Continuous-Time Recurrent Neural Networks for Quadratic Programming — Theory and Engineering Applications

LIU Shubao

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Automation and Computer-Aided Engineering

August 2005

Abstract

In this thesis, the design, analysis and application of recurrent neural networks (RNNs) are discussed. The design is motivated by the time-varying optimzation problems' real-time solution requirement. The analysis is mainly concentrated on the convergence property, which can be studied from the dynamic system viewpoint, and computational complexity. The analysis has twofold effects. First, it can show the solution capability of RNNs from the theoretical viewpoint; Second, it can show methodology of how to analysis analog neural networks, which is very important in analysis of the biological neural networks. The application part will show the applications in wide engineering areas.

In the past half century, many research efforts have been put into the constrained optimization problem. And as the outcome, several efficient methods have been proposed, such as the simplex methods, active set methods, interior point methods and etc. They have been successfully applied to the static constrained optimization problems. In principle, they can also be used to tackle the time-varying constrained optimization problems. But the high computational burden prohibits their practical deployment in the complex circumstances. Usually the problems in these circumstances can be formulated as time-varying constrained optimization problems, such as in robot control, signal process, biomedical engineering, and etc. Because the real-time requirement always accompanies the time-varying constrained optimization problems, the discrete serial methods face many challenges.

Recurrent neural networks have the property of high parallelism, adaptivity and analog VLSI implementability. These characteristics can be utilized to solve the time-varying constrained optimization problems. Remarkable advances have been made in the area of RNN during the past twenty years, both in theory and application aspects. As a new analog computing paradigm, Continuous-time Recurrent Neural Networks (CRNNs) have several advantages compared with digital ones. It can handle analog inputs directly; It works in asynchronous mode; It can tackle large-scale and real-time applications; It is energy efficient.

In this thesis, several CRNN models are proposed for quadratic opti-

mization. A simplified dual neural network suitable for solving time-varying convex linearly constrained quadratic programming (LCQP) problems is proposed. This recurrent neural network's computational ability is analyzed from the perspectives of convergence and number of neurons. It is shown to be globally convergent to the exact optimal solutions. Compared with existing neural networks capable of solving QP problems, the complexity of neural networks is reduced with the number of neurons is only equal to the number of inequality constraints. And two recurrent neural networks are investigated to solve the quadratically constrained quadratic programming (QCQP) problems.

Their applications to engineering problems are also discussed. The K-winners-take-all (KWTA) operation is first converted to an equivalent quadratic optimization problem and then the neural network model is tailed to design the analog circuit to do the KWTA operation. In robotics, the redundant robot arm path planning and control can be formulated as a equality constrained time-varying optimization problem with the joint angle velocity or the joint torque as the variables. Also the obstacle avoidance can also be incorporated into the time-varying optimization framework. It can be formulated as an time-varying inequality constraints. In the wireless communication, the optimal multiuser detection is a NP-hard problem. After dedicated derivation and some practical hypothesis, it can be formulated as a time-varying quadratically constrained qudaratic programming problem. The details of the formualtions are discussed in the thesis. The simulation results show that CRNN can complete the time-varying on-line engineering optimization tasks in real-time.

摘要

本論文初步探討了遞歸/回饋神經網絡的設計、分析和應用。設計是建立在時變優化問題的實時求解的需求之上的。分析主要集中在動態系統的穩定性和計算複雜度兩點。分析部分有兩層意義，一可以得出所設計神經網絡的求解能力；二可以爲分析普遍意義上的模擬神經網絡提供方法基礎。應用部分則介紹了如何將工程上的問題轉化到時變優化問題上來。

從原理上講，傳統的串行數值方法也可以求解時變約束優化問題，但是巨大的運算量和實時性要求使其在複雜的現實應用（如機器人控制、信號處理、生物醫學影像等）中是基本不可行的。遞歸/回饋神經網絡具有高度並行性、自適應性、可模擬大規模集成電路實現等特點。這些特點可以充分利用來設計特定的網絡結構以求解時變信號的處理問題。

本論文針對約束二次優化提出了幾種遞歸神經網絡模型。適於求解線形不等約束二次優化問題的簡化對偶神經網絡是基於對偶神經網絡提出來的。此網絡在保留了全局收斂性的基礎上簡化了神經網絡的結構，使得神經元和突觸連接的數目得到了大幅減少。簡化神經網絡的神經元的個數恰好等於不等約束的個數。兩個針對二次不等約束二次優化問題的兩個神經網絡模型也已經提出。

神經網絡在工程上的應用在本論文中也作了大量探討。KWTA 運算首先被轉化爲一個離散時變二次優化問題，之後又進一步被轉化爲連續時變二次優化問題。基於前面建立的網絡模型提出了 KWTA 的一個電路實現。在機器人領域，自由度冗餘機械臂的軌跡規劃和控制能夠被轉化爲關於關節角速度和力矩的等式約束時變優化問題；避障問題則可轉化爲一個不等約束時變問題。在無線通訊中，最優多用戶檢測問題本身是一個 NP-hard 問題，在經過了嚴密推演和近似假設後，能夠轉化爲一個時變二次約束二次優化問題。這個問題的解就是多用戶檢測問題的一個近似解。其中的轉化細節在文中都作了細緻探討。通過仿真實驗，遞歸神經網絡能夠實時可靠地處理各種工程問題。

Acknowledgement

During my time at CUHK, I have been very fortunate to have the guidance, friendship and cooperations from lots of people, who have helped me very much in both academic and personal affairs. Without them, this thesis is no way in its current form.

Here, I would like to thank Professor Jun Wang for admitting me to this program and being my advisor in the last two years. Intellectually he introduced me to the field of computational intelligence and helped me do research in more rigorous way. His high standards for research always stimulate me to produce better work, and thereby learn more. During the past two years, Professor Wang has provided me with invaluable insights on research and paper writing. For all of these, I will always be grateful.

Many people have helped shape my understanding of research and mathematics in the last two years. Dr Yousheng Xia helped me a lot at the beginning of my MPhil study and gave me a lot of advice on research. I also greatly benefited much from discussions with Dr Zhigang Zeng. With frequent discussion with Dandan Li and Yinhui Yan, my interest in several branches of mathematics arose and the role of math in engineering research is becoming clearer to me. The math taste and appreciation cultured during the past two years is a great harvest during the period. All these transfers are impossible without their sincere help. Thanks in this aspect should also be forwarded to the ACAE department for its multi-discipline research culture.

Besides research and learning, there's much more content in life. The life is colorful and enjoyable with the presence of Dandan Li, Jian Liang, Xiaolin Hu, Tong Liu, Qinhe Zhang, Yinhui Yan and many other people without space to list.

As an important role in the MPhil study, teaching gave me many challenges, in terms of oral presentation, communication with students and etc. In this respect, I thank my partners Ming and Turkey for their kind help.

Finally, my deepest thanks go to my parents. From the early on, they instilled in me the importance of education. I should 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

Summary

In this chapter, the motivations of this research are first described. With the rapid industry development, more and more procedures are needed to be optimized for economical reasons. Many of these optimization problems are not static problems, they are inherently dynamic, i.e., time-varying. The parameters of the processes are changing from time to time. The real-time optimization of this dynamic process constitutes a major challenge for the traditional serial optimization methods. Recurrent neural networks have good properties that can be utilized to effectively solve the problem. This chapter mainly answers the "Why?" problems. Why does the serial methods face challenges? Why the RNN is effective? Is the RNN capable of real-time solution in principles? Is the RNN implementable?

1.1 Time-Varying Quadratic Optimization

Constrained optimization is concerned with optimizing an objective function over a domain. The domain can be described by a group of equality and/or inequality constraints. With mathematical notations, it can be written as

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

where $x \in R^n$.

If the objective and/or domains are not static, they can be described as having a list of parameters, and these parameters are time-varying. Then the optimization problem is called a time-varying quadratic optimization problem.

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

where $\theta(t)$ and $\Omega(t)$ are time-varying parameters and domain.

If the objective function is also convex quadratic and the domain is a convex set, the optimization problem is often called as a quadratic optimization (or quadratic programming) problem.

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2} x^T W(t) x + c(t)^T x, \\ \text{subject to} \quad & A(t)x = b(t), \\ & l(t) \leq Ex \leq h(t). \end{aligned}$$

where $W(t)$ is a symmetric positive definite matrix.

This paper deals with time-varying convex quadratic programming problems with equality and inequality constraints by recurrent neural networks. The interest in such problems comes from two facts. First, many engineering problems can be formulated as time-varying quadratic optimization problems. For example, in the root control, desired position is changing with time, the obstacle is moving. In wireless communication, the channel is under rapid change because of environment and user signals. They can all be incorporated in a time-varying quadratic optimization framework. In Part II, you will see some detailed descriptions of these application problems. In some sense, if the dynamic problem is to be optimized locally, they can usually be formulated as a time-varying optimization problem. Second, in many algorithms for general nonlinear programming, a search direction is determined at each iteration as a solution of a quadratic problem.

There are a lot of publications devoted to quadratic programming, see e.g. [3], such as prime and dual simplex method, active set method, interior point method, and etc. All these methods are a collection of various detailed methods. The prime and dual simplex methods can only deal with bounded inequality constraints. The active set method proceeds by partitioning the inequality constraints into two sets: active and inactive. Here active means the equality holds. The inactive constraints are ignored during the iteration. Then, the new value is generated by moving on the surface defined by the working set. Contrary to the active set method, the interior point method achieves optimization by moving through the middle of the region defined by the problem rather than around its surface as the active method. The

interior-point method was developed in the 1980s to solve linear programming problems. Later it was found that it can be used to solve convex optimization problems as well. These new methods allow us to solve certain new classes of convex optimization problems, such as semidefinite programs and second-order cone programs, almost as earilier as linear programs.

All these serial numerical methods have been successfully applied to the static constrained optimization problems. In principle, they can also be used to tackle the time-varying constrained optimization problems. But the high computational burden prohibits their practical deployment in the complex circumstances. Usually the problems in these circumstances can be formulated as time-varying constrained optimization problems, such as in robot control, signal process, biomedical engineering, and etc. Because the real-time solution requirement always accompanies the time-varying constrained optimization problems, the discrete serial methods face many challenges.

1.2 Recurrent Neural Networks

1.2.1 From Feedforward to Recurrent Networks

Neural networks have become a very popular research field in computer science, engineering, cognitive science and mathematics. They represent a broad range of information processsing models which mimic the information processing of the biological neural networks. Often the "neural network" is characterized by a assemblies of weighted interconnected (connections are called weights or synaptics) simple processing components (the components are called neurons). By embedding a vast number of simple neurons in an interacting system, it is possible to provide computational power for very sophisticated information processing. Usually the neural networks have the adaptive and parallel processing characters.

There are two streams of neural networks research, one considering neural networks as a model to help understand the neural systems/brain, the other considering neural networks as a powerful family of nonlinear statistical model and analog computing paradigm. Historically, the neural network was first proposed to study the brain theory, but the application potentials have shifted the balance with the developments of computer technology. Currently this field is experiencing rapid development because of its applications. The applications include robotics, pattern recognition (for speech and vision systems), and understanding human brain-mind processes.

The strength of application-driven neural networks hinges upon three main characteristics:

Adaptiveness and self-organization: It offers robust and adaptive processing capabilities by adopting adaptive learning or self-organizing rules or recurrent feeback.

Nonlinear network processing: By utilizing the nonlinear behavior of natural phenomena, like the exponential increase of current with voltage in silicon transistors, it enhances the networks' computational capability.

Parallel processing: It usually employs a large number of procesisng cells enhanced by a extensive interconnectivity.

Historically, neural implementations have focused on feed-forward networks which is composed of layers of neurons, with information following from the input layer through the hidden layer to the output layer. There is no feedback mechanism within this network itself (although in the training period, the error information can be propagated backwards.) This is in distinct contrast to biological neural networks, which has no training period and no particular back-propagation mechanism. They are inherently recurrent. The reasons for this preference of feed-forward networks are manifold, including the difficulty in understanding and control of the highly nonlinear complex recurrent behaviors. The other reason is that the feed-forward networks have showed adequate performance in many engineering applications, such as data processing, pattern recognition, and robot control. For example, neural network researchers often exploit the following easy and successful strategy. Given a problem currently solved with a standard algorithm, interpret computations performed by the algorithm as a parameterized mapping from an input to an output, and call this mapping a neural network; then adapt the parameters using the available data samples so as to produce another mapping that solves the task better.

But the most biologically plausible of these, which has just recently become feasible and attract many interests, is the recurrent neural networks, which we would like to discuss in the context of constrained optimization.

In recurrent neural network, some of the signals come from the environment, acting as the input to the NN, while other signals go out of the NN to the environment, encoding the end result of the computation, acting as output, just as shown in Fig.1.1.

The status of the weights, whether unknown parameters or fixed constants, prompts two different working mode of neural networks. When the weights are considered unknown parameters, the network is a semiparametric statistical model, able to approximate input-output relations by means of parameter estimation. While the weights are considered constant

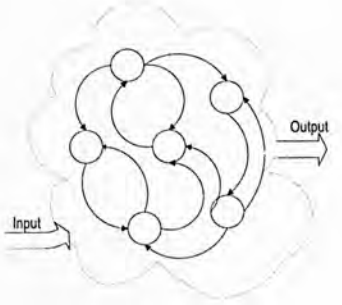


Figure 1.1: Recurrent Neural Network General Model

after or without a process of adaptation, the network can perform exact computations rather than mere approximations.

Different from most feedforward neural networks, which operate on discrete time, synchronous mode, and have layered structure, the continuous neural network operate on continuous time, asynchronous mode and have interconnected structure. Traditional neural networks are described by the difference equations, while the continuous neural network is described by ordinary differential equations. So the traditional neural network is studied by the arithmetic methods, while the continuous neural network should be studied by the ordinary differential equation theory. From the system theory viewpoint, CRNN can be regarded as a continuous dynamic system. From this viewpoint, we can investigate its stability property. The curious quality of recurrent neural networks is that they retain traces of the previous inputs, and in this sense incorporate context. The previous states have an impact on the current states, even without additional learning algorithms. What exactly is being represented, or how it is being represented, is largely unknown. The representations must be drawn from the features that are fed as input, and temporally, these must be short-term memories of some kind. It turns out that this is exactly what we need in time-varying optimization: short-term memory. Metaphorically we can think of this short-term tracking as 'catching the thread' of time-varying trends. Indeed, recurrent neural networks seems to be one of the most promising methods for resolving the hard problem.

The reason why we study continuous-time neural network, not discrete or digital neural network can be partly explained by the following quote.

> "We first note that the assumption of continuity may be beneficial, even when the hardware is digital. A prime example is the linear programming problem with its polyhedral structure of solutions. A search in the space of vertices by the simplex algorithm

> has exponential worst case behavior. On the other hand, interior point algorithms, such as Karmarkar's algorithm, approach the solution from the inside of the continuous polytope and require polynomial time only. This exemplifies the speedup gain from considering a continuous phase space."[18]

As we have pointed out, the continuous recurrent neural network can be seen as a dynamical system. The state a dynamical system settles into is called an *attractors*. Dynamical systems are called dissipative if their dynamics converge to attractors when a dissipative system has an energy (Lyapunov) functional, the attractors are fixed points; otherwise, more complex attractors may appear. If we want use the continuous recurrent neural network to do computation, we can use the settled down state. This computing paradigm is called computing with attractors.

As a special case of computing with attractors, continuous recurrent neural network works on continuous time and asynchronous mode. The advantages of this paradigm can be summarized as:

- it handles analog input directly;
- it works in asynchronous mode, i.e. without global clock to synchronize all the processes;
- its computing complexity is very low and it has large computing density for some parallelizable problems;
- it is fault tolerant and robust;
- it is energy efficient compared with digital computing.

There is a large class of problems that are at best poorly solved. These problems involve the transformation of data across the boundary between the real world and the digital world. And they occur whenever a computer is sampling and/or acting on real world data. These are difficult problems to solve on a computer, since they require the computer to find complex structures and relationships in massive quantities of low precision, ambiguous, noisy data. Analog recurrent neural networks meet with these challenges and requirements.

1.2.2 Computational Power and Complexity

Although neural networks are based on continuous operations, as computing machines, we still need analyze their computational power using the computational complexity theory.

Traditional computational complexity deals with the concept of computability. Historically, each computational class of functions was associated with an automaton. The most popular automaton is the Turing machine. It is neither the weakest nor the strongest, but it is the mathematical equivalent of the modern computer . It is shown the the analog recurrent neural network is more powerful than Turing machine, although their upper limit is unknown [18].

The prime historic computational computability of neural networks research is Minsky and Papert's 1969 study of the computational limitations of single-layer perceptrons, which has played a major role in driving the researchers away from the neural network research.

The modern theory of computational complexity does not deal only with the ultimate power of a machine, but also with its expressive power under constraints on resources, such as time and space. One of the basic problems of the neural networks in the computational complexity is their scalability. Early research has show that they work adequately good enough on small problems, but when they are scaled up to larger problems they often need more neurons than possible with current technology. In thiscase the resources needed to scaled up grow up too rapidly, i.e., the model does not scale up well. If they grow too fast (usually we consider the number of resources grow up faster than log-linear as too fast), it may be not feasible to expect that advances in technology can keep pace. Here the resources for neural computation may include number of neurons, number of connections, and convergence time. [17]

"The prime contribution of neural networks is not their mode of computation, but the capability for efficient computation of certain problems." That means, for some certain problems, neural networks' computational complexity is lower compared with digital computer. For example, the time-varying signal processing elements like integrator can be easily implemented using current silicon technology, while for digital technology, it means a hard computational burdern by numerical integration.

1.2.3 Implementation Issues

The rapid developing technology of very large scale integrated (VLSI) circuits have given us a medium to fabricate thousands of millions of transistors interconnected on a single silicon wafer. "The Integrated-circuit fabrication has evolved to the point where systems of the scale of small, but identifiable, parts of the nervous system can be emulated on a single piece of silicon. We are not limited by the constraints inherent in our fabrication technology; we are limited by the paucity of our understanding."[13]

"To create silicon structures for emulating biologically inspired computing, we need a better understanding of biological computing models, and we need VLSI design techniques that emulate these models efficiently. We also need to identify those aspects of computational neurobiology which are necessary and which are not. For example, analog computation has advantage in low precision computations and low power applications, and impressive computational density. However, analog computation also has disadvantages in stability, temperature sensitivity, communication, and ease of design. And it is not clear that analog's computational density is an advantage in sparsely activated, sparsely connected networks.

Digital technology is less area efficient, especially for certain types of functionality (e.g., leaky integration). It is also power intensive, and the representation of time tends to be more complicated (events are typically synchronized to a global clock). But digital allows for the efficient multiplexing of scarce computational and communication resources." [9] Hybrid, analog/digital or "mixed-signal" techniques may be the optimum design point.

But usually two critics for the implementability of recurrent neural networks argue that

1. Because the analog circuits and analog signals are easy to be corrupted by noise, the precision of computation will not be guaranteed.

2. It is difficult to implement a highly connected neural network for current technology. Because the current VLSI circuits lies on 2 dimensional plane, whether there are 3 or 4 layers of plane.

Certainly analog neural networks face these shortcomings. But when we analyze these two questions deeply, you can see that the conclusion is not correct. First, the easy corruption problem can be easily overcome by redundancy. We can observe that the biological neural network is resistent to noise even to failures. Their robustness comes from the highly redundancy of connections and neurons. In the contrary, we can expect that systems with extraordinary reliability and robustness will result; so much so that useful integration at the scale of a complete wafer is feasible. Secondly, the highly connection can be implemented by hybrid models.

Of these, connectivity is one of the most important characteristics of biological neural structures. Unfortunately, connectivity is perhaps the one area where silicon is significantly less robust than biology. Communication in silicon is generally limited to a two-dimensional plane (though with several levels, 6-8 with today's semiconductor technologies). It is still one of the most important problems as we consider scaling to very large models.

Concurrently, Carver Mead's group at Caltech and others developed "Address Event Representation" or ARR communication [10][11]. The address-event technique has also been expanded into a hierarchical structure by Lazzaro and Wawrznyk [12]. When analog computation is used, signals can be represented by action-potential-like "spike' (generally a neuron unit exceeding its threshold). These signal "packets" or "pulses" are transmitted asynchronously at the most they occur, by sending the originating unit's address on a single multiplexed bus. This "pseudo-digital" representation allows multiplexing of the bus and retention of temporal information, if contention for the units sharing the bus is minimal.

"Perhaps the most rewarding aspect of analog computation is the extent to which elementary computational primitives are a direct consequence of fundamental laws of physics."

"It is essential to recognize that neural systems evolved without the slightest notion of mathematics or engineering analysis. Nature knew nothing of bits, Boolean algebra, or linear system theory. But evolution had access to a vast array of physical phenomena that implemented important functions. It is evident that the resulting computational metaphor has a range of capabilities that exceeds by many orders of magnitude the capabilities of the most powerful digital computers."

1.3 Thesis Organization

This thesis begins with the introduction chapter. In this chapter, we introduce the elements of constraints optimization and recurrent neural networks, from the perspectives of the differences of feedforward and recurrent neural networks, the computational power and complexity of RNN, and the implementation issues, especially the analog implementation. We have emphasized the challenges that motivate this research and showed the characteristics of RNN which is useful for the challenge.

Then comes Part I on theory and models of RNN. In this section, the development history of RNN for optimization is firstly reviewed. Then the linearly constrained quadratic optimization and the quadratically constrained quadratic optimization are respectively studied in terms of design procedure, analysis of convergence and complexity and validation by numerical simulations.

Engineering applications are discussed in Part II. There you will find applications in circuit design, robotics and wireless communication. More detailedly, Chapter 4 discusses the KWTA circuit design through steadily reformulating the KWTA problem into a LCQP problem. Chapters 5 and

6 discuss the applications in robotics, respectively on dynamics control and obstacle avoidance. The multiuser detection in wireless communication is tackled in Chapter 7.

At last, the conclusions and future works to be done is described in Chapter 8. My personal view of the prospect of RNN is also discussed in Chapter 8.

□ **End of chapter.**

Part I

Theory and Models

Summary

A continuous recurrent neural network model is proposed to solve the linearly constrained quadratic programming problems and is proven to be globally convergent to the exact solution. And two continous recurrent neural network models are proposed to solve the quadratic constrained quadratic programming problems and its convergence behavior are being investigated.

Chapter 2

Linearly Constrained QP

A *strictly convex quadratic program* (QP) is an optimization problem whose objective function is strictly convex quadratic, and the constraint functions are affine. In QP, a convex quadratic function is minimized over a polyhedron.

Quadratic program problems arise in many fields such as signal processing, robot control, machine learning, economic analysis, transport planning and etc [1][54][84][89][95]. A number of numerical algorithms have been proposed to solve the problem. But the computational complexity of these serial algorithms may limit their applications in large-scale and/or online optimization applications, such as in on-line learning(classification), fluid dynamics and robotics path planning, etc. In the above fields the ability to solve large-scale and/or on-line optimization problems is essential.

In the past nearly two decades, the parallel computing abilities of recurrent neural networks, known as natural computing, is recognized by researchers in many disciplines [89][95][54][84]. The essence of the neural computing is that a dynamic system can be designed to be globally convergent to the solution of the desired problem. And the dynamic system can be easily implemented by the electronic circuits or more recently by analog or digital VLSI circuits technologies. The good property of this computing paradigm is that it can tackle large-scale problems and time-varying problems without great efforts.

Tank and Hopfield proposed the first working recurrent neural network implemented on analogue circuits [26][27], which opened the avenue of solving optimization problems by neural computing. These years, various neural network models have been developed for solving the quadratic program problems. According to their design method, these neural network can be categorized as the penalty-parameter NN[28], the Lagrange NN[29], the primal-dual[31] and the dual NN[32][49][53]. It is known that the neural network model [28] contains finite penalty parameters and generate approximate so-

lutions only. The Lagrangian neural network is not suitable for solving the inequality constrained optimization problems. In addition, the dimensionaltiy of Lagrange network is much larger than that of original problems, due to the introduction of slack and surplus variables. As a much flexible tool for exactly solving quadratic program problems, the primal-dual neural network [31] was developed with the feature that it handles the primal quadratic program and its dual problem simultaneously by minimizing the duality gap with Karush-Kuhn-Tacker condition (KKT-condition) and using the gradient method. Unfortunately, the dynamic equations of the primal-dual neural network are usually complicated, and may contain high-order nonlinear terms. Moreover, the network size are usually larger than the dimensionality of the primal quadratic program plus its dual problem. In [32][49], a neural network called dual neural network is presented to solve the convex quadratic problem utilizing only the dual variables. Its dynamic equation is piecewise linear. In this chapter, we will give out a new neural network called improved dual neural network based on dual neural network to solve the quadratic program problem. It can further eliminate the architecture complexity of neural network.

