Neural Network Controllers for Robot Manipulators

by

Seul Jung

B.S.E.C.E. (Wayne State University) 1988
M.S.E.E (University of California, Davis) 1991

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY of CALIFORNIA

DAVIS

Approved:

_____

_____

_____

Committee in Charge

1996

Committee in Charge:

Dr. Richard C. Dorf, Chair
Professor of Electrical Engineering

Dr. Tien C. Hsia
Professor of Electrical Engineering

Dr. Shih-ho Wang
Professor of Electrical Engineering

Dr. Gary E. Ford
Professor of Electrical Engineering

Dr. Andrew A. Frank
Professor of Mechanical Engineering

Abstract

The fast development in factory automation has resulted in involving a lot of industrial robot manipulators in manufacturing line in the industry in order to increase the productivity and quality of products. The automation process has also demanded the dextrous tools and the sophisticated control algorithms for the robot manipulators to handle various complex situations.

The primary task of the robot manipulators to be performed is the motion control such as carrying an object, painting the surface of an object or welding materials that require only the accurate positioning control. The major position control technique is known to be the computed-torque control or inverse dynamic control which decouples each joint of the robot and linearizes it based on the estimated robot dynamic models. Therefore, the performance of position control is mainly dependent upon the accurate estimations of robot dynamics. In practice, it is true that exact estimation of robot dynamic models is not possible so that the performance of position control is deteriorated. In addition to that, the performance is also subjected to uncertainties in mechanical robot itself.

Many researches have been explored to solve this problem by compensating for those uncertainties. The adaptive and robust control algorithms have been investigated intensively and well formed in theory as well as in applications. These control algorithms have been used to deal with uncertainties that cause due to the nature of non-linear behavior of the robot manipulators and uncertain nature of the envi-

ronment when performed automated tasks in the constrained space. The nonlinear behavior is known to be difficult to control with the linear controller such as the well known PID(proportional-integral-derivative) controller.

It is proper to say that the nonlinear controller is suitable for controlling the nonlinear dynamical system such as robot manipulator. One good candidate of nonlinear controllers is a neural network(NN). Universal approximation capability and adaptive learning capability of NN are so suitable for controlling a nonlinear plant by nonlinear mapping in control process.

The purpose of this research is to develop neural network control algorithms to solve the nonlinear problems by compensating for uncertainties in robot manipulator control so that accurate position and force control can be achieved. In control systems, since nonlinear time varying behavior is present, on-line learning and control is preferred. Within these frameworks, two basic on-line neural network control algorithms have been proposed. The difference between two algorithms is made by minimizing different objective functions: One is to minimize the output error(desired - actual) directly, another is to minimize the output error indirectly by minimizing the model error(real model - estimated model). These two control algorithms were implemented in the joint space control as well as in the Cartesian space control of PUMA 560 robot manipulator.

Furthermore, this compensating idea using neural network has been extended to the force control of robot manipulator as well. Force control is more complicated and

difficult than position control because of controlling position and force simultaneously when the robot manipulator interacts with the environment. The immediate goal of force control is to realize the ideal impedance function by compensating for all uncertainties using neural networks. Usually, for conventional impedance force control, the exact informations on the environment characteristics such as environment stiffness and environment location are required in advance. Therefore, the ultimate goal of neural network force controller in this research is to develop a robust neural network controller that has the force tracking capability as well as compensating capability under uncertainties from both partially known robot dynamics and the totally unknown environment. In this thesis, we have proposed a solution for force control that solves all of the problems mentioned above.

The back-propagation algorithm has been used for training two layered feedforward neural network. The training signal for each control algorithm has been developed. Simulation results demonstrated that neural network controllers perform better than the adaptive control when the partial robot dynamic models are given. Even when no informations of robot dynamics are available neural network controller performs excellently compared with the conventional PD controller.

The tracking performances under NN controllers are extremely improved when compared with the uncompensated nominal case and the well known Craig's adaptive control scheme as well as other NN control schemes proposed by Ishiguro et al.. The proposed method of compensating at input trajectory level gives not only a better per-

formance but also simpler implementation in practice. So the internal controller that has been built already for a certain system is not necessary to be modified at all. It was demonstrated in this thesis that both position and force control using the proposed NN control algorithms perform better than the conventional control methods, and among different NN control schemes the proposed reference compensation technique (RCT) performed best.

As an alternative control algorithm, uncertainties can also be compensated by robust position control algorithm which uses the time delayed information to cancel out any uncertainties. In order to deal with unknown characteristics of environment, the simple robust adaptive impedance control has been proposed and implemented in PUMA 560 robot. The experimental results of this robust adaptive impedance control algorithm prove its robustness as well as its excellent performance.

Tien C. Hsia, *Professor*

Thesis advisor

To Mom and Dad

And my God

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

This research has been supported in part by ***NITTA*** Corporation of Japan. I thank ***NITTA*** Corporation for helping me financially. I would like to thank all my friends in Davis, specially Davis Korean Church members who keep praying for me. I also would like to thank all my colleagues in the Control and system group in Davis, ***Dr. Z.Q. Mao*** and ***Dr. Ty Lasky*** for their valuable discussions with me, and specially ***Dr. R.B. Bonitz*** who actually helped me on experimental works. Also sincere thanks go to our control group professors : ***Dr. R.C. Dorf***, ***Dr. S.H. Wang***, ***Dr. N. Gundes***, ***Dr. T.S. Chang***, and ***Dr. D.Q. Mayne*** who helped me to understand clearly from the basic fundamentals of control systems to the advanced control theory. Specially, I thank ***Dr. Dorf*** and ***Dr. Wang*** for their encouraging me all the time. I also thank ***Dr. Ford*** for his kind advise on neural networks.

Foremost, I would like to thank ***Dr. T.C. Hsia*** who is my advisor since I came here Davis for giving me a chance to keep studying in Ph.D program. He has given a lot of intellectual way of thinking and doing researches which helped me to change my thinking attitude as a researcher. Like a father, he pointed out and corrected my weak points with honest opinions to help me to grow. Working together for almost 8 years has been present since 1988 and debating on some topics has motivated me to see in broader view and to be productive.

Sincerely, I would like to thank my parents for their invaluable consistent helps.

My parents, ***Chang Hi Jung*** and ***Chung Ja Yu***, are always there to support me when I needed. I deeply appreciate for everything they have done to me. Without them, I was not able to finish this research. Finally, I would like to thank ***God*** for giving me a blessing of studying opportunity, helping me to realize how to live and of being with me all the time.

Neural Network Controllers for Robot Manipulators

Seul Jung

June 1996

Electrical & Computer Engineering

Abstract

The fast development in factory automation has resulted in involving a lot of industrial robot manipulators in manufacturing line in the industry in order to increase the productivity and quality of products. The automation process has also demanded the dextrous tools and the sophisticated control algorithms for the robot manipulators to handle various complex situations.

The primary task of the robot manipulators to be performed is the motion control such as carrying an object, painting the surface of an object or welding materials that require only the accurate positioning control. The major position control technique is known to be the computed-torque control or inverse dynamic control which decouples each joint of the robot and linearizes it based on the estimated robot dynamic models. Therefore, the performance of position control is mainly dependent upon the accurate estimations of robot dynamics. In practice, it is true that exact estimation of robot dynamic models is not possible so that the performance of position control is deteriorated. In addition to that, the performance is also subjected to uncertainties in mechanical robot itself.

Many researches have been explored to solve this problem by compensating for

those uncertainties. The adaptive and robust control algorithms have been investigated intensively and well formed in theory as well as in applications. These control algorithms have been used to deal with uncertainties that cause due to the nature of non-linear behavior of the robot manipulators and uncertain nature of the environment when performed automated tasks in the constrained space. The nonlinear behavior is known to be difficult to control with the linear controller such as the well known PID(proportional-integral-derivative) controller.

It is proper to say that the nonlinear controller is suitable for controlling the nonlinear dynamical system such as robot manipulator. One good candidate of nonlinear controllers is a neural network(NN). Universal approximation capability and adaptive learning capability of NN are so suitable for controlling a nonlinear plant by nonlinear mapping in control process.

The purpose of this research is to develop neural network control algorithms to solve the nonlinear problems by compensating for uncertainties in robot manipulator control so that accurate position and force control can be achieved. In control systems, since nonlinear time varying behavior is present, on-line learning and control is preferred. Within these frameworks, two basic on-line neural network control algorithms have been proposed. The difference between two algorithms is made by minimizing different objective functions: One is to minimize the output error(desired - actual) directly, another is to minimize the output error indirectly by minimizing the model error(real model - estimated model). These two control algorithms were implemented

in the joint space control as well as in the Cartesian space control of PUMA 560 robot manipulator.

Furthermore, this compensating idea using neural network has been extended to the force control of robot manipulator as well. Force control is more complicated and difficult than position control because of controlling position and force simultaneously when the robot manipulator interacts with the environment. The immediate goal of force control is to realize the ideal impedance function by compensating for all uncertainties using neural networks. Usually, for conventional impedance force control, the exact informations on the environment characteristics such as environment stiffness and environment location are required in advance. Therefore, the ultimate goal of neural network force controller in this research is to develop a robust neural network controller that has the force tracking capability as well as compensating capability under uncertainties from both partially known robot dynamics and the totally unknown environment. In this thesis, we have proposed a solution for force control that solves all of the problems mentioned above.

The back-propagation algorithm has been used for training two layered feedforward neural network. The training signal for each control algorithm has been developed. Simulation results demonstrated that neural network controllers perform better than the adaptive control when the partial robot dynamic models are given. Even when no informations of robot dynamics are available neural network controller performs excellently compared with the conventional PD controller.

The tracking performances under NN controllers are extremely improved when compared with the uncompensated nominal case and the well known Craig's adaptive control scheme as well as other NN control schemes proposed by Ishiguro et al.. The proposed method of compensating at input trajectory level gives not only a better performance but also simpler implementation in practice. So the internal controller that has been built already for a certain system is not necessary to be modified at all. It was demonstrated in this thesis that both position and force control using the proposed NN control algorithms perform better than the conventional control methods, and among different NN control schemes the proposed reference compensation technique (RCT) performed best.

As an alternative control algorithm, uncertainties can also be compensated by robust position control algorithm which uses the time delayed information to cancel out any uncertainties. In order to deal with unknown characteristics of environment, the simple robust adaptive impedance control has been proposed and implemented in PUMA 560 robot. The experimental results of this robust adaptive impedance control algorithm prove its robustness as well as its excellent performance.

Tien C. Hsia, *Professor*

Thesis advisor

Chapter 1

Introduction

1.1 Problem Statement

Identification and control of a nonlinear system (a robot manipulator) have been a difficult challenging problem to be solved for decades. Due to the lack of mathematical system theory for representing nonlinear system characteristics, the exact dynamics of a nonlinear system are usually unknown so that it is difficult to satisfy the requirements of stability and desired performance. As the most popular approach, computed-torque method or inverse dynamic control method is used most for robot dynamic control when the estimated robot dynamic models are available [1, 2, 3]. Since this method uses the inverse dynamic model of the robot, which is impossible to get the exact model in most practical cases, in order to linearize and decouple the robot dynamic equations, exact linearization and decoupling can not be achieved. Then, this results in the steady state position tracking error or even instability [2]. The adaptive control [4, 5] and the robust control [6, 7, 8] have been studied to solve these problems for many years.

The linear adaptive control approach tries to adjust the controller characteristics by estimating the parameters of the unknown system in order to stabilize the system while the robust control approach has a fixed controller which stabilizes the system over a range of the acceptable performance for uncertain parameters [7]. The adaptive control approach such as the model reference adaptive control (MRAC) and the self tuning regulator (STR) is already well developed in the literature [9]. The adaptive law is derived from the Lyapunov function or the regression function which are usually

difficult to find. And they have limitations of performance on complex nonlinear system even though they have been successful for linear systems or some nonlinear systems. As a robust control approach the second method of Lyapunov technique has been designed [2]. The bound of uncertainties are required to be known by limiting certain accuracy of inertia matrix. The sliding mode control is also designed for nonlinear system and it also requires the information of bounds of uncertainties [8]. Another robust control, disturbance rejection scheme, based on decentralized linear system by using previous dynamic informations to cancel out uncertainties has been proposed by Hsia [10, 11, 12, 13, 14]. Most recent works on robust position control has been investigated in this thesis.

On the other hand, for last several years, there have been a lot of interests and advances of researches on artificial neural networks (NNs) for control system to solve the nonlinearity and complexity of a nonlinear system through nonlinear mapping abilities [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. NNs have been successfully proved its potential capability of identifying and learning the nonlinearity of a system during process of execution [28]. NN applications to control robot manipulators can be found in many papers [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]. Specially, NN controls for underwater vehicles [53, 54, 55, 56] and mobile robot [57, 58] have been also investigated. The neural controller has been actively used in induction motor control as well because the motor dynamic is nonlinear and difficult to control with linear controllers such as PID and

PI [59, 60, 61, 62].

It is also proved that any continuous function in compact domain can be approximated by two layer feedforward NN(one hidden layer) with a certain accuracy under the condition that sufficient hidden neurons are available [63, 64]. This universal approximation capability may also solve the problem of the limitation of the adaptive linear filter for the nonlinear system in adaptive system theory. This may also give more flexibility to the system by tolerating the stability range over uncertain parameters to robust control system.

In this thesis, NN controllers perform the adaptive control process in on-line fashion to control, to identify a nonlinear dynamical system and to compensate for any structured and unstructured uncertainties of a nonlinear system that discovered during learning process by forming a multilayer neural network. Therefore, they achieve a robust adaptive control by gradually adjusting the internal weights of the neural networks such that the error between the actual and desired value is minimized by back-propagation (BP) learning algorithm which is based on gradient decent method. In general NN training methods can be classified into three groups based on implementation of their learning algorithms : on-line(OL) learning method, off-line(FL) learning method, and combination of OL and FL. OL and FL can be defined by whether the training data are available at the time $\boldsymbol{t}$ or not. FL method allows the operation with the training data collected before the analysis of parameters while OL operates as the data become available to the system every sampling time. By

the nature of the control system OL method is desired because OL method can deal with any time-varying parameters in robot dynamics during process while FL method can deal with only the time-invariant behavior. Therefore, FL method needs to be retrained from task to task because learning process is different from the adaptive process unless one training set can represent a whole universal set of working space. So generalization ability of NN degrades.

Our interest is mainly limited to on-line NN control. Within this framework, many neural network control schemes have been proposed in the literature. In the paper [65], they compared CMAC(cellebellar model articulated controller or cellebellar model arithmetic computer) neural network and traditional adaptive control systems. They showed that CMAC neural network performs well over all and it presented superiority in robustness under dynamic model mismatches. In other papers they showed the real time control possibility using CMAC scheme [29, 37]. Chen [18] used error back propagation NN for a nonlinear self-tuning adaptive control and demonstrated its good performance. In the papers [23, 24] as a modified version of the model reference adaptive control scheme (MRAC) represented in [19], the indirect adaptive control which uses two recurrent type neural networks, a neural identifier and a neural controller, and dynamic back-propagation algorithms [20] to train these two NNs is presented. A neural identifier provides system Jacobian informations of a plant to the neural controller so that the neural controller can be used as an inverse controller of the system without a primary feedback controller.

For on-line robot dynamic control using NN controller(compensator), NN control schemes can be classified into two groups with respect to availability of informations of the robot dynamics : a model-based approach (computed-torque based control) and a non model-based approach(PD type control). Kawato's feedback error learning (FEL) control [31] and Ishiguro et. al's neural compensator proposed in the paper [43] fall into model-based approach. The idea of the Kawato's scheme is so attractive that on-line control is possible which learning and execution of the process is executed simultaneously. This scheme uses the method of partitioning the priori known robot inverse dynamic formulation into several components, so called subsystems. Hence, this scheme can be considered as the model-based nonlinear adaptive control system.

Another model based approach to robot dynamic control is to compensate for all uncertainties using NNs [42, 44]. The feedback or feedforward configurations of neural compensator can likely be used to perform the good computed-torque method in robotics. Since the inexact cancellation of dynamic model causes the poor tracking performance it is necessary to compensate for the approximated parameters in dynamic models. In the paper [42] they use two neural networks(one for inertia and another for centripetal, coriolis, and disturbances) to replace the approximated model matrices in order to compensate for any parameter variations. In first phase, neural networks are trained off-line using the mathematical model to learn the structure of the robot manipulator and used on-line in the second phase to learn real parameters and unstructured uncertainties. This is a combination of generalized learning and

specialized learning method discussed in the paper [15].

Instead of learning whole operation data as shown in [42], it is better for NN controller to compensate for those uncertainties [43, 44]. Even though NN compensating uncertainties may have loss of it's generalization characteristic it is acceptable if the adaptation rate is fast by using small number of neurons. A feedback neural compensator combined with computed-torque method was proposed to eliminate any uncertainties by compensating any sudden unmodeled dynamic changes based on learning algorithms [43]. The feedback neural compensator generates the compensating signal to make the over all system be a good computed-torque controlled system. In order to train NN the robot dynamic model is required to compute teaching signal which is time consuming.

In many nonlinear dynamic systems including the robot manipulators it is not easy to obtain their dynamic models. Very often partial-model based or non-model based approach is appropriate when the robot model is not available or little knowledge is available. The direct inverse neural control and the indirect inverse neural control method with primary feedback controller falls into partial or non model based approach category in terms of whether gravity is compensated or not. The direct way means that the internal weights of NN controller are directly adjusted by output errors, while the indirect way means that NN weights are indirectly adjusted by using a plant estimation by another NN. Non-model based NN controller is universal in a sense that it can be used for other dynamical systems as well as robot manipulator,

and universal approximation capability of NN is fully used.

The performance of NN is effected by NN parameters. In addition to those parameters the magnitude of inputs is also a factor because NN is a numerical processing unit and it's nonlinear function has certain operational bounds (e.g. $\pm\mathbf{1}$). Usually NN compensating signals of previous control designs proposed are added to torque level [31, 43]. This might need normalization/de-normalization process for better mapping. The problem of normalization process is that the normalizing factor is not accurately available in practice. Here we are proposing NN controller which compensates at input trajectory level called the reference compensation technique.

Force control is an extension of position control. One of the well known force control algorithm is impedance control that uses the mechanical stiffness between position and force. In order to achieve good force tracking accurate informations on environment are required. But, in practice, it is very difficult to know those informations in advance. Therefore, new force control algorithm is desired to solve those problems as well as uncertainty problems. In this thesis, we also address those issues and propose a new solution.

1.2 Thesis Organization

The thesis is composed of 8 chapters. In chapter 2, a brief introduction and a history of neural network are presented. The back propagation learning algorithm is introduced for two layer feedforward neural network structure. In Chapter 3, various

general NN controller schemes that have been proposed in the literature are investigated first and then the comprehensive studies of those schemes are presented. NN controllers are designed to generate the compensating signals of the range from small uncertainty terms for the model based to full dynamic models for the non-model based. In these frameworks, computer simulation studies of robot position tracking under both model-based and non-model based are carried out. The chapter 4 begins with presenting a comprehensive study of differently designed feedforward NN controllers mentioned above. All designs are about trajectory control of robot manipulator using neural network as a robust tool in order to perform the robust control under nonlinear uncertainties presence. Extended implementation works are delivered and compared with other NN controller scheme by using computer simulations [66]. In addition to that, novel NN control schemes, which are conceptually different from those schemes [66] in the literature, for robot manipulator are proposed [67]. The proposed schemes are considered as the inverse NN control of stabilized plants. The slightly modified scheme of [67] is also proposed to perform faster convergence and better stability [68]. The proposed inverse model control schemes have structural advantages over generic schemes [43] in that compensation is done outside control loop so that it can be implemented easily at the command trajectory level external to an existing controller without having to modify the primary controller's internal structure. Since this technique is compensating at reference trajectory it is called Reference Compensation Technique(RCT).

Another primitive robot control technique is the non-model based PD control. As an extension of model based NN control, the non model based NN control is presented in Chapter 5. The goal of NN controllers for non-model based case is to identify the inverse dynamics of robot manipulator by minimizing system output error $\boldsymbol{\varepsilon} = \boldsymbol{q_d} - \boldsymbol{q}$ directly or indirectly. Non-model based NN controllers are universal in a sense that they can be used to control other nonlinear plants besides robot manipulators. NN shows excellent performance even under no knowledge of robot dynamic model at all.

The similar concepts of NN control in joint space have been extended to the Cartesian space as well. In Chapter 6 the robot tracking performances according to different compensating locations are investigated. The different magnitudes of compensating signals depending on different compensating locations also have different performances. Then the proposed NN control technique is applied to non-model based PD control case [69]. The relationships between PD gains and performances are studied under the proposed scheme and the FEL scheme. Simulating trajectory tracking of a three link rotary robot manipulator shows that the proposed neural network controllers improves the performance under uncertainties presence.

As the final stage of my research, extension of the proposed NN controllers to impedance force control of robot manipulator has been performed in Chapter 7. As an extension of Cartesian NN control NN is applied to impedance control which regulates position and exerted force at the same time by the mechanical impedance relationship between position and sensed force. A simple trajectory modification

technique is proposed to give the impedance controller more robustness under the unknown environment stiffness. The proposed robust NN force control algorithm is able to track a desired force and to compensate for uncertainties in robot dynamic model and the environment. Separate from neural network control, in Chapter 8, the robust position control is presented to show as an alternative way of achieving the same goal of compensating for uncertainties. The adaptive robust force control based on the robust position control is also introduced. The robust control algorithm is quite simple and is easily implemented in real time. Experimental results along with simulation results are presented to show the controller's performance.

Chapter 2

Neural Network Model

2.1 Introduction

Artificial neural networks are parallel computing elements that mimic biological neural systems. Research on artificial neural network has been motivated to give human intelligence to computing machine in order to make computers be smart. Our brain as a parallel processing element is functioning in terms of a lot of neuron connections. Recently, artificial neural networks or simply neural networks have been used as major tools for solving many complex problems in many applications such as control, signal processing, pattern recognition and economics etc. The major function of neural networks is a mapping capability of nonlinear function through a training algorithm based on teaching data. Once neural network is trained based on previous data it can also predict the future behavior of that system in the area of stock market and weather forecast. In control area, neural network is used as an identifier that identifies the plant dynamics or a controller that controls the plant. Since most systems are nonlinear, neural network is a good candidate for a nonlinear controller. Their capabilities of parallel computation, learning, adaptation, generalization are very important characteristics used in many areas.

History of neural Networks

The first artificial neuron model is known as "M-P Neuron" named after the first characters of MaCulloch and Pitts [70]. This is a simple model that sums all input signals and performs logical operations. Later Herb's rule was developed to give adaptability to M-P neuron by modifying the strength of weights that fire. In 1957

more refined neuron model called "Perceptron" was developed by Rosenbratt [71]. It consists of summing junction and threshold function. The internal weights are trainable by performing the gradient decent algorithm. The perceptron was sensational at that time because of its simplicity and promising potential capability. At about same time "ADALINE" was introduced by Widrow. Since it is based on the adaptive linear filter it was used for adaptive signal processing with least-mean-square(LMS) algorithm. In 1969, however, Minsky and Papert proved that the perceptron could not simulate the simple XOR logic function [72]. This was a big attack on neural network society. After this attack, the era of neural network has become dark.

The discovery of back-propagation algorithm ended the dark era of neural network. Initially Werbos discovered the principles of back-propagation algorithm in his Ph.D thesis [73], but it was not published in public until Parker rediscovered this beautiful algorithm in 1982 [74]. Later this back-propagation algorithm was rediscovered and well developed for feedforward neural network by Rumelhart, Hinton and Williams in their book of parallel distributed processing [75]. After discovery of back-propagation algorithm the new era of neural network begins.

The historical perspectives of neural network can be summarized as follows:

2.2 A Single Artificial Neuron

An artificial neuron (AN) unit has the imitation structure of a biological neuron in human brain. It consists of processing input elements, a summing junction, weights,

Table 2.1: History of Neural Network

1943	"M-P Neuron" by MaCulloch and Pitts
1949	Herbs rule $\Delta w_{ij} = \epsilon a_i a_j$
1957	"The Perceptron" by Rosenbratt
1960	"ADALINE"(ADAptive LINear Element) by Widrow and Hoff
1962	"MADALINE" by Widrow
1969	Minsky and Papert' Attack
1974	"BP Algorithm" by Werbos
1975	"CMAC" by Albus
1978	"Neogonitron" by K. Fukushima
1982	"Self Organizing map" by kohonen
1982	"Hopfield Net"
1982	Rediscovered BP by Parker
1983	"ART1" by Carpenter and Grossberg
1986	"A generalized delta rule" by Rumelhart, Hilton, William
1987	"BAM(Bidirectional associative memory)" by Kosko
1989	"Universal Approximation proof" by Cybenko, Fruhashi
1991	"Universal approximation proof" by Honik

and outputs, which approximately corresponds to dendrites, axons, strengths between neurons and synapses in the biological neuron as shown in Figure 2.1. A neuron unit performs learning by adjusting weight strengths between neurons. In general, the artificial neural networks (NNs) are formed in layers and weight connections by a number of AN unit.

The signal is activated as follows: The input, x_i, from other neuron's synapse is multiplied by its weight w_i, and added together with other inputs and propagate through axon by activation function. The threshold units fire when their total summation value of inputs exceeds certain bias levels. The output of a neuron can be

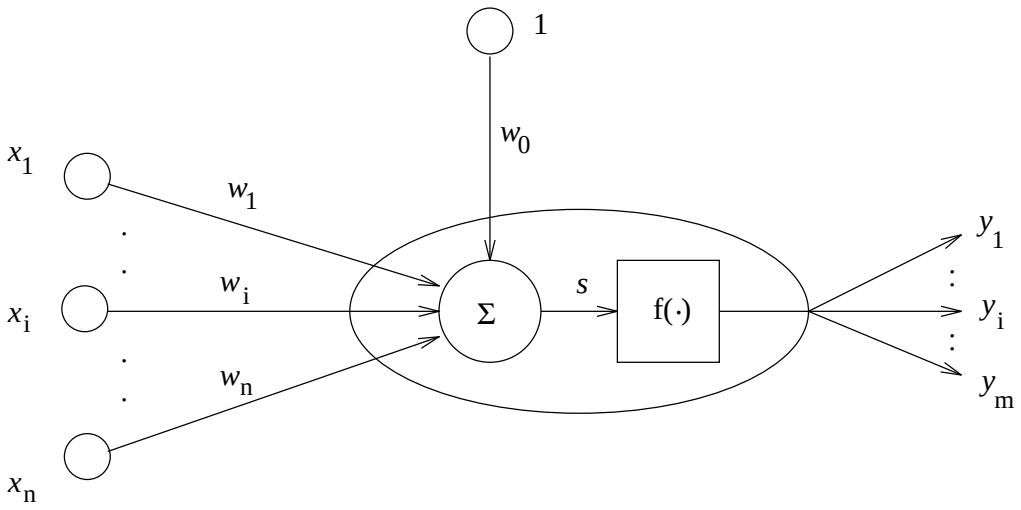


Figure 2.1: Basic structure of a single neuron

represented as :

$$\boldsymbol{y_p = f(\sum_{i=1}^{n} x_i w_i + w_0)} \tag{2.1}$$

where $\boldsymbol{w_i}$ is an $\boldsymbol{i}$th weight and $\boldsymbol{w_0}$ is the bias. The bias can be considered as a weight so that it can be updated as same as other weights. The activation function $\boldsymbol{f(x)}$ can be a linear, a hard limiter, and sigmoid functions depending on their applications. The linear function is used when the output is expected to be continuous values.

$$\underline{\textbf{\textit{Linear function}}} \;:\; \boldsymbol{f(x) = ax} \qquad \boldsymbol{-B \le f(x) \le B} \tag{2.2}$$

$$\underline{\textbf{\textit{Sigmoidal function}}} \;:\; \boldsymbol{f(x) = \frac{1}{1 + e^{(-\lambda(x-\theta))}}} \qquad \boldsymbol{0 \le f(x) \le 1} \tag{2.3}$$

$\boldsymbol{\lambda}$ determines the steepness of an activation function $\boldsymbol{f(x)}$ and $\boldsymbol{\theta}$ is a bias that shift the activation function with respect to given input, $\boldsymbol{x}$ so that it gives another degree of freedom. Sigmoidal function has the saturation at 0 and 1, so it limits the output to

be positive values. Typically, $\boldsymbol{\lambda = 1}$, is often used and refereed as a logistic function.

$$\underline{\textbf{\textit{Tangent hyperbolic function}}} : \boldsymbol{f(x) = \frac{1 - e^{(-\lambda(x-\theta))}}{1 + e^{(-\lambda(x-\theta))}}} \qquad \boldsymbol{-1 \leq f(x) \leq 1} \tag{2.4}$$

Tangent hyperbolic function has the saturation at $\boldsymbol{-1}$ and $\boldsymbol{1}$. The effect of $\boldsymbol{\lambda}$ on performance has been investigated in real control application [40, 35]. In addition to that the adaptive method of updating $\boldsymbol{\lambda}$ has been proposed. The output of this function can have a range between negative values and positive values.

The artificial neural networks are formed by connecting a number of a single neuron unit together. The neural networks which use error-back-propagation algorithm can be classified into two groups such as feedforward network and recurrent network based on their feed back connection structure. Specially, in this thesis, the feedforward NN is used, and linear function is used at input and output layer and nonlinear sigmoid function is used in hidden layer.

2.2.1 Feedforward Neural Networks

The feedforward network proceeds from input to output without any feedback connections from output to input. Figure 2.2 shows the typical two layer feedforward neural network which are composed of an input layer, one hidden layer, and an output layer. The number of neurons are chosen to be $\boldsymbol{l, m,}$ and $\boldsymbol{n}$ in the input layer, the hidden layer, and the output layer, respectively. The output of the hidden layer, $\boldsymbol{Y^1}$

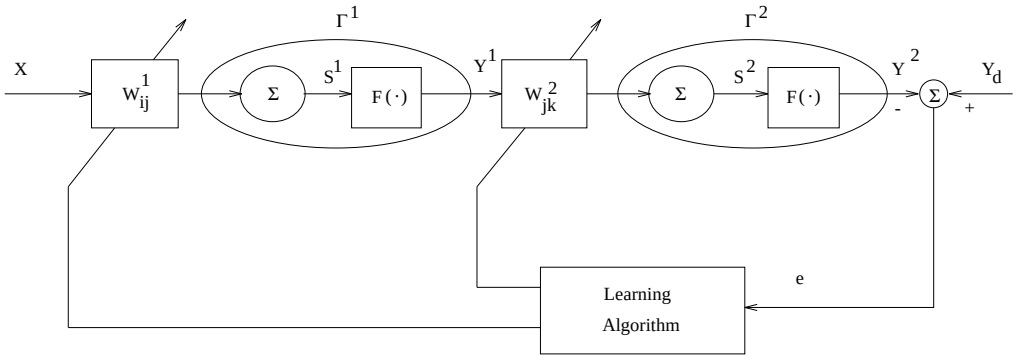


Figure 2.2: Two layer feedforward neural Network

, is represented as :

$$\boldsymbol{Y^1 = \Gamma^1[W_{ij}^{1^T} X]} \tag{2.5}$$

where $\boldsymbol{X}$ is $\boldsymbol{l}$ x 1 input vector, $\boldsymbol{W_{ij}^1}$ is the $\boldsymbol{l}$ x $\boldsymbol{m}$ weight matrix between input layer and hidden layer, $\boldsymbol{Y^1}$ is a $\boldsymbol{n}$ x 1 output vector of hidden layer, and $\boldsymbol{\Gamma^1}$ is $\boldsymbol{m}$ x $\boldsymbol{m}$ diagonalized operator matrix which operates between input layer and hidden layer. T is denotes as the transpose of a matrix. The operator, $\boldsymbol{\Gamma}$, can be represented as

$$\boldsymbol{\Gamma = f(S)} \tag{2.6}$$

The mapping from input layer to output layer can be represented by the operators as follows:

$$\begin{aligned} \boldsymbol{Y^2} &= \boldsymbol{\Gamma^2[W_{jk}^{2^T} Y^1]} \\ &= \boldsymbol{\Gamma^2[W_{jk}^{2^T} \Gamma^1[W_{ij}^{1^T} X]]} \end{aligned} \tag{2.7}$$

where $\boldsymbol{W_{jk}^{2^T}}$ is a $\boldsymbol{m}$ x $\boldsymbol{n}$ weight matrix between hidden layer and output layer, $\boldsymbol{Y^2}$ is a $\boldsymbol{n}$ x 1 output vector, and $\boldsymbol{\Gamma^2}$ is the $\boldsymbol{n}$ x $\boldsymbol{n}$ diagonalized operator matrix between

the hidden layer and the output layer. The error vector $\boldsymbol{e}$ is formed by subtracting actual output vector from desired vector

$$\boldsymbol{E = Y_d - Y^2} \tag{2.8}$$

The learning algorithm is to minimize the objective function

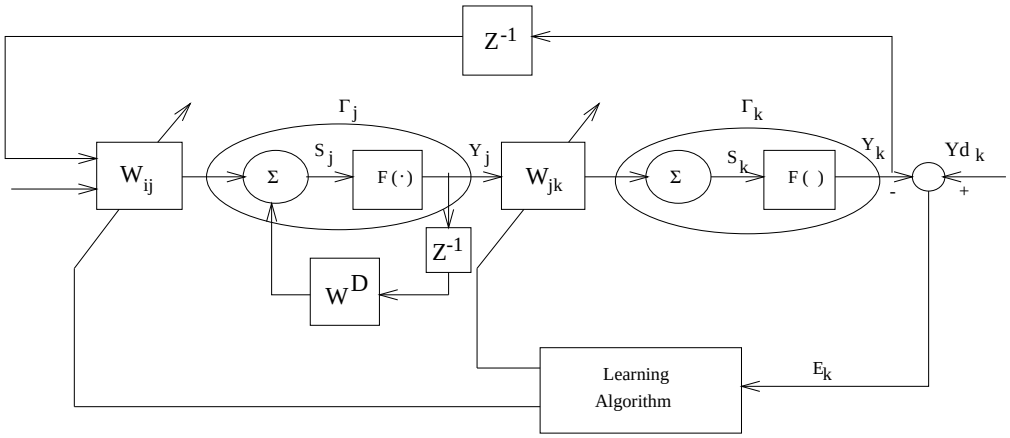
$$\mathcal{J} = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{E^T E} \tag{2.9}$$

2.2.2 Recurrent Neural Networks

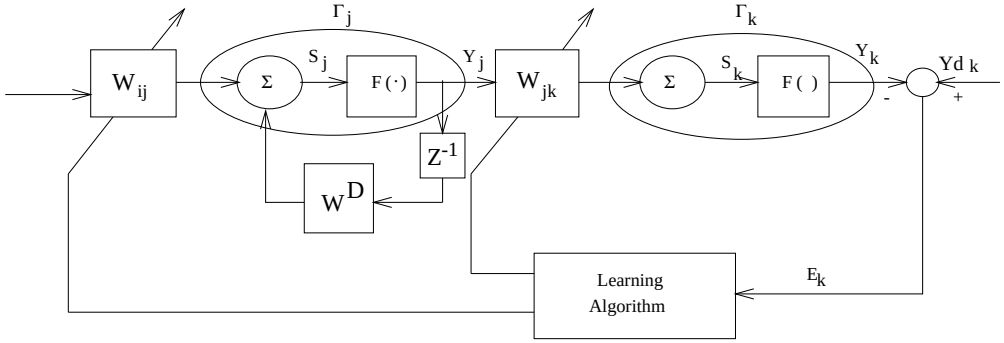
The recurrent network has dynamic nonlinear mapping ability since it has the recursive structure in it, which is suitable for dynamical system while the feedforward network represents the static nonlinear mapping. However, the fully connected recurrent network should pay longer computing time comparing with feedforward neural network. Figure 2.3 shows two types of recurrent neural networks : (a) has the feedback from previous output to input and recursive hidden layer, (b) has only recursive hidden layer. The output of the hidden layer has a recursive structure with new weight vector,$\boldsymbol{W_j^D}$, so that it has dynamic memory of previous state, $\boldsymbol{Y_j(t)}$. The new state,$\boldsymbol{Y_j(t+1)}$, is represented as :

$$\boldsymbol{Y_j(t+1) = \Gamma_j[W_{ij}^T I(t) + W_{dj}^T Y_j(t)]} \tag{2.10}$$

where $\boldsymbol{W_j^D}$ is the $\boldsymbol{m}$ x 1 weight vector, $\boldsymbol{I(t)}$ is $\boldsymbol{(l+n)}$ x 1 input vector for (a) and $\boldsymbol{l}$ x 1 input vector for (b) such that $\boldsymbol{dim(I) = dim(X) + dim(Y)}$ and $\boldsymbol{I(t+1) =}$



(a)



(b)

Figure 2.3: Two Types of Recurrent Networks (a) full configuration(b) diagonal configuration

$[\boldsymbol{Y_k(t), X_i(t+1)}]^{\boldsymbol{T}}$ for (a). The mapping from input layer to output layer can be represented by the operator as following:

$$\boldsymbol{Y_k(t+1) = \Gamma_k[W_{jk}^T I(t) Y_j(t)] = \Gamma_k[W_{jk}^T \Gamma_j[W_{ij}^T I(t) + W_{dj}^T Y_j(t)]]} \qquad (2.11)$$

where $\boldsymbol{W_{jk}}$ is a $\boldsymbol{m}$ x $\boldsymbol{n}$ weight matrix.

2.3 Back-propagation Algorithms

The back-propagation training algorithm for neural networks is based on gradient decent method that searches for the minimum of the objective function specified. From the objective function (2.9), we assume that this learning rule is analyzed based on a certain input pattern $\boldsymbol{p}$ defining $\boldsymbol{y_k}$ as a $\boldsymbol{k}$th output of NN and $\boldsymbol{yd_k}$ as a desired output. Then the objective function $\mathcal{J}_{\boldsymbol{p}}$ is

$$\mathcal{J}_{\boldsymbol{p}} = \frac{\mathbf{1}}{\mathbf{2}} \sum (\boldsymbol{yd_k} - \boldsymbol{y_k})^2 \tag{2.12}$$

The gradient becomes

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} = - \sum \boldsymbol{e_k} \frac{\partial \boldsymbol{y_k}}{\partial \boldsymbol{w}} \tag{2.13}$$

where $\boldsymbol{e_k} = \boldsymbol{yd_k} - \boldsymbol{y_k}$. From the gradient decent method, the weight update rule called the delta rule is formed to update the current weight values based on the error and input data.

2.3.1 Delta rule

If network has one layer structure from input $\boldsymbol{x_j}$ to output $\boldsymbol{y_k}$ and the output is linear function, then the weights $\boldsymbol{w_{jk}}$ between input and output layer can be updated by the delta rule as follows:

$$\begin{aligned} \boldsymbol{\Delta w_{jk}} &= -\boldsymbol{\eta} \frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} \\ &= \boldsymbol{\eta} (\boldsymbol{yd_k} - \boldsymbol{y_k}) \frac{\partial \boldsymbol{y_k}}{\partial \boldsymbol{w_{jk}}} \\ &= \boldsymbol{\eta e_k x_i} \end{aligned} \tag{2.14}$$

where yd_k is the desired output and η is a update rate.

2.3.2 Generalized delta rule

From the objective function (2.9), the gradient yields

$$\Delta w = -\eta \frac{\partial \mathcal{J}_p}{\partial w} = \eta e_k \frac{\partial y_k}{\partial w} \tag{2.15}$$

2.3.2.1 Feedforward neural network

If the network has two or more layer structures from input to output, then each layer's weights have to be updated. For simplicity, consider two layered network which consists of input layer i, hidden layer j, and output layer k. The number of inputs is N_I, the number of hidden units is N_H, and the number of output units is N_O.

(a) The output y_k The input signals x_i are multiplied by weights w_{ij} and added.

$$s_j = \sum_{i=1}^{N_I} w_{ij} x_i \tag{2.16}$$

And summed signal goes through the Sigmoidal function and becomes inputs to the next layer

$$y_j = f(s_j) = \frac{1}{1 + e^{-s_j}} \tag{2.17}$$

The input to hidden layer are multiplied by the weights w_{jk} and added. Hence the signal transition from input x_i to output y_k is

$$y_k = f(s_k) = \frac{1}{1 + e^{-\sum_{j=1}^{N_H} w_{jk} y_j}} \tag{2.18}$$

(b) Updating the weight $\boldsymbol{w_{jk}}$ between hidden and output layer

$$\begin{aligned} \boldsymbol{\Delta w_{jk}} &= -\boldsymbol{\eta}\frac{\boldsymbol{\partial \mathcal{J}_p}}{\boldsymbol{\partial w_{jk}}} \\ &= \boldsymbol{\eta e_k}\frac{\boldsymbol{\partial y_k}}{\boldsymbol{\partial w_{jk}}} \\ &= \boldsymbol{\eta e_k f'(s_k)}\frac{\boldsymbol{\partial s_k}}{\boldsymbol{\partial w_{jk}}} \end{aligned} \tag{2.19}$$

Since $\frac{\partial s_k}{\partial w_{jk}} = y_j$, $\boldsymbol{\Delta w_{jk}}$ is

$$\boldsymbol{\Delta w_{jk} = \eta \delta_k y_j} \tag{2.20}$$

where

$$\boldsymbol{\delta_k = e_k f'(s_k),\ where\ f'(s_k) = f(s_k)(1 - f(s_k))} \tag{2.21}$$

where if the output layer is Sigmoidal function.

$$\boldsymbol{\delta_k = e_k, since\ f'(s_k) = 1} \tag{2.22}$$

where the output layer is linear.

(c) Updating the weights $\boldsymbol{w_{ij}}$ between input and hidden layer

$$\begin{aligned} \boldsymbol{\Delta w_{ij}} &= -\boldsymbol{\eta}\frac{\boldsymbol{\partial \mathcal{J}_p}}{\boldsymbol{\partial w_{ij}}} \\ &= \boldsymbol{\eta e_k}\frac{\boldsymbol{\partial y_k}}{\boldsymbol{\partial w_{ij}}} \end{aligned} \tag{2.23}$$

Since $\frac{\partial y_k}{\partial w_{ij}}$ is not available, a chain rule is applied so that

$$\begin{aligned} \frac{\boldsymbol{\partial y_k}}{\boldsymbol{\partial w_{ij}}} &= \frac{\boldsymbol{\partial y_k}}{\boldsymbol{\partial y_j}}\frac{\boldsymbol{\partial y_j}}{\boldsymbol{\partial w_{ij}}} \\ &= \frac{\boldsymbol{\partial y_k}}{\boldsymbol{\partial y_j}}\frac{\boldsymbol{\partial y_j}}{\boldsymbol{\partial s_j}}\frac{\boldsymbol{\partial s_j}}{\boldsymbol{\partial w_{ij}}} \\ &= \frac{\boldsymbol{\partial y_k}}{\boldsymbol{\partial y_j}}\boldsymbol{f'(s_j)x_i} \end{aligned} \tag{2.24}$$

since $\frac{\partial y_j}{\partial s_j} = f'(s_j), \frac{\partial s_j}{\partial w_{ij}} = x_i$.

$$\begin{aligned} \frac{\partial y_k}{\partial y_j} &= \sum_{k=1}^{N_O} \frac{\partial y_k}{\partial s_k}\frac{\partial s_k}{\partial y_j} \\ &= \sum_{k=1}^{N_O} f'(s_k) w_{jk} \end{aligned} \tag{2.25}$$

since $\frac{\partial y_k}{\partial s_k} = f'(s_k), \frac{\partial s_k}{\partial y_j} = w_{jk}$.

$$\Delta w_{ij} = \eta f'(s_j) x_i \sum_{k=1}^{N_O} e_k f'(s_k) w_{jk} \tag{2.26}$$

2.3.2.2 Diagonal recurrent neural network

The back propagation updating algorithm for diagonal recurrent NN shown in Figure 2.3 (b) is more complicated than that of feedforward NN because it has another internal weights w_j^D.

(a) The output y_k

The signal transition from input x_i to output y_k is

$$y_k(t) = f(s_k) = \frac{1}{1 + e^{-\sum_{j=1}^{N_H} w_{jk} y_j}} \tag{2.27}$$

$$y_j(t) = f(s_j) = \frac{1}{1 + e^{-s_j}} \tag{2.28}$$

$$s_j(t) = w_j^D y_j(t-1) + \sum_{i=1}^{N_I} w_{ij} x_i \tag{2.29}$$

where $y_j(t-1)$ is the previous value of $y_j(t)$. In this case an extra term $w_j^D y_j(t-1)$ is added.

(b) Updating the weight w_{jk} between hidden and output layer is same as feedfor-

ward neural network.

$$\begin{aligned} \Delta w_{jk} &= -\eta \frac{\partial \mathcal{J}_p}{\partial w_{jk}} \\ &= \eta e_k \frac{\partial y_k}{\partial w_{jk}} \\ &= \eta e_k f'(s_k) \frac{\partial s_k}{\partial w_{jk}} \end{aligned} \tag{2.30}$$

Since $\frac{\partial s_k}{\partial w_{jk}} = s_j$, Δw_{jk} is

$$\Delta w_{jk} = \eta \delta_k s_j \tag{2.31}$$

where

$$\delta_k = e_k f'(s_k) \tag{2.32}$$

(c) Updating the weights w_j^D

$$\Delta w_j^D = -\eta \frac{\partial \mathcal{J}_p}{\partial w} \tag{2.33}$$

Since $\frac{\partial y_k}{\partial w_{ij}}$ is not available, chain rule is applied so that

$$\begin{aligned} \frac{\partial \mathcal{J}_p}{\partial w_j^D} &= \frac{\partial \mathcal{J}_p}{\partial y_k} \frac{\partial y_k}{\partial w_j^D} \\ &= -e_k \frac{\partial y_k}{\partial s_k} \frac{\partial s_k}{\partial y_j} \frac{\partial y_j}{\partial w_j^D} \\ &= -e_k f'(s_j) w_{jk} \frac{\partial y_j}{\partial w_j^D} \end{aligned} \tag{2.34}$$

Let

$$\begin{aligned} P_j(t) &= \frac{\partial y_j(t)}{\partial w_j^D} = \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_j^D} \\ &= f'(s_j) \frac{\partial}{\partial w_j^D} [w_j^D y_j(t-1) + \sum_{i=1}^{N_I} w_{ij} x_i] \end{aligned}$$

$$= f'(s_j)[y_j(t-1) + w_j^D \frac{\partial y_j(t-1)}{\partial w_j^D}] \quad (2.35)$$

$$= f'(s_j)[y_j(t-1) + w_j^D P_j(t-1) \quad (2.36)$$

where $P_j(t-1) = \frac{\partial y_j(t-1)}{\partial w_j^D}$. Therefore,

$$\Delta w_j^D = \eta P_j(t) e_k \sum_{j=1}^{N_H} w_{jk} \quad (2.37)$$

(d) Updating the weights w_{ij} between input and hidden layer

$$\begin{aligned} \Delta w_{ij} &= -\eta \frac{\partial \mathcal{J}_p}{\partial w_{ij}} \\ &= \eta e_k \frac{\partial y_k}{\partial w_{ij}} \end{aligned} \quad (2.38)$$

Since $\frac{\partial y_k}{\partial w_{ij}}$ is not available, a chain rule is applied so that

$$\frac{\partial y_k}{\partial w_{ij}} = \frac{\partial y_k}{\partial y_j} \frac{\partial y_j}{\partial w_{ij}} \quad (2.39)$$

$$\begin{aligned} \frac{\partial y_k}{\partial y_j} &= \sum_{k=1}^{N_O} \frac{\partial y_k}{\partial s_k} \frac{\partial s_k}{\partial y_j} \\ &= \sum_{k=1}^{N_O} f'(s_k) w_{jk} \end{aligned} \quad (2.40)$$

since $\frac{\partial y_k}{\partial s_k} = f'(s_k), \frac{\partial s_k}{\partial y_j} = w_{jk}$.

Defining $Q_{ij}(t)$ as

$$\begin{aligned} Q(t)_{ij}(t) &= \frac{\partial y_j(t)}{\partial w_{ij}} = \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_{ij}} \\ &= f'(s_j) \frac{\partial}{\partial w_{ij}} [w_j^D y_j(t-1) + \sum_{i=1}^{N_I} w_{ij} x_i(t)] \\ &= f'(s_j)[w_j^D \frac{\partial y_j(t-1)}{\partial w_i j} + x_i(t)] \\ &= f'(s_j) \sum_{j=1}^{N_H} w_{jk}[x_i(t) + w_j^D Q_{ij}(t-1)] \end{aligned} \quad (2.41)$$

where $\boldsymbol{Q_{ij}(t-1) = \frac{\partial y_j(t-1)}{\partial w_{ij}}}$.

Finally,

$$\boldsymbol{\Delta w_{ij} = \eta Q(t)_{ij}(t) \sum_{k=1}^{N_O} e_k f'(s_k) w_{jk}} \tag{2.42}$$

Chapter 3

Identification and Control

In this chapter, NN is used as an adaptive filter which identifies the forward and inverse of a plant. Based on identification technique inverse model control structure is presented.

3.1 Introduction

For the past decade, many attempts have been made to solve the problems of identification and control of complex nonlinear system by exploiting the nonlinear mapping abilities of the NN [16, 18, 19, 40, 22, 26, 24, 60, 59]. At the same time, extensive investigations have been carried out to design NN controllers for robot manipulators as well [31, 43, 44, 42, 36, 32, 30]. As a result, the nonlinear mapping and learning capabilities allow neural networks to replace all the functions of the conventional controllers(C, F, and O) shown in Figure 3.1.

The main goal of neural networks in control area is to identify the inverse dynamic model of the system and to perform as an inverse controller in on-line sitting. The direct inverse control structure can be shown from Figure 3.1 when $\boldsymbol{F} = \boldsymbol{0}, \boldsymbol{P} = \boldsymbol{I}, \boldsymbol{O} = \boldsymbol{0}$ and $\boldsymbol{C} = \boldsymbol{N}$ where $\boldsymbol{N}$ represents a neural network. This structure requires the informations about the dynamic Jacobian of a dynamical system in order to apply back propagation algorithm directly by minimizing output errors. Unfortunately, the Jacobian is not exactly available in most cases so that off-line pre-training is required[44, 42].

Several methods have been proposed to approximate the dynamic Jacobian on-

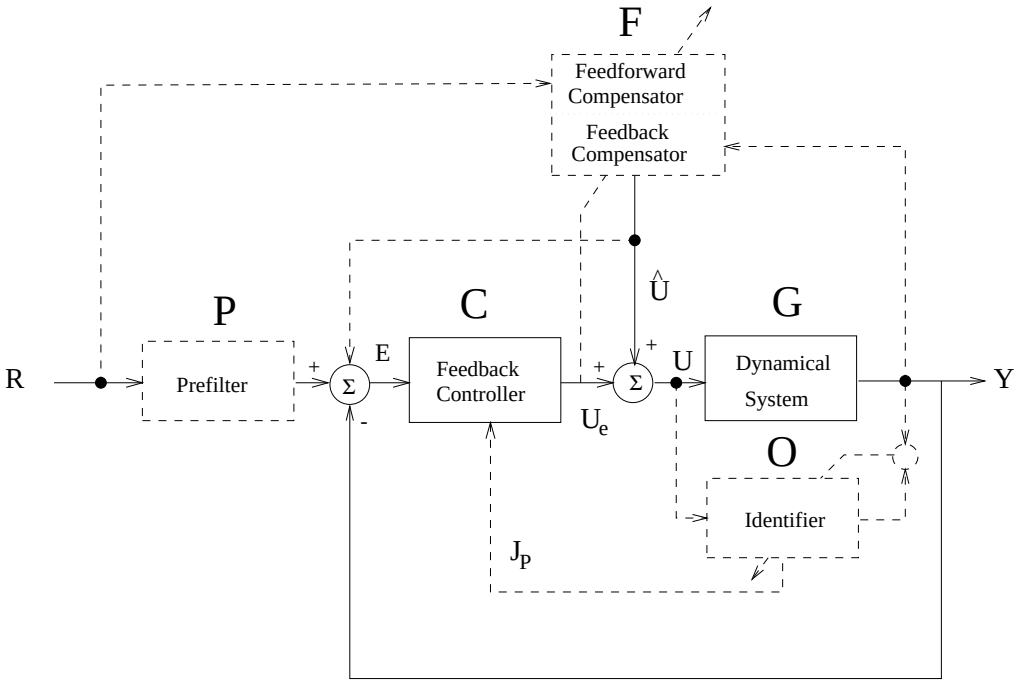


Figure 3.1: Basic Control Structure

line. The simplest method of approximating the Jacobian is the direct calculation from exact dynamic equations if available. Finite difference approximation is based on changes between control input $\boldsymbol{U}$ and system output $\boldsymbol{Y}$[15]. Unfortunately, this method can not assure the stability because the dynamical system is not always stable under non-zero constant control input signal. In this configuration, it has been shown that the stability of the system becomes very sensitive to choices of leaning rate and initial weights of NN. Another approximation of Jacobian can be achieved by an NN as shown in Figure 3.1 when $\boldsymbol{P = I, F = 0,}$ and $\boldsymbol{C = O = N}$ [23]. This method is also unreliable for multi-input multi-output complex nonlinear system such as a robot manipulator because the NN controller($\boldsymbol{C}$) and the NN identifier($\boldsymbol{O}$) can have

different convergence rates so that the NN identifier can provide wrong estimated Jacobian($\boldsymbol{J_p}$) to the NN controller. This fails on-line control of the system. As a modified version of the direct inverse NN control, an adjustable gain layer method has been proposed[54]. One gain layer is added between the NN controller and the plant to adjust approximated Jacobian. In this technique, linearization of the system, small update rate and small initial weights must be assured to maintain stability at the initial step. Thus this is not suitable for the application where fast convergence in real time is needed such as an industrial robot manipulator.

Thus in on-line inverse NN control application the stability of the dynamical system must be assured before applying NN controllers. This has been done simply by employing a feedback controller and NN feedforward compensator ($\boldsymbol{F}$) [31]. This scheme is called feedback error learning(FEL) that provides the solution for Jacobian problem as well as stability at initial process. However, compensating at control input $\boldsymbol{U}$ is very sensitive to feedback controller gains and the performance of NN is degraded. Here, new neural network control schemes are proposed.

3.2 Research Objectives

- The new strategy is to stabilize the system with feedback controllers first and then do inverse control over that closed loop on line. Therefore, the controlled system is guaranteed to be stable with the selection of an appropriate learning rate.

- A new method replaces a prefilter with a neural network to achieve the same goal of that of FEL($\boldsymbol{F} = \boldsymbol{0}, \boldsymbol{O} = \boldsymbol{0}$ and $\boldsymbol{P} = \boldsymbol{N}$) as shown in Figure 3.8. Compensating at reference input provides the superior performance over compensating at torque as well as the unified structure that can be used for the pre-existed control systems.

- The purpose of this chapter is to present the new on-line inverse control scheme that implemented based on a stabilized closed loop system and to investigate the principal difference from FEL structure.

- The purpose of this chapter is to see the feasibility of implementing this technique for real time control.

3.3 System Output Error and System Model Error

Two basic errors, system output error and system model error, are considered to be minimized according to their control structure and implementation as shown in Figure 3.2. The system output error defined as $\boldsymbol{\varepsilon} = \boldsymbol{q}_d - \boldsymbol{q}$ between desired and actual trajectories minimizes the output error directly, while the system model error $\boldsymbol{\tau}_e = \boldsymbol{\tau} - \hat{\boldsymbol{\tau}}$, between actual and estimated torques, minimizes the output error indirectly. Different objective functions formed from different errors dictate the differences in learning rule and training signal, but the same goal can be achieved as

$\boldsymbol{q_d} \cong \boldsymbol{q}$. Defining the objective function from $\boldsymbol{\varepsilon}$ as

$$\mathcal{J}_\varepsilon = \frac{1}{2}\varepsilon^T\varepsilon \tag{3.1}$$

In order to apply back-propagation algorithm, the gradient of the objective function with respect to weight is required. The gradient is obtained by differentiating (3.1)

$$\frac{\partial \mathcal{J}_\varepsilon}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{q}^T}{\partial \boldsymbol{w}}\varepsilon \tag{3.2}$$

Since the gradient $\frac{\partial q^T}{\partial w}$ is not available, (3.2) can be expanded by chain-rule as

$$\frac{\partial \mathcal{J}_\varepsilon}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{\tau}^T}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{q}^T}{\partial \boldsymbol{\tau}^T}\varepsilon = -\frac{\partial \boldsymbol{\tau}^T}{\partial \boldsymbol{w}}\boldsymbol{J}_p^T\varepsilon \tag{3.3}$$

where $\frac{\partial \tau^T}{\partial w}$ is always available and $\frac{\partial q^T}{\partial \tau^T}$ needs to be estimated. We define $\boldsymbol{J_p} = \frac{\partial q}{\partial \tau}$ as the dynamic Jacobian of robot manipulator. This Jacobian is not easy to achieve in practice and several solutions have been proposed based on numerical estimation [15] or neural estimation [23, 24]. Their performances on complex nonlinear multi-input multi output dynamical system are yet to be successful. Since this is one difficulty of minimizing system output error directly many NN control structures proposed are designed to avoid estimation of Jacobian.

An alternative solution is to avoid the Jacobian estimation by minimizing the objective function formed differently. Another objective function can be formed based on system model error as shown in Figure 3.2(b). Instead of minimizing system output error, the same goal can be achieved by minimizing model error as well. In this case the Jacobian estimation is not necessary. Define the objective function as

$$\mathcal{J}_\tau = \frac{1}{2}\tau_e^T\tau_e \tag{3.4}$$

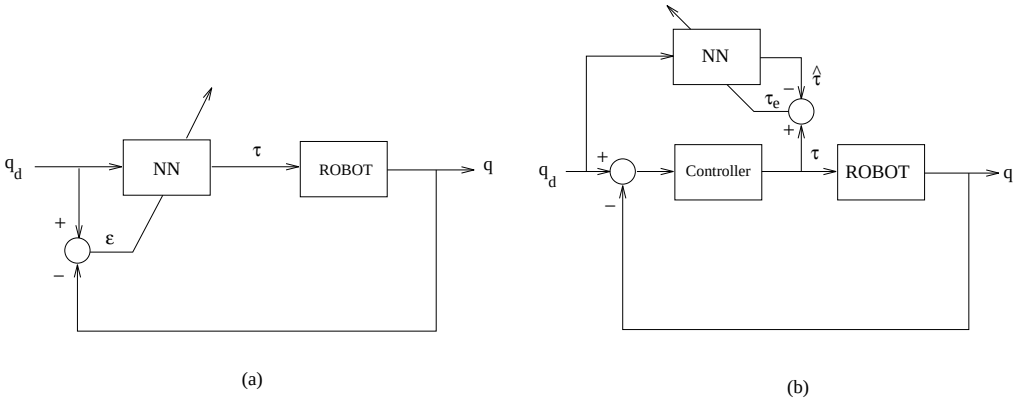


Figure 3.2: (a) System output error and (b) system model error

where $\boldsymbol{\tau_e} = \boldsymbol{\tau} - \hat{\boldsymbol{\tau}}$, $\hat{\boldsymbol{\tau}}$ is estimated output from FFNN controller and $\boldsymbol{\tau}$ is actual output of the system. Similarly, the gradient of (3.4) is

$$\frac{\partial \mathcal{J}_\tau}{\partial \boldsymbol{w}} = -\frac{\partial \hat{\boldsymbol{\tau}}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} \boldsymbol{\tau_e} \tag{3.5}$$

Since $\hat{\boldsymbol{\tau}} = \boldsymbol{f}(\boldsymbol{w})$ and the term $\frac{\partial \hat{\tau}^T}{\partial \boldsymbol{w}}$ is always available Jacobian is not necessary.

3.4 Neural Identifier

Neural identifier is to identify the plant in forward or inverse manners with respect to input and output configuration with the dynamical systems. The goal of the forward neural identifier is to copy the plant, $\boldsymbol{P}$, into neural network, $\boldsymbol{N}(\boldsymbol{\tau}, \boldsymbol{w})$, such that $\boldsymbol{N}(\boldsymbol{\tau}, \boldsymbol{w}) \approx \hat{\boldsymbol{P}}$ after convergence. The inputs to the neural identifier is same as inputs to the plant as shown in Figure 3.3(a). If the training set of inputs are whole universal set and infinite neurons are available, then it is possible to identify the plant within some accuracy such that $||\boldsymbol{P} - \boldsymbol{N}(\boldsymbol{\tau}, \boldsymbol{w})|| < \boldsymbol{\epsilon}$ with respect to any

bounded inputs [63, 64]. In real world, however, since it is not allowed we always have the approximate plant, $\hat{\boldsymbol{P}}$ which has been partially identified in operating point.

In the same way, the goal of inverse neural identifier shown in Figure 3.3(b) is to copy the inverse of a plant, $\boldsymbol{P^{-1}}$, into neural network, $\boldsymbol{N(q,w)}$, so that $\boldsymbol{N(q,w)} \approx \hat{\boldsymbol{P}}^{-1}$ after convergence. In control system, it is desired to use the inverse neural identifier as a controller after identifying the plant by putting it in front of the plant. Therefore, the output of the plant always tracks the reference input. If the plant is linear this can be done easily by adaptive control theory (i.e. $\boldsymbol{NP} = \boldsymbol{I}$ since $\boldsymbol{N} = \hat{\boldsymbol{P}}^{-1}$ where I is an identity matrix). However, since the robot manipulator and the inverse neural identifier are nonlinear systems. they are not commutable each other at all. Copying inversely identified neural identifier to a neural controller is hardly valid for a nonlinear plant. In order to identify the robot or system properly, we borrow the concept of the conventional adaptive control theory. One method uses delayed state variables (or previous state variables) as well as the present state variables as inputs to the neural identifier so that NN identifier is a dynamic system which can be represented as differential equations [19, 23]. Another identification method is to add noise to input and output measurement of the plant [76].

For both forward and inverse identification, let us define the objective function to be minimized as follows:

$$\mathcal{J} = \frac{1}{2} e(t)^T e(t) \tag{3.6}$$

where $\boldsymbol{e(t)} = \boldsymbol{d(t)} - \boldsymbol{y(t)}$, $\boldsymbol{d(t)} = \boldsymbol{f(x(t))}$ and $\boldsymbol{y(t)} = \boldsymbol{f(x(t), x(t-1), d(t-}$

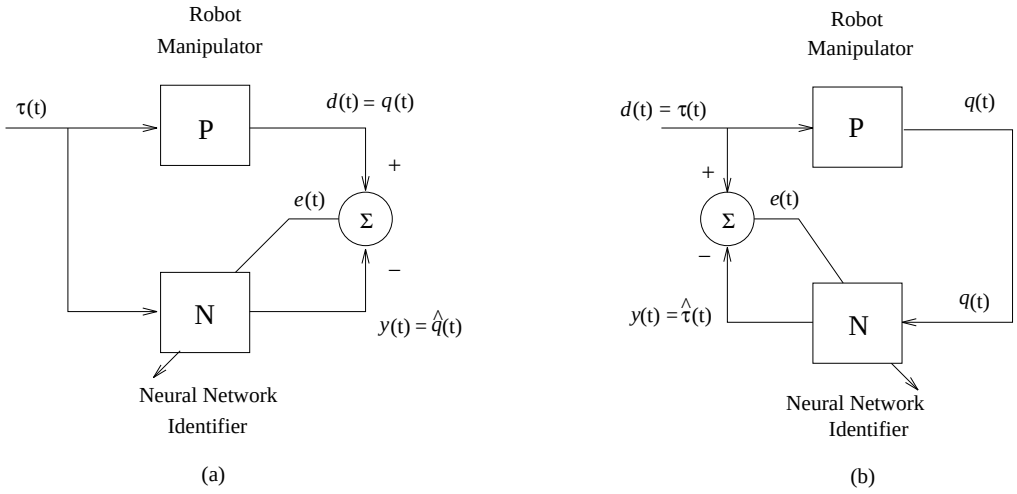


Figure 3.3: Identification of a nonlinear plant(robot) (a) forward (b) Inverse

$\mathbf{1)}, \boldsymbol{w})$. From now on, $\boldsymbol{d(t)}$ or $\boldsymbol{y(t)}$ are used as $\boldsymbol{d}, \boldsymbol{y}$ interchangeably. Differentiating (3.6) to get the gradient of the objective function

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{e}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} \boldsymbol{e} = -\frac{\partial \boldsymbol{y}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} \boldsymbol{e} \tag{3.7}$$

Therefore, the weights of neural network can be updated to minimize the objective function since $\frac{\partial \boldsymbol{y}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}$ can be computed.

3.5 Neural Controller

The basic structure of neural controller is the direct inverse neural controller or specialized learning structure as shown in Figure 3.4. This control structure is very simple but needs the Jacobian of the plant when error back-propagation algorithm is used. There are four ways to support this Jacobian informations : $\boldsymbol{i})$ direct calculation from dynamic equation of a plant if they are known $\boldsymbol{ii})$ estimation of dynamic

Jacobian by finite difference approximation method from input and output relationship ***iii***) using PD feedback controller as a teaching signal to avoid the requirement of estimating dynamic Jacobian ***iv***) estimation of dynamic Jacobian by another neural network by identifying the forward model of a plant. The following subsections deal with each way of getting dynamic Jacobian.

3.5.1 Direct Inverse Neural Controller

In most control system applications, it is desirable to place neural network in front of plant so that it can control the plant directly as shown in Figure 3.4. This scheme is called the direct neural control or specialized learning scheme which on-line control is possible [15]. The internal weights of the neural controller are updated to minimize the error between desired and actual position. Since the plant (a robot manipulator) is located between the controller and the output and generalized delta rule is used in neural controller, we need the system Jacobian of a robot manipulator.

