dot{D} = -\gamma f_a(\mathbf{y})\mathbf{y}^T$ using two sine waveforms, $\gamma = 0.005$ and $D_0 = 0$

Other nonlinear function, reported in [8], were also considered. Computer simulation of the following algorithms were performed

$$\dot{D} = -\gamma \mathbf{y}^3 \mathbf{y} \tag{5.66}$$

$$\dot{D} = -\gamma \sinh \mathbf{y} \tanh \mathbf{y} \tag{5.67}$$

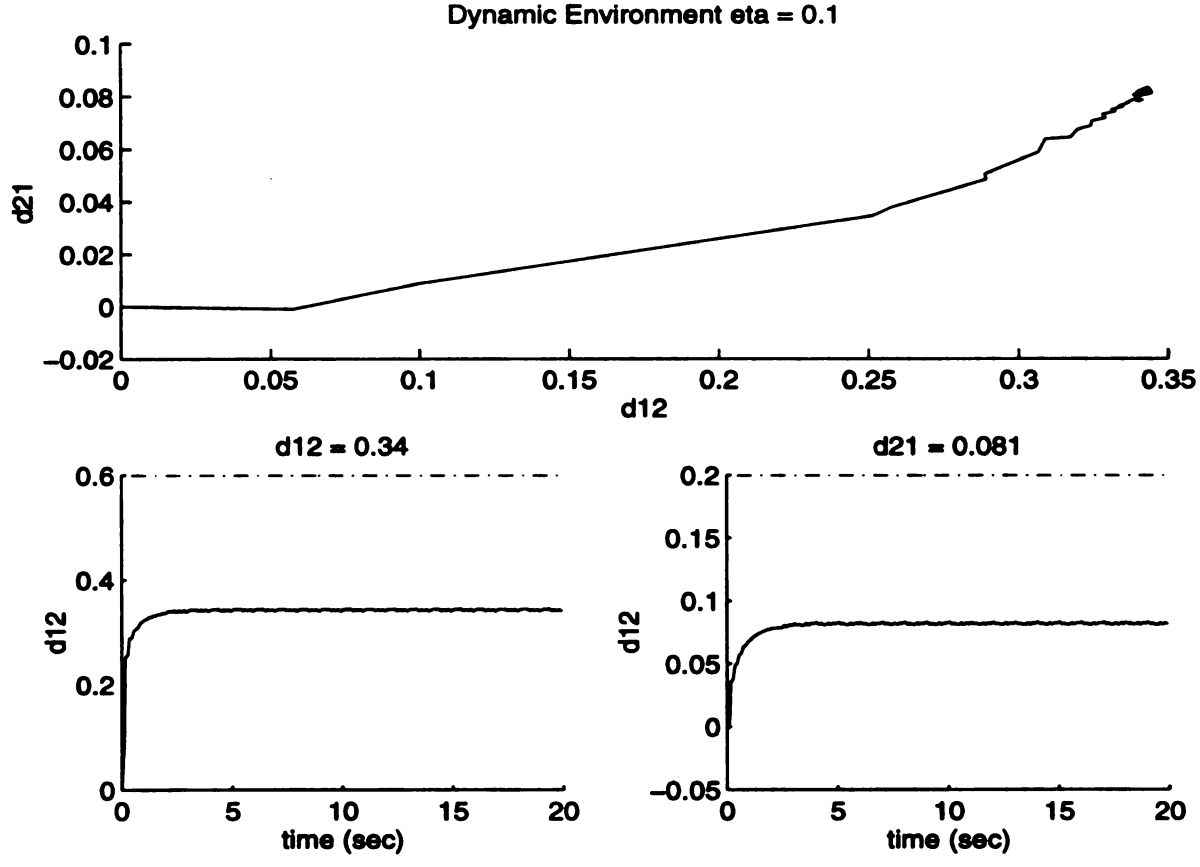


Figure 5.10. Dynamic Forward Structure. Update only the D matrix according to $\dot{D} = -\gamma f_d(\mathbf{y})\mathbf{y}^T$ using two sine waveforms, $\gamma = 0.5$ and $D_0 = 0$

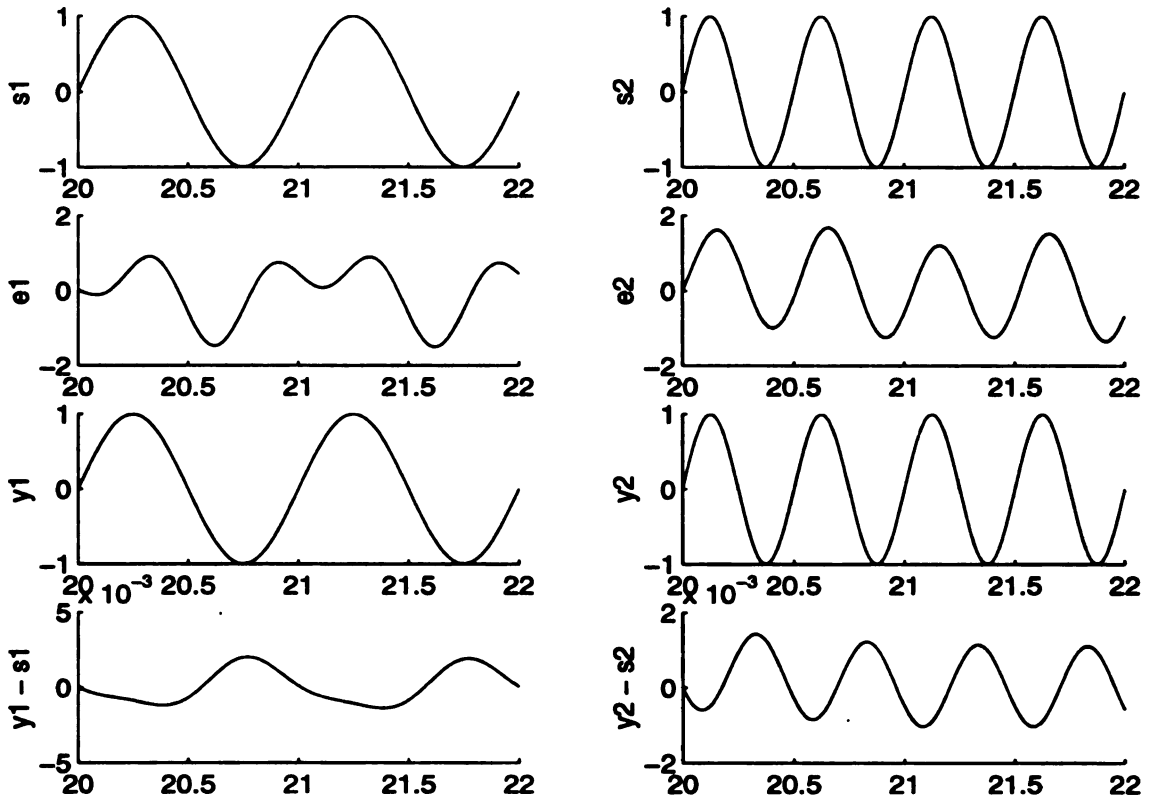


Figure 5.11. Dynamic Forward Structure. Test Simulation of the update law $\dot{D} = -\gamma f_a(\mathbf{y})\mathbf{y}^T$ at convergence using two sine waveforms

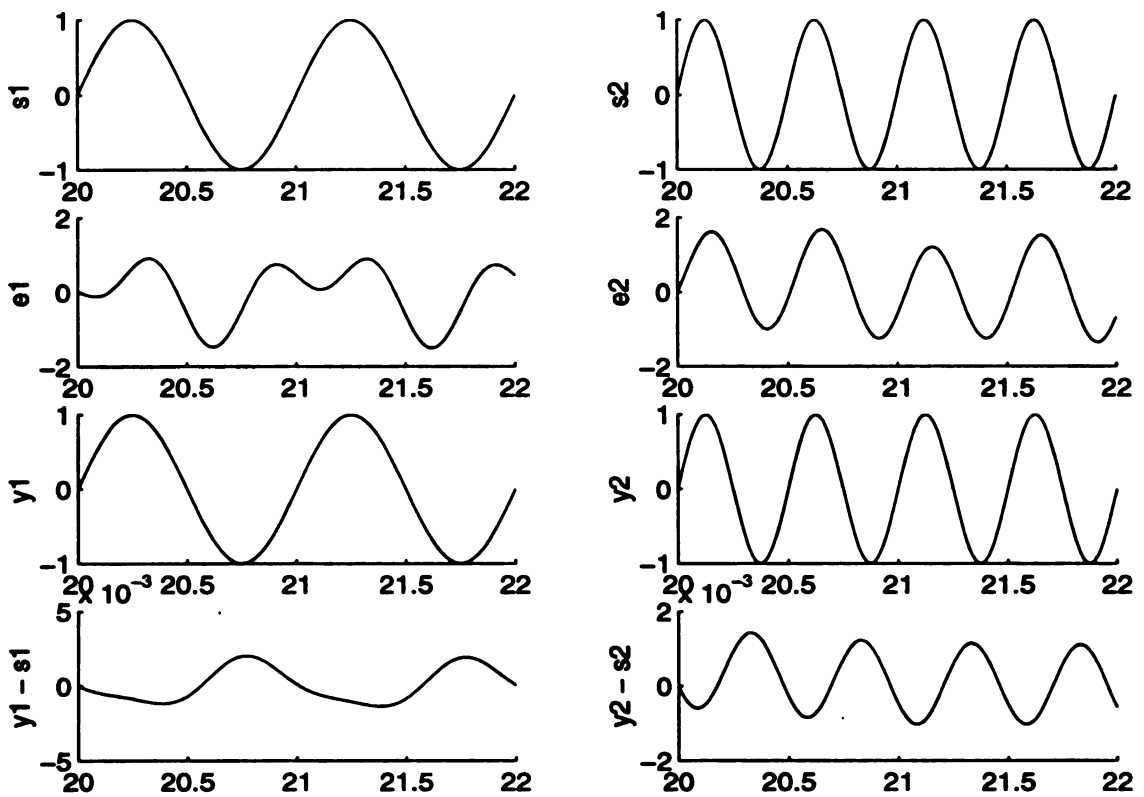


Figure 5.11. Dynamic Forward Structure. Test Simulation of the update law $\dot{D} = -\gamma f_a(\mathbf{y})\mathbf{y}^T$ at convergence using two sine waveforms

Figures 5.12 and 5.13 show their performance. Observe that both nonlinear functions are capable of separating signals. The reason of exploring the algorithms defined by equations (5.66) and (5.67) is to show that the nonlinear function f_a performs as equal to other nonlinear functions that have shown to provide excellent performance, but that were chosen heuristically[8]. This is in contrast to the function f_a that was developed based on a well defined energy function and independence criterion. However, the function f_d, reported in [24], did not provide the same performance. This is due to the poor approximation of entropy approximation and/or the assumption of the unit variance.

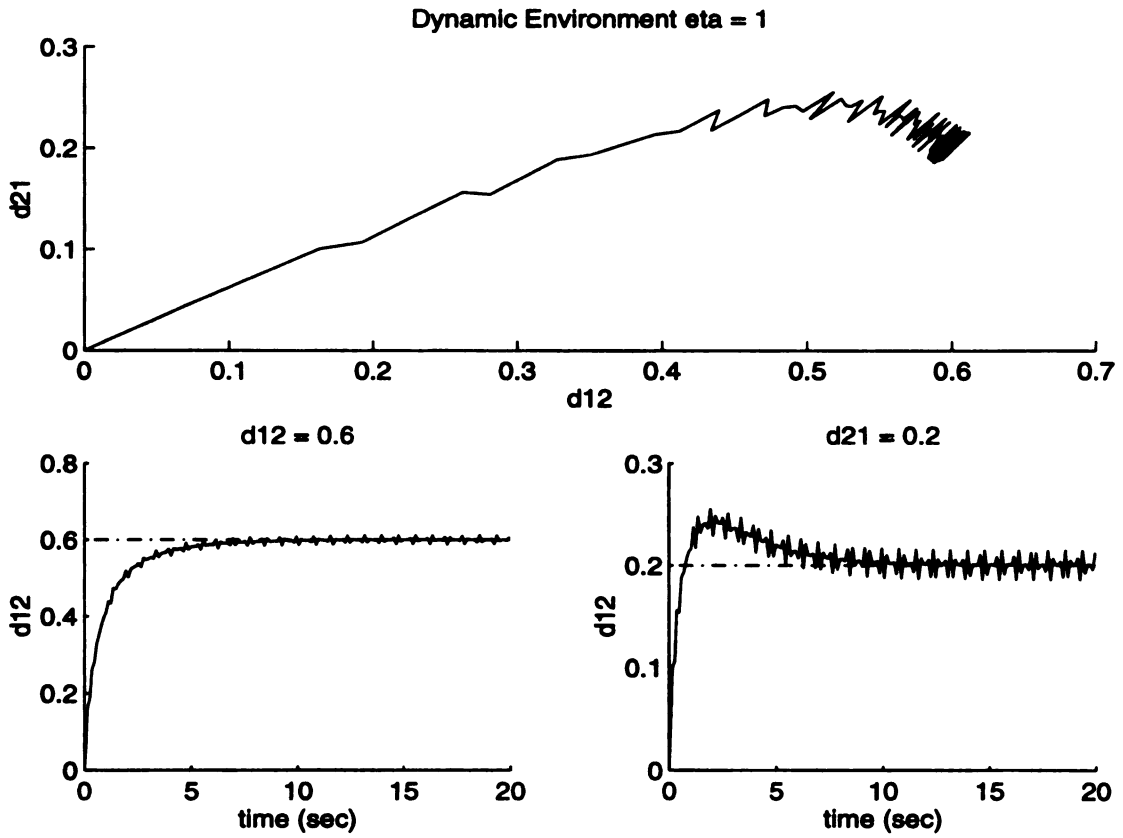


Figure 5.12. Update of the D Matrix using $\dot{D} = -\gamma \mathbf{y}^3\mathbf{y}$

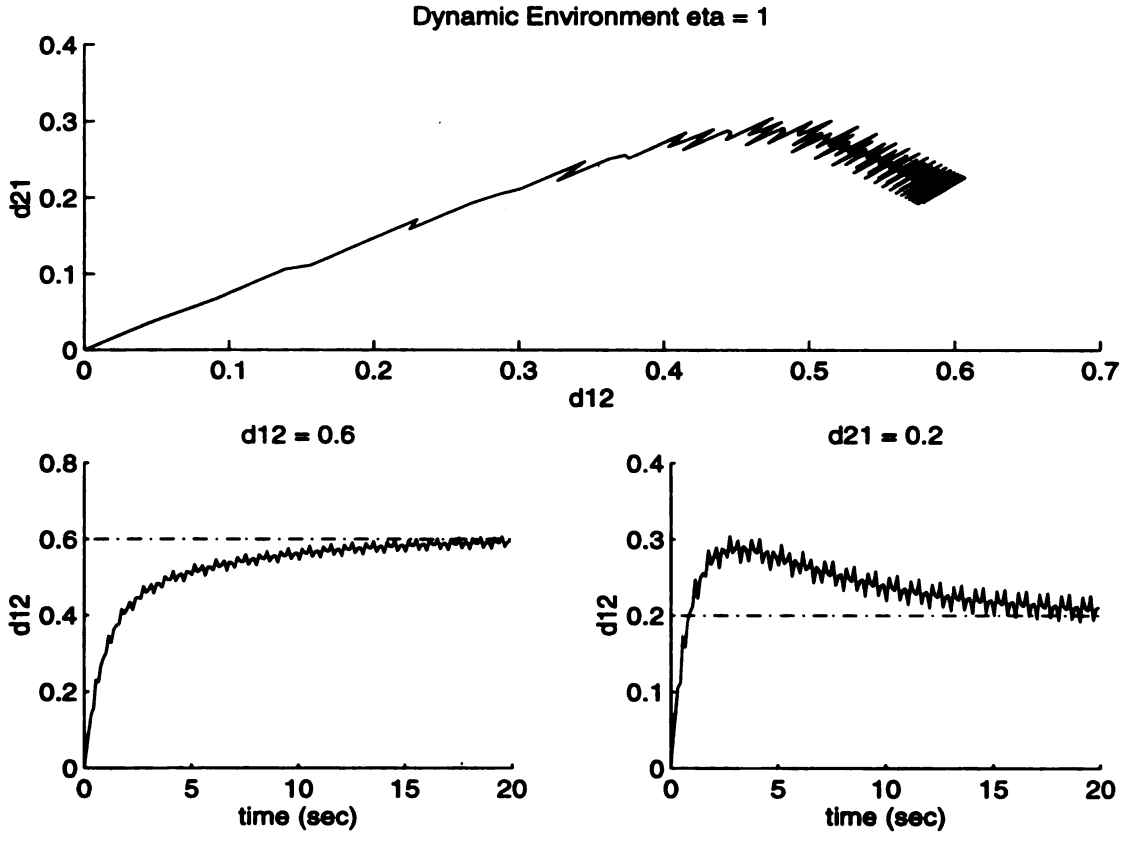


Figure 5.13. Update of the D Matrix using $\dot{D} = -\gamma \sinh \mathbf{y} \tanh \mathbf{y}$

5.7 Observations

Despite the limited results that one obtained in this chapter, a general framework to derive an update law was developed. Such a framework can be applied to any network structure. It was also observed that the introduced forward structure , in this chapter, is subject to an identifiability problem due the freedom of the possible solutions that the network could converge to. Thus, the task is to define an identifiable structure and to use the herein developed tools to define the update law for that structure. It is intended to do so in the following chapter.

CHAPTER 6

Blind Separation in a Dynamic Environment: Feedback Structure

In this chapter, an architecture different from the one discussed in the previous chapter is considered. It does not represent a particular network architecture, nor does it limit the type of environments that can be considered. In this research, however, it will be shown that any type of environment can be considered. The state space representation of the network will be presented and the existence of a theoretical solution to the problem shown. Then, an update law for the parameters of the network based on an analogy of the static case will be developed.

6.1 Architecture

General systems are defined by their input-output relationship. For time-invariant systems, such a relation is described by a transfer function $L(s)$ which relates the input $S(s)$ to the output $E(s)$

$$E(s) = L(s)S(s) \tag{6.1}$$

If $\mathbf{s} \in \mathcal{R}^r$ and $\mathbf{e} \in \mathcal{R}^m$, then $L(s)$ is an $m \times r$ matrix

$$L(s) = \begin{bmatrix} L_{11} & \cdots & L_{1i} & \cdots & L_{1r} \\ \vdots & \ddots & & & \vdots \\ L_{i1} & \cdots & L_{ii} & \cdots & L_{ir} \\ \vdots & & & \ddots & \vdots \\ L_{m1} & \cdots & L_{mi} & \cdots & L_{mr} \end{bmatrix} \tag{6.2}$$

which can be expressed as

$$L(s) = \begin{bmatrix} 1 & \cdots & \bar{H}_{1i} & \cdots & \bar{H}_{1r} \\ \vdots & \ddots & & & \vdots \\ \bar{H}_{i1} & \cdots & 1 & \cdots & \bar{H}_{ir} \\ \vdots & & & \ddots & \vdots \\ \bar{H}_{m1} & \cdots & \bar{H}_{mi} & \cdots & 1 \end{bmatrix} \begin{bmatrix} L_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & \cdots & L_{ii} & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & L_{mr} \end{bmatrix} \tag{6.3}$$

where

$$\bar{H}_{ij} = \frac{L_{ij}}{L_{ii}} \tag{6.4}$$

Let's denote $\tilde{L}(s)$ to be the diagonal matrix of $L(s)$. Thus, equation (6.1) is equivalent to the following two equations:

$$E(s) = \bar{H}(s)\tilde{S}(s) \tag{6.5}$$

and

$$\tilde{S}(s) = \tilde{L}(s)S(s) \tag{6.6}$$

Therefore, the system described by equation (6.1) is equivalent to the cascaded processing of two systems described by equations (6.5) and (6.6). Equation (6.5) performs

the pre-filtering process, while equation (6.6) performs the mixing and/or filtering process that describes the model of the environment.

In the blind separation of signals, the goal is to recover the unknown sources. This requires the development of a network that performs the inverse function of the pre-filtering and mixing/filtering processes. Thus, such a network is also a cascade of two processes. The first one will be devoted to reverse the mixing and/or filtering process, while the second one will be devoted to the inverse filtering. Let's denote $\tilde{\mathbf{y}}$ and $\mathbf{y}$ to be the respective outputs of these processes. A feedback structure that models the inverse process of the mixing and/or filtering is as follows

$$\tilde{Y}(s) = E(s) - H(s)\,\tilde{Y}(s) \tag{6.7}$$

where $H(s)$ is a matrix with zero diagonal entries. Thus, the goal of such a process is to recover a replica of the $\tilde{\mathbf{s}}$. Therefore, using equations (6.5) and (6.7), the following equation is obtained:

$$I + H(s) = \bar{H}(s) \tag{6.8}$$

Thus, with an appropriate update law of the parameters of the matrix transfer function $H(s)$, the problem has the solution

$$H_{ij}(s) = \bar{H}_{ij}(s) \qquad \forall i \neq j \tag{6.9}$$

Finally, to recover the original signal $\mathbf{s}$, some filter $K(s)$ will be applied to $\tilde{\mathbf{y}}$, which would result in an output

$$Y(s) = K(s)\tilde{\mathbf{y}} \tag{6.10}$$

$$= K(s)\tilde{\mathbf{s}} \tag{6.11}$$

$$= K(s)\tilde{L}(s)S(s) \tag{6.12}$$

If the goal is to recover $S(s)$, then

$$K(s) = \tilde{L}(s)^{-1} \tag{6.13}$$

However, no knowledge of the $\tilde{L}(s)$ is available. Thus, one could suffice with a solution to the problem that is up to a filter of the unknown sources. Figure 6.1 shows a schematic diagram of these processes.

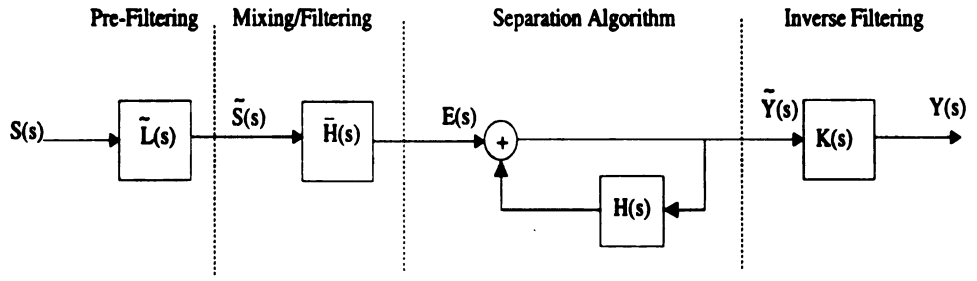


Figure 6.1. Feedback Processing Structure

6.2 State Representation

In the last chapter, the canonical controllable realization to define the state space realization of a given transfer function matrix was considered. Such realization will also be used here to model the network. However, one should keep in mind that the transfer matrix $H(s)$ has zero diagonal entries. Therefore, the state space representation of $H_{ii}(s)$ is $\mathcal{D}_{ii} = (0, 0, 0, 0)$. Consequently, the i^{th} entries of the matrices A_i,

$\mathbf{b}_i$, $\mathbf{c}_i^T$ and $\mathbf{d}_i^T$, as defined in equations (5.19) and (5.19), are null. Despite the fact that these entries are redundant to the network model, they will be preserved for the sake of clarity of the presentation of the model.

Here is a canonical controllable realization $\mathcal{D} = (A, B, C, D)$ for $H(s)$. Such a realization is defined by equations (5.21) and (5.21). The state equation is described by

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{y} \tag{6.14}$$

while the output equation is described by

$$\mathbf{y} = \mathbf{e} - C\mathbf{x} - D\mathbf{y} \tag{6.15}$$

Having defined the structure and the realization of the network model, the update laws for the parameters C and D are now ready to be developed.

6.3 Update Law Derivation

To develop the update law for the feedback structure, one needs first to define the vectors $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{e}}$ and the matrix $\tilde{W}$ as

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} \qquad \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{e} \\ \mathbf{0} \end{bmatrix} \qquad \tilde{\mathbf{W}} = \begin{bmatrix} D & C \\ 0 & I \end{bmatrix} \tag{6.16}$$

However,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{e} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} D & C \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} \tag{6.17}$$

Therefore, equation (6.15) can be rewritten as

$$\tilde{\mathbf{y}} = \tilde{\mathbf{e}} - \tilde{W}\tilde{\mathbf{y}} \tag{6.18}$$

Using Theorem 2, the parameters of the matrix $\tilde{W}$ are updated according to

$$\dot{\tilde{W}} = \eta f_\alpha(\tilde{\mathbf{y}})(\tilde{\mathbf{y}})^T \qquad \alpha \in \{a,\ b,\ c,\ d\} \tag{6.19}$$

This will reduce to

$$\dot{C} = \eta f_\alpha(\mathbf{y})\mathbf{x}^T \tag{6.20}$$

$$\dot{D} = \gamma f_\alpha(\mathbf{y})\mathbf{y}^T \tag{6.21}$$

where η and γ are respectively the learning rates of the matrices C and D.

6.4 Computer Simulation Results

To study the performance of the algorithm and the feedback structure, one needs to consider two approaches depending on the origin of the nonlinear function. Two different approaches will be considered. One approach will be based on the nonlinear functions f_a and f_d, which were developed in Chapter 4, based on the mutual information functional. The other will based on selecting the nonlinear functions reported in [8]. These two parts define functions that were developed based on the mutual information approach and others that were based on the neuromemitic approach. This comparative study will be considered as a mean to quantify the performance of the developed nonlinear function f_a vis-à-vis other existing nonlinear functions.

Computer simulations were conducted for the proposed architecture. A two dimensional network was considered. The two unknown sources were filtered and mixed

according to the following transfer function defined by

$$\bar{H}(s) = \begin{bmatrix} 1 & \frac{0.6(s+1)(s+4)}{(s+6)(s+7)} \\ \frac{0.4(s+2)(s+3)}{(s+5)(s+8)} & 1 \end{bmatrix} = \begin{bmatrix} 1 & \bar{H}_{12}(s) \\ \bar{H}_{21}(s) & 1 \end{bmatrix}$$

Therefore, the canonical controllable realization is $\bar{\mathcal{D}} = (\bar{A}, \bar{B}, \bar{C}, \bar{D})$ defined below as

$$\bar{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -42 & -13 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -40 & -13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \bar{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\bar{C} = \begin{bmatrix} 0 & -22.8 & -4.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & -13.6 & -3.2 & 0 \end{bmatrix} \qquad \bar{D} = \begin{bmatrix} 1.0 & 0.6 \\ 0.4 & 1.0 \end{bmatrix}$$

The bode plots of $\bar{H}_{12}(s)$ and $\bar{H}_{21}(s)$ are shown in Figure 6.2. Also, the bode plot of $H(s) = \bar{H}(s)^{-1}$ is shown in Figure 6.3.

6.4.1 The Approach Based on Mutual Information

The algorithm defined by equations (6.20) and (6.21) will be studied by exploring the nonlinear functions f_a and f_d.

Algorithm defined by f_a

Computer simulations of the algorithm defined by the equations (6.20) and (6.21) and the nonlinear function f_a was studied.

$$\dot{C} = \eta f_a(\mathbf{y})\mathbf{x}^T \tag{6.22}$$

$$\dot{D} = \gamma f_a(\mathbf{y})\mathbf{y}^T \tag{6.23}$$

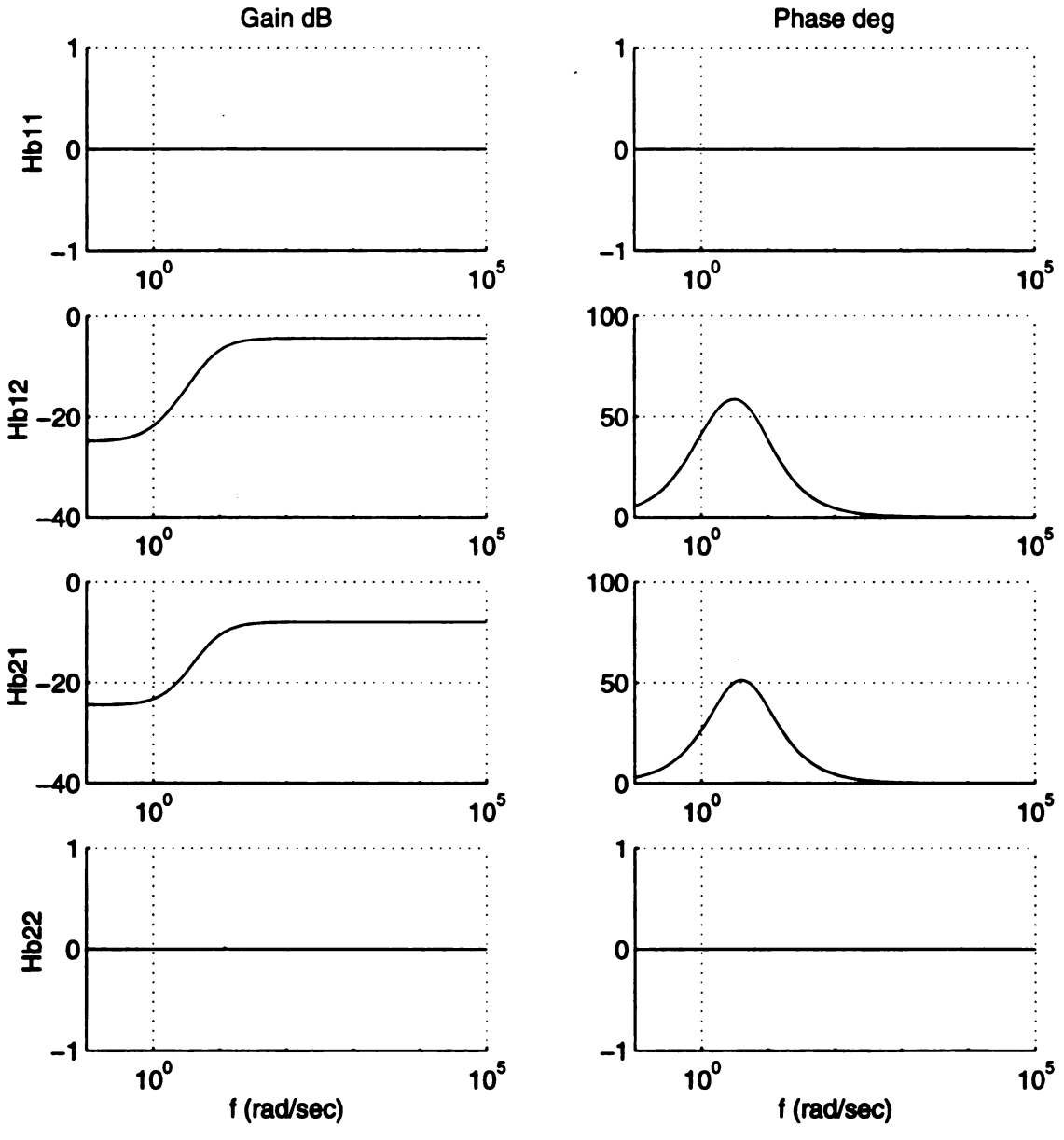


Figure 6.2. Frequency response of the environment model

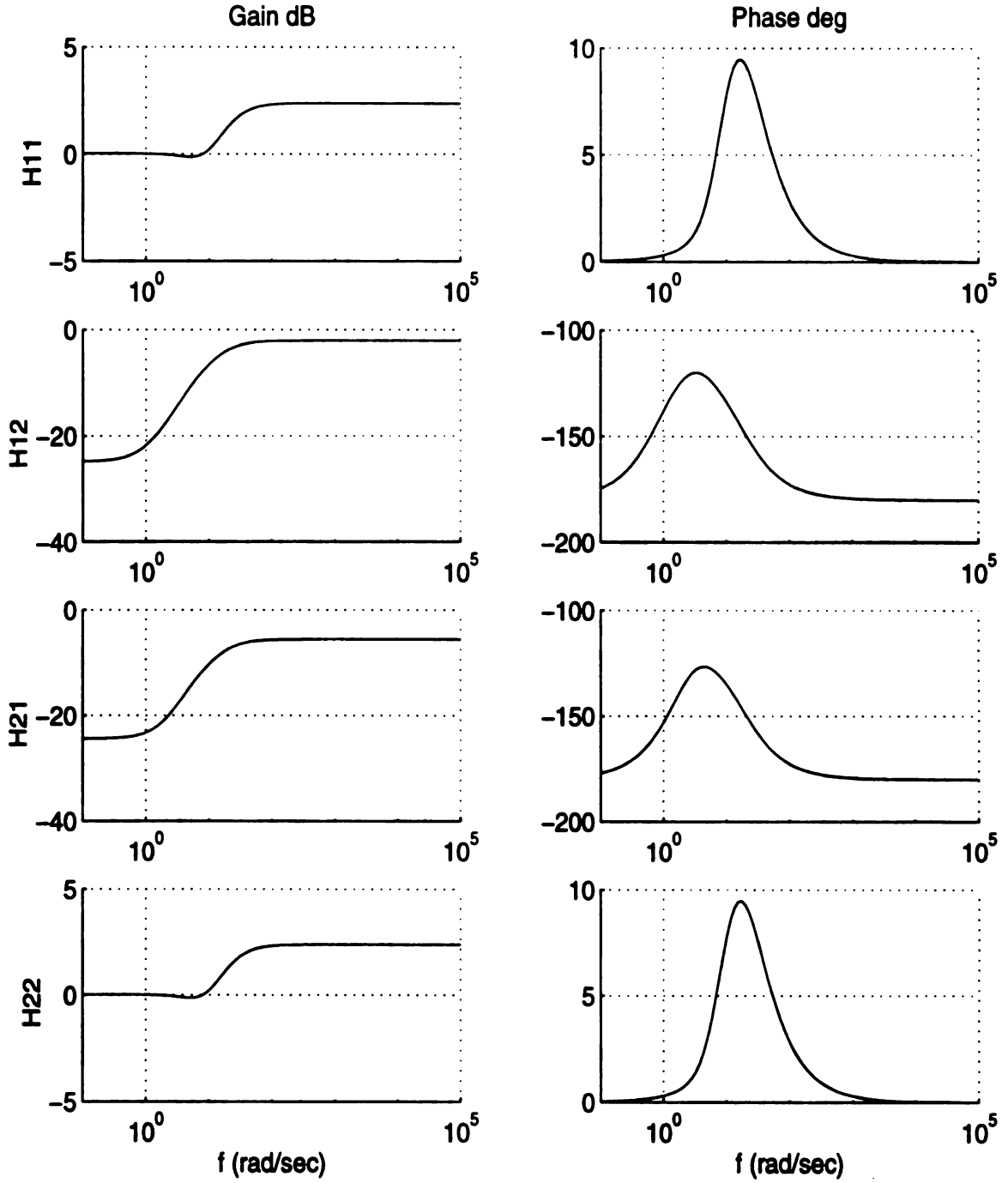


Figure 6.3. Frequency response of the network model

In order to study its performance thoroughly, the parameters of the network will be initially updated separately. This was accomplished by updating some parameters of the network, while holding the remaining parameters, constant to their nominal values.

The first set of simulations considered the update of the individual parameters of the C matrix, namely c_{12}, c_{13}, c_{21} and c_{22}. The initial value of each of these parameters was equal to 0.8 of their nominal values. Also, the unknown sources were assumed to be 2 sine waveforms with respective frequencies $1Hz$ and $2Hz$. Figures 6.4 through 6.7 display the results of such simulation. One could observe that all the parameters converge individually to the desired values. It is important to note that the learning rates of each of the parameters of the C matrix are different, $[\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}] = [10^{1}\ 10^{-1}\ 10^{0}\ 10^{-2}]$.

Once it was demonstrated that each of the parameters converged to the desired value, the update law of all the parameters of the C matrix were combined. Computer simulation were conducted assuming different initial conditions of the parameters of the C matrix. Figures 6.8 through 6.11 represent the results of simulations when the initial conditions are respectively $0.8C^{*}$, $0.5C^{*}$, $0.1C^{*}$ and 0. In all these simulations, one could observe that all the parameters of the C matrix converge to the desired values.

So far, only simulation results of the performance of the parameters of the C matrix were considered. These simulations showed excellent results in terms of the performance of the algorithm in separating the unknown sources. The performance of the algorithm in updating the parameters of the D matrix is considered next.

Computer simulations for different initial conditions of the D matrix, while the parameters of the C matrix are held at their nominal values. were considered. Figures 6.12 through 6.15 show the results of the simulations when the initial conditions are respectively $(d_{12},\ d_{21}) = (0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$. Figure 6.16 combines the

performance of the algorithm for all these tests. One could observe that that the parameters of the D matrix also converged to the desired values.

Despite the fact that the matrices C and D converged to the desired parameters when each one of these matrices was updated separately, some parameters of the network failed to converge when both matrices are updated simultaneously. Therefore, sequential update will be considered in the next section to resolve this problem.

Algorithm defined by f_d

When the nonlinear function f_d is assumed to define the nonlinear function of the algorithm described by equations (6.20) (6.21), the parameters of the matrices C and D converged to some undesirable solution, causing the algorithm to fail in separating the unknown sources. Figures 6.17 and 6.18 display the performance of the algorithm when each matrix is updated separately.

This last result shows again the superior performance of the algorithm defined by the nonlinear function f_a compared to the one defined by f_d, as developed in [24].