2.1 Model Description

A general time-varying convex quadratic programming problem can be expressed as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T W(t)x + c(t)x \\ \text{subject to} \quad & A(t)x = b, \\ & l \leq E(t)x \leq h, \end{aligned} \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the variable, $W(t) \in \mathbb{S}^n_{++}$[1], $c(t) \in \mathbb{R}^n$, $A(t) \in \mathbb{R}^{m\times n}$, $b(t) \in \mathbb{R}^m$ $(m < n)$, $E(t) \in \mathbb{R}^{p\times n}$, $l, h \in \mathbb{R}^p$. l and h can respectively be $-\infty$ and ∞, which corresponds to one-side inequality or no inequality constraint.

From now on, we simply denote $W(t)$ as W, $b(t)$ as b, and so on. Here, for simplicity, suppose A is a full rank matrix (i.e., $\text{rank}(A) = m$) without loss of generality. When $\text{rank}(A) = m' < m$ (i.e., not full rank), we can find a maximum linearly independent subset of row vectors of A. The subset vectors constitute a row matrix A' ($\text{rank}(A') = m$) with the corresponding new vector b'. Then the equality constraint is equivalent to

$$A'x = b',$$

[1]Here, $W(t) \in \mathbb{S}^n_{++}$ means that $W(t)$ is symmetric and positive definite.

where A' is a full rank matrix.

Consider (2.1) as the primal problem, its dual problem is

$$\begin{aligned} \text{maximize} \quad & b^T y - \frac{1}{2} x^T W x + l^T v - h^T w \\ \text{subject to} \quad & Wx + c - A^T y - E^T v + E^T w = 0, \end{aligned} \tag{2.2}$$

where $y \in \mathbb{R}^m$, $v, w \in \mathbb{R}^p$ are dual variables.

Define $u := v - w$, the equality constraint in (2.2) becomes

$$Wx + c - A^T y - E^T u = 0.$$

According to the Karush-Kuhn-Tucker (KKT) conditions for convex optimization [1], the following set of equations have the same solution as problem P

$$\begin{aligned} & Wx + c - A^T y - E^T u = 0, \\ & Ax = b, \\ & \begin{cases} (Ex)_i = l, & \text{if } u_i > 0, \\ (Ex)_i = h, & \text{if } u_i < 0, \\ l \le (Ex)_i \le h, & \text{if } u_i = 0. \end{cases} \end{aligned}$$

That is,

$$Wx + c - A^T y - E^T u = 0, \tag{2.3}$$

$$Ax = b, \tag{2.4}$$

$$Ex = g(Ex - u), \tag{2.5}$$

where $g(z)$ is a piecewise linear function, defined as

$$g(z_i) = \begin{cases} l, & \text{if } z_i < l, \\ z_i, & \text{if } l \le z_i \le h, \quad i = 1, \cdots, p \\ h, & \text{if } z_i > h. \end{cases}$$

If $l = -\infty$, $g(z_i)$ degenerates to

$$g(z_i) = \begin{cases} z_i, & \text{if } z_i \le h, \\ h, & \text{otherwise.} \quad i = 1, \cdots, p \end{cases}$$

If $h = \infty$, $g(z_i)$ degenerates to

$$g(z_i) = \begin{cases} v_i, & \text{if } z_i \ge l, \\ l, & \text{otherwise.} \quad i = 1, \cdots, p \end{cases}$$

If $r = -\infty$, $h = \infty$, $g(z_i)$ degenerates to a linear function

$$g(z_i) = z_i.$$

From equation (2.3), because W is invertible, we can get

$$x = W^{-1}(A^T y + E^T u - c). \tag{2.6}$$

Substitute (2.6) into (2.4),

$$AW^{-1}(A^T y + E^T u - c) = b.$$

Because rank$(A) = m$ and W is invertible, then $AW^{-1}A^T$ is invertible. So y can be explicitly expressed by u as

$$y = (AW^{-1}A^T)^{-1}\left[-AW^{-1}E^T u + AW^{-1}c + b\right]. \tag{2.7}$$

Based on (2.5), (2.6), (2.7), by the projection theorem, the dynamic equation of the neural network for solving the primal problem (2.1) can be designed as

- State equation

$$\epsilon\frac{du}{dt} = -Ex + g(Ex - u), \tag{2.8}$$

- Output equation

$$\begin{aligned} x &= W^{-1}(A^T y + E^T u - c), \\ y &= (AW^{-1}A^T)^{-1}\left[-AW^{-1}E^T u + AW^{-1}c + d\right], \end{aligned} \tag{2.9}$$

where $u \in \mathbb{R}^p$ is the state vector, $\epsilon > 0$ is a scaling parameter that controls the convergence rate of the neural network. It can be rewritten in the more compact and explicit form as

- State equation

$$\epsilon\frac{du}{dt} = -EME^T u + g(EME^T u - u + Es) - Es, \tag{2.10}$$

- Output equation

$$x = ME^T u + s, \tag{2.11}$$

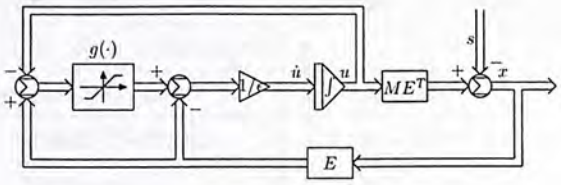


Figure 2.1: Block diagram of the simplified dual neural network.

where $M := W^{-1}(I - A^T G)$, $s := W^{-1}(A^T h - c)$, $G := (AW^{-1}A^T)^{-1}(AW^{-1})$, $h := (AW^{-1}A^T)^{-1}(AW^{-1}c + b)$. Because the analytic expression of A, W, F can be obtained in the design stage, the analytic expressions of M and s can be computed beforehand. In view of this, though expression (2.10) appears complicated, it is not computationally complex.

The block diagram of the neural network is shown in Fig. 2.1.

In this neural network, u is the state variable, x is the output of the neural network, and the time-varying parameters W, b, c, E, l, h are the inputs. In the circuit implementation, u is electrical signal, which operates on $ns - \mu s$ time scale, while the time-varing parameters usually work in $ms - s$ time scale. So for the neural network, the time-varying optimization problem can be seen as static problems. This is the rational why we can use the electronic neural network to solve the time-varying optimization problem. There are several advantages compared with traditional serial digital methods. (1) It is a parallel solution, compared with traditional methods. With the parallelizable ability, it can tackle large scale problems; (2) With the global convergence property, the neural network solution is stable no matter how the inputs change; (3) It can be implemented by analog VLSI which means that the neural network can tackle practical problems directly without A/C converteer with high speed to ensure online operations and produce continuous outputs.

In the simplified dual neural network, the number of neurons is equal to the number of inequality constraints, whereas the number of neurons is equal to the number of equality and inequality constraints in the original dual neural network [32] [49]. A complete comparison with several neural networks for solving the quadratic program problem (2.1) is shown in Table 2.1.

2.2 Convergence Analysis

A neural network is said to be globally convergent if starting from any initial point taken in a given domain of the whole state space, any trajectory of

Neural Network Type	Number of Neurons	Reference
Simplified Dual Neural Network	p	this paper
Lagrange Neural Network	$n+m+4p$	[29]
Prime-Dual Neural Network	$n+m+2p$	[31]
Dual Neural Network	$m+p$	[49]

Table 2.1: Comparison of architecture complexity among various neural networks

the corresponding dynamic system converges to an equilibrium point that depends on the initial state of the trajectory.

To analyze the convergence of the simplified dual neural network, three lemmas are first introduced.

Lemma 1 *[2] Assume that the set $\Omega \subset R^m$ is a closed convex set, then the following two inequalities hold*

$$(P_\Omega(\alpha)-\beta)^T(\alpha-P_\Omega(\alpha))^T \geq 0, \quad \forall \alpha \in R^m, \beta \in \Omega;$$
$$\|P_\Omega(\beta)-P_\Omega(\alpha)\| \leq \|\beta-\alpha\|, \quad \forall \alpha, \beta \in R^m,$$

where $P_\Omega : R^m \rightarrow \Omega$ is a projection operator defined as $P_\Omega(\gamma) = \min_{\zeta\in\Omega} \|\gamma - \zeta\|$.

Remark 1 It is clear that the set $\Omega := \{u \in \mathbb{R}^p | l \leq u \leq h\}$ and $g(u)$ satisfy the above projection property.

Lemma 2 *For $\Lambda = diag\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ with $\lambda_i > 0$ $(i = 1, 2, \cdots, n)$, $P \in \mathbb{R}^{m\times n}$ $(m < n)$, rank$(P) = m$, the following matrix inequality holds*

$$\Lambda - P^T(P\Lambda^{-1}P)^{-1}P \succeq 0.^2$$

Proof Denote $P = [p_1\ p_2\ \cdots\ p_m]^T$, we can find $n-m$ column vectors $\tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_{n-m}$, such that

$$p_i^T\Lambda^{-1}\tilde{p}_i = 0.$$

Define $\tilde{P} := [\tilde{p}_1\ \tilde{p}_2\ \cdots\ \tilde{p}_{n-m}]$.

$$\begin{aligned} & \tilde{P}\Lambda^{-1}P^T = 0_{(n-m)\times m} \\ \Rightarrow\ & (\tilde{P}\Lambda^{-1}P^T)^T = 0_{m\times(n-m)} \\ \Rightarrow\ & P\Lambda^{-1}\tilde{P}^T = 0_{m\times(n-m)}. \end{aligned}$$

[2]Here, $H \succeq 0$ means matrix H is positive semi-definite.

Because $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ is full rank and $\text{rank}(P) = m$, then $p_1, p_2, \cdots, p_m, \tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_{n-m}$ are linearly independent vectors. Define: $Q := \begin{bmatrix} P \\ \tilde{P} \end{bmatrix} \in R^{n\times n}$, then Q is invertible. For $\forall y$, we have

$$\begin{aligned}
& y^T\Lambda y \\
= \; & y^TQ^T(Q^T)^{-1}(\Lambda^{-1})^{-1}Q^{-1}Qy \\
= \; & (Qy)^T(Q\Lambda^{-1}Q^T)^{-1}(Qy) \\
= \; & (\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} y)^T(\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} \Lambda^{-1} \begin{bmatrix} P^T & \tilde{P}^T \end{bmatrix})^{-1}(\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} y) \\
= \; & \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix}^T \begin{bmatrix} P\Lambda^{-1}P^T & P\Lambda^{-1}\tilde{P}^T \\ \tilde{P}\Lambda^{-1}P^T & \tilde{P}\Lambda^{-1}\tilde{P}^T \end{bmatrix}^{-1} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\
= \; & \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \begin{bmatrix} P\Lambda^{-1}P^T & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & \widetilde{P}\Lambda^{-1}\tilde{P}^T \end{bmatrix}^{-1} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\
= \; & \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \begin{bmatrix} (P\Lambda^{-1}P^T)^{-1} & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & (\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1} \end{bmatrix} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\
= \; & y^TP^T(P\Lambda^{-1}P^T)^{-1}Py + y^T\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\widetilde{P}y
\end{aligned}$$

Move $y^TP^T(P\Lambda^{-1}P^T)^{-1}Py$ to the left side,

$$y^T\Lambda y - y^TP^T(P\Lambda^{-1}P^T)^{-1}Py = y^T\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\tilde{P}y. \tag{2.12}$$

Because $\Lambda^{-1} \succeq 0$,

$$\tilde{P}\Lambda^{-1}\tilde{P}^T \succeq 0.$$

Then

$$\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\tilde{P} \succeq 0.$$

That is,

$$y^T\tilde{P}^T(\tilde{P}\Lambda^{-1}\tilde{P}^T)^{-1}\widetilde{P}y \geq 0, \quad \forall y \in R^n. \tag{2.13}$$

From (2.12) and (2.13),

$$y^T\Lambda y - y^TP^T(P\Lambda^{-1}P^T)^{-1}Py \geq 0, \quad \forall y \in R^n.$$

i.e.,

$$\Lambda - P^T(P\Lambda^{-1}P)^{-1}P \succeq 0.$$

The proof is complete □

Lemma 3 *Let* $W \in \mathbb{S}^n_{++}, A \in \mathbb{R}^{m\times n}, F \in \mathbb{R}^{p\times n}(m < n)$, *then*

$$F(W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1})F^T \succeq 0$$

Proof By the matrix spectrum theorem, because $W \in \mathbb{S}^n_{++}$, it can be decomposed as

$$W = T\Lambda T^T \tag{2.14}$$

where $\Lambda = \text{diag}\,\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ with $\lambda_i > 0$ $(i = 1, 2, \cdots, n)$ the eigenvalue of W and T is an orthogonal matrix. Then, take the inverse of both side of (2.14)

$$W^{-1} = T\Lambda^{-1}T^T. \tag{2.15}$$

Define $P := AT \in R^{m\times n}$ $(m < n)$, by Lemma (2),

$$\Lambda - P^T(P\Lambda^{-1}P^T)^{-1}P \succeq 0.$$

$\Lambda^{-1} \succ 0$, so

$$I - P^T(P\Lambda^{-1}P^T)^{-1}P\Lambda^{-1} \succeq 0.$$

Furthermore,

$$\Lambda^{-1}(I - T^TA^T(AT\Lambda^{-1}T^TA^T)^{-1}AT\Lambda^{-1})T^T \succeq 0.$$

In view of the equation (2.15), we can get

$$W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1} \succeq 0.$$

The proof is complete. □

Theorem 1 *The simplified dual neural network (2.10) is globally convergent to an equilibrium point u^*.*

Proof At u^*, we have the following inequality

$$(z - Ex^*)^T u^* \geq 0, \quad \forall z \in \Omega,\ \Omega := \{u \in \mathbb{R}^p | l \leq u \leq h\}, \tag{2.16}$$

which can be obtained by considering the following three cases:

- If for some $i \in \{1, 2, \cdots, p\}$, $u_i^* = 0$, $l- \leq [Ex^*]_i \leq h$, then $(z_i - [Ex^*]_i)u_i^* = 0$;
- If for some $j \in \{1, 2, \cdots, p\}$, $u_j^* > 0$, $[Ex^*]_j = l$ and $l \leq z_j \leq h$, then $z_j - [Ex^*]_j \geq 0$ and thus $(z_j - [Ex^*]_j)u_j^* \geq 0$;
- If for some $k \in \{1, 2, \cdots, p\}$, $u_k^* < 0$, $[Ex^*]_k = h$ and $l \leq z_k \leq h$, then $z_k - [Ex^*]_k \leq 0$ and thus $(z_k - [Ex^*]_k)u_k^* \geq 0$.

Therefore, by subsituting $z = g(EME^T u + Es - u)$, we can get from (2.16)

$$\left[g(EME^T u + Es - u) - (EME^T u^* + Es)\right]^T u^* \geq 0. \tag{2.17}$$

On the other hand, from Lemma 1, it follows that, $\forall u \in R^p$

$$\begin{aligned} &\left[g(EME^T u + Es - u) - (EME^T u^* + Es)\right]^T \\ &\left[(EME^T u + Es - u) - g(EME^T u + Es - u)\right] \geq 0. \end{aligned} \tag{2.18}$$

Combining (2.17) and (2.18), we have

$$\begin{aligned} &\left[g(EME^T u + Es - u) - (EME^T u^* + Es)\right]^T \\ &\left[u^* + (EME^T u + Es - u) - g(EME^T u + Es - u)\right] \geq 0. \end{aligned} \tag{2.19}$$

Define $\tilde{g} := g(EME^T u + Es - u) - (EME^T u + Es)$, (2.19) becomes

$$\left[\tilde{g} + EME^T(u - u^*)\right]^T \left[(u - u^*) + \tilde{g}\right] \leq 0. \tag{2.20}$$

From (2.20), we can get

$$\begin{aligned} &(u - u^*)^T \tilde{g} + \tilde{g}^T EME^T(u - u^*) \leq \\ &-\|\tilde{g}\|^2 - (u - u^*)^T EME^T(u - u^*). \end{aligned} \tag{2.21}$$

And

$$\begin{aligned} EME^T &= EW^{-1}(I - A^T G)E^T \\ &= E(W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1})E^T. \end{aligned}$$

According to Lemma 2, EME^T is positive semidefinite, i.e.,

$$(u - u^*)^T EME^T(u - u^*) \geq 0. \tag{2.22}$$

From (2.21) and (2.22), we get

$$(u - u^*)^T \tilde{g} + \tilde{g}^T EME^T(u - u^*) \leq 0, \tag{2.23}$$

and if and only if $u = u^*$, the equality holds.

Now choose the following radially unbounded Lyapunov functional candidate

$$V(u(t)) = \frac{1}{2}\|Q(u(t) - u^*)\|_2^2, \tag{2.24}$$

where Q is a symmetric and positive definite matrix with $Q^2 = (I + EME^T)$.

Then from (2.23), we get

$$\begin{aligned} \frac{dV}{dt} &= (u-u^*)^T Q^2 \dot{u} \\ &= (u-u^*)^T (I+EME^T)\tilde{g} \\ &= (u-u^*)^T \tilde{g} + \tilde{g}^T EME^T (u-u^*) \\ &\leq 0. \end{aligned} \tag{2.25}$$

By the Lyapunov stability theorem, the simplified dual neural network is globally stable.

The proof is complete. □

Theorem 2 *$x^* = ME^T u^* + s$ is an optimal solution to the quadratic programming problem (2.1), where u^* is an equilibrium point of dynamic equation (2.10).*

Proof Since u^* is an equilibrium point of the dynamic equation (2.10),

$$g(EMEu^* + Es - u^*) - (EME^T u^* + Es) = 0.$$

By $x^* = ME^T u^* + s$,

$$g(Ex^* - u^*) = Ex^*. \tag{2.26}$$

Define $y^* := Gu^* + h$, then

$$\begin{aligned} x^* &= ME^T u^* + s \\ &= W^{-1}(A^T(Gu^* + h) + E^T u^* - c) \\ &= W^{-1}(A^T y^* + E^T u^* - c). \end{aligned} \tag{2.27}$$

That is,

$$Wx^* + c - A^T y^* - E^T u^* = 0. \tag{2.28}$$

Substituting G and h into the expression of y^*, we have

$$y^* = (AW^{-1}A^T)^{-1}[(-AW^{-1}E^T)u^* + AW^{-1}c + b].$$

$AW^{-1}A^T$ is invertible, then

$$AW^{-1}A^T y^* = (-AW^{-1}E^T)u^* + AW^{-1}c + b.$$

Leaving only b on the right side,

$$AW^{-1}A^T y^* + AW^{-1}E^T u^* - AW^{-1}c = b.$$

i.e.,

$$AW^{-1}(A^T y^* + E^T u^* - c) = b.$$

In view of x^* in equation (2.27), we have

$$Ax^* = b. \tag{2.29}$$

Equations (2.26), (2.28), (2.29) constitute the KKT conditions (2.3), (2.4), (2.5) of the problem (2.1). So $x^* = ME^T u^* + s$ is the optimal solution to the quadratic programming problem (2.1).

The proof is complete. □

From Theorems 1 and 2, we can conclude that the simplified dual neural network in (2.10), (2.11) is globally convergent to the exact optimal solution of problem (2.1).

To show the convergence behavior of the neural network, let's consider a simple numerical example.

$$\begin{aligned}
\text{minimize} \quad & 3x_1^2 + 3x_2^2 + 4x_3^2 + 5x_4^2 + 3x_1x_2 + 5x_1x_3 + \\
& x_2x_4 - 11x_1 - 5x_4 \\
\text{subject to} \quad & 3x_1 - 3x_2 - 2x_3 + x_4 = 0, \\
& 4x_1 + x_2 - x_3 - 2x_4 = 0, \\
& -x_1 + x_2 \leq -1, \\
& -2 \leq 3x_1 + x_3 \leq 4.
\end{aligned}$$

Written in the standard form, the corresponding parameters are

$$W = \begin{bmatrix} 6 & 3 & 5 & 0 \\ 3 & 6 & 0 & 1 \\ 5 & 0 & 8 & 0 \\ 0 & 1 & 0 & 10 \end{bmatrix}, \; c = \begin{bmatrix} -11 \\ 0 \\ 0 \\ -5 \end{bmatrix},$$

$$A = \begin{bmatrix} 3 & -3 & -2 & 1 \\ 4 & 1 & -1 & -2 \end{bmatrix}, \; b = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$E = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \end{bmatrix}, \; l = \begin{bmatrix} -\infty \\ -2 \end{bmatrix}, \; h = \begin{bmatrix} -1 \\ 4 \end{bmatrix}.$$

For solving this quadratic program problem, the improved dual neural network needs only 2 neurons, while the Lagrange neural network [29] needs 12 neurons, the primal-dual neural network [31] 9 neurons, the dual neural network [49] 4 neurons.

The simulation results with $\lambda = 10^8$ are shown in Figs. 2.2, 2.3 and 2.4. Figs. 2.2 and 2.3 illustrate respectively the convergence behaviors of

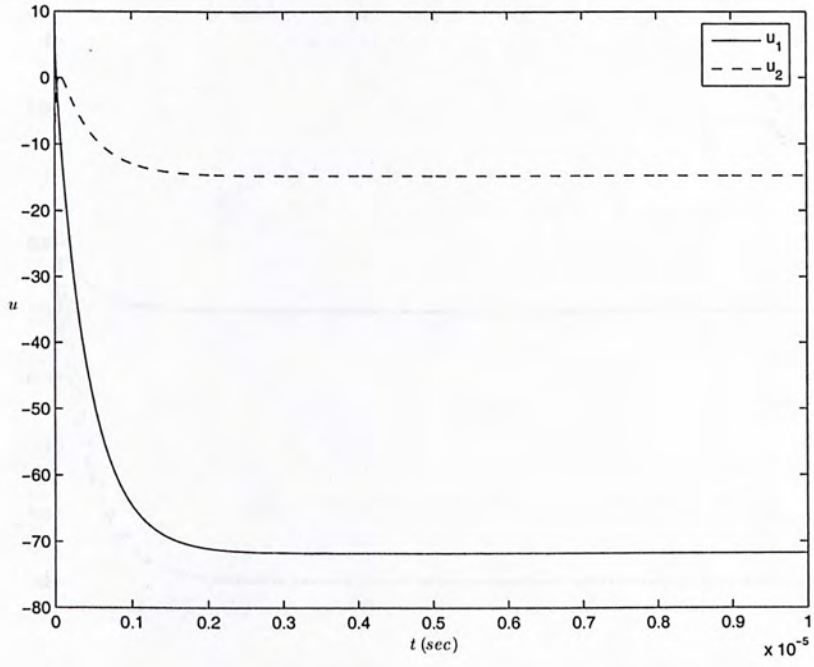


Figure 2.2: Transient behavior of u.

variables u and x. From the initial state $[10, -10]$, the improved dual neural network converges to the optimal solution $[0, -2, 2, -2]$ within 10^{-5} second. And the norm of the difference between the optimal solution and the neural network output value at time instance 10^{-5} second is less than 3×10^{-10}. Fig. 2.4 shows the state trajectories of the improved dual neural network converging to the optimal solution from different initital states.