The objective function can be represented as

$$\mathcal{J} = \frac{1}{2}\boldsymbol{\varepsilon}^{T}\boldsymbol{\varepsilon} \tag{3.8}$$

Then, using chain rules, we get the gradient of the objective function as follows:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{\varepsilon}^{T}}{\partial \boldsymbol{w}}\boldsymbol{\varepsilon} = -\frac{\partial \boldsymbol{\tau}^{T}}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{q}^{T}}{\partial \boldsymbol{\tau}^{T}}\boldsymbol{\varepsilon} \tag{3.9}$$

where $\frac{\partial \boldsymbol{\tau}^{T}}{\partial \boldsymbol{w}}$ is generally known since $\boldsymbol{\tau} = \boldsymbol{f}(\boldsymbol{q}_d, \boldsymbol{w})$. If the plant is known, then the dynamic Jacobian of plant, $\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\tau}}$, can be expressed analytically and its value can be

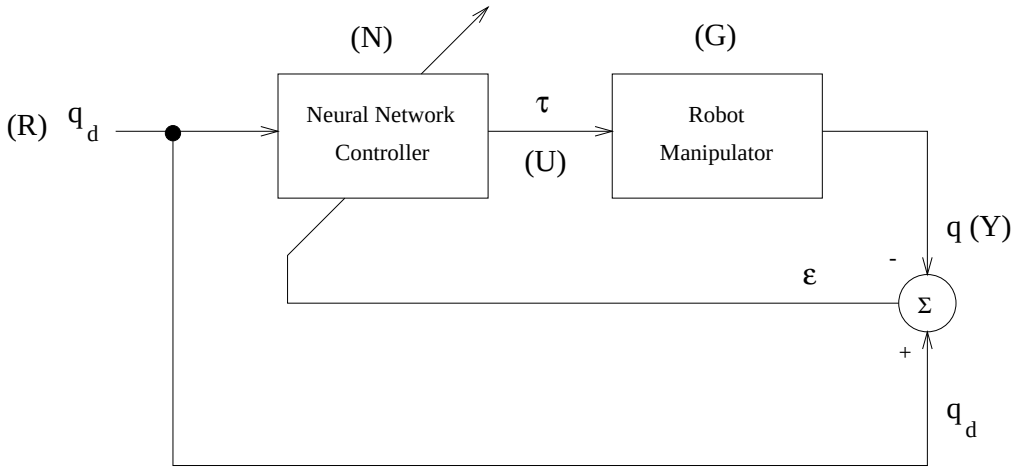


Figure 3.4: Direct Inverse Neural Controller

computed. But most of the cases, the plant is unknown, then we need to compute $\frac{\partial q}{\partial \tau}$. This can be done numerically from finite difference approximation $\frac{\partial q}{\partial \tau} = \frac{\Delta q}{\Delta \tau}$ on line [15].

$$\frac{\Delta q}{\Delta \tau} = \frac{q(t) - q(t-1)}{\tau(t) - \tau(t-1)} \tag{3.10}$$

Therefore, this scheme can be used without a model of the plant. In this scheme, however, there is no guarantee to satisfy stability and to perform desired response because the robot dynamics is not always stable under non zero constant input torque (i.e. if a step torque is applied at a joint which has a magnitude greater than the gravity force, then a link can rotate with increasing angular velocity). Yabuta and Yamada [40, 35] showed that the back-propagation method in direct inverse neural controller cannot guarantee the stability since it is very much depends on the learning rate and initial weights. It is not easy to find right parameters by trial and error basis. Hence, it is better to stabilize the robot with PD feedback control plus grav-

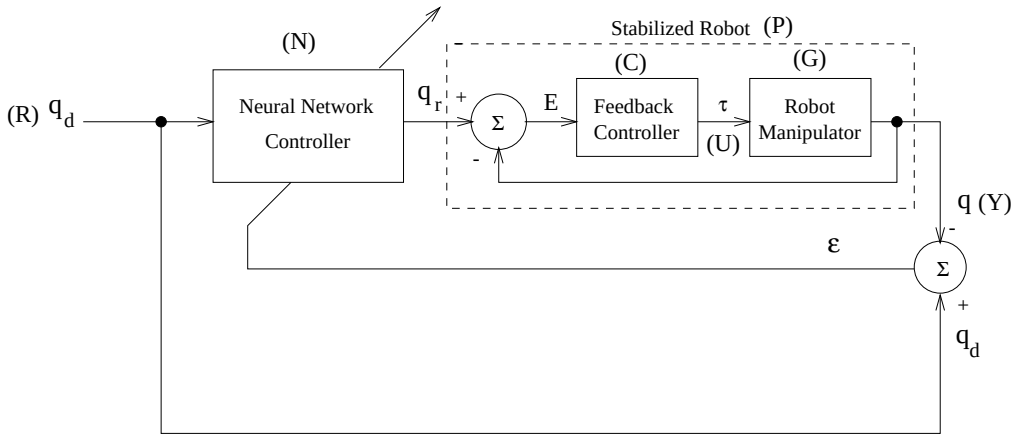


Figure 3.5: Direct Inverse Neural Control Structure with Stabilized System

ity compensation(always possible) or PD feedback controller only first before NN is trained [68]. Figure 3.5 shows the proposed direct inverse control structure with a stabilized robot (called a plant). In this proposed structure the learning can be done by either defining Jacobian estimated from ideal differential closed loop equation [66] or neglecting Jacobian. The proposed control structure has several advantages over other NN control structures: First, the performance is better. Second, implantation outside can be done without modifying the internal control structure. Third, it has the robustness to feedback gain variations. These advantages will be proven in later chapters by computer simulations.

3.5.2 Indirect Inverse Neural Controller

The dynamic Jacobian can be obtained from neural identifier by forming input and output configuration for forward identification. In first phase, it is training off-

line by uniformly distributed random data set and then the neural network is trained on-line to compensate for any changes in dynamic Jacobian of the plant. The indirect neural adaptive control uses two neural networks : a neural identifier and a neural controller as shown in Figure 3.6. The purpose of a neural identifier is to provide the estimation of dynamic Jacobian of the plant to the neural controller so that a neural controller can generate proper control input signal without any feedback loop.

$$\mathcal{J} = \frac{1}{2}\varepsilon^T\varepsilon \tag{3.11}$$

Then, using chain rules, we get the gradient of the performance index as follows:

$$\frac{\partial \mathcal{J}}{\partial w} = -\frac{\partial q^T}{\partial w}\varepsilon = -\frac{\partial \tau^T}{\partial w}\frac{\partial q^T}{\partial \tau^T}\varepsilon \tag{3.12}$$

where $\frac{\partial \tau^T}{\partial w}$ is generally known. The neural identifier provides the dynamic Jacobian value, $\frac{\partial q}{\partial \tau}$, to the neural controller.

For on-line control, it is suggested that using different time sampling intervals for a identifier and a controller such that the neural identifier is trained and updated at faster sampling time than a neural controller. But this strategy has unpredictable results in nonlinear systems. So, it may still have convergence problems overall because it takes time for the plant model to converge before the controller NN can be properly converged. To avoid this difficulty, experience in linear system problems has suggested to identify the plant first, and then followed by the control action. In identification phase, the plant is to be identified to a certain desired convergent accuracy with a uniformly distributed random input. In control phase, control action took place. As

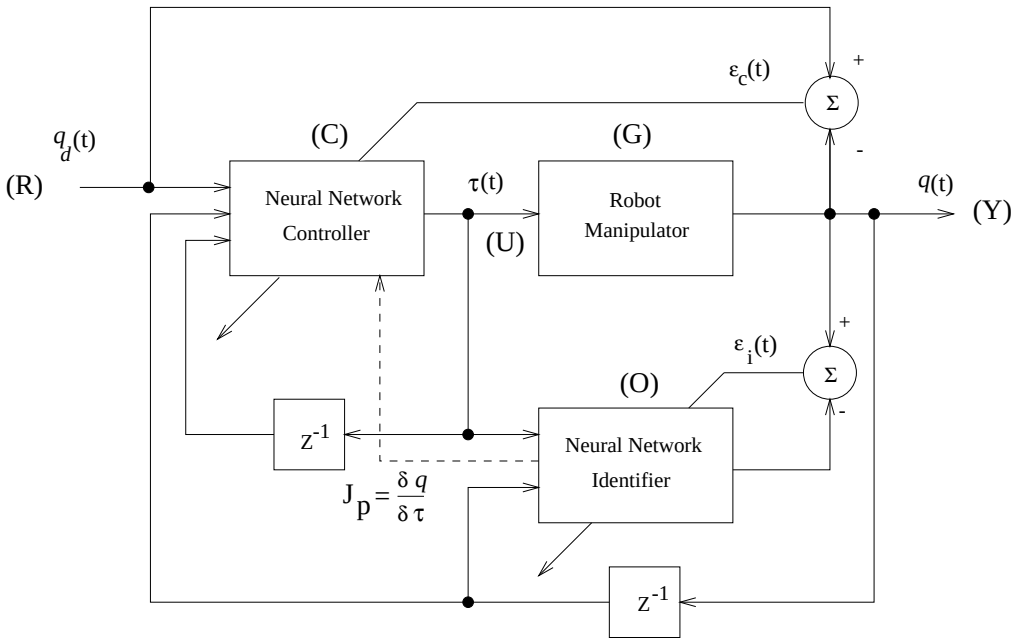


Figure 3.6: Indirect Inverse Control Structure

shown in Figure 3.6, the plant model must be trained with $\boldsymbol{q(t)}$ compatible with the intended set $\boldsymbol{q_d(t)}$. In general, the control input $\boldsymbol{q_d(t)}$ is difficult to know in advance before the controller NN is known. Therefore if the plant model is first trained based on some training set of $\boldsymbol{\tau(t)}$ which is not necessarily related to $\boldsymbol{q_d(t)}$, then the model is not going to provide the correct dynamic Jacobian values for different $\boldsymbol{\tau(t)}$ during the training of the controller. In this scheme, to model the plant using another NN and evaluate the dynamic Jacobian, $\frac{\partial q}{\partial \tau}$, from the model is too complicated and unreliable for a complex multi-input multi-output nonlinear system.

3.5.3 Feedback Error Learning Controller

Figure 3.7 shows that a direct inverse neural control structure with a feedback controller, so called a feedback error learning structure [31]. Unlike previous schemes this scheme uses the feedback command as the error signal to be minimized in inverse dynamic model so that the back-propagation of the error through controlled system is not necessary (i.e. avoid to calculate the Jacobian). Another advantage is that on-line control is possible which is very important subject in real time control system. This control structure is same as Kawato's feedback error learning structure except that the general neural network is used instead of specific neural network having priori knowledge of dynamic components as subsystems. The structure of this scheme is more likely neural control and simpler than Kawato's feedback error learning method, which can be called the nonlinear adaptive controller because their coefficients of subsystems are only adapted. In order to avoid the dynamic Jacobian, the objective function is defined differently as a system model error.

The control input signal $\boldsymbol{U}$ is

$$\boldsymbol{U} = \boldsymbol{U}_e + \hat{\boldsymbol{U}} \tag{3.13}$$

where $\hat{\boldsymbol{U}}$ is output of NN and $\boldsymbol{U}_e$ is feedback error signal. Ideally, $\boldsymbol{U}_e = \boldsymbol{0}$ is desired. Then, minimizing $\boldsymbol{U}_e = \boldsymbol{U} - \hat{\boldsymbol{U}}$ yields

$$\hat{\boldsymbol{U}} = \boldsymbol{U} \tag{3.14}$$

so that NN is an inverse model of the dynamical system. The feedback error signal,

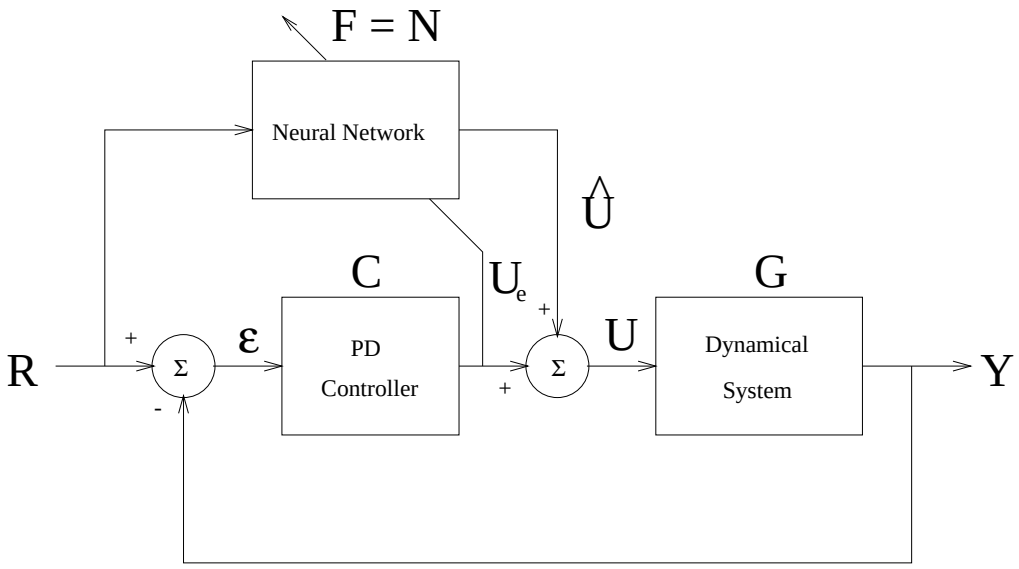


Figure 3.7: FEL Structure (FEL I)

$\boldsymbol{U_e}$, is used to adjust internal weights of neural network.

A feed forward neural network combined with a PD feedback controller provides the stability over a direct inverse control scheme in the previous section because the proportional and derivative (PD) feedback controller plus gravity compensation stabilizes the plant at the beginning of control action [2]. The recent research proved that only PD control can stabilize the system [77]. And afterward, control is transferred from a PD feedback controller to a feed forward neural controller.

The main disadvantage of this scheme is that there is lack of priori knowledge of robot dynamic so that the closed loop performance may be affected by selecting feedback gains which depends on the inertia matrix, $\boldsymbol{D(q)}$. If gains are selected such that the robot is stable but not asymptotically stable then it may not converge or the

speed of convergence may be very slow. In general, high feedback gains are desired for better tracking performance and robustness because it de-centralizes the system more completely, reduces the nonlinearity effect and the noise effect if the closed loop system is stable [78].

Since the inverse dynamic model is trained with certain set of input other than whole universal input sets, the internal weights in neural network should be retrained each time whenever the task is changed. It is not even certain that the neural network represents the exact inverse of the robot manipulator after training [42]. If the robot is designed to do repetitive works all the time, it is true in most practical cases, this will not be a disadvantage at all. But, for general cases, the neural compensator structure is introduced as a substitute solution for the inverse control. In chapter 5, non-model based NN control section, the detailed studies of this scheme will be reexamined.

3.5.4 Reference Compensation Technique(RCT)

Figure 3.8 shows the proposed reference compensation technique. Compensating at reference input provides the better performance as well as the simpler control structure. From Figure 3.8, the control input signal can be derived as

$$\boldsymbol{U} = \boldsymbol{C}\boldsymbol{E} \tag{3.15}$$

where $\boldsymbol{E} = \boldsymbol{R_c} - \boldsymbol{Y}$.

Substituting $\boldsymbol{E} = \boldsymbol{R} + \boldsymbol{\Phi} - \boldsymbol{Y}$ into (3.15) and defining $\boldsymbol{v} = \boldsymbol{C\varepsilon}$ yields

$$\boldsymbol{v} = \boldsymbol{C\varepsilon} = \boldsymbol{U} - \boldsymbol{C\Phi} \tag{3.16}$$

where $\boldsymbol{\varepsilon} = \boldsymbol{R} - \boldsymbol{Y}$ and $\boldsymbol{\Phi}$ is output of NN. Compensating signals are amplified through primary feedback controller gains so that the magnitudes of compensating signals from NN are expected to be very small compared with those of the FEL controller which compensates at control input signal. At convergence $\boldsymbol{\varepsilon} = \boldsymbol{0}$, when inverse identification is done, the ideal compensation of NN is same as

$$\boldsymbol{\Phi} = \boldsymbol{C^{-1}U} \tag{3.17}$$

which is much smaller in magnitude than $\hat{\boldsymbol{U}}$ in (3.14). Hence comparing with the FEL controller, the performance(accuracy and convergence) of the proposed scheme is better not only because the dimensionality of searching degrees-of-freedom(DOF) for optimal value is larger than the FEL's($\boldsymbol{\phi_p}$ and $\boldsymbol{\phi_d}$ instead of $\hat{\boldsymbol{U}}$) but also because the magnitude of disturbances to be compensated is reduced by $\boldsymbol{C^{-1}}$. Above all the real advantage of the proposed RCT over the FEL scheme is the simpler practical implementation. Functioning as a prefilter, compensation can be done outside of control loop without modifying the internal controller structure, while compensation of the FEL controller is done inside control loop. NN compensation outside control loop also allows robustness to controller gain change so that the performance under low primary controller gains is excellent.

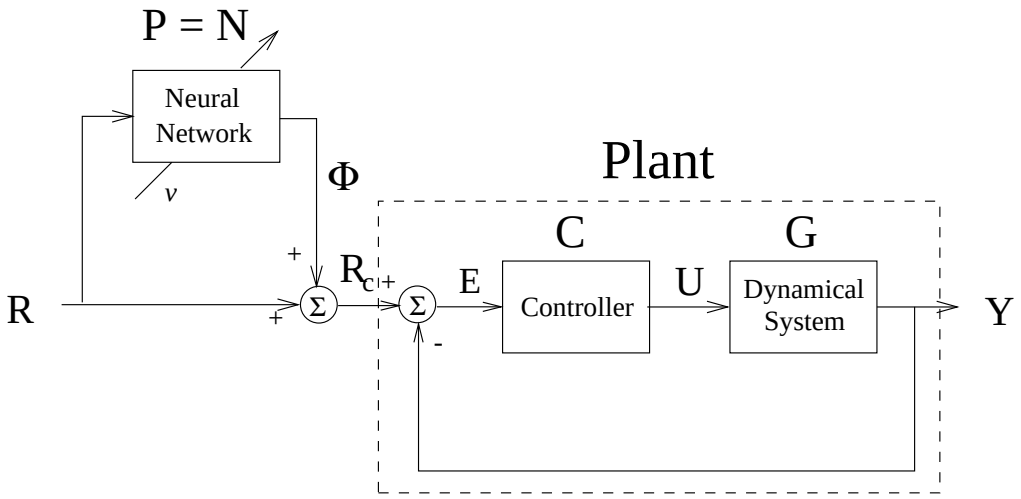


Figure 3.8: The Proposed RCT Neural Control Structure

The objective function $\mathcal{J}$ is a quadratic function of the unified training signal $\boldsymbol{v}$.

$$\mathcal{J} = \frac{1}{2}\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v} \tag{3.18}$$

where $\boldsymbol{v} = \boldsymbol{C\varepsilon}$ from (3.16). Differentiating equation(3.18) and making use of (3.16) yields the gradient of $\boldsymbol{E}$ as follows:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{v} = -\frac{\partial \boldsymbol{\Phi}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{C}^{\boldsymbol{T}}\boldsymbol{v} \tag{3.19}$$

since $\frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{\Phi}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{C}^{\boldsymbol{T}}$.

3.6 Structural comparison between FEL and RCT

The structural difference between FEL and RCT affects the performance of NN controllers by allowing the different amount of works allocated for NN in the control structure and the different sensitivity to the feedback controller. Here we are investigating two aspects : compensation and sensitivity.

Compensation for uncertainties in system $\boldsymbol{G}$

We assume that the linear system representation from input to output

$$\boldsymbol{Y} = \frac{\boldsymbol{GC}}{\boldsymbol{1+GC}}\boldsymbol{R} \tag{3.20}$$

from Figure 3.8 is an ideal system such that the controller is selected to satisfy the desired performance. If there are uncertainties such as $\boldsymbol{\Delta G}$ in the system $\boldsymbol{G}$, then the equation (3.20) is changed to

$$\hat{\boldsymbol{Y}} = \frac{(\boldsymbol{G}+\boldsymbol{\Delta G})\boldsymbol{C}}{\boldsymbol{1}+(\boldsymbol{G}+\boldsymbol{\Delta G})\boldsymbol{C}}\boldsymbol{R} \tag{3.21}$$

In order to compensate those uncertainties in $\boldsymbol{G}$ we use NN to compensate those uncertainties for two different NN control structures : FEL and RCT. For FEL, the output response due to input with the compensator $\boldsymbol{F}$ can be represented as follows:

$$\hat{\boldsymbol{Y}} = \frac{(\boldsymbol{G}+\boldsymbol{\Delta G})(\boldsymbol{C}+\boldsymbol{F})}{\boldsymbol{1}+(\boldsymbol{G}+\boldsymbol{\Delta G})\boldsymbol{C}}\boldsymbol{R} \tag{3.22}$$

Equating (3.22) with (3.20) and arranging for $\boldsymbol{F}$ yields

$$\boldsymbol{F} = \frac{\boldsymbol{T}(\boldsymbol{1}+(\boldsymbol{G}+\boldsymbol{\Delta G})\boldsymbol{C}) - \boldsymbol{C}(\boldsymbol{G}+\boldsymbol{\Delta G})}{\boldsymbol{G}+\boldsymbol{\Delta G}} \tag{3.23}$$

where $\boldsymbol{T} = \frac{\boldsymbol{GC}}{\boldsymbol{1+GC}}$. Therefore the NN compensator $\boldsymbol{F}$ is required to compensate those amount of (3.23) in order for the closed loop system to perform as desired as (3.20).

Similarly for the proposed RCT structure we have

$$\hat{\boldsymbol{Y}} = \frac{(\boldsymbol{G}+\boldsymbol{\Delta G})\boldsymbol{C}(\boldsymbol{P}+\boldsymbol{1})}{\boldsymbol{1}+(\boldsymbol{G}+\boldsymbol{\Delta G})\boldsymbol{C}}\boldsymbol{R} \tag{3.24}$$

Equating (3.24) with (3.20) and arranging for $\boldsymbol{P}$ yields

$$\boldsymbol{P} = \frac{\boldsymbol{T}(\mathbf{1} + (\boldsymbol{G} + \boldsymbol{\Delta G})\boldsymbol{C}) - \boldsymbol{C}(\boldsymbol{G} + \boldsymbol{\Delta G})}{\boldsymbol{C}(\boldsymbol{G} + \boldsymbol{\Delta G})} \tag{3.25}$$

The compensated term by the FEL compensator $\boldsymbol{F}$ is larger than that of the proposed one $\boldsymbol{P}$ by $\boldsymbol{C}$ so that the relationship $\boldsymbol{P} = \boldsymbol{C^{-1}F}$ holds. Therefore the NN of the FEL is required to compensate larger magnitude. If $\boldsymbol{P}$ and $\boldsymbol{F}$ controllers are the same and has the capability of compensating certain rate of partial uncertainties, their effects on the output $\boldsymbol{Y}$ in each structure are different. Obviously we immediately see the effect of $\boldsymbol{P}$ controller is much smaller than that of the $\boldsymbol{F}$ controller. Therefore, the performance of the RCT scheme is better when the same neural networks with a certain accuracy are given.

System sensitivity to controller $\boldsymbol{C}$

One way of checking the robustness of the system is the system sensitivity due to parameter perturbation. We like to see the robustness of the system by investigating the relationship between the compensator $\boldsymbol{F}$ or $\boldsymbol{P}$ and the controller $\boldsymbol{C}$ with small perturbation in $\boldsymbol{C}$. The system sensitivity due to small perturbation in the controller $\boldsymbol{C}$ is defined in [79] as

$$\boldsymbol{S_C^T} = \frac{\frac{\partial \boldsymbol{T}}{\boldsymbol{T}}}{\frac{\partial \boldsymbol{C}}{\boldsymbol{C}}} = \frac{\partial \boldsymbol{T}}{\partial \boldsymbol{C}} \frac{\boldsymbol{C}}{\boldsymbol{T}} \tag{3.26}$$

where $\boldsymbol{T}$ is the transfer function of the system. For FEL scheme in Figure 3.7 we have

$$\boldsymbol{S_C^T} = \frac{(\mathbf{1} - \boldsymbol{GF})\boldsymbol{C}}{(\mathbf{1} + \boldsymbol{GC})(\boldsymbol{C} + \boldsymbol{F})} \tag{3.27}$$

The system sensitivity due to controller perturbation is very sensitive because it depends on not only the system itself but also the compensator $\boldsymbol{F}$. Hence system sensitivity due to the controller perturbation is also effected by the NN compensator if $\boldsymbol{F}$ is a neural network.

Similarly for the proposed RCT scheme in Figure 3.8 the system sensitivity due to the controller perturbation is obtained as

$$\boldsymbol{S_C^T} = \frac{\mathbf{1}}{(\mathbf{1} + \boldsymbol{GC})} \tag{3.28}$$

which is only dependent upon the system not on the compensator $\boldsymbol{P}$.

Borrowing the concept of the linear control theory we may expect the same behavior for the nonlinear control with neural network compensators $\boldsymbol{P}$ and $\boldsymbol{F}$. The analysis shows that there are two advantages of the RCT over the FEL which are less work load of neural network and robustness due to the controller perturbation. Therefore, the proposed NN RCT has the better structure such that NN controller not only performs better but is robust to controller gain variations. In addition, the most important advantage of the RCT is not modifying the internal control structure to implement but compensating outside the controller. This advantage can be applied to most of the control systems that have been operating, but needs compensation for uncertainties. The performances of two control structures due to the controller variations are shown in simulation section.

Chapter 4

Model Based Neural Network Robot Position Control

4.1 Introduction

In previous chapter, we studied different neural network control structures for generic control applications. Extensive investigations have been carried out to design NN controllers for robot manipulators as well [31, 43, 44, 42, 36, 32, 30, 48]. In this chapter, model based neural control structures for robot manipulators are implemented. Accordingly, various design schemes have been proposed and compared with other neural control schemes. The differences in these schemes are largely in the role that NN is playing in the control system and the way it is trained for achieving desired trajectory tracking performance.

The most popular control scheme is one which uses NN to generate auxiliary joint control torques to compensate for the uncertainties in the computed-torque based primary robot controller that is designed based on a nominal robot dynamic model. This is accomplished by implementing the NN controller in either a feedforward or a feedback configuration, and the NN is trained on-line as shown in Figure 4.1. Satisfactory results of this approach have been reported in the literature[31, 43, 30]. Thus, depending on the amount of a priori knowledge of the robot model that is available, the NN controller would be required to compensate for a wide range of model uncertainties. Another approach that has been tried is to use the NN as the computed-torque controller where it is trained to follow an accurate estimate of the robot model as shown in Figure 4.2 [16, 44]. Implementation of this approach requires off-line training of the NN based on a nominal robot model, and then followed

by on-line training to match the actual robot dynamics [44, 42].

One major obstacle in carrying out on-line training is the requirement of knowing the robot system Jacobian in order to apply the error back-propagation algorithm for updating the internal weights. Such Jacobian is difficult to determine and compute in practice, and various solutions have been proposed [15, 16, 23, 66]. The success of this control approach is yet to be demonstrated.

Here we propose two basic NN techniques, auxiliary control and inverse control, whose objective functions are differently formed. Thus, depending on the objective function to be minimized, different learning algorithms and training signals have been developed. The objective function for auxiliary control is composed of system model errors and the objective function for inverse control is composed of system output error ε. The gradient of the objective function for inverse control requires the information of system Jacobian to apply back-propagation algorithm correctly while the gradient of the objective function for auxiliary control avoids the system Jacobian.

We also propose one of inverse control approach called the reference compensation technique(RCT) that does not require any Jacobian estimations. Performance of the RCT is excellent.

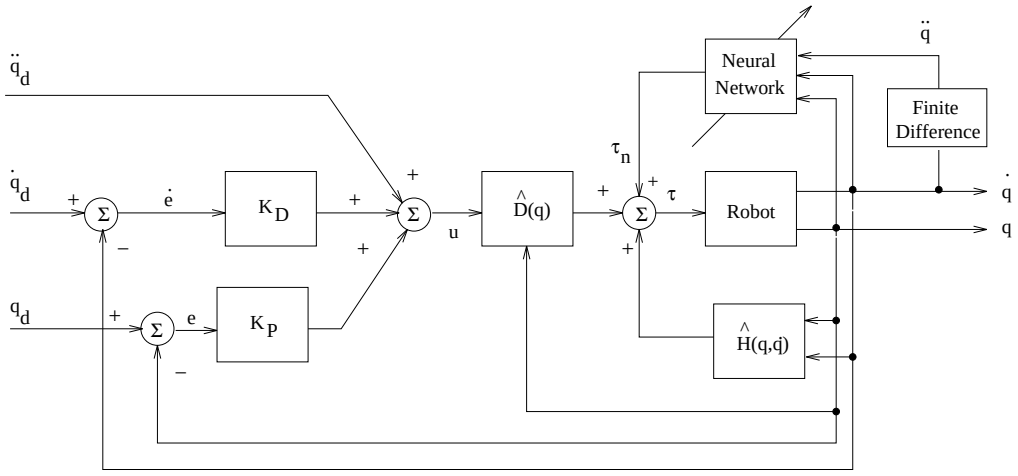


Figure 4.1: On-Line Torque Compensation Scheme

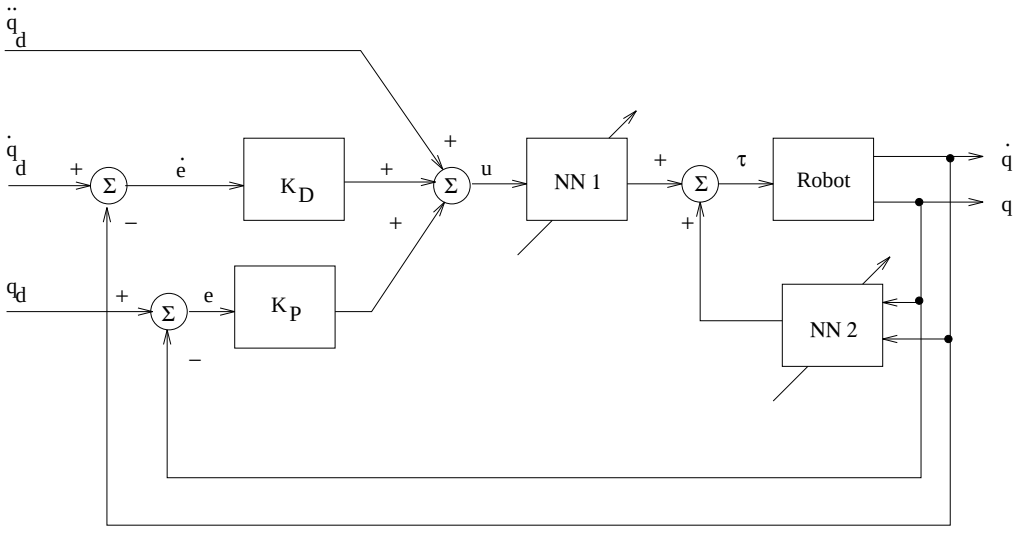


Figure 4.2: Off-Line Learning and On-Line Modification Scheme

4.2 Robot Dynamic Model

The dynamic equation of an $\boldsymbol{n}$ degrees-of-freedom manipulator in joint space coordinates are given by :

$$\boldsymbol{D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + \tau_f(\dot{q}) = \tau} \tag{4.1}$$

where the vectors $\boldsymbol{q, \dot{q}, \ddot{q}}$ are the joint angle, the joint angular velocity, and the joint angular acceleration, respectively; $\boldsymbol{D(q)}$ is the $\boldsymbol{n}$ x $\boldsymbol{n}$ symmetric positive definite inertia matrix; $\boldsymbol{C(q,\dot{q})}$ is the $\boldsymbol{n}$ x $\mathbf{1}$ vector of Coriolis and centrifugal torques; $\boldsymbol{g(q)}$ is the $\boldsymbol{n}$ x $\mathbf{1}$ gravitational torques; $\boldsymbol{\tau_f(\dot{q})}$ is an $\boldsymbol{n}$ x $\mathbf{1}$ vector representing Coulomb friction and viscous friction forces : $\boldsymbol{\tau_f(\dot{q}) = K_1 sgn(\dot{q}) + K_2\dot{q}}$; and $\boldsymbol{\tau}$ is the $\boldsymbol{n}$ x $\mathbf{1}$ vector of actuator joint torques. For simplicity, let us denote $\boldsymbol{h(q,\dot{q}) = C(q,\dot{q})\dot{q} + g(q)}$ so that (4.1) can be rewritten as

$$\boldsymbol{D(q)\ddot{q} + h(q,\dot{q}) + \tau_f(\dot{q}) = \tau} \tag{4.2}$$

The robot dynamic equation (4.2) represents a highly nonlinear, coupled, and multi-input multi-output system. In most practical cases, the model is not exactly known. Thus only nominal estimates of the model are available for controller design.

4.3 Computed-Torque Control

Computed torque method is the basic approach for robot motion control when nominal robot dynamic model is available. In this case the control law can be written

as

$$\boldsymbol{\tau} = \hat{\boldsymbol{D}}(\boldsymbol{q})\boldsymbol{U} + \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{4.3}$$

where $\hat{\boldsymbol{D}}(\boldsymbol{q})$ and $\hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are estimates of $\boldsymbol{D}(\boldsymbol{q})$ and $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, respectively, and the control input $\boldsymbol{U}(\boldsymbol{t})$ is given by

$$\boldsymbol{U} = \ddot{\boldsymbol{q}}_{\boldsymbol{d}} + \boldsymbol{K}_{\boldsymbol{D}}(\dot{\boldsymbol{q}}_{\boldsymbol{d}} - \dot{\boldsymbol{q}}) + \boldsymbol{K}_{\boldsymbol{P}}(\boldsymbol{q}_{\boldsymbol{d}} - \boldsymbol{q}) \tag{4.4}$$

where $\boldsymbol{K}_{\boldsymbol{P}}$ and $\boldsymbol{K}_{\boldsymbol{D}}$ are $\boldsymbol{n}$ x $\boldsymbol{n}$ symmetric positive definite gain matrices, and $\boldsymbol{q}_{\boldsymbol{d}}$ is the desired joint trajectory. The control system is depicted in Figure 4.3. Combining (4.2),(4.3), and (4.4) yields the closed loop tracking error dynamic equation

$$\ddot{\boldsymbol{E}} + \boldsymbol{K}_{\boldsymbol{D}}\dot{\boldsymbol{E}} + \boldsymbol{K}_{\boldsymbol{P}}\boldsymbol{E} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\tau}_{\boldsymbol{f}}(\dot{\boldsymbol{q}})) \tag{4.5}$$

where $\boldsymbol{\Delta D}(\boldsymbol{q}) = \boldsymbol{D}(\boldsymbol{q}) - \hat{\boldsymbol{D}}(\boldsymbol{q}), \boldsymbol{\Delta h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$, and $\boldsymbol{E} = (\boldsymbol{q}_{\boldsymbol{d}} - \boldsymbol{q})$. For simplicity, $\boldsymbol{D} = \boldsymbol{D}(\boldsymbol{q}), \boldsymbol{h} = \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are used. In the ideal case where $\boldsymbol{\Delta D} = \boldsymbol{\Delta h} = \boldsymbol{0}$, $\boldsymbol{\tau}_{\boldsymbol{f}} = \boldsymbol{0}$, equation (4.5) becomes the following ideal second order error equation, which is denoted as $\boldsymbol{v}$.

$$\boldsymbol{v} = \ddot{\boldsymbol{E}} + \boldsymbol{K}_{\boldsymbol{D}}\dot{\boldsymbol{E}} + \boldsymbol{K}_{\boldsymbol{P}}\boldsymbol{E} = \boldsymbol{0} \tag{4.6}$$

Thus, the goal of computed torque control is to make robot be decoupled and be linearized. However, since there are always uncertainties in the robot dynamic model, the ideal error response (4.6) can not be achieved in general. The actual system performance is governed by (4.5) which is degraded and unpredictable. Thus the computed-torque based control is not robust in practice. To improve robustness, NN

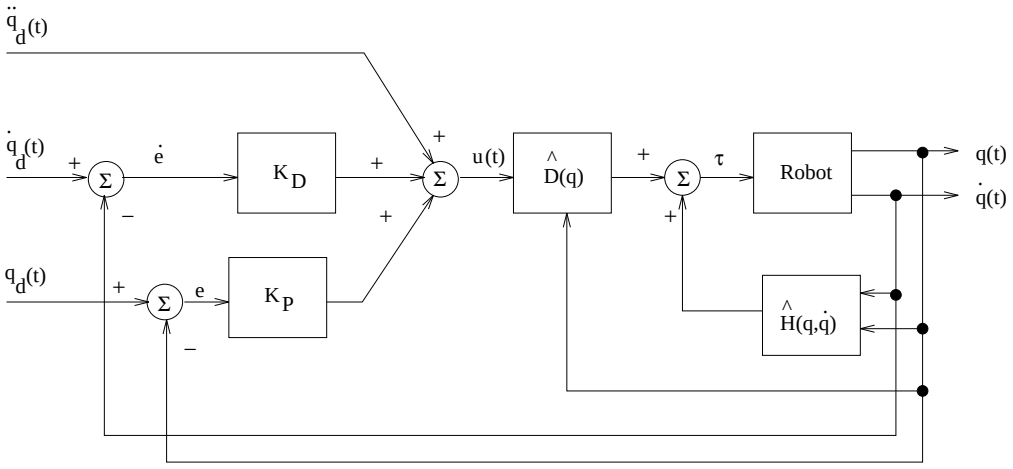


Figure 4.3: Computed Torque Control Structure

controller can be introduced to generate additional joint torques to compensate for the disturbances due to model uncertainties. In the following sections, two NN control techniques including Ishiguro's compensation technique which are conceptually different in compensation method are presented.

4.4 Ishiguro's Torque Level Compensation Scheme

One on-line neural control scheme for robot manipulator is torque level compensation scheme shown in Figure 4.4. This scheme has an objective function that minimizes the error between the actual torque $\boldsymbol{\tau}$ and the torque $\boldsymbol{\tau_m}$ generated from the inverse model of robot manipulator. The implementation of this method is based on the paper [43]. In order to compensate for uncertainties in robot dynamics, the compensated signal from neural network is added to the torque control law signal

such that

$$\boldsymbol{\tau} = \hat{\boldsymbol{D}}\boldsymbol{U} + \hat{\boldsymbol{h}} + \boldsymbol{\tau_n} \tag{4.7}$$

where $\boldsymbol{\tau_n}$ is output of NN. Combining (4.2) and (4.7), we have the error equation as

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f} - \boldsymbol{\tau_n}) \tag{4.8}$$

The way of generating a teaching signal is to use inverse dynamic robot model. Based on model estimation, $\boldsymbol{\tau_m}$, the teaching signal can be obtained between the actual torque and the estimated torque as following

$$\begin{aligned} \boldsymbol{\tau} - \boldsymbol{\tau_m} &= \boldsymbol{\tau_t} \\ &= \boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f} \end{aligned} \tag{4.9}$$

where $\boldsymbol{\tau_t}$ is the teaching signal and $\boldsymbol{\tau_m}$ is the approximated model, $\boldsymbol{\tau_m} = \hat{\boldsymbol{D}}\ddot{\boldsymbol{q}} + \hat{\boldsymbol{h}}$. Thus, the training signal $\boldsymbol{\tau_e}$ becomes $\boldsymbol{\tau_e} = \boldsymbol{\tau_t} - \boldsymbol{\tau_n}$. Ideally, at convergence, $\boldsymbol{\tau_e} = \mathbf{0}$ the output of neural network $\boldsymbol{\tau_n} = \boldsymbol{\tau_t}$ which is same as uncertainties in (4.9).

Therefore, minimizing the error $\boldsymbol{\tau_e}$ between $\boldsymbol{\tau_t}$ and $\boldsymbol{\tau_n}$ eliminates those uncertainty terms. The objective function is represented as :

$$\mathcal{J} = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{\tau_e}^{\boldsymbol{T}}\boldsymbol{\tau_e} \tag{4.10}$$

The internal weight updated law can be written as

$$\boldsymbol{\Delta w} = -\boldsymbol{\eta}\frac{\boldsymbol{\partial}\mathcal{J}}{\boldsymbol{\partial w}} = \boldsymbol{\eta}\frac{\boldsymbol{\partial \tau_n}}{\boldsymbol{\partial w}} \tag{4.11}$$

This gradient can be used in back-propagation learning algorithm.

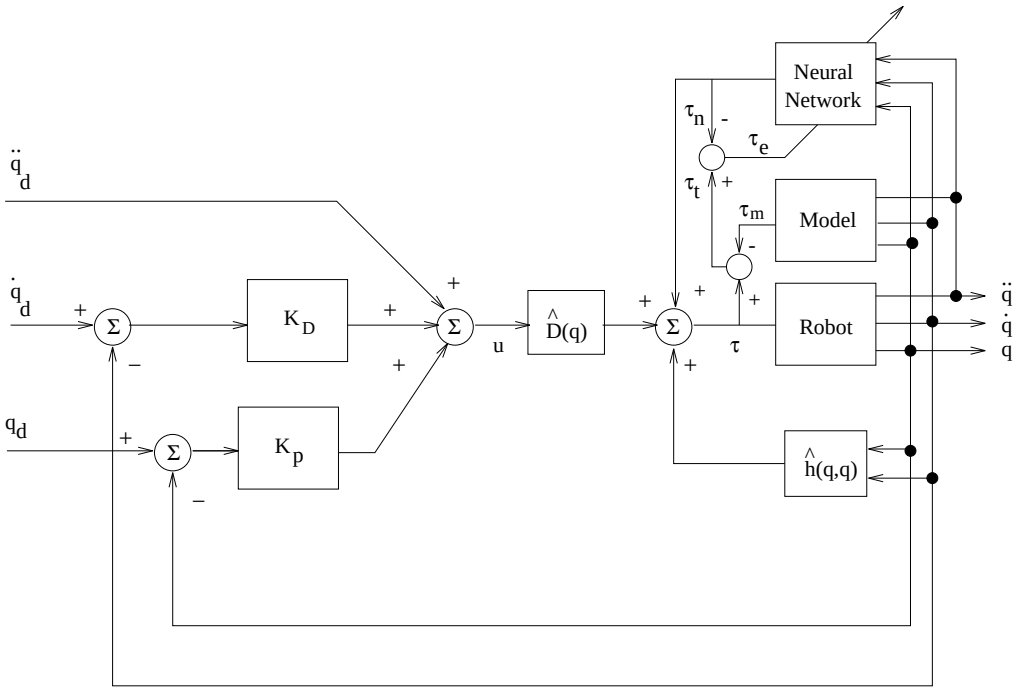


Figure 4.4: Ishiguro ***et. al***'s Neural Compensator Structure

The block diagram of this control scheme is shown in Figure 4.4. Since there is a requirement of model to generate the teaching signal to form learning algorithm the overall structure become complicated and the calculating dynamic model is time consuming. In next section, we are proposing new neural network control scheme that has the same goal, but has the different training signal. Two type of control approach will be presented : auxiliary and prefilter type.

4.5 Neural Network Auxiliary Control

The objective of this section is to present a comprehensive study of the design of NN controller when used as an auxiliary controller. Based on the computed-torque

control concept, a unified training signal is derived for an NN controller with a two-layer architecture. Comparison studies based on a three joint robot have been made for the NN controller implemented in both feedforward and feedback configurations. In order to apply back-propagation algorithm properly, a new training signal, $\boldsymbol{v}$ is developed.

Furthermore, the proposed NN controllers are compared in terms of accuracy and convergence rate with one existing NN control scheme [43] presented in the previous section 4.4 and one adaptive control scheme [4] presented in Appendix A. Since the NN controller performance is also effected by the NN input types, the number of hidden neurons, the back-propagation algorithm update rate, and the initial weight values, these factors are also examined by extensive simulation studies.

4.5.1 Feedforward and Feedback NN Controller Schemes

In this section, we present NN controller designs to achieve disturbance rejection for a computed-torque control system. For completeness two NN control schemes, feedforward and feedback, are examined, and they are depicted in Figures 4.5 and 4.6. In both schemes, the NN output $\boldsymbol{\Phi}$ cancels out the uncertainties caused by inaccurate robot model in the computed-torque controller. The main difference between the two schemes is that the joint variables used in the NN inputs and the computed-torque controller are either the desired values $\boldsymbol{q_d(t)}$ or the actual values $\boldsymbol{q(t)}$. It is also noted that the NN inputs can be either $\boldsymbol{q_d(t), \dot{q}_d(t), \ddot{q}_d(t)}$ or $\boldsymbol{q(t)\ \dot{q}(t)\ \ddot{q}(t)}$, or the time-

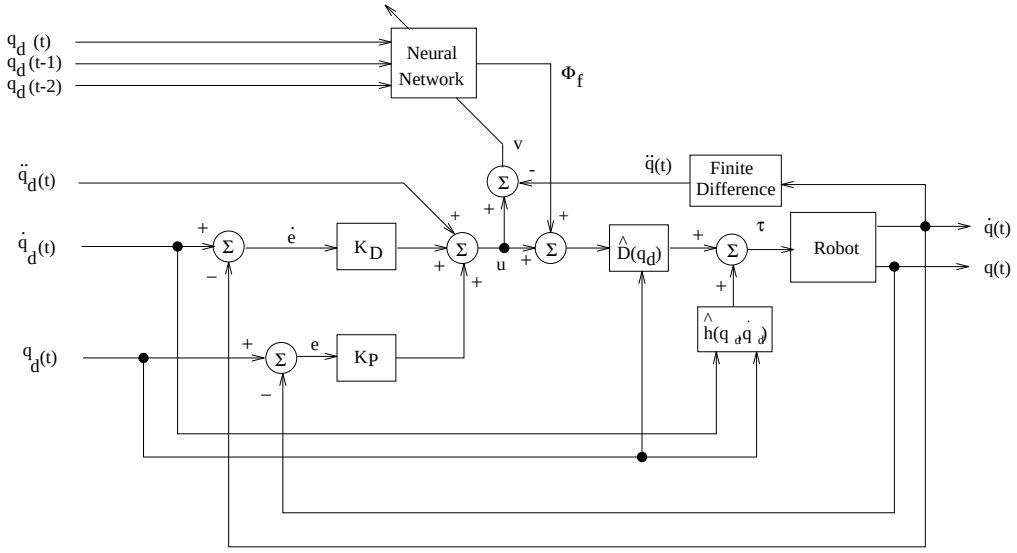


Figure 4.5: Proposed Feedforward Neural Compensator Structure

delayed values $\boldsymbol{q_d(t)\ q_d(t-1)\ q_d(t-2)}$ or $\boldsymbol{q(t)\ q(t-1)\ q(t-2)}$. The latter case is called time-delayed NN in the literature [19]. Delay time is chosen as the sampling period of the controller. We found in our simulations (see section 4.5.3) that the NN performs better when time-delayed joint values are used instead of the velocity and acceleration values calculated from finite difference approximations based on samples of $\boldsymbol{q(t)}$.

The corresponding robot joint torques are

Feedforward scheme :

$$\boldsymbol{\tau^f = \hat{D}(q_d)(U + \Phi_f) + \hat{h}(q_d, \dot{q}_d)} \tag{4.12}$$

Feedback scheme :

$$\boldsymbol{\tau}^b = \hat{\boldsymbol{D}}(\boldsymbol{q})(\boldsymbol{U} + \boldsymbol{\Phi}_b) + \hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{4.13}$$

For the sake of simplicity, let us use the following torque equation to represent both schemes (4.12) (4.13):

$$\boldsymbol{\tau} = \hat{\boldsymbol{D}}(\boldsymbol{U} + \boldsymbol{\Phi}) + \hat{\boldsymbol{h}} \tag{4.14}$$

The corresponding closed loop error system is (under the assumption $\boldsymbol{q} \cong \boldsymbol{q_d}$ in the feedforward case)

$$\boldsymbol{v} = \ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f}) - \boldsymbol{\Phi} \tag{4.15}$$

Since the control objective is to generate $\boldsymbol{\Phi}$ to reduce $\boldsymbol{v}$ to zero in (4.15), we therefore propose here to use $\boldsymbol{v}$

$$\boldsymbol{v} = \ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} \tag{4.16}$$

as the new error signal for training the NN in both schemes. The ideal value of $\boldsymbol{\Phi}$ at $\boldsymbol{v} = \boldsymbol{0}$ then is

$$\boldsymbol{\Phi} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f}) \tag{4.17}$$

Clearly minimizing the error signal $\boldsymbol{v}$ allows us to achieve ideal computed torque control directly. We note that in an existing result shown in Figure 4.6 [43] which is closely related to the feedback scheme of Figure 4.5, it used an error signal equivalent to

$$\boldsymbol{\tau_e} = \boldsymbol{\tau} - (\hat{\boldsymbol{D}}\ddot{\boldsymbol{q}} + \hat{\boldsymbol{h}}) - \hat{\boldsymbol{D}}\boldsymbol{\Phi} \tag{4.18}$$

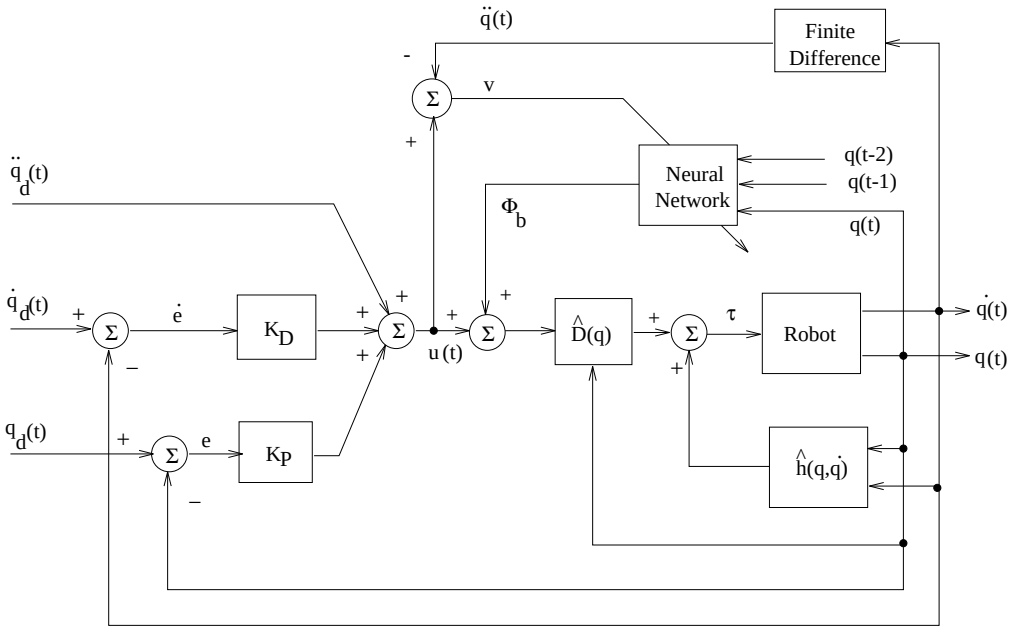


Figure 4.6: Proposed Feedback Neural Compensator Structure

This provides the relationship of $\boldsymbol{v} = \hat{\boldsymbol{D}}^{-1}\boldsymbol{\tau_e}$, so the ideal system performances for both cases are identical. However, the use of error signal $\boldsymbol{\tau_e}$ as suggested in [43] requires on-line computation of the nominal robot model $\boldsymbol{\tau_m} = \hat{\boldsymbol{D}}\ddot{\boldsymbol{q}} + \hat{\boldsymbol{h}}$ which is far more time consuming than computing $\boldsymbol{v}$ in (4.16) as proposed.

4.5.2 Neural Network Compensator Design

The two-layer feedforward neural network shown in Figure 4.7 is used as the compensator. It is composed of an input buffer, a nonlinear hidden layer, and a linear output layer. The inputs $\boldsymbol{X} = [\boldsymbol{x(t)^T}\ \boldsymbol{x(t-1)^T}\ \boldsymbol{x(t-2)^T}]^{\boldsymbol{T}}$ are multiplied by weights($\boldsymbol{w^1_{ij}}$) and summed at each hidden node. Then the summed signal at a node activates a nonlinear function $\boldsymbol{f(\cdot)}$, called a sigmoid function, which is bounded

in magnitude between $\mathbf{0}$ and $\mathbf{1}$:

$$\boldsymbol{f(\cdot) = \frac{1}{1 + exp(-(\cdot))}} \tag{4.19}$$

The outputs from all $\boldsymbol{f(\cdot)}$ are weighted($\boldsymbol{w^2_{jk}}$) and summed at each output node. Thus, the output $\boldsymbol{\phi_k}$ at a linear output node can be calculated from its inputs as follows:

$$\boldsymbol{\phi_k = [\sum_{j=1}^{n_H} w^2_{jk}(\frac{1}{1 + exp(-(\sum_{i=1}^{n_I} x_i w^1_{ij} + b^1_j))})] + b^2_k} \tag{4.20}$$

where $\boldsymbol{n_I}$ is the number of inputs, $\boldsymbol{n_H}$ is the number of hidden neurons, $\boldsymbol{x_i}$ is the $\boldsymbol{i}$th element of input of X, $\boldsymbol{w^1_{ij}}$ is the first layer weight between $\boldsymbol{i}$th input and $\boldsymbol{j}$th hidden neuron, $\boldsymbol{w^2_{jk}}$ is the second layer weight between $\boldsymbol{j}$th hidden neuron and $\boldsymbol{k}$th output neuron, $\boldsymbol{b^1_j}$ is a biased weight for $\boldsymbol{j}$th hidden neuron and $\boldsymbol{b^2_k}$ is a biased weight for $\boldsymbol{k}$th output neuron. If $\boldsymbol{n}$ is the number of output neurons, the total number of weights($\boldsymbol{w_T}$) is $\boldsymbol{w_T = (n_I + 1)n_H + (n_H + 1)n}$. The total number of neurons is $\boldsymbol{n_T = n_H + n}$. Thus, our typical structure shown in Figure 4.7 has $\boldsymbol{n_I = 9, n = 3, n_T = 6}$ and $\boldsymbol{w_T}$ = 81. The weight updating law minimizes the objective function $\boldsymbol{\mathcal{J}}$ which is a quadratic function of the training signal $\boldsymbol{v}$.

$$\boldsymbol{\mathcal{J} = \frac{1}{2} v^T v} \tag{4.21}$$

Differentiating equation(4.21) and making use of (4.15) yields the gradient of $\boldsymbol{\mathcal{J}}$ as follows:

$$\boldsymbol{\frac{\partial \mathcal{J}}{\partial w} = \frac{\partial v^T}{\partial w} v = -\frac{\partial \Phi^T}{\partial w} v} \tag{4.22}$$

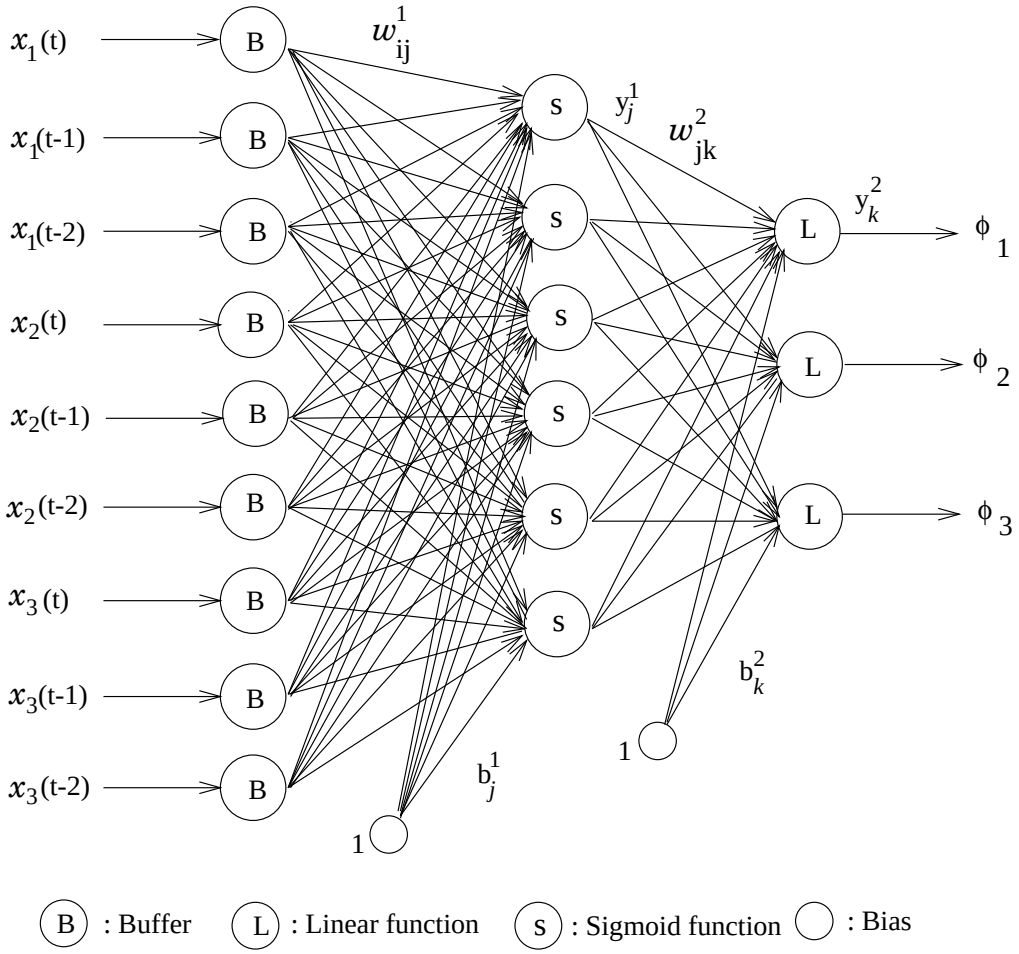


Figure 4.7: Multilayered Feedforward Neural Network Structure

in which the fact $\frac{\partial v^T}{\partial w} = -\frac{\partial \Phi^T}{\partial w}$ is used. The back-propagation update rule for the weights with a momentum term is

$$\Delta w(t) = \eta \frac{\partial \Phi^T}{\partial w} v + \alpha \Delta w(t-1) \tag{4.23}$$

where η is the update rate and α is the momentum coefficient. Specifically,

$$\Delta w_{ij}^1(t) = \eta y_j(1-y_j)x_i[\sum_{k=1}^{n} v_k w_{jk}^2] + \alpha \Delta w_{ij}^1(t-1) \tag{4.24}$$

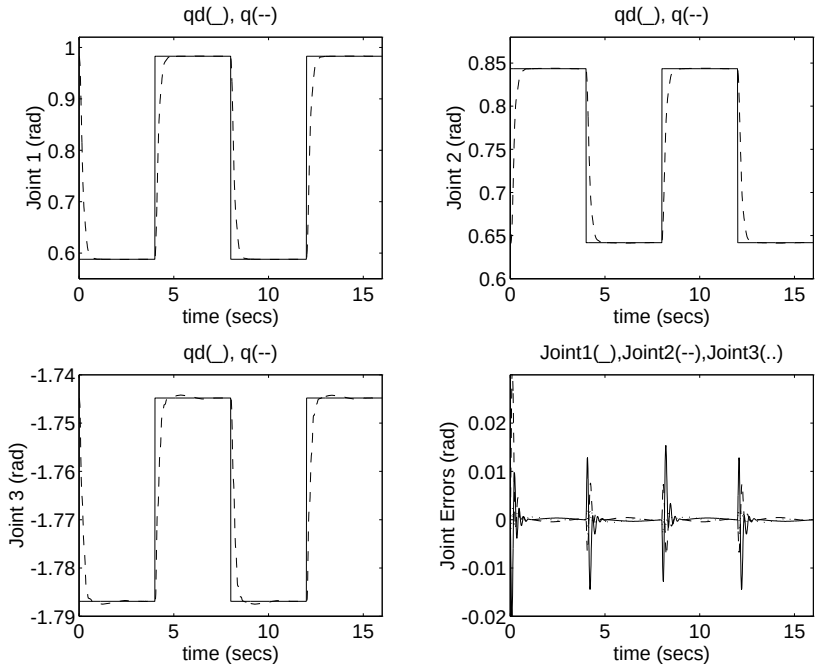


Figure 4.8: Feedback Scheme for Rectangular Trajectory

$$\Delta w_{jk}^2(t) = \eta v_k y_j + \alpha \Delta w_{jk}^2(t-1) \tag{4.25}$$

$$\Delta b_j^1(t) = \eta y_j(1-y_j)[\sum_{k=1}^{n} v_k w_{jk}^2] + \alpha \Delta b_j^1(t-1) \tag{4.26}$$

$$\Delta b_k^2(t) = \eta v_k + \alpha \Delta b_k^2(t-1) \tag{4.27}$$

$$y_j = \frac{1}{1+exp(-(\sum_{i=1}^{n_I} x_i w_{ij}^1 + b_j^1))} \tag{4.28}$$

$$y_k = \sum_{j=1}^{n_H} y_j w_{jk}^2 + b_k^2 \tag{4.29}$$

where y_j is the output of jth hidden neuron and y_k is the output of kth output neuron.

4.6 Robot Model for Simulation Studies

Through this thesis, all comprehensive simulation studies have been carried out using a three link rotary robot manipulator whose parameters are taken from the first three links of Puma 560 arm and the nominal system parameters are used as the basis in forming the robot model $\hat{\boldsymbol{D}}(\boldsymbol{q})$ and $\hat{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. The kinematic equations and Jacobian matrix of a three link rotary robot manipulator are shown in Appendix B. The robot dynamic equations derived from Lagrangian formulation are also shown in Appendix B.

Model uncertainties included a 10 Kg payload attached to the third link, Coulomb friction and viscous friction forces $\boldsymbol{\tau_f}(\dot{\boldsymbol{q}})$ added to each joint where $\boldsymbol{\tau_f}(\dot{\boldsymbol{q}}) = \mathbf{5.0}\boldsymbol{sgn}(\dot{\boldsymbol{q}}) + \mathbf{8.0}(\dot{\boldsymbol{q}})$.

4.6.1 NN Controller Performance

The performance of the proposed NN control schemes is extensively investigated in this section. For the NN controller, we have chosen six hidden neurons $(\boldsymbol{n_H} = \mathbf{6})$. The back propagation algorithm parameters are: $\boldsymbol{w_{ij}^k}(\mathbf{0}) = \mathbf{0}, \boldsymbol{\eta} = \mathbf{0.01}$ and $\boldsymbol{\alpha} = \mathbf{0.9}$. Choices of $\boldsymbol{n_H}, \boldsymbol{w_{ij}^k}(\mathbf{0}),$ and $\boldsymbol{\eta}$ are justified in the next section. The controller gains are selected as $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{20}, \mathbf{20}, \mathbf{20}]$ and $\boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{100}, \mathbf{100}, \mathbf{100}]$ which give identical critically damped motions at the three joints. The performances of the proposed schemes are tested by tracking three different joint trajectories : rectangular, circular and composite trajectory as shown in Figure 4.8, 4.9, and 4.10, respectively.

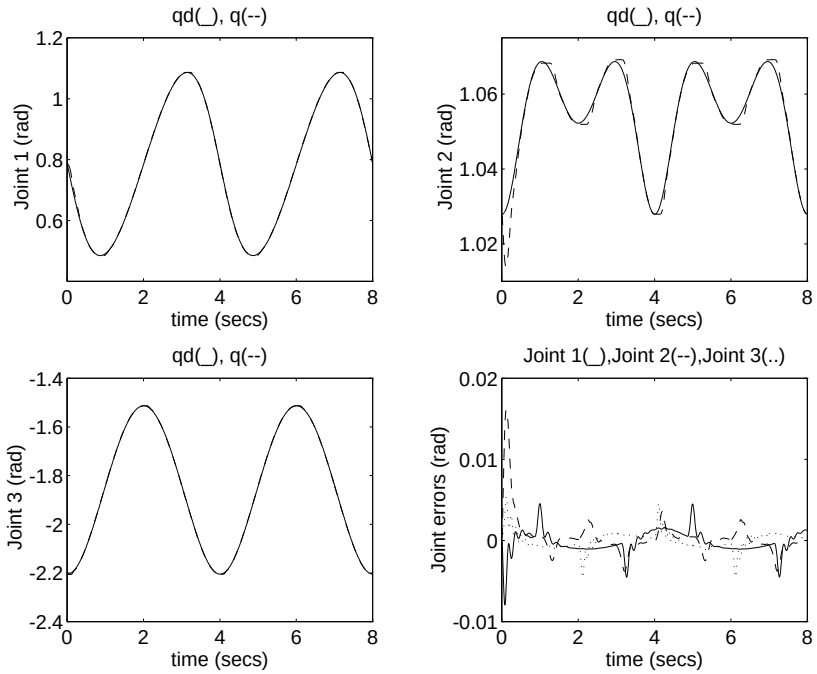


Figure 4.9: Feedforward Scheme for Circular Trajectory

The tracking error is defined as the difference between the ideal computed-torque response $\boldsymbol{q_f}$ and actual response $\boldsymbol{q}$. For each $\boldsymbol{i}$th joint, $\boldsymbol{q_{fi}}$ is related to the desired trajectory $\boldsymbol{q_{di}}$ by (4.6) as

$$\ddot{\boldsymbol{q}}_{\boldsymbol{di}} + \mathbf{20}\dot{\boldsymbol{q}}_{\boldsymbol{di}} + \mathbf{100}\boldsymbol{q}_{\boldsymbol{di}} = \ddot{\boldsymbol{q}}_{\boldsymbol{fi}} + \mathbf{20}\dot{\boldsymbol{q}}_{\boldsymbol{fi}} + \mathbf{100}\boldsymbol{q}_{\boldsymbol{fi}} \tag{4.30}$$

System performance is measured by the following saves of tracking error-squares over one training cycle of a trajectory as follows:

$$\boldsymbol{E_p} = \sum_{\boldsymbol{t=1}}^{\boldsymbol{P/T}} \parallel \boldsymbol{q_f} - \boldsymbol{q} \parallel^2 (\boldsymbol{rad})^2 \qquad \boldsymbol{E_v} = \sum_{\boldsymbol{t=1}}^{\boldsymbol{P/T}} \parallel \dot{\boldsymbol{q}}_{\boldsymbol{f}} - \dot{\boldsymbol{q}} \parallel^2 (\boldsymbol{rad/sec})^2 \tag{4.31}$$

where $\boldsymbol{E_p}$ is the position error; $\boldsymbol{E_v}$ is the velocity error; $\boldsymbol{P}$ is one training cycle elapse time, where $\boldsymbol{P} = \mathbf{8, 4,}$ and $\mathbf{16}$ seconds for rectangular, circular, and composite trajec-

tory respectively; and $\boldsymbol{T}$ is the sampling period of $\boldsymbol{T = 5ms}$. Table 4.1 summarizes the converged performance results of both feedforward and feedback schemes. We see that both schemes performed extremely well with feedback scheme doing slightly better in composite trajectory tracking and with feedforward scheme doing slightly better in rectangular trajectory tracking (see also Figure 4.12). Sample tracking results are

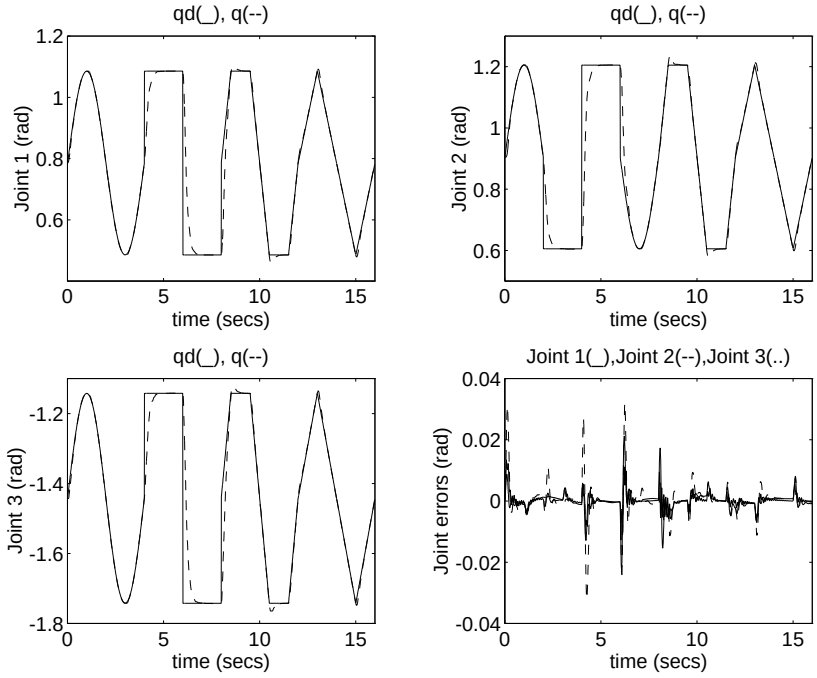


Figure 4.10: Feedback Scheme for Composite Trajectory

plotted in Figures 4.8-10. The simulation data showed that the rate of convergence for all trajectories are very fast with a convergence time less than one second. Thus on-line trajectory control of the proposed NN schemes is completely acceptable in practice. We note that the joint angle responses of the feedback scheme to a step command shown in Figure 4.8 follow very closely to the ideal response (critically

damped) according to (4.6) as expected. As a base line comparison, we also listed the performance of the uncompensated computed-torque control system in Table 4.1. It is seen that the tracking errors are very large.

