# A Neurodynamic Optimization Approach to Constrained Pseudoconvex Optimization

GUO, Zhishan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Mechanical and Automation Engineering

The Chinese University of Hong Kong
July 2011

**<u>Thesis/Assessment Committee</u>**

**Professor YAM, Yeung (Chair)**

**Professor WANG, Jun (Thesis Supervisor)**

**Professor LIU, Yunhui (Committee Member)**

**Professor JIANG, Zhongping (External Examiner)**

Abstract of thesis entitled:

A Neurodynamic Optimization Approach to Constrained Pseudoconvex Optimization

Submitted by GUO, Zhishan

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in June 2011

In this thesis, based on an existing model for constrained convex optimization, a one-layer recurrent neural network is proposed for solving pseudoconvex optimization problems subject to linear equality constraints.

The global convergence of the reconstructed neural network can be guaranteed for any pseudoconvex objective function. The finite-time state convergence to the feasible region defined by the equality constraints is also proved. In addition, global exponential convergence is proved when the objective function is strongly pseudoconvex on the feasible region.

Besides simulation results on illustrative samples, an application on chemical process data reconciliation is provided to demonstrate the effectiveness and characteristics of the neural network.

The model is then modified to solve pseudoconvex optimization with linear equality constraints and also bound constraints. Application on real-time portfolio optimization is shown.

# 摘要

本論文將一種單層回饋神經網路改進，使之能夠解決含線性等式約束條件的偽凸優化問題。

本文從理論上證明了在優化函數是偽凸的情況下，重新構造的神經網路模型仍然可以保證全域收斂到最優解。同時本論文還證明了到可行域的有限時間收斂，以及在優化目標函數是強偽凸的情況下，到最優解的全域指數收斂。

在列舉了一些簡單數值例子的方針結果之後，本論文研究了該神經網路模型的一類工程應用：化工過程資料和解。通過模擬實驗，該神經網路優化方法的有效性和相關特徵得以闡述。

此外，在隨後的一章，該神經網路優化模型得到了進一步的改進。改進之後的模型不僅對優化函數的條件放寬了，同時可以解決包含上下界約束的偽凸優化問題。這一章以證券投資優化作為該模型求解相關優化問題的一個應用，具體闡述了模擬結果，並做出相關分析。

# Acknowledgement

It is an honor for me to study for M. Phil. degree in the Chinese University of Hong Kong during the past two years. I am indebted to the faculty of the Department of Mechanical and Automation Engineering for their contribution to my M. Phil. education.

I am heartily thankful to my supervisor, Prof. Jun Wang, for providing me the opportunity to study here in Hong Kong. It is him who led me into the world of research and introduced me to the field of computational intelligence. His guidance, support, and encouragement always stimulate me to learn more in doing research and writing better papers.

Many people have helped me during my M. Phil. study. I am lucky to have Dr. Qingshan Liu as a labmate for several months, from whom I benefited much in mathematical proving and dynamical system analysis. Mr. Jin Hu often kindly helps me with math problems during my research, and he also helped me a lot at the beginning of my M. Phil study. With discussions with Prof. Zhigang Zeng and Mr. Zheng Yan, the picture of math and computational intelligence is getting clearer.

Besides research and learning, life is colorful and enjoyable with the presence of my colleagues and friends Ms. Ning Sui, Mr. Chengwei Guo, Mr. Jingwen Dai, Mr. Xuefeng Hu, Mr. Sze Ho Chim, Mr. Kwun Ming Cheng and also many other people without space to list.

Finally, my special thanks go to my parents. I would never be here without their unconditional love, encouragement and support. This thesis is dedicated to them.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

**Chapter Outline**

In this chapter, basic backgrounds and the motivations of this thesis are described. With the rapid development in industry, optimization of more and more procedures are necessary. Many of these optimization problems are time-varying ones, where the parameters of the processes are changing from time to time. Real-time optimization of these dynamic processes is a major challenge for traditional optimization methods. Recurrent neural networks have good properties that can be utilized to solve some of these problems effectively. The last section also gives the organization of this thesis.

## 1.1 Constrained Pseudoconvex Optimization

Constrained optimization is optimizing an objective function over a domain. With mathematical notations, it can be written as

$$\begin{aligned} \text{minimize} \quad & f(x), \\ \text{subject to} \quad & x \in \Omega, \end{aligned} \tag{1.1}$$

where $x \in \mathbb{R}^n$, and the domain $\Omega$ can be described by a group of equality and inequality constrains.

Many problems in science and engineering can be formulated as Constrained optimization problems, such as robot control, manufacturing system design, signal and image processing, and pattern recognition [9] [8]. Most of the time, the objective function and domains are not static, which can be described as having a list of time-varying parameters. Then the problem is time-varying constrained optimization one.

$$\begin{aligned} \text{minimize} \quad & f(x, \theta(t)), \\ \text{subject to} \quad & x \in \Omega(t), \end{aligned} \tag{1.2}$$

In this thesis, the following nonlinear optimization problem with equality constraints will be considered.

$$\begin{aligned} \text{minimize} \quad & f(x), \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{1.3}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables, $A \in \mathbb{R}^{m \times n}$ is a coefficient matrix with full row-rank (i.e., $\text{rank}(A) = m \leq n$), and the objective function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is differentiable, bounded below, locally Lipschitz continuous [51] [60] and pseudoconvex on the feasible region $\{x | Ax - b = 0\}$. In this thesis, we assume that problem (1.3) has at least one finite solution.

While most approaches to optimization focus on convex optimization, nonconvex optimization is rarely investigated. In particular, among nonconvex optimization problems, constrained pseudoconvex optimization will be studied in this thesis, and the definitions of pseudoconvex, strictly pseudoconvex, and strongly pseudoconvex are given below.

*Definition 1*: A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be pseudoconvex on a set $\Omega$ if $\forall x, y \in \Omega, x \neq y$,

$$\nabla f(x)^T(y-x) \geq 0 \Rightarrow f(y) \geq f(x).$$

The function $f$ is said to be strictly pseudoconvex on $\Omega$ if $\forall x \neq y \in \Omega$,

$$\nabla f(x)^T(y-x) \geq 0 \Rightarrow f(y) > f(x);$$

and strongly pseudoconvex on $\Omega$ if there exists a constant $\beta > 0$ such that $\forall x \neq y \in \Omega$,

$$\nabla f(x)^T(y-x) \geq 0 \Rightarrow f(y) > f(x) + \beta\|x-y\|_2^2,$$

where $\|\cdot\|_2$ is the $L_2$-norm, which will be written as $\|\cdot\|$ hereafter.

From the definition, pseudoconvex is weaker than convex, which means approaches on constrained pseudoconvex optimization are promising ones to convex optimization problem, but not so vice versa. Constrained pseudoconvex optimization has widespread applications, such as fractional programming [19] [11], frictionless contact analysis [7], production planning, financial and corporate planning [24], computer vision [57], healthcare and hospital planning.

Real-time solutions are often required in engineering applications, especially for time-varying optimization problems. Current methods on pseudoconvex optimization mostly face specific problems, but not generalized ones, while a few of them may not provide optimal solution real-time. One promising approach for solving optimization problems in real time is recurrent neural networks (RNNs).

## 1.2 Recurrent Neural Networks

Neural networks have become a popular research field in computer science, mathematics, and cognitive science. Historically, the neural network was first proposed to study the brain theory. Then, with the developing computer technologies, neural networks are considered more as a powerful family of nonlinear model and analog computing paradigm. As the counterparts of biological neural systems, properly designed artificial neural networks can serve as goal-seeking computational models for solving various optimization problems.

Among various neural networks, a couple of networks have been utilized for optimization, such as Hopfield neural networks [29], self-organizing feature maps [40], and Boltzmann machines [1]. Compared with traditional numerical methods for constrained optimization, recurrent neural network has several advantages:

- it can solve many optimization problems with time-varying parameters;
- it can handle large-scale problems with its parallelizable ability;
- it can be implemented effectively using VLSI and optical technologies [15].

Therefore, neural network can solve optimal control problems in running times at the orders of magnitude much faster than the most popular optimization algorithms executed on general-purpose digital computers.

In 1940s, the first conceptual elements of neural networks were introduced. Since then, numerous neural network models have been developed. Most feedforward neural networks have layered structure, synchronous mode, operate on discrete time, are described by the difference equations, and thus are studied by arithmetic methods. However, recurrent neural networks have interconnected structure, asynchronous mode, operate on continuous time, are described by the ordinary differential equations, and thus are studied by ordinary differential equation theory. From the system theory viewpoint, continuous recurrent neural networks can be regarded as a continuous dynamic system, and thus its stability property can be investigated.

Moreover, the continuous model is superior to the discrete one in terms of the local minimum problem, because of its smoother energy surface. Hence, the continuous recurrent networks have dominated the solving techniques for optimization problems, especially for combinatorial problems, and will be the major concern of this thesis.

Consider recurrent neural network as a system, the inputs are signals coming from the environment, while outputs are the ones go out of the network to the environment, encoding the end result of the computation. As a "computing machine", which was associated with automation, the computability and complexity of neural networks are also concerned. A most popular automation is the Turing machine, which is mathematical equivalent to modern computer. It is shown that the analog recurrent neural network is more powerful than Turing machine, although their upper limits are both unknown [68]. Besides the ultimate power, theories in computational complexity also contains expressive power constraints on resources, such as time and space. Thus scalability of neural networks is very important: some neural networks has shown great performance on small problems, but when the are scaled up to larger problems, too more neurons are often necessary. The number of neurons, number of connections, and also convergence time are all resources for neuron computation.

In this thesis, a one-layer recurrent neural network with very simple architecture is proposed to solve constrained pseudoconvex optimization problem. Among existing neural network models to related problems, it has the least number of neurons and connections. It's finite time convergence to the feasible region also shows its low computational complexity.

## 1.3 Thesis Organization

This thesis consists of six chapters. After the introduction of backgrounds and motivations, Chapter 2 gives the literature review. Then, an existing recurrent neural network model is extended for solving pseudoconvex optimization problems subject to linear equality constraints in Chapter 3, where theoretical analysis of the neural network model will also be stated. Chapter 4 shows several simple simulation results based on numerical examples, and two applications of the proposed neural network model on chemical process data reconciliation and robot motion planning will be discussed afterwards in Chapters 5 and 6. Chapter 7 introduces another recurrent neural network model and shows a promising application on real-time portfolio optimization. Finally Chapter 8 concludes the thesis.

□ **End of chapter.**

# Chapter 2

# Literature Review

**Chapter Outline**

In this chapter, researches related to the work described in this thesis are reviewed. The first section mainly focuses on studies on pseudoconvex optimization. The second section reviews works on neurodynamic optimization approaches.

## 2.1 Pseudoconvex Optimization

In classical scalar optimization theory, convexity plays a fundamental role since it guarantees the validity of important properties: a local minimizer is also a global minimizer, a stationary point is a global minimizer and the usual first order necessary optimality conditions are also sufficient for a point to be a global minimizer. Many algorithms have been proposed to solve constrained optimization problems in the past decades, such as Rosen's gradient projection method in 1960 [65] and Zangwill's penalty function method in 1967 [83].

Pseudoconvexity was formulated and first studied in 1960s [51], where it is pointed out that pseduconvexity turns out to be weaker than strict convexity and stronger than quasiconvexity. In particular it is shown that a local minimum pseduconvexity function over a convex set of Euclidean n-space is at the same time a global minimum [60]. With this properties, the vanishing of the gradient insures a global minimum, which made the extension of several basic results of convex programming to pseudoconvex ones possible.

However, from a practical point of view, it is still very difficult to recognize pseudoconvexity. From Definition 1, pseudoconvexity is defined by conditions involving infinitely many inequalities and that there is no general criterion requiting the examination of finitely many conditions. Necessary and sufficient second order conditions are obtained in this thinking [17] [55].

Since the definition of pseudoconvexity in 1960s, its strict relationship with fractional programming has been highlighted and from the beginning, fractional programming has benefited from advances in generalized convexity, and vice versa [69]. Pseudoconvex fractional programming are extremely important for its good properties and numerous applications such as Data Envelopment Analysis, tax programming, risk and portfolio theory, logistics and location theory [6] [12] [13] [56]. Many papers about fractional programming have been published in the last decades and both theoretical and algorithmic point of views have been handled [54] [22] [41] [8] [10]. Several of these results deal with the pseudoconvexity of the objective function, since this property plays a key role in the study of minimization problems. Among them, most algorithms solve the pseudoconvex optimization problem by transforming it into convex one, and many of them requires various kinds of properties of the objective function.

## 2.2 Recurrent Neural Networks

In the past two decades, recurrent neural networks for optimization and their engineering applications have been widely investigated. Since Tank and Hopfield's pioneering work on a neural network approach to linear programming [70], the theory and applications of recurrent neural networks for optimization have been widely investigated. Based on the penalty functions and gradient methods, constrained optimization problems were first converted approximately or exactly to unconstrained optimization problems, then some gradient-based neural network models were constructed to compute the approximate or exact optimal solutions. The gradient method was widely used in the neural network design. Two classical recurrent neural network models for optimization are the Hopfield neural network [70] for linear programming, and the one by Kennedy and Chua [38] for nonlinear programming.

The common LP problems have the following formulation:

$$\begin{aligned} \text{minimize} \quad & f(x) = c^T x, \\ \text{subject to} \quad & Ax \geq b(g_j(x) \geq b_j, j = 1, \ldots, m), \end{aligned} \tag{2.1}$$

where $x, c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$. Tank and Hopfield first proposed the neural network structure to solve the LP problem, and its energy function can be defined as

$$E_{TH}(x) = f(x) + \sum_{j=1}^{m} (g_j^+(x)^2 + \sum_{i=1}^{n} x_i^2/2sR_{ii}, \tag{2.2}$$

where $R$ is an $n \times n$ diagonal matrix, $g^+ = (g_1^+, \ldots, g_m^+)^T$ is a penalty vector when $g_(x) < b_j$, and $s > 0$ is a penalty parameter which must be sufficiently large.

The Hopfield neural network emulates the behavior of neurons through an electronic circuit. After the circuit is established with many similar sets of components, a parallel computation can be realized for an on-line solution. These special characteristics are beneficial for real-time optimization and have been applied to solve optimization problems.

Kennedy and Chua proposed a kind of neural network structure based on Hopfield network with an inexact penalty function, where is energy function is

$$E_{KC}(x) = f(x) + \frac{s}{2} \sum_{j=1}^{m} (g_j^+(x))^2, \tag{2.3}$$

where $s > 0$ is the penalty parameter, and for an inexact penalty rule, the states converges to the solution as $s \to \infty$. To improve this, Rodríguez-Vázquez et al. [62] directly use another penalty method and transform the LP problem into an unconstraint optimization problem, where its energy function is illustrated as

$$E_{RV}(x) = f(x) + \alpha \left| \sum_{j=1}^{m} \min\{0, g_j(x) - b_j\} \right|, \tag{2.4}$$

where $\alpha > 0$. It is shown that once the feasible region is reached, the trajectory moves toward the minimal direction of the objective function.

In addition to the gradient method, other methods have been developed for neurodynamic optimization. The penalty function method is a popular technique for optimization in which it is used to construct a single unconstrained problem or a sequence of unconstrained problems. Although non-linear programming problems are complex in computation, neural network approaches with a penalty function offer a faster convergence [28]. A general non-linear programming problem is in the following form:

$$\begin{aligned} \text{minimize} \quad & f(x), \\ \text{subject to} \quad & h_i(x) = 0, i = 1, \ldots, n; \\ & g_j(x) \le 0, j = 1, \ldots, m; \\ & x \in \Omega. \end{aligned} \tag{2.5}$$

In general, the penalty function $f_p(x)$ is represented as

$$f_p(x) = f(x) + \sum_{j=1}^{m} K_j \max\{0, g_j(x)\}^\beta + \sum_{i=1}^{n} K_i(|h_i(x)|^\alpha/\alpha), \tag{2.6}$$

where $\alpha, \beta > 0$, and the penalty parameters are $K_i, K_j$, which are usually sufficiently large or increase as network training. A searching method can be adopted to search for the solution in the decision space, and the steepest descent approach is the popular technique for obtaining the searching direction [15].