□ **End of chapter.**

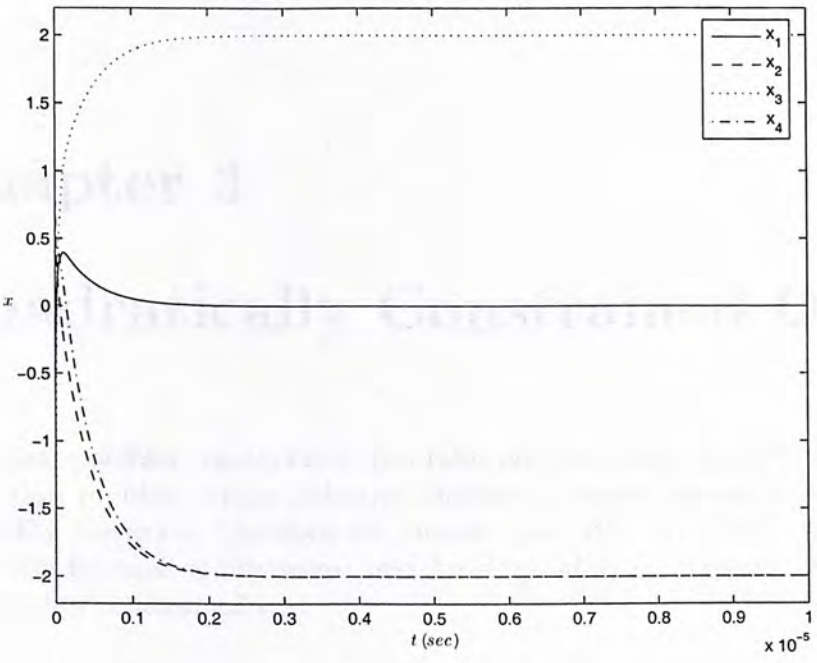


Figure 2.3: Transient behavior of x.

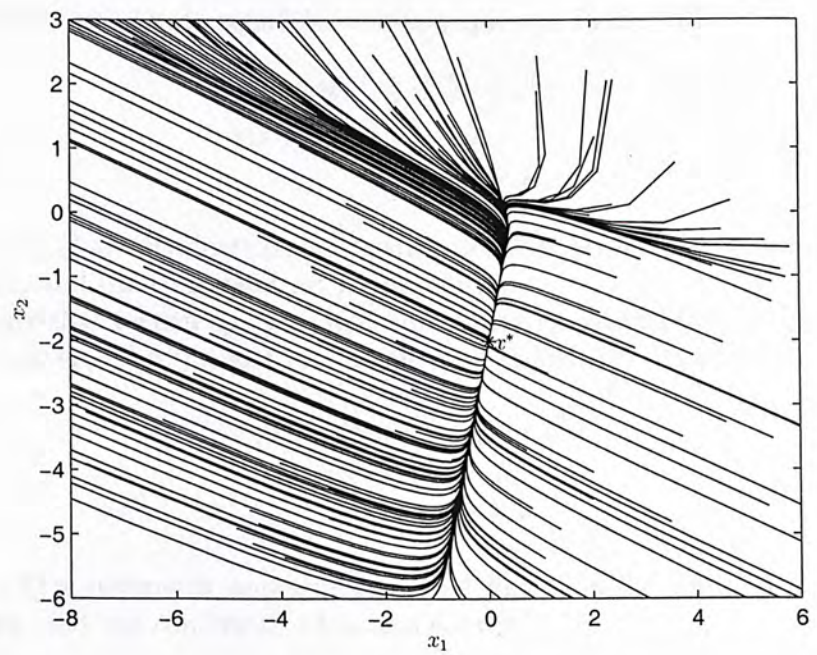


Figure 2.4: Trajectory of x_1 and x_2 from different initial point.

Chapter 3

Quadratically Constrained QP

A convex quadratic constrained quadratic programming (QCQP) is an optimization problem whose objective function is convex quadratic, and the inequality constraint functions are convex quadratic. In QCQP, a convex quadratic function is minimized over an ellipsoid or an ellipsoid surface in the equality constrained one.

3.1 Problem Formulation

A QCQP with linear equalities can be expressed in the form:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}x^T\bar{Q}x + \bar{c}^T x, \\ \text{subject to} \quad & \bar{A}x = \bar{b}, \\ & \tfrac{1}{2}x^T\bar{R}x + \bar{d}^T x \leqslant \bar{r}. \end{aligned}$$

where Q, R are symmetric and positive definite. So the objective function is convex, and the constraint set is also convex.

Note that we can apply an invertible linear transformation to the problem that converts the ellipsoid into a sphere of radius $\sqrt{r}$. This result in

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}x^T Q x + c^T x, \\ \text{subject to} \quad & Ax = b, \\ & \|x\|_2^2 \leqslant r. \end{aligned} \tag{3.1}$$

where Q is symmetric and semipositive definite. So the objective function is convex, and the constraint set is also convex.

3.2 Model Description

The Lagrangian dual problem is to maximize $\theta(u)$ over $u \geqslant 0$, where

$$\theta(u) = \inf_x \{\frac{1}{2}x^T Qx + c^T x + u(\frac{1}{2}x^T x - \frac{1}{2}r) - (Ax - b)^T v : x \in R^n\}$$

Note that for a given $u \geqslant 0$, the function $\frac{1}{2}x^T Qx + c^T x + u(\frac{1}{2}x^T x)$ is convex, so a necessary and sufficient condition for a minimum is that the gradient must vanish, that is

$$Qx + c + ux - A^T v = 0 \tag{3.2}$$

Thus the dual problem can be written as follows:

$$\begin{array}{ll} \text{maximize} & \frac{1}{2}x^T Qx + c^T x + u(\frac{1}{2}x^T x - \frac{1}{2}r) - (Ax - b), \\ \text{subject to} & Qx + c + ux - A^T v = 0, \\ & u \geqslant 0, \quad v \geqslant 0. \end{array} \tag{3.3}$$

Now from (3.2), we have

$$x^T Qx + c^T x + u(x^T x) - x^T A^T v = 0$$

Substituting this into (3.3), we derive the dual quadratic programming given below

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2}x^T Qx - \frac{1}{2}x^T x - \frac{1}{2}ur - b^T v, \\ \text{subject to} & Qx + c + ux - A^T v = 0, \\ & u \geqslant 0, \quad v \geqslant 0. \end{array} \tag{3.4}$$

KKT conditions

$$\begin{array}{l} Qx + c + ux = 0, \\ \begin{cases} \|x\|_2^2 < r, & u = 0; \\ \|x\|_2^2 = r, & u > 0. \end{cases} \end{array} \tag{3.5}$$

That is,

$$Qx + c + ux - A^T v = 0, \tag{3.6}$$

$$Ax = b, \tag{3.7}$$

$$\|x\|_2^2 = g(\|x\|_2^2 - u). \tag{3.8}$$

Because $Q \in \mathbb{S}_{++}^n$, by the matrix spetrum theorem, Q can be decomposed as

$$Q = T\Lambda T^T$$

$$\begin{array}{ll} & Qx + ux + c - A^T v = 0 \\ \Rightarrow & T\Lambda T^T x + uTIT^T x = A^T v - c \\ \Rightarrow & T(\Lambda + uI)T^T x = A^T v - c \\ \Rightarrow & x = T^T(\Lambda + uI)^{-1}T(A^T v - c) \end{array}$$

NN Type	# of Neurons	Convergence	Solution	Reference
Simplified Dual NN	1	G.C.[1]	Exact	this chapter
Dual NN	m+1	G.E.C.[2]	Exact	this chapter
Nonlinear NN	n+m+1	G.E.C.	Exact	[29]
Lagrangian NN	n+m+2	G.C.	Approximate	[91]

Table 3.1: Comparison between various NNs capable of solving QCQP

3.2.1 Model 1 (Dual Model)

- State Equation

$$\frac{d}{dt}\begin{pmatrix} u \\ v \end{pmatrix} = \lambda \begin{pmatrix} g(\|x\|_2^2 - u) - \|x\|_2^2 \\ -Ax + b \end{pmatrix}$$

- Output Equation

$$x = T^T(\Lambda + uI)^{-1}T(A^T v - c)$$

No. of Neuron: m+1

Supposed Convergence Property: Globally Exponential Convergence

3.2.2 Model 2 (Improved Dual Model)

$$\begin{aligned} & Ax = b \\ \Rightarrow\ & AT^T(\Lambda + uI)^{-1}T(Av - c) = b \\ \Rightarrow\ & AT^T(\Lambda + uI)^{-1}TA^T v = AT^T(\Lambda + uI)^{-1}Tc + b \\ \Rightarrow\ & v = [AT^T(\Lambda + uI)^{-1}TA^T]^{-1}[AT^T(\Lambda + uI)^{-1}Tc + b] \end{aligned}$$

- State Equation

$$\frac{d}{dt}u = \lambda(g(\|x\|_2^2 - u) - \|x\|_2^2)$$

- Output Equation

$$v = [AT^T(\Lambda + uI)^{-1}TA^T]^{-1}[AT^T(\Lambda + uI)^{-1}Tc + b]$$

$$x = T^T(\Lambda + uI)^{-1}T(A^T v - c)$$

No. of Neuron: 1

Supposed Convergence Property: Globally Convergence

□ End of chapter.

Part II

Engineering Applications

Summary

In this part, several application examples are given out to show the ability of solving engineering problems of the proposed neural networks. First, they shows how to properly formulate the desired problems into time-varying quadratic optimizations through careful derivation and thorough consideration of the physical feasibility. Then the correspdonging neural networks are tailed according to the specific case of each application. The simulation results demonstrated the feasbility and efficicies of the solutions.

Chapter 4

KWTA Network Circuit Design

Summary

In this chapter, the K-Winners-Take-All (KWTA) operation is converted to an equivalent constrained convex quadratic optimization formulation. A simplified dual neural network, called KWTA network, is further developed for solving the convex quadratic programming (QP) problem. The KWTA network is shown to be globally convergent to the exact optimal solution of the QP problem. Simulation results are presented to show the effectiveness and performance of the KWTA network.

4.1 Introduction

Winner-take-all (WTA) is an operation that identifies the largest value from the input signals. Such an operation has many applications including associative memories [6], cooperative models of binocular stereo [55], Fukushima's neocogniton for feature extraction, and etc [56]. In the combinatorial optimization, this operation is also needed.

As an extension of winner-take-all operation, k-winners-take-all (KWTA) selects the k largest inputs from the total n inputs. It can be considered as a generalized version of winner-take-all operation. It has recently been shown that KWTA is computationally powerful compared with standard neural network models with threshold logic gates [57][58]. Any boolean function can be computed by a single k-winners-take-all unit applied to weighted sums of input variables. Beside the applications in neural network model, the KWTA operation has important applications in machine learning, such

as k-neighborhood classification, k-means clustering, etc. As the number of inputs becomes large and/or the selection process should be operated in real time, parallel hardware implementation is desirable. There have been many attempts to design very large scale integrated (VLSI) circuits to do the KWTA operation[59][60][61][62][63][64][65][66]

This chapter proposes a new neural network implementation of KWTA operation based on the equivalent quadratic optimization formulation, which has the O(N) complexity. For this network, global convergence is guaranteed and time-varying signals can be tackled. The rest of this chapter is organized as following. Section II derives an equivalent formulation of KWTA, which is suitable for neural network design. Section III introduces the neural network design procedure, architecture and properties. Simulation results are reported in Section IV. Section V concludes this chapter.

4.2 Equivalent Reformulation

The optimization capability of the recurrent neural network has been widely investigated. After the seminal work of Tank and Hopfield [26][27], various neural networks have been proposed. They can be categorized as the penalty-parameter neural network [28], the Lagrange neural network [29], the deterministic annealing neural network [30], the primal-dual neural network [31][52] and the dual neural network [32][49].

Mathematically, KWTA can be formulated as a function

$$x_i = f(v_i) = \begin{cases} 1, & \text{if } v_i \in \{k \text{ largest elements of } v\}; \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Fig. 4.1 shows the KWTA operation graphically. In this section, we will reformulate the KWTA operation as a quadratic programming problem, which is suitable for neural network design. Toward this goal, hereafter two theorems are given and proved.

Theorem 3 *The solution of (4.1) is the same as the solution to the following discrete quadratic programming problem (4.2).*

$$\begin{aligned} &\textit{minimize} && ax^T x - v^T x \\ &\textit{subject to} && e^T x = k \\ & && x_i \in \{0,1\}, \quad i = 1,2,\cdots,n \end{aligned} \tag{4.2}$$

where a is a positive constant, $v := [v_1, v_2, \cdots, v_n]^T$, $x := [x_1, x_2, \cdots, x_n]^T$, $e := [1, 1, \cdots, 1]^T$.

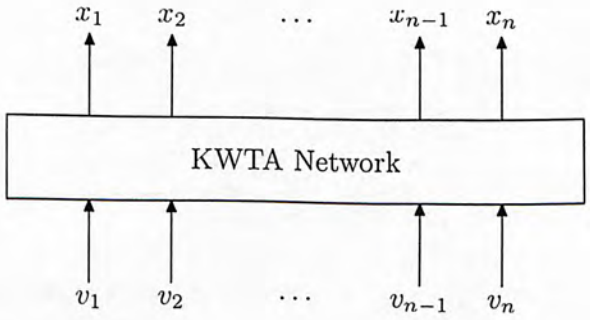


Figure 4.1: The diagram of KWTA operation.

Proof $e^T x = k$ can be written as

$$\sum_{i=1}^{n} x_i = k \tag{4.3}$$

Because $x_i \in \{0, 1\}$, we have

$$x_i^2 = x_i \tag{4.4}$$

From (4.3) and (4.4), we get

$$\sum_{i=1}^{n} x_i^2 = k$$

i.e.,

$$ax^T x = a\sum_{i=1}^{n} x_i^2 = ak$$

is a constant.

So the cost function can be rewritten as

$$\text{maximize} \quad v^T x$$

Suppose that the solution of (4.2) x^* is not the solution of (4.1). Without loss of generality, we assume that $x_i^* = 0$, v_i in the top k largest inputs; And $x_l^* = 1$, v_l is not in the top k largest inputs.

Because v_i is in the top kth largest inputs, and v_l is not, then

$$v_i > v_l$$

Define $\bar{x}_i := 1$, $\bar{x}_l := 0$, $\bar{x}_j := x_j^*, j \neq i, l$. $\bar{x} := [\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n]^T$ also satisfies

the constraints.

$$\begin{aligned} v^T\bar{x} &= v_i\bar{x}_i + v_l\bar{v}_l + \textstyle\sum_{j\neq i,l} v_j\bar{x}_j \\ &= v_i + \textstyle\sum_{j\neq i,l} v_j x_j^* \\ &> v_k + \textstyle\sum_{j\neq i,l} v_j x_j^* \\ &= v_i x_i^* + v_l x_l^* + \textstyle\sum_{j\neq i,l} v_j x_j^* \\ &= v^T x^* \end{aligned}$$

This contradicts with the assumption that x^* is the solution of (4.2).

The proof is complete. □

Denote the kth largest element as $\tilde{v}_k$, $(k+1)$th largest element as $\tilde{v}_{k+1}$. Then we have the following theorem:

Theorem 4 *If $\tilde{v}_k - \tilde{v}_{k+1} \geq 2a$, then the discrete quadratic programming problem (4.2) and the following continuous quadratic programming problem (4.5) have the same solution.*

$$\begin{aligned} &\textit{minimize} && ax^Tx - v^Tx \\ &\textit{subject to} && e^Tx = k \\ & && x_i \in [0,1], \quad i = 1, 2, \cdots, n \end{aligned} \tag{4.5}$$

where a is a positive constant.

Proof If we can show that the solution of the problem (4.5) is in the set $\{0,1\}^n$, then the theorem is proved.

From the equality constraint $e^Tx = k$, we can get

$$x_l = k - \sum_{j\neq l} x_j \tag{4.6}$$

Substituting (4.6) into $ax^Tx - v^Tx$, we have

$$\begin{aligned} & ax^Tx - v^Tx \\ = \;& a(\textstyle\sum_{j=1}^n x_j^2) - \sum_{j=1}^n v_j x_j \\ = \;& a(\textstyle\sum_{j\neq l} x_j^2 + (k - \sum_{j\neq l} x_j)^2) - \\ & \textstyle\sum_{j\neq l} v_j x_j - v_l(k - \sum_{j\neq l} x_j) \\ = \;& a(x_i^2 + \cdots + x_i^2 - 2kx_i + 2x_i \textstyle\sum_{j\neq l,i} x_j + \cdots) - \\ & v_i x_i + v_l x_l + \cdots \\ = \;& 2ax_i^2 + (2a \textstyle\sum_{j\neq l,i} x_j - 2ak + v_l - v_i)x_i + \cdots \end{aligned}$$

From the above derivation, we can take $ax^Tx - v^Tx$ as the function of variable x_i. It can be further written as

$$ax^Tx - v^Tx = 2ax_i^2 + (2a\sum_{j\neq l,i} x_j - 2ak + v_l - v_i)x_i + e(x), \tag{4.7}$$

where $e(x)$ has nothing to do with x_i and x_l.

If the following condition is satisfied, $ax^Tx - v^Tx$ can only reach its minimum at its boundary, that is $x_i = 0$ or $x_i = 1$.

$$\frac{v_i - v_l + 2a(k - \sum_{j\neq l,i} x_j)}{2a} \in (-\infty, 0] \cup [1, \infty).$$

Considering $k - \sum_{j\neq l,i} x_j = x_i + x_l$, the condition is equivalent to

$$\frac{v_i - v_l + 2a(x_i + x_l)}{4a} \in (-\infty, 0] \cup [1, \infty). \tag{4.8}$$

If $v_i \in \{k$ largest elements of inputs$\}$, then we can choose $v_l \notin \{k$ largest elements of inputs$\}$. To let $x_i = 1$ and $x_l = 0$, the following conditions should be satisfied.

$$\frac{v_i - v_l + 2a(x_i + x_l)}{4a} \geq 1,$$

and

$$\frac{v_l - v_i + 2a(x_l + x_i)}{4a} \leq 0.$$

We can get

$$x_i - x_l \geq 2a.$$

If $v_i \notin \{k$ largest elements of inputs$\}$, we can choose $v_l \in \{k$ largest elements of inputs$\}$. In the same way, we can get

$$x_l - x_i \geq 2a.$$

So, if $\tilde{v}_k - \tilde{v}_{k+1} \geq 2a$, the solution of problem (4.5) is in the set $\{0, 1\}^n$. The proof is complete. □

From Theorems 3 and 4, we can easily get that if $\tilde{v}_k - \tilde{v}_{k+1} \geq 2a$, the solution to the continuous quadratic optimization problem (4.5) is the solution of (4.1).

Remark 2 Let's consider the solution of problem (4.5) when $\tilde{v}_k - \tilde{v}_{k+1} < 2a$. In this case

$$\frac{\tilde{v}_k - \tilde{v}_{k+1} + 2a(\tilde{x}_k + \tilde{x}_{k+1})}{4a} \in [0, 1].$$

Then when

$$\tilde{x}_k = \frac{\tilde{v}_k - \tilde{v}_{k+1} + 2a(\tilde{x}_k + \tilde{x}_{k+1})}{4a},$$

the cost function reaches its minimum. That is,

$$\tilde{x}_k - \tilde{x}_{k+1} = \frac{\tilde{v}_k - \tilde{v}_{k+1}}{2a}. \tag{4.9}$$

If other elements can be successfully separated, then

$$\tilde{x}_k + \tilde{x}_{k+1} = 1. \tag{4.10}$$

From (4.9) and (4.10), we can get

$$\tilde{x}_k = 0.5 + \frac{\tilde{v}_k - \tilde{v}_{k+1}}{4a}, \quad \tilde{x}_{k+1} = 0.5 - \frac{\tilde{v}_k - \tilde{v}_{k+1}}{4a}.$$

In particular, if $\tilde{v}_k = \tilde{v}_{k+1}$, $\tilde{x}_k = \tilde{x}_{k+1} = 0.5$.

4.3 KWTA Network Model

In [32][49], a neural network called dual neural network is presented to solve convex quadratic problems utilizing the dual variables. In this section, we will simplify the dual neural network for solving the quadratic programming problem (4.5). Here we call the network as KWTA network. It reduces the architecture complexity while preserving the desirable convergence property compared with the dual neural network.

Consider the problem (4.5) as the primal problem P, then its dual problem D can be written as

$$\begin{aligned} \text{maximize} \quad & k^T y - a x^T x - e^T w_2 \\ \text{subject to} \quad & ax - v - ey - w_1 + w_2 = 0 \\ & u \geqslant 0 \end{aligned} \tag{4.11}$$

where $y \in R, w_1 \in R^n, w_2 \in R^n$ are dual variables.

Define $u = w_1 - w_2$, the equality constraints in (4.11) becomes

$$ax - v - ey - u = 0.$$

By the Karush-Kuhn-Tucker (KKT) conditions for convex optimization [1], the following set of equations have the same solution as problem P

$$ax - v - ey - Iu = 0 \tag{4.12}$$

$$e^T x = k \tag{4.13}$$

$$\begin{cases} x_i = 0 & \text{if } u_i > 0 \\ x_i = 1 & \text{if } u_i < 0 \\ 0 \leqslant x_i \leqslant 1 & \text{if } u_i = 0 \end{cases} \tag{4.14}$$

(4.14) can be rewritten as

$$x = g(x - u), \tag{4.15}$$

where

$$g(v_i) = \begin{cases} 0 & \text{if } v_i < 0 \\ v_i & \text{if } 0 \leqslant v_i \leqslant 1 \\ 1 & \text{if } v_i > 1. \end{cases}$$

From (4.12),

$$x = \frac{1}{a}(ey + u + v) \tag{4.16}$$

Substitute (4.16) into (4.13),

$$\frac{1}{a}e^T(ey + u + v) = k$$

y can be explicitly expressed by u.

$$y = \frac{1}{n}(ak - e^T u - e^T v) \tag{4.17}$$

Substitute (4.17) into (4.16),

$$x = \frac{1}{a}(I - \frac{1}{n}ee^T)u + \frac{1}{a}(I - \frac{1}{n}ee^T)v + \frac{ek}{n} \tag{4.18}$$

Based on (4.15, 4.16, 4.17), by projection theorem, the neural network that can solve the original problem can be designed as

$$\begin{aligned} \epsilon\tfrac{du}{dt} &= -Mu + g(Mu - u + s) + s \\ x &= Mu + s \end{aligned} \tag{4.19}$$

where $M = (I - ee^T/n)/a, s = Mv + (k/n)e$.

The architecture of the KWTA network is shown in Fig 4.2. A circuit implementing this network consists of summers, integrators and operational amplifiers.

The properties of convergence and optimality are studied below.

Theorem 5 *The KWTA network (4.19) is globally convergent to an equilibrium point u^* which depends on the initial state of the trajectory.*

Proof At u^*, we have the following inequality

$$(v - x^*)^T u^* \geq 0, \forall v \in \Omega. \tag{4.20}$$

which can be obtained by considering the following three cases:

- Case 1: If for some $i \in \{1, 2, \cdots, p\}$, $u_i^* = 0$, $0 \leq x_i^* \leq 1$, then $(v_i - x_i^*)u_i^* = 0$;

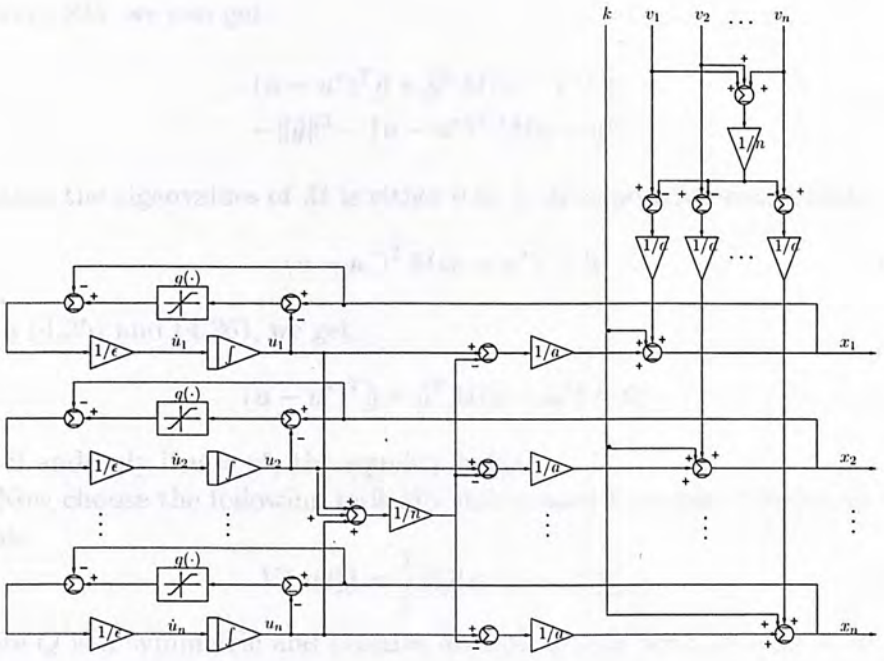


Figure 4.2: Architecture of the KWTA network.