Table 4.1: Tracking errors after convergence ($\boldsymbol{\eta = 0.01, \alpha = 0.9, n_H = 6, w_{ij}^k(0) = 0}$)

Schemes	Errors	Rectangular	Circular	Composite
FEED FORWARD	$\boldsymbol{E_p}$	0.0056	0.0036	0.1989
	$\boldsymbol{E_v}$	1.9520	0.5069	68.4828
FEED BACK	$\boldsymbol{E_p}$	0.0157	0.0036	0.1279
	$\boldsymbol{E_v}$	6.7260	0.5022	44.9980
Uncompensated Controller	$\boldsymbol{E_p}$	31.3614	27.8037	109.0905
	$\boldsymbol{E_v}$	12.6291	25.3965	261.7838
Ishiguro's Method	[a]BR	$\boldsymbol{\pm 45}$	$\boldsymbol{\pm 50}$	$\boldsymbol{\pm 60}$
	$\boldsymbol{E_p}$	0.5489	0.0036	3.2605
	$\boldsymbol{E_v}$	21.0631	0.4465	169.1020
Craig's Method [c]($\boldsymbol{\beta = 10}$)	[b]$\boldsymbol{CT}$	14 secs	6 secs	5 secs
	[d]$\boldsymbol{\gamma}$	0.01	0.05	0.01
	$\boldsymbol{E_p}$	0.5467	0.0661	0.5938
	$\boldsymbol{E_v}$	2.7800	1.0733	7.1166

[a] BR is optimized bound of NN output. [b]$\boldsymbol{CT}$ is convergence time. [c] the filter parameter. [d] the adaptive gain.

Figure 4.11 shows the end point trajectories of the circular command signal for the uncompensated controller and the feedforward NN controller. The improvement achieved by the NN controller is clearly demonstrated. For further comparison, we also listed in Table 4.1 the results of the closely related NN control method of Ishiguro et al. [43]. In that method, the system performance is affected by the bound of NN compensator output. Thus the bound can be chosen appropriately in accordance

with the bounds of uncertainties and the type of trajectory. We found the optimal bound(BR) experimentally for each trajectory for the same $\boldsymbol{\eta = 0.01}$ and $\boldsymbol{\alpha = 0.9}$. The performance is not as good as the proposed schemes in general. The convergence rate is comparable however.

Another important comparison is with the well known adaptive controller by Craig ***et al.*** [4] under the same control environment. For this method we have chosen the filter parameter $\boldsymbol{\beta = 10}$ and the adaptive gain γ is optimized for each trajectory (see Appendix A). We see from Table 4.1 that Craig's method is comparable over all in tracking accuracy, but it is much slower in convergence rate (a factor of 5 to 14 times). This comparative study shows that the proposed NN control schemes are far more practical and effective than adaptive controllers for robot manipulator control.

4.6.2 Effects of Input Types, Hidden Neurons, Update Rate and Initial Weights on NN controller performance

In designing the NN controller, there are a number of parameters that should be judiciously selected. They are, among others, the composition of NN inputs, the number of hidden neurons, the weight update rate and the initial weight values. We decided to evaluate via simulation for the three-link robot case the effects of these parameters to the controller performance. The results are discussed below.

(a) ***Input types*** :

As a general case, we have proposed in Figure 4.4 and 4.5 to set the NN inputs

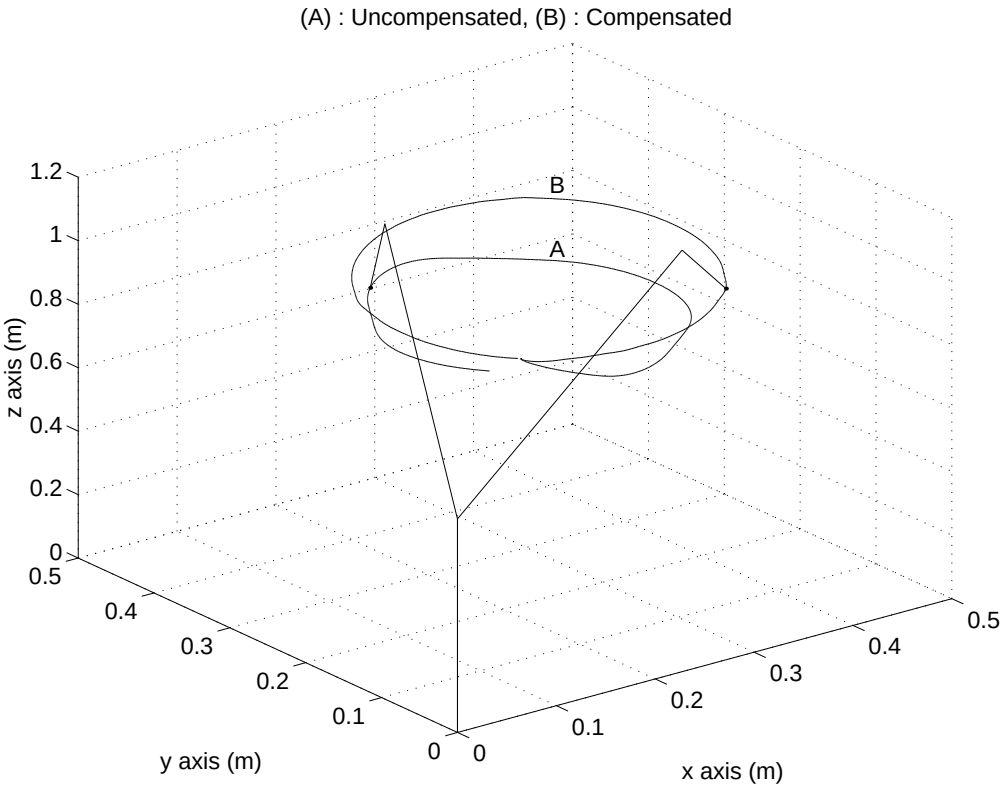


Figure 4.11: End Point Tracking of a Circular Trajectory for Uncompensated control and the Feedforward NN Control

as $\boldsymbol{X(t) = [x(t)^T, x(t-1)^T, x(t-2)^T]^T}$ where $\boldsymbol{x}$ is either $\boldsymbol{q_d}$ or $\boldsymbol{q}$. Shown in Table 4.2 are the tracking performances for three different compositions of $\boldsymbol{X(t)}$ where we used $\boldsymbol{n_H = 6, \alpha = 0.9,}$ and $\boldsymbol{\eta = 0.01}$ in the simulation. When $\boldsymbol{X(t) = [q(t)^T, \dot{q}(t)^T, \ddot{q}(t)^T]^T}$ is used, we found that the control system can easily become unstable. This is largely due to the fact that approximation of $\boldsymbol{\ddot{q}(t)}$ by finite difference is too noisy which inadvertently causes large fluctuations on the weights. The abbreviated input $\boldsymbol{X(t) = [q(t)^T, \dot{q}(t)^T]^T}$ has no stability problem however, the results are shown in Table 4.2. In view of the overall results it appears that the input can be reduced to simply $\boldsymbol{q(t)}$ or $\boldsymbol{q_d(t)}$ without appreciable loss of performance. This fact is important in that it allows us to reduce the complexity and computation time of the NN controller in practical applications.

Table 4.2: Performances with different inputs and trajectories ($\boldsymbol{\eta = 0.01, \alpha = 0.9, n_H = 6, w_{ij}^k(0) = 0}$)

Trajectory	Error	$\boldsymbol{x(t), \cdot, x(t-2)}$		$\boldsymbol{x(t), x(t-1)}$		$\boldsymbol{x(t)}$		$\boldsymbol{x(t), \dot{x}(t)}$	
		FF	FB	FF	FB	FF	FB	FF	FB
Rectangular	$\boldsymbol{E_p}$	0.0056	0.0158	0.0816	0.0158	0.0818	0.0158	0.0158	0.0158
	$\boldsymbol{E_v}$	1.9520	6.7260	3.8336	6.6968	3.8807	6.7033	6.7087	6.7151
Circular	$\boldsymbol{E_p}$	0.0036	0.0036	0.0036	0.0036	0.0036	0.0036	0.0036	0.0036
	$\boldsymbol{E_v}$	0.5069	0.5022	0.5055	0.5013	0.5129	0.5034	0.5056	0.5043
Composite	$\boldsymbol{E_p}$	0.1989	0.1279	0.1990	0.1279	0.1988	0.1278	0.8125	0.1278
	$\boldsymbol{E_v}$	68.483	44.998	68.744	45.012	68.664	44.907	72.819	44.961

(b) ***Hidden neurons*** :

The effects of the number of hidden neurons have also been investigated experimentally for the proposed schemes, and the results are displayed in Figure 4.12. The

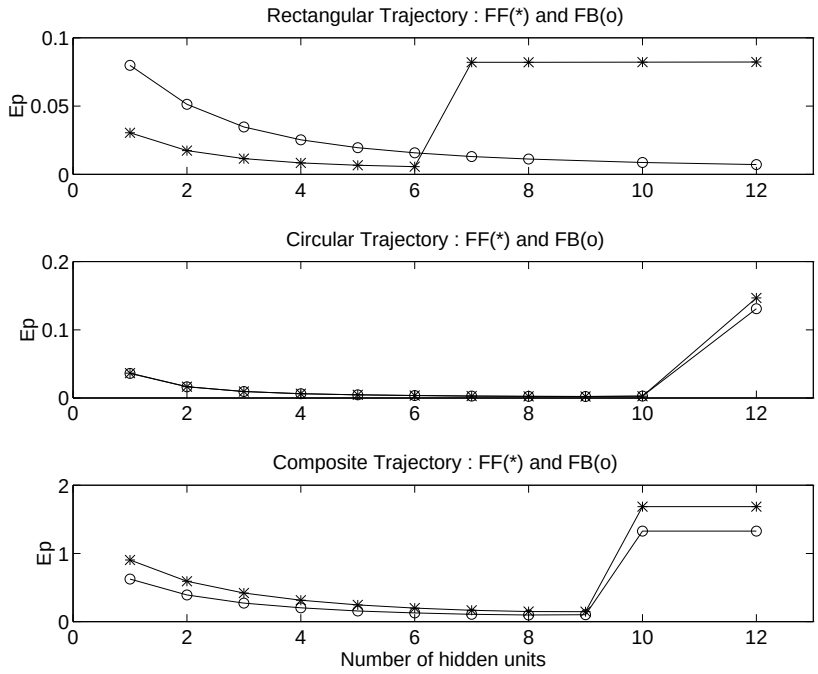


Figure 4.12: Errors versus hidden neurons $\boldsymbol{n_H}$ for $\boldsymbol{\alpha = 0.9, \eta = 0.01}$ and $\boldsymbol{w_{ij}^k(0) = 0}$

$\boldsymbol{E_P}$ data show that the optimal number of hidden neurons is different for each trajectory type. However a range of $\boldsymbol{n_H, 6 \leq n_H \leq 9,}$ is quite acceptable. In order to reduce the NN complexity, the number of hidden neurons can be selected as $\boldsymbol{n_H = 6}$. This choice has been used for all of our simulations.

(c) ***Update rate*** :

Along with momentum coefficient $\boldsymbol{\alpha}$, weight update rate $\boldsymbol{\eta}$ in the back-propagation algorithm also effects the performance (convergence rate and tracking accuracy) including stability. Optimal solutions of $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ are generally complicated issues yet to be dealt with theoretically, although some fundamental results have been obtained recently [80]. Here we fix $\boldsymbol{\alpha}$ at the commonly used value of $\boldsymbol{\alpha = 0.9}$, and evaluate

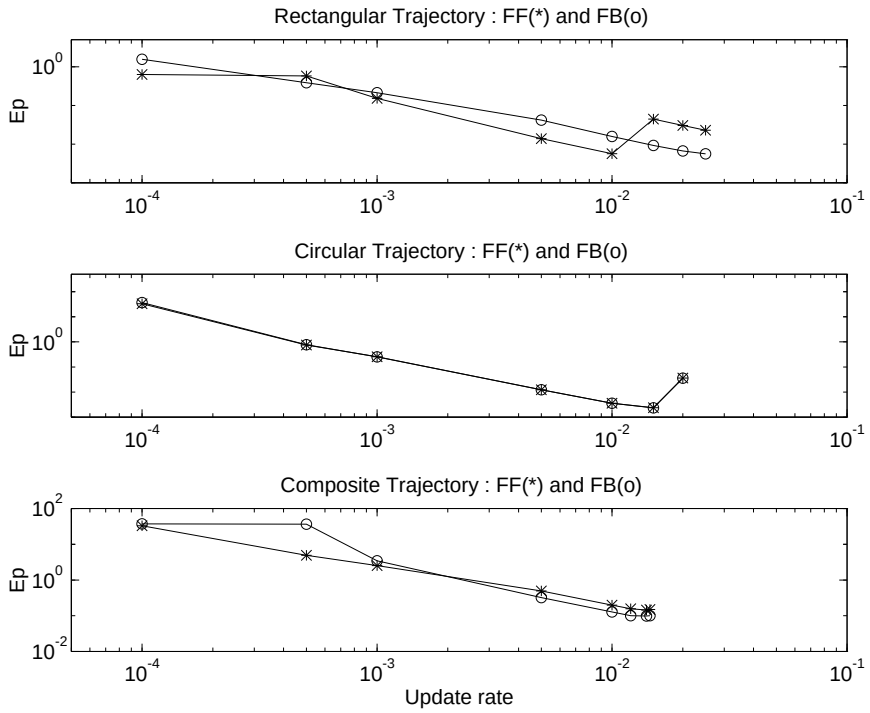


Figure 4.13: Joint errors versus Update rate $\boldsymbol{\eta}$ for $\boldsymbol{\alpha = 0.9, n_H = 6}$ and $\boldsymbol{w_{ij}^k(0) = 0}$

tracking errors with varying values of $\boldsymbol{\eta}$. The results are displayed in Figure 4.13. The data show that $\boldsymbol{\eta = 0.01}$ is a good choice for all cases. We note that a larger $\boldsymbol{\eta}$ provides better tracking accuracy, but a value slightly greater than $0.01(\boldsymbol{i.g.\ \eta \geq 0.02})$ would quickly destabilize the system.

(d) ***Initial weight values*** :

Finally the effects of initial weight value $\boldsymbol{w_{ij}^k(0)}$ choice have also been examined. All our simulations above used $\boldsymbol{w_{ij}^k(0) = 0}$ which implies that the compensating control is initially zero. Here, in order to investigate the effects of initial weights, initial weight values $\boldsymbol{w_{ij}^k(0)}$ are chosen either as $\boldsymbol{w_{ij}^k(0) = 0}$ or as zero mean uniformly

distributed random variables in the range of ± 0.5. Their performances have been evaluated in terms of E_p. The results show no significant difference. Thus $w_{ij}^k(0) = 0$ is considered as a good all around choice.

4.6.3 Discussion

A comprehensive study of neural network robot control is presented in this section. The NN compensator serves as an auxiliary control for the computed-torque controller to counteract the uncertainties in the robot dynamics and disturbances occurring during application. A new teaching signal is developed in this section for the back-propagation algorithm, and delayed joint variables are proposed as inputs to the NN. In addition several NN design parameters such as number of inputs, number of hidden neurons, weight update rate, and initial weight values have been investigated in an extensive simulation on a three-link puma 560 like robot arm. Judging from the training and tracking performances of three different trajectory types, it is reasonable to conclude that a good and practical set of choice of the NN controller parameters is $X(t) = [x(t)], n_H = 6, \eta = 0.01,$ and $w_{ij}^k(0) = 0$. The system performances under these conditions are excellent and the convergence rates are under one second. Thus the proposed NN controllers are feasible for on-line robot control. Comparisons with Ishiguro et. al's NN control scheme as well as Craig ***et. al***'s adaptive control scheme show that the proposed control technique is superior in either tracking accuracy or convergence rate.

4.7 Neural Network Inverse Control

Here we are proposing two new NN control approaches as an extension of direct inverse NN control scheme shown in Figure 3.3 : one is Jacobian based approach that requires the estimation of Jacobian informations and another is feedback error based approach that does not require Jacobian informations.

4.7.1 Jacobian Approach

In this section we investigate a new robot control technique as depicted in Figure 4.14. This control approach is conceptually different from those schemes presented previously in Figures 4.1 and 4.2. Compensation of robot model uncertainty is achieved by modifying the desired trajectory, $\boldsymbol{q_d}$, using an appropriately trained NN serving as a nonlinear filter. The filtered trajectories $\boldsymbol{q_r}$ $\dot{\boldsymbol{q}}_{\boldsymbol{r}}$ $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$ are the desired reference inputs to the computed-torque robot control system (referred to as the plant). The trajectory tracking training error $\boldsymbol{\varepsilon} = \boldsymbol{q_d} - \boldsymbol{q}$ is used as training signal. In this arrangement, the NN acts as the inverse of the plant. Hence we refer the proposed technique as neural network inverse control.

The control problem posed in Figure 4.14 is an example of the general problem of NN control of nonlinear plants. For this class of problems, plant dynamics must be known (or a model is identified) to implement the error-back-propagation algorithms for training the NN. We also note that the control system of Figure 4.14 can be viewed as an inverse system model identification problem [19]. Thus a necessary condition

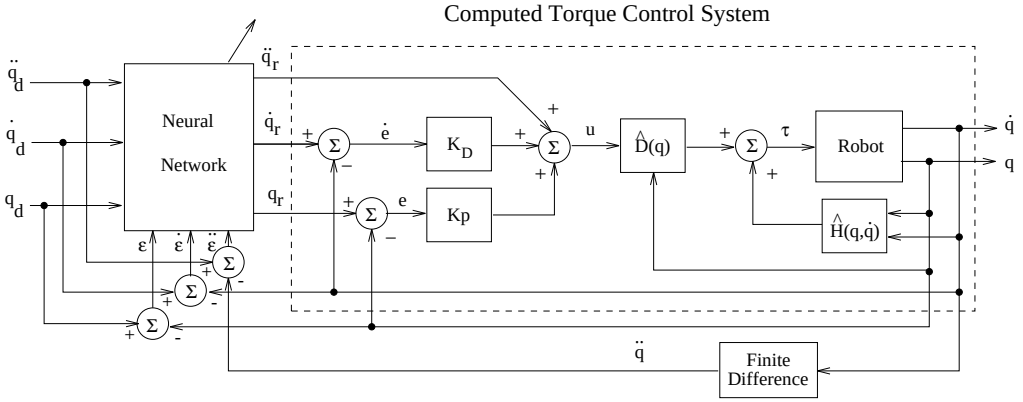


Figure 4.14: Scheme A : Inverse Control Structure for Computed-Torque-Control

by the proposed scheme is that the plant be stable.

4.7.1.1 Neural Network Inverse Control

The basic concept of this scheme is that the NN controller acts as the inverse of the plant(the robot under computed-torque control) so that the robot response $\boldsymbol{q}$ tracks $\boldsymbol{q_d}$ with minimal distortion. So when the NN controller is updated on-line to account for the uncertainties in the robot dynamics, optimal trajectory tracking by the robot can be achieved. The tracking errors $(\boldsymbol{\varepsilon}, \dot{\boldsymbol{\varepsilon}}, \ddot{\boldsymbol{\varepsilon}})$ are used to update the NN weights. From Figure 4.14 (called Scheme A) the control input vector $\boldsymbol{U}$ is

$$\boldsymbol{U} = \ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \boldsymbol{K_D}(\dot{\boldsymbol{q}}_{\boldsymbol{r}} - \dot{\boldsymbol{q}}) + \boldsymbol{K_P}(\boldsymbol{q}_{\boldsymbol{r}} - \boldsymbol{q}) \tag{4.32}$$

where $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}\ \dot{\boldsymbol{q}}_{\boldsymbol{r}}\ \boldsymbol{q}_{\boldsymbol{r}}$ are reference input vectors to the plant and they are independently generated by the NN (so they are independent to one another, $\boldsymbol{i.e.}$derivative relations

among them do not necessarily hold). Combining (4.2),(4.3) and (4.32) yields

$$(\ddot{q}_r - \ddot{q}) + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q) = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) \qquad (4.33)$$

To analyze how the NN controller works, let us denote its three outputs as ϕ_p, ϕ_d, and ϕ_a where $\ddot{q}_r = \phi_a$, $\dot{q}_r = \phi_d$, $q_r = \phi_p$ as shown in Figure 4.14. Using the definition of tracking error $\varepsilon = q_d - q$, (4.33) can be rewritten as

$$\ddot{\varepsilon} + K_D\dot{\varepsilon} + K_P\varepsilon = \Delta_M + \ddot{q}_d + K_D\dot{q}_d + K_P q_d - \Psi_A \qquad (4.34)$$

where $\Delta_M = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f)$ and $\Psi_A = (\phi_a + K_D\phi_d + K_P\phi_p)$ which represents the total contribution of the NN outputs to the tracking error equation (4.34). When the NN controller is converged, $\varepsilon = 0$ and the ideal NN outputs $\phi_p\ \phi_d\ \phi_a$ satisfy the relationship

$$\Psi_A = \Delta_M + \ddot{q}_d + K_D\dot{q}_d + K_P q_d \qquad (4.35)$$

It is seen that Ψ_A is equal to a combination of the desired trajectories and robot model uncertainties. This means that the function of the NN controller is to modify the desired trajectories such that its outputs $\phi_p\ \phi_d\ \phi_a$ satisfy the relation (4.35). In view of the above analysis it is clear that the NN controller can be redesigned as shown in Figure 4.15 (called Scheme B). In this scheme, the NN outputs are added to the desired trajectories such that $\ddot{q}_r = \phi_a + \ddot{q}_d$, $\dot{q}_r = \phi_d + \dot{q}_d$, and $q_r = \phi_p + q_d$. Substituting $q_r\ \dot{q}_r\ \ddot{q}_r$ into (4.33) yields

$$\ddot{\varepsilon} + K_D\dot{\varepsilon} + K_P\varepsilon = \Delta_M - \Psi_B \qquad (4.36)$$

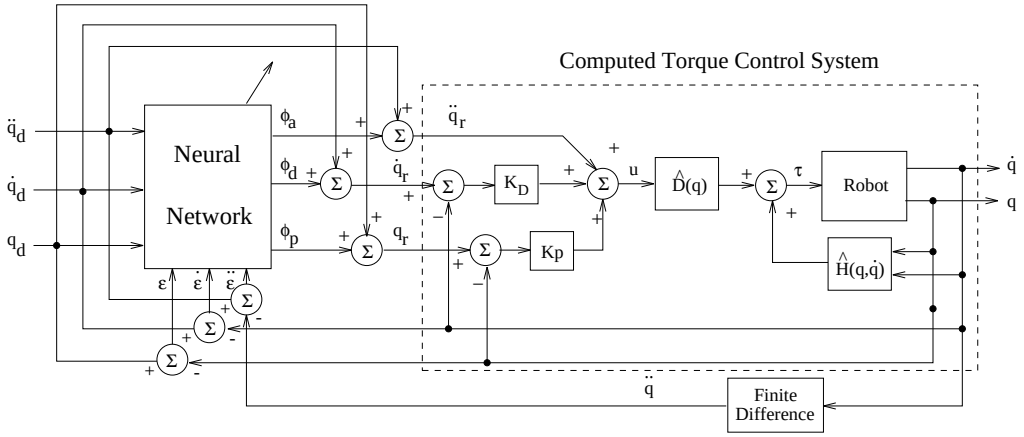


Figure 4.15: Scheme B : Modified Inverse Control Structure for Computed-Torque-Control

where $\boldsymbol{\Psi_B} = [\boldsymbol{\phi_a} + \boldsymbol{K_D}\boldsymbol{\phi_d} + \boldsymbol{K_P}\boldsymbol{\phi_p}]$. Thus, for Scheme B, the ideal output of NN at convergence is

$$\boldsymbol{\Psi_B} = \boldsymbol{\Delta_M} \tag{4.37}$$

In this case the NN is only required to cancel out the robot model uncertainty. Simulations presented later show that Scheme B is more efficient than Scheme A. Comparing the new results with those schemes which modify the robot joint torques as shown in Figure 4.15, the NN in Schemes A and B have higher dimensional output than that of scheme by Ishiguro et. al [43]. This increased NN complexity should provide better controller performance. Our simulations show that this is true. We propose to use multilayer feedforward neural network with back-propagation updating algorithms.

4.7.1.2 Neural Controller Design

The standard two layer feedforward neural network shown in Figure 4.16 is used as the controller. The inputs $\boldsymbol{X} = [\boldsymbol{q_d^T}\ \boldsymbol{\dot{q}_d^T}\ \boldsymbol{\ddot{q}_d^T}]^{\boldsymbol{T}}$.

The weight updating law minimizes the objective function $\boldsymbol{E(\mathcal{E})}$ which is a quadratic function of tracking errors $\boldsymbol{\varepsilon}$:

$$\mathcal{J}(\mathcal{E}) = \frac{1}{2}(\mathcal{E}^T\mathcal{E}) \tag{4.38}$$

where $\mathcal{E} = [\varepsilon^T \dot{\varepsilon}^T \ddot{\varepsilon}^T]^T, \varepsilon = (q_d - q), \dot{\varepsilon} = (\dot{q}_d - \dot{q})$, and $\ddot{\varepsilon} = \ddot{q}_d - \ddot{q}$. The gradient of $\mathcal{J}(\mathcal{E})$ is

$$\frac{\partial \mathcal{J}(\mathcal{E})}{\partial w} = \frac{\partial \mathcal{E}^T}{\partial w}\mathcal{E} = -[\ \frac{\partial q^T}{\partial w}\ \frac{\partial \dot{q}^T}{\partial w}\ \frac{\partial \ddot{q}^T}{\partial w}]\ \mathcal{E} \tag{4.39}$$

In both Scheme A and Scheme B, $q_r\ \dot{q}_r\ \ddot{q}_r$ are functions of w so

$$\begin{aligned} \frac{\partial \mathcal{E}^T}{\partial w} &= -[\ \frac{\partial q^T}{\partial w}\ \frac{\partial \dot{q}^T}{\partial w}\ \frac{\partial \ddot{q}^T}{\partial w}] \\ &= -[\ \frac{\partial q^T}{\partial q_r^T}\frac{\partial q_r^T}{\partial w}\ \frac{\partial \dot{q}^T}{\partial \dot{q}_r^T}\frac{\partial \dot{q}_r^T}{\partial w}\ \frac{\partial \ddot{q}^T}{\partial \ddot{q}_r^T}\ \frac{\partial \ddot{q}_r^T}{\partial w}] \end{aligned} \tag{4.40}$$

Evaluation of the derivatives $\frac{\partial q^T}{\partial q_r^T}\ \frac{\partial \dot{q}^T}{\partial \dot{q}_r^T}\ \frac{\partial \ddot{q}^T}{\partial \ddot{q}_r^T}$ require the knowledge of the plant model. Clearly the plant dynamics is not precisely known in our case. Our proposal here is to use the ideal linear error equation(q_d is replaced by q_r in (4.6))

$$(\ddot{q}_r - \ddot{q}) + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q) = 0 \tag{4.41}$$

as an approximation to the plant dynamics by neglecting the plant uncertainties. The most significant consequence of this approximation is that we have a linear plant model which provides simple results to (4.40). In view of (4.41), we have

$$\frac{\partial \mathcal{J}(\mathcal{E})}{\partial w} = -[\ \frac{\partial q_r^T}{\partial w}\ \frac{\partial \dot{q}_r^T}{\partial w}\ \frac{\partial \ddot{q}_r^T}{\partial w}]\ J_p^T \begin{bmatrix} \varepsilon \\ \dot{\varepsilon} \\ \ddot{\varepsilon} \end{bmatrix} \tag{4.42}$$

where

$$\boldsymbol{J_p} = \begin{bmatrix} \frac{\partial q}{\partial q_r} & \frac{\partial q}{\partial \dot{q}_r} & \frac{\partial q}{\partial \ddot{q}_r} \\ \frac{\partial \dot{q}}{\partial q_r} & \frac{\partial \dot{q}}{\partial \dot{q}_r} & \frac{\partial \dot{q}}{\partial \ddot{q}_r} \\ \frac{\partial \ddot{q}}{\partial q_r} & \frac{\partial \ddot{q}}{\partial \dot{q}_r} & \frac{\partial \ddot{q}}{\partial \ddot{q}_r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{K_P^{-1} K_D} & \boldsymbol{K_P^{-1}} \\ \boldsymbol{K_D^{-1} K_P} & \boldsymbol{I} & \boldsymbol{K_D^{-1}} \\ \boldsymbol{K_P} & \boldsymbol{K_D} & \boldsymbol{I} \end{bmatrix} \tag{4.43}$$

$\boldsymbol{J_p}$ is a Jacobian for the plant which depends only on the PD gains. The gradient rule for weight update is

$$\boldsymbol{\Delta w(t) = -\eta \frac{\partial \mathcal{J}}{\partial w} + \alpha \, \Delta w(t-1)} \tag{4.44}$$

where $\boldsymbol{\eta}$ is the update rate and $\boldsymbol{\alpha}$ is the momentum coefficient.

4.7.1.3 Simplification for Controller Design

The control Schemes A an B involve simultaneous generation of the robot reference trajectories $\boldsymbol{q_r, \dot{q}_r,}$ and $\boldsymbol{\ddot{q}_r}$ using three separate tracking errors, $\boldsymbol{\varepsilon, \dot{\varepsilon}}$, and $\boldsymbol{\ddot{\varepsilon}}$. Thus the total number of weights is $\boldsymbol{(n_I + 1)n_H + (n_H + 1)n_O}$. But in both $\boldsymbol{\Psi_A}$ and $\boldsymbol{\Psi_B}$, the NN outputs $\boldsymbol{\phi_p}$ and $\boldsymbol{\phi_d}$ are weighted more heavily than $\boldsymbol{\phi_a}$ due to the large values of $\boldsymbol{K_D}$ and $\boldsymbol{K_P}$. Therefore we can reduce the complexity of the NN if we eliminate $\boldsymbol{\phi_a}$ from the output and replace it with the finite difference approximation

$$\boldsymbol{\phi_a(t) \cong \frac{\phi_d(t) - \phi_d(t-1)}{T}} \tag{4.45}$$

where $\boldsymbol{T}$ is the sampling time. This simplified controller, called Scheme C, is shown in Figure 4.17 in which only $\boldsymbol{\varepsilon}$ and $\boldsymbol{\dot{\varepsilon}}$ are employed for NN training. The NN controller in Scheme C is represented as

$$\boldsymbol{\Psi_C = \dot{\phi}_d + K_D \phi_d + K_P \phi_p} \tag{4.46}$$

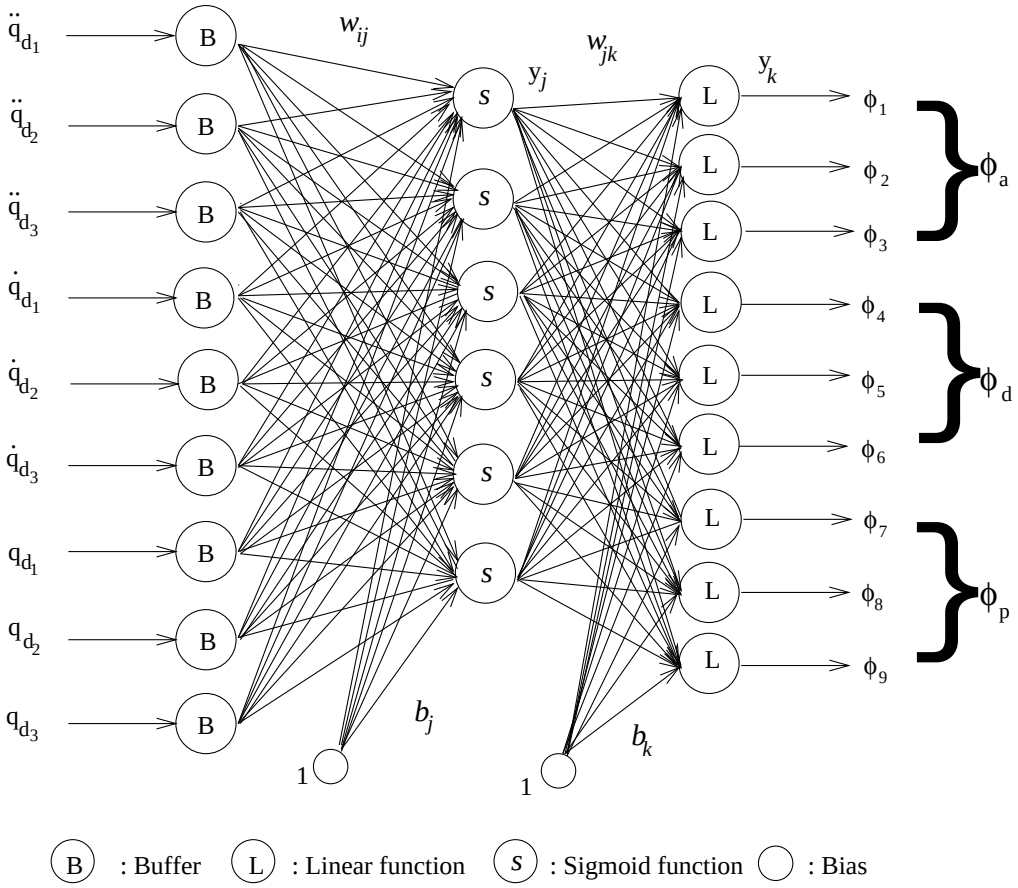


Figure 4.16: Neural Controller Structure

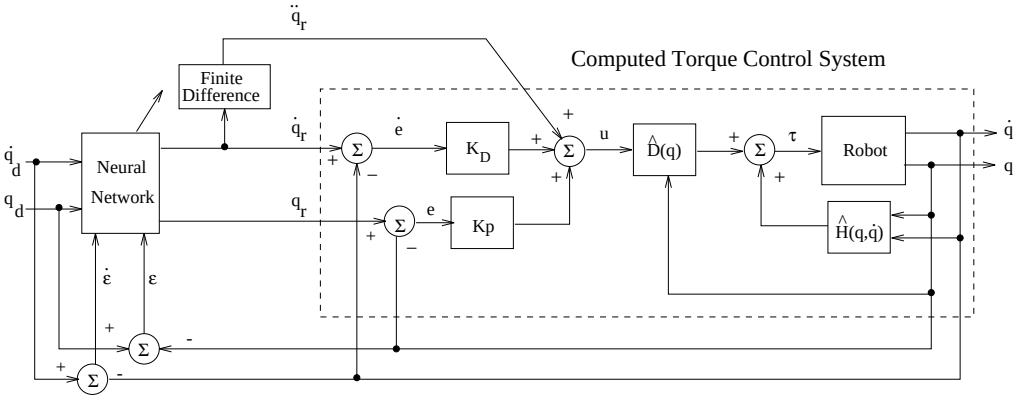


Figure 4.17: Scheme C:Inverse Control Structure for Computed-Torque Control

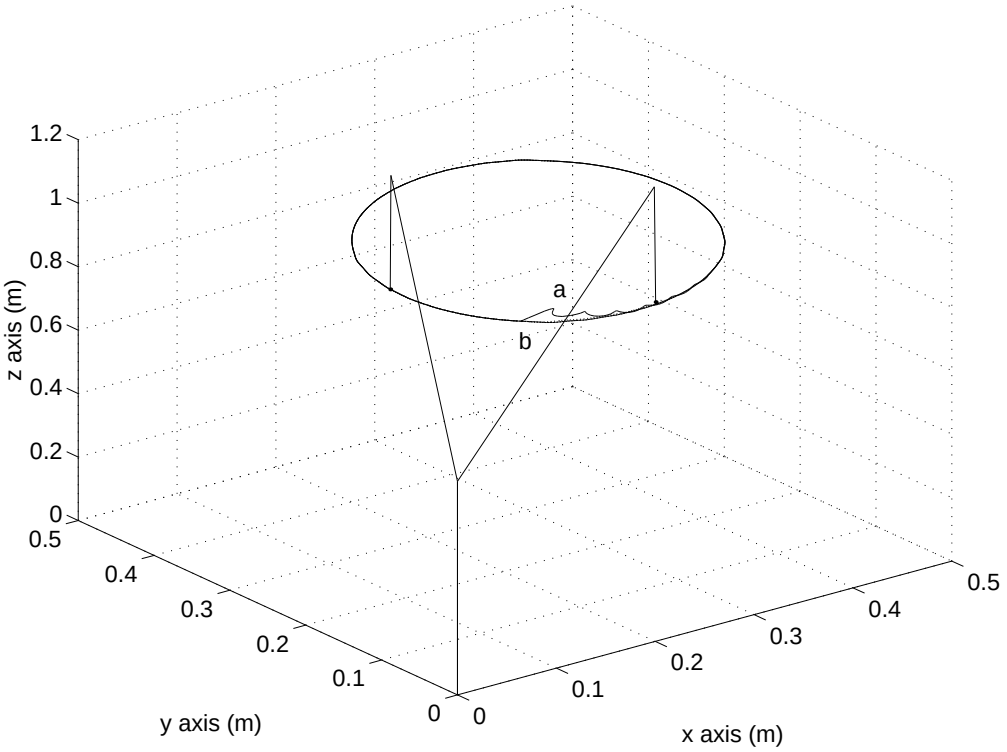


Figure 4.18: End Point Tracking of a Circular Trajectory for a Three Link Robot using control (a) Scheme A and (b) Scheme B

Consequently the number of internal weights is reduced approximately by one third. We note however that one degree of freedom is lost in (4.46) so that its effectiveness in compensating robot uncertainties is reduced. Simulations have confirmed this point.

4.7.1.4 Simulation Results

The values for $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$ are listed in Table 4.3. The performances of the basic control Scheme A and Scheme B are tested by commanding the end point to track a circular trajectory as shown in Figure 4.18. The circle has a period of 4 ***secs***. The

corresponding joint position and velocity errors for 2 cycles are shown in Figure 4.19. It is clear that Scheme B converges faster with better accuracy than Scheme A. It is also seen that the robot joint trajectories converged rapidly in about 1 $\boldsymbol{sec}$ (one fourth of a cycle). The simulation results demonstrate the fast adaptive rate of the NN controller for meaningful on-line application.

Additional simulation studies have been carried out using Scheme C and Ishiguro et. al's scheme [43]. The results are listed in Table 4.3. The errors $\boldsymbol{E_p}$ $\boldsymbol{E_v}$ $\boldsymbol{E_c}$ are the total tracking errors computed over one cycle of the trajectory as follows:

$$\boldsymbol{E_p = \sum_{t=1}^{P/T} \| q_f - q \|^2 \ (rad)^2 \quad E_v = \sum_{t=1}^{P/T} \| \dot{q}_f - \dot{q} \|^2 \ (rad/sec)^2} \tag{4.47}$$

$$\boldsymbol{E_c = \sum_{j=1}^{P/T} [(x_d - x)^2 + (y_d - y)^2 + (z_d - z)^2](m)^2} \tag{4.48}$$

where $\boldsymbol{q_c}$ is the ideal computed-torque response given by

$$\boldsymbol{\ddot{q}_{di} + 20\dot{q}_{di} + 100q_{di} = \ddot{q}_{fi} + 20\dot{q}_{fi} + 100q_{fi} \ \ i = 1, 2, 3} \tag{4.49}$$

$\boldsymbol{P = 4sec}$ and $\boldsymbol{T = 5ms}$. The results of $\boldsymbol{E_p}$ $\boldsymbol{E_v}$ $\boldsymbol{E_c}$ in Table 4.3 are computed during the second cycle of tracking (controller is well converged after the first cycle). The update rate $\boldsymbol{\eta}$ is optimized for each case while $\boldsymbol{\alpha}$ is fixed at $\boldsymbol{\alpha = 0.9}$. We see that Schemes A and B all performed better than Ishiguro et. al' scheme, and Scheme B is the best as expected.

The performance of the uncompensated computed torque system (Figure 4.3 with $\boldsymbol{\tau_n = 0}$) controller is also listed in Table 4.3 for comparison. The improvement in accuracies provided by all the NN controller is clearly demonstrated.

Table 4.3: Circular tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized.)

	Scheme A	Scheme B	Scheme C	Ishiguro's	Uncompensated
$\boldsymbol{\eta}$	0.006	0.06	0.05	0.01	N/A
$\boldsymbol{\alpha}$	0.9	0.9	0.9	0.9	N/A
$\boldsymbol{E_p}$	0.0124	0.0034	0.0103	0.0244	27.8037
$\boldsymbol{E_v}$	0.5330	0.4586	2.9928	1.9926	25.3965
$\boldsymbol{E_c}$	0.0011	0.0002	0.0020	0.0045	8.5890

Table 4.4: Composite tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized.)

	Scheme A	Scheme B	Scheme C	Ishiguro's	Uncompensated
$\boldsymbol{\eta}$	0.005	0.03	0.01	0.0008	N/A
$\boldsymbol{\alpha}$	0.9	0.9	0.9	0.9	N/A
$\boldsymbol{E_p}$	0.20940	0.07511	3.5344	4.8749	109.0905

Table 4.4 lists the simulation results of tracking a composite trajectory. Again both Schemes A and B outperform Ishiguro et. al's scheme and Scheme B is the best. In both simulation cases, Scheme C has comparable performance as Ishiguro et. al's scheme. The tracking responses of Schemes A, B and Ishiguro et. al's scheme are shown in Figures 4.21-23, , respectively.

4.7.1.5 Discussion

A new neural network control technique for robot manipulator control is presented in this section. The NN controller serves as the inverse model of the computed-torque controlled robot system. Three possible schemes are introduced for implementing the control strategy with varying degree of complexities. Simulation studies on

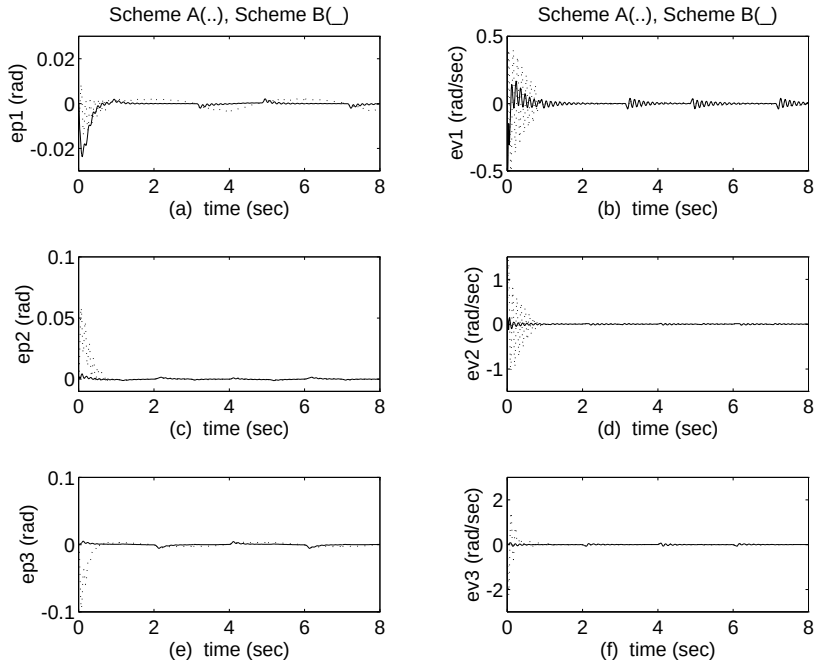


Figure 4.19: Joint Angle Tracking Errors for Circle Trajectory in Figure 4.18 where $\boldsymbol{e_p = q_d - q}$ $\boldsymbol{e_v = \dot{q}_d - \dot{q}}$.

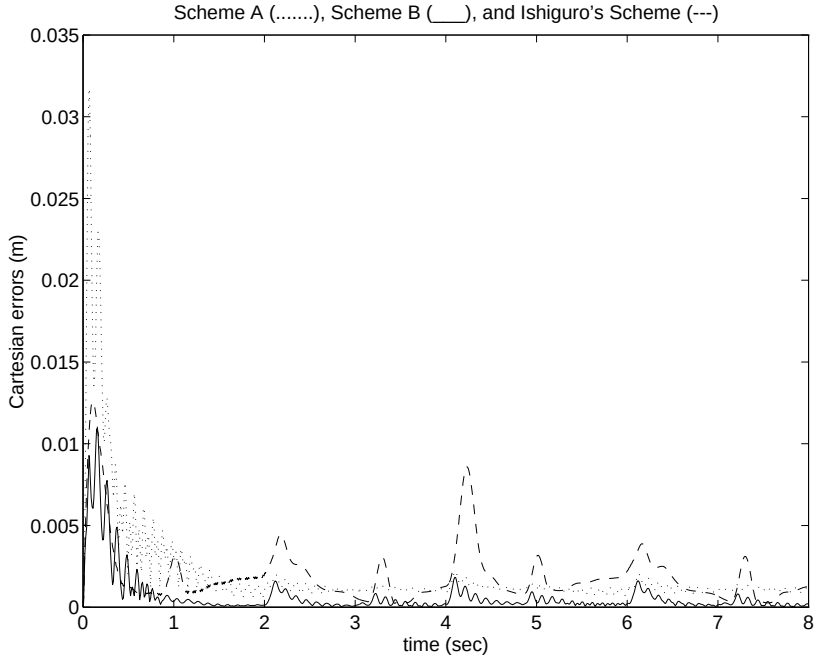


Figure 4.20: Cartesian errors of Circular Tracking for Scheme A,B, and Ishiguro's Scheme

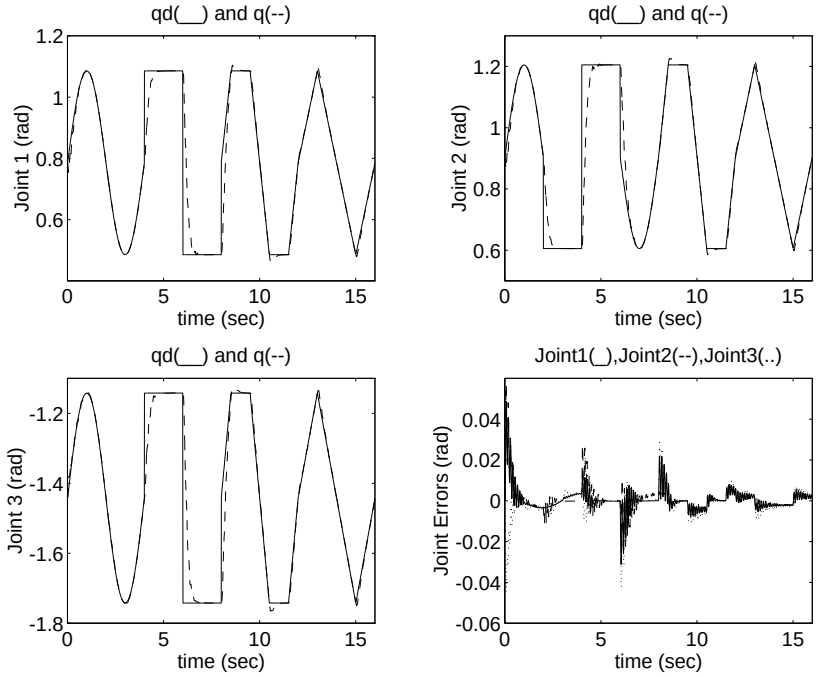


Figure 4.21: Composite Trajectory Tracking Response and Joint errors for Scheme A

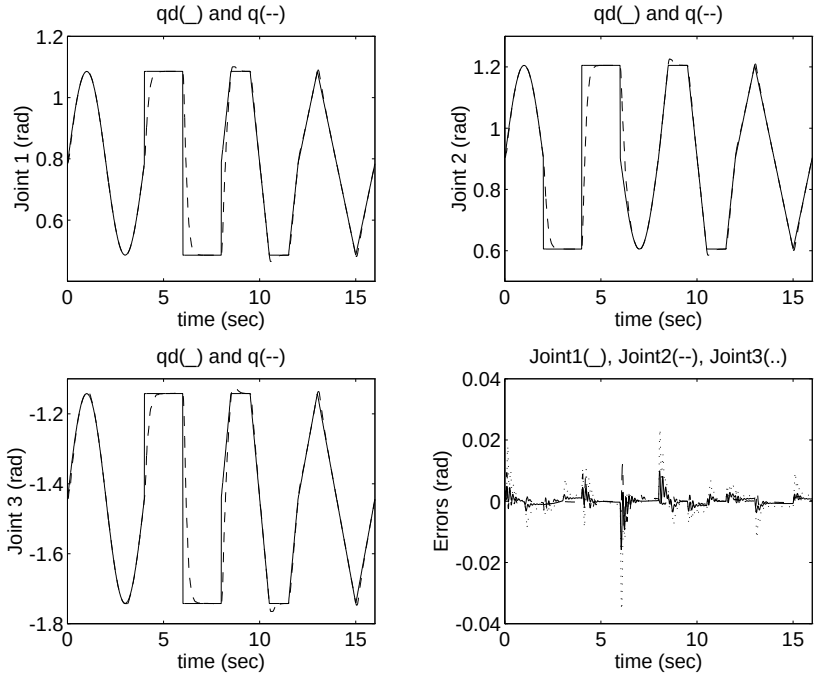


Figure 4.22: Composite Trajectory Tracking Response and Joint errors for Scheme B

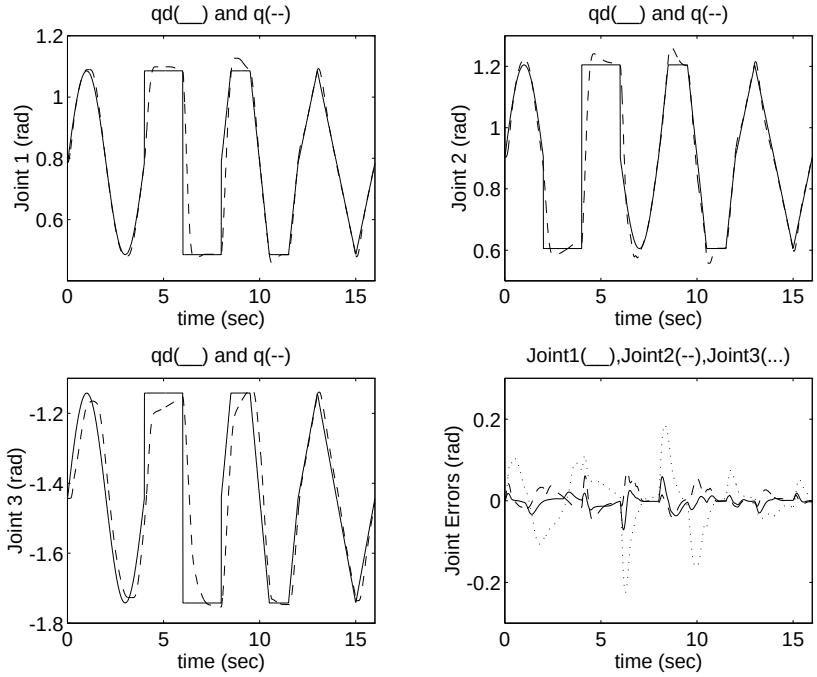


Figure 4.23: Composite Trajectory Tracking Response and Joint errors for Ishiguro's Scheme

trajectory tracking showed that the proposed control technique works extremely well and the controller converges very fast. The performance of Scheme B clearly demonstrates the superiority of the proposed technique over the existing technique such as that of Ishiguro et. al. Since the proposed schemes are aimed at modifying the desired input trajectory, it can be easily implemented at the command trajectory level planning of an existing controller without having to modify the controller's internal structure as would be required by other existing schemes.

4.7.2 Feedback Error Based Approach

In this approach, compensating locations are same as the previous Jacobian based approach, but the minimizing errors are differently formed. Instead of minimizing output errors separately, a feedback error function is minimized. By doing so, one benefit over Jacobian based approach is that Jacobian estimation is not required.

4.7.2.1 Control Law

This control structure is shown in Figure 4.24. The inputs to the neural controller are the desired trajectories, $\ddot{\boldsymbol{q}}_{\boldsymbol{d}}, \dot{\boldsymbol{q}}_{\boldsymbol{d}}, \boldsymbol{q}_{\boldsymbol{d}}$. The compensating signals from neural compensator, $\boldsymbol{\phi_a}, \boldsymbol{\phi_d}, \boldsymbol{\phi_q}$ are added to the desired trajectories.

The control law is

$$\boldsymbol{\tau} = \hat{\boldsymbol{D}}(\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + \boldsymbol{\phi_a} + \boldsymbol{K_D}(\dot{\boldsymbol{\varepsilon}} + \boldsymbol{\phi_d}) + \boldsymbol{K_P}(\boldsymbol{\varepsilon} + \boldsymbol{\phi_q})) + \hat{\boldsymbol{h}} \tag{4.50}$$

where $\boldsymbol{\phi_a}, \boldsymbol{\phi_q}$ and $\boldsymbol{\phi_d}$ are the outputs of NN. Combining (4.50) with (4.2) yields

$$\boldsymbol{v} = \ddot{\boldsymbol{\varepsilon}} + \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f}) - \boldsymbol{\Phi_A} \tag{4.51}$$

where $\boldsymbol{\Phi_A} = \boldsymbol{\phi_a} + \boldsymbol{K_D}\boldsymbol{\phi_d} + \boldsymbol{K_P}\boldsymbol{\phi_q}$. Ideally, at $\boldsymbol{v} = \boldsymbol{0}$, the ideal value of $\boldsymbol{\Phi_A}$ output of NN is

$$\boldsymbol{\Phi_A} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f}) \tag{4.52}$$

which are uncertainties in robot dynamic models and same as $\boldsymbol{\Psi_B}$.

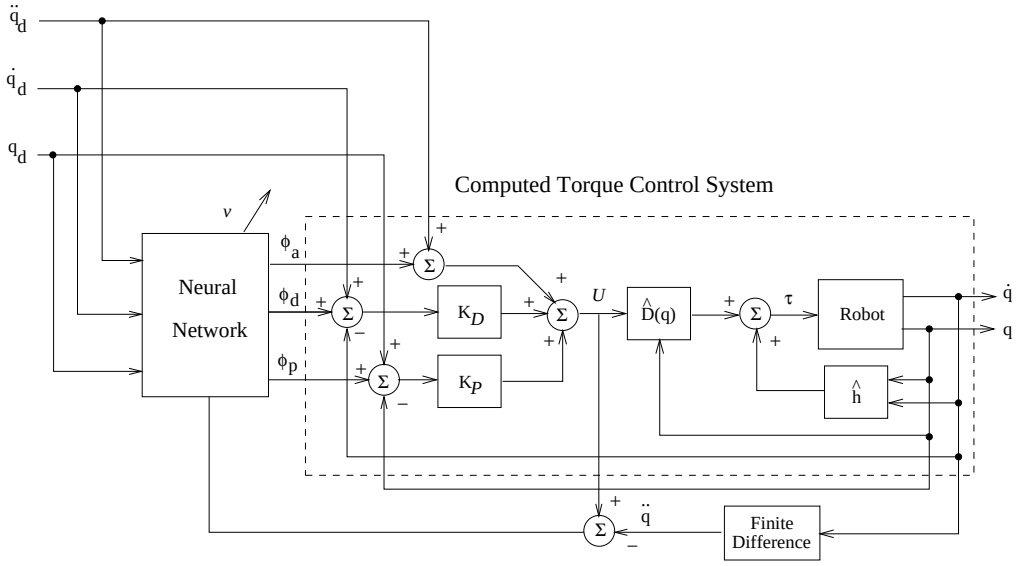


Figure 4.24: The Proposed Neural Network Control Structure

4.7.2.2 Simplification of NN Controller

The structure of NN can be simplified based on the number of outputs of NN. Only compensating at position and velocity trajectory reduces the number of neurons inside without losing accuracy. We call this two state compensation scheme as Scheme B. In this case, the control law is

$$\boldsymbol{\tau} = \hat{\boldsymbol{D}}(\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + \boldsymbol{K_D}(\dot{\boldsymbol{\varepsilon}} + \boldsymbol{\phi_d}) + \boldsymbol{K_P}(\boldsymbol{\varepsilon} + \boldsymbol{\phi_q})) + \hat{\boldsymbol{h}} \tag{4.53}$$

And the closed loop equation is

$$\boldsymbol{v_{PD}} = \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f}) - \ddot{\boldsymbol{\varepsilon}} - \boldsymbol{\Phi_B} \tag{4.54}$$

where $\boldsymbol{\Phi_B} = (\boldsymbol{K_D}\boldsymbol{\phi_d} + \boldsymbol{K_P}\boldsymbol{\phi_q})$. The neural controller in Scheme B has to compensate for more complicated signal than that of Scheme A.

It can be further simplified by compensating only at position trajectory such as

$$\boldsymbol{\tau} = \hat{\boldsymbol{D}}(\ddot{\boldsymbol{q}}_d + \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}(\boldsymbol{\varepsilon} + \boldsymbol{\phi}_q)) + \hat{\boldsymbol{h}} \tag{4.55}$$

And the closed loop equation is

$$\boldsymbol{v_P} = \boldsymbol{K_P}\boldsymbol{\varepsilon} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f}) - \ddot{\boldsymbol{\varepsilon}} - \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} - \boldsymbol{\Phi_B} \tag{4.56}$$

where $\boldsymbol{\Phi_C} = \boldsymbol{K_P}\boldsymbol{\phi}_q$. This is called Scheme C. Note that Scheme C minimizes only the position errors.

4.7.2.3 Neural Network Controller Design

Since we are minimizing the error function $\boldsymbol{v}$ the objective function is

$$\mathcal{J} = \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v} \tag{4.57}$$

Differentiating (4.57) yields the gradient as

$$\frac{\boldsymbol{\partial}\mathcal{J}}{\boldsymbol{\partial w}} = \frac{\boldsymbol{\partial v}^{\boldsymbol{T}}}{\boldsymbol{\partial w}}\boldsymbol{v} = -\frac{\boldsymbol{\partial \Phi}^{\boldsymbol{T}}}{\boldsymbol{\partial w}}\boldsymbol{v} \tag{4.58}$$

in which the fact $\frac{\boldsymbol{\partial v}^{\boldsymbol{T}}}{\boldsymbol{\partial w}} = -\frac{\boldsymbol{\partial \Phi}^{\boldsymbol{T}}}{\boldsymbol{\partial w}}$ is used. The back-propagation updating rule for the weights with a momentum term is

$$\boldsymbol{\Delta w(t)} = \boldsymbol{\eta}\frac{\boldsymbol{\partial v}^{\boldsymbol{T}}}{\boldsymbol{\partial w}}\boldsymbol{v} + \boldsymbol{\alpha}\boldsymbol{\Delta w(t-1)} \tag{4.59}$$

where $\boldsymbol{\eta}$ is the update rate and $\boldsymbol{\alpha}$ is the momentum coefficient.

The neural network structure is same as shown in Figure 4.16.

Table 4.5: Circular tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized.) under high feedback controller gains ($\boldsymbol{K_D}$ = diag[**50 50 50**], $\boldsymbol{K_P}$ = diag[**625 625 625**])

	Scheme A	Scheme B	Scheme C
$\boldsymbol{\eta}$	0.0001	0.0002	0.00001
$\boldsymbol{\alpha}$	0.9	0.9	0.9
$\boldsymbol{E_p}$	0.00043	0.00028	0.00436
$\boldsymbol{E_v}$	0.82856	1.10915	0.63987
$\boldsymbol{E_c}$	0.000086	0.000055	0.000679

4.7.2.4 Comparison Studies between Different Neural Controllers

The robot is required to follow the same circular trajectory shown in the previous simulations. Two tables are provided here : one is when the feedback controller gains are large, and another one is when the feedback controller gains are low. We like to test three schemes shown above and to see the relationship between the locations of compensation and the performances. Table 4.5 shows the results when high gains are used. All controllers' performances are very good. The performances of Scheme A and B are excellent and very comparable while that of Scheme C performs about 10 times worse. This result shows that Scheme B is the best choice among three schemes in performance and efficiency of neural network structure.

Table 4.6 lists the results when the feedback gains are low. The notable poor performance of Scheme C can be observed while those of Scheme A and B are still comparable. As we can from the results in Table 4.6, only one state compensation is not good idea even though neural controller structure becomes simpler. It is also proved that at least two state compensation, position and velocity, is required to have

Table 4.6: Circular tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized.) under low feedback controller gains($\boldsymbol{K_D}$ = diag[**10 10 10**], $\boldsymbol{K_P}$ = diag[**25 25 25**])

	Scheme A	Scheme B	Scheme C
$\boldsymbol{\eta}$	0.0045	0.0045	0.00001
$\boldsymbol{\alpha}$	0.9	0.9	0.9
$\boldsymbol{E_p}$	0.006155	0.00665	32.0186
$\boldsymbol{E_v}$	0.82415	0.083	80.1902
$\boldsymbol{E_c}$	0.00093	0.00097	6.4688

the robustness to feedback gain variations as well as good tracking performances.

4.7.2.5 Comparison Studies between Different Training Signals

It is interesting to see the performances for different training signals. In the previous simulation studies, the second order error equation $\boldsymbol{v}$ is used as a training signal for Scheme A. The acceleration term $\ddot{\boldsymbol{q}}$ is obtained from the finite difference method such that

$$\ddot{\boldsymbol{q}}(\boldsymbol{t}) = \frac{\dot{\boldsymbol{q}}(\boldsymbol{t}) - \dot{\boldsymbol{q}}(\boldsymbol{t} - \boldsymbol{\lambda})}{\boldsymbol{\lambda}} \tag{4.60}$$

where $\boldsymbol{\lambda}$ is a sampling time.

Table 4.7 lists the circular trajectory tracking results of Scheme B with three different training signals : $\boldsymbol{v}$, $\boldsymbol{v_{PD}} = \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon}$, and $\boldsymbol{v_P} = \boldsymbol{K_P}\boldsymbol{\varepsilon}$. It clearly shows that the performance with the training signal $\boldsymbol{v}$ is the best, and then $\boldsymbol{v_{PD}}, \boldsymbol{v_P}$ in order. It is true from (4.54) that using the $\boldsymbol{v_{PD}}$ training signal has to compensate for the acceleration error in addition to robot uncertainties.

Table 4.7. Performances with different inputs and trajectories during the second iteration($\boldsymbol{\eta}$ is optimized.)