The deterministic annealing neural networks were developed for solving linear and nonlinear convex programming problems by Wang [72], which were utilized to solve the assignment problems [73] and shortest path problems [74]. Some differentiable and convex penalty functions are given and used, for example

$$f_p(x) = \frac{1}{2}[g(x) + v^+]^T[g(x) + v^+] + \frac{1}{2}h(v)^T h(v). \tag{2.7}$$

Based on the Lagrangian function and Lagrangian optimality conditions, the Lagrangian network [84] was proposed for solving the general optimization problems, in which the local convergence was guaranteed. For convex optimization, the global convergence of the Lagrangian network was analyzed and proven by Xia [77]. Similar to the penalty function methods, Lagrange multiplier and augmented Lagrange multiplier methods merge an objective function and constraints into an energy function of the target networks. This category uses two kinds of dynamic structures to handle real variables and Lagrange multiplier (dual variables), and its energy function for linear programming is

$$E_L(x) = f(x) + \sum_{j=1}^{m} \lambda g_j(x). \tag{2.8}$$

In recent years, based on the Karush-Kuhn-Tucker (KKT) optimality conditions, the primal-dual network [76], dual network [80] and simplified dual network [45] were developed for solving convex optimization problems. It is suggested that the equilibrium point is the same as the exact solution when simultaneously solving the primal problem and its dual problem. The energy function of the primal-dual network for linear programming is

$$\begin{aligned} E_{PD}(x,y) = \frac{1}{2}(c^T x - b^T y)^2 + \frac{1}{2}x^T(x - |x|) + \frac{1}{2}y^T(y - |y|) \\ + \frac{1}{2}\|Ax - b\|_2^2 + \frac{1}{2}(A^T y - c)^T[(A^T y - c) - |(A^T y - c)|]. \end{aligned} \tag{2.9}$$

These neural network approaches have three important advantages: (i) avoiding a significant amount of parameters, (ii) escaping from an infeasible solution, and (iii) being able to estimate the duality gap indirectly.

Furthermore, based on the projection method (e.g., [78] [82], [32]), optimality conditions for constrained optimization problems can be written in the form of linear (or nonlinear) variational inequalities, and transformed into projection equations.

Then neural networks based on the projection equations were constructed for solving the constrained optimization problems. Moreover, for convex optimization problems, the global convergence of the projection neural networks can be guaranteed for the global optimal solutions [79] [33] [5]. To reduce the model complexity, some one-layer recurrent neural networks were proposed for solving linear and quadratic programming problems [46] [47]. As there is no bound constraints, the projection neural network will degeneration to the Lagrangian network [84][85] which is given by the following equations for problem (4.1) where $x$ is the output state vector and $y$ is the hidden state vector.

$$\begin{array}{l} \frac{dx}{dt} = -\nabla f(x) + A^T y, \\ \frac{dy}{dt} = -Ax + b. \end{array} \tag{2.10}$$

Another projection network, the two-layer recurrent neural network in [81] is proposed for quadratic programming with linear inequality constraints, and its equivalent form for problem (4.1) can be described as follows:

$$\begin{array}{l} \frac{dx}{dt} = -\nabla f(x) + [A^T, -A^T]y^+, \\ \frac{dy}{dt} = -y + y^+ + \begin{bmatrix} A \\ -A \end{bmatrix} x + \begin{bmatrix} -b \\ b \end{bmatrix}; \end{array} \tag{2.11}$$

where $y^+ = [y_1^+, ..., y_m^+]^T$, and $y_i^+ = \max(0, y_i)$.

Apart from the recurrent neural networks for solving smooth constrained optimization problems, neurodynamic approaches to nonsmooth constrained optimization were investigated by some researchers recently. The generalized nonlinear programming circuit for solving nonsmooth nonconvex optimization problems were proposed by Forti et al. [25]. Specifically, some recurrent neural networks for solving linear and quadratic programming with discontinuous activation functions were proposed [45] [48].

$$\epsilon\frac{dx}{dt} \in -Px - (I-P)\partial f(x) + q, x_0 = x(t_0), \qquad (2.12)$$

where $x$ is the state vector, $\epsilon$ is a positive scaling constant, $I$ is the identity matrix, $P = A^T(AA^T)^{-1}A$, $q = A^T(AA^T)^{-1}b$, and $\partial f(x)$ is the sub-differential of $f(x)$.

While most neural network approaches to optimization focus on convex optimization, nonconvex optimization is rarely investigated. In particular, among nonconvex optimization problems, pseudoconvex optimization has many applications. Hu and Wang [31] extended the projection neural network for optimization with differentiable pseudoconvex objective functions and bound constraints.

However, for more general pseudoconvex optimization problems subject to linear equality constraints, the projection neural network is not applicable for solving these problems due to its convergence conditions.

---

□ **End of chapter.**

# Chapter 3

# Model Description and Convergence Analysis

**Chapter Outline**

In this chapter, a new RNN model is proposed for solving pseudoconvex optimization problems subject to linear equality constraints. After the model is given, some necessary preliminary concepts and results including the definition of global convergence are stated for further discussion. Theorem 1 discusses the finite-time convergence to the feasible region of problem (1.3). Then in Theorems 2 and 3, the Lyapunov stability of the proposed neural network is proved, based on which the globally convergence to the optimal solution of problem (1.3) is then shown. Theorem 4 analyzes the exponential convergence of the neural network when the gradient of the optimal function $\nabla f(x)$ is strongly pseudomonotone.

## 3.1 Model Descriptions

Consider problem (1.3), and an existing recurrent neural network model (2.12) [48], when $f$ in problem (1.3) is differentiable, $\nabla f(x)$ is used instead of $\partial f(x)$ as the differential of $f(x)$. Therefore, we have:

$$\epsilon \frac{dx}{dt} = -Px - (I-P)\nabla f(x) + q, x_0 = x(t_0).$$

However, its global convergence results are established for convex optimization problems only, and theoretically its state reaches the feasible region only when time approaches to infinity.

In this thesis, in order to guarantee the finite-time convergence to the feasible region and global convergence to optimal solutions for pseudoconvex optimization problems, the model is modified to:

$$\epsilon \frac{dx}{dt} = -(I-P)\nabla f(x) - A^T g(Ax-b), x_0 = x(t_0), \qquad (3.1)$$

where $g = (g(x_1), g(x_2), \cdots, g(x_m))^T$ and its component is defined as

$$g(x_i) = \begin{cases} 1, & \text{if } x_i > 0, \\ 0, & \text{if } x_i = 0, \ (i = 1, 2, \ldots, m). \\ -1, & \text{if } x_i < 0; \end{cases} \qquad (3.2)$$

The structure of one-layer recurrent neural network (3.1) is shown in Figure 3.1, where $[p_{ij}]_{n\times n} = P$, $[c_{ij}]_{n\times n} = I - P$, $[a_{ij}]_{m\times n} = A$, $[b_1, b_2, \ldots, b_m]^T = b$, and $h_i(\cdot) = \partial f(\cdot)/\partial x_i$ for

$i = 1, 2, \ldots, n$.

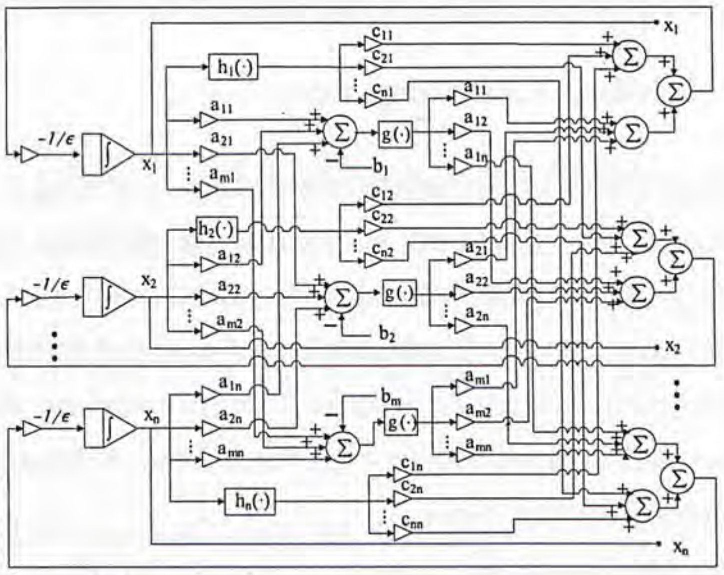

Figure 3.1: The structure of one-layer recurrent neural network (3.1).

For the convenience of later discussions, several definitions and theorems on pseudoconvex optimization are introduced at first.

*Definition 2*: A function $F : \mathbb{R}^n \to \mathbb{R}^n$ is said to be pseudomonotone on a set $\Omega$ if $\forall x, x' \in \Omega, x \neq x'$,

$$F(x)^T(x' - x) \geq 0 \Rightarrow F(x')^T(x' - x) \geq 0. \tag{3.3}$$

A very important result on pseudoconvex optimization is given by the following lemma, and its proof follows directly from Theorem 4.3.8 in [8].

*Lemma 1* [8]: For problem (1.3), if the Karush-Kuhn-Tucker conditions hold at a feasible solution $\bar{x}$; i.e., $\exists y \in \mathbb{R}^m, \nabla f(\bar{x}) - A^T y = 0$, then $\bar{x}$ is a global optimal solution to problem (1.3).

## 3.2 Global Convergence

The state vector of the neural network (3.1) is said to be globally convergent to an optimal solution of problem (1.3) if for any $x(t)$ of the neural network with initial point $x_0 \in \mathbb{R}^n$, such that $\lim_{t \to +\infty} x(t) = x^*$, where $x^*$ is an optimal solution. Since the dynamical system is described by an ordinary differential equation with discontinuous right hand side, Filippov solution is considered in this thesis.

The existence of the solution can be derived from the locally Lipschitz continuity of the objective function $f(\cdot)$ and Proposition 3 in [16]. The solution for discontinuous system may not be unique [3], and the LaSalle invariant set theorem does not require the uniqueness of the solution.

Denote the feasible region as $\mathcal{S} = \{x | Ax = b\}$.

*Theorem 1:* The state vector of the recurrent neural network (3.1) is globally convergent to the feasible region $\mathcal{S}$ in finite time by $t_{\mathcal{S}} = \epsilon \|Ax_0 - b\|_1 / \lambda_{\min}(AA^T)$ and stays there thereafter, where $x_0$ is the initial value, and $\lambda_{\min}$ is the minimum eigenvalue of the matrix.

*Proof:* Note that $\mathcal{B}(x) = \|Ax - b\|_1$, which is convex and regular, by using the chain rule [2] [25], we have

$$\frac{d}{dt}\mathcal{B}(x) = \zeta^T \frac{dx(t)}{dt}, \forall \zeta \in \partial\mathcal{B}(x(t)) = A^T K[g(Ax - b)].$$

where $K(\cdot)$ denotes the closure of the convex hull; i.e., the Filippov set-valued map [16], and $\dot{x}(t)$ is given by (3.1).

From the definition of $P$, we know that $A(I-P) = A - AA^T(AA^T)^{-1}A = 0$. Thus for any $x_0 \in \mathbb{R}^n$, when $x(t) \in \mathbb{R}^n \backslash \mathcal{S}$, we have

$$\frac{d}{dt}\mathcal{B}(x) = -\frac{1}{\epsilon}\|A^T\eta\|^2, \forall \eta \in K[g(Ax-b)].$$

For any $x \in \mathbb{R}^n \backslash \mathcal{S}$, $Ax - b \neq 0$. So at least one of the components of $\eta$ is 1 or $-1$. On one hand, since $A$ has full row-rank, $AA^T$ is invertible. It follows that

$$\|(AA^T)^{-1}AA^T\eta\| = \|\eta\| \geq 1.$$

Since $AA^T$ is positive definite, we have

$$\|A^T\eta\|^2 = \eta^T AA^T\eta \geq \lambda_{\min}(AA^T)\|\eta\|^2 \geq \lambda_{\min}(AA^T).$$

Thus

$$\frac{d\mathcal{B}(x(t))}{dt} \leq -\frac{1}{\epsilon}\lambda_{\min}(AA^T) < 0. \tag{3.4}$$

Integrating the both sides of (3.4) from $t_0 = 0$ to $t$, we have

$$\|Ax(t) - b\|_1 \leq \|Ax_0 - b\|_1 - \frac{1}{\epsilon}\lambda_{\min}(AA^T)t.$$

Thus, $Ax(t) - b = 0$ as $t = \epsilon\|A(x_0) - b\|_1/\lambda_{\min}(AA^T)$. That is, the state vector of neural network (3.1) reaches $\mathcal{S}$ in finite time and an upper bound of the hit time is $t_{\mathcal{S}} = \epsilon\|A(x_0) - b\|_1/\lambda_{\min}(AA^T)$.

Next, we prove that, when $t \geq t_{\mathcal{S}}$, the state vector of neural network (3.1) remains inside $\mathcal{S}$ thereafter. If not so, assume that the trajectory leaves $\mathcal{S}$ at time $t_1$ and stays outside of $\mathcal{S}$ for almost all $t \in (t_1, t_2)$, where $t_1 < t_2$. Then, $\|A(x(t_1)) - b\|_1 = 0$, and from above analysis, $\|A(x(t)) - b\|_1 < 0$ for almost all $t \in (t_1, t_2)$, which is a contradiction. That is, the state vector of neural neural (3.1) reaches the equality feasible region $\mathcal{S}$ by $t_{\mathcal{S}}$ at the latest and stays there thereafter. □

*Theorem 2:* Let $f(x)$ be pseudoconvex on $\mathcal{S}$. The state vector of the neural network (3.1) is stable in the sense of Lyapunov and globally convergent to the equilibrium point set for any $x_0 \in \mathbb{R}^n$. In particular, assume that $f(x)$ is strictly pseudoconvex on $\mathcal{S}$, then the neural network (3.1) is globally asymptotically stable.

*Proof:* Denote $\bar{x}$ as an equilibrium point of system (3.1); i.e., $0 \in A^T K[g(A\bar{x} - b)] + (I - P)\nabla f(\bar{x})$. Since by Theorem 1, any trajectory $x(t)$ will convergent to the feasible region $\mathcal{S}$ in finite time $t_{\mathcal{S}} = \epsilon\|A(x_0) - b\|_1/\lambda_{\min}(AA^T)$, and will remain in $\mathcal{S}$ forever; i.e., $\forall t \geq t_{\mathcal{S}}$, $x(t) \in \mathcal{S}$. As a result, it suffices to show the stability of the system with $x(t) \in \mathcal{S}$.

Consider the following Lyapunov function:

$$V_1(x) = f(x) - f(\bar{x}) + \frac{1}{2}\|x - \bar{x}\|^2. \tag{3.5}$$

Clearly, $\forall x \in S$ and $x \neq \bar{x}$, $V_1(x) > 0$, $0 = \eta \in K[g(Ax - b)]$, and $Ax - b = 0$.. so $A(\bar{x} - x) = 0$, and $(x - \bar{x})^T P = (x - \bar{x})^T A^T (AA^T)^{-1} A = [A(x - \bar{x})]^T (AA^T)^{-1} A = 0$, as well as $P^T(x - \bar{x}) = 0$. Since $x = P^T x + (I - P)^T x$ and $(I - P)\nabla f(\bar{x}) = 0$, as a result:

$$\begin{aligned}\nabla f(\bar{x})^T(x-\bar{x}) &= \nabla f(\bar{x})^T[P^T(x-\bar{x})+(I-P)^T(x-\bar{x})] \\ &= [(I-P)\nabla f(\bar{x})]^T(x-\bar{x})=0. \end{aligned} \tag{3.6}$$

By the pseudoconvexity of $f(x)$ on $\mathcal{S}$, we know that $\nabla f(x)$ is a pseudomonotone mapping on $\mathcal{S}$ [37]. Thus from (3.6), we know that for any $x \in \mathcal{S}$ and $x \neq \bar{x}$, $\nabla f(x)^T(x-\bar{x}) \geq 0$.

$$\begin{aligned}\frac{dV_1(x)}{dt} &= \nabla V_1(x)^T \cdot \frac{dx}{dt} \\ &= -(\nabla f(x)+x-\bar{x})^T\left((I-P)\nabla f(x)+A^T\eta\right) \\ &= -\nabla f(x)^T(I-P)\nabla f(x)-(x-\bar{x})^T\nabla f(x)+ \\ &\quad (x-\bar{x})^T P\nabla f(x) \\ &\leq -\|(I-P)\nabla f(x)\|^2 \leq 0. \end{aligned} \tag{3.7}$$

Furthermore, $d(V_1(x))/dt=0$ if and only if $(I-P)\nabla f(x)=0$, since $f(\cdot)$ is locally Lipschitz continuous, from LaSalle invariant set Theorem [42][39][16], $x(t) \to \bar{\Omega} = \{x|d(V_1(x))/dt=0\}$.

Now we show that $\{x|dV_1(x)/dt=0\}$ is the same set as $\{x|dx/dt=0\}$.

From equation (3.7), it's obvious that $d(V_1(x))/dt=0 \Rightarrow (I-P)\nabla f(x)=0$. Since the assumption that $x(t) \in \mathcal{S}$ has been made at the beginning of the proof, we have $0 \in K[g(Ax-b)]$. Thus $d(V_1(x))/dt=0 \Rightarrow (I-P)\nabla f(x)=0 \Rightarrow dx/dt=0$. For any $x$ that satisfies $dx/dt=0$, it is clear that $d(V_1(x))/dt = d(V_1(x))/dx \cdot dx/dt=0$. As a result, $x(t) \to \bar{\Omega} = \{x|d(V_1(x))/dt=$

$0\} = \{x|dx/dt = 0\}$, thus the neural network is stable in the sense of Lyapunov and globally convergent to the equilibrium points set.