- Case 2: If for some $j \in \{1, 2, \cdots, p\}$, $u_j^* > 0$, $x_j^* = 0$ and $0 \leq v_j \leq 1$, then $v_j - x_j^* \geq 0$ and thus $(v_j - x_j^*)u_j^* \geq 0$;
- Case 3: If for some $k \in \{1, 2, \cdots, p\}$, $u_k^* < 0$, $x_k^* = 1$ and $0 \leq v_k \leq 1$, then $v_k - x_k^* \leq 0$ and thus $(v_k - x_k^*)u_k^* \geq 0$.

Therefore, it follows from (4.20) that

$$[g(Mu + s - u) - (Mu^* + s)]^T u^* \geq 0. \tag{4.21}$$

On the other hand, from projection theorem [2], it follows that, $\forall u \in R^n$

$$\begin{aligned} &[g(Mu + s - u) - (Mu^* + s)]^T \\ &[(Mu + s - u) - g(Mu + s - u)] \geq 0. \end{aligned} \tag{4.22}$$

Combining (4.21) and (4.22), we have

$$\begin{aligned} &[g(Mu + s - u) - (Mu^* + s)]^T \\ &[u^* + (Mu + s - u) - g(Mu + s - u)] \geq 0. \end{aligned} \tag{4.23}$$

Define $\tilde{g} := g(Mu + s - u) - (Mu + s)$, (4.23) becomes

$$[\tilde{g} + M(u - u^*)]^T [(u - u^*) + \tilde{g}] \leq 0. \tag{4.24}$$

From (4.24), we can get

$$\begin{aligned}(u-u^*)^T\tilde{g}+\tilde{g}^TM(u-u^*)\leq\\ -\|\tilde{g}\|^2-(u-u^*)^TM(u-u^*).\end{aligned} \tag{4.25}$$

Because the eigenvalues of M is either 0 or 1, M is positive semidefinite, i.e.,

$$(u-u^*)^TM(u-u^*)\geq 0. \tag{4.26}$$

From (4.25) and (4.26), we get

$$(u-u^*)^T\tilde{g}+\tilde{g}^TM(u-u^*)\leq 0, \tag{4.27}$$

and if and only if $u=u^*$, the equality holds.

Now choose the following radically unbounded Lyapunov functional candidate

$$V(u(t))=\frac{1}{2}\|Q(u(t)-u^*)\|_2^2, \tag{4.28}$$

where Q is a symmetric and positive definite matrix with $Q^2=(I+M)$.

Then from (4.27), we get

$$\begin{aligned}\tfrac{dV}{dt}&=(u-u^*)^TQ^2\dot{u}\\ &=(u-u^*)^T(I+M)\tilde{g}\\ &=(u-u^*)^T\tilde{g}+\tilde{g}^TM(u-u^*)\\ &\leq 0.\end{aligned} \tag{4.29}$$

By the Lyapunov stability theorem, the simplified dual neural network is globally stable. □

Remark 3 The convergence speed of the neural network is determined by the eigenvalues of M which are independent of n and inversely proportional to a.

Theorem 6 *$x^*=Mu^*+s$ is an optimal solution to the quadratic programming problem (4.5), where u^* is an equilibrium point of the dynamic equation (4.19).*

This theorem can be verified by substituting x^* into the KKT conditions (4.12,4.13,4.14). The three equations are satisfied, which means that x^* is the optimal solution to the quadratic programming problem (4.5).

From the Theorems 5 and 6, we can conclude that the KWTA network is globally convergent to the exact solution of the problem (4.5). Further, from

Theorems 3 and 4, the KWTA network is globally convergent to the exact solution of the KWTA operation. As a comparison, the KWTA circuit in [62] will oscillate under some conditions.

From the convergence study of the simplified dual neural network, we can get the result that the KWTA network is also globally Lyapunov stable and globally convergent to the solution of KWTA operation with resolution $2a$. Moreover, because the convergence speed of the KWTA network is dominated by the linear term of (4.19) (i.e., eigenvalues of M), which is independent of number of inputs n, the convergence speed is independent of the problem scale. In addition, the eigenvalues of M are inversely proportional to a, so the convergence speed is also inversly proportional to a.

4.4 Simulation Results

Example 1: First a static KWTA problem is tested. The inputs are $v_i = i$ $(i = 1, 2, \cdots, n)$, $k = 2$ and $\epsilon = 10^{-8}$. When $n = 5$, and $a = 0.25$, the transient behaviors of u and v are shown in Figs. 4.3 and 4.4. In Fig. 4.4, the curves from bottom to top correspond respectively v_1, v_2, $\cdots$, v_5. It can be seen that the outputs are [0 0 0 1 1]. The 2 largest elements are successfully selected within 5×10^{-8} second. Figs. 4.5 and 4.6 show the relationship between convergence rate and the parameters. In Fig. 4.5, it can be observed that when the parameter a increases exponentially, the convergence time also increases exponentially (note that the horizontal axis is in log scale). On the contrary, the convergence rate remains steady with respect to the changing of problem scale n, which can be observed in Fig. 4.6.

Next, consider a set of 5 sinusoidal input signals with the following instantaneous values $v_p(t) = 10\sin[2\pi(1000t - 0.25\pi(p-1)](p = 1, 2, 3, 4)$ and $k = 2$. Fig. 4.7 illustrates the 5 input signals and the transient outputs of the KWTA network with $\epsilon = 10^{-6}$ and $a = 10^{-3}$. The simulation results show that the KWTA network can effectively determine the two largest signals from the time-varying signals in real time.

4.5 Conclusions

A KWTA network is developed for K-winners-take-all operation based on the equivalent quadratic programming formulation. The KWTA network is shown to be stable and can implement the KWTA operation in real time. Compared with the dual neural network, KWTA network has lower architecture complexity. The KWTA network is shown to be effective by analysis

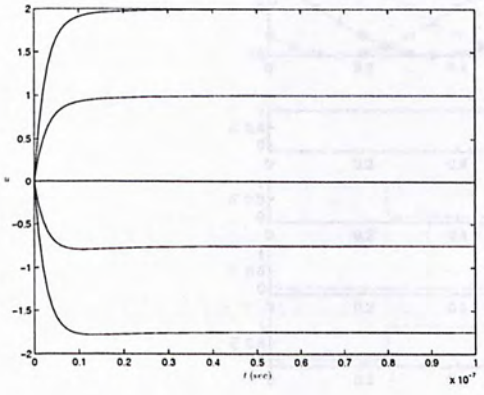


Figure 4.3: The transient behavior of u.

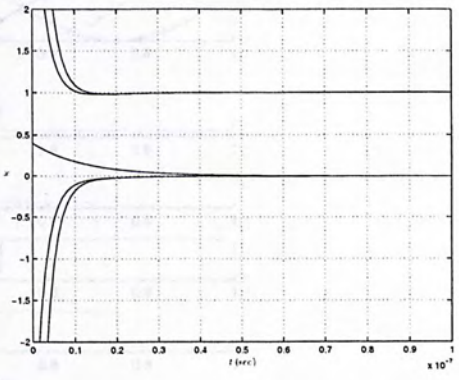


Figure 4.4: The transient behavior of x.

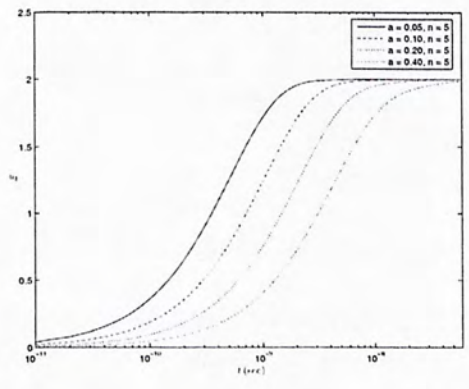


Figure 4.5: Convergence behavior of the KWTA network with respect to different a in Example 1

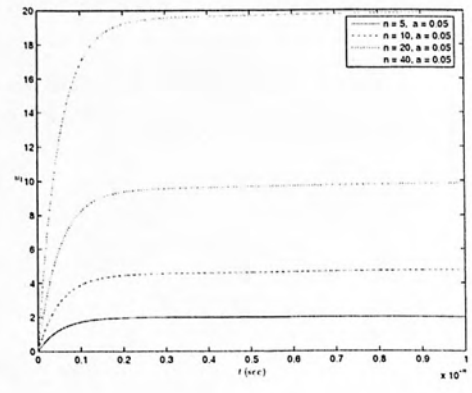


Figure 4.6: Convergence behavior of the KWTA network with respect to different n in Example 1

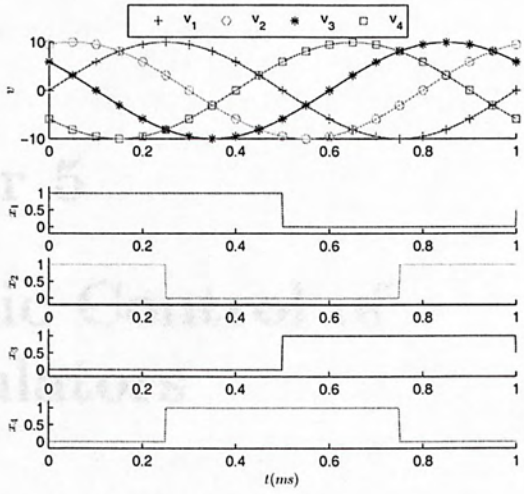


Figure 4.7: Inputs and outputs of the KWTA network.

and simulation.

□ **End of chapter.**

Chapter 5

Dynamic Control of Manipulators

Summary

The bi-criteria joint torque optimization of kinematically redundant manipulators balances between the energy consumption and the torque distribution among the joints. In this chapter, a simplified dual neural network is proposed to solve this problem. Joint torque limits are incorporated simultaneously into the proposed optimization scheme. The simplified dual network has less numbers of neurons compared with other recurrent neural networks and is proved to be globally convergent to optimal solutions. The control scheme based on the recurrent neural network is simulated with the PUMA 560 robot manipulator to demonstrate effectiveness.

5.1 Introduction

Kinematically redundant manipulators are those having more degrees of freedom (DOFs) than required to perform a given task. The redundancy of such manipulators includes intrinsic redundancy and functional redundancy, which can be utilized to optimize various performance criteria, while performing the given motion task. Among performance criteria, the optimization of joint torques is an appealing one since it is equivalent to effective utilization of actuator powers. An initial study was conducted by Hollerbach ans Suh [70] who presented the null-space algorithm that instantaneously minimizes the joint torque. Neungadi and Kazerounian [71] presented an approach that

locally minimizes joint torques weighted by the inverse of inertia matrix. This optimization criterion corresponds to global kinetic energy minimization, and the local solutions are thus optimal and internally stable. But in practice, because of the large computational load of computing inverse matrix, the optimization function is often simplified as the 2-norm of the torque. The minimum-effort solution was also proposed to explicitly minimize the largest torque. This solution is consistent with physical limits and enables a better direct monitoring and control of the magnitude of individual joint torques than other norms of joint torques [72]. It is thus more desirable in applications where low individual joint torques is of primary concern. Recently, neural network approaches have been developed for the optimization of redundant manipulators. Tang and Wang [45] and Zhang and Wang [73] have proposed several recurrent neural networks (Lagrangian, primal-dual and dual neural network) for the torque optimization. In [74], a dual neural network was presented by Zhang *et al.* for kinematic control for redundant manipulators by minimizing the bi-criteria of Euclidean and infinity norm of joint velocities. In [75], the results have been extended to minimize both the weighted norm and infinity norm of the joint torques to get a balance between the total energy consumption and the joint torque distribution, and at the same time take into account of the joint torque limits. In this chapter, the neural network structure is simplified, while preserving the global convergence property.

5.2 Problem Formulation

Consider the forward kinematics relations between the joint variables and the poses of the end-effector in Cartesian space

$$r = f(\theta) \tag{5.1}$$

where $\theta \in R^n$ is the joint variable vector; $r \in R^m$ is the position and orientation vector of the end-effector in the Cartesian space ($m < n$ in redundant manipulators); $f(\cdot)$ is a smooth nonlinear mapping function, known for a given manipulator.

Differentiating (5.1) with respect to time gives the linear relation between the Cartesian velocity $\dot{r}$ and joint velocity $\dot{\theta}$

$$\dot{r} = J(\theta)\dot{\theta} \tag{5.2}$$

where $J(\theta) \in R^{m \times n}$ is the Jacobian matrix defined as $J(\theta) = \partial f(\theta)/\partial\theta$.

Differentiating (5.2) with respect to time yields the relation between the joint acceleration $\ddot{\theta}$ and Cartesian acceleration $\ddot{r}$

$$J(\theta)\ddot{\theta} = \ddot{r} - \dot{J}(\theta)\dot{\theta} \tag{5.3}$$

where $\dot{J}(\theta) \in R^{m\times n}$ is the time derivative of the Jacobian matrix. In a redundant manipulator, (5.1) and (5.2) are underdetermined since $m < n$ and hence they may admit an infinite number of solutions.

It is well known that the revolute joint robot dynamics is

$$\tau = H(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) \tag{5.4}$$

where $H(\theta) \in R^{n\times n}$ is the symmetric positive definite inertia matrix; $c(\theta, \dot{\theta}) \in R^n$ is the component of the torque depending on Coriolis centrifugal forces; $g(\theta) \in R^n$ is the component depending on gravity forces.

The 2-norm optimization of joint torques can be formulated as

$$\begin{array}{ll} \text{minimize} & \tau^T\tau, \\ \text{subject to} & J(\theta)\ddot{\theta} + \dot{J}(\theta)\dot{\theta} - \ddot{r} = 0. \end{array} \tag{5.5}$$

Inverting (5.4), we have the joint acceleration for given joint torques $\ddot{\theta} = H^{-1}(\tau - c - g)$. Substituting it into (5.3), the joint torque can be expressed in terms of $\ddot{r}$ as

$$JH^{-1}\tau = JH^{-1}(c + g) + \ddot{r} - \dot{J}\dot{\theta}. \tag{5.6}$$

Equation (5.6) can be simplified by introducing two terms $\ddot{r}_\tau = JH^{-1}(c + g) + \ddot{r} - \dot{J}\dot{\theta}$ and $J_\tau = JH^{-1}$. Hence, we have a linear torque-based constraint

$$J_\tau\tau = \ddot{r}_\tau. \tag{5.7}$$

The inertia inverse weighted joint torque optimization problem (5.5) can thus be reformulated to a time-varying quadratic program subject to the linear torque-based constraints (5.7) as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\tau^T\tau, \\ \text{subject to} & J_\tau\tau = \ddot{r}_\tau. \end{array} \tag{5.8}$$

Further incorporating the infinity-norm optimization and joint torque limits, the bi-criteria torque optimization can be formulated as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(\alpha\tau^T\tau + (1 - \alpha)||\tau||_\infty^2), \\ \text{subject to} & J_\tau\tau = \ddot{r}_\tau, \\ & \tau^- \le \tau \le \tau^+. \end{array} \tag{5.9}$$

where $||\tau||_\infty$ denotes the infinity norm, $\alpha \in (0, 1)$ the weight coefficient. τ^- and τ^+ denote respectively upper and lower limits of torque.

Next, let us convert the minimum infinity-norm part of (5.9) into a quadratic program. By defining $s = ||\tau||_\infty$, the minimization of $(1-\alpha)||\tau||_\infty^2$ can be rewritten equivalently [74] as

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}(1-\alpha)s^2, \\ &\text{subject to} && \begin{bmatrix} I & -e \\ -I & -e \end{bmatrix} \begin{bmatrix} \tau \\ s \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{5.10}$$

where $e := [1, 1, \ldots, 1]^T$ and $0 := [0, 0, \ldots, 0]^T$ are vectors, respectively, of ones and zeros with appropriate dimensions, I is the identity matrix.

Thus, by defining the variable vector $x = [\tau, s]^T \in R^{n+1}$, the bi-criteria torque optimization problem (5.9) can be expressed as the following quadratic program:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}x^T W x, \\ &\text{subject to} && Ax = b, \\ & && Cx \leq d, \\ & && x^- \leq x \leq x^+, \end{aligned} \tag{5.11}$$

where the coefficient matrices and vectors are

$$\begin{aligned} W &:= \begin{bmatrix} \alpha I & 0 \\ 0 & (1-\alpha) \end{bmatrix} \in R^{(n+1)\times(n+1)}, \\ A &:= \begin{bmatrix} J_\tau & 0 \end{bmatrix} \in R^{m\times(n+1)}, \\ b &:= \ddot{r}_\tau \in R^m, \\ C &:= \begin{bmatrix} I & -e \\ -I & -e \end{bmatrix} \in R^{2n\times(n+1)}, \\ d &:= 0 \in R^{2n}, \\ x^- &:= \begin{bmatrix} \tau^- \\ 0 \end{bmatrix}, \\ x^+ &:= \begin{bmatrix} \tau^+ \\ \max_{1\leq j\leq n} |\tau_j^\pm| \end{bmatrix} \in R^{n+1}. \end{aligned}$$

Since the objective function in the formulation (5.11) is strictly convex (due to $0 < \alpha < 1$ and W is positive definite) and the feasible region of linear constraints is a closed convex set, the solution to the bi-criteria quadratic program (5.9) is unique and satisfies the Karush-Kuhn-Tucker optimality conditions. Hence the continuity of the bi-criteria solution is guaranteed. As $\alpha \to 0$, the bi-criteria solution reaches the infinity-norm solution and $\alpha \to 1$, the bi-criteria solution becomes the 2-norm solution, which illustrates that

the proposed bi-criteria optimization scheme is much more flexible than a single-criterion optimization scheme.

5.3 Simplified Dual Neural Network

A number of numerical algorithms have been proposed to solve the time-varying quadratic problem (5.11). But the computational complexity of the serial algorithms may limit their applications in this online dynamic optimization application.

In the past two decades, the parallel computing capability of recurrent neural networks, known as computing with attractors, is recognized by researchers in many disciplines. The essence of the computing with attractors is that a neuro-dynamic system can be designed to globally convergent to the solutions of the desired problems [76]. And the neuro-dynamic systems can be implemented by the electronic circuits such as analog and/or digital VLSI circuits. The superior property of this computing paradigm is that it can tackle large-scale and time-varying problems.

Over years, various neural network models have been developed for solving quadratic programming problems. According to their design methods, these neural networks can be categorized as the penalty-parameter neural network [28], the Lagrange neural network [29], the primal-dual neural network [31][52] and the dual neural network [32][49]. As a much flexible tool for exactly solving quadratic programming problems, the primal-dual neural network [31] was developed with the feature that it handles the primal quadratic program and its dual problem simultaneously by minimizing the duality gap and using the gradient method. Unfortunately, the dynamic equations of the primal-dual neural network are usually complicated, and may contain high-order nonlinear terms. Moreover, the network size is usually larger than or equal to the dimensionality of the primal quadratic program and its dual problem. In [32][49], a neural network called the dual neural network is presented to solve convex quadratic problems utilizing only the dual variables. In this section, we will give a new neural network called simplified dual neural network based on dual neural network for solving the QP problem (5.11). It can further reduce the architecture complexity while preserving the desirable convergence property.

The inequality constraints can be unified in more consistent forms. Define

$$F := \begin{bmatrix} C \\ I \end{bmatrix}, \ \xi^- := \begin{bmatrix} -\infty \\ x^- \end{bmatrix}, \ \xi^+ := \begin{bmatrix} d \\ x^+ \end{bmatrix},$$

then the inequalities evolve to this form

$$\xi^- \leq Fx \leq \xi^+.$$

Based on the above transformation, (5.11) can be reformulated as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T W x, \\ \text{subject to} & Ax = b, \\ & \xi^- \leq Fx \leq \xi^+, \end{array} \tag{5.12}$$

where $F \in R^{(3n+1)\times(n+1)}$, $\xi^- \in R^{(3n+1)}$, $\xi^+ \in R^{(3n+1)}$.

Hereafter, we will concentrate on formulation (5.12).

Consider problem (5.12) as the primal problem P. Its dual problem D can be written as

$$\begin{array}{ll} \text{maximize} & b^T y - \frac{1}{2}x^T W x + \xi^{-T} v - \xi^{+T} w, \\ \text{subject to} & Wx - A^T y - F^T v + F^T w = 0, \\ & v \geq 0, \quad w \geq 0, \end{array} \tag{5.13}$$

where $y \in R^m, v \in R^p, w \in R^p$ are dual variables.

Define $u = v - w$, the equality constraints in (5.13) becomes

$$Wx - A^T y - F^T u = 0.$$

According to the Karush-Kuhn-Tucker (KKT) conditions for convex optimization [1], the following set of equations have the same solution as problem P

$$\begin{array}{l} Wx - A^T y - F^T u = 0, \\ Ax = b, \\ \begin{cases} (Fx)_i = \xi_i^- & \text{if } u_i > 0, \\ (Fx)_i = \xi_i^+ & \text{if } u_i < 0, \\ \xi_i^- \leq (Fx)_i \leq \xi_i^+ & \text{if } u_i = 0. \end{cases} \end{array}$$

That is,

$$Wx - A^T y - F^T u = 0, \tag{5.14}$$

$$Ax = b, \tag{5.15}$$

$$Fx = g(Fx - u), \tag{5.16}$$

where

$$g(v) = [g_1(v_1), g_2(v_2), \cdots, g_p(v_p)]^T.$$

$$g_i(v_i) = \begin{cases} \xi_i^- & \text{if } v_i < \xi_i^-, \\ v_i & \text{if } \xi_i^- \le v_i \le \xi_i^+, \\ \xi_i^+ & \text{if } v_i > \xi_i^+. \end{cases} \quad i = 1, \cdots, p$$

If $\xi^- = -\infty$, $g(v_i)$ degenerates to

$$g_i(v_i) = \begin{cases} v_i & \text{if } v_i \le \xi_i^+, \\ \xi_i^+ & \text{otherwise.} \end{cases} \quad i = 1, \cdots, p$$

From equation (5.14), because W is invertible,

$$x = W^{-1}(A^T y + F^T u). \tag{5.17}$$

Substituting (5.17) into (5.15), we have

$$AW^{-1}(A^T y + F^T u) = b.$$

As $\text{rank}(A) = m$ and W is invertible, $AW^{-1}A^T$ is invertible. Then y can be explicitly expressed by u.

$$y = (AW^{-1}A^T)^{-1}\left[(-AW^{-1}F^T)u + b\right] \tag{5.18}$$

Based on (5.16), (5.17), (5.18), by the projection theorem, the dynamic equation of the neural network for solving the primal problem (5.12) can be designed as

- State equation

$$\epsilon \frac{du}{dt} = -Fx + g(Fx - u), \tag{5.19}$$

- Output equation

$$\begin{aligned} x &= W^{-1}(A^T y + F^T u), \\ y &= (AW^{-1}A^T)^{-1}\left[(-AW^{-1}F^T)u + b\right], \end{aligned} \tag{5.20}$$

where $u \in R^p$ is the state vector, $\epsilon > 0$ is a scaling parameter that controls the convergence rate of the neural network. In implementation, ϵ should be set as small as the hardware permits. The recurrent neural network can be rewritten in the more compact and explicit form as

- State equation

$$\epsilon \frac{du}{dt} = -FMF^T u + g(FMF^T u - u + Fs) - Fs, \tag{5.21}$$

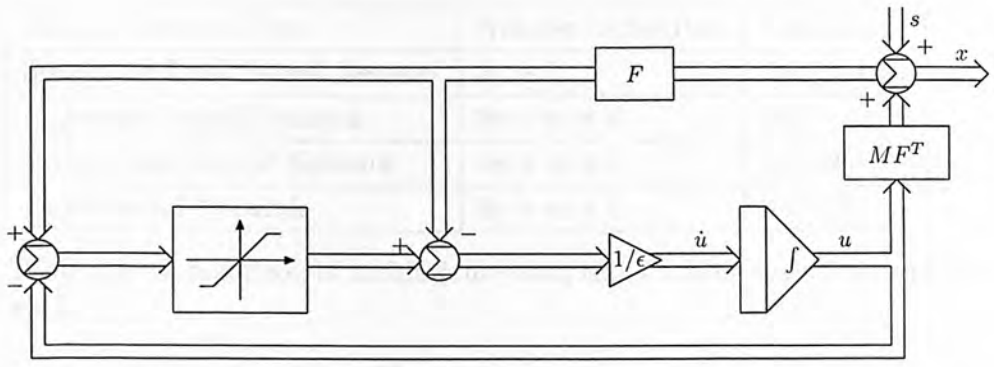


Figure 5.1: Block diagram of the simplified dual neural network.