	High Gains			***Low Gains***		
	$\boldsymbol{v}$	$\boldsymbol{v_{PD}}$	$\boldsymbol{v_P}$	$\boldsymbol{v}$	$\boldsymbol{v_{PD}}$	$\boldsymbol{v_P}$
$\boldsymbol{\eta}$	0.0002	0.000009	0.000005	0.0045	0.0045	0.00001
$\boldsymbol{E_p}$	0.00028	0.00609	0.061715	0.006652	0.013218	76.2080
$\boldsymbol{E_v}$	1.10915	1.80222	7.8955	0.083	10.6534	189.6862
$\boldsymbol{E_c}$	0.000055	0.00106	0.012793	0.000974	0.003	13.0064

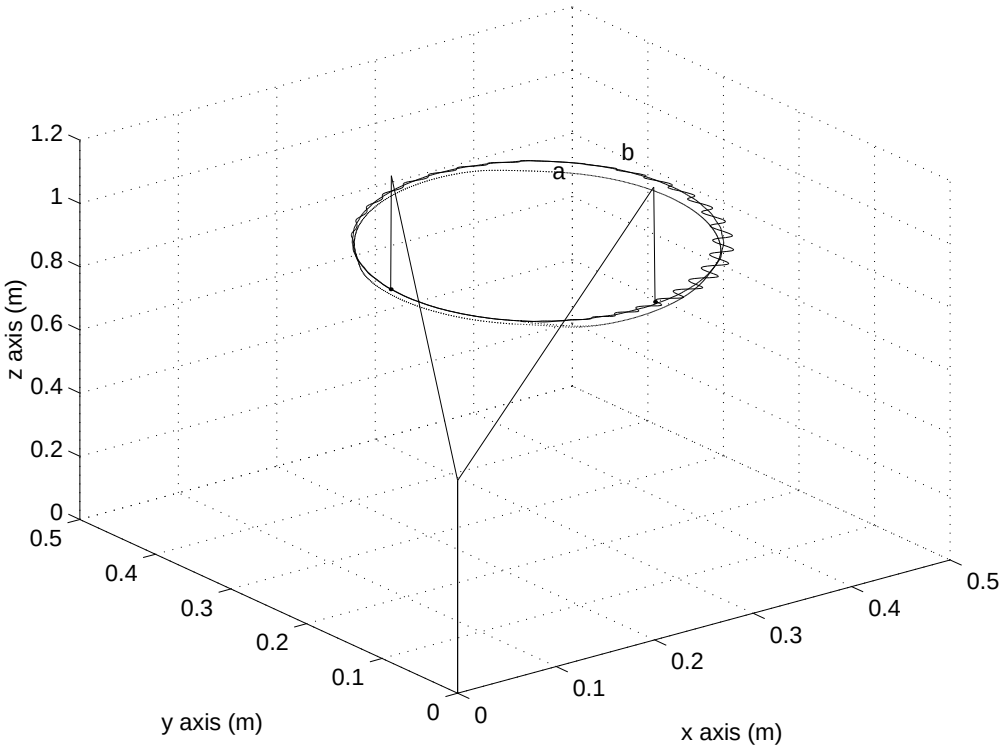


Figure 4.25: End point trajectory with high PD gains : (a)uncompensated (b) Scheme A (c) Reference

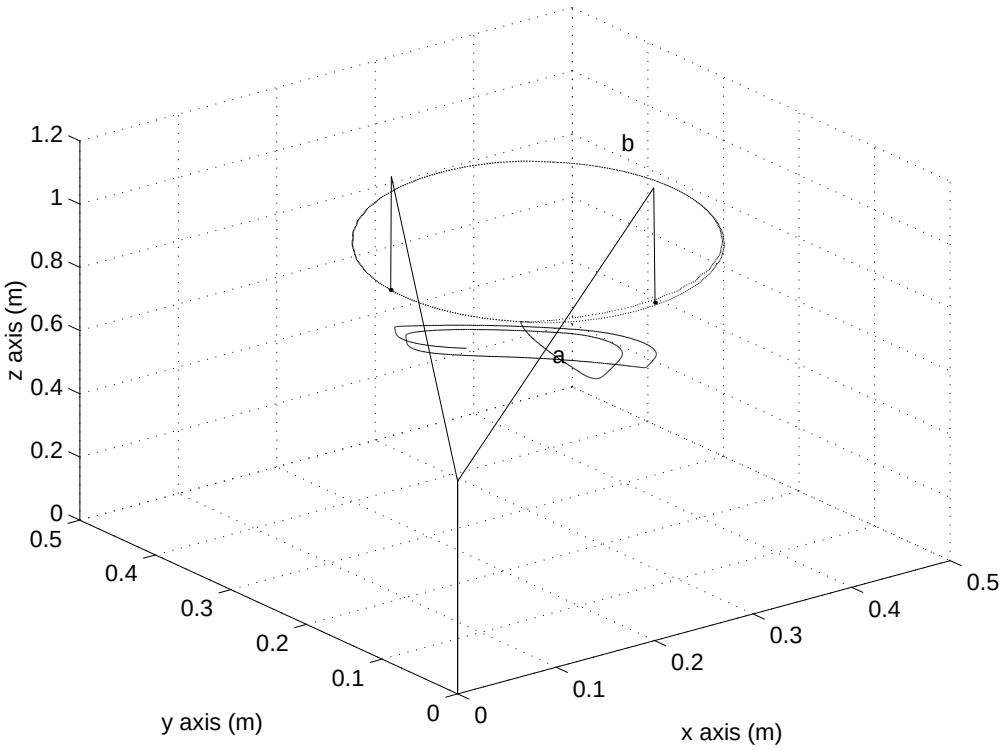


Figure 4.26: End point trajectory with low PD gains : (a)uncompensated (b) Scheme A

Table 4.7: Different Model Based Feedforward schemes

	Auxiliary Compensator		Inverse Compensator		
Schemes	FF/FB	Ishiguro	Prefilter	Modified	**RCT**
Objective Function	$\frac{1}{2}\boldsymbol{v}^T\boldsymbol{v}$	$\frac{1}{2}\boldsymbol{\tau}_e^T\boldsymbol{\tau}_e$	$\frac{1}{2}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}$	$\frac{1}{2}\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}$	$\frac{1}{2}\boldsymbol{v}^T\boldsymbol{v}$
Jacobian	No	No	Yes	Yes	**No**
Training Signal	$\boldsymbol{v}$	$\boldsymbol{\tau} - \boldsymbol{\tau}_m - \boldsymbol{\tau}_n$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{v}$
No. of hidden neurons	6	6	6	6	6
No. of output neurons	$\boldsymbol{n}$	$\boldsymbol{n}$	$\boldsymbol{3n}$	$\boldsymbol{3n}$	$\boldsymbol{3n}$
No. of weights	81	81	123	123	123
Compensating location	$\boldsymbol{u}$	$\boldsymbol{\tau}$	$\boldsymbol{q_d}$	$\boldsymbol{q_d}$	$\boldsymbol{q_d}$
Output at convergence	$\hat{\boldsymbol{D}}\boldsymbol{\delta}$	$\boldsymbol{\delta}$	$\boldsymbol{delta + Q}$	$\boldsymbol{\delta}$	$\boldsymbol{\delta}$
Output of NN	$\boldsymbol{\Phi}$	$\boldsymbol{\tau_n}$	$\boldsymbol{\Psi_A}$	$\boldsymbol{\Psi_B}$	$\boldsymbol{\Phi}$
$\boldsymbol{E_P}$ for Circle	0.0036	0.0244	0.0124	0.0034	**0.00043**
$\boldsymbol{E_P}$ for Composite	0.1279	4.4875	0.2095	0.07511	**0.0235**

4.8 Summary of Model Based Approach

Here, different characteristics of schemes discussed previous section are tabulated. The robot model has three rotary joints.

We note that the relationships between NN output magnitudes $\|\boldsymbol{\tau_n}\| > \|\boldsymbol{\Psi_A}\| > \|\boldsymbol{\Phi}\| = \|\boldsymbol{\Psi_B}\|$ hold. The performances listed in Table 4.5 for each scheme shows that the smaller magnitude of NN is the smaller error results. Based on performances the modified prefilter type is the best choice among other controllers. This phenomena is investigated more extensively in next chapter. In next chapter, we also propose a prefilter type NN control scheme that does not require Jacobian information.

4.9 Summary

In this chapter, neural network compensators are used to compensate for partially unknown robot dynamic models. The proposed FF/FB NN control schemes were compared with the well known adaptive control technique and one neural network control technique proposed by Ishiguro et. al.. Simulation results showed the better tracking performances of the proposed scheme. Neural network parameters are optimized base on trial and error basis. The learning rate is desired as large as possible just before the system goes instability. We also found that the larger number of hidden units do not always give good performance. The initial weights can be either set zeros or small random numbers that do not effect initial behavior of the system.

As the second half of this chapter, conceptually different technique called the inverse control technique has also been proposed. Depending on the way of minimizing the errors, two approaches were presented. One is Jacobian based approach, another is feedback error approach. Obviously, the latter approach has advantage of simple implementation by not requiring Jacobian informations, at the same time of comparable performances. Within feedback error approach, tracking performances were compared based upon different structures of neural controller and different training signals were tested. It is found that at least PD feedback signal is required to achieve good performance as well as robustness.

Most of all, one very good advantage of the proposed inverse control technique is

that it can be implemented as an auxiliary device without modifying inside control structure. Thus, practical implementation is easy.

Chapter 5

Non-Model Based Neural Network Robot Position Control

5.1 Introduction

We have shown in the previous chapter that model based NN controllers perform very well. However, the real value of NN application to robot control system is not requiring any knowledge of the robot model at all. In practical situations, many robots have many joints and difficult configurations so that the exact estimation of dynamic equation becomes very difficult. In such a case it is better to control the robot without knowing any robot dynamics. This requires the full usages of the potential NN ability of universal approximation.

Within this framework, the use of NN controllers for non-model based robot control is basically to identify inverse dynamics of robot system. One approach called the direct NN inverse control technique which uses the NN as an inverse controller has been proposed (see when $\boldsymbol{K_P} = \boldsymbol{K_D} = \mathbf{0}$ and $\boldsymbol{v} = [\boldsymbol{\varepsilon}^{\boldsymbol{T}}\ \dot{\boldsymbol{\varepsilon}}^{\boldsymbol{T}}]^{\boldsymbol{T}}$ in Figure 5.1) [15]. This approach has difficulty of on-line implementation because of unavailability of Jacobian of the robot and lack of stability, whose performance is heavily dependent upon its initial weight values and update rate. Providing Jacobian information by another NN which identifies the forward model of a plant helps the performance at the beginning of process, but this may have still convergence problems overall because it takes time for the plant model to converge before the NN controller can be properly converged [19, 23, 24].

Thus in on-line inverse NN control application the stability of the robot system must be assured before applying NN controllers. This can be done simply by em-

ploying PD feedback controllers as shown in Figure 5.1. The scheme in Figure 5.1 called feedback error learning [31]can be considered as an auxiliary type inverse NN controller while the scheme proposed in [66] is considered as a prefilter type inverse NN controller.

Here we propose a new auxiliary type NN inverse control scheme and a prefilter type NN inverse control scheme for PD controlled robot manipulator. Identification and control of a stabilized robot system is achieved by modifying the desired trajectory for each joint, $\boldsymbol{q_d}$ $\dot{\boldsymbol{q}}_{\boldsymbol{d}}$, using appropriately trained NN serving as a nonlinear filter as depicted in Figure 5.2. In auxiliary type control, the signal $\boldsymbol{v} = \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon}$, is used as a unified training signal while $\boldsymbol{\mathcal{E}} = [\boldsymbol{\varepsilon}^{\boldsymbol{T}}\dot{\boldsymbol{\varepsilon}}^{\boldsymbol{T}}]^{\boldsymbol{T}}$ is used as a training signal for prefilter type control. In this arrangement, the NNs of both types act as the inverse model of the robot system. Hence, we refer the proposed techniques as a non-model based neural network inverse control.

We can assume that a nominal PD control system is stable when encountered in performing tasks [77]. Under this condition, the robot tracking error satisfies approximately the first order linear differential equation. In the following, we will present implementations of the FEL scheme and the proposed NN control schemes. The performance of PD control is very sensitive to gains of controller so that high gains are usually desirable for better tracking as long as the closed loop is stable [78]. We will also present controller training algorithms as well as simulation results for on-line trajectory tracking under high and low feedback controller gains to investigate the NN

controller's robustness to gain changes which is important in practical implementation and cost.

5.2 PD Control

PD control compensation for robot dynamic control has been studied in many papers because of its simplicity and popularity [78]. The desired control law is composed of position and velocity displacements such as (see when $\boldsymbol{\Phi_\tau = 0}$ in Figure 5.1)

$$\boldsymbol{\tau = K_D\dot{\varepsilon} + K_P\varepsilon} \tag{5.1}$$

where $\boldsymbol{\varepsilon = q_d - q}$. Combining the robot dynamic equation (4.2) and (5.1), we have a closed loop error equation as follows

$$\boldsymbol{D\ddot{\varepsilon} + K_D\dot{\varepsilon} + K_P\varepsilon = D\ddot{q}_d + h + \tau_f} \tag{5.2}$$

Multiplying both sides by $\boldsymbol{D^{-1}}$ yields

$$\boldsymbol{\ddot{\varepsilon} + D^{-1}K_D\dot{\varepsilon} + D^{-1}K_P\varepsilon = D^{-1}(D\ddot{q}_d + h + \tau_f)} \tag{5.3}$$

Clearly, there is no way to control the robot dynamics correctly since the error, $\boldsymbol{\varepsilon}$, behaves differently as configuration of the robot arm changes. Diagonal high feedback gains are usually desired to achieve better tracking performance as long as the closed loop system is stable since coupling effects due to other joint motion are minimized. If gains are selected such that $\boldsymbol{K_D \gg 1}$ and $\boldsymbol{K_P \gg 1}$, then (5.3) can be approximated

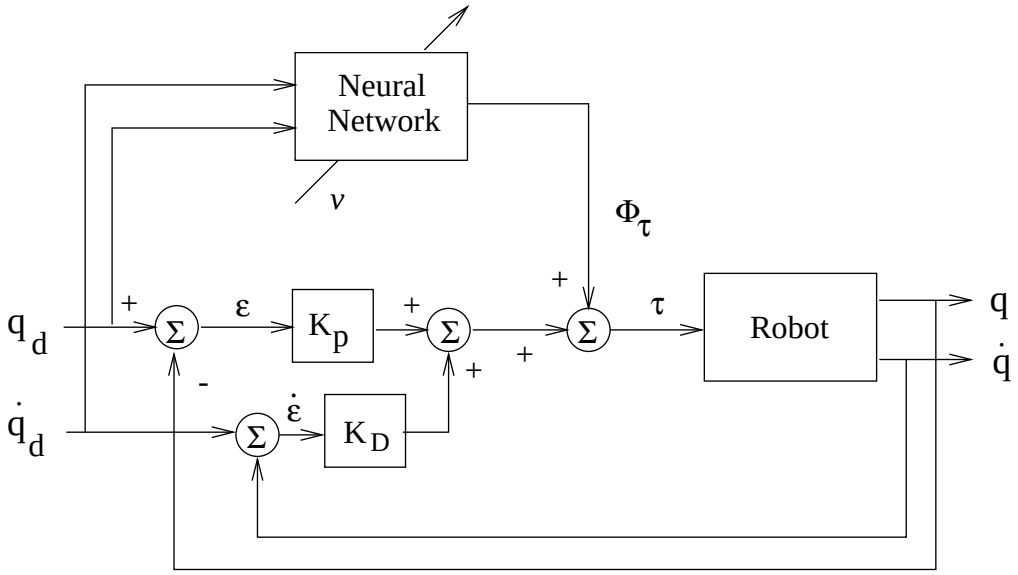


Figure 5.1: Feedback Error Learning Scheme(FEL I)

as the first order differential equation

$$\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_D^{-1}K_P\varepsilon} \cong -\boldsymbol{K_D^{-1}D\ddot{\varepsilon}} + \boldsymbol{K_D^{-1}}(\boldsymbol{D\ddot{q}_d + h + \tau_f}) \cong \mathbf{0} \tag{5.4}$$

In practice, there are limitations of selecting high gains due to the stability and hardware cost. Thus the choice of feedback gains has the crucial effects on performance in PD type controlled robot manipulators, and this has been proved in simulation studies.

5.3 Auxiliary Type FFNN Controller Scheme

5.3.1 Feedback Error Learning Controller Scheme

The FEL control scheme is examined here, and it is depicted in Figure 5.1. The FEL scheme presented here is slightly different from one in [31] in that pure feedforward NN is used instead of modified NN having subsystems of priori known robot dynamic elements. In this scheme, the NN output $\boldsymbol{\Phi_\tau}$ is a torque signal from inverse dynamic model identified by NN controller. The compensating control law is

$$\boldsymbol{\tau = K_D\dot{\varepsilon} + K_P\varepsilon + \Phi_\tau} \tag{5.5}$$

Combining the robot dynamic equation(4.2) and the control law(5.5), error dynamic equation can be written as

$$\boldsymbol{K_D\dot{\varepsilon} + K_P\varepsilon + \Phi_\tau = D\ddot{q} + h + \tau_f} \tag{5.6}$$

Define the training signal $\boldsymbol{v}$ as $\boldsymbol{v = K_D\dot{\varepsilon} + K_P\varepsilon}$, then (5.6) is rewritten as

$$\boldsymbol{\tau_e = v = D\ddot{q} + h + \tau_f - \Phi_\tau} \tag{5.7}$$

Thus, at the convergence, the ideal output of NN is expected to be the inverse dynamics of robots as

$$\boldsymbol{\Phi_\tau = D\ddot{q} + h + \tau_f} \tag{5.8}$$

The FEL control performs identification and control at the same time in on-line fashion.

5.3.2 Proposed Reference Compensation Technique

Here we propose a new auxiliary NN control scheme for PD controlled robot manipulator as depicted in Figure 5.2. This scheme is same as the reference compensation technique(RCT) that have been implemented for model based case in Chapter 4. In non-model based approach, model estimations are not required.

From Figure 5.2, the closed loop equation can be derived as

$$\boldsymbol{K_D\dot{E} + K_P E = D\ddot{q} + h + \tau_f} \tag{5.9}$$

where $\boldsymbol{E = q_r - q}$. Since a linear relationship between $\boldsymbol{E}$ and $\boldsymbol{\phi_p}$ is hold $\boldsymbol{\dot{e}}$ can be obtained by differentiating $\boldsymbol{e}$ such that $\boldsymbol{\dot{E} = \dot{q}_r - \dot{q} = \dot{q}_d + \dot{\phi}_p - \dot{q}}$ where $\boldsymbol{\dot{\phi}_p}$ is estimated from finite difference method as

$$\boldsymbol{\dot{\phi}_p(t) \cong \frac{\phi_p(t) - \phi_p(t-1)}{T}} \tag{5.10}$$

where $\boldsymbol{T}$ is a sampling time. The reason of using finite difference approximation for $\boldsymbol{\dot{\phi}}$ instead of using direct output from NN is to be consistent in dimension of outputs with FEL scheme.

Substituting $\boldsymbol{\dot{E} = \dot{q}_d + \dot{\phi}_p - \dot{q}}$ and $\boldsymbol{E = q_d + \phi_p - q}$ into (5.9) yields

$$\boldsymbol{v = D\ddot{q} + h + \tau_f - \Phi_q} \tag{5.11}$$

where $\boldsymbol{\Phi_q = K_D\dot{\phi}_p + K_P\phi_p}$. Compensation is done in input trajectory level and those compensating signals are amplified through primary feedback controller gains so that the magnitudes of those compensating signals from NN are expected to be very

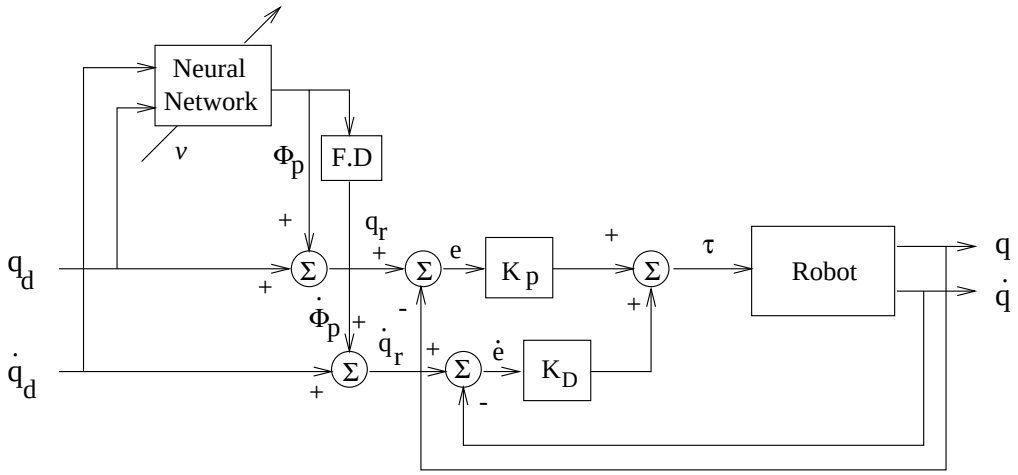


Figure 5.2: The Proposed Neural Control Scheme 1 : RCT I

small compared with one in the FEL controller which actually compensates at torque level. At convergence, when inverse identification is done, the ideal compensation of NN is also same as

$$\boldsymbol{\Phi_q} = \boldsymbol{D\ddot{q} + h + \tau_f} \tag{5.12}$$

which is same as (5.8). Thus the proposed Scheme 1 and the FEL scheme generate the same compensating term of inverse model through either input level or joint level so that the robot response $\boldsymbol{q}$ tracks $\boldsymbol{q_d}$ with minimal distortion. However, the ideal output of NN from (5.11) is

$$\boldsymbol{\phi_p = K_P^{-1}[D\ddot{q} + h + \tau_f - K_D\dot{\phi}_p]} \tag{5.13}$$

which is much smaller than $\boldsymbol{\Phi_\tau}$ in (5.8). Comparing with the FEL controller, the performance(accuracy and convergence) of the proposed scheme will be better not only because the dimensionality of searching degrees-of-freedom(DOF) for optimal

value is larger than the FEL's(ϕ_p and ϕ_d instead of ϕ_τ) but also the magnitude of disturbances to be compensated are reduced by K_P^{-1} even though the dimension of NN outputs is limited to same as FEL's for fair comparison. The real advantage of the proposed NN controller over the FEL NN controller is simpler practical implementation. Functioning as a trajectory generator, compensation can be done outside of control loop, while compensation of the FEL controller is done inside control loop. NN compensation outside control loop allows more robustness so that the performance under low primary controller gains is excellent.

5.3.3 Auxiliary Type Neural Network Controller Design

In order to design NN controller, we use multilayer feedforward NN(FFNN) and BP learning algorithms. It is composed of an input buffer, a non-linear hidden layer and a linear output layer. The weight updating law minimizes the objective function $\mathcal{J}$ which is a quadratic function of the unified training signal v which is composition of system errors $\varepsilon, \dot{\varepsilon}$.

$$\mathcal{J} = \frac{1}{2}\tau_e^T \tau_e = \frac{1}{2} v^T v \tag{5.14}$$

where $v = \tau_e = K_D\dot{\varepsilon} + K_P\varepsilon$ from (5.7). Differentiating equation(5.14) and making use of (5.11) yields the gradient of $\mathcal{J}$ as follows:

$$\frac{\partial \mathcal{J}}{\partial w} = \frac{\partial v^T}{\partial w} v = -\frac{\partial \Phi_q^T}{\partial w} v \tag{5.15}$$

since $\frac{\partial v^T}{\partial w} = -\frac{\partial \Phi_q^T}{\partial w}$ and $\frac{\partial v^T}{\partial w} = -\frac{\partial \Phi_\tau^T}{\partial w}$ for the FEL scheme.

The BP update rule for the weights with a momentum term is

$$\Delta \boldsymbol{w}(t) = \eta \frac{\partial \boldsymbol{\Phi}^T}{\partial \boldsymbol{w}} \boldsymbol{v} + \alpha \Delta \boldsymbol{w}(t-1) \tag{5.16}$$

where η is the update rate and α is the momentum coefficient. The selection of η is very important for performance of the NN controller as well as stability. Usually a larger η is desired for better performance as long as the system is stable [67]. How to select the optimal η is one area of on-going researches.

5.3.4 Simplification for Controller Design

Simplification of Figure 5.2 can be designed as shown in Figure 5.3. This is called RCT II. From Figure 5.3, the inverse model control is performed by adjusting position trajectory only and no compensation is carried out on velocity trajectory such that $\dot{\boldsymbol{q}}_r = \dot{\boldsymbol{q}}_d$. Substituting $\dot{\boldsymbol{E}} = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}$ and $\boldsymbol{E} = \boldsymbol{q}_d + \boldsymbol{\phi}_p - \boldsymbol{q}$ into (5.6) yields

$$\boldsymbol{v} = \boldsymbol{D}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{\tau}_f - \boldsymbol{K}_P \boldsymbol{\phi}_p \tag{5.17}$$

which is a simplification of (5.13). Thus it has less dimensionality of DOF for optimization than (5.13), but same as (5.12). The ideal output of NN at convergence is

$$\boldsymbol{\phi}_p = \boldsymbol{K}_P^{-1}(\boldsymbol{D}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{\tau}_f) \tag{5.18}$$

which is reduced from (5.12) by the factor of $\boldsymbol{K}_P^{-1}$. Further simplification of Figure 5.3 can be done as shown in Figure 5.2 (b). The inputs to NN are limited to position variables only so that NN structure is further simplified. The corresponding closed

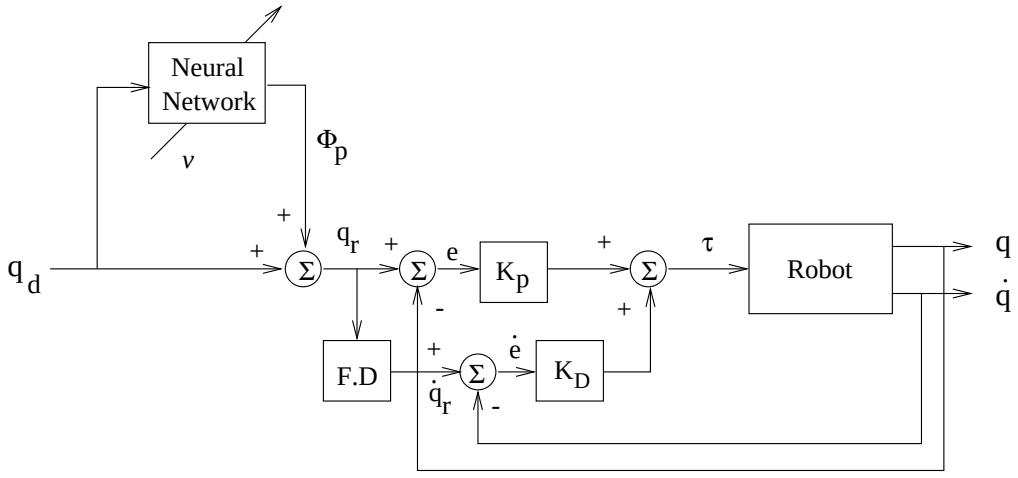


Figure 5.3: Reference compensation technique : Scheme II where $\boldsymbol{v} = \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} = \boldsymbol{q_d} - \boldsymbol{q}$, and F.D is a finite difference method

loop equation is same as (5.13) of Scheme 1 except that instead of using $\dot{\boldsymbol{q}}_{\boldsymbol{r}} = \dot{\boldsymbol{q}}_{\boldsymbol{d}} + \dot{\boldsymbol{\phi}}_{\boldsymbol{p}}$ calculated directly from a linear relationship $\boldsymbol{q_r} = \boldsymbol{q_d} + \boldsymbol{\phi_p}$, $\dot{\boldsymbol{q}}_{\boldsymbol{r}}$ is approximated by finite difference method based on the reference position trajectory $\boldsymbol{q_r}$ as

$$\dot{\boldsymbol{q}}_{\boldsymbol{r}}(\boldsymbol{t}) \cong \frac{\boldsymbol{q_r(t)} - \boldsymbol{q_r(t-1)}}{\boldsymbol{T}} \tag{5.19}$$

so that dimensionality of DOF for optimization is same as Scheme 1(RCT I). At convergence, the ideal output of NN is

$$\boldsymbol{\phi_p} = \boldsymbol{K_P^{-1}}(\boldsymbol{D}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{\tau_f} - \boldsymbol{K_D}\dot{\boldsymbol{\phi}}_{\boldsymbol{p}}) \tag{5.20}$$

which is same as Scheme 1. Thus Scheme 3 can not only perform as good as Scheme 1 but also reduce the cost of NN by reducing the number of internal weights by about $\frac{1}{3}$. However, a little larger initial errors are expected because of one state compensation.

Figure 5.4 shows the scheme that is similar to FEL scheme. This scheme is called FEL II. n this scheme the NN outputs are multiplied by PD gains. With this

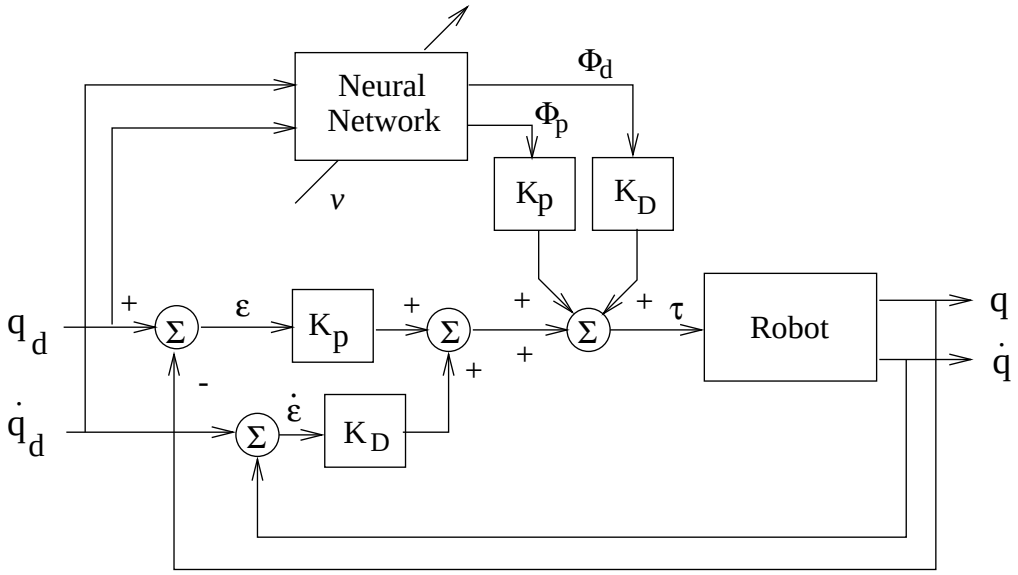


Figure 5.4: Feedback error learning control structure equivalent to RCT Scheme I(FEL II)

formulation, the structure of Figure 5.2 and Figure 5.4 are equivalent so that the same performances are expected in linear control theory. However, we note from simulation studies that the performances between two schemes are different. More detailed comparison studies are shown in next section.

5.4 NN Controller Performances

Here three different neural controllers, RCT I, RCT II, and FEL, are tested under the same conditions. The PD gains are selected as high and low. The high controller gains are selected as $\boldsymbol{K_D = diag[50, 50, 50]}$ and $\boldsymbol{K_P = diag[625, 625, 625]}$ while the low controller gains are selected as $\boldsymbol{K_D = diag[10, 10, 10]}$ and $\boldsymbol{K_P = diag[25, 25, 25]}$. The performances are measured by the total tracking errors com-

puted over one cycle of the trajectory as follows:

$$\boldsymbol{E_p} = \sum_{t=1}^{P/T} \| \boldsymbol{q_d} - \boldsymbol{q} \|^2 \; \boldsymbol{E_v} = \sum_{t=1}^{P/T} \| \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}} \|^2 \; \boldsymbol{E_c} = \sum_{j=1}^{P/T} [(\boldsymbol{x}_d - \boldsymbol{x})^2 + (\boldsymbol{y}_d - \boldsymbol{y})^2 + (\boldsymbol{z}_d - \boldsymbol{z})^2] \tag{5.21}$$

where $\boldsymbol{P = 4secs}$ for circular trajectory and $\boldsymbol{P = 12secs}$ for composite trajectory. Under these specifications all NN controllers are tested. First, the performances of NN controllers are tested by commanding the end point to track a circular trajectory as shown in Figure 7. Table 5.1 lists the results of three different schemes for circular trajectory with $\boldsymbol{E_p}\ \boldsymbol{E_v}\ \boldsymbol{E_c}$ computed during the second cycle of tracking under large primary controller gains in joint space. We see that all NN schemes performed very well compared with uncompensated PD control scheme. The FEL scheme II which is equivalent to RCT scheme I poorer performance than that of RCT schemes. The plots shown in Figure 5.5 visualizes the difference in performances between the proposed RCT Schemes I, II and FEL scheme I. The improvement in accuracies provided by the NN controller is clearly demonstrated. Also plotted in Figure 5.6 is the Cartesian errors for circular trajectory. The convergences of RCT NN controllers are faster than that of FEL. Figure 5.7 shows the detailed joint errors. We clearly see that the errors of FEL are the largest.

Also listed in Table 5.2 is that performances of NN controllers for composite trajectory in joint space under the same control environment. All NN schemes perform well with much better performance of the proposed RCT Scheme I. For simplicity, the tracking responses of the proposed RCT Scheme I are only plotted in Figure

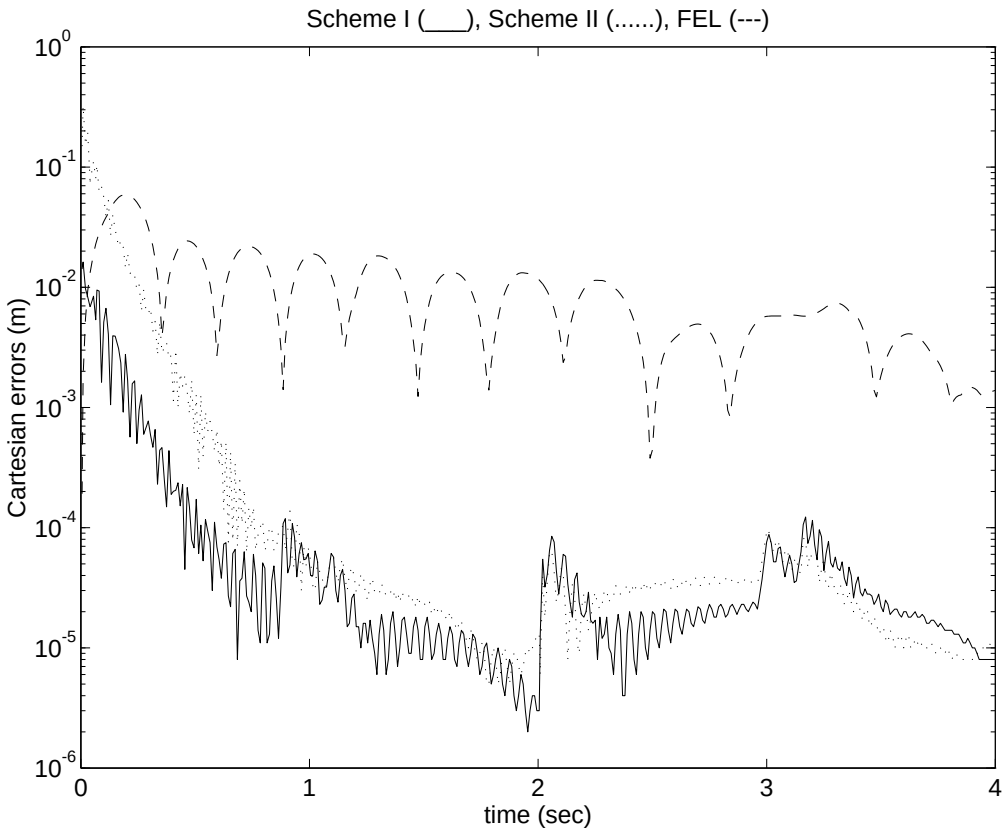


Figure 5.5: Cartesian errors of End Point Circular Trajectory

5.8. The desired and actual tracking plots of RCT I and II are not distinguishable clearly because of good tracking. It also shows that the NN controller converges immediately. The joint errors of composite trajectory tracking are clearly shown in Figure 5.9. In order to see the robustness of NN controllers to variation of feedback gains the same trajectory tracking tests are performed with relatively small magnitude of primary controller gains under the same control environment. The performance of FEL for composite trajectory tracking is so poor that the results are not listed in

Table 5.1: Circular tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized, $\boldsymbol{\alpha} = \mathbf{0.9}$ is used, $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{50\ 50\ 50}], \boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{625\ 625\ 625}]$.)

Schemes	***RCT I***	***RCT II***	***FEL I***	***FEL II***	***PD Control***
$\boldsymbol{\eta}$	0.0008	0.0002	0.0008	0.000003	N/A
$\boldsymbol{E_p}$	0.000002	0.000003	0.023981	0.02839	23.0640
$\boldsymbol{E_v}$	0.00444	0.00343	1.790924	1.75617	2.228624
$\boldsymbol{E_c}$	0.0000001	0.0000001	0.00605	0.00759	7.798476

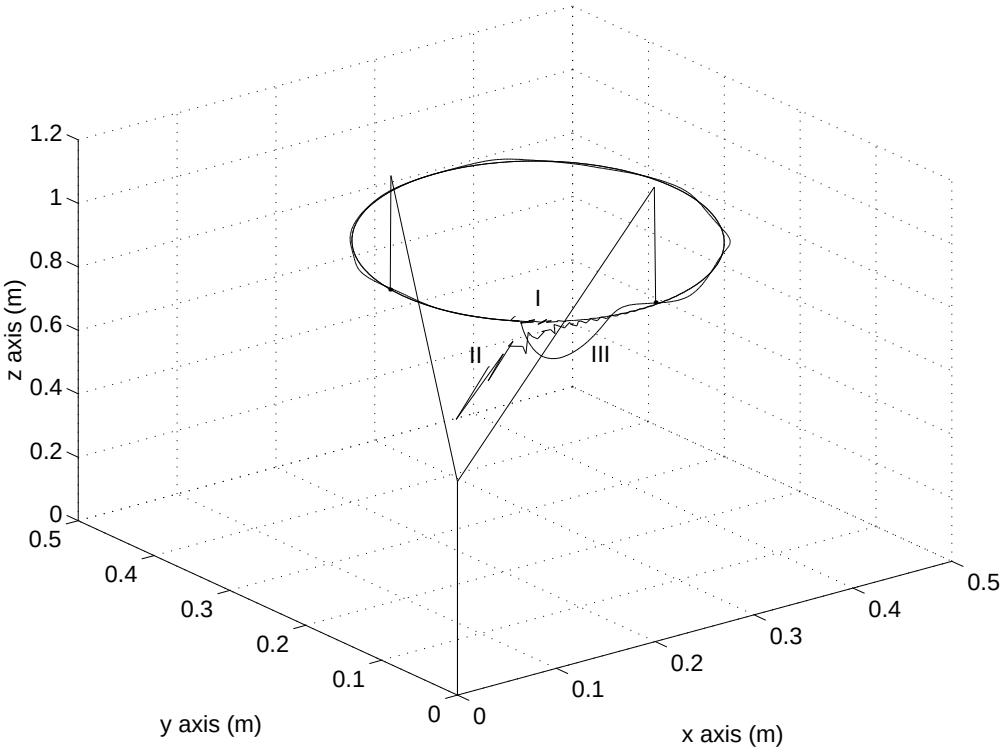


Figure 5.6: Cartesian Errors of End Point Circular Trajectory

Table 5.2: Composite tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized, $\boldsymbol{\alpha} = \mathbf{0.9}$ is used, $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{50\ 50\ 50}], \boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{625\ 625\ 625}]$.)

Schemes	***RCT I***	***RCT II***	***FEL I***	***FEL II***	***PD***
$\boldsymbol{\eta}$	0.0005	0.0001	0.0005	0.000004	N/A
$\boldsymbol{E_p}$	0.011361	0.018119	2.006162	1.963067	98.639343
$\boldsymbol{E_v}$	112.8616	100.6445	173.321945	168.1299	115.47089

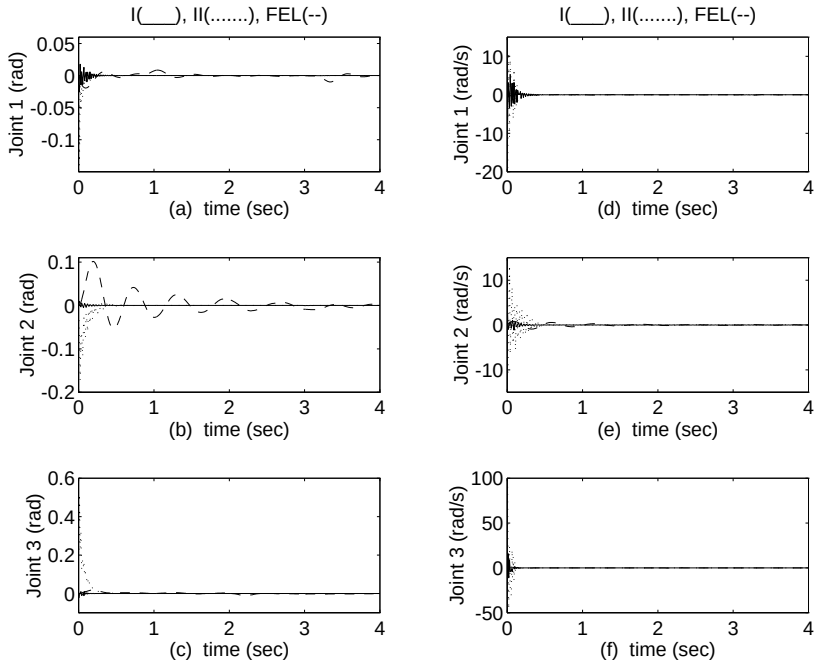


Figure 5.7: Joint Position Errors of Circular Trajectory Tracking

Table 3. Table 5.3 shows the excellent tracking performances of RCT Scheme I and II. Since the proposed RCT controllers allow the larger $\boldsymbol{\eta}$ as lowering feedback gains the tracking performances within stability the performances are even better than those under large primary controller gains(see Figure 5.10). Figure 5.10 (a) shows that the plot of the optimal update rates $\boldsymbol{\eta}$ versus PD gains and the corresponding Cartesian

Table 5.3: Tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized,$\boldsymbol{\alpha = 0.9}$ is used, $\boldsymbol{K_D = diag[10\ 10\ 10], K_P = diag[25\ 25\ 25]}$.)

Trajectory	***Circular***		***Composite***	
Schemes	***RCT I***	***RCT II***	***RCT I***	***RCT II***
$\boldsymbol{\eta}$	0.01	0.005	0.015	0.005
$\boldsymbol{E_p}$	0.000339	0.000229	0.009719	0.009503
$\boldsymbol{E_v}$	0.004881	0.003760	60.050266	37.12189
$\boldsymbol{E_c}$	0.000085	0.000061	N/A	N/A

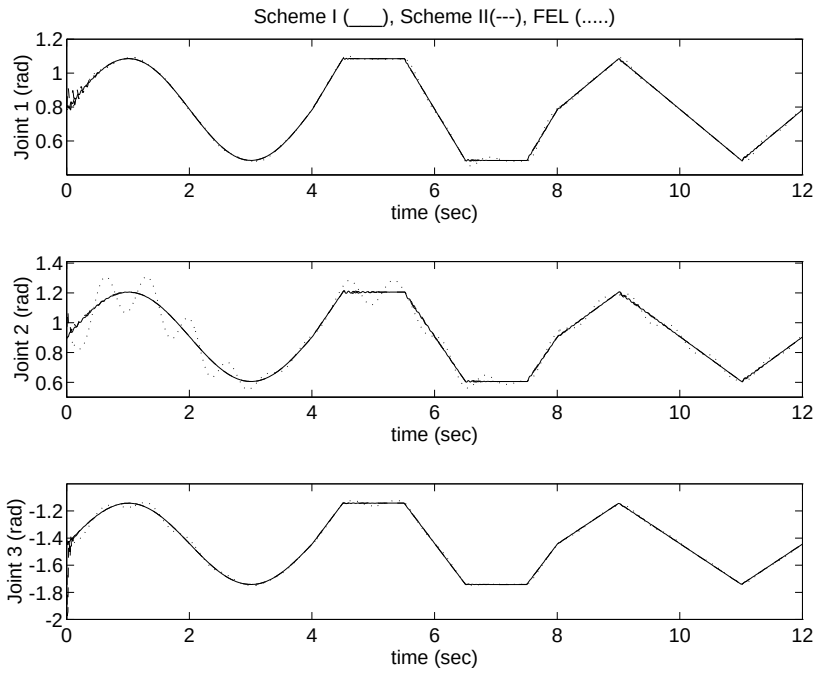


Figure 5.8: Composite Trajectory Tracking

errors versus PD gains are shown in Figure 5.10 (b). The numbers A on $\boldsymbol{X}$ axis represents PD gain factor of gain $\boldsymbol{K_P}$ and $\boldsymbol{K_D}$. The lowest PD gains are $\boldsymbol{K_P = 25I}$ and $\boldsymbol{K_D = 10I}$ when $\boldsymbol{A = 1}$. If A = 5 means 5 times of $\boldsymbol{K_D = 20I}$ and $\boldsymbol{K_P}$ can be calculated to yield critically damped behavior so that $\boldsymbol{K_P = 650I}$ and $\boldsymbol{K_D = 50I}$). The severe dependence on primary feedback gains for FEL schemes I,II can be clearly seen from Figure 5.10 (b) while the performance of the proposed RCT scheme I,II does not depend on the magnitude of feedback gains at all. This result is consistent with the structural analysis in the previous section. In addition to that the magnitude of the compensating signals from NN controller may be another reason since small changes in $\boldsymbol{\eta}$ do not have relatively large effects on large compensating output and

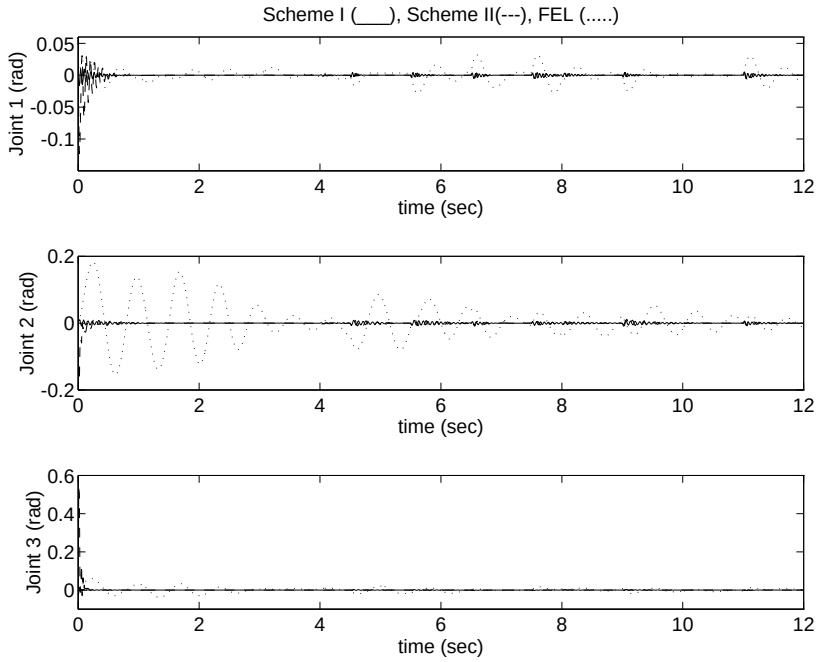


Figure 5.9: Joint Position Errors of Composite Trajectory Tracking

vise versa according to (5.8) and (5.13). Thus the robustness of the proposed RCT scheme I and II is demonstrated with a robot manipulator. It is interesting to note that the performance of FEL scheme II is as good as FEL scheme I, but worse than that of RCT I which is equivalent structure even though they are equivalent structure in linear-wise. This simply confirms the previous statement that the performance of NN that compensates for larger magnitudes of uncertainties is worse than that of NN for smaller magnitudes when the same percent accuracy of NNs are given. The magnitude of the same percent error in FEL NN is actually larger that that in RCT.

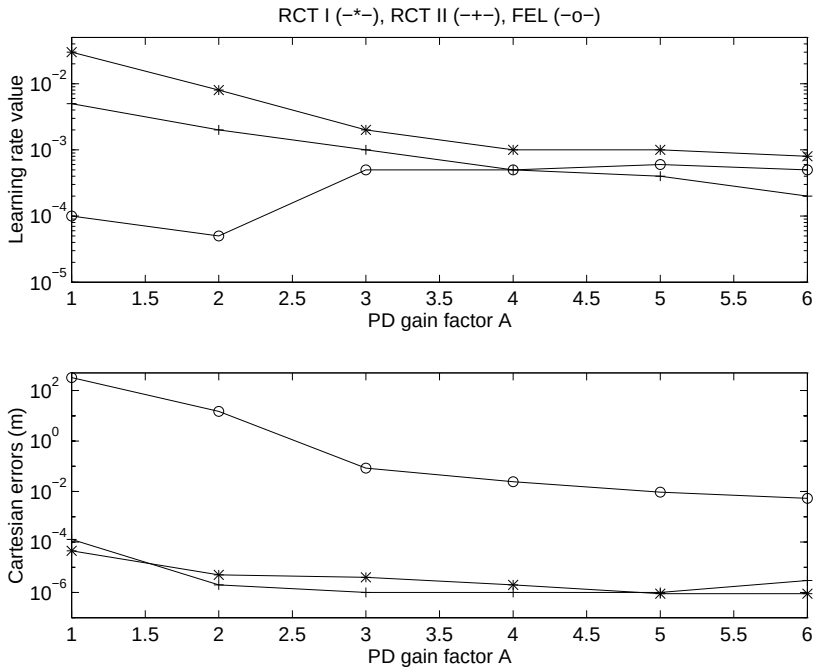


Figure 5.10: PD gains versus (a) Update Rate and (b) Cartesian Errors for Circular Trajectory The number on $\boldsymbol{X}$ axis means times of $\boldsymbol{K_P K_D}$ gains

5.4.1 Discussion

New inverse neural network control techniques for a dynamical robot system are presented along with the well known FEL scheme in this paper. The proposed RCT NN controllers identify the stabilized dynamical system by regulating the reference input command while the FEL NN controller identifies by generating the auxiliary compensating control input signal. Simulation studies on trajectory tracking of the robot manipulator showed that all of the NN schemes worked extremely well when compared with uncompensated PD control so that their controllers converged very fast with good accuracy. Among all NN controller schemes RCT Schemes are superior to FEL.

The best of all, the design of the proposed scheme has structural advantage over FEL scheme in that compensation is done outside control loop so that it can be implemented easily at the command trajectory planning level external to an existing controller without having to modify the controller's internal structure as would be required by FEL scheme. Another advantage is the robustness to fixed primary small controller gains since small gain control has much advantages in practical implementation. The proposed RCT Scheme 1 is robust to achieve good performances within stability with regardless of magnitudes of feedback primary gains while FEL controller is sensitive to feedback gains.

5.5 Prefilter Type NN Inverse Control Scheme

A new prefilter type robot control scheme is presented in here as depicted in Figure 5.11. The basic concept of this scheme is that the NN controller acts as an inverse of the robot under PD control by minimizing system output error vector so that the robot response $\boldsymbol{q}$ tracks $\boldsymbol{q_d}$ with minimal distortion. From Figure 5.11 the control input $\boldsymbol{\tau}$ is

$$\boldsymbol{\tau} = \boldsymbol{K_D}(\phi_d - \dot{\boldsymbol{q}}) + \boldsymbol{K_P}(\phi_p - \boldsymbol{q}) \tag{5.22}$$

where $\boldsymbol{\phi_d}, \boldsymbol{\phi_p}$ are output vectors of an NN controller. Compensation is done in input trajectory level and those compensating signals are amplified through gains so that the magnitudes of those prefiltered signals are small compared with one in the auxiliary

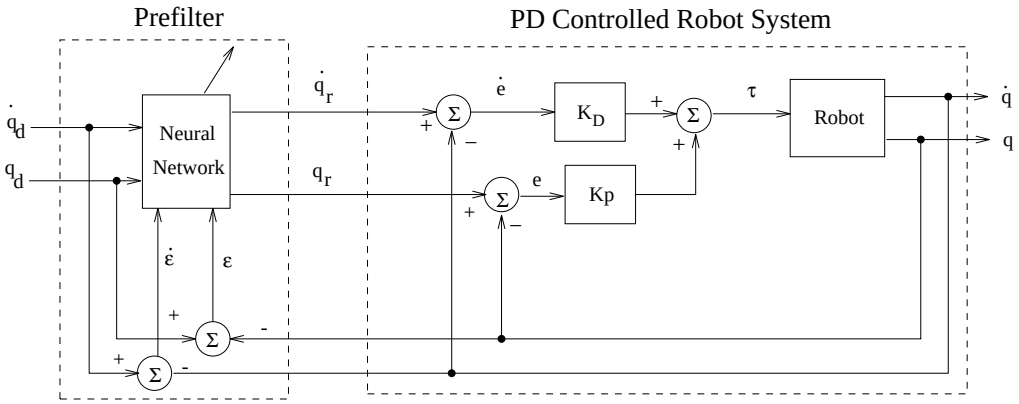


Figure 5.11: Scheme I : Prefilter Type NN Inverse Control Structure for PD Controlled Robot

type controller.

Combining (4.2) and (5.22) yields

$$\boldsymbol{K_D\dot{\varepsilon} + K_P\varepsilon = D\ddot{q} + h + \tau_f + K_D\dot{q}_d + K_P q_d - (K_D\phi_d + K_P\phi)} \tag{5.23}$$

Denoting $\boldsymbol{\Delta = D\ddot{q} + h + \tau_f}$ and $\boldsymbol{\Psi = K_D\phi_d + K_P\phi_p}$ yields the closed loop dynamic equations as follows:

$$\boldsymbol{K_D\dot{\varepsilon} + K_P\varepsilon = \Delta + K_D\dot{q}_d + K_P q_d - \Psi} \tag{5.24}$$

where $\boldsymbol{\dot{\varepsilon} = \dot{q}_d - \dot{q}}$, and $\boldsymbol{\varepsilon = q_d - q}$. At convergence, when inverse identification is done, the ideal output of NN is

$$\boldsymbol{\Psi = \Delta + K_D\dot{q}_d + K_P q_d} \tag{5.25}$$

which has the extra terms $\boldsymbol{K_D\dot{q}_d + K_P q_d}$ to be compensated comparing with previous schemes. This scheme identifies the inverse model of PD controlled robot manipulator rather than inverse model of robot manipulator. These extra terms can be eliminated

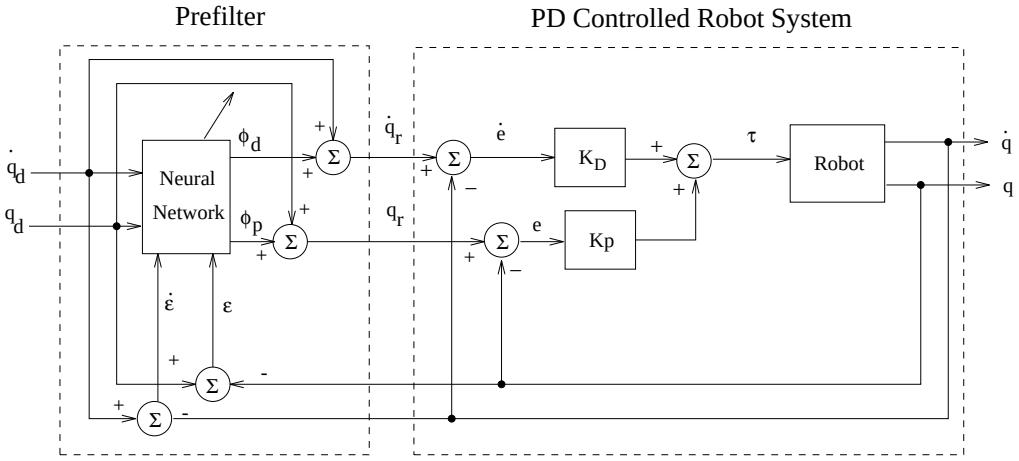


Figure 5.12: Scheme II : Modified Prefilter Type NN Inverse Control Structure

by modifying the NN prefilter structure as shown in Figure 5.12. Similarly, $\mathbf{\Psi}$ in modified NN prefilter scheme is reduced as

$$\mathbf{\Psi} = \mathbf{\Delta} \tag{5.26}$$

which is same as (5.12). Thus the modified scheme as depicted in Figure 5.12 has advantage of generating simpler compensating terms compared with (5.25), and this can be one of factors to help this modified scheme converge faster than the original scheme without loss of accuracy. Comparing with the previous auxiliary type NN controller the disadvantage of is that the Jacobian is required to use back propagation algorithm.

We use multilayer FFNN and BP updating algorithms. In view of system configuration of Figure 5.11, development of error back-propagation algorithm requires accurate knowledge of the plant in order to evaluate the gradient functions such as $\frac{\partial q}{\partial q_r}$. Clearly the plant dynamics is not precisely known in our case. One technique is

to use the ideal first order linear error equation.

$$\boldsymbol{K_D}(\dot{\boldsymbol{q}}_r - \dot{\boldsymbol{q}}) + \boldsymbol{K_P}(\boldsymbol{q}_r - \boldsymbol{q}) = \boldsymbol{0} \tag{5.27}$$

as an approximation to the plant dynamics by neglecting the plant uncertainties. The most significant consequence of this approximation is that the linear plant model (5.27) brings about a simple static Jacobian problem instead of a dynamic Jacobian problem when the plant is nonlinear [19].

The weight updating law minimizes the objective function $\mathcal{J}(\mathcal{E})$ for joint which is a function of joint tracking error:

$$\mathcal{J}(\mathcal{E}) = \frac{\mathbf{1}}{\mathbf{2}}(\mathcal{E}^T\mathcal{E}) \tag{5.28}$$

where $\mathcal{E} = [\varepsilon^T \dot{\varepsilon}^T]^T, \varepsilon = (\boldsymbol{q}_d - \boldsymbol{q}), \dot{\varepsilon} = (\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}})$. The gradient of $\mathcal{J}(\mathcal{E})$ is

$$\frac{\partial \mathcal{J}(\mathcal{E})}{\partial \boldsymbol{w}} = \frac{\partial \mathcal{E}^T}{\partial \boldsymbol{w}}\mathcal{E} \tag{5.29}$$

In view of plant model (5.27), we have

$$\frac{\partial \mathcal{J}(\mathcal{E})}{\partial \boldsymbol{w}} = -[\frac{\partial \boldsymbol{q}_r}{\partial \boldsymbol{w}} \; \frac{\partial \dot{\boldsymbol{q}}_r}{\partial \boldsymbol{w}}] \; \boldsymbol{J}_p^T \begin{bmatrix} \varepsilon \\ \\ \dot{\varepsilon} \end{bmatrix} \tag{5.30}$$

where

$$\boldsymbol{J_p} = \begin{bmatrix} \frac{\partial q}{\partial q_r} & \frac{\partial q}{\partial \dot{q}_r} \\ \\ \frac{\partial \dot{q}}{\partial q_r} & \frac{\partial \dot{q}}{\partial \dot{q}_r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{K_P^{-1}K_D} \\ \\ \boldsymbol{K_D^{-1}K_P} & \boldsymbol{I} \end{bmatrix} \tag{5.31}$$

$\boldsymbol{J_p}$ is a system Jacobian of the robot manipulator. The gradient rule for weight update is

$$\boldsymbol{\Delta w}(\boldsymbol{t}) \;\; = \;\; -\boldsymbol{\eta} \, \frac{\partial \mathcal{J}(\varepsilon)}{\partial \boldsymbol{w}} + \boldsymbol{\alpha \Delta w}(\boldsymbol{t-1}) \tag{5.32}$$

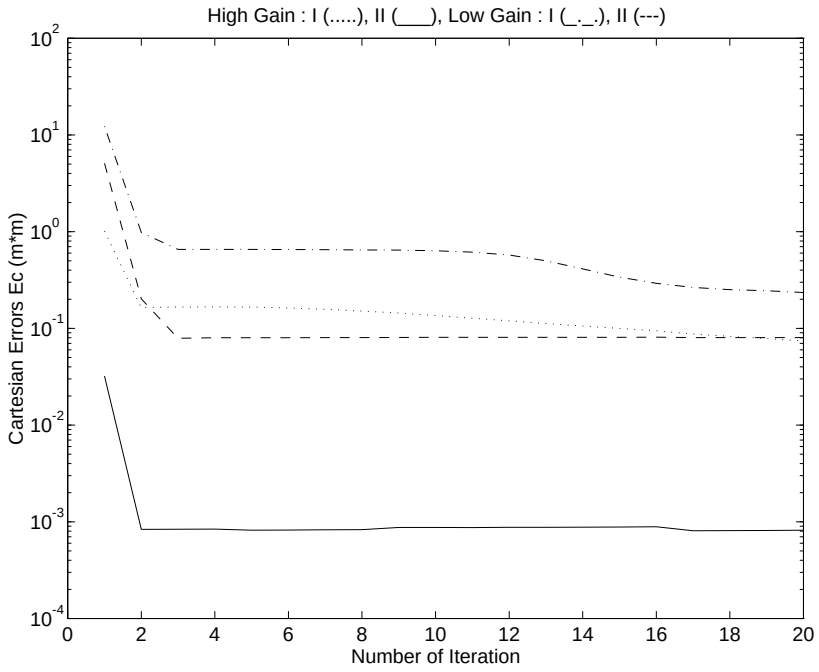


Figure 5.13: Cartesian Errors of Circular Trajectories for Prefilter type Scheme I and II

$$= [\frac{\partial \boldsymbol{q_r}}{\partial \boldsymbol{w}} \frac{\partial \dot{\boldsymbol{q}}_{\boldsymbol{r}}}{\partial \boldsymbol{w}}] \boldsymbol{J_p^T} \begin{bmatrix} \boldsymbol{\varepsilon} \\ \dot{\boldsymbol{\varepsilon}} \end{bmatrix} + \boldsymbol{\alpha \Delta w(t-1)} \tag{5.33}$$

(5.34)

where $\boldsymbol{\eta}$ is the update rate.

Figure 5.13 shows the circular trajectory tracking performances of prefilter type I and II under either high and low feedback gains. The modified prefilter controller (Scheme II) performs better with both high and low gains. The corresponding end point trajectory tracking plots are shown in Figure 5.14 and 15. The better performance is clearly visualized.

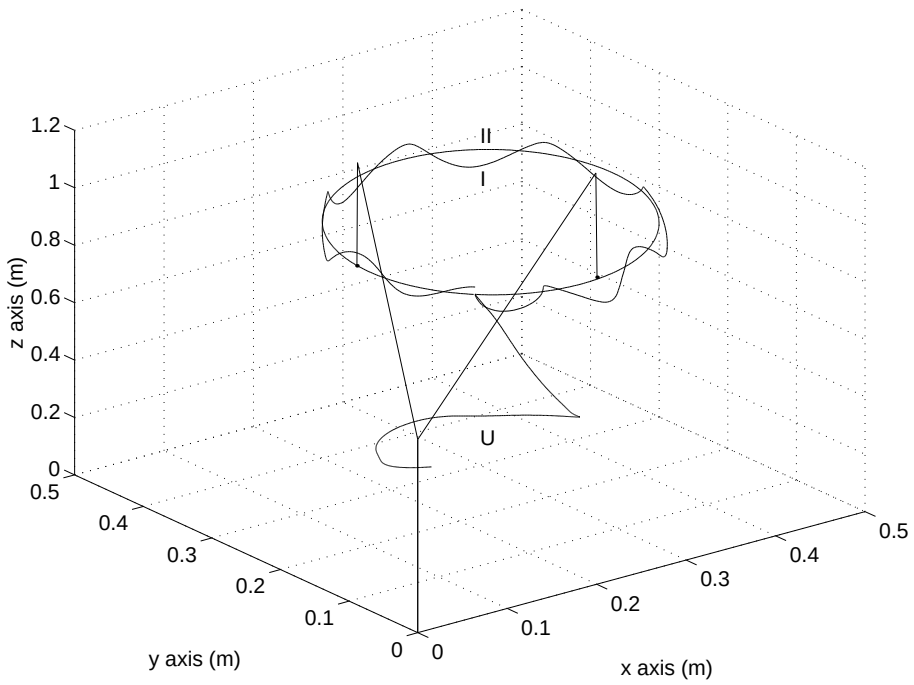


Figure 5.14: Circular Trajectory with low gains:Prefilter type Scheme I, II, and Uncompensated U

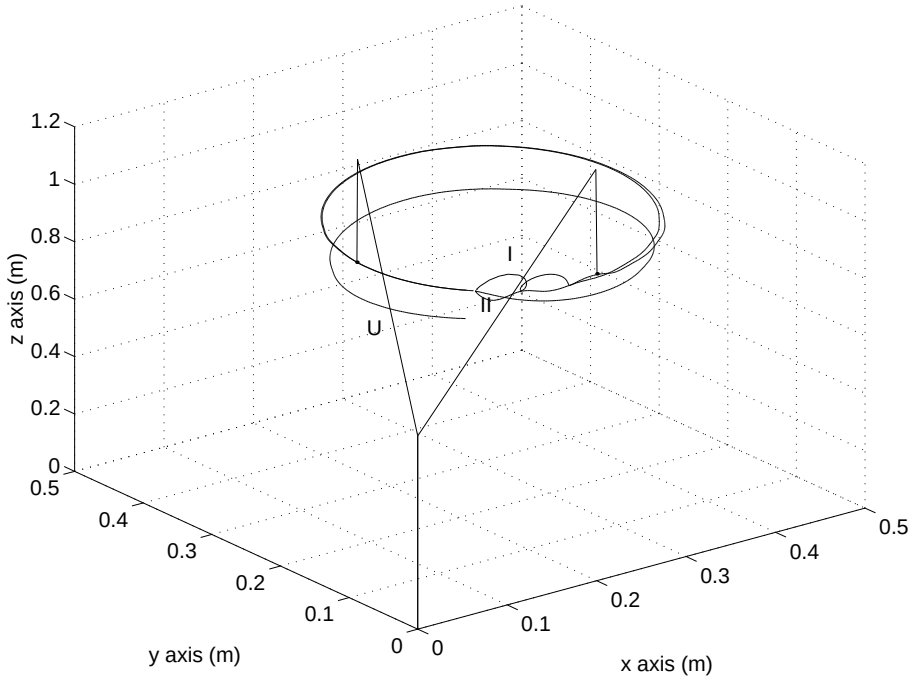


Figure 5.15: Circular Trajectory with high gains:Prefilter type Scheme I, II, and Uncompensated U

Table 5.4: RCT Schemes and Prefilter Type Schemes

	RCT	$Prefilter$
$Objective\ Function$	$\frac{1}{2}v^T v$	$\frac{1}{2}\varepsilon^T \varepsilon$
$Training\ Signal$	v	ε
$No\ of\ Weights$	$(3n+1)N_H + (N_H+1)n$	$(3n+1)N_H + (N_H+1)3n$
$Jacobian$	No	Yes
$Robustness$	Yes	No

Table 5.5: Circular tracking errors during the second iteration(η is optimized, $\alpha = 0.9$ is used, $K_D = diag[200\ 200\ 200], K_P = diag[500\ 500\ 500]$.)

Types	$Auxiliary$			$Prefilter$		$Uncomp$
$Scheme$	FEL	$RCTI$	$RCTII$	$Orig(I)$	$Mod(II)$	PD
η	0.005	0.000018	0.00001	0.01	0.009	N/A
E_p	0.0005	0.0004	0.0077	0.1219	0.0004	37.1022
E_v	0.1308	0.0035	0.0411	0.4512	0.2514	1.6018
E_c	0.0001	0.0001	0.0020	0.0249	0.0001	12.1274

5.6 Comparison Studies between RCT Scheme and Prefilter Scheme

Here is a summary of comparing structures of two schemes : RCT and Prefilter.

In performance comparison studies, different PD gains are used. Table 5.5 lists the results of circular trajectory tracking when PD gains are high. The performances of most NN controllers are comparable.

Also listed in Table 5.6 is that performances of NN controllers for composite trajectory in joint space under the same control environment. All NN schemes perform well

Table 5.6: Composite tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized, $\boldsymbol{\alpha} = \mathbf{0.9}$ is used, $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{200\ 200\ 200}], \boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{500\ 500\ 500}]$.)