If $f(x)$ is strictly pseudoconvex on $\mathcal{S}$, $\nabla f(x)$ is a strictly pseudomonotone mapping on $\mathcal{S}$ [37], then $\forall x \in \mathcal{S}$ and $x \neq \bar{x}$, $(x-\bar{x})^T \nabla f(x) > 0$. From (3.7), we know that $\forall x \in S$ and $x \neq \bar{x}$, $dV_1(x)/dt < 0$, and $dV_1(x)/dt = 0$ if and only if $x = \bar{x}$. Also $f(x) > f(\bar{x})$ can be derived from $\nabla f(x)^T(x-\bar{x}) = 0$ for any $x \in \mathcal{S}$ since $f(x)$ is strictly pseudoconvex. As a result, $\bar{x}$ is a unique equilibrium point. Thus if $f(x)$ is strictly pseudoconvex on $\mathcal{S}$, the neural network (3.1) is globally asymptotically stable.
□

*Theorem 3:* Let $f(x)$ be pseudoconvex on $\mathcal{S}$. The state vector of the neural network (3.1) is globally convergent to optimal solution set of the problem (1.3) for any $x_0 \in \mathbb{R}^n$. In addition, when $f(x)$ is strictly pseudoconvex on $\mathcal{S}$, the neural network (3.1) is globally convergent to the unique optimal solution $x^*$ of problem (1.3).

*Proof:* From Theorem 2, we know that the system (3.1) is stable in the sense of Lyapunov, and globally convergent to equilibrium point set $\bar{\Omega} = \{x|dx/dt = 0\}$. As $0 \in -(I-P)\nabla f(\bar{x}) - A^T K[g(A\bar{x}-b)]$ holds for any $\bar{x}$ and $P = A^T(AA^T)^{-1}A$, we have

$$0 \in \nabla f(\bar{x}) - A^T(AA^T)^{-1}A\nabla f(\bar{x}) + A^T K[g(A\bar{x}-b)].$$

Let $y \in (AA^T)^{-1}A\nabla f(\bar{x}) - K[g(A\bar{x}-b)]$. Then $\nabla f(\bar{x}) -$

$A^T y = 0$, which means $\bar{x}$ satisfies KKT condition of problem (1.3). Considering Lemma 1, we can conclude that any equilibrium point $\bar{x}$ of system (3.1) is an optimal solution $x^*$ of problem (1.3). Thus the neural network (3.1) is globally convergent to the optimal solution set of problem (1.3).

For strictly pseudoconvex optimization, since the solution $x^*$ is unique, it is obvious that the neural network (3.1) is convergent to the optimal solution of problem (1.3). □

*Theorem 4:* Let $\nabla f(x)$ be strongly pseudomonotone on $\mathcal{S}$. For any initial point $x_0 \in \mathbb{R}^n$, the state vector of the neural network (3.1) is exponentially convergent to the optimal solution $x^*$ of the problem (1.3) after $t \geq t_{\mathcal{S}}$.

*Proof:* By the strongly pseudomonotone of $\nabla f(x)$ on $\mathcal{S}$, since in (3.6) we have $\nabla f(\bar{x})^T (x - \bar{x}) = 0$, $\exists \gamma > 0$, such that $\forall t > t_{\mathcal{S}}$,

$$\nabla f(x)^T (x - \bar{x}) \geq \gamma \|x - \bar{x}\|^2,$$

where $\bar{x}$ is an equilibrium point that satisfies $0 \in -A^T K[g(A\bar{x} - b)] - (I - P)\nabla f(\bar{x})$.

Consider the following Lyapunov function:

$$V_2(x) = \frac{1}{2}\|x - \bar{x}\|^2. \tag{3.8}$$

We have

$$\frac{dV_2(x)}{dt} \leq -(x - \bar{x})^T \nabla f(x) \leq -\gamma \|x - \bar{x}\|^2 = -2\gamma V_2(x).$$

As a result, $\forall t > t_{\mathcal{S}}$,

$$V_2(x(t)) \leq V_2(x(t_S)) \exp\left(-2\gamma(t - t_S)\right).$$

From Lemma 1 and the proof in Theorem 3, as $\nabla f(x)$ is strongly pseudomonotone on $\mathcal{S}$, $f(x)$ is pseudoconvex on $\mathcal{S}$. Thus we know that $\bar{x}$ satisfies KKT condition and is the optimal solution $x^*$. Because $V_2(x) = 0$ if and only if $x = \bar{x}$, the neural network (3.1) is exponentially convergent to the optimal solution $x^*$ of the problem (1.3) after $t \geq t_S$. $\square$

Note that strictly convex quadratic function is also strongly pseudoconvex (since the matrix in a strictly convex quadratic function has a minimum positive eigenvalue). Thus the state vector of the neural network (3.1) is also exponentially convergent to the optimal solution for strictly convex quadratic optimization subject to linear equality constraints.

---

□ **End of chapter.**

# Chapter 4

# Numerical Examples

**Chapter Outline**

To demonstrate the performance of the proposed one-layer neural network in solving pseudoconvex optimization problems with linear equality constraints, several illustrative examples are given in this section.

Many functions in nature are pseudoconvex, such as Butterworth filter functions, fractional functions, and some density functions in probability theory. Among them, the Gaussian function as shown in Figure 4.1 is chosen in Example 4.1, and quadratic fractional function is chosen as the objective function for Example 4.2. Example 4.3 considers a strictly convex objective funtion, which is also strongly pseudoconvex.

In the following simulations, the differential equation defined by (3.1) is solved using MATLAB r2008a ode45 algorithm on a 2.4GHZ Intel CoreTM2 Qrad PC running Windows Vista with 2.0GB main memory.

## 4.1 Gaussian Optimization

*Example 4.1:* Consider the following pseudoconvex optimization problem with linear equality constraints:

$$\begin{aligned} &\text{minimize} && -\exp(-\sum_{i=1}^{2} x_i^2/\sigma_i^2), \\ &\text{subject to} && Ax = b, \end{aligned} \tag{4.1}$$

where $x \in \mathbb{R}^2$, $\sigma = (1,1)^T$, the elements of $A = [0.787,\ 0.586]$ and $b = 0.823$ are randomly drawn from the uniform distribution over $(0,1)$. Obviously, the objective function is locally Lipschitz continuous and strictly pseudoconvex on $\mathbb{R}^2$.

Since the conditions in Theorems 1, 2 and 3 hold, the one-layer recurrent neural network (3.1) is globally asymptotically stable and capable of solving this optimization problem. Figure 4.2 is the state phase plot of the neural network (3.1) from 20 random initial points converging to the feasible set $\mathcal{S} = \{x | Ax - b = 0\}$ in finite time, and then converging to $x^*$. It is also obvious that the state variables stay in the feasible region $\mathcal{S}$ once reaching it. Figure 4.3 shows the transient states of the neural network (3.1) with $\epsilon = 10^{-6}$ in Example 4.1, where 20 random initial points are generated from the uniform distribution over $(-1,1)$.

The projection neural network [82] and the two-layer recurrent neural network [81] are also used for solving the same problem (4.1).

The global convergence of the Lagrangian network for convex

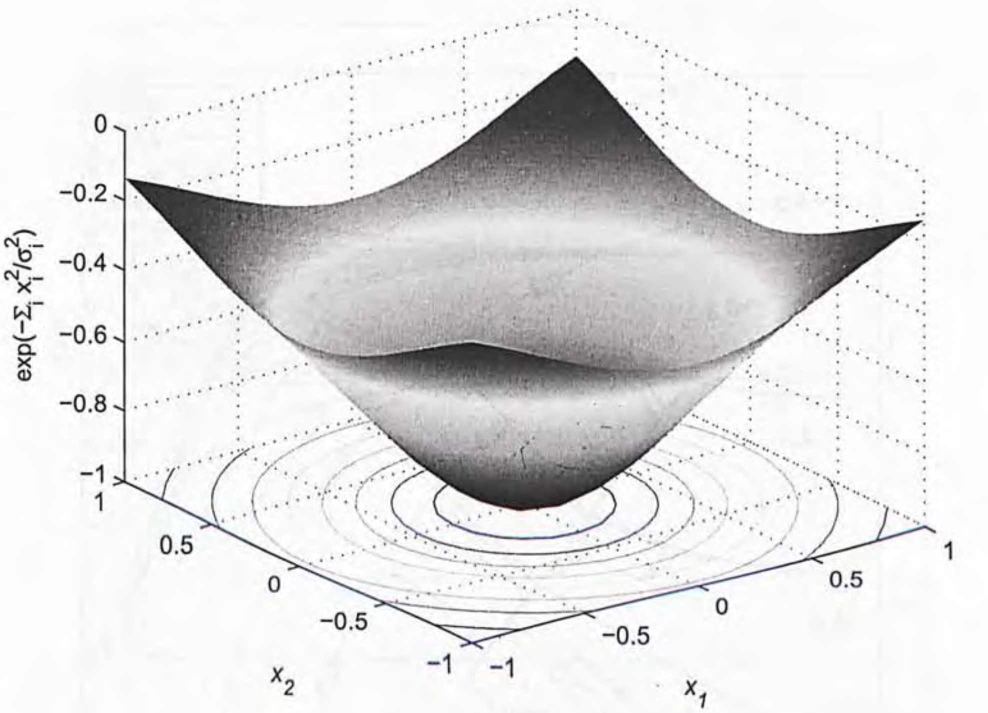

Figure 4.1: Isometric of of inverted $2D$ non-normalized Gausssian function with $\sigma = [1, 1]^T$.

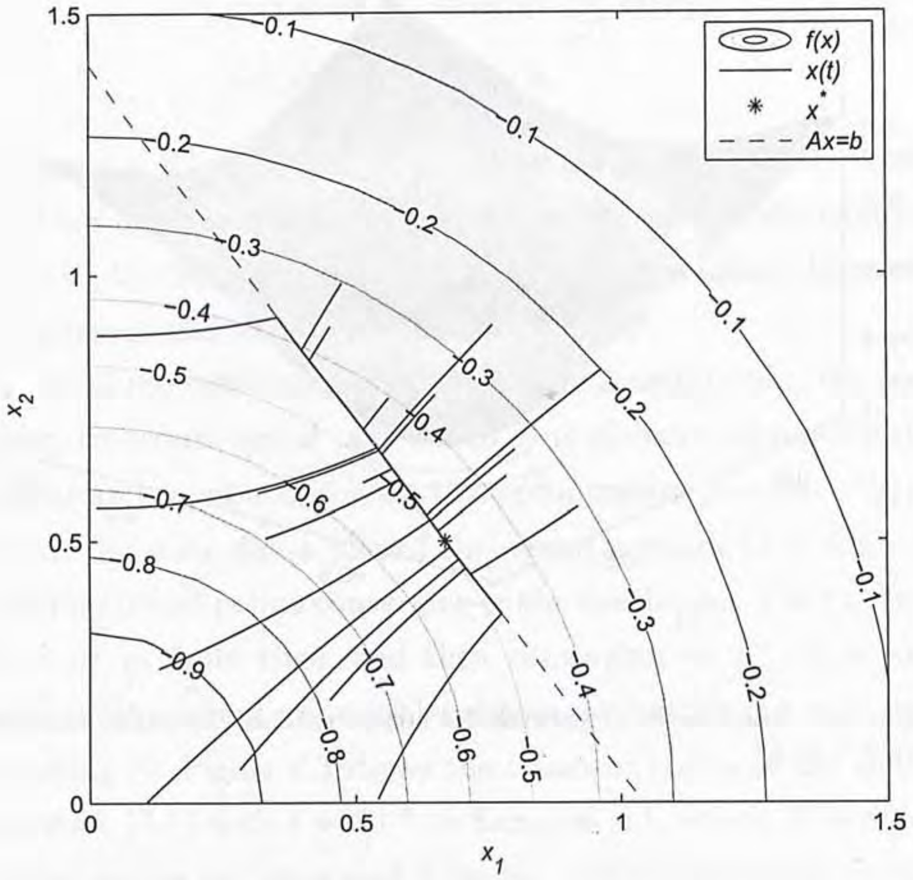

Figure 4.2: Transient behaviors of the neural network (3.1) with 20 random initial points in Example 4.1.

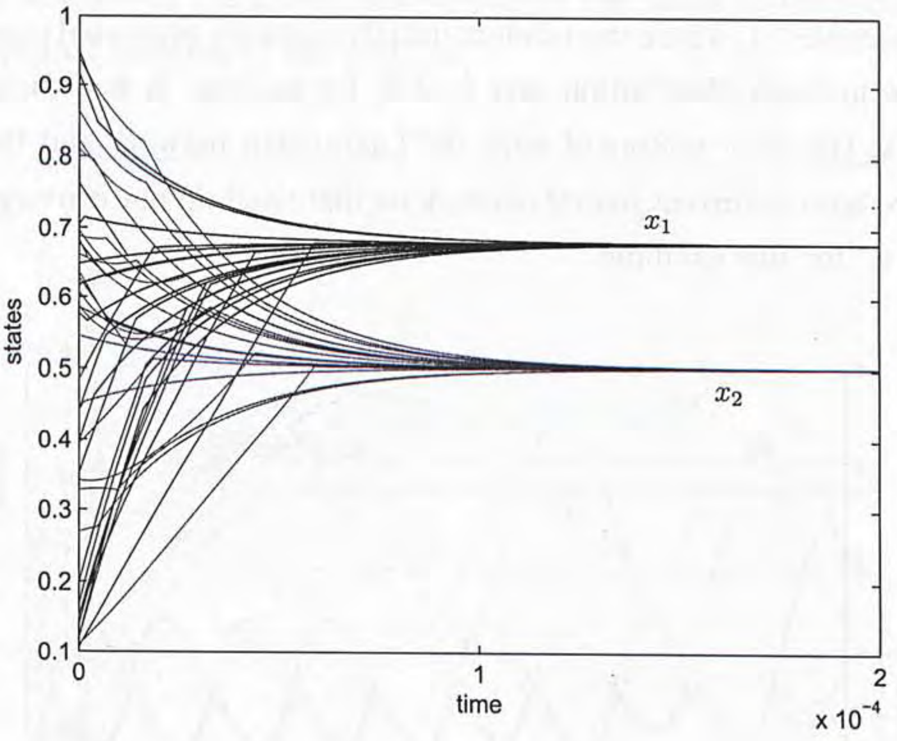

Figure 4.3: Transient states of the neural network (3.1) in Example 4.1.

optimization was studied in [75]. However, global convergence is not guaranteed for pseudoconvex problems. Figure 4.4 shows the transient states of both the Lagrangian network (in dashed line) and the two-layer recurrent neural network (in continues line) in Example 4.1, where two random initial points are generated from the uniform distribution over $(-2, 2)$ for models. It is obvious that the state vectors of both the Lagrangian network and the two layer recurrent neural network oscillate and do not converge to $x^*$ for this example.

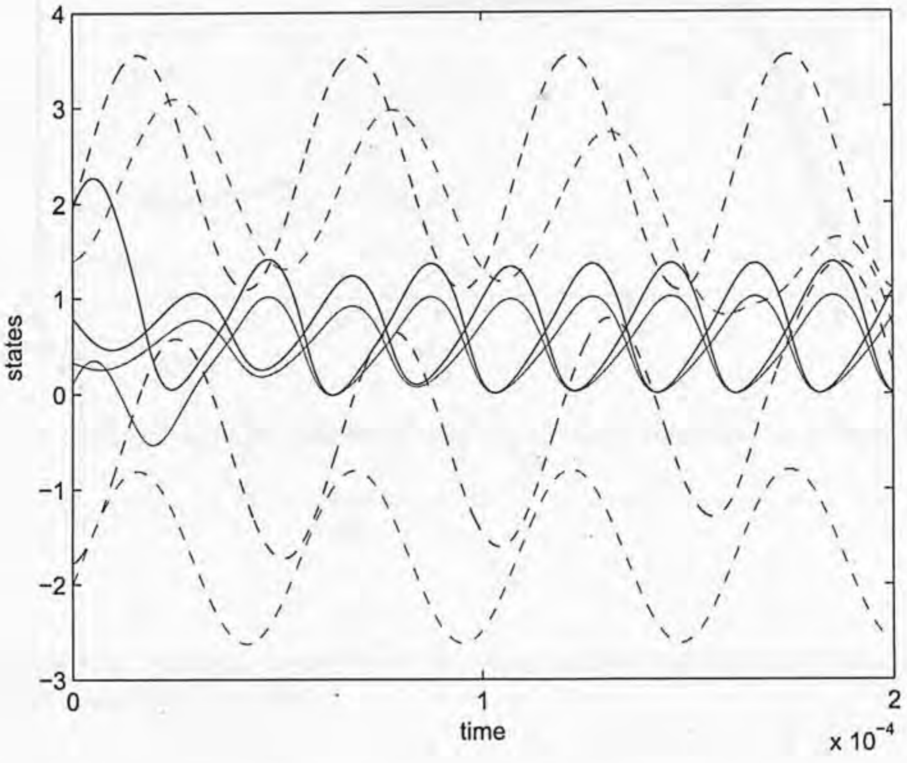

Figure 4.4: Transient states of both the Lagrangian network (in dashed line) and the two-layer recurrent neural network (in continues line) in Example 4.1.