- Output equation

$$x = MF^T u + s, \tag{5.22}$$

where $M := W^{-1}(I - A^T G)$, $s := W^{-1}(A^T h)$, $G := (AW^{-1}A^T)^{-1}(AW^{-1})$, $h := (AW^{-1}A^T)^{-1}b$. Because the analytic expression of A, W, F can be obtained in the design stage, the analytic expressions of M and s can be computed beforehand. In view of this, though expression (5.21) appears complicated, it is not computationally complex. The block diagram of the neural network is shown in Fig. 5.1.

As a comparison, the original dual neural network [32] [49] for solving problem (5.12) is

$$\begin{aligned} \epsilon \tfrac{du}{dt} &= -EW^{-1}E^T u + g(EW^{-1}E^T u - u), \\ x &= W^{-1}E^T u, \end{aligned} \tag{5.23}$$

where

$$E := \begin{bmatrix} A \\ C \\ I \end{bmatrix} \in R^{(3n+m+1)\times(n+1)}.$$

In the simplified dual neural network, the number of neurons is equal to the number of inequality constraints, whereas the number of neurons is equal to the number of equality and inequality constraints in the original dual neural network (5.23). A complete comparison with several existing neural networks for solving the quadratic program problem (5.11) is shown in Table 5.1. The recurrent neural network (5.19) can be proven to be stable in the sense of Lyapunov and asymptotically globally convergent to the exact optimal solution of the quadratic programming problem.

Neural Network Type	Number of Neurons	References
Simplified Dual Neural Network	$3n+1$	this chapter
Lagrange Neural Network	$9n+m+4$	[29]
Prime-Dual Neural Network	$5n+m+1$	[31][52]
Dual Neural Network	$3n+m+1$	[74][75]

Table 5.1: Comparison of architecture complexity among various neural networks

5.4 Simulation Results

The Unimation PUMA560 manipulator has six joints. When the pose of the end-effector is considered, PUMA560 is not a redundant manipulator. However, if we consider only the position of the end-effector, PUMA560 becomes a redundant manipulator with the associated Jacobian matrix $J(\theta) \in R^{3\times 6}$.

In this section, we discuss the bi-criteria torque optimization of the PUMA 560 when its end-effector tracks circular paths, by means of the proposed recurrent neural network.

The desired motion of the end-effector is a circle of radius $r = 10cm$ with the revolute angle about the x axis $\pi/6$. The task time of the motion is $10s$ and the initial joint variables $\theta(0) = [0\ 0\ 0\ 0\ 0\ 0]^T$. Fig. 5.4 illustrates the simulated motion of the PUMA 560 manipulator in the 3D workspace, which is sufficiently close to the desired one. Fig. 5.11 shows the end effector position error more clearly. From this figure, we can see that the error is within 0.025mm. Fig. 5.5, 5.6 and 5.7 illustrate respectively the transient behaviors of joint variables, joint velocities an joint accelearations. Figs. 5.8 and 5.9 show the infinity norm and the 2-norm of joint torques while moving when $\alpha = 0.5, 0.99, 0.01$ respectively. As described in Section 5.2, when $\alpha = 0.01$, the bi-criteria solution is approximate to the infinity-norm solution; while $\alpha = 0.99$, the bi-criteria solution becomes nearly the inertia 2-norm solution. From Figs. 5.8 and 5.9, we can see that the bi-criteria solution always make a balance between the infinity-norm solution and the weighted-norm solution. In view of discontinuity of pure infinity-norm solution, the bi-criteria solution is smooth, which implies no sudden change of torque, which can be observed from Fig. 5.10. Hence, compared with single-criterion torque optimization, the bi-criteria scheme based on the simplified dual neural network is much more flexible in the sense that it can yield any combination of the minimum-effort and minimum-power solutions as needed; at the same time eliminate the discontinuity of minimum-effort solution.

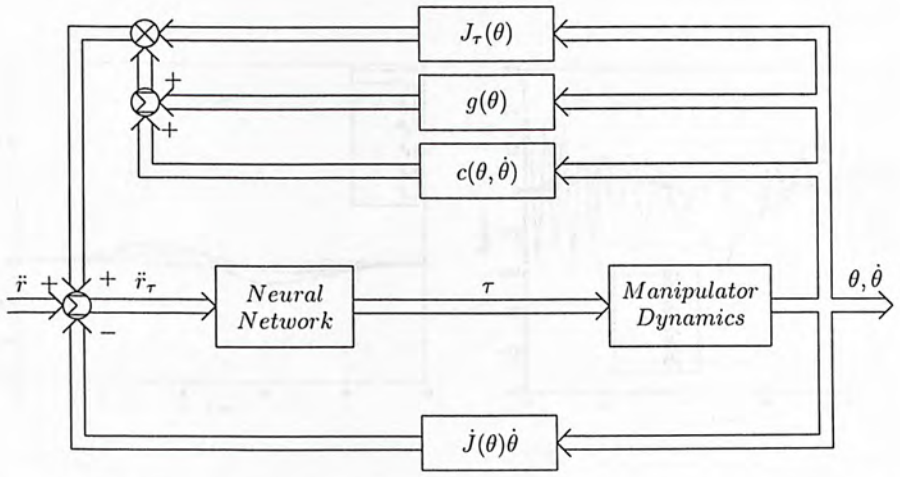


Figure 5.2: Block diagram of neural network based torque optimization of manipulators

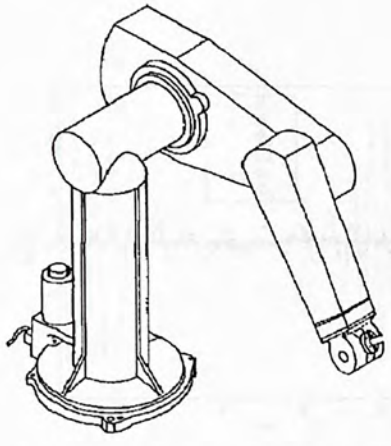


Figure 5.3: PUMA560 robot manipulator

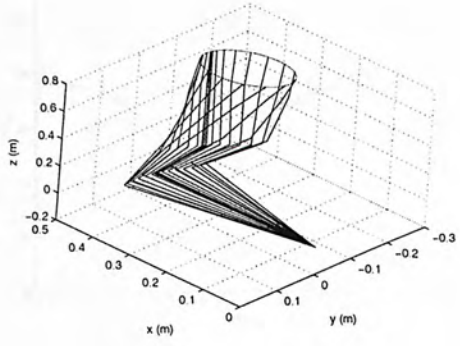


Figure 5.4: Motion trajectory of PUMA560 manipulator while tracking a circle

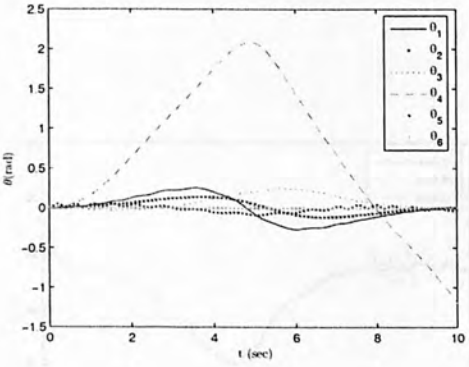


Figure 5.5: Transient behaviors of joint variables

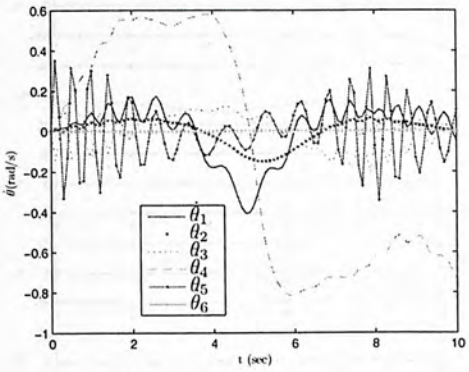


Figure 5.6: Transient behaviors of joint velocities

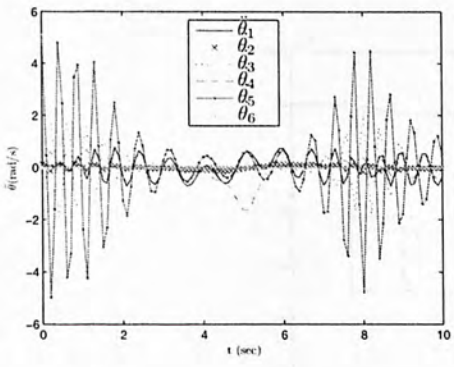


Figure 5.7: Transient behaviors of joint accelerations

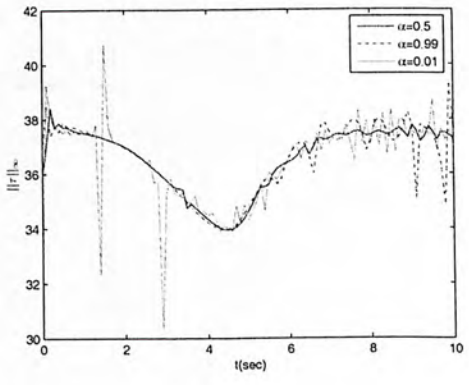


Figure 5.8: Infinity norm of torques when $\alpha = 0.5, 0.99, 0.01$

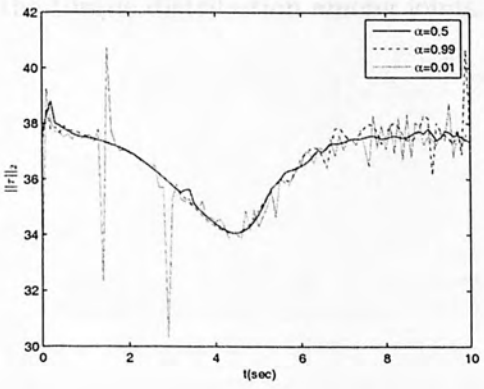


Figure 5.9: 2-norm of torques when $\alpha = 0.5, 0.99, 0.01$

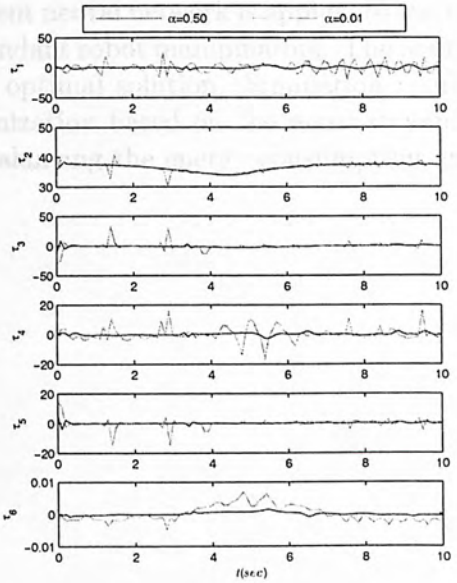


Figure 5.10: Joint torques when $\alpha = 0.5$ and 0.01

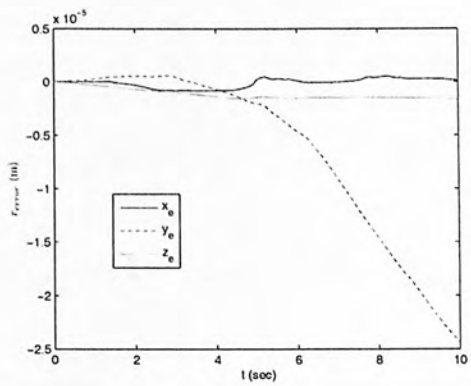


Figure 5.11: Position error of the end effector while tracking the circle

5.5 Concluding Remarks

In this chapter, a simplified dual recurrent neural network is applied to the bi-criteria torque optimization of the redundant robot manipulators. The neural network is globally convergent to the optimal solution. Simulation results show that the bi-criteria torque optimization based on the neuro-dynamic approach is effective and efficient in balancing the energy consumption and the torque distribution among joints.

□ **End of chapter.**

Chapter 6

Robot Arm Obstacle Avoidance

summary

With the wide deployment of redundant manipulators in complex working environments, obstacle avoidance emerges as an important issue to be addressed in robot motion planning. In this chapter, the obstacle avoidance scheme for redundant manipulators proposed in [50] is improved. In the improved scheme, obstacle avoidance is mathematically formulated as a time-varying inequality constrained quadratic programming problem taking advantage of the manipulators' kinematic redundancy. To solve this time-varying quadratic optimization problem in real time, a recurrent neural network called simplified dual neural netwok is proposed, which has lower structural complexity compared with previously proposed neural networks for solving the problem. The effectivity of the improved scheme and the real time solution capability of the simplified dual neural network is demonstrated through a simulation case on the Mitsubishi PA10-7C manipulator.

6.1 Introduction

As the development of automation industry, robot manipulators are required to work in more and more complex and dynamic environments, where an important issue to be considered is how to effectively avoid the static or moving objects in the workspace of the manipulators. Kinematically redundant manipulators are those having more degrees of freedom than required to perform given end-effector moving tasks. Being dexterous and flexible, they have been

used for avoding obstacles, singularity, and optimizing various performance criteria in addition to tracking desired end-effector trajectories. Of those versatile applications, obstacle avoidance is extremely important for successful motion control when obstacles exist in the workspace. Many studies have been reported on using kinematically redundant manipulators for motion control and obstacle avoidance (e.g., [77][67][68][78][79][69][80][81][82][83]). One class of approaches to obstacle avoidance for redundant manipulators is based on high-level motion planning by means of roadmaps, potential fields, cell decomposition, or mathematical programming. This class of methods is most suitable for avoidance of fixed obstacles in known structured working environments. For obstacle avoidance in uncertain or dynamic environments, low-level local motion control in real time is necessary. The popular methods for real-time obstacle avoidance employ the pseudoinverse for obtaining a general solution at velocity level, which contains a minimum L_2-norm solution and a homogeneous solution. The homogeneous solution is selected such that a secondary goal of obstacle avoidance is achieved while accomplishing the primary goal of trajectory tracking, e.g., [77][67][68][78]. Khatib [79] developed the artificial potential method to control robots in the presence of obstacles. However, the use of this approach is limited due to the existence of local minima and its inability to handle arbitrarily shaped obstacles. Sciavicco and Siciliano [69] augmented the Jacobian matrix to fully constrain the system by including constraints for collision avoidance. This method may induce algorithmic singularity to make solutions infeasible. Guo and Hsia [80] presented a method to optimize the distance between the robot links and obstacles. The intensive computation in the construction of the distance function rules out a real-time application of this method.

The real-time obstacle avoidance problem in robotic motion control is concerned with determining the joint variables of a kinematically redundant manipulator in real time to follow the desired trajectory accurately without any collision with obstacles in the workspace. To perform successful robotic manipulations in dynamic and/or uncertain environments, it is highly desirable to determine the optimal motion of the manipulators in real time. Such a real-time optimization process entails extremely extensive on-line computation. The existing optimization techniques are usually incompetent because of the time-varying nature of the optimization problem. Parallel and distributed approaches to real-time obstacle avoidance are deemed necessary as well as desirable. In the past fifteen years or so, recurrent neural networks for optimization have been widely explored. Reported results of numerous investigations have shown many advantages over the traditional optimization algorithms, especially in real-time applications.

In this chapter, we will develop a recurrent neural network capable of ob-

stacle avoidance for real-time motion planning and control of kinematically redundant manipulators. Compared with supervised learning feedforward neural networks, the proposed recurrent neural network eliminates the need for iterative training. Unlike the recurrent neural networks based on penalty parameters, the proposed recurrent neural networks are able to converge to exact optimal solutions without using any penalty parameters. This chapter addresses the key issues and solve the key problems including the formulation of the real-time obstacle avoidance problem as real-time optimization problems using different objective functions and constraints, the determination of appropriate energy functions and dynamic equations of the recurrent neural network models which underlie the development of the recurrent neural networks, the analysis of neural network dynamics and overall intelligent robotic systems in terms of stability and optimality, the evaluation of system characteristics and performance by means of simulations of the recurrent neural networks for motion control of redundant manipulators.

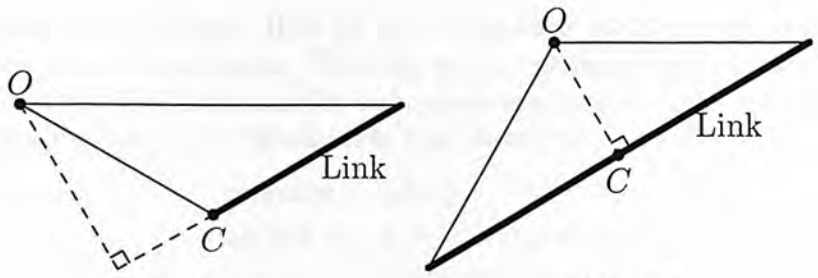
The remainder of this chapter is organized as follows. In Section 6.2, an improved obstacle avoidance scheme is proposed based on the formulation in [77]. The scheme is then compared with the one in [50]. In Section 6.3, we propose a simplified dual neural network for solving the time-varying quadratic optimization formulation derived in Section 6.2. The convergence behavior and the structural complexity of the network is also studied. A simulation case is reported in Section 6.4. At last, Section 6.5 concludes this chapter.

6.2 Obstacle Avoidance Scheme

6.2.1 Equality Constrained Formulation

The obstacle avoidance scheme proposed in this section is based on the algorithm proposed by Maciejewski and Klein [77]. The obstacle avoidance algorithm is to identify from time to time the critical points, which is defined as the points on the manipulator every link which are closest to the obstacles, and then assign them desired velocities which direct the critical point away from the obstacle.

Each critical point C is defined as the point on the link L which is closest to the obstacle point O. As shown in Fig. 6.1, corresponding to different relative positions of the obstacle and the link, there are two possible cases for locating the critical point C on the vulnerable link L. Here the obstacle point O is representative of the obstacle object. In model-based control, the position of the obstacle O is *a priori* available; while in sensor-based control, it is determined by synthetic information of sensor fusion technology, e.g.,



(a) Case 1 (b) Case 2

Figure 6.1: Critical point location

utilizing vision, ultrasonic, and infra-red sensors. The critical point C is thus derived via the online distance minimization between the manipulator link and the obstacle object.

Taken obstacle avoidance into consideration, the forward kinematics of a serial-link manipulator can be described by the following augmented forward kinematics equation:

$$r_e(t) = f_e(\theta(t)), \quad r_c(t) = f_c(\theta(t)) \tag{6.1}$$

where $r_e \in R^m$ is the end-effector's m-dimensional position and/or orientation vector, and $r_c \in R^3$ is the critical point's position vector, $\theta(t)$ is the joint vector of the manipulator, and f_e and f_c are nonlinear functions of the manipulator with respect to respectively the end-effector and the critical point. If there exist more than one critical points, multiple equations, similar to the second one, are present.

The manipulator path planning problem (also called inverse kinematics problem or kinematic control problem) is to find the joint variable $\theta(t)$ for any given $r_e(t)$ and $r_c(t)$ through the inverse mapping of (6.1). Unfortunately, it is usually impossible to find an analytic solution due to the nonlinearity of $f_e(\cdot)$ and $f_c(\cdot)$. The inverse kinematics problem is thus usually solved at the velocity level with the relation

$$J_e(\theta)\dot{\theta} = \dot{r}_e, \quad J_c(\theta)\dot{\theta} = \dot{r}_c, \tag{6.2}$$

where $J_e(\theta) = \partial f_e(\theta)/\partial\theta$ and $J_c(\theta) = \partial f_c(\theta)/\partial\theta$ are the Jacobian matices, and $\dot{r}_e$ is the desired velocity of the end-effector, $\dot{r}_c$ is the desired velocity of the critical point which should be properly selected to effectively direct the link away from the obstacle.

The desired motion of the critical point is regarded as the secondary goal when the minimum distance between the link and the obstacle is shorter than

a specified safety margin. Here we put the obstacle avoidance algorithm into an optimization framework. Then the motion planning and obstacle avoidance problem of a kinematically redundant manipulator can be formulated in a time-varying local optimization framework as:

$$\begin{aligned} \text{minimize} \quad & c(\dot{\theta}(t)) \\ \text{subject to} \quad & J_e(\theta(t))\dot{\theta}(t) = \dot{r}_e(t) \\ & J_c(\theta(t))\dot{\theta}(t) = \dot{r}_c(t) \\ & l \leqslant \dot{\theta}(t) \leqslant u \end{aligned} \tag{6.3}$$

where $c(\cdot)$ is an appropriate convex cost function, and l and u are respectively lower and upper bounds of the joint velocity vector [46].

In the above problem formulation, the convex cost function can be selected according to the task needs. For example, various vector norms of the joint velocity vector (e.g., $\|\dot{\theta}(t)\|_1$, $\|\dot{\theta}(t)\|_2^2$, $\|\dot{\theta}(t)\|_\infty$) are often used as a cost function [42][43][44][45][46][47]. To avoid singularity configurations or drifty in repeated operations, a linear term of $\dot{\theta}$ can be added. The first constraint corresponds to the primary goal of tracking a given end-effector motion, and the second constraint is to achieve the secondary goal of obstacle avoidance, which is included whenever the shortest distance between the obstacle and the manipulator is within a prespecified safety margin. When there exist more than one obstacle, multiple constraints, similar to the second constraint, that associate with the obstacles, should be included in the problem formulation. By putting the motion requirement of the critical points as a constraint instead of as a part of a part of the objective function, the solution obtained from (6.3) is ensured to satisfy both the goals of tracking a specified end-effector trajectory and avoiding the obstacles simultaneously.

6.2.2 Inequality Constrained Formulation

The previous obstacle avoidance method formulated in (6.3) can make the manipulator link move away from the obstacle once it enters the danger zone (i.e., the region inside the prespecified safety margin). But it has several drawbacks. First, how to determine the suitable magnitude of escape velocity $\dot{r}_c$? This value should be selected large enough to ensure that the link moves faster than the obstacle (if the obstacle is moving) to ensure the minimum distance between them enlarge as time goes on. At the same time, to be safe and energy efficient, the velocity should not be too large. Second, suppose that there are p critical points. If $m + p > n$, the optimization problem (6.3) is overdetermined, i.e., it has no solution. From this sense, the equaltiy constraints unnecessarily reduce the solution space, sometimes even make it null.

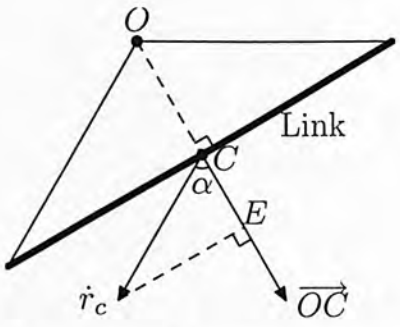


Figure 6.2: Escape direction and magnitude

One possible approach to overcome these drawbacks is to replace the equality constraints by inequality constraints. As shown in Fig. 6.2, we can just constrain the direction and magnitude of $\dot{r}_c$ in a range and let the optimization process to determine its accurate direction and magnitude.