NNi Types	***Auxiliary***			***Prefilter***		***Uncomp***
Schemes	***FEL***	***RCTI***	***RCTII***	***Orig(I)***	***Mod(II)***	***PD***
$\boldsymbol{\eta}$	0.001	0.000009	0.00001	0.01	0.005	N/A
$\boldsymbol{E_p}$	0.1080	0.0222	0.0619	0.7633	0.0779	158.8399
$\boldsymbol{E_v}$	18.7428	10.7714	12.1089	139.4903	38.6660	19.0308

Table 5.7: Tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized,$\boldsymbol{\alpha} = \mathbf{0.9}$ is used, $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{20\ 20\ 20}], \boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{50\ 50\ 50}]$.)

Traj	***Circular***					***Composite***	
Type	***Auxiliary***			***Prefilter***		***Auxiliary***	
Scheme	***RCTI***	***RCTII***	***FEL***	***Orig(I)***	***Mod(II)***	***RCTI***	***RCTII***
$\boldsymbol{\eta}$	0.005	0.005	0.001	0.003	0.001	0.005	0.005
$\boldsymbol{E_p}$	0.00018	0.00039	1.0286	10.8480	1.6354	0.0064	0.0064
$\boldsymbol{E_v}$	0.00313	0.00455	12.3819	196.3187	14. 2213	34.4641	35.5069
$\boldsymbol{E_c}$	0.00004	0.00010	0.24604	N/A	N/A	N/A	N/A

with slightly better performance of the proposed RCT Scheme I. In order to see the robustness of NN controllers to variation of feedback gains the same trajectory tracking tests are performed with relatively small magnitude of primary controller gains under the same control environment. For the FEL, the performances of composite trajectory tracking are so poor that the results are not listed in Table 5.7($\boldsymbol{e.g.E_P} > \mathbf{2000}$). Table 5.7 shows the excellent tracking performances of RCT Scheme I and II which belong to the auxiliary type NN. Since the proposed NN controllers allow the larger $\boldsymbol{\eta}$ as lowering feedback gains the tracking performances within stability are even better than those under large primary controller gains. Figure 5.16 (a) shows that the plot

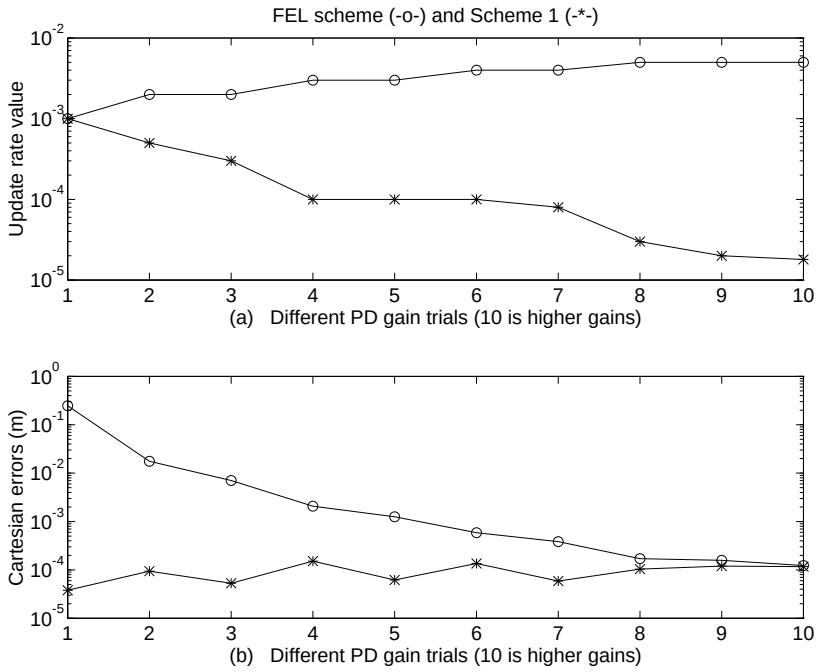


Figure 5.16: PD gains versus (a) Update Rate $\boldsymbol{\eta}$ and (b) Cartesian Errors $\boldsymbol{E_c}$ for Circular Trajectory: FEL I and RCT I

of the optimal update rates $\boldsymbol{\eta}$ versus PD gains and the corresponding Cartesian errors versus PD gains are shown in Figure 5.16 (b). The numbers 1-10 on $\boldsymbol{X}$ axis represents how many times of low gain $\boldsymbol{K_P}$ and $\boldsymbol{K_D}$ (e.g 5 means 5 times of $\boldsymbol{K_P = 50I}$ and $\boldsymbol{K_D = 20I}$ so that $\boldsymbol{K_P = 250I}$ and $\boldsymbol{K_D = 100I}$). The severe dependence on primary feedback gains for FEL scheme can be clearly seen from Figure 5.16 (b) while the performance of the proposed RCT scheme I does not depend on the magnitude of feedback gains at all. The reason is that the smaller primary controller gains are used the larger magnitudes should be compensated by NN because uncertainties are assumed to be bounded. For FEL case, however, the corresponding optimal $\boldsymbol{\eta}$ for each gains selected remains about same magnitude while $\boldsymbol{\eta}$ of the proposed scheme

increases as selected gains are smaller in order to achieve desired performances. In addition to that the magnitude of the compensating signals from NN controller may be another reason since small changes in $\boldsymbol{\eta}$ do not have relatively large effects on large compensating output and vise versa.

5.7 Neural Network as Computed Torque Element without Model

This scheme is a simplified version of computed torque control compensator structure. One neural compensator takes a role of two separate neural compensators. Therefore, this neural compensator is expected to require more hidden units or more time to converge. On-line control is not certain because the output signal from neural compensator is not expected to be bounded at the beginning of the process. The neural compensator can be trained to learn the structure of the robot manipulator off line, and used on line to learn uncertainties. One neural network should pay longer time to converge. The control law is simply represented as

$$\boldsymbol{\tau} = \boldsymbol{\tau_n} \tag{5.35}$$

where $\boldsymbol{\tau_n} = \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{w})$. The goal of neural network is to make $\boldsymbol{\tau_n}$ be

$$\boldsymbol{\tau_n} = \boldsymbol{W_d U} \tag{5.36}$$

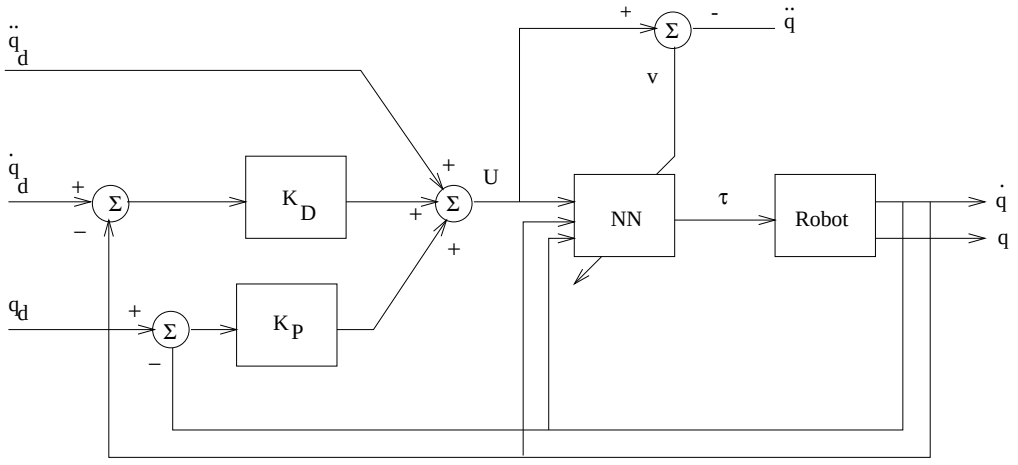


Figure 5.17: Neural Network as Computed Torque Element

where $\boldsymbol{W_d} \approx \boldsymbol{D(q)}$ at the convergence. $\boldsymbol{W_d}$ are the internal weights of the neural compensator. Then $\boldsymbol{v} = \boldsymbol{0}$ is achieved.

5.8 Discussion

New inverse neural network control techniques for non-model based robot manipulator control are presented along with the well known FEL scheme in this paper. Two basic types of schemes were presented : auxiliary and prefilter. These two types of schemes have the same goal of identifying inverse dynamics of robot manipulator by minimizing either system output error or system model error. The proposed auxiliary type NN controllers identify the robot system by regulating the desired joint trajectory while the FEL NN controller identifies by generating the auxiliary compensating torques to each joint. The prefilter type control scheme requires the estimation of Jacobian information in order to apply back propagation algorithm correctly.

Simulation studies on trajectory tracking showed that all of the NN schemes worked extremely well when compared with uncompensated PD control so that their controllers converged very fast with good accuracy. Among all NN controller schemes presented RCT Scheme 1 of the auxiliary type NN is the best choice for good performance.

The best of all, the design of the proposed schemes has structural advantage over FEL scheme in that compensation is done outside control loop so that it can be implemented easily at the command trajectory planning level external to an existing controller without having to modify the controller's internal structure as would be required by Kawato's FEL scheme. Another advantage is the robustness to fixed primary small controller gains since small gain control has much advantages in practical implementation. The RCT Scheme 1 is robust enough to achieve good performances within stability with regardless of magnitudes of feedback primary gains while FEL and prefilter type controller are sensitive to feedback gains.

Chapter 6

Cartesian Space NN Control

6.1 Introduction

It is clear that the higher the degree of nonlinearity exists in the uncertainties, the greater benefits NNs can contribute. In Cartesian space control, more robust control is required due to the following problems :

- The inverse dynamic in Cartesian space is more complicated than that in joint space.
- The singularity of Jacobian($\boldsymbol{J}$) becomes problem at hand when the positions in Cartesian space are calculated from joint measurements.
- More uncertainties are present.

Due to the fact that bounded nonlinear functions(such as Sigmoidal function) are used in the hidden neurons, it is also clear that NN performs better when the signal levels in the NN are small (say within $\pm\mathbf{1}$ or other normalized values) so that full exploitation of the nonlinear mapping capabilities can be achieved. Thus maximum payoff can be obtained from NNs when these two features are fully explored. In this chapter we examine these issues by studying the Cartesian space robot manipulator control problem. Unlike the joint space control problem shown in the previous chapters, model uncertainties in Cartesian space control are more nonlinear and complex. Although NN compensation for model uncertainties has been traditionally carried out by modifying the joint torque/force of the robot [43, 49], it is also possible to achieve the same objective by using the NN to modify the reference Cartesian trajectory as

we presented in the previous chapters(see Figure 6.2)[66]. We show that, for the same neural network, the required internal signal level for the trajectory modification approach is much smaller. Thus we are able to demonstrate via simulations that very significant performance improvement is obtainable by applying the NN compensation at the reference trajectory level instead of at the joint torque/force level. This approach is also very attractive in practice because it can be incorporated in the trajectory planner of any existing robot control system without having to alter the internal structure of the controller.

In the following, we will analyze the properties of these compensation approaches, discuss the NN architecture and training, and present the simulation results.

6.2 Robot Dynamic Equations in Cartesian Space

The kinematic relationships between the joint space coordinates $\boldsymbol{q}$ and the end effector Cartesian space coordinates $\boldsymbol{X}$ are

$$\dot{\boldsymbol{X}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}},\ \ddot{\boldsymbol{X}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} \tag{6.1}$$

where $\boldsymbol{J}(\boldsymbol{q})$ is the $\boldsymbol{n}$ x $\boldsymbol{n}$ nonsingular Jacobian matrix. Thus robot dynamic equation in joint space can be expressed as

$$\boldsymbol{D}(\boldsymbol{q})\boldsymbol{J}^{-1}(\boldsymbol{q})(\ddot{\boldsymbol{X}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\tau_f}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} \tag{6.2}$$

Since the end point forces $\boldsymbol{F}$ are related to the joint torques $\boldsymbol{\tau}$ by

$$\boldsymbol{\tau} = \boldsymbol{J}^{\boldsymbol{T}}\boldsymbol{F} \tag{6.3}$$

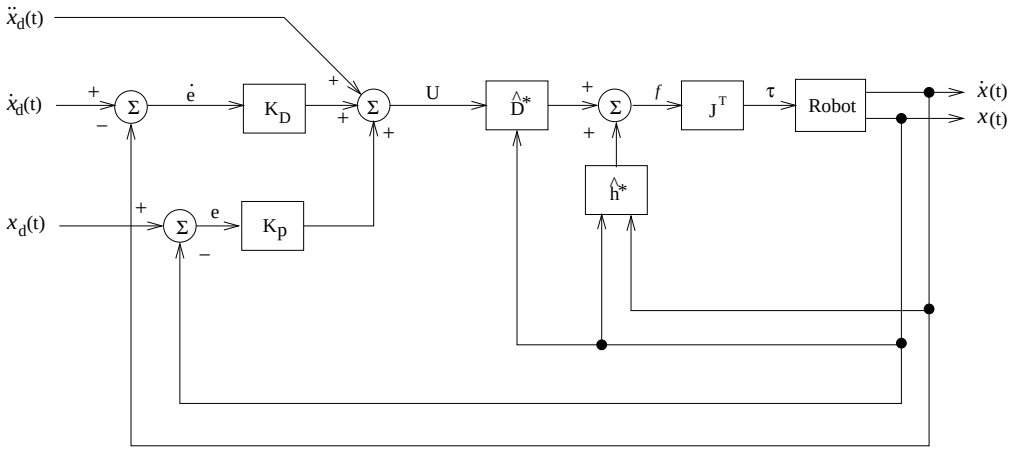


Figure 6.1: Position Control Structure in Cartesian Space

we obtain the Cartesian space robot dynamic equation

$$\boldsymbol{D^*(X)\ddot{X} + h^*(X,\dot{X}) + F_f^*(\dot{X}) = F} \tag{6.4}$$

where $\boldsymbol{D^* = J^{T^{-1}}(q)D(q)J^{-1}(q)}$, $\boldsymbol{h^* = J^{T^{-1}}h(q,\dot{q}) - D^*(X)\dot{J}J^{-1}\dot{X}}$ and $\boldsymbol{F_f^* = J^{T^{-1}}\tau_f(\dot{q})}$. The robot dynamic equation (6.4) represents a highly nonlinear, coupled, and multi-input multi-output system. In most practical cases, the model functions $\boldsymbol{D(q), h(q,\dot{q})}$, and $\boldsymbol{\tau_f(\dot{q})}$ are not exactly known, only the nominal estimates of $\boldsymbol{\hat{D}(q)}$ and $\boldsymbol{\hat{h}(q,\dot{q})}$ are available for controller design. For simplicity, $\boldsymbol{D^* = D^*(X), h^* = h^*(X,\dot{X}), F_f^* = F_f^*(\dot{X})}$ are used later on.

6.3 Computed Torque Control in Cartesian Space

The computed-torque control in Cartesian space is shown in Figure 6.1. The control law $\boldsymbol{F}$ in Cartesian space is

$$\boldsymbol{F} = \hat{\boldsymbol{D}}^* \boldsymbol{U} + \hat{\boldsymbol{h}}^* \tag{6.5}$$

And the control input $\boldsymbol{U}$ is given by

$$\boldsymbol{U} = \ddot{\boldsymbol{X}}_d + \boldsymbol{K_D}(\dot{\boldsymbol{X}}_d - \dot{\boldsymbol{X}}) + \boldsymbol{K_P}(\boldsymbol{X}_d - \boldsymbol{X}) \tag{6.6}$$

where $\boldsymbol{K_P}$ and $\boldsymbol{K_D}$ are $\boldsymbol{n}$ x $\boldsymbol{n}$ symmetric positive definite desired damping, and stiffness gain matrices, respectively and $\boldsymbol{x_d}$ is the desired joint trajectory. Combining (6.4),(6.5), and (6.6) yields the closed loop tracking error dynamic equation

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \hat{\boldsymbol{D}}^{*^{-1}}[\boldsymbol{\Delta D}^* \ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^* + \boldsymbol{F}_{\boldsymbol{f}}^*] \tag{6.7}$$

where $\boldsymbol{\Delta D}^* = \boldsymbol{D}^* - \hat{\boldsymbol{D}}^*, \boldsymbol{\Delta h}^* = \boldsymbol{h}^* - \hat{\boldsymbol{h}}^*$, and $\boldsymbol{E} = (\boldsymbol{X}_d - \boldsymbol{X})$. In the ideal case where $\boldsymbol{\Delta D}^* = \boldsymbol{\Delta h}^* = \boldsymbol{0}$, and $\boldsymbol{F}_{\boldsymbol{f}}^* = \boldsymbol{0}$, the closed loop robot behavior satisfies the second order differential equation

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \boldsymbol{0} \tag{6.8}$$

Since there are always uncertainties in the robot dynamic model, the ideal equation (6.8) can not be achieved in general. The actual system performance is governed by (6.7) which is degraded and unpredictable. Thus the computed torque based position control in Cartesian space is not robust in practice. To improve robustness,

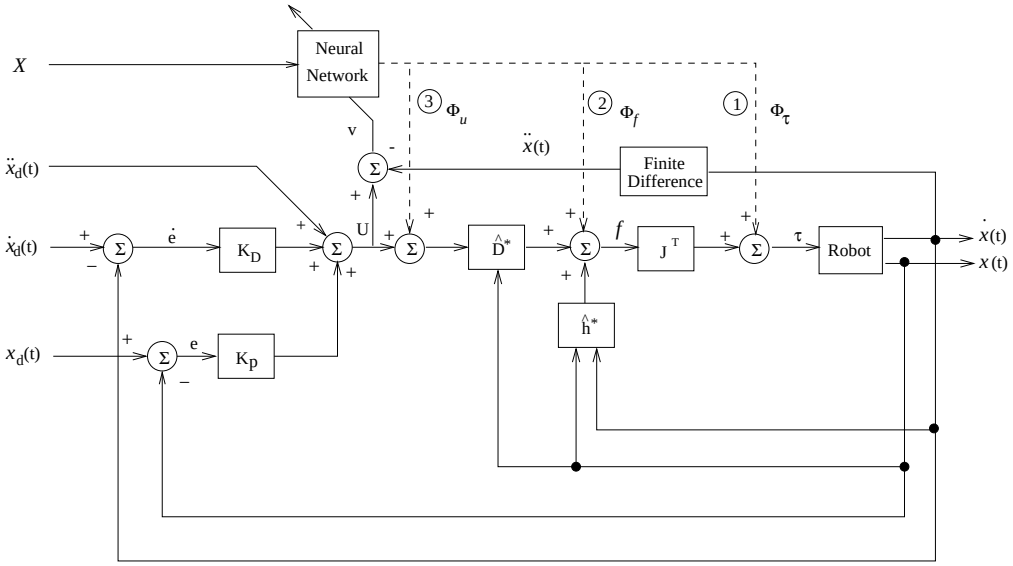


Figure 6.2: Proposed NN Control Structure in Cartesian Space with Different Locations of Compensation

NN controller can be introduced to generate additional input to compensate for the disturbances due to model uncertainties. These additional compensating inputs can be placed in four different locations as presented in the next section. The proposed control structure is shown in Figure 6.2.

6.4 Cartesian Space NN Controller Schemes

In this section, we present four different NN controller designs to achieve disturbance rejection for an uncertain system. The design perspectives are dependent upon the compensated position by NN as shown in Figure 6.2 and Figure 6.3. There are four locations that can be compensated : first, at torque level $\boldsymbol{\tau}$, second, at force level $\boldsymbol{F}$, third at control input level $\boldsymbol{U}$. Finally compensation at input tra-

jectory level $\boldsymbol{X_d}$ is depicted in Figure 6.3 separately. By defining a unified training signal all NN control schemes have one same goal of minimizing the same objective functions. The NN outputs $\boldsymbol{\Phi}$ of all the schemes try to cancel out the uncertainties caused by inaccurate robot model in the computed-torque controller. It is noted that the NN inputs $\boldsymbol{X}$ can be either $\boldsymbol{X_d(t), \dot{X}_d(t), \ddot{X}_d(t)}$ or the time-delayed values $\boldsymbol{X_d(t)\ X_d(t-1)\ X_d(t-2)}$. The latter case is called time-delayed NN in the literature [19]. Delay time is chosen as the sampling period of the controller. We found in our simulations that the NN performs slightly better or comparable when time-delayed joint values are used instead of the velocity and acceleration values. Here we lay out each scheme analytically.

Scheme 1 : Compensating at $\boldsymbol{\tau}$

The control law of compensating at torque level, called Scheme 1, is

$$\boldsymbol{\tau(t) = J^T[\hat{D}^* U + \hat{h}^*] + \Phi_\tau} \tag{6.9}$$

where $\boldsymbol{\Phi_\tau}$ is output of NN at torque level. Combining with the robot dynamic equation yields the corresponding closed loop error system as

$$\boldsymbol{v = \ddot{E} + K_D \dot{E} + K_P E = \hat{D}^{*^{-1}}(\Delta D^* \ddot{X} + \Delta h^* + F_f^* - J^{T^{-1}} \Phi_\tau)} \tag{6.10}$$

Since the control objective is to generate $\boldsymbol{\Phi}$ to reduce $\boldsymbol{v}$ to zero in (6.10), we therefore propose here to use $\boldsymbol{v}$

$$\boldsymbol{v = \ddot{E} + K_D \dot{E} + K_P E} \tag{6.11}$$

as the new error signal for training the NN in all schemes. The ideal value of $\boldsymbol{\Phi_\tau}$ at

$\boldsymbol{v} = \mathbf{0}$ then is

$$\boldsymbol{\Phi}_{\tau} = \boldsymbol{J}^{\boldsymbol{T}}(\boldsymbol{\Delta D}^{*}\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^{*} + \boldsymbol{F}_{\boldsymbol{f}}^{*}) \tag{6.12}$$

Thus NN is required to learn robot Jacobian and this degrades the performance comparing with other schemes. Clearly minimizing the error signal $\boldsymbol{v}$ allows us to achieve ideal position control directly. However, the use of error signal in Cartesian space as suggested in [43] requires on-line computation of the nominal robot model $\boldsymbol{F_m} = \boldsymbol{D}^{*}\ddot{\boldsymbol{X}} + \boldsymbol{h}^{*}$ in Cartesian space which is far more time consuming than computing $\boldsymbol{v}$ in (6.11) as proposed.

Scheme 2 : Compensating at $\boldsymbol{F}$

In order to avoid learning kinematic Jacobian compensating location can be moved from 1 torque level to 2 force level in Figure 6.2. Similarly, the control law from Figure 6.2 is derived as

$$\boldsymbol{\tau}(\boldsymbol{t}) = \boldsymbol{J}^{\boldsymbol{T}}[\hat{\boldsymbol{D}}^{*}\boldsymbol{U} + \hat{\boldsymbol{h}}^{*} + \boldsymbol{\Phi}_{\boldsymbol{f}}] \tag{6.13}$$

where $\boldsymbol{\Phi}_{\boldsymbol{f}}$ is output of NN at force level. Substituting (6.13) into (6.4) produces

$$\boldsymbol{v} = \hat{\boldsymbol{D}}^{*^{-1}}(\boldsymbol{\Delta D}^{*}\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^{*} + \boldsymbol{F}_{\boldsymbol{f}}^{*} - \boldsymbol{\Phi}_{\boldsymbol{f}}) \tag{6.14}$$

At convergence, the ideal value of $\boldsymbol{\Phi}_{\boldsymbol{f}}$ is

$$\boldsymbol{\Phi}_{\boldsymbol{f}} = \boldsymbol{\Delta D}^{*}\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^{*} + \boldsymbol{F}_{\boldsymbol{f}}^{*} \tag{6.15}$$

which is a simplification of (6.12). Thus NN does not need to learn robot Jacobian.

Scheme 3 : Compensating at $\boldsymbol{U}$

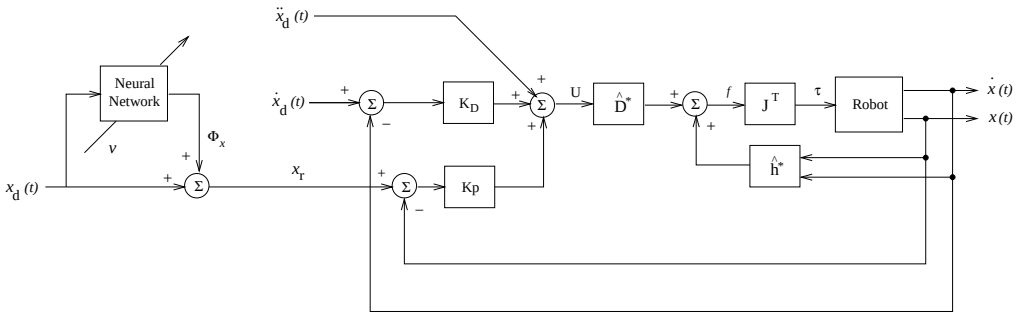


Figure 6.3: Proposed NN Control Structure that Compensates at Input Trajectory Level

Scheme 3 is to compensate at control input $\boldsymbol{U}$. Accordingly, the control law is given by

$$\boldsymbol{\tau(t) = J^T[\hat{D}^*(U + \Phi_u) + \hat{h}^*]} \tag{6.16}$$

where $\boldsymbol{\Phi_u}$ is output of NN at control input level. Substituting (6.16) into (6.4) produces

$$\boldsymbol{v = \hat{D}^{*^{-1}}(\Delta D^* \ddot{X} + \Delta h^* + F_f^*) - \Phi_u} \tag{6.17}$$

The ideal NN output is

$$\boldsymbol{\Phi_u = \hat{D}^{*^{-1}}(\Delta D^* \ddot{X} + \Delta h^* + F_f^*)} \tag{6.18}$$

The compensating magnitude is further reduced from (6.15) by $\boldsymbol{\hat{D}^{*^{-1}}}$ since $\| \boldsymbol{\hat{D}^*} \|$ is positive definite and usually greater than 1. At least it is true for the Puma 560 robot manipulator that $\| \boldsymbol{\hat{D}^*} \| > 1$.

Scheme 4 : Compensating at $\boldsymbol{X_d}$

Finally, for compensating input trajectory, the control law becomes

$$\boldsymbol{\tau(t) = J^T[D^*(\ddot{X} + K_D(\dot{X}_d - \dot{X}) + K_P(X_d - X + \Phi_x)) + \hat{h}^*]} \tag{6.19}$$

where $\boldsymbol{\Phi_x}$ is output of NN at input trajectory level. Combining with (4.2) and (6.4) yields

$$\boldsymbol{v} = \hat{\boldsymbol{D}}^{*^{-1}}(\boldsymbol{\Delta D}^*\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^* + \boldsymbol{F}_f^*) - \boldsymbol{K_P}\boldsymbol{\Phi_X} \tag{6.20}$$

At $\boldsymbol{v} = \boldsymbol{0}$, the ideal output of NN is

$$\boldsymbol{\Phi_x} = \boldsymbol{K_P^{-1}}\hat{\boldsymbol{D}}^{*-1}(\boldsymbol{\Delta D}^*\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^* + \boldsymbol{F}_f^*) \tag{6.21}$$

where $\boldsymbol{\Phi_x}$ has the smallest magnitude among all schemes presented above. The performance is also dependent on feedback gain $\boldsymbol{K_P}$. Ideally, it is true that $\boldsymbol{\Phi_u} = \boldsymbol{K_P} \cdot \boldsymbol{\Phi_x}$. Thus the compensating magnitudes of NN output are different from Scheme 1 to Scheme 4 under the same NN structure with Scheme 4 having the smallest. This simply means that Scheme 4 is the best scheme in terms of performance since NN prefer to operate in the same region that nonlinear function employed in NN operates in terms of numerical processing. Then the small magnitude of NN output is enough to cancel out uncertainties to satisfy the desired performance. In a view of practical implementation point Scheme 4 has also a great advantage over other schemes because implementation can be done easily not interrupting internal primary controller. In simulation section results are presented to prove these analytical explanations.

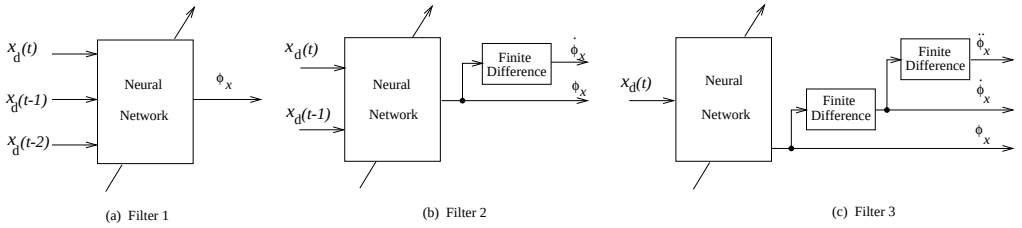


Figure 6.4: Various Design of NN Controller

6.5 Neural Network Compensator Design

The weight updating law minimizes the objective function $\mathcal{J}$ which is a quadratic function of the training signal $\boldsymbol{v}$.

$$\mathcal{J} = \frac{1}{2}\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v} \tag{6.22}$$

Differentiating equation(6.22) and making use of (6.12) yields the gradient of $\mathcal{J}$ for Scheme 1 as follows:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{v} = -\frac{\partial \boldsymbol{\Phi}_{\boldsymbol{\tau}}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{J}^{\boldsymbol{T}^{-\boldsymbol{T}}}\hat{\boldsymbol{D}}^{*^{-\boldsymbol{T}}}\boldsymbol{v} \tag{6.23}$$

in which the fact $\frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{\Phi}_{\boldsymbol{\tau}}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{J}^{\boldsymbol{T}^{-\boldsymbol{T}}}\hat{\boldsymbol{D}}^{*^{-\boldsymbol{T}}}$ is used where $\boldsymbol{A}^{-\boldsymbol{T}} = [\boldsymbol{A}^{-\mathbf{1}}]^{\boldsymbol{T}}$. For other schemes, $\frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{\Phi}_{\boldsymbol{f}}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\hat{\boldsymbol{D}}^{*^{-\boldsymbol{T}}}$ for Scheme 2, $\frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{\Phi}_{\boldsymbol{u}}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}$ for Scheme 3,$\frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}} = -\frac{\partial \boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{K}_{\boldsymbol{P}}^{\boldsymbol{T}}$ for Scheme 4 can be found from previous section. Thus the learning algorithm based on gradient shows a decreasing order of complexity from Scheme 1 to Scheme 3 or 4. The back-propagation updating rule for the weights with a momentum term is

$$\boldsymbol{\Delta w(t)} = \boldsymbol{\eta}\frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{v} + \boldsymbol{\alpha}\boldsymbol{\Delta w(t-1)} \tag{6.24}$$

where $\boldsymbol{\eta}$ is the update rate and $\boldsymbol{\alpha}$ is the momentum coefficient.

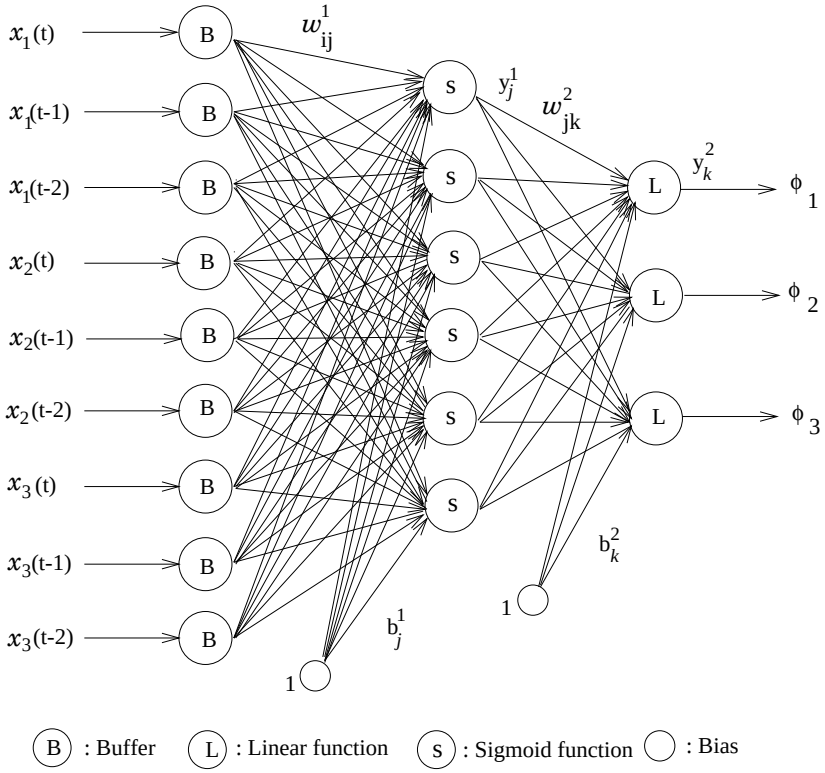


Figure 6.5: NN Structure

6.6 NN Controller Performance

For the NN controller, we have chosen six hidden neurons $(n_H = 6)$. The back propagation algorithm parameters are: $w_{ij}^k(0) = 0$ and $\alpha = 0.9$. The controller gains are selected as $K_D = diag[20, 20, 20]$ and $K_P = diag[100, 100, 100]$ which give identical critically damped motions at the three axis. The performances of the proposed schemes are tested by tracking circular trajectory which has radius of 30 cm and is tilted by 45 degrees. The tracking error is defined as the difference between the desired response x_d and actual response x. System performance is measured by the following saves of tracking error-squares over one training cycle of a trajectory as follows:

$$E_p = \sum_{t=1}^{P/T} \| X_d - X \|^2 \ (rad)^2 \qquad E_v = \sum_{t=1}^{P/T} \| \dot{X}_d - \dot{X} \|^2 \ (rad/sec)^2 \tag{6.25}$$

where E_p is the position error; E_v is the velocity error; P is one training cycle elapse time, where $P = 4$ seconds, and T is the sampling period of $T = 5ms$. Table 6.1 summarizes the converged performance results of proposed schemes. The corresponding Cartesian error plots are shown in Figure 6.6. The performance depending on compensating location is clearly demonstrated. The simulation data of Scheme 4 in Figure 6.6 showed that the rate of convergence is very fast with a convergence time less than a half second. Thus on-line trajectory control of the proposed NN scheme is completely acceptable in practice. Also plotted in Figure 6.7 are outputs of NN for

all the schemes. The compensating magnitudes prove the previous analysis with the NN output of Scheme 4 having the smallest magnitude. The separate plots of NN

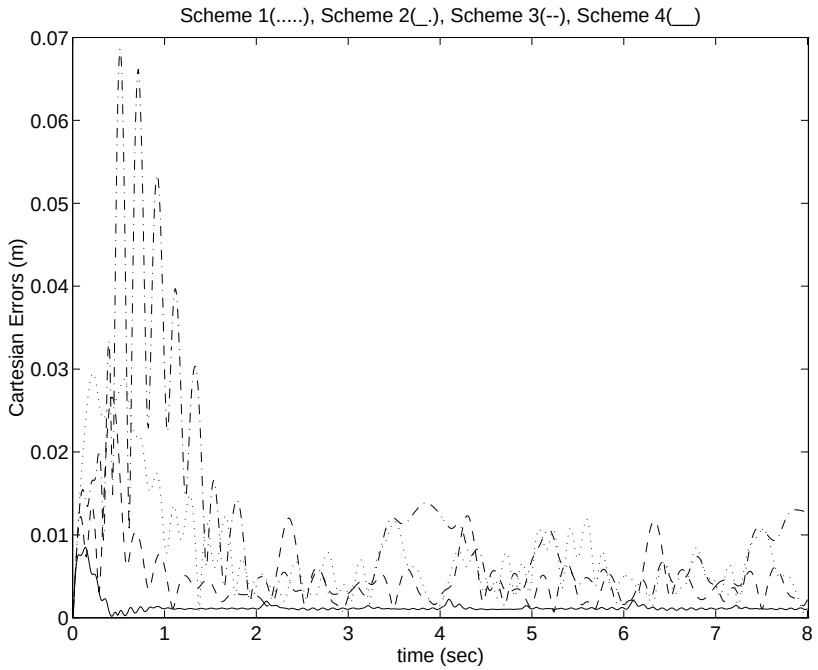


Figure 6.6: Cartesian Errors calculated as $\sqrt{(x_d - x)^2 + (y_d - y)^2 + (z_d - z)^2}$

output of Scheme 4 are shown in Figure 6.7(d,e,f) because the magnitudes are too small compared with other schemes. The chattering behavior occurred to compensate for friction disturbances accurately. Also plotted in Figure 6.8 (a,b,c) is NN outputs for Scheme 3 and 4 together. The magnitude differs by approximate $\boldsymbol{J^T}$. The plots shown in Figure 6.8 (d,e,f) represents the relationship of $\boldsymbol{\Phi_u \cong K_P \cdot \Phi_x}$ between Scheme 3 and 4. The compensating magnitudes of both schemes after multiplying $\boldsymbol{K_P}$ to $\boldsymbol{\Phi_x}$ are close to each other after NN compensator converges. However, Scheme 4 compensates more accurately. From these plots we can conclude the important fact

that NN compensator prefer the same operating range in terms of numerical processing which falls into the dynamic range of nonlinear function employed in NN. Note that the compensating magnitude range of Scheme 3 are $\boldsymbol{K_P}$ times of that of Scheme 4.

As a base line comparison, we also listed the performance of the uncompensated computed-torque control system in Cartesian space in Table 6.1. It is seen that the tracking errors are very large.

Table 6.1: Circular Trajectory Tracking errors after convergence ($\boldsymbol{\eta}$ is optimized. $\boldsymbol{\alpha = 0.9, n_H = 6, w_{ij}^k(0) = 0}$)

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Uncomp
$\boldsymbol{\eta}$	0.01	0.05	0.01	0.00001	N/A
$\boldsymbol{E_p}$	0.0317	0.0411	0.0197	0.0011	7.9104
$\boldsymbol{E_v}$	4.4693	1.9081	1.6838	0.0717	4.8632
[a] $\boldsymbol{\Phi}$	-15< $\boldsymbol{\Phi}$ <41	-15< $\boldsymbol{\Phi}$ <43	-3< $\boldsymbol{\Phi}$¡15	-0.03< $\boldsymbol{\Phi}$ <0.15	N/A

[a] Bound of NN output after convergence.

For non-model based approach, Schemes 1 and 2 are automatically reduced to feedback error learning scheme in Cartesian space. The results are listed in Table 6.3.

Table 6.2: Model Based Approach for different filters of Scheme 3 ($\boldsymbol{\eta}$ is optimized, $\boldsymbol{\alpha = 0.9}$ is used)

$\boldsymbol{Schemes}$	$\boldsymbol{Filter1}$	$\boldsymbol{Filter2}$	$\boldsymbol{Filter3}$
$\boldsymbol{\eta}$	0.00001	0.0000015	0.0000001
$\boldsymbol{E_p}$	0.0011	0.0016	0.1927
$\boldsymbol{E_v}$	0.0721	0.0186	0.9922
$\boldsymbol{\Phi}$	$\boldsymbol{-0.025 < \Phi_x < 0.144}$	$\boldsymbol{-0.019 < \Phi_x < 0.14}$	$\boldsymbol{-0.019 < \Phi_x < 0.14}$

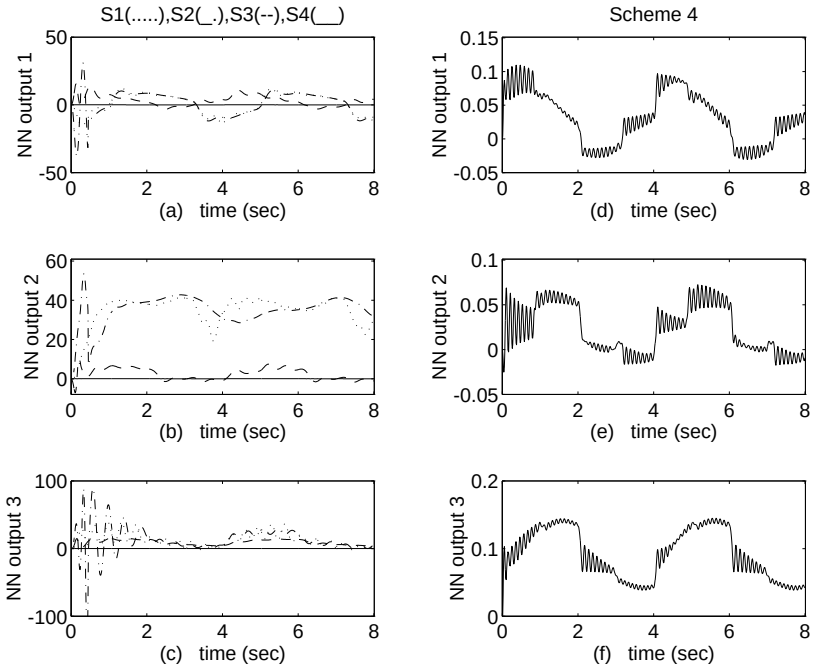


Figure 6.7: NN Outputs : (a)(b)(c) Scheme 1,2,3, and 4 (d)(e)(f) Scheme 4 only

The NN controller in non-model based approach performs the best tracking when filter 2 is used, and is comparable to that of model based approach. The magnitude of compensating signal is the smallest when filter2 is used.

Figure 6.9 shows the end point trajectories of the circular command signal for the uncompensated controller and the Scheme 4 NN controller. The improvement achieved by the NN controller is clearly demonstrated. The NN compensator serves as an auxiliary control for the computed-torque controller in Cartesian space to counteract the uncertainties in the robot dynamics and disturbances occuring during application. A new teaching signal $\boldsymbol{v}$ is developed for the proper back-propagation

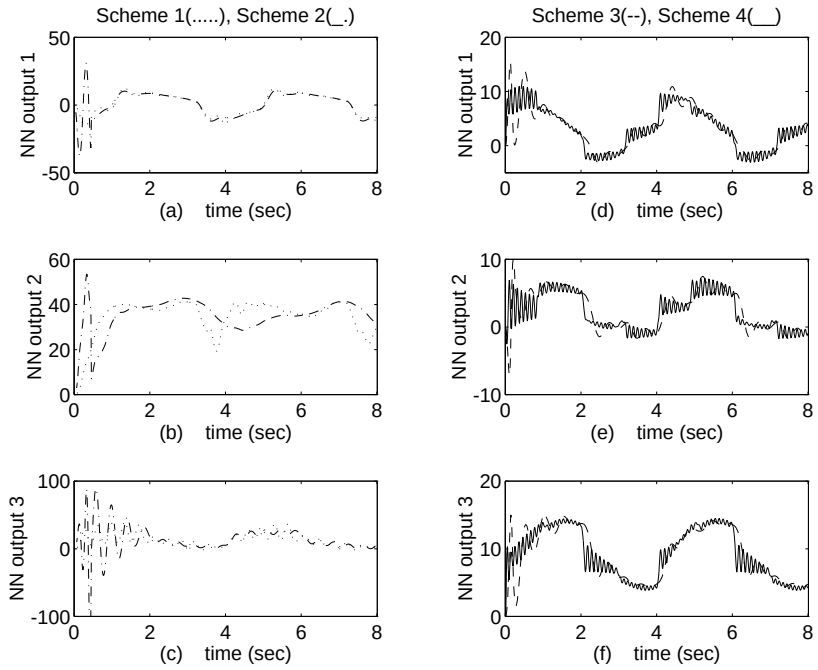


Figure 6.8: NN Outputs : (a)(b)(c) Scheme 1 and 2, (d)(e)(f) Scheme 3 and 4 with NN output of Scheme 4 being multiplied by $\boldsymbol{K_P}$

Table 6.3: Non-Model Based Approach for Circular Tracking errors during the second iteration($\boldsymbol{\eta}$ is optimized, $\boldsymbol{\alpha = 0.9}$ is used)

$\boldsymbol{Schemes}$	$\boldsymbol{Filter1}$	$\boldsymbol{Filter2}$	$\boldsymbol{FEL}$	$\boldsymbol{Uncompensated}$
$\boldsymbol{\eta}$	0.0000008	0.0003	0.0008	N/A
$\boldsymbol{E_p}$	2.1729	0.0011	4.2179	814.8922
$\boldsymbol{E_v}$	25.4444	0.0037	25.8204	42.1146
$\boldsymbol{\Phi}$	$\boldsymbol{0 < \Phi < 2.7}$	$\boldsymbol{-1 < \Phi < 2.2}$	$\boldsymbol{-1 < \Phi < 263}$	N/A

algorithm application, and delayed sample variables are proposed as inputs to the NN. In addition several NN controller designs based on their compensating locations have been investigated in a analysis and proved by simulations on a three-link puma 560 like robot arm. The system performances under this scheme are excellent and the convergence rates are under a half second. Thus the proposed NN controller is feasible for on-line robot control. We also experimentally found that NN prefers small magnitude which falls in dynamic range of sigmoidal function employed in NN controller. Thus, judging from the training and tracking performances of trajectory, it is reasonable to conclude that a good and practical choice of the NN controller design is Scheme 4 which compensates at input trajectory level in terms of performance and simpler structure.

6.7 Extension to Non-Model Based

Here we extend the same technique to non-model based robot control. Schemes 1 and 2 from the previous section can be automatically reduced to non-model based PD control structure as shown in Figure 6.10 (a). This is called Cartesian feedback error learning(FEL) NN control scheme. In same manner, Scheme 3 becomes the reduced structure of Figure 3 shown in Figure 6.10 (b). The purpose of these non-model based Cartesian NN controllers is to identify the inverse dynamics in Cartesian space. The same analysis as the model based approach is carried out by defining the

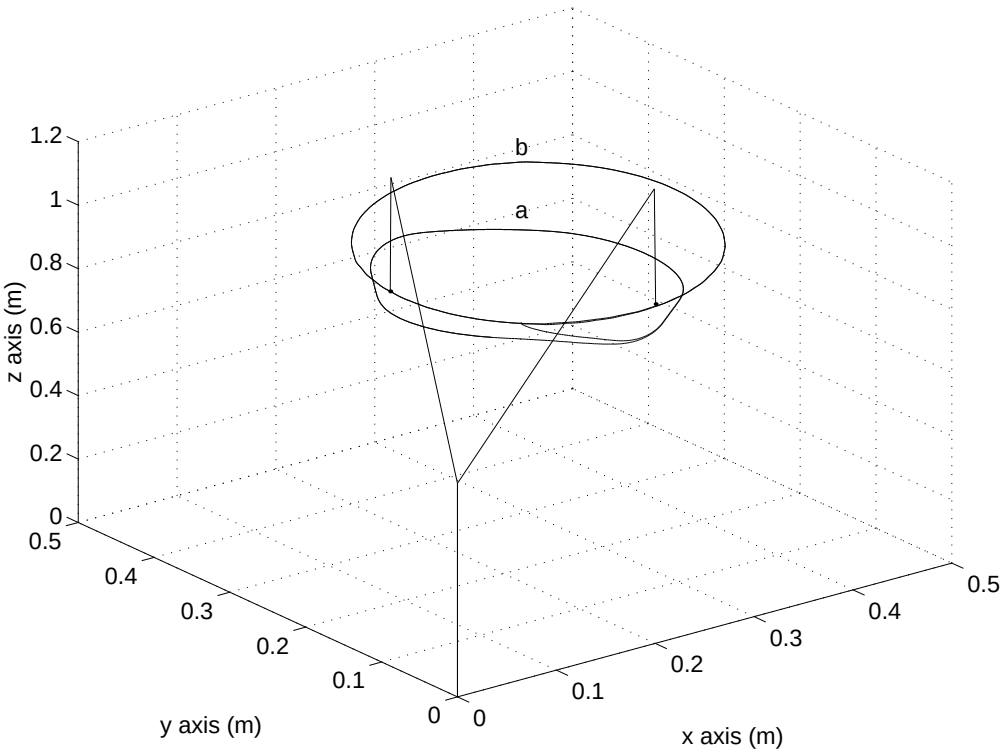


Figure 6.9: End point Circular Trajectory : (a) Nominal Control Scheme in Figure 6.1 (b) Proposed Scheme in Figure 6.2

control law equations in Cartesian space as follows for FEL scheme :

$$\boldsymbol{F} = \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon} + \boldsymbol{\Phi_f} \tag{6.26}$$

where $\boldsymbol{\varepsilon} = \boldsymbol{x_d} - \boldsymbol{x}$. Combining (6.4) and (6.26) yields

$$\boldsymbol{v'} = \boldsymbol{K_D}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{K_P}\boldsymbol{\varepsilon} = \boldsymbol{D^*}\ddot{\boldsymbol{X}} + \boldsymbol{h^*} + \boldsymbol{F_f^*} - \boldsymbol{\Phi_f} \tag{6.27}$$

where $\boldsymbol{v'}$ is a newly defined unified training signal for non-model based control in Cartesian space.

Thus, at convergence, $\boldsymbol{\Phi_f} = \boldsymbol{D^*}\ddot{\boldsymbol{X}} + \boldsymbol{h^*} + \boldsymbol{F_f^*}$ which is the dynamic model of robot manipulator. Similarly, for Scheme 3 in non-model based, two filter structures

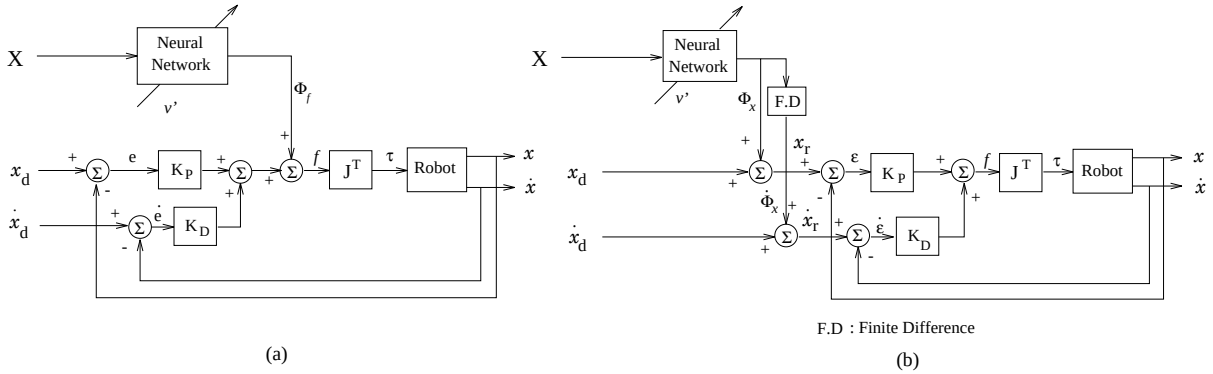


Figure 6.10: Cartesian NN Controllers : (a) FCC Scheme (b) TMC Scheme: Training error is $\boldsymbol{v' = K_D\dot{E} + K_P E}$

are used:filter 1 and 2. And the input to the NN is reduced to $[\boldsymbol{X_d^T(t)X_d^T(t-1)}]^{\boldsymbol{T}}$. The closed loop equation is

$$\boldsymbol{K_D\dot{E} + K_P E = D^*\ddot{X} + h^* + F_f^*} \tag{6.28}$$

Substituting $\boldsymbol{\dot{E} = \dot{\Phi}_x + \dot{X}_d, E = \Phi_x + X_d}$ for filter 2 into (6.28) yields the following closed loop equation

$$\boldsymbol{v' = D^*\ddot{X} + h^* + F_f^* - \Phi_p} \tag{6.29}$$

where $\boldsymbol{\Phi_p = K_D\dot{\Phi}_x + K_P\Phi_x}$.

At convergence, the relationship $\boldsymbol{\Phi_p = K_D\dot{\Phi}_x + K_P\Phi_x = \Phi_f}$ can be found. Thus it is clear to see that $\| \boldsymbol{\Phi_x} \| << \| \boldsymbol{\Phi_f} \|$ at convergence.

6.8 Discussion

In this chapter, the auxiliary control of compensating for uncertainties in Cartesian space has been introduced. It is found that different compensation locations have

different effects on performances. The Cartesian reference compensation technique shows the best performance as expected. We also investigated the performances of different types of filters in RCT scheme. Similar idea has been extended to non-model based case as well.

Successful neural network application to Cartesian space position control opens the door of possible neural network application to force control which is formed in the Cartesian space. It will be presented in the next chapter.

Chapter 7

Force Control

7.1 Introduction

In the previous chapters, we have studied several neural network control schemes for robot motion control. However, the present tasks in industries are more complicated and are required to cope with unknown contact with environment. Examples of such tasks are assembling parts into products, turning a crank, driving a screw, positioning meta surfaces, opening a door, deburring, drilling a hole in a bone, and so on. Here the same idea of using NN applied in position control is introduced to force control of robot manipulators.

When the robot motion is constrained by the given task and requires that the trajectory be modified continuously by contact forces in the environment, the position control by itself is not sufficient to perform a given task. The control of constrained motion requires a controller that combines both position and force control, which is called compliance control or simply force control. When robot manipulator is in contact with a constraining surface(called the environment), force is generated between the end effector and the environment. The amount of forces exerted by the robot to the environment depends on how much the end-effector position is physically constrained by the environment to reach its goal position. Hence, the contact force can be regulated by appropriate control of the end-effector position [81].

One simple way of achieving a compliant motion is to use the mechanical device, whose the end-effector is equipped with a passive mechanical device composed of springs and dampers so that the robot would maintain an appropriate position

and orientation of the end-effector. This passive mechanical device is known as the RCC(remote compliance center) which is based on this principle. Since the RCC is typically capable of responding quickly, and is relatively inexpensive, it is useful for introducing the compliance motion between the robot and the environment. However, their application is necessarily limited to very specific tasks. A programmable active device or active compliance control would allow the manipulator of various types of parts, or the modification of the end-effector' elastic behavior according to different phases of an assembly task. With these reasons, the sophisticated force control algorithms are desired.

One of main stream of force control is hybrid position/force control based on Mason's constrained frame work [82]. The hybrid position/force control combines force and torque information with position and orientation data to satisfy simultaneous position and force trajectory constrains specified in a convenient task related coordinate system. In other words, the basic idea behind hybrid position/force control is that the physical constraints of the task should guide those axes along which force is controlled and those axes along which position is controlled. The main problem of the hybrid position/force control is the ignorance of the dynamic behavior between the robot and the environment.

Another most frequently used approach is to adjust the robot end-effector position in response to the sensed contact force in such a way that a target impedance relationship is satisfied. This is the well known impedance force control concept [83].

Design of impedance control law is straight forward if the models of the robot manipulator and the environment are known with high precision. Uncertainties in these models degrade the system performance. In practice, the complete dynamic model of a robot is not known exactly, and the environment stiffness is also only approximately known. Accurate knowledge of the environment stiffness is very important in that it determines the robot target trajectory in order to attain the desired contact force. The objective of this chapter is to present a solution to impedance force control assuming that there are uncertainties in robot dynamics and environment stiffness. The proposed solution employs the sensed contact force and neural network technique in the controller design.

In the previous chapters, it has been shown that neural network(NN) are very effective in compensating for trajectory tracking in free space. Here we wish to extend these NN compensating concepts to force control in constrained space. One example of such kind was reported in [49] in which NN position/force control has been implemented for cooperative multiple robots. Adaptive impedance control technique proposed in [84] helps the mapping of Cartesian control input to joint control torque so that dynamic model of robot is not required under the position/force control. However, both studies did not address the issue of environment uncertainties.

There have been many studies made to solve the problem of stiffness uncertainties such as by employing separate trajectory modification control loop using integral control [85], or by generating reference position using adaptive control [86]. Although

these techniques do work, they tend to increase the complexity of the system dynamics which require special attention to system stability.

7.2 Hybrid Position/Force Control

One of he well known position/force control architecture is the hybrid position/force control. Position and force are separately controlled through a selection matrix that forms with zeros and ones. Position control loop is same as inverse dynamic control in Cartesian space, while the force control loop is to minimize the force error between the desired force and the actual force directly. The immediate concern is how to define the control law for each direction. This is one of problems of hybrid position/force control. The selection matrix which is a diagonal and its complement are used to represent a constrained frame. The selection matrix separates the position and force controllable directions in the end effector space by setting ones for force controlled direction and zeros for position controlled direction.

Let the robot dynamic model in Cartesian space be

$$\boldsymbol{D}^*\ddot{\boldsymbol{X}} + \boldsymbol{h}^* + \boldsymbol{F}_f^* = \boldsymbol{F} - \boldsymbol{F}_e \tag{7.1}$$

The control law can simply be formed as a combination of position and force control.

$$\boldsymbol{F} = [\boldsymbol{I} - \boldsymbol{S}]\boldsymbol{F}_p + [\boldsymbol{S}]\boldsymbol{F}_f + \boldsymbol{F}_h + \boldsymbol{F}_e \tag{7.2}$$

where $\boldsymbol{S}$ is a selection matrix and $\boldsymbol{F}_h$ is an estimation of $\boldsymbol{h}^* + \boldsymbol{F}_f^*$. Assuming that

$\boldsymbol{F_h} = \boldsymbol{h}^* + \boldsymbol{F}_f^*$ and combining (7.1) with (7.2) yields

$$\boldsymbol{D}^*\ddot{\boldsymbol{X}} = [\boldsymbol{I} - \boldsymbol{S}]\boldsymbol{F}_p + [\boldsymbol{S}]\boldsymbol{F}_f \tag{7.3}$$

This is a closed loop dynamic model which has the explicit relationship between position input and torque output. The robot can be force controlled or position controlled by switching the control law.

For the position control law, consider a PD controller with acceleration feedforward term such as

$$\boldsymbol{F_p} = \hat{\boldsymbol{D}}^*(\ddot{\boldsymbol{X}}_d + \boldsymbol{K_D}\delta\dot{\boldsymbol{X}} + \boldsymbol{K_P}\delta\boldsymbol{X}) \tag{7.4}$$

where $\delta\boldsymbol{X} = \boldsymbol{X_d} - \boldsymbol{X}$. Combining (7.3) with (7.4) results in the second order position error equation

$$\delta\ddot{\boldsymbol{X}} + \boldsymbol{K_D}\delta\dot{\boldsymbol{X}} + \boldsymbol{K_P}\delta\boldsymbol{X} = \boldsymbol{0} \tag{7.5}$$

The same results of computed torque control is achieved.

The similar control law can be used for force control. Consider a PD force feedback controller with a feedforward term.

$$\boldsymbol{F_f} = \hat{\boldsymbol{D}}^*\boldsymbol{K_e^{-1}}(\ddot{\boldsymbol{X}}_d + \boldsymbol{K_D}\delta\dot{\boldsymbol{X}} + \boldsymbol{K_P}\delta\boldsymbol{X}) \tag{7.6}$$

Substituting (7.6) into (7.3) yields the second order force error equation as

$$\delta\ddot{\boldsymbol{F}} + \boldsymbol{K_D}\delta\dot{\boldsymbol{F}} + \boldsymbol{K_P}\delta\boldsymbol{F} = \boldsymbol{0} \tag{7.7}$$

where $\delta\boldsymbol{F} = \boldsymbol{F}_d - \boldsymbol{F}$.

In practical situations, the force control law (7.6) can not be obtained because the environment stiffness $\boldsymbol{K_e^{-1}}$ is not exactly known to us. Another problem is that the dynamic relationship between the position and force is ignored. Hence undesirable large overshoot of impact force is expected when the robot makes a contact with the environment.

7.3 Torque-Based Impedance Force Control

Direct application of position controllers designed for free space robot motion usually results in instability when robot hand is in contact with a constraining environment because these controllers ignore the fundamental dynamic relationship of interaction between the robot and the environment. Impedance control allows us to specify the robot stiffness for a given task under contact with the environment. The interaction between the environment and the end-point of the robot in terms of a desired mechanical stiffness, called impedance is regulated. Thus the impedance control regulates the relationship between force and position by setting suitable gains of impedance parameters.

The control law $\boldsymbol{F}$ for impedance force control then is

$$\boldsymbol{F} = \hat{\boldsymbol{D}}^* \boldsymbol{U} + \hat{\boldsymbol{h}}^* + \boldsymbol{F}_e \tag{7.8}$$

where $\hat{\boldsymbol{D}}^*, \hat{\boldsymbol{h}}^*$ are estimates of $\boldsymbol{D}^*, \boldsymbol{h}^*$ and $\boldsymbol{F}_e$ is the exerted force on the environment,

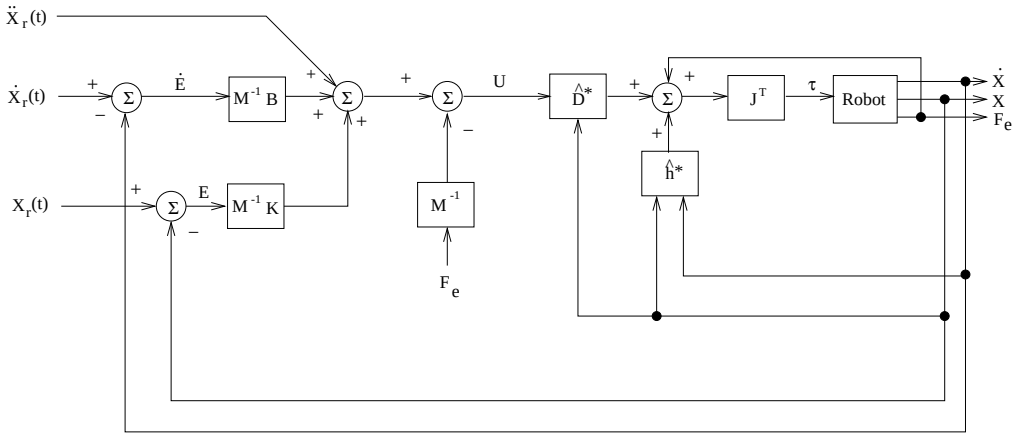


Figure 7.1: Torque-Based Force Control Structure

and $\boldsymbol{U}$ is given by

$$\boldsymbol{U = \ddot{X}_r + M^{-1}[B(\dot{X}_r - \dot{X}) + K(X_r - X) - F_e]} \tag{7.9}$$

where $\boldsymbol{M}$ $\boldsymbol{B}$ and $\boldsymbol{K}$ are $\boldsymbol{n}$ x $\boldsymbol{n}$ symmetric positive definite desired inertia, damping, and stiffness gain matrices, respectively, and $\boldsymbol{X_r}$ is the reference end-point trajectory and $\boldsymbol{X}$ is the actual end-point trajectory. Combining (7.1),(7.8), and (7.9) yields the closed loop tracking error equation

$$\boldsymbol{\ddot{E} + M^{-1}[B\dot{E} + KE - F_e] = \hat{D}^{*^{-1}}[\Delta D^* \ddot{X} + \Delta h^* + F_f^*]} \tag{7.10}$$

where $\boldsymbol{\Delta D^* = D^* - \hat{D}^*, \Delta h = h^* - \hat{h}^*}$, and $\boldsymbol{E = (X_r - X)}$. In the ideal case where $\boldsymbol{\Delta D^* = \Delta h^* = 0}$, and $\boldsymbol{F_f^* = 0}$, the closed loop robot satisfies the target impedance relationships

$$\boldsymbol{F_e = M\ddot{E} + B\dot{E} + KE} \tag{7.11}$$

Since there are always uncertainties in the robot dynamic model($\boldsymbol{\Delta D^* \neq 0, \Delta h^* \neq 0}$), the ideal target impedance relationships (7.11) can not be achieved in general.

Thus the impedance based force control is not robust in practice. To improve robustness, NN controller can be introduced in (7.10) to compensate for the disturbances due to model uncertainties.

Another problem is the difficulty of knowing the environment stiffness accurately in advance for us to design the reference trajectory $\boldsymbol{X_r}$ to achieve the desired contact force. We will address this issue in section 7.4.

7.4 Position-Based Impedance Force Control

The position-based impedance control is implemented outside the computed torque controlled loop. Position-based impedance control scheme has the advantage of easy implementation on a position-controlled robot system [87]. However, the position controller has to be highly accurate in order to achieve accurate force control.

In position-based force control, the impedance controller is added to form an additional control loop around the position controlled manipulator as shown in Figure 7.2. The reference trajectory $\boldsymbol{X_r}$ is adjusted by $\boldsymbol{X_a}$ which is determined based on sensed contact force $\boldsymbol{F_e}$ and impedance parameters. The desired relationship between $\boldsymbol{X_a}$ and $\boldsymbol{F_e}$ can be expressed as the impedance function

$$\boldsymbol{F_e} = \boldsymbol{M}\ddot{\boldsymbol{X}}_a + \boldsymbol{B}\dot{\boldsymbol{X}}_a + \boldsymbol{K}\boldsymbol{X_a} \tag{7.12}$$

In frequency domain the impedance function is represented by

$$\boldsymbol{X}_a(\boldsymbol{s}) = [\boldsymbol{M}\boldsymbol{s}^2 + \boldsymbol{B}\boldsymbol{s} + \boldsymbol{K}]^{-1}\boldsymbol{F}_e(\boldsymbol{s}) \tag{7.13}$$

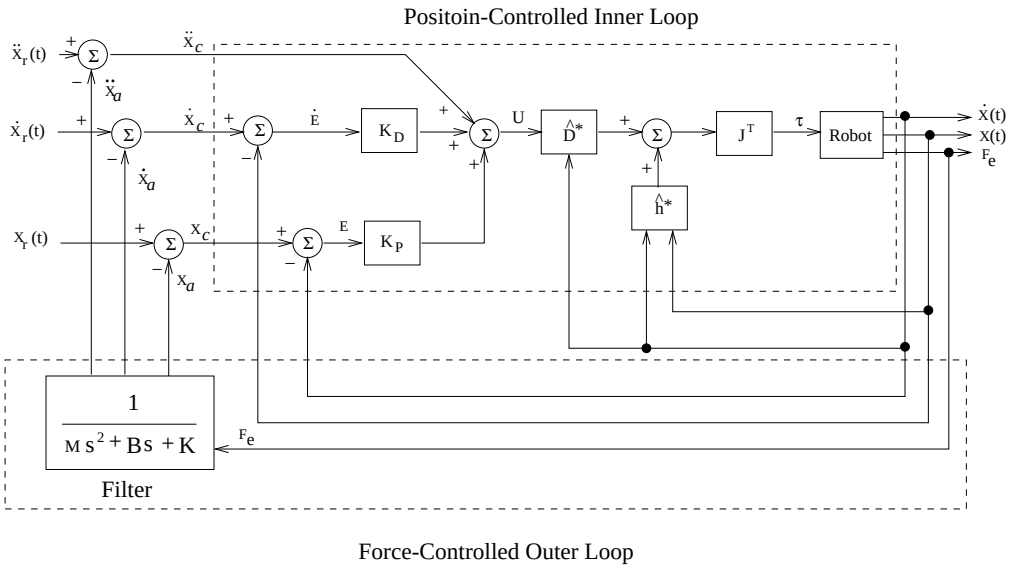


Figure 7.2: Position-Based Force Control Structure

The reference command trajectory $\boldsymbol{X_r}$ and the robot position command $\boldsymbol{X_c}$ are related by

$$\boldsymbol{X_c} = \boldsymbol{X_r} - \boldsymbol{X_a} \tag{7.14}$$

In order to regulate $\boldsymbol{F_e}$ to track a desired force $\boldsymbol{F_d}$, the reference trajectory $\boldsymbol{X_r}$ should be appropriately designed [2].