Furthermore, consider problem (4.1) in a higher-dimension case with $n = 5$, where $\sigma = [1, 1/2, 1/4, 1/2, 1]^T$, $A \in \mathbb{R}^{3\times5}$ and $b \in \mathbb{R}^3$ are drawn from the uniform distribution over $(0, 1)^5$. Figure 4.5 depicts the transient states of the one-layer recurrent neural network (3.1) with $\epsilon = 10^{-6}$, where five random initial points are generated from the uniform distribution over $(-1, 1)^5$. It also shows the global convergence of the states to the unique optimal solution of the problem.

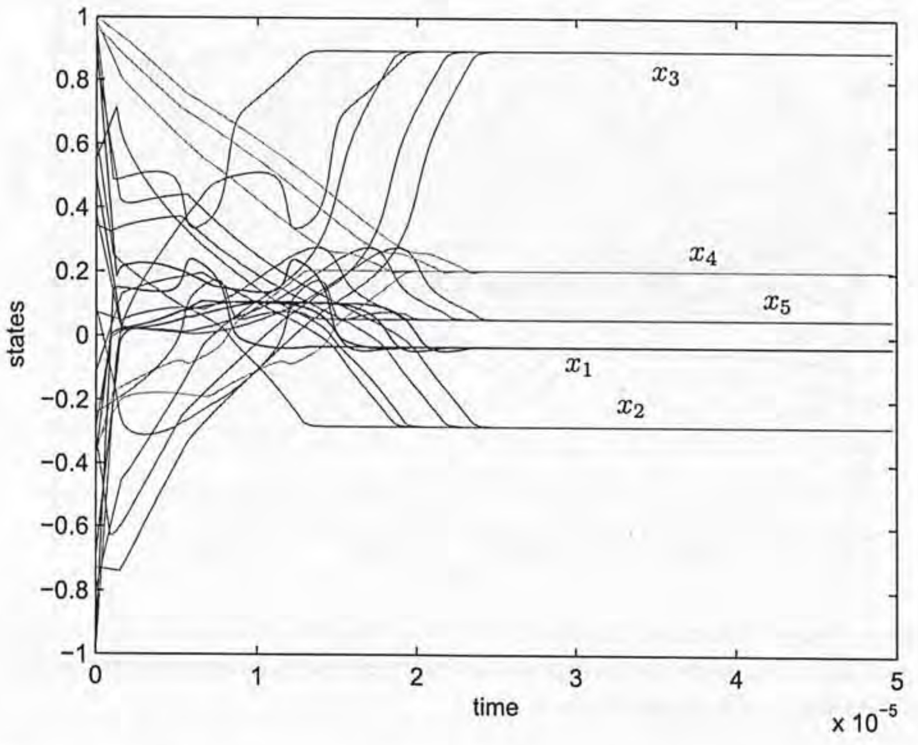

Figure 4.5: Transient states of the one-layer recurrent neural network (3.1) from 5 random initial points in Example 4.1($n = 5$).

Again, the Lagrangian network [85] is applied to solve the problem (4.1). In Figure 4.6, 5 random initial points are generated from the uniform distribution over $(-1, 1)^5$, and the states are drawn in different colors. It is obvious that the Lagrangian network does not converge.

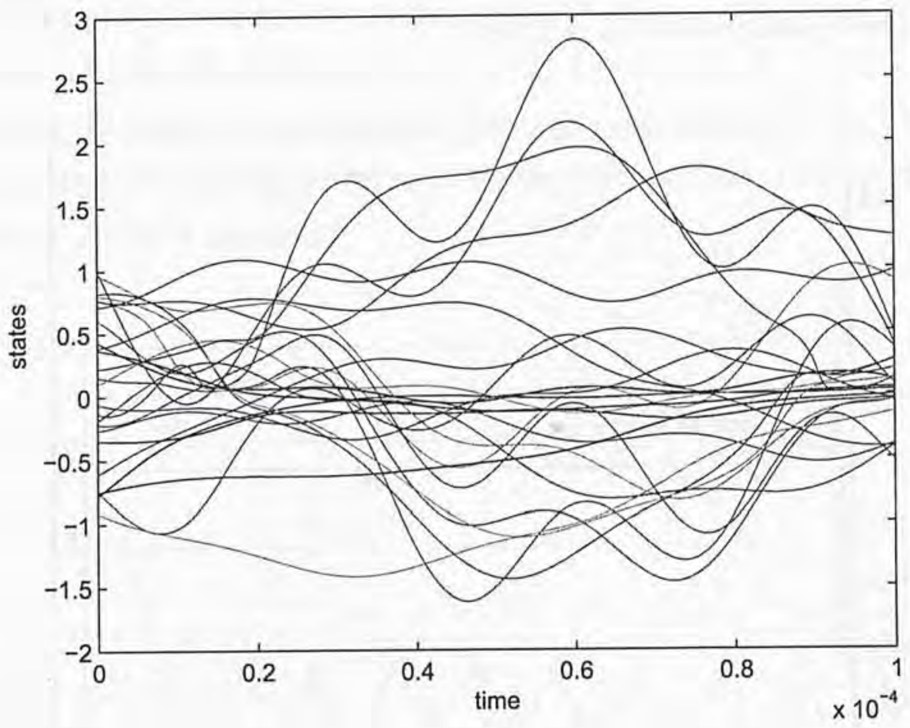

Figure 4.6: Transient behaviors of the Lagrangian network with 5 random initial points in Example 4.1($n = 5$).

Figure 4.7 shows the transient behaviors of the two-layer recurrent neural network [81] in Example 4.1, where five random initial points are generated from the uniform distribution over $(-1, 1)^5$. It shows that the state vectors of the neural network oscillate or even diverge for the example.

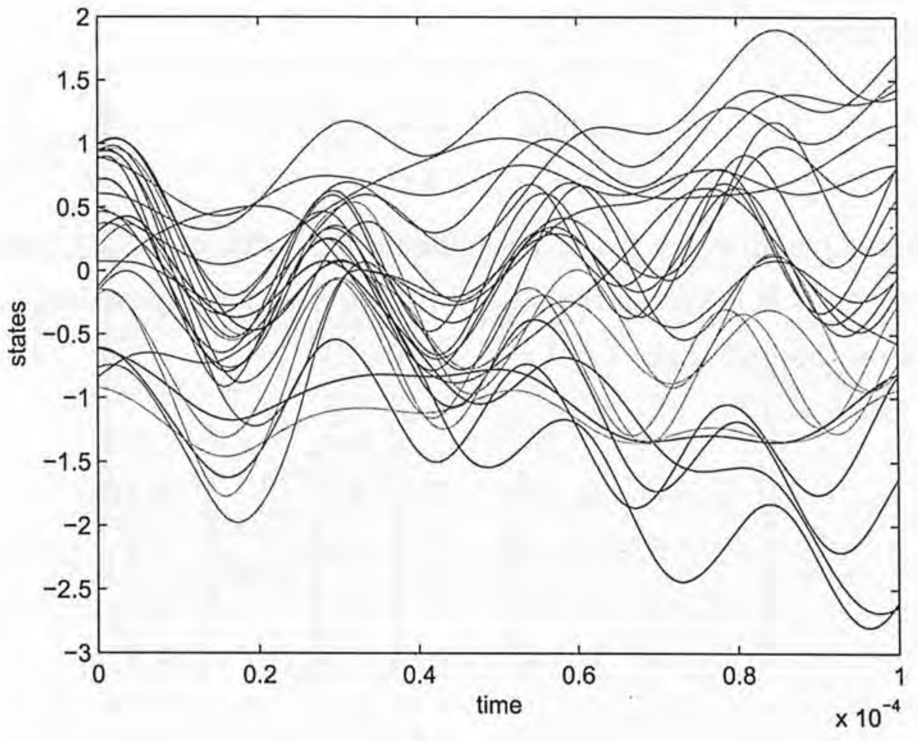

Figure 4.7: Transient behaviors of the two-layer recurrent neural network (2.11) with 5 random initial points in Example 4.1($n = 5$).

## 4.2 Quadratic Fractional Programming

*Example 4.2:* One of the important classes of pseudoconvex optimization problems is the quadratic fractional programming problem:

$$\begin{array}{ll} \text{minimize} & \frac{x^TQx+a^Tx+a_0}{c^Tx+c_0}, \\ \text{subject to} & Ax = b, \end{array} \tag{4.2}$$

where $Q$ is an $n \times n$ positive semidefinite matrix, $a, c \in \mathbb{R}^n$, and $a_0, c_0 \in \mathbb{R}$. It is known that the objective function is pseudoconvex on the half space $\{x|c^Tx + c_0 > 0\}$ [19].

Let $n = 4$,

$$Q = \begin{pmatrix} 5 & -1 & 2 & 0 \\ -1 & 5 & -1 & 3 \\ 2 & -1 & 3 & 0 \\ 0 & 3 & 0 & 5 \end{pmatrix}, a = \begin{pmatrix} 1 \\ -2 \\ -2 \\ 1 \end{pmatrix}, c = \begin{pmatrix} 2 \\ 1 \\ -1 \\ 0 \end{pmatrix},$$

$$A = \begin{pmatrix} 2 & 1 & -1 & 0 \\ 1 & 0 & 2 & -2 \end{pmatrix}, b = \begin{pmatrix} 4 \\ 5 \end{pmatrix}, a_0 = -2, c_0 = 5.$$

As $Q$ is symmetric and positive definite in $\mathbb{R}^4$, the objective function is pseudoconvex on the feasible region $\{x|Ax = b\}$. Figure 4.8 depicts the transient states of the one-layer recurrent neural network (3.1) with $\epsilon = 10^{-6}$, where 10 random initial points are generated from the uniform distribution over $(0, 5)^4$. It shows the global convergence to the unique optimal solution of the problem.

Figure 4.9 shows the transient states of both the Lagrangian

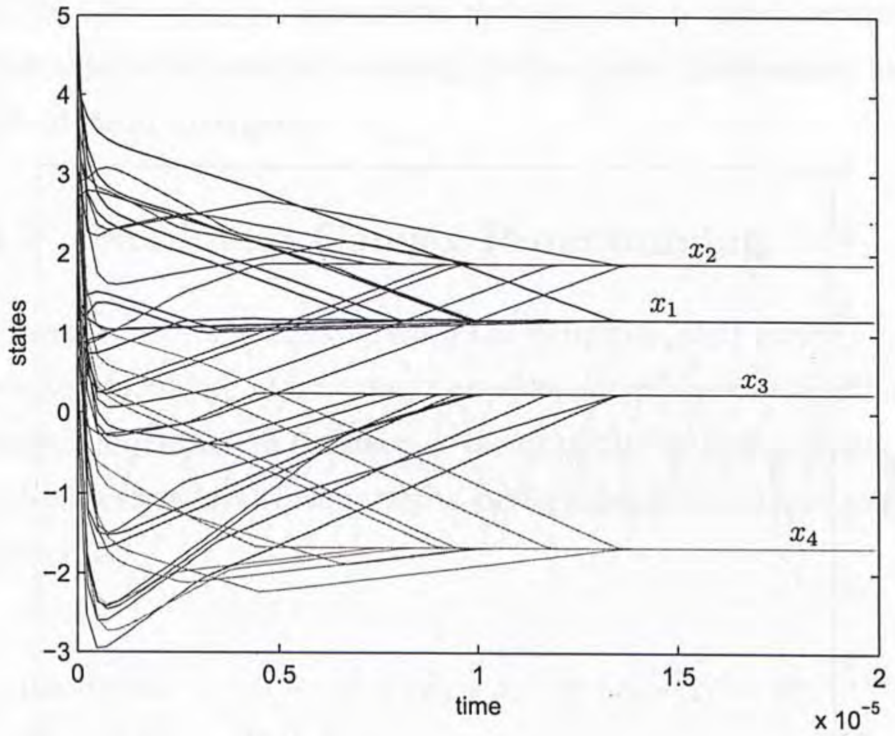

Figure 4.8: Transient behaviors of the one-layer recurrent neural network (3.1) with 10 random initial points in Example 4.2.

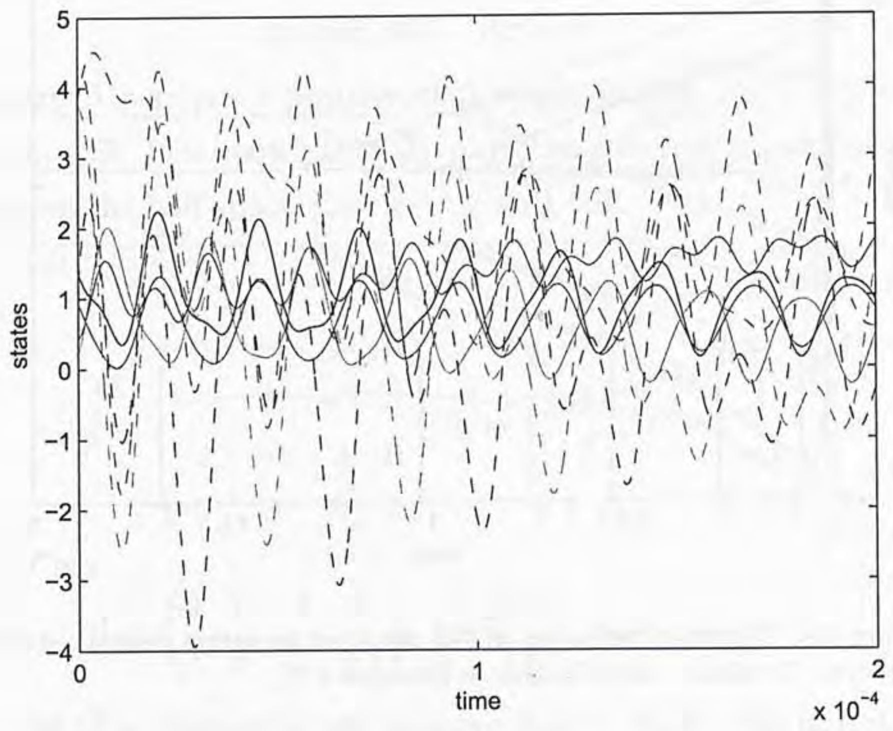

Figure 4.9: Transient states of both the Lagrangian network (dashed line) and the two-layer recurrent neural network (continues line) in Example 4.2.

network (in dashed line) and the two-layer recurrent neural network (in continues line) in Example 4.2, with a random initial point generated from the uniform distribution over $(0, 5)$.

These two examples have shown that for strictly pseudoconvex optimization problem, the one-layer recurrent neural network is globally asymptotically stable at $x^*$. While other networks such as the Lagrangian network and a novel recurrent neural network may not converge to the global minimum or even vibrate and divergence.

## 4.3 Nonlinear Convex Programming

*Example 4.3:* It is obvious from the definition that strictly convex functions are also strongly pseudoconvex ones. A nonlinear convex function will be taken as the objective in this example, in order to show and compare the performances of related neural networks.

$$\begin{array}{ll} \text{minimize} & (x_1-4)^4+(x_2+x_3)^6+(x_4+2)^4+e^{x_1+x_2+x_3+x_4}, \\ \text{subject to} & Ax=b, \end{array} \tag{4.3}$$

where $A = \begin{pmatrix} 2 & -3 & 1 & 0 \\ 0 & 1 & 2 & -1 \end{pmatrix}$, and $b = [1, -3]^T$.

Figure 4.10 depicts the transient states of the one-layer recurrent neural network (3.1) with $\epsilon = 10^{-6}$, where 5 random initial points are generated from the uniform distribution over $(-1, 1)^4$. It shows the global convergence to the unique optimal solution of the problem.

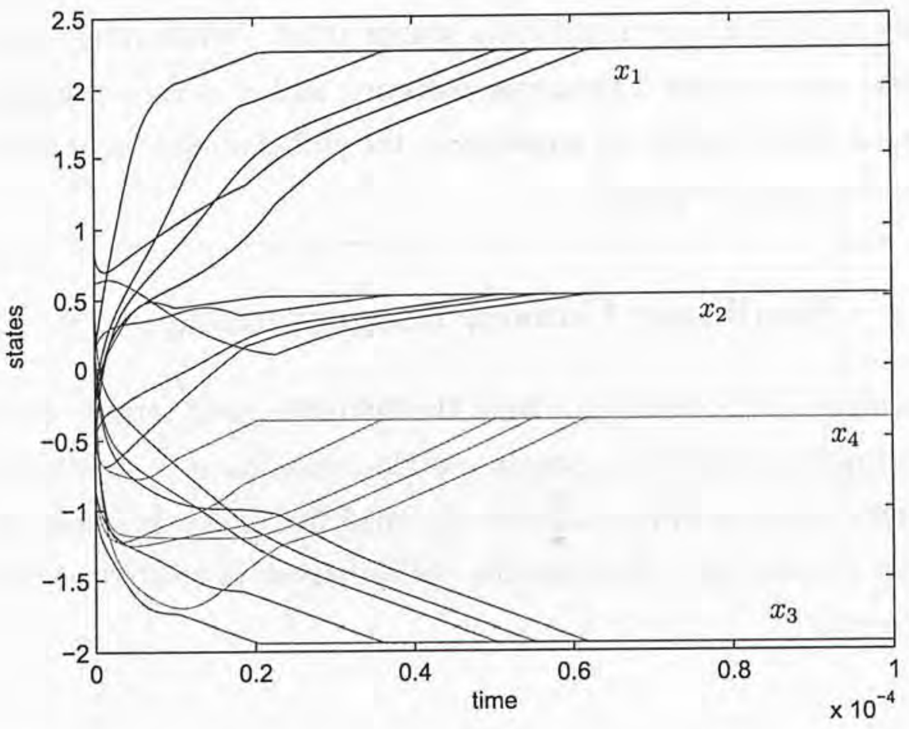

Figure 4.10: Transient states of the one-layer recurrent neural network (3.1) with 5 random initial points in Example 4.3.

