INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

NEUROCONTROL OF ROBOTIC MANIPULATORS

by
Pramod Gupta, M.Eng.

A Thesis

Submitted to the School of Graduate Studies

In Partial Fulfilment of the Requirements

For the Degree

Doctor of Philosophy

McMaster University

NEUROCONTROL OF ROBOTIC MANIPULATOR

DEDICATED

to

my parents and my wife

DOCTOR OF PHILOSOPHY (1997)
(Electrical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE: Neurocontrol of Robotic Manipulators

AUTHOR: Pramod Gupta
M.Eng. (McGill University, Montreal, Canada)

SUPERVISORS: Dr. Naresh K. Sinha
Professor Emeritus
Ph.D. (Manchester)

Dr. M.A. Elbestawi
Professor (Mechanical Engineering Department)
Ph.D. (McMaster University)

NUMBER OF PAGES: xiii, 144

ABSTRACT

The thesis is devoted to investigating neurocontrol of nonlinear systems with uncertain and unknown dynamic models. The aim of my research in the neural network area is to search for fast learning algorithm with reduced computation burden. Novel theoretical synthesis and analysis of neurocontrol systems have been conducted and applied to control a robotic manipulator. Modified back propagation learning algorithm making use of the changeable shape of the nonlinear function of node is introduced. The resulting algorithm results in a better accuracy and faster convergence. Neural network based control scheme is used to control the motion of a manipulator. The neural network plays the role of an approximate inverse model of a robot and are then used in conjunction with a conventional proportional plus derivative (PD) controller. To demonstrate the feasibility of the proposed algorithm and neurocontrol scheme, intensive computer simulations were conducted. Different types of adaptive tracking problems and regulation problems are considered. The proposed scheme possesses robustness to systems model uncertainty and changing conditions of operations. Simulation results are quite promising. It is concluded that neural networks, by virtue of their natural ability to learn from data, are well suited for dynamic reconstruction,, bringing the world of nonlinear dynamics closer to practical use.

To demonstrate the practicability of the proposed scheme, experiments were

conducted on an existing two-link manipulator and a single link manipulator. Results confirm the practicability of the proposed scheme.

The thesis concludes that the neurocontrol approach is capable of learning the tasks of reasonable complexity and it should be possible to train a system for a variety of operations using a neural network of practical size.

ACKNOWLEDGEMENTS

I wish to thank many people who contributed in various ways in the completion of my work. My principal thesis advisor; Prof. Naresh K. Sinha, deserves special thanks for this technical guidance and wisdom. It is my pleasure to have Prof. Sinha as an advisor at McMaster. I would also like to thank Professors Simon Haykin, M.A. Elbestawi and Dr. Mike Liu, members of my supervisory committee for their continuing interest and support.

Special thanks are extended to Dr. Gary Bone and Mr. Rafael Bravo of Mechanical Engineering Department who helped me in carrying out the experiments. Many thanks are due to friends and colleagues for their advice and encouragement. I would like to extend my sincere thanks to Ms. Cheryl Gies and Ms. Barbara McDonald of Department of Electrical and Computer Engineering for helping me during my course of stay.

I would like to thank my family especially my parents for always encouraging, supporting and helping me in my studies. Words cannot express my gratitude to them for many sacrifices they have made over the years for my sake. Finally, I am deeply grateful and most indebted to my wife, Shaloo, for her love, patience and support during the course of my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This chapter gives a brief introduction to neural networks and neurocontrol, and forms the motivation of the thesis. Section 1.1 presents a brief overview of neural networks and their applications. Section 1.2 presents the definition of neurocontrol, Section 1.3 discusses the control of robotic manipulators. Section 1.4 presents the motivation of the thesis. Based on the literature reviews, the objectives and the scope of the thesis are defined in section 1.5. In Section 1.6 organization of the thesis is over viewed.

1.1 NEURAL NETWORKS

In recent years there has been an increasing interest in studying the mechanisms and structure of the human brain. This has led to the development of new computational tools known as neural networks, for solving complex problems which are difficult to solve. Neural networks are both numerical model-free estimators and dynamical systems. They have the ability to improve the intelligence of systems working in an uncertain, imprecise and noisy environment. The applications where neural networks have the most promise are those with a real-world flavour, such as, speech recognition, image processing, fast information processing and control of nonlinear systems. In 1986 the parallel distributed processing (PDP)

group in MIT published a series of results and algorithms (Rumelhart and PDP research group 1986). This work gave a strong impetus to the area and provided the catalyst for much of the subsequent research in this field. For a fine collection of key papers in the development of models of neural networks see *Neurocomputing: Foundations of Research* (Anderson and Rosenfeld 1988). Since the work of the PDP group several well-defined architectures have been proposed to tackle a variety of problems. Many examples of real-world applications ranging from finance to aerospace are currently being explored (Hecht-Nielsen 1988). The present work concentrates on the application of neural networks to the modeling and control of nonlinear systems.

A neural network is an information processing system that is nonalgorithmic, nondigital, and massively parallel. A neural network is a adaptable dynamical system whose learning, noise-tolerance, and generalization abilities grow out of its connectionist structure, dynamics, and distributed data representation. A neural network is a statistical associative model. A typical network model shown in Figure 1.1 has a set of input patterns and a set of output patterns. The role of the network is to perform a function that associates each input pattern with an output pattern. A learning algorithm uses the statistical properties of a set of input/output pairs - called the training set - to generalize. With this model, statistical inference can be developed which has some distinct advantages over rule-based inference. Statistical inference allows for exception and randomness in the association between two variables, whereas rules are deterministic. In a neural network

model, the history of the system - that is, what training it has seen - decides the system's response to a new stimulus, but rule-based systems are often nonadaptive, i.e., they do not respond to observed changes in the stimulus environment. Rule-based systems can be made adaptive as well, at the expense of making the rules more complex. It is particularly difficult to develop specific rules for assessing the required time for mastery by each student. A neural network can be defined as (Haykin 1994):

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.

2. Interneuron connection strengths known as weights are used to store the knowledge.

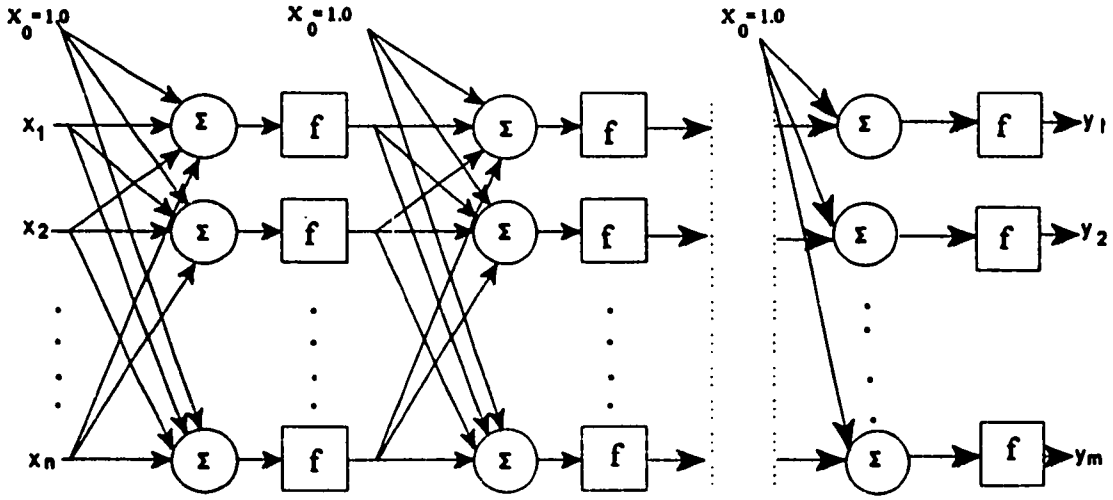

Figure 1.1 A Typical Neural Network with *n* inputs and *m* outputs

The network architecture is defined by the processing elements and the way in which they are connected. The basic processing element of the neural network architecture is often called a neuron by analogy with neurophysiology, but other names such as perceptron (Rosenblatt 1958) or ADALINE (Adaptive Linear Element) is also used. The neurons by themselves are not very powerful in terms of computation or representation but their interconnection allows them to encode relations between the different variables giving powerful processing capabilities. There are many different paradigms available in the literature by which a network may be connected and trained. The important forms of interconnections are:

- Multilayer Feedforward Perceptrons
- Radial Basis Function (RBF) networks
- Associative memory networks
- Cerebellar Model Articulation Controller (CMAC) networks
- Recurrent networks
- Fuzzy neural networks

After initialization, a neural network contains no information reflecting the system it is to approximate. Therefore, at each time instant when new observations are made available, it is desirable to incorporate the additional information into the current parameter estimate. Neural networks learn to solve a problem; they are not programmed to do so. Learning and training are thus fundamental to nearly all neural networks. Training is the procedure by which the network learns; learning is the end result of that

procedure. Training is a procedure external to the network; learning is an internal process or activity. Learning is achieved not by modifying the neurons in the network, but by adjustment of the synaptic weights of the neurons in the network. The procedure for adjustment of the weights is called the learning rule. To start with, when the weights are not calibrated, the network may perform badly at its task. However, after a series of learning processes that the network goes through, the weights are adjusted and the network should perform at a desired level. Typically the learning rules do not change, only the weights do. After the learning period, the weights are usually not changed any further, unless there is a change in the operating environment. Generally in the implementation of a neural network, two regimes can be distinguished: the knowledge acquisition (or training) phase and the knowledge diagnosis (or examination) phase.

Training a neural network may be supervised or unsupervised. With supervised training, the network is provided with an input stimulus pattern along with the corresponding desired output pattern. The learning rule for such networks typically computes an error - that is, how far from the desired output the network's actual output really is. The error is then used to modify the synaptic weights. Unsupervised training involves presenting the data to the network until the network has discovered the emergent collective properties of the data by itself.

Details of various architectures and of various learning algorithms have not been given as they can be found in the references (Albus 1975a, 1975b, Haykin 1994, Hopfield 1982,

Rumelhart *et al.* 1986, Waibel *et al.* 1989).

The distinguishing features of the neural networks which are important from modeling and control point of view are:

1. *Distributed Nonlinearity*

2. Ability to *learn* from experience: They can learn to do tasks based on training data or initial experience, and can adapt to environmental changes

3. *Parallel distributed processing*: They can perform fast information processing

4. *Arbitrary function approximators*: They can be applied to modeling and classification problems

5. *Generalizing* the performance over inputs for which no training has been received

6. *Model-free estimators:* Neural networks are model-free estimators and dynamical systems.

It is clear that a modeling paradigm which has all of the above features has great promise. From modeling and control point of view the ability of neural networks to deal with nonlinear systems is perhaps the most significant.

1.2 NEUROCONTROL

Neurocontrol is defined as the use or study of well-specified neural networks as *controllers*, as devices that output a vector of control signals as a function of time. A neurocontroller performs a special form of adaptive control, with the controller taking the form of a nonlinear multilayer network and the adaptable parameters being the strengths of the interconnections between the neurons. The corresponding neurocontrollers can be implemented on computers or neural net chips. A neurocontroller and the controlled plant

form a neurocontrol system. A neural network makes use of nonlinearity, learning, parallel processing, and generalization capabilities for application to intelligent control. Werbos outlines the five major methods such as supervised control, inverse control, neural adaptive control, backpropagation through time and adaptive critic methods. All of these basic methods have valid applications, reviewed in more detail elsewhere (Werbos, 1992). Most of the approaches to neurocontrol can be considered to belong to one of these five. These methods can be applied, in principle, to any network made up of differentiable functions; they are not restricted to the case of a neural network as such. These approaches can also be used in combination thereby increasing suitability for the control of complex systems.

In supervised control, a neural network learns the mapping from sensor inputs to desired actions adapting to a training set of examples of what it should do. In inverse control, a neural network learns the inverse dynamics of a system. This neural network is used in the control loop. Inverse control has application in the control of a robotic manipulator. In this case, the neural network learns the mapping from the position of a robot arm or something similar, back to the actuator signals which would move the arm to that position. The neural network is then used to make the arm follow a desired trajectory or reach a desired target point.

In neural adaptive control, neural networks are substituted for the mappings used in conventional adaptive control. Conventional adaptive control includes designs like the Self-Tuning Regulator and Model-Reference Adaptive Control. These designs, like inverse control, try to achieve a prespecified target.

In backpropagation through time, the user specifies a utility function or performance measure to be maximized and a model of the external environment. The backpropagation through time is often used to train the recurrent net. Backpropagation is used to calculate the derivative of utility summed across all future times with respect to current actions. These derivatives are then used to adapt the neural network which outputs the actions, or to adapt a schedule of actions.

In adaptive critic methods, the user supplies a function or measure to be maximized. The long term optimization problem is solved by adapting an additional neural network called a critic network, which evaluates the progress that the system is making. In other words, the output of the critic may be seen as a kind of secondary utility function (or its derivatives) that somehow represents the sum of the original utility function across all future time. The network which outputs the actions is adapted to maximize this secondary utility function in the immediate future.

There are three main levels of analysis in neurocontrol. At the lowest level, people try to build supervised learning systems (SLSs). An SLS is any system that learns a nonlinear function or static mapping from a vector **X** to a vector **Y**. At the middle level, in neurocontrol proper, one builds a complex systems made up of SLS components, and other similar components; one tries to develop general purpose designs to perform the tasks of tracking or dynamic optimization. Finally, in applications research, one uses neurocontrol systems in combination with other systems to build complex systems for specific applications. The books *Neural Network for Control* (Miller *et al.* 1990) and *Neural Network Applications in*

Control (Irwin *et al.* 1995) provide a broad overview of the field.

1.3 ROBOT CONTROL

Robotics is a relatively new field of modern technology and is becoming increasingly important in the field of flexible manufacturing. Understanding the complexity of robots and their applications require knowledge of electrical engineering, mechanical engineering, industrial engineering, computer engineering, economics and mathematics. Robotics is the science and engineering dealing with the design and application of reprogrammable multi functional manipulators to move objects through variable programmed motions for the performance of a variety of tasks. Robotics constitutes the study of a finite number of rigid mechanical links which represent a multivariable nonlinear coupled system. A robot manipulator is typically modeled as a serial chain of n rigid bodies. In general, one end of the chain is fixed to some reference surface while the other end is free, thus forming an open kinematic chain of moving rigid bodies. Dynamics of manipulators involves nonlinear mapping between joint torques applied and the robot and the joint positions, velocities and accelerations. Dynamics of a robotic manipulator consists of a set of second-order, nonlinear and highly coupled differential equations with uncertainty as a robot may work under unknown and changing environments and execute different tasks.

The common control tasks involved in robot manipulators are: Position and orientation control, trajectory control, force/moment control, constrained motion control. A position control problem is the problem of determining the time history of joint inputs required to cause the end-effector to execute a commanded motion. If the motion of the end-

effector is constrained, for example, to move along the surface of a table, then the motion is referred to as constrained motion. The motion of the manipulator must comply with the environmental constraints. Often the compliant motion is associated with a task such as an assembly requiring fine motion control. The design of a controller for a given manipulator is based on the design objectives and a dynamical model. The objective in designing a controller for free or constrained motion is to make the center point and the orientation of the end-effector follow the desired trajectory. For controlling the motion of the manipulator along a desired path, the actuator torque functions have to be calculated from the equations of motion. This is a difficult task as it requires incorporating effects of inertia, coupling between joints (Coriolis and centrifugal forces), gravity loading and backlash, gear friction and the dynamics of the control devices. Inverse dynamic approach is particularly important for control, since it allows to compensate for highly coupled and nonlinear arm dynamics. As stated before, the dynamic model of a manipulator is described by nonlinear coupled differential equations. It is difficult to determine the precise dynamical model of the robot.

Many strategies have been developed for controlling its motion. The particular control method chosen as well as the manner in which it is implemented can have a significant impact on the performance of the manipulator and consequently on the range of its possible applications. For example, continuous path tracking requires a different implementation in terms of the hardware and software of the computer interface than does point-to-point control. In addition, the mechanical design itself will influence the type of control scheme needed. For example, the control problems encountered with a Cartesian manipulator are

fundamentally different from those encountered with an elbow type manipulator. Existing robotic manipulators use simple Proportional - Integral - Derivative (PID) controllers whose gains are tuned to make the manipulator critically damped. Controllers with fixed gains suffer from the need to retune the gains periodically. Even if the optimum gains are initially selected, changes in the process dynamics warrant constant retuning. This can result in a poor part quality and frequent shutdown. The inability to achieve and maintain an accurate linear approximation of the true nonlinear process dynamics prevents accurate and efficient control. There are advanced modern approaches to design controllers for robot motion control. They include computed torque control, robust control, model based adaptive control and variable structure control. However, model based adaptive control is too complicated and expensive for industry to use. A heavy computational burden impedes it for real-time control applications. On the other hand, the computed torque method needs an accurate dynamic model which is not always available especially when the robot is performing under different operating conditions. Moreover, variations of payload and low precision of manipulator parameters restrict the usefulness of the computed torque method. Adaptive approaches have been proposed to overcome this problem (Craig *et al.* 1987, Slotine and Li 1987). These adaptive methods have the advantage that, in general, they require no *a priori* knowledge of robot dynamics. A general drawback for the adaptive controller is that the computational requirements for real time parameter estimation are high and sensitivities to numerical precision and measurement noise creates problems. In summary we can say that conventional control methods for manipulators suffer from two difficulties. First we must

have detailed *a priori* knowledge of individual manipulators. Secondly, the computational load is very heavy when advanced control methods such as computed torque and adaptive control are used to achieve good performance. Thus, there is a need to develop a controller that can "learn" to control a robotic manipulator.

1.4 MOTIVATION FOR NEUROCONTROL

1.4.1 Challenges in Robotic Control

A general trend in designing robotic manipulators is to make them lightweight relative to their payloads, able to work at higher speed with high precision, adaptable to different tasks, and efficient in applications. The dynamics of these advanced manipulators cannot be simply modeled as rigid bodies. The dynamic modeling errors due to neglecting the structural flexibility of various components in mechanical manipulators have a significant effect on the performance of robot systems, and sometimes even lead to system instability if they are not taken into account in the controller design. It has been observed that joint flexibility plays a significant role in determining the end deflection of robot arms. The actuators themselves also exhibit some flexibility. Another difficulty is structural and parametric uncertainty. Parametric uncertainty may arise from irregular shapes of the robotic components, nonuniform materials, non-symmetric motor or transmission installation, part wear-out and pay-load changes. Structural uncertainty may result from neglected actuator dynamics, internal moving parts, friction and backlash, and external disturbances. Friction is present at the joint and difficult to model. Sudden control action may excite unmodelled high frequency characteristics, such as link flexibility. Non-symmetric motor axes, for example, result in coupling between link

dynamics and motor dynamics. Finally, the desired motion is usually specified in terms of the Cartesian space coordinates, whereas the torques needed for the motion are applied by actuators at the joint level. Thus, a transformation from Cartesian space to joint space is required. This is computationally intensive.

1.4.2 The Potential of Neurocontrol

Neural network computing systems with capabilities for supervised learning, matching, and generalization are being developed and explored in a variety of simulated and experimental contexts. Robotic systems offer a promising domain for this exploration since the practical application of complex robotic systems may require adaptive and learning behaviour in order to achieve their desired functionality. Technical issues which must be addressed in order to expand the capabilities of robotics and automation technology are grouped into four separate areas: mechanisms, control, representation and planning, and architecture and implementation. While the mechanisms themselves are not directly related to the implementation of adaptive and learning systems, it is clear that improvements in sensing technology, motor technology, and new mechanisms such as flexible arms and sophisticated hands, will place increasingly stronger demands on the corresponding control and planning systems to incorporate adaptive capabilities for utilization in specific tasks. There are important opportunities in the development of more robust controllers by utilizing learning systems to more accurately identify robot kinematics and dynamics, to more efficiently adapt dynamic control parameters to particular tasks, and to more effectively integrate sensory information into the control process. The capability to adapt to an inherently

uncertain representation or model of the task, is the key to the improved reliability of these systems. An automated system for learning of accurate kinematic or dynamic parameters of current robot arms, would have immediate impact interims of the potential performance of these arms. Newer systems that incorporate lightweight, flexible arms or multi arm interactions will also require such on-line identification in order to function effectively. Thus, there is a need for an intelligent controller that can learn how to control a dynamic system. Recently, there has been considerable progress in the field of intelligent control systems. Recent innovations in learning control using neural networks display much potential for nonlinear dynamic processes. Neural networks can adapt to nonlinear and changing processes, can integrate sensory information, and are robust in the presence of noise. These and other strengths may enable neural networks to address the need of existing and future manufacturing systems.

Advances in nonlinear function approximation methods (e.g., neural networks) and the application of these methods to robotics systems may yield substantial payoffs in more accurate and adaptable control of engineering systems. Neural network-based function approximation techniques are capable of learning complex nonlinear control mappings, may be robust in the presence of noise, and behave smoothly to ensure process stability. Neural networks can be used to model the quantitative relationships of process parameters and, when combined with the qualitative knowledge of the process engineer, can be used to develop adaptive nonlinear process control systems. The implementation of efficient on-line learning techniques that may achieve stable operation of an unmodelled nonlinear process, with noisy

feedback, in a relatively short span and at reasonable computational expense, will make simple, adaptive, accurate, and cost-effective control of complex processes possible. On-line learning provides the flexibility to learn the control of an unknown, dynamically changing, multivariate processes with arbitrary inputs and feedback data. The rigid constraints imposed by real-time learning strongly favour the use of a neural network paradigm that converge very quickly and require a minimal amount of computational overhead. Off-line learning may be used to pretune a learning control system to reduce the start up oscillations.

The main advantages of neurocontrol approach are: (i) ability to deal with difficulties arising from uncertainty, and noise; (ii) model-free estimator and dynamical system, i.e., quantitative system models are not required to design a neurocontroller; (iii) ability to model the controlled system itself; (iv) it is easy to incorporate any a priori information about controlled systems into neurocontroller designs; (v) controllers are based on learning from input-output measurements and not on parametric model-based dynamics. In robotics, e.g., it may be possible to implement a complete inverse dynamics model of the robot without any need of parametric model-based dynamics. In fact there is no need to estimate the parameters of the system as in the case of direct and indirect adaptive controls

Based on the existing problems of nonlinear control, the potentials of neurocontrol, and incapability of the existing approaches, this thesis proposes to study neurocontrol and apply it to control a robotic manipulator.

1.5 OBJECTIVES AND SCOPE OF THE THESIS

The objectives of my thesis are: (1) to develop an algorithm which has faster rate of

convergence with better learning performance, and (2) to apply the neural network-based scheme to the trajectory control of a robotic manipulator.

Based on the above objectives, the scope of the thesis is defined as follows.

1) Modification of Backpropagation Learning Algorithm

Conventional backpropagation learning algorithm suffers from a slow rate of convergence. A modified backpropagation algorithm based on the use of an independent, adaptive learning rate parameter for each synaptic weight with adaptable nonlinear function has been introduced.

2) to develop a robust neurocontrol scheme for the control of a robotic manipulator

Due to the existence of modelling errors, most conventional control schemes for robotic manipulators' result in limited control precision. A neurocontrol scheme used in the thesis suggests that neural network-based scheme is quite robust to the changing conditions of operation. Simulation results suggest that the neural network-based scheme is quite promising.

3) to test the real-time features of a neurocontrol scheme by an experiment

For the validation of the results obtained by simulations, experiments were conducted using an available two-link manipulator and a single motor.

1.6 ORGANIZATION OF THE THESIS

The remaining chapters of the thesis are organized as follows. Chapter 2 presents literature review on neurocontrol and its application to robotic control. Based on the review, the objectives and scope of the thesis are proposed and defined.

Chapter 3 presents a modification to the conventional backpropagation learning algorithm. The new algorithm results in the fast rate of convergence and reduced computation. Also, different training strategies are fully investigated and compared. Based on these, a training method is selected, which is most suitable for applications in this thesis.

Chapter 4 presents modeling of dynamics of robotic manipulators. Simulation results show the effectiveness of the method.

In Chapter 5, motion control of robotic manipulators is studied using a neurocontrol scheme based on backpropagation. The employed neural network is trained as an approximate model of the robot. The learning process gradually transfers the control action from a feedback PD controller to the neural controller. The control performance is improved greatly. Simulation results show that the suggested method is promising.

Experimental tests are performed on a two-link manipulator to evaluate the real-time performance of the neurocontroller. The results are presented in Chapter 6.

The thesis concludes with Chapter 7, where a summary of contributions and suggestions for future work are given.

CHAPTER 2

LITERATURE OVERVIEW

2.1 INTRODUCTION

Neural network computing methods provide one approach to the development of adaptive and learning behaviour of robotic systems. Neural networks are capable of learning to control an unknown plant by extracting the necessary information from the plant. This chapter provides an overview of the current research in the application of neural networks to the control of robotic manipulators. Applications of some neural network architectures in robot control are surveyed. The chapter discusses several approaches, indicating distinctive features and successful demonstrations. The chapter is organized as follows: A brief overview of robotics is provided in Section 2.2. Section 2.3 deals with the neurocontrol and Section 2.4 provides the review of the neurocontrol of robotic manipulators.

2.2 AN OVERVIEW OF ROBOTICS

Robotics is a relatively new field of modern technology and is becoming increasingly important in the field of automated manufacturing, including flexible automation. It constitutes the study of a finite number of rigid mechanical links which represent a multivariable nonlinear coupled system. Major problem areas in robotics include trajectory

planning, kinematics, dynamics, and control. Details of the issues in robotics can be found in (Craig 1986, Fu *et al.* 1987, Spong and Vidyasagar 1989). The purpose of robot arm control is to maintain the dynamic response of the manipulator in accordance with some prespecified performance criterion. The solution of this problem is difficult because this requires the solution of nonlinear coupled differential equations which are complex functions of joint variables. Although our main interest is in control, the knowledge of some aspects of kinematics and dynamics of robots is essential for our purpose.

Kinematics is the study of motion without regard to the forces or torque which cause the motion. Kinematics of manipulators refers to all of the geometrical and time-based properties of motion. Two subproblems are distinguished: the *forward kinematics problem* and the *inverse kinematics problem*. The forward kinematics problem can be stated as: Given the joint variables of the robot, determine the position and orientation of the end-effector. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link's extension in the case of prismatic or sliding joints. It involves a nonlinear mapping from joint space of the robot to Cartesian space and can be represented as

$$\boldsymbol{x} = \boldsymbol{f}(\theta) \tag{1}$$

where θ is the vector of individual joint variables, and $\boldsymbol{x}$ is the end-effector location in Cartesian space. The function $\boldsymbol{f}$ is nonlinear and consists of some trigonometric functions as well as matrix multiplications. The computation is relatively straightforward. On the other hand, an inverse problem is somewhat difficult. The inverse kinematics problem can be stated as follows: Given a desired position and orientation for the end-effector of the robot,

determine a set of joint variables that achieve the desired position and orientation, that is:

$$\text{Given } x, \text{ find } \theta \text{ such that } f(\theta) = x$$

Since the kinematic equations are nonlinear, the problem does not have a unique solution. This is a mathematically ill-posed problem; since the forward kinematic mapping $f(\theta)$ is many-to-one, $f^{-1}(x)$ will not be unique. The inverse kinematics problem may be solved either explicitly, by directly computing a closed-form regularized inverse mapping $f^{-1}(x)$, or implicitly (i.e., numerically), by, for example, the use of differential methods. A solution can be obtained in a closed form using the spatial geometry of the manipulator, or by solving the matrix algebraic Eq. (1). Because of the complex nature of Eq. (1), there are cases wherein a closed from solution does not exist. For nonredundant manipulators which do not have closed form solution, or for those manipulators which have redundant degrees of freedom, numerical methods are commonly used. Due to their iterative nature, numerical solutions are generally much slower than the corresponding closed-form solution.

Dynamical models of a manipulator specify the equations of motion relative to a chosen coordinate system. In many applications, the motion of the end-effector is of primary interest. The center of the end-effector traces a path in the world coordinate system when the manipulator is moved by the actuators. Controlling this path is called the gross motion control. It can be determined based on the dynamical model of the manipulator. Robot arm dynamics deals with the mathematical formulations of the equations of robot arm motion. The dynamic equations of motion of a manipulator are a set of mathematical equations describing the dynamic behaviour of the manipulator. Such equations of motion are useful for computer

simulation of the robot arm motion, the design of suitable control equations of a robot arm, and the evaluation of the kinematic design and structure of a robot arm.