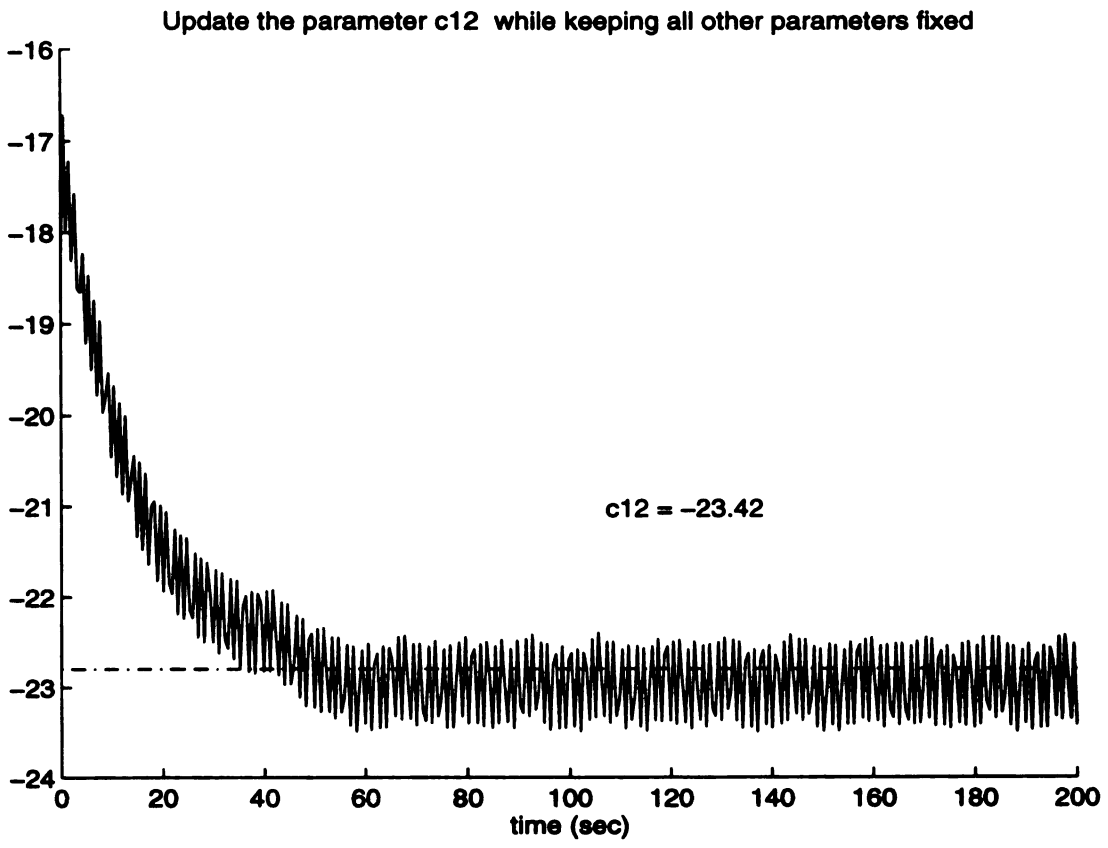


Figure 6.4. $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$ and $\eta_{ij} = 10$

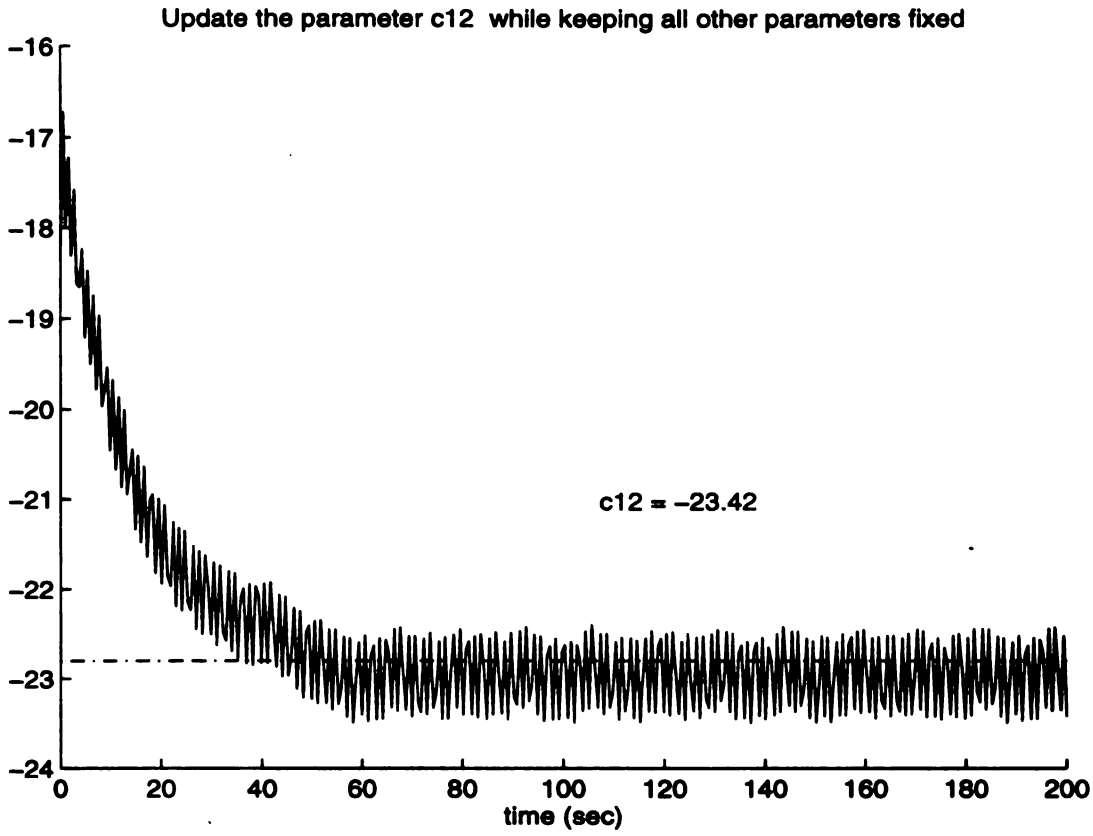


Figure 6.5. $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$ and $\eta_{ij} = 10^1$

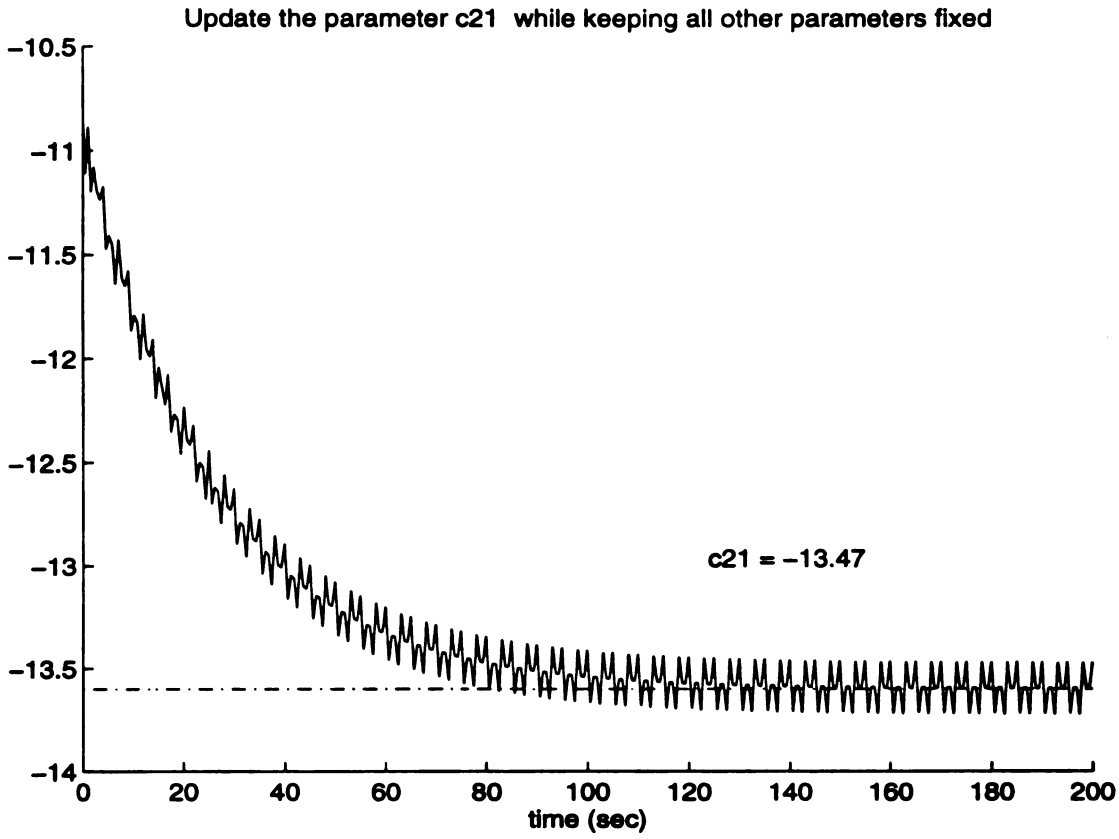


Figure 6.6. $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$ and $\eta_{ij} = 10^0$

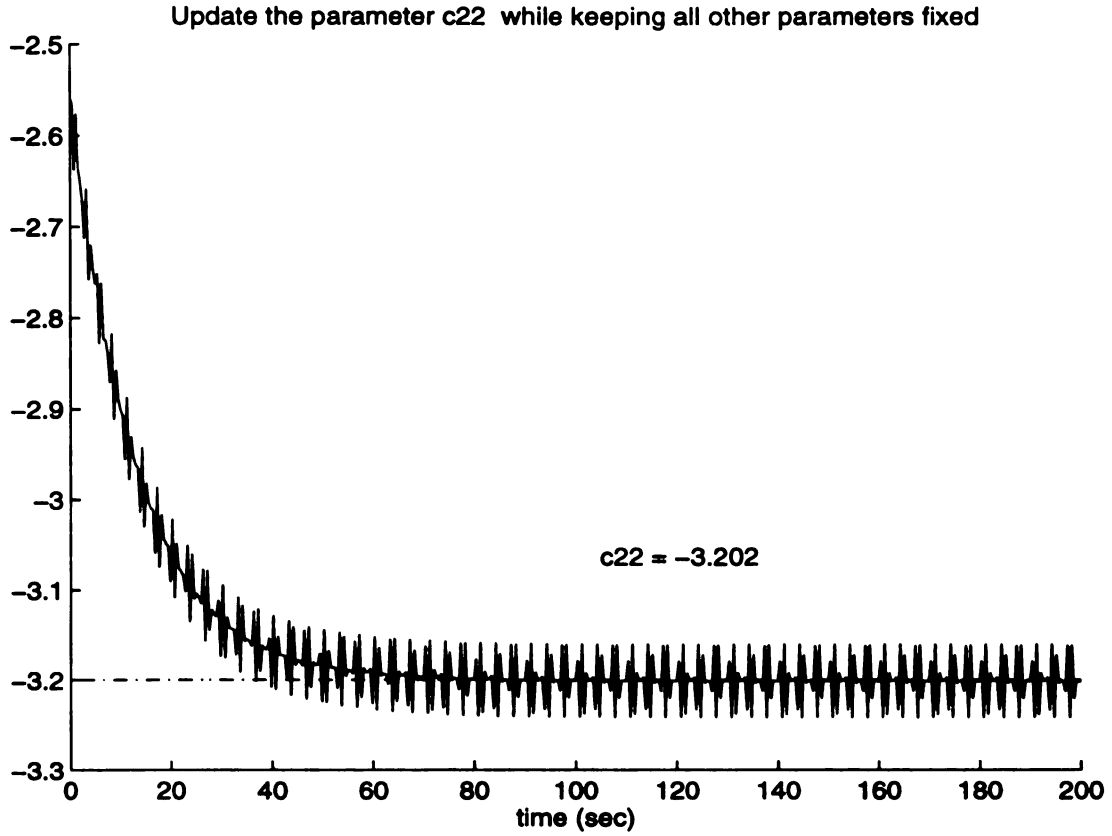


Figure 6.7. $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$ and $\eta_{ij} = 10^{-2}$

6.4.2 An Approach Based on Neuromemitic

Update of C Matrix

The parameters of the C matrix will be updated according to a decorrelation between the output and the state. Therefore, its parameters will be updated according to:

$$\dot{c}_{ij} = \eta_{ij} y_i x_{ij} \tag{6.24}$$

Computer simulations were performed for each of the parameters of the matrix C individually. Then, the update of all of these parameters together was performed

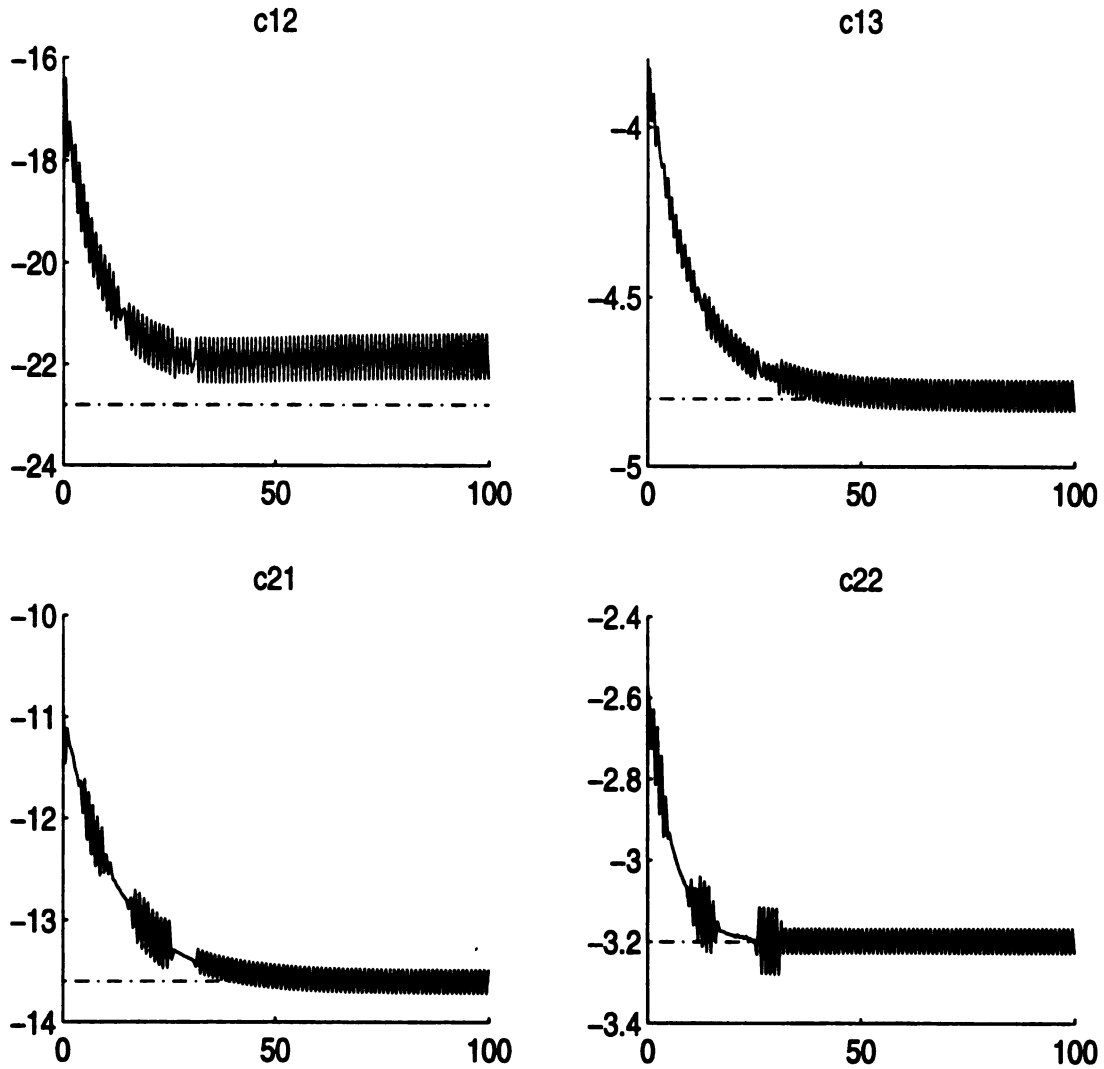


Figure 6.8. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10,\ 0.1,\ 2,\ 0.1)$ and $C_0 = 0.8\bar{C}$

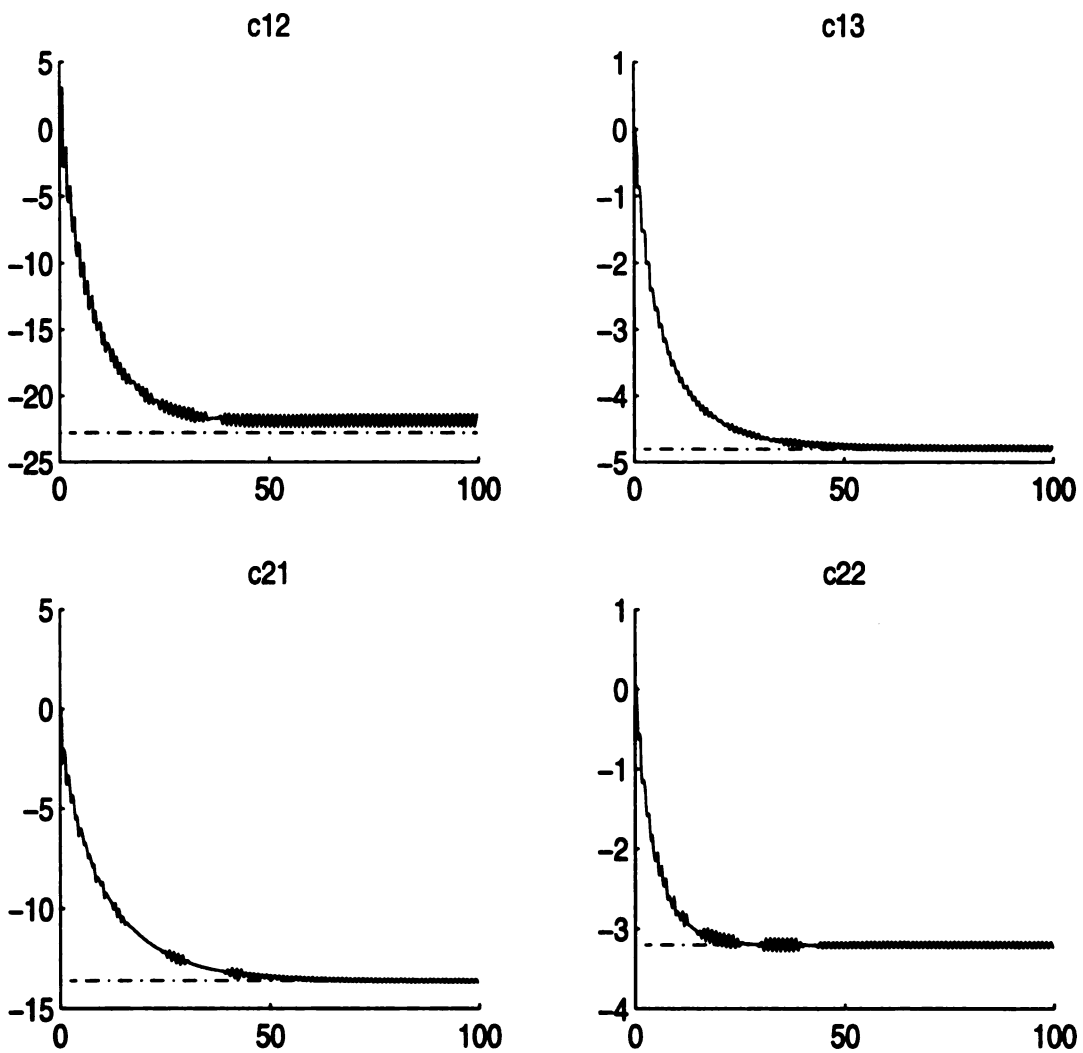


Figure 6.9. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10,\ 0.1,\ 2,\ 0.1)$ and $C_0 = 0.5\bar{C}$

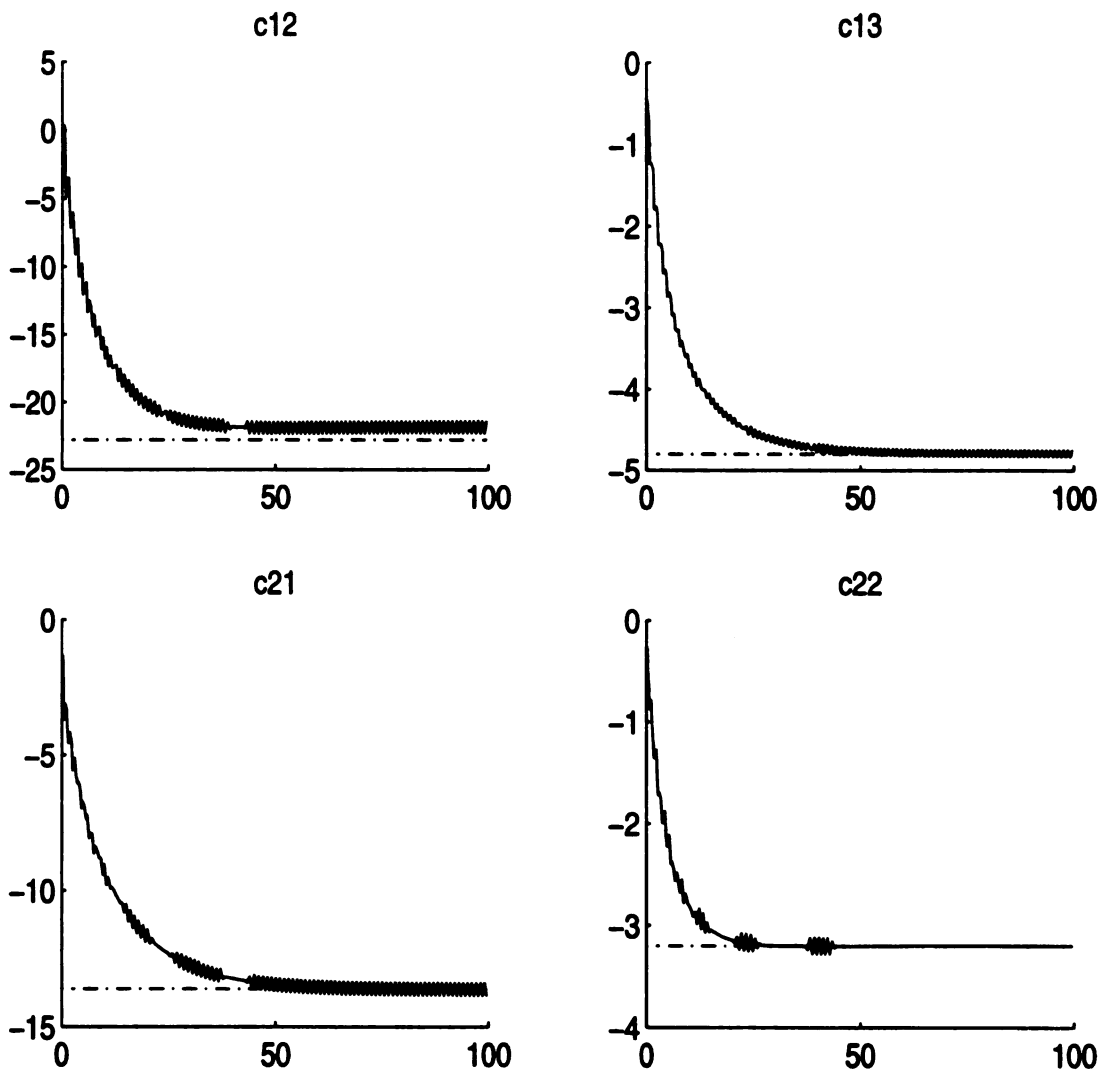


Figure 6.10. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10,\ 0.1,\ 2,\ 0.1)$ and $C_0 = 0.1\bar{C}$

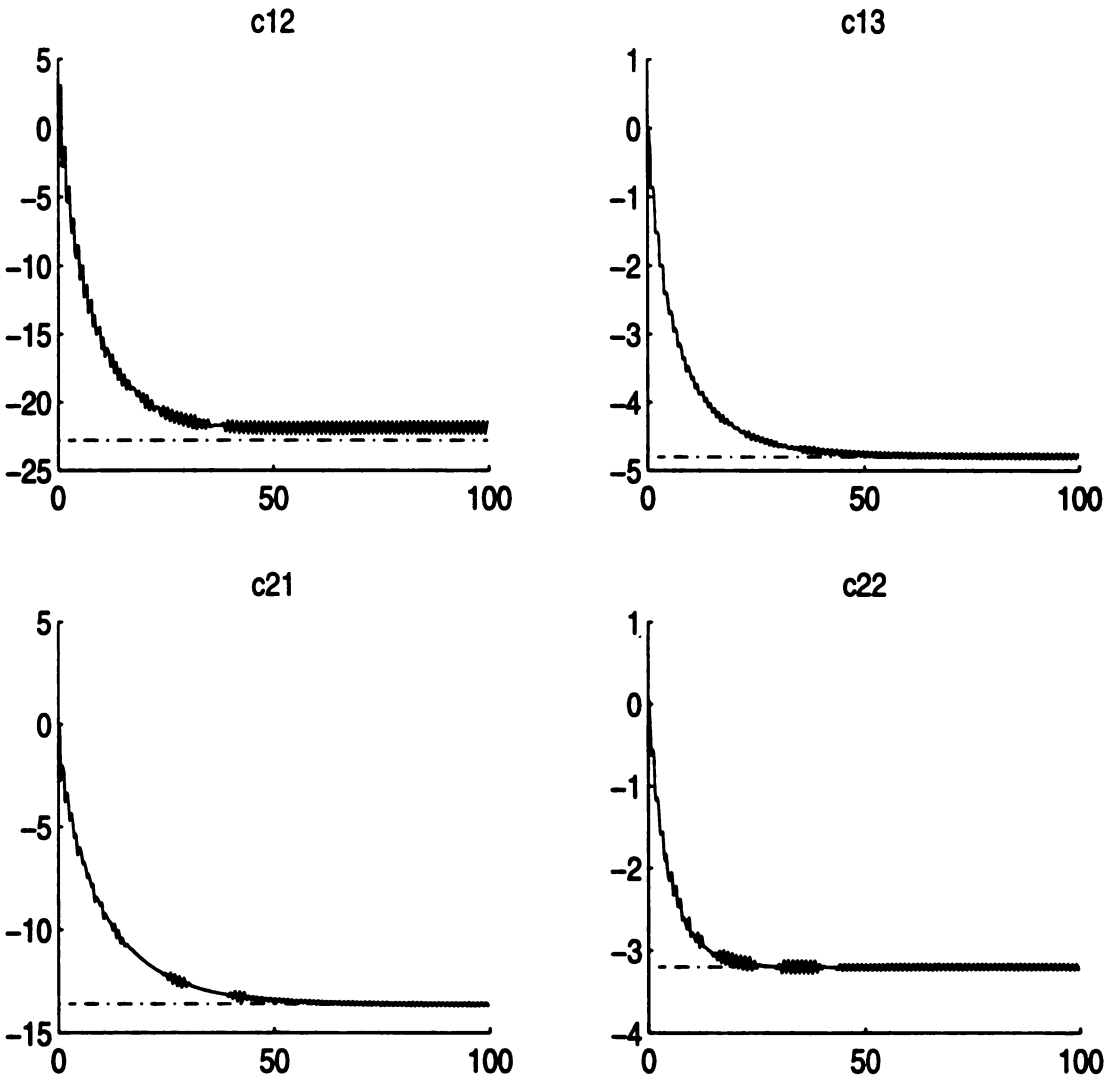


Figure 6.11. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} f_a(y_i) x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10,\ 0.1,\ 2,\ 0.1)$ and $C_0 = 0$

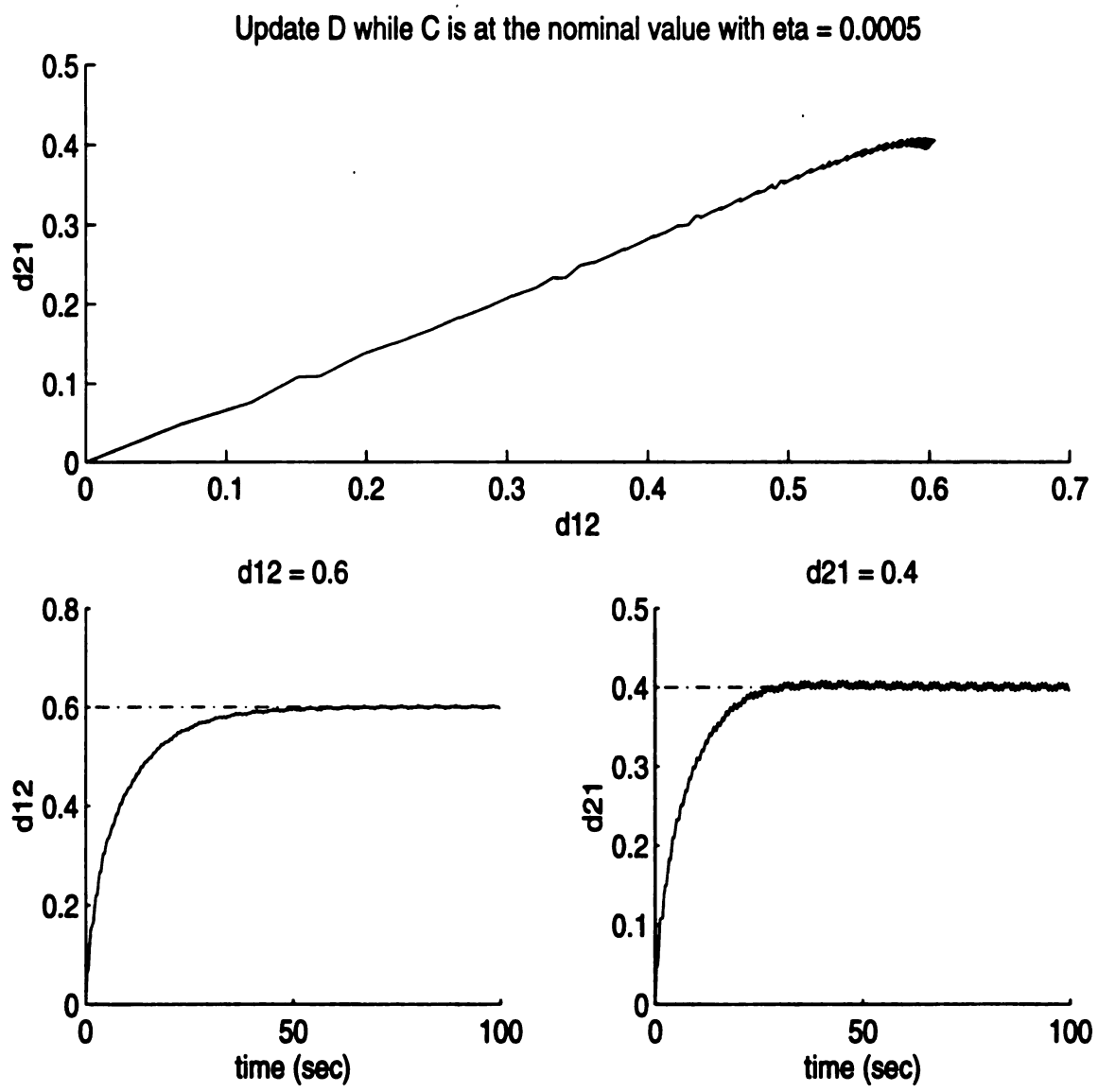


Figure 6.12. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma f_a(y_i) y_j$, $\gamma = 0.0005$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

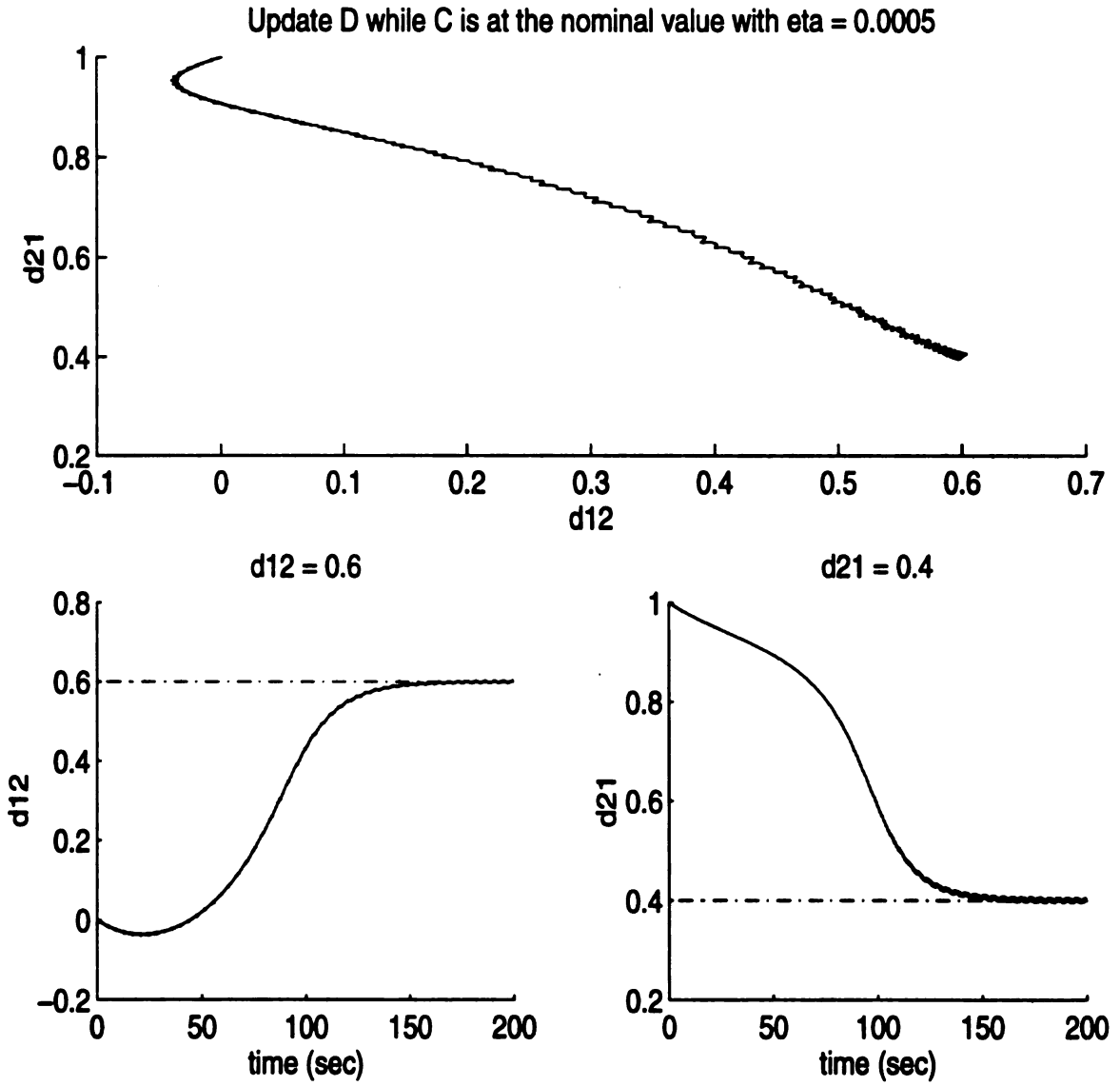


Figure 6.13. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma f_a(y_i)y_j$, $\gamma = 0.0005$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 1.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

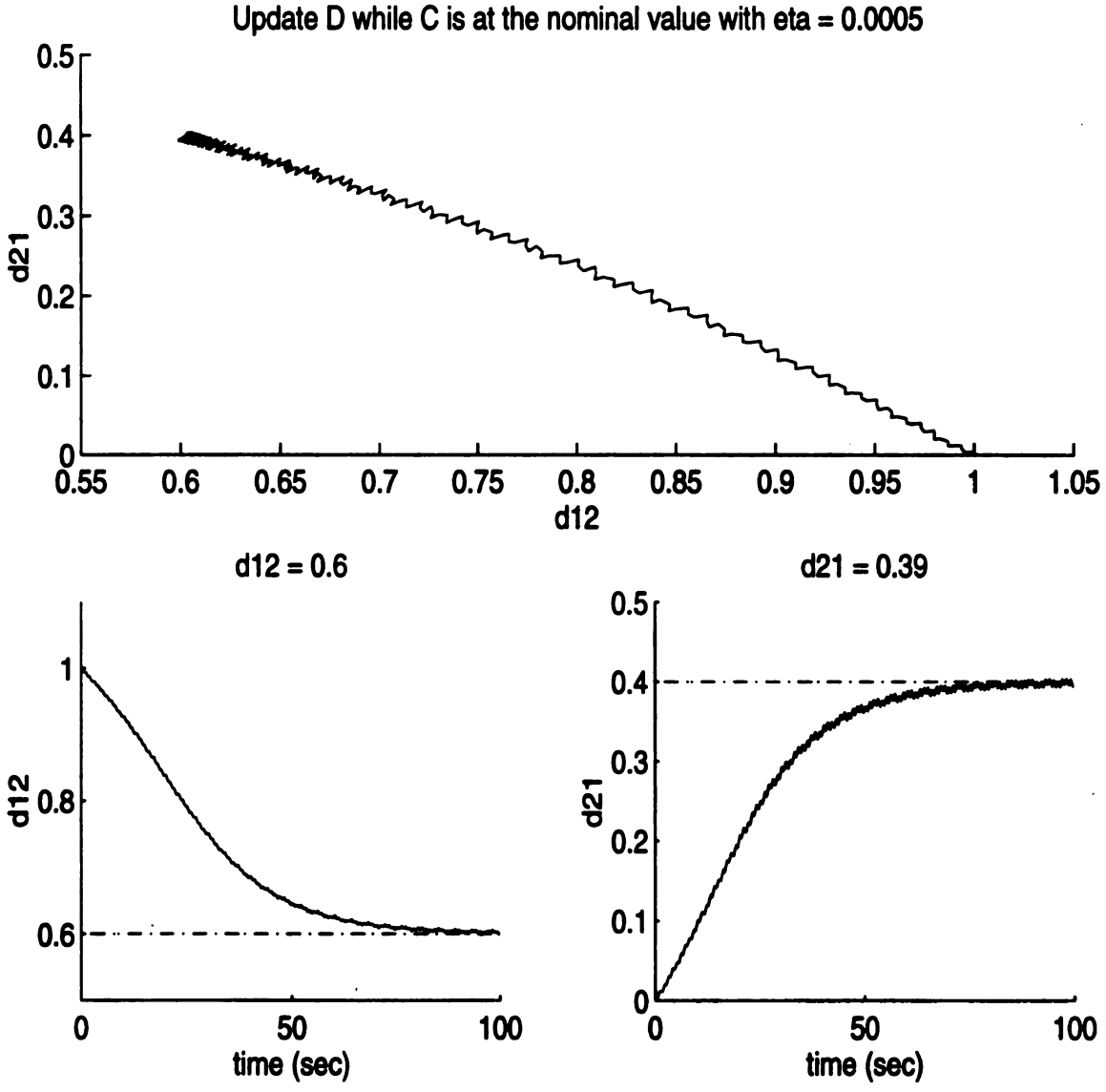


Figure 6.14. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma f_a(y_i)y_j$, $\gamma = 0.0005$ and $(d_{12_0},\ d_{21_0}) = (1.0,\ 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

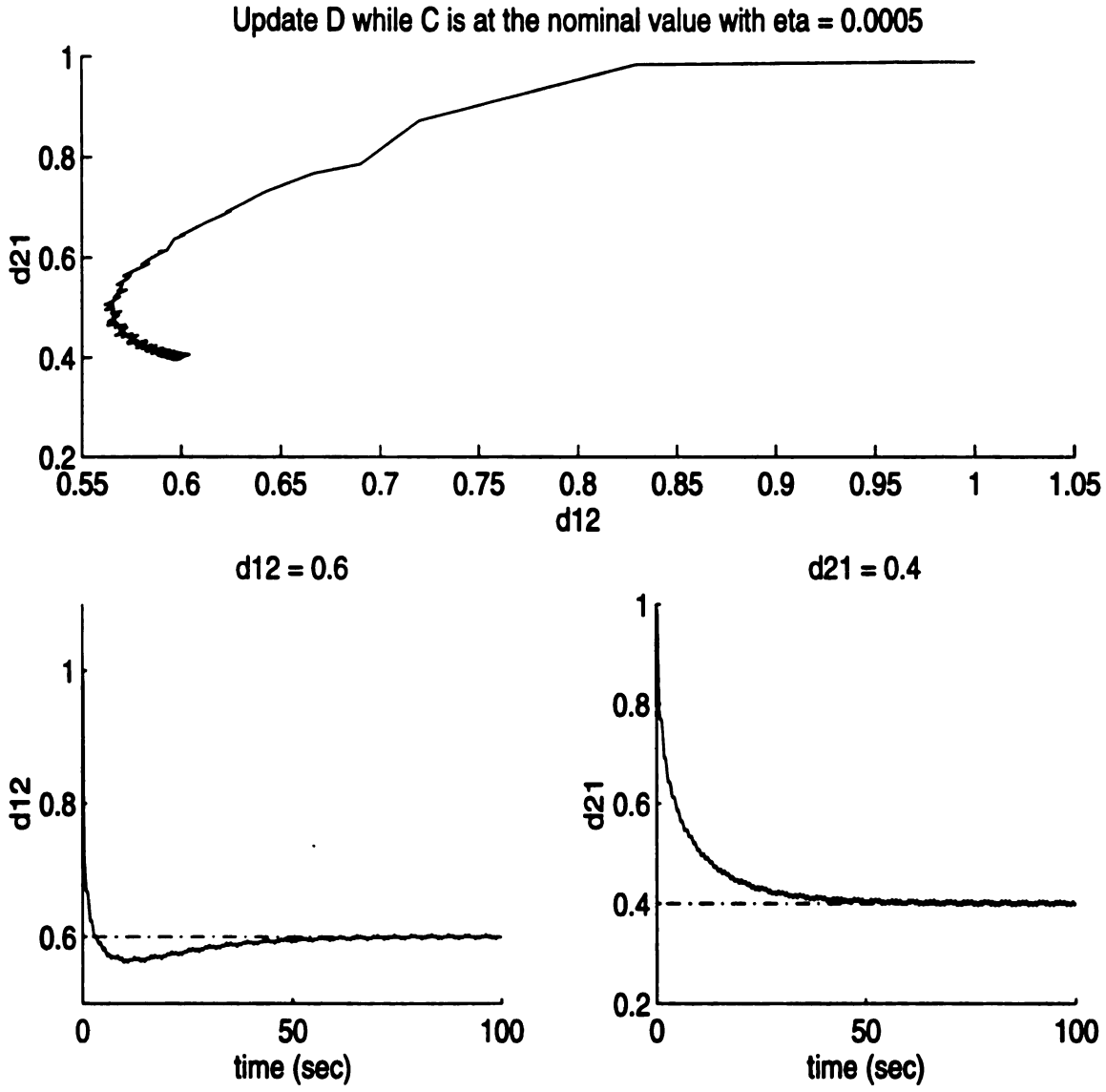


Figure 6.15. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma f_a(y_i) y_j$, $\gamma = 0.0005$ and $(d_{12_0},\ d_{21_0}) = (1.0,\ 1.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

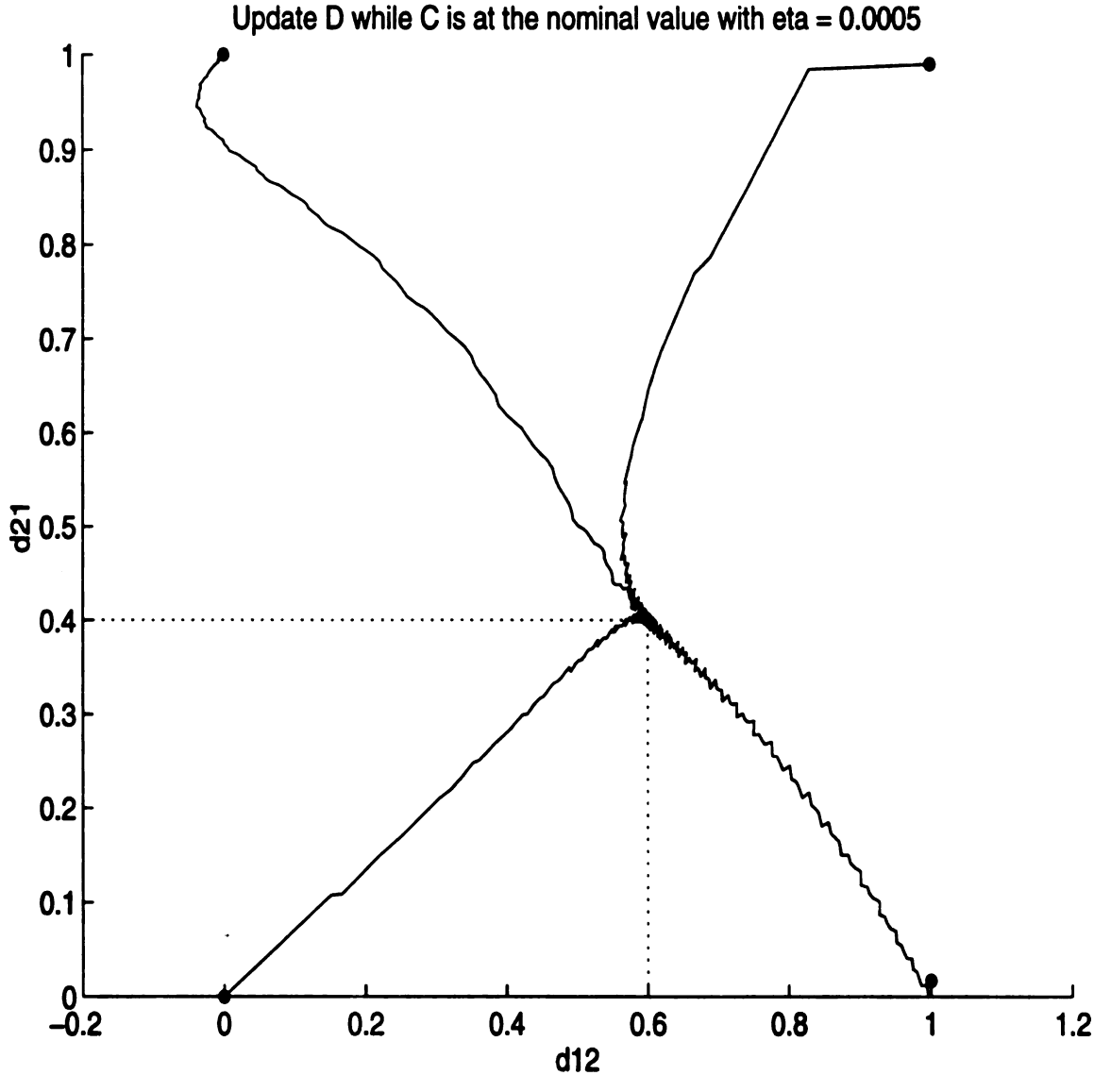


Figure 6.16. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma f_a(y_i)y_j$ and $\gamma = 0.0005$ for different initial conditions. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