Using the same parameters, the problem is solved with two other neural networks. Figure 4.11 shows the transient states of both the Lagrangian network (in dashed line) and the two-layer recurrent neural network (in continues line) in Example 4.3. These networks are designed for constrained convex optimization, and the convergence of their states to the optimal solution has been proved theoretically, and Figure 4.11 shows exactly the transient behaviors. However, when comparing Figures 4.10 and 4.11, we can see that the proposed neural network converges faster than the other existing method.

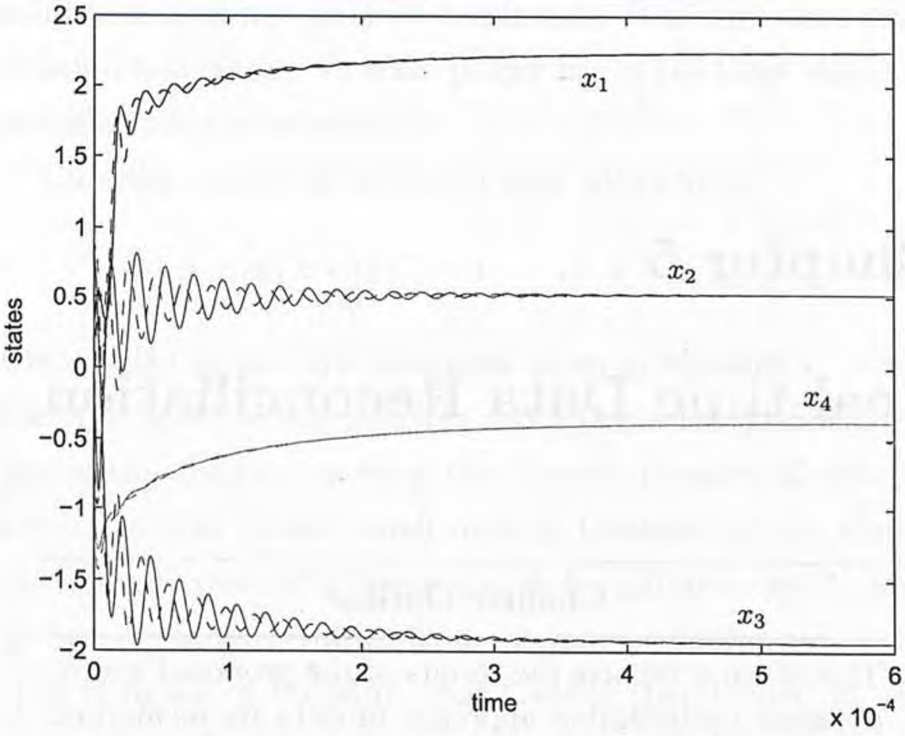

Figure 4.11: Transient states of both the Lagrangian network (dashed line) and the two-layer recurrent neural network (continues line) in Example 4.3.

□ **End of chapter.**

# Chapter 5

# Real-time Data Reconciliation

**Chapter Outline**

This chapter reports the results of the proposed neurodynamic optimization approach in data reconciliation. It is shown that data reconciliation problem can be formulated as pseudoconvex optimization one with linear equality constraints. Based on a performance index, simulation results on three industrial examples are shown and compared.

## 5.1 Introduction

Measured process data usually contain several types of errors. It is important to understand what is wrong with the values obtained by the measurement and how they can be adjusted [64]. Data reconciliation is a means to adjust process data measurements by minimizing the error and ensuring constraint satisfaction, which is a way to improve the quality of distributed control systems. A good estimation is usually defined as the optimal solution to a constrained maximum likelihood objective function subject to data flow balance constraints. Real-time data reconciliation is necessary to make proper use of the large amount of available process information.

Consider a series of measured data with errors:

$$y_i(k) = z_i(k) + e_i(k), i = 1, \cdots, n, k = 1, 2, \cdots \tag{5.1}$$

where $y_i(k)$ is the $k$th measured value of element $i$, $z_i(k)$ is the true value of the element, and $e_i(k)$ is the identically independent distribution error that usually consists of three different types of errors: small random Gaussian errors, Cauchy distributed systematic biases and drift, and gross errors which are usually large, resulting from instrument malfunction.

It is shown in literature that Cauchy (Lorentzian) function is the most effective generalized maximum likelihood objective function with higher data reconciliation performance [18]. Consider equation (5.1), data reconciliation can be stated in the following form for each given $k$:

$$\begin{array}{ll} \text{maximize} & \Pi_i\{\frac{1}{\pi\sigma_i(1+(y_i-x_i)^2/\sigma_i^2)}\}, \\ \text{subject to} & \sum\limits_{j=1}^{n} a_{ij}x_j = b_i, i = 1, \cdots, n, \end{array} \tag{5.2}$$

where $y_i$ is the measurement of variable $i$, $x_i$ is the reconciled estimate, $\sigma_i$ is a scaler statistical parameter of the error.

The proposed neurodynamic optimization method can be used for solving the data reconciliation problem in chemical processes.

The benefit of using neural networks for data reconciliation is that the proposed neural dynamic system can achieve the optimal solution in very short time, which makes real-time data reconciliation possible.

## 5.2 Theoretical Analysis and Performance Measurement

First, we need to show that Cauchy function (which is also the objective function of Problem (5.2)) $g(x) = \Pi_i 1/[\pi\sigma_i(1 + (y_i - x_i)^2/\sigma_i^2)]$ is strictly pseudoconcave on $\mathbb{R}^n$. It is shown in [37] that a differentiable function is strictly pseudoconcave if and only if its negative gradient is a strictly pseudomonotone mapping where the definition is given below.

From the definition of Cauchy function, we have

$$\frac{\partial g}{\partial x_i} = g(x)\frac{2(x_i - y_i)}{\sigma_i^2[1 + (y_i - x_i)^2/\sigma_i^2]}.$$

Since $g(x) \geq 0$ always holds, $\forall x, x' \in \mathbb{R}^n$, from $-\nabla g(x)^T(x' - x) \geq 0$, we have

$$\sum_{i=1}^{n} \frac{2(x_i - y_i)(x_i' - x_i)}{[1 + (y_i - x_i)^2/\sigma_i^2]} \geq 0,$$

which simply leads to $\nabla g(x')^T(x' - x) > 0$. Thus $g(x)$ is strictly pseudoconvex on $\mathbb{R}^n$, and problem (5.2) is a pseudoconvex optimization problem with linear equality constraints.

Total error reductions (TER) [66] is often used to evaluate the data validation performance.

$$TER = \max\{0, \frac{\sqrt{\sum_{i=1}^{n}((y_i - z_i)/\sigma_i)^2} - \sqrt{\sum_{i=1}^{n}((x_i^* - z_i)/\sigma_i)^2}}{\sqrt{\sum_{i=1}^{n}((y_i - z_i)/\sigma_i)^2}}\}. \tag{5.3}$$

The range of $TER$ is [0, 1] and it reaches its maximum when the optimal solution $x^*$ is exactly the same as the true value $z$.

## 5.3 Examples

In the following experiments, the measurement sets $y_i$ are generated for each variable by adding noise from Cauchy and normal distributions with equal probability to true value $z_i$. For the gross errors, outliers are created in 10 percent randomly selected measurements by adding or subtracting $10 - 100$ percent of the true values. The lower bounds on the measurement variables are set to 50 percent of the true values and the upper bounds to twice of the true values.

*Example 5.1:* Consider a chemical reactor with two entering and two leaving mass flows [61], The four variables are related by three linear mass balance equations, where

$$A = \begin{pmatrix} 0.1 & 0.6 & -0.2 & -0.7 \\ 0.8 & 0.1 & -0.2 & -0.1 \\ 0.1 & 0.3 & -0.6 & -0.2 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\sigma = \mathrm{diag}(0.00289, 0.0025, 0.00576, 0.04),$$

$$z = (0.1850, 4.7935, 1.2295, 3.880)^T.$$

Figures 5.1 and 5.2 show, respectively, the transient states of the neural network (3.1) and the performance index value $TER$ with five random initial states and the same errors. It shows the global convergence of the neurodynamic optimization approach.

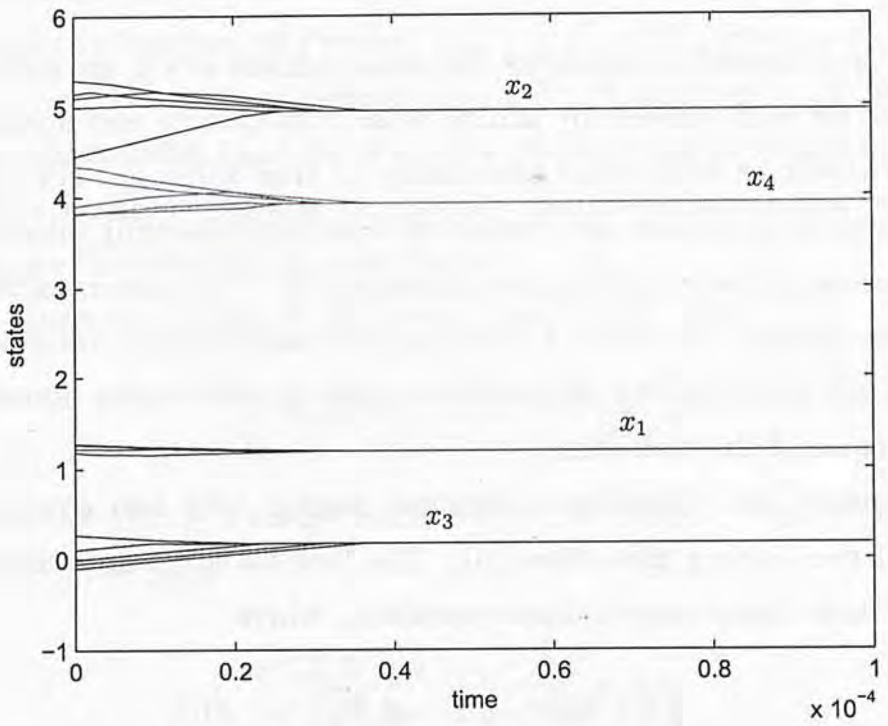

Figure 5.1: Transient states of the neural network (3.1) for data reconciliation with 5 random initial states in Example 5.1.

*Example 5.2:* Consider a recycle process network, where seven streams are identified with overall material balance as four linear

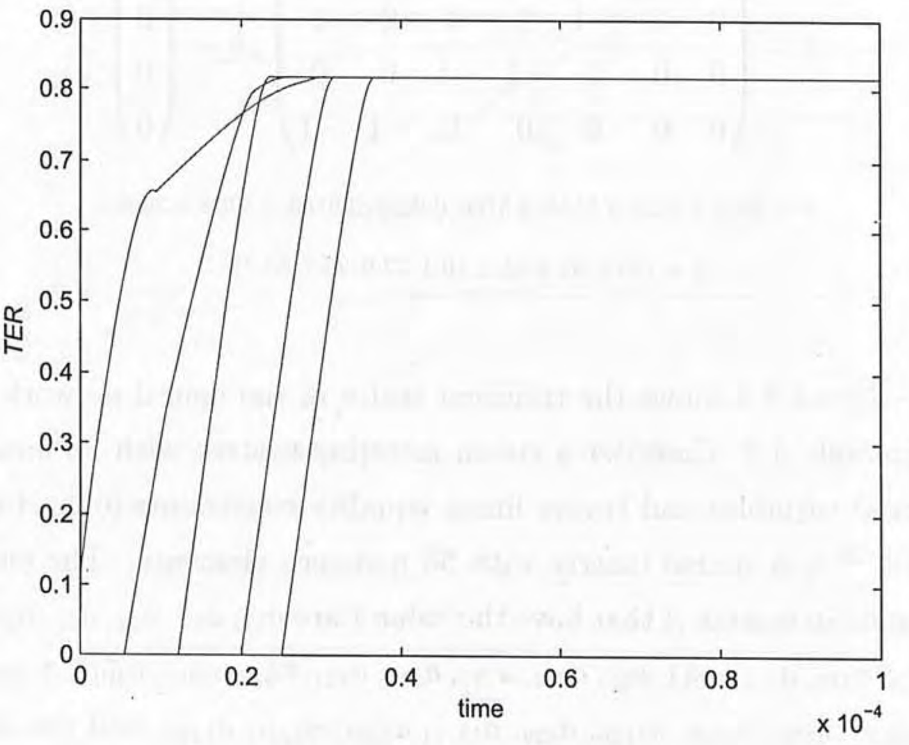

Figure 5.2: Transient behaviors of the performance index $TER$ in Example 5.1.

equality constraints [63], where

$$A = \begin{pmatrix} 1 & -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\sigma = \mathrm{diag}(1.5625, 4.5156, 4.5156, 0.0625, 3.5156, 0.3906, 0.3906),$$

$$z = (49.5, 81.5, 85.3, 10.1, 72.9, 25.7, 50.7)^T.$$

Figure 5.3 shows the transient states of the neural network.

*Example 5.3:* Consider a steam metering system with 28 measured variables and twelve linear equality constraints [67]. $A \in \mathbb{R}^{12\times 28}$ is a sparse matrix with 56 non-zero elements. The elements in matrix $A$ that have the value 1 are $a_{11}$, $a_{12}$, $a_{14}$, $a_{27}$, $a_{28}$, $a_{35}$, $a_{410}$, $a_{411}$, $a_{513}$, $a_{66}$, $a_{714}$, $a_{717}$, $a_{815}$, $a_{821}$, $a_{912}$, $a_{916}$, $a_{1019}$, $a_{1023}$, $a_{1027}$, $a_{1120}$, $a_{1126}$, $a_{1128}$, $a_{129}$, $a_{1211}$, $a_{1221}$, $a_{1224}$, $a_{1225}$, and the elements that have the value $-1$ are $a_{13}$, $a_{25}$, $a_{26}$, $a_{29}$, $a_{31}$, $a_{310}$, $a_{412}$, $a_{511}$, $a_{514}$, $a_{515}$, $a_{516}$, $a_{517}$, $a_{62}$, $a_{613}$, $a_{77}$, $a_{719}$, $a_{720}$, $a_{721}$, $a_{818}$, $a_{823}$, $a_{824}$, $a_{922}$, $a_{925}$, $a_{1026}$, $a_{118}$, $a_{124}$, $a_{1227}$, $a_{1228}$. $b = \mathbf{0} \in \mathbb{R}^{12}$, $z$ is given in Table II, and $\sigma = \mathrm{diag}(0.025z)$.

Figure 5.4 shows transient states of the neural network.

Figure 5.5 depicts the performance index $TER$ during the convergent processes in Examples 5.2 and 5.3. It shows that there are mainly two parts in this transient behaviors: at first, the state vector $x$ converges to a feasible point (that satisfies

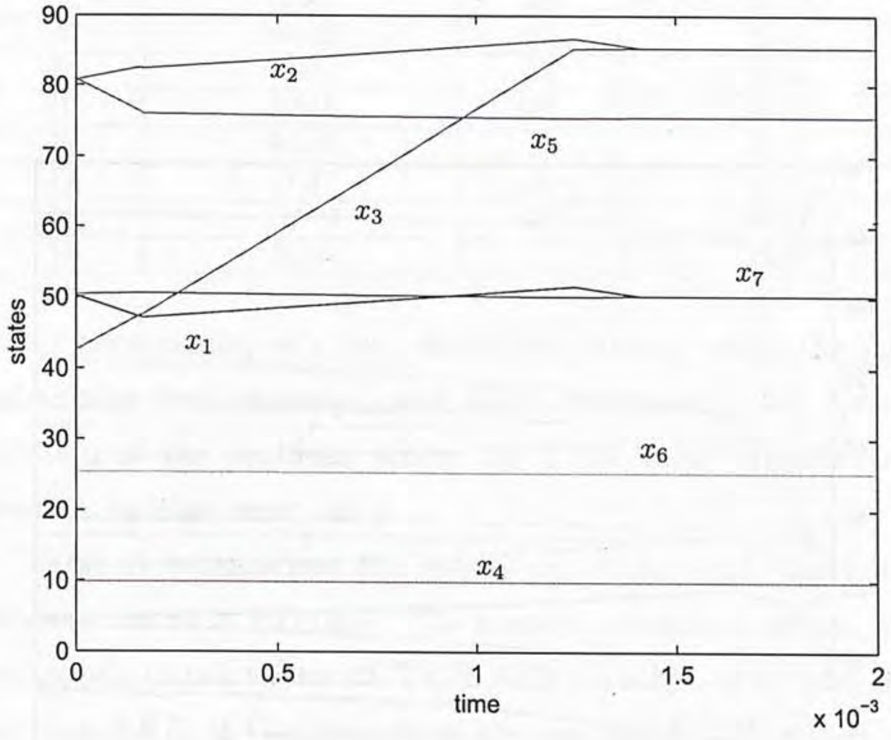

Figure 5.3: Transient states of the neural network (3.1) for data reconciliation in Example 5.2.