The actual dynamic model of a robot can be obtained from known physical laws such as the laws of Newtonian mechanics and Lagrangian mechanics. This leads to the development of the dynamic equations of motion for the various articulated joints of the manipulator in terms of specified geometric and inertial parameters of the links. Conventional approaches like the Lagrange-Euler and Newton-Euler formulations could then be applied systematically to develop the actual robot arm motion equations. Euler-Lagrange's equations of motion are obtained on the basis of Lagrange's energy function. The resulting differential equations describe the motion in terms of the joint variables and the structural parameters of the manipulator. An alternative approach to the modelling of the manipulator dynamics is to consider each link as a free body and obtain the equations of motion for each link in succession on the basis of Newton's and Euler's laws. Thus, the recursive differential equations of motion can be determined for the entire manipulator with serial links. The general form of the dynamic equations of a robotic system can be written as:

$$\tau(t) = M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + G(q(t)) \qquad (2)$$

where $\tau(t)$ is a $n \times 1$ vector of joint torques

$q(t)$ is a $n \times 1$ vector containing the joint angles

$M(q(t))$ is the inertia matrix of the manipulator

$C(q(t), \dot{q}(t))$ is the contribution of the Coriolis and centrifugal forces and

$G(q(t))$ is the gravitational torque due to the gravity.

Both formulations are based on fundamental physical laws of dynamics and provide the designer insight in the dynamical behaviour of the system. This is an important aspect in the engineering, analysis and design. When Lagrange's formulation is used, the designer has the dynamical model for the whole robot system. Therefore, the interactions between the variables and the couplings between the dynamical equations of the joints are quite apparent. When Newton-Euler's formulation is used, the effects of external and internal forces and torques on the single link are transparent to the designer. In many cases, however, it is difficult to visualize the interrelationships between different parts of the entire robot system, because they don't appear in the model explicitly. The choice of the model used in a study is usually dependent on the specific application and task to be performed by the robot manipulator system.

For controlling the motion of the manipulator along a desired path, the actuator torque functions have to be calculated from the equations of motion. The equations of motion obtained for a manipulator can be used to solve the problem of inverse dynamics, i.e., to determine the generalized joint torques when the desired positions, velocities, and the accelerations of the joints are specified. It is difficult as it requires incorporating effects of inertia, coupling between joints (Coriolis and centrifugal forces), gravity loading and backlash, gear friction and the dynamics of the control devices. The inverse dynamic approach is particularly important for control, since it allows to compensate for highly coupled and nonlinear arm dynamics Solution of equations of motion requires large numbers of trigonometric and nonlinear functions of the joint coordinated, velocities and accelerations.

Hence, it is considerably more complex than the kinematic problem.

2.3 LITERATURE REVIEW ON NEUROCONTROL

Models of dynamic systems and their inverses have immediate utility for control. Direct inverse control utilizes an inverse system model. The inverse model is simply cascaded with the system to be controlled in order that the whole system results in an identity mapping between desired response (i.e., the network input) and the controlled system output. Thus, the network acts directly as the controller in such a configuration. A neural network is trained to minimize some error function. Since the desired action is unknown, it is not possible to determine the error signal required to train the network by back propagation algorithm. Hence, appropriate training schemes need to be developed for the use of neural networks for direct control. Psaltis *et al.* (1987, 1988) proposed learning architectures for training the neural network inverse dynamic controller to provide the appropriate inputs to the plants so that the desired output is obtained. The schemes are shown in Figures 2.1 to 2.3.

2.3.1 Indirect Inverse Neurocontrol

Figure 2.1 shows the feedforward controller implemented as neural network architecture, with its output driving the plant. The real plant output is used as input to a copy of the neural network. The difference between the two networks' outputs is a measure of the controller's error and is used to adapt weights of both networks so that this difference is minimized. This learning architecture is called indirect learning. The main feature of this scheme is that the network is trained only in the region of interest since we start with the desired response and all other signals are generated from it.

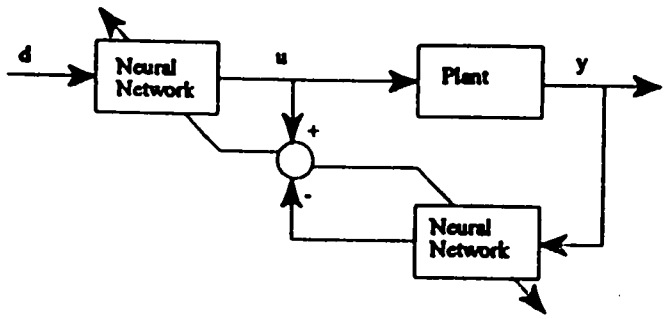

Figure 2.1 Indirect Learning Architecture

2.3.2 Direct Inverse Neurocontrol

Another type of learning architecture is based on an explicit learning stage while the controller uses a neural network as a simulator in order to learn its behaviour. The controller knows what should have been the output values of the plant and uses the difference to adapt itself. This method is known as general learning. The scheme is shown in Figure 2.2. The success of this method depends on the ability of the neural network to generalize correctly so that it may respond to inputs for which it has not been trained. Thus, the training samples will have to cover the entire input space of the plant. Evidently this procedure is not efficient since the network will have to learn the responses of the plant over a wider range than what may be actually necessary. The other disadvantages of this scheme are (1) the controller is not operational during the learning stage, (2) the plant is assumed to be a static physical process and (3) if the nonlinear system mapping is not one-to-one then an incorrect inverse can be obtained. These drawbacks can be avoided by using the specialized learning approach

shown in Figure 2.3.

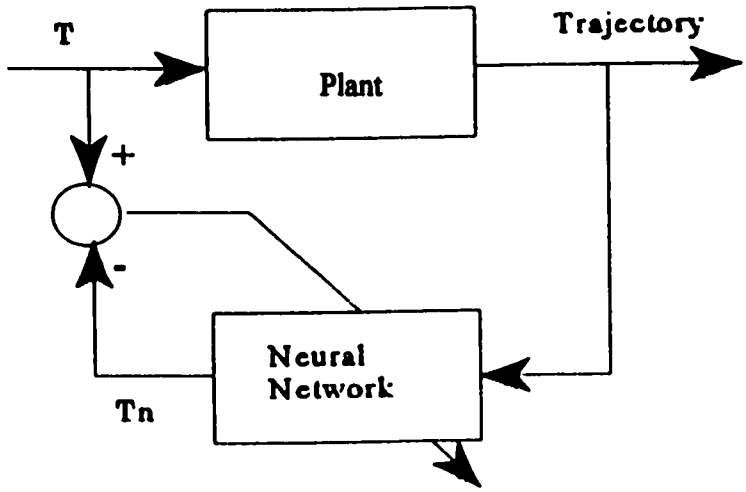

Figure 2.2 Learning Scheme of an Inverse Dynamical Model

2.3.3 Specialized Inverse Neurocontrol

In this approach the network inverse model precedes the system and receives as input a training signal which spans the desired operational output space of the controlled system (i.e., it corresponds to the system reference or command signal). In this case, the controller learns by directly evaluating the accuracy of the network. The error between the actual and desired output of the plant is used to change the weights of the network. The controller learns continuously and therefore it is able to control plants with time varying characteristics. Psaltis *et al.* (1988) proposed that the plant can be viewed as an additional and not modifiable layer of the neural controller. This method, however, requires the knowledge of the plant Jacobian since the error has to be back propagated through the plant before it can be used to update

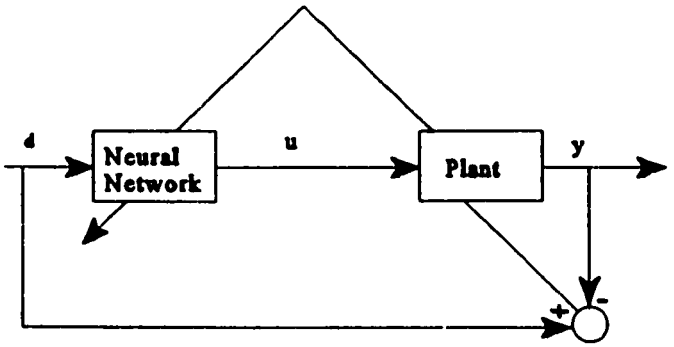

Figure 2.3 Specialized Learning Architecture

the weights. The plant Jacobian can be obtained in a neural network form (Jagannathan and Lewis 1996, Jin *et al.* 1993). This architecture can specifically learn in the region of interest, and it may be trained to fine-tune itself on-line while actually performing useful tasks. In comparison with generalized learning the specialized learning approach has the following features:

- The procedure is a goal directed since it is based on the error between desired system outputs and actual outputs. In other words, the system receives inputs during training which correspond to the actual operational inputs it will subsequently receive.
- In cases where the system forward mapping is not one-to-one a particular inverse suitable to the particular application will be found. Jordan and Rumelhart (1992) discuss ways in which learning can be biased to find particular inverse models with desired properties.

Psaltis *et al.* (1988) recommended that a hybrid approach where a general learning is first performed to learn the approximate behaviour of the plant, and then specialized learning is performed to fine tune the network in the operating regime of the system. General training

will have a tendency to create better initial weights for specialized training. Thus, starting with general training can speed the learning process by reducing the number of iterations of ensuing specialized training.

Narendra and Parthasarthy (1990) analysed in details neural network-based models for both identification and control. They proposed four different neural network models for discrete nonlinear systems. These generalized neural network models result from a unified treatment of multilayer and recurrent neural network models. They show that neural networks can be used effectively in the identification and control of nonlinear dynamic systems.

2.3.4 Robust Neuorcontrol Scheme

Model-based neurocontrol optimization is a nominal control design procedure without an on-line adaptive component, it does not allow for plant drifts or other factors that could adversely affect the performance of the control system. In fact, a controller that is highly optimized for a specific process cannot be expected to tolerate deviations from the nominal process gracefully. However, model-based neurocontroller design can incorporate robustness. This approach combines conventional robust control schemes with neural network models. Neurocontrollers can be optimized for robust performance. The scheme is shown in Figure 2.4.

Raibert (1978) proposed a model of the central nervous system that is capable of learning the dominant dynamic parameters of a limb while performing a specific movement. He uses the idea of a state vector addressable memory to store this parametric data during learning. This data was then accessed as a look-up table to generate parameters

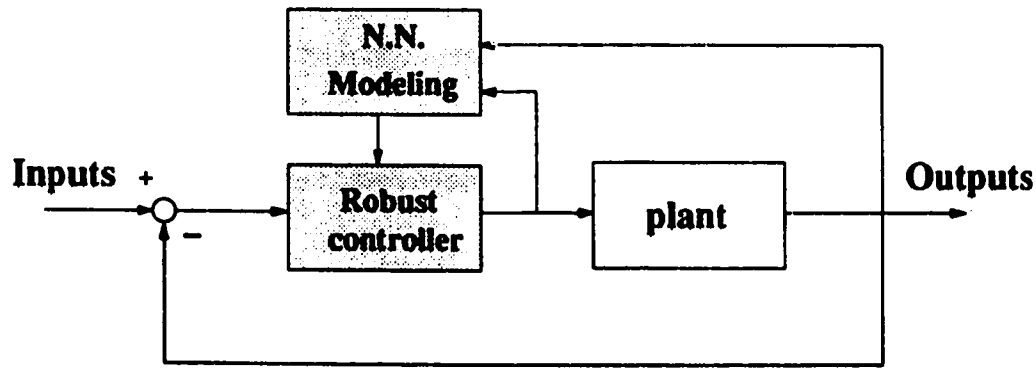

Fig. 2.4 The Block diagram of robust neurocontrol

necessary to compute joint torques for the desired movements. Albus (1975a, 1975b) generalized this approach in his cerebellar model articulation controller (CMAC) to store parametric data in locally overlapping regions so that each table entry is influenced by neighbouring data points during leaning. Atkeson and Reinkensmeyer (1988) used an alternate approach in which the parametric data is stored locally during learning but the parameter value obtained during recall is averaged over neighbouring entries. All of these techniques essentially store parametric data in a large look-up table

Jacobs and Jordan (1993) described a multi-network, or modular, neural network architecture that uses a piecewise control strategy to perform control tasks. The architecture's network competes to learn the training patterns. As a result, the parameter space of the plant is adaptively partitioned into a number of regions, and a different network learns a control law in each region. The main advantage of modular networks is that they can be structured more easily then fully connected networks. Simulation results

show that the modular architecture performs quite well on a multi-payload robot motion control task.

There is also reinforcement neurocontrol, adaptive critic neurocontrol, fuzzy neurocontrol hybrid and hierarchical neurocontrol etc. The structures of these neurocontrol systems are similar to the above schemes. However, their learning algorithms may have great differences. The learning strategies can be classified into pure on-line learning, pure off-line learning, and hybrid on-line and off-line learning. In pure off-line trained neurocontrollers, the system performance is dependent on the generalization property of the neurocontroller. Pure on-line trained neurocontrollers have bad initial control performance. Hybrid on-line and off-line trained neurocontrollers share the merits and demerits of both.

2.4 NEUROCONTROL OF ROBOTIC MANIPULATORS

One goal in robotic research is to develop classes of robots capable of duplicating human performance in uncertain situations. This requires the ability to adapt to variations in various parameters throughout the operation. The mechanism responsible for this ability is the control system. A basic control block diagram for a robot manipulator is shown in Figure 2.5.

Control of manipulators involves solving coupled nonlinear differential equations. As more degrees of freedom and nonlinearities are introduced these become less and less tractable. The manipulation tasks performed by living organisms seem to suggest that solving differential equation is a very inefficient way of controlling the manipulator. Connectionist approaches to robot control are grounded in these observations of biological systems. The

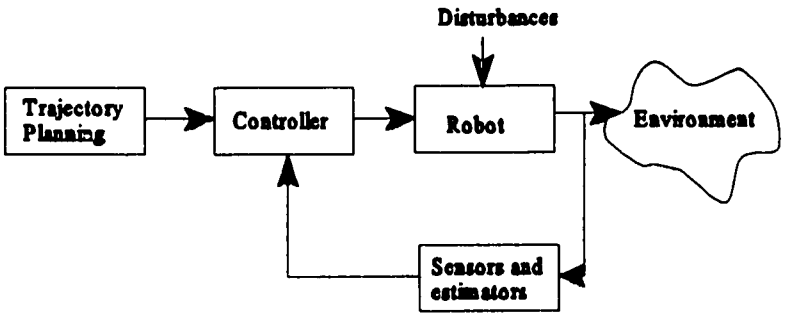

Fig. 2.5 Basic Control diagram for a robot manipulator

emphasis is on learning the mapping between the various variables without an accurate knowledge of the system parameters or the equations governing the system. Neural networks, which are not model based, have become recognized as viable alternatives to traditional identification and control of robotic manipulators. The most promising property of neural networks is their ability to adaptively learn complex mappings. One of the advantages of this property is that it allows us to avoid deriving some closed form analytic function by hand. But more importantly, the system could learn mappings which are mathematically intractable. In addition, the system would be portable since it adapts to the robot to which it is applied. Moreover, as neural networks have the ability to learn, they can be trained to emulate human skills, utilize qualitative control rules, and adapt to environmental and system internal changes. Therefore, they offer an exciting alternative for solving the robot control problem by just learning from examples of the robot's behaviour.

Neural network models have some features particularly useful in a robotic setting: they can be trained 1) to handle the inverse and forward kinematics; 2) to represent any

complex relationship between inputs and outputs of n-joint robot arms; i.e., dynamics of a robot can be modelled; and 3) to design any trajectory for a robot as a model reference. Much research effort has been put into the design of neural network applications for manipulator control. All these results showed that neural networks do have the potential to overcome the difficulties in manipulator control experienced by classical control theory.

2.4.1 Neural Networks and Kinematics of Robots

The problem of finding the joint variables from a specification of the desired end effector's position and orientation is called the inverse kinematic problem. It is difficult because it is computationally intensive and has multiple solutions. Moreover, it becomes more complex as the degree of freedom of the manipulator increases. Also, the robot kinematic parameters need to be known accurately. Neural networks offer an exciting alternative for kinematic control of a manipulator. There has been a growing interest in this area for some time. Attempts have been made to train neural networks for learning inverse kinematics based on the capability of neural networks to store input-output associations or to approximate input-output mapping.

Elsley (1988) reported the use of neural networks in the solution of an inverse kinematic problem. He reported that the average number of steps required for the manipulator to reach the desired position and the average error were more for the analytically calculated exact inverse Jacobian method of control. This is due to the fact that the inverse Jacobian is optimum for small motions, but not for large ones. The neural network learns the Jacobian for large motion.

Guo and Cherkassky (1989) presented a solution algorithm, using a neural computational scheme to implement the Jacobian control technique in real time as often required in practical control problems. They have used the neural computation scheme of Hopfield and Tank to implement the Jacobian control technique. Their method has an advantage over numerical iterative algorithms in that the solution time is independent of the number of degrees of freedom.

Guez and Ahmed (1989) presented a hybrid approach using a multi-layered feedforward network (MFN) to the iterative solution of robotic manipulators which resulted in accelerated convergence in the inverse kinematics. They showed that by employing a three layered a neural network; the inverse kinematic problem can be solved by learning. Neural network solution is used as an initial guess to the iterative procedure (Newton-Raphson or Gradient descent method) which provides the final solution. An MFN was trained in a supervised manner using back propagation algorithm. They carried out a test on PUMA 560. The proposed method combines the advantages of neural networks and iterative methods. The method is independent of the type of manipulator and simple to implement. The proposed method is also computationally efficient.

Lee and Kil (1994) proposed a method for inverse kinematic solution based on the iterative update of joint vectors. The proposed method is based on the Jacobian transpose that uses pseudo inverse of the gradient of a Lyapunov function in joint space to update the joint vector. The proposed neural network consists of a feedforward network and a feedback network forming a recurrent loop. The feedforward network computes the forward kinematic.

The feedback network is derived from the feedforward network and computes joint vector updates. The proposed scheme has advantages over conventional neural networks for IKP in that it provides an accurate computation of forward and inverse kinematic solution with very simple training. Simulation results demonstrate that the proposed method is effective for real-time kinematic control of a redundant arm, as well as for real-time generation of collision-free joint trajectories.

A neural network theory is applied to theoretical robot kinematics to learn accurate transformation between joint variables and the Cartesian coordinates of position. The network is trained on accuracy data that characterize the actual robot kinematics. The network learns the differences in the joint variables to improve the accuracy between an endpoint from theoretically calculated joint variables and the desired endpoints. Results show that a neural network, as discussed here, can increase both the accuracy and the positional repeatability of robots. In summary, these results confirm that neural networks can solve coordinate transformation problems and arbitrary input-output mapping problems.

2.4.2 Neural Networks and Dynamics

The nonlinear mapping property of neural networks is ideal for learning robot dynamics. The basic idea is that the neural network learns the inverse dynamical relationship of the robot directly so as to map the nonlinear relationship between the robot joint torques and joint variables (position and speed). Thus, it can be used as an inverse dynamics controller. A general diagram for a control system is shown in Figure 2.6. The feedback and feedforward controllers and the filter can all be implemented as multilayered neural networks.

The learning process gradually tunes the synaptic weights of the neural network so that the error between the desired and actual plant responses is minimized. Since the error signal is the input to the feedback controller, the training of the network will lead to a gradual switching from feedback to the feedforward action as the error signal becomes small.

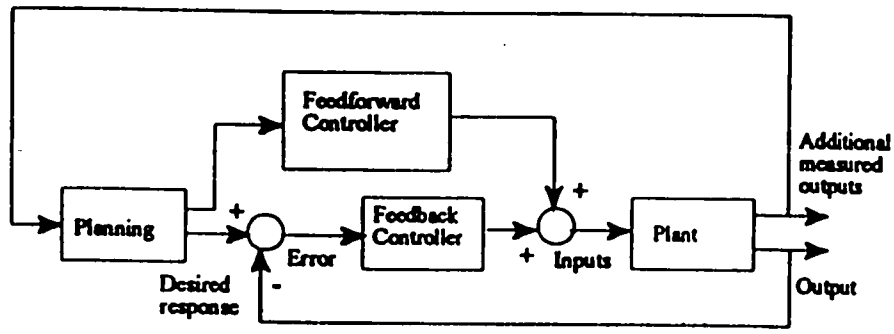

Figure 2.6 General Control System Diagram

2.4.3 Inverse Dynamic Robotic Control

Kawato *et al.* (1988) proposed a hierarchical neural network model based on physiological models. It contains internal neural models of dynamics and inverse-dynamics of the musculoskeletal system as essential learning parts. To check the efficacy of the proposed neural network model to learn control of an object with highly nonlinear coupled dynamics with multiple degrees of freedom they chose an industrial robotic manipulator with two degrees of freedom. They show that once the neural network model learns some movement, it could control quite different and faster movements also. The reason for this success is that the proposed model learns the dynamics as well as the

inverse dynamics of a control object instead of a specific command or a movement pattern. The model can adapt to sudden changes in the dynamics of the controlled system.

Miyamoto *et al.* (1988) have successfully applied the proposed method to control an industrial manipulator (Kawasaki-Unimate PUMA 260) with the neural network model in a microcomputer (Hewlett-Packard 9000-300-320). Their neural network-based scheme is shown in Figure 2.7. The total torque $T_t(t)$ applied to an actuator of the manipulator is a sum of the feedforward torque $T_i(t)$ generated by the inverse dynamic model, and the feedback torque $T_f(t)$. The desired trajectory is input to the model. The feedback torque $T_f(t)$ can be considered an indication of the inaccuracy of the inverse model. This signal

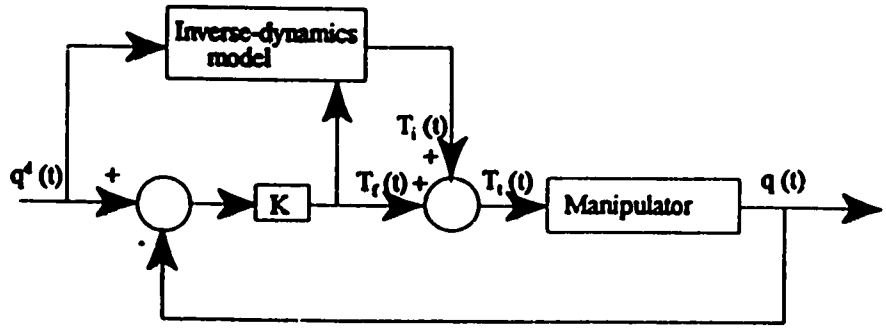

Figure 2.7 Feedback Error Learning Scheme for Robot Control

is used as an error signal for training the inverse model. It is expected that the feedback signal will approach to zero as learning proceeds. This learning scheme is called feedback error learning, emphasizing the importance of using the feedback torque as the error signal

of the hetero-synaptic learning.

Miller *et al.* (1987) reported on using a CMAC neural network in a computed torque controller. The robot dynamic model required by the computed torque controller is provided by the neural network. The CMAC network is used to predict the feedforward torques required for the desired trajectory, which is used in parallel with a fixed gain linear feedback controller. The scheme was implemented for the control of a GE P-5 five axis robot using a VAX-11/730 minicomputer with a TMS32010 auxiliary processor.

Karakaşoğlu *et al.* (1993) proposed a neural network structure which includes dynamical nodes in the hidden layer of network for trajectory tracking of a manipulator. A supervised learning scheme that employs a simple distributed updating rule is used for the on-line identification and decentralized adaptive control. Although they used the specific example of robot trajectory tracking for the illustration of the proposed scheme, they point out that the controller design procedure is general and can be used for the adaptive control of other complex dynamical systems.

2.4.4 Modelling Uncertainty

When there is uncertainty in a robotic system, a neural network can be applied to model it and used to compensate for such uncertainty. Leahy, Johnson and Rogers (1991) modified the inverse dynamics control scheme for robots under unknown payload conditions by using neural networks to estimate the dynamic parameters of an unknown load and to compensate for the uncertainty. The estimate of the payload adapts the

feedforward compensator to unmodelled system dynamics and payload variations.

Kuperstein *et al.* (1990) described a neural controller that learns to accurately move and position a single joint link carrying a payload. The neural controller can move a link carrying an unforeseen payload from any starting angle to any ending joint angle without oscillations of an end point. The learning does not require practice with different payloads.

Yegerlehner (1993) developed a neurocontroller for a two-link manipulator subject to changes in the payload. The control architecture is based on the computed torque method, but uses a feedforward neural network in place of the system model to generate the inverse dynamics. A second neural network is used to estimate the payload mass, which is used as an additional input to the inverse dynamics neural network.

2.4.5 Hybrid Intelligent Robotic Control

Handelman *et al.* (1989) tried to integrate knowledge-based systems and neural networks for autonomous robots. They discussed the possibility of robot intelligence and some schemes to realize it.

Rabelo and Avula (1991) proposed a hierarchial neurocontroller architecture consisting of two neural network systems for the manipulation of a robotic arm. The higher level neural system participated in the coordinates transformation and motion decision making. The lower one provided the control action sequence.

Fukuda and Shibata (1991) gave a concept and strategy of hierarchical intelligent control of robots. Basically, they regarded the hierarchical intelligent control system as a

hybrid system of neural networks, fuzzy logic, and AI, and divided the system into three levels: learning level, skill level and adaptation level. The adaptation level is realized by a neuromorphic controller in an uncertain environment. The skill level consists of trained fuzzy neural networks which emulate the human operation skills. The fuzzy logic works as the intermediate connecting neural networks and symbolic reasoning systems. The learning level recognizes the environment and the manipulated objects and makes decisions based on the sensed information.

2.4.6 Teacher-replacing Neurocontroller

Asada and Liu (1991) proposed a teacher-replacing scheme to transfer the skill of human operators into a neural network and use the generalization ability of the network to build the skill-based controller. This scheme can find wide applications in industrial manufacturing.

This chapter discusses various approaches of neural networks to the robotic control. These studies establish the efficacy of using a neural network-based controller for controlling the complex dynamics of manipulators. These controllers are based on learning from input output measurements and not on parametric model-based dynamics. The results permit reaching goals with arbitrarily low error even though the inverse dynamics, which was learned by the neural network is only an approximation. A neural network is used for the estimation of the inverse dynamics without any need of parametric model-based dynamics. In fact, there is no need to estimate the parameters of the system as in the case

of direct and indirect adaptive controls. The network takes into account the whole dynamics in addition to possible perturbations. In robotics, e.g., it may be possible to implement a complete inverse dynamics model of the robot which could possibly incorporate dynamics of the control device, backlash, and gear friction. This model would be computed without the need for analytic modeling.

Studies show that the neurocontroller maintains a very good trajectory tracking performance even in the presence of large parameter uncertainties and external disturbance. Neurocontroller achieves the desired control of the manipulators. The controller can learn to generate accurate movements of robot links without information about link mass, link length, and with indirect, uncalibrated information about payload and actuator limits.

The design concept for the neurocontroller is generic because it does not require knowledge of the physical plant or actuator characteristics and is designed to be extended to an arbitrarily large number of joints. Each new joint in the robot system will require adding new neural input and weight maps to the architecture.

2.4.7 Needs for Research

Autonomous robots, which achieve tasks without human operators, are required in many field. The autonomous robots carry out tasks in various environments by themselves like human beings. They have to be intelligent to determine their own motions in unknown environments based on sensory information. Neural networks show great potential for

controlling such systems.

Conventional control schemes require the knowledge of the mathematical model of the manipulator. In practice, this information is seldom available. When robots' task is changed, control algorithm adjustment and related tedious planning are required to make the robot execute the new task. Control engineers have attempted to solve this problem by developing techniques for system identification, but this usually requires a great deal of computation. On the other hand, a human controller can learn how to control a system without knowing its mathematical model. Therefore, it is evident that we need an intelligent controller which can learn how to control a dynamic system. Instead of basing the control algorithm on a known model, the neural network can configure itself with appropriate connection weights in order to learn a control scheme that generates desired system performance. The theories of neural network-based modelling and control are still under development. The main problems are the slow learning convergence of neural networks. Most existing neurocontrol schemes are difficult to implement in real-time due to their complexity.

CHAPTER 3

A MODIFICATION OF BACKPROPAGATION LEARNING ALGORITHM

3.1 INTRODUCTION

The ability of neural networks to realize some complex nonlinear functions makes them attractive for system identification. In the recent past, neural networks trained with the back-propagation learning algorithm have gained attention for the identification and control of nonlinear dynamic systems. However, the conventional back-propagation algorithm suffers from a slow rate of convergence. In this chapter, we present an improvement to the back-propagation algorithm based on the use of an independent, adaptive learning rate parameter for each weight with adaptable nonlinear function. The usefulness of including the adaptive slope of the nonlinearity of the nodes in the delta-bar-delta rule (Jacobs 1988) is studied. In the present scheme, the hyperbolic tangent activation function with changeable shape has been used. As a result, the sigmoid function attains flexibility, in contrast to all of the former studies in which neural networks have a fixed sigmoid function. Our modified updating rule, a variation on that originally proposed by

Jacobs (1988), allows adaptable independent learning rates for individual weights in the algorithm together with the updates of the slopes of the nonlinearities. After discussing the learning algorithm, a runtime comparison of the algorithms is made. This includes an estimation of the 'optimal' parameters of standard back-propagation and the comparison of this optimal standard algorithm with other back-propagation variations. It is shown that the learning speed is increased significantly by making the slope of the nonlinearity adaptive since it amplifies those directions in weight space that are successfully chosen by gradient descent. Comparison of algorithms showing the error convergence with the number of epochs is presented. The results demonstrate that the suggested method gives better error minimization and faster convergence. The chapter is organized as follows: Section 3.2 presents a brief introduction of backpropagation learning algorithm. Section 3.3 deals with the description of the proposed algorithm. Section 3.4 presents results of simulation and finally results are discussed.