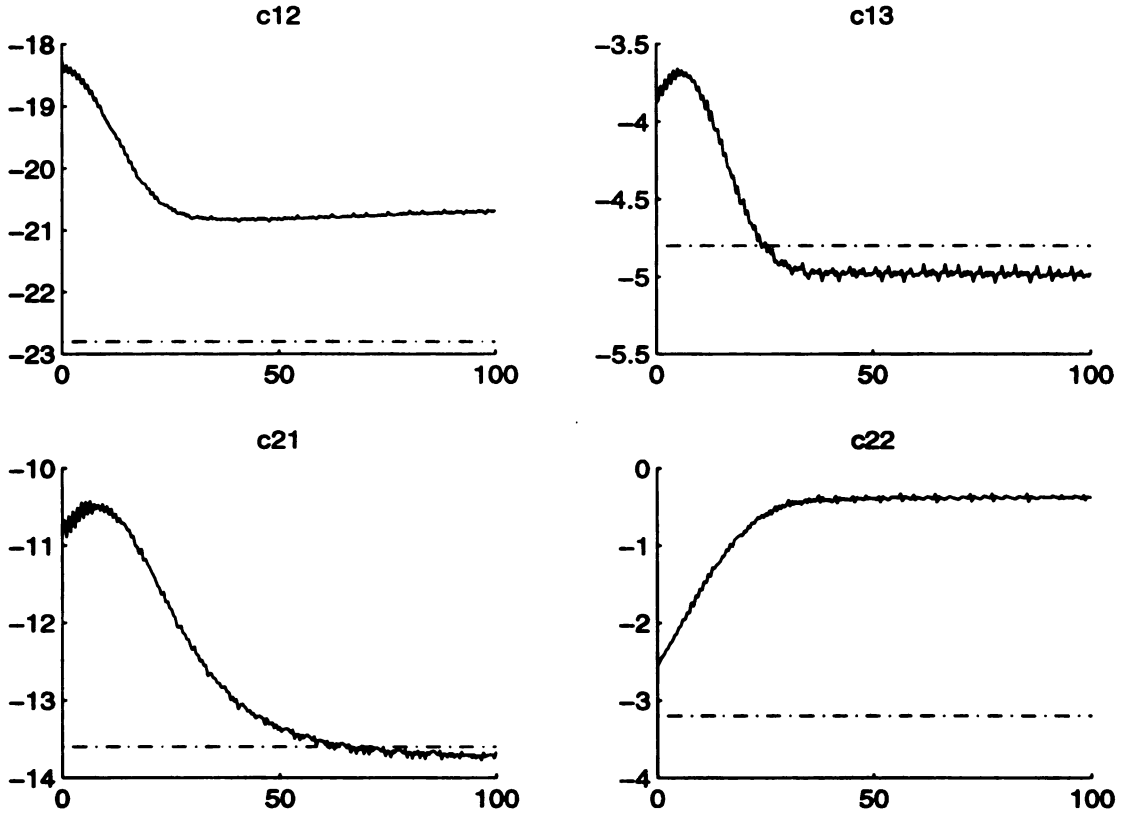


Figure 6.17. Dynamic Feedback Structure. Update only the C matrix according to $\dot{C} = \eta f_d(\mathbf{y})\mathbf{x}^T$ using two sine waveforms, $\eta = [100\ 10\ 100\ 10]$ and $C_0 = 0.8C^*$

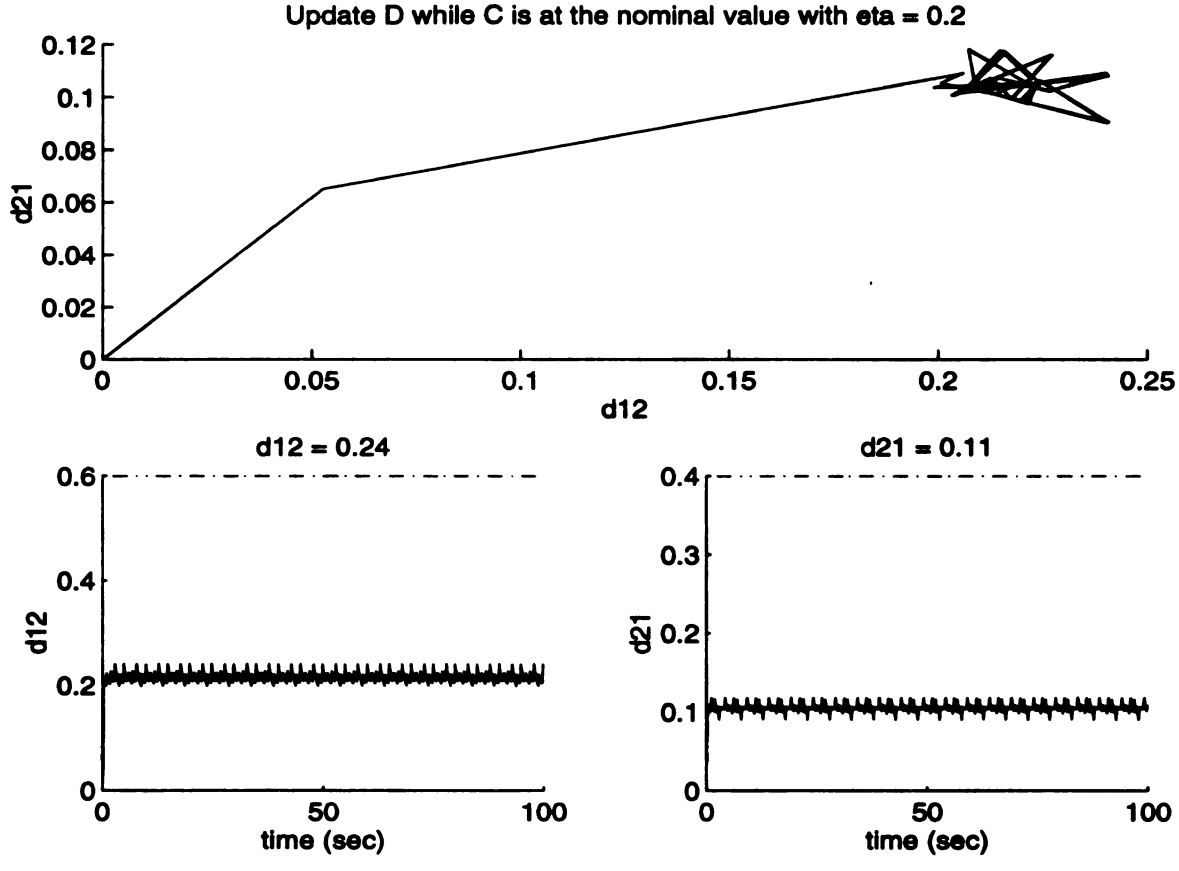


Figure 6.18. Dynamic Feedback Structure. Update only the D matrix according to $\dot{D} = \eta f_d(\mathbf{y})\mathbf{y}^T$ using two sine waveforms, $\eta = 0.2$ and $D_0 = 0$

while keeping the parameters of the matrix D fixed at their nominal values at all times. These simulations were performed with two sine waveforms of respective frequencies $f_1 = 1Hz$ and $f_2 = 2Hz$. The initial conditions for each case were such that the parameters to be updated were 0.8 of the nominal values. Figures 6.19 through 6.22 show, respectively, the convergence of the parameters c_{12}, c_{13}, c_{21}, c_{22}. Each one of these parameters required a different learning rate. The fact that each of these parameters is the coefficient of some power s, where s is the time derivative operator, may reveal the reason for this behavior. Therefore, there is an order of time scale between these parameters. Later in this chapter, this concept will be introduced. In addition, it will be shown that, under a well defined transformation, such a time scale could be reduced, as well as all the parameters of C, to be in the same order of magnitude. Figure 6.23 shows the convergence of the parameters of the matrix C when all the parameters were updated simultaneously. Observe that all the parameters have converged to the correct values.

Using the update law defined by (6.24), a set of simulations was conducted to study the performance of the algorithm for the C matrix based on the choice of the initial conditions. Figures 6.23 through 6.26 show the performance when the initial conditions were respectively 0.8, 0.5, 0.1 and 0.0 of the nominal value of the C matrix. Regardless of the initial conditions, the parameters converge to the desired values. These simulations were also repeated with square and sawtooth waveform functions. The same results were obtained as when using sine waveforms. Some sample results of using different unknown sources were as follows. For instance, Figure 6.27 shows the performance of a sine waveform of frequency $1Hz$ and a square waveform of frequency $2Hz$. The initial conditions were all at 0. Figure 6.28 shows the performance of a sawtooth waveform of frequency $1Hz$ and a square waveform of frequency $2Hz$. The initial conditions were all at 0.

The update law defined by equation (6.24) was modified to include some nonlin-

earities. This was motivated from the work of Herault and Jutten [8, 9]. The update equations were the following:

$$\dot{c}_{ij} = \eta_{ij} y_i^3 x_{ij} \tag{6.25}$$

and

$$\dot{c}_{ij} = \eta_{ij} \sinh y_i \tanh x_{ij} \tag{6.26}$$

Computer simulation of both algorithms were conducted using the same initial conditions, and mixing matrix, and unknown sources. The results of these two simulations are displayed respectively in Figures 6.29 and 6.30. One observes that both algorithms converge.

Update of Matrix D

Starting from the energy function that defines the total power in the signal, one would derive an derive an update law for the parameters of the matrix D similar to (5.40)

$$\dot{d}_{ij} = \gamma_{ij} y_i e_j$$

Computer simulations were performed using 2 sine waveforms of respective frequencies $1Hz$ and $2Hz$. Zero initial conditions were considered. The results of such a simulation is displayed in Figure 6.31. It should be noted that the network failed to converge to the desired values.

On the other hand, successful update of the D matrix, based on the rules developed in [8], was also accomplished.

$$\dot{d}_{ij} = \gamma_{ij} y_i^3 y_j \tag{6.27}$$

and

$$\dot{d}_{ij} = \gamma_{ij} \sinh y_i \tanh y_j \tag{6.28}$$

The performance of the algorithms defined by equations (6.27) and (6.28) are shown respectively in Figures 6.32 and 6.33. The learning rate γ is taken to be the same for all the parameters in each of the above simulations, and is equal to 0.2.

The algorithm defined by equation (6.28) will now be considered and its performance will be studied. First, its performance for different initial conditions will be investigated. Figures 6.33 through 6.36 represent a set of simulations for 4 different initial conditions. The algorithm always converges to the desired values regardless of the initial condition. Figure 6.37 shows the convergence of the algorithm defined by equation (6.28) for the above initial conditions when plotted together.

The performance of the algorithm for various unknown types of sources, sawtooth, square and sine functions was also studied. Figures 6.38 through 6.40 show respectively the performance of the algorithm for the following scenarios: (i) using a sine and a square of respective frequencies $1Hz$ and $2Hz$, (ii) using a sine and a sawtooth of respective frequencies $1Hz$ and $2Hz$, and (iii) using a a sawtooth and a square of respective frequencies $1Hz$ and $2Hz$. One can observe that the parameters converge to the desired values in all these scenarios.

In conclusion, it was demonstrated that the algorithms for updating the matrices C and D have converged when each of these parameters is updated individually by considering the mutual information approach or the neuromimetic approach. It was also shown that both algorithms perform well under various initial conditions, for different types of unknown sources, as well as for different learning rates in the case of the C matrix parameters. It was also shown that the algorithm, defined by the function f_d, which was developed in [24] failed to perform the separation task. It

was also shown that the algorithm defined by f_a has equivalent performance to those defined based on the neuromimetic approach. The novelty of our approach resides in the fact that the nonlinear function f_a was developed based on the minimization of an energy function that defines the independence of signals while the other functions in the literature were motivated from biological inspirations.

Update C and D

In the previous two sections, the performance of the algorithms for updating the matrices C and D, separately was explored. In this section, both update laws will be combined. In this case, the parameters will be updated simultaneously. The parameters of the C matrix were updated according to

$$\dot{c}_{ij} = \eta_{ij} y_i x_{ij} \tag{6.29}$$

and those of the D matrix were updated according to

$$\dot{d}_{ij} = \gamma_{ij} \sinh y_i \tanh y_j \tag{6.30}$$

One parameter at a time

In the first set of simulations, only one parameter of each of the matrices C and D, namely c_{13} and d_{12} were updated. These two parameters will be updated according to the equations (6.29) and (6.30).

In the first simulation, the parameter d_{12} was initially set at its nominal value, $d_{12}^0 = d_{12}^*$, while the parameter c_{13} is initially set to be 0.8 of its nominal value, $c_{13}^0 = 0.8 c_{13}^*$. The results of this simulations are presented in Figure 6.41. In this case, both parameters converge to the desired values as it can be observed in Figure 6.41.

In the next simulation, the initial conditions of the parameters were reversed. The parameter c_{13} was initially set at its nominal value, $c_{13}^0 = c_{13}^*$, while the parameter

c_{12} is initially set to be 0.0, $d_{12}^0 = 0.0$. The results of this simulations are presented in Figure 6.42. One, however, observes that the parameter c_{13} did not converge yet, as it is shown in Figure 6.42. This is due to its small learning rate $\eta_{13} = 20$. Thus, when the learning rate is increased to $\eta_{13} = 40$, both parameters converge, as displayed in Figure 6.43. It should be noted that, in Figure 6.43, despite the fact that the parameter c_{13} was initially at the desired value, it was pushed away from it while the parameter d_{12} was still searching for the path to correct solution. Ultimately, both parameters converge to the desired values. Figure 6.44 shows the phase plot of the parameters d_{12} and c_{13}.

All the parameters together

The previous section proved that the update laws of the matrices C and D were successful when only some parameters of each matrix were updated. The idea behind this set of simulations was to study any coupling effect between the two update laws. It was concluded that the algorithm performs well. In this section, all the parameters of the matrices C and D will be updated and the algorithm's performance will be observed. A set of simulations will be conducted in which some parameters were given their nominal values as the initial conditions, while starting others at some values other than their nominal values. The unknown sources used, for this case, were the sawtooth and the square waveforms with respective frequencies $f_1 = 1Hz$ and $f_2 = 2Hz$. Figures 6.45 through 6.48 show the performance of the network for different initial conditions, while the learning rates for the matrices C and D were held constant at all simulations. Figure 6.45 shows that the algorithm could not converge for the specified time range and learning rate. Thus, a simulation with a higher learning rate for the parameters c_{21} and c_{22} was required, as shown in Figure 6.46. However, a longer running time was also needed. Thus, when $T = 1000$, the algorithm was able to converge. Figure 6.47 shows that all the parameters did

converge, except for one, c_{21}. One possible route to solve this problem is to consider sequential update.

6.5 Sequential Update

In Figure 6.47, it was noted that all the parameters converged except for one parameter, c_{21}. In order to eliminate this phenomenon, we consider updating the parameters "sequentially". This was accomplished by updating the set of parameters of the C and D matrices at alternate time intervals of length ΔT. This means that starting at time t_0, and without loss of generality, the parameters of the matrix D will be updated during the time interval $[t_0+(k-1)\Delta T, t_0\ k\Delta T]$ while those of the C matrix will be held constant. However, during the time interval $[t_0+k\Delta T,\ t_0+(k+1)\Delta T]$, the order of update will be switched so that the parameters of C will now be updated, while those of D will remain constant at their nominal values.

Figures 6.49 through 6.51 show the performance of the algorithm when such a sequential training method is applied with $\Delta T = 50$ seconds. Figure 6.49 shows the performance when the learning rates of the parameters of the C matrix were $\eta = [10^3\ 10\ 10^2\ 10]$ and that of the D matrix is $\gamma = 0.2$. Observe in this case that the parameter c_{21} is converging but a longer time is required for it to converge, or a larger learning rate is required. Also, one observes that in Figure 6.49, all the parameters of the C matrix, except c_{21}, converged in a small period of time, while exhibiting some oscillation. This is due to the large learning rates at which they were updated. Thus, smaller update rates were considered and taken to be $\eta = [200\ 10\ 200\ 10]$. The results of these simulations are presented in Figure 6.51. Also, Figure 6.50 shows a simulation when $\eta = [1000\ 10\ 500\ 10]$.

Figure 6.52 shows the response of the network at the start of the update. One observes that the output signals $\mathbf{y}(t)$ were different from the desired signals $\mathbf{s}(t)$.

Recall that the initial condition of the system was not set at the nominal values, but rather $C_0 = 0.8C^*$ and $D_0 = 0$. Figure 6.53 shows the performance of the algorithm after all the parameters have converged. It can clearly be observed that the original signals were recovered and that the error was in the order of 10^{-3}. The network is tested by changing the unknown source to a pair of a sawtooth and a sine functions. Figure 6.54 shows the signal recovery performed by the network. The final values to which the network converged were $[c_{12}\ c_{13}\ c_{21}\ c_{22}] = [-21.9783\ -4.7968\ -13.4989\ -3.3635]$ and $[d_{12}\ d_{21}] = [0.3992\ 0.6006]$

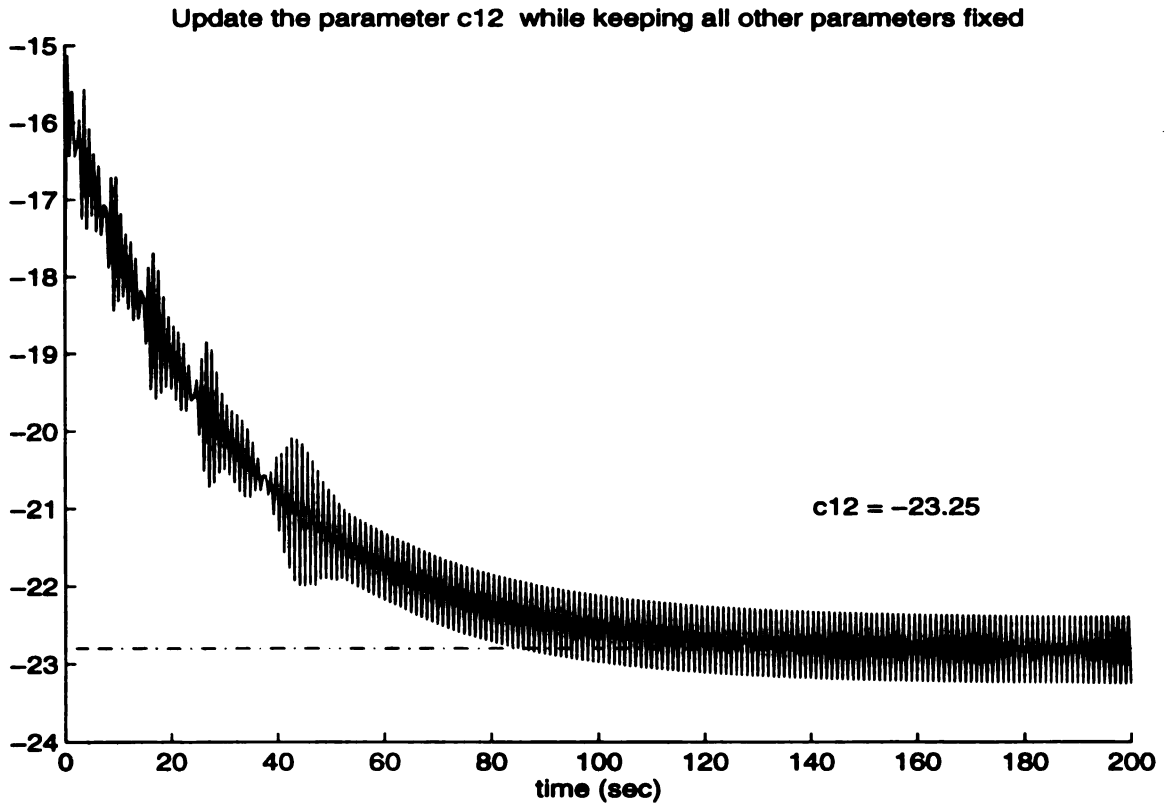


Figure 6.19. $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$ and $\eta_{ij} = 2000$

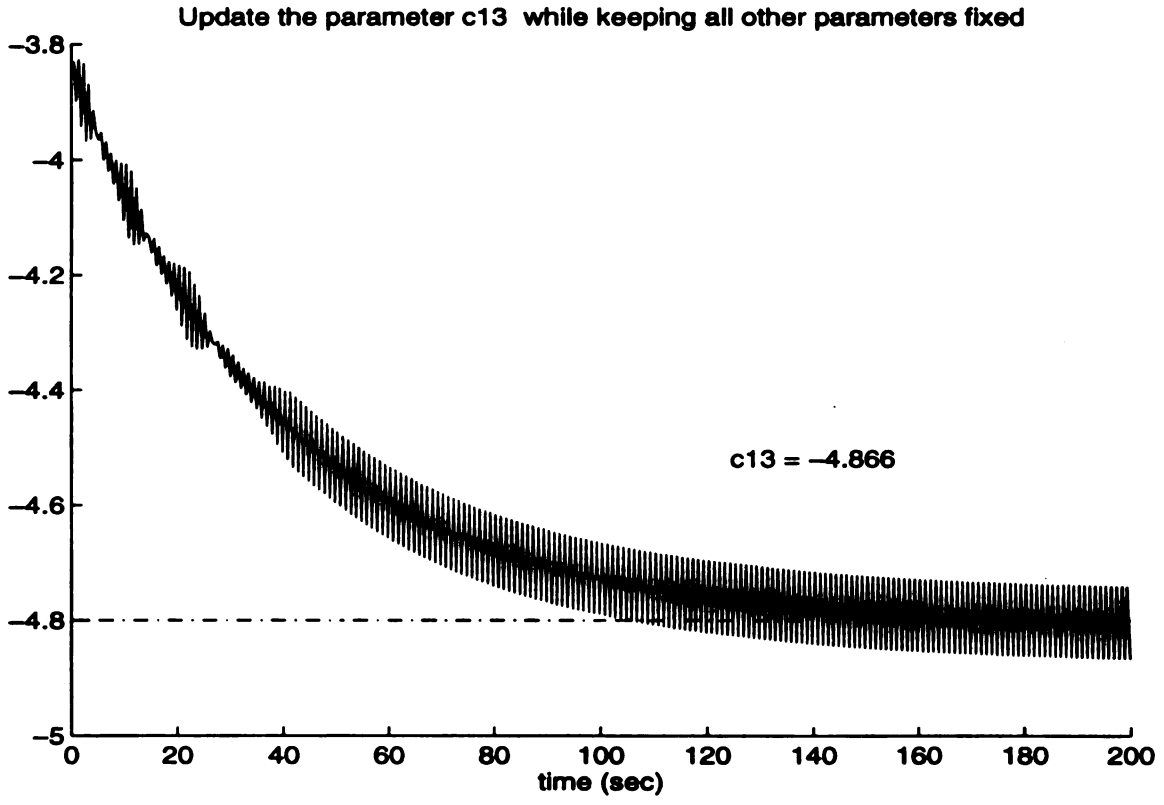


Figure 6.20. $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$ and $\eta_{ij} = 20$

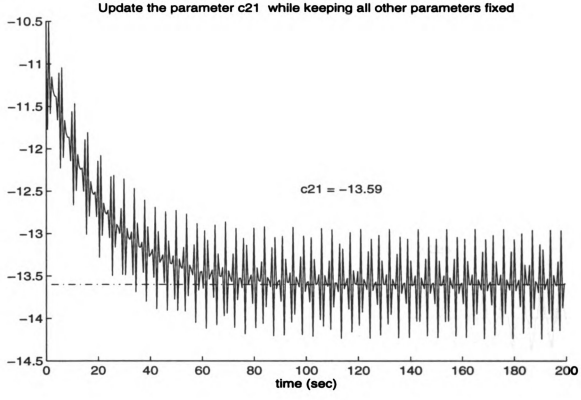


Figure 6.21. $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$ and $\eta_{ij} = 500$

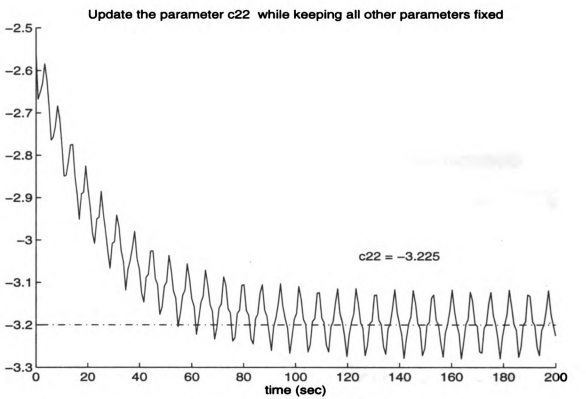


Figure 6.22. $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$ and $\eta_{ij} = 10$

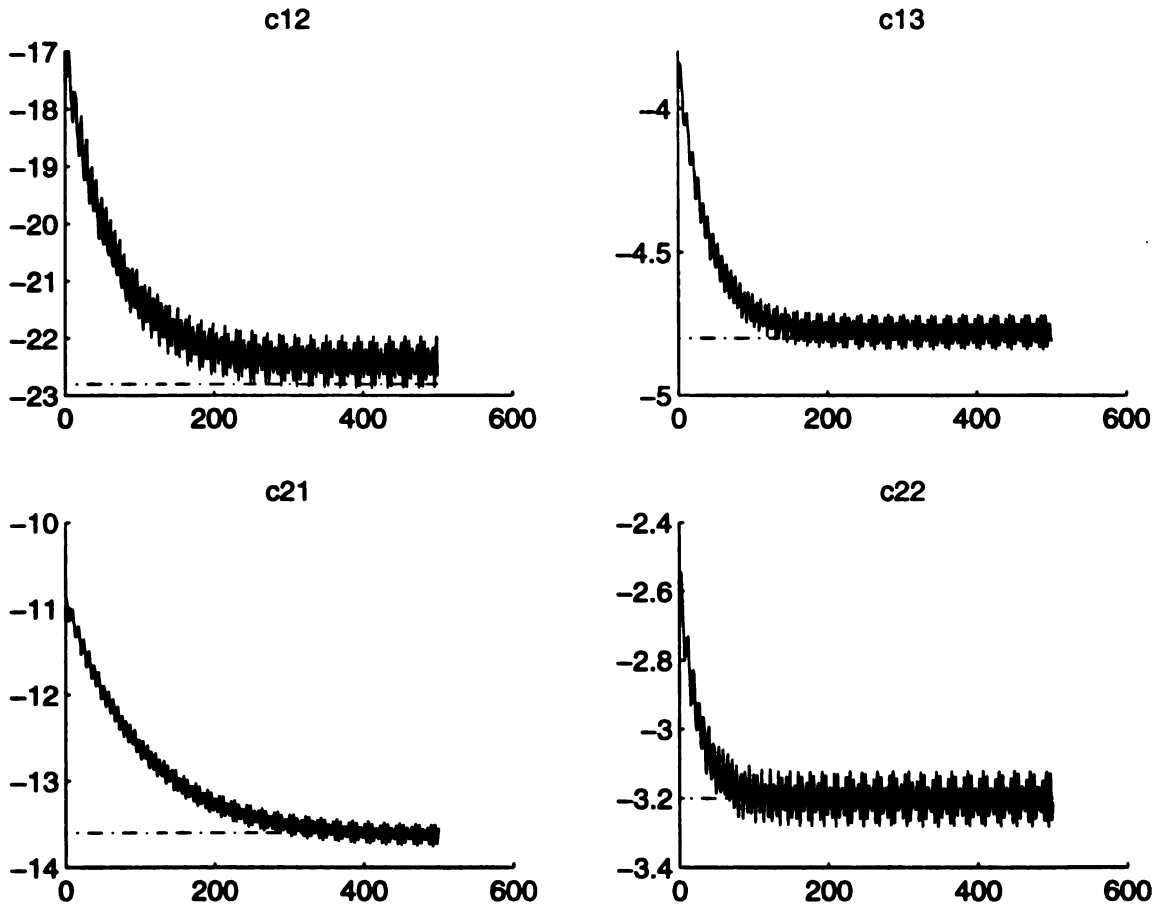


Figure 6.23. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 1)$ and $C_0 = 0.8\bar{C}$

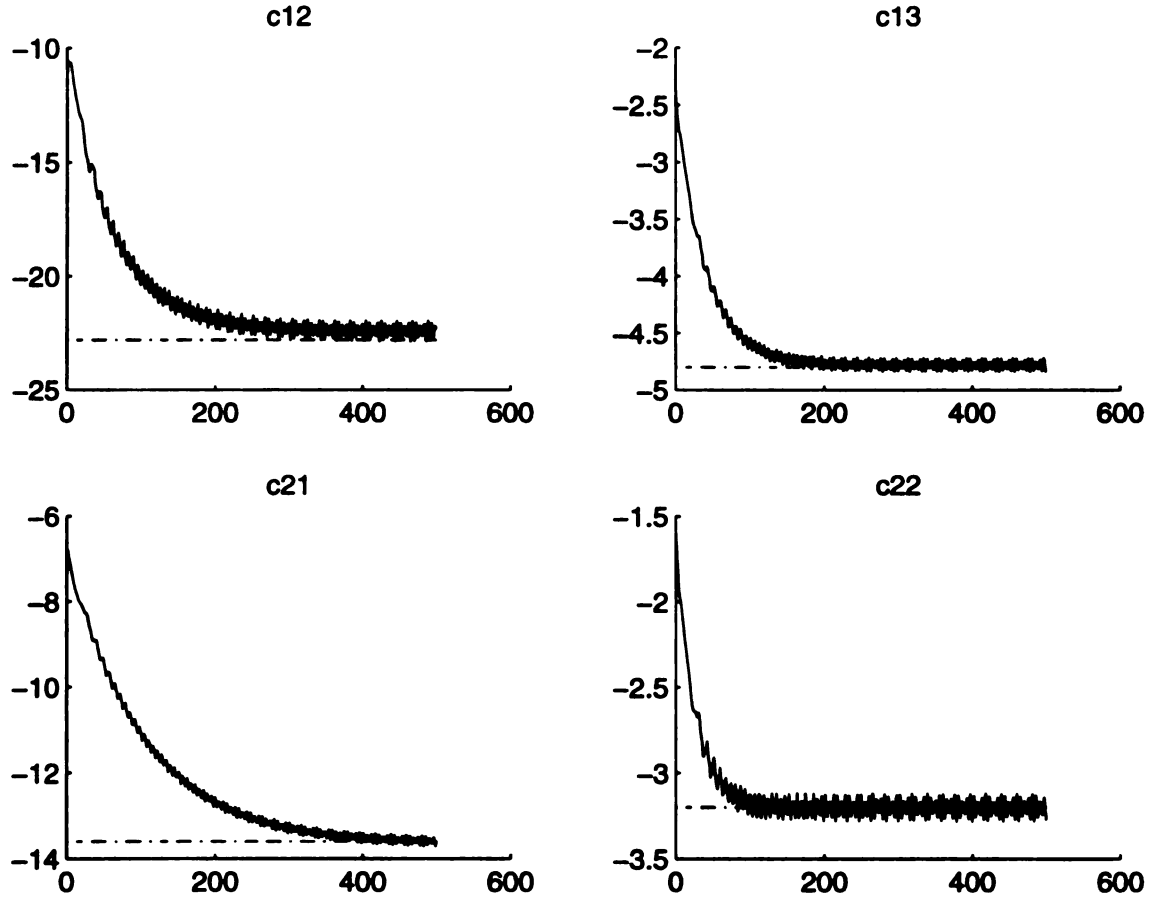


Figure 6.24. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 1)$ and $C_0 = 0.5\bar{C}$

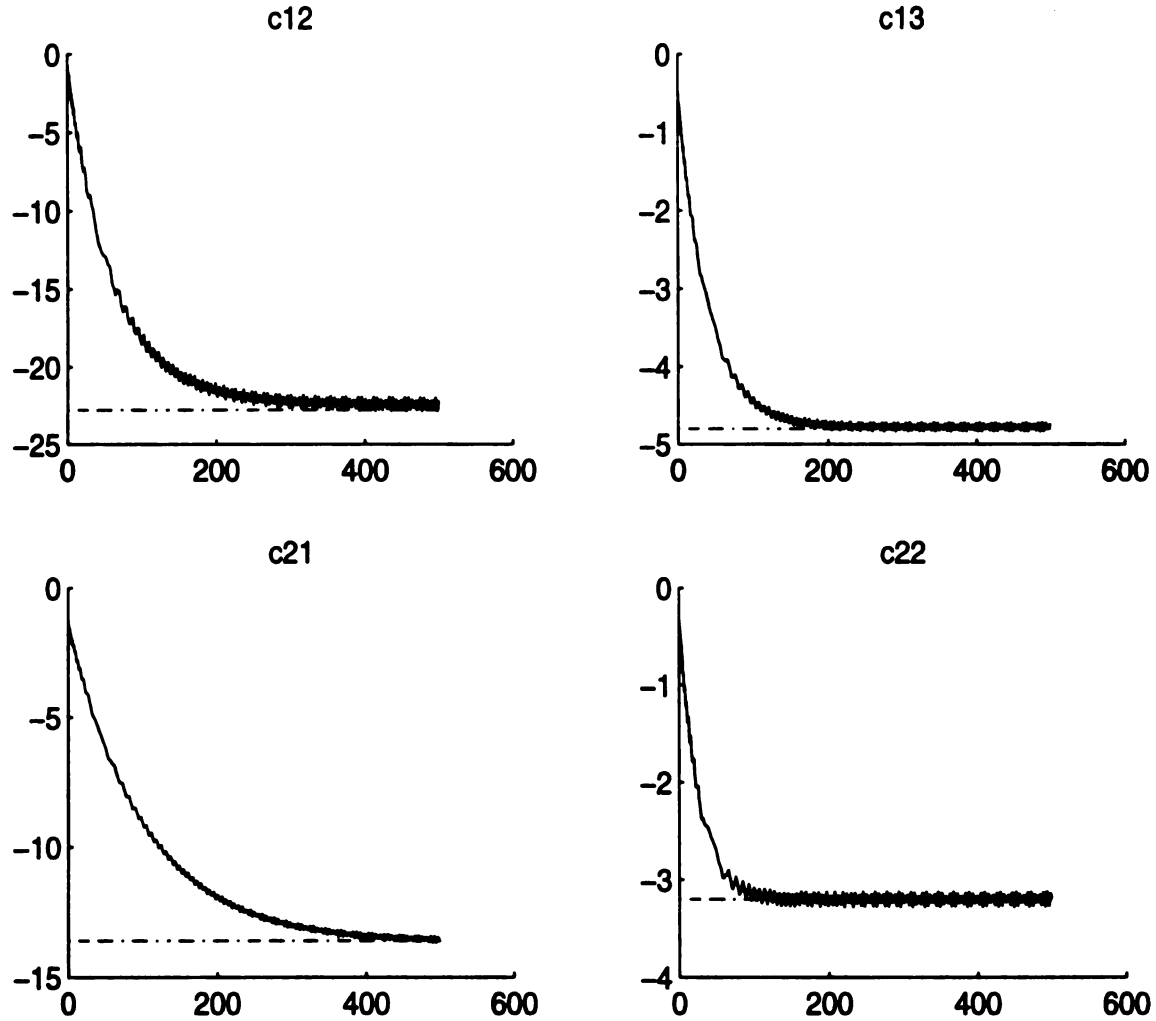


Figure 6.25. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 1)$ and $C_0 = 0.1\bar{C}$

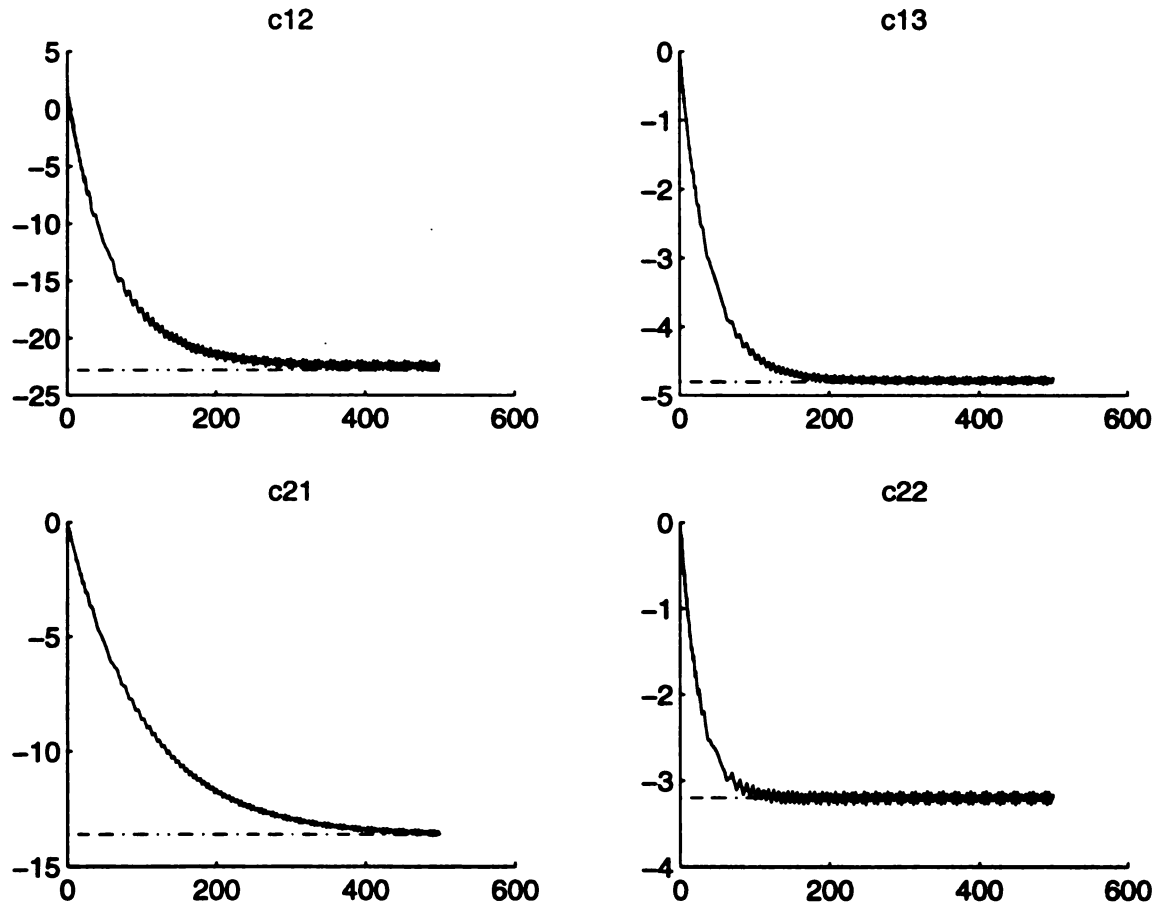


Figure 6.26. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 1)$ and $C_0 = 0$

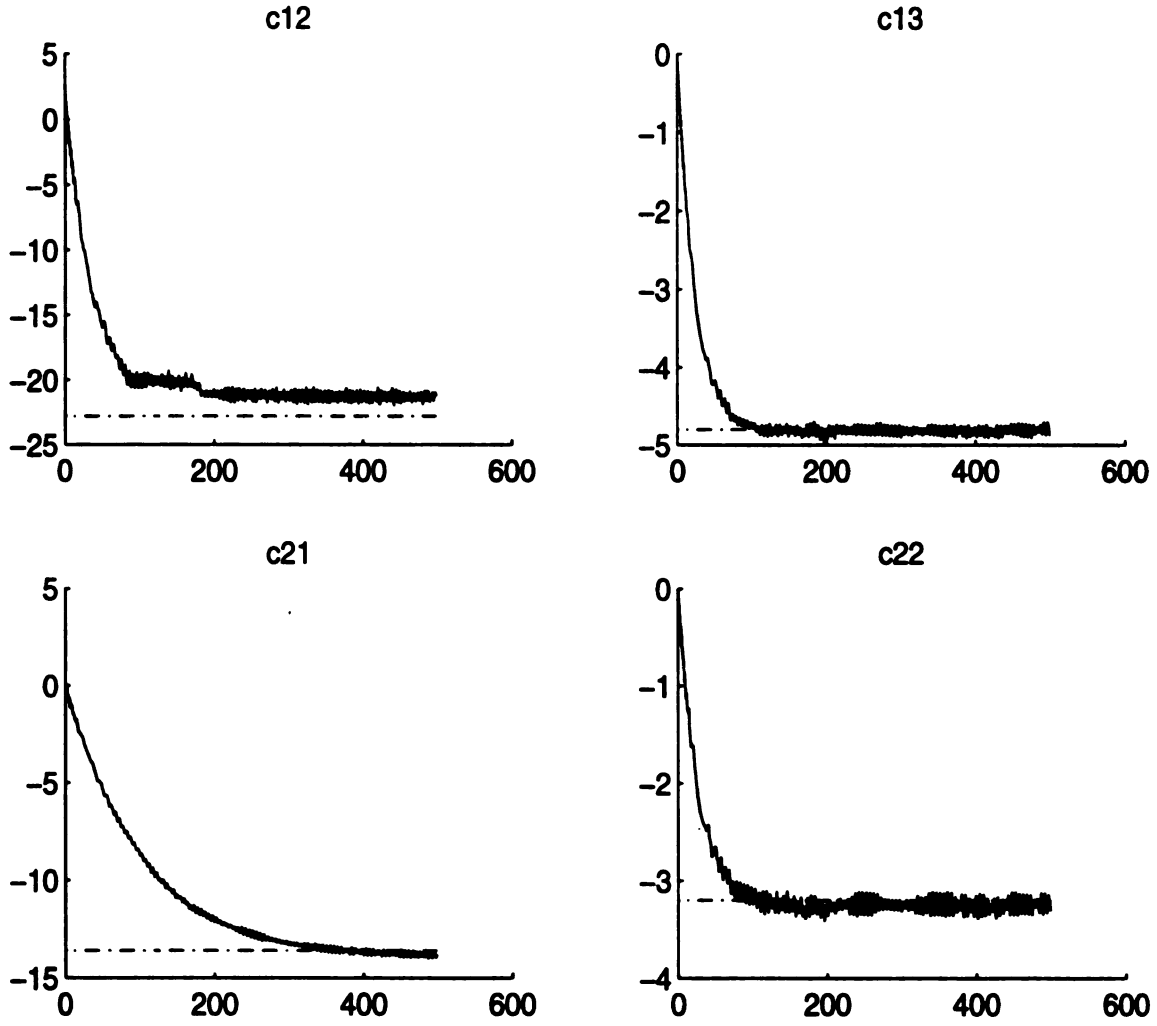


Figure 6.27. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 1)$ and $C_0 = 0$. The unknown sources are $s_1(t) = sin(2\pi t)$ and $s_2(t) = square(4\pi t)$

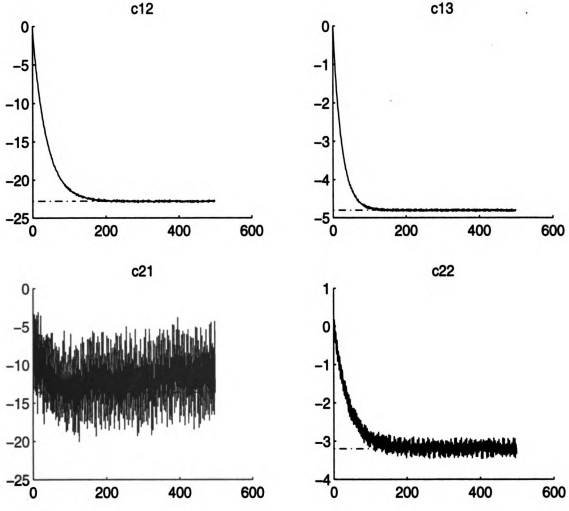


Figure 6.28. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i x_{ij}$, $(\eta_{12}, \eta_{13}, \eta_{21}, \eta_{22}) = (10^3, 10, 10^5, 10^2)$ and $C_0 = 0$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