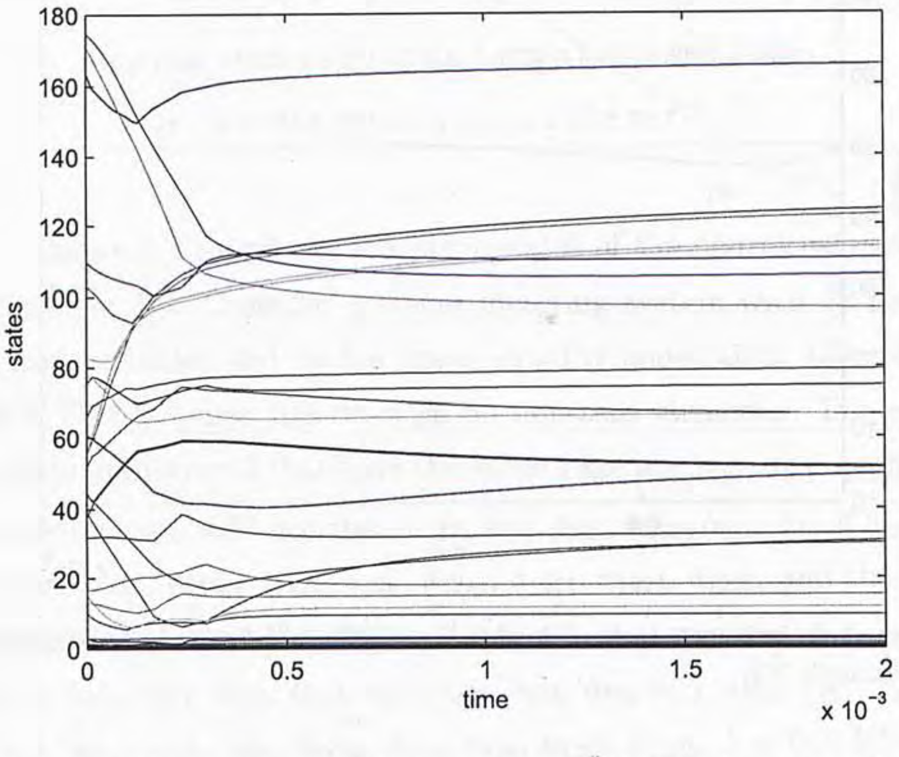

Figure 5.4: Transient states of the neural network (3.1) for data reconciliation in Example 5.3.

Table 5.1: True values of flow rates for the steam system in Example 5.3 [67].

| Steam No. | Flow Rate (1000Kg/h) | Steam No. | Flow Rate (1000Kg/h) |
|---|---|---|---|
| 1 | 0.86 | 15 | 60.00 |
| 2 | 1.00 | 16 | 23.64 |
| 3 | 111.82 | 17 | 32.73 |
| 4 | 109.95 | 18 | 16.23 |
| 5 | 53.27 | 19 | 7.95 |
| 6 | 112.27 | 20 | 10.50 |
| 7 | 2.32 | 21 | 87.27 |
| 8 | 164.05 | 22 | 5.45 |
| 9 | 0.86 | 23 | 2.59 |
| 10 | 52.41 | 24 | 46.64 |
| 11 | 14.86 | 25 | 85.45 |
| 12 | 67.27 | 26 | 81.32 |
| 13 | 111.27 | 27 | 70.77 |
| 14 | 91.86 | 28 | 72.23 |

linear constraints) in a very short time, during which the $TER$ value may even decrease, and then converges to the optimal solution of the problem, where the $TER$ value increases and reaches its maximum value.

Table II summarizes the results of Monte Carlo tests with random errors of 100 runs. The average, maximum (max), and minimum (min) values of $TER$ with Cauchy errors and also average $TER$ of Gaussian ones are compared. Obviously, the results with Cauchy errors are better than those with Gaussian ones.

□ **End of chapter.**

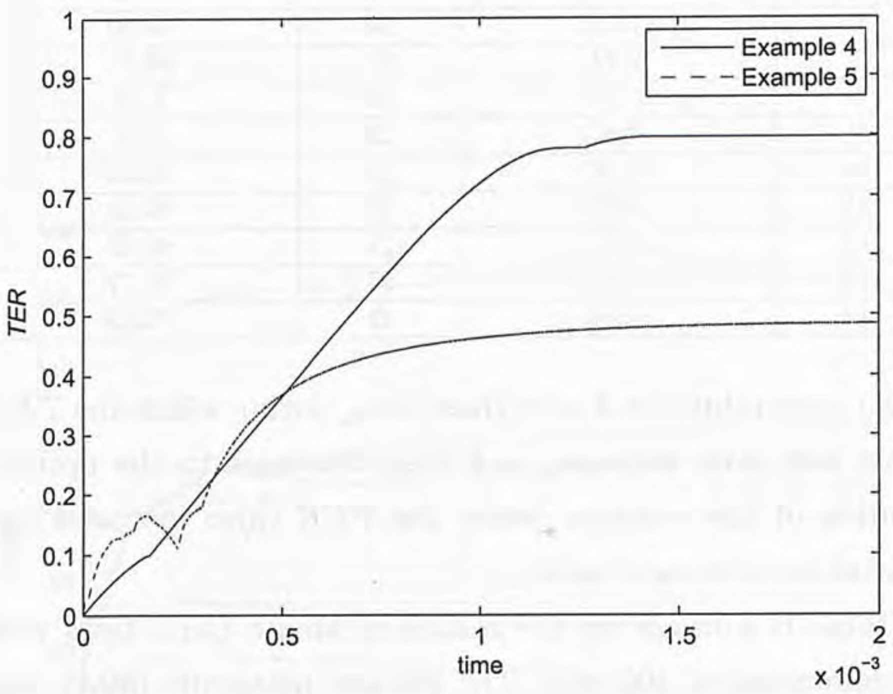

Figure 5.5: Transient behaviors of the performance index $TER$ in Examples 5.2 and 5.3.

Table 5.2: Performance of Monte Carlo tests in terms of $TER$ in Examples 5.1-5.3.

| Example | Gaussian | Cauchy | | |
|---|---|---|---|---|
| | average | average | max | min |
| 5.1 | 0.751 [18] | 0.757 | 0.992 | 0.424 |
| 5.2 | 0.764 [21] | 0.789 | 0.898 | 0.260 |
| 5.3 | 0.466 [18] | 0.526 | 0.558 | 0.205 |

# Chapter 6

# Real-time Portfolio Optimization

**Chapter Outline**

In this chapter, another neurodynamic optimization approach is proposed and used to achieve the optimal allocate scheme of portfolio selection. First, another neural network model is described. Then it will be proved that the objective function in certain portfolio selection model is pseudoconvex, and the settings of initial points are discussed, which guarantee that the recurrent neural network will achieve the optimal solution to the problem. Finally, simulation results are given and compared.

## 6.1 Introduction

Portfolio optimization [53] is a means to optimize a set of financial instruments held to achieve goals by spreading the risk of possible loss due to below expected performance. A good portfolio is not only a long list of good stocks and bonds, but also a balanced whole that provides protections and opportunities with respect to a wide range of contingencies.

Since Markowitz's pioneering work of Mean-Variance (MV) model in portfolio investment [52], many studies have been done to enhance the model. In particular, a portfolio model is proposed to maximize the probability that the rate of return is no less than an expected one [44][43].

As the market is changing, real-time portfolio optimization approach is both necessary and rewarding.

## 6.2 Model Description

For $n$ securities that the rate of return is a random vector $\xi = (\xi_1, \xi_2, \cdots, \xi_n)^T$ with normal distribution; i.e., $\xi \sim N(\mu, Q)$. Here $\mu = (\mu_1, \mu_2, \cdots, \mu_n)^T > 0$ is the mean vector of $\xi$, and $Q \in \mathbb{R}^{n\times n}$ is the positive definite covariance matrix of $\xi$, which is usually considered as a measurement of risk. Let $x = (x_1, x_2, \cdots, x_n)^T$ be the investment ratio vector, such that $\sum_i x_i = 1$. Thus, the rate of return is $\eta = x^T\xi$, $\eta \sim N(\mu^T x, x^T Q x)$, and $(\eta - \mu^T x)/\sqrt{x^T Q x} \sim N(0,1)$. Thus the optimization model of portfolio investment with probability criterion is

$$\begin{array}{ll} \text{maximize} & p(\eta \geq r) = \Phi\left(\frac{\mu^T x - r}{(x^T Q x)^{1/2}}\right), \\ \text{subject to} & 0 \leq x_i \leq 1, \quad \sum_i x_i = 1, \end{array} \tag{6.1}$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} exp(-\frac{t^2}{2})dt$ is the standard normal distribution function, and $r \geq 0$ is the expected rate of return. Since $\Phi(\cdot)$ is monotone increasing on $\mathbb{R}$, the following equivalent optimization problem can be formulated.

$$\begin{array}{ll} \text{minimize} & f(x) = \frac{r-\mu^T x}{(x^T Q x)^{1/2}}, \\ \text{subject to} & 0 \leq x_i \leq 1, \quad \sum_i x_i = 1, \end{array} \qquad (6.2)$$

For an expected rate of return $r$, $\Phi(-f(x^*))$ gives the investors the maximum probability of the portfolio investment on current $n$ securities with respect to certain expected rate of return $r$.

Since bound constraints are necessary in this model, the neural network model (3.1) is modified to achieve the optimal solution of the problem (6.2):

$$\epsilon\frac{dx}{dt} \in -\partial f(x) - \sigma_1 A^T g(Ax-b) - \sigma_2 h_{[u,v]}(x), x_0 = x(t_0), \quad (6.3)$$

where $g = (g(x_1), g(x_2), \cdots, g(x_m))^T$ and its components are defined as

$$g(x_i) = \begin{cases} 1, & \text{if } x_i > 0, \\ 0, & \text{if } x_i = 0, \ (i = 1, 2, \ldots, m) \\ -1, & \text{if } x_i < 0; \end{cases} \qquad (6.4)$$

$\sigma_1$ and $\sigma_2$ are nonnegative constants, and another discontinuous activation function $h_{[u,v]}(x)$ is defined as

$$h(x_i) = \begin{cases} 1, & \text{if } x_i > u_i, \\ 0, & \text{if } u_i \le x_i \le v_i, \ (i = 1, 2, \ldots, n) \\ -1, & \text{if } x_i < v_i. \end{cases} \tag{6.5}$$

Note that for problem (6.2), $u_i = 0$ and $v_i = 1$ for all $i = 1, 2, \ldots, n$.

## 6.3 Theoretical Analysis

*Lemma 2:* If $w : \mathcal{S} \to \mathbb{R}$ and $v : \mathcal{S} \to \mathbb{R}$ are convex, then $f(x) = w(x)/v(x)$ is pseudoconvex on $\mathcal{S} = \{x | w(x) < 0, v(x) > 0\}$.
*Proof:* By setting $\nabla f(x_2)(x_1 - x_2) \ge 0$, we have

$$\frac{1}{v(x_2)} \left( \nabla w(x_2) - \frac{w(x_2)}{v(x_2) \nabla v(x_2)} \right) (x_1 - x_2) \ge 0,$$

Since $w$ and $v$ are both convex, $\forall x_1 x_2 \in \mathcal{S}$, we have $w(x_1) - w(x_2) \ge \nabla w(x_2)(x_1 - x_2)$ and $v(x_1) - v(x_2) \ge \nabla v(x_2)(x_1 - x_2)$, and $w(x_2)/v(x_2) < 0$, we have

$$\begin{aligned} 0 &\le \nabla w(x_2)(x_1 - x_2) - \frac{w(x_2)}{v(x_2)} \nabla v(x_2)(x_1 - x_2) \\ &\le w(x_1) - w(x_2) - \frac{w(x_2)}{v(x_2)} (v(x_1) - v(x_2)) \\ &= w(x_1) - \frac{w(x_2)}{v(x_2)} v(x_1). \end{aligned}$$

Since $v(x_2) > 0$, $w(x_1)/v(x_1) \ge w(x_2)/v(x_2)$ follows directly

which indicates that $f(x) = w(x)/v(x)$ is pseudoconvex on $\mathcal{S}$.
$\square$

According to Lemma 2, the objective function $f(\cdot)$ in (6.2) is pseudoconvex on $\mathcal{S}_0 = \{x|r - \mu^T x < 0\}$.
*Theorem 7:* For all $r < \max_i \mu_i$, and large enough $\sigma_1$ and $\sigma_2$, the states of the neural network (6.3) will converge to the global optimal solution of (6.2) for any $x_0 \in \mathcal{S} = \{x|r - \mu^T x < 0, \sum_i x_i = 1\}$ and $x_0 \in \mathcal{C} = \{x|0 \le x_i \le 1\}$.
*Proof:* First, the feasible region $\mathcal{S} \cap \mathcal{C}$ is not empty for any $r < \max_i \mu_i$ since the vector with the $j$th element of 1 and the rest ones 0; i.e., $e_j = (0^1, \cdots, 0^{j-1}, 1^j, 0^{j+1}, \cdots, 0^n)^T$ is a feasible state, where $j = \max_{\mu_i} i$. Thus we can always find a feasible initial state $x_0$.

Second, since the denominator of the objective function is positive for all $x \in \mathbb{R}^n$, $f(x_1) < f(x_2)$ always holds if $r - \mu^T x_1 < 0$ and $r - \mu^T x_2 \ge 0$. As $\mathcal{S} \cap \mathcal{C}$ is not empty, it is clear that the optimal solution of problem (6.2) under condition $r - \mu^T x < 0$ is the optimal solution of the original problem (6.2).

Finally, it has been proved (see Theorems 3 and 5 in [50]) that for any $x_0 \in \mathcal{S} \cap \mathcal{C}$, $\sigma_1$ and $\sigma_2$ are large enough, the states of the neural network (6.3) will remain in the feasible region forever and converge to the optimal solution of the problem (6.2) under condition $r - \mu^T x < 0$, which is also the optimal solution to problem (6.2) as just stated. $\square$

## 6.4 Illustrative Examples

The numerical example is generated randomly in the following steps. First, the expected rate of return to a particular security varies over time. Thus the function $\mu(t)_i = r_i + k_i t + R_i \sin(t/T_i + \omega_i), i = 1, 2, \ldots, 5$ is used to describe the mean of expected rate of return at time $t$, where $k \sim U(2\times10^{-4}, 6\times10^{-4})$, $R \sim U(0.3, 1)$, $T \sim U(2, 5)$, and $\omega \sim U(0, 2\pi)$ are all randomly generated from uniform distributions. Figure 6.1 shows the statistically expected rate of return $\mu$ over a time period.

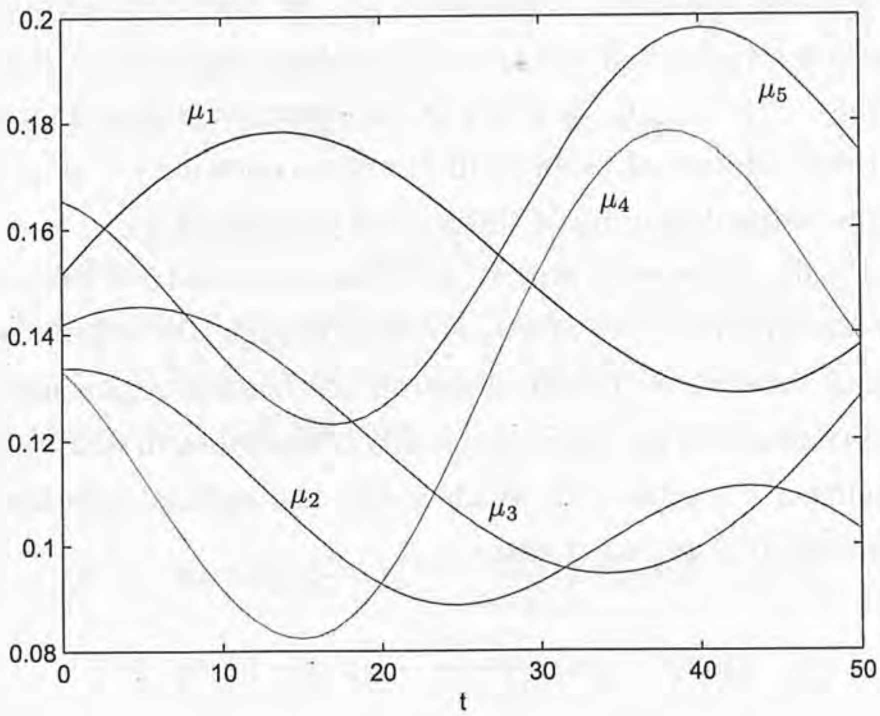

Figure 6.1: Mean of expected rate of return $\mu$.