3.2 A BASIC BACKPROPAGATION LEARNING ALGORITHM

A neural network is a computational structure inspired by knowledge from neuroscience. It represents an alternative computational paradigm to the programmed instruction sequence paradigm of Von Neumann. Although there are various types of neural networks available, feedforward neural networks are most commonly used for identification and control. Either a feedforward neural network or a multilayer neural network consisting of layers of neurons with weighted links connecting the outputs of

neurons in one layer to the inputs of neurons in the next layer is very popular and is used in identification and control. A typical multilayer neural network is shown in Fig. 3.1. It can be considered as static mapping from an input vector space $X \subset R^n$ to an output vector space $Y \subset R^m$. The input layer of a neural network will be denoted as the 1st layer and each subsequent layer will be referred to as the $(l+1)$th layer. A given layer, l, consists of a set of weights' $w_{ji}^l(k)$ $\{j = 1 \text{ to } n^l, i = l \text{ to } n^{l-1}\}$ where n^l is the number of neurons in the lth layer and n^{l-1} is the number of neurons in $(l-1)$th layer. Each, $w_{ji}^l(k)$, is an adjustable parameter that propagates and scales the output of the ith neuron in the previous layer to the jth neuron in the lth layer. The summation of these weighted inputs for a

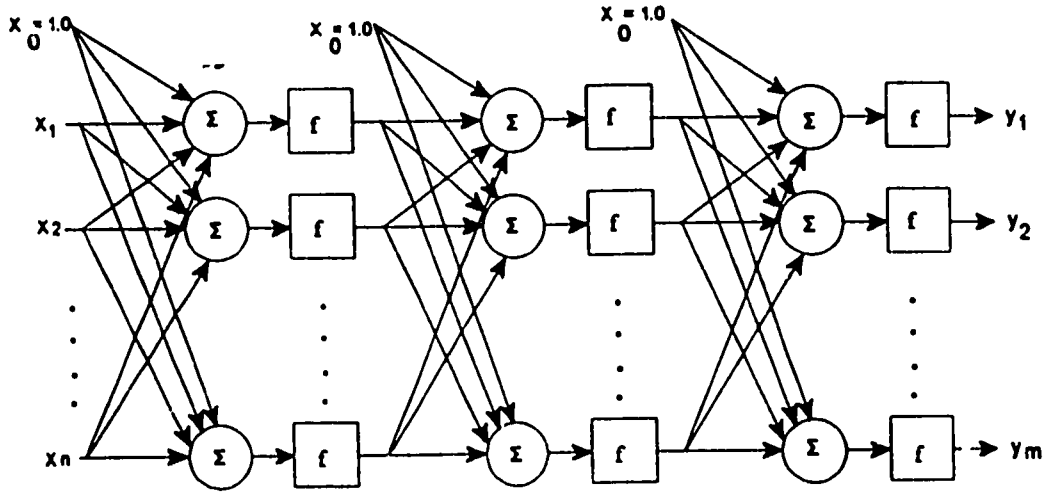

Figure 3.1 A Feedforward Neural Network

particular neuron is the activation level $v_j^l(k)$ that is transformed through a nonlinear activation function f: $R \to R$ to generate the output:

$$y_j^l(k) = f[\sum_{i=1}^{N_{l-1}} w_{ji}^l(k) y_i^{(l-1)}] = f[v_j^l(k)] \quad (1)$$

where f is the sigmoidal nonlinearity.

The input signal propagates through the network in the forward direction, on a layer by - layer basis. The network is trained in a supervised manner with a back-propagation algorithm. This algorithm is based on an error-correction learning rule. The error signal is propagated backward through the network. Weights of the network are adjusted until the desired result is obtained.

The details of the derivation of the back propagation algorithm are well known in the literature and have not been included here. The reader is referred to (Haykin 1994, Rumelhart *et al.* 1986) for details. A review of the important steps of the algorithm is presented here. The function to be minimized is the sum of the squared error of the output vector

$$\mathscr{E}_{av} = \frac{1}{N}\sum_{n=1}^{N} \mathscr{E}(n) \quad (2)$$

where N is the number of the data points and $\mathscr{E}(n)$ is the sum of squared errors at all nodes in the output layer, i.e.,

$$\mathscr{E}(n) = \frac{1}{2}\sum_{j=1}^{N_l} (d_j(n)-y_j(n))^2 \quad (3)$$

For an optimum weight configuration, $\mathscr{E}(n)$ is minimized with respect to the synaptic weight w, so that for each data set,

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}^{l}} = 0 \tag{4}$$

The weights of the network are updated using the relationship

$$w_{ji}^{l}(n+1) = w_{ji}^{l}(n) - \eta \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} \tag{5}$$

where η is a constant that determines the rate of learning; it is called the learning rate parameter of a back-propagation algorithm. Using back-propagation, partial derivatives can be found and update equation (5) can be written as

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k) + \eta \delta_{j}^{l}(k) y_{i}^{(l-1)}(k) \tag{6}$$

where δ_j is the local gradient of node j. For the output layer

$$\delta_{j}^{l}(n) = e_{j}^{l}(n) f'(v_j(n)) \tag{7}$$

while for other layers

$$\delta_{j}^{l}(n) = f_{j}'(v_j(n)) \sum_{k} \delta_{k}^{(l+1)} w_{kj}^{(l+1)}(n) \tag{8}$$

Hence, the partial derivatives of equation (5) are implicitly calculated in eqs. (7) and (8) as part of a recursive error propagation algorithm.

These procedures have some well known drawbacks. The first, and the simplest, is just the difficulty in the choice of the value of η. The learning parameter should be chosen small to provide minimization of the total error function $\mathscr{E}$. However, for a small η the learning process becomes slow. On the other hand, a large value of η corresponds to rapid learning, but leads to oscillations that prevent the algorithm from converging to

the desired solution. The choice of the best value is especially hard when the problem being addressed is very large, involving perhaps a few hours of simulation for a single sweep through the whole training set.

Another, more important drawback, is the slowness when the hypersurface that represents the cost function being minimized, presents deep and narrow valleys (ravines). One of the earliest methods to be proposed for overcoming this limitation, was the use of the momentum term, which corresponds in fact to recursive lowpass filtering of the gradient of the cost function. The resulting update formula is

$$w_{ji}^{l}(k+1) = w_{ji}^{l}(k)+\eta\delta_{j}^{l}(k)y_{i}^{(l-1)}(k)+\alpha\Delta w_{ji}^{l}(k-1) \qquad (9)$$

where α is the momentum constant that determines the relative contribution of the current and past partial derivatives to the current weight changes.

The third term in equation (9) is the so-called momentum term which may improve the convergence rate and the steady state performance of the algorithm. This term determines the relative contribution of the past partial derivatives to the current weight updates. It has the effect of attenuating "high frequency" oscillation across the ravine, and/or amplifying the "DC" component of the gradient along the bottom of the ravine.

3.3 PROPOSED MODIFICATIONS TO THE LEARNING ALGORITHM

In this section, an approach which led to an improved rate of convergence of the back-propagation algorithm is discussed. The appealing properties of the back-propagation algorithm are its simplicity and its ability to generalize to novel inputs. The slowness of

this algorithm is elucidated by examining the update law in eq.(6) which modifies the previous weight estimate on the basis of the current observation alone, ignoring all possibly relevant historical information. Various combinatorial learning methods exist that modify equation (5) in different ways (Baldi 1995, Battiti 1992, Eaton and Olivier 1992, Jacobs 1988, Kruschke and Movellen 1991). In this section we discuss the effect of the shape of nonlinearity of the node on the speed of convergence of the learning algorithm.

The general form of the hyperbolic tangent function is expressed as

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}} \qquad (10)$$

This function is monotonically increasing with asymptotes to ± 1, as the $x \to \pm\infty$, and is useful when bipolar outputs are required. We use the sigmoid function as hyperbolic tangent function type with an additional parameter which is given by

$$f(x,\gamma) = \frac{1-e^{-\gamma x}}{1+e^{-\gamma x}} \qquad (11)$$

This function can be made to have different shapes by changing the parameter γ, known as the gain of the nonlinearity of the node. The effect of different values of γ on the shape of the function is shown in Fig. 3.2. Another property of this function is that it becomes almost linear as $\gamma \to 0$, and is nonlinear for large values of γ. The saturation region of the sigmoid function can be controlled by the parameter γ. This means that the nonlinearity of the sigmoid function can be controlled by this parameter.

Because the unsaturated range of the nonlinearity (dynamic range) is predetermined

and kept fixed throughout (generally γ is made equal to 1); in the classical version of the back-propagation algorithm it is very common that the initial set of weights causes the nonlinearities to saturate and therefore the derivative of the nonlinearities becomes so small that the weights update very slowly, if at all. This causes the algorithm to fail to converge, or to converge after a relatively large number of iterations. On the other hand, if the slope is made very small so as to have all the inputs to nonlinearity fall in the dynamic range,

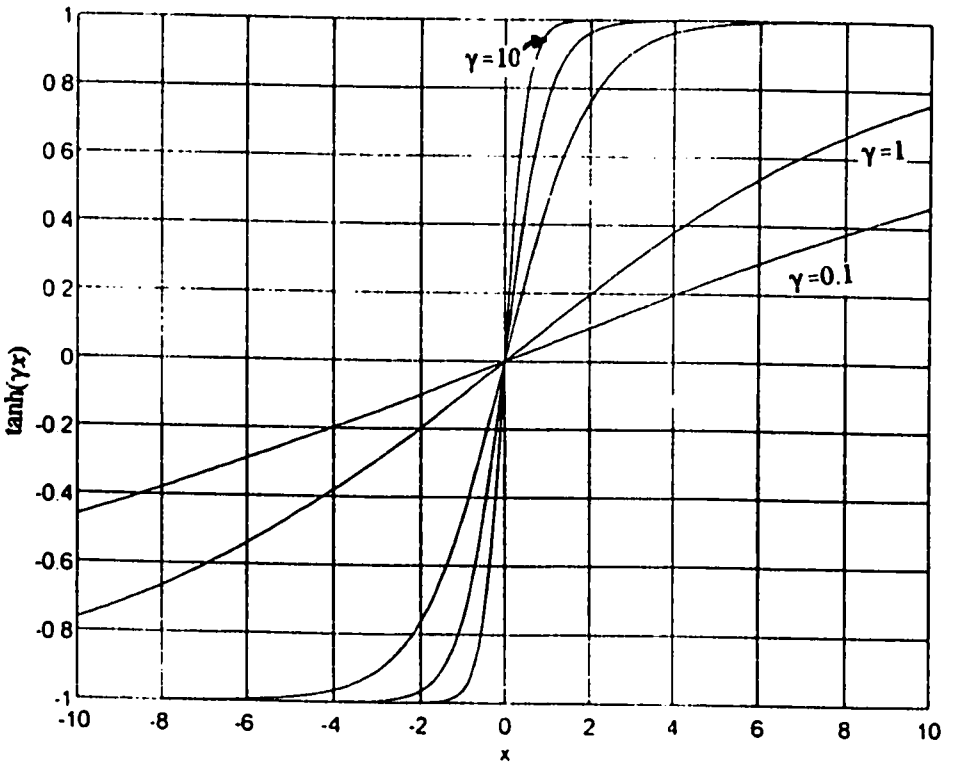

Figure 3.2 Sigmoidal Function

the inputs might be so small that it would take a relatively large number of iterations for

the actual outputs to move out from the dynamic range and fall in the correct saturated regions. Therefore, there must be an optimal value for the slope of the nonlinearity for different initial weights. Since there is no way to determine the optimal slope, it would be desirable to modify the back-propagation algorithm such that the slopes of the nonlinearities used in the net are made adaptive as well. The delta-bar-delta learning rule, proposed by Jacobs (1988), based on four heuristics achieved faster rate of convergence of algorithm while adhering to the locality constraint. These heuristics suggest that every synaptic weight of a network should be given its own learning rate and that these rates should be allowed to vary over time. Additionally, the heuristics suggest how the learning rates should be adjusted. The delta-bar-delta learning rule uses the fixed slope/gain of the nonlinearity of the neuron. We introduce adaptive gain into the delta-bar-delta rule and show that it can yield additional benefits for the learning speed and generalization. The learning rule for gains is easily incorporated into the delta-bar-delta learning rule as all the quantities appearing in the formula for update of gain are locally available to the affected gains. The modified algorithm consists of a weight update rule, a learning rate update rule and a gain update rule.

Learning the synaptic weights:

Following the same approach as in the standard back-propagation the weight update rule can be written

$$w_{ji}^{l}(n+1) = w_{ji}^{l}(n) + \eta_{ji}(n+1)\delta_{j}^{l}(n)y_{i}^{(l-1)}(n) + \alpha\Delta w_{ji}^{l}(n-1) \qquad (12)$$

In this case the only difference is in the formula for δ_j which is given by

$$\delta_j^l(n) = \gamma e_j^l(n) f'(v_j^l(n)) \tag{13}$$

The formula (13) for δ_j for output nodes is the same as in the standard back-propagation (7) except for the appearance of parameter γ. Similarly, the formula for δ_j for hidden nodes also gets modified by parameter γ.

The Learning Rate update:

Following Jacob's heuristics, the learning rate update rule is:

$$\Delta\eta_{ji}(n+1) = \begin{cases} \kappa & \text{if } S_{ji}(n-1)D_{ji}(n)>0 \\ -\beta\eta_{ji}(n) & \text{if } S_{ji}(n-1)D_{ji}(n)<0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where $D_{ji}(n)$ and $S_{ji}(n)$ are defined as, respectively

$$D_{ji}(n) = \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} \tag{15}$$

and

$$S_{ji}(n) = (1-\zeta)D_{ji}(n) + \zeta S_{ji}(n-1) \tag{16}$$

where ζ is a positive constant. The quantity $D_{ji}(n)$ is the current value of the partial derivative of the error surface with respect to the $w_{ji}(n)$. The quantity $S_{ji}(n)$ is an exponentially weighted sum of the current and past derivatives of the error surface with respect to $w_{ji}(n)$, and ζ as the base and iteration number n as the exponent.

The learning rate parameters are updated based on both the partial derivatives of

the error with respect to the parameters and on an estimate of the curvatures of the error surface at the current point in parameter space along each parameter dimension unlike the traditional delta rule that performs steepest descent on the local error surface, the error gradient vector $\delta_j(n)$ and the weight update vector Δw_{ji} have different directions. This learning rule assures that the learning rate η_{ji} will be incremented by constant number κ if the error derivatives of the consecutive epochs have the same sign, which generally means a smooth local error surface. On the other hand, if the error derivatives keep on changing signs, the algorithm decreases the learning rates. The learning rate parameter is incremented linearly but decremented exponentially. A linear increase prevents the learning rate parameter from growing too fast, whereas an exponential decrease means that the learning rate parameter remains positive and that it is reduced rapidly.

Learning the sigmoid function parameter: Now, we need to find parameters γ's in the sigmoid function to minimize $\mathscr{E}$. By employing the gradient descent algorithm, the increment of γ_j^l denoted by $\Delta\ \gamma_j^l$ can be obtained as

$$\Delta\gamma_j^l(n) = -\eta_\gamma \frac{\partial \mathscr{E}(n)}{\partial \gamma_j^l} \tag{17}$$

where $\eta_\gamma > 0$ is a learning rate given by a small positive number. Using the chain rule, it is easy to show that

$$\Delta\gamma_j^l = \frac{\eta_\gamma \delta_j^l v_j^l}{\gamma_j^l} \qquad \eta_\gamma > 0 \tag{18}$$

The learning rule for parameter the γ is easily incorporated into the standard back-propagation program. In particular, all the quantities that appear in (18) are easily locally available.

Summary of the training procedure

The following steps summarize the procedure to be adopted while training the neural network.

1. Start with a reasonable network configuration, and set all the synaptic weights and threshold levels of the network to small random numbers that are uniformly distributed.
2. Present the network with input vectors and output vectors of the system.
3. For each example in the set ordered in some fashion, perform the forward and backward computations using the algorithm described above.
4. Iterate the computation by presenting new set of training examples to the network until the free parameters of the network stabilize their values and the average squared error $\mathscr{E}_{av}$ computed over the entire training set is at a minimum or acceptable small value.

3.4 RESULTS OF SIMULATION

To evaluate the performance of the algorithm, we report results of simulation of identification of a suitable model of a nonlinear system. In this type of identifier, the output of the neural network almost coincides with the output of the plant after learning and the model of the plant is composed in the neural network. The following algorithms were simulated: A gradient descent algorithm (standard back-propagation), delta-bar-delta

procedure and delta-bar-delta with adaptive gain of the neurons. In the example considered here, the plant is assumed to be of the form:

$$y_p(k+1) = f[y_p(k-1), y_p(k-2), u(k), u(k-1), u(k-2)]$$

In the identification model, a three-layer neural network with 5 input nodes, 20 nodes in the first hidden layer, 10 nodes in second hidden layer and 1 node in the output layer, was used. The hidden neurons are nonlinear whereas the output neuron acts as a linear combiner. Since the initial values of the weights may affect convergence, all the simulations have been performed with the same initial weights. The input to the model and plant consists of uniformly distributed random numbers in the interval [-1, 1].

A) Gradient Descent Algorithm (Standard back-propagation)

The gradient descent algorithm was applied with a constant step size, η, of 0.2 in each layer. Initially the weights were set uniformly distributed in the interval [-0.5, 0.5]. Convergence to a mean squared error over an epoch versus the number of epochs is shown in Fig. 3.4. From Figure 3.4 it is observed that the speed of convergence is fast initially but after few iterations it is very slow and it takes long to converge to the desired tolerance.

B) Delta-bar-Delta Rule

A delta-bar-delta rule back-propagation algorithm was employed with $\eta = 0.2$, $\alpha = 0.9$, $\kappa = 0.01$, $\beta = 0.1$ and $\zeta = 0.7$. Initially the weights were uniformly distributed

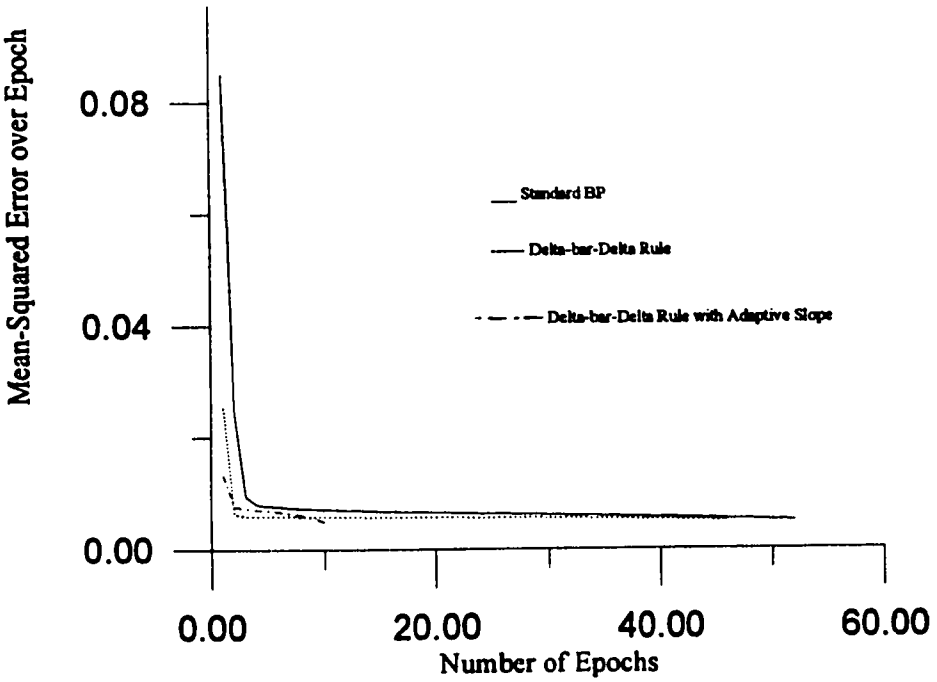

Figure 3.3 Learning Curves

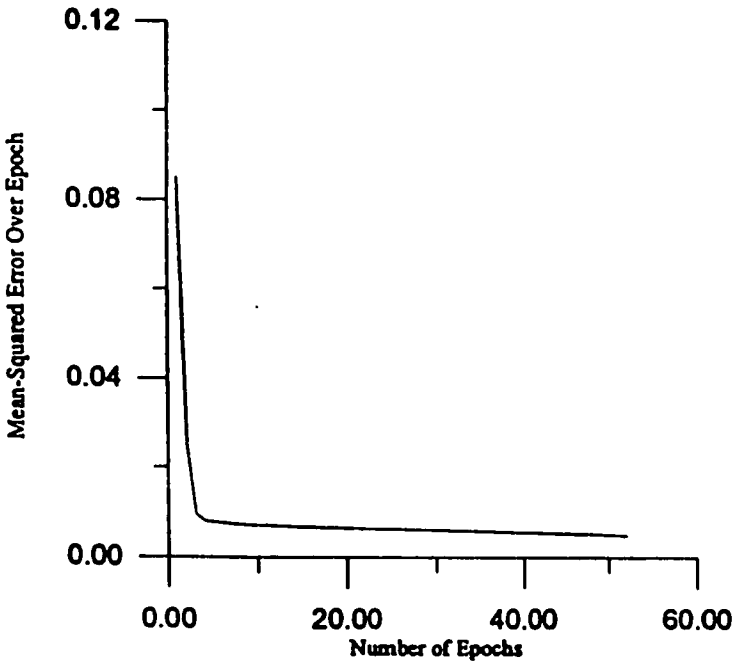

Figure 3.4 Learning Curve (Standard Back-propagation)

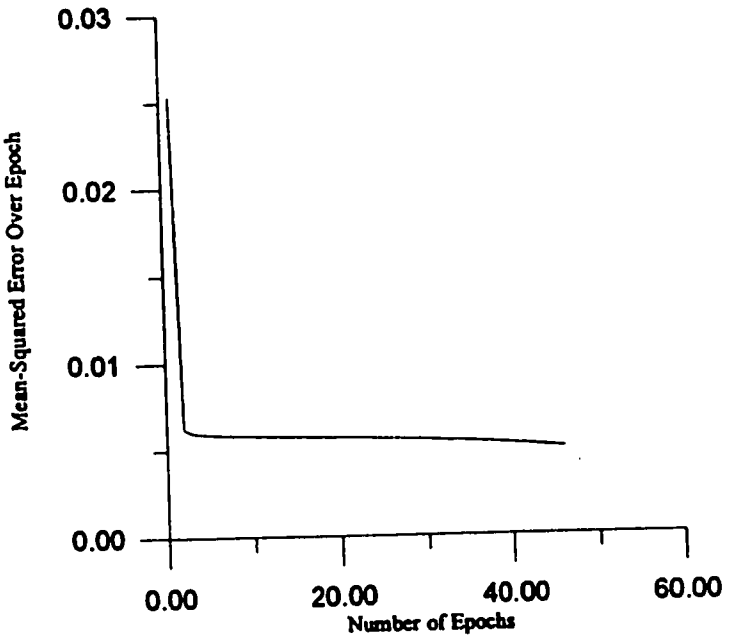

Figure 3.5 Learning Curve (Delta-bar-Delta Rule)

in the interval [-0.5 0.5]. Although the initial rate of convergence was rapid, as shown in Figure 3.5, the algorithm did not achieve a final error as low as the other algorithm or it took long to achieve the desired error tolerance.

C) Proposed Delta-bar-Delta Rule with Adaptive Gain

The delta-bar-delta rule with adaptive gain of the neurons was implemented with the same value of the parameters as the delta-bar delta rule. Initially the gain of neurons was set to 1.0. The results from delta-bar-delta rule with adaptive gain, shown in Fig. 3.6, demonstrate an improved rate of convergence over the existing algorithms. From the figure it is observed that the convergence is comparable to other algorithms initially but after few iterations it is very fast and converges to the prespecified tolerance in about 10

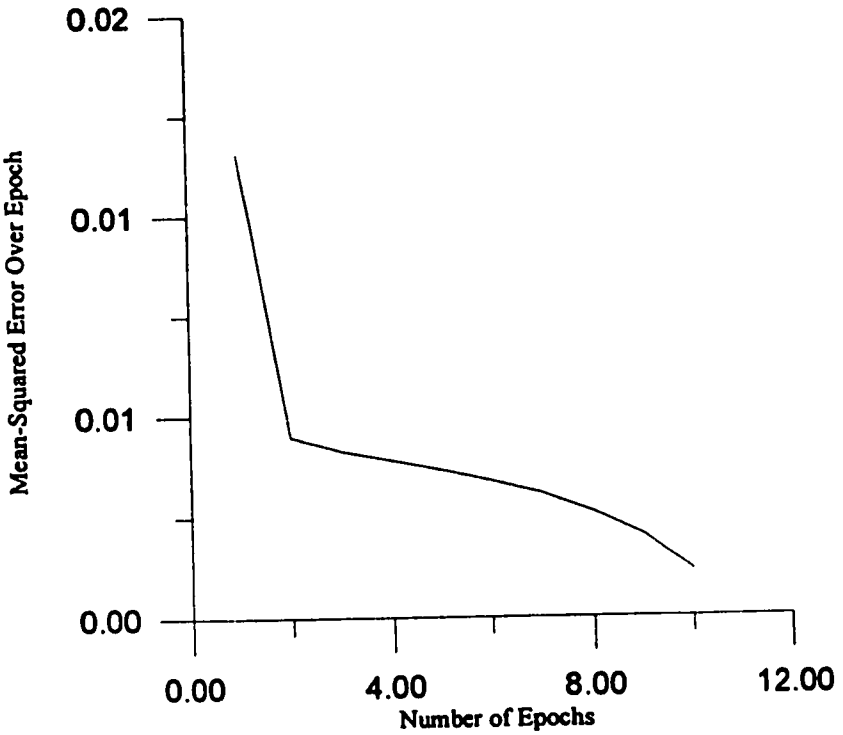

Figure 3.6 Learning Curves (Delta-bar-Delta rule with Adaptive Gain)

epochs. This improvement in the rate of convergence can be attributed to the adaptive gain of the neurons which has a catalytic effect in the learning process by modifying the magnitude of the weight change.

(D) Comparison

Figs. 3.3 to 3.6 show the learning curves with mean squared average error over an epoch $\mathscr{E}_{av}$ plotted as a function of the number of epochs using various approaches discussed above. Table 3.1 shows the learning time required by various algorithms to converge to the preset value of the error, i.e., tolerance. From the learning curves it may

be observed that the algorithm based on delta-bar-delta rule with adaptive gain of the node shows significant improvement in the learning speed as compared to the other algorithms. In the case of standard back-propagation, we see from Figure 3.4 that the learning is

TABLE 3.1

Type of Back-propagation	Number of epochs
Std. Back-propagation	52
Delta-bar-delta rule	46
Delta-bar-delta rule with adaptive gain	10

comparable and after a few iterations it is very slow. In sharp contrast, learning in case of the proposed delta-bar-delta rule with adaptive gain of the node shown in Figure 3.6 is quite fast. It takes about 50 epochs to converge to the prespecified tolerance with the standard back-propagation whereas with the proposed algorithm it takes about 10 epochs to converge. Simulations confirmed that adaptive gain significantly accelerated a back-propagation algorithm.

3.5 CONCLUSIONS

The chapter has presented a method for improving the learning ability of neural networks by using an activation function with changeable shape. The proposed updating laws, involving the adaptive selection of the step size η and adaptive gain γ, have been shown to learn much more efficiently the dynamics of a nonlinear system. Results of simulation suggest that the proposed delta-bar-delta rule with adaptive gain of the node

significantly improves the convergence behavior when compared with conventional back-propagation, making the proposed algorithm less computationally intensive. Because of the fast convergence of the learning algorithm, it is possible to use a neural network for modeling nonlinear dynamic systems in real-time.

CHAPTER 4

MODELING ROBOT DYNAMICS USING NEURAL NETWORKS

4.1 INTRODUCTION

During the past 40 years, great advances have been made in control theory and many of these have now been applied in developing controllers for industrial robots. All of these approaches, however, require the knowledge of a suitable mathematical model of the process to be controlled. In most practical cases, the system is also subject to random noise in the form of errors in sensor measurements, disturbances and parameters variations. Some types of stochastic models are needed in these cases for developing a suitable controller.

In view of the above, in this chapter we will discuss various methods that have been developed for modelling robot dynamics, and in particular how neural networks can be employed to perform this task efficiently.