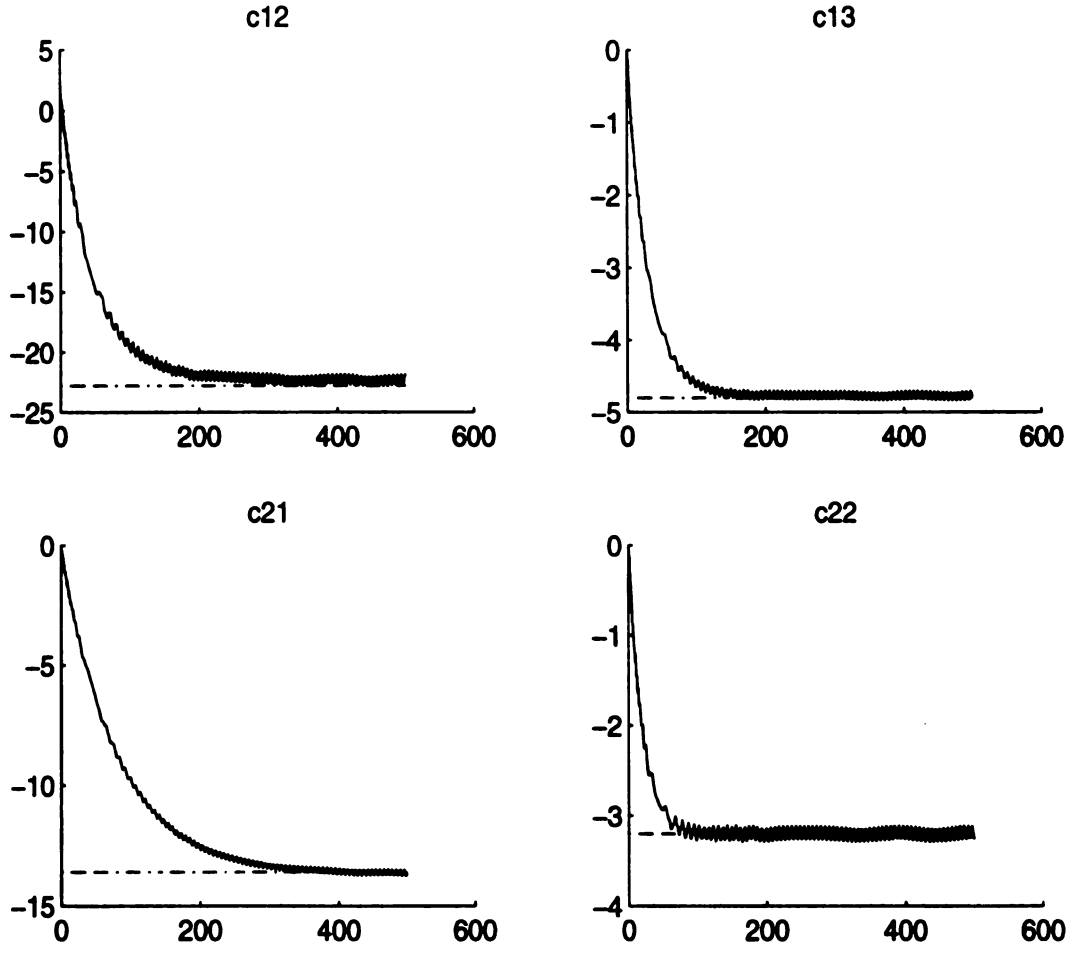


Figure 6.29. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} \sinh y_i \tanh x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 10)$ and $C_0 = 0$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

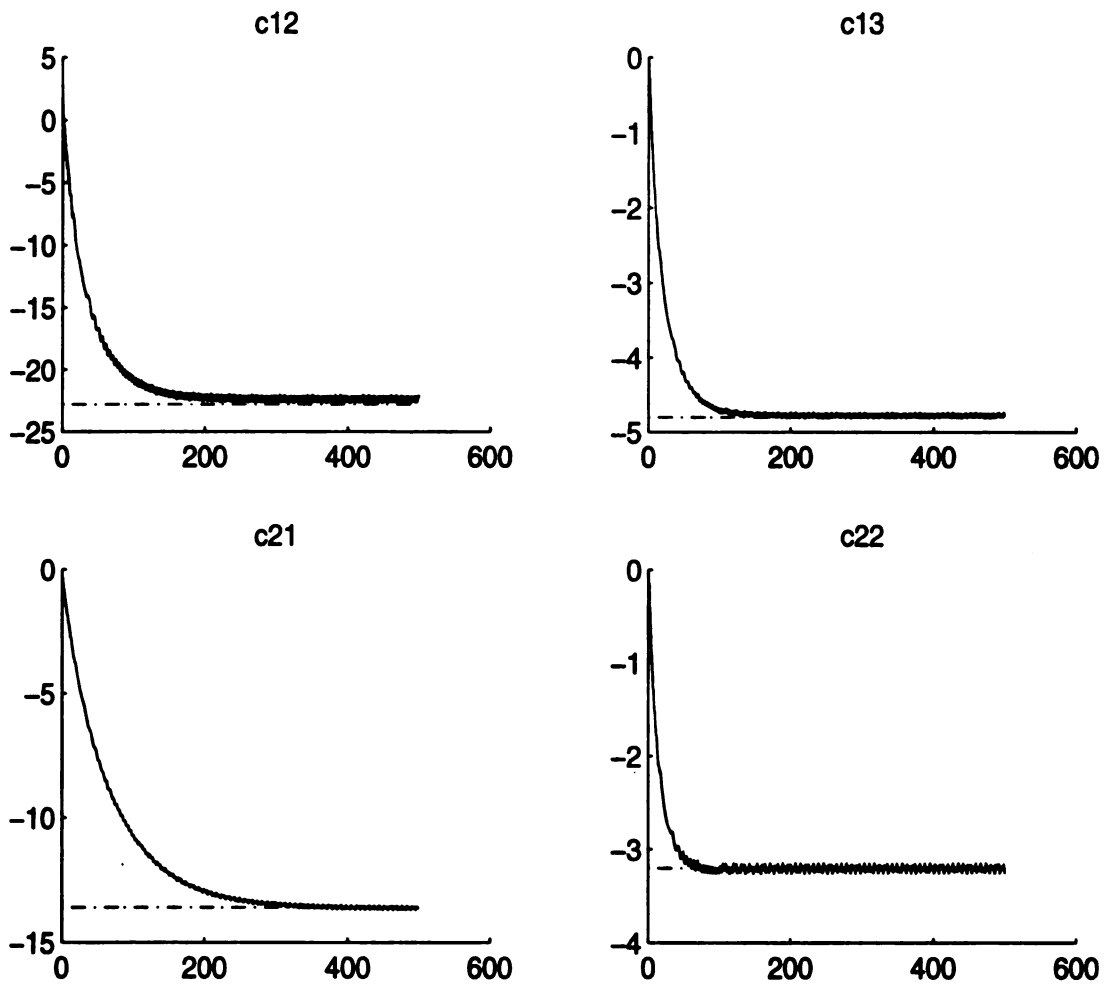


Figure 6.30. Update all the parameters of the C matrix according to $\dot{c}_{ij} = \eta_{ij} y_i^3 x_{ij}$, $(\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{22}) = (10^3, 10, 10^2, 10)$ and $C_0 = 0$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

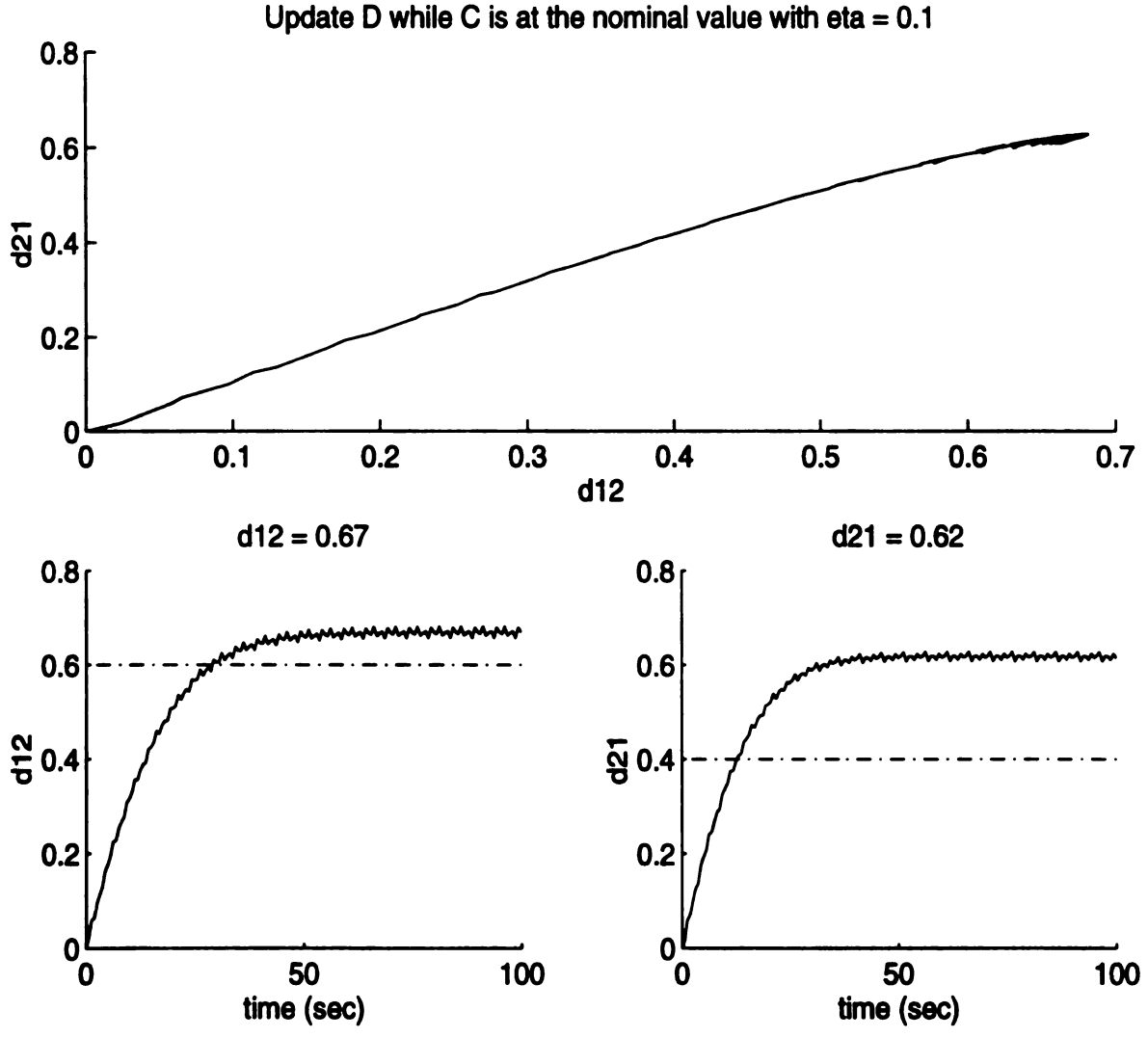


Figure 6.31. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma y_i e_j$, $\gamma = 0.2$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

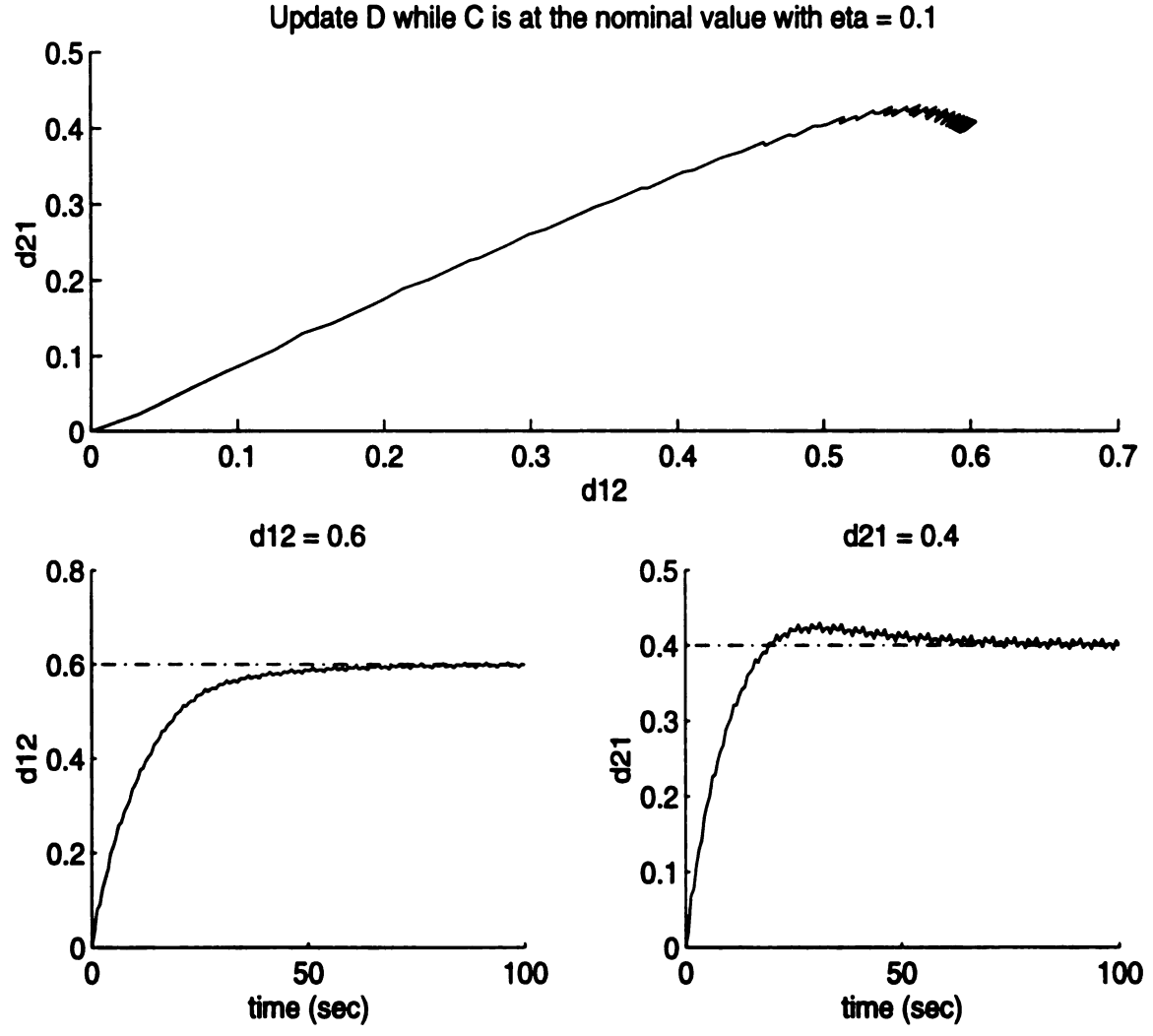


Figure 6.32. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma y_i^3 y_j$, $\gamma = 0.2$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

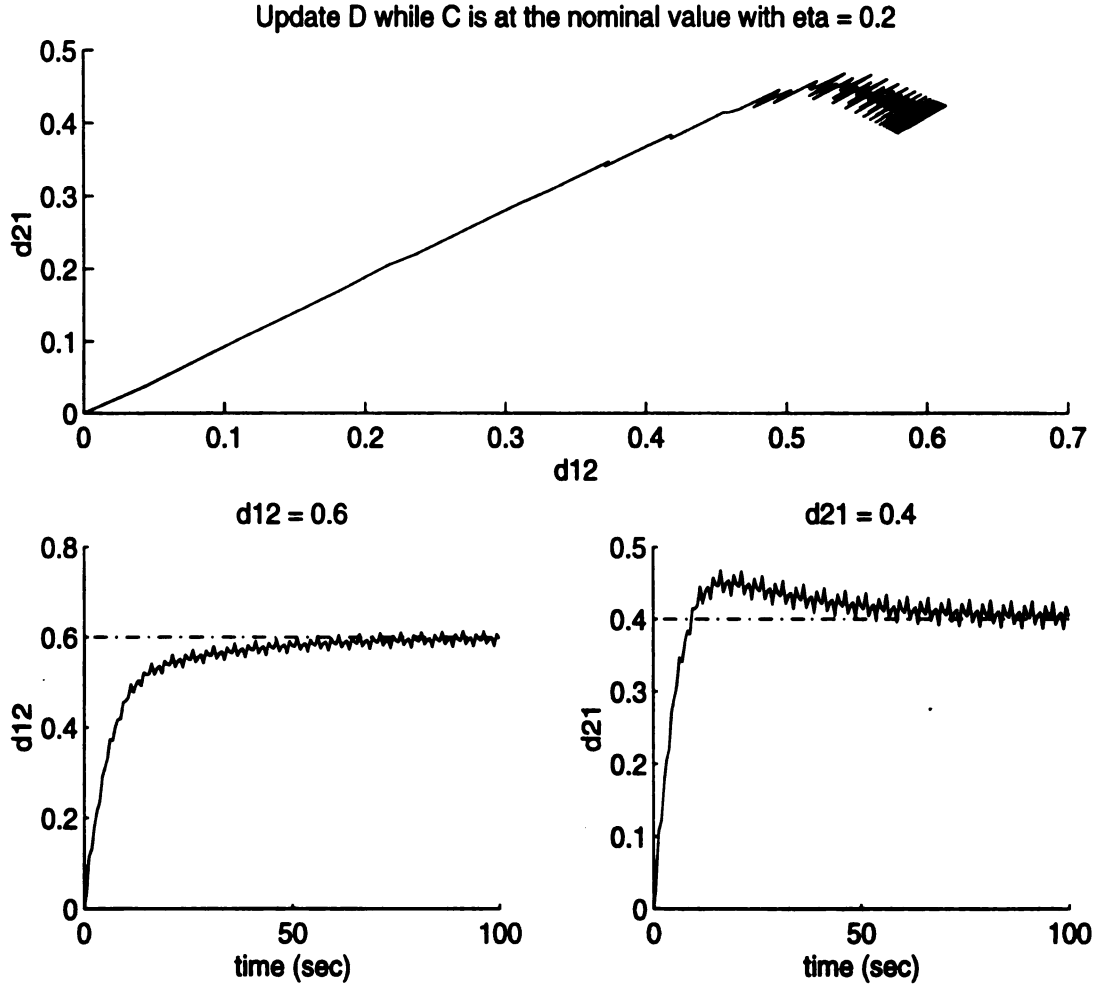


Figure 6.33. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0}, d_{21_0}) = (0.0, 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

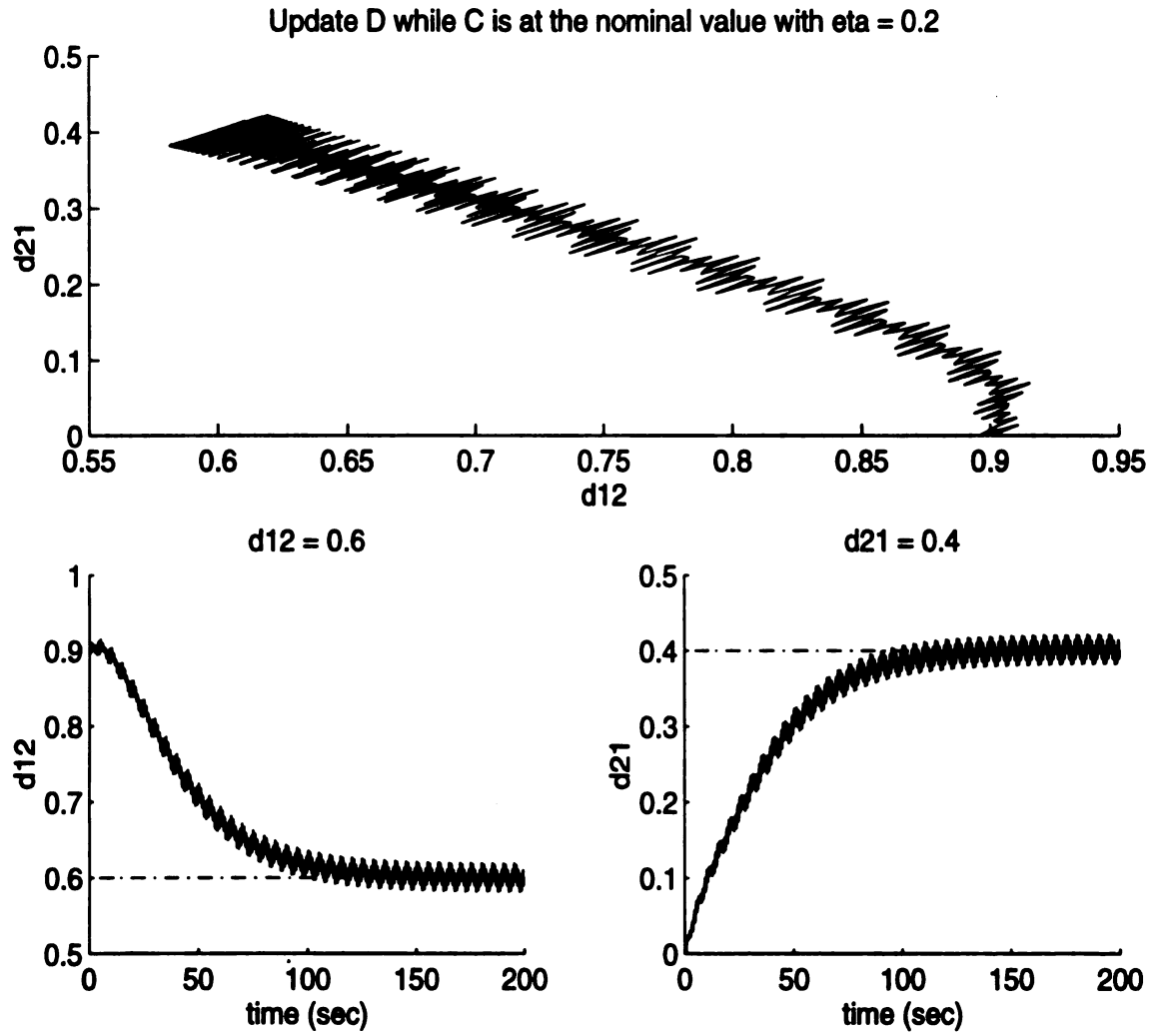


Figure 6.34. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 0.9)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

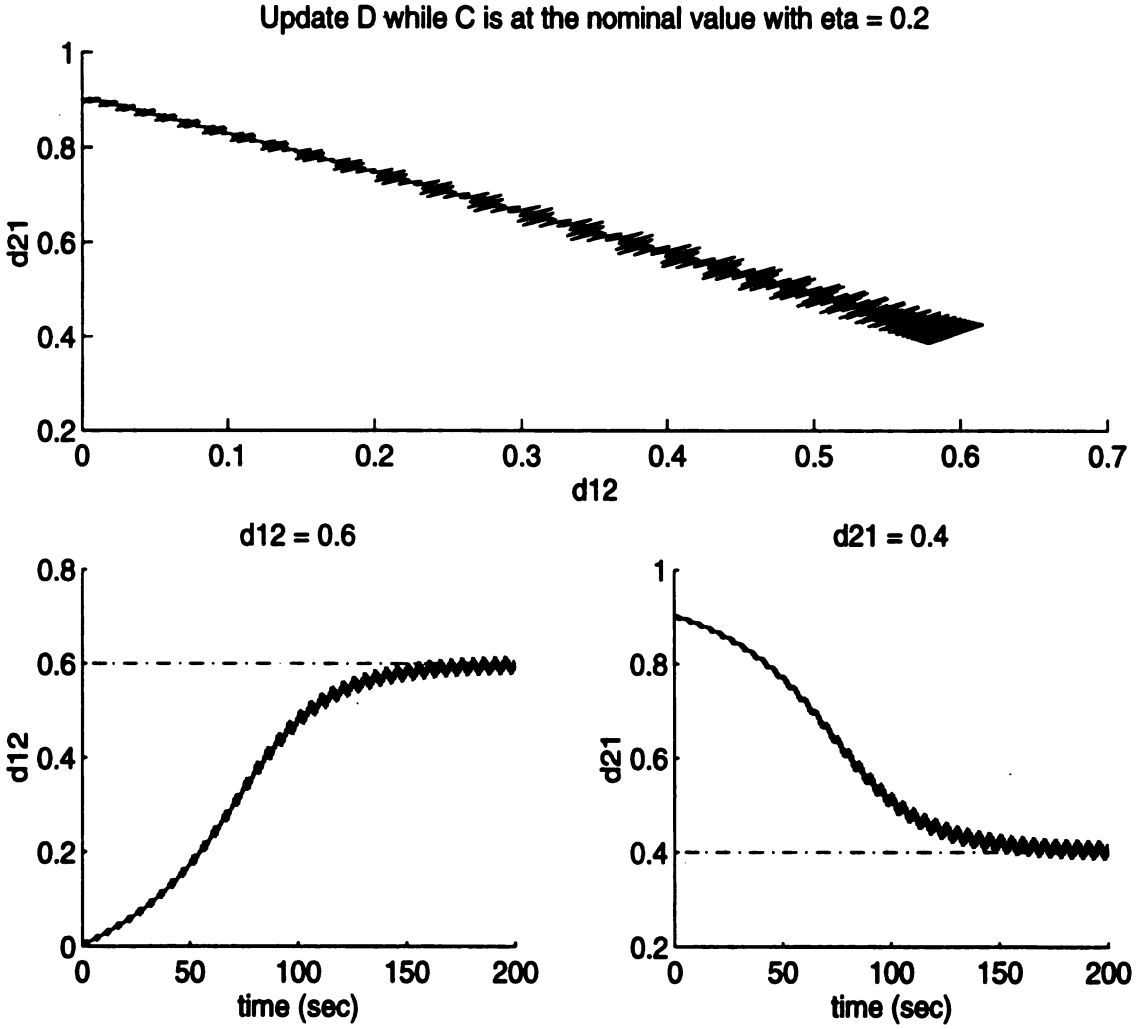


Figure 6.35. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0}, d_{21_0}) = (0.9, 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

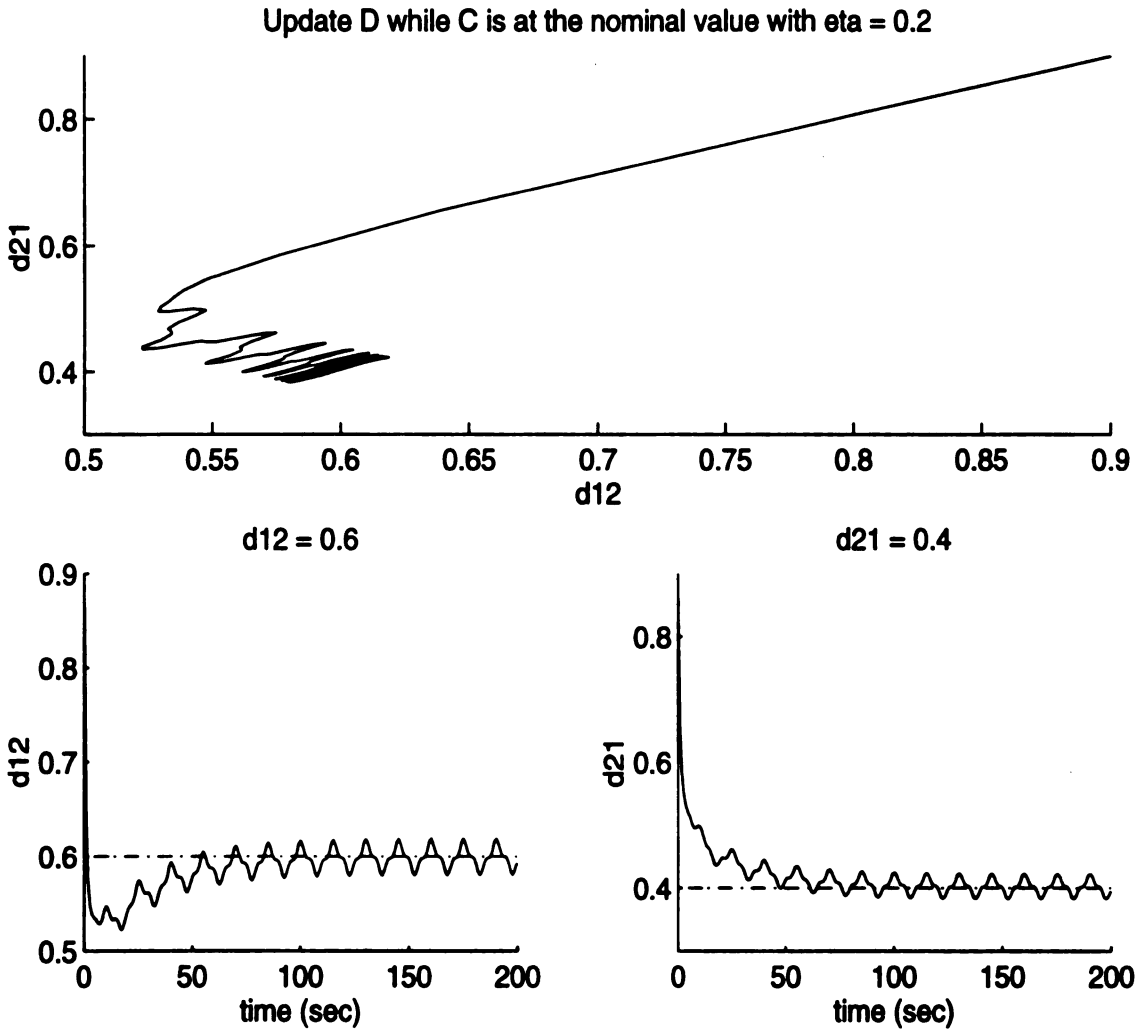


Figure 6.36. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0}, d_{21_0}) = (0.9, 0.9)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

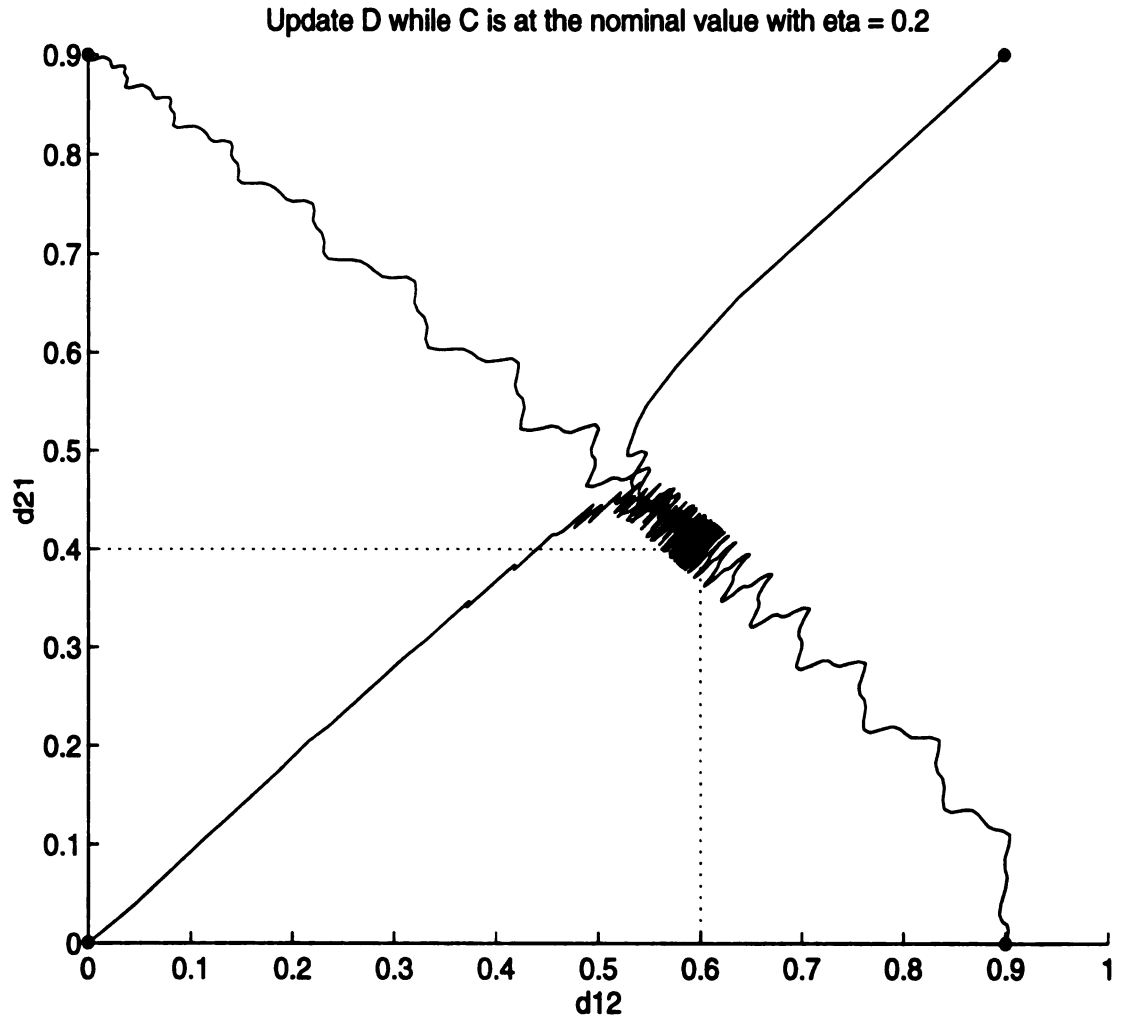


Figure 6.37. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$ and $\gamma = 0.2$ for different initial conditions. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = \sin(4\pi t)$

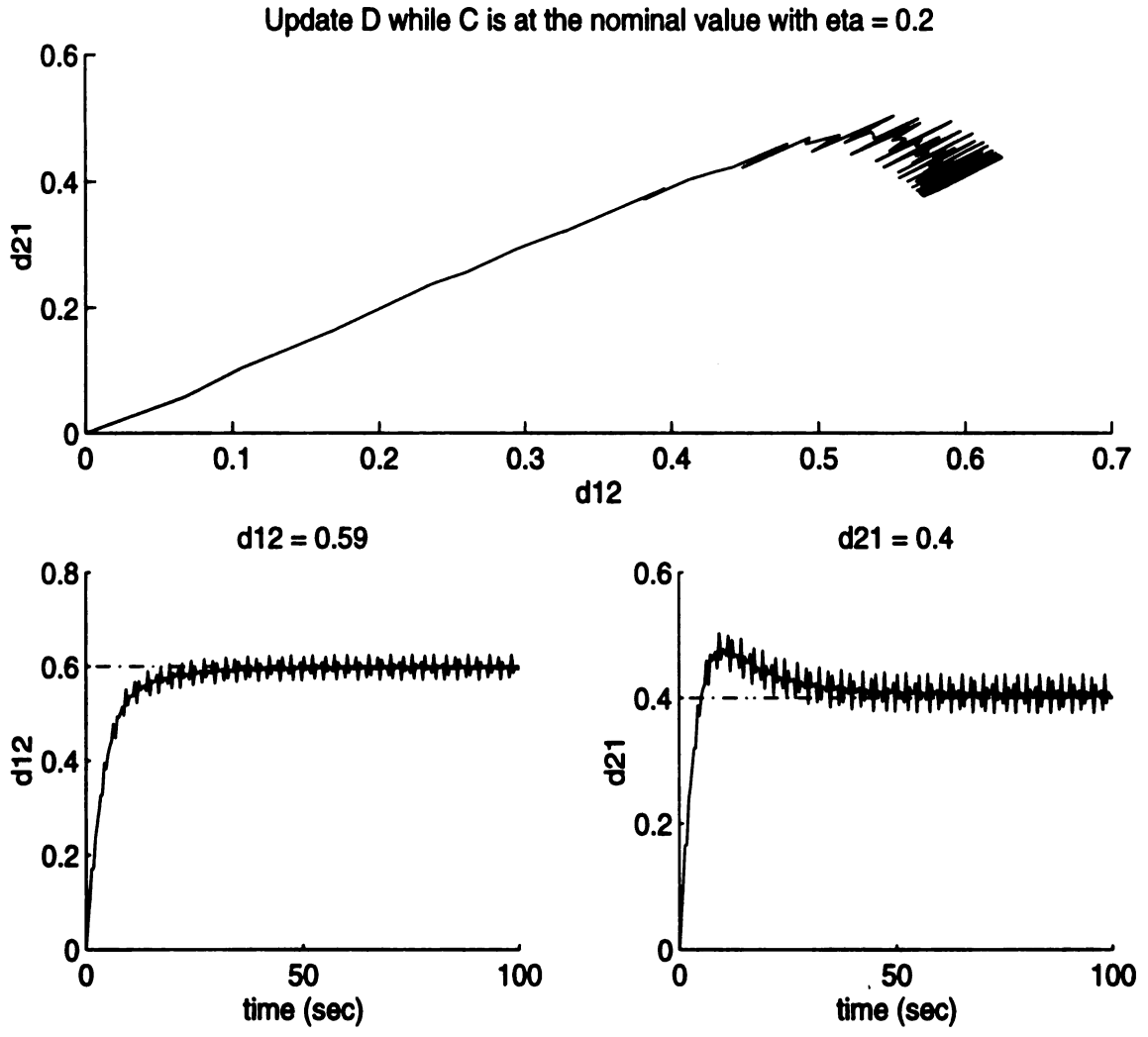


Figure 6.38. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = square(4\pi t)$

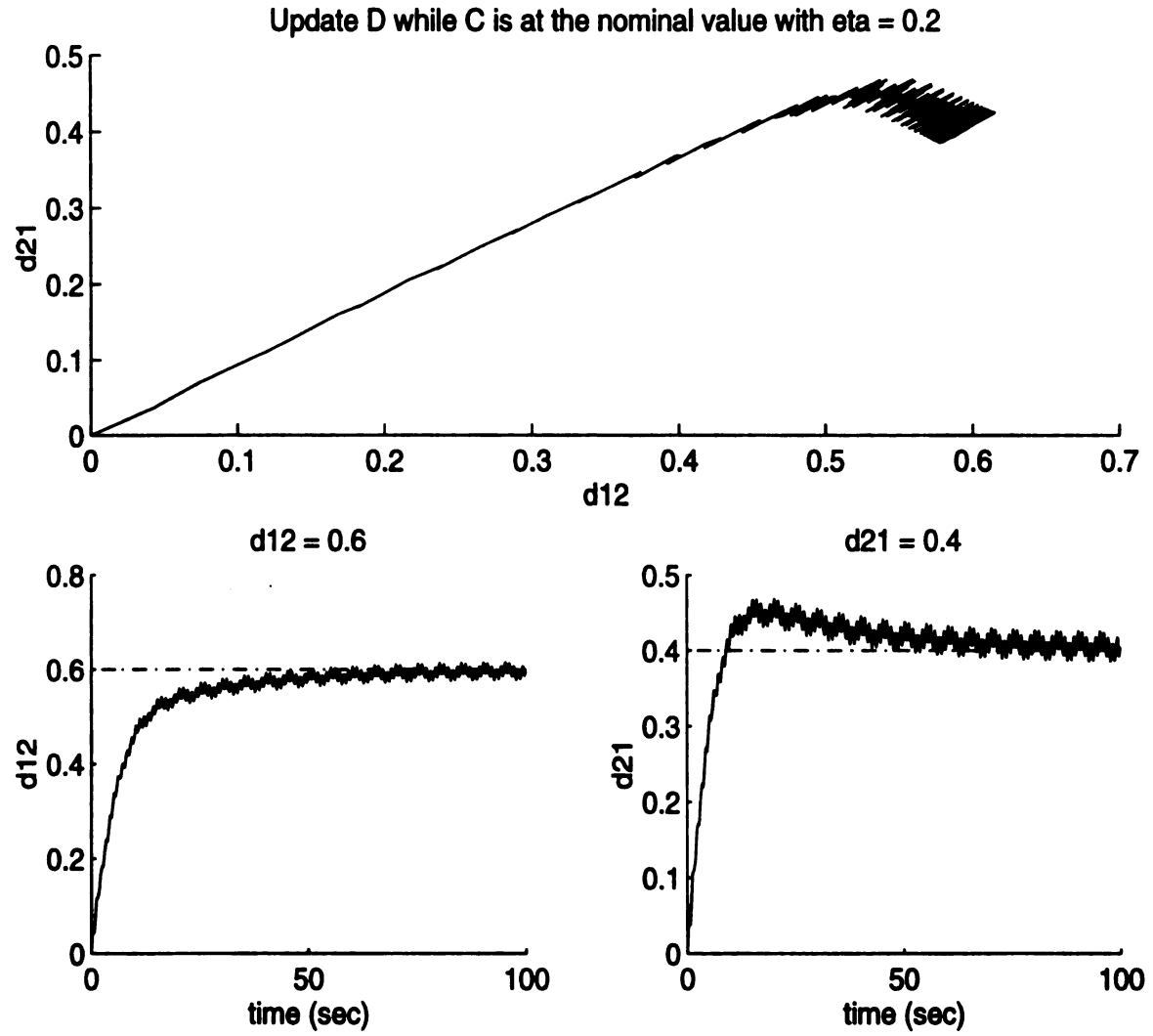


Figure 6.39. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0},\ d_{21_0}) = (0.0,\ 0.0)$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = sawtooth(4\pi t)$