The constraints of direction and magnitude of $\dot{r}_c$ can be considered as follows. The component of $\dot{r}_c$ projected on $\overrightarrow{OC}$ (i.e., $\overrightarrow{CE}$ in Fig. 6.2) should change with the minimum distance $|OC|$ between the obstacle and the link. When the distance is large, $|CE|$ can be small, but when the distance is near small, especially less than the safety margin, $|CE|$ should be big enough to direct the link away from the obstacle promptly. Because the moving velocity of the obstacle is not known, $|CE|$ should be large enough when the distance $|OE|$ is very small. A natural candidate function is the hyperbolic function $\frac{c}{|OC|}$, where c is a positive parameter controlling the velocity magnitude as shown in Fig. 6.3. Then the obstacle avoidance constraint can be formulated as

$$|\dot{r}_c| \cos\alpha \geqslant \frac{c}{|OC|}. \tag{6.4}$$

The above inequality is equivalent to

$$|\dot{r}_c||OC| \cos\alpha \geqslant c.$$

It can be further written in the dot product of the two vectors $\dot{r}_c$ and $\overrightarrow{OC}$ as

$$\dot{r}_c \cdot \overrightarrow{OC} \geqslant c.$$

Because

$$\overrightarrow{OC} = \begin{bmatrix} x_c - x_o & y_c - y_o & z_c - z_o \end{bmatrix}^T$$

and

$$\dot{r}_c = J_c(\theta)\dot{(\theta)},$$

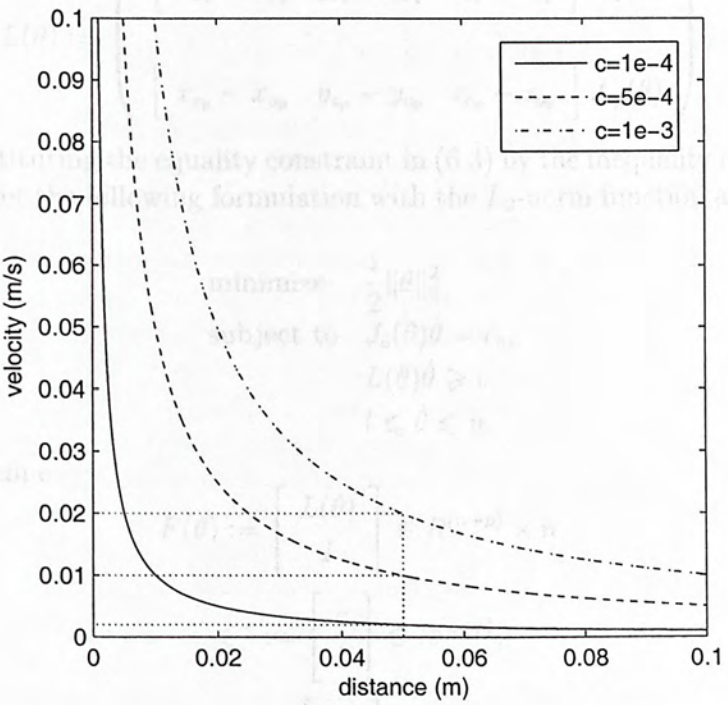


Figure 6.3: Minimum escape velocity component on the escape direction

$\dot{r}_c \cdot \overrightarrow{OC}$ can be written in the expression of $\dot{\theta}$,

$$\dot{r}_c \cdot \overrightarrow{OC} = \begin{bmatrix} x_c - x_o & y_c - y_o & z_c - z_o \end{bmatrix} J_c(\theta)\dot{\theta}.$$

Then the constraint (6.4) evolves to the following form

$$\begin{bmatrix} x_c - x_o & y_c - y_o & z_c - z_o \end{bmatrix} J_c(\theta)\dot{\theta} \geqslant c. \tag{6.5}$$

If there are p critical points, define

$$L(\theta) := \begin{pmatrix} \begin{bmatrix} x_{c_1} - x_{o_1} & y_{c_1} - y_{o_1} & z_{c_1} - z_{o_1} \end{bmatrix} J_{c_1}(\theta) \\ \vdots \\ \begin{bmatrix} x_{c_p} - x_{o_p} & y_{c_p} - y_{o_p} & z_{c_p} - z_{o_p} \end{bmatrix} J_{c_p}(\theta) \end{pmatrix}.$$

After substituting the equality constraint in (6.3) by the inequality constraint (6.5), we get the following formulation with the L_2-norm function as the cost function,

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\dot{\theta}\|_2^2 \\ \text{subject to} \quad & J_e(\theta)\dot{\theta} = \dot{r}_e, \\ & L(\theta)\dot{\theta} \geqslant c, \\ & l \leqslant \dot{\theta} \leqslant u. \end{aligned} \tag{6.6}$$

Further define

$$F(\theta) := \begin{bmatrix} L(\theta) \\ I \end{bmatrix} \in R^{(n+p)} \times n,$$

$$\xi^- := \begin{bmatrix} c \\ l \end{bmatrix} \in R^{(n+p)},$$

$$\xi^+ := \begin{bmatrix} \infty \\ u \end{bmatrix} \in R^{(n+p)},$$

then the above formulation can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\dot{\theta}\|_2^2 \\ \text{subject to} \quad & J_e(\theta)\dot{\theta} = \dot{r}_e, \\ & \xi^- \leqslant F(\theta)\dot{\theta} \leqslant \xi^+. \end{aligned} \tag{6.7}$$

Now let's compare the proposed scheme with the one proposed in [50] where the obstacle avoidance is formulated as an inequality constraint

$$J_N(\theta)\dot{\theta} \leqslant 0, \tag{6.8}$$

where $J_N(\theta) = -\text{sgn}(\overrightarrow{OC}) \diamond J_C(\theta)$.[1] The vector matrix multiplication operator $\diamond$ is defined as $u \diamond V = [u_1 V_1, u_2 V_2, \cdots, u_p V_p]^T$, where the column vector $u = [u_1, u_2, \cdots, u_p]^T$ and the row vector V_i denotes the ith row of matrix V.

From (6.8), we can see that the direction of escape velocity is confined in the quadrant where $\overrightarrow{OC}$ lies, which varies with the Cartesian corrdinates setted up. While in this proposed scheme, the direction is only required to be on the perpendicular half plane on which side $\overrightarrow{OC}$ lies. That is to say, compared with the scheme in [50], the present scheme can double the solution space.

6.3 Simplified Dual Neural Network Model

6.3.1 Existing Approaches

In the past two decades, the theory, methodology and applications of neural networks have been widely investigated. As dense parallel computational models, neural networks possess many desirable properties such as real-time information processing. In particular, recurrent neural networks for optimization, control, and signal processing received tremendous interests [26][33][34][35]. In recent years, neural networks have been developed and applied to robot control in intelligent robotic systems. Specifically, feedforward and recurrent neural networks have been used for kinematic control and dynamic control of redundant manipulators. In particular, Wang, Xia, Zhang, et al, have developed several recurrent neural networks for kinematic control and torque optimization of redundant manipulators using different objective functions under various conditions or constraints [43][44][45][46][47][32][49][51][50][52]. These recurrent neural networks are theoretically proven to be asymptotically stable and are demonstrated to be suitable for real-time motion control of redundant manipulators. For example, a two-layer recurrent neural network called the Lagrangian network is developed based on a quadratic optimization formulation of differential inverse kinematic problem and Lagrange optimality condition for pseudoinverse control of kinematically redundant manipulators without directly computing the pseudoinverse of a Jacobian matrix [44][46]. Such a neural network is proven to be capable of generating optimal joint rate signals to track given trajectories, even though the Jacobian matrix is rank-deficient. A single-layer recurrent neural network called the dual network is recently developed based on the primal-dual formulation of the differential inverse kinematic problem and the duality theory in optimization [27][30]. This neural network has much lower architectural complexity

[1] The notations in [50] are revised to be consistent with the notations in this chapter.

and is also proven to be able to generate optimal joint rate signals like the Lagrangian network. Recently, a recurrent neural network called the primal-dual network with much reduced network complexity is also applied for the minimum infinity-norm kinematic control of redundant manipulators based on an improved problem formulation [26][33]. Compared with feedforward neural networks for robot control based on supervised learning, the presented recurrent neural networks eliminate the need for off-line learning. Compared with other recurrent neural network approaches to robot control, the stability and optimality of the above models are theoretically guaranteed.

The above recurrent neural network approaches to robot kinematic control mainly aim at the primary task of redundancy resolution for the trajectory tracking by the end-effectors of redundant manipulators without consideration of the secondary task of obstacle avoidance. To equip redundant manipulators with the capability of obstacle avoidance based on neural networks, the following investigation on recurrent neural networks for real-time obstacle avoidance of redundant manipulators is absolutely necessary and rewarding.

6.3.2 Model Derivation

In [32][49], a neural network called dual neural network is presented to solve the convex quadratic problems utilizing the dual variables. In this subsection, we will simplify the dual neural network for solving the quadratic programming problem (6.7). Here we call the network as simplified dual neural network, which reduces the dual neural network's structural complexity while preserving its desirable convergence and optimality properties.

Taken the problem (6.7) as the primal problem, its dual problem can be written as

$$\begin{aligned} \text{maximize} \quad & \dot{r}_e^T y - \frac{1}{2}\dot{\theta}^T\dot{\theta} + \xi^{-T}v - \xi^{+T}w, \\ \text{subject to} \quad & \dot{\theta} - J_e(\theta)^T y - F(\theta)^T v + F(\theta)^T w = 0, \\ & v \geqslant 0, \quad w \geqslant 0, \end{aligned} \tag{6.9}$$

where $y \in R^m, v \in R^{n+p}, w \in R^{n+p}$ are dual variables.

Define $u = v - w$, the equality constraint in (6.9) becomes

$$\dot{\theta} - J_e(\theta)^T y - F(\theta)^T u = 0.$$

According to the Karush-Kuhn-Tucker (KKT) conditions for convex optimization, the following set of equations have the same solution as problem (6.7).

$$\dot{\theta} - J_e(\theta)^T y - F(\theta)^T u = 0, \tag{6.10a}$$

$$J_e(\theta)\dot{\theta} = \dot{r}_e, \tag{6.10b}$$

$$\begin{cases} (F(\theta)\dot{\theta})_i = \xi_i^-, & \text{if } u_i > 0, \\ (F(\theta)\dot{\theta})_i = \xi_i^+, & \text{if } u_i < 0, \\ \xi_i^- < (F(\theta)\dot{\theta})_i < \xi_i^+, & \text{if } u_i = 0. \end{cases}$$

The last equation can be rewritten as

$$F(\theta)\dot{\theta} = g(F(\theta)\dot{\theta} - u), \tag{6.10c}$$

where $g(v)$ is a piecewise linear function, defined as

$$g(v) = [g_1(v_1), g_2(v_2), \cdots, g_p(v_p)]^T,$$

and

$$g_i(v_i) = \begin{cases} \xi_i^-, & \text{if } v_i < \xi_i^-, \\ v_i, & \text{if } \xi_i^- \leqslant v_i \leqslant \xi_i^+, \\ \xi_i^+ & \text{if } v_i > \xi_i^+. \end{cases} \quad i = 1, \cdots, p.$$

If $\xi_i^+ = \infty$, $g(v_i)$ degenerates to

$$g_i(v_i) = \begin{cases} v_i, & \text{if } v_i \geqslant \xi_i^- \\ \xi_i^-, & \text{otherwise.} \end{cases} \quad i = 1, \cdots, p$$

From equation (6.10a), $\dot{\theta}$ can be explicitly expressed by y and u,

$$\dot{\theta} = J_e(\theta)^T y + F(\theta)^T u. \tag{6.11}$$

Substitute the expression of $\dot{\theta}$ into (6.10b),

$$J_e(\theta)J_e(\theta)^T y + J_e(\theta)F(\theta)^T u = \dot{r}_e.$$

Then y can be explicity expressed by u,

$$y = (J_e(\theta)J_e(\theta)^T)^{-1}[-J_e(\theta)F(\theta)^T u + \dot{r}_e]. \tag{6.12}$$

Based on (6.10c),(6.11),(6.12), by the projection theorem [2], the simplified dual neural network for solving the primal problem (6.7) can be designed as

- State equation

$$\epsilon\frac{du}{dt} = -F(\theta)\dot{\theta} + g(F(\theta)\dot{\theta} - u)$$

- Output equation

$$\begin{aligned} \dot{\theta} &= J_e(\theta)^T y + F(\theta)^T u \\ y &= (J_e(\theta)J_e(\theta)^T)^{-1}[-J_e(\theta)F(\theta)^T u + \dot{r}_e] \end{aligned}$$

where $u \in R^p$ is the state vector, $\epsilon > 0$ is a scaling parameter that controls the convergence rate of the neural network. It can be rewritten in the more compact and explicit form as

- State equation

$$\begin{aligned} \epsilon \frac{du}{dt} &= -F(\theta)MF(\theta)^T u + g(F(\theta)MF(\theta)^T u - u \\ &\quad + F(\theta)s) + F(\theta)s \end{aligned} \tag{6.13a}$$

- Output equation

$$\dot{\theta} = MF(\theta)^T u + s \tag{6.13b}$$

where $M := I - J_e(\theta)^T (J_e(\theta) J_e(\theta)^T)^{-1} J_e(\theta), s := J_e(\theta)^T (J_e(\theta) J_e(\theta)^T)^{-1} \dot{r}_e$.

In this context, the desired end-effector velocity $\dot{r}_e$ is fed into the neural network as its input. And the neural network parameters (e.g. J_e, J_o, $g(\cdot)$) are time-varying, determined by the pose of the manipulator and the position of the obstacle. The optimal joint rate $\dot{\theta}$ that could make the manipulator avoid obstacles is generated as the neural network output. By further taking integration of the joint velocities with the known initial values, we can get the joint motions that the manipulator should follow.

Because the analytic expression of $J_e(\theta), F(\theta)$ can be obtained in the design stage, the analytic expressions of M and s can be computed beforehand. In the simplified dual neural network, the number of neurons is equal to the number of inequality constraints, whereas the number of neurons is equal to the number of equality and inequality constraints in the original dual neural network [32] [49].

6.3.3 Convergence Analysis

The properties of convergence and optimality of the solution are studied below.

Theorem 7 *The simplified dual neural network (6.13a) is globablly convergent to an equilibrium pont u^*.*

Proof We have the following inequality at u^*

$$(v - F(\theta)\dot{\theta}^*)^T u^* \geqslant 0, \forall v \in \Omega,$$

where $\dot{\theta}^*$ is the optimal solution of problem (6.7).

Therefore, it follows that

$$\begin{aligned}&[g(F(\theta)MF(\theta)^T u + F(\theta)s - u)\\ &-(F(\theta)MF(\theta)^T u^* + F(\theta)s)]^T u^* \geqslant 0.\end{aligned} \tag{6.14}$$

On the other hand, from the projection theorem [2], it follows that $\forall u \in R^n$,

$$\begin{aligned}(g(F(\theta)MF(\theta)^T u + F(\theta)s - u) - (F(\theta)MF(\theta)^T u^* +\\ F(\theta)s))^T((F(\theta)MF)\theta)^T u + F(\theta)s - u)\\ -g(F(\theta)MF(\theta)^T u + F(\theta)s - u)) \geqslant 0.\end{aligned} \tag{6.15}$$

Combining (6.14) and (6.15), we have

$$\begin{aligned}(g(F(\theta)MF(\theta)^T u + F(\theta)s - u) - (F(\theta)MF(\theta)^T u^* +\\ F(\theta)s))^T(u^* + (F(\theta)MF(\theta)^T u + F(\theta)s - u)\\ -g(F(\theta)MF(\theta)^T u + F(\theta)s - u)) \geqslant 0.\end{aligned}$$

Defining

$$\tilde{g} := (F(\theta)MF(\theta)^T u + F(\theta)s - u) - (F(\theta)MF(\theta)^T + F(\theta)s),$$

it can be written as

$$[\tilde{g} + F(\theta)MF(\theta)^T(u - u^*)]^T[(u - u^*) + \tilde{g}] \leqslant 0.$$

Then we can get

$$\begin{aligned}&(u - u^*)^T\tilde{g} + \tilde{g}^T F(\theta)MF(\theta)^T(u - u^*)\\ &\leqslant -\|\tilde{g}\|^2 - (u - u^*)^T F(\theta)MF(\theta)^T(u - u^*).\end{aligned} \tag{6.16}$$

Because

$$F(\theta)MF(\theta)^T = F(\theta)(I - J_e(\theta)(J_e(\theta)J_e(\theta)^T)^{-1}J_e(\theta))F(\theta)^T$$

is positive semidefinite (see Lemma 4 in the Appendix), i.e.,

$$(u - u^*)^T F(\theta)MF(\theta)^T(u - u^*) \geqslant 0. \tag{6.17}$$

From (6.16) and (6.17), we can get

$$(u - u^*)^T\tilde{g} + \tilde{g}^T F(\theta)MF(\theta)^T(u - u^*) \leqslant 0. \tag{6.18}$$

If and only if $u = u^*$, the equality holds.

Now choose the following radially unbounded Lyapunov function candidate

$$V(u) = \frac{1}{2}\|Q(u) - u^*\|_2^2, \tag{6.19}$$

where Q is symmetric and positive definite matrix with $Q^2 = (I+F(\theta)MF(\theta)^T)$. Then from (6.18), we get

$$\begin{aligned} \frac{dV}{dt} &= (u-u^*)^T Q^2 \dot{u} \\ &= (u-u^*)^T (I + F(\theta)MF(\theta)^T)\tilde{g} \\ &= (u-u^*)^T \tilde{g} + \tilde{g}^T F(\theta)MF(\theta)^T (u-u^*) \\ &\leqslant 0. \end{aligned}$$

By the Lyapunov theorem, the simplified dual neural network is globally convergent. □

Theorem 8 *$\dot{\theta}^* = Mu^* + s$ is an optimal solution of the quadratic programming problem (6.7), where u^* is an equilibrium point of the dynamic equation (6.13a).*

This theorem can be proven by substituting $\dot{\theta}^*$ into the KKT conditions (6.10). The three conditions are satisified, which means that $\dot{\theta}^*$ is the optimal solution of the problem (6.7).

From Theorems 7 and 8, we can conclude that the simplifed dual neural network (6.13) is globally convergent to the exact solution of the problem (6.7).

6.3.4 Model Comparision

As pointed out at the beginning of this section, previously three recurrent neural network models have been proposed for kinematically redundant manipulators, respectively Lagrangian neural network [46], dual neural network [50] and primal-dual neural network [52]. In this subsection, we will compare the different aspects of these recurrent neural network models with the simplified dual neural network proposed in this chapter.

In [46], the obstacle avoidance is formulated as the equality constraints and not considering the joint velocity limits. The formulation can be written as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\dot{\theta}^T\dot{\theta}, \\ \text{subject to} \quad & J_e(\dot{\theta}) = \dot{r}_e, \\ & J_c(\dot{\theta}) = \dot{r}_c. \end{aligned} \tag{6.20}$$

If we define $J := [J_e{}^T, J_c{}^T]^T$, $\dot{r} := [\dot{r}_e^T, \dot{r}_c^T]^T$, then the Lagrangian neural network for solving (6.20) can be written as

$$\mu \frac{du}{dt} = -u - J^T v, \tag{6.21a}$$

$$\mu \frac{dv}{dt} = Ju - \dot{r}, \tag{6.21b}$$

where μ is a positive scaling constant to scale the convergence rate of the neural network. The Lagrangian neural network can only tackle equality constraints easily. And the number of neuron is equal to $3m$. Another drawback of Lagrangian neural network is that it can only converge to the approximate optimal value, not the exact optimal value.

If we define

$$H := \begin{bmatrix} J_e(\theta) \\ F(\theta) \end{bmatrix}, \quad \eta^- := \begin{bmatrix} \dot{r}_e \\ \xi^- \end{bmatrix}, \quad \eta^+ := \begin{bmatrix} \dot{r}_e \\ \xi^+ \end{bmatrix},$$

the dual neural network proposed in [50] for solving (6.7) can be written as

$$\mu \frac{du}{dt} = g(HH^T u - u) - HH^T u, \tag{6.22a}$$

$$\frac{d\theta}{dt} = H^T u, \tag{6.22b}$$

where μ is a positive design parameter to scale the convergence rate of the dual neural network, and g is defined similiarly as (6.10c). The number of neurons is $n + m + 1$. This neural network is globally convergent to the optimal solution.

In [52], a new recurrent neural network called primal-dual neural network is proposed. In the paper, the neural network can only tackle bound inequality constraints, not general inequality constraints. But if new variables are introduced, this neural network can be used to solve (6.7). The number of neurons is $3n+4$. This neural network is simplifier than dual neural network only if the inequality constraints is bound constraints, instead of the general ones. This neural network is also globally convergent to the optimal solution.

In summary, the simplified dual neural network is simplifier than these previously proposed neural networks with $n+1$ neurons while preserving the globally optimal solution property.

6.4 Simulation Results

In this section, the validity of the proposed obstacle avoidance scheme and the real-time solution capability of the simplified dual neural network is shown

through a simulation case with the Mistsubishi 7-DOF PA10-7C manipulator. The coordinates are setted up according to [44], and the structure parameters, joint limits, and joint velocity limits can also be found there. In this study, only the position of the end-point is concerned, then $m = 3$ and $n = 7$. The parameter c is chosen as $5e-4$. The parameter that controls the convergence rate of the simplified dual neural network is chosen as $\epsilon = 10^{-6}$.

Example 1: The desired motion of the end-effector is a circle of radius $r = 20$cm with the revolute angle about the x axis $\pi/6$. The task time of the motion is 10s and the initial joint variables $\theta(0)$=[0; $-\pi/4$; 0; $\pi/2$; 0; $-\pi/4$; 0]. In the workspace there are two obstacle points, respectively [-0.0454m; -0.0737m; 0.8367m], and [-0.1541m; -0.0609m; 0.5145m]. Fig. 6.4 illustrates the simulated motion of the PA10 manipulator in the 3D workspace, which is sufficiently close to the desired one with the tracking error less than 0.6mm, as seen from Fig. 6.5.

The minimum distances between obstacles and links are shown in Figs. 6.6, 6.7, and 6.8, where the links always tend to move away from the obstacle once they enter the danger zone. Fig. 6.9 shows the comparison between the case with and without obstacle avoidance scheme. Without obstacle avoidance, the link will collide with an obstacle as shown by the dotted line in Fig. 6.9. But when the proposed scheme is adopted, the link will move away from the obstacle once entering the danger zone, which is shown by the solid line in Fig. 6.9.

Example 2: In this example, The desired motion of the end-effector is a straight line. The task time of the motion is 10s and the initial joint variables $\theta(0)$=[0; $-\pi/4$; 0; $\pi/2$; 0; $-\pi/4$; 0]. In the workspace there are two obstacle points, respectively [0; 0; 0.6000m], and [-0.1541m; -0.0609m; 0.4500m]. Figs. 6.12 and 6.13 show respectively the motion trajectories of PA10 when there are no obstacles and when there are two obstacles in the workspace. The minimum distances between obstacles and links are shown in Fig. 6.15, where the links always tend to move away from the obstacle once they enter the danger zone.

6.5 Concluding Remarks

A new obstacle avoidance scheme is proposed based on quadratic programming formulation. The solution to this time-varying optimization problem using a new recurrent neural network called simplified dual neural network is compared with other neural network models and is simulated with the PA10 robot manipulators. The results show that this scheme is effective and can be performed in real time.