The covariance matrix $Q = Q_1 + Q_2$ is randomly generated and fixed over time to compare the results directly, where $Q_1 = 0.01U^TU \succeq \mathbf{0}$, $u_{ij} \sim U(-1,1)$ are i.i.d variables, and $Q_2 = diag(R_1, R_2, \ldots, R_5)$:

$$Q = \begin{pmatrix} 1.849 & 0.548 & 0.089 & 0.437 & 0.207 \\ 0.548 & 2.159 & -0.011 & -1.155 & 0.004 \\ 0.089 & -0.011 & 1.697 & -0.010 & -0.547 \\ 0.437 & -1.155 & -0.010 & 3.252 & 0.372 \\ 0.207 & 0.004 & -0.547 & 0.372 & 1.523 \end{pmatrix}.$$

Then, based on the statistic results of the expected return rates and their covariance matrices, the probabilities of expected returning rate are maximized based on the proposed neurodynamic optimization model.

First we only consider a singe case when $t = 0$, and test the performance and properties of the neural network model in this application. Figure 6.2 shows the transient states of the one-layer recurrent neural network (6.3) for a single case portfolio optimization when $t = 0$ and $r = 0.1$.

Sometimes, saving money in the bank is considered as a portfolio selection as well. In this case, a sixth selection which implies saving money in the bank is added, and the expected rate of return is set to be two percent, and $q_{66} = 10^{-5}$ since there is very little risk when putting money into the bank, and $q_{i6} = q_{6i} = 0$ for $i = 1, \ldots, 5$ since the risk of bank saving has nothing to do with other investments. Figure 6.3 shows the transient states of the one-layer recurrent neural network (6.3) for a single case portfolio optimization with bank saving when $t = 0$ and $r = 0.1$. We can see that $x_6$ remains the value 0 dur-

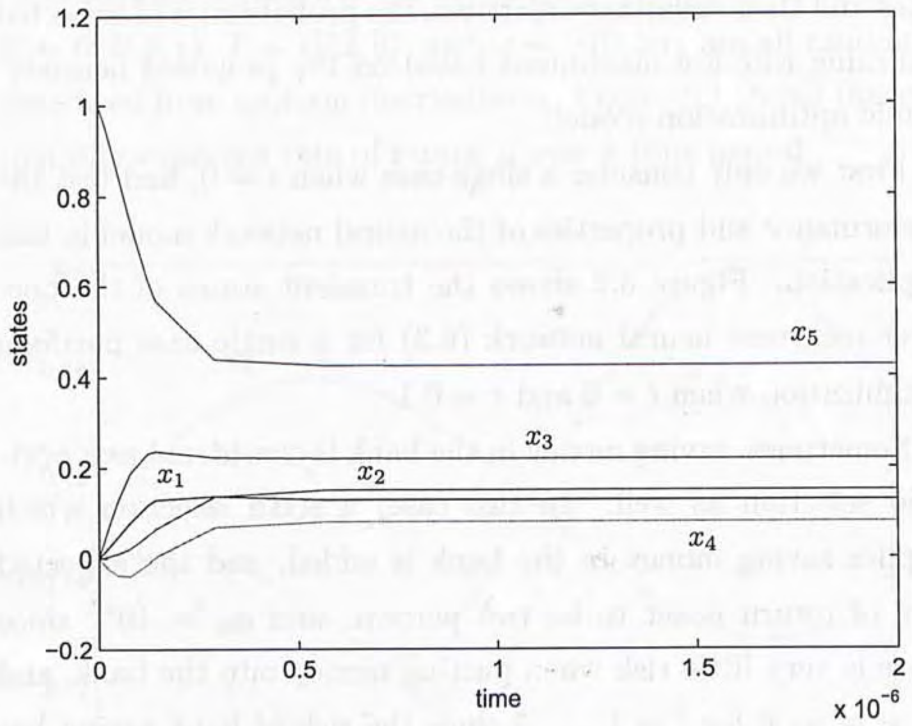

Figure 6.2: Transient states of the neural network (6.3) in portfolio selection, where $t = 0$ and $r = 0.1$.

ing convergence since the returning rate for putting the money into the bank ($\mu_6 = 0.02$) is much lower than the expected rate of return. By comparing Figures 6.3 and 6.2, we can see that in this case, adding bank saving as a choice causes no changing to the optimal portfolio selection, and the maximum probability $\Phi(-f(x^*))$ that returning rate of the investment larger than expected ($r = 0.1$) also remains 77.58%.

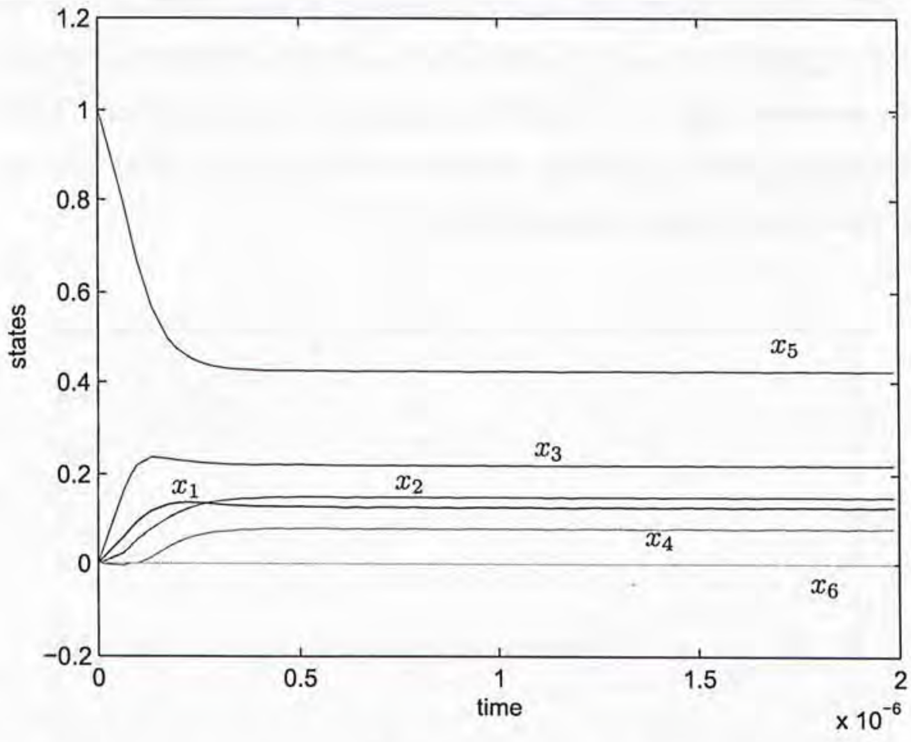

Figure 6.3: Transient states of the neural network (6.3) in portfolio selection with choice of saving money into the bank, where $t = 0$ and $r = 0.1$.

Then the expected rate of return is set to a very low value ($r = 0.02$), which is the same as the real returning rate of the bank. Figure 6.4 shows the transient states of the one-layer recurrent neural network (6.3) for this case. Not surprisingly, the optimal selection scheme suggests that one put almost all his or her money into the bank, since this is the safest way with 0 risk. The maximum probability $\Phi(-f(x^*))$ that returning rate of the investment larger than expected ($r = 0.02$) increases to 99.79%. The reason that this probability is not 100% is that in order to guarantee that the numerator of the objective function to be positive, $q_{66} = 10^{-5}$ instead of exact 0 in simulation. Thus the selection rate of putting money into the bank is only around 0.9, and there is still about 0.3% risk.

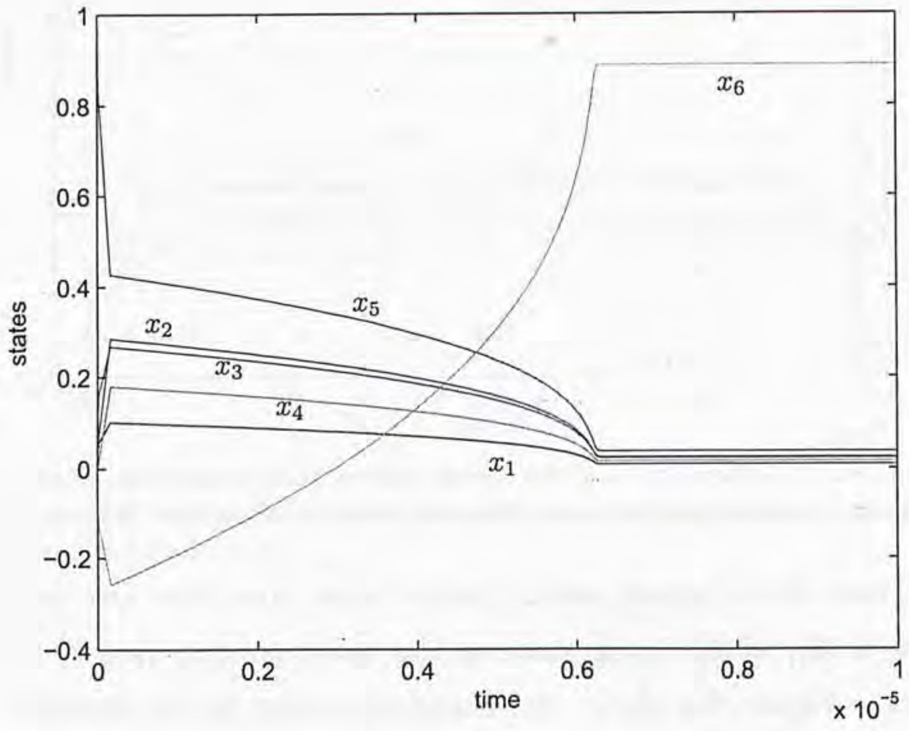

Figure 6.4: Transient states of the neural network (6.3) in portfolio selection with choice of saving money into the bank, where $t = 0$ and $r = 0.02$.

Now we consider time-varying portfolio optimization problem. As Figure 6.1 shows, the rate of return $\mu$ changes over time. Figure 6.5 shows the optimal selection scheme of the five securities over time. The probabilities that rate of return is greater or equal than an expected value ($r = 0.1$) is also calculated as $p(x^{*T}\xi \geq r) = \Phi((\mu^T x - r)/(x^T Q x)^{1/2})$ and shown in dotted line with '+' mark in the figure.

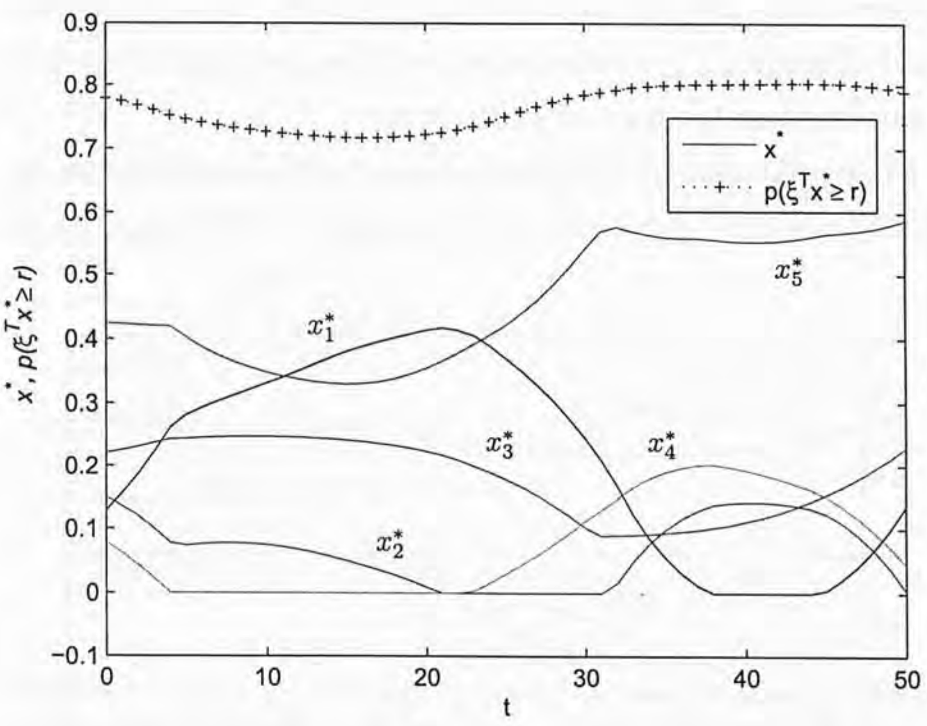

Figure 6.5: Optimal portfolio selection and the probability when the expected rate of return is at least $r = 0.1$.

From Figures 6.5 and 6.1, we can see that the selection rate $x_i^*$ of a particular security $i$ is higher if its expected rate of return $\mu_i$ is larger and also with less fluctuating ($q_{ii}$). On one hand, though $\mu_4$ is much larger than $\mu_3$ when $t > 25$, its optimal selection rate $x_4^*$ is still quite low since its return rate fluctuates more than others (with larger $q_{33}$). On the other hand, though $\mu_3 < \mu_1$ all the time, $x_3^* > x_1^*$ sometimes since $q_{33} < q_{11}$. For some time (i.e., $5 < t < 25$), most of the rates of return for the securities are not large enough ($< 0.14$). As a result, the possibility $p(x^{*T}\xi \geq r)$ of earnings of 10 percent more ($r = 0.1$) is smaller than the ones on other time.

Figures 6.6 and 6.7 shows the optimal portfolios as well as the possibilities $p(x^{*T}\xi \geq r)$ for $r = 0.12$ and $r = 0.08$ respectively.

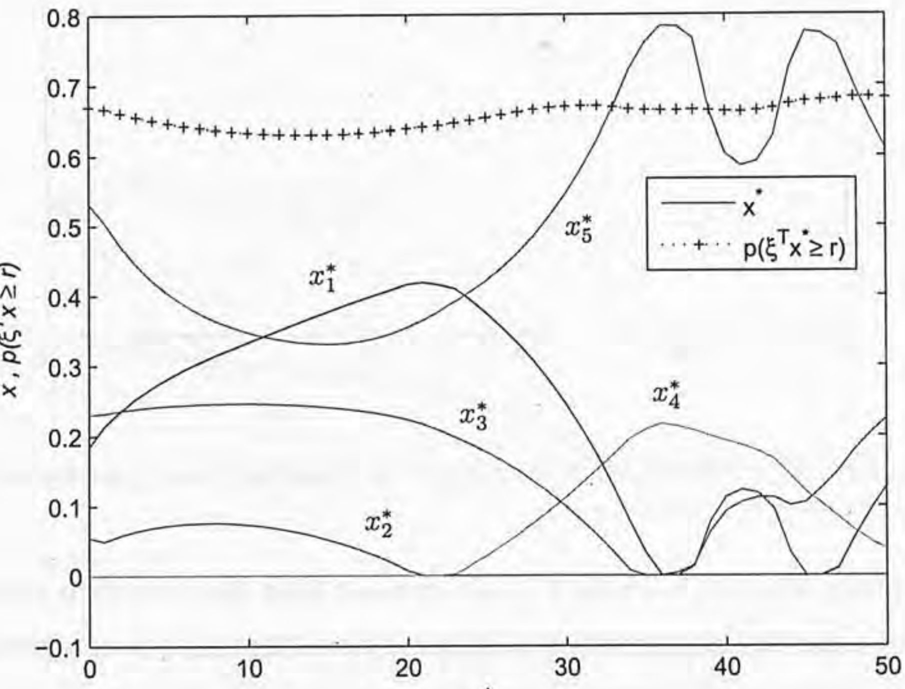

Figure 6.6: Optimal portfolio selection and the probability when the expected rate of return is at least $r = 0.12$.

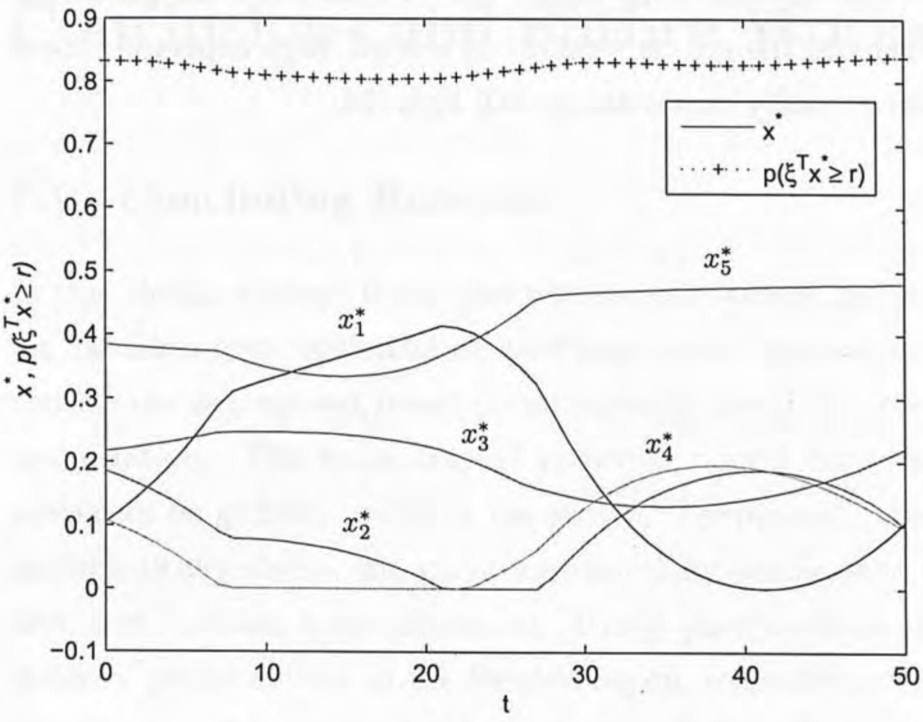

Figure 6.7: Optimal portfolio selection and the probability when the expected rate of return is at least $r = 0.08$.