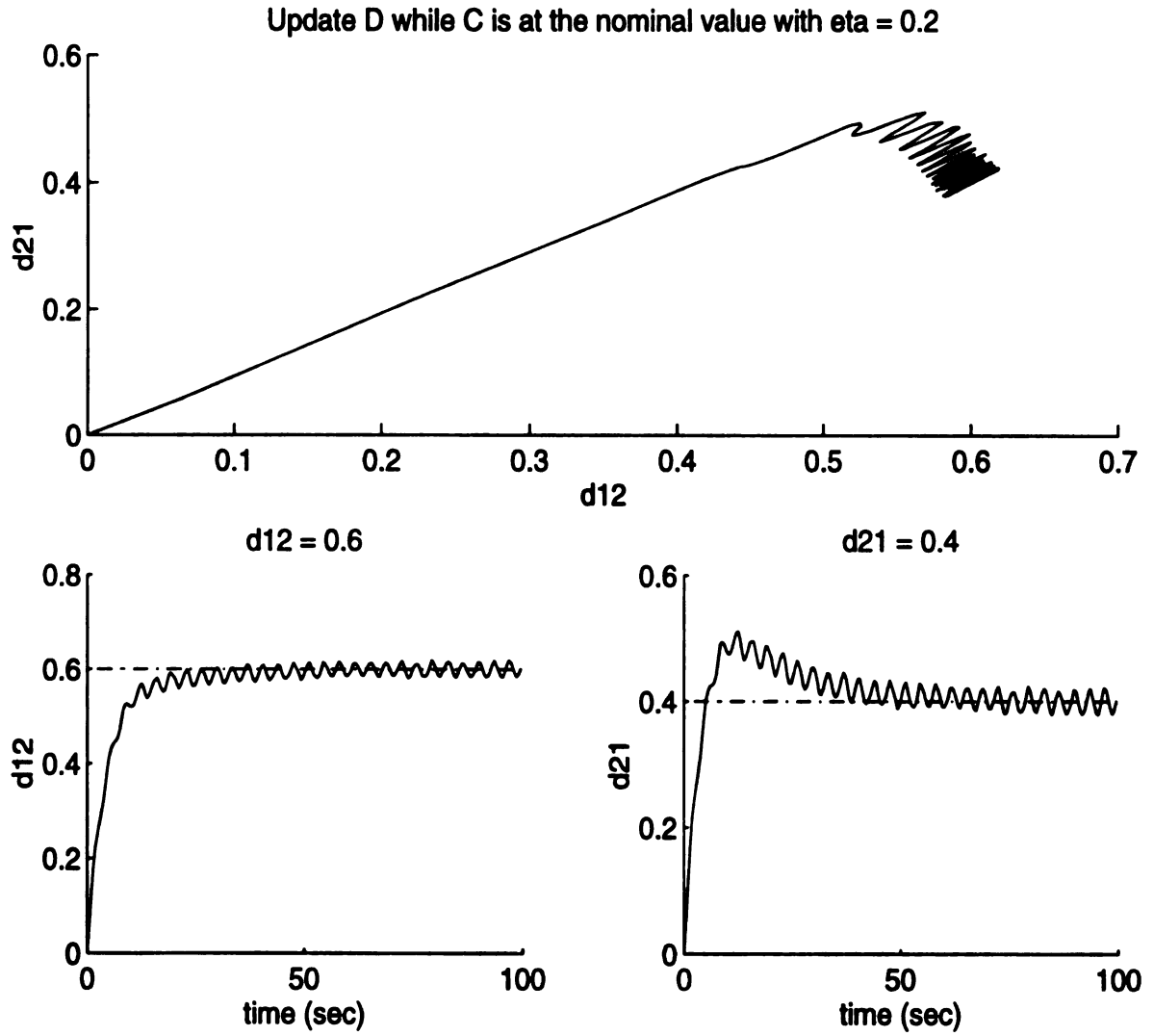


Figure 6.40. Update all the parameters of the D matrix according to $\dot{d}_{ij} = \gamma \sinh y_i \tanh y_j$, $\gamma = 0.2$ and $(d_{12_0}, d_{21_0}) = (0.0, 0.0)$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

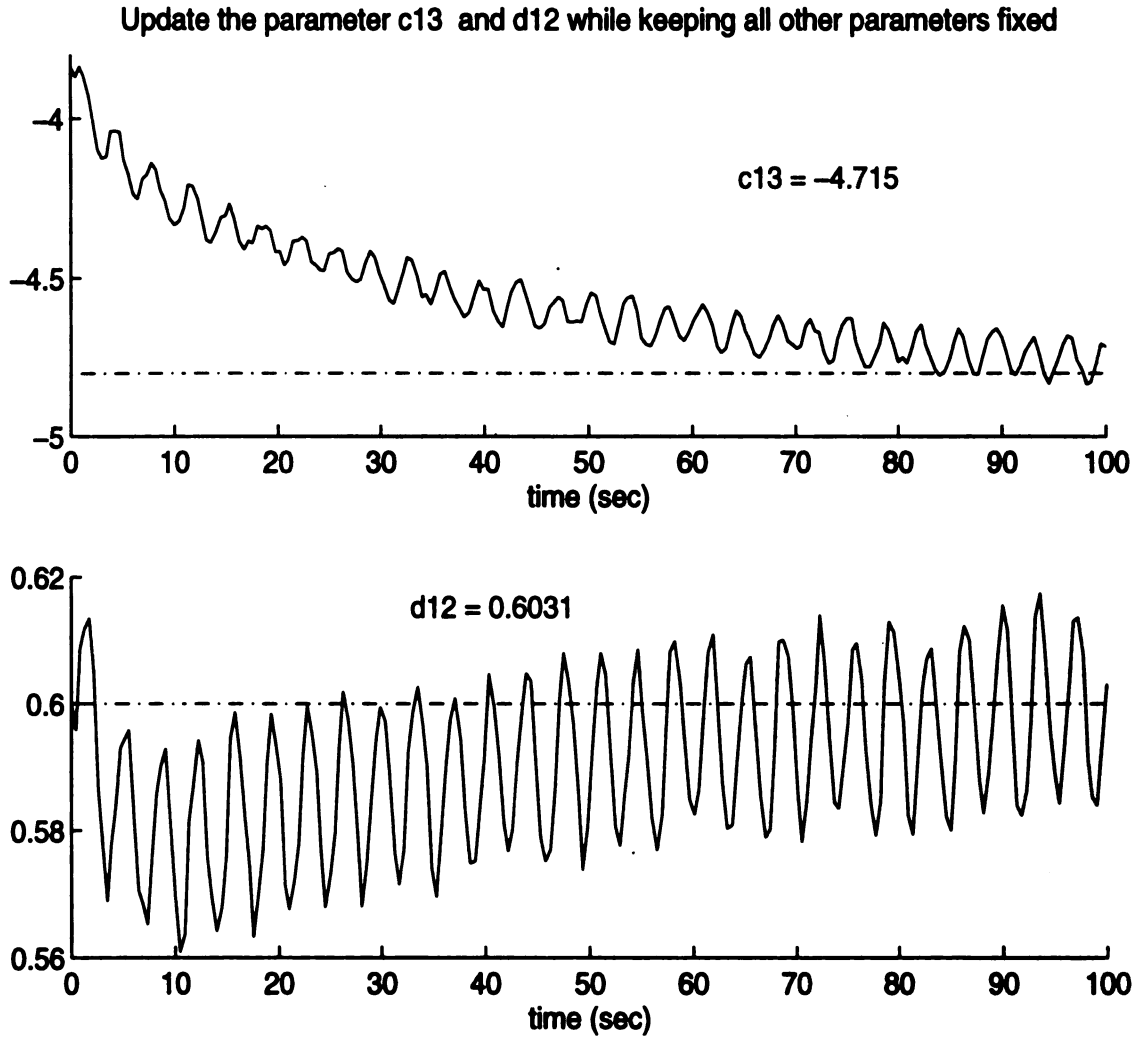


Figure 6.41. Update only one parameter of each matrix: update the parameters c_{13} and d_{12} with $\eta = 20$, $\gamma = 0.2$ and $(c_{13_0},\ d_{12_0}) = (0.8c_{13}^*,\ d_{12}^*)$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

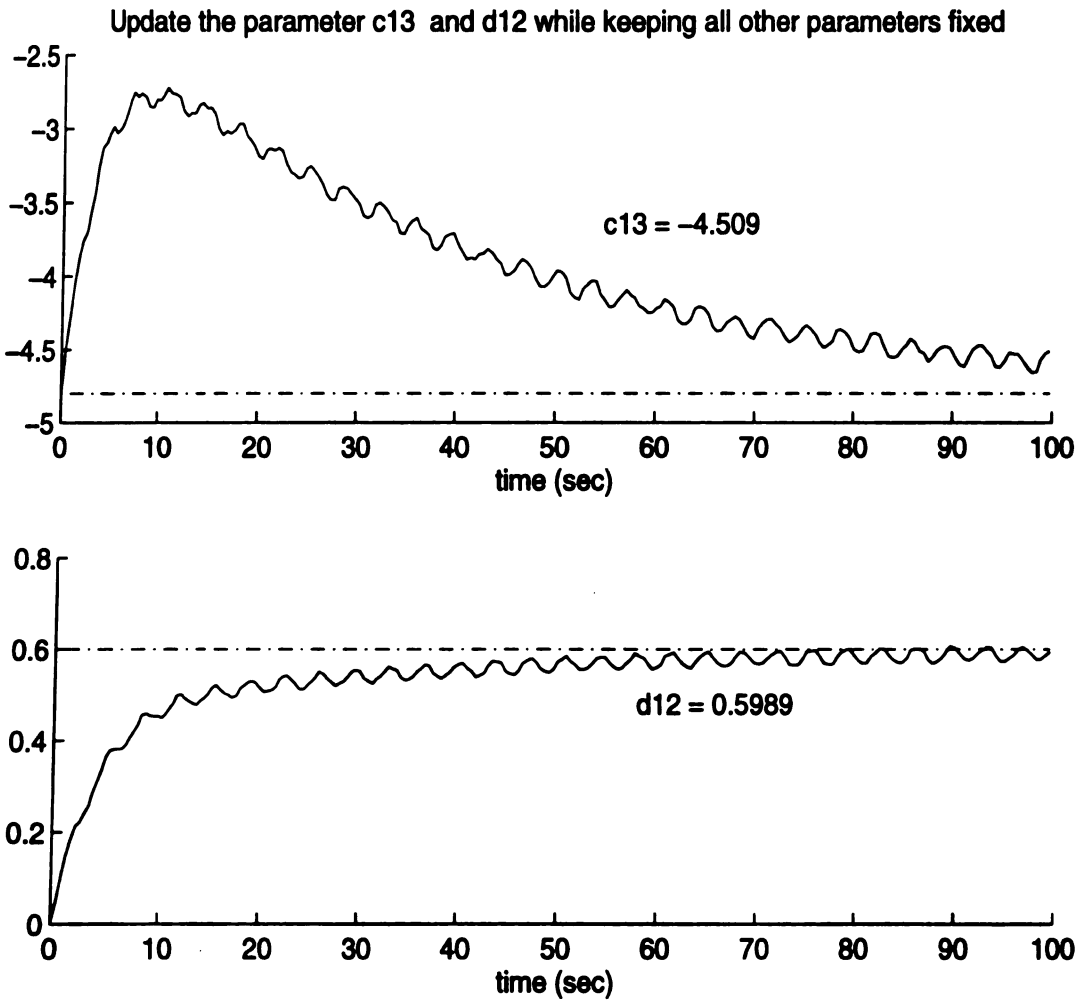


Figure 6.42. Update only one parameter of each matrix: update the parameters c_{13} and d_{12} with $\eta = 20$, $\gamma = 0.2$ and $(c_{13_0},\ d_{12_0}) = (c^*_{13},\ 0.0)$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

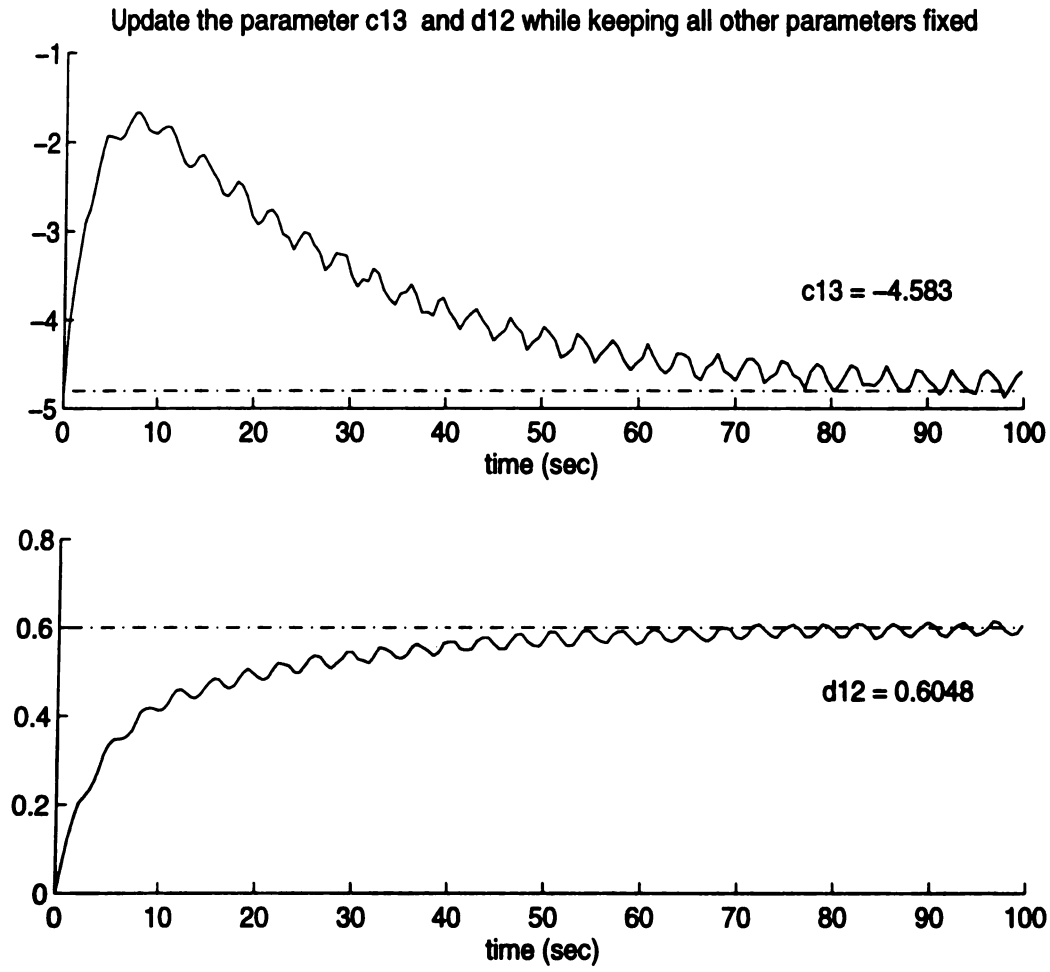


Figure 6.43. Update only one parameter of each matrix: update the parameters c_{13} and d_{12} with $\eta = 40$, $\gamma = 0.2$ and $(c_{13_0},\ d_{12_0}) = (c_{13}^*,\ 0.0)$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

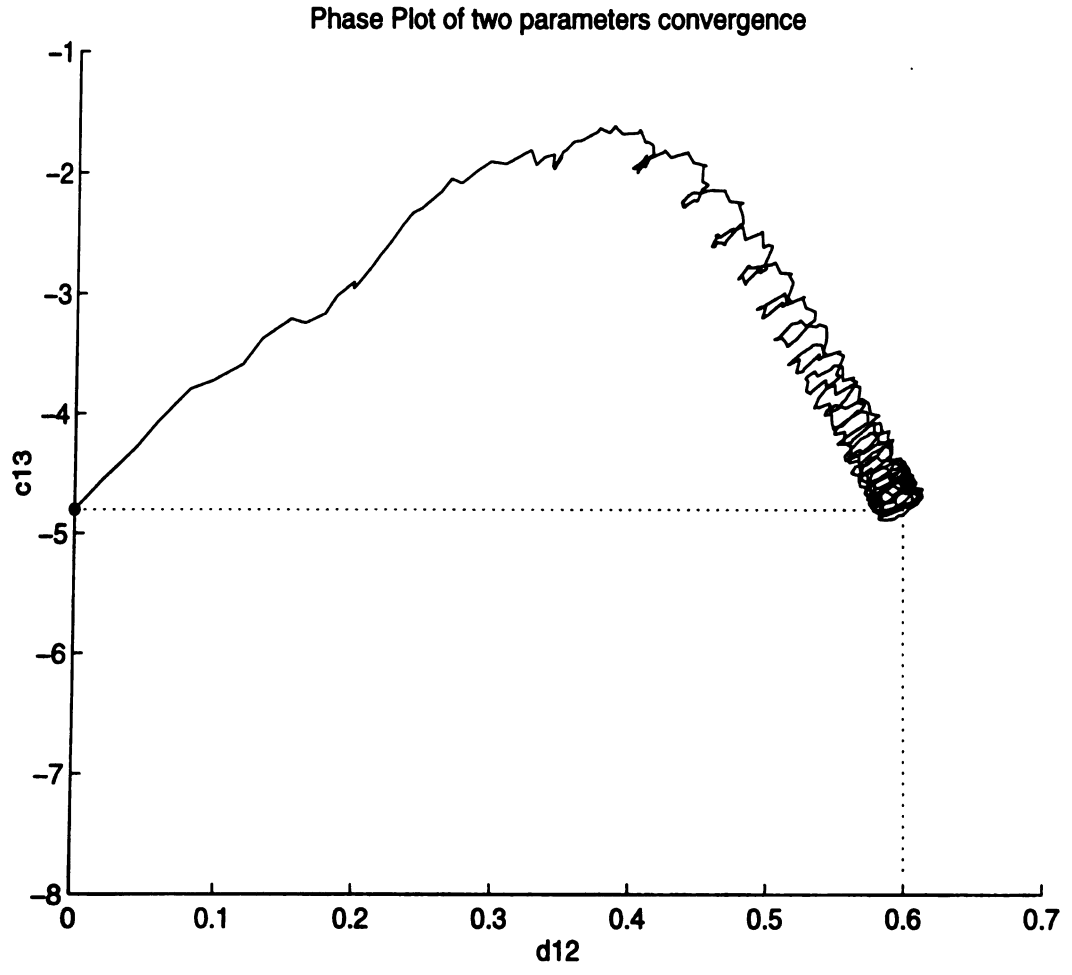


Figure 6.44. Update only one parameter of each matrix: update the parameters c_{13} and d_{12} with $\eta = 40$, $\gamma = 0.2$ and $(c_{13_0},\ d_{12_0}) = (c^*_{13},\ 0.0)$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

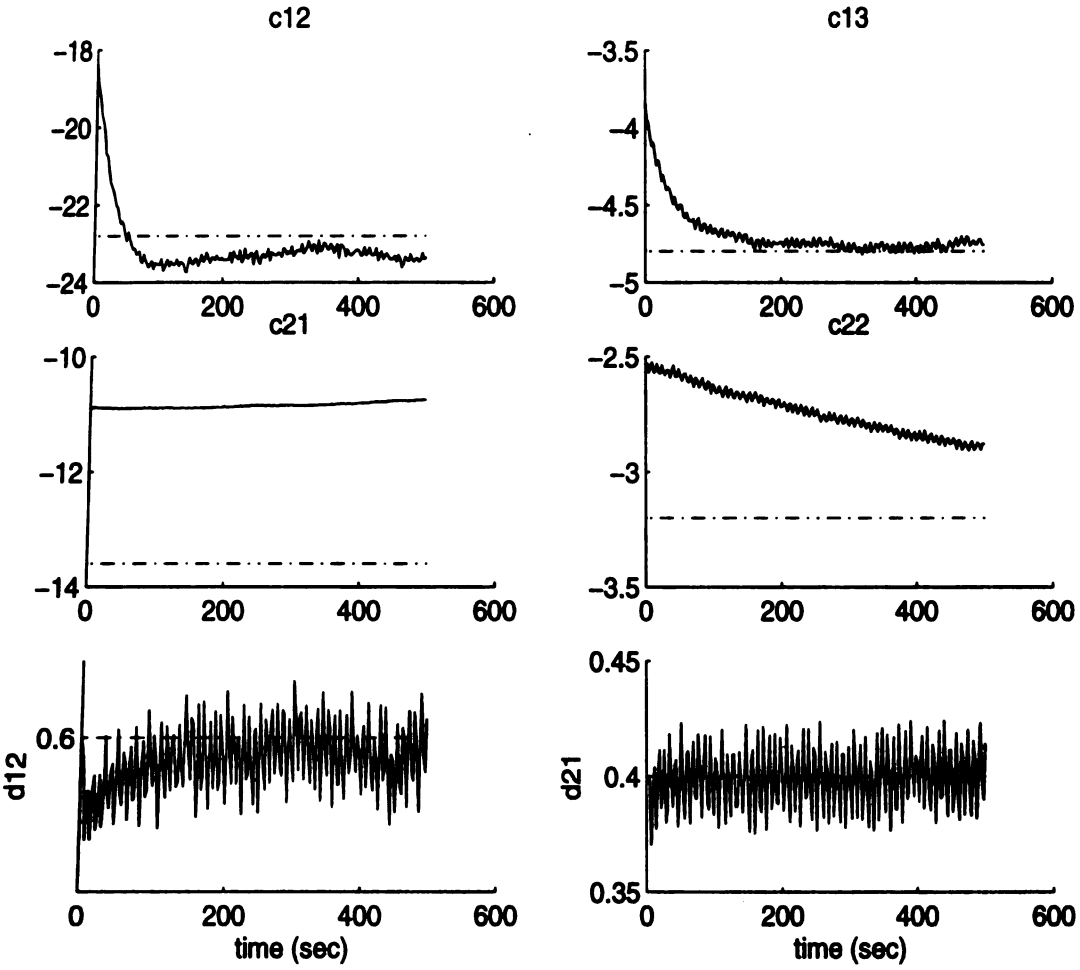


Figure 6.45. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = D^*$, $\eta = [10^3\ 10\ 10^2\ 10]$, $\gamma = 0.2$ and $T = 500$ The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

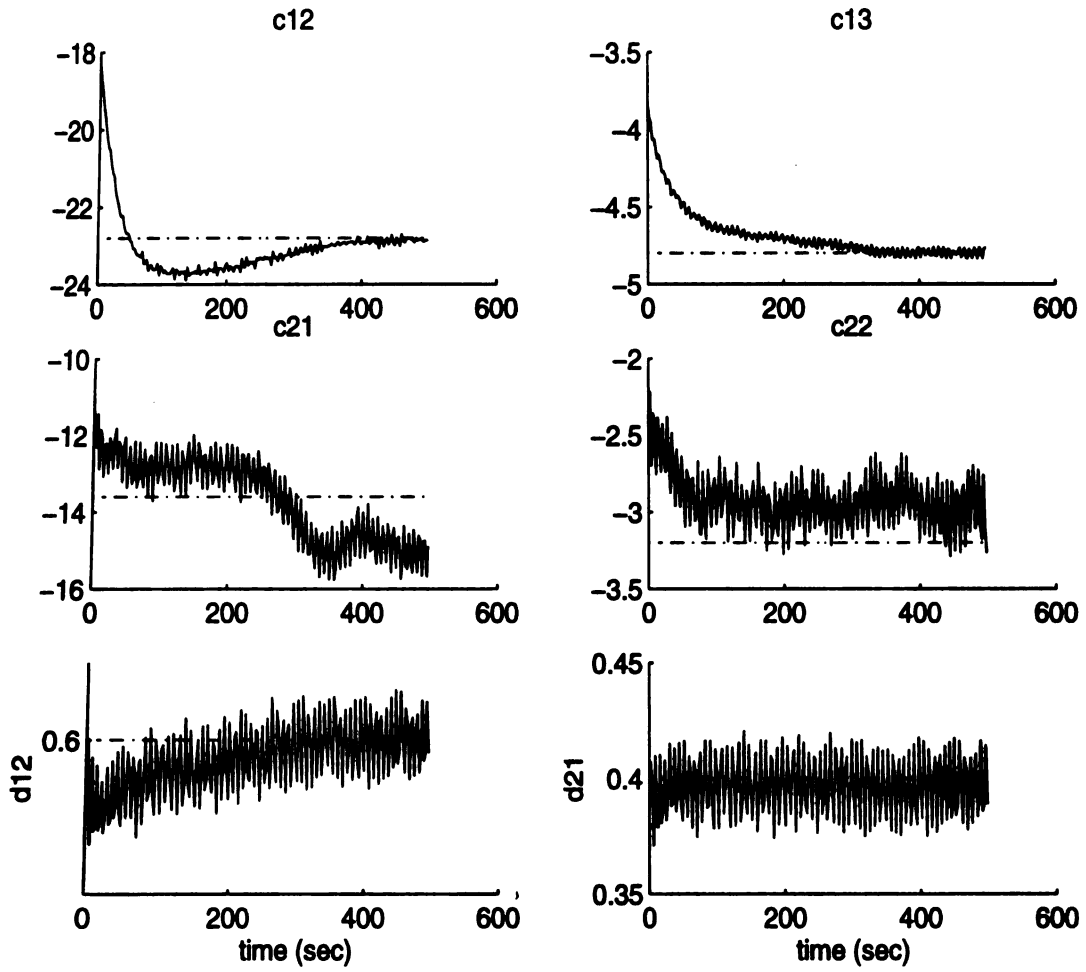


Figure 6.46. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = D^*$, $\eta = [10^3\ 10\ 10^4\ 10^2]$, $\gamma = 0.2$ and $T = 500$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

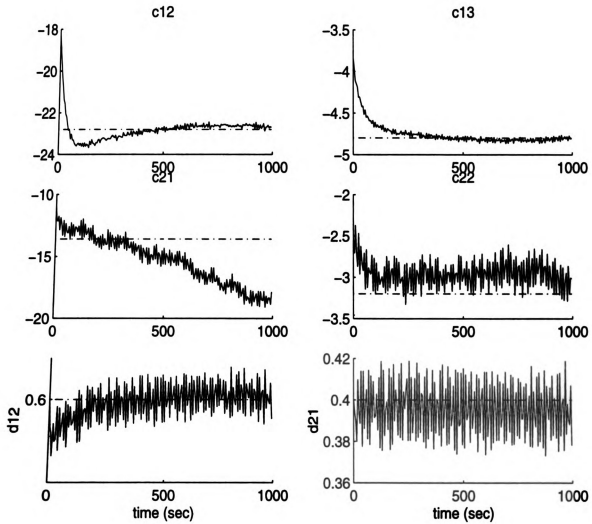


Figure 6.47. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = D^*$, $\eta = [10^3\ 10\ 10^4\ 10^2]$, $\gamma = 0.2$ and $T = 1000$. The unknown sources are $s_1(t) = \boldsymbol{sawtooth}(2\pi t)$ and $s_2(t) = square(4\pi t)$

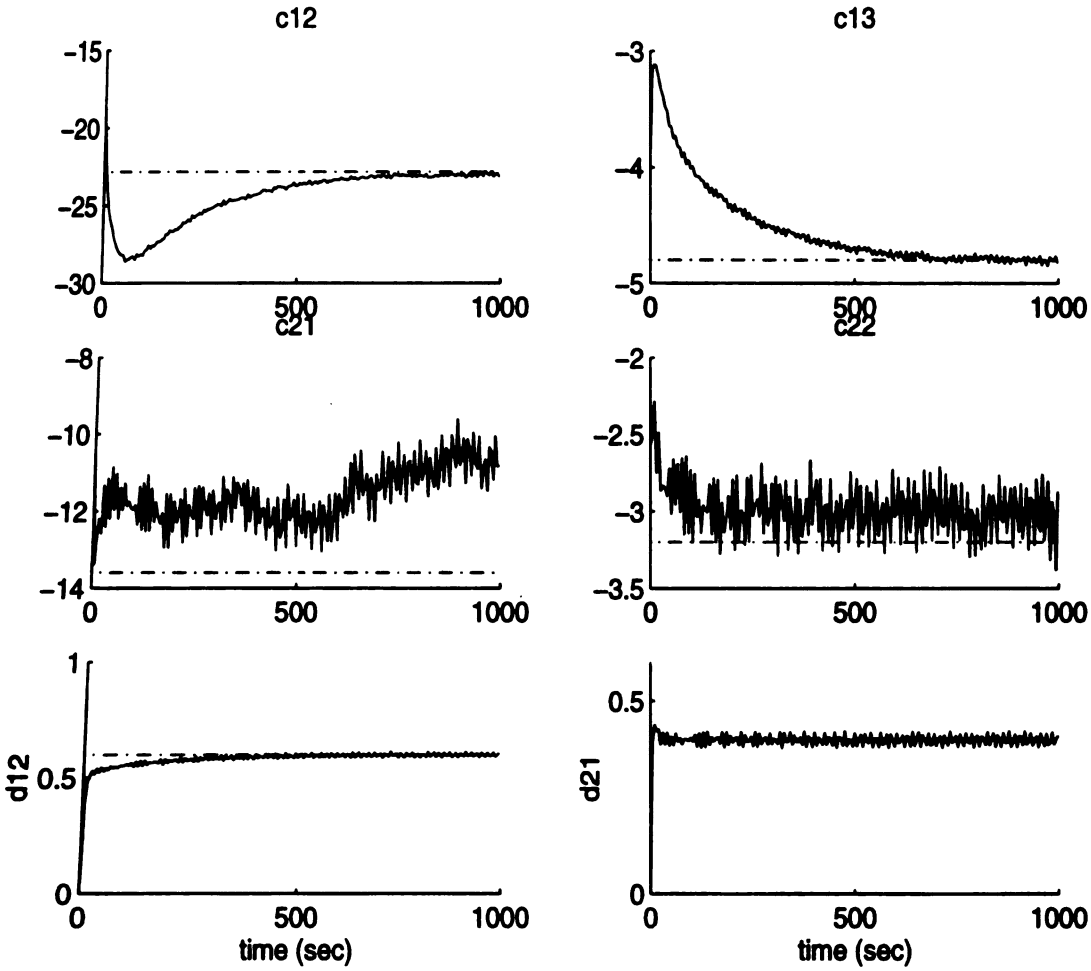


Figure 6.48. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [10^3 \ 10 \ 10^4 \ 10^2]$, $\gamma = 0.2$ and $T = 1000$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

6.6 Time and Weight Scaling

It has been observed that the algorithm fails to converge when the parameters of the C matrix are too large. This is not a limitation of the algorithm, but rather a limitation of the computer implementation. Thus, a scaling down of these parameters is proposed in order to make them equal in order of magnitude and in a range that is acceptable for digital hardware implementation.

6.6.1 Mathematical Development

Given a single input-single output dynamical system described by its transfer function

$$H(s) = \frac{\beta_1 s^{n-1} + \beta_2 s^{n-2} + \cdots + \beta_{n-1} s + \beta_n}{s^n + \alpha_1 s^{n-1} + \alpha_2 s^{n-2} + \cdots + \alpha_{n-1} s + \alpha_n} + d$$

Consider its canonical controllable realization $\mathcal{D} = (A, \mathbf{b}, \mathbf{c}^T, d)$. Such realization was presented in Chapter 5 to be of the form

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ -\alpha_n & \cdots & \cdots & \cdots & -\alpha_1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} \beta_n & \beta_{n-1} & \cdots & \beta_2 & \beta_1 \end{bmatrix} \mathbf{x} + du$$

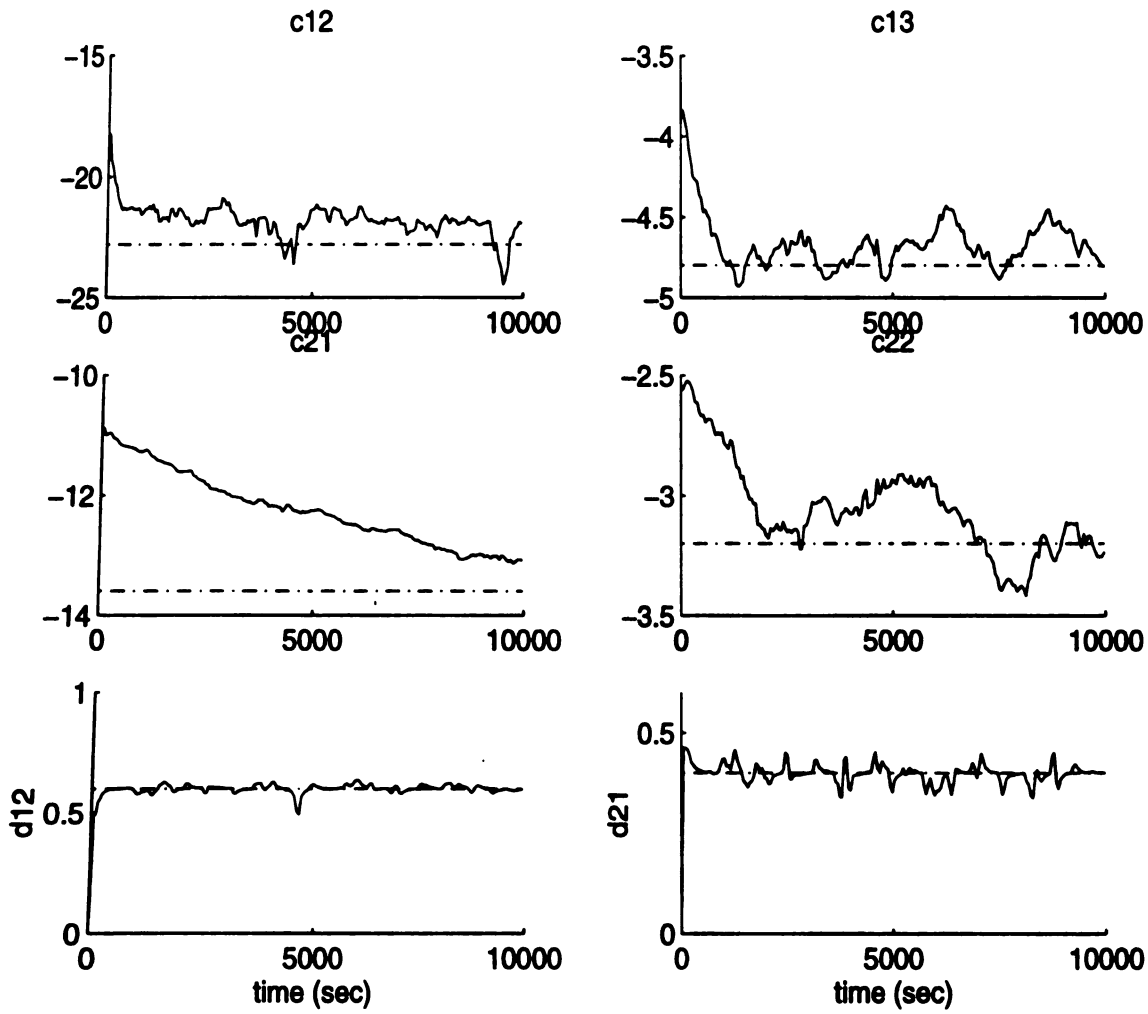


Figure 6.49. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [1000\ 10\ 100\ 10]$, $\gamma = 0.2$ and $T = 10000$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

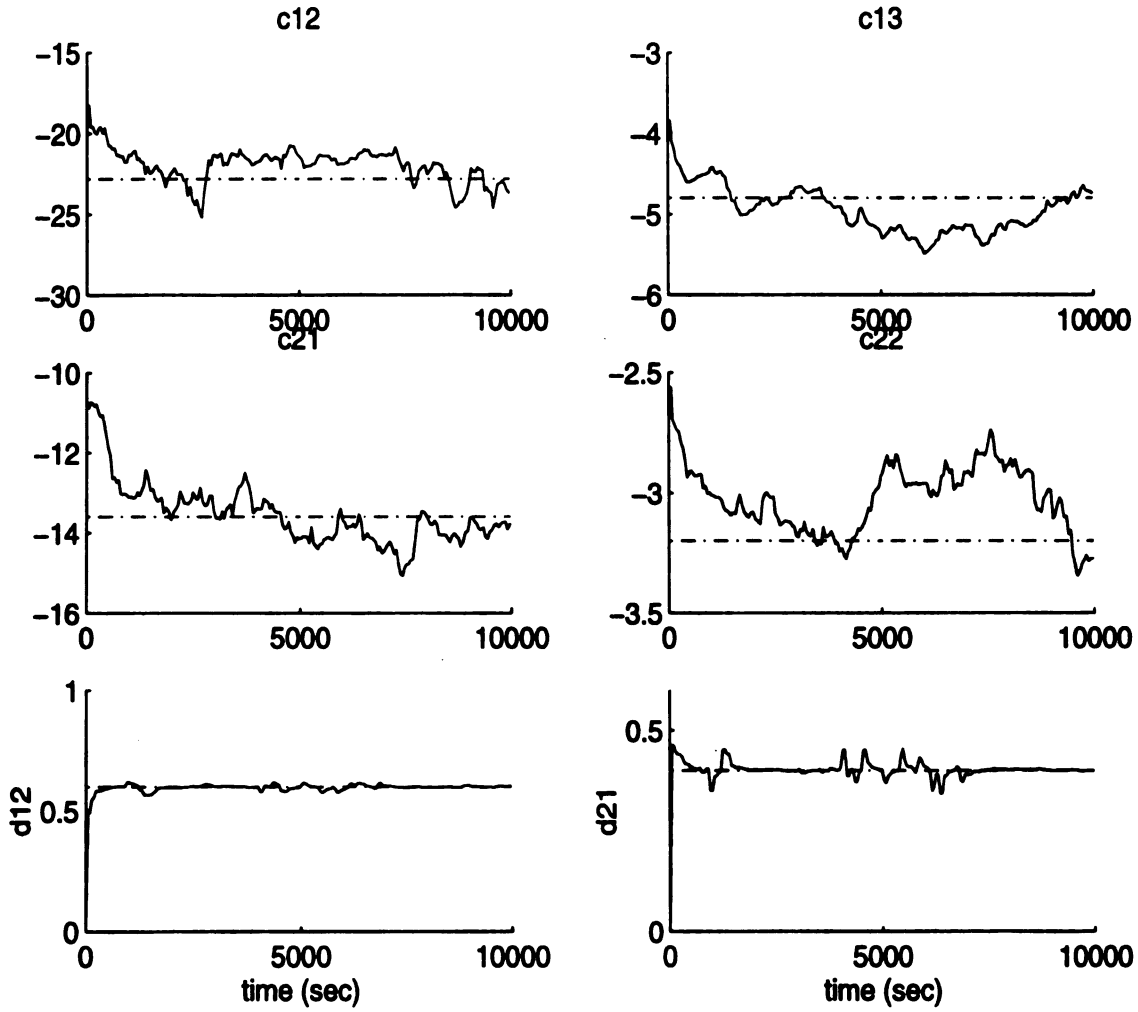


Figure 6.50. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [1000\ 10\ 500\ 10]$, $\gamma = 0.2$ and $T = 10000$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

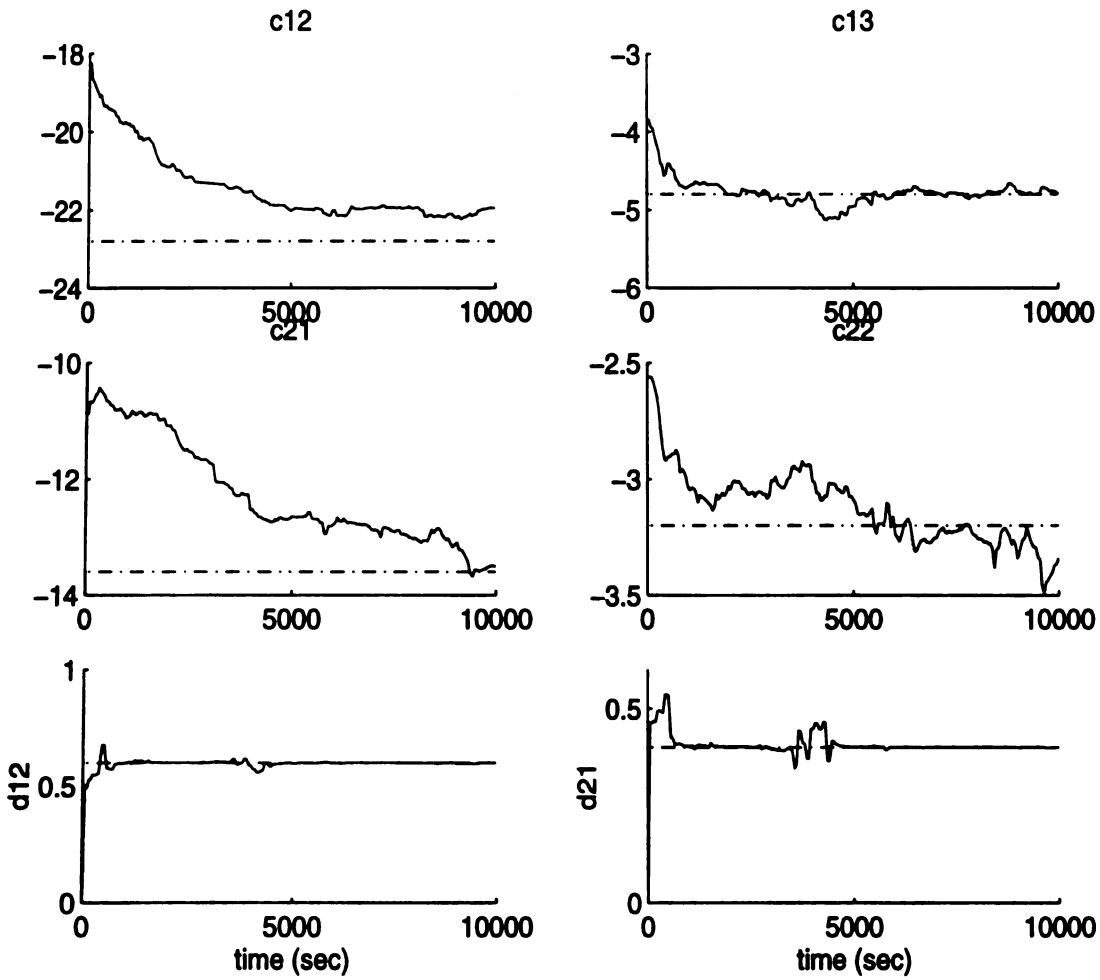


Figure 6.51. Update all parameters of each matrix. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [200\ 10\ 200\ 10]$, $\gamma = 0.2$ and $T = 10000$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