The control law for implementing the above impedance control concept is

$$\boldsymbol{F} = \hat{\boldsymbol{D}}^* \boldsymbol{U} + \hat{\boldsymbol{h}}^* + \boldsymbol{F_e} \tag{7.15}$$

and

$$\boldsymbol{U} = \ddot{\boldsymbol{X}}_c + \boldsymbol{K_D}(\dot{\boldsymbol{X}}_c - \dot{\boldsymbol{X}}) + \boldsymbol{K_P}(\boldsymbol{X_c} - \boldsymbol{X}) \tag{7.16}$$

where $\hat{\boldsymbol{D}}^*$ and $\hat{\boldsymbol{h}}^*$ are nominal values for $\boldsymbol{D}^*$ and $\boldsymbol{h}^*$, $\boldsymbol{K_D}$ and $\boldsymbol{K_P}$ are the desired positive definite diagonal matrices for damping and stiffness gains. Combining

(7.1),(7.15), and (7.16) yields the closed loop position tracking error equation

$$\ddot{\boldsymbol{E}}+\boldsymbol{K_D}\dot{\boldsymbol{E}}+\boldsymbol{K_P}\boldsymbol{E}=\hat{\boldsymbol{D}^*}^{-1}[\boldsymbol{\Delta D^*}\ddot{\boldsymbol{X}}+\boldsymbol{\Delta h^*}+\boldsymbol{F_f^*}]+\ddot{\boldsymbol{X}}_a+\boldsymbol{K_D}\dot{\boldsymbol{X}}_a+\boldsymbol{K_P}\boldsymbol{X}_a \quad (7.17)$$

where $\boldsymbol{\Delta D^*} = \boldsymbol{D^*} - \hat{\boldsymbol{D}^*}, \boldsymbol{\Delta h} = \boldsymbol{h^*} - \hat{\boldsymbol{h}^*}$, and $\boldsymbol{E} = (\boldsymbol{X_r} - \boldsymbol{X})$. The term $\hat{\boldsymbol{D}^*}^{-1}[\boldsymbol{\Delta D^*}\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h^*} + \boldsymbol{F_f^*}]$ is the robot dynamic uncertainty. In the ideal case where $\boldsymbol{\Delta D^*} = \boldsymbol{\Delta h^*} = \boldsymbol{F_f^*} = \boldsymbol{0}$, the closed loop robot motion satisfies

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \ddot{\boldsymbol{X}}_a + \boldsymbol{K_D}\dot{\boldsymbol{X}}_a + \boldsymbol{K_P}\boldsymbol{X}_a \quad (7.18)$$

By defining $\boldsymbol{B} = \boldsymbol{M}\boldsymbol{K_D}$ and $\boldsymbol{K} = \boldsymbol{M}\boldsymbol{K_P}$ the desired impedance relation (7.6) can be satisfied by

$$\boldsymbol{F_e} = \boldsymbol{M}\ddot{\boldsymbol{E}} + \boldsymbol{B}\dot{\boldsymbol{E}} + \boldsymbol{K}\boldsymbol{E} \quad (7.19)$$

Letting $\boldsymbol{F_e} = \boldsymbol{K_e}(\boldsymbol{X} - \boldsymbol{X_e})$ where $\boldsymbol{K_e}$ is the environment stiffness constant and $\boldsymbol{X_e}$ is the environment position, we have

$$\boldsymbol{M}\ddot{\boldsymbol{E}} + \boldsymbol{B}\dot{\boldsymbol{E}} + \boldsymbol{K}\boldsymbol{E} = \boldsymbol{K_e}(\boldsymbol{X} - \boldsymbol{X_e}) \quad (7.20)$$

Replacing $\boldsymbol{X} - \boldsymbol{X_e}$ with $\boldsymbol{X_r} - \boldsymbol{X_e} - \boldsymbol{X_r} + \boldsymbol{X}$ in (7.20) yields

$$\boldsymbol{M}\ddot{\boldsymbol{E}} + \boldsymbol{B}\dot{\boldsymbol{E}} + (\boldsymbol{K} + \boldsymbol{K_e})\boldsymbol{E} = \boldsymbol{K_e}(\boldsymbol{X_r} - \boldsymbol{X_e}) \quad (7.21)$$

We note that the damping condition of (7.21) varies for different environment stiffnesses. In order to assure that the system behaves over-damped, impedance gains $\boldsymbol{M}, \boldsymbol{B}$ and $\boldsymbol{K}$ should be adjusted in accordance with environment stiffness $\boldsymbol{K_e}$. For example, the damping gain $\boldsymbol{B}$ should be increased to suppress the overshoot at initial

contact of the robot with the environment if the environment is stiff. We show in the simulations that different impedance gains for different environment stiffnesses can be appropriately selected.

Since there are always uncertainties in the robot dynamic model such that $\Delta D^*, \Delta h^*$ and F_f^* are not identically zero, the ideal target impedance relationships (7.13) can not be achieved in general. The actual system performance is governed by (7.17) which has an unpredictable dynamics behavior. Thus impedance force control is not robust. To improve robustness, NN controller can be introduced to generate additional signals to compensate for the disturbances due to model uncertainties. In addition to that, it is necessary to develop an algorithm that can specify the reference trajectory X_r in order to achieve a desired force tracking under unknown environment stiffness. In the next section, we like to develop such an algorithm that computes the reference trajectory with respect to the environment stiffness.

7.5 Computing Reference Trajectory using Sensed Force

Regulating F_e to track a desired contact force F_d, X_r must be appropriately designed based on F_d, X_e (environment position), K, and K_e (environment stiffness) as follows:

$$x_r = x_e + \Delta x_r \qquad \Delta x_r = \frac{f_d}{k_{eff}} \tag{7.22}$$

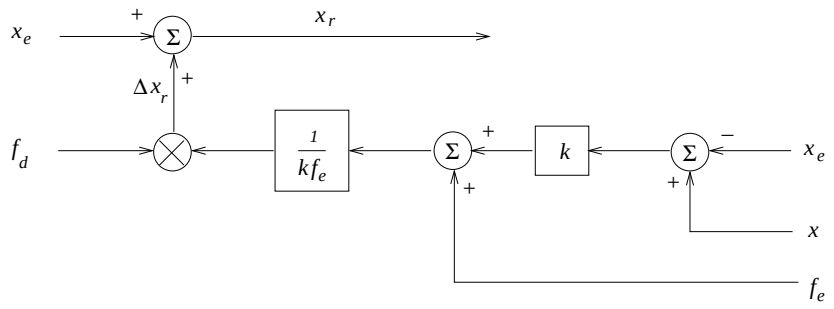


Figure 7.3: The Simple Trajectory Modification Loop

where $\boldsymbol{x_r}, \boldsymbol{x_e}, \boldsymbol{f_d}$ represents an element of $\boldsymbol{X_r}, \boldsymbol{X_e}, \boldsymbol{F_d}$, and $\boldsymbol{k_{eff}}$ is the combined stiffness of the environment and the robot [2]

$$\boldsymbol{k_{eff}} = \frac{\boldsymbol{k k_e}}{\boldsymbol{k + k_e}} \tag{7.23}$$

in which $\boldsymbol{k}$ and $\boldsymbol{k_e}$ are corresponding elements of $\boldsymbol{K}$ and $\boldsymbol{K_e}$. So when the environment stiffness $\boldsymbol{k_e}$ is known exactly, it is easy to calculate the reference trajectory $\boldsymbol{x_r}$ from (7.22) and (7.23). In practice, $\boldsymbol{k_e}$ is not accurately measurable so that $\boldsymbol{x_r}$ is also inaccurate resulting poor force control performance.

We propose here not to use $\boldsymbol{k_e}$ directly in (7.23), rather to identify the environment stiffness $\boldsymbol{k_e}$ through the relationship(assume the environment is an ideal spring)

$$\boldsymbol{f_e} = \boldsymbol{k_e}(\boldsymbol{x} - \boldsymbol{x_e}) \tag{7.24}$$

Solving for $\boldsymbol{k_e}$ yields

$$\boldsymbol{k_e} = \frac{\boldsymbol{f_e}}{\boldsymbol{x} - \boldsymbol{x_e}} \tag{7.25}$$

where $\boldsymbol{f_e}$ is the measured contact force obtained from a wrist force sensor. In practice (7.25) is only a close estimate of $\boldsymbol{k_e}$ as $\boldsymbol{f_e}$ may be subjected to some measurement

error. Substituting (7.25) into (7.23) yields

$$\boldsymbol{k_{eff}} = \frac{\boldsymbol{k}\frac{\boldsymbol{f_e}}{(\boldsymbol{x}-\boldsymbol{x_e})}}{\boldsymbol{k}+\frac{\boldsymbol{f_e}}{(\boldsymbol{x}-\boldsymbol{x_e})}} = \frac{\boldsymbol{k f_e}}{\boldsymbol{k}(\boldsymbol{x}-\boldsymbol{x_e})+\boldsymbol{f_e}} \tag{7.26}$$

Combining (7.26) with (7.22) yields

$$\boldsymbol{x_r} = \boldsymbol{x_e} + \boldsymbol{f_d}[\frac{\boldsymbol{k}(\boldsymbol{x}-\boldsymbol{x_e})+\boldsymbol{f_e}}{\boldsymbol{k f_e}}] \tag{7.27}$$

This is the desired results for designing $\boldsymbol{x_r}$. We see that the reference trajectory $\boldsymbol{x_r}$ is modified from $\boldsymbol{x_e}$ based on $\boldsymbol{f_d}, \boldsymbol{f_e}, \boldsymbol{k}, \boldsymbol{x},$ and $\boldsymbol{x_e}$, all of which are assumed to be known at all times. We note however that (7.27) is invalid when $\boldsymbol{f_e} = \boldsymbol{0}$. This case represents the moment just before the robot is exerting force on the environment and $\boldsymbol{x} = \boldsymbol{x_e}$. So we see that the trajectory modification term $\boldsymbol{\Delta x_r}$

$$\boldsymbol{\Delta x_r} = \boldsymbol{f_d}\frac{\boldsymbol{k}(\boldsymbol{x}-\boldsymbol{x_e})+\boldsymbol{f_e}}{\boldsymbol{k f_e}} \tag{7.28}$$

is indefinite at $\boldsymbol{f_e} = \boldsymbol{0}$ when $\boldsymbol{x} - \boldsymbol{x_e} = \boldsymbol{0}$. To take care of this case, we determine the limits of $\boldsymbol{\Delta x_r}$ for $\boldsymbol{f_e} = \boldsymbol{0}$ at $\boldsymbol{x} - \boldsymbol{x_e} = \boldsymbol{0}$ using $\boldsymbol{L'Hopital'}$s rule which yields $\boldsymbol{\Delta x_r} = \frac{\boldsymbol{f_d}}{\boldsymbol{k}}$. Therefore (7.28) is modified as

$$\boldsymbol{x_r} = \begin{cases} \boldsymbol{x_e} + \frac{\boldsymbol{f_d}}{\boldsymbol{k}} & \text{if } \boldsymbol{f_e} = \boldsymbol{0} \\ \boldsymbol{x_e} + \boldsymbol{f_d}[\frac{\boldsymbol{k}(\boldsymbol{x}-\boldsymbol{x_e})+\boldsymbol{f_e}}{\boldsymbol{k f_e}}] & \text{if } \boldsymbol{f_e} \neq \boldsymbol{0} \end{cases} \tag{7.29}$$

$\boldsymbol{\dot{x}_r}\boldsymbol{\ddot{x}_r}$ can be obtained by differentiating (7.29). This result is applicable to robot position control for both contact and non contact phases. Simulations based on (7.29) are carried out to demonstrate the performance of the proposed design.

7.6 Proposed NN Controller Scheme

In this section, we present an NN controller design to achieve the desired impedance function (7.13) in the presence of robot model uncertainty. The NN replaces the conventional impedance controller.

7.6.1 Torque-Based NN Controller Scheme

In this section, we present an NN controller design for the impedance control system as shown in Figure 7.4. Let the NN output be denoted as $\mathbf{\Phi}$. A new control law is defined as

$$\boldsymbol{F} = \hat{\boldsymbol{D}}^*(\boldsymbol{U} + \mathbf{\Phi}) + \hat{\boldsymbol{h}^*} + \boldsymbol{F_e} \tag{7.30}$$

where $\boldsymbol{U}$ is the same as (7.3). Combining (7.30) with (7.1) yields the corresponding closed loop error system as

$$\ddot{\boldsymbol{E}} + \boldsymbol{M}^{-1}[\boldsymbol{B}\dot{\boldsymbol{E}} + \boldsymbol{K}\boldsymbol{E} - \boldsymbol{F_e}] = \hat{\boldsymbol{D}^*}^{-1}(\boldsymbol{\Delta D}^*\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^* + \boldsymbol{F_f^*}) - \mathbf{\Phi} \tag{7.31}$$

Define an error signal $\boldsymbol{v}$ as

$$\boldsymbol{v} = \ddot{\boldsymbol{E}} + \boldsymbol{M}^{-1}[\boldsymbol{B}\dot{\boldsymbol{E}} + \boldsymbol{K}\boldsymbol{E} - \boldsymbol{F_e}] \tag{7.32}$$

which is a desired impedance equation. The NN control objective is to generate $\mathbf{\Phi}$ to reduce $\boldsymbol{v}$ to zero in (7.31). We therefore propose here to use $\boldsymbol{v}$ as the training signal for the NN. The ideal value of $\mathbf{\Phi}$ at $\boldsymbol{v} = \mathbf{0}$ then is

$$\mathbf{\Phi} = \hat{\boldsymbol{D}^*}^{-1}(\boldsymbol{\Delta D}^*\ddot{\boldsymbol{X}} + \boldsymbol{\Delta h}^* + \boldsymbol{F_f^*}) \tag{7.33}$$

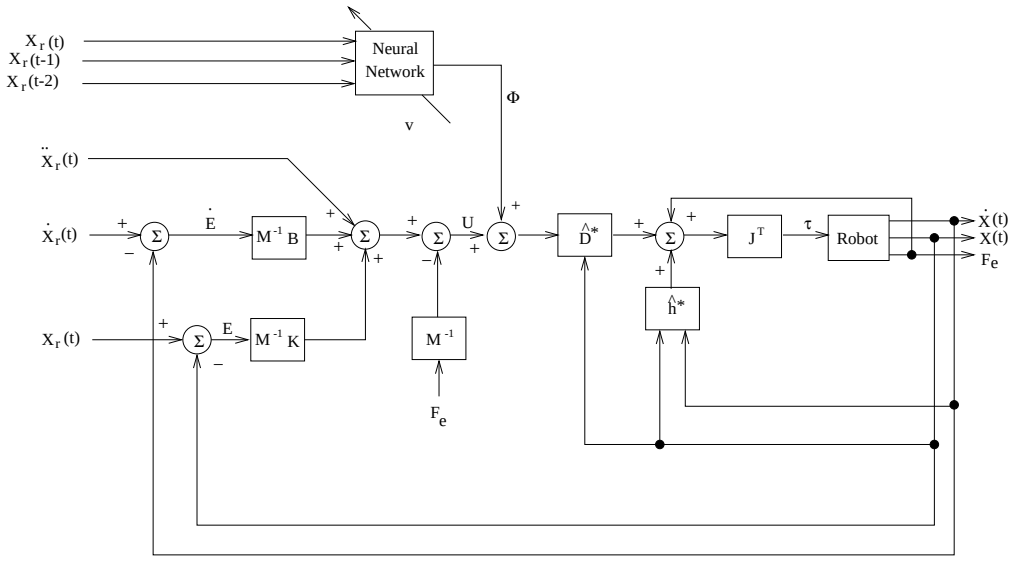


Figure 7.4: The proposed Torque-based NN Force Control Structure

Clearly minimizing the error signal $\boldsymbol{v}$ allows us to achieve ideal impedance relationship (7.13).

7.6.1.1 Neural Network Compensator Design

The weight updating law minimizes the objective function $\mathcal{J}$ which is a quadratic function of the training signal $\boldsymbol{v}$.

$$\mathcal{J} = \frac{1}{2}\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v} \tag{7.34}$$

Differentiating (7.34) and making use of (7.31) yields the gradient of $\mathcal{J}$ as follows:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{v}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{v} = -\frac{\partial \boldsymbol{\Phi}^{\boldsymbol{T}}}{\partial \boldsymbol{w}}\boldsymbol{v} \tag{7.35}$$

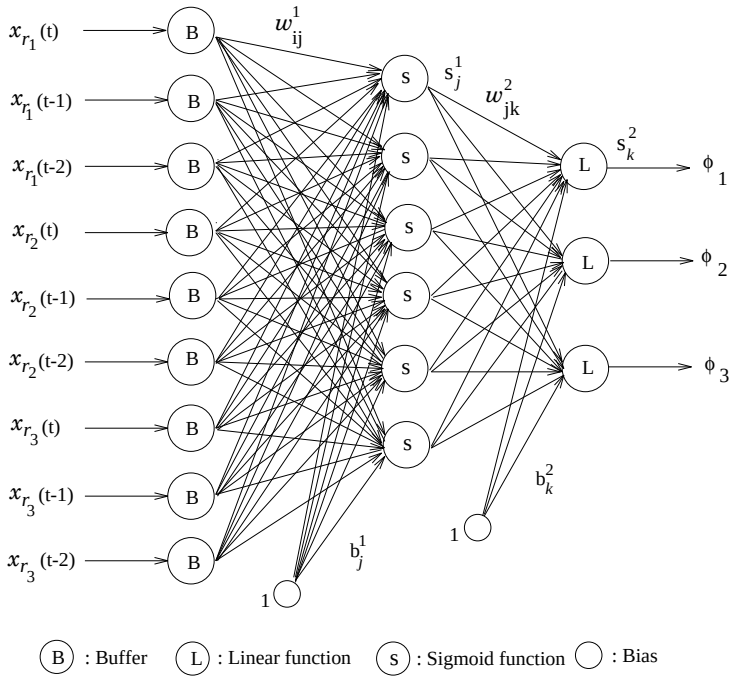


Figure 7.5: NN structure

in which the fact $\frac{\partial v^T}{\partial w} = -\frac{\partial \Phi^T}{\partial w}$ is used. The back-propagation update rule for the weights with a momentum term is

$$\Delta w(t) = \eta \frac{\partial \Phi^T}{\partial w} v + \alpha \Delta w(t-1) \tag{7.36}$$

where η is the update rate and α is the momentum coefficient. Specifically,

$$\Delta w^1_{ij}(t) = \eta s^1_j (1 - s^1_j) x_{r_i} [\sum_{k=1}^{n} v_k w^2_{jk}] + \alpha \Delta w^1_{ij}(t-1) \tag{7.37}$$

$$\Delta w^2_{jk}(t) = \eta v_k s^1_j + \alpha \Delta w^2_{jk}(t-1) \tag{7.38}$$

$$\Delta b^1_j(t) = \eta s^1_j (1 - s^1_j) [\sum_{k=1}^{n} v_k w^2_{jk}] + \alpha \Delta b^1_j(t-1) \tag{7.39}$$

$$\Delta b^2_k(t) \;\; = \;\; \eta v_k + \alpha \Delta b^2_k(t-1) \tag{7.40}$$

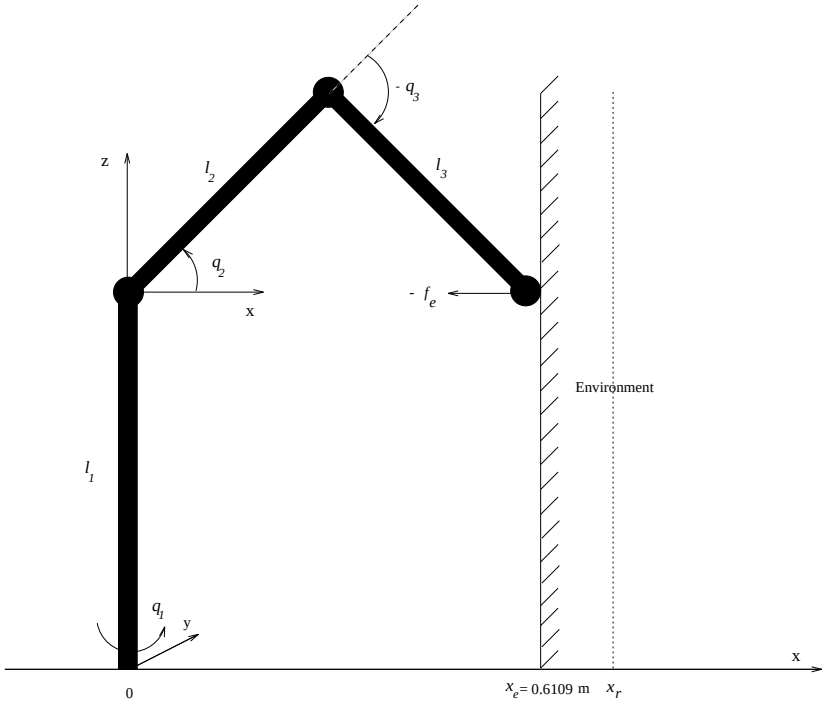


Figure 7.6: Task 1 : Sine Wave Tracking on Flat Surface Environment

$$s_j^1 = \frac{1}{1 + exp(-(\sum_{i=1}^{n_I} x_{r_i} w_{ij}^1 + b_j^1))} \tag{7.41}$$

$$s_k^2 = \sum_{i=1}^{n_H} s_j^1 w_{jk}^2 + b_k^2 \tag{7.42}$$

where s_j^1 is the output of jth hidden neuron and s_k^2 is the output of kth output neuron.

7.6.1.2 Simulation Results

The performances of the proposed schemes are tested by tracking two different tasks : sine wave path tracking(called Task 1) and circular path tracking(called Task 2) as shown in Figure 7.6 and 7.7, respectively. For the NN controller, we have chosen six hidden neurons $(n_H = 6)$. The back-propagation algorithm parameters are:

$\boldsymbol{w_{ij}^k(0) = 0, \alpha = 0.9}$. The environment stiffness $\boldsymbol{k_e}$ is $\boldsymbol{1000N/m}$. The controller gains are selected as $\boldsymbol{K_D = diag[60, 20, 20]}$ and $\boldsymbol{K_P = diag[100, 100, 100]}$ for Task 1, and $\boldsymbol{K_D = diag[60, 60, 60]}$ and $\boldsymbol{K_P diag[100, 100, 100]}$ for Task 2, all of which give critically damped or over-damped motions at the end-effector. For the Task 1, the robot manipulator is required to move up and down sinusoidally in $\boldsymbol{z}$ direction while exerting a constant 5N force on $\boldsymbol{x}$ direction. The one cycle time is 4 secs. The update rate $\boldsymbol{\eta = 0.08}$ is used. Sample tracking results for Task 1 are plotted in Figures 7.9. The simulation data showed that the rate of convergence is very fast with a convergence time less than 0.5 second. Thus on-line trajectory control of the proposed NN scheme is completely acceptable in practice. As a base line comparison, we also plotted the performance of the uncompensated system under the same condition in Figure 7.9. It is seen that the force tracking errors in this case are very large, so improvement of using NN control is clearly demonstrated.

The corresponding position tracking in $\boldsymbol{z}$ axis is shown in Figure 10. Task 2 is to require the robot to move a circular trajectory as shown in Figure 7.11. Since the environment is tilted by 45 degrees, the total force normal to the surface has three components $\boldsymbol{f_{e_x}, f_{e_y}, f_{e_z}}$ in $\boldsymbol{x, y, z}$ direction. The update rate $\boldsymbol{\eta}$ for this task is 0.05. Figure 7.11 shows the end-point reference trajectories $\boldsymbol{x_r}$ and actual trajectories $\boldsymbol{x}$. This plot demonstrates that position tracking under NN control is excellent. The corresponding force tracking response is plotted in Figure 7.12. Force tracking of the uncompensated control system is plotted as well for comparison. The improvement

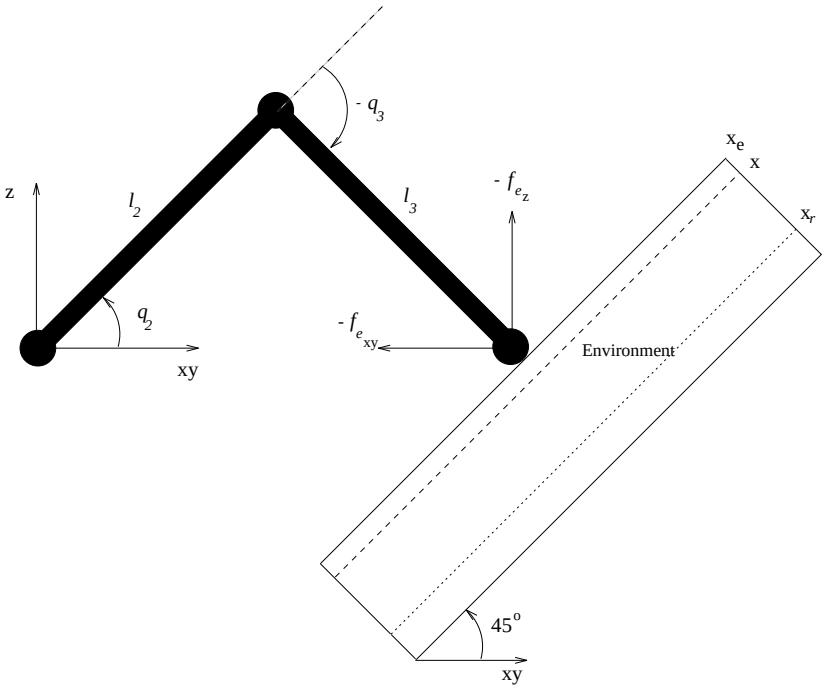


Figure 7.7: Task 2 : Circular Tracking on $\mathbf{45^o}$ Tilted Surface Environment

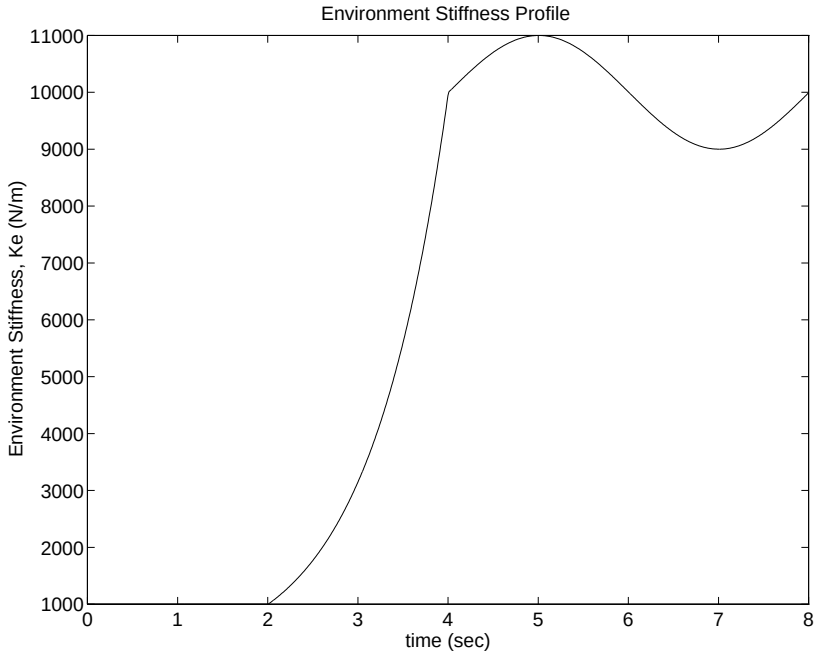


Figure 7.8: Unknown Environment Stiffness Profile

achieved by the NN controller is again clearly demonstrated. Figure 7.13 shows the position tracking of each axes. Plotted in Figure 7.14 are the forces $f_{e_x}, f_{e_y}, f_{e_z}$. We see that the small force overshoots at initial impact settled down within 1 sec.

Finally, in order to show the versatility of design technique(Figure 7.8), we tested the system performance for a time varying environment stiffnesses with the stiffness profile as

$$k_e = \begin{cases} 1000 & 0 \leq t < 2 \\ 1000\ e^{1.1513(t-2)} & 2 \leq t < 4 \\ 10000 + 1000\ sin(\pi t) & 4 \leq t \end{cases} \tag{7.43}$$

As expected, the impedance controller is able to track effectively the desired contact force as shown in Figures 7.15 and 16. For these tasks, the update rate $\eta = 0.015$ and $\eta = 0.03$ are used. We see that the reference modification algorithm works well under unknown environment stiffness.

7.6.2 Proposed Position-Based NN Controller Scheme

The NN outputs are (ϕ_a, ϕ_d, ϕ_p), and the NN inputs can be either $(X_r(t),\ \dot{X}_r(t),\ \ddot{X}_r(t),\ F_e(t))$ or the time-delayed version $(X_r(t),\ X_r(t-1),\ X_r(t-2),\ F_e(t))$ [19]. Delay time is chosen as the sampling period of the controller. We found in our simulations that the NN performs better when time-delayed values of X_r are used instead of its velocity and acceleration values calculated from finite difference approximations [67].

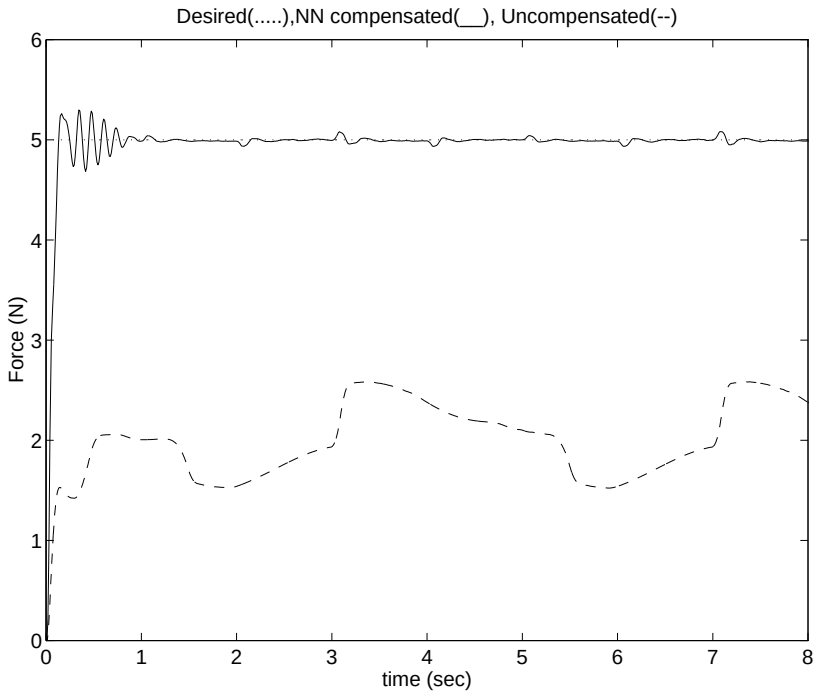


Figure 7.9: Task 1 (Flat Wall) : Force Tracking

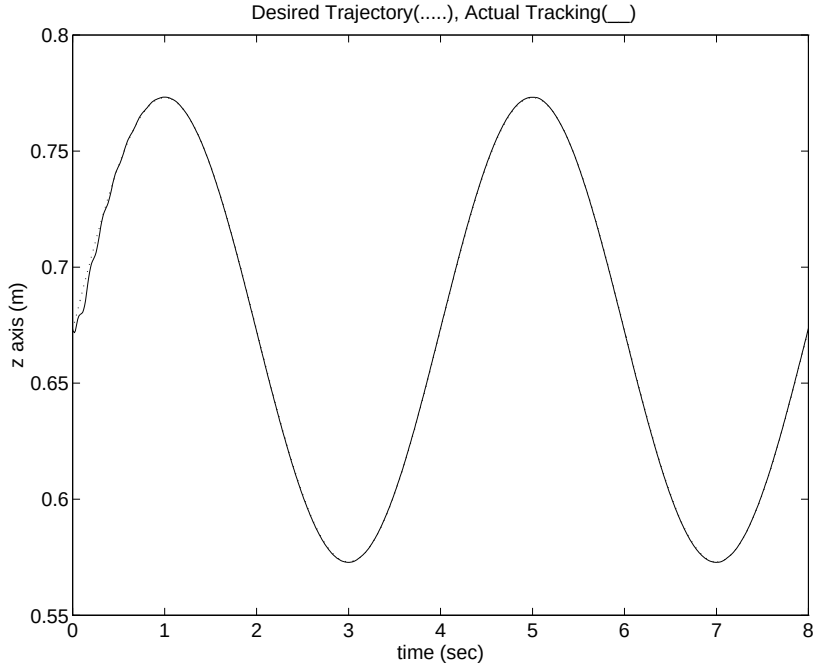


Figure 7.10: Task 1 (Flat Wall) : Position Tracking

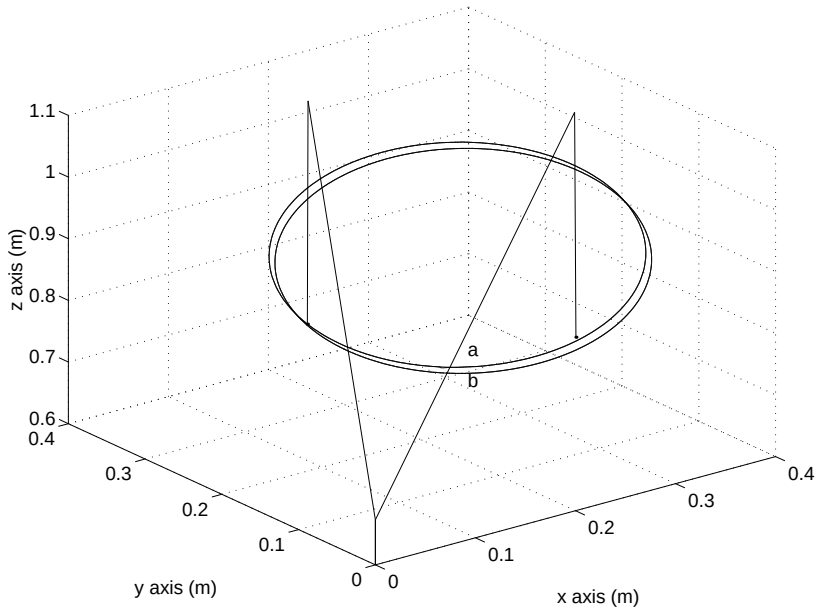


Figure 7.11: Task 2 : End point Tracking of Circular Trajectory (a) Actual Tracking (b) Reference Trajectory

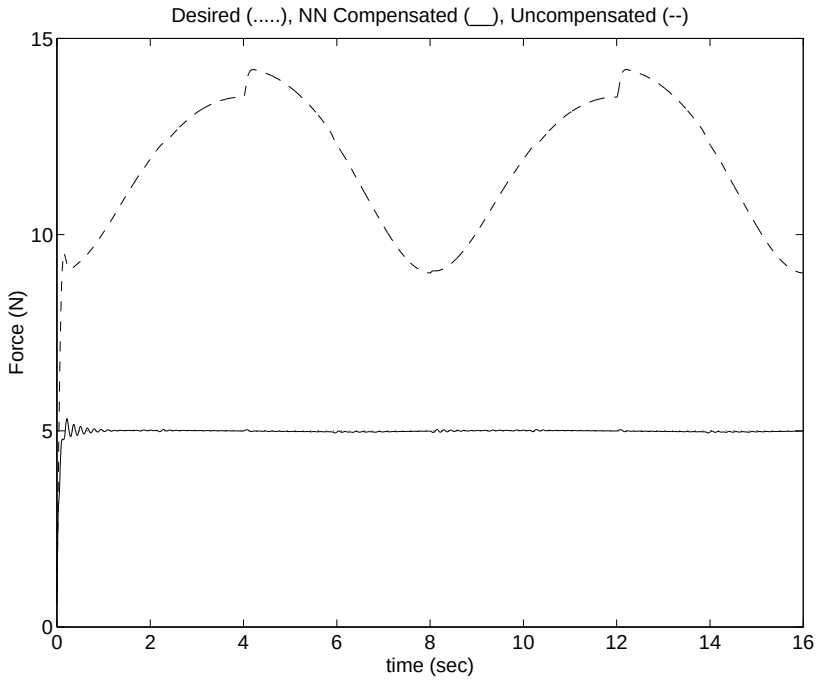


Figure 7.12: Task 2 (Circular Trajectory Tracking) : Force Tracking

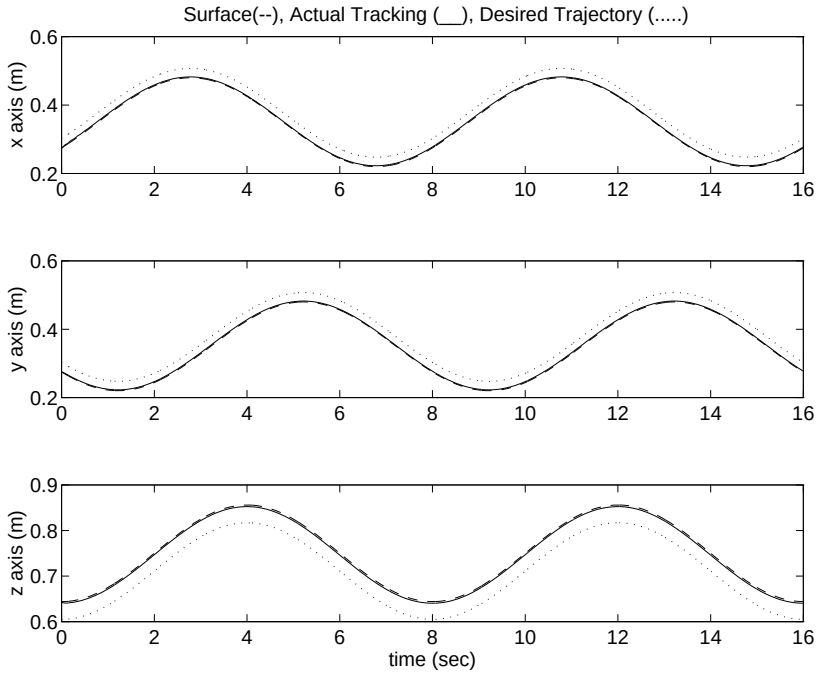


Figure 7.13: Task 2 (Circular Trajectory Tracking) : Position Tracking

The proposed control law for $\boldsymbol{F}$ is

$$\boldsymbol{F} = \hat{\boldsymbol{D}}^{*}\boldsymbol{U} + \hat{\boldsymbol{h}}^{*} \tag{7.44}$$

and $\boldsymbol{U}$ is

$$\boldsymbol{U} = \ddot{\boldsymbol{X}}_{\boldsymbol{c}} + \boldsymbol{K_D}(\dot{\boldsymbol{X}}_{\boldsymbol{c}} - \dot{\boldsymbol{X}}) + \boldsymbol{K_P}(\boldsymbol{X}_{\boldsymbol{c}} - \boldsymbol{X}) - \boldsymbol{M^{-1}F_e} \tag{7.45}$$

We note that $\boldsymbol{F_e}$ is introduced in $\boldsymbol{U}$ so that the impedance controller is outside the robot position control system.

Substituting $\boldsymbol{X_c} = \boldsymbol{X_r} + \boldsymbol{\phi_p}, \dot{\boldsymbol{X}}_{\boldsymbol{c}} = \dot{\boldsymbol{X}}_{\boldsymbol{r}} + \boldsymbol{\phi_d}, \ddot{\boldsymbol{X}}_{\boldsymbol{a}} = \ddot{\boldsymbol{X}}_{\boldsymbol{r}} + \boldsymbol{\phi_a}$ into $\boldsymbol{U}$ in (7.39) yields

$$\boldsymbol{U} = \ddot{\boldsymbol{X}}_{\boldsymbol{r}} + \boldsymbol{\phi_a} + \boldsymbol{K_D}(\dot{\boldsymbol{X}}_{\boldsymbol{r}} - \dot{\boldsymbol{X}} + \boldsymbol{\phi_d}) + \boldsymbol{K_P}(\boldsymbol{X_r} - \boldsymbol{X} + \boldsymbol{\phi_p}) - \boldsymbol{M^{-1}F_e} \tag{7.46}$$

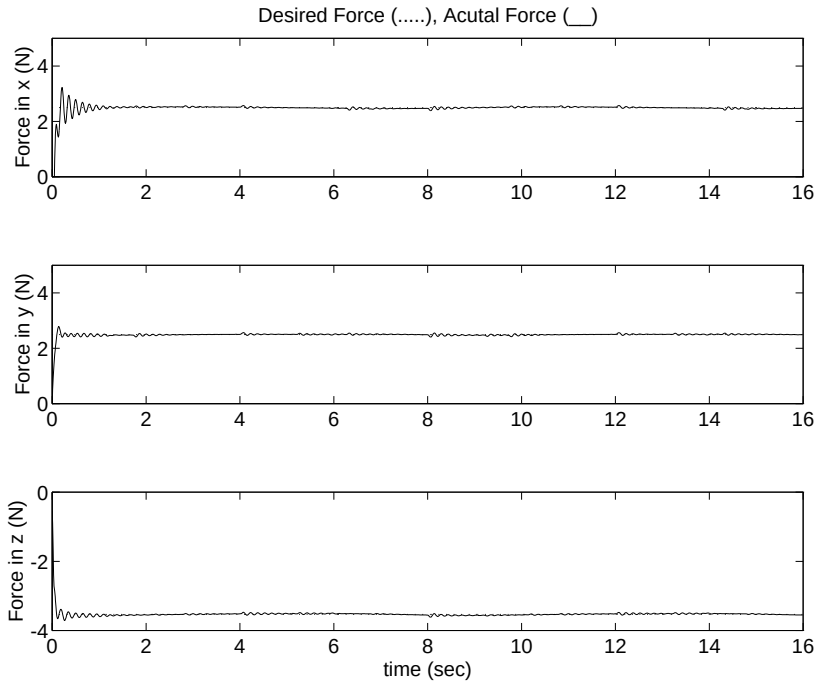


Figure 7.14: Task 2 (Circular Trajectory Tracking) : Force Tracking in x,y,z axis

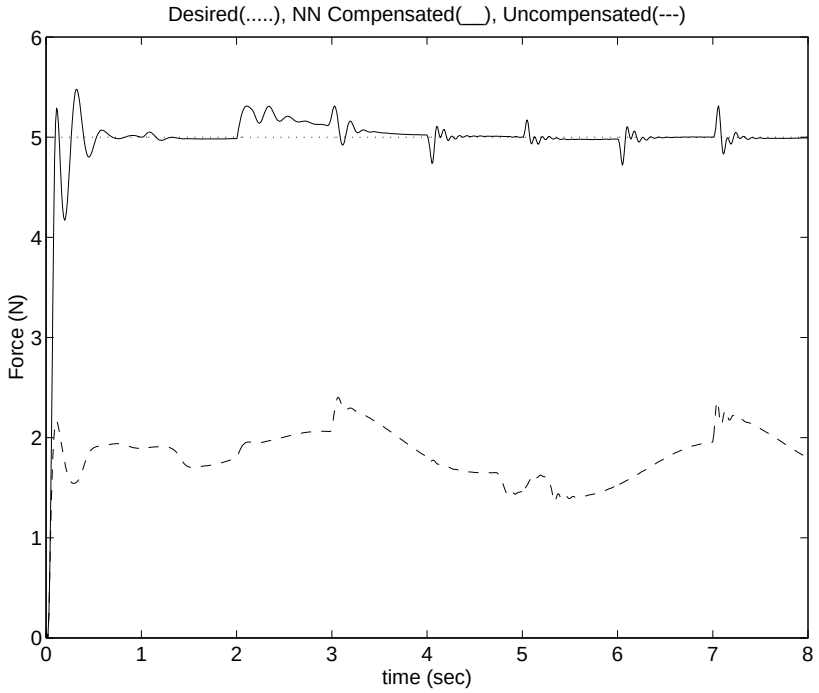


Figure 7.15: Force Tracking Flat Wall Tracking with Environment Stiffness Identification Method

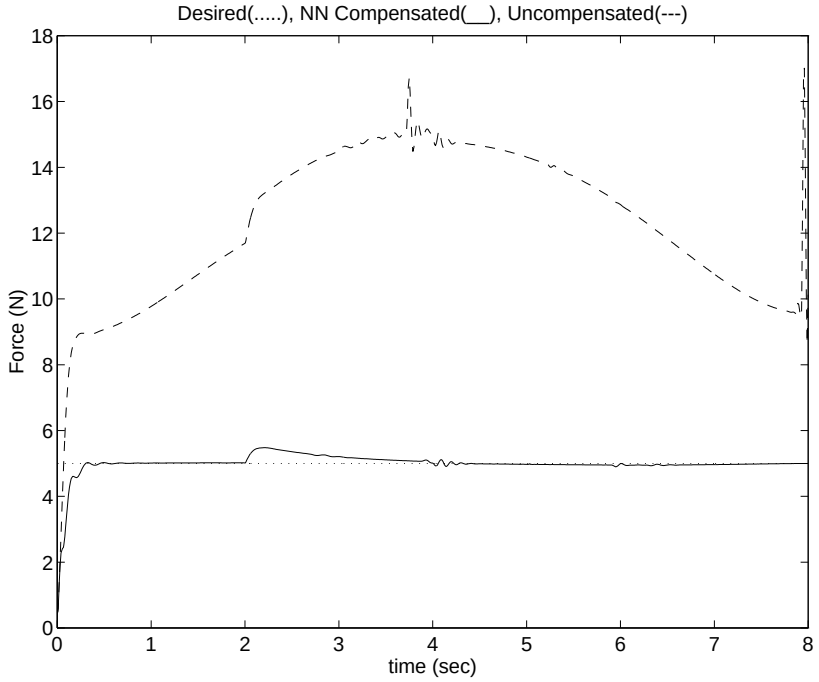


Figure 7.16: Force Tracking Circular Trajectory with Environment Stiffness Identification Method

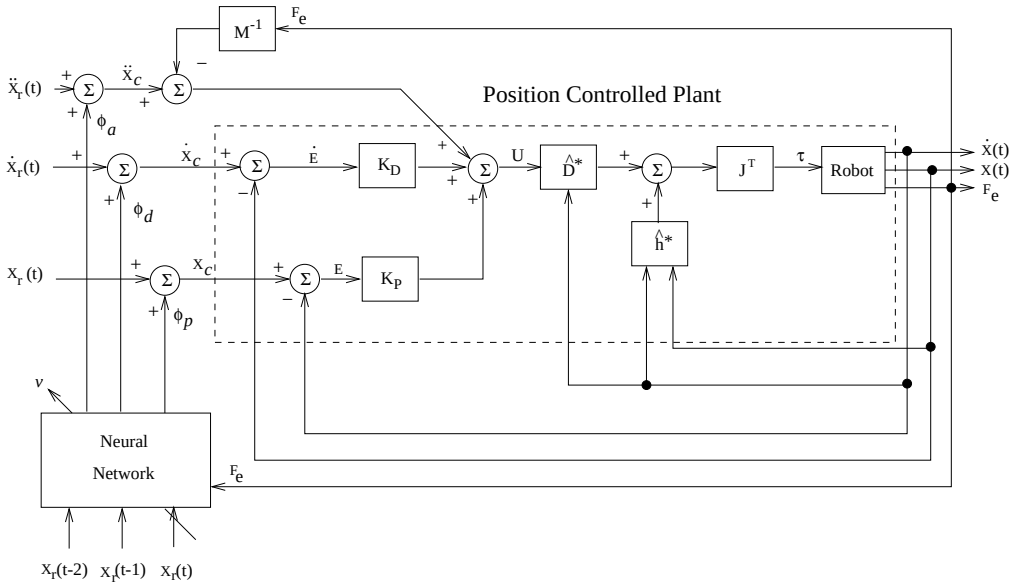


Figure 7.17: The proposed Position-Based NN Force Control Structure

Combining (7.44)(7.46) with (7.1) yields the corresponding closed loop error system as

$$\boldsymbol{M\ddot{E} + B\dot{E} + KE - F_e = M(\hat{D^*}^{-1}(\Delta D^* \ddot{X} + \Delta h^* + F_e + F_f^*) - \Psi)} \quad (7.47)$$

where $\boldsymbol{\Psi = \phi_a + K_D\phi_d + K_P\phi_p}$. In view of (7.13), the target impedance

$$\boldsymbol{F_e = M\ddot{E} + B\dot{E} + KE} \quad (7.48)$$

can be realized if $\boldsymbol{\Psi}$ is properly defined. Thus the NN controller design objective is to generate $\boldsymbol{\phi_a, \phi_d, \phi_p}$ such that $\boldsymbol{\Psi}$ satisfies $\boldsymbol{\hat{D^*}^{-1}(\Delta D^* \ddot{X} + \Delta h^* + F_e + F_f^*) = 0}$. We therefore propose here an NN training signal $\boldsymbol{v}$ to be

$$\boldsymbol{v = M\ddot{E} + B\dot{E} + KE - F_e} \quad (7.49)$$

It is clear that the ideal value of $\boldsymbol{\Psi}$ at $\boldsymbol{v = 0}$ is

$$\boldsymbol{\Psi = \hat{D^*}^{-1}(\Delta D^* \ddot{X} + \Delta h^* + F_e + F_f^*)} \quad (7.50)$$

Then, minimizing $\boldsymbol{v}$ allows us to achieve the ideal force control objective (7.13).

7.6.2.1 Simulation Results

For the NN controller, we have chosen six hidden neurons $(\boldsymbol{n_H = 6})$ and $\boldsymbol{\alpha = 0.9}$. The other parameters are $\boldsymbol{n_I = 12, n = 9}$ and $\boldsymbol{w_T = 161}$. The performances of the proposed schemes are tested by tracking two different tasks : one is flat sine wave tracking(called Task 1) and another is circular path tracking(called Task 2) as shown in Figure 7.6 and 7, respectively. The desired force $\boldsymbol{F_d}$ for both

Table 7.1: Task 1 (M = I)

K_e	K_D	K_P	η
1000	$diag[70, 20, 20]$	$diag[100, 100, 100]$	0.0001
10000	$diag[200, 20, 20]$	$diag[400, 100, 100]$	0.0001
50000	$diag[450, 20, 20]$	$diag[900, 100, 100]$	0.0001

Table 7.2: Task 2 (M = I)

K_e	K_D	K_P	η
1000	$diag[60, 60, 60]$	$diag[100, 100, 100]$	0.0001
5000	$diag[60, 60, 60]$	$diag[100, 100, 100]$	0.00005
10000	$diag[60, 60, 60]$	$diag[100, 100, 100]$	0.00003

tasks are $5N$. The reference trajectory X_r are approximately designed based on F_d and the environment stiffness K_e.

The control parameters for these two tasks are given in Tables 7.1 and 2. The results show that the controller converges less than 0.5 secs for all cases. Thus the control schemes is very effective for on-line application.

It is shown by simulation in Figure 7.18 and 19 that the NN controller performed well in compensating the uncertainties in the robot dynamics. The convergence rates are under one half second. Thus the proposed NN controllers are feasible for on-line robot control. Since the impedance law is implemented outside the position control loop it is simple to modify a position controlled robot system to perform force control application.

It is desirable to develop a new NN force control algorithm that is robust in the

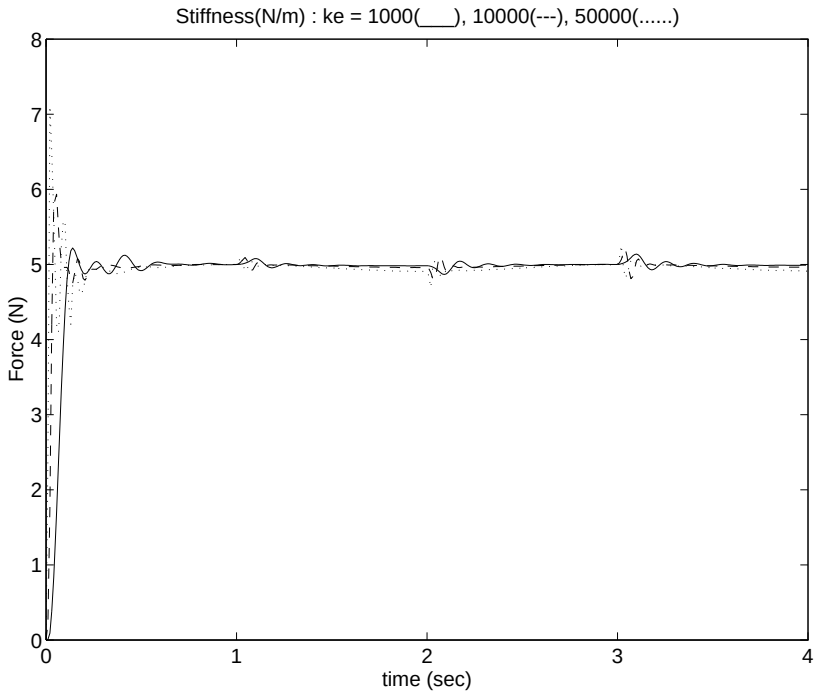


Figure 7.18: Task 1 : Force Tracking under different environment stiffnesses

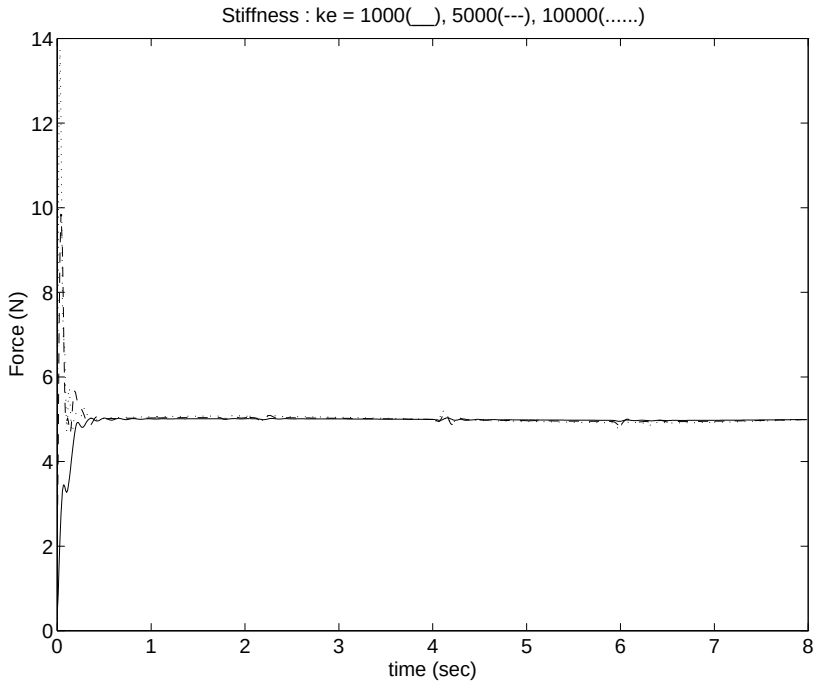


Figure 7.19: Task 2 : Force Tracking under different environment stiffnesses

presence of the uncertainties in both robot dynamics and in environment stiffness. Some new results are presented in the next nonlinear impedance control section.

7.7 Robustness Analysis due to Sensor Noises

7.7.1 Force sensor noise

It is important to examine the practical question as to how robust the proposed result (7.29) is when force sensor noise is present. Because the force measurement is usually noisier than position measurement.

Let $\boldsymbol{n_f}$ be the zero mean random force sensor noise. Adding $\boldsymbol{n_f}$ to (7.28) yields

$$\boldsymbol{\Delta x'_r} = \boldsymbol{f_d}\frac{\boldsymbol{k(x - x_e) + f_e + n_f}}{\boldsymbol{k(f_e + n_f)}} \tag{7.51}$$

Calculating the ratio $\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}}$ yields

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = \frac{\boldsymbol{f_e + n_f}}{\boldsymbol{f_e}}\left(\frac{\boldsymbol{k(x - x_e) + f_e}}{\boldsymbol{k(x - x_e) + f_e + n_f}}\right) \tag{7.52}$$

Substituting the relationship $\boldsymbol{x - x_e} = \frac{\boldsymbol{f_e}}{\boldsymbol{k_e}}$ in (7.24) into (7.52) yields

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = \frac{\boldsymbol{f_e + n_f}}{\boldsymbol{f_e}}\left(\frac{\boldsymbol{1}}{\boldsymbol{1 + \frac{n_f}{\frac{kf_e}{k_e} + f_e}}}\right) \tag{7.53}$$

For simplicity, let $\boldsymbol{k'} = \frac{\boldsymbol{k}}{\boldsymbol{k_e}}$ and $\boldsymbol{f'} = \frac{\boldsymbol{n_f}}{\boldsymbol{f_e}}$. We can show that (7.53) becomes

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = \frac{\boldsymbol{k'f'}}{\boldsymbol{k' + f' + 1}} + \boldsymbol{1} \tag{7.54}$$

This result exhibits the difference between $\boldsymbol{\Delta x_r}$ and $\boldsymbol{\Delta x'_r}$ due to force measurement noise $\boldsymbol{n_f}$. The difference depends on the ratios $\boldsymbol{k'}$ and $\boldsymbol{f'}$. For most practical cases

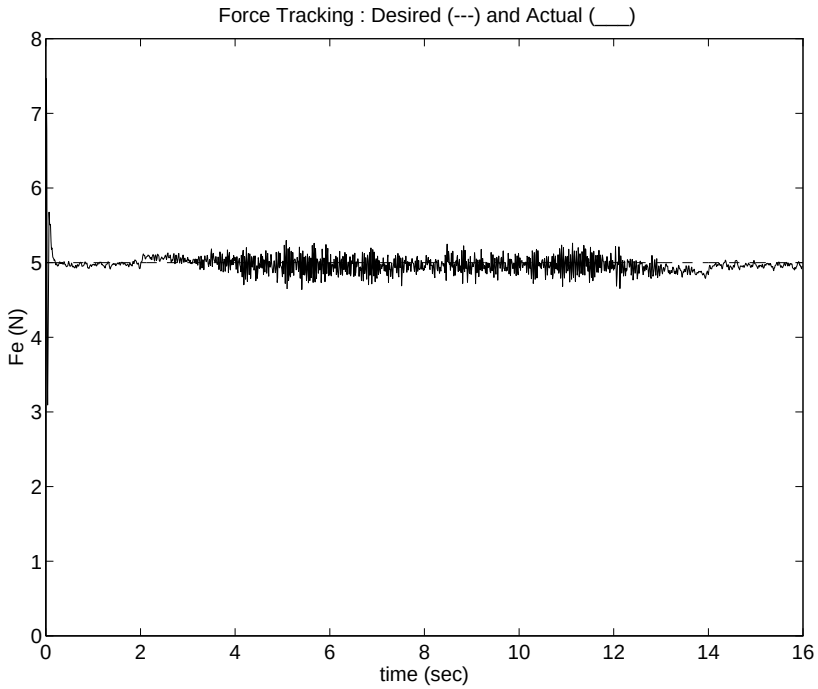


Figure 7.20: Task 2 : Force Tracking of Circular Trajectory of PBNNIC under $\pm 5\%$ Force Sensor Noise

where the noise $\boldsymbol{n_f}$ is small and environment is sufficiently stiff, both $\boldsymbol{k'}$ and $\boldsymbol{f'}$ are small so that $\boldsymbol{\Delta x'_r \approx \Delta x_r}$ It also shows that $\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} \rightarrow \boldsymbol{1}$ as $\boldsymbol{k_e \rightarrow \infty}$ (a rigid environment). So in the case of $\boldsymbol{k_e >> k}$, noise $\boldsymbol{n_f}$ has very little effect on the accuracy of $\boldsymbol{\Delta x'_r}$, and force tracking is sufficiently accurate. In general we can calculate the ratio $\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}}$ for a given $\boldsymbol{k_e}$ and $\boldsymbol{n_f}$. For example, if $\boldsymbol{k_e = 1000, k = 100}$ and force sensor noise is limited to $\boldsymbol{\pm 10\%}$ of the force measurement, then $\boldsymbol{k' = 0.1}$ and $\boldsymbol{f' = \pm 0.1}$, and the ratio $\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = \boldsymbol{1.00833 \sim 0.99}$. This represents an error of less than $\boldsymbol{1\%}$ in $\boldsymbol{\Delta x'_r}$.

Figure 7.20 shows the force tracking of Position based neural network force control scheme under $\pm 5\%$ force sensor noise.

7.7.2 Environment Position inaccuracy

The environment position is another form of uncertainty that causes poor force tracking. Even though with the help of vision system the position error can be minimized the uncertainty still remains. We like to see the effect of this uncertainty on performance.

Let $\boldsymbol{\delta x_e}$ be the inaccurate measurement of the environment position. Adding $\boldsymbol{\delta x_e}$ to (7.28) yields

$$\boldsymbol{\Delta x'_r} = \boldsymbol{f_d}\frac{\boldsymbol{k(x + \delta x_e - x_e) + f_e}}{\boldsymbol{k f_e}} \tag{7.55}$$

Calculating the ratio $\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}}$ yields

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = (\frac{\boldsymbol{k(x - x_e) + f_e}}{\boldsymbol{k(x + \delta x_e - x_e) + f_e}}) \tag{7.56}$$

Dividing the relationship by $\boldsymbol{k(x - x_e) + f_e}$ yields

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = \frac{\mathbf{1}}{\mathbf{1} + \frac{\boldsymbol{k k_e \delta x_e}}{\boldsymbol{(k+k_e) f_e}}} \tag{7.57}$$

For simplicity, let $\boldsymbol{k_{eff}} = \frac{\boldsymbol{k k_e}}{\boldsymbol{k+k_e}}$. We can show that (7.57) becomes

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} = \frac{\mathbf{1}}{\mathbf{1} + \frac{\boldsymbol{k_{eff} \delta x_e}}{\boldsymbol{f_e}}} \tag{7.58}$$

If the environment is stiff such that $\boldsymbol{k_e >> k}$ then $\boldsymbol{k_{eff} \approx k}$.

$$\frac{\boldsymbol{\Delta x_r}}{\boldsymbol{\Delta x'_r}} \cong \frac{\mathbf{1}}{\mathbf{1} + \frac{\boldsymbol{k \delta x_e}}{\boldsymbol{f_e}}} \tag{7.59}$$

In order to minimize the position sensor inaccuracy the term $\frac{\boldsymbol{k \delta x_e}}{\boldsymbol{f_e}}$ is required to be close to zero. For example, let the reasonable position measurement error be

0.5mm. The desired force is $5\boldsymbol{N}$ and $\boldsymbol{k = 500}$. Then $\frac{k\delta x_e}{f_e} = \frac{500*0.0005}{5} = \mathbf{0.05}$ and $\frac{\Delta x_r}{\Delta x_r'} = \mathbf{0.9524}$ which is about $\mathbf{5\%}$ error in force which is less than $\mathbf{1\%}$ error. Therefore, the performance of the proposed algorithm is more sensitive to the position measurement inaccuracy than to the force sensor noise.

7.8 Comparison Studies between Torque-based and Position Based NN Impedance Control

The controller gains are selected as $\boldsymbol{K_D = diag[300, 20, 20]}$ and $\boldsymbol{K_P = diag[900, 900, 900]}$ for Task 1, and $\boldsymbol{K_D = diag[200, 200, 200]}$ and $\boldsymbol{K_P = diag[500, 500, 500]}$ for Task 2 which give critically damped or over-damped motion s at the end-effector. The update rate $\boldsymbol{\eta = 0.00008}$ for PBNNIC and $\boldsymbol{\eta = 0.08}$ for TBNNIC is used. Sample tracking results for Task 1 are plotted in Figures 10 and 11. The simulation data showed that the rate of convergence is very fast with a convergence time less than 0.5 second. As a base line comparison, we also plotted the performance of the uncompensated system under same condition in Figure 7.22 and 23. It is seen that the force and position tracking errors are very large, and improvement of using NN is clearly demonstrated. We also see that the performance of the PBNNIC is slightly better than that of the TBNNIC.

Next task is that the robot is required to move circular trajectory with continuous environment stiffness profile shown in Figure 7.24. Since the environment is tilted by

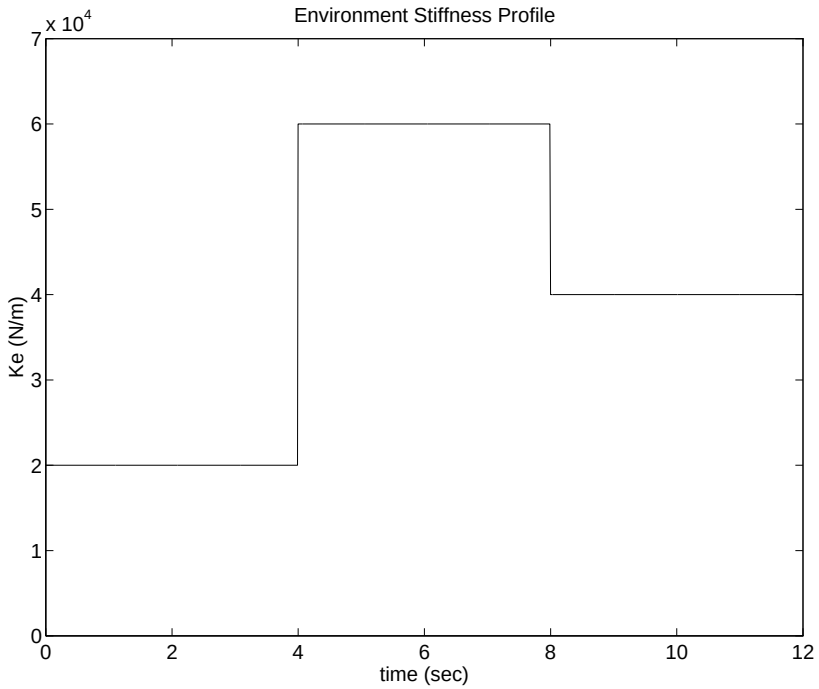


Figure 7.21: Time-Varying Discontinuous Environment Stiffness Profile I

45 degrees the total force normal to the surface has three components: $\boldsymbol{f}_{e_x}, \boldsymbol{f}_{e_y}, \boldsymbol{f}_{e_z}$ in $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ direction. The update rate $\boldsymbol{\eta}$ for this task is 0.05 for TBNNIC and 0.0001 for PBNNIC. The force tracking response is plotted in Figure 7.25. The force tracking response of the uncompensated control system is not plotted because the error is too large.