The problem of system modelling and identification has attracted considerable attention during the past four decades mostly because of a large number of applications in diverse fields like chemical processes, biomedical systems, ecology, econometrics, and social sciences. In each of these cases, a model consists basically of mathematical equations which can be used for understanding the behaviour of the system, and wherever possible, for

prediction and control. Most processes encountered in the real world are nonlinear to some extent, and in many practical applications nonlinear models may be required to achieve an acceptable predictive accuracy. The choice of model is vitally important since it influences its usefulness in prediction and control. Practical applications have shown that nonlinear models cannot only provide a better fits to the data but can also reveal rich behaviour such as limit cycles and bifurcations, which cannot be captured by linear models.

Two basic types of modelling problems arise. In the first type one can associate with each physical phenomenon, a small number measurable causes (inputs) and a small number of measurable effects (outputs). The outputs and the inputs can generally be related through a set of mathematical equation, in most cases nonlinear partial differential equations. The determination of these equations is the problem of modelling in such cases. These can be obtained by using either a set of equilibrium equations based on mass and energy balance and other physical laws, or one may use the "black box" approach which consists of determining the equations from the past records of the inputs and the outputs. Modelling problems of this type appear quite often in engineering practice. This type of problem is referred as system identification.

When formulating and solving an identification problem, it is important to have the purpose of the identification in mind. The interest in this subject has different roots, e.g.,

(i) Definite needs by engineers in process industries to obtain a better knowledge about their plants for improved control (This holds not only for the chemical but also for the mechanical and other production industries).

(ii) The task of studying high performance aero and space vehicles, as well as the dynamics of more down-to-earth objects like railway carriages and hydrofoils.

(iii) Study of a human being in tracking action and in other types of control.

In control problems the final goal is often to design control strategies for a particular system. There are, however, also situations where the primary interest is to analyse the properties of a system. Even if the purpose of the identification is to design a control system, the character of the problem might vary widely depending on the nature of the control problem. A few examples are given below:

Design a stable regulator

Design a control program for optimal transition from one state to another.

Design a regulator which minimizes the variations in process variables due to disturbances.

In the first case it might be sufficient to have a fairly crude model of the system dynamics. The second control problem might require a fairly accurate model of the system dynamics. In the third problem it is also necessary to have a model of the environment of the system. Assuming that the ultimate aim of the identification is to design a control strategy for a system, what would constitute a satisfactory solution from a practical point of view?

The identification problem is defined by Zadeh (1962) as "the determination, on the basis of input and output, of a system within a specified class of systems, to which the system under a test is equivalent." Using Zadeh's formulation it is necessary to specify a class of systems, a class of input signals, and the meaning of equivalence. The problem can also be

represented diagrammatically as Figure 4.1.

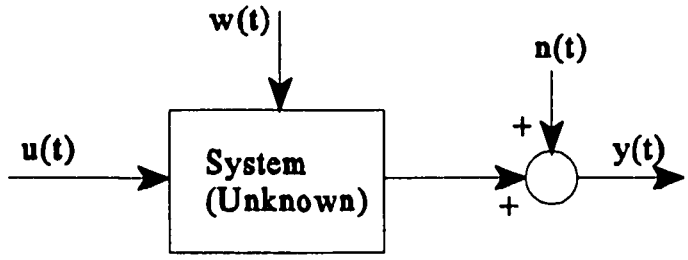

Figure 4.1 Formulation of system identification

where

u(t) is the input

w(t) is the input observation noise

n(t) is the observation noise

y(t) is the output.

Thus, roughly speaking, the problem of system identification is the determination of the system model from records of input u(t) and the output y(t).

A system identification problem consists of three steps: structure determination, parameter estimation and model validation (Box and Jenkins 1970; Sinha and Kuszta 1983). In the first step one tries to find the order of the model (discrete or continuous) by which the system can be closely represented. The second step is the application of a suitable algorithm to estimate the parameters of the model as accurately as possible. The last step is the

application of some criterion (e.g., the Akaike information criterion) to check how closely the fitted model represents the system under consideration.

A number of methods have been proposed in the literature during the past few decades. These methods can be classified in several different ways, for instance, continuous time type or discrete time type; on-line methods or off line methods; parametric methods or non-parametric methods etc. (Sinha and Kuszta 1983). In the past three decades major advances have been made in adaptive identification and control for identifying and controlling linear time-invariant plants with unknown parameters. The choice of the identifier and controller structures is based on well established results in linear system theory. Stable adaptive laws for the adjustment of parameters in these cases which assure the global stability of the relevant overall systems are also based on properties of linear systems as well as stability results that are well known for such systems (Narendra and Annaswamy 1989). The theory of identification and control of linear systems is well established but very few results exist in nonlinear systems theory which can be directly applied. Considerable care has to be exercised in the statement of the problems, the choice of the identifier and controller structures as well as the generation of adaptive laws for the adjustments of the parameters. In short some of the problems in system identification using conventional methods are:

(i) determining the order of the system

(ii) selection of a suitable criterion for determining the "accuracy" of the model

(iii) designing an input signal which will maximize the accuracy of the estimates of the parameters of the model.

The capability of trained neural networks for approximating arbitrary input-output mappings can find an important application in devising simple procedures for the identification of unknown dynamical plants in order to control them. This chapter is concerned with modelling the complex nonlinear dynamics of a robotic system using neural networks. A neural network-based scheme has been used for the identification of inverse dynamics of a manipulator. A learning scheme using a model of known dynamics of manipulators is used. A modified form of the back propagation learning algorithm introduced in the previous chapter has been used. To demonstrate the validity of the proposed method, we apply it to identify the inverse dynamics of a two-link manipulator and a three-link manipulator. The simulation results confirm that this modelling can be satisfactorily achieved. The chapter is organized as follows: Section 4.2 deals with the identification using neural networks. In Section 4.3 modelling of inverse dynamics of manipulator using a neural network is discussed. Simulation results are presented in Section 4.4 and finally results are discussed.

4.2 IDENTIFICATION WITH NEURAL NETWORKS

One of the main objectives of modelling a process is to learn something about it, perhaps by identifying the key parameters that influence it. Another is for control, either as an on-line monitor or for use in a feedback controller. In computerized control systems the control algorithm often assumes the form of a set of control rules specifying the relationship between measured parameters of the system or the environment vs. control action taken. The rules are derived as a result of the analysis of the model. In many applications however, the model is either not available or too complex to be useful, e.g., biological processes, robotics,

etc. are difficult to model accurately to obtain satisfactory control algorithms.

It has been observed that trained human operators, after some practice, achieves very good results in controlling complex systems. They know what control actions are to be taken under different conditions, but typically they are unable to formalize their knowledge in control logic that can be used to control the system automatically. One way to capture the control logic of the system operators are the accumulation and analysis of the operational data. The data, represented in a form of a table of measurement vectors sampled at specific intervals, reflect the states of the system, environment and control action taken. The objective of the analysis of data is to identify and describe the real relationship between the measured variables and the actions taken and to derive the control logic as an optimized expression. Models can be developed by utilizing either first principles such as material and energy balances, or process input and output information. The advantages of first principle models include the ability to incorporate the scientist's view of the process into the model, the capacity to describe the internal dynamics of the process, and the capability to explain the behaviour of the process. Their disadvantages are the high cost of model development, the bias that may have because of the model developer's choices, and the limitations in including the details due to lack of information about specific model parameters. As details are added to the model, it may become complex and too large to run the model on the computer within an acceptable amount of time. However, this constraint has a moving upper limit, since new developments in computer hardware and software technologies permit faster execution times. Often, some physical, chemical or transport parameters are computed by using empirical

relations, or they are derived from experimental data. In either case, there is some uncertainty about the actual value of the parameter.

An input-output model is a means of describing the input-output relationship of a system and an important question regarding the model is how to relate the input to the output in some straightforward way that will provide an adequate approximation to a large class of systems for a reasonable computational cost. For linear systems, it is well known that a linear difference equation model exists that involves only a fixed and finite number of calculations at each stage. Once the model is decided and verified to be correct, it can be used to aid in understanding the nature of some unknown disturbances, forecast future values of the time series, derive unknown transfer functions, or to design optimal control strategies. On the other hand most systems encountered in the real world are nonlinear to some extent and in many practical applications nonlinear models may be required to achieve an acceptable prediction accuracy.

The nonlinear autoregressive moving average (NARMAX) model (Leontaritis and Billings 1985, Chen and Billings 1989a) provides a unified representation for a wide class of discrete-time nonlinear systems. In an NARMAX description the system is modelled in terms of a nonlinear functional expansion of lagged inputs, outputs and prediction errors. Two considerations are of practical importance for the application of the NARMAX approach. The function describing a real-world system can be very complex and the explicit form of this function is usually unknown, so that any practical modelling of a real-world process must be based upon a chosen model set of known functions. Obviously this model set should be

capable of approximating the underlying process within an acceptable accuracy. Secondly, an efficient identification procedure must be developed for the selection of a parsimonious model structure, because the dimension of a nonlinear model can easily become extremely large. Without efficient subset selection, the identification would entail excessive computation and the resulting model would have little practical value. Previous research (Chen *et al.* 1989b) has investigated the polynomial NARMAX model and several identification procedures based upon this model have been developed (Chen and Billings 1989a). Because the derivation of the NARMAX model was independent of the form of the nonlinear functional, other choices of expansion can easily be investigated within this framework and neural networks offer an exciting alternative.

In recent years, neural networks, which are not model based, have become recognized as viable alternatives to traditional identification and control in many applications. This is particularly true for complex, large order, or highly nonlinear systems which pose significant challenges to mathematical modelling and model-based control. The nonlinear functional mapping properties of neural networks are central to their use in control and identification. Training a neural network using input-output data from a nonlinear plant can be considered as a nonlinear functional approximation problem.

As discussed in the previous section system identification usually consists of two stages - model selection, and parameter estimation. In case of neural network-based identification, the selection of the number of hidden nodes corresponds to the model selection stage. The backpropagation algorithm utilizes gradient descent to determine the

weights of the network and thus corresponds to the parameter estimation stage. Thus we see that in case of the identification using neural networks we do not need to know the model structure especially important with nonlinear systems like robotic manipulators. Neural networks are trained to approximate relations between variables regardless of their analytical dependency, they are usually referred to as model-free estimators. The main properties of the neural networks are:

(1) *Ability* to learn from experience.

(2) *Generalization* for untrained inputs.

(3) Capability to *approximate* to arbitrary specified accuracy given sufficient number of neurons.

Neural networks have become of interest because of these properties and have been used in several applications for performing various mappings. The properties outlined above are the same as those of interest to researchers in the area of system identification.

Neural networks are trained to minimize the error energy function. Neural networks, trained to minimize squared error energy function between neural network output and a plant output. In this type of identifier, the neural network output converges with the plant output after learning and the direct/forward transfer function is composed in the neural network. The other type of neural network identifier is trained to minimize the squared error between the neural network output and the plant input. In this type of identifier, the neural network output converges with the plant input after learning and the inverse

transfer function of the plant is composed in the neural network. If we can get the inverse model of the system by neural network, the neural network can be used as a feedforward controller. In almost cases, to get the input which realizes a desired output of the system is impossible, so that we cannot get learning patterns of the neural network for inverse modelling of the system. We can get only the error between the output of the system and the desired output which is used as the input value of the neural network. We must calculate the error between the output of the neural network and input of the system for the desired output, using the error between the output of the system and the desired output for learning in inverse modelling. The most important application of the identified inverse model result is in the direct controller. In robotics, e.g., it may be possible to implement a complete inverse dynamics model of the robot which could possibly incorporate dynamics of the control device, backlash, and gear friction. This model would be computed without the need for analytic modelling.

There are different neural networks used in the identification. Some of them are

- Feedforward multilayer neural networks
- Radial basis function networks
- Recurrent neural networks

Multilayer networks have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm know as the Back-propagation algorithm. The structure of a multilayer neural network has been described in (Haykin 1994 and Rumelhart *et al.* 1986). A multilayer neural network is

composed of hierarchy of processing units organized in a series of two or more mutually exclusive sets of neurons or layers. The first layer acts as a receiving site for the values applied to the network. At the last layer, the results of the computation are read off. Between these two layers lie one or more layers of hidden units. The function of the hidden neurons is to intervene between the external input and the network output. The number of layers in the network, and the number of neurons in each layer, are important parameters of the network. Once these have been selected, only the adjustable weights have to be determined to specify the network completely. These weights are generally adjusted to minimize the error between the output of the network and some desired output according to some criterion. Due to the presence of the nonlinear function in the network, it follows that the output of the network depends nonlinearly on the parameters. This, in turn, implies that the parameters have to be adjusted using some gradient-type method such as backpropagation. Another important feature of the multilayer networks is that the networks contain only the linear operations of multiplication by a scalar constant and summation in addition to the single nonlinear function which is known. This makes it mathematically attractive for the approximation of functions and consequently for the modelling of systems nonlinear differential equations.

A viable alternative to multilayer neural network is the radial basis function network (RBFN). The RBFN can be considered as a two-layer network in which the hidden layer performs a fixed nonlinear transformation with no adjustable parameters, so that the input space is mapped into a new space. The output layer then combines the outputs in the latter

space linearly. The details of the network can be found in (Haykin 1994 and Rumelhart *et al.* 1986). When the radial functions are specified, the only adjustable parameters of the network are the weights. Since these are linearly related to the output and the output error, they can be adjusted using a straightforward least square approach. This, in turn, has made radial basis networks attractive in signal processing, identification, and control.

Multilayer neural network has been considered as providing a nonlinear mapping between an input vector and a corresponding output vector. Most of the work in this area has been devoted to obtaining this nonlinear mapping in a static setting. Many practical problems may be modelled by static models, e.g., pattern recognition. On the other hand, many real-life problems such as time series prediction, vision, speech recognition, and motor control require dynamic modelling, i.e., the current output depends on previous inputs and outputs. There have been number of attempts to extend the multilayer networks to encompass this class of problems (Lapedes and Farber 1987, Narendra and Prathasarthy 1990). Waiebel *et al.* (1989) used a time delay neural network architecture that involves successive delayed inputs to each neuron. All these attempts use only feedforward architecture, i.e., no feedback from latter layers to previous layers. There are other approaches that involve feedback from either the hidden layer or the output layer to the input layer. These define the class of recurrent networks. Recurrent or feedback networks allow information to flow from the output to the input field, so that the previous state of the network can be fed back to the input. The current input, therefore, can be processed based upon past as well as future inputs. Recurrent networks share the following distinctive features: (i) nonlinear computing units; (ii) symmetric

synaptic connections; and (iii) abundant use of feedback. Details can be found in (Williams and Zipser 1990, Hopfield 1982, 1984).

4.3 MODELLING INVERSE DYNAMICS OF A MANIPULATOR

Model-based robot control schemes require the inclusion of the dynamical model of the robot arm in the control law. Hence, it is necessary for robot dynamics to be modelled accurately. However, the dynamics consists of a set of second-order, nonlinear, and highly coupled partial differential equations. The dynamics are time variant, and undergo changes such as variations in payloads, changes in the friction coefficients of the joints etc. These facts have always posed a major problem in the implementation of dynamic-based controllers. Many algorithms have been developed for the computation of the dynamics in the last decade, and this can be seen in the literature on the subject (Koivo 1989, Fu *et al.* 1987).

The reliance of most model-based control algorithms on the accurate derivation of the arm dynamics is barely practical when operating in real-time because of the time-varying nature of the robot arm. This is due to several reasons, for example, the nonuniform mass distribution of the different links, the high degree of coupling among links, and the effects of handling loads. These issues need to be addressed and solutions incorporated during the model development stage. Robot dynamic models are usually defined using two kinds of parameters. The first is kinematic parameters, which define homogenous transformations between successive links. The second kind is the dynamic or inertial parameters consisting of mass, centre of mass and the inertial parameters of each link. The measurement of the dynamic parameters involves excessive computations due to the large number of

multiplications and additions involved. The computation of the dynamics must be made fast, efficient, and viable in terms of cost to enable real-time implementations. Several techniques for the measurements of these properties can be found in the literature (Khosla 1988, Atkeson 1988). However, most of these techniques incur a heavy computational load which hinders real time applications. Application of neural networks to dynamical modelling has created a lot of interest in the past decade. What makes neural networks a viable tool is the fact that autonomous systems, such as robot manipulators, require a high degree of flexibility to deal with significant variations in the environment. These variations are, often, unpredictable and difficult to formulate with traditional mathematical tools.

4.3.1 Problem Definition

In the case of a robot system, the dynamics are complex and nonlinear. The general form of the dynamic equations of a robotic system can be written as:

$$\tau(t) = M(q(t))\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + G(q(t)) \qquad (1)$$

where $\tau(t)$ is a $n \times 1$ vector of joint torques

$q(t)$ is a $n \times 1$ vector containing the joint angles

$M(q(t))$ is the inertia matrix of the manipulator

$C(q(t), \dot{q}(t))$ is the contribution of the Coriolis and centrifugal forces and

$G(q(t))$ is the gravitational torque due to the gravity.

Concerning the system represented by equation (1), the following two problems are encountered:

(i) The *forward dynamics* problem: Given the torque vector $\tau(t)$, obtain the output vector $q(t)$

and $\dot{q}(t)$. This is equivalent to the problem of modelling the dynamical system of eq. (1).

(ii) The *inverse dynamic* problem: Given the vectors of the joint angles $q(t)$, joint velocities $\dot{q}(t)$, and accelerations $\ddot{q}(t)$, calculate the vector of torque inputs $\tau(t)$. An inverse problem is important to control problems since it allows one to find the appropriate inputs necessary for producing the desired outputs.

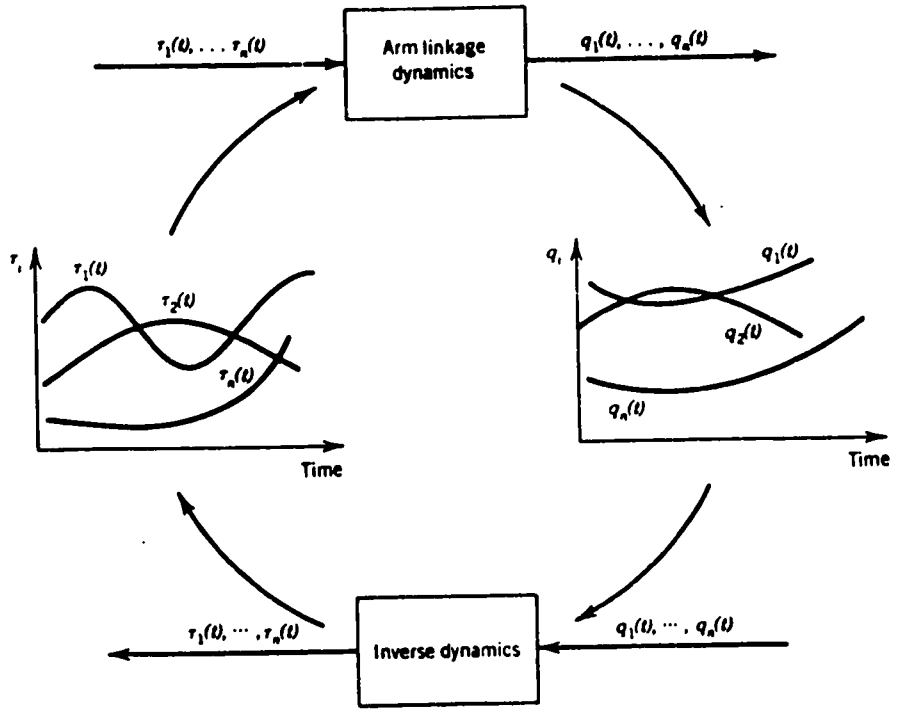

Figure 4.2 Inverse Dynamic

The recent resurgence of research and application of artificial neural networks to diverse range of disciplines makes it possible to seek out solutions of robotic problems by employing neural networks. Application of neural networks to the inverse dynamics problem has created a lot of interest recently (Kawato *et al.* 1987, Miyamoto *et al.* 1988). The basic

idea is that the neural network learns the inverse dynamical relationship of the robot directly so as to map the nonlinear relationship between the robot torques and joint variables. Once the inverse dynamic model of the manipulator is known, it can be used as a feedforward controller.

4.3.2 Model Learning

Psaltis *et al.* (1988) proposed a two-stage learning procedure. The first stage is called generalized learning with its configuration shown in Fig. 4.3, and the second stage is called specialized learning with a different configuration shown in Fig. 4.4. For generalized learning, the robotic manipulator should be actually operated, and operating data should be recorded. Then, the neural network receives the obtained trajectories and is trained to yield the desired torque command. This training can be fulfilled off line. After the neural network is well trained, the neural network is installed in the feedforward loop of the manipulator controller, and specialized learning is used to fine tune the neural network on line. This procedure is inefficient in generalized learning because of the following:

1) For recording the learning data, the robotic manipulator should actually be operated. This is time consuming.

2) It is difficult to obtain data that are uniformly distributed over the working space of the endpoint of the manipulator.

As far as robotic manipulators are concerned, we can obtain approximate dynamical models to a certain extent from either the operating data or the design specifications. The neural network learning consists of two steps. Step 1 consists of generalized learning of the

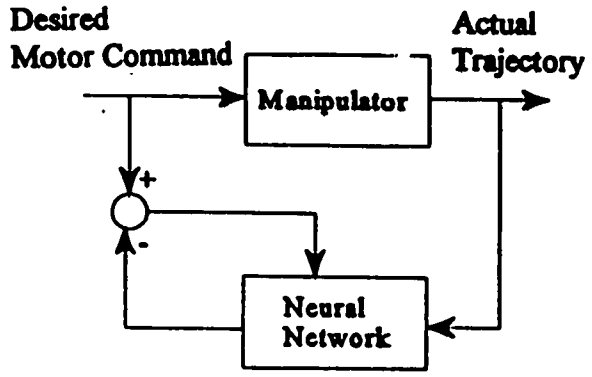

Figure 4.3 Generalized Learning

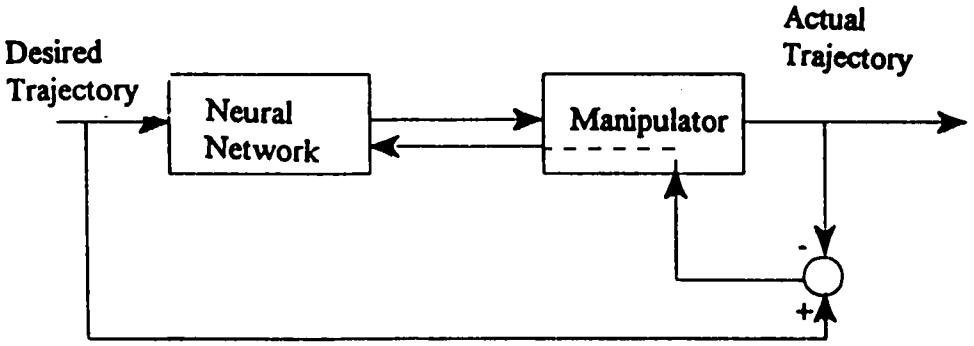

Figure 4.4 Specialized Learning

neural networks from the obtained dynamical model. The scheme is shown in Fig. 4.5. Since no actual operation of the manipulator is necessary in generalized learning, model learning is efficient. In step 2 the neural network can be trained to learn structured/unstructured uncertainties by actually operating the manipulator on line.

Once the neural network finishes learning, it produces an approximate inverse dynamic model, described as

$$\tau_{nn} = \hat{M}(q_d)\ddot{q}_d + \hat{C}(q_d, \dot{q}_d) + \hat{G}(q_d)$$

where $\hat{M}(q_d)$, $\hat{C}(q_d, \dot{q}_d)$ and $\hat{G}(q_d)$ are the estimated values of the various parameters of the equation (1).

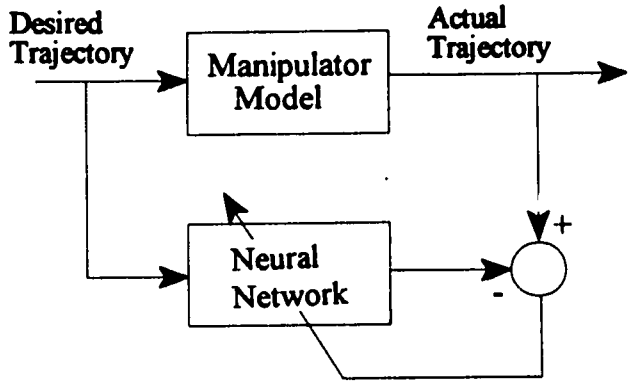

Figure 4.5 Model Learning

Thus, a neural network learns the following nonlinear function

$$\tau_{nn} = f(\ddot{q}_d, \dot{q}_d, q_d, w, \gamma)$$

where w and γ are the synaptic weights between nodes and sigmoid function parameter respectively.

4.4 Results of Simulation

This section presents the modeling of the nonlinear robot dynamics of eq. (1) by using a neural network. A three-link and a two-link manipulators have been simulated.

4.4.1 Two-Link manipulator

The simulated two-link manipulator is shown in Figure 4.5. For this manipulator the terms appearing in the dynamical equation (1) are given as

$$
\begin{aligned}
m_{11} &= m_1 l_{c1}^2 + m_2(l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(q_2)) + I_1 + I_2 \\
m_{12} &= m_{21} = m_2(l_{c2}^2 + l_1 l_{c2}\cos(q_2)) + I_2 \\
m_{22} &= m_2 l_{c2}^2 + I_2 \\
g_1 &= m_1 g l_{c1}\cos(q_1) + m_2 g l_1 \cos(q_1) + m_2 g l_{c2}\cos(q_1 + q_2) \\
g_2 &= m_2 g l_{c2}\cos(q_1 + q_2) \\
h &= -m_2 l_1 l_{c2}\sin(q_2)
\end{aligned}
$$

with

$$
C = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix}
$$

The details of the derivation of equations can be found in any good book on robotics (for example see Spong and Vidyasagar 1989).

Link 1 was moved from 15 degrees to 75 degrees and link 2 was moved from 20 degrees to 80 degrees in 3 seconds. Cubic trajectories are specified between the various goal points according to standard trajectory planning schemes (Craig 1986). The desired trajectories of are expressed as

$$
\begin{bmatrix} q_1^d(t) \\ q_2^d(t) \end{bmatrix} = \begin{bmatrix} 15.0 + 20.0t^2 - 4.44t^3 \\ 20.0 + 20.0t^2 - 4.44t^3 \end{bmatrix}
$$

A three-layered neural network has been used. The network consisted of 6 nodes in the input layer, 25 nodes in the first hidden layer, 35 nodes in the second hidden layer and 2 nodes in the output layer. The number of nodes in the input and the output layer are

determined from the problem at hand.

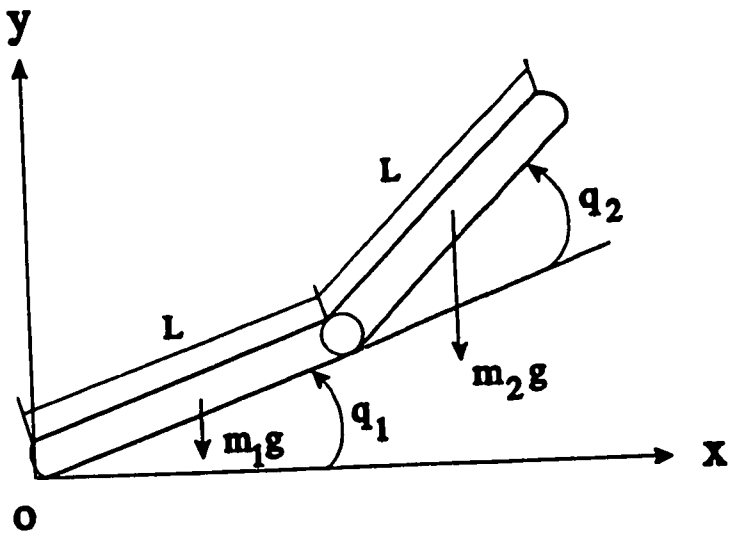

Figure 4.5 Two-Link Manipulator

Structure of the training set

The training set for the network consisted of

Inputs: desired position, desired velocity and the desired acceleration of the two joints, and

Target outputs: required torque at the two joints.

The output of the neural network and the output of the plant are given in Figures 4.7 and 4.8. Figure 4.7 shows the time history of the torque of the motor shaft 1 and figure 4.8 shows the time history of torque for link 2. The dotted line shows the control torque generated by the inverse dynamic controller and the solid line depicts the torque learned by the neural network. The two lines are almost the same showing that the desired and actual torques follow each other quite closely and the training results are very good.