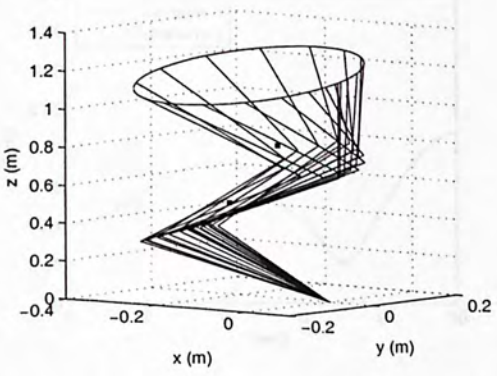


Figure 6.4: The PA10 manipulator in a circular motion

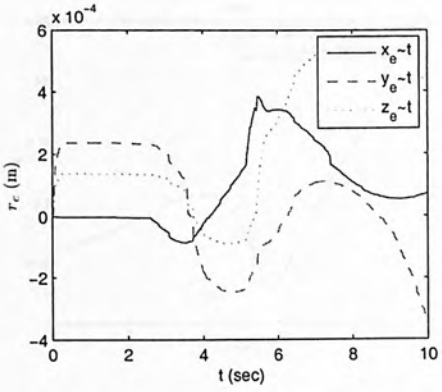


Figure 6.5: Position Error of the end-effector

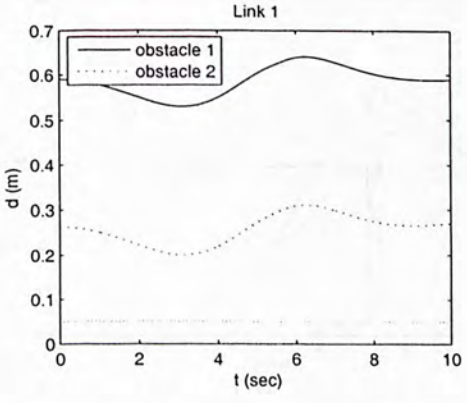


Figure 6.6: Minimum distance between obstacles and Link 1

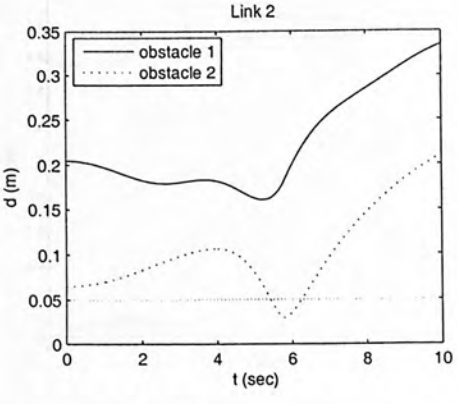


Figure 6.7: Minimum distance between obstacles and Link 2

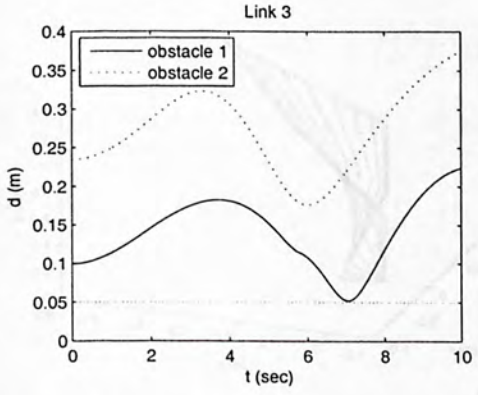


Figure 6.8: Minimum distance between obstacles and Link 3

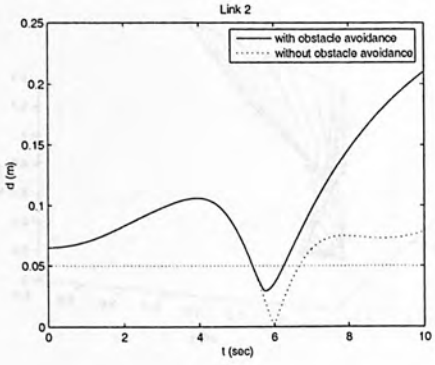


Figure 6.9: Comparision of minimum distance between obstacle 2 and link 2 with and without obstacle avoidance

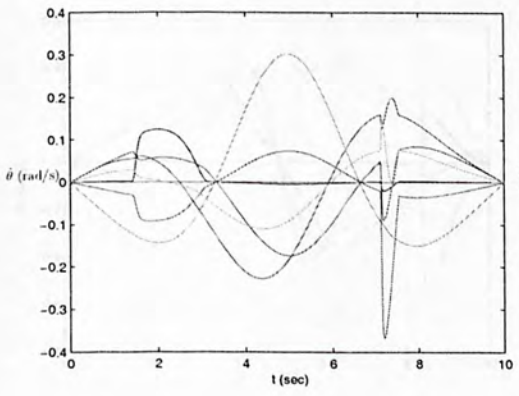


Figure 6.10: Joint velocities

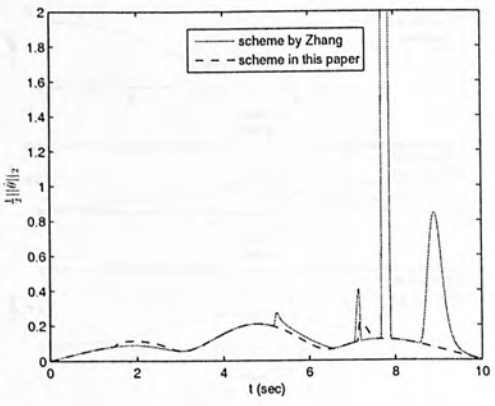


Figure 6.11: Comparision of minimization of the objective function between two schemes

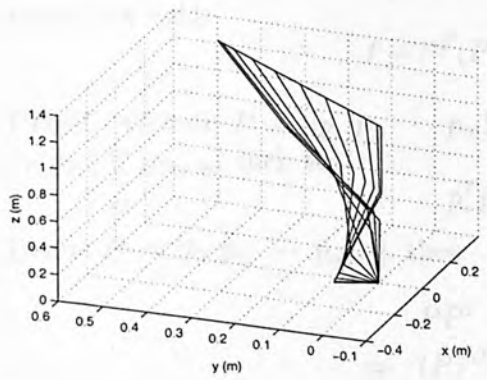


Figure 6.12: The motion trajectory of PA10 when there are no obstacles while tracking a straight line

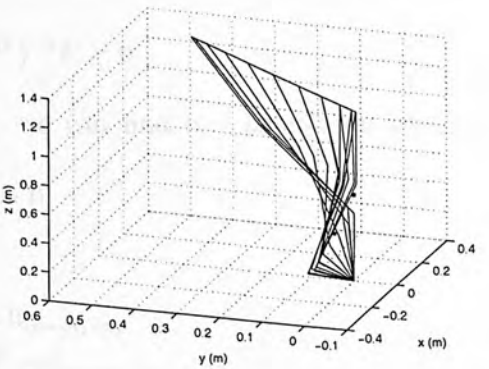


Figure 6.13: The motion trajectory of PA10 when there are two obstacles while tracking a straight line

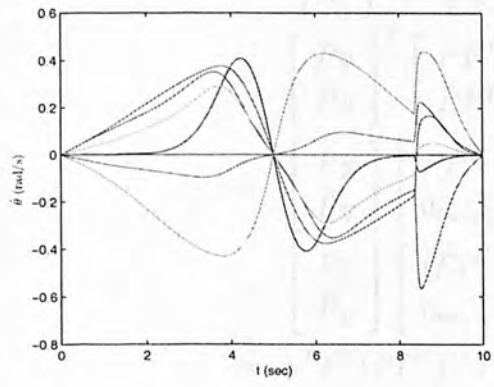


Figure 6.14: The joint velocities of PA while tracking a straight line

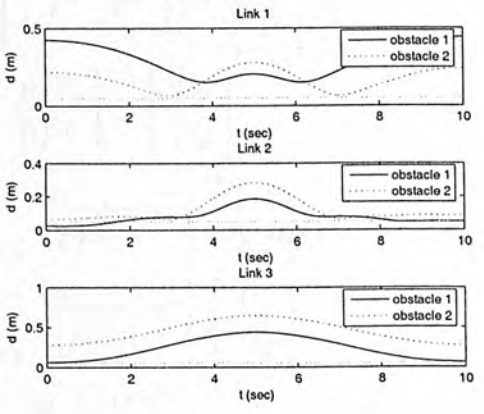


Figure 6.15: The distances between obstacles and links while tracking a straight line

Appendix

Lemma 4 *For $P \in R^{m\times n}$ $(m < n)$, rank$(P) = m$, the following matrix inequality holds*

$$I - P^T(PP^T)^{-1}P \succeq 0$$

Proof Denote $P = [p_1\ p_2\ \cdots\ p_m]^T$, we can find $n-m$ column vectors $\tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_{n-m}$, such that

$$p_i^T \tilde{p}_i = 0.$$

Define $\tilde{P} := [\tilde{p}_1\ \tilde{p}_2\ \cdots\ \tilde{p}_{n-m}]$, then

$$\begin{aligned} & \tilde{P}P^T = 0_{(n-m)\times m} \\ \Rightarrow\ & (\tilde{P}P^T)^T = 0_{m\times(n-m)} \\ \Rightarrow\ & P\tilde{P}^T = 0_{m\times(n-m)}. \end{aligned}$$

Define $Q := \begin{bmatrix} P \\ \tilde{P} \end{bmatrix} \in R^{n\times n}$. Because rank$(P) = m$, and $p_1, p_2, \cdots, p_m, \tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_{n-m}$ are linearly independent vectors, then matrix Q is invertible.

$$\begin{aligned} y^T y &= y^T Q^T (Q^T)^{-1} Q^{-1} Q y \\ &= (Qy)^T (QQ^T)^{-1} (Qy) \\ &= (\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} y)^T (\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} \begin{bmatrix} P^T & \tilde{P}^T \end{bmatrix})^{-1} (\begin{bmatrix} P \\ \tilde{P} \end{bmatrix} y) \\ &= \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix}^T \begin{bmatrix} PP^T & P\tilde{P}^T \\ \tilde{P}P^T & \tilde{P}\tilde{P}^T \end{bmatrix}^{-1} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\ &= \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \begin{bmatrix} PP^T & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & \tilde{P}\tilde{P}^T \end{bmatrix}^{-1} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\ &= \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \begin{bmatrix} (PP^T)^{-1} & 0_{m\times(n-m)} \\ 0_{(n-m)\times m} & (\tilde{P}\tilde{P}^T)^{-1} \end{bmatrix} \begin{bmatrix} Py \\ \tilde{P}y \end{bmatrix} \\ &= y^T P^T (PP^T)^{-1} Py + y^T \tilde{P}^T (\tilde{P}\tilde{P}^T)^{-1} \tilde{P}y \end{aligned}$$

Move $y^T P^T (PP^T)^{-1} Py$ to the left side,

$$y^T y - y^T P^T (PP^T)^{-1} Py = y^T \tilde{P}^T (\tilde{P}\tilde{P}^T)^{-1} \tilde{P}y. \tag{6.23}$$

Because

$$\tilde{P}\tilde{P}^T \succeq 0,$$

then

$$\tilde{P}^T(\tilde{P}\tilde{P}^T)^{-1}\tilde{P} \succeq 0.$$

That is,

$$y^T\tilde{P}^T(\tilde{P}\tilde{P}^T)^{-1}\tilde{P}y \geq 0, \quad \forall y \in R^n. \tag{6.24}$$

From (6.23) and (6.24),

$$y^Ty - y^TP^T(PP^T)^{-1}Py \geq 0, \quad \forall y \in R^n,$$

i.e.,

$$I - P^T(PP^T)^{-1}P \succeq 0.$$

□

□ **End of chapter.**

Chapter 7

Multiuser Detection

Multiuser detection has gained much attention in recent years for its potential to greatly improve the capacities of CDMA communication systems. In this chapter, a recurrent neural network is presented for solving the nonlinear optimization problem involved in the multiuser detection in CDMA. Compared with other neural networks, the presented neural network can globally converge to the exact optimal solution of the nonlinear optimization problem with nonlinear constraints and has relatively low structural complexity. Computer simulation results are presented to show the optimization capability. The performance in CDMA communcation systems is also verified through a simulation example.

7.1 Introduction

In wireless communication systems, Direct-Sequence Code Division Multiple Access (DS-CDMA) is a promising technology, with several advantages over others: asynchronous multiple access, robustness to frequency selective fading, and etc [85]. But to permit a high number of user communicating simultanously with high bit rate in 3G mobile communication systems, the capacity has to be increased. The capacity of DS-CDMA is limited by signal interference. Therefore it can be increased by using techniques that suppress interference. Many research activities have been focused on the MultiUser Detection (MUD) techniques [85][86]. Among them, blind detection is the most promising one because it requires no more knowledge than does the conventional single-user detection: the desired user's signature waveform and its timing. The blind detection techniques turns the MUD problem to a constrained nonlinear optimization of the objective function Minimum Output Energy (MOE). Based on this, Verdú proposed the adaptive blind detection which gives an adaptive solution to the nonlinear optimization to reduce the

computational complexity [87]. As the cost, adaptive blind detection needs a certain number of bit intervals to reach the optimal solution, since the path towards the optimum filter coefficients set is performed in a step by step approximation, based on the steepest descent gradient technique.

Wireless communication channels are changing with time rapidly, because of both natural communicating conditions (like multipath fading, mobile terminal moving, etc.) and random asynchronous access of other interfering users. So the wireless communication systems are typical real-time systems. This requires that the detection methods should adapt the parameters rapidly to achieve good performance.

Neural network approaches have shown to be able to handle real-time applications, because of VLSI implementability and parallel processing capability [88]. The idea proposed in this chapter is to employ a recurrent neural network in order to accelerate the convergence process of the adaptive filter coefficients towards the optimum solution in the blind detection algorithm.

After Hopfield and Tank's seminal work, Kennedy and Chua developed a neural network with a finite penalty parameter for solving nonlinear programming problems [88]. Although this work actually fullfills both the Kuhn-Tucker optimality conditions in terms of penalty function, this network is not capable to find an exact optimal solution due to a finite penalty parameter and is difficult to implement when the penalty parameter is very large. Fantacci, Forti, and et al, developed a blind detector/receiver based on Kennedy and Chua's neural network [89]. Kechriotis and Manolakos [94] investigated the application of Hopfield neural networks (HNN's) to the problem of multiuser detection in spread spectrum/CDMA (code division multiple access) communication systems. Recently, Xia and Wang proposed a recurrent neural network that can solve nonlinear convex optimization problems [90][91]. Xia and Wang's neural network is stable in the sense of Lyapunov and globally convergent to the exact optimal solution. Applying Xia and Wang's neural network, we proposed a neural network approach to blind multiuser detection. The simulation results show that it is efficient in blind detection of CDMA.

7.2 Problem Formulation

Multiaccess communication, in which several transmitters share a common channel, is common nowadays, like mobile communication systems, satellite communication systems, packet-radio networks, and etc. A common feature of those communication channels is that the receiver obtains a noisy version

of the superposition of the signal sent by active transmitters. How to detect the desired signal is important for these communication systems.

The conventional DS-CDMA system treats each user as signal, with other users considered as noise or MAI (Mulitple Access Interference). This technique has several inherent shortcomings. Simply considering other users as noise makes the capacity interference-limited. For the same reason the near/far effect is very serious, so the system needs good power control [85].

To solve these problems, the MultiUser Detection (MUD) method was proposed by Verdú. MUD considers all users as signals when detecting a particular user signal; so this is a joint detection (in comparison to the sperate detection disucssed previous). This technique can reduce interference and hence lead to increase capacity. At the same time it alleviates the near/far problem.

The optimum multiuser detector for asynchronous mulitple-access Gaussian channels is discussed in [85] where it is shown that the near/far problem suffered by the conventional CDMA receiver can be overcome by a more sophisticated receiver which accounts for the presence of other interferers in the channel. This receiver is shown to attain essentially single-user performance upon knowing the following [86]:

① The signature waveform of the desired user.

② The signature waveform of the interfering users.

③ The timing of the desired user.

④ The timing of each interfering user.

⑤ The received relative amplitudes of the interfering users to that of the desired user.

The conventional receiver only requires ① and ③, but it is severely limited by the near/far problem, even in the presence of perfect power control, the bit-error-rate is orders of magnitude far from optimal.

To alleviate the need to know interferers' signature ②, timing ④, and amplitudes ⑤, some attention has been focused recently on adaptive multiuser detection. The adaptive multiuser detector in [87] is based on the minimization of mean-square-error (MMSE) between the outputs and the data. But it needs traing data sequences for each user to approximate the unknown parameters. However, at any time there may be a drastic change in the communication environment (e.g. a deep fade or the access of other interfering users), at this time the parameters becomes unreliable. Then data transmission of the desire user must be temporarily suspened and a

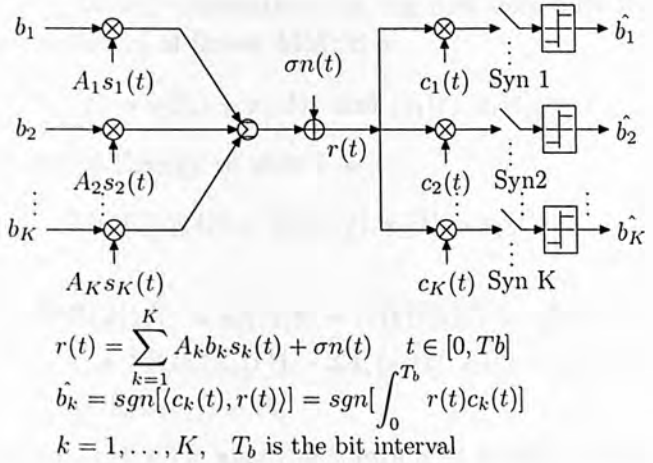


Figure 7.1: Simplified K-user DS-CDMA Synchronous Communication Model

fresh training sequence must be retransmitted. Thus retransmitting of the training sequence is cumbersome in most CDMA systems, where one of the most important advantages is the ability to have completely asynchronous and uncoordinated transmissions that switch on and off autonomously.

The foregoing observation implies that the need for blind adaptive receivers is even more evident in multiaccess channels than in single-user channels subject to intersymbol interference.

The formulation of blind adaptive multiuser detection is discussed as follows.

Let's consider the DS-CDMA systems as illustrated in Figure 7.1, where $b_k \in \{-1, 1\}$ is the bits to be transmitted, $s_k(t)$ is the kth user's signature waveform, A_k is the modulation amplitude, $\sigma n(t)$ is the additive Gaussian white noise, $r(t)$ is the received signal, and $c_k(t)$ is the matched filter coefficients. The commonly used objective function to be minimized in multiuser detection is:

$$\text{MMSE}(c_k(t)) = \min_{c_k} E[(b_k - \langle c_k(t), r(t)\rangle)^2]$$

where $\langle c_k(t), r(t)\rangle = \int_0^{T_b} c_k(t)r(t)dt$, $c_k(t)$ is the waveform used to demodulate $r(t)$. The output decision is

$$\hat{b_k} = sgn[\langle c_k(t), r(t)\rangle]$$

It can be proven that the solution of this optimization problem has no relation with b_k.

For simplicity, we will concentrate on the first user only from now. The canonical representation of linear MMSE is

$$c_1(t) = s_1(t) + x_1(t), \text{ and } \langle s_1(t), x_1(t)\rangle = 0$$

Define Mean Output Energy of user 1 as

$$\text{MOE}(x_1(t)) = \text{E}[(\langle r(t), s_1(t) + x_1(t)\rangle)^2]$$

Then

$$\begin{aligned}&\text{MSE}(x_1(t)) = \text{E}[(A_1 b_1 - \langle r(t), s_1(t) + x_1(t)\rangle)^2] \\ &= A_1^2 + \text{MOE}(x_1(t)) - 2A_1\langle s_1(t), s_1(t) + x_1(t)\rangle \\ &= \text{MOE}(x_1(t)) - A_1^2\end{aligned}$$

So the solution of MMOE is also the solution of MMSE. While MMOE has no nothing to do with b_k, then the detection problem evolves to the following form:

$$\begin{aligned}\min \quad & \text{MOE}(x_1(t)) = \text{E}[(\langle r(t), s_1(t) + x_1(t)\rangle)^2] \\ \text{s.t.} \quad & \langle s_1(t), x_1(t)\rangle = 0\end{aligned}$$

Because the received signature $\hat{s}_1$ is not always the same as s_1, so we need to add the surplus energy constraints [87]:

$$||x_1(t)||^2 < \chi$$

where χ is the surplus energy which is a positive constant. So the formulation for Blind MMOE Detector with surplus energy constraints can be expressed as follows,

$$\begin{aligned}\min \quad & \text{MOE}(x_1(t)) = E[(\langle r(t), s_1(t) + x_1(t)\rangle)^2] \\ \text{s.t.} \quad & \langle s_1(t), x_1(t)\rangle = 0\ , \\ & ||x_1(t)||^2 \leq \chi\end{aligned} \tag{7.1}$$

It is proven that $\text{MOE}(x_1)$ is a convex function [85].

The nonlinear optimization problem (7.1) is a general form. Particularly, when the signature waveforms $s_k(t), (k = 1, \ldots, K)$ are binary PN sequences, the vector form of (7.1) is as follows.

$$\begin{aligned}\min \quad & \text{MOE}(x_1) = E[(\langle r, s_1 + x_1\rangle)^2] \\ \text{s.t.} \quad & \langle s_1, x_1\rangle = 0\ , \\ & ||x_1||^2 \leq \chi\end{aligned} \tag{7.2}$$

where $x_1, s_1, y_1 \in R^n$, n is the number of chips per bit for the PN sequences. From now on, we will design the neural network based on formulation (7.2).

7.3 Neural Network Architecture

Problem (7.2) can be generalized to the following formulation:

$$\begin{array}{ll} \min & f(u) \\ \text{s.t.} & g(u) \leq 0 \\ & h(u) = 0 \end{array} \tag{7.3}$$

where, $u \in R^n$, $f(u), g(u), h(u)$ are scalar functions.

In [91], Xia and Wang developed a neural network for solving the following nonlinear optimization problem with inequality constraints.

$$\begin{array}{ll} \min & f(u) \\ \text{s.t.} & c(u) \leq 0, \quad u \geq 0 \end{array}$$

In addition in [90], Xia and Wang developed a neural network for solving the following nonlinear optimization problem with both equality and inequality constraints.

$$\begin{array}{ll} \min & \frac{1}{2}u^T Q u + q^T u \\ \text{s.t.} & g(u) \leq 0, \quad G^T u = -f_{ext} \end{array}$$

Following these design methods, here we present a recurrent neural network for solving the nonlinear program (7.3). For complete proof, please refer to [90][91].

To derive a neural network model for solving (7.3), we first give a equivalent form of (7.3).

$$\left\{ \begin{array}{rcl} \nabla f(u) + v\nabla g(u) - wh(u) & = & 0, \\ (v + g(u))^+ - v & = & 0, \\ h(u) & = & 0, \end{array} \right. \tag{7.4}$$

where $v \in R$, $w \in R$ are both auxiliary one-dimensional variables.

It can be derived as follows. First define the Lagrangian function

$$L(u, v, w) = f(u) + vg(u) - wh(u)$$

According to the well-known saddle point theorem [92], u^* is a solution to (7.3) if and only if there exists $v^* \in R^+$ and $w^* \in R$, such that for any $(u, v, w) \in R^n \times R^+ \times R$, (u^*, v^*, w^*) satisfies

$$L(u^*, v, w) \leq L(u^*, v^*, w^*) \leq L(u, v^*, w^*)$$

where $R^+ = \{v \in R \mid v \geq 0\}$. Then we can get

$$L(u^*, v, 0) \leq L(u^*, v^*, w^*) \leq L(u, v^*, w^*)$$

Because $h(u^*) = 0$, so

$$\begin{aligned} & f(u^*) + vg(u^*) \\ \leq\ & f(u^*) + v^*g(u^*) \\ \leq\ & f(u) + v^*g(u) - w^*h(u) \end{aligned}$$

From the first inequality above, we can derive that

$$(v - v^*)(-g(u^*)) \geq 0, \quad \forall v \geq 0$$

On the other side, let

$$\phi(u) = f(u) + v^*g(u) - wh(u)$$

Then the right inequality above implies:

$$\phi(u) \geq \phi(u^*), \quad \forall u \in R^n$$

this means,

$$\nabla\phi(u^*) = 0$$

that is,

$$\nabla f(u^*) + v^*\nabla g(u^*) - w^*\nabla h(u^*) = 0$$

so, u^* is a solution to (7.3), if and only if (u^*, v^*, w^*) satisfies

$$\begin{cases} \nabla f(u^*) + v^*\nabla g(u^*) - w^*\nabla h(u^*) = 0 \\ (v - v^*)(-g(u^*)) \geq 0, \quad \forall v \geq 0 \\ h(u^*) = 0 \end{cases} \tag{7.5}$$

From the projection theorem [2], it can be seen that the above formulation is equivalent to (7.4).

Based on the equivalent formulation in (7.4), we propose a recurrent neural network for solving (7.3) with its dynamical equation given by

$$\frac{d}{dt}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \lambda \begin{pmatrix} -\nabla f(u) - v\nabla g(u) + w\nabla h(u) \\ (v + g(u))^+ - v \\ -h(u) \end{pmatrix} \tag{7.6}$$

where λ is a positive scaling constant.

The neural network is guaranteed to be globally convergent to the exact optimal solution [90][91].

Now for the problem (7.2), the specific neural network can be defined by the following dynamic state equation.

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ v \\ w \end{pmatrix} = \lambda \begin{pmatrix} -2\mathrm{E}[r^T(x_1+s_1)r] - 2vx_1 - ws_1 \\ (v+||x_1||^2-\chi)^+ - v \\ -s_1^T x_1 \end{pmatrix} \tag{7.7}$$

where $x_1, s_1, r \in R^n$, n is the number of chips per bit. $v \in R$, $w \in R$, $\lambda > 0$ is a scalar parameter.