7.8.1 Discussion

Two new robust neural network impedance control schemes of robot manipulator are presented in this chapter. One is the torque-based NN impedance control and another is the position-based NN impedance control. The NN compensators serve

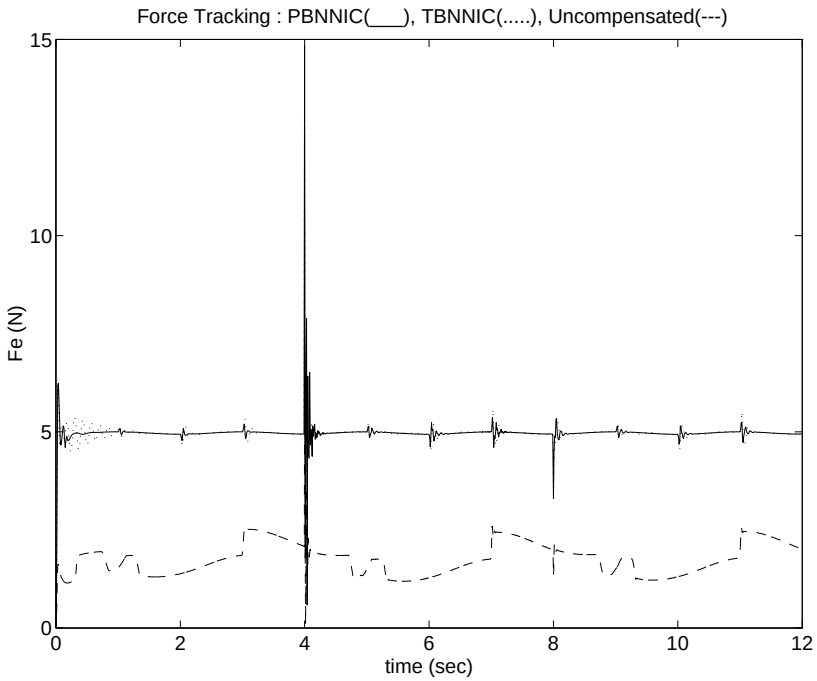


Figure 7.22: Task 1 (Flat Wall) : Force Tracking

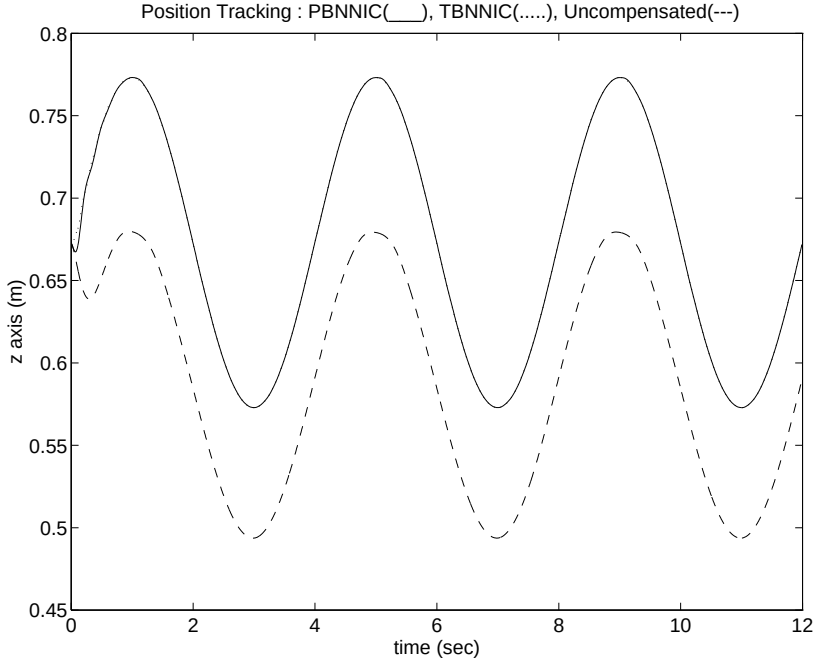


Figure 7.23: Task 1 (Flat Wall) : Position Tracking

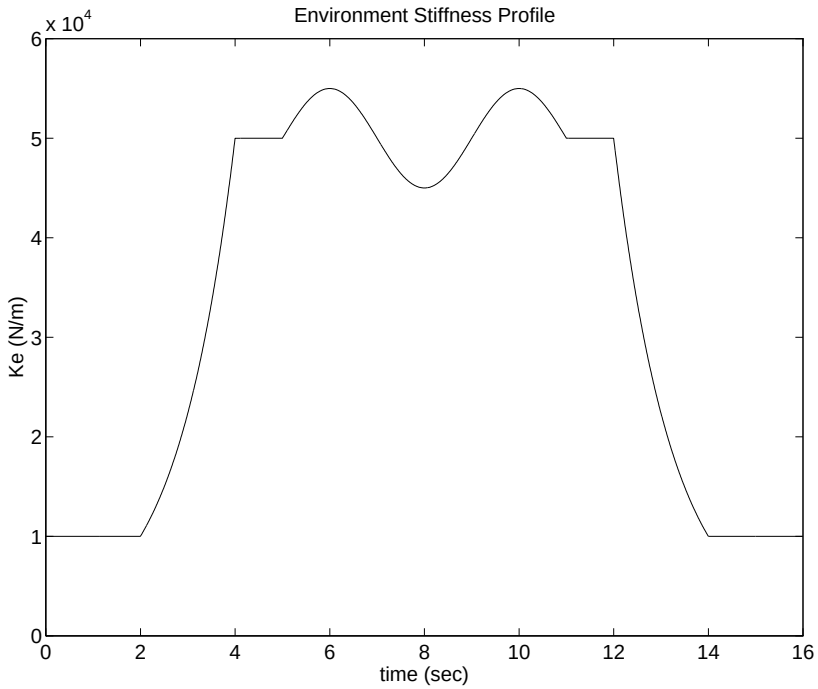


Figure 7.24: Time-Varying Continuous Environment Stiffness Profile II

as auxiliary controllers for both the impedance control structures to counteract the uncertainties in the robot dynamics and disturbances occurring from environment during application. Also introduced is a simple technique to design the reference trajectory using the sensed contact force instead of environment stiffness to compensate for the uncertainties in environment stiffness. A unified training signal for both torque-based and position based impedance control schemes is developed for training the back-propagation algorithm, and time delayed reference trajectories are used as inputs to the NN. Also introduced is a simple technique to design the reference trajectory using the sensed contact force instead of environment stiffness. Simulations show that system performances under these conditions are excellent and the conver-

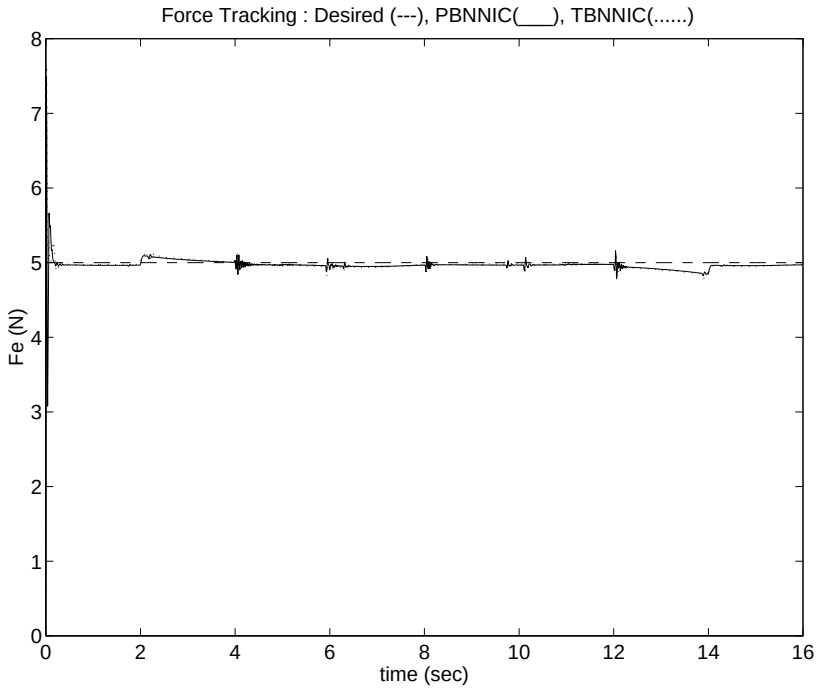


Figure 7.25: Task 2 (Circular Trajectory Tracking) : Force Tracking

gence rates are under one second. Thus the proposed control technique is feasible for on-line robot control application. Simulations based on a three link robot show that system performances of both NN impedance control schemes under these conditions are excellent and the convergence rates are under one second. Thus the proposed NN controllers are feasible for on-line robot control. The performance of PBNNIC is slightly better than that of TBNNIC with paying more weights in NN. In addition, the PBNNIC has a structural advantage so that the implementation of the robust force control from position-controlled robot system can be done without modifying inside control structure.

7.9 Non-linear Impedance Control Approach

7.9.1 Control Law

In this section, we present an NN controller for force tracking impedance control as shown in Figure 7.26. For simplicity, let $\boldsymbol{m}, \boldsymbol{b}, \boldsymbol{k}, \boldsymbol{f_e}, \boldsymbol{\delta_m}, \boldsymbol{\psi}, \boldsymbol{e}$ be the elements of the diagonal matrices $\boldsymbol{M}, \boldsymbol{B}, \boldsymbol{K}$ and vectors $\boldsymbol{F_e}, \boldsymbol{\Delta_M}, \boldsymbol{\Psi}, \boldsymbol{E}$, respectively. Then, each element in equation (7.31) becomes

$$\ddot{\boldsymbol{e}} + \frac{\mathbf{1}}{\boldsymbol{m}}(\boldsymbol{b}\dot{\boldsymbol{e}} + \boldsymbol{k}\boldsymbol{e} - \boldsymbol{f_e}) = \boldsymbol{\delta_m} - \boldsymbol{\psi} \tag{7.60}$$

Let $\boldsymbol{x_r}$ be denoted as $\boldsymbol{x_r} = \boldsymbol{x_e} + \boldsymbol{\Delta x_r}$ where $\boldsymbol{x_e}$ is the environment position. The elements in $\boldsymbol{x_r}$ and $\boldsymbol{x_e}$ satisfy the relationships $\boldsymbol{x_r} = \boldsymbol{x_e} + \boldsymbol{\Delta x_r}$, $\dot{\boldsymbol{x}}_r = \dot{\boldsymbol{x}}_e + \boldsymbol{\Delta}\dot{\boldsymbol{x}}_r$ and $\ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}_e + \boldsymbol{\Delta}\ddot{\boldsymbol{x}}_r$. If we substitute these relationships into (7.60) and define $\boldsymbol{\varepsilon} = \boldsymbol{x_e} - \boldsymbol{x}$, then

$$\ddot{\boldsymbol{\varepsilon}} + \frac{\mathbf{1}}{\boldsymbol{m}}[\boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{k}\boldsymbol{\varepsilon}] + \boldsymbol{\Delta}\ddot{\boldsymbol{x}}_r + \frac{\mathbf{1}}{\boldsymbol{m}}[\boldsymbol{b}\boldsymbol{\Delta}\dot{\boldsymbol{x}}_r + \boldsymbol{k}\boldsymbol{\Delta x_r} - \boldsymbol{f_e}] = \boldsymbol{\delta_m} - \boldsymbol{\psi} \tag{7.61}$$

This equation indicates that the reference trajectory for the robot control system is $\boldsymbol{x_e}$(see Figure 7.26) instead of $\boldsymbol{x_r}$ used in the traditional design.

By defining $\boldsymbol{\delta_f} = \boldsymbol{\Delta}\ddot{\boldsymbol{x}}_r + \frac{\mathbf{1}}{\boldsymbol{m}}\boldsymbol{b}\boldsymbol{\Delta}\dot{\boldsymbol{x}}_r$ (7.61) can be simplified to be

$$\ddot{\boldsymbol{\varepsilon}} + \frac{\mathbf{1}}{\boldsymbol{m}}[\boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{k}\boldsymbol{\varepsilon}] + \frac{\mathbf{1}}{\boldsymbol{m}}[\boldsymbol{k}\boldsymbol{\Delta x_r} - \boldsymbol{f_e}] = \boldsymbol{\delta_m} - \boldsymbol{\delta_f} - \boldsymbol{\psi} \tag{7.62}$$

It is well known that $\boldsymbol{\Delta x_r}$ satisfies the relationship $\boldsymbol{\Delta x_r} = \frac{\boldsymbol{f_d}}{\boldsymbol{k_{eff}}}$ where $\boldsymbol{k_{eff}} = \frac{\boldsymbol{k}\boldsymbol{k_e}}{\boldsymbol{k}+\boldsymbol{k_e}}$ where $\boldsymbol{k_e}$ is the environment stiffness [2]. Substituting $\boldsymbol{\Delta x_r}$ into (7.62) yields

$$\ddot{\boldsymbol{\varepsilon}} + \frac{\mathbf{1}}{\boldsymbol{m}}(\boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{k}\boldsymbol{\varepsilon} + \boldsymbol{f_d}\frac{\boldsymbol{k}}{\boldsymbol{k_e}} + \boldsymbol{f_d} - \boldsymbol{f_e}) = \boldsymbol{\delta_m} - \boldsymbol{\delta_f} - \boldsymbol{\psi} \tag{7.63}$$

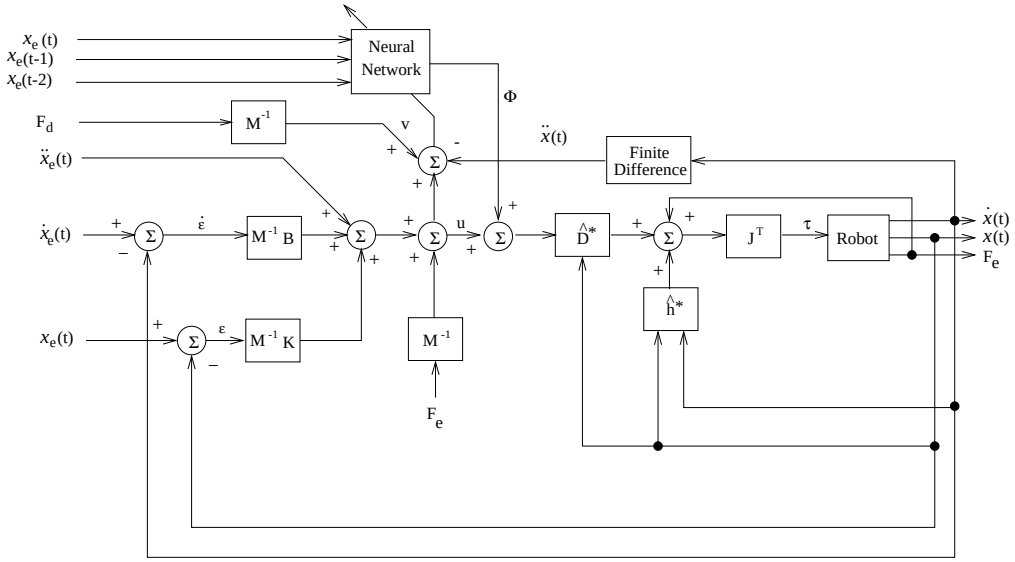


Figure 7.26: NN Force Control Structure

In this section we assume that $\boldsymbol{k_e}$ is unknown. As a solution, we estimate $\boldsymbol{k_e}$ based on the measured force $\boldsymbol{f_e}$ and the relationship $\boldsymbol{f_e = k_e(x - x_e) = -k_e\varepsilon}$. So replacing $\boldsymbol{k_e}$ by $-\frac{\boldsymbol{f_e}}{\boldsymbol{\varepsilon}}$ in (7.63) results in

$$\ddot{\boldsymbol{\varepsilon}} + \frac{\mathbf{1}}{\boldsymbol{m}}(\boldsymbol{b\dot{\varepsilon}} + \boldsymbol{k\varepsilon}(\mathbf{1} - \frac{\boldsymbol{f_d}}{\boldsymbol{f_e}}) + \boldsymbol{f_d} - \boldsymbol{f_e}) = \boldsymbol{\delta_m} - \boldsymbol{\delta_f} - \boldsymbol{\psi} \tag{7.64}$$

The control objective is to require $\boldsymbol{\delta_m - \delta_f - \psi = 0}$ such that the ideal result

$$\boldsymbol{m\ddot{\varepsilon} + b\dot{\varepsilon} + k\varepsilon}(\mathbf{1} - \frac{\boldsymbol{f_d}}{\boldsymbol{f_e}}) + \boldsymbol{f_d} - \boldsymbol{f_e} = \mathbf{0} \tag{7.65}$$

is obtained. We noted that (7.66) satisfies the target impedance relationship (7.13) in which $\boldsymbol{f_e}$ tracks $\boldsymbol{f_d}$ in steady state. The new results (7.66) is more useful in that it allows $\boldsymbol{f_d}$ to be specified directly and no knowledge of $\boldsymbol{k_e}$ is required.

In order for the NN output $\boldsymbol{\psi}$ to cancel out the uncertainties $\boldsymbol{\delta_m}$ and $\boldsymbol{\delta_f}$, we choose

a training signal $\boldsymbol{v}$ for the NN controller as

$$\boldsymbol{v} = \boldsymbol{m}\ddot{\boldsymbol{\varepsilon}} + \boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{k}\boldsymbol{\varepsilon}(\mathbf{1} - \frac{\boldsymbol{f_d}}{\boldsymbol{f_e}}) + \boldsymbol{f_d} - \boldsymbol{f_e} \tag{7.66}$$

We see that by minimizing $\boldsymbol{v}$ to zero, the NN will force $\boldsymbol{\psi}$ to cancel $(\boldsymbol{\delta_m} - \boldsymbol{\delta_f})$ as indicated in (7.64)(7.66). Note that the training signal $\boldsymbol{v}$ is not valid when $\boldsymbol{f_e} = \mathbf{0}$ or $\boldsymbol{f_e} = \boldsymbol{f_d} = \mathbf{0}$. So for free space motion control when $\boldsymbol{f_e} = \mathbf{0}$, we will use a different training signal in the form of

$$\boldsymbol{v} = \boldsymbol{m}\ddot{\boldsymbol{\varepsilon}} + \boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{k}\boldsymbol{\varepsilon} + \boldsymbol{f_d} \tag{7.67}$$

Thus combination of (7.66) and (7.67) allows us to train the NN for both contact and non-contact robot motion control.

It is important to note that in the contact phase, the steady state of (7.66) is

$$\begin{aligned} \boldsymbol{k}\boldsymbol{\varepsilon}(\mathbf{1} - \frac{\boldsymbol{f_d}}{\boldsymbol{f_e}}) + \boldsymbol{f_d} - \boldsymbol{f_e} &= \mathbf{0} \\ \boldsymbol{or}\ \boldsymbol{f_e^2} - (\boldsymbol{f_d} + \boldsymbol{k}\boldsymbol{\varepsilon})\boldsymbol{f_e} - \boldsymbol{k}\boldsymbol{\varepsilon}\boldsymbol{f_d} &= \mathbf{0} \end{aligned} \tag{7.68}$$

which is a second order quadratic equation of $\boldsymbol{f_e}$. Solving for $\boldsymbol{f_e}$ leads to two possible solutions

$$\boldsymbol{f_e} = \boldsymbol{f_d}, \qquad \boldsymbol{f_e} = \boldsymbol{k}\boldsymbol{\varepsilon} \tag{7.69}$$

However, $\boldsymbol{f_e} = -\boldsymbol{k_e}\boldsymbol{\varepsilon}$ by definition, so equating this solution to that of $\boldsymbol{f_e} = \boldsymbol{k}\boldsymbol{\varepsilon}$ requires that $-\boldsymbol{k_e}\boldsymbol{\varepsilon} = \boldsymbol{k}\boldsymbol{\varepsilon}$ or $\boldsymbol{k_e} = -\boldsymbol{k}$ for all $\boldsymbol{\varepsilon} \neq \mathbf{0}$. Since this is physically impossible, the desired solution $\boldsymbol{f_e} = \boldsymbol{f_d}$ is then guaranteed. It is also necessary to investigate force solutions when the environment location $\boldsymbol{x_e}$ is not exactly known.

7.9.2 Robustness Analysis to Inexact Environment Location

Here we investigate the robustness of the proposed controller when the environment location $\boldsymbol{x_e}$ is not exactly known. This is an important concern in practice because there are always uncertainties about the environment locations. Our interest is to know how these uncertainties effect the behavior of the controller performance. We proceed our analysis as follows. Let the reference input to the control system be $\boldsymbol{x'_e}$ where $\boldsymbol{x'_e = x_e + \Delta x_e}$, $\boldsymbol{\Delta x_e}$ being uncertainty of $\boldsymbol{x_e}$. Define $\boldsymbol{\varepsilon' = \varepsilon + \Delta x_e}$. The corresponding steady state of (7.66) and (7.67) becomes

$$\boldsymbol{k\varepsilon' + f_d = 0} \qquad \boldsymbol{free\ space} \tag{7.70}$$

$$\boldsymbol{f_e^2 - (f_d + k\varepsilon')f_e - k\varepsilon' f_d = 0} \qquad \boldsymbol{contact\ space} \tag{7.71}$$

We examine two cases of $\boldsymbol{\Delta x_e}$

i) $\boldsymbol{\Delta x_e < 0}$ case : Off the surface

For convenience of analysis, we assume $\boldsymbol{x, x_e > 0}$ so that $\boldsymbol{\Delta x_e < 0}$ implies that $\boldsymbol{x'_e}$ is off the surface of the environment defined by $\boldsymbol{x_e (x'_e < x_e)}$. Suppose the end effector is initially in free space, i.e. $\boldsymbol{x < x_e}$. From (7.70), we get

$$\boldsymbol{k(x_e + \Delta x_e - x) + f_d = 0}$$

$$\boldsymbol{or \quad x = x_e + \Delta x_e + \frac{f_d}{k}} \tag{7.72}$$

Two possibilities exist:

a) $\boldsymbol{x < x_e}$ if $\boldsymbol{\Delta x_e + \frac{f_d}{k} < 0}$ (or $\boldsymbol{\Delta x_e < -\frac{f_d}{k}}$). This case means that the steady state

$\boldsymbol{x}$ remains outside the environment, so no contact between robot and environment would ever occur as it would be when $\boldsymbol{\Delta x_e = 0}$. The free space controller using $\boldsymbol{x'_e}$ instead of $\boldsymbol{x_e}$ would fail to function properly in this case.

b) $\boldsymbol{x > x_e}$ if $\boldsymbol{\Delta x_e + \frac{f_d}{k} > 0}$ (or $\boldsymbol{\Delta x_e > -\frac{f_d}{k}}$). This says that contact would occur if $\boldsymbol{0 > \Delta x_e > -\frac{f_d}{k}}$. This condition can usually be met in practice since $|\boldsymbol{\Delta x_e}|$ would be generally smaller than $\boldsymbol{\frac{f_d}{k}}$.

Once the contact is made and the robot is under contact space control, two steady state $\boldsymbol{f_e}$ solutions exist : $\boldsymbol{f_e = f_d}$ and $\boldsymbol{f_e = k\varepsilon'}$. It can be shown that $\boldsymbol{f_e = f_d}$ is the only valid solution in that the solution $\boldsymbol{f_e = k\varepsilon'}$ requires $\boldsymbol{k_e = -k}$ which is not possible. Thus desired force tracking can be achieved.

ii) $\boldsymbol{\Delta x_e > 0}$ case : Inside the surface

Starting from free space $\boldsymbol{x < x_e}$, the robot end effector steady state position of (7.70) is $\boldsymbol{x > x_e}$ because $\boldsymbol{\Delta x_e + \frac{f_d}{k} > 0}$. So contact is always made if $\boldsymbol{x'_e > x_e}$. Again, during contact, $\boldsymbol{f_e}$ has two values $\boldsymbol{f_e = f_d}$ or $\boldsymbol{f_e = k\varepsilon'}$. In this case however, both solutions can be valid depending on the magnitude of $\boldsymbol{\Delta x_e}$. Thus contact force tracking may be problematic. In next section, the boundary condition of the correct force tracking for this case will be found.

7.9.3 Boundary Conditions

From the nonlinear impedance function given in (7.64) the environment position uncertainty $\boldsymbol{\delta x_e}$ can be added as :

$$\ddot{\varepsilon}' + \frac{1}{m}(b\dot{\varepsilon}' + k\varepsilon'(1 - \frac{f_d}{f_e}) + f_d - f_e) = 0 \tag{7.73}$$

where $\varepsilon' = x_e - x + \delta_e$. Let ε' be a function of force f_e such that $\varepsilon' = -\frac{f_e}{k_e} + \delta x_e, \dot{\varepsilon}' = -\frac{\dot{f}_e}{k_e} + \delta \dot{x}_e, \ddot{\varepsilon}' = -\frac{\dot{f}_e}{k_e} + \delta \ddot{x}_e$. Substituting $\varepsilon', \dot{\varepsilon}', \ddot{\varepsilon}'$ into (7.73) yields

$$\ddot{f}_e + \frac{b}{m}\dot{f}_e + \frac{k}{m}f_e(1 - \frac{f_d}{f_e}) - \frac{kk_e}{m}\delta x_e(1 - \frac{f_d}{f_e}) + k_e\frac{(f_e - f_d)}{m} = 0 \tag{7.74}$$

Rearranging (7.74) in terms of force

$$\ddot{f}_e + \frac{b}{m}\dot{f}_e + \frac{(k + k_e)}{m}f_e + \frac{kk_e f_d \delta x_e}{m f_e} = \delta \ddot{x}_e + \frac{1}{m}(f_d + kk_e\delta x_e + f_d k_e + bk_e\delta \dot{x}_e) \tag{7.75}$$

For simplicity, let $\delta \ddot{x}_e = \delta \dot{x}_e = 0$, then (7.75) becomes

$$\ddot{f}_e + \frac{b}{m}\dot{f}_e + \frac{(k + k_e)}{m}f_e + \frac{kk_e f_d}{m f_e}\delta x_e = \frac{1}{m}(f_d + kk_e\delta x_e + f_d k_e) \tag{7.76}$$

We know that the nonlinear equation (7.76) have two solutions at equilibrium points such as $f_e = f_d, f_e = k\varepsilon' = k_{eff}\delta x_e$. Here we like to find the conditions when forces are two solutions.

The steady state representation of (7.76) can be shown by defining states $y_1 = f_e, y_2 = \dot{f}_e$ at an equilibrium point $f_e = f_d$ as

$$\dot{y} = Ay + Bu = \begin{bmatrix} 0 & 1 \\ -\frac{(k+k_e)}{m} + \frac{kk_e\delta x_e}{mf_d} & -\frac{b}{m} \end{bmatrix} y + \frac{1}{m}(f_d + kk_e\delta x_e + f_d k_e) \tag{7.77}$$

The characteristic equation is calculated by determining

$$\boldsymbol{det(sI - A) = s^2 + \frac{b}{m}s + \frac{k + k_e}{m} - \frac{kk_e\delta x_e}{mf_d} = 0} \tag{7.78}$$

The stability conditions are found as

$$\boldsymbol{0 < \frac{b}{m}, \; and \; \delta x_e < \frac{f_d}{k_{eff}}} \tag{7.79}$$

Since we choose $\boldsymbol{b, m > 0}$, $\boldsymbol{\frac{f_e^2}{f_d k_{eff}} > \delta x_e}$ is the only condition to be considered. Thus, at the desirable equilibrium point $\boldsymbol{f_e = f_d}$, the condition (7.79) becomes

$$\boldsymbol{\delta x_e < \frac{f_d}{k_{eff}} \;\; at \; f_e = f_d} \tag{7.80}$$

In the same manner, the characteristic equation evaluated at another equilibrium point $\boldsymbol{f_e = k_{eff}\delta x_e}$ which is not desirable is

$$\boldsymbol{det(sI - A) = s^2 + \frac{b}{m}s + \frac{k + k_e}{m} - \frac{(k + k_e)f_d}{mk_{eff}\delta x_e} = 0} \tag{7.81}$$

For stability, the condition of $\boldsymbol{\delta x_e}$

$$\boldsymbol{\frac{f_d}{k_{eff}} < \delta x_e \;\; at \; f_e = k_{eff}\delta x_e} \tag{7.82}$$

should hold when $\boldsymbol{b, m > 0}$.

Therefore, when the environment location uncertainty is specified inside the environment($\boldsymbol{0 < \delta x_e}$) the condition (7.80) should be satisfied in order to achieve the stable correct force control. Combining the results from the previous analysis [88] and the above results (7.80) the uncertainty location $\boldsymbol{\delta x_e}$ should be located within the bound

$$\boldsymbol{-\frac{f_d}{k} < \delta x_e < \frac{f_d}{k_{eff}}} \tag{7.83}$$

The left side of the bound is obtainable only when the bounds of δx_e are known priori because f_d, k are user specified variables while the right side of the bound is not available immediately since k_e is not known. However, when the real environment for the robot manipulator to be dealt with is very stiff so that the approximation $k_{eff} \approx k$ is acceptable. Then the bound becomes

$$-\frac{f_d}{k} < \delta x_e < \frac{f_d}{k} \tag{7.84}$$

which can be specified by the user. In practice, the position uncertainty δx_e can be minimized as small as possible (eg. $\delta x_e < 1mm$) so that the bound can be chosen sufficiently large by the user. The solution to this problem is to use the adaptive law to minimize the stiffness gain so that there is only one solution $f_e = f_d$ available. The proposed adaptive control will be presented in the next section.

7.9.4 Simulation Studies

Two comprehensive simulation studies of polishing process have been carried out using a three link rotary robot manipulator whose parameters are taken from the first three links of Puma 560 arm. One simulation study has been carried out when there are no uncertainties in order to investigate the effects on force tracking by the theoretical bounds. Another case is for uncertainties presence. The nominal system parameters are used as the basis in forming the robot model $\hat{D}(q)$ and $\hat{h}(q, \dot{q})$. In order to investigate the analysis shown in section 5, the robot is tested under the ideal condition without any uncertainty presence. Within each case, several environment

surface locations are tested to confirm the analytical bounds founded from stability analysis. The environment stiffness is 50000(N/m) which is fairly large. The PD controller gains are selected as $\boldsymbol{K_A = I}$, $\boldsymbol{K_D = diag[200, 20, 20]}$ and $\boldsymbol{K_P = diag[500, 100, 100]}$ The performances of the proposed schemes are tested by flat sine wave tracking in order to polish the surface as shown in Figure 2.

7.9.5 Performances without uncertainties

Two cases of environment surface location, outside surface and inside surface, were considered.

7.9.5.1 Off surface

This case is when the inaccurate estimation of environment surface location is outside the surface such that $\boldsymbol{x'_e < x_e}$. Several surface locations were tested and some of them were plotted in Figures 7.27 and 28. The critical bound found in our analysis was $\boldsymbol{-\frac{f_d}{k} = -\frac{10}{500} = -0.02m}$. When the uncertainty of environment surface location is $\boldsymbol{-20mm < \delta x_e}$, the robot does not make contact with the surface at all. Even though it satisfies the contact condition, for example $\boldsymbol{\delta x_e = -10mm > -20mm}$, the desired force tracking was not guaranteed and it bounces back and forth which is considered as an unstable condition. For those cases$\boldsymbol{(-8, -5, -1mm)}$ plotted in Figure 7.27, the force tracking performances are excellent as well as the contact was successful. We also note that the more accurate estimation is the faster

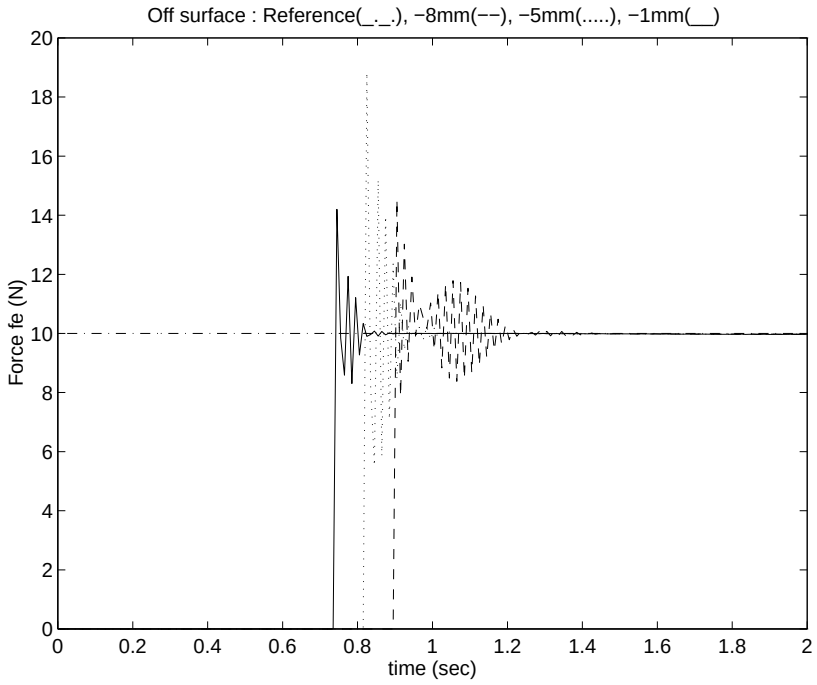


Figure 7.27: Force Tracking performances under no uncertainties when estimated environment surface locations $\boldsymbol{x}'_e$ are outside the surface

the contact occurs. Figure 7.28 shows the excellent position tracking also.

7.9.5.2 Inside surface

Another case is when the estimation of environment surface location is inaccurately estimated inside the surface so that $\boldsymbol{x}_e < \boldsymbol{x}'_e$. The performances are shown in Figures 7.29 and 30. When $\boldsymbol{\delta x_e = 10mm < \frac{f_d}{k_{eff}} = 25.2mm}$ the force tracking is very good. More oscillatory behavior can be observed as $\boldsymbol{\delta x_e}$ is close to the bound $\boldsymbol{25.2mm}$. Further than that, the interesting nonlinear behavior can be observed specially when the uncertainty $\boldsymbol{\delta x_e}$ exceed the critical bound $\frac{f_d}{k_{eff}}$. In Figure 7.29, when $\boldsymbol{\delta x_e = 30mm > \frac{f_d}{k_{eff}} = 25.2mm}$, the actual contact force $\boldsymbol{f_e}$ is rest at an

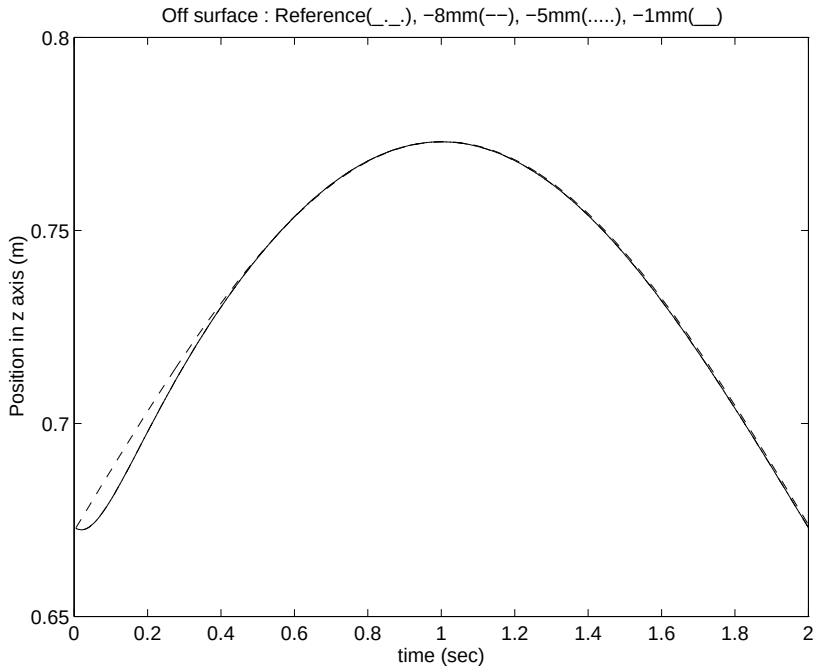


Figure 7.28: Position Tracking in z axis with different estimated environment surface locations of Fig. 7.28

equilibrium point $\boldsymbol{f_e = k_{eff}\delta x_e = 14.85(N)}$ which matches our analytical result. Note that the force initially tries to go to the equilibrium point $\boldsymbol{f_e = f_d}$ and then settles at another equilibrium point $\boldsymbol{f_e = k_{eff}x_e}$. When uncertainties are added, it is more difficult to satisfy the bound.

7.9.6 Performances with uncertainties

Model uncertainties included a 10 Kg mechanical tool attached to the third link, Coulomb friction and viscous friction forces $\boldsymbol{\tau_f(\dot{q})}$ added to each joint where $\boldsymbol{\tau_f(\dot{q}) = 0.5sgn(\dot{q}) + 0.8(\dot{q})}$.

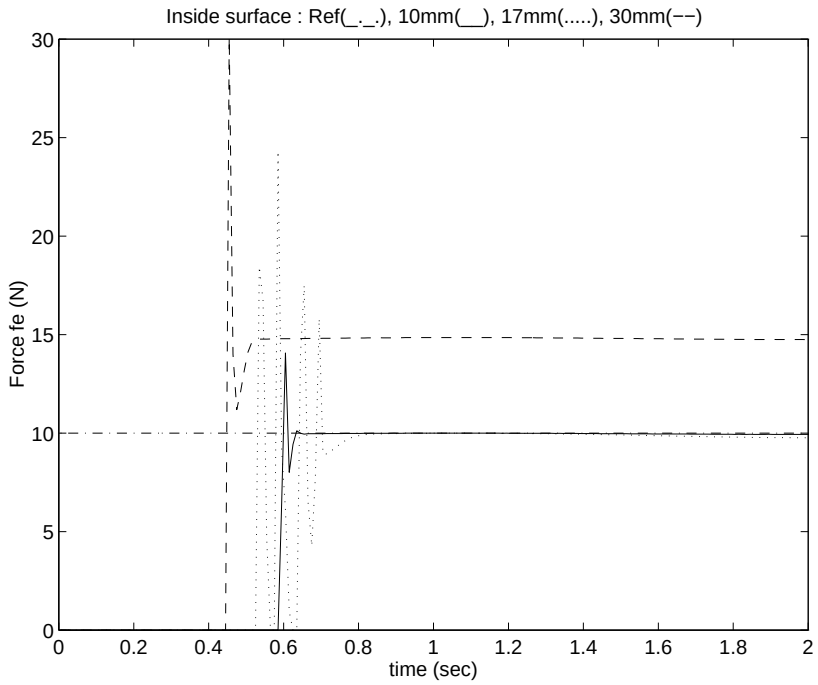


Figure 7.29: Force Tracking Performances under no uncertainties when estimated environment surface locations $\boldsymbol{x}'_e$ are inside the surface

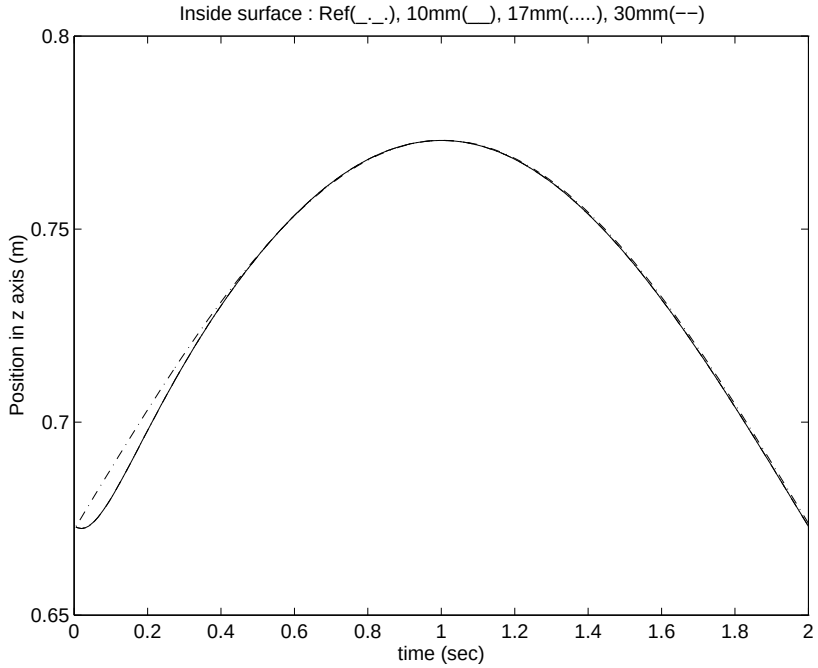


Figure 7.30: Position Tracking in z axis with different estimated environment surface locations of Fig. 7.30

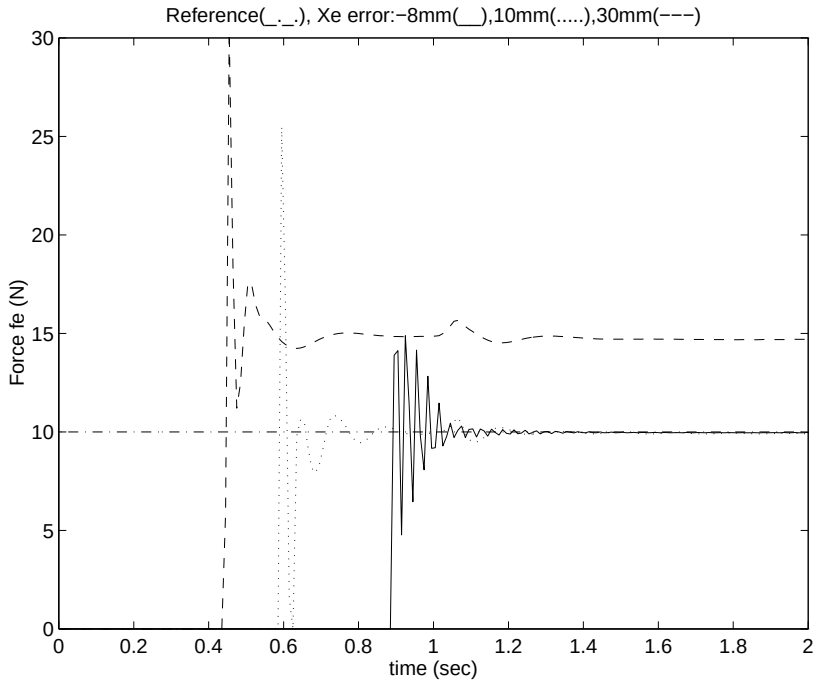


Figure 7.31: Force Tracking Performances under uncertainties when environment surface locations $\boldsymbol{x}_e'$ are estimated

7.9.6.1 Constant environment stiffness

The sample tracking performances for different environment surface location are shown in Figures 7.31 and 32. The optimized learning rates $\boldsymbol{\eta = 0.01, 0.022, 0.01}$ are used for $\boldsymbol{\delta x_e = -8mm, 10mm, 30mm}$, respectively. The force tracking performances within the bound are very good while those of uncompensated case are always unstable. The same observation is obtained when $\boldsymbol{\delta x_e}$ exceeds the bound. The corresponding good position tracking are shown in Figure 7.32. The neural controller enable the system to be stable by compensating for those uncertainties.

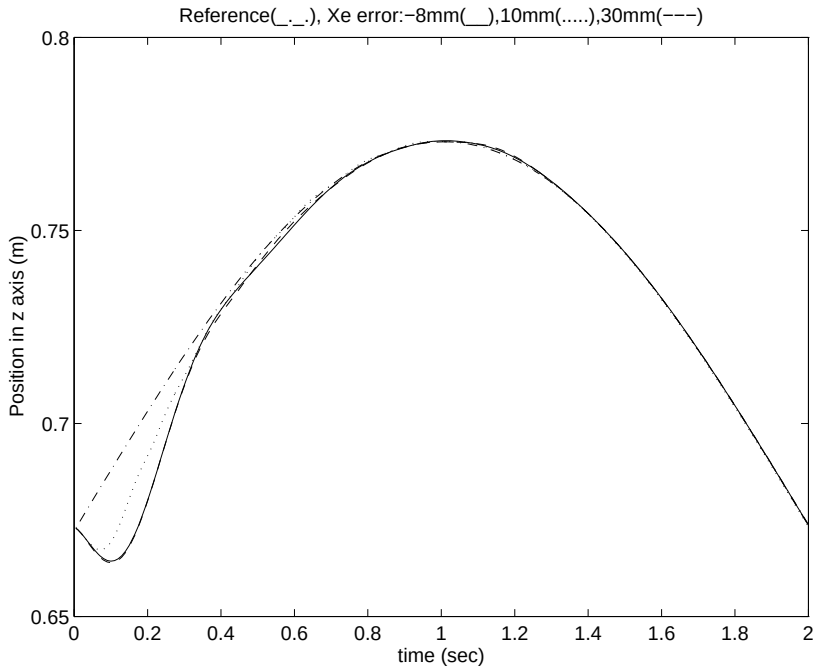


Figure 7.32: Position Tracking in z axis with different estimated environment surface locations of Fig. 7.31

7.9.6.2 Time-varying environment stiffness

In order to show the robustness to unknown environment stiffness of the proposed scheme, we tested the system performance for abruptly changing environment stiffnesses with the stiffness profile as

$$\boldsymbol{k_e} = \begin{cases} \mathbf{50000} & \mathbf{0} \leq \boldsymbol{t} < \mathbf{2} \\ \mathbf{100000} & \mathbf{2} \leq \boldsymbol{t} < \mathbf{4} \\ \mathbf{50000} & \mathbf{4} \leq \boldsymbol{t} < \mathbf{6} \\ \mathbf{100000} & \mathbf{6} \leq \boldsymbol{t} < \mathbf{8} \end{cases} \tag{7.85}$$

which is shown in Figure 7.33. The controller gains are selected as $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{400, 20, 20}]$ and $\boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{500, 100, 100}]$ which give more over-damped motions at the force

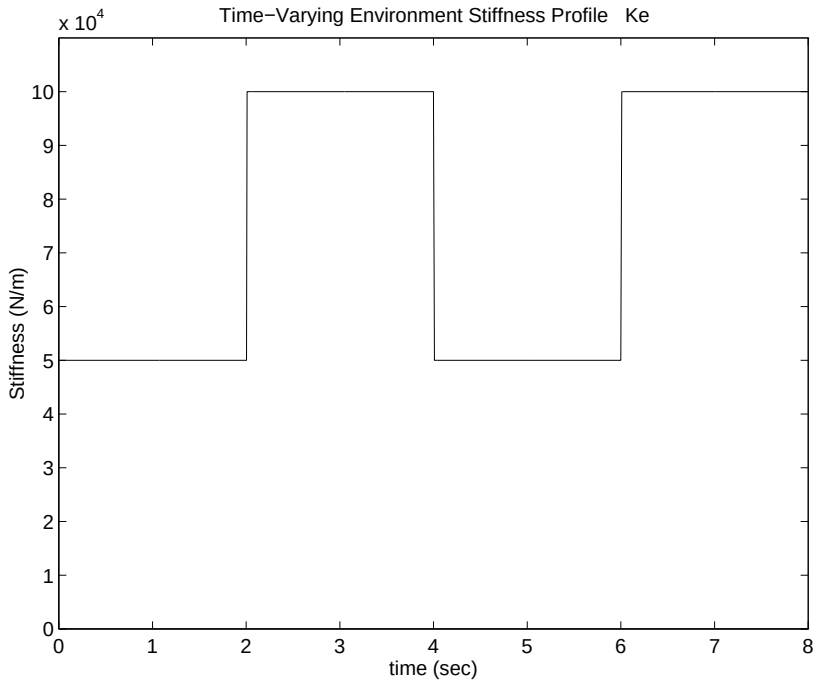


Figure 7.33: Environment Stiffness Profile

controlled direction than the previous case. The learning rate $\boldsymbol{\eta = 0.01}$ is used. At the same time, the desired force is also time-varying such as

$$\boldsymbol{f_d = 20 + 4sin(\frac{\pi t}{4})} \tag{7.86}$$

Sample tracking results are plotted in Figures 7.34 and 35. The simulation data showed that the force tracking results are very effective. Note that overshoots occur when the environment stiffness is changed abruptly. Figures 7.35 shows the reference trajectory and actual trajectory in $\boldsymbol{z}$ axis.

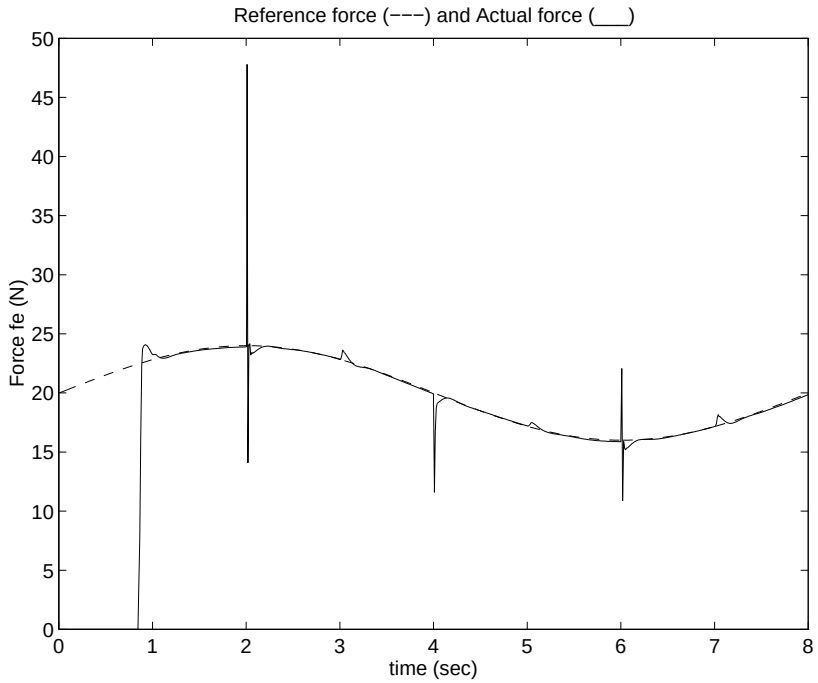


Figure 7.34: Time-varying force tracking under the stiffness of Fig. 7.33

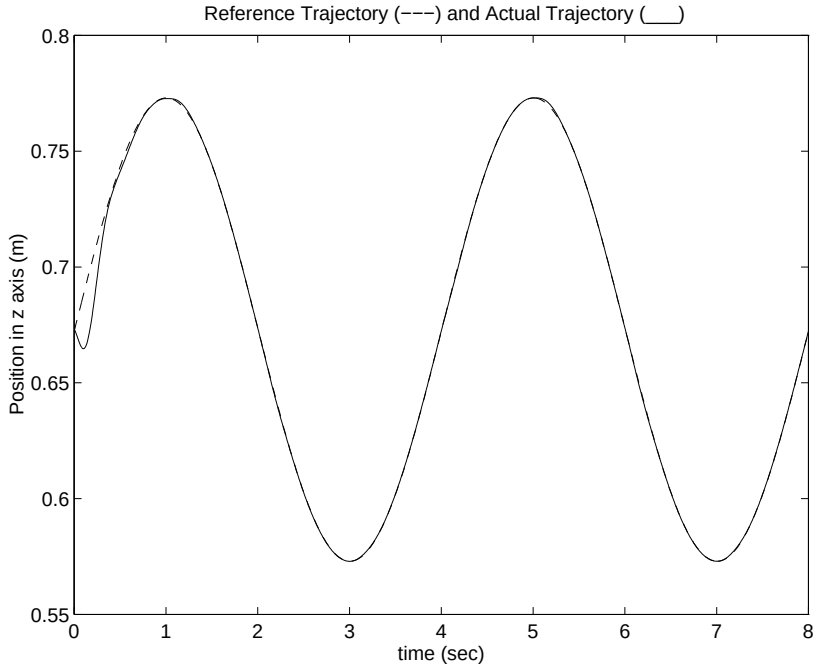


Figure 7.35: Position Tracking in z axis of Fig. 7.34

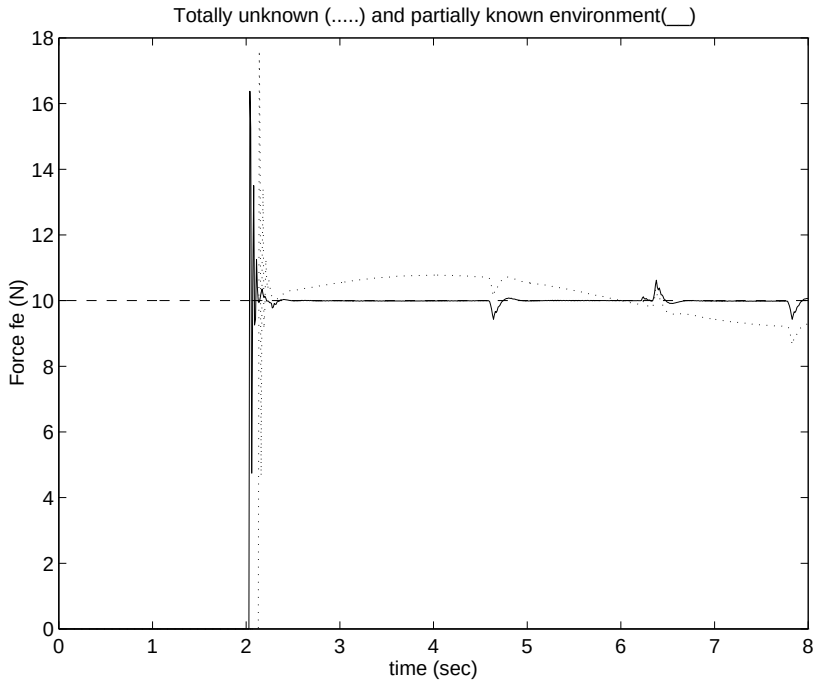


Figure 7.36: Force Tracking with time-varying environment surface location as shown of Fig. 13

7.9.6.3 Time-varying environment surface location

It is more interesting to see the performance of the proposed controller when the environment surface location is also time varying. Two force tracking performances when $\boldsymbol{\eta = 0.01}$ are shown in Figure 12. One is when environment surface location is totally unknown(called A). The force tracking offset error occurs due to not knowing the surface velocity profile of environment location as we assume $\dot{\boldsymbol{x}}'_e = \ddot{\boldsymbol{x}}'_e = \mathbf{0}$. When the surface velocity profile of environment location is given the force tracking is much improved as shown in Figure 7.36. This is called B in Figure 7.37. The position tracking shown in Figure 7.38 are also excellent.

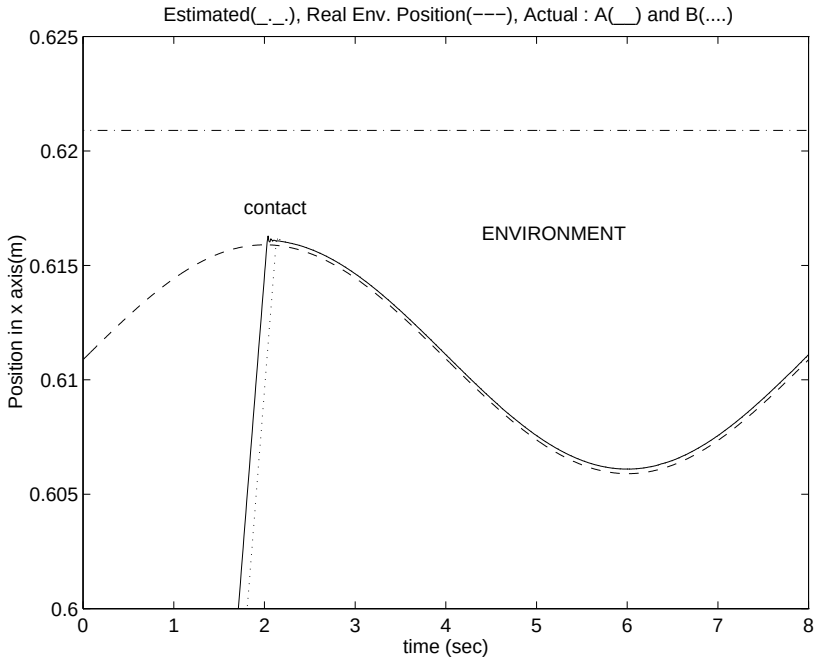


Figure 7.37: Position Tracking in $\boldsymbol{x}$ axis with estimated and real environment surface position

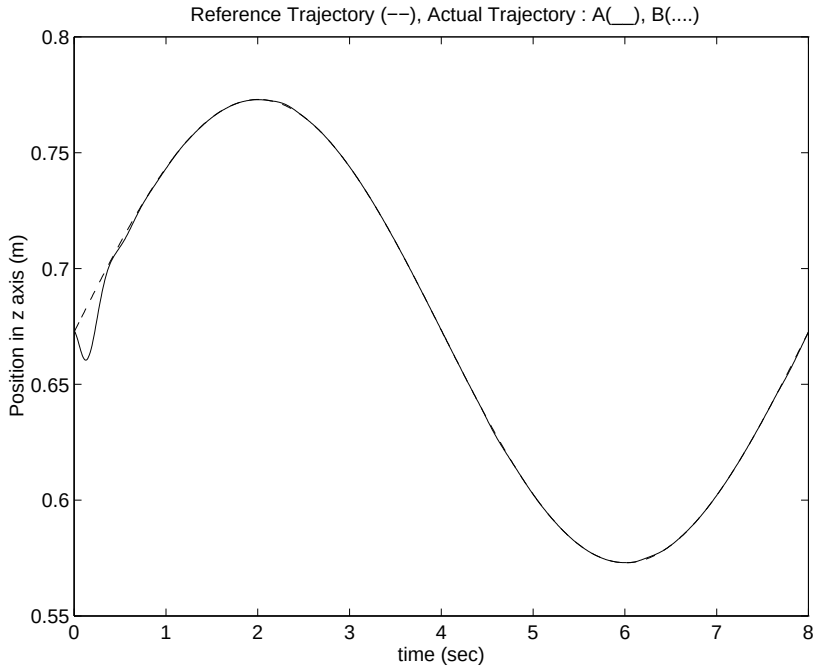


Figure 7.38: Position Tracking in $\boldsymbol{z}$ axis of Fig. 12

7.9.7 Discussion

A neural network control technique for impedance control of robot manipulator is presented in this paper. Stability analysis has been presented to show the boundary conditions for good force tracking. Intensive simulation studies have been done to confirm those analytical results. The critical bound is found to be $-\frac{f_d}{k} < \delta x_e < \frac{f_d}{k}$. The proposed NN controller showed the robustness under unknown environment stiffness and location by compensating for the uncertainties in the robot dynamics. In next section, a simple NN control algorithm that solves the problem of force tracking error occurred when the surface velocity profile of environment location is not given.

7.10 Simple Robust NN Force Control

In this section, a new robust force tracking impedance control scheme that uses neural network as a compensator is proposed. The proposed scheme has the capability of tracking a specified desired force as well as of compensating for uncertainties from unknown environment location and stiffness. In addition, at the same time, the uncertainties in robot dynamics are also compensated by the same neural network. In order to train the neural compensator, switching learning algorithm is employed to deal with transitional conditions from free space to contact space. Simulation studies with three link rotary robot manipulator are carried out to demonstrate the robustness of the proposed scheme under uncertainties in robot dynamics, and no

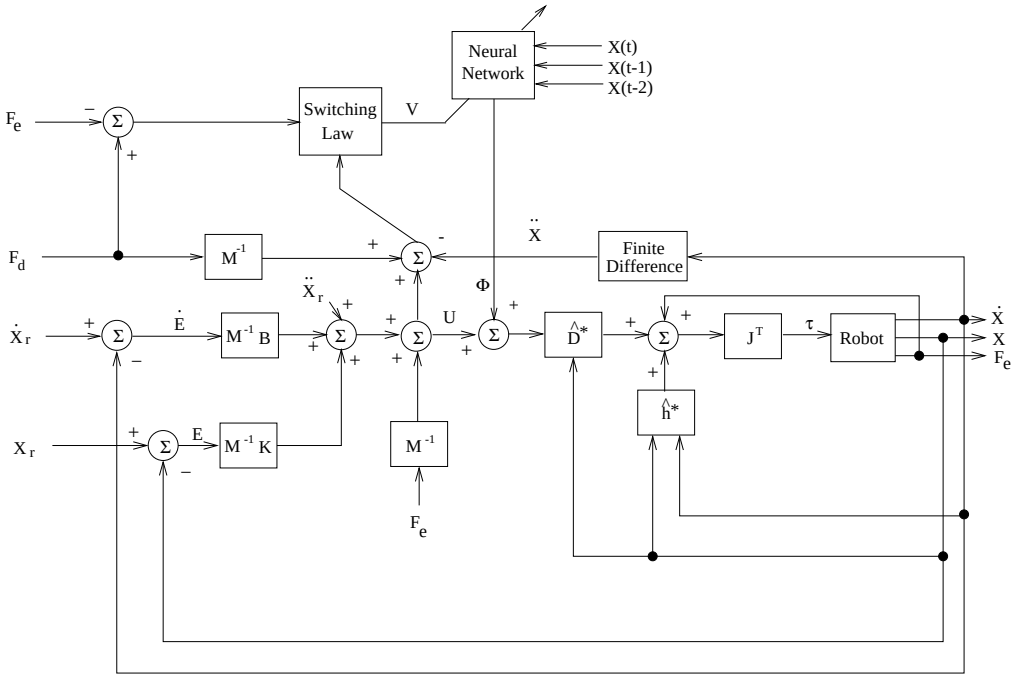


Figure 7.39: NN Force Control Structure

knowledges of environment position and environment stiffness. The results show that the excellent force tracking is achieved by single neural network.

In the previous section, we proposed a nonlinear impedance control (7.64) as

$$\ddot{\boldsymbol{\varepsilon}} + \frac{\mathbf{1}}{\boldsymbol{m}}(\boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{k}\boldsymbol{\varepsilon}(\mathbf{1} - \frac{\boldsymbol{f_d}}{\boldsymbol{f_e}}) + \boldsymbol{f_d} - \boldsymbol{f_e}) = \boldsymbol{\delta_m} - \boldsymbol{\delta_f} - \boldsymbol{\phi} \tag{7.87}$$

The careful investigation of (7.87) suggests to drop the nonlinear term in order to linearize and simplify the impedance function without loss of the main concept of the impedance control. By setting $\boldsymbol{k} = \mathbf{0}$ yields the new impedance function is realized as

$$\boldsymbol{m}\ddot{\boldsymbol{\varepsilon}} + \boldsymbol{b}\dot{\boldsymbol{\varepsilon}} + \boldsymbol{f_d} - \boldsymbol{f_e} = \boldsymbol{m}(\boldsymbol{\delta_m} - \boldsymbol{\delta_f} - \boldsymbol{\phi}) \tag{7.88}$$

Let us define $\boldsymbol{x}'_{\boldsymbol{e}} = \boldsymbol{x_e} + \boldsymbol{\delta x_e}$ as an estimation of environment position so that there

is a uncertainty $\boldsymbol{\delta x_e}$. Then, the proposed impedance function becomes

$$\ddot{\varepsilon}' + \frac{1}{m}(b\dot{\varepsilon}' + f_d - f_e) = \delta_m - \delta_f - \phi \tag{7.89}$$

where $\ddot{\varepsilon}' = \ddot{\varepsilon} + \delta\ddot{x}_e$ and $\dot{\varepsilon}' = \dot{\varepsilon} + \delta\dot{x}_e$. Since the environment location is not known exactly, $\ddot{x}_e{}' = \dot{x}_e{}' = 0$ The only approximation can be made is the estimation of the environment position x_e' which is assumed to be parallel to the environment. Then, in free space, v_f becomes

$$v_f = -\ddot{x} + \frac{1}{m}(-b\dot{x} + k\varepsilon') + f_d) \tag{7.90}$$

where $\varepsilon' = x_e' - x$. In contact space, the equation (7.89) becomes

$$v_c = -\ddot{x} + \frac{1}{m}(-b\dot{x} + f_d - f_e) = \delta_m - \delta_f - \phi \tag{7.91}$$

If the training signal defined in (7.91) is used directly then the force tracking will have the offset of the magnitude of $\ddot{x} + \frac{1}{m}b\dot{x}$ due to not specifying $\ddot{x_e}$ and $\dot{x_e}$ correctly. In order to compensate for this offset it is suggested to use the force error directly as

$$v_c = f_d - f_e = \delta_m - \delta_f + \delta_p - \phi \tag{7.92}$$

where $\delta_p = \ddot{x} + \frac{1}{m}b\dot{x}$. So the neural compensator compensates for the uncertainties $\delta_m - \delta_f + \delta_p$ in environment position as well as in robot dynamics.

7.10.1 Performances

For the NN controller, we have chosen six hidden neurons $(n_H = 6)$. The back propagation algorithm parameters are: $\eta = 0.01$ and $\alpha = 0.9$. Weights

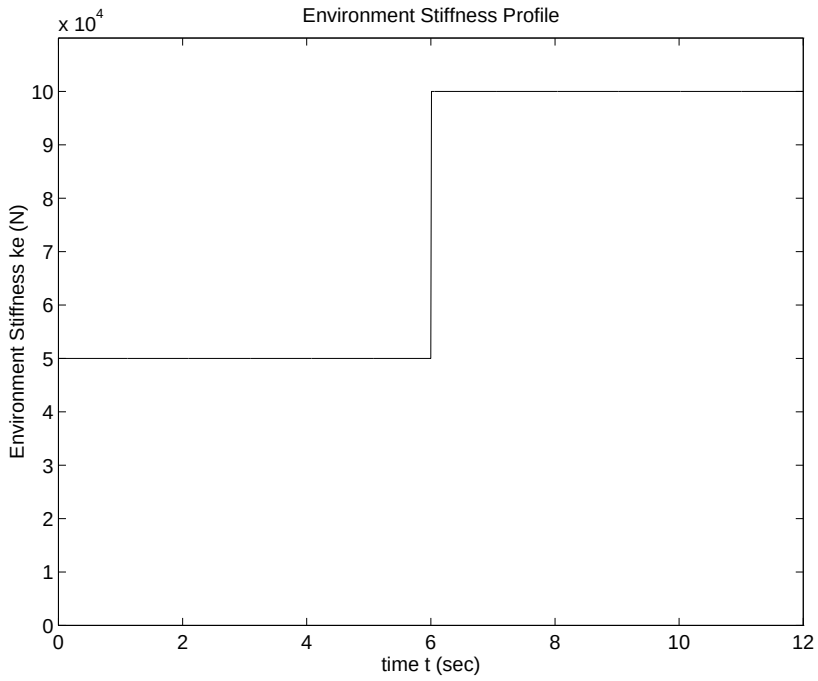


Figure 7.40: Time Varying Environment Stiffness Profile

are randomly selected.The controller gains in free space are selected as $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{60, 40, 40}]$ and $\boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{100, 100, 100}]$ which give over-damped motions at the three joints. In contact space, the controller gains are $\boldsymbol{K_D} = \boldsymbol{diag}[\mathbf{400, 40, 40}]$ and $\boldsymbol{K_P} = \boldsymbol{diag}[\mathbf{0, 100, 100}]$. In order to show the robustness to unknown environment stiffness of the proposed scheme, we tested the system performance for abruptly changing environment stiffnesses with the stiffness profile as

$$\boldsymbol{k_e} = \begin{cases} \mathbf{50000} & \mathbf{0 \leq t < 6} \\ \mathbf{100000} & \mathbf{6 \leq t < 12} \end{cases} \tag{7.93}$$

which is shown in Figure 7.40.