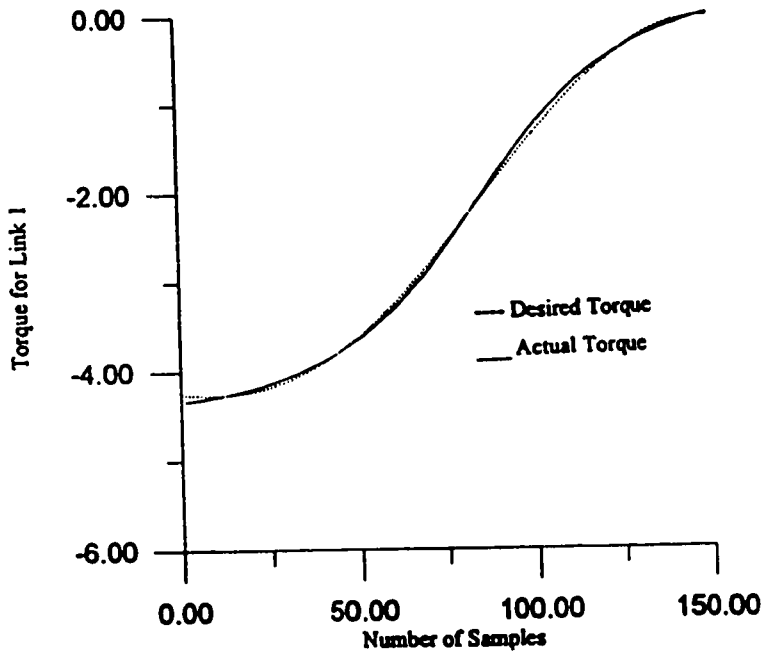

Figure 4.7 Comparison of the desired and the predicted torque for link1

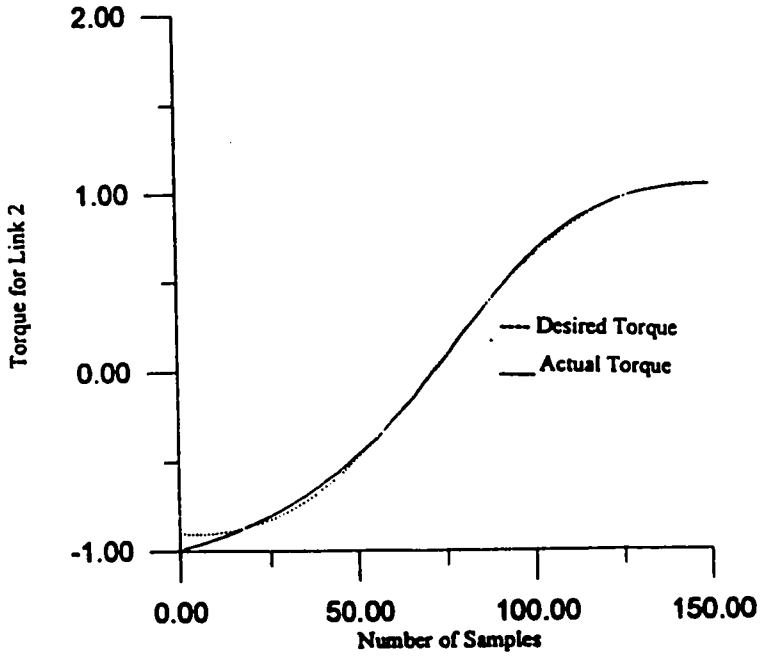

Figure 4.8 Comparison of the desired and predicted torque for link 2

4.4.2 Three-link Manipulator

A three-link manipulator is used for the simulation. The first three links (waist, shoulder, and elbow) are considered for the sake of simplicity. The remaining three links (wrist assembly) do not contribute significantly to the dynamics of the arm, but would add considerable computational complexity. Link 2 and 3 have dimensions 0.457×0.102×0.0127 m, and mass 0.9 kg. Link 1 is cylindrical with height 0.5 m, and diameter and mass of 0.1 m and 10.6 kg respectively. Various components of the equations of motion are not included as they can be found in any good book on robotics. Link 1 was moved from 0 to 90 degrees, link 2 was moved from 15 to 75 degrees and link 3 was moved from 20 to 80 degrees in 3 seconds. The desired trajectory of the robot is expressed as

$$\begin{bmatrix} q_1^d(t) \\ q_2^d(t) \\ q_3^d(t) \end{bmatrix} = \begin{bmatrix} 30.0t^2 - 6.66t^3 \\ 15.0 + 20.0t^2 - 4.44t^3 \\ 20.0 + 20.0t^2 - 4.44t^3 \end{bmatrix}$$

After the neural network learns the inverse dynamic model of the manipulator, the output of the neural network and plant is shown in Figs. 4.9 to 4.11. It is observed that the estimated torque (network output) is almost equal to the desired torque. As a result of the learning, the actual responses converge to the desired trajectories.

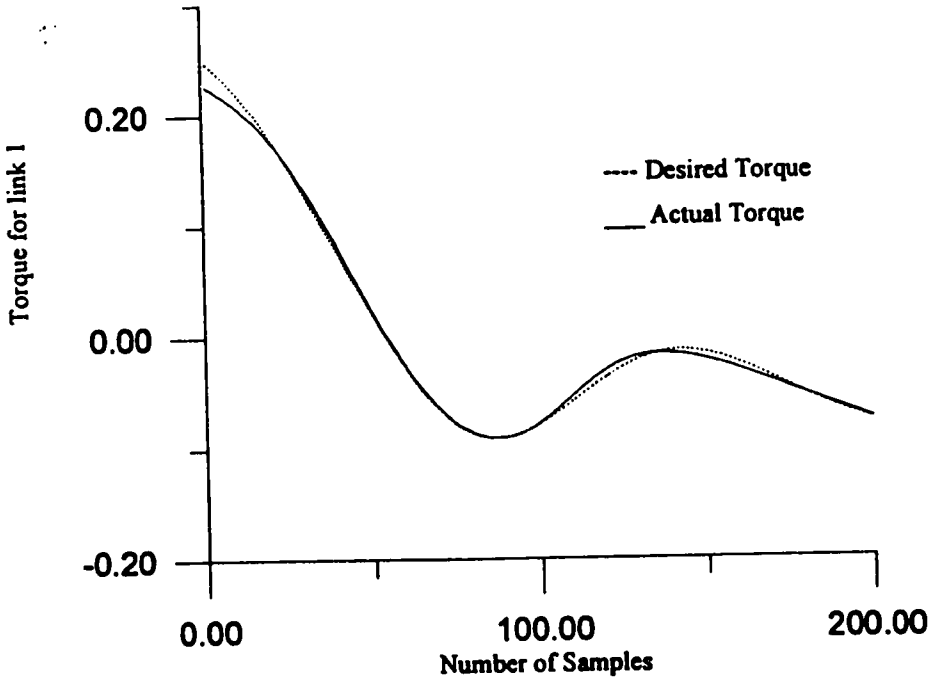

Figure 4.9 Comparison of the desired and predicted torque for link 1

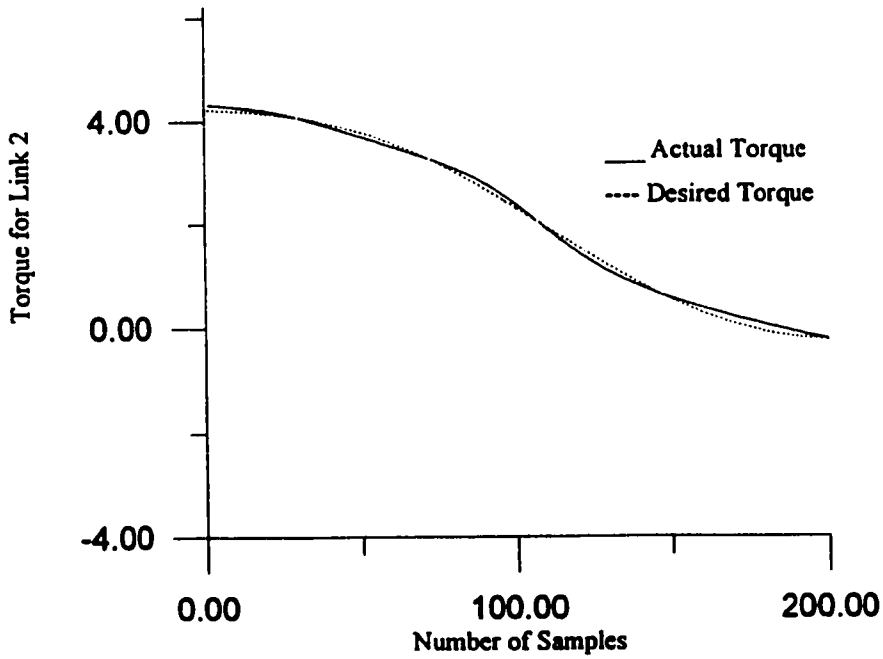

Figure 4.10 Comparison of the desired and predicted torque for link 2

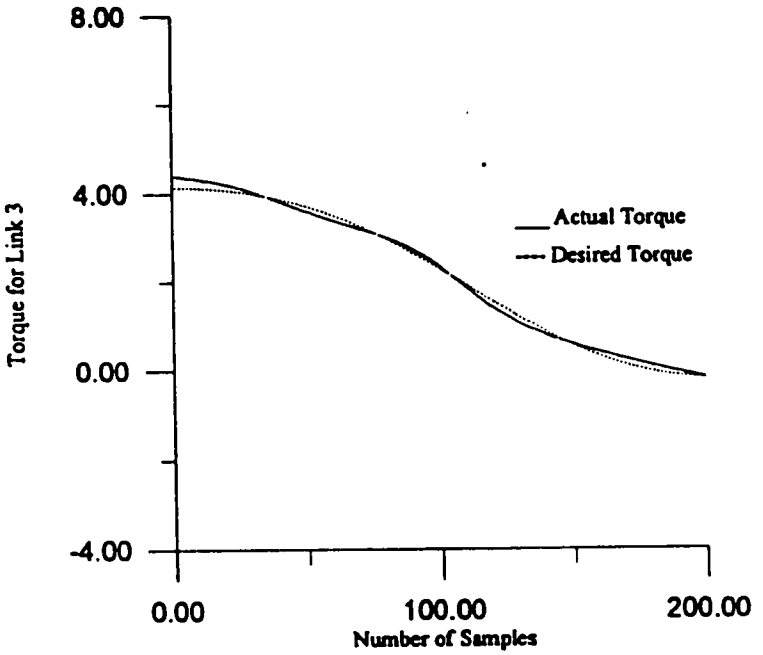

Figure 4.11 Comparison of the desired and predicted torque for link 3

4.5 Conclusions

This chapter presented a procedure to solve the problem of computing the complex nonlinear dynamics of a robotic system using a neural network. The neural network was used to estimate the torque, i.e., input to the robot, therefore, avoiding the computationally tedious task of using parameters' estimators. Simulation results show that modelling can be satisfactorily achieved. They also confirm that neural network identifier can represent nonlinear system characteristic very well. Because of its ability to identify nonlinear relationships and because it is fundamentally a parallel technique, a backpropagation algorithm is a particularly promising neural algorithm for application in robotics. A model learning

scheme was also proposed. The elementary training of the neural network using an obtained model can be fulfilled off line to give an approximate model and needs no data recording of actual manipulator operation. After the model learning is finished, the neural network learns structured/unstructured uncertainties on line. This learning procedure is effective and efficient in learning the manipulator dynamics, and an error convergence rate is fast, making the learning less computationally intensive. In summary we can say that neural networks are powerful-tool in modelling and identification of unknown dynamical systems.

CHAPTER 5

MOTION CONTROL OF ROBOTIC MANIPULATORS

5.1 INTRODUCTION

In control system design, solutions to problems encountered generally involve derivations of dynamic equations of system under study and design of real-time control algorithms to meet certain system specifications. To be more specific, the solutions involve modelling of the system based on physical laws or system identification based on control inputs applied to the system. Typically, the system to be controlled is described by a set of linear or nonlinear differential equations. Implementation of controllers can be carried out either in hardware or software.

Robotic manipulators have become increasingly important in the field of flexible automation. The dynamic behaviour of robotic manipulators is defined by coupled nonlinear differential equations. One of the important and fundamental tasks of the robot manipulators is position control. Present day industrial robots operate with very simple controllers, which do not yet take full advantage of the inexpensive computer power that has become available. The result is that these fairly expensive mechanisms are not being utilized to their full potential in terms of the speed and precision of their movement. With a more powerful control

computer it is possible to use a dynamic model of the manipulators the heart of a sophisticated control algorithm. This dynamic model allows the control algorithm to know how to control the manipulator's actuators in order to compensate for the complicated effects of inertial, centrifugal, Coriolis, gravity, and friction forces when the robot is in motion. The result is that the manipulator can be made to follow a desired trajectory through space with smaller tracking errors, or perhaps move faster while maintaining good tracking.

Through the years, considerable research efforts have been made in control of the robotic manipulator. In order to achieve accurate trajectory tracking and good control performance, a number of different control schemes have been developed. Computed torque control is one of the most intuitive schemes, which relies on the exact cancellation of the nonlinear dynamics of the manipulator system. Such a scheme has the disadvantage that the exact dynamic model is required. Furthermore, the inertia properties and gravitational loads of the robot manipulator vary during operation and are dependent on its payload, which may not be necessarily known in advance. To overcome this problem, adaptive control strategies for robot manipulators have been developed, and have attracted the interests of many researchers (Craig 1988, Slotine and Li 1988). These adaptive control methods have the advantage that, in general, they require no *a priori* knowledge of unknown parameters such as the mass of the payload. These techniques are based on using the known model form to construct a control law with unknown parameters and then using the system data to estimate these parameters. Although adaptive control procedures have been applied in the robot control, they are limited by the need to assume that the forms of the system equations are

known. For a complex process, however, the forms of the system equations may be unknown, making it impossible to determine the required control law for use in existing adaptive control procedures. This problem provides the motivation for considering the use of neural networks in adaptive control.

Neural networks have recently attracted much attention for their potential to address a number of difficult problems in modelling. One of the areas receiving a significant portion of the attention is the use of neural networks for controlling and regulating nonlinear dynamic systems. Traditionally, developing controllers for nonlinear dynamic systems has been difficult, even in deterministic settings where the equations governing the system dynamics are fully known. Neural networks however, offer potential for addressing control problems even broader than this, including the control of stochastic systems with unknown nonlinear dynamics. The basic idea of this sort of control strategy is to combine humans' experience and knowledge of the control system in order to learn, adapt and make decisions to meet the desired performance criterion. Neural network control design is done in two steps. First a neural network is used to approximate the dynamic model for the system. This approximation is usually carried out off-line and then when sufficiently an accurate approximation is obtained, an appropriate control strategy using this approximation can be constructed.

A neural network-based scheme for the control of a robotic manipulator is investigated. The main idea is that by using a neural network to learn the characteristics of the robot system (or specifically its inverse dynamics) accurate trajectory following and good performance results are obtained. The intention is to demonstrate the potential of a neural

network in conjunction with a linear compensator to perform such a function. A neural network is used in a feedforward loop with a conventional feedback proportional plus derivative (PD) controller. As learning went on, the feedforward/feedback compensation tended to move to a feedforward path with little feedback compensation. A linear compensator (PD) is unable to control a robotic manipulator under different conditions of operation. A neural network can be trained to produce the large portion of the control input; however, a hybrid combination of the neural network and the linear compensator gives the best results. Furthermore, we demonstrate that such a hybrid system can tolerate changes in the operating conditions. Finally, we demonstrate generalization within the training domain through accurately predicting a case that was absent in the training domain. The modified backpropagation learning algorithm introduced in chapter 3 is used to train the neural network. The suggested method is applied to the control of a two-link manipulator and a three-link manipulator. The results demonstrate that the proposed method gives better error minimization and faster convergence. The resulting controller is sufficiently robust with respect to the changing conditions.

5.2 NEURAL NETWORK-BASED CONTRCLLER

Dynamics of a manipulator involves nonlinear mapping between applied joint torques and the joint positions, velocities and accelerations. These relationships can be described by a set of second-order, nonlinear and highly coupled differential equations with uncertainty as a robot may work under unknown and changing environments and execute different tasks.

The equations of motion for a manipulator can be written as

$$M(q)\ddot{q} + C(q,\dot{q}) + G = \tau \qquad (1)$$

where $\tau \in R^n$ is the joint actuator torque and $q \in R^n$ is the generalized joint angles, and $M(q)$ is a matrix, usually referred to as the manipulator mass matrix containing the kinetic energy functions of the manipulator. $C(q,\dot{q})$ represents forces arising from Coriolis and centrifugal forces and G is the force due to gravitation. Generally equation (1) is very complicated for all but the simplest manipulator configurations.

The manipulator control problem is a problem in which the end-effector is made to track the reference values representing the desired value. The inputs to the manipulator system can be chosen as the generalized torques produced by the joint actuators, and the outputs to be controlled are the positions. The actual positions and velocities of the joints in a manipulator can be measured by encoders and tachometers, respectively. The values of the accelerations can be calculated by hardware or software programming. These variables can then be compared with their desired values using feedback loops to from the position, velocity, and acceleration errors for driving the plant.

For controlling the motion of the manipulator along a desired path, the actuator torque functions have to be calculated from the equations of motion. This requires incorporating the effects of inertia, coupling between joints (Coriolis and centrifugal forces), gravity loading, gear friction and backlash, and the dynamics of the control devices. The inverse dynamic approach is particularly important for control of robots, and can be used to compensate for highly coupled and nonlinear arm dynamics.

To make the end-effector of a manipulator track the desired nominal trajectory, the generalized torques applied to the system should have the appropriate (nominal) values that result in the desired motion under ideal conditions. The controller generating these values can be referred to as the primary controller. Thus, it compensates for the nonlinear effects. Since the mathematical model used is usually not exact, and since the system is subject to disturbances, undesirable deviations (errors) of the actual motion from the nominal trajectory can be corrected by means of an additional controller called a secondary controller. Many strategies have been developed for controlling the motion of a robot. These require the knowledge of the mathematical model of the robot manipulator to be controlled. Considerable effort has been devoted in the past for developing dynamical models of the from given by (1) for different kinds of manipulators, which are in turn used for designing appropriate control strategies for executing desired motions. Due to the highly nonlinear functions and the existing coupling between the joint motions, evaluation of the model parameters is not a simple task, however. Several schemes have been devised which incorporate the full dynamic model of the arm in the controller design (Craig 1988). Adaptive techniques are essential for the stable and robust performance of the manipulator. This is due to the fact that manipulators are systems of nonlinear and time-varying nature. In general, the aim of a model-based adaptive control algorithm is to estimate the parameters of the model, and then use the estimates to compute the control scheme by an appropriate design method. The literature on adaptive control is vast and several techniques were proposed by researchers in the last few years (Narendra and Annaswamy 1989, Slotine and Li 1988). Adaptive control techniques

are divided into two approaches, namely, (1) direct adaptive control and (2) indirect adaptive control. In the former approach, the parameters of the controller are directly adjusted to reduce some norm of the output error. While in the latter one, the parameters of the plant are estimated and accordingly, the controller is updated so that to minimize the error between the model and the plant. These techniques are based on nonlinear laws which make them difficult to derive. For implementing an adaptive control that can deliver satisfactory performance in the face of variations in plant parameters and external dynamics, parameters' estimation followed by updating of control is needed at each adaptation step which further compounds the complexities. Furthermore, their complexity grows geometrically with the number of unknown parameters which makes them non-robust. In addition, all these techniques incur a heavy computational load which hinders their real-time applicability. Several researchers have presented learning control schemes for improving the performance in trajectory following tasks over successive attempts at following the same trajectory (Arimoto *et al.* 1984, Atkeson and McIntyre 1986). Typically, control torques for each time instant in the trajectory are adjusted iteratively based on observed trajectory errors at similar times during previous attempts. In the results presented by these researchers, the trajectories followed consistently converged on the ideal trajectories over several repetitions. A drawback to such control techniques is that they are only applicable to operations which are repetitive.

Recently there has been considerable interest in learning in the form of simple models of networks of neurons. The recent resurgence of neural networks makes it possible to seek solutions for robotic problems. The basic theme is that of using the network to learn the

characteristics of the robot/sensor system, rather than having to specify explicit robotic models. We will be discussing neural network based control scheme for the motion control of a robotic manipulator. The block diagram of the control system is shown in Fig. 1. No parameter estimation is required as the control law does not depend on the parameter estimates. The control signals are the outputs of a neural network where the network's parameters (synaptic weights) are adjusted by an error signal that quantifies the amount of deviation between the model and the system. In addition, the neural network is also robust.

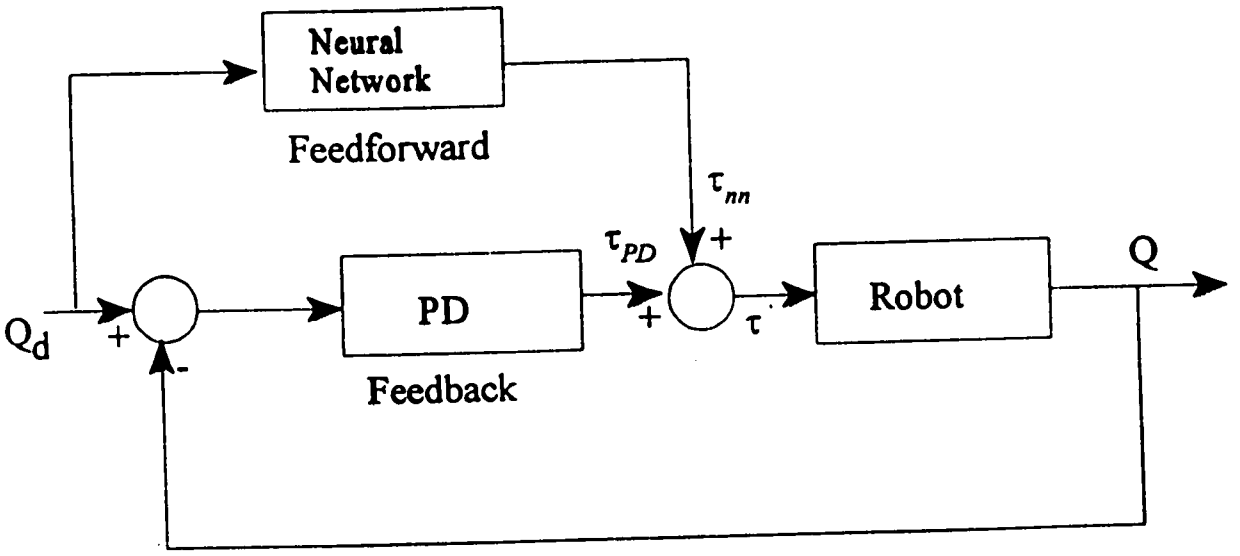

Figure 5.1 Neural network based control scheme

In this scheme neural network controller acts as a primary controller and PD controller is used as a secondary controller. It will be seen that most of the system input, i.e., torque is provided by the neural network controller. The fixed-gain PD controller ensures adequate performance prior to convergence of the network weights and provides the training signal to the network for the purpose of adapting the weights. It also reduces steady-state

output errors due to disturbance inputs. As can be seen from the figure 1, the sum of the outputs of the neural network and the feedback controller will be the actual input torque to the robotic manipulator. This can be expressed as

$$\tau = \tau_{PD} + \tau_{nn} \tag{2}$$

where τ_{nn} is the output of the neural network and τ_{PD} is the output of feedback (PD) controller. The feedback controller plays a role in making the whole system stable. The neural network has been trained off-line to approximate the inverse dynamic model of the manipulator. The learning scheme to get the inverse dynamic model of the manipulator has been discussed in the last chapter. Once the neural network finishes learning, it produces an approximate inverse dynamic model, described as

$$\tau_{nn} = \hat{D}(q_d)\ddot{q}_d + \hat{C}(q_d,\dot{q}_d) + \hat{G} \tag{3}$$

where $D(q_d)$, $C(q_d,\dot{q}_d)$ and $G(q_d)$ are the estimated values of the various parameters of the equation (1). Thus, a neural network learns the following nonlinear relations

$$\tau_{nn} = f(\ddot{q}_d, \dot{q}_d, q_d, w, \gamma)$$

where w and γ are the synaptic weights between nodes and sigmoid function parameter respectively. The total torque to the manipulator is given by

$$\tau_{nn} + \tau_{pd} = \hat{D}(q_d)\ddot{q}_d + \hat{C}(q_d,\dot{q}_d) + \hat{G} + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) \tag{4}$$

Writing the equation for the closed loop system, we have

$$\ddot{E} + K_v\dot{E} + K_pE = \hat{D}(q_d)^{-1}[(D(q) - \hat{D}(q_d))\ddot{q} + (C(q,\dot{q}) - \hat{C}(q_d,\dot{q}_d)) + (G - \hat{G})] \tag{5}$$

If the model were exact then the right-hand side of (5) would be zero and the errors would disappear. When the model is not exact, the mismatch between the actual and modeled

parameters will produce these error signals that can be utilised to train the neural network on-line so that the errors go to zero.

5.3 RESULTS OF SIMULATION

Simulations were done to verify the neural controller compensating unstructured uncertainties. The proposed scheme has been applied to a two-link manipulator and a three-link manipulator.

5.3.1 Two-link Manipulator

Various components of equation (1) for a two-link manipulator are given in the previous chapter. The desired trajectory of the robot is expressed as

$$\begin{bmatrix} q_1^d(t) \\ q_2^d(t) \end{bmatrix} = \begin{bmatrix} 15.0+20.0t^2-4.44t^3 \\ 20.0+20.0t^2-4.44t^3 \end{bmatrix}$$

The number of input and output nodes is determined from the problem at hand whereas the numbers of nodes in the hidden layers are flexible. The neural network employed in the simulation consists of an input layer with six nodes, the first hidden layer with 25 nodes, the second hidden layer with 35 nodes and an output layer with 2 nodes. This is symbolized as $N_{6,25,35,2}$. Learning the dynamics of manipulator has been discussed in chapter 4.

Comparisons of the desired angular position with that produced by the neurocontroller with linear compensation are shown in Figs. 5.2 and 5.3. The figures show the time history of the shaft angle. It is observed that the predicted trajectory and the desired trajectory are coinciding and the results are almost indistinguishable. This clearly shows that a combination of a neural network with a linear compensator is more than capable of controlling the motion

of a robotic manipulator.

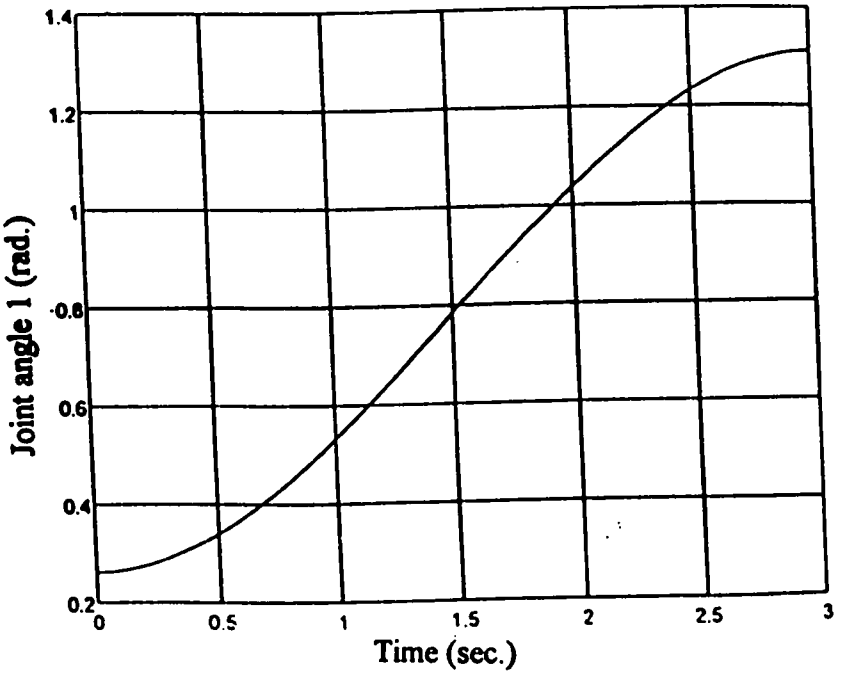

Figure 5.2 Comparison of the desired and predicted time histories of the joint angle 1

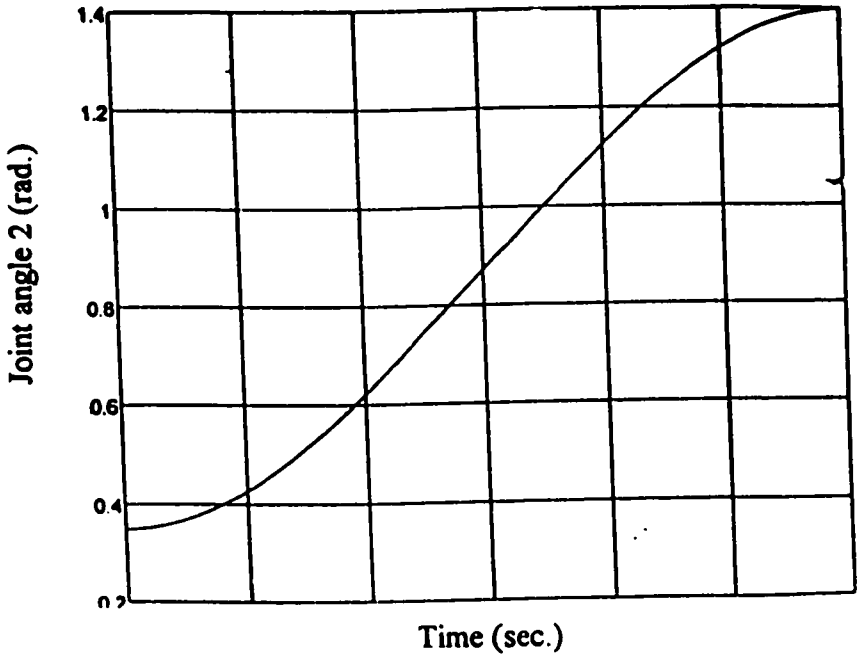

Figure 5.3 Comparison of the desired and predicted time histories of the joint angle 2

Robustness of Control Strategy

Robustness of the model was tested without retuning the neural network. Although most industrial robots perform repetitive tasks, often robotic systems are subject to the payload variations. This may arise from the fact that the standard articles handled by the robot may have slightly different masses due to the variations in the manufacturing process. Thus, it is important to determine how the controller deals with these uncertainties. As expected, simulations have shown that the proposed neural network controller has better robustness and is less sensitive to the payload variations. Figs. 5.4 and 5.5 show the results of simulation with about 10% variation from standard payload used in the training process.