In [89], R. Fantacci, et al. also investigated the neural network approach to solve the problem (7.2). They proposed a recurrent neural network with nonobvious modification of Kennedy and Chua's neural network [88]. It can be defined by the following equation.

$$\dot{x_1} = -Gx_1 - \nabla V(x_1) + \langle \nabla V(x_1), s_1 \rangle s_1 \tag{7.8}$$

where

$$\begin{aligned} V(x_1) &= \mathrm{MOE}(x_1) + \textstyle\int_0^{f(x_1)} g(\rho)d\rho \ , \\ f(x_1) &= \chi - ||x_1||^2 \ , \\ g(\rho) &= \begin{cases} 0, & \text{for } \rho \le 0 \\ K\rho, & \text{for } \rho < 0 \end{cases} \end{aligned}$$

$x_1 \in R^n$, $G \in R^+$ models the neuron parasitic capacitances, and K is the positive penalty parameter.

7.4 Simulation Results

To verify the neural network (7.8), Fantacci, et al. used the following example problem.

$$\begin{aligned} \min \quad & \mathrm{MOE}(u) = \tfrac{1}{2}u^T Q u + q^T u \\ \text{s.t.} \quad & s^T u = 0 \ , \\ & ||u||^2 \le \chi \end{aligned}$$

where,

$$Q = \begin{pmatrix} 10 & 2 & 1 \\ 2 & 12 & 3 \\ 1 & 3 & 7 \end{pmatrix}, \quad q = \begin{pmatrix} 23 \\ -27 \\ -24 \end{pmatrix},$$

$$s = (1 \ \ 0 \ \ 0)^T, \quad \chi = 4, \quad u \in R^3$$

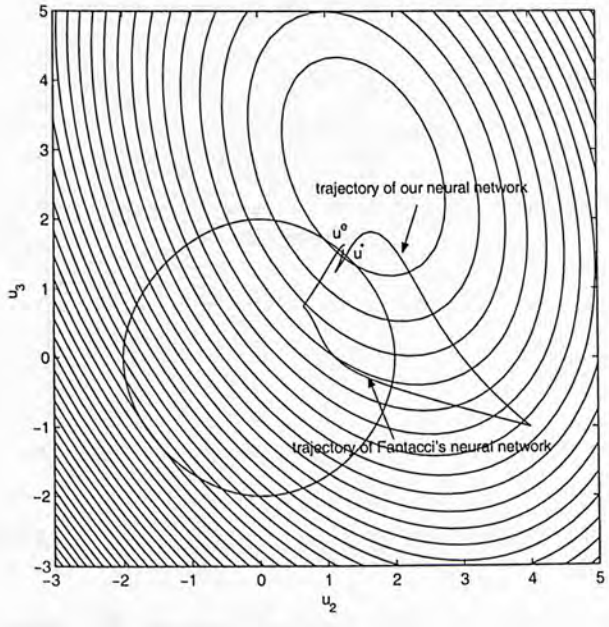


Figure 7.2: Trajectories of two Neural Networks

We will also use this problem to demonstrate the optimization capability of the neural network (7.7).

Figure 7.2 depicts the contour lines of the objective function, and the constraint circle $||u||^2 = u_2^2 + u_3^2 = \chi = 4$. In this figure, two simulated trajectories are plotted, separately of Fantacci's neural network (7.8) and of the neural network (7.7) both starting from point $(2, 4, -1)$. It can be seen that the trajectory of Fantacci's neural network (7.8) converges toward the equilibrium point u^e, which is close to the minimum u^*, while the trajectory of neural network (7.7) converges exactly toward the optimal point u^*. For Fantacci's neural network (7.8), the accuracy of solution, i.e., the closeness of u^e and u^* depends on the value of the penalty parameter K.

In Figure 7.3, the convergence of u_1 with time is illustrated. From this figure, it can be easily seen that the neural network (7.7) converges with more accuracy than the Fantacci's neural network (7.8) within the same period of time and under the same corresponding parameters.

The performance of the neural network (7.7) in a DS-CDMA mobile communication systems is also studied by means of computer simulation. It shows that the neural network is efficient in blind adaptive multiuser detection. Compared with the classic blind adaptive detector [87], the neural network permits to gain better performance in terms of bit error rate (BER).

The simulated system uses 31 chips per bit Gold PN sequences [85] as

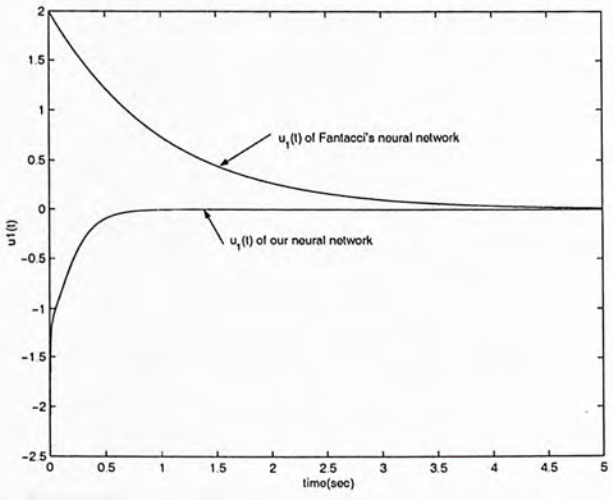


Figure 7.3: Convergence of $u_1(t)$ of two Neural Networks

the signature codes. We assume that it is under perfect power control; i.e., $A_1 = A_2 = \ldots = A_K$. The communcation channel is an additive Gaussian white noise (AGWN) channel and the signal noise ratio (SNR) is 20dB. For brevity, we only detect the first user's signal.

Figure 7.4 depicts the case when only one user is transmitting in the first 4500 bit time periods, and suddenly 19 other interfering users begin to transmit. In this circumstance, the communication channel for the first user at 4500 bit interval is under a rapid change. As shown in Figure 7.4, the neural network (7.7) can rapidly adapt its parameters to the optimum, while the classic blind adaptive detector needs a few steps for its parameters to reach the steady state and therefore in the adaptation period the error bit rate (EBR) is much higher.

□ **End of chapter.**

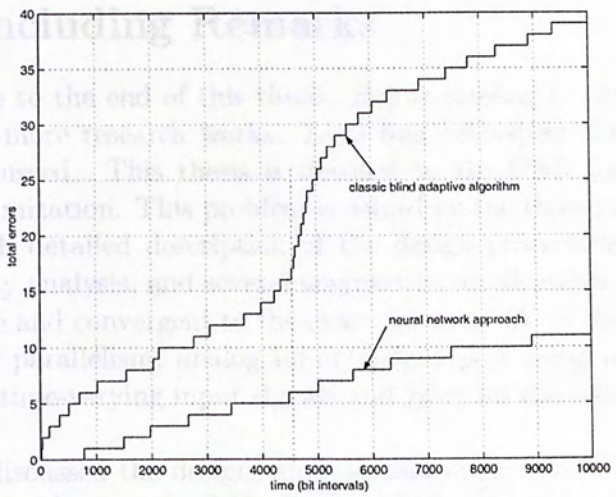


Figure 7.4: Total number of errors versus number of bits

Chapter 8

Conclusions and Future Works

8.1 Concluding Remarks

We have come to the end of this thesis. Before moving to the future works that deserves more research works. Let's first retrospect the content that has been discussed. This thesis is devoted to the RNN for time-varying quadratic optimization. This problem is solved by the three proposed neural networks, with detailed description of the design procedures, convergence and complexity analysis, and several engineering applications. The SDNN is globally stable and convergent to the exact solution. With this capacity and RNN's highly parallelism, analog input/output processing ability, the NN can track the time-varying input signals and gave out the solution as output in real time.

We have discussed the design, analysis and applications. But we didn't have time to implement the NN. So one of the future works that can be done is to design an ASIC which can works as a stand-alone processor or co-processor in the robot control, or wireless communication.

Another possible direction of research work is to extend the work on robot control to multirobot coordinations which can be formulated as equality constraints. Other applications, like vision and hearing perception is also a promising direction of research.

8.2 Future Prospects

Indeed, as the years pass, our ambitions and demands grow, we want to tackle more and more complex problems that are maybe not imaginable in the past. For example, we want to get a multiple sensor equipped computer which has visual, audio, and other perceptions; we want to get a highly dexterous robots

that can cook, care for babies, work together with human being. All these challenges needs for computing machines that can deal with noisy, incomplete and analog input signals. On the other hand, as the technologies advances, its' more and more easy to prototype an circuit desin, which makes the validation easy and cheap. This is a great opportunity for RNN to show its promise in this area with great demands and easy to implementation. Then a very promising research direction is to design and analysis more general RNNs that can tackle more general and complex problems.

□ **End of chapter.**

Bibliography

[1] S. Boyd and L. Vandenbeghe, *Convex Optimization*, Cambridge Uniersity Press, Cambridge, UK, 2004.

[2] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*, Academic Press, New York, 1980.

[3] N.I.M. Gould and etc, "A Quadratic Programming Bibliography," `http://www.optimization-online.org/DB_HTML/2001/02/285.html`, (2001).

[4] D. Goldfarb and A. Idnani, "Dual and Primal-Dual Methods for Solving Strictly Convex Quadratic Programs," *Lecture Notes in Mathematics,* vol.909, Springer-Verlag, Berlin, pp.226-239.

[5] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming,* **27**: 1-33.

[6] A. Krogh, J. Hertz and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, CA, 1991.

[7] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

[8] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to condution and excitation in nerve," *J. Physiol.*, pp. 500–544, 1952.

[9] D. Hammerstrom, "The coming revolution: The merging of computational neural science and semiconductor," *Electrical and Computer Engineering Department, Oregon Graduate Institute.*

[10] M. Mahowald, "Computation and neural systems," *California Institute of Technology*, 1992.

[11] A. Mortara and E. Vittoz, "A communications architecture tailored for analog vlsi artificial neural networks – intrinsic performance and limitations," *IEEE Journal of Neural Networks*, vol. 5, no. 3, pp. 459–466, 1994.

[12] J. Lazzaro and J. Wawrzynek, "A multi-sender asynchronous extension to the address-event protocol," in *16th Conference on Advanced Research in VLSI.*

[13] C. Mead, *analog VLSI and neural systems,* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989

[14] P. Orponen, "An Overview of the Computational Power of Recurrent Neural Networks," University of Jyväskylä, Department of Mathematics, FIN-40351 Jyväskylä, Finland

[15] P. Orponen, "Computational complexity of neural networks: A survey," Nordic Journal of Computing 1 (1994), 94-110

[16] P. Orponen, "Computing with truly asynchronous threshold logic networks," Theoretical Computer Science 174 (1997), 123-136

[17] I. Parberry, *Circuit Complexity and Neural Networks*, The MIT Press, Cambridge, MA, 1994

[18] H.T. Siegelmann, *Neural Netowrks and Analog Computation: Beyond the Turing Limit*, Birkhuser, Boston, MA, 1999

[19] H.T. Siegelmann and E.D. Sontag, "Analog computation via neural networks," Theoretical Computer Science 131 (1994), 331-360

[20] H.T. Siegelmann and E.D. Sontag, "Compuational power of neural networks," Journal of Computer System Science 50, 132-150, 1995

[21] C. Branicky, "Analog computation with continuous ODEs," Proc. Workshop on Physics and Computation 1994, 265-274, IEEE Computer Society Press, Los Alamitos, CA, 1994

[22] J. Burck, "On the convergence properties of the Hopfield model," Proc. of the IEEE 78 (1990), 1579-1585

[23] J. Kroll, "A Survey of Recurrent Neural Networks for Speech Recognition," June 2002

[24] M. Schuster, "Bidirectional Recurrent Neural Networks for Speech Recognition," ATI Interpreting Telecommunications Research Lab, Tokyo, Japan, 1996

[25] S. Parveen and P.D. Green, "Speech Recognition with Missing Data using Recurrent Neural Nets," University of Sheffield, Sheffield, Scotland, UK, 2001

[26] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 5, pp. 533–541, 1986.

[27] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model," *Science*, vol. 233, pp. 625–633, 1986.

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

[29] S. Zhang and A. G. Constantinides, "Lagrange programming neural networks," *IEEE Transactions on Circuits and Systems II*, vol. 39, no. 7, pp. 441–452, July 1992.

[30] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 5, no. 4, pp. 962–971, 1994.

[31] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1544–1547, November 1996.

[32] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 31, no. 1, pp. 147–154, February 2001.

[33] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley, New York, 1993.

[34] J. Wang, "Recurrent neural networks for optimization," in *Fuzzy Logic and Neural Network Handbook*, C. H. Chen (ed.), McGraw-Hill, New York, pp 4.1-4.35, 1996.

[35] Y. Xia and J. Wang, "A general methodology for designing globally convergent optimization neural networks," *IEEE Trans. on Neural Networks*, vol. 9, pp. 1331-1343, 1998;

[36] H. Wang, T. T. Lee, W. A. Gruver, "A neuromorphic controller for a three-link biped robot," *IEEE Trans. on Systems, Man and Cybernetics,* vol. 22, pp. 164-169, 1999.

[37] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems,* Taylor & Francis, London, UK, 1999.

[38] Y. Li and N. Zeng, "A neural network based inverse kinematics solution in robotics," in *Neural Networks in Robotics*, G. A. Bekey and K. Y. Goldberg (eds.), pp. 97-111, Kluwer, Boston, MA, 1993.

[39] S. Lee and R. M. Kim, "Redundant arm kinematic control with recurrent loop," *Neural Networks*, vol. 7, pp. 643-659, 1994.

[40] H. Ding and S.P. Chan, "A real-time planning algorithm for obstacle avoidance of redundant robots," *Journal of Intelligent and Robotic Systems*, vol. 12, pp. 1-15, 1996.

[41] Z. Mao and T. C. Hsia "Obstacle avoidance inverse kinematics solution of redundant robots by neural networks," *Robotica*, vol. 15, pp. 3-10, 1997.

[42] H. Ding and S. K. Tso, "Redundancy resolution of robotic manipulators with neural computation," *IEEE Trans. on Industrial Electronics,* vol. 29, no. 3, pp. 230 - 233, 1999.

[43] H. Ding and J. Wang, "Recurrent neural networks for minimum infinity-norm kinematic control of redundant manipulators," *IEEE Trans. on Systems, Man and Cybernetics - Part A: Systems and Humans,* vol. 30, pp. 269-276, 1999.

[44] J. Wang, Q. Hu and D. Jiang, "A Lagrangian network for kinematic control of redundant robot manipulators," *IEEE Trans. Neural Networks*, vol. 10, pp. 1123-1132, 1999.

[45] W. S. Tang and J. Wang, "Two recurrent neural networks for local joint torque optimization of kinematically redundant manipulators," *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 30, pp. 120-128, 2000;

[46] W. S. Tang, M. Lam, and J. Wang, "Kinematic control and obstacle avoidance for redundant manipulators using a recurrent neural network," *Proc. of Intl. Conf. on Artificial Neural Networks*, Vienna, Austria, pp. 922-929, 2001.

[47] W. S. Tang and J. Wang,"A primal-dual neural network for kinematic control of redundant manipulators subject to joint velocity constraints," *Proc. of Intl. Conf. on Neural Infor. Processing,* Perth, Australia, pp. 801-806, 1999.

[48] Maciekewski, A.A., Klein C.A.: Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamicall Varying Enviorements. International J. of Robotics Research 4 (1985) 109–117

[49] Y. Zhang and J. Wang, "A dual neural network for convex quadratic programming subject to linear equality and inequality constraints," *Physics Letters A*, pp. 271–278, June 2002.

[50] Y. Zhang and J. Wang, "Obstacle Avoidance for Kinematically Redundant Manipulators Using a Dual Neural Network," *IEEE Trans. Syst. Man, Cybern. – Part B: Cybernetics*, vol. 34 (2004) 752-759.

[51] Y. Zhang, J. Wang, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," IEEE Transactions on Neural Networks, vol. 14, no. 3, pp. 658-667, 2003.

[52] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for on-line resolving constrained kinematic redundancy in robot motion control," IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, vol. 35, no. 1, pp. 54-64, 2005.

[53] Y. Xia and G. Feng, "A modified neural network for quadratic programming with real-time applications," *Neural Information Processing*, vol. 3, no. 3, June 2004.

[54] Y. Xia and J. Wang, "A one-layer recurrent neural network for support vector machine learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 34, no. 2, pp. 1–10, April 2004.

[55] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 195, pp. 283–328, 1977.

[56] A. L. Yuille and D. Geiger, *The Handbook of Brain Theory and Neural Networks*, chapter Winner-Take-All Networks, pp. 1228–1231, The MIT Press, 2 edition, 2002.

[57] W. Maass, "Neural computation with winner-take-all as the only nonlinear operation," *Advances in Neural Information Processing Systems*, vol. 12, 2000.

[58] W. Maass, "On the computational power of winner-take-all," *Neural Computation*, vol. 12, pp. 2519–2535, 2000.

[59] W. J. Wolfe, D. Mathis, and et al, "K-winner networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 310–315, March 1991.

[60] J. Wang, "Analogue winner-take-all neural networks for determining maximum and minimum signals," *Int. J. Electronics*, vol. 77, no. 3, pp. 355–367, 1994.

[61] K. Urahama and T. Nagao, "K-winners-take-all circuit with O(N) complexity," *IEEE Transactions on Neural Networks*, vol. 6, pp. 776–778, May 1995.

[62] B. Sekerkiran and U. Cilingiroglu, "A CMOS k-winners-take-all circuit with O(N) complexity," *IEEE Transactions on Circuits and Systems*, vol. 46, no. 1, January 1999.

[63] Jayadeva and S. A. Rahman, "A neural network with O(N) neurons for ranking N numbers in O(1/N) times," *IEEE Transactions on Circuits and Systems I: in press*, 2004.

[64] J. Yen, J. Guo, and H. Chen, "A new k-winners-take-all neural network and its array architecture," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 901–912, Sept. 1998.

[65] B.D. Calvert and C.A. Marinov, "Another k-winners-take-all analog neural network," *IEEE Transactions on Neural Networks*, vol. 11, no. 4, pp. 829–838, July 2000.

[66] C.A. Marinov and B.D. Calvert, "Performance analysis for a k-winners-take-all analog neural network: basic theory," *IEEE Transactions on Neural Networks*, vol. 14, no. 4, pp. 766–780, July 2003.

[67] Y. Nakamura, H. Hanafusa and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *Intl. J. Robotics Research*, vol. 6, pp. 3-15, 1987.

[68] I. D. Walker and S. I. Marcus, "Subtask performance by redundancy resolution for redundant robot manipulators," *IEEE J. Robotics and Automation*, vol. 4, pp. 350-354, 1988.

[69] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE J. Robotics and Automation*, vol. 4, pp. 403-410, 1988.

[70] J.M. Hollerbach and K.C. Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE Trans. on Robot. and Automat.*, vol. RA-3, pp. 308–315, 1987.

[71] A. Nedungadi and K. Kazerouinian, "A local solution with global characteristics for joint torque optimization of a redundant manipulator," *J. Robot. Syst.*, vol. 6, no. 5, pp. 631–654, 1989.

[72] I.C. Shim and Y.S. Yoon, "Stabilized minimum infinity-norm torque solution for redundant manipulators," *Robotica*, vol. 16, pp. 1993–205, 1998.

[73] Y. Zhang and J. Wang, "A dual neural network for constrained torque optimization of kinematically redundant manipulators," *IEEE Trans. Syst. Man Cyber.*, vol. 32, no. 5, pp. 654–662, Oct. 2002.

[74] Y. Zhang, J. Wang, and Y.S. Xu, "A dual neural network for bi-criteria kinematic control of redundant manipulators," *IEEE Trans. Robot. Automat.*, vol. 18, no. 6, pp. 923–931, Dec. 2002.

[75] S. Liu and J. Wang, "A dual neural network for bi-criteria torque optimization of redundant robot manipulators," in *Proceedings of International Conferene on Neural Information Processing.* 2004, vol. LNCS 3316, pp. 1142–1147, Springer-Verlag, Berlin, Germany.

[76] J. Hertz, *The Handbook of Brain Theory and Neural Networks*, chapter Computing with Attractors, pp. 248–252, The MIT Press, 2 edition, 2002.

[77] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Intl. J. Robotics Research*, vol. 4, pp. 109-117, 1985.

[78] F. T. Cheng, T. H. Chen, and Y. Y. Sun, "Obstacle avoidance for redundant manipulators using the compact QP method," *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 262-269, 1993.

[79] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Intl. J. Robotics Research*, vol. 5, pp. 90-99, 1986.

[80] J. Guo and T. C. Hsia, "Joint trajectory generation for redundant robots in an environment with obstacles," *J. of Robotics Systems*, vol. 10, pp. 199-215, 1993.

[81] M. Kircanski and M. Vukobratovic, "Contributions to control of redundant robotic manipulators in an environment with obstacles," *Intl. J. Robotics Research*, vol. 5, pp. 112-119, 1986.

[82] K. Glass, R. Colbaugh, D. Lim, and H. Seraji, "Real-time collision avoidance for redundant manipulators," *IEEE Trans. Robotics and Automation*, vol. 11, pp. 448-456, 1995.

[83] F. T. Cheng, Y. T. Lu, and Y. Y. Sun, "Window-shaped obstacle avoidance for a redundant manipulator," *IEEE Trans. Systems, Man, Cybernetics*, vol. 28, pp. 806-815, 1998.

[84] E. Hons, "Constrained quadratic optimization: theory and application for wireless communication systems," Ph.D. dissertation, University of Waterloo, Waterloo, Ontario, Canada, 2001.

[85] S. Verdú, *Multiuser Detection*, Cambridge University Press, New York, 1998.

[86] U. Madhow, "Blind adaptive interference suppression for Direct-Sequence CDMA," *Proceedings of the IEEE*, vol. 86, no. 10, October 1998.

[87] M. Honig, U. Madhow, and S. Verdú, "Blind adaptive multiuser detection," *IEEE Trans. Info. Theory*, vol. 41, no. 1, July 1995.

[88] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circ. Syst.*, vol. 35, no. 5, pp. 554–562, May 1988.

[89] R. Fantacci, M. Forti, M. Marini, D. Tarchi, and G. Vannuccini, "A neural network for constrained optimization with application to CDMA communication systems," *IEEE Trans. Circ. Syst.*, vol. 50, no. 8, August 2003.

[90] Y. Xia, J. Wang, and L. Fok, "Grasping force optimization for multi-fingered robotic hands using an recurrent neural network," *IEEE Trans Rob. Aut.*, in press, 2004.

[91] Y. Xia and J. Wang, "A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints," *IEEE Trans. Circ. Syst.*, in press, 2004.

[92] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming–Theory and Algorithms*, John Wiley, New York, 2nd edition, 1993.

[93] Y. Wang, J. Li, and L. Jiao, "A novel multiuser detector using the stochastic Hopfield network in CDMA communication system," *Proc. of IEEE International Conf. on Commun., Circ. and Syst.*, vol. 2, pp. 1132 – 1135, 2002.

[94] G.I. Kechriotis and E.S. Manolakos, "Hopfield neural network implementation of the optimal CDMA multiuser detector," *IEEE Trans. on Neural Networks*, vol. 7, pp. 131-141, January 1996.

[95] S. Liu and J. Wang, "Blind adaptive multiuser detection using a recurrent neural network," in *Proceedings of ICCCAS 2004*, June 2004, pp. 1071–1075.

[96] S. Liu and J. Wang, "A Simplified Dual Neural Network for Quadratic Programming with Application for K-Winners-Take-All Operation", submitted to IEEE Transactions on Neural Networks in November 2004.

[97] S. Liu and J. Wang, "A Dual Neural Network for Bi-Criteria Torque Optimization of Redundant Robot Manipualtors", Proceedings of International Conference on Neural Information Processing, LNCS vol. 3316, pp. 1142-1147, Springer-Verlag, Berlin, Germany, November 2004.

[98] S. Liu and J. Wang, "Obstacle Avoidance for Kinematically Redundant Manipulators Using the Deterministic Annealing Neural Network", International Symposium on Neural Networks, May 2005.

[99] S. Liu and J. Wang, "Bi-Criteria Torque Optimization of Redundant Manipulators Based on a Simplified Dual Neural Network," International Joint Conference on Neural Networks 2005.

[100] S. Liu and J. Wang, "A New K-Winners-Take-All Neural Network," International Joint Conference on Neural Networks 2005.

[101] S. Liu and J. Wang, "Obstacle Avoidance for Kinematically Redundant Manipulators Based on an Improved Formulation and a Simplified Dual Neural Network," submitted, June 2005.