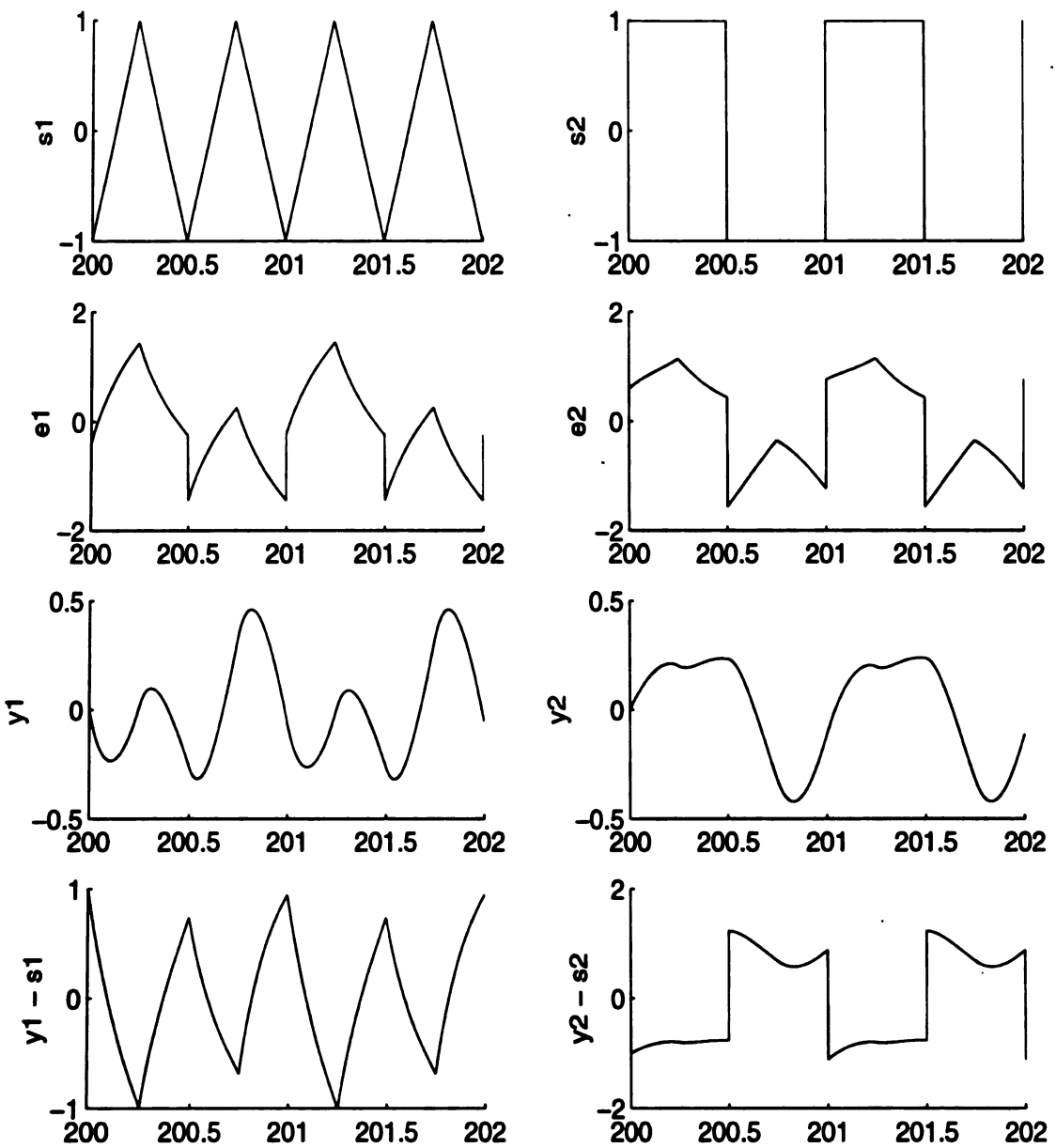


Figure 6.52. Unknown sources, environment output and Network output at initial time $t = 0$. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [200\ 10\ 200\ 10]$, $\gamma = 0.2$ and $T = 10000$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

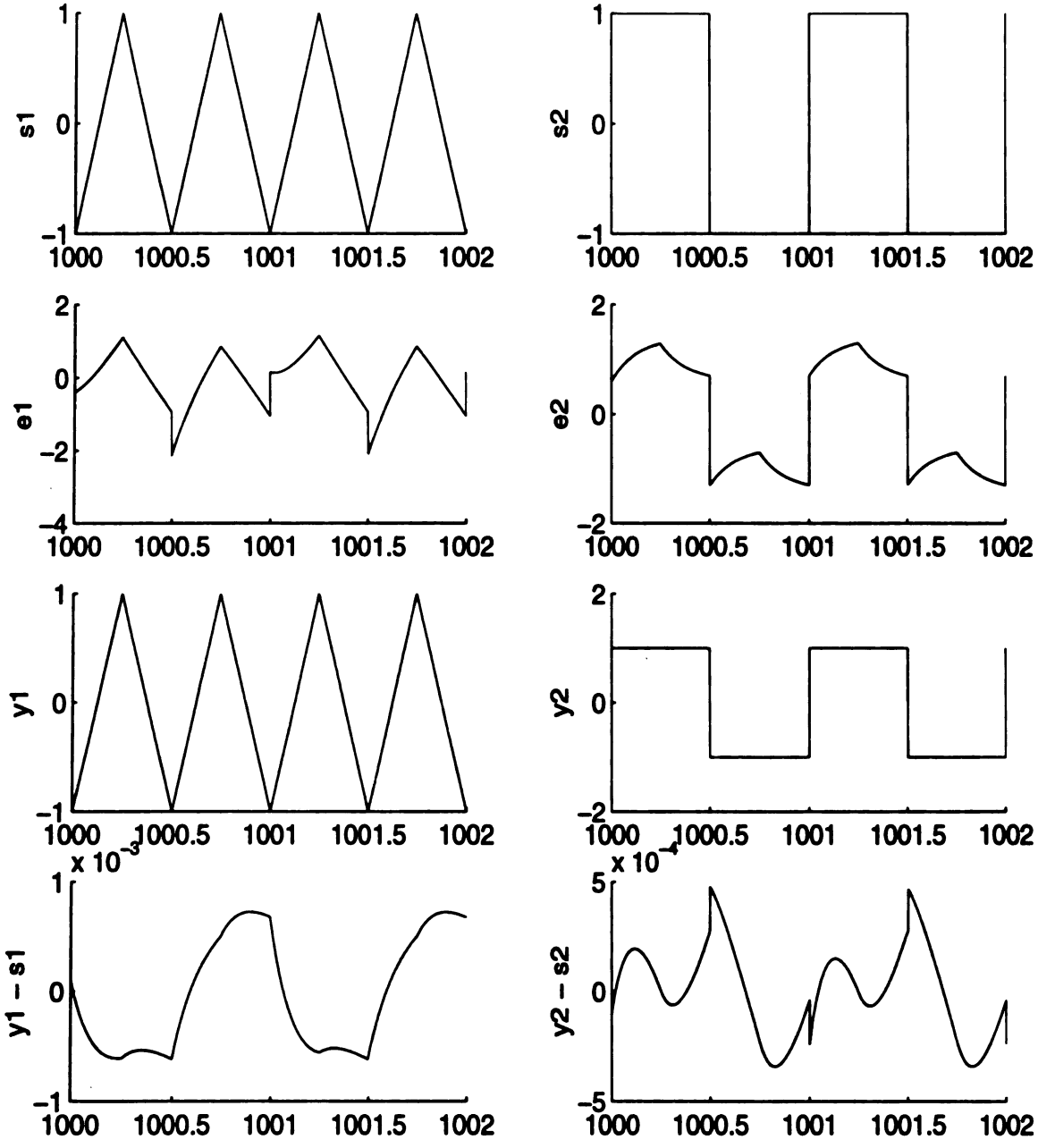


Figure 6.53. Unknown sources, environment output and Network output at final time $t = T$. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [200\ 10\ 200\ 10]$, $\gamma = 0.2$ and $T = 10000$. The unknown sources are $s_1(t) = sawtooth(2\pi t)$ and $s_2(t) = square(4\pi t)$

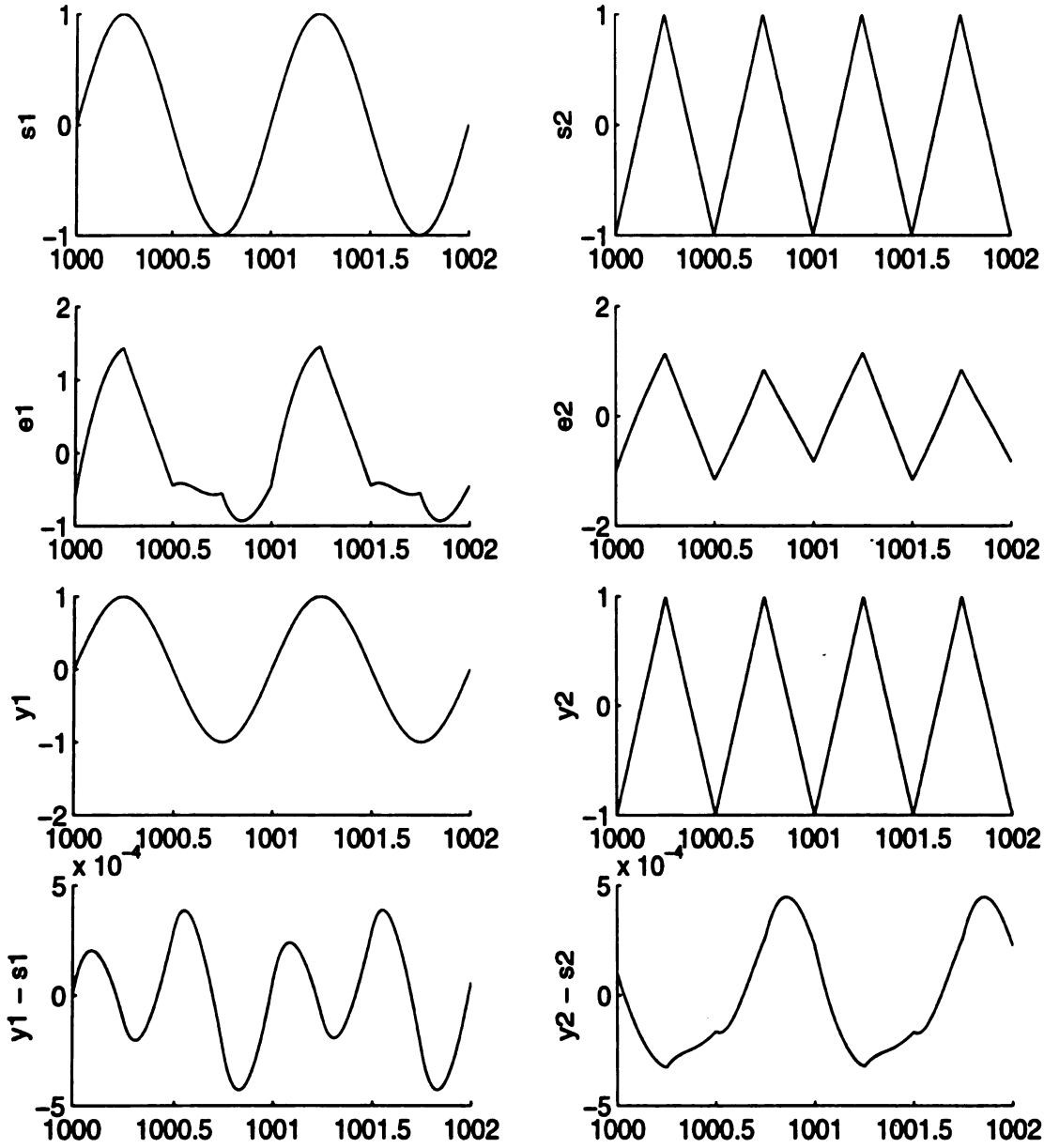


Figure 6.54. Test case: Unknown sources, environment output and Network output at final time $t = T$. $C_0 = 0.8C^*$ and $D_0 = 0$, $\eta = [200\ 10\ 200\ 10]$, $\gamma = 0.2$ and $T = 10000$. The unknown sources are $s_1(t) = \sin(2\pi t)$ and $s_2(t) = sawtooth(4\pi t)$

The diagonal transformation $T_{\epsilon,\kappa}$ is defined as

$$T_{\epsilon,\kappa} = \kappa \begin{bmatrix} \epsilon^{n-1+} & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & 0 \\ \vdots & 0 & \epsilon^{n-i} & 0 & \vdots \\ 0 & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Under the transformation $T_{\epsilon,\kappa}$, the new realization of $H(s)$ in the new coordinate system is $\tilde{\mathcal{D}} = (\tilde{A},\ \tilde{\mathbf{b}},\ \tilde{\mathbf{c}}^T,\ \tilde{d})$, defined as

$$\tilde{A} = T_{\epsilon,\kappa} A T_{\epsilon,\kappa}^{-1} = \begin{bmatrix} 0 & \epsilon & 0 & \cdots & 0 \\ 0 & 0 & \epsilon & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \epsilon \\ -\tilde{\alpha}_n & \cdots & \cdots & \cdots & -\tilde{\alpha}_1 \end{bmatrix} \qquad \tilde{\mathbf{b}} = T_{\epsilon,\kappa}\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \kappa \end{bmatrix}$$

$$\tilde{\mathbf{c}}^T = \mathbf{c}^T T_{\epsilon,\kappa}^{-1} = \begin{bmatrix} \tilde{\beta}_n & \tilde{\beta}_{n-1} & \cdots & \tilde{\beta}_2 & \tilde{\beta}_1 \end{bmatrix} \qquad \tilde{d} = d$$

where

$$\tilde{\alpha}_i = \alpha_i \epsilon^{1-i},\ \forall i = 1, \cdots, n$$

and

$$\tilde{\beta}_i = \frac{\beta_i \epsilon^{n-i}}{\kappa},\ \forall i = 1, \cdots, n$$

One observes that under such transformation, the coefficients of

- $\tilde{A}$ and $\tilde{\mathbf{c}}$ are both inversely proportional to some power of ϵ,

- the nonzero parameter of $\tilde{\mathbf{b}}$ is proportional to κ while those of $\tilde{\mathbf{c}}$ are inversely proportional to κ,
- the parameters of $\tilde{d}$ are independent of ϵ and κ,
- the parameters of $\tilde{A}$ are independent of κ.

Using this transformation, one could scale the parameters that are to be updated up or down in order to make their numerical computation possible by digital machines. The scaling is achieved by the appropriate choice of ϵ and depends on the application for which the problem is being set. For example, the parameters can be scaled down by choosing a value for ϵ less than 1. Thus, there is no optimum value of ϵ that can be used for all possible environments. However, one should have some knowledge of the desired environment and use that information to select the appropriate value of ϵ. Such information could be based on the frequency bandwidth of the environment. In the following example, it will be shown how the choice of ϵ can be considered. κ can be considered as a normalizing coefficient for the parameters of $\tilde{\mathbf{c}}$.

6.6.2 Example

The following example illustrates the necessity of scaling in order to make the digital implementation of the algorithm possible. Given the transfer function

$$H(s) = \frac{(s+100)(s+200)}{(s+1000)(s+2000)(s+3000)} + 0.6$$

or in an expanded form

$$= \frac{s^2 + 3 \times 10^2 s + 2 \times 10^4}{s^3 + 6 \times 10^3 s^2 + 11 \times 10^6 s + 6 \times 10^9} + 0.6$$

The realization of $H(s)$ is

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6\times10^{9} & -11\times10^{6} & -6\times10^{3} \end{bmatrix} \qquad \mathbf{b} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c}^T = \begin{bmatrix} 2\times10^{4} & 3\times10^{2} & 1 \end{bmatrix} \qquad d = 0.6$$

However, by considering the transformation $T_{\epsilon,\kappa}$ with $\epsilon = 10^3$ and $\kappa = 1$, $H(s)$ can be realized by

$$\tilde{A} = \begin{bmatrix} 0 & 10^{3} & 0 \\ 0 & 0 & 10^{3} \\ -6\times10^{3} & -11\times10^{3} & -6\times10^{3} \end{bmatrix} \qquad \tilde{\mathbf{b}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{c}}^T = \begin{bmatrix} 0.02 & 0.3 & 1 \end{bmatrix} \qquad \tilde{d} = 0.6$$

and by considering the transformation $T_{\epsilon,\kappa}$ with $\epsilon = 10^2$ and $\kappa = 1$, $H(s)$ can be realized by

$$\tilde{A} = \begin{bmatrix} 0 & 10^{2} & 0 \\ 0 & 0 & 10^{2} \\ -6\times10^{5} & -11\times10^{4} & -6\times10^{3} \end{bmatrix} \qquad \tilde{\mathbf{b}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{c}}^T = \begin{bmatrix} 2 & 3 & 1 \end{bmatrix} \qquad \tilde{d} = 0.6$$

One may conclude that in order to scale the parameters of the matrix $\tilde{\mathbf{c}}$, one should choose ϵ on the order of the zeros. Whereas ϵ should be on the order of the poles of the transfer function if one wishes to scale the parameters of the matrix A. While ϵ accomplishes the time scaling task, κ performs magnitude scaling. By taking a value

for κ different than 1, one could increase (decrease) the parameters of the $\tilde{\mathbf{c}}$ ($\tilde{\mathbf{b}}$) vector and vice versa.

6.7 Observations

In this chapter, a solution to the blind separation of signals in dynamical network model was developed. The network parameters of the proposed feedback structure, with the update law as defined by Theorem 2, has shown to converge to the desired values. In addition, other update laws based on neuromemitic approach were considered in order to provide a comparative performance between the developed algorithm and the existing ones in the literature. Thanks to the special structure of the state space realization of the network model, an update law of the C matrix based on the decorrelation between the output vector and the state vector was developed. Also, such a representation of the network model is suitable for circuit implementation which will be presented in the following chapter.

CHAPTER 7

Micro-electronic Implementation

7.1 CMOS Building Blocks

7.1.1 Transistor

The transistor represents the basic building block of a circuit. Designers must have a good understanding of its functionality in order to produce any circuit. Models which accurately define the characteristics of the transistor can be extremely complex and not at all possible for manual calculation. On the other hand, if a computer is employed to handle such a complex model, circuit design can become simple enough to model it.

The general expression for the drain-to-source current of a transistor operation in a sub-threshold region is given below

$$I_{ds} = I_o \, e^{\kappa V_g} \left(e^{-V_s} - e^{-V_d} \right)$$

At saturation, the drain voltage V_d is much higher than the gate voltage V_g. Therefore, the above expression of drain-to-source current at saturation is

$$I_{ds} = I_o \, e^{\kappa V_g - V_s}$$

7.1.2 Current mirror

In circuit design, it is often necessary to have a copy of a current to perform different operations on that particular current. Thus, a circuit that provides a replica of a current is needed. The circuit shown in Figure 7.1 performs such a function. Both n-channel and p-channel current mirrors are shown.

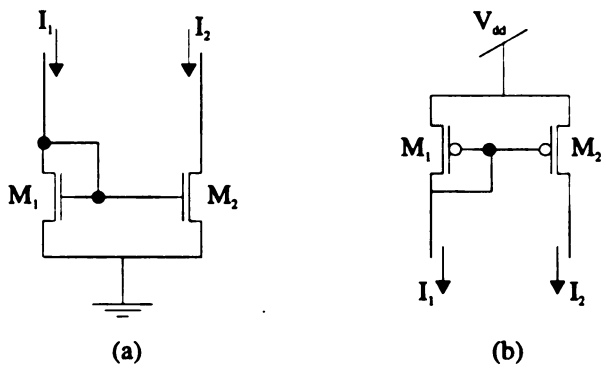


Figure 7.1. Current Mirror (a) n-channel, (b) p-channel

7.1.3 Transconductance Amplifier

Figure 7.2(a) represents the circuit diagram of a transconductance amplifier. The circuit will exhibit a sigmoidal function of the differential voltage V_1 and V_2 presented at the gates of the transistors M_1 and M_2 , respectively. The circuit is being driven by a biasing current I_b which is controlled by the voltage V_b applied to the transistor M_b.

$$I_{out} = I_b \tanh \frac{\kappa}{2}(V_1 - V_2) \tag{7.1}$$

where

$$I_b = wI_o\, e^{\kappa V_b} \tag{7.2}$$

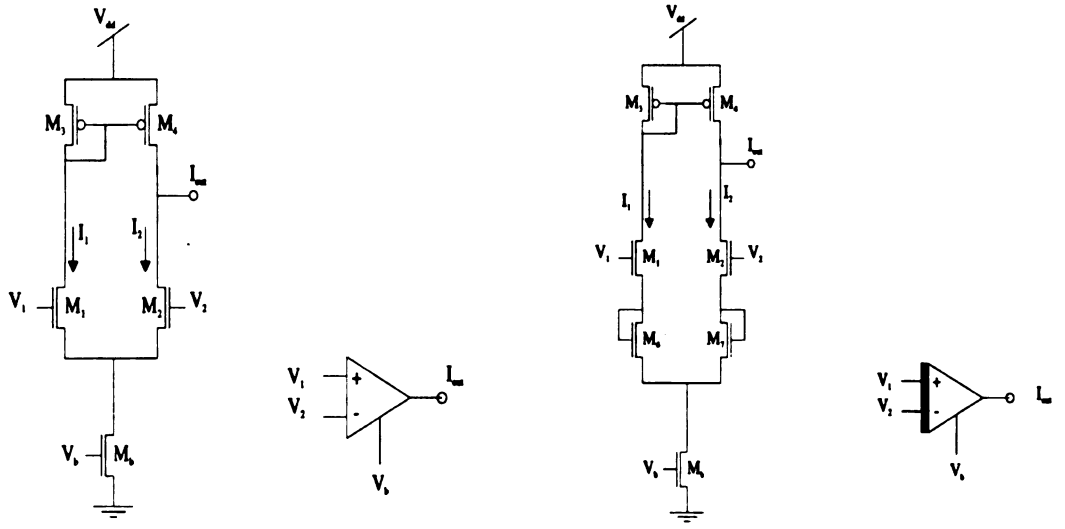
w is the width to length ratio of the biasing transistor M_b. For small-signal analysis the current-voltage relationship of a transconductance amplifier described by equation (7.1) is simply

$$I_{out} = g_m V_{in} \tag{7.3}$$

where $V_{in} = V_1 - V_2$ and g_m is the transconductance defined as

$$g_m = \frac{\partial I_{out}}{\partial V_{in}}|_{V_{in}=0} = \frac{I_b}{2kT/(q\kappa)} \tag{7.4}$$

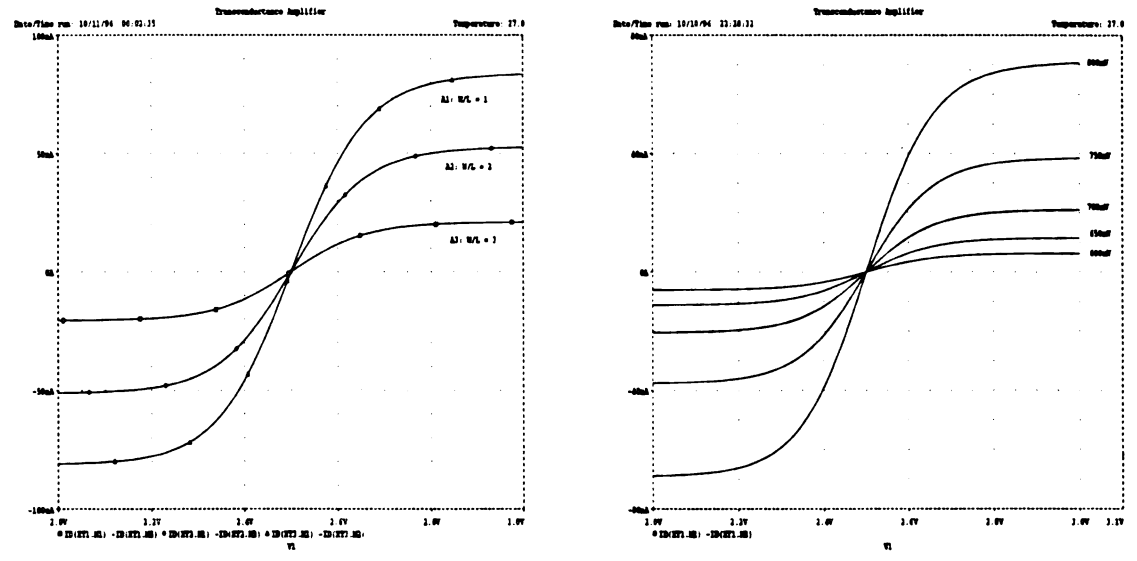
It is important that the transconductance depends on the biasing voltage V_b. This pa-



(a) Original Transconductance Amplifier (b) Improved Transconductance Amplifier

Figure 7.2. Transconductance Design

rameter will be the key to adjusting the behavior of circuits that include the transconductance amplifiers as will be shown in the next section. In [69], the author proposed an improved design of the transconductance amplifier as shown in Figure 7.2(b). This design enables the transconductance amplifier to increase its dynamic range. Circuit simulations of this design are shown in Figure 7.3. Figure 7.3(a) shows the output current of the transconductance amplifier for different width to length ratios of the biasing transistor M_b, while Figure 7.3(b) shows the same response of the amplifier for different biasing voltages V_b.



(a) $\frac{W}{L} = 2,\ 3,\ 4,\ 5,\ 8$ (b) $V_b = 625,\ 250,\ 675,\ 700,\ 725mV$

Figure 7.3. PSPICE simulation of a transconductance amplifier

7.1.4 Hyperbolic Sine Function

A circuit that exhibits the sine hyperbolic function is shown in Figure 7.4. In [48], it

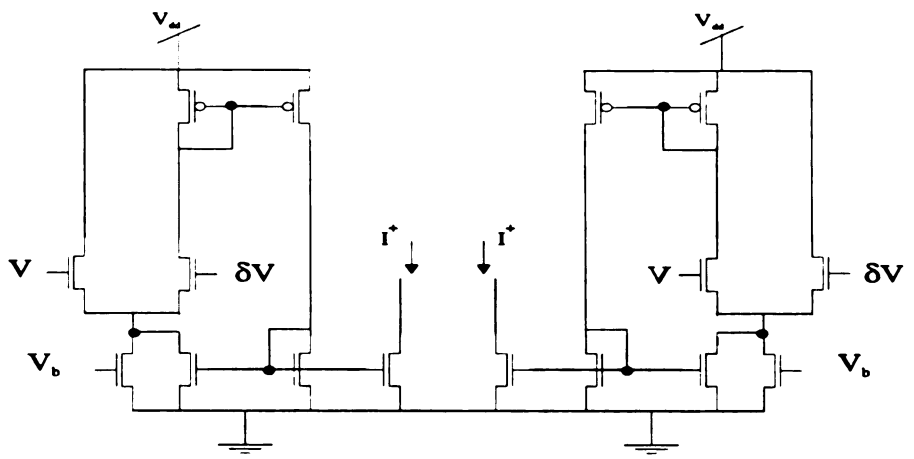


Figure 7.4. Circuit Realization of tan and sine hyperbolic functions

was derived that the output current of this circuit can be expressed as

$$I_{out} = 2wI_0 e^{\kappa(V_b - \delta V)} \sinh \kappa V$$

where V_b is the biasing voltage. A PSPICE simulation of the circuit is shown in Figure 7.5

7.1.5 Gilbert Multiplier

A circuit that produces the product of two voltages is needed in order to perform the implementation of algorithms that involves such an operation. In [70], the author presented the 1-dimensional multiplier as shown in Figure 7.6. It is now known in literature as the Gilbert multiplier. The output current of this circuit is

$$I_{out} = I_b \tanh \frac{\kappa}{2}(V_1 - V_2) \tanh \frac{\kappa}{2}(V_3 - V_4) \tag{7.5}$$

For small-signal analysis, the current described by equation (7.5) can be expressed as

$$I_{out} \propto I_b(V_1 - V_2)(V_3 - V_4) \tag{7.6}$$

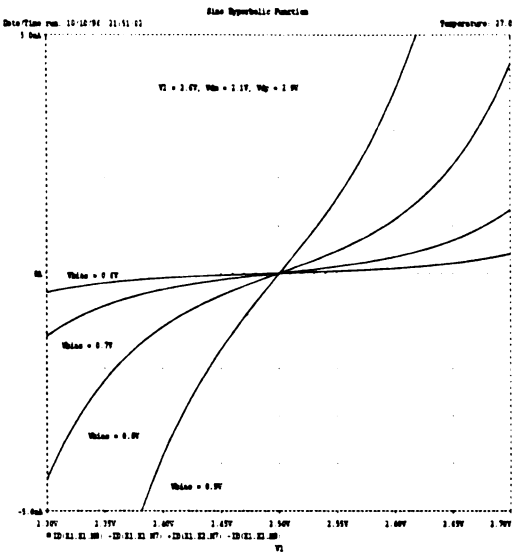


Figure 7.5. PSPICE simulation of the sine hyperbolic circuit for different biasing voltages V_b

PSPICE simulation of the circuit for various biasing voltages is presented in Figure 7.7. Observe the linear range for which the current is linear. It is desirable that one defines the range of operation to be within that linear range.

7.1.6 Vector Multipliers

Using the design of a single dimensional Gilbert multiplier, one may design the multi-dimensional multiplier. Such a design is shown in the Figure 7.8easily. In this case, the output current is

$$I_{out} = I_b \prod_i \tanh \frac{\kappa}{2}(X_i - V_{ref}) \ \tanh \frac{\kappa}{2}(Y_i - V_{ref}) \tag{7.7}$$

For small-signal analysis, the current expressed by equation (7.7) is approximated by

$$I_{out} \propto I_b \prod_i (X_i - V_{ref})(Y_i - V_{ref}) \tag{7.8}$$

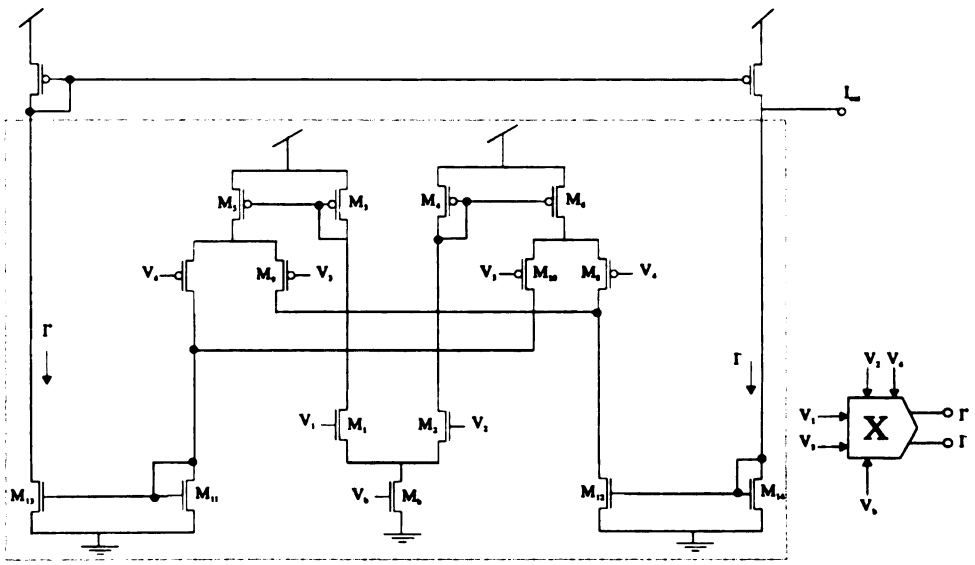


Figure 7.6. 1-dimensional Gilbert multiplier

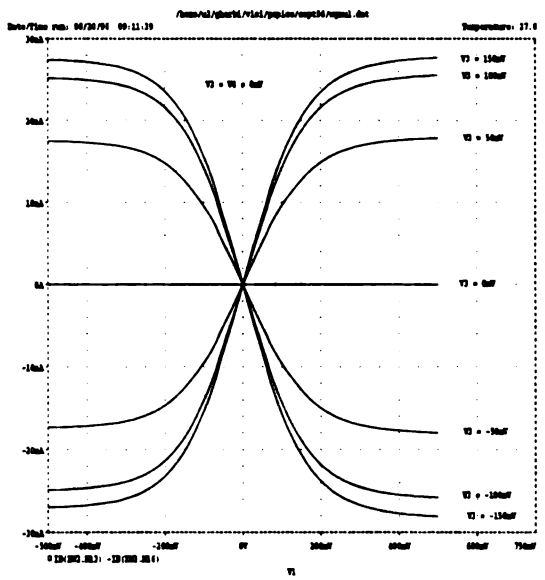


Figure 7.7. PSPICE simulation of of Modified Gilbert Multiplier using different biasing voltages

Thus, this circuit realizes the dot product of two n-dimensional vectors X and Y and gives such a value as a current. If one wishes to convert such a quantity to a voltage, then s/he can utilize the linear resistors to do so.

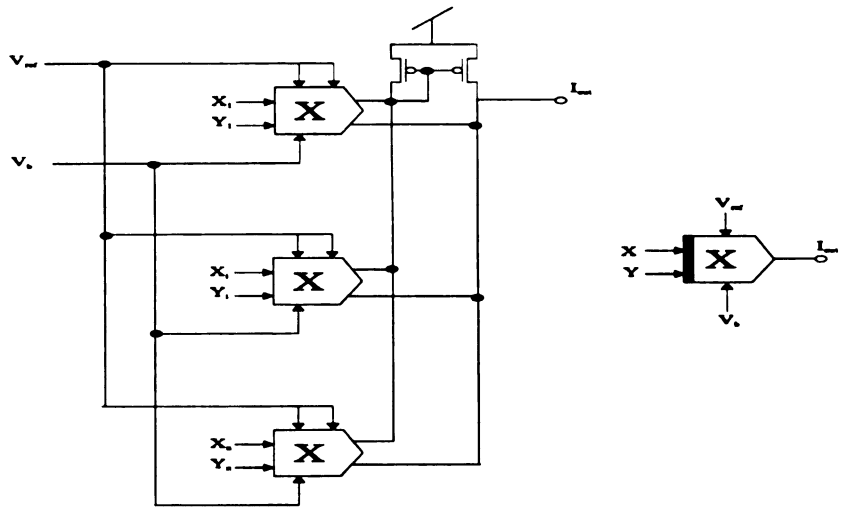


Figure 7.8. n-dimensional Gilbert multiplier

7.1.7 Current to Voltage Converter

All the basic analog components that have been presented so far produce a current output function. If one wishes to convert the current quantity to a voltage quantity, he or she can rely on the linear resistors. Linear resistor can be used to realize the conversion of a current to a voltage. A circuit realization of such a block is shown in Figure 7.9.

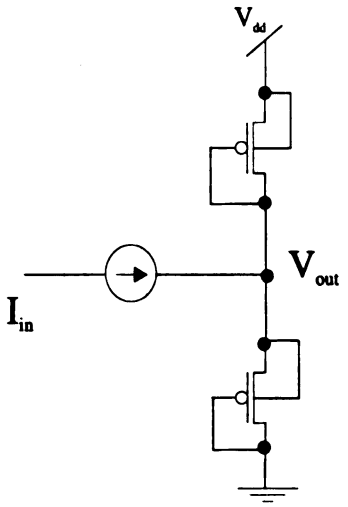


Figure 7.9. Current to Voltage Converter

7.1.8 Second Order Section

The second order section is a circuit that can be used to generate a response that can be represented by two poles in the complex plane. By adjusting its parameters, one can position the locations of these poles anywhere in the complex plane. A realization of such a circuit was presented by Mead in [71] and is shown in Figure 7.10. By performing a small-signal analysis, one can derive the transfer function of such a circuit.

The transconductance amplifier A_2 is a follower integrator and is described by the following equation

$$C_2\dot{v}_2 = g_2(v_2 - v_1) \tag{7.9}$$

At the node (1), the current generated by the transconductance amplifiers A_1 and A_3

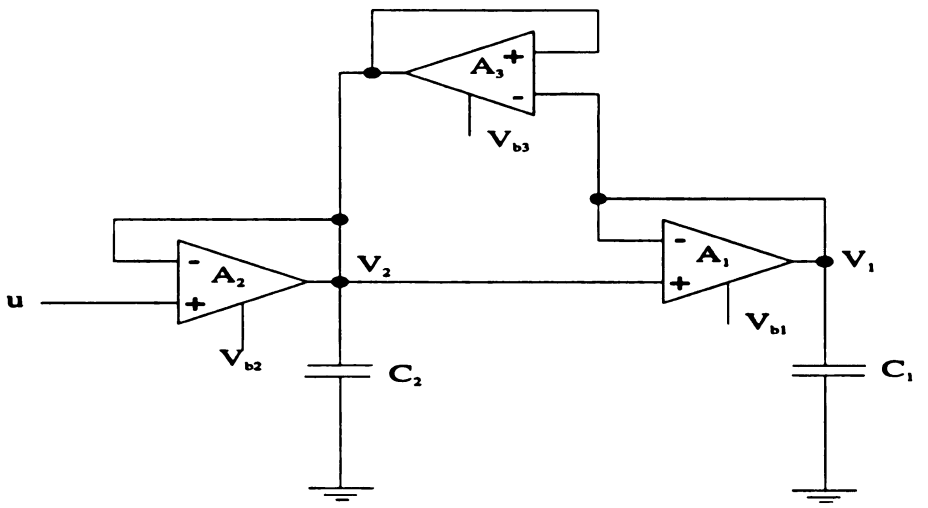


Figure 7.10. Second Order Section Circuit

will charge or discharge the capacitor C_1

$$C_1\dot{v}_1 = g_1(u - v_1) + g_3(v1 - v_2) \tag{7.10}$$

By taking the Laplace transform of equations (7.9) and (7.10)

$$sC_2V_2 = g_2(V_2 - V_1) \tag{7.11}$$

$$sC_1V_1 = g_1(U - V_1) + g_3(V1 - V_2) \tag{7.12}$$

and combining these two equations, the following relationship relating the output V_2

to the input u can be obtained

$$\begin{aligned}\frac{V_2}{U} &= \frac{g_1}{(sC_1 + g_1 - g_3)(1 + s\tau_2) + g_3} \\ &= \frac{g_1}{\tau_2 C_1 s^2 + (C_1 + \tau_2 g_1 - \tau_2 g_3)s + g_1} \\ &= \frac{1}{\tau_1 \tau_2 s^2 + (\tau_1 + \tau_2)(1 - \alpha)s + 1} \end{aligned} \tag{7.13}$$

where

$$\tau_i = \frac{C_i}{g_i} \tag{7.14}$$

and

$$\alpha = \frac{\tau_1}{\tau_1 + \tau_2}\frac{g_3}{g_1} \tag{7.15}$$

In [71], the authors concentrated their analysis on the case when $C_1 = C_2 = C$. In this particular case, the expression for α in (7.15) is, therefore, reduced to

$$\alpha = \frac{g_3}{g_1 + g_2} \tag{7.16}$$

and the transfer function described by (7.13) is expressed as

$$\frac{V_2}{U} = \frac{1}{\tau^2 s^2 + 2\tau(1 - \alpha)s + 1} \tag{7.17}$$

Figure 7.11 shows the frequency response of the second order section for different values of α.

Based on the small-signal analysis, the second order section is stable for any α satisfying $0 < \alpha < 1$. Also, in the case where $\alpha = 1$, the poles are purely imaginary and the circuit is unstable. However, a large-signal analysis for the circuit developed experimentally by Watts [69] shows that the second order section can be unstable

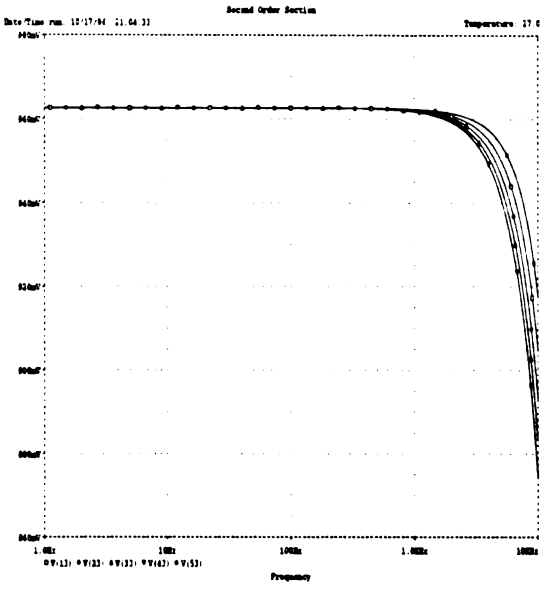


Figure 7.11. PSPICE Simulations of Second Order Section for V_{b_3} = 625, 650, 675, 700, 725mV

within the specified range of stability based on the small-signal analysis. It was proven that this nonlinear circuit exhibits some instabilities. Therefore, an improved design of the second order section was developed based on the improved design of the transconductance amplifier that is shown in Figure 7.2(b).

7.1.9 Second Order Filter Design

The design of the second order section developed by Mead and Watts birthed the inspiration to develop the circuit realization of the state equation for the canonical controllable state representation of a dynamical system. To start, a circuit realization for a 2-dimensional system using the second order section will be developed. Then, the development of the circuit realization for higher dimensions will be straight forward.

The second order section uses the transconductance amplifier A_1 as an integrator follower and thus it realizes a low pass filter. However, here a pure integrator for A_1 will be implemented. Therefore, A_1 was modified to produce a pure integrator circuit

instead.