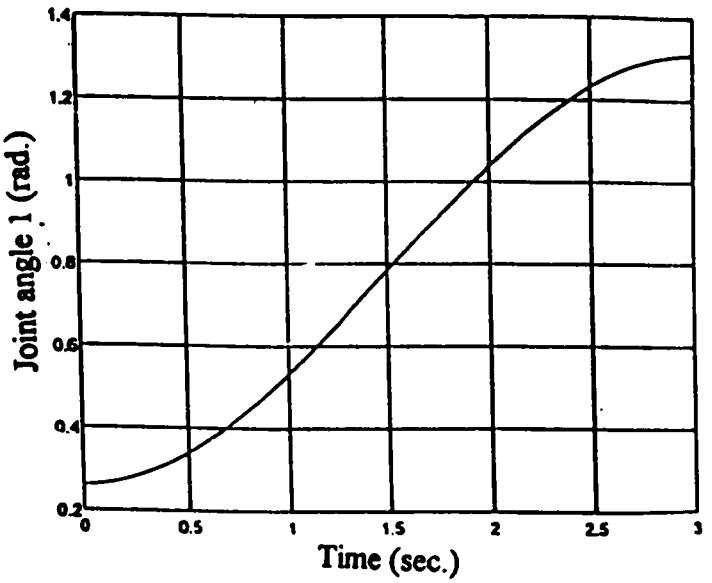

Figure 5.4 Comparison of the desired and predicted time histories of the joint angle 1 with 10% payload variation

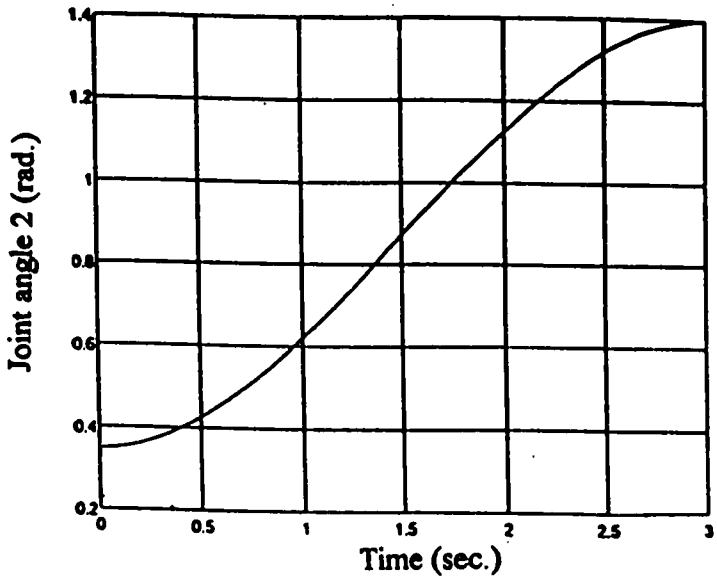

Figure 5.5 Comparison of the desired and predicted time histories of the joint angle 2 with 10% payload variation

In Figs 5.6 and 5.7 describe the simulation results with about 50% variation from the standard payload. It is observed that even in this case the desired and predicted results are almost indistinguishable. In most cases, a robot will be performing the same task, i.e., it will be following the same trajectory. Simulations have shown that even if the trajectory is different from the one used in training, the proposed scheme is very robust. The results are shown in Figs. 5.8 and Figs. 5.9.

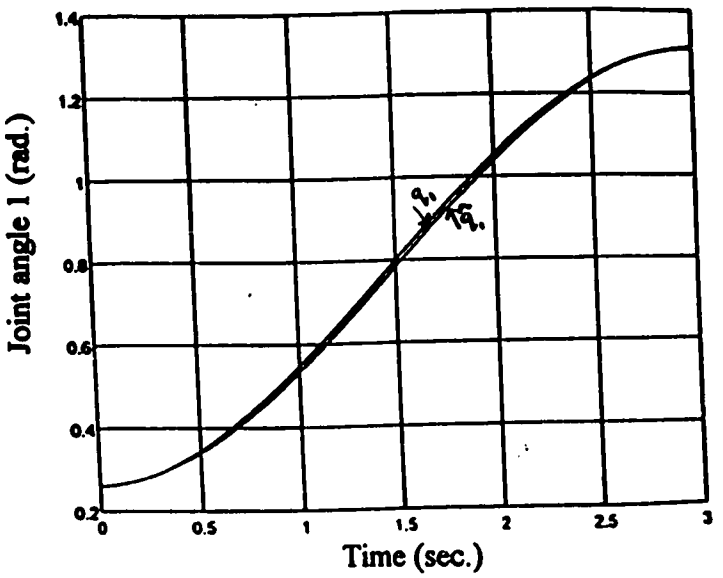

Figure 5.6 Comparison of the desired and predicted time histories of the joint angle 1with 50% payload variation

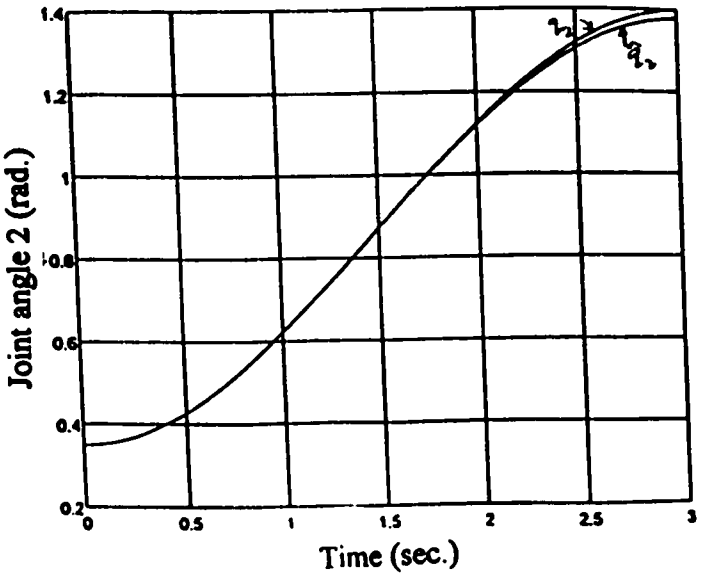

Figure 5.7 Comparison of the desired and predicted time histories of the joint angle 2 with 50% payload variation

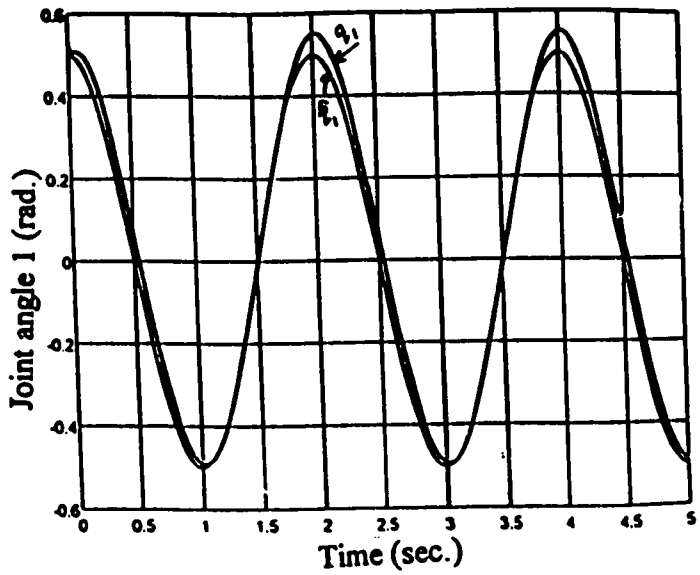

Figure 5.8 Comparison of the desired and predicted time histories of the joint angle 1

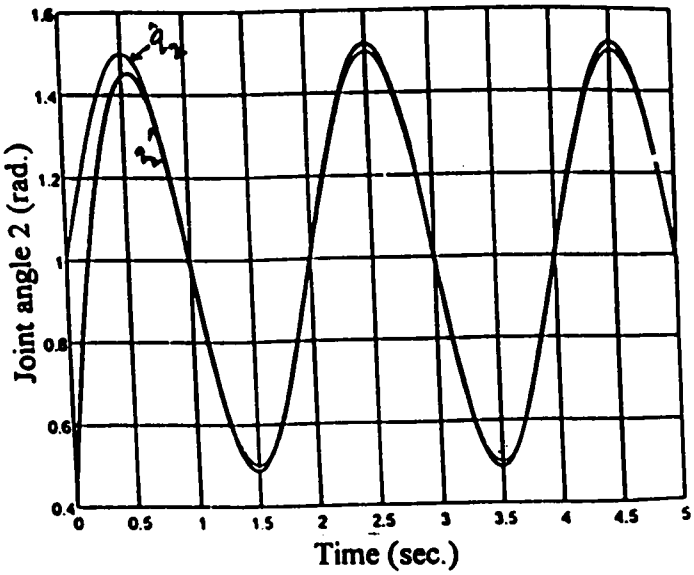

Figure 5.9 Comparison of the desired and predicted time histories of the joint angle 2

To see the effect of dynamics changing fast i.e., when the manipulator is moved fast or the same task is required to be completed in less time. Simulation were performed and the results are shown in figures 5.10 and 5.11. Even in this case the proposed controller has better robustness.

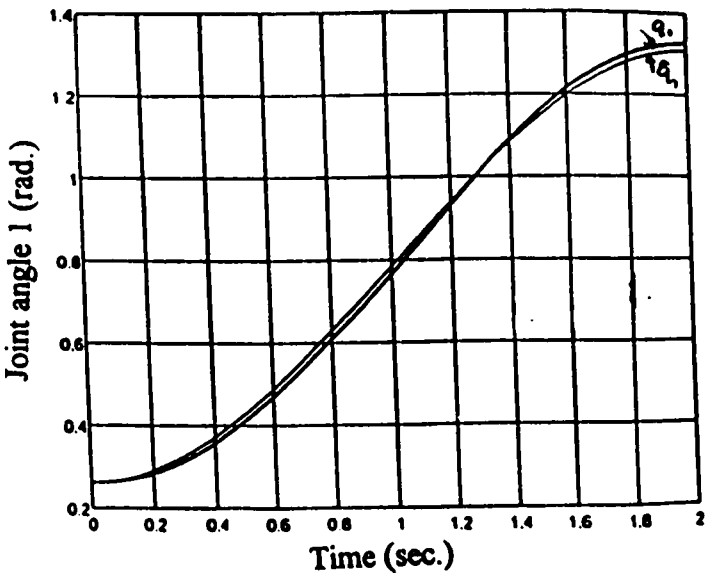

Figure 5.10 Comparison of the desired and predicted time histories of the joint angle 1

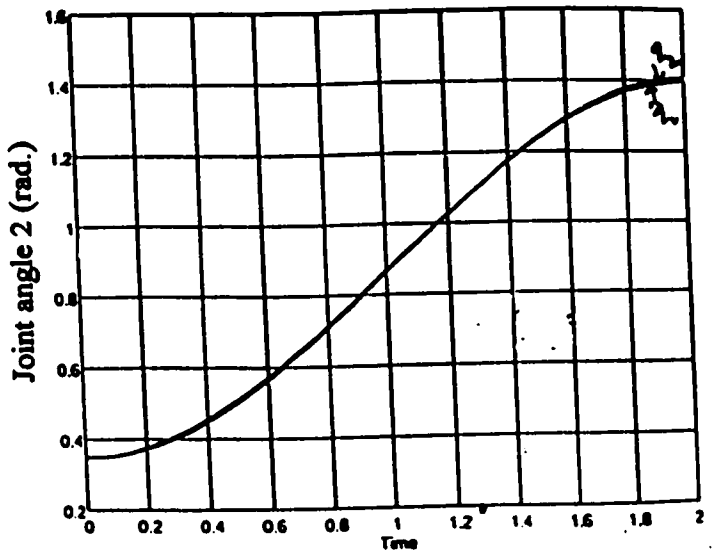

Figure 5.11 Comparison of the desired and predicted time historics of the joint angle 2

5.3.2 Three-Link Manipulator

A three-link manipulator shown in figure 1 is used for the simulation. The first three links (waist, shoulder, and elbow) are considered for the sake of simplicity. The remaining three links (wrist assembly) do not contribute significantly to the dynamics of the arm, but would add considerable computational complexity. Various components of the equations of motion are not included. The value of the various parameters used in the simulation has been given in section .

Link 1 was moved from 0 to 90 degrees, link 2 was moved from 15 to 75 degrees and link 3 was moved from 20 to 80 degrees in 3 seconds. The desired trajectory of the robot is expressed as

$$\begin{bmatrix} q_1^d(t) \\ q_2^d(t) \\ q_3^d(t) \end{bmatrix} = \begin{bmatrix} 30.0t^2 - 6.66t^3 \\ 15.0 + 20.0t^2 - 4.44t^3 \\ 20.0 + 20.0t^2 - 4.44t^3 \end{bmatrix}$$

The tracking responses of the proposed scheme are shown in Figs. 5.12 to 5.14 . As a result of the learning, the actual responses converge to the desired trajectories. In most of the cases, a manipulator will be performing the same task repeatedly, i.e., it will be following the same trajectory. Simulations have shown that even if the trajectory is different from the one used in training, the proposed scheme behaves well. The excellent correlation between the desired trajectory and that produced by the combination of the network and the linear compensator is quite evident. Producing the correct results suggest that the network has learned to generalize.

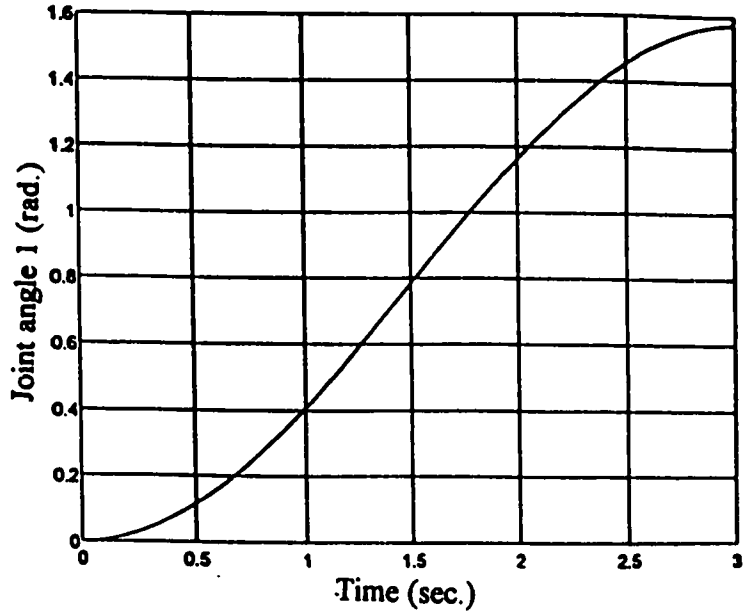

Figure 5.12 Comparison of the desired and predicted time histories of the joint angle 1

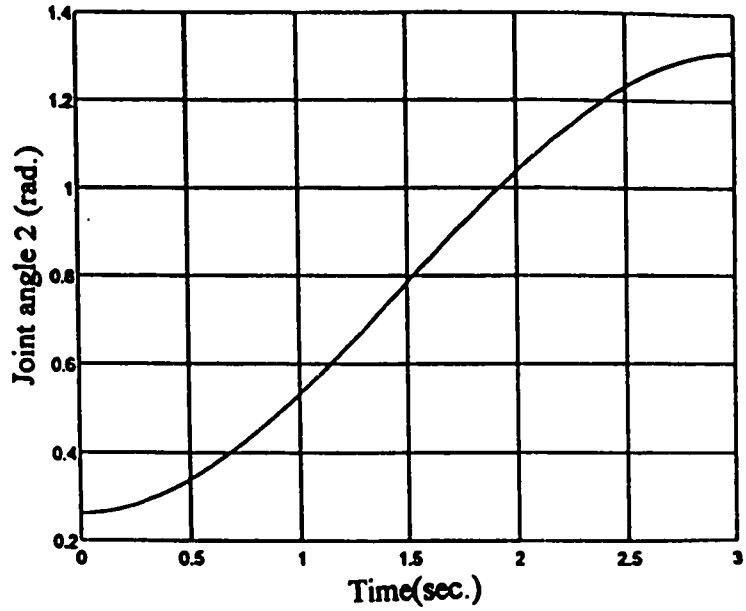

Figure 5.13 Comparison of the desired and predicted time histories of the joint angle 2

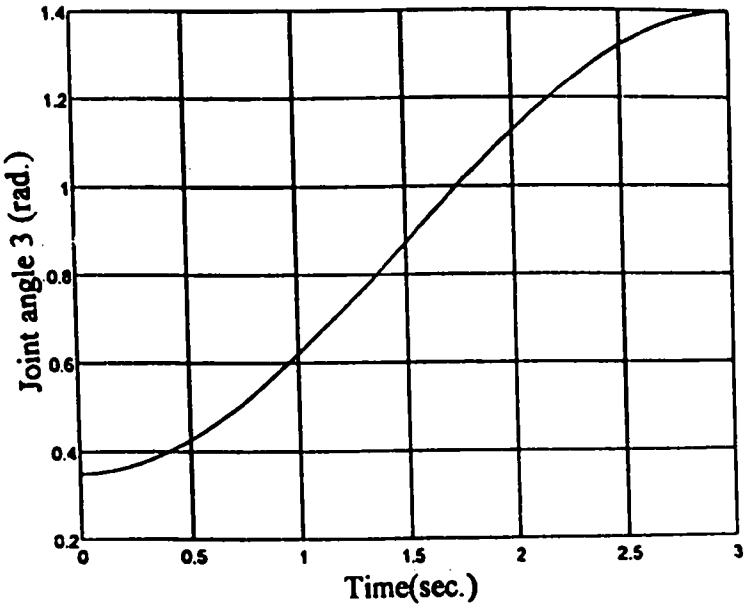

Figure 5.14 Comparison of the desired and predicted time histories of the joint angle 3

5.4 CONCLUSIONS

The feasibility of developing a controller based on neural networks for nonlinear dynamical plants has also been demonstrated. A case study of a two-link robot and a three-link manipulator has demonstrated that the proposed scheme is very promising, which achieved satisfactory performance. The inverse model with neural network structure found to possess the good performance. After the neural network has compensated perfectly or partially for the nonlinearity of the controlled object through learning, the responses of the controlled object follow the desired responses supplied by inverse reference model implemented in the conventional feedback controller. The combination of the neural network and linear compensator proved to be very valuable in controlling the manipulator. In most cases, corrections provided by this scheme resulted in exact reproduction of the desired results. Generalization was shown through reproduction of a case that was absent from the training set. The results permit reaching goals with arbitrarily low error even though the inverse dynamics, which was learned by the neural network is only an approximation. The inverse model with neural network structure gave better performance. After the neural network has compensated perfectly or partially for the nonlinearity of the controlled object through learning, the responses of the controlled object follow the desired responses supplied by the inverse reference model implemented in the conventional feedback controller. In most cases, corrections provided by this scheme resulted in exact reproduction of the desired results. The advantage of the proposed controller is that there is no need of on-line training as the neural network is trained off-line. Thus, this avoids the problem of heavy on-line

computation required by adaptive controllers and robust controllers. Furthermore, the proposed controller has the potential to handle the effect of changes in the payload and variations in the operating conditions. As a result, the adaptive capabilities of the neural network controller to the unstructured effects are clarified.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 INTRODUCTION

The purpose of conducting experiments was to test the real-time features, and numerical characteristics of the proposed neurocontrol scheme. While there seems to be a widespread interest in the robotic control problem within the neural network and robotics communities, little has been reported in the nature of actual robot control experiments. This is due, at least in part, to the computational speed and stability problems encountered when using typical neural models in networks of sufficient complexity to be useful for a realistic robot control problem. This chapter presents the results of real-time experiments which involved learning the dynamics of a manipulator. Experiments were conducted using FLEXROD, an experimental two-link revolute joint manipulator and a single motor fixed with a multi-degree of freedom gripper designed and built in the Mechanical Engineering Department of McMaster University. Also, there are many uncertain factors in a real time setup than found in simulation models, which often present challenges to model-based control approaches.

The general procedure of control systems development involves the following steps:

(i) control task definition; (ii) control system hardware setups; (iii) modeling, parametric identification; (iv) control scheme selection and simulation tests; (v) real-time control code programming and implementation; and (vi) system testing.

Since FLEXROD and single link robot have been set up already and a neurocontrol scheme has been developed and tested by simulation, only steps (v) and (vi) remain to be done.

6.2 Experimental Set up of a FLEXROD

Figures 6.1 to 6.3 show the FLEXROD. It consists of three main parts:

(1) The mechanical arm, complete with DC motors, encoders, transmission gears and a light weight aluminium structure.

(2) The controller-amplifier package, composed of a three processor parallel computer controller and a Pulse width modulated amplifier unit

(3) The personal computer interface, that allows for communication and programming of the controller.

The mechanical arm is a two-link elbow arm made of aluminum, and is actuated by two permanent magnet DC motors installed on the shoulder, as shown in Figure 6.1. One of the motors drives the first link, through a harmonic drive speed reducer, while the second motor remotely drives the second link through a harmonic drive and a pulley and a timing belt. Each motor has a built-in differential encoder, to provide the motor's position. The parameters of the arm are shown in table 6.1:

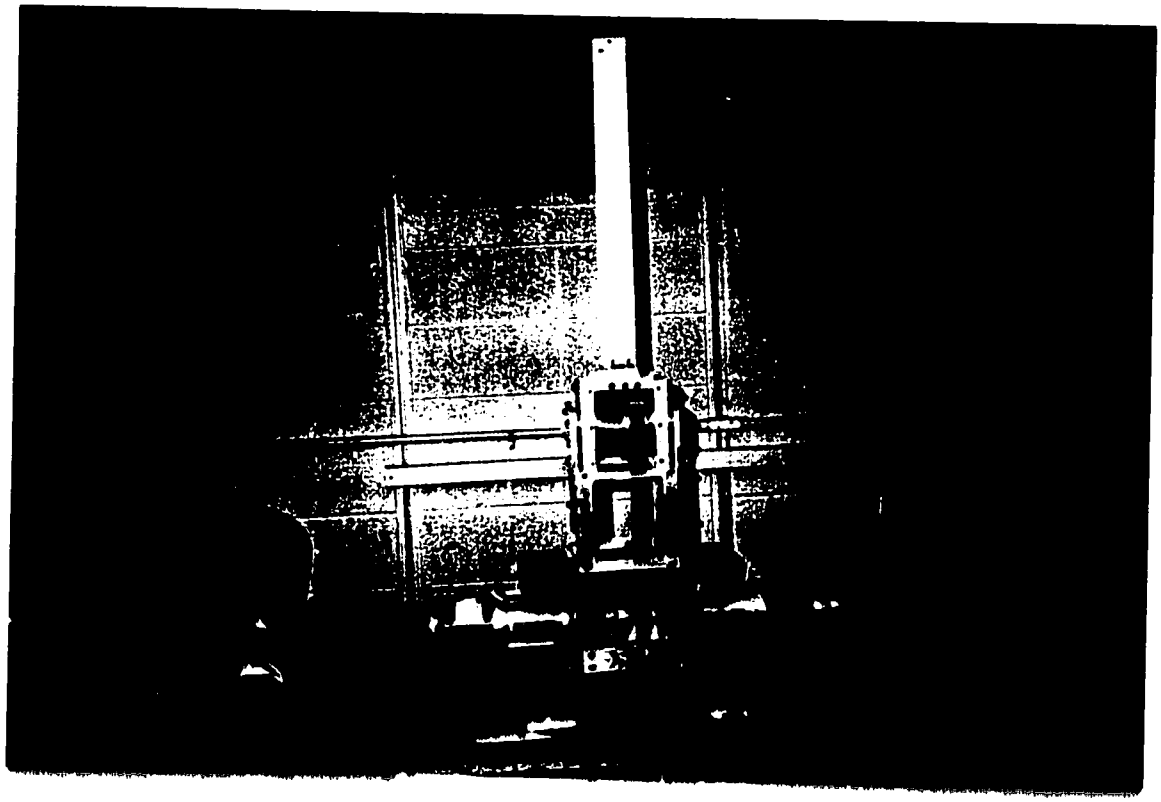

Figure 6.1 Mechanical Arm in Home Position

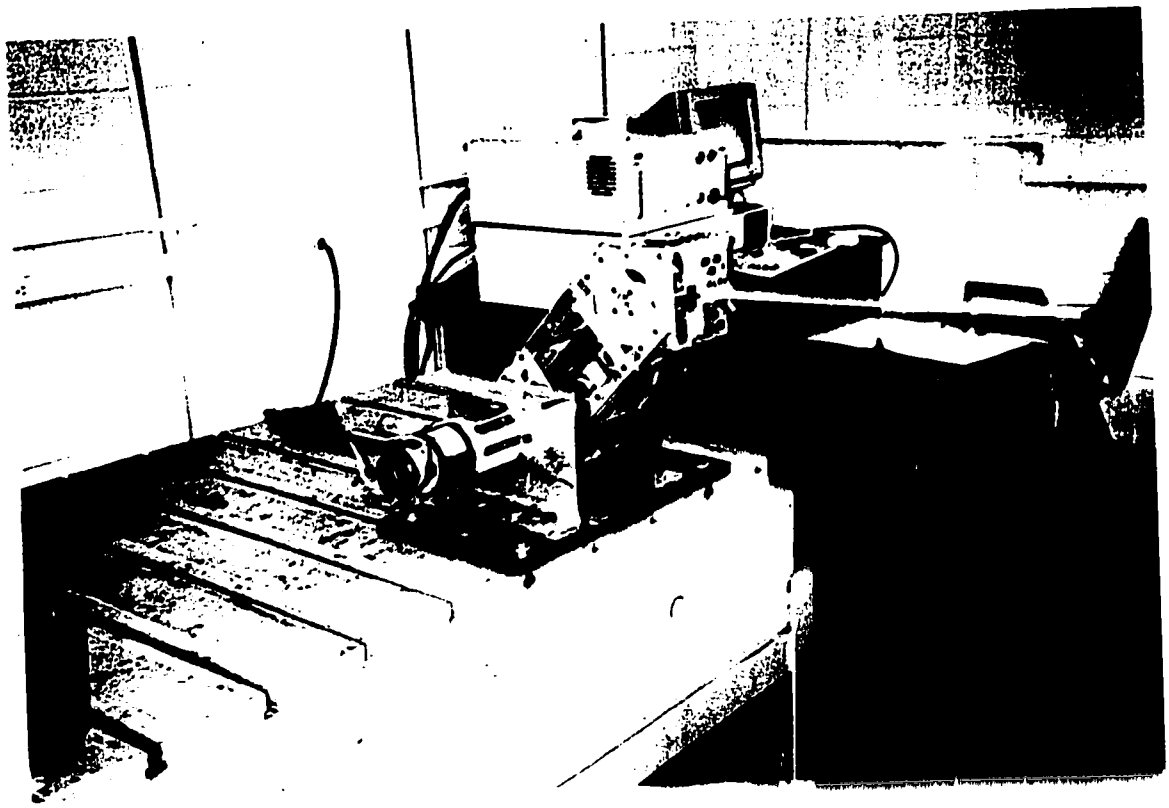

Figure 6.2 Arm in Final Position

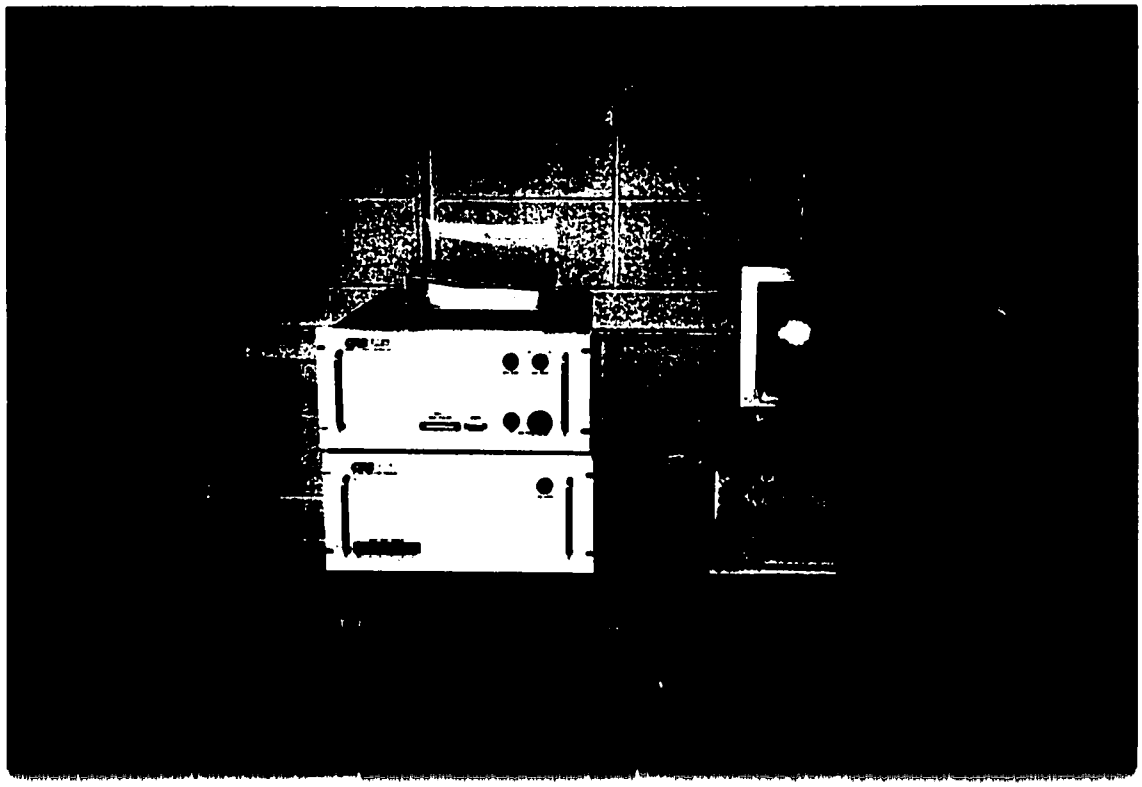

Figure 6.3 Controller-Amplifier and PC Interface

Table 6.1

Parameter	Mass (Kg)	Centroidal Inertia (kg-m^2)	Length (m)	Position of the Centre of Mass (m)	Actuator Inertia (kg-m^2)	Gear ratio
Link 1	4.8	0.025	0.305	0.145	9.532×10^{-5}	100
Link 2	4.8	0.025	0.585	0.220	9.532×10^{-5}	100