By comparing Figures 6.5 - 6.7, we can see that even under the same condition (i.e., exactly the same securities), the optimal selections are different with different expectations. If one is more greedy expecting higher rate of return ($r$), the possibility to achieve the goal is smaller. In a word, high expected rate of return usually comes along with high risk.

□ **End of chapter.**

# Chapter 7

# Conclusions and Future Works

## 7.1 Concluding Remarks

In this thesis, a single-layer recurrent neural network for solving pseudoconvex optimization problems with linear equality constraints is proposed based on an existing model for convex optimization. The reconstructed recurrent neural network is proven to be globally stable in the sense of Lyapunov, globally asymptotically stable, and global exponentially stable when the objective function is pseudoconvex, strictly pseudoconvex, and strongly pseudoconvex in the feasible region, respectively. The convergence of its states to the optimal solution of the problem is also derived. Simulation results on numerical examples and application to chemical process data reconciliation is elaborated to substantiate the effectiveness of the recurrent neural network. Moreover, application on real-time portfolio optimization of a improved recurrent neural network model is done in this thesis.

## 7.2 Future Works

In fact, the objective functions of many real world optimization problems are non-smooth . There are still challenges on theoretical works for the one-layer recurrent neural networks to deal with non-smooth objective functions. Besides linear equality constraints and bound constraints, linear inequality constraints may have much wider application areas, such as 3-D reconstruction [57]. Thus, modifying the model for solving pseudoconvex optimization problem with inequality constraints might also be our further investigation.

□ **End of chapter.**

# Appendix A

# Publication List

- Qingshan Liu, Jun Wang, and Zhishan Guo, "A one-layer recurrent neural network for pseudoconvex optimization subject to linear equality and bound constraints", *Neural Networks*, Submitted, June, 2011.
- Zhishan Guo, Qingshan Liu, and Jun Wang, "A one-layer recurrent neural network for constrained optimization with pseudoconvex objective function subject to linear equality constraints", *IEEE Trans. on Neural Networks*, Resubmitted, June, 2011.
- Zhishan Guo and Jun Wang, "Information retrieval from large data sets via multiple-winners-take-all", In *Proc. of ISCAS2011*, Rio de Janeiro, Brazil, May, 2011.
- Zhishan Guo and Jun Wang, "A neurodynamic optimization approach to constrained sparsity maximization based on alternative objective functions", In *Proc. of IJCNN2010*, pp. 2932-2939, Barcelona, Spain, 2010.

- Jun Wang and Zhishan Guo, "Parametric sensitivity and scalability of k-winners-take-all networks", In *Proc. of ISNN2010*, Shanghai, China, 2010.

---

☐ **End of chapter.**

# Bibliography

[1] D. Ackley, G. Hinton, T. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, pp. 147-169, 1985.

[2] J. Aubin and A. Cellina, *Set-Valued Analysis*, Springer, New York, 1990.

[3] A. Bacciotti and F. Ceragioli, "Stability and stabilization of discontinuous systems and nonsmooth lyapunov functions", *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 4, pp. 361-376, 1999.

[4] M.P. Barbarosou and N.G. Maratos, "A non-feasible gradient projection recurrent neural network for equality constrained optimization," in *Proc. of IJCNN2004*, pp. 2251-2256, 2004.

[5] M.P. Barbarosou and N.G. Maratos, "A nonfeasible gradient projection recurrent neural network for equality-constrained optimization problems," *IEEE Trans. on Neural Networks*, vol. 19, no. 10, pp. 1665-1677, 2008.

[6] E. Bajalinov, *Linear-Fractional Programming: Theory, Methods, Applications, and Software*, Kluwer Academic Publishers, Boston, 2004.

[7] J.R. Barber, "Solid mechanics and its applications," *Elasticity, 2nd Edition*, chap. 26, pp. 351-357, 2004.

[8] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms (3nd Ed.)*, John Wiley, New York, 2006.

[9] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, New York: Academic, 1982.

[10] R. Cambini, "A class of non-linear programs: theoretical and algorithmical results", *Generalized Convexity: Lecture Notes in Economics and Mathematical Systems*, vol. 405, pp. 294-310, 1994.

[11] L. Carosi and L. Martein, "Some classes of pseudoconvex fractional functions via the Charnes-Cooper transformation," *Generalized Convexity and Related Topics*, Springer, Berlin, pp. 177-188, 2006.

[12] A. Charnes, W. Cooper W, and E. Rhodes, "Measuring the efficiency of decision making units," *European Journal of Operational Research 2*, pp. 429-444, 1978.

[13] A. Charnes, W. Cooper W, and L. Seiford, *Data Envelopment Analysis: Theory, Methodology and Applications*, Kluwer Academic Publishers, Boston, 1995.

[14] L. Cheng, Z. Hou, M. Tan, Y. Lin, W. Zhang, and F. Wang, "A recurrent neural network for non-smooth convex optimization problem with an application of genetic regulatory networks," *IEEE Trans. on Neural Networks*, to appear, 2011.

[15] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, London, U.K. Wiley, 1993.

[16] J. Cortes, "Discontinuous dynamical systems," *IEEE Control Systems Magazine*, vol. 28, no. 3, pp. 36-73, 2008.

[17] R. Cottle and J. Ferland, "Matrix-theoretic criteria for the quasi-convexity and pseudo-convexity of quadratic functions," *Linear Algebra and Its Applications*, vol. 5, pp. 123-136, 1972.

[18] B. Derya and W. Ralph, "Theory and practice of simultaneous data reconciliation and gross error detection for chemical processes," *Computers and Chemical Engineering*, vol. 28, no. 3, pp. 381-402, 2004.

[19] W. Dinkelbach, "On nonlinear fractional programming," *Management Science*, vol. 13, no. 7, pp. 492-498, 1967.

[20] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289-1306, 2006.

[21] T. Edgar, D. Himmelblau, and L. Lasdon, *Optimization of Chemical Processes (2nd Ed.)*, McGraw-Hill Inc., New York, 2002.

[22] A. Ellero, "The optimal level solutions method," *Journal of Information and Optimization Sciences*, vol. 17, no. 2, pp. 355-372, 1996.

[23] A.P. Eriksson, C. Olsson, and F. Kahl, "Normalized cuts revisited: a reformulation for segmentation with linear grouping constraints," In *Proc. International Conference on Computer Vision (ICCV)*, 2007.

[24] F. Forgo and I. Joó, "Fixed point and equilibrium theorems in pseudoconvex and related spaces," *Journal of Global Optimization*, vol. 14, pp. 27-54, 1999.

[25] M. Forti, P. Nistri, and M. Quincampoix, "Generalized neural network for nonsmooth nonlinear progranmming problems," *IEEE Trans. Circuis Syst. I*, vol. 51, pp. 1741-1754, 2004.

[26] Z. Guo and J. Wang, "A neurodynamic optimization approach to constrained sparsity maximization based on alternative objective functions," In *Proc. of IJCNN2010*, pp. 2932-2939, 2010.

[27] N. Hadjisavvas and S. Schaible, "On strong pseudomonotonicity and (semi)strict quasimonotonicity," *J. Optim. Theory Appl.*, vol. 79, no. 1, pp. 139-155, 1993.

[28] M. Hagan, H. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Co., Massachusetts, 1996.

[29] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554-2558, 1982.

[30] X. Hu, C. Sun, and B. Zhang, "Design of recurrent neural networks for solving constrained least absolute deviation problems," *IEEE Trans. on Neural Networks*, vol. 21, pp. 1073-1086, 2010.

[31] X. Hu and J. Wang, "Solving pseudomonotone variational inequalities and psudoconvex optimization problems using the projection neural network," *IEEE Trans. on Neural Networks*, vol. 17, pp. 1487-1499, 2006.

[32] —, "A recurrent neural network for solving a class of general variational inequalities," *IEEE Transactions on Systems, Man and Cybernetics-B*, vol. 37, no. 3, pp. 528-539, 2007.

[33] —, "Design of general projection neural networks for solving monotone linear variational inequalities and linear and quadratic optimization problems," *IEEE Transactions on Systems, Man and Cybernetics-B*, vol. 37, no. 5, pp. 1414-1421, 2007.

[34] X. Hu and B. Zhang, "An alternative recurrent neural network for solving variational inequalities and related optimization problems," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 21, pp. 1073-1086, 2010.

[35] P. Huber, *Robust statistics*, New York: Wiley, 1981.

[36] R. Jamisola, M. Ang, T. Lim O. Khatib and S. Lim, "Dynamics identification and control of an industrial robot" In *Proc. of ICAR99*, pp. 323-328, 1999.

[37] S. Karamardian and S. Schaible, "Seven kinds of monotone maps," *J. Optim. Theory Appl.*, vol. 66, no. 1, pp. 37-46, 1990.

[38] M. Kennedy and L. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 5, pp. 554-562, 1988.

[39] H.K. Khalil, *Nonlinear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1996.

[40] T. Kohonen, "Self-organized formation of topologically correct feature maps"' *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[41] S. Komlósi, "First and second order characterizations of pseudolinear functions", *European Journal of Operation Research*, vol. 67, pp. 278-286, 1993.

[42] J.P. LaSalle, "Some extensions of Liapunov's second method," *IRE Transactions on Circuit Theory*, CT-7, pp. 520-527, 1960.

[43] J. Liang and W. Tang, "Probability criterion in portfolio investment model with commisions," *Systems Engineering*, vol. 19, pp 5-10, 2001.

[44] B. Liu and C. Ku, "Probability criterion in inventory systems," *Journal of Systems Science and Mathematical Science*, vol. 13, no. 1, pp 70-75, 1993.

[45] Q. Liu and J. Wang, "A recurrent neural network for non-smooth convex programming subject to linear equality and bound constraints," In *I. King et al. (eds.): ICONIP 2006*, Part II, LNCS 4233, pp. 1004-1013, 2006.

[46] —, "A one-layer recurrent neural network with a discontinuous activation function for linear programming," *Neural Computation*, vol. 20, pp. 1366-1383, 2008.

[47] —, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming," *IEEE Trans. Neural Networks*, vol. 19, no. 4, pp. 558-570, 2008.

[48] —, "A one-layer recurrent neural network for non-smooth convex optimization subject to linear equality constraints," In *M. Köppen et al. (eds.): ICONIP 2008*, Part II, LNCS 5507, pp. 1003-1010, 2009.

[49] —, "Finite-time convergent recurrent neural network with a hard-limiting activation function for constrained optimization with piecewise-linear objective functions," *IEEE Trans. Neural Networks*, vol. 22, no. 4, pp. 601-603, 2011.

[50] Q. Liu, J. Wang, and Z. Guo "A one-layer recurrent neural network for pseudoconvex optimization subject to linear

equality and bound constraints," Preprint submitted to *Neural Networks*, May, 2011.

[51] O. L. Mangasarian, "Pseudo-convex functions," *SIAM Journal on Control*, vol. 3, pp. 281-290, 1965.

[52] H. Markowitz, "Portfolio selection," *Journal of Finance*, vol. 3, pp. 151-158, 1952.

[53] H. Markowitz, *Portfolio selection: efficient diversification of investments*, Yale University Press, 1990.

[54] B. Martos, *Nonlinear Programming Theory and Methods*, North Holland, Amsterdam, 1975.

[55] P. Mereau and J. Paquet, "Second order conditions for pseudo-convex functions," *SIAM Journal on Applied Mathematics*, vol. 27, no. 1, pp. 131-137, 1974.

[56] K. Mjelde , "Allocation of resources according to a fractional objective," *European Journal of Operational Research*, pp.116-124, 1978.

[57] C. Olsson and F. Kahl, "Generalized convexity in multiple view geometry," *J. Math Imaging Vis.*, vol. 38, pp. 35-51, 2010.

[58] C. Olsson, A. P. Eriksson, and F. Kahl, "Efficient optimization for $L_\infty$-problems using pseudoconvexity," In *Proc. International Conference on Computer Vision (ICCV)*, 2007.

[59] C. Olsson, A. P. Eriksson, and F. Kahl, "Improved spectral relaxation methods for binary quadratic optimization problems," In *Proc. Computer Vision and Image Understanding*, 2007.

[60] J. Ponstein, "Seven kinds of convexity," *SIAM Rev.*, vol. 9, no. 1, pp. 115-119, 1967.

[61] D. Ripps, "Adjustment of experimental data," *Chemical Engineering Progress Symposium Series*, vol. 61, pp. 8-13, 1965.

[62] A. Rodríguez-Vzquez, R. Domínguez-Castro, A. Rueda, J. Huertas, E. Sanchez-Sinencio, "Switched-capacitor neural networks for linear programming," *Electronics Letters*, vol. 24, pp. 496-498, 1988.

[63] D. Rollins and J. Davis, "Gross error detection when variance-covariance matrices are unknown." *Journal of American Institute of Chemical Engineers*, vol. 39, no. 8, pp. 1335-1341, 1993.

[64] J. Romagnoli and M. Sanchez, *Data Processing and Reconciliation for Chemical Process Operation*, Academic Press, New York, 1999.

[65] J. Rosen, "The gradient projection method for nonlinear programming. part i: linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181-217, 1960.

[66] R. Serth, C. Valero, and W. Heenan, "Detection of gross errors in nonlinearly constrained data: A case study," *Chemical Engineering Communications*, vol. 51, pp. 89-104, 1987.

[67] R. Serth and W. Heenan, "Gross error detection and data reconciliation in steam-metering systems," *Journal of American Institute of Chemical Engineers*, vol. 32, pp. 733-742, 1986.

[68] H. Seigelmann, *Neural Networks and Analog Computation: Beyond the Turing Limit*, Birkhuser, Boston, 1999.

[69] S. Schaible, "Fractional programming," In *Handbook of Global Optimization*, pp. 495-608, 1995.

[70] D.W. Tank and J.J. Hopfield, "Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits and Systems*, vol. 33, pp. 533-541, 1986.

[71] W. Tang, Y. Wang, and J. Liang, "Fractional programming model for portfolio with probability criterion," In *IEEE International Conference on Systems, Man and Cybernetics*, pp 516-519, 2002.

[72] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, pp. 629-641, 1994.

[73] —, "Primal and dual assignment networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 784-790, 1997.

[74] —, "Primal and dual neural networks for shortest-path routing," *IEEE Transactions on Systems, Man, and Cybernetic-A*, vol. 28, no. 6, pp. 864-869, 1998.

[75] J. Wang, Q. Hu, and D. Jiang, "A Lagrangian network for kinematic control of redundant robot manipulators," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1123-1132, 1999.

[76] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Networks*, vol. 7, no. 6, pp. 1544-1547, 1996.

[77] —, "Global convergence analysis of Lagrangian networks," *IEEE Transactions on Circuits and Systems-I*, vol. 50, no. 6, pp. 818-822, 2003.

[78] Y. Xia and J. Wang, "A recurrent neural network for solving linear projection equations," *Neural Netw.*, vol. 13, no. 3, pp. 337-350, 2000.

[79] —, "A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints," *IEEE Transactions on Circuits and Systems-I*, vol. 51, no. 7, pp. 1385-1394, 2004.

[80] Y. Xia, F. Gang, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations," *Neural Networks*, vol. 17, pp. 1003-1015, 2004.

[81] —, "A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints," *IEEE Trans. Neural Networks*, vol. 19, no. 8, pp. 1340-1353, 2008.

[82] Y. Xia, H. Leung, and J. Wang, "A projection neural network and its application to constrained optimization problems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 49, no. 4, pp. 447-458, 2002.

[83] W. Zangwill, "Non-linear Programming via penalty functions," *Management Science*, vol. 13, no. 5, pp. 344-358, 1967.

[84] S. Zhang and A.G. Constantinides, "Lagrange programming neural networks," *IEEE Trans. Circuits and Systems II*, vol. 39, pp. 441-452, 1992.

[85] S. Zhang, X. Zhu, and L.H. Zou, "Second-order neural nets for constrained optimization," *IEEE Trans. Neural Networks* vol. 3, pp. 1021-1024, 1992.