7.1.10 Circuit Implementation of Second Order Filter

Consider the second order filter described by the transfer function

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_1}{s^2 + a_1 s + a_2} \tag{7.18}$$

The canonical controllable state representation of H(s) is

$$\dot{x}_1 = x_2 \tag{7.19}$$

$$\dot{x}_2 = -a_2 x_1 - a_1 x_2 + b_1 u \tag{7.20}$$

and the output equation is

$$y = x_1 \tag{7.21}$$

It will be shown that a circuit realization of the filter described by equation (7.18) is shown in Figure 7.12. A_1 is a pure integrator. Therefore

$$C_1 \dot{v}_1 = g_1(v_1 - V_{ref}) \tag{7.22}$$

At node (2), Kirchoff's current law suggests that

$$C_2 \dot{v}_2 = g_2(u - v_2) + g_3(V_{ref} - v_1) \tag{7.23}$$

Let $z_i = v_i - V_{ref}$ and $\hat{u} = u - V_{ref}$. Therefore, equations (7.22) and (7.22) become

$$C_1 \dot{z}_1 = g_1 z_2$$

$$C_2 \dot{z}_2 = g_2(\hat{u} - z_2) - g_3 z_1)$$

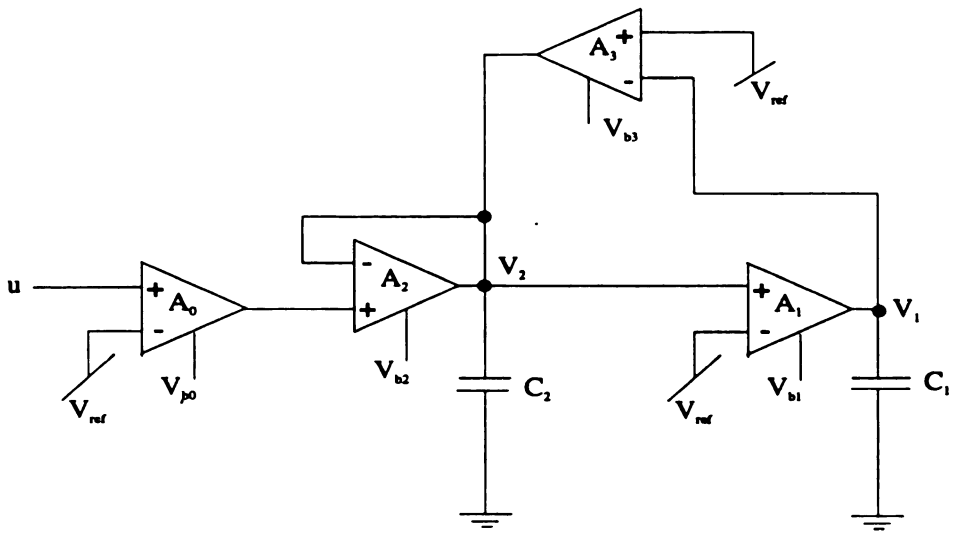


Figure 7.12. Filter Design. Circuit realization of controllable canonical form of a second order filter

which give us the generalized canonical controllable realization defined by

$$\dot{z}_1 = \frac{g_1}{C_1} z_2 \tag{7.24}$$

$$\dot{z}_2 = -\frac{g_3}{C_2} z_1 - \frac{g_2}{C_2} z_2 + \frac{g_2}{C_2} \hat{u} \tag{7.25}$$

The canonical controllable realization is obtained by using the transformation $T_{\frac{g_1}{C_1},1}$ defined by $x_1 = z_1$ and $x_2 = \frac{1}{\tau_1} z_2$. Therefore

$$\dot{x}_1 = x_2 \tag{7.26}$$

$$\dot{x}_2 = -\frac{1}{\tau_1 \tau_3} x_1 - \frac{1}{\tau_2} x_2 + \frac{1}{\tau_1 \tau_2} \hat{u} \tag{7.27}$$

where

$$\tau_1 = \frac{C_1}{g_1} \qquad \tau_2 = \frac{C_2}{g_2} \qquad \tau_3 = \frac{C_2}{g_3}$$

Equations (7.26) and (7.27) describe the canonical controllable of (7.18) where

$$a_2 = \frac{1}{\tau_1 \tau_3} \qquad a_1 = \frac{1}{\tau_2} \qquad b_1 = \frac{1}{\tau_1 \tau_3}$$

It is important to note that the developed second order filter described by equation (7.25) is always stable since the coefficients a_1 and a_2 are positive. In circuit implementation, it is often assumed that all the capacitors C_i are equal to C, which is in the order of a few pico Fayrads. The characteristics of the second order filter, i.e. its pole locations and gain, are controlled by the biasing voltages of the transconductance amplifiers shown in Figure 7.12. Figure 7.13 shows the transistor level design of the filter. PSPICE simulations of this circuit were performed. Figures 7.14(a) and 7.14(b) show the frequency response of the filter for various biasing voltages of transconductance $A2$ and $A3$, respectively. Observe how changing V_{b_2} and V_{b_3} impacts the gain as well as the pole locations of the filter. Having developed the circuit realization of a second order filter, that of an n-dimensional filter becomes straight forward as it is shown in the next section.

7.1.11 Circuit Realization for n-dimensional Filter

Figure 7.15 represents the circuit realization of an n-dimensional filter described by the equation

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_1}{s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \tag{7.28}$$

By looking at Figure 7.15, A_i is a pure integrator. Therefore,

$$C_i \dot{v}_i = g_i (v_{i+1} - V_{ref}) \tag{7.29}$$

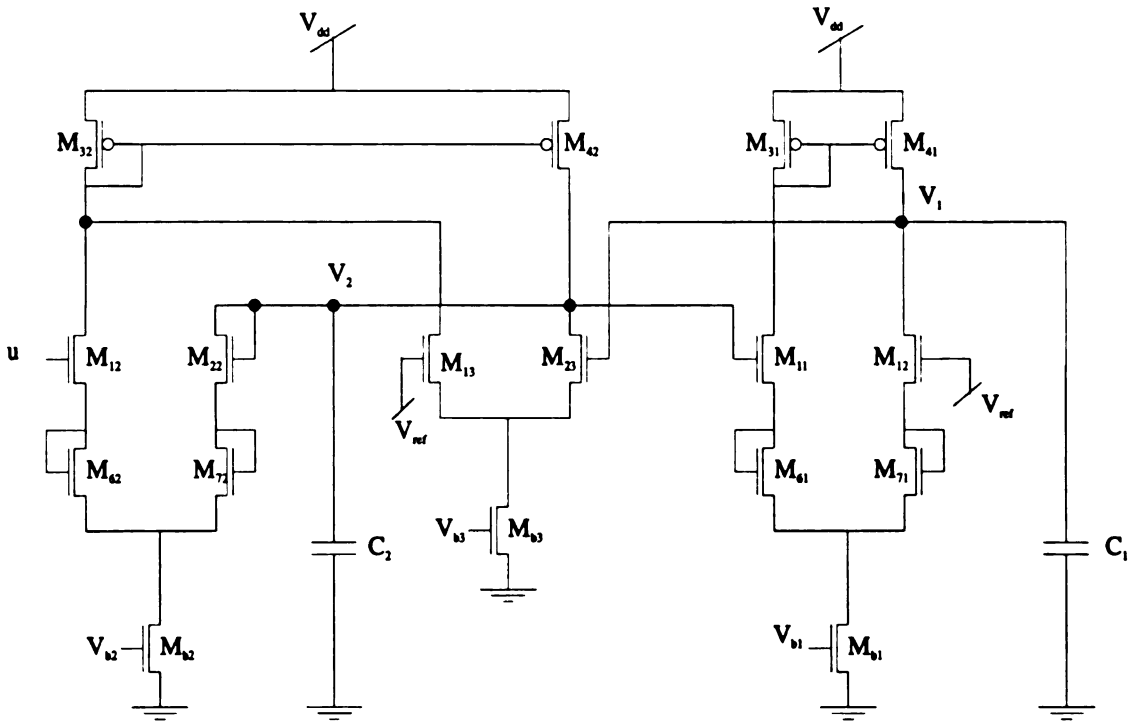
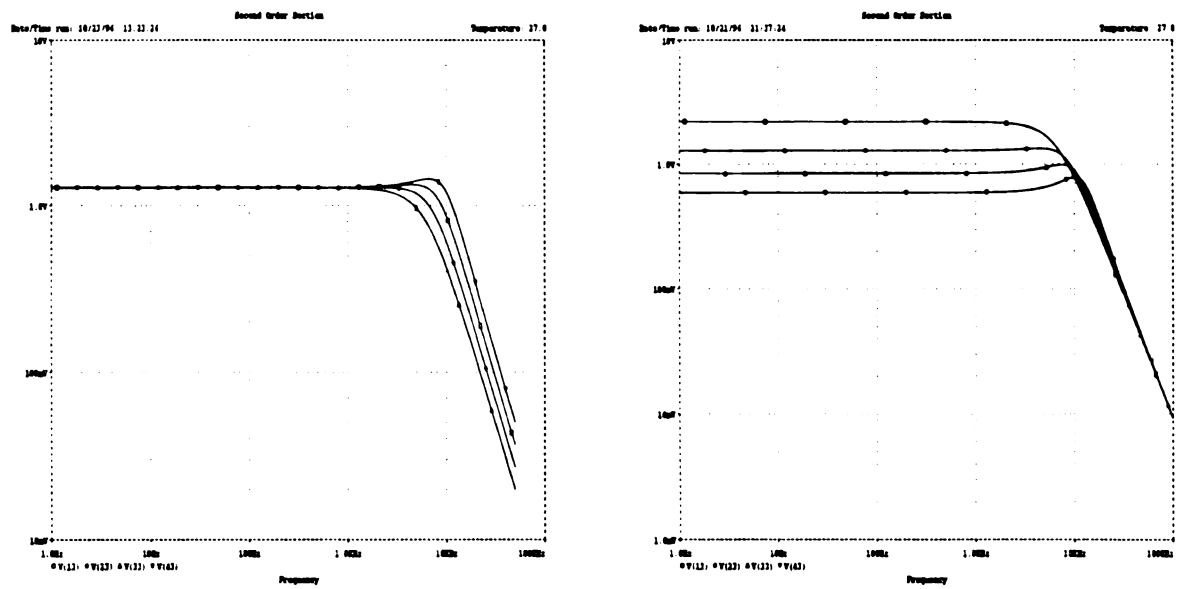


Figure 7.13. Filter Design. Circuit realization of controllable canonical form of a second order filter



(a) $V_{b_2} = 650,\ 675,\ 700,\ 725mV$ (b) $V_{b_3} = 650,\ 675,\ 700,\ 725mV$

Figure 7.14. PSPICE Simulations of second order filter

At node (v_n), Kirchoff's current law suggests that

$$C_n\dot{v}_n = \sum_{i=1}^{n-1} \bar{g}_i(V_{ref} - v_i) + g_n(u - v_n) \tag{7.30}$$

Let $z_i = v_i - V_{ref}$ and $\hat{u} = u - V_{ref}$. Therefore, equations (7.29) and (7.30) become

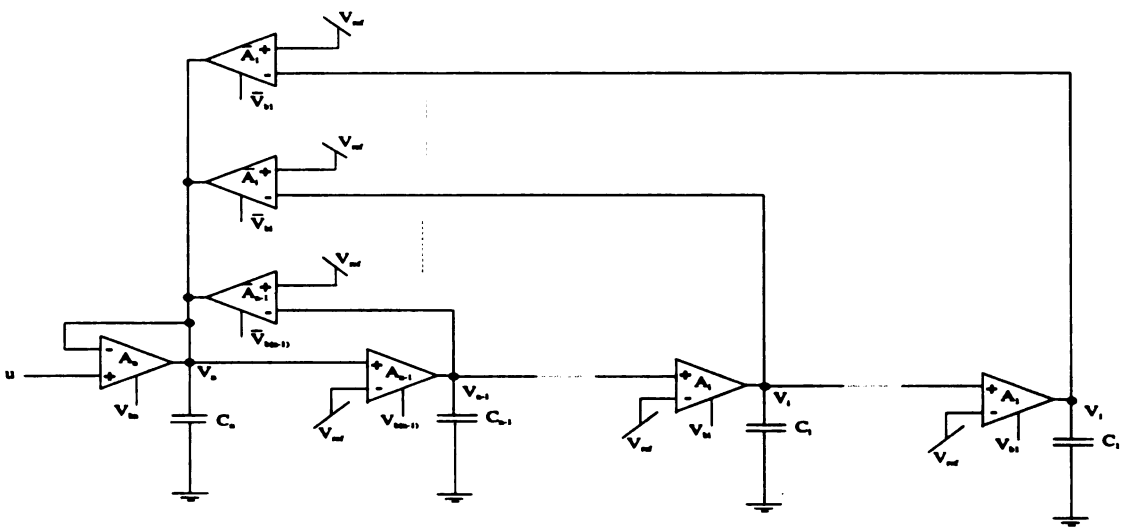


Figure 7.15. Filter Design, Circuit realization of a controllable canonical form of a n^{th} order filter

$$\begin{aligned} C_i\dot{z}_i &= g_i z_{i+1} \qquad i = 1, \cdots, n-1 \\ C_n\dot{z}_n &= -\sum_{i=1}^{n-1} \bar{g}_i z_i + g_n(\hat{u} - z_n) \end{aligned}$$

which gives us the generalized canonical controllable realization defined by

$$\dot{z}_i = \frac{g_i}{C_i} z_{i+1} \qquad i = 1, \cdots, n-1 \tag{7.31}$$

$$\dot{z}_n = -\frac{\bar{g}_1}{C_n} z_1 - \cdots - \frac{\bar{g}_{n-1}}{C_n} z_{n-1} - \cdots - \frac{g_n}{C_n} z_n + \frac{g_n}{C_n} \hat{u} \tag{7.32}$$

Based on the developed small-signal analysis, one can use the Routh Hurwitz criteria to design the necessary parameters to obtain a stable filter.

Define $\tau_i = \frac{g_i}{C_i}$, $\bar{\tau}_i = \frac{\bar{g}_i}{C_n}$, $\forall i = 1, \cdots, n-1$ and $\bar{\tau}_n = \tau_n$. Therefore, equations (7.31) and (7.32) become

$$\dot{z}_i = \tau_i z_{i+1} \qquad i = 1, \cdots, n-1 \tag{7.33}$$

$$\dot{z}_n = -\bar{\tau}_1 z_1 - \cdots - \bar{\tau}_{n-1} z_{n-1} - \cdots - \bar{\tau}_n z_n + \tau_n \hat{u} \tag{7.34}$$

In matrix form

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_i \\ \dot{z}_n \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & \tau_1 & 0 & \cdots & 0 \\ 0 & 0 & \tau_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \tau_{n-1} \\ -\bar{\tau}_1 & \cdots & \cdots & -\bar{\tau}_{n-1} & -\bar{\tau}_n \end{bmatrix}}^{A_z \in \mathcal{R}^{n \times n}} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_i \\ z_n \end{bmatrix} + \overbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \tau_n \end{bmatrix}}^{b_z \in \mathcal{R}^n} \hat{u}$$

Define the transformation T as

$$T = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & 0 \\ \vdots & 0 & \prod_{j=1}^{i-1} \tau_j & 0 & \vdots \\ 0 & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \prod_{j=1}^{n-1} \tau_j \end{bmatrix}$$

Therefore

$$A = TA_zT^{-1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ -a_n & \cdots & \cdots & \cdots & -a_1 \end{bmatrix} \qquad \mathbf{b} = T\mathbf{b}_z = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b_1 \end{bmatrix}$$

where

$$a_i = \bar{\tau}_{n-i+1} \frac{\prod_{j=1}^{n-1} \tau_j}{\prod_{j=n-i}^{n-1} \tau_j} = \bar{\tau}_{n-i+1} \prod_{j=n-i+1}^{n-1} \tau_j$$

and

$$b_1 = \tau_n \prod_{j=1}^{n-1} \tau_j = \prod_{j=1}^{n} \tau_j$$

7.2 Circuit Realization

Having introduced the library of basic building blocks for analog CMOS implementations, it is possible to develop a circuit realization of the network and the update laws.

7.2.1 Implementation of the output equation

The output equation

$$y_i = e_i - \mathbf{c}_i^T \mathbf{x}_i - \mathbf{d}_i^T \mathbf{y} \tag{7.35}$$

is simplified to

$$y_i = e_i - \hat{\mathbf{c}}_i^T \hat{\mathbf{x}}_i - \hat{\mathbf{d}}_i^T \hat{\mathbf{y}} \tag{7.36}$$

where $\hat{\mathbf{z}}_i$ is a vector having one dimension less than $\mathbf{z}$ and is obtained by eliminating the i^{th} component of $\mathbf{z}$. For the algorithm development, equation (7.35) was used rather than (7.36) in order to create a matching in the dimensions for clarity. It should be recalled that $\mathbf{x}_{ii} = 0$, $\mathbf{c}_{ii} = 0$ and $d_{ii} = 0$. Equation (7.36) is realized by using a transconductance amplifier to convert the input voltage e_i into a current. In addition, two multi-dimensional vector multipliers are used to compute the product $\hat{\mathbf{c}}_i^T\hat{\mathbf{x}}_i$ and $\hat{\mathbf{d}}_i^T\hat{\mathbf{y}}$. The currents coming out of the vector multipliers are subtracted from that of the transconductance amplifier to obtain the y_i in current form. Then, a linear resistor is used to convert the current into a voltage. Figure 7.16 shows the schematic of circuit realization.

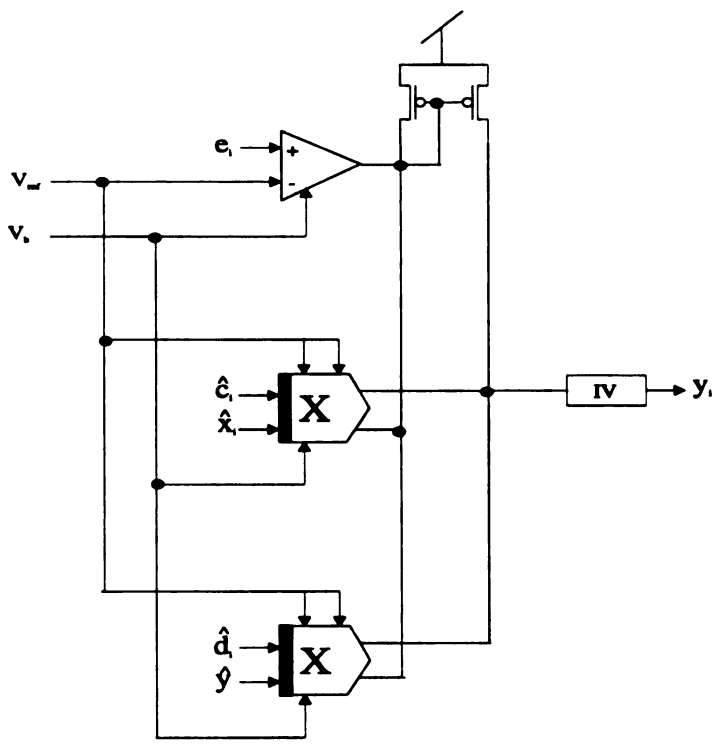


Figure 7.16. Circuit Realization of the output equation

7.2.2 Implementation of the state equation

The state equation is described by the transfer function

$$\begin{bmatrix} \dot{x}_{ij}^{(1)} \\ \dot{x}_{ij}^{(2)} \\ \vdots \\ \dot{x}_{ij}^{(\mu_{ij}-1)} \\ \dot{x}_{ij}^{(\mu_{ij})} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ -\alpha_{ij}^{(\mu_{ij})} & \cdots & \cdots & \cdots & -\alpha_{ij}^{(1)} \end{bmatrix} \begin{bmatrix} x_{ij}^{(1)} \\ x_{ij}^{(2)} \\ \vdots \\ x_{ij}^{(\mu_{ij}-1)} \\ x_{ij}^{(\mu_{ij})} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \beta_{ij} \end{bmatrix} u_j \quad (7.37)$$

The circuit implementation of this transfer function was completed in the previous section. Figure 7.15 shows the schematic for a filter of order n. Thus, to implement the equation (7.37) one needs to designs a μ_{ij}th order filter. In this case

$$\alpha_{ij}^{(k)} = \bar{\tau}_{\mu_{ij}-k+1} \prod_{l=\mu_{ij}-k+1}^{\mu_{ij}-1} \tau_l \quad (7.38)$$

and

$$\beta_{ij} = \prod_{l=1}^{\mu_{ij}} \tau_l \quad (7.39)$$

7.2.3 Implementation of the C update equation

The update law of the parameters of the C matrix is described by

$$\dot{c}_{ij}^{(k)} = \eta_{ij}^{(k)} y_i x_{ij}^{(k)} \qquad \forall j \neq i \text{ and } k = 1, \cdots, \mu_{ij} \quad (7.40)$$

Equation (7.40) is implemented using a 1-dimensional gilbert multiplier. The output current of the multiplier is dumped to a capacitor $C_{ij}^{(k)}$ as shown in Figure 7.17. In this case

$$C_{ij}^{(k)} \dot{c}_{ij}^{(k)} = I_{b_{ij}^{(k)}} \tanh \kappa \frac{x_{ij}^{(k)} - V_{ref}}{2} \tanh \kappa \frac{y_i - V_{ref}}{2} \quad (7.41)$$

Therefore

$$\dot{c}_{ij}^{(k)} = \eta_{ij}^{(k)}\left(x_{ij}^{(k)} - V_{ref}\right)\left(y_i - V_{ref}\right) \tag{7.42}$$

where

$$\eta_{ij}^{(k)} = \frac{I_{b_{ij}^{(k)}}}{C_{ij}^{(k)}}\left(\frac{\kappa q}{2kT}\right)^2 \tag{7.43}$$

Thus, the leaning rate of such parameter is controlled the bias voltage of the multiplier, $VC_{b_{ij}^{(k)}}$

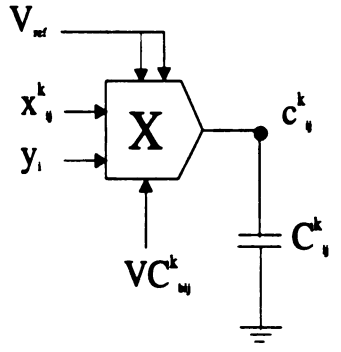


Figure 7.17. Circuit Realization the update equation of the C matrix parameters

7.2.4 Implementation of the D update equation

The update law of the parameters of the C matrix is described by

$$\dot{d}_{ij} = \gamma_{ij} \sinh y_i \tanh y_j \qquad \forall j \neq i \tag{7.44}$$

Equation (7.44) is implemented using the sine hyperbolic circuit and the transconductance amplifier that were described in the previous section. Figure 7.18 shows the

circuit implementation. The algebraic sum of all the current at the (+) node of the capacitor C_{ij} is

$$I_{out} = I_{out}^{sinh} \tanh \frac{\kappa}{2}(y_j - V_{ref}) \tag{7.45}$$

where

$$I_{out}^{sinh} = 2wI_0 e^{\kappa(V_b+\delta V)} \sinh \kappa(y_i - V_{ref}) \tag{7.46}$$

The current described by equation (7.45) is dumped to the capacitor C_{ij}. Therefore, when equations (7.45) and (7.46) are combined together, one obtains

$$\dot{d}_{ij} = \gamma_{ij} \sinh \kappa(y_i - V_{ref}) \tanh \frac{\kappa}{2}(y_j - V_{ref}) \tag{7.47}$$

where

$$\gamma_{ij} = \frac{2wI_0 e^{\kappa(VD_{b_{ij}}+\delta V)}}{C_{ij}} \tag{7.48}$$

It is important to observe the parameters that the the learning rate γ_{ij} depends on. They are the biasing voltage $VD_{b_{ij}}$; the ratio of width of the biasing transistor to the width of the arm transistors w; and the offset voltage δV. In the circuit implementation, all these parameters will be held constant except the bias gate voltage $VD_{b_{ij}}$, which will control the learning rate of the parameter d_{ij}.

7.3 Circuit Simulation

In this section, the circuit simulation of a 2×2 network is performed using the circuit simulator PSPICE. The block diagram of the overall circuit including network and learning is shown in Figure 7.19.

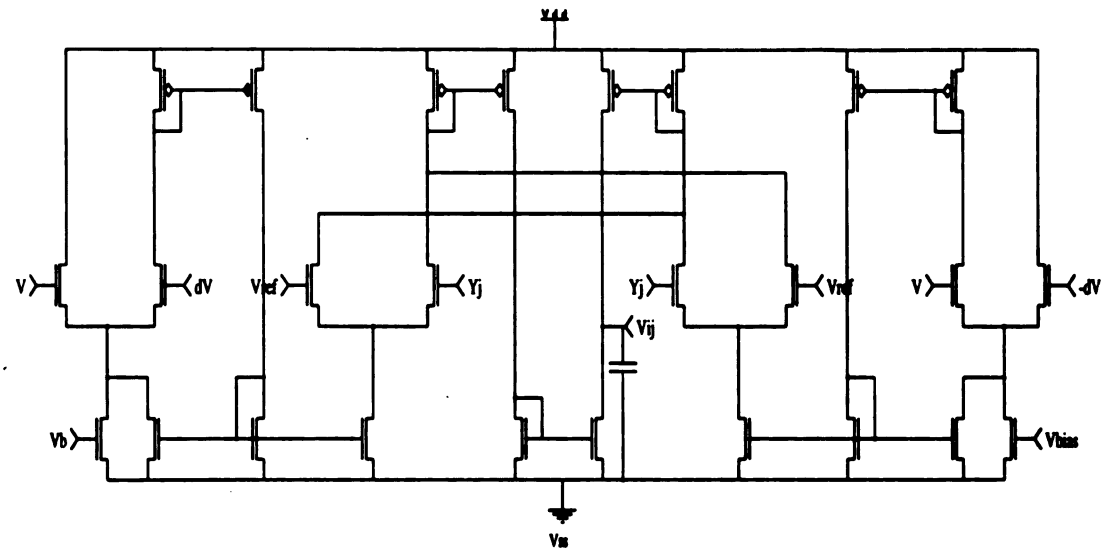


Figure 7.18. Circuit Realization of tan and sine hyperbolic functions

The unknown signals are two sine waveforms with respective frequencies $1kHz$ and $2kHz$ as shown in Figure 7.20. These two signals are filtered and mixed in order to obtain the signals $e_1(t)$ and $e_2(t)$ which are the output signals of the environment as presented in Figure 7.21. The transfer functions $\bar{H}_{12}(s)$ and $\bar{H}_{21}(s)$, shown in Figure 7.21, are two second order filters whose state and output equations are respectively defined by equations (7.37) and (7.35), with the exception that $d_{ii} = 1$ in this case.

$$\dot{\mathbf{x}}_{ij} = \bar{A}_{ij}\mathbf{x}_{ij} + \bar{\beta}_{ij}u_j \tag{7.49}$$

$$e_i = \bar{\mathbf{c}}_{ij}\mathbf{x}_{ij} + u_i + \bar{d}_{ij}u_j \tag{7.50}$$

The circuit implementation of equation (7.49) is shown in Figure 7.15. However, the circuit implementation of equation (7.50) is obtained in a similar fashion as that of the output equation of the network model described by equation (7.35) whose circuit implementation is shown in Figure 7.16.

The parameters of the matrix $\bar{A}_{ij}$ are defined by the biasing voltages of the transconductance amplifiers and the capacitors shown in Figure 7.15. The parameters of the column vector $\bar{\mathbf{c}}_{ij}$ and the parameter $\bar{d}_{ij}$ are defined by the voltages at the cor-

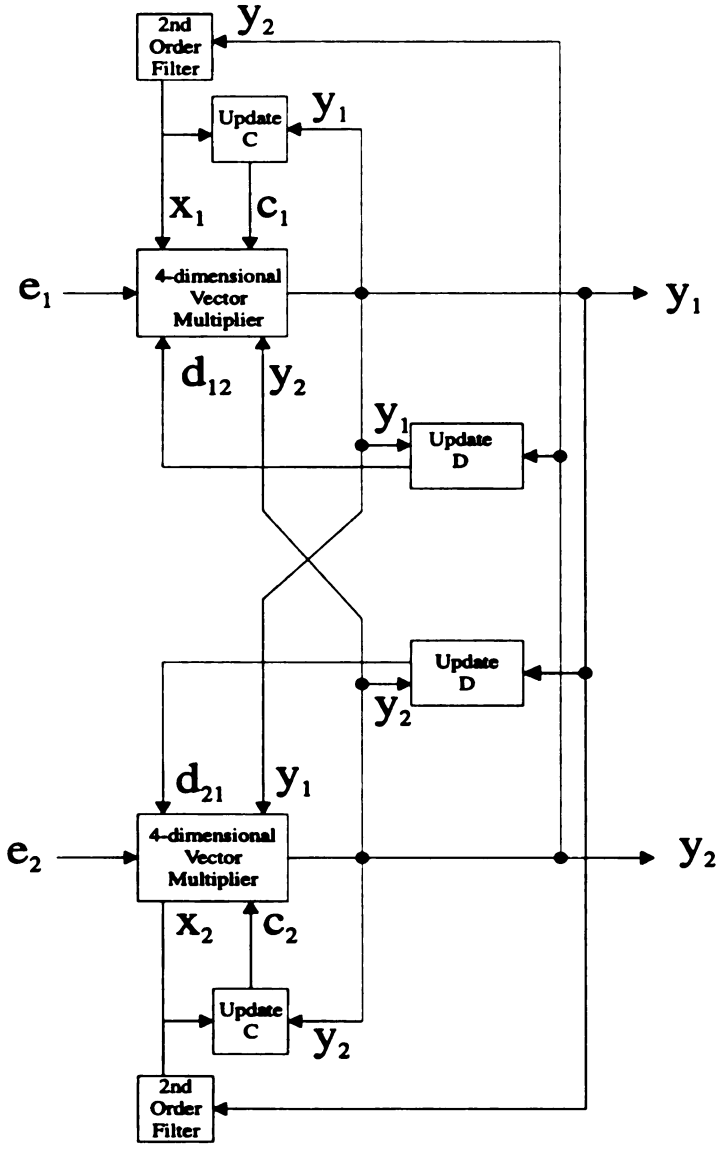


Figure 7.19. Block diagram of overall circuit including network and learning

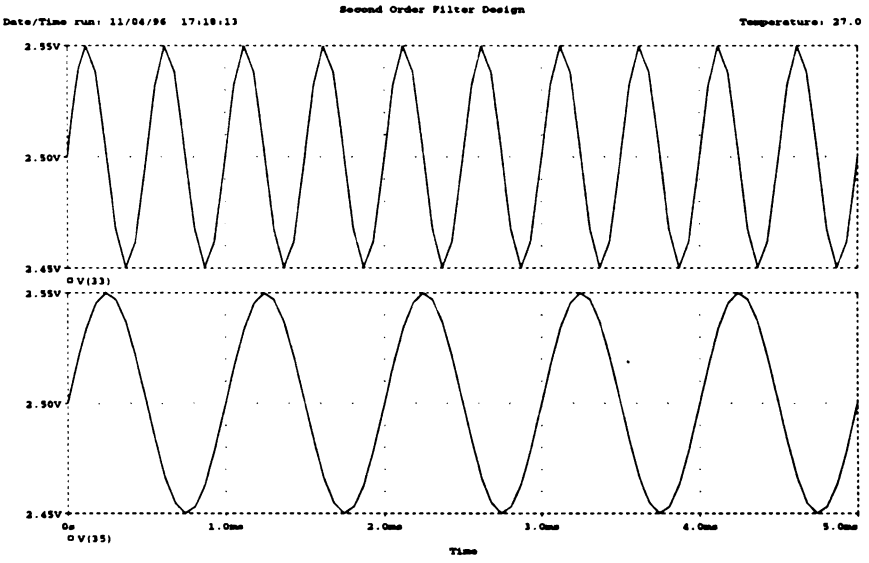


Figure 7.20. Unknown sources

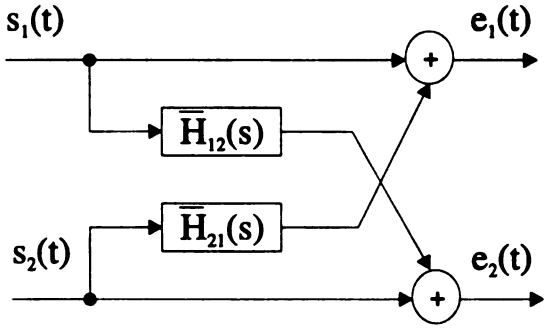


Figure 7.21. Environment Circuit Model

Table 7.1. Parameters of the Environment Circuit Realization

$\bar{A}_{12}$		$\bar{A}_{21}$	
V_b^1	0.69V	V_b^1	0.69V
V_b^2	0.69V	V_b^2	0.69V
V_b^3	0.71V	V_b^3	0.71V
C_1	5pF	C_1	4pF
C_2	5pF	C_2	4pF
$\bar{\mathbf{c}}_{12}$		$\bar{\mathbf{c}}_{21}$	
$V_{\bar{c}_{11}}$	2.60V	$V_{\bar{c}_{21}}$	2.65V
$V_{\bar{c}_{12}}$	2.40V	$V_{\bar{c}_{22}}$	2.45V
$\mathbf{d}_1$		$\mathbf{d}_2$	
$V_{\bar{d}_{11}}$	2.70V	$V_{\bar{d}_{21}}$	2.56V
$V_{\bar{d}_{12}}$	2.54V	$V_{\bar{d}_{22}}$	2.70V

responding gates of the transistor at the positive arm of the gilbert multiplier. Table 7.3 shows all these parameters. A PSPICE simulation of the frequency response of the filters $\bar{H}_{12}(s)$ and $\bar{H}_{21}(s)$ was conducted. The results of this simulation are shown in Figure 7.22. Observe that the cutoff frequencies of the filters $\bar{H}_{12}$ and $\bar{H}_{21}$ are within the $1kHz$ and the $5kHz$ range, which is the operating frequency range. The transient response of these filters is shown in Figure 7.23 where the input signals are the two sine waveforms shown in Figure 7.20.

The output of the environment is fed to the transconductance amplifier as shown in Figure 7.16. Unlike the environment circuit model whose parameters $\bar{\mathbf{c}}_{ij}$ and $\bar{\mathbf{d}}_i$ are constant and are defined in Table 7.3, the parameters $\hat{\mathbf{c}}_{ij}$ and $\hat{\mathbf{d}}_i$ of the network model are controlled by the time-varying charges, accross the capacitors shown in figures 7.17 and 7.18, defined by equations (7.42) and (7.47). The initial voltages across these capacitors are shown in Table 7.3. Figure 7.24 shows the results of

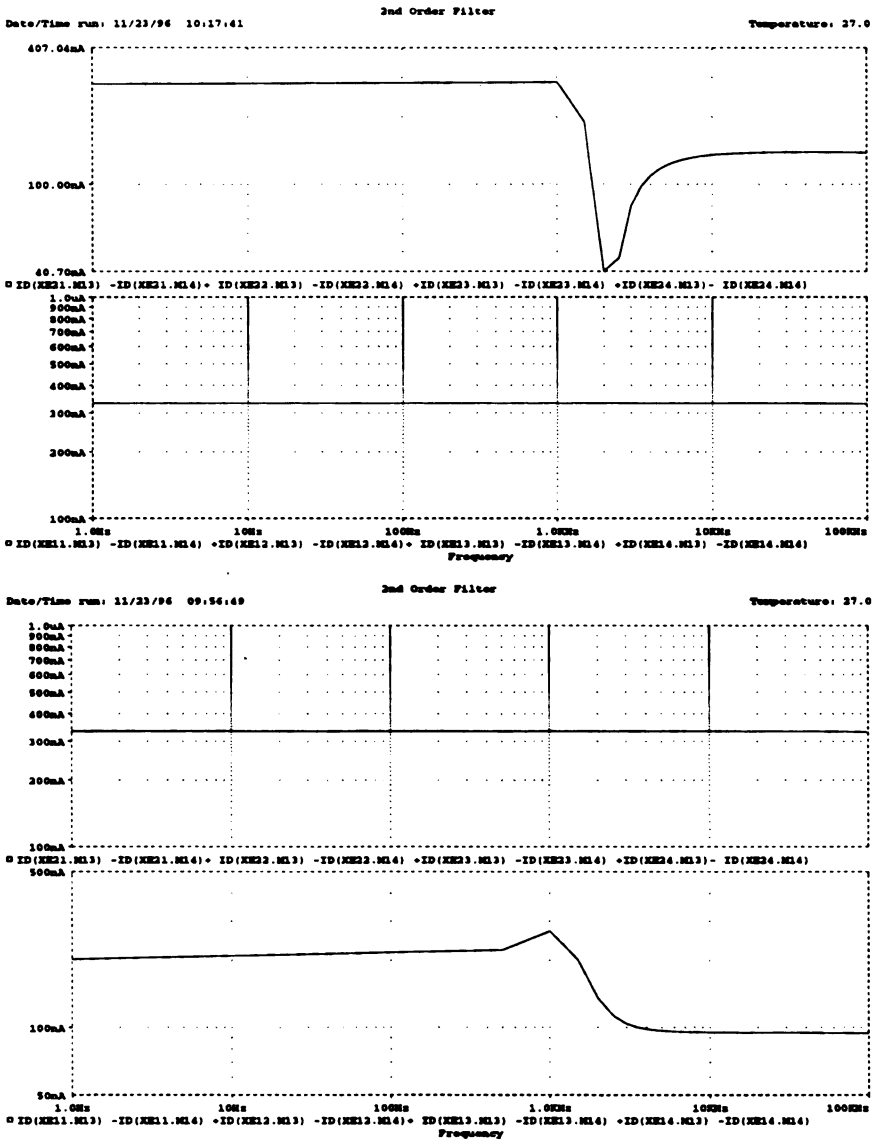


Figure 7.22. Frequency Response of the Environment Circuit Realization

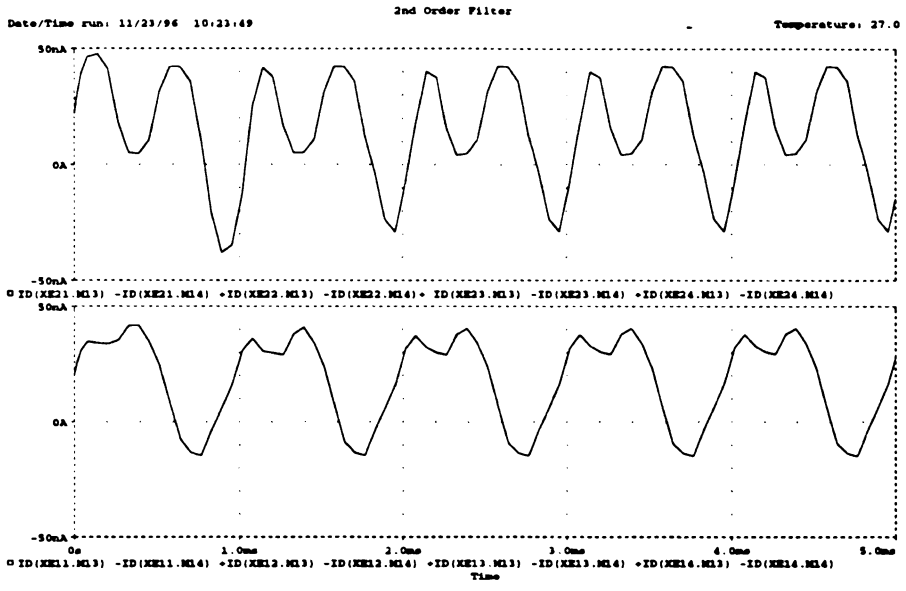


Figure 7.23. Transient Response of The Environment Circuit Realization

the network performance for the first $50msec$. Observe that all the parameters have converged to the values as shown in Table 7.3. By fixing the parameters of the network to these final values, the network reproduced the original signals as shown in Figure 7.25. The transistor PSPICE model and the PSPICE source code used to simulate the developed circuit implementation is given in Appendix D.

In this chapter, a subthreshold circuit implementation of the dynamic feedback network and the learning algorithm was developed. The performance analysis of circuit realization was conducted using two prototype signals as the unknown sources. It was observed that the network parameters converge to a separating solution and thus recover the original unknown sources.

Table 7.2. Parameters of the Environment Circuit Realization

Parameter	Initial	Final
d_{12}	1.0V	4.80V
d_{21}	4.0V	4.80V
c_{11}	1.0V	2.31V
c_{12}	4.0V	2.20V
c_{21}	1.0V	2.73V
c_{22}	4.0V	2.36V

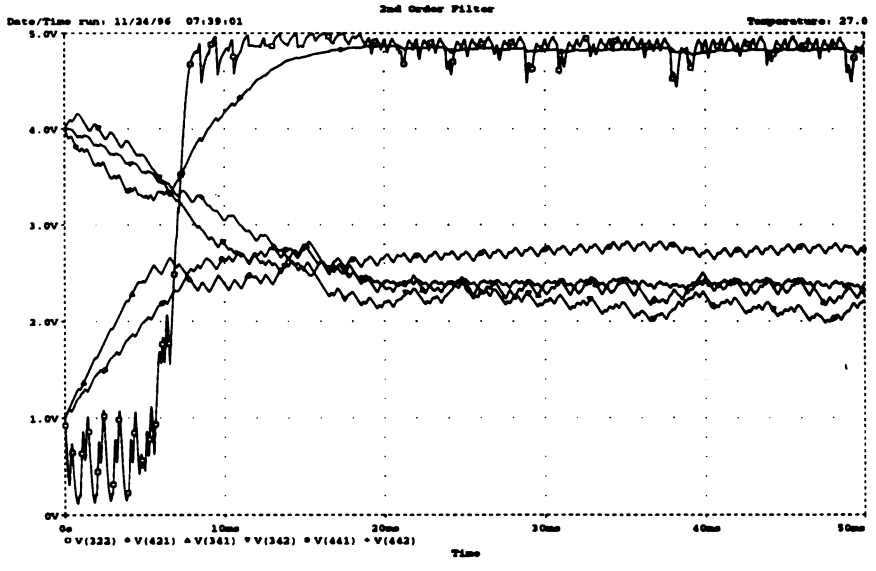


Figure 7.24. Parameters' Convergence of the Network Circuit Realization

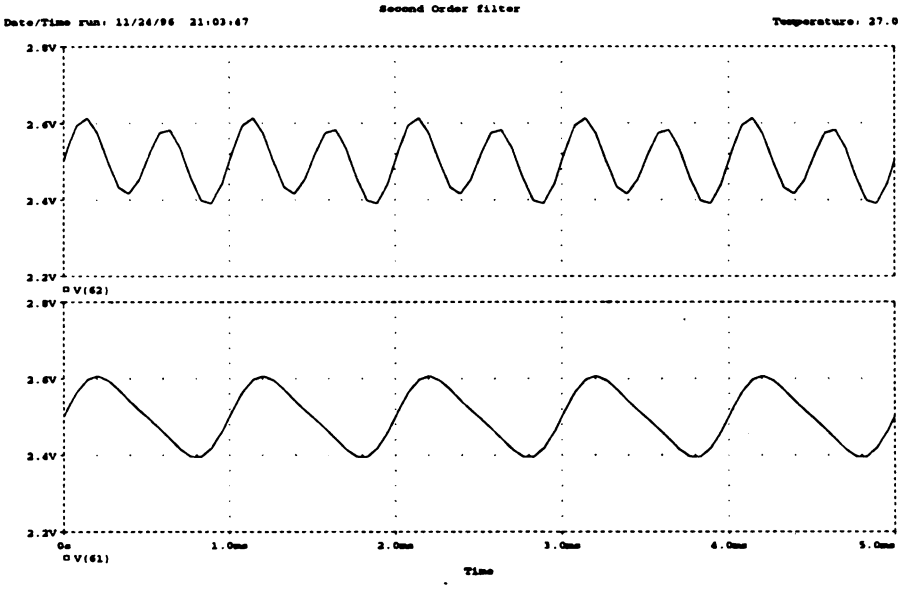


Figure 7.25. Output of the Network Circuit Realization at Convergence

CHAPTER 8

Conclusion

In this work, a novel algorithm for the blind separation of signals in static environment based on the decorrelation condition of the output signals was developed. It was then improved to test for independence. An energy function for the problem based on the minimization of the mutual information functional was also developed. A fourth order Edgeworth expansion was used in order to find an approximate expression of the probability density function. The new energy function considered a larger set of signals as compared to the existing energy function, since the assumption of unit variance of the output was not considered. Also, the energy function represented a better estimate of the mutual information functional.

A mathematical framework for the development of update laws for the network parameters based on the theory of adaptive optimal control theory was also developed. A realization of the environment that represents the least number of parameters was also introduced. A forward and an feedback structure were considered to model the environment and hence the network. Computer simulations showed that the update of the parameters of the matrix D converge for both structures. However, successful learning of the C matrix was realized only for the feedback structure. The coupling of both learning rules was eliminated by considering a sequential update method. Basic building blocks for the circuit implementation of an algorithm in the dynamic feedback

network model were developed. Overall circuit implementation of the algorithm was also developed and tested for prototype waveforms.