More information about the experimental set up can be found in (Carrara 1995). In the

control law joint angles in radians and joint velocities in radians per second are required. Both can be determined from the readings of the encoder. The joint angles in radians from the encoder readings are found using the following relations:

$$q_1 = \frac{2\pi\eta}{E_r} E_1$$

$$q_2 = \frac{q_1}{\eta} - \frac{2\pi\eta}{E_r} E_2$$

where E is the encoder resolution (lines/rev) and E_1 and E_2 are encoder readings of motor 1 and motor 2 respectively. The joint velocities have to be determined by numerical differentiation of the joint angles. Since numerical differentiation is inherently noisy, this estimate is smoothed out by using a lowpass digital filter, with a cutoff frequency of 100 Hz. A fourth order Butterworth filter is used for this purpose. Another issue related to the implementation of the control law is the conversion of the output torque, τ, received from the control routine into a digital command signal which in turn is converted into a voltage signal which is sent to the PWM amplifier. This involves using the digital to analog gain ($D/A_{const.}$) of the controller, the voltage to current gain (V/C_{amp}) of the amplifier, and the torque constant K_T (current to torque gain) of the DC motors, and the transmission ration η of the harmonic drives. This is done according to the following formula:

$$DCS = (K_{DSC}\tau) = \frac{(D/A_{const})\times(V/C_{amp})\times K_T}{\eta}\tau$$

The value of the gain K_{DSC} = 11.4285 Digital Units/Nm.

6.2.1 Experimental Results

For the experimental verification of the neurocontrol strategy, the previously developed control law routines are adapted and implemented into the controller code, using the same parameters used in the simulation phase. The neurocontroller was first trained using the nominal model of the manipulator. Then the off-line learned weights were used in the control algorithm. Different trajectories were used to test the robustness of the neurocontrol. The link 1 and 2 were moved were 0 to 45 degrees and back to the home position. Cubic trajectories are specified between the various goal points according to standard trajectory planning schemes (Craig 1989). Fig. 6.4 and 6.5 show the desired and actual angular positions of the link 1 and 2. It is observed that the qualitative behaviour is very close to that obtained in the simulation. Excellent agreement between the actual values of the joint positions and those given by the neural control scheme is quite evident from these figures. Figures 6.6 and 6.7 shows an intermediate case, absent in the training set. Another experiment was performed in which links were moved from 0 to 45 degrees and back home and this was repeated. The results are shown in the figures 6.6 and 6.7. Again, excellent correlation between the desired angular position of the joint and that given by the combination of a neural network and the linear compensator is quite evident. Producing the correct results for this interpolative case suggests that the network has learned to generalize. A more rigorous demonstration would require presentation of many cases. This clearly shows that a combination of a neural network with a linear compensator is capable of controlling the motion of a robotic manipulator. In summary, the control precision of experiments is quite acceptable.

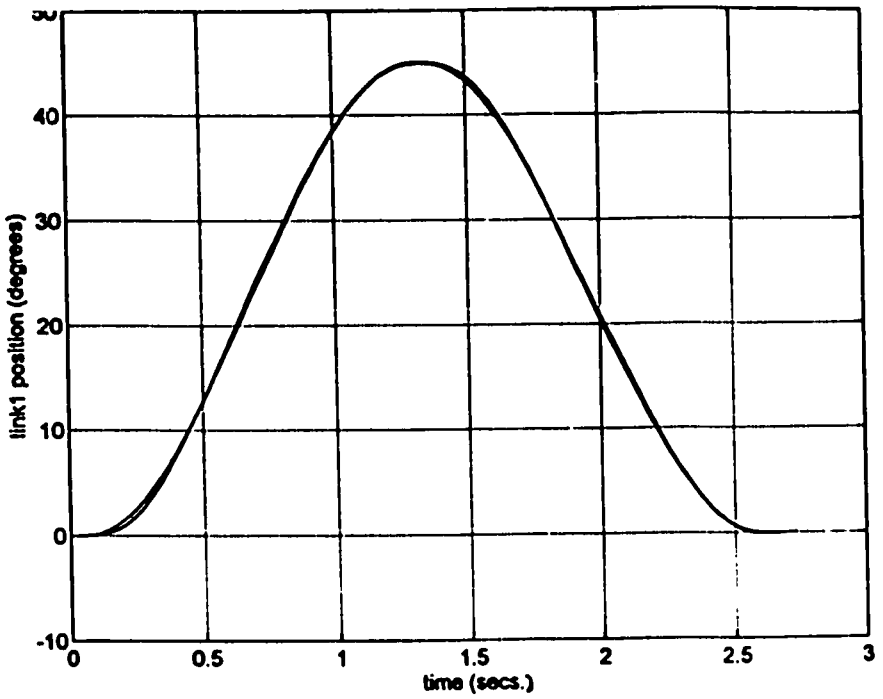

Figure 6.4 Time History of the Joint Angle 1

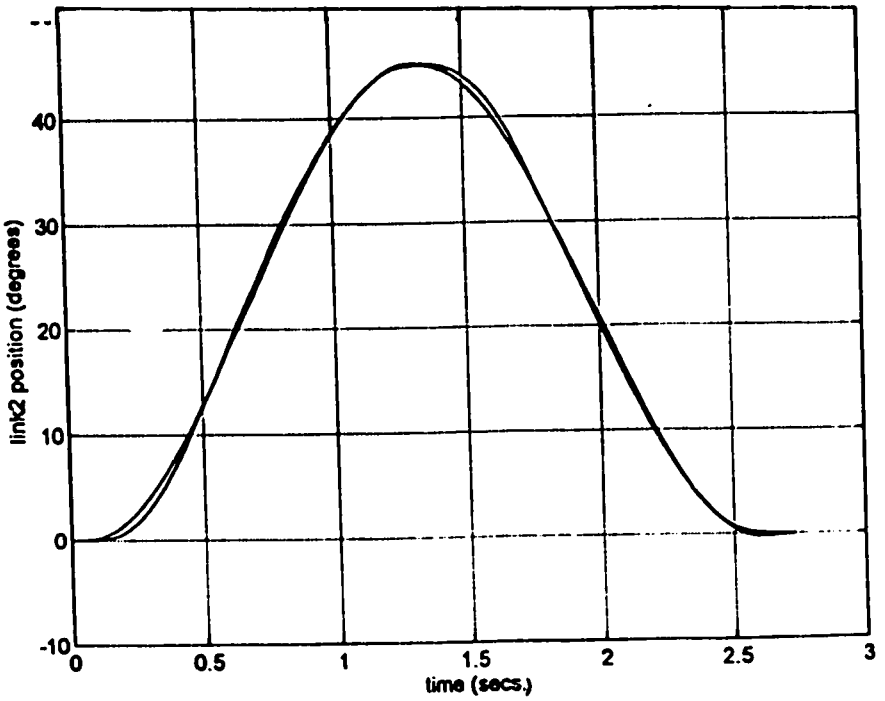

Figure 6.5 Time History of the Joint Angle 2

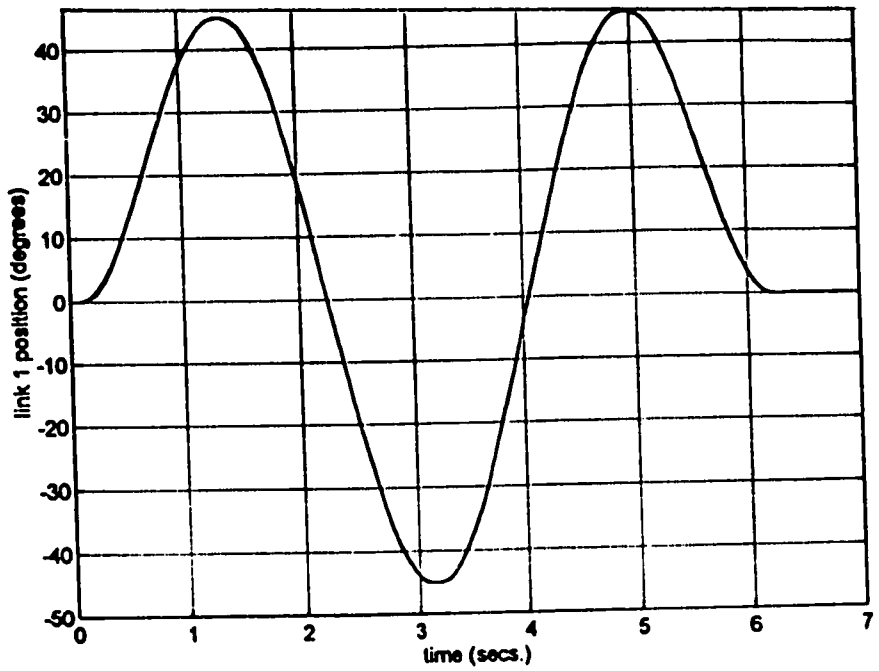

Figure 6.6 Time History of the Joint Angle 1

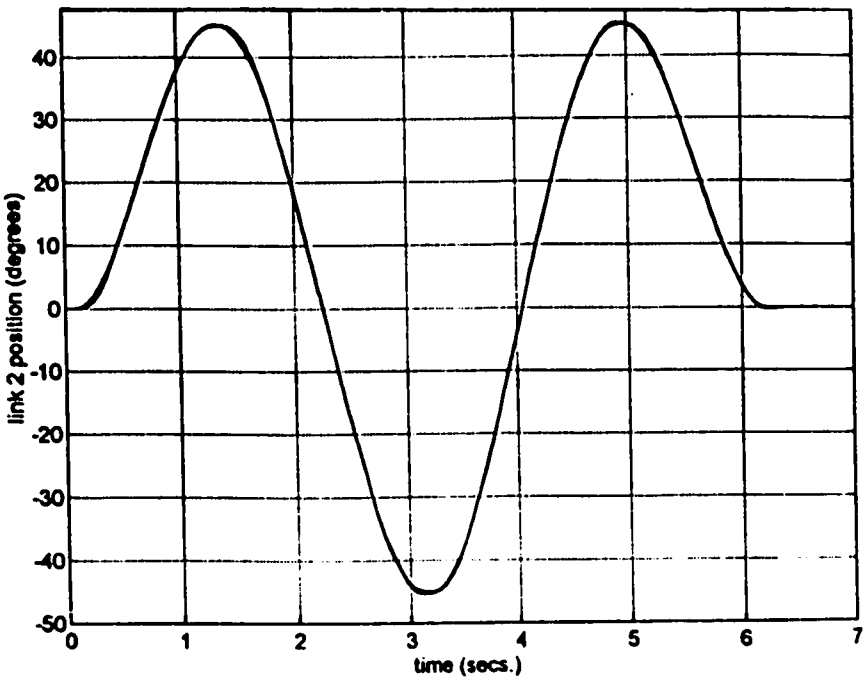

Figure 6.6 Time History of the Joint Angle 2

6.3 Results on a Single Link manipulator

Experiments were performed on a single motor fitted with a multi-degree of freedom gripper. Experimental set up is shown in Fig. 6.8.

Figure 6.8 Experimental Set up of a Single-link Manipulator

6.3.1 Experimental Results

It was hypothesized that the neural network would produce the largest part of the control input with linear compensator generating small corrections. This scheme proved to be effective. However, the possibility of the linear compensator alone being able to correct the changes in the operating conditions raised some questions about the necessity of having an artificial network as the main controller. To explore the limits of the linear compensator, the neural network was removed from the system and the manipulator was controller only with PD controller. The motor was moved from 0 to 90 degrees. The task was completed in 1 second. Experiments were performed using a PD controller and neurocontroller. The results are shown in figures 6.9 and 6.10. Fig 6.9 shows the angular position with PD controller and Fig. 6.10 shows the results using neurocontroller-based scheme. It is observed that neurocontroller gives smooth results as compared to PD controller. There are fewer oscillations as compared to PD controller. This shows that neural network-based scheme can handle unmodelled factors better than PD controller. Another experiment was performed to see the effect of payload variations. The variation may arise from the fact that the standard articles handled by the robot may have slightly different masses due to the variations in the manufacturing process. Thus, it is important to determine how the controller deals with these uncertainties. As expected, neural network controller has better robustness and is less sensitive to the payload variations. Figs. 6.11 and 6.12 show the results with the payload variation. It is observed that the neurocontrol is robust to dynamic or load change of the robot. The performance of the robot which uses conventional controller (PD controller)

deteriorates greatly when subject to such a load change. Another experiment was carried out to see the effect of doing the same task faster, i.e., the dynamics of the manipulator was changing fast. In this case also the proposed controller has better robustness (Figs. 6.13 and 6.14).

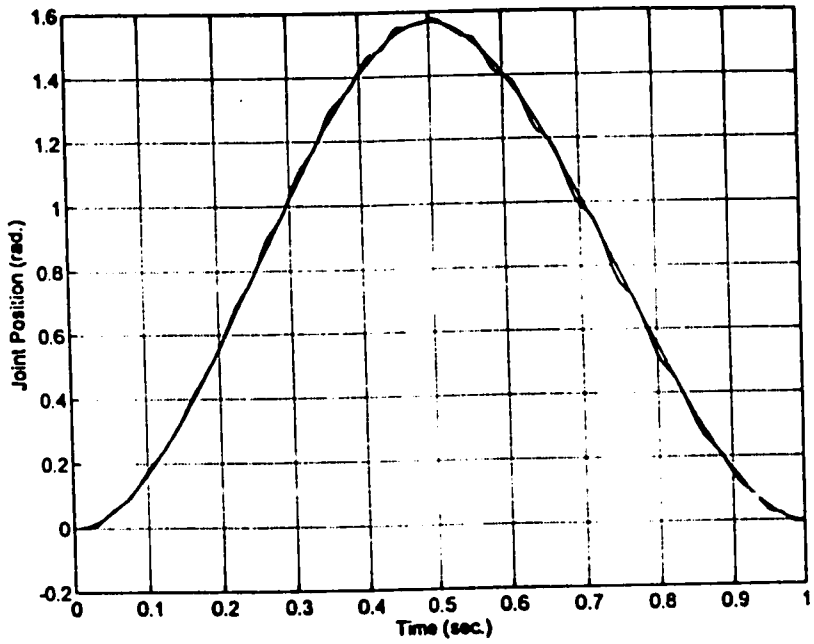

Figure 6.9 Time History of the Joint Angle with PD Controller

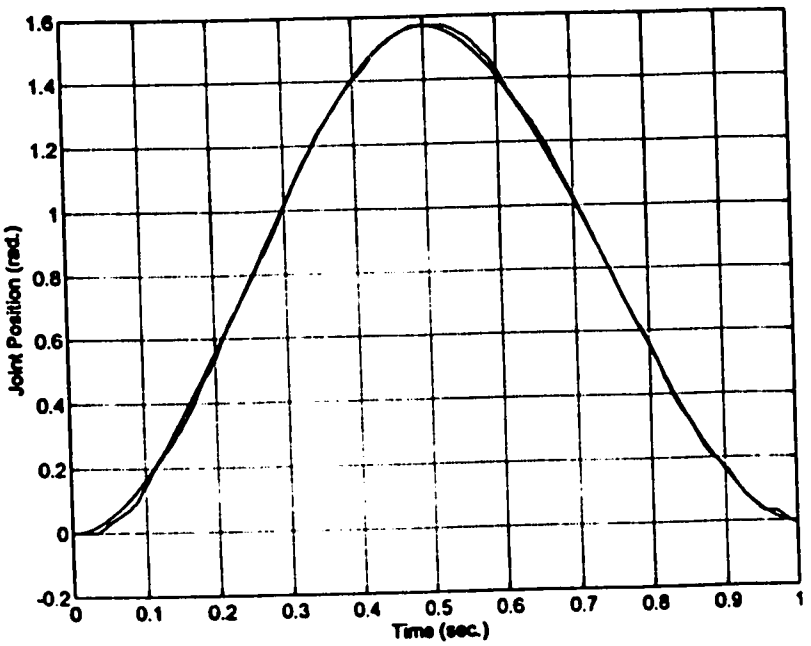

Figure 6.10 Time History of the Joint Angle

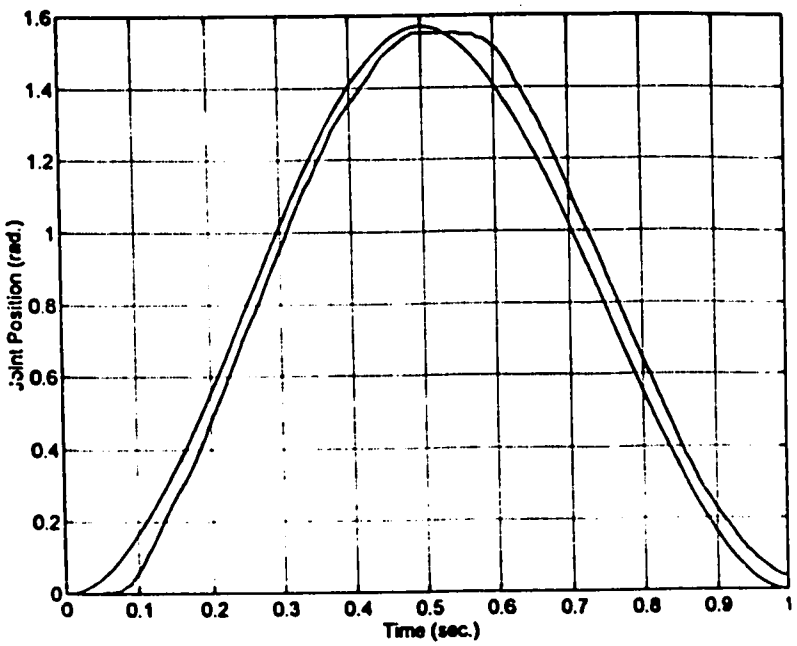

Figure 6.11 Time History of the Joint Angle with PD Controller with Payload

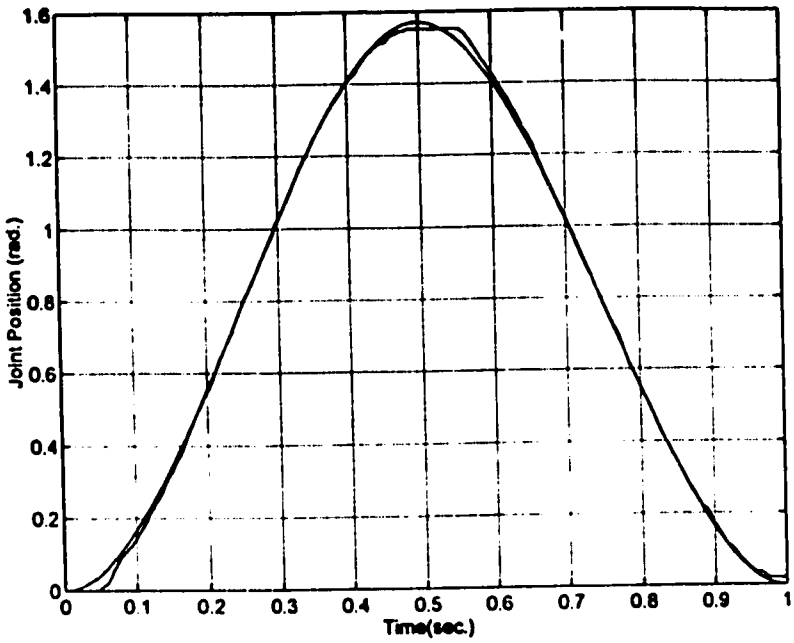

Figure 6.12 Time History of the Joint Angle with Payload

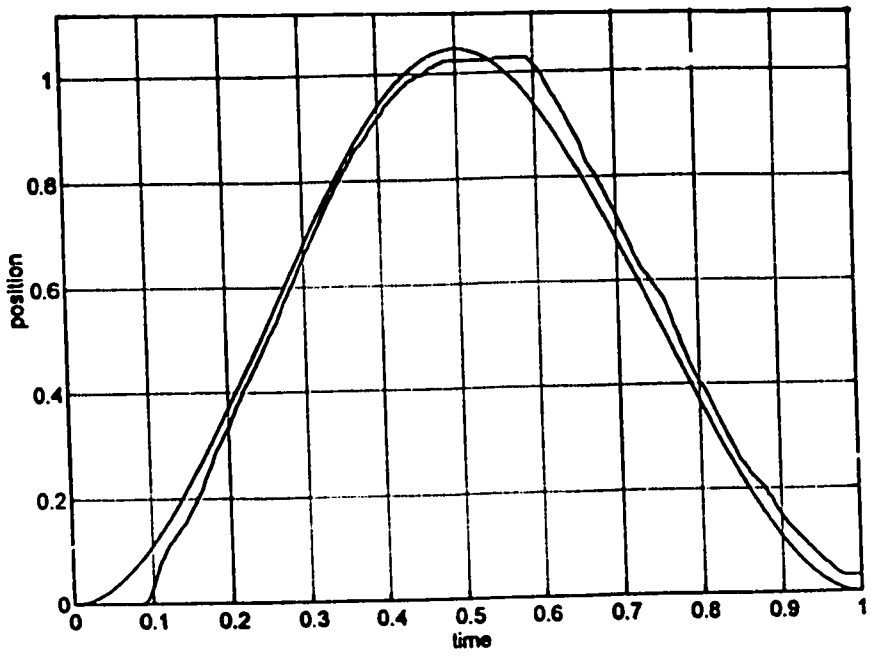

Figure 6.13 Time History of the Joint Angle with PD controller

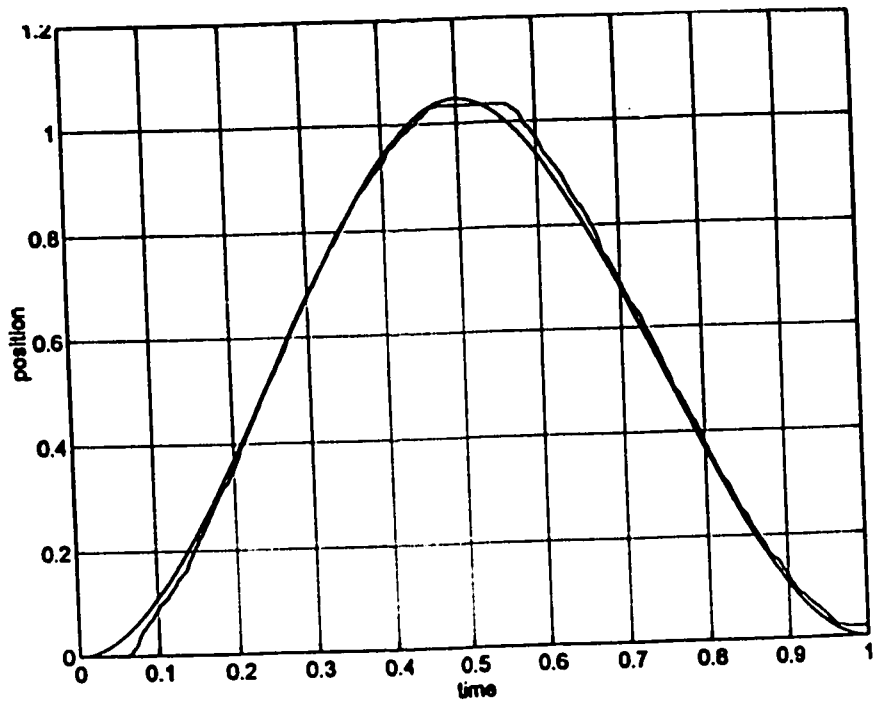

Figure 6.14 Time History of the Joint Angle

6.4 CONCLUSIONS

Compared with the approaches based on model of the system, a neurocontrol scheme requires the least a priori information about the system dynamics. A neurocontrol is able to improve its performance and approach the best results. If there are uncertainties and/or time varying factors (which are difficult to model) conventional schemes usually deteriorate. The results presented clearly indicate that the learning controller converges to a low error. This observation is consistent with the results of previous simulations. While good performance was generally possible with the carefully adjusted fixed-gain controller, control system performance without learning was highly sensitive to the changes in the operating conditions. In contrast, control system performance with learning was relatively insensitive to parameter

selection, resulting in control errors lower than or comparable to the fixed-gain controller, even for severe changes in the parameters. The learning controller presented here is well suited for practical application to the control of industrial robotic manipulators. The learning algorithm structure is simple and can be easily extended to any manipulator. This makes adaptation of the control software to accommodate system changes unnecessary or relatively easy, and makes it possible to transport large portions of the control software from one robot to another. In summary

- Robotic manipulators can be controlled by learning.
- The control strategy developed is tested experimentally. A reasonable agreement can be found between theory and simulations.
- PD controller alone does not provide proper control.
- The combination of neural network and PD controller proved to be very valuable in controlling the motion of the manipulator. It is seen that corrections provided by this scheme resulted in exact reproduction of the desired motion. The errors resulting from the use of the combination of the neural network and PD controller were very acceptable.
- Since the dynamics of the manipulator itself is used for learning, the dynamics of the system Can be identified accurately to improve the control performance.
- Generalization was shown through the reproduction of a case that was absent from the training set.

Through experimental testing, it was found that the neurocontrol approach can be

used to control a system without detailed modelling and parameter identification of the system dynamics.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER WORK

Almost all real world systems are nonlinear. It is often quite difficult, if not impossible to obtain an accurate dynamic model of such systems. Robotic manipulators belong to the class of highly coupled nonlinear dynamic systems, and linear control approaches usually produce poor control performance. There are only a few nonlinear control design approaches available to date and most of these can only be applied to feedback linearizable nonlinear systems. The neurocontrol approach opens a new area that may present general solutions to nonlinear control systems. In this thesis, effort has been made toward obtaining the robust control of robotic manipulators. The major emphasis has been on studying the effects of changing conditions of operation of manipulators and the introduction of new combination of backpropagation learning algorithm. The work is significant to the development of advanced industrial robots, where high speed and high control precision are required.

7.1 SUMMARY OF CONTRIBUTIONS

In chapter 3, modification of the conventional backpropagation learning algorithm has been introduced. The proposed delta-bar-delta rule with adaptive gain of the node significantly improves the convergence behavior when compared with the conventional

back-propagation, making the proposed algorithm less computationally intensive. The proposed updating laws, involving the adaptive selection of the step size η and adaptive gain γ, have been shown to learn much more efficiently the dynamics of a nonlinear system. This benefit is tremendous for complex high order nonlinear systems, especially real-time control applications.

Model learning scheme is introduced in chapter 4. The elementary training of the neural network using an obtained dynamic model can be fulfilled off-line. After the model learning is finished, the neural network learns structured/unstructured uncertainties on-line. The learning procedure is effective and efficient in learning the manipulator dynamics, and the error convergence rate with an untrained trajectory is fast.

The feasibility of developing a controller based on neural networks for nonlinear dynamical plants has been demonstrated in chapter 5. A control architecture based on the combination of a neural network and a linear compensator has been presented. It has been hypothesized that a neural network can be trained to produce the large nonlinear portion of the required system control input. A linear compensator (PD controller) can be used to provide the necessary corrections to these control inputs. The combination of the neural network and linear compensator proved to be very valuable in controlling the manipulator. In most cases, corrections provided by this scheme resulted in exact reproduction of the desired results. In most cases, the errors resulting from the use of the combination of the neural network and the linear compensator were very acceptable. Generalization was shown through reproduction of a case that was absent from the training set. A case studies have

demonstrated that the proposed scheme is very promising, which achieved satisfactory performance. The results permit reaching goals with arbitrarily low error even though the inverse dynamics, which was learned by the neural network is only an approximation. The inverse model with neural network structure found to possess the good performance. After the neural network has compensated perfectly or partially for the nonlinearity of the controlled object through learning, the responses of the controlled object follow the desired responses supplied by inverse reference model implemented in the conventional feedback controller. The advantage of the proposed controller is that there is no need of on-line training as the neural network is trained off-line. Thus, this avoids the problem of heavy on-line computation required by adaptive controllers and robust controllers. Furthermore, the proposed controller has the potential to handle the effect of changes in the payload and the variation in the operating conditions. As a result, the adaptive capabilities of the neural network controller to the unstructured effects are clarified. The neural controller has been designed in such a way that it can be extended to an arbitrarily large number of joints. Each new joint in the robot system will require adding new neural input and weight maps to the architecture. The number of neurons in each neural weight map is arbitrary and increasing the number should increase the resolution of performance. Furthermore, the neural controller is a parallel mechanism which can be highly optimized with parallel hardware. It is expected that an optimized hardware will result in dynamic control which is faster than the mechanical reaction time of the robot for an arbitrary number of joints.

It is well known that most industrial robots perform repetitive tasks. The drawbacks

are that the tracking errors are also repeated. An intelligent robot should have the capability to learn from previous operations and improve performance by itself. The research in this thesis ensures that a robot has this desirable feature.

Through experimental testing presented in chapter 6, it was found that the neurocontrol approach can be used to control a system without detailed modeling and parameter identification of the system dynamics. Compared with the approaches based on model of the system, a neurocontrol scheme requires the least a priori information about the system dynamics. A neurocontrol is able to improve its performance and approach the best results. If there are uncertainties and/or time varying factors (which are difficult to model) conventional schemes usually deteriorate. In summary

- PD controller alone doesn't provide proper control.
- The combination of neural network and PD controller proved to be very valuable in controlling the motion of the manipulator. It is seen that corrections provided by this scheme resulted in exact reproduction of the desired motion. The errors resulting from the use of the combination of the neural network and PD controller were very acceptable.
- The present method requires neither an accurate model nor parameter estimation. It possesses a great ability to generalize.
- After the neural network has compensated perfectly or partially for the nonlinearity of the controlled object through learning, the responses of the controlled object follow the desired responses supplied by inverse reference model implemented in the

conventional feedback controller.

- The control strategy developed is tested experimentally. A reasonable agreement can be found between theory and simulations.
- Generalization is shown through the reproduction of a case that was absent from the training set.
- Neurocontroller is generalized to accurately move an unforeseen payload to arbitrary targets without endpoint oscillations.
- The present model can be easily implemented in a parallel distributed processing machine, since both the nonlinear transformations in subsystems and the synaptic modifications are essentially parallel.
- Neurocontroller reduces the computational power, the robot calibration time, maintenance cost and engineering time when developing controllers of new robots by virtue of its firmly established emergent generalization, fault tolerant and self organization properties.

The author acknowledges that neural network is an exploding research area and advances very quickly. It is not surprising that the approaches proposed here may not be the best with the latest results. However, the neural network employed in this research work is simple yet quite effective. Also, emphasis of this work is laid on the application of neural networks to control of a robot, rather than on research in the theory of neural networks. A neural network here provides an alternative approach to the existing controllers of robots.

While the focus of this research was the dynamic control of industrial manipulators,

the technique described is applicable to a wide range of robotic control problems which will be increasingly important in the future. For example, the use of low-mass materials in the construction of robots, for applications in space, will almost certainly require the use of high-performance learning controllers, since the control characteristics will be highly payload/task dependent.

The research outlined in this thesis should not be taken as the final word in the application of neural networks to robotics. Instead much of this work should be considered as proof of principle that a neural network can solve some interesting problem in robotics.

Generally, this work has fulfilled the author's originally proposed research plans and achieved satisfactory results. The answer to some important, yet unresolved problems are given in this thesis, but not all. Here, the author wants to point out some unresolved remaining issues and give suggestions for the further research.

7.2 SUGGESTIONS FOR FURTHER RESEARCH

Most of the work done so far can be considered to be experimental in nature. It lacks a firm theoretical foundation. The convergence of a neural network is still an open problem.

Although the author's modification of the back-propagation algorithm is quite effective, yet a strict mathematical proof is not given. Also, some partial results are reported recently, but they provide little help for practical training of the neural networks. In the author's view much research is needed before the potential of neural networks is fully utilized.

Most of the results in this thesis apply to deterministic dynamic systems. Further study should consider stochastic dynamic systems, since there is always measurement noise in the

feedback signals.

The various neural network approaches need to be compared to conventional approaches and it needs to be determined if neural networks provide any cost and/or performance benefits.

As more complex problems are attempted to be solved using neural networks, a better understanding of the network's function is necessary. A relationship needs to be established between the system under study (i.e., the nonlinear function to be approximated) and the structure of a suitable network.

These problems can be attributed to the fact that the neural network field is in its developing stages and has not been understood fully. It is expected that most of the above-mentioned problems would disappear once more is known about how the neural networks work.

In this thesis, most of the work carried out relates to the trajectory control of robotic manipulators. Further extensions to force control and constrained motion control are another interesting topic for research and can be investigated.

Due to the powerful mapping ability of neural networks, the proposed neural network-based control scheme can be enhanced with neural networks performing computer vision-to-task space position and orientation mapping and collision avoidance tasks.

BIBLIOGRAPHY

[1] Albus, J.S., Data Storage in the Cerebellar Model Articulation Controller (CMAC), *Transactions of ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 97, no. 3, pp. 228-233, 1975a.

[2] Albus, J.S., A New Approach to Manipulator Control: The Cerebellar Model Controller (CMAC), *Transactions of ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 97, no. 3, pp. 220-227, 1975b.

[3] Annaswamy, A.M. and Yu, S.H. Yu, θ - Adaptive Neural Networks: A New Approach to parameter Estimation, *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 907-918, 1996.

[4] Anderson, J.A. and Rosenfeld, E., Neurocomputing: Foundations of Research, Cambridge, MA: MIT Press, 1988.

[5] Arimoto, S., Kawamura, S., and Miyazaki, F., Bettering Operation of Robots by Learning, *Journal of Robotic Systems*, pp. 123-140, 1984

[6] Asada, H. and Liu, S., Transfer of Human Skills to Neural Net Robot, *Proc. IEEE Int'l Conf. on Robotics and Automation*, Sacramento, CA, pp. 2442-2448, 1991

[7] Atkeson, C.G. and McIntyre, J., Robot Trajectory Learning Through Practice,

Proceedings of the IEEE Conference on Robotics and Automation, San Francisco, 1986.

[8] Atkeson, C.G. and Reinkensmeyer, D.J., Using Associative Content-Addressable Memories to Control Robots, *Proceeding of the IEEE Conference on Decision and Control*, pp. 792-797, 1988.

[9] Back, A.D., and Tsoi, A.C., FIR and IIR Synapses, a new neural network architecture for time series modelling, *Neural Computation*, vol. 3, pp. 375-385, 1991.

[10] Baldi, P.F. and Hornik, K., Learning in Linear Neural Networks - A Survey, *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 837-858, 1995.

[11] Barron, A.R., Neural net approximation, *Proc. of the 7th Yale Workshop on Adaptive and Learning Systems*, New Haven, CT: Yale University, pp. 69-72, 1992.

[12] Barto, A.G., Sutton, R.S. and Anderson, W.Ch., Neuron like Adaptive Element That can Solve Difficult Learning Control Problems. *IEEE Transactions on System, Man, and Cybernetics*, SMC-13, pp. 834-846, 1983.

[13] Billings, S. A., Jamaluddin, H.B. and Chen, S., Properties of neural networks with applications to modelling nonlinear dynamical systems. *International Journal of Control*, vol. 1, pp. 193-224, 1992.

[14] Bolt, G.R., Fault Tolerance in Artificial Neural Networks, D. Phil. Thesis, York

University, Ontario, 1992.

[15] Box, George, E.P and Jenkins, Gwilym M., *Time Series Analysis: Forecasting and Control*, San Francisco, Holden-Day Inc., 1970.

[16] Carrara, A.R.D.S., Dynamics, Control and Simulation of Flexible Robotic Systems, Ph.D. Thesis, McMaster University, 1995.

[17] Churchland, P.S. and Sejnowski, T.J., *The Computational Brain*, Cambridge, MA: MIT Press, 1992.

[18] Chen, S. And Billings, S.A., Representation of Non-linear Systems: the NARMAX Model, *International Journal of Control*, pp. 1013-1032, 1989a.

[19] Chen, S. And Billings, S.A.,, Modeling and Analysis of Non-linear Time Series, *International Journal of Control*, vol. 50, pp. 2151-2171, 1989b.

[20] Craig, J.J., *Introduction to Robotics: Mechanics and Control*, Reading, MA: Addison-Wesley Publishing Company, 1986.

[21] Craig, J.J., *Adaptive Control of Mechanical Manipulators*, Reading, MA: Addison-Wesley Publishing Company, 1988.

[22] Cybenko, G., Approximations by superpositions of a sigmoidal Function. *Math.Control Signal* Systems, vol. 2, pp. 303 - 314, 1989.

[23] Elsley, R., A learning architecture for control based on back-propagation neural networks, *IEEE Conference on Neural Networks*, vol. 2, pp. 584-587, 1988.

[24] Foslien, W., Konar, A.F. and Samad, T., Optimization with neural memory for

process parameter estimation, *Applications of Artificial Neural Networks III*, S.K. Rogers (Ed.), Proc. SPIE 1709.

[25] Fu, K.S., Ganzalez, R.C., and Lee, C.S.G., *Robotics: Control, Sensing, Vision, Intelligence*, New York, McGraw-Hill Inc., 1987.

[26] Fukuta, T. and Shibata, T., Adaptation and Learning for Hierarchical Intelligent Control, *Proc. IJCNN'91*, Baltimore, MD, pp. 269-274, 1991.

[27] Funahasi, K., On the approximate realisation of continuous mappings by neural networks, *Neural Networks*, vol. 2, pp. 183-192, 1989.

[28] Guez, A. and Ahmad, Z., Accelerated Convergence in the Inverse Kinematics Via Multilayer Feedforward Networks, *IEEE International Joint Conference on Neural Networks, Washington, D.C.*,pp. 341-344, 1989.

[29] Guez, A., Elibert, J.L. and Kam, M., Neural Network architecture for Control. *IEEE Control System Magazine*, vol. 8, no. 3, pp. 22-25, 1988.

[30] Guez, A. and Selinsky, J.W., A Neurocontroller with Guaranteed Performance for Rigid Robots, *IEEE Int'l Joint Conf. on Neural Networks*, Washington, DC, pp. 511-514, 1990.

[31] Gomi, H. and Kawato, M., Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning. *Neural Networks*, vol. 6, pp. 933-946, 1993.

[32] Gupta, Madan M. and Sinha, Naresh K. (Eds.), *Intelligent Control Systems: Theory and Applications*, IEEE Press, 1996.

[33] Gupta, Pramod and Sinha, Naresh K.,Control of Robotic Manipulators Using Neural Networks - A Survey, in *Methods and Applications of Intelligent Control*, S. Tzafestas (Ed.), Kluwer Academic Publishers, The Netherlands, pp. 103-133, 1997.

[34] Gupta, Pramod and Sinha, Naresh K., Intelligent Control of Robotic Manipulator- A Neural Network Approach, accepted for publication in *International Journal of Systems Science*, 1997.

[35] Gupta Pramod, Sinha, Naresh K., Elbestawi, M.A. and Bone, G., Intelligent Control of Industrial Robots For Manufacturing Applications, IPMM'97, Australasia-Pacific forum on Intelligent Processing and Manufacturing of Materials, Gold Coast, Australia, July, 1997.

[36] Gupta, Pramod and Sinha, Naresh K., Modeling Robot Dynamics Using Dynamic Neural Networks, *SYSID '97 11th IFAC Symposium on System Identification*, Fukuoka, Japan, pp. 783-788, July, 1997, (Invited Paper).

[37] Gupta, Pramod and Sinha, Naresh K., Intelligent Control of Robotic Manipulator Using Neural Networks, *IFAC SYSID '97 11th IFAC Symposium on System Identification*, Fukuoka, Japan, pp. 889-894, July, 1997, (Invited Paper).

[38] Gupta, Pramod, Sinha, Naresh K. and Elbestawi, Mohammed A., Identification of Industrial Robots Using Neural Networks, in *Proceedings of Canadian Society for Mechanical Engineering (CSME)* Forum, Hamilton (Canada), May 7-9, 1996, pp. 275-278.

[39] Gupta, Pramod and Sinha, Naresh K., A New Algorithm for Efficient Identification of

Nonlinear Systems Using Neural Networks, in *Proceedings of TIMA-96, International Conference on Trends in Industrial Measurements and Automation*, Madras (India),pp. 195-199, Jan. 96.

[40] Gupta, Pramod and Sinha, Naresh K., Fast Identification of Nonlinear Systems Using Neural Networks, in *Proceedings of International Conference on Automation '95,ICAUTO'95*, held in India Dec. 95, pp. 141-144.

[41] Guo, J. and Cherkassky, V., A Solution to the inverse Kinematic Problem in Robotics Using Neural Network Processing, *IEEE International Joint Conference on Neural Networks, Washington, D.C.*, PP 299-304, 1989.

[42] Handelman, D.A., Lane, S.H. and Gelfand, J.J., Integration of Knowledge-Based System and Neural Network Techniques for Automation Learning Machines, *Proc. IJCNN'89*, Washington, D.C., pp. 683-688, 1989

[43] Haykin, S., *Neural Networks, A Comprehensive Foundation*, Toronto, Maxwell Macmillan, 1994.

[44] Hecht-Nielsen, R., Neurocomputer applications, In R. Eckmiller and Ch. V. D. Malsburg (Eds.), *Neural Computers*, Springer-Verlag, Berlin, pp. 445-453, 1988.

[45] Hopfield, J.J., Neural Networks and Physical Systems with Emergent Collective Computation Abilities, *Proceedings of the National Academy of Science USA*, pp. 2554-2558, 1982.

[46] Hopfield, J.J., Neurons with Graded Response have Collective Computational

Properties like Those of Two-State Neurons, *Proceedings of the National Academy of Science USA*, pp. 3088-3092, 1984.

[47] Hornik, K., Stinchombe, M. and White, H., Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, vol. 2, pp. 359-366, 1989.

[48] Hunt, K.J., Sbarbaro, D., Zbikowski, R. and Gawthrop, P.J., Neural Networks for Control Systems - A Survey, *Automatica*, vol. 28, pp. 1083-1112, 1992.

[49] IEEE, Special Issue on Neural Networks, *IEEE Control System Magazine*, vol. 8, no. 2, 1988.

[50] IEEE, Special Issue on Neural Networks, *IEEE Control System Magazine*, vol. 9, no. 3, 1989.

[51] IEEE, Special Issue on Neural Networks, *IEEE Control System Magazine*, vol. 10, no. 3, 1990.

[52] Irwin, G.W., Warwick, K. and Hunt, K.J. (editors), *Neural Network Applications in Control*, IEE Control Engineering Series, London, 1995.

[53] Jacobs, R.A. and Jordan, M.I., Learning Piecewise Control Strategies in a Modular Neural Network Architecture, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, pp. 337-345, 1993.

[54] Jagannathan, S. and Lewis, F.L.,Multilayer Discrete-Time Neural Net Controller With Guaranteed Perfromance, *IEEE Transactions on Neural Networks*, vol. 7, no.1, pp. 107-130, 1996

[55] Jin, Y., Pipe, T. and Winfield, A., Stable Neural Network Control for Manipulator, *Proc. IJCNN*, pp. 2775-2778, 1993.

[56] Jin, L. and Gupta, M.M., Globally Asymptotical Stability of Discrete-Time Analog neural Networks, *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 1024-1031, 1996

[57] Johnson, M.A. and Leahy, M.B., Adaptive Model-Based Neural Network Control, *Proc. IEEE Int'l Conference on Robotics and Automation*, Cincinnati, OH, pp. 1704-1709, 1990.

[58] Jordan, M. I., Supervised Learning and Systems with Excess Degrees of Freedom. Technical Reports 88-27, University of Massachusetts, Amherst, MA, 1988.

[59] Jordan, M. I. and Rumelhart, D.E., Forward Models: Supervised Learning with a Distal Teacher, *Cognitive Science*, vol. 16, pp. 307-354, 1992.

[60] Jacobs, R.A. and Jordan, M.I., Learning Piecewise Control Strategies in a Modular Neural Network Architecture, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 337-345, 1993.

[61] Karakaşoğlu, A., Sudharsanan, S.I. and Sudharsanan, M., Identification and Decentralized Adaptive Control Using Dynamical Neural Networks with Application to Robotic Manipulators, *IEEE Transactions on Neural Networks*, vol. 4, pp. 919-930, 1993.

[62] Kawato, M., Uno, Y., Isobe, M. and Suzuki, R.,Hierarchical Neural Network

Model for Voluntary Movement with Applications to Robotics, *IEEE Control System Magazine*, vol. 8, pp. 8-16, 1988.

[63] Khosla, P.K., Estimation of Robot Dynamic Parameters: Theory and Application, *International Journal of Robotics and Automation*, vol. 3, pp. 35-41, 1988

[64] Kohonen, T., *Self-Organization and Associative Memory*, Berlin, Springer-Verlag, 1987.

[65] Konar, A.F., Samad, T. and Foslien, W., Hybrid Neural Network/Algorithmic Approaches to System Identification, *Preprints of the IFAC Symposium on DYCORD 92*, College Park, MD, 1992.

[66] Kosmatopoulos, E.B., Polycorpou, M.M., Christodoulo, M.A. and Ioannou, P.A., High-Order Neural Network Structures for Identification of Dynamical Systems, *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 422-431, 1995.

[67] Kraft, L.G. and Campagna, D.P., A Comparison Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems, *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 36-43, 1990.

[68] Kuperstein, M. and Wang, J., Neural Controller for Adaptive Movements with unforseen payloads, *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 137-142, 1990.

[69] Lapedes, A. and Farber, R., Nonlinear signal processing using neural networks: prediction and system modeling, Report LA-UR-87-2662, Los Alamos National

Laboratory, 1987.

[70] Leahy, M.B., Johnson, M.A. and Rogers, S.K., Neural Network Payload Estimation for Adaptive Robot Control, *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 93-100, 1991.

[71] Lee, S. and Kil, Rhee M., Redundant Arm Kinematic Control with Recurrent Loop, *Neural Networks*, vol. 7, pp. 643-659, 1994.

[72] Leontaritis, I.J. and Billings, S.A., Input-Output Parametric Models for Nonlinear Systems, *International Journal of Control*, vol. 41, pp. 303-344, 1985.

[73] Levin, A.U. and Narendra, K.S., Control of Nonlinear Dynamical Systems Using Neural Networks - Part II: Observability, Identification and Control, *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 30-42, 1996.

[74] Levin, A.U. and Narendra, K.S., Recursive Identification Using Feedforward Neural Networks, *International Journal of Control*, vol. 61, pp. 533-547, 1995.

[75] Lewis, F.L., Liu, K. and Yeşildirek, A., Neural Net Robot Controller with Guaranteed Tracking Performance, *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 703-715, 1995.

[76] Lewis, F.L., Yeşildirek, A. and Liu, K.,Multilayer Neural Net Robot Controller with Guaranteed Tracking Performance, *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 388-399, 1996.

[77] Li, C. James and Kim, T., A New Feedforward Neural Network Structural Learning

Algorithm - Augmentation by Training with Residuals, *ASME Transactions on Dynamic Systems, Measurement and Control*, vol. 117, no. 3, pp. 411-414, 1995.

[78] Liu, C.J. and Lin, C.T., Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network, *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 709-731, 1996.

[79] Liu, C.S. and Kim, H., Selection of Learning Parameters for CMAC-Based Aadptive Critic Learning, *IEEE Transactions on Neural Networks*, vol. 6, no. 4, pp. 642-647, 1995.

[80] Miller, W.T., Glanz, F.H. and Kraft, L.G., Application of a General Learning Algorithm to the Control of Robotic Manipulator, *International Journal of Robotic Research*, vol. 6, no. 2, pp. 84-98, 1987.

[81] Miller, W.T., Hewes, Robert P., Glanz, F.H. and Kraft, L.G., Real-Time Dynamic Control of an Industrial Manipulator Using a Neural-Network-Based Learning Controller, *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp.1-9, 1990.

[82] Miller, W.T., Glanz, F.H. and Kraft, L.G., CMAC: An Associative Neural Network Alternative to Backpropagation, *IEEE Proceedings*, vol. 78, pp. 1561-1567, 1990.

[83] Miller, W.T., Sutton, R.S. and Werbos, P.J. (Eds.), *Neural Networks for Control*, Cambridge, MA: MIT Press,1990.

[84] Mistry, S.I., Chang, S.L. and Nair, S.S., Indirect Control of a Class of Nonlinear Dynamic Systems, *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 1015-1023, 1996.

[85] Miyamoto, H., Kawato, M, Setoyama, T. and Suzuki, R., Feedaback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator, *Neural Networks*, vol. 1, pp. 251-265, 1988.

[86] Moody, J.O. and Antsaklis, P.J., The Dependence Identification Neural Network Construction Algorithm, *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 3-15, 1996.

[87] Mukhopsdhyay, S. and Narendra, K.S., Intelligent Control Using Neural Networks, In Madan M. Gupta and Naresh K. Sinha (Eds.) *Intelligent Control Systems: Theory and Applications,*, IEEE Press, 1995.

[88] Narendra, K.S. and Parthasarthy, K., Identification and Control of dynamical Systems using Neural Networks, *IEEE Transactions on Neural Networks*, vol. 1, pp. 4-27, 1990.

[89] Narendra, K.S. and Annaswamy, A.M., Stable Adaptive Systems, Prentice-Hall, Englewood, NJ, 1989.

[90] Nguyen, D.H. and Widrow, B., Neural Networks for Self-Learning Control Systems, *IEEE Control System Magazine*, vol. 10, no. 3, pp. 18-23, 1990.

[91] Nordgren, R.E. and Meckl, Peter H., An Analytical Comparison of a Neural

Network and a Model-Based Adaptive Controller, *IEEE Transactions on Neural Networks*, vol. 4, pp. 685-694, 1993.

[92] Nowlan, Steven J., Gain Variation in Recurrent Error Propagation Networks, *Complex Systems*, vol. 2, pp. 305-320, 1988.

[93] Psaltis, D., Sideris, A., and Yamamura, A.A., Neural Controllers, *Proc. IEEE Int'l Conference on Neural Networks*, San Diego, CA, vol. 4, pp. 551- 558, 1987.

[94] Psaltis, D., Sideris, A. and Yamamura, A.A., A Multilayered Neural Network Controller, *IEEE Control System Magazine*, vol. 8, pp. 17-21, 1988.

[95] Rabelo, L.C. and Avula Xavier, J.R., Hierarchical Neurocontroller Architecture for Robotic Manipulation, *Proc. IEEE Int'l Conf. on Robotics and Automation*, Sacramento, CA, , pp. 2656-2661, 1991.

[96] Raibert, M.H., Analytical equations vs. Table look-up for Manipulation: A unifying Concept, *Proceedings of the IEEE Conference on Decision and Control*, pp. 576-579, 1977.

[97] Raibert, M.H., A Model for Sensorimotor Control and Learning, *Biological Cybernetics*, vol. 29, pp. 29-36, 1978.

[98] Rosenblatt, F., The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review*, vol. 65, pp. 386-408, 1958.

[99] Rumelhart, McClelland and the PDP Research Group, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, vol. 1: Foundations*,

Cambridge MA, MIT Press, 1986.

[100]Sanner, R.M. and Slotine, J.-J.E., Stable Adaptive Control of Robot Manipulators Using Neural Networks, *Neural Computation*, vol. 7, no. 4, pp. 753-790, 1995.

[101] Sekiguchi, M., Sugasaka, T. and Nagata, S., Control of a Multivariable System by Neural Network, *Proceedings IEEE Int'l Conference on Robotics and Automation, Sacramento, CA*, pp. 2644-2649, 1991.

[102] Sinha, N.K. and Kuszta, B., *Modeling and Identifictaion of Dynamic Systems*, New York, Van Nostrand Reinhold Company Inc., 1983.

[103] Slotine, J.-J.E. and Li, W., Adaptive Manipulator Control: A Case Study, *IEEE Transactions on Automatic Control*, vol. 33, no. 11, pp. 995-1003, 1988.

[104] Spong, M. and Vidyasagar, M., *Robot Dynamics and Control*, New York, John Wiley & Sons, 1989.

[105] Tanak, K., An Approach to Stability Criteria of Neural Network Control Systems, *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 629-642, 1996.

[106] Tank, D. and Hopfield, J., Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.

[107] Tsoi, Ah. C. and Back, A.D., Locally Recurrent Globally Feedforward Networks: A critical Review of Architectures, *IEEE Trans. on Neural Networks*, vol. 5, pp. 229-239, 1994.

[108] Waibel, A., Hanazawa, T., Hinton, G.S., Shikano, G. and Lang, K., Phonemic

recognition using time delay neural networks, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 328-339, 1989.

[109] Wang, Q. and Broome, D.R., A Novel Neural Adaptive Controller for Robots, *Proceedings IEE Int'l Conference on Control'94*, Coventry, UK, pp. 486-491, 1994.

[110] Werbos, P.J., Neurocontrol and Supervised Learning: An Overview and Evaluation, In D.A. White and D.A. Sofge (Eds.), *Handbook of Intelligent Control, Neural, Fuzzy, and Adaptive approaches*, Van Nostrand Reinhold, pp. 65-89, 1992.

[111] Werbos, P.J., Overview of Designs and Capabilities, In W.T. Miller, R.S. Sutton and P.J. Werbos (Eds.), *Neural Networks for Control*, Cambridge MA, MIT Press, pp. 59-65, 1990.

[112] Werbos, P.J., Neurocontrol and Related Techniques, A. Maren (Ed.), In *Handbook of Neural Computing Applications*, New York: Academic Press, 1990.

[113] White, D.A. and Sofge, D.A. (Eds.), Handbook of Intelligent Control, Neural Fuzzy, and Adaptive approaches, Van Nostrand Reinhold, 1992.

[114] Widrow, B. and Lehr, M.A., 30 Years of Adaptive Neural Networks: Perceptron, Medaline, and Backpropagation, *Proc. of the IEEE*, vol. 78, 1415-1441, 1990.

[115] Williams, R.J. and Zipser, D., A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, vol. 1, pp. 270-280, 1989.

[116] Yabuta, T. and Yamada, T., Possibility of Neural Networks Controller for Robot

Manipulators, *Proceedings of International Conference on Robotics and Automation*, Cincinnati, OH, pp. 1686-1691, 1990.

[117] Yamada, T. and Yabuta, T., Dynamic System Identification Using Neural Networks, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 204-211, 1993.

[118] Yegerlehner, James D., Experimental Implementation of Neural Network Controller for Robot Undergoing Large Payload Changes, *Proceedings of IEEE International Conference on Robotics and Automation, Atlanta, GA, USA*, pp. 744-749, 1993.