APPENDICES

APPENDIX A

Definitions

A.1 Statistical Definitions

Definition 4 (Entropy) *Entropy is a measure of uncertainty of the occurrence of an event in an ensemble of experiments described by the random variable (rv)* $\mathbf{x}$ *[72]. It is defined as:*

$$H(\mathbf{x}) = -E\{\ln f_{\mathbf{x}}(\mathbf{x})\}$$

where $f_{\mathbf{x}}(\mathbf{x})$ *is the probability density function (pdf) of the rv* $\mathbf{x}$.

Definition 5 (Joint Entropy) *The joint entropy of two random variables* $\mathbf{x}$ *and* $\mathbf{y}$ *is [72]:*

$$H(\mathbf{x},\mathbf{y}) = -E\{\ln f_{\mathbf{xy}}(\mathbf{x},\mathbf{y})\}$$

where $f_{\mathbf{xy}}(\mathbf{x},\mathbf{y})$ *is the joint pdf of the rv's* $\mathbf{x}$ *and* $\mathbf{y}$

Definition 6 (Mutual Information) *The mutual information among the components of a random vector* $\mathbf{x}$ *is defined as [72]:*

$$I(\mathbf{x}) = E\{\ln \frac{f_{\mathbf{x}}(\mathbf{x})}{\prod_i f_{\mathbf{x}_i}(\mathbf{x}_i)}\} \tag{A.1}$$

In the above definitions, $E\{.\}$ denotes the expected value and is defined as:

Definition 7 (Expected Value) *The expected value of a function of an rv* $\mathbf{x}$ *is defined as [72]:*

$$E\{g(\mathbf{x})\} = \int_{-\infty}^{+\infty} g(\mathbf{x}) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}$$

Definition 8 (Conditional Probability Denesity)

$$f(\mathbf{x},\mathbf{y}) = f(\mathbf{x}|\mathbf{y})f(\mathbf{y}) = f(\mathbf{y}|\mathbf{x})f(\mathbf{x})$$

Proposition 1 (Properties of Entropy)

P1 $H(\mathbf{x}) \geq 0$

P2 $H(\mathbf{x},\mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}|\mathbf{x}) = H(\mathbf{y}) + H(\mathbf{x}|\mathbf{y})$

P3 $H(\mathbf{x},\mathbf{y}) \leq H(\mathbf{x}) + H(\mathbf{y})$

P4 $H(\mathbf{x}|\mathbf{y}) \leq H(\mathbf{x})$

P5 $H(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n) = H(\mathbf{x}_1) + H(\mathbf{x}_2|\mathbf{x}_1)$
$+H(\mathbf{x}_3|\mathbf{x}_2,\mathbf{x}_1) + \cdots + H(\mathbf{x}_n|\mathbf{x}_{n-1},\cdots,\mathbf{x}_1)$

P6 $H(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n) \leq H(\mathbf{x}_1) + H(\mathbf{x}_2) + \cdots + H(\mathbf{x}_n)$

P7 *Equality in* **P6** *will hold if* $\mathbf{x}_i$*'s are statistically independent.*

Proof of *P7*

In this case, by the definition of statistical independence, one has $f_{\mathbf{x}}(\mathbf{x}) = \prod_i f_{\mathbf{x}_i}(\mathbf{x}_i)$

$$\begin{aligned} H(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n) &= -E\{\ln f_{\mathbf{x}}(\mathbf{x})\} \\ &= -E\{\ln \prod_i f_{\mathbf{x}_i}(\mathbf{x}_i)\} \\ &= -\sum_i E\{\ln f_{\mathbf{x}_i}(\mathbf{x}_i)\} \end{aligned}$$

$$= \sum_i H(\mathbf{x}_i)$$

A.2 Matrix Notations

Definition 9 *A permutation matrix is a square matrix whose rows and columns contain only one nonzero entry that is equal to* 1.

Definition 10 *A generalized permutation matrix is a square matrix whose rows and columns contain only one nonzero entry.*

APPENDIX B

Proofs

B.1 Proof of Theorem 3.5

The proof of this theorem is conducted by induction. However, the following theorem is needed.

Theorem 3 *Let A be a positive definite $n \times n$ matrix, and B be the $(n+1) \times (n+1)$ matrix*

$$B = \begin{pmatrix} A & b \\ b^T & \alpha \end{pmatrix}$$

Then

(i) $|B| \leq \alpha |A|$, with equality if and only if $b = 0$;

(ii) B is positive definite if and only if $|B| > 0$

Proof of theorem 3:

Define the $(n+1) \times (n+1)$ matrix

$$P = \begin{pmatrix} I_n & -A^{-1}b \\ 0^T & 1 \end{pmatrix}$$

Then

$$P^T B P = \begin{pmatrix} A & 0 \\ 0^T & \alpha - b^T A^{-1} b \end{pmatrix}$$

so that

$$|B| = |P^T B P| = |A| \bullet (\alpha - b^T A^{-1} b)$$

A is positive definite. Then, A^{-1} is positive definite. Therefore, $b^T A^{-1} b > 0, \forall b$. Consequently, we prove (i). To prove (ii), we observe that $|B| > 0$ if and only if $\alpha - b^T A^{-1} b > 0$, which is the case if and only if $P^T B P$ is positive definite. Since P is positive definite, so is B. □

Proof of theorem 1:

As stated earlier, the proof will be conducted by induction. If M is a positive scalar, then equation (3.5) is always true. Now, we will assume that the equation (3.5) true for matrix M of rank n. Theorem 3 shows that equation (3.5) is also true for $(n+1)$. □

B.2 CMOS Circuit Function Derivation

Derivation of The Transconductance Amplifier Function

Figure 7.2(a) represents the circuit diagram of a transconductance amplifier. The circuit will exhibit a sigmoidal function of the differential voltage V_1 and V_2 presented at the gates of the transistors M_1 and M_2, respectively. The circuit is being driven by a biasing current I_b which is controlled by the voltage V_b applied to the transistor M_b. Using KCL at node (1), we have:

$$I_1 + I_2 = Ib$$

where

$$
\begin{aligned}
I_1 &= I_o\, e^{\kappa V_1 - V_x} \\
I_2 &= I_o\, e^{\kappa V_2 - V_x} \\
I_b &= w I_o\, e^{\kappa V_b} \left(1 - e^{-V_x}\right)
\end{aligned}
$$

Solving for e^{-V_x} to obtain:

$$
e^{-V} = \frac{w\, e^{\kappa V_b}}{e^{\kappa V_1} + e^{\kappa V_2} + w e^{\kappa V_b}}
$$

Substituting the above expression in the expressions of currents I_1 and I_2, one obtains:

$$
\begin{aligned}
I_1 &= w I_o\, e^{\kappa V_b} \frac{e^{\kappa V_1}}{e^{\kappa V_1} + e^{\kappa V_2} + w e^{\kappa V b}} \\
I_2 &= w I_o\, e^{\kappa V_b} \frac{e^{\kappa V_2}}{e^{\kappa V_1} + e^{\kappa V_2} + w e^{\kappa V b}}
\end{aligned}
$$

To acquire the difference of the two currents, a current mirror circuit is employed. The current mirror circuit produces a replica of the current I_1 flowing through M_4. Thus,

$$
\begin{aligned}
I_{out} &= I_1 - I_2 \\
&= w I_o\, e^{\kappa V_b} \frac{e^{\kappa V_1} - e^{\kappa V_2}}{e^{\kappa V_1} + e^{\kappa V_2} + w e^{\kappa V b}}
\end{aligned}
$$

Multiply numerator and denominator by $e^{\kappa(V_1+V_2)/2}$

$$
I_{out} = w I_o\, e^{\kappa V_b} \frac{e^{\kappa(V_1-V_2)/2} - e^{-\kappa(V1-V_2)/2}}{e^{\kappa(V_1-V_2)/2} + e^{-\kappa(V_1-V_2)/2} + w e^{\kappa(Vb-(V_1+V_2)/2)}}
$$

However,

$$
V_b << \frac{V_1 + V_2}{2}
$$

Therefore,

$$I_{out} = wI_o\, e^{\kappa V_b} \frac{e^{\kappa(V_1-V_2)/2} - e^{-\kappa(V1-V_2)/2}}{e^{\kappa(V_1-V_2)/2} + e^{-\kappa(V_1-V_2)/2}}$$

which is

$$I_{out} = wI_o\, e^{\kappa V_b} \tanh \kappa(V_1 - V_2)/2$$

Derivation of Hyperbolic Sine Function

A circuit that exhibits the sine hyperbolic function is shown in figure 7.4. The output current is expressed as

$$I_{out} = I^+ - I^-$$

where

$$I^+ = h(V, \delta V) \qquad \text{and} \qquad I^+ = h(-\delta V, V)$$

Now, it is desired to determine the function $h(V_1, V_2)$.

$$I_1 = I_0 e^{\kappa V_1 - V_x}$$

$$I_2 = I_0 e^{\kappa V_2 - V_x}$$

$$I_b = wI_0 e^{\kappa V_1}(1 - e^{V_x})$$

Using KCL at node V_x:

$$I_1 + I_2 = I_b + A\, I_1$$

Substituting for all expressions and solve for e^{-V_x} to obtain:

$$e^{-V_x} = \frac{w\ e^{\kappa V_b}}{(1-A)e^{\kappa V_1} + e^{\kappa V_2} + w\ e^{\kappa V_b}}$$

Therefore with unity feedback ($A = 1$):

$$I_1 = wI_0 e^{\kappa V_b}\ \frac{e^{\kappa V_1}}{e^{\kappa V_2} + w\ e^{\kappa V_b}}$$

Multiplying numerator and denominator by $e^{-\kappa V_2}$ to obtain:

$$I_1 = wI_0 e^{\kappa V_b}\ \frac{e^{\kappa(V_1 - V_2)}}{1 + w\ e^{\kappa(V_b - V_2)}}$$

However, $V_b << V_2$. Thus, $e^{\kappa(V_b - V_2)} << 1$. Consequently,

$$I = h(V_1, V_2) = I_1 = wI_0 e^{\kappa V_b}\ e^{\kappa(V_1 - V_2)}$$

Coming back to find the output current of figure 7.4:

$$\begin{aligned} I_{out} &= h(V, \delta V) - h(-\delta V, V) \\ &= wI_0 e^{\kappa V_b}\ \left[e^{\kappa(V - \delta V)} - e^{\kappa(-\delta V - V)}\right] \\ &= wI_0 e^{\kappa(V_b - \delta V)}\ \left[e^{\kappa V} - e^{-\kappa V}\right] \\ &= 2wI_0 e^{\kappa(V_b - \delta V)}\ \sinh \kappa V \end{aligned}$$

APPENDIX C

Matlab

C.1 Derivation of f_a

Consider the assumption that the output signals have zero mean. Therefore, the cumulants are expressed as follows:

$$\begin{aligned}
\kappa_{2i} &= \mu_{2i} = E[y_i^2] \\
\kappa_{3i} &= \mu_{3i} = E[y_i^3] \\
\kappa_{4i} &= \mu_{4i} - 3\mu_{2i}^2 = E[y_i^4] - 3E[y_i^2]^2
\end{aligned}$$

Express the approximation of the entropy as

$$\begin{aligned}
\mathcal{H}_i &= \frac{288 \ln 2\pi - 288\kappa_{2i} - 48\kappa_{3i}^2 - 12\kappa_{4i}^2 + 12\kappa_{3i}^2\kappa_{4i}}{576} \\
&\quad + \frac{12\kappa_{4i}^3 - 1704\kappa_{3i}^2\kappa_{4i}^2 - 496\kappa_{4i}^4 - 213\kappa_{4i}^4}{576}
\end{aligned}$$

Compute

$$\begin{aligned}
\frac{\partial \mathcal{H}_i}{\kappa_{2i}} &= \frac{288}{576} \\
\frac{\partial \mathcal{H}_i}{\kappa_{3i}} &= \frac{-96\kappa_{3i} + 24\kappa_{3i}\kappa_{4i} - 3408\kappa_{3i}\kappa_{4i}^2 - 1984\kappa_{3i}^3}{576}
\end{aligned}$$

$$\frac{\partial \mathcal{H}_i}{\kappa_{4i}} = \frac{-24\kappa_{4i} + 12\kappa_{3i}^2 + 36\kappa_{4i}^2 - 3408\kappa_{3i}^2\kappa_{4i} - 852\kappa_{4i}^3}{576}$$

Compute also

$$\begin{aligned}
\frac{\partial \kappa_{2i}}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}[\mu_{2i}] = \frac{\partial}{\partial w_{ij}} E[y_i^2] = E[2y_i x_j] \\
\frac{\partial \kappa_{3i}}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}[\mu_{3i}] = \frac{\partial}{\partial w_{ij}} E[y_i^3] = E[3y_i^2 x_j] \\
\frac{\partial \kappa_{4i}}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}[\mu_{4i} - 3\mu_{2i}^2] = E[4y_i^3 x_j] - 12\mu_{2i} E[y_i x_j]
\end{aligned}$$

C.2 Matlab Source Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function F = func(unit_var,order);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if unit_var == 1
k3 = [0     1     0     0     0]';
k4 = [1     0     0     0    -3]';
p3 = [0     0     3     0    -3]';
p4 = [0     4     0   -12     0]';
else
k3 = [0     1     0     0     0]';
k4 = [1     0     0     0    -3]';
p3 = [0     0     3     0     0]';
p4 = [0     4     0     0     0]';
end
if order == 3
n = 13;
H3 = 96*pad(k3,n) - 24*pad(conv(k3,k4),n)
+ 3408*pad(conv(conv(k4,k4),k3),n)
+ 1984*pad(conv(conv(k3,k3),k3),n);
H4 = 24*pad(k4,n) - 12*pad(conv(k3,k3),n)
+ 3408*pad(conv(conv(k3,k3),k4),n)
- 36*pad(conv(k4,k4),n) + 852*pad(conv(conv(k4,k4),k4),n);
H3 = conv(H3,p3);
H4 = conv(H4,p4);
c = 576;
elseif order == 2
n = 9;
H3 = -8*pad(k3,n) + 60*pad(conv(k3,k4),n);
H4 = -2*pad(k4,n) + 30*pad(conv(k3,k3),n) + 9*pad(conv(k4,k4),n);
```

```
H3 = conv(H3,p3);
H4 = conv(H4,p4);
c = 48;
else
fprintf('\n No order is defined ..\n');
break;
end
if unit_var == 1
H2 = c*pad([1 0],length(H3));
else
H2 = pad(0,length(H3));
end
F = (H2 + H3 + H4)/c;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = pad(x,n);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y = zeros(n,1);
x = x(:);
m = length(x);
y(n-m+1:n) = x;
```

APPENDIX D

PSPICE Files

D.1 Circuit File

```
2 by 2 Separation Network Circuit
*****************
* Apply Voltages
*****************
Vdd     80   0   5.0V
Vss     70   0   0.0V

Vref    3    0   2.50
Vbmul   4    0   0.70
VbIV 5    0   1.10

Vdp     6    0  2.8
Vdm     7    0  2.2

Vs1     11   0  sin(2.5 0.1 1k)
Vs2     12   0  sin(2.5 0.1 2k)

Vdb11 121  0 2.70
Vdb12 122  0 2.54
Vcb11   141  0  2.60
Vcb12   142  0 2.40

Vdb21   221  0  2.56
Vdb22   222  0  2.70
Vcb21   241  0  2.65
Vcb22   242  0  2.45
```

```
*Vd11   321  0  2.70
*Vd12   322  0  2.54
*Vc11   341  0  2.60
*Vc12   342  0  2.40

*Vd21   421  0  2.56
*Vd22   422  0  2.70
*Vc21   441  0  2.65
*Vc22   442  0  2.45

Vbsos11 151  0 0.69
Vbsos12 152  0  0.69
Vbsos13 153  0  0.71

Vbsos21 251  0  0.69
Vbsos22 252  0  0.69
Vbsos23 253  0  0.71
*******************************
* Identify Nodes
*******************************
*       Vref    3
* Vbmul 4
*
* Vb1sos1 151 Vb2sos1 251
*       Vb1sos2 152 Vb2sos2 252
*       Vb1sos3 153 Vb2sos3 253
*
*       s1      11
*       s2      12
*
*       db11    121     db21    221
*       db12    122     db22    222
*
*       xb11    131     xb21    231
*       xb12    132     xb22    232
*       cb11    141     cb21    241
*       cb12    142     cb22    242
*
* e1+ 21 e2+ 31
* e1- 22 e2- 32
*
*       d11     321     d21     421
*       d12     322     d22     422

*       x11     331     x21     431
```

```
*       x12     332     x22     432
*       c11     341     c21     441
*       c12     342     c22     442

* y1 61 y2 62
************************************************************
* MYSOS s2 xb12 xb11 Vb1sos1  Vb1sos2  Vb1sos3 Vdd Vgnd Vref
*        1    2    3      11       12       13   8    7    9
************************************************************
XSOSE1 12  132  131  151 152 153 80 70 3 MYSOS
CE11     131   0   5pF
CE12     132   0   5pF
*********************************************************
* Subcircuit GMULD1  Vdd Vss V1 Vref V2 Vref Vb I+ I-
*                     8   7   1  2    3  4    5  21 22
*********************************************************
* realize e1 = cb11*x11 + cb12*xb12 + db11*s1 + db12*s2
XE11   80  70  141  3  131  3  4 21 22         GMULD1
XE12   80  70  142  3  132  3  4 21 22         GMULD1
XE13   80  70  121  3  11   3  4 21 22         GMULD1
XE14   80  70  122  3  12   3  4 21 22         GMULD1
************************************************************
* MYSOS s1 xb22 xb21 Vb1sos1  Vb1sos2  Vb1sos3 Vdd Vgnd Vref
************************************************************
XSOSE2   11  232  231  251 252 253 80 70 3       MYSOS
CE21     231   0   4pF
CE22     232   0   4pF
* realize e2 = cb21*x21 + cb22*xb22 + db21*s1 + db22*s2
XE21   80  70  241  3  231  3  4 31 32         GMULD1
XE22   80  70  242  3  232  3  4 31 32         GMULD1
XE23   80  70  221  3  11   3  4 31 32         GMULD1
XE24   80  70  222  3  12   3  4 31 32         GMULD1
************************************************************
* MYSOS y2 x12 x11 Vb1sos1  Vb1sos2  Vb1sos3 Vdd Vgnd Vref
************************************************************
XSOSY1  62  332  331  151 152 153 80 70 3       MYSOS
CY11     331   0   5pF
CY12     332   0   5pF
*********************************************************
* Subcircuit GMULD1  Vdd Vss V1 Vref V2 Vref Vb I+ I-
*********************************************************
* realize y1 = e1 - c11*x11 - c12*x12 - d12*y2
XY11   80  70  3  341  331  3  4 21 22         GMULD1
XY12   80  70  3  342  332  3  4 21 22         GMULD1
XY13   80  70  3  322  62   3  4 21 22         GMULD1
```

```
*********************************************************
* MYSOS y1 x22 x21 Vb1sos1  Vb1sos2  Vb1sos3 Vdd Vgnd Vref
*********************************************************
XSOSY2   11  432  431  251 252 253 80 70 3       MYSOS
CY21     431   0   4pF
CY22     432   0   4pF
*********************************************************
* Subcircuit GMULD1  Vdd Vss V1 Vref V2 Vref Vb I+ I-
*                     8   7   1  2    3  4    5  21 22
*********************************************************
* realize y2 = e2 - c21*x21 - c22*x22 - d21*y1
XY21   80  70  3  441  431  3  4 31 32         GMULD1
XY22   80  70  3  442  432  3  4 31 32         GMULD1
XY23   80  70  3  421  61   3  4 31 32         GMULD1
*********************************************************
* Subcircuit MYIV      Vdd Vss  I+   I-  Vbias    Vout
*                       8   7   1    2      4       3
*********************************************************
XIV1    80 70 21 22 5 7 MYIV
XIV2    80 70 31 32 5 72   MYIV
*********************************************
* Subcircuit TRANSAM1 V1 V2 Vb Vout Vdd Vgnd
*********************************************
* Use Follower
XF1 71 61 4 61 80 70 TRANSAM1
XF2     72 62 4 62 80 70  TRANSAM1
***************************
* Update equation of dij
***************************
* Subcircuit BLOCK Vdd Vss  yi Vdm  yj  Vb Vdp dij Vref
XB12    80  70   61 6  62  4  7  322 3       FGB
CD12    322 0   5nF
* Subcircuit BLOCK Vdd Vss  y2 Vdm  y2  Vb Vdp d21 Vref
XB21     80  70   62 6  61  4  7   421 3       FGB
CD21     421 0   5nF
***************************
* Update equations of cij
***************************
* Subcircuit WGMUL  Vdd Vss yi Vref xij Vref Vbias cij
XMUL11  80 70  61 3 331 3  4  341 WGMUL
XMUL12  80 70  61 3 332 3  4  342       WGMUL
XMUL21  80 70  62 3 431 3  4  441       WGMUL
XMUL22  80 70  62 3 432 3  4  442       WGMUL
CC11 341 0 1nf
CC12    342 0   1nf
```

```
CC21    441 0   1nf
CC22    442 0   1nf
*************
* Simulations
*************
.LIB comp.lib
.TRAN .5m  50m
.IC V(322) = 1
.IC V(421) = 4
.IC V(341) = 1
.IC V(342) = 4
.IC V(441) = 1
.IC V(442) = 4
.OP
.PROBE
.END
```

D.2 Library File

```
*****************************************************
* Transistor Models
*****************************************************
.MODEL nmos NMOS LEVEL=2
+ VTO=0.783736 KP=5.46E-5        GAMMA=0.5262      LAMBDA=3.533329E-2
+ TOX=412E-10  CGSO=2.669565E-10 CGDO=2.669565E-10 CJ=1.134E-4
+ MJ=0.708     CJSW=4.77E-10     MJSW=0.253        XJ=0.1500U
+ TPG=1.0000   LD=0.212340U      NSUB=5.860E+15    NFS=9.544270E+11
+ NEFF=1.0     NSS=1.0000E+12    DELTA=1.99612     VMAX=57874.1
+ UO=651       UEXP=0.177364     UCRIT=30664.6     PB=0.800  PHI=0.6
+ RSH=33.400   CGBO=4.250255E-10
*
.MODEL pmos PMOS LEVEL=2
+ VTO=-0.807   KP=2.13E-5        GAMMA=0.5644      LAMBDA=5.659491E-2
+ TOX=412E-10  CGSO=3.143031E-10 CGDO=3.143031E-10 CJ=2.54E-4
+ MJ=0.553     CJSW=3.31E-10     MJSW=0.352        XJ=0.05U
+ TPG=-1.0     LD=0.25U          NSUB=6.7400E+15   NFS=1.000E+11
+ NEFF=1.001   NSS=1.0000E+12    DELTA=1.001368E-6 VMAX=37082.8
+ UO=253.977   UEXP=0.2458       UCRIT=16929.2     PB=0.800  PHI=0.6
+ RSH=121.6000 CGBO=4.574377E-10
*****************************************************
* Subcircuit FGB  Vdd Vss S1 Vdm S2 Vb Vdp Vout Vref
*                   1   2  3   4  5  6   7    8    9
*****************************************************
.SUBCKT FGB  1  2  3  4  5  6  7  8 9
```

```
X1      1  2  3  4  9  6  11  12  5  FG
X2      1  2  7  3  9  6  12  11  5  FG
X3      1  12 10    PMIRROR
X4      2  10 8     NMIRROR
X5      1  11 8     PMIRROR
.ENDS
*********************************************************
* Subcircuit FG  Vdd Vss V1  Vd  V2  Vb Vout1 Vout2 Vref
*                  1   2  3   4   5   6     7     8    9
*********************************************************
.SUBCKT FG 1 2 3 4 5 6 7 8 9
M1  1   3   12  2   nmos W=6.0U L=6.0U
M2  11  4   12  2   nmos W=6.0U L=6.0U
M3  12  6   2   2   nmos W=6.0U L=6.0U
M4  12  13  2   2   nmos W=6.0U L=6.0U
M5  13  13  2   2   nmos W=6.0U L=6.0U
M6  14  13  2   2   nmos W=6.0U L=6.0U
M7  7   9   14  2   nmos W=6.0U L=6.0U
M8  8   5   14  2   nmos W=6.0U L=6.0U
M9  1   11  11  1   pmos W=6.0U L=6.0U
M10 1   11  13  1   pmos W=6.0U L=6.0U
.ENDS
***********************************
* Subcircuit PMirror Vdd Vref Vmir
*                      1    2    3
***********************************
.SUBCKT  PMIRROR   1  2  3
M1  1   2   2   1   pmos W=6.0U L=6.0U
M2  1   2   3   1   pmos W=6.0U L=6.0U
.ENDS
***********************************
* Subcircuit NMirror Vss Vref Vmir
*                      1    2    3
***********************************
.SUBCKT  NMIRROR   1  2  3
M1  2   2   1   1   nmos W=6.0U L=6.0U
M2  3   2   1   1   nmos W=6.0U L=6.0U
.ENDS
*************************************************
* Subcircuit WGMUL  Vdd Vss V1 V2 V3 V4 Vb Vout
*                   8   7   1  2  3  4  5  6
* Schematic: Mead's book page 95
*************************************************
.SUBCKT WGMUL    8  7  1  2  3  4  5 6
M1 12  1   13  7 nmos  W=6.0U L=6.0U
```

```
M2 16  2   13  7  nmos  W=6.0U L=6.0U
M3 12  12  8   8 pmos  W=6.0U L=6.0U
M4 16  16  8   8 pmos  W=6.0U L=6.0U
M5 15  12  8   8   pmos  W=6.0U L=6.0U
M6 18  16  8   8  pmos  W=6.0U L=6.0U
M7 11  4   15  8  pmos  W=6.0U L=6.0U
M8 14  4   18  8 pmos  W=6.0U L=6.0U
M9 14  3   15  8 pmos  W=6.0U L=6.0U
M10 11  3   18  8 pmos  W=6.0U L=6.0U
M11 11  11  7   7  nmos  W=6.0U L=6.0U
M12  14  14  7   7  nmos  W=6.0U L=6.0U
M13 17  11  7   7  nmos  W=6.0U L=6.0U
M14 6   14  7   7 nmos  W=6.0U L=6.0U
M15 17  17  8   8   pmos  W=6.0U L=6.0U
M16 6   17  8   8  pmos  W=6.0U L=6.0U
Mb  13  5   7   7 nmos  W=6.0U L=6.0U
.ENDS
*************************************************
* Subcircuit GMULD1  Vdd Vss V1 V2 V3 V4 Vb I+ I-
*                      8   7   1  2  3  4  5  21 22
* Schematic: Mead's book page 95
*************************************************
.SUBCKT GMULD1    8  7  1  2  3  4  5  21 22
M1      12  1   13  7   nmos  W=6.0U L=6.0U
M2      16  2   13  7   nmos  W=6.0U L=6.0U
M3      12  12  8   8   pmos  W=6.0U L=6.0U
M4      16  16  8   8   pmos  W=6.0U L=6.0U
M5      15  12  8   8   pmos  W=6.0U L=6.0U
M6      18  16  8   8   pmos  W=6.0U L=6.0U
M7      11  4   15  8   pmos  W=6.0U L=6.0U
M8      14  4   18  8   pmos  W=6.0U L=6.0U
M9      14  3   15  8   pmos  W=6.0U L=6.0U
M10     11  3   18  8   pmos  W=6.0U L=6.0U
M11     11  11  7   7   nmos  W=6.0U L=6.0U
M12     14  14  7   7   nmos  W=6.0U L=6.0U
M13     21  11  7   7   nmos  W=6.0U L=6.0U
M14     22  14  7   7   nmos  W=6.0U L=6.0U
Mb      13  5   7   7   nmos  W=6.0U L=6.0U
.ENDS
.ENDS
*********************************************************
* Subcircuit MYIV      Vdd Vss  I+   I-  Vbias   Vout
*                       8   7   1    2      4       3
*********************************************************
.SUBCKT MYIV            8   7   1    2  3  4
```

```
* Mirror current out of 1, add to to current coming from 2
M1      8   1   1  8    pmos  W=4.0U L=12.0U
M2      8   1   2  8    pmos  W=4.0U L=12.0U
* Output gain Stage
M3  8   2   4  7 nmos  W=4.0U L=12.0U
M4 4   3   7  7   nmos  W=4.0U L=12.0U
.ENDS
************************************************
* Subcircuit STRANSAM V1 V2 Vb Vout Vdd Vgnd
*                        1  2  3    4   5    6
*       W/L = 1/1
************************************************
.SUBCKT STRANSAM 1 2 3 4 5 6
M1      7  1  9  6      nmos  W=6.0U L=6.0U
M2      4  2  10 6      nmos  W=6.0U L=6.0U
M5 9  9  8  6 nmos  W=6.0U L=6.0U
M6 10 10 8  6 nmos  W=6.0U L=6.0U
Mb      8  3  6  6      nmos  W=6.0U L=6.0U
M3      7  7  5  5      pmos  W=6.0U L=6.0U
M4      4  7  5  5      pmos  W=6.0U L=6.0U
.ENDS
************************************************
* Subcircuit TRANSAM1  V1 V2 Vb Vout Vdd Vgnd
*                      1  2  3    4   5    6
* W/L = 1/1
************************************************
.SUBCKT TRANSAM1 1 2 3 4 5 6
M1      7  1  8  6      nmos  W=6.0U L=6.0U
M2      4  2  8  6      nmos  W=6.0U L=6.0U
Mb      8  3  6  6      nmos  W=6.0U L=6.0U
M3      7  7  5  5      pmos  W=6.0U L=6.0U
M4      4  7  5  5      pmos  W=6.0U L=6.0U
.ENDS
************************************************
* Subcircuit TRANSAM V1 V2 Vb Vout Vdd Vgnd
*                     1  2  3    4   5    6
*       W/L = 1/3
************************************************
.SUBCKT TRANSAM13 1 2 3 4 5 6
M1      7  1  8  6      nmos  W=6.0U L=6.0U
M2      4  2  8  6      nmos  W=6.0U L=6.0U
Mb      8  3  6  6      nmos  W=6.0U L=18.0U
M3      7  7  5  5      pmos  W=6.0U L=6.0U
M4      4  7  5  5      pmos  W=6.0U L=6.0U
.ENDS
```

```
*********************************************************
* Subcircuit MYSOS V1 V2 V3 Vb1  Vb2  Vb3 Vdd Vgnd Vref
*                  1  2  3  11   12   13   8    7    9
* G(A1) > G(A3)
*********************************************************
.SUBCKT MYSOS 1 2 3  11 12 13 8 7 9
XA1     1   2   11   2    8  7          STRANSAM
XA2     2   9   12   3    8  7          STRANSAM
XA3     2   3   13   2    8  7          TRANSAM13
.ENDS
```

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] B. Widrow *et al.*, "Adaptive noise cancellation: principles and applications," *IEEE Proceedings*, vol. 63, pp. 1691–1717, Apr. 1975.

[2] F. M. Salam, "An adaptive network for blind separation of independent signals," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 431–434, May 1993.

[3] K. S. Narendra and K. Parthasaraphy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, vol. 1, pp. 4–27, March 1991.

[4] C. Jutten, *Calcul neuromimétique et traitement du signal, analyse en composantes indépendantes.* Thèse de doctorat d'etat, Université Scientifique et Médicale-Institut Nationale Polytechnique, Grenoble, France, 1987.

[5] C. Jutten and J. Herault, "Space or time signal processing by neural networks models," *Proceedings on Neural Networks for Computing*, vol. 1, pp. 151 – 170, Apr. 1986.

[6] C. Jutten, J. Herault and A. Guerin, "In.c.a.: An independent components analyser based on neuromimetic network," *Artificial Intelligence and Cognitive Sciences*, pp. 201–219, 1988.

[7] J. Herault and C. Jutten, "Space or time adaptive signal processing by neural network models," *Neural Network for computing, AIP Conference Proceedings*, pp. 207–211, Apr. 1986.

[8] J. Herault and C. Jutten, "Blind separation of sources, part 1: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, pp. 1–10, July 1991.

[9] P. Comon, J. Herault and C. Jutten, "Blind separation of sources, part 2: Problem statement," *Signal Processing*, vol. 24, pp. 11–20, July 1991.

[10] E. Sorouchyari, "Blind separation of sources, part 3: Stability analysis," *Signal Processing*, vol. 24, pp. 21–29, July 1991.

[11] J.C.Fort, "Stability of the jh sources separation algorithm," *Signal Processing*, vol. 8, no. 1, pp. 35–42, 1991.

[12] J. Karhunen and J. Joutsensalo, "Representation and separation of signals using nonlinear pca type learning," *Neural Networks*, vol. 7, no. 1, pp. 113–127, 1993.

[13] A. Chichocki and L. Moszcznski, "New learning algorithm for blind separation of sources," *Electronic Letters*, vol. 28, pp. 1986–1887, 1992.

[14] K. Meriam, A. Belouchrani and J. F. Cardoso, "Assymptotic performance of second order blind source separation," *Proc. of IEEE International Symposium on Speech and Signal Processing*, vol. 4, pp. 277–280, 1994.

[15] E. Moreau and O. Macchi, "New self-adaptive algorithms for source separation based on contrast functions," *Proc. of IEEE International Symposium on Speech and Signal Processing*, vol. 1, pp. 515–519, 1993.

[16] N. Delfosse and P. Loubaton, "Adaptive separation of independent sources: A deflection approach," *Proc. of IEEE International Symposium on Speech and Signal Processing*, vol. 4, pp. 41–44, April 1994.

[17] J. Lacome and P. Ruiz, "Source identification: A solution based on cumulants," *Proc. of IEEE International Symposium on Speech and Signal Processing*, vol. Workshop, 1988.

[18] J. F. Cardoso, "Source separation using higher order moments," *Proc. of IEEE International Symposium on Speech and Signal Processing*, pp. 2109–2112, 1989.

[19] J. F. Cardoso, "Super-symmetric decomposition of the fourth order cumulant tensor, blind identification of more sources than sensors," *Proc. of IEEE International Symposium on Speech and Signal Processing*, pp. 14–17, 1991.

[20] J. F. Cardoso, "Iterative techniques for blind source separation using only fourth order cumulants," *European Signal Processing Conference*, pp. 739–742, 1992.

[21] J. F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *submitted to IEEE Trans. on Signal Processing*, 1996.

[22] J. F. Cardoso and P. Comon, "Independent component analysis, a survey of some algebraic methods," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 93–96, May 1996.

[23] P. Comon, "Independent component analysis: A new concepts?," *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[24] S. Amari, A. Cichocki ans H. Yang, "A new learning algorithm for blind signal separation," *MIT Press (in press)*, 1996.

[25] A. Bell and J. Sejnowski, "Fast blind separation based on information theory," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 43–47, December 1995.

[26] T. Bell and T. J. Sejnowski, "A non-linear information maximisation algorithm that performs blind separation," *Advances in Neural Information Processing Systems 7, MIT Press*, pp. 467–474, 1995.

[27] G. Giannakis, Y. Inouye and J.M. Mendel, "Cumulants based identification of multichannel moving average models," *IEEE Trans. on Automatic Control*, vol. 34, pp. 783–787, 1989.

[28] L. Tong, R. Liu, V. C. Soonand Y. Huang, "Inderminacy and identifiability of blind identification," *IEEE Trans. on Signal Processing*, vol. 38, pp. 409–509, May 1991.

[29] L. Tong, Y. Inouye and R. Liu, "Wave-preserving blind estimation of multiple independent sources," *IEEE Trans. on Signal Processing*, vol. 41, pp. 2461–2470, July 1993.

[30] X. Ling, Y. Huang and R. Liu, "A neural network for blind signal separation," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 27, pp. 69–72, May 1994.

[31] V. C. Soon, L. Tong, F. Huang and R. Liu, "An extended fourth order blind identification algorithm in spatially correlated noise," *Proc. of IEEE International Symposium on Speech and Signal Processing*, pp. 1365–1368, 1990.

[32] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE trans. on Antennas Propagation*, vol. AP, pp. 276–280, March 1986.

[33] R. Roy and T. Kailath, "Esprit-estimation of signal paramater via rotational invariance techiques," *IEEE trans. on signal Processing*, vol. 37, pp. 984–993, July 1989.

[34] B. Agee, "Blind separation and capture of communication signals using a multitarget constant modulus beamformer," *IEEE Millitary Communication Conference*, pp. 340–346, 1989.

[35] A. Swindlehurst, S Daas and J. Yang, "Analysis of a decision directed beamformer," *IEEE Trans. on Signal Processing*, vol. 43, pp. 2920–2927, December 1995.

[36] B. Agee, A. Schell and W. Gardner, "Spectral self-coherence restoral: A new approach to blind adaptive signal extraction using antenna arrays," *IEEE Proceedings*, vol. 78, pp. 753–767, April 1990.

[37] S. Talwar, M. Viberg and A. Paulraj, "Blind separation of synchronous cochannel digital signals using an antenna array," *IEEE Trans. on Signal Processing*, vol. 44, pp. 1184–1197, May 1996.

[38] A. Belouchrani and F. F. Cardoso, "Maximum likelihood source separation by the expectation-maximization technique: Deterministic and stochastic implementation," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 49–53, December 1995.

[39] D. Pham, P. Garrat and C. Jutten, "Separation of a mixture of independent sources through a maximum likelyhood approach," *European Signal Processing Conference*, pp. 771–774, 1992.

[40] G. Giannakis and M. Tsatsanis, "A unifying maximum likelyhood view of cumulants and polyspectral measures for non-gaussian signal classification and estimation," *IEEE Trans. on Information Theory*, vol. 38, pp. 386–406, March 1992.

[41] E. Moulinès, P. Duhamel, J. F. Cardoso and S. Marrague, "Subspace methods for the blind identification of multichannel fir filters," *IEEE Trans. on Signal Processing*, vol. 43, pp. 516–525, February 1995.

[42] S. Gerven and D. Compernolle, "Feedforward and feedback in a symmetric adaptive noice canceller: Stability analysis in simplified case," *Signal Processing*, pp. 1081–1084, 1992.

[43] H. Thi and C. Jutten, "Blind source separation for convolutive mixtures," *Signal Processing*, vol. 45, pp. 209–229, 1995.

[44] T. Bell and T. J. Sejnowski, "Blind separation and blind deconvolution: An information-theoretic approach,," *Proc. of IEEE International Symposium on Speech and Signal Processing*, pp. 3415–3418, May 1995.

[45] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[46] A. B. Gharbi and F. M. Salam, "Blind separation of independent sources in linear dynamical media," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, December 5-9 1993.

[47] A. B. Gharbi and F. M. Salam, "Separation of mixed signals: Formulation and some implementations," *Proc. of IEEE Midwest Symposium on Circuits and Systems*, August 4-6 1994.

[48] E. Vittoz and X. Arreguit, "Cmos integration of herault-jutten cells for separation of sources," *Proceedings Workshop on Analog VLSI and Neural Systems*, May 1989.

[49] M. H. Cohen and G. Andreou, "Current-mode subthreshold mos implementation of herault-jutten autoadaptive network," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 714–727, May 1992.

[50] A. B. Gharbi and F. M. Salam, "Implementation and experimental results of a chip for the separation of mixed and filtered signals," *Journal of Circuits, Systems and Computers*, vol. 6, pp. 115–139, April 1996.

[51] A. B. Gharbi and F. M. Salam, "Test results of a chip for the separation of mixed and filtered signals," *IEEE Trans. on Circuits and Systems*, vol. 42, pp. 748–751, November 1995.

[52] A. B. Gharbi and F. M. Salam, "Implementation and test results of a chip for the separation of mixed signals," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 271–274, May 1995.

[53] E. Oja, "A simplified neuron model as a principal componenet analyzer," *Journal of Mathematical Biology*, vol. 15, pp. 267–273, 1982.

[54] E. Oja, "Principal components, minor components, and subspaces," *Neural Networks*, vol. 1, pp. 61–68, 1992.

[55] S. Amari, *Differential-Geometry Methods in Statistics, Lecture Notes in Statistics, vol. 28*, vol. 28. New York: Springer, 1985.

[56] J. F. Cardoso, "The invariant approach to source separation," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 55–60, December 1995.

[57] C. Jutten and J. F. Cardoso, "Separation of sources: Really blind?," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 79–84, December 1995.

[58] S. Kullback, *Topics in statistical information theory*. New York: Springer-Verlag, 1987.

[59] Chi-Tsong Chen, *Linear System Theory and Design*. New York: Holt, Rinehart and Winston Publishing, 1984.

[60] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Economics*. New York: John Wiley Series in Probability and Mathematical Statistics, 1988.

[61] D. Yellin, E. Weinstein, "Multichannel signal separation: Methods and analysis," *IEEE Trans. on Signal Processing*, vol. 44, pp. 106–118, January 1996.

[62] John E. Kolassa, *Series Approximation Methods in Statistics*. New York: Springer-Verlag, 1994.

[63] A. Cichocki, S. Amari, M. Adachi and W. Kasprzak, "Self-adaptive neural networks for blind separation of sources," *Proc. of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 157–160, May 1996.

[64] S. Amari, A. Cichocki and H. H. Yang, "Recurrent neural networks for blind separation of sources," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 37–42, December 1995.

[65] A. Cichocki, W. Kasprzak and S. Amari, "Multi-layer neural network with local adaptive learning rules for blind separation of source signals," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 61–65, December 1995.

[66] L. De Lathauwer, P. Comon, B. De More and J. Vandewalle, "Application in independent component analysis," *Proc. Of IEEE International Symposium on Nonlinear Theory and its Applications*, pp. 91–96, December 1995.

[67] Peter McCullagh, *Tensor methods in statistics.* New York: Chapman and Hall, 1987.

[68] Gram Charlier, *Application, de la theorie des probabilites a l'astronomie, Part of the traite.* Paris: Gauthier-Villars, 1931.

[69] L. Watts, D. A. Kerns, R. F. Lyon and C. A. Mead, "Improved implementation of the silicon cohhlea," *IEEE Journal on Solid-State Circuits*, vol. 27, pp. 692–700, May 1992.

[70] B. Gilbert, "A precise four-quadrant multiplier with subnanoseconds response," *IEEE Journal of Solid-State Circuits*, vol. SC, no. 3, pp. 365–372, 1968.

[71] C. Mead, *Analog VLSI and Neural Systems.* New York: Prentice Hall, 1989.

[72] A. Papoulis, *Probability, random variables and Stochastic Processes.* New York: McGraw-Hill, 1984.