Two tasks are performed. Task 1 is when the environment surface is flat. The

environment position is not correctly estimated as $\boldsymbol{x}_e' = \boldsymbol{x}_e + \mathbf{0.01}\boldsymbol{m}$ which is inside the environment. In order to test the force tracking capability the desired reference force is specified as

$$\boldsymbol{f}_d = \mathbf{10} + \mathbf{2}\boldsymbol{sin}(\frac{\boldsymbol{\pi} * \boldsymbol{t}}{\mathbf{4}}) \tag{7.94}$$

Sample tracking results are plotted in Figures 7.41 and 7.42. The simulation data showed that the force tracking result is very effective and the corresponding position tracking is also good. In Task 1, the control algorithm is governed by the equation (7.91) which does not require $\dot{\boldsymbol{x}}_e, \ddot{\boldsymbol{x}}_e$ informations ($\dot{\boldsymbol{x}}_e = \ddot{\boldsymbol{x}}_e = \mathbf{0}$) since the environment is flat and parallel to estimated environment position $\boldsymbol{x}_e'$. As we expected from the previous analytical section the performance is excellent with small delay at the beginning. The proposed NN controller works well even when the environment location is not estimated correctly.

The more interesting task is when the environment is sinusoidally shaped. As same as task 1, only information available is the estimated environment position $\boldsymbol{x}_e'$ which is specified inside the environment and parallel to the environment. Using the training signal defined in (7.92) the force tracking is shown in Figure 7.43 which is very good. The position tracking plots are shown in Figures 7.44 and 7.45.

7.10.2 Discussion

A robust neural network scheme for position and force tracking control of robot manipulator is presented in this section. The proposed NN controller as an exten-

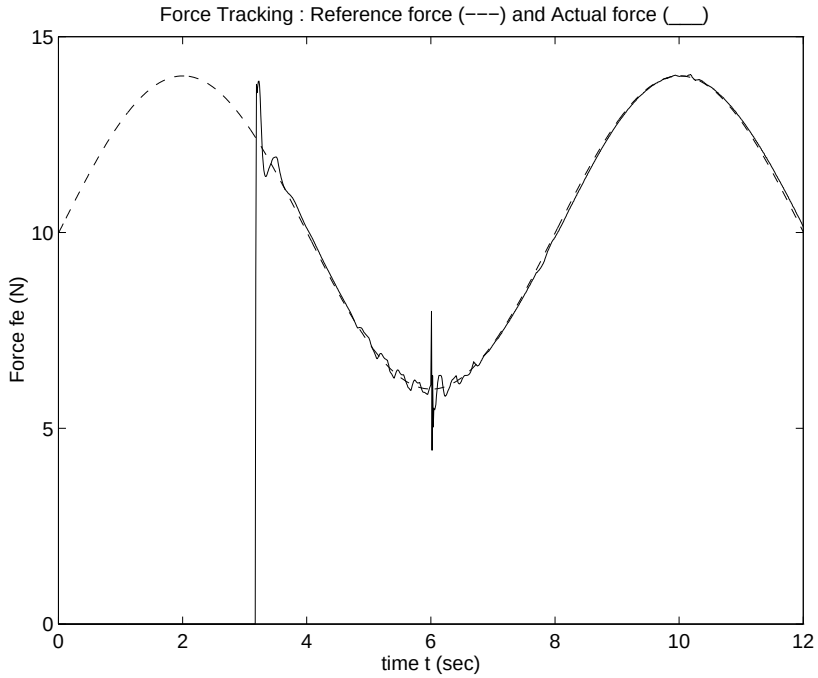


Figure 7.41: Flat surface : Force Tracking

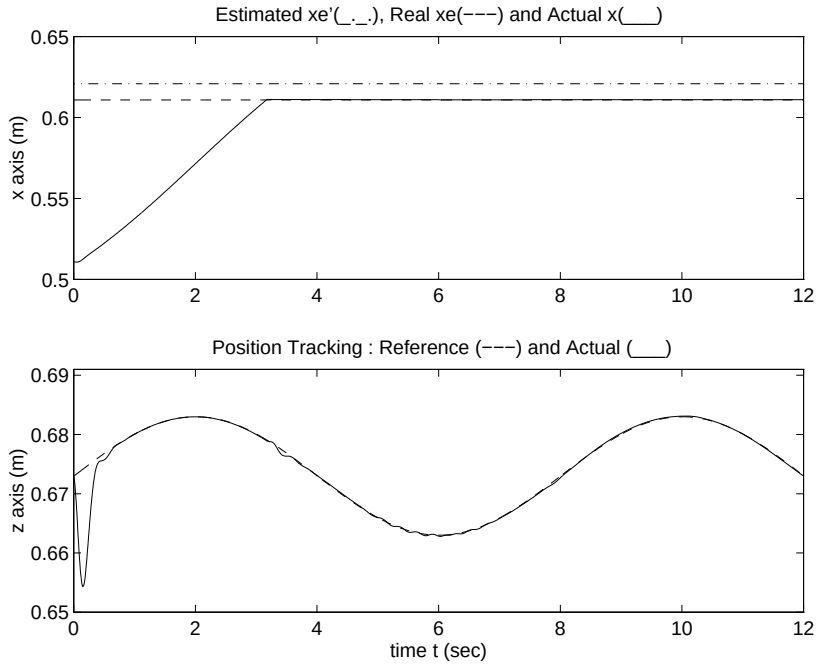


Figure 7.42: Flat surface : Position Tracking

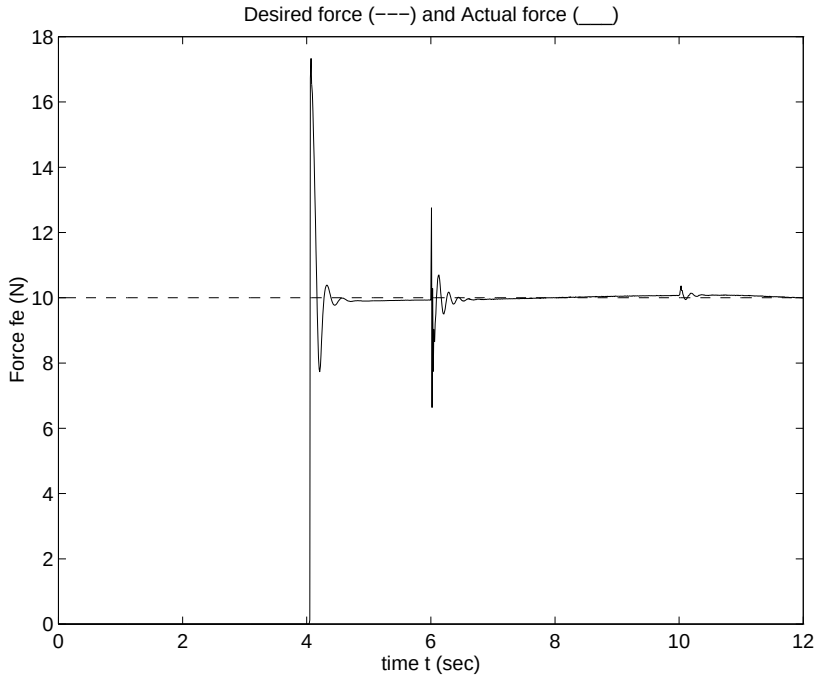


Figure 7.43: Time varying surface : Force Tracking

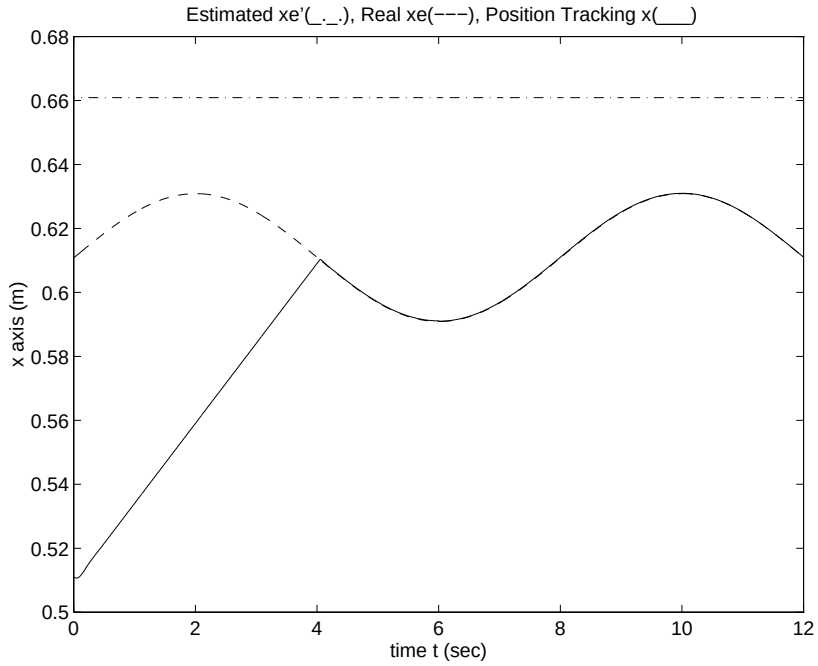


Figure 7.44: Time varying surface : Position Tracking in x axis

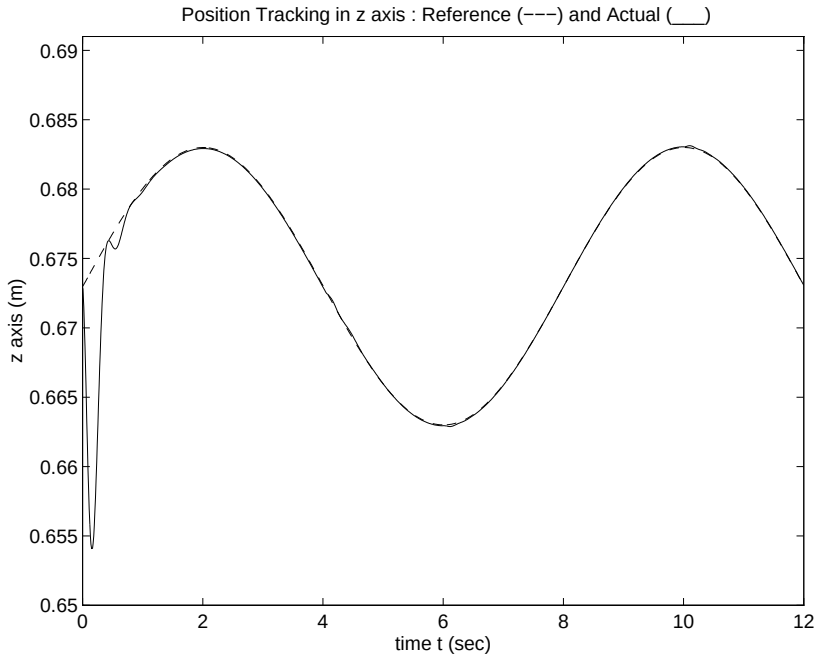


Figure 7.45: Time varying surface : Position Tracking in z axis

sion of the previous nonlinear impedance control. is capable of compensating for the uncertainties in both the robot dynamics and the environment. Neural network training signals are proposed for both free space and contact space control. Computer simulation results show that the proposed control is able to achieve excellent force/position tracking in the presence of robot dynamic uncertainties, unknown environment stiffness and position. Controller robustness with respect to environment position uncertainties was also analyzed and confirmed by the simulation results. The system performance under the proposed control scheme is excellent. Thus, the proposed NN controller is feasible for on-line robot control and one final solution that solves all uncertainty problems. In next chapter, an alternative way of implementing

robust control is introduced.

Chapter 8

Time-Delayed Robust Control

In this Chapter, robust control schemes for both position and force control are presented. The proposed control schemes are not using neural network but using time delayed informations to compensate for uncertainties. The control algorithm is so simple that it can be easily implemented in real time.

8.1 Robust position Control

The previous results in Chapter 3 have shown that neural network techniques are effective in compensating highly nonlinear uncertainties in the robot model where computed torque method is used for robot motion control. The propose of this section is to present a simple alternate solution to the same control problem which eliminates the need of a neural network. The solution is based on the robust position control technique by the authors [6, 11, 12]. Computer simulations show that the alternate control method works better.

8.1.1 Introduction

It is a well established fact that model-based computed-torque control scheme is the basic technique for robot control system design. But it is also a common knowledge that the scheme has poor robustness when uncertainties exist in the robot model. To improve the robustness, application of both adaptive control techniques and neural network control techniques have been studied in Chapter 3. The latter

approach has been actively investigated in this research. Eliminating the effects of model uncertainties by adaptive control scheme is to estimate the robot parameters on-line. On the other hand, a disturbance rejection torque is generated on-line by a neural network to cancel out the effects of uncertainties in robot dynamics [43]. Both of these approaches require extensive computation. The purpose of this section is to examine the basic idea behind the neural network scheme proposed by Ishiguro et. al [43], and to present an extremely simple alternate scheme to solve the same problem.

8.1.2 An Alternative to Neural Network Control Scheme

The robot control scheme proposed in [43] can be summarized as follows:

Given the robot manipulator dynamics as

$$\boldsymbol{\tau} = \boldsymbol{D}\ddot{\boldsymbol{q}} + \boldsymbol{h} + \boldsymbol{\tau_f} \tag{8.1}$$

and employing a computed-torque controller plus a neural network compensator results in a robot tracking error equation

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \hat{\boldsymbol{D}}^{-1}(\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f} - \boldsymbol{\tau_N}) \tag{8.2}$$

where $\boldsymbol{\Delta D} = \boldsymbol{D} - \hat{\boldsymbol{D}}, \boldsymbol{\Delta h} = \boldsymbol{h} - \hat{\boldsymbol{h}}$ and $\boldsymbol{\tau_N}$ is the neural network compensator output. The basic control idea is to train the neural network such that

$$\boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f} - \boldsymbol{\tau_N} = \boldsymbol{0} \tag{8.3}$$

Hence a teaching signal $\boldsymbol{\tau_t}$ can be defined as

$$\boldsymbol{\tau_t} = \boldsymbol{\Delta D}\ddot{\boldsymbol{q}} + \boldsymbol{\Delta h} + \boldsymbol{\tau_f} \tag{8.4}$$

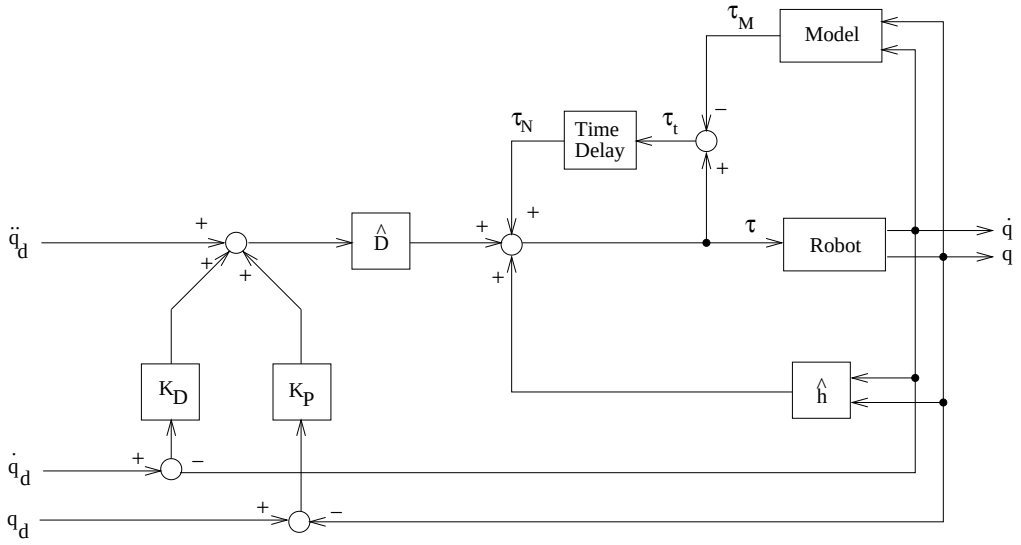


Figure 8.1: The proposed alternate control scheme to that in Ishiguro's

or

$$\begin{aligned} \boldsymbol{\tau_t} &= \boldsymbol{\tau} - (\hat{\boldsymbol{D}}\ddot{\boldsymbol{q}} + \hat{\boldsymbol{h}}) \\ &= \boldsymbol{\tau} - \boldsymbol{\tau_M} \qquad\qquad \boldsymbol{\tau_M} = \hat{\boldsymbol{D}}\ddot{\boldsymbol{q}} + \hat{\boldsymbol{h}} \end{aligned} \tag{8.5}$$

When the training is successful, we can achieve the ideal results

$$\boldsymbol{\tau_N} = \boldsymbol{\tau_t} = \boldsymbol{\tau} - \boldsymbol{\tau_M} \tag{8.6}$$

and

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \boldsymbol{0} \tag{8.7}$$

A multilayer feedforward neural network with back propagation updating rule has been suggested for the compensator. An alternate technique is presented below which accomplishes the same task with extreme simplicity.

The term $\hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f)$ in (8.2) is essentially a highly nonlinear and complex disturbance in the computed-torque control system caused by robot model uncertainties $\Delta D, \Delta h$, and τ_f, and τ_N is generated by the neural network to cancel out this disturbance. From this standpoint, other disturbance rejection techniques which fulfill the requirement (8.3) could also be applicable. One such simple technique has been studied by the authors' research group for several years for robot motion control [12]. We now show how to adapt that idea to the problem at hand.

Based on the concept in [12], the teaching signal τ_t at time t can be closely approximated by its time delayed value $\tau_t(t-\lambda)$ when the delay time λ is sufficiently small and $\tau_t(t)$ is a continuous function. Then instead of generating the signal τ_N from the neural network compensator, we simply approximate τ_N at time t in (8.6) as

$$\tau_N(t) \cong \tau_t(t-\lambda) = \tau(t-\lambda) - \tau_M(t-\lambda) \tag{8.8}$$

where $\tau_M(t)$ is defined by the nominal robot model (8.5). In this way, the disturbance rejection torque τ_N is instantly computed from the time-delayed values of τ and τ_M. In practice λ can be conveniently chosen as the sampling period in the digitally implemented controller, and λ is usually very small. So previous samples of $\tau(t-\lambda)$ and $\tau_M(t-\lambda)$ can be stored at every sampling period for computing (8.8). The alternate control scheme is depicted in Figure 8.1. This scheme requires no neural network. The approximation of (8.8) leads to a closed loop error equation

$$\ddot{E} + K_D\dot{E} + K_P E = \hat{D}^{-1}\Delta\tau \tag{8.9}$$

where $\boldsymbol{\Delta\tau(t) = (\tau(t) - \tau(t-\lambda)) - (\tau_M(t) - \tau_M(t-\lambda))}$. Since in practice $\boldsymbol{\tau(t)}$ and $\boldsymbol{\tau_M(t)}$ are reasonably smooth function, and $\boldsymbol{\Delta\tau(t)}$ is bounded and small for small $\boldsymbol{\lambda}$, the compensated error equation (8.9) is reasonably close to the ideal result (8.7). It is easy to see that $\boldsymbol{\Delta\tau(t) \rightarrow 0}$ when $\boldsymbol{\lambda \rightarrow 0}$. With the aid of DSP hardware, it is possible to compute $\boldsymbol{\tau_M}$ at a sampling rate of 1000 Hz [89]. Thus $\boldsymbol{\Delta\tau}$ can be made sufficiently small in practice. In our controller implemented at UCD for PUMA560 the sampling rate becomes 200 $\boldsymbol{Hz}$. Simulation results of the proposed scheme are presented below.

8.1.3 Computer Simulation Study

A robot manipulator with three rotary joint is used in the simulation. Robot model parameters are taken from the first three links of a Puma 560 arm. Model uncertainties are introduced by adding a payload of 10 kg at the end of third link, and a friction torque of $\boldsymbol{\tau_f(\dot{q}) = 5.0sgn(\dot{q}) + 8.0\dot{q}}$ at each joint. The end point is commanded to track a circle of radius 30 cm which is tilted 45 degrees in the 3-D Cartesian space with a cycle time of 4 secs. Using the proposed control scheme, the joint position and velocity tracking errors for all three joints are plotted in Figure 8.2. Also plotted in Figure 8.2 for comparison are the errors of the neural network control scheme in [43] where the update rate $\boldsymbol{\eta}$, momentum coefficient $\boldsymbol{\alpha}$, a number of hidden units $\boldsymbol{N_H}$, and the output layer bound B are optimized as : $\boldsymbol{\eta = 0.001, \alpha = 0.9, N_H = 6, B = \pm 50}$. Initial weights are randomly selected.

Another simulation that the robot manipulator is required to follow composite trajectory which is composed of four different trajectories as shown in Figure 8.3 is carried out. The proposed control scheme outperforms the Ishiguro's neural network control scheme in tracking as shown in Figure 8.3 where the neural network parameters are optimized as : $\boldsymbol{\eta = 0.0005, \alpha = 0.9, N_H = 6, B = \pm 90}$ in this case. The corresponding error plots in Figure 8.4 are clearly shown the superior performance of the robust position control. One of the reason that Ishiguro's scheme performs poor is due to using accelerations as neural network controller inputs as indicated in Chapter 3. Therefore, it is interesting to investigate the performance without acceleration inputs to NN. As expected, eliminating acceleration inputs to NN the tracking performances are improved for both circular and composite trajectory as shown Figure 8.5 and Figure 8.6, respectively. Also plotted in Figure 8.7 is the corresponding joint errors to Figure 8.6. The NN parameters are optimized as : $\boldsymbol{\eta = 0.02, \alpha = 0.9, N_H = 6, B = \pm 60}$ for circular trajectory and $\boldsymbol{\eta = 0.01, \alpha = 0.9, N_H = 6, B = \pm 100}$ for composite trajectory. The detailed simulation studies regarding NN input types have been found in Chapter 3.

The PD gains in both schemes are $\boldsymbol{K_D = diag[20, 20, 20]}$ $\boldsymbol{K_P = diag[100, 100, 100]}$, and the sampling rate is 200 $\boldsymbol{Hz}$. We see that not only the alternate scheme works, but it also works better with tracking errors kept steadily near zero at all times.

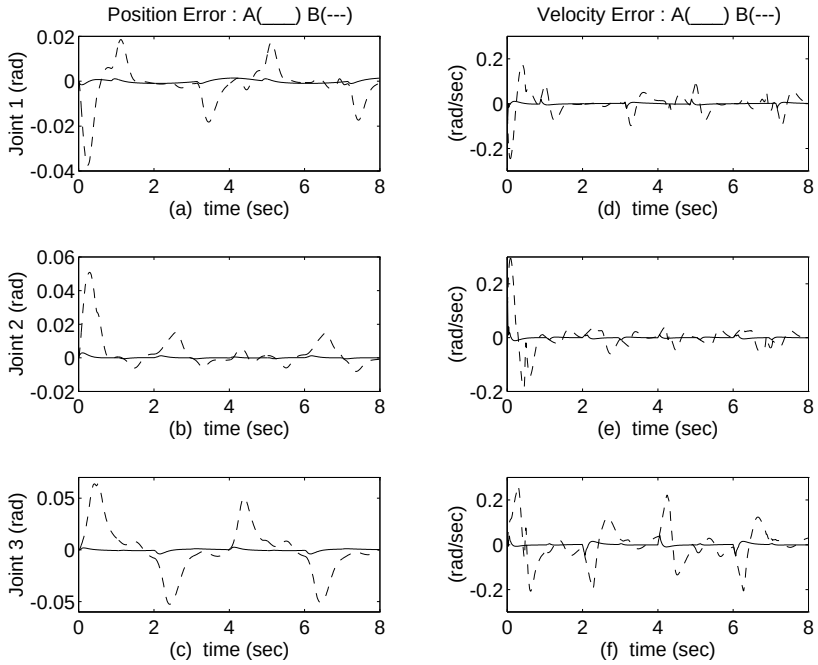


Figure 8.2: Circular Tracking Errors for (A) Proposed Scheme (B) Neural Network Scheme : Joint Position error (a)(b)(c), and Joint Velocity Error (d)(e)(f)

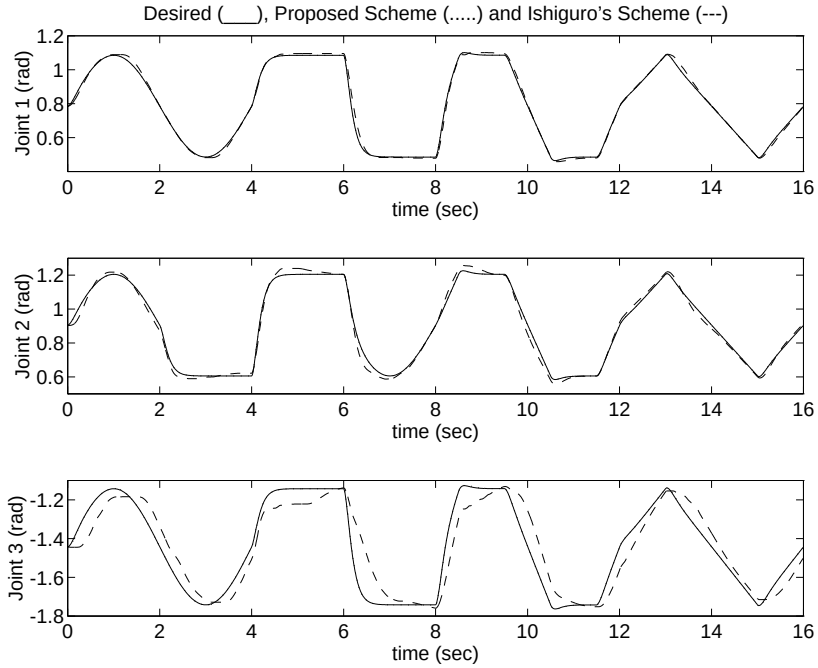


Figure 8.3: Composite Trajectory Tracking for (A) Proposed Scheme (B) Neural Network Scheme

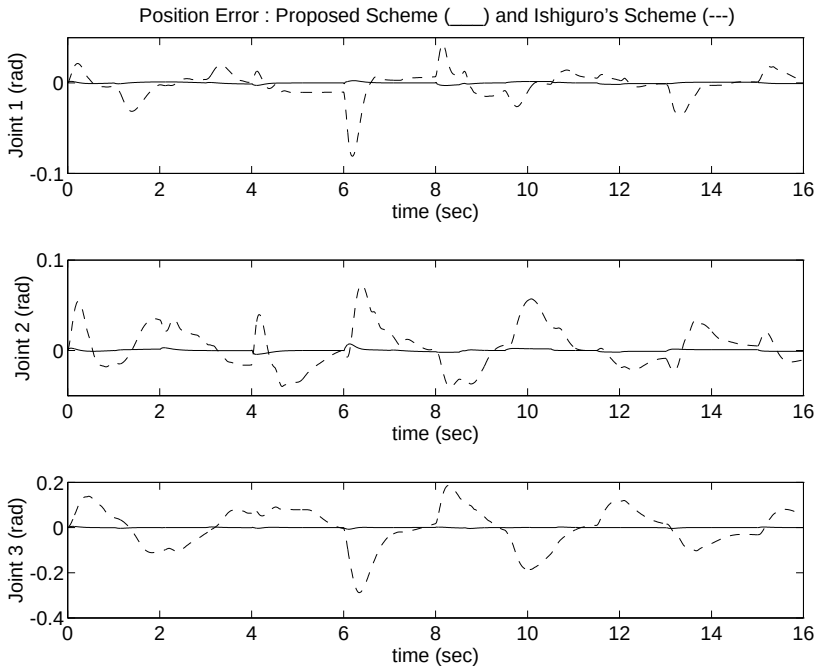


Figure 8.4: Composite Trajectory Tracking Error for Proposed Scheme and Neural Network Scheme

8.1.4 Discussion

Having been shown by simulation that the alternate scheme is working extremely well in comparison with the original neural network scheme under the same control environment, we may wish to ask the following interesting question. Given that the nominal robot parameters $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{h}}$ are available and compensation of uncertainties in the computed-torque controller can be achieved by the proposed scheme as well as other robust schemes, would it not be unfair to say that neural network schemes such as [43] are unnecessarily complicated solutions to the problem? It is undeniable that neural network is a powerful technique which can solve complicated nonlinear control problems in robotics. The authors of [43] have stated that the neural network

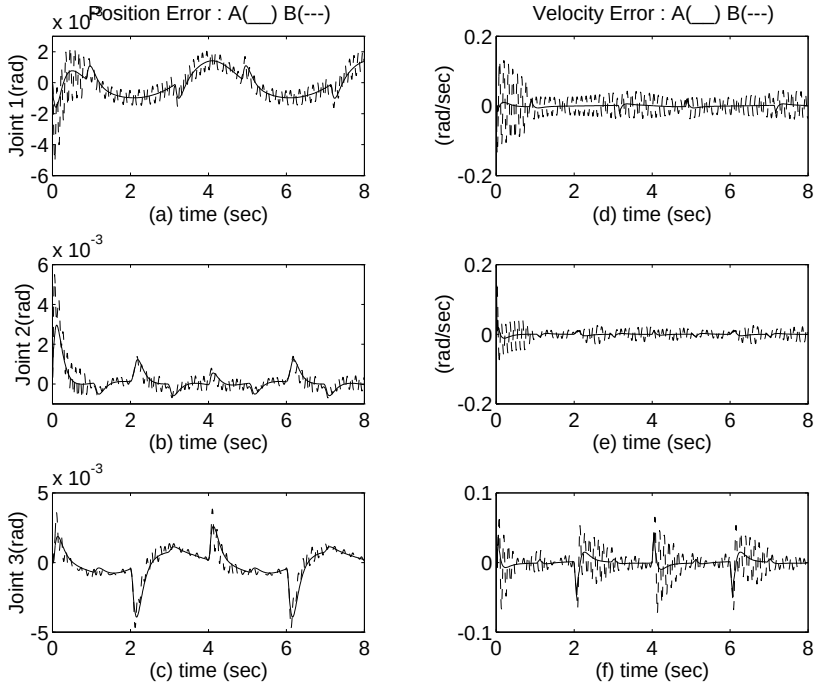


Figure 8.5: Circular Tracking Errors for (A) Proposed Scheme (B) Neural Network Scheme when Acceleration Inputs to NN are Eliminated : Joint Position error (a)(b)(c), and Joint Velocity Error (d)(e)(f)

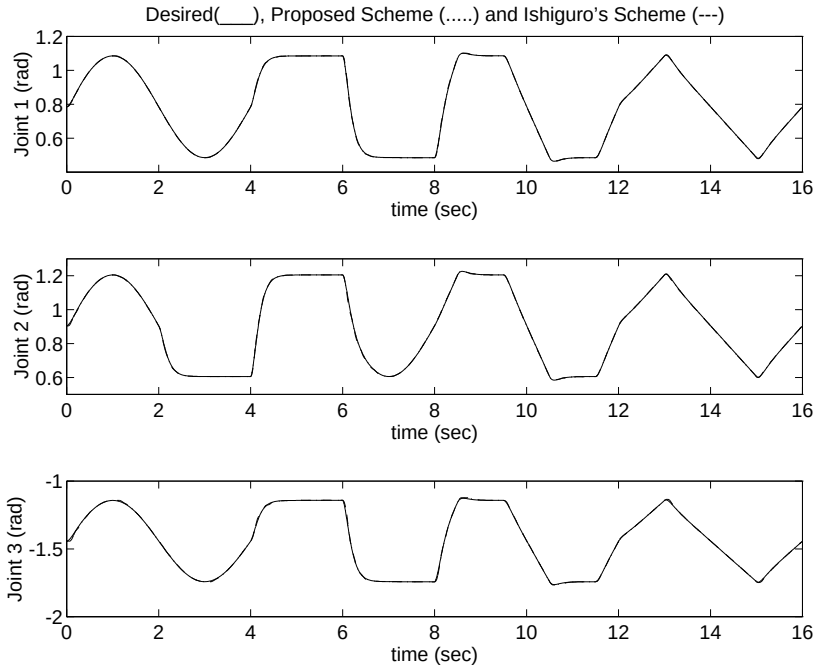


Figure 8.6: Composite Trajectory Tracking for (A) Proposed Scheme (B) Neural Network Scheme when Acceleration Inputs to NN are Eliminated

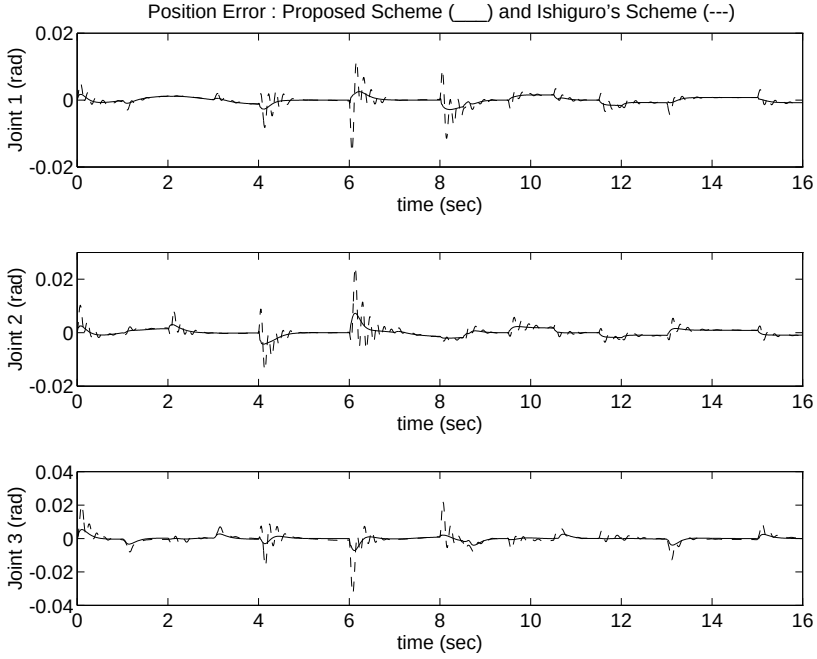


Figure 8.7: Composite Trajectory Tracking Error for Proposed Scheme and Neural Network Scheme when Acceleration Inputs to NN are Eliminated

works better when it is required to learn model uncertainties instead of the complete robot inverse dynamics. In our view, the neural network is under-used in this type of application. So it is important for us researchers to carefully examine the full potential and special benefits that can be derived from using neural network for robot control when $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{h}}$ are available. As of now, a strong case has been made that neural network solution is meaningful for non-model based robot control in Chapter 4.

8.2 Robust Adaptive Force Control

8.2.1 Introduction

After the pioneering work of impedance control by Hogan [83] there have been two major research issues to be solved on the impedance control: one is to give the force tracking capability to impedance control and another is to compensate for the uncertainties occurred in both robot dynamics and the environment(position and stiffness).

Within this impedance force control framework, many control algorithms have been proposed. Lasky and Hsia [85] have proposed the inner/outer loop control scheme where the robot dynamics uncertainties are compensated by a robust position control algorithm in the inner loop and estimated environment position is modified using an integral control of the force tracking error in the outer loop. The generalized impedance control of considering a general dynamic relation between a position error

and a force error to deal with the unknown environment stiffness has been proposed [90]. The adaptive techniques of using force tracking errors have been proposed to estimate environment stiffness or adjust controller gains to compensate for unknown environment stiffness [86, 91, 92]. The authors have proposed a simple trajectory modification scheme using the robot controller to compensate for robot dynamics uncertainties as in [85] and environment stiffness is replaced by contact force information [93]. Later analysis has shown that accurate force tracking is not always guaranteed unless the accuracy of the estimated environment position is within certain bounds [88].

The purpose of this section is to provide a simple solution to the force tracking impedance control algorithm that is robust with respect to uncertainties in both robot dynamic model and environment position and stiffness. The main idea is to set the stiffness gain in the impedance function immediately to zero in force control direction as soon as contact to the environment is made, and the nominal environment position is chosen in such a way that the robot will be assured to make contact with the environment from any position in free space. Environment stiffness knowledge is not required by the algorithm. The proposed control law is very simple that it can be implemented with ease.

Simulation studies of a three link rotary robot manipulator are carried out to demonstrate the robustness under uncertainties in robot dynamics, environment position, and environment stiffness. Experimental results using a PUMA 560 robot arm

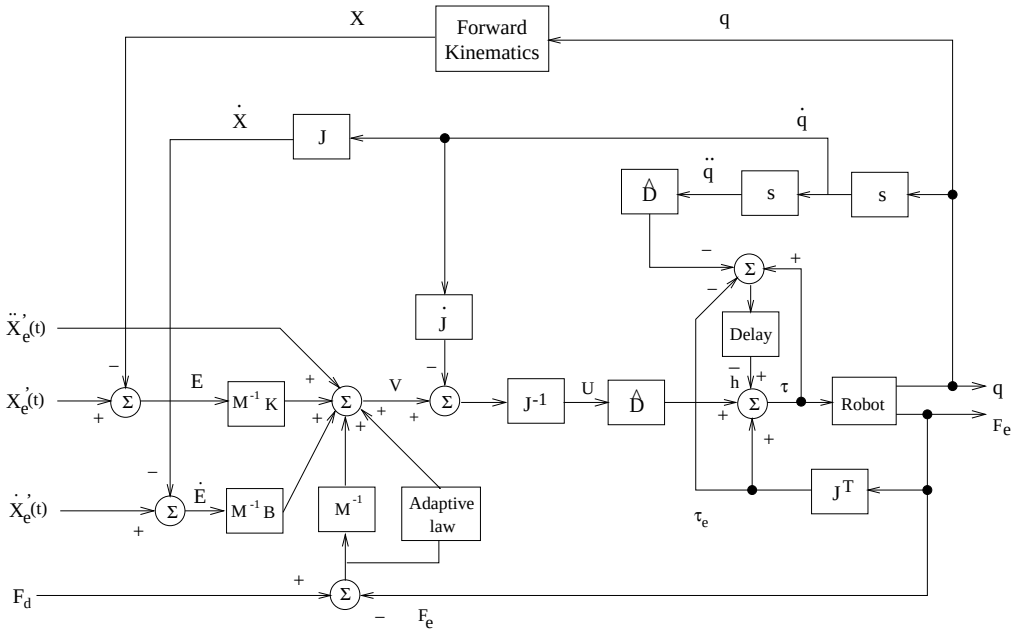


Figure 8.8: Force Control Structure with Robust Position Control Algorithm

are presented to confirm the performance of the control scheme.

$$\boldsymbol{F_e = M\ddot{E} + B\dot{E} + KE} \tag{8.10}$$

where $\boldsymbol{F_e}$ is equal to the desired force $\boldsymbol{F_d}$. Since there are always uncertainties in the robot dynamic model the ideal target impedance relationships (8.10) can not be achieved in general. Thus the impedance based force control is not robust in practice.

Another problem is the difficulty of knowing the environment stiffness accurately in advance for us to design the reference trajectory $\boldsymbol{X_r}$ to achieve the desired contact force. These difficulties can be solved by the force tracking impedance control in the next section.

8.2.2 The Proposed Outer Loop Control Scheme

8.2.2.1 Control law

The proposed control structure is shown in Figure 8.8. Replacing $\boldsymbol{X_r}$ with $\boldsymbol{X_e}$ in (8.10) and subtracting the desired force $\boldsymbol{F_d}$ from $\boldsymbol{F_e}$ yields the new impedance function

$$\boldsymbol{F_e - F_d = M\ddot{E} + B\dot{E} + KE} \tag{8.11}$$

where $\boldsymbol{E = X_e - X}$ and $\boldsymbol{X_e}$ is the environment position.

For simplicity, we consider that force is applied to only one direction. Let $\boldsymbol{f_d, f_e, m, b, k}$ be elements of $\boldsymbol{F_d, F_e, M, B, K}$ respectively. Then, equation (8.11) becomes

$$\boldsymbol{m\ddot{e} + b\dot{e} + ke - f_e + f_d = 0} \tag{8.12}$$

where $\boldsymbol{e = x_e - x}$.

We propose a two-phase control algorithm: the first phase is free-space control when the robot is approaching toward the environment and the second phase is contact-space control in which the end-effector is in contact with the environment. In free space control, the control law can be obtained from (8.12) since $\boldsymbol{f_e = 0}$ as follows:

$$\boldsymbol{m\ddot{e} + b\dot{e} + ke = -f_d} \tag{8.13}$$

If the desired force is set to zero in the equation (8.13) when the environment position is known exactly, then the robot would just make contact with the environment since the position tracking $\boldsymbol{e}$ is required to track the environment position, not the

trajectory inside the environment. The desired force $\boldsymbol{f_d}$ is the driving force to enable the robot to exert a force on the environment.

In contact space, the careful investigation of the impedance relation (8.13) suggests that eliminating the stiffness gain $\boldsymbol{k}$ will satisfy the ideal impedance function directly in terms of a real environment stiffness $\boldsymbol{k_e}$ and a desired force $\boldsymbol{f_d}$ so that one does not need to select the stiffness gain $\boldsymbol{k}$. The proposed law is designed to set the stiffness gain $\boldsymbol{k} = \boldsymbol{0}$ in the force controllable direction. In the position controllable direction $\boldsymbol{k}$ will not be changed. Setting $\boldsymbol{k} = \boldsymbol{0}$ upon contact and substituting $\boldsymbol{f_e} = \boldsymbol{k_e}(\boldsymbol{x} - \boldsymbol{x_e}) = -\boldsymbol{k_e}\boldsymbol{e}$ into (8.13) realizes the new impedance law as

$$\boldsymbol{m}\ddot{\boldsymbol{e}} + \boldsymbol{b}\dot{\boldsymbol{e}} + \boldsymbol{k_e}\boldsymbol{e} = -\boldsymbol{f_d} \tag{8.14}$$

which is stable and ideal. Even though the environment stiffness $\boldsymbol{k_e}$ is not known exactly the ideal impedance function can be realized by selecting the proper gains $\boldsymbol{m}$ and $\boldsymbol{b}$ based on approximation of $\boldsymbol{k_e}$. For high stiffness environment, one could have a large bandwidth by selecting a suitable damping gain $\boldsymbol{b}$ such that the closed loop poles of the equation (8.14) can be located far from the origin. Therefore, our proposed impedance function is simple, stable and robust in terms of gain selection, control structure, and force tracking capability under unknown environment stiffness.

8.2.2.2 Inaccurate environment position

Consider that we have an inaccurate environment position estimation such that $\boldsymbol{e'} = \boldsymbol{e} + \boldsymbol{\delta x_e}$ where $\boldsymbol{\delta x_e}$ is the inaccuracy of the environment position which can

be specified by the control designer. In free space, our previous analysis has shown that when $\boldsymbol{\delta x_e} < \mathbf{0}$ a boundary condition has to be satisfied in order for the robot manipulator to make contact with the environment [88]. If the condition is not satisfied the robot end-effector stays in free space without making contact with the environment. As an alternative condition, when $\boldsymbol{\delta x_e} > \mathbf{0}$, the contact can always be made. Therefore, the better strategy is to specify the estimated environment position, $\boldsymbol{x'_e}$ inside the environment ($\boldsymbol{\delta x_e} > \mathbf{0}$) such that contact is guaranteed to occur. Replacing $\boldsymbol{e'}$ into (8.14), in contact space, yields

$$\boldsymbol{m\ddot{e}' + b\dot{e}' = f_e - f_d} \tag{8.15}$$

When the environment is a flat surface such that $\boldsymbol{x_e}$ is constant, equation (8.15) becomes

$$\boldsymbol{m\ddot{x} + b\dot{x} = f_d - f_e} \tag{8.16}$$

since $\boldsymbol{\dot{x}_e = \ddot{x}_e = 0}$ and $\boldsymbol{\delta\ddot{x}_e = \delta\dot{x}_e = 0}$. The steady state of equation (8.16) is $\boldsymbol{f_e = f_d}$. We note that when the environment is not flat the equation (8.15) requires the information on $\boldsymbol{\dot{x}_e, \ddot{x}_e}$ which are not available. Controlling with $\boldsymbol{\delta\ddot{x}_e = \delta\dot{x}_e = 0}$ yields the force tracking errors because $\boldsymbol{\dot{x}}$ and $\boldsymbol{\ddot{x}}$ have certain values. In order to eliminate force tracking error, a compensating term $\boldsymbol{\Omega}$ is added to the control input signal such that (8.15) becomes

$$\boldsymbol{m\ddot{e}' + b\dot{e}' + b\Omega = f_e - f_d} \tag{8.17}$$

Here we propose to use the following simple adaptive law for Ω to compensate for $m\ddot{x}'_e + b\dot{x}'_e$:

$$\Omega(t) = \Omega(t-\lambda) + \eta\frac{(f_d(t-\lambda) - f_e(t-\lambda))}{b} \qquad \eta > 0 \tag{8.18}$$

where η is the update rate and λ is the sampling rate.

8.2.3 Stability and Convergence of the adaptive control law

When the adaptive law (8.18) is employed in (8.17) control loop, it is necessary to see whether the new control law remains stable. Combining (8.17) and (8.18) yields

$$\begin{aligned} m\ddot{e}(t)' \;+\; & b\dot{e}(t)' + k_e e(t) + b\Omega(t-\lambda)) + \eta k_e e(t-\lambda) \\ = \; & -(f_d(t) + \eta f_d(t-\lambda)) \end{aligned} \tag{8.19}$$

From the relationship $f_e = k_e(x - x_e)$, we can replace position with force such that $x = x_e + \frac{f_e}{k_e}, \dot{x} = \dot{x}_e + \frac{\dot{f}_e}{k_e}$, and $\ddot{x} = \ddot{x}_e + \frac{\ddot{f}_e}{k_e}$.

Substituting $x, \dot{x}, \ddot{x}$ into (8.19) yields the second order force error equation as

$$\begin{aligned} & m(\ddot{f}_d - \ddot{f}_e) + b(\dot{f}_d - \dot{f}_e) + b\Omega(t-\lambda) + k_e(f_d - f_e) \\ = \; & m\ddot{f}_d + b\dot{f}_d - mk_e\delta\ddot{x}_e - bk_e\delta\dot{x}_e - \eta k_e(f_d(t-\lambda) - f_e(t-\lambda)) \end{aligned} \tag{8.20}$$

Defining $\epsilon(t) = f_d(t) - f_e(t)$ and rewriting (8.20) becomes

$$\begin{aligned} & m\ddot{\epsilon}(t) + b\dot{\epsilon}(t) + k_e(\epsilon(t) + b\Omega(t-\lambda) + \eta\epsilon(t-\lambda)) \\ & = m\ddot{f}_d + b\dot{f}_d - mk_e\delta\ddot{x}_e - bk_e\delta\dot{x}_e \end{aligned} \tag{8.21}$$

Consider $\boldsymbol{k}$ elements of $\boldsymbol{\Omega}$ series.

$$\begin{aligned} \boldsymbol{b\Omega(t-\lambda)} &= \boldsymbol{b\Omega(t-(k-1)\lambda)+\eta k_e\epsilon(t-(k-2)\lambda)} \\ &+ \boldsymbol{\cdots+\eta k_e\epsilon(t-2\lambda)} \end{aligned} \tag{8.22}$$

We assume that the initial $\boldsymbol{\Omega}$ is zero such that $\boldsymbol{\Omega(t-(k-1)\lambda)=0}$. Combining (8.21) with (8.22) yields

$$\begin{aligned} &\boldsymbol{m\ddot{\epsilon}+b\dot{\epsilon}+k_e e+\eta k_e(\epsilon(t-(k-1)\lambda)+\cdots+\epsilon(t-\lambda)} \\ &\qquad\qquad \boldsymbol{=m\ddot{f}_d+b\dot{f}_d-m\delta\ddot{x}_e-b\delta\dot{x}_e} \end{aligned} \tag{8.23}$$

Define the force occurred due to estimation of the environment position as $\boldsymbol{\bar{f}_e = k_e\delta x_e}$. Then (8.23) becomes

$$\begin{aligned} &\boldsymbol{m\ddot{\epsilon}+b\dot{\epsilon}+k_e e+\eta k_e(\epsilon(t-(k-1)\lambda)+\cdots+\epsilon(t-\lambda)} \\ &\qquad\qquad \boldsymbol{=m\ddot{\varepsilon}+b\dot{\varepsilon}} \end{aligned} \tag{8.24}$$

where $\boldsymbol{\varepsilon = f_d - \bar{f}_e}$. Laplace transform of (8.24) is

$$\frac{\boldsymbol{\epsilon(s)}}{\boldsymbol{\mathcal{E}(s)}} = \frac{\boldsymbol{ms^2+bs}}{\boldsymbol{(ms^2+bs+k_e+k_e\eta(e^{-(k-1)\lambda s}+\cdots+e^{-\lambda s}))}} \tag{8.25}$$

When there is a delay $\boldsymbol{\lambda}$, the stability of the system (8.25) can be assured by the characteristic equation of (8.25).

$$\boldsymbol{(ms^2+bs+k_e+k_e\eta(e^{-(k-1)\lambda s}+\cdots+e^{-\lambda s}))=0} \tag{8.26}$$

Since $|e^{-\lambda s}| < 1$ where $\lambda < 1$, the sum of the series can be represented as

$$e^{-(k-1)\lambda s} + \cdots + e^{-\lambda s} = \sum_{n=1}^{k-1} e^{-\lambda s n} = \frac{1}{1 - e^{-\lambda s}} - 1 \tag{8.27}$$

Substituting the sum (8.27) into (8.26) yields

$$(ms^2 + bs + k_e + k_e\eta(\frac{e^{-\lambda s}}{1 - e^{-\lambda s}}))\epsilon(s) = 0 \tag{8.28}$$

When the sampling is fast, the delayed term can be approximated as $e^{-\lambda s} \approx 1 - \lambda s$ by Taylor series expansion. Rearranging (8.28) with Taylor series expansion yields

$$\lambda m s^3 + b\lambda s^2 + k_e\lambda(1 - \eta)s + k_e\eta = 0 \tag{8.29}$$

For stability, the Routh-Hurwitz array is constructed.

$$\begin{array}{lll} s^3 & \lambda m & k_e\lambda(1-\eta) \\ s^2 & b\lambda & \eta k_e \\ s^1 & c_1 & 0 \\ s^0 & c_0 & 0 \end{array} \tag{8.30}$$

And the following conditions are found.

$$c_1 = \frac{bk_e\lambda^2(1-\eta) - \lambda\eta m k_e}{b\lambda} > 0 \;\; and \;\; c_0 = \eta k_e > 0 \tag{8.31}$$

since $m, b, \lambda > 0$. In order for the equation (8.31) to be satisfied with the stability condition, η should be bound ed by

$$0 < \eta < \frac{b\lambda}{b\lambda + m} \tag{8.32}$$

For convergence, we investigate the steady state error $\boldsymbol{E_{ss}(s)}$ of (8.25).

$$\boldsymbol{E_{ss}(s) = lim_{s\rightarrow 0}\, s(\epsilon(s) - \mathcal{E}(s)) = -1} \tag{8.33}$$

when the input is step such that $\boldsymbol{\mathcal{E}(s) = \frac{1}{s}}$. This means that when the input $\boldsymbol{\mathcal{E}}$ is a step the output $\boldsymbol{\epsilon(s)}$ goes to zero so that

$$\boldsymbol{lim_{t\rightarrow\infty}\epsilon(t) = 0} \tag{8.34}$$

Therefore, when $\boldsymbol{t \rightarrow \infty}$, $\boldsymbol{f_e \rightarrow f_d}$. The actual force converges to the desired force.

8.2.4 Compensation for Robot Model Uncertainty

The basic idea of the disturbance rejection algorithm is to use the previous informations to cancel the highly complex uncertainties. We can write the robot dynamic equation (4.2) as

$$\boldsymbol{\hat{D}\ddot{q}(t) + \bar{h}(t) = \tau(t)} \tag{8.35}$$

where $\boldsymbol{\bar{h} = h(t) + \tau_f(t) + \tau_e(t) + (D - \hat{D})\ddot{q}(t)}$.

Select the control law as

$$\boldsymbol{\tau(t) = \hat{D}U(t) + \bar{h}(t)} \tag{8.36}$$

From the equation (8.35), $\boldsymbol{\bar{h}(t)}$ can be rewritten as

$$\boldsymbol{\bar{h}(t) = \tau(t) - \hat{D}(q)\ddot{q}(t)} \tag{8.37}$$

The above idea can not be exactly implemented in practice, however since the values for $\boldsymbol{\tau(t)}$ and $\boldsymbol{\ddot{q}(t)}$ are not available at the time $\boldsymbol{t}$. It is suggested to use the delayed

sample values $\boldsymbol{\tau(t-\lambda)}$ and $\boldsymbol{\ddot{q}(t-\lambda)}$. The new estimation of $\boldsymbol{\bar{h}(t)}$, $\boldsymbol{\hat{\bar{h}}(t)}$ is defined as follows:

$$\boldsymbol{\hat{\bar{h}}(t) = \bar{h}(t-\lambda) = \tau(t-\lambda) - \hat{D}(q)\ddot{q}(t-\lambda)} \tag{8.38}$$

The delayed samples $\boldsymbol{\tau(t-\lambda)}$ and $\boldsymbol{\ddot{q}(t-\lambda)}$ can be easily stored at each sampling period.

Now the control law is

$$\boldsymbol{\tau(t) = \hat{D}U(t) + \hat{\bar{h}}(t)} \tag{8.39}$$

and

$$\boldsymbol{U(t) \cong \ddot{q}(t) = J^{-1}(V - \dot{J}\dot{q})} \tag{8.40}$$

where $\boldsymbol{V = \ddot{x}}$. From the impedance law (8.17), $\boldsymbol{\ddot{x}}$ can be obtained as

$$\boldsymbol{V} = \begin{cases} \boldsymbol{\ddot{X}_e' + M^{-1}(F_d + B\dot{E}' + KE')} & \text{Position Control} \\ \boldsymbol{\ddot{X}_e' + M^{-1}(F_d - F_e + B(\dot{E}' + \Delta\Omega))} & \text{Force Control} \end{cases} \tag{8.41}$$

The control law becomes

$$\boldsymbol{\tau(t) = \hat{D}J^{-1}[V - \dot{J}\dot{q}] + \tau(t-\lambda) - \hat{D}\ddot{q}(t-\lambda)} \tag{8.42}$$

For stability analysis of the robust position control algorithm, let us define the control input error $\boldsymbol{\varepsilon}$ caused from time sample delay from (8.40) and (8.41) as follows:

$$\boldsymbol{\varepsilon = \ddot{q} - U = \hat{D}^{-1}(\hat{\bar{h}}(t) - \bar{h}(t))} \tag{8.43}$$

where $\boldsymbol{\hat{\bar{h}}(t) = h(t-\lambda)}$. From (8.41),(8.42) and (8.35) the control input error $\boldsymbol{\varepsilon}$ at contact space has the following relationship with the closed loop equation

$$\boldsymbol{M\ddot{E}' + B\dot{E}' + K_eE = -MJ\varepsilon - F_d} \tag{8.44}$$

Since $\boldsymbol{m}, \boldsymbol{b},$ and $\boldsymbol{k_e}$ are positive definite the forced error system left hand side of equation (8.44) is asymptotically stable with the result shown in earlier section. Only concern for stability is the boundness of $-\boldsymbol{mJ\varepsilon}$ since the $\boldsymbol{f_d}$ is bounded. The boundness of $-\boldsymbol{mJ\varepsilon}$ can be shown in same manner of the stability proof shown in [13].

It has been demonstrated in these studies that controller design approach such as (8.42) works very well for the PUMA robot manipulator. With the aid of DSP hardware, $\boldsymbol{\lambda}$ can be made significantly small in practice. For example, our experiments with PUMA 560 arm have achieved a sampling rate of 200 $\boldsymbol{Hz}$. Consequently, the control system (8.42) is stable and the performance is very close to that of the ideal case. Further information regarding practical controller implementation can be found in [6, 89].

8.2.5 Simulations

In this section, the proposed control algorithm is tested by simulating the tracking performance with a three link rotary robot manipulator whose parameters are taken from the first three links of Puma 560 arm. The nominal system parameters are used as the basis in forming the robot model $\hat{\boldsymbol{D}}(\boldsymbol{q})$ and $\hat{\boldsymbol{H}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. Model uncertainties included a 10 Kg mechanical tool attached to the third link, Coulomb friction and viscous friction forces $\boldsymbol{\tau_f}(\dot{\boldsymbol{q}})$ added to each joint where $\boldsymbol{\tau_f}(\dot{\boldsymbol{q}}) = \mathbf{5.0}\boldsymbol{sgn}(\dot{\boldsymbol{q}}) + \mathbf{8.0}(\dot{\boldsymbol{q}})$.

The performances of the proposed schemes are tested by tracking a flat wall. The controller gains are selected as $\boldsymbol{M} = \boldsymbol{I}, \boldsymbol{B} = \boldsymbol{diag}[\mathbf{500}, \mathbf{40}, \mathbf{500}]$ and $\boldsymbol{K} =$

$\boldsymbol{diag}[\mathbf{300, 100, 300}]$ which give over-damped motions at the three joints. At contact, $\boldsymbol{K}$ becomes $\boldsymbol{K} = \boldsymbol{diag}[\mathbf{0, 100, 100}]$ such that the gain in force controllable direction, in this case the $\boldsymbol{x}$ axis, is set to zero. In order to show the robustness to unknown environment stiffness of the proposed scheme, we tested the system performance for abruptly changing environment stiffnesses with the stiffness profile as

$$\boldsymbol{k_e} = \begin{cases} \mathbf{40000}(\boldsymbol{N/m}) & \mathbf{0} \leq \boldsymbol{t} < \mathbf{3} \\ \mathbf{80000}(\boldsymbol{N/m}) & \mathbf{3} \leq \boldsymbol{t} < \mathbf{6} \end{cases} \tag{8.45}$$

which is considered as a stiff environment. In this task, we test the different values of inaccurate estimation of environment position such as $\boldsymbol{\delta x_e} = \mathbf{0.03, 0.06}$ and $\mathbf{0.1}\boldsymbol{m}$. The resulting force trackings are excellent as shown in Figure 8.9. We note that the larger estimation of $\boldsymbol{\delta x_e}$ yields the shorter contact time but the larger force overshoot at initial contact.

Another task is tested for the robot to follow the flat wall with a large triangle shaped crack in the middle. The initial controller gains are selected as $\boldsymbol{M} = \boldsymbol{I}, \boldsymbol{B} = \boldsymbol{diag}[\mathbf{440, 40, 440}]$ and $\boldsymbol{K} = \boldsymbol{diag}[\mathbf{300, 100, 300}]$. The environment position is not correctly estimated but specified inside the environment to guarantee the contact. Since the deepest point of the crack is $\mathbf{0.02}\boldsymbol{m}$ from the surface the user specified $\boldsymbol{\delta x_e}$ is selected at $\mathbf{0.03}\boldsymbol{m}$ such that the estimated environment position $\boldsymbol{x_e'} = \boldsymbol{x_e} + \mathbf{0.03}\boldsymbol{m}$. Since we assume that the environment profile is not available, $\boldsymbol{\delta x_e}$ is constant and $\boldsymbol{\delta \dot{x}_e}, \boldsymbol{\delta \ddot{x}_e}$ are assumed to be zero. In this case, the control behavior is controlled by (11) and the plots are shown in Figure 8.10. Even though the force tracking is

excellent when the environment is flat, the force tracking is not effective with notable errors when the robot follows the crack position. The deviation in force tracking in Figure 8.10 results from not specifying $\dot{\boldsymbol{x}}_e$ and $\ddot{\boldsymbol{x}}_e$ in the control law because it is assumed to be unknown. The approximate value of force deviation can be calculated from $\boldsymbol{\delta f = f_d - f_e \approx b * \dot{x}_e}$ in Figure 8.10 since $\boldsymbol{m * \ddot{x}_e \ll b * \dot{x}_e}$. In our simulation, since $\boldsymbol{b = 440}$ are used and $\boldsymbol{\dot{x}_e = 0.02}$ is assumed to be unknown the $\boldsymbol{\delta f \approx 8.8(N)}$ which matches the force deviation in the plot. Therefore, it is better to specify more information on the environment. In most practical case, the environment is not assumed to be known so that we adopt the adaptive idea (13) to estimate $\dot{\boldsymbol{x}}_e$ which is a major contribution term to the force deviation. The force tracking plot using the adaptive law is shown in Figure 8.11. The update rate is selected as $\boldsymbol{\eta = 0.3 < \frac{440}{80000*0.005} = 1.1}$ which is satisfied the bound condition of $\boldsymbol{\eta}$. The corresponding position tracking plot is shown in Figure 8.12. The performance of force tracking is excellent. Another simulation is carried out for the sinusoidally shaped environment. The force and position tracking are shown in Figure 8.13 and 8.14, respectively. In this case, the update rate $\boldsymbol{\eta}$ is set to $\boldsymbol{0.2 < \frac{600}{80000*0.005} = 1.5}$ and the controller gains $\boldsymbol{B = diag[600, 20, 20], K = diag[500, 100, 100]}$ are used in free space and $\boldsymbol{K = diag[0, 100, 100]}$ is used in contact space. Again, the tracking performance is excellent.

8.2.6 Experimental Results

To demonstrate the performance of the proposed algorithm, experiments were conducted using the UCD Robotics Research Laboratory (RRL) experimental testbed consisting of a PUMA 560 industrial manipulator fitted with a JR3 wrist force sensor. The manipulator is controlled by a UCD RRL-designed controller consisting of: 1) a 486 personal computer; 2) electronics to read the voltages from the joint potentiometers, to read each joint's digital encoder signals, and to output control signals to the Unimate motor voltage amplifiers; 3) a force sensor interface card; 4) a TMS320C40 DSP board; and 5) the control software.

Single arm experiments were first conducted on the PUMA manipulator operating in both free space and in contact with a rigid surface using conventional impedance control which resulted in the impedance chosen as follows: $\boldsymbol{M} = \mathit{diag}\ \{\mathbf{50\ 50\ 50\ 1\ 1\ 1}\}$, $\boldsymbol{B} = \mathit{diag}\ \{\mathbf{3142\ 3142\ 1571\ 31.4\ 31.4\ 31.4}\}$, and $\boldsymbol{K} = \mathit{diag}\ \{\mathbf{31583\ 31583\ 11844\ 237\ 237}\}$. The inertia matrix was chosen to satisfy a lower bound requirement which results from digital implementation during constrained operation as shown in [94]. The preceding impedance yields good trajectory tracking and force step response performance. Damping in the z direction was reduced to achieve faster response in the force tracking direction.

Two experiments were conducted each consisting of moving in the negative z direction onto a surface, and then moving 25cm along the surface in the x direction in 15 seconds with $\boldsymbol{f_d} = [\mathbf{0\ 0\ -20}\boldsymbol{N}\ \mathbf{0\ 0\ 0}]$. The first surface is a flat hard surface

and the second surface is stiff curved sheet of galvanized steel. The exact location and stiffness of each surface is unknown. For the flat hard surface, η was chosen as 0.001 and for the curved stiff surface η was increased to 0.02 since the surface is more compliant. The results of the experiments are shown in Figures8.15 and 9. Initially, the end effector is commanded to move to a point beyond the surface. Upon contact with the surface, the initial overshoot is approximately 60% and the force settles to -20N in approximately two seconds. During the motion the force is controlled to within $\pm$20%. The experiments demonstrate that the algorithm exhibits good force tracking in the presence of environmental uncertainty.

8.2.7 Discussion

A simple robust force control scheme of robot manipulator is presented in this chapter. The robust position control algorithm is used to compensate for the uncertainties in the robot dynamics. The new impedance controller realized by setting the stiffness gain to zero performs very well under the condition of inaccurate estimation of the environment position and unknown stiffness. Even though the proposed impedance function is so simple, when the environment position is unknown, the estimation of velocity profile of the environment position is necessary to achieve the better force tracking. One simple solution is suggested to use an adaptive control concept that adjusts the environment velocity profile by a force error. Simulation results prove that the proposed controller is excellent to achieve the best force/position track-

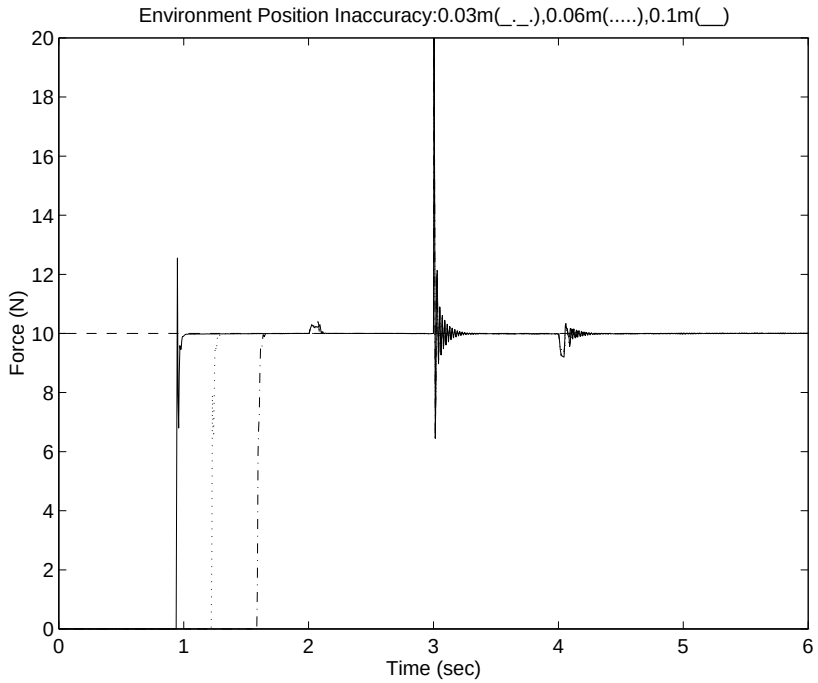


Figure 8.9: Force Tracking with different environment position inaccuracies δx_e

ing under the situation that the environment is not known. The proposed controller works well when the environment stiffness is abruptly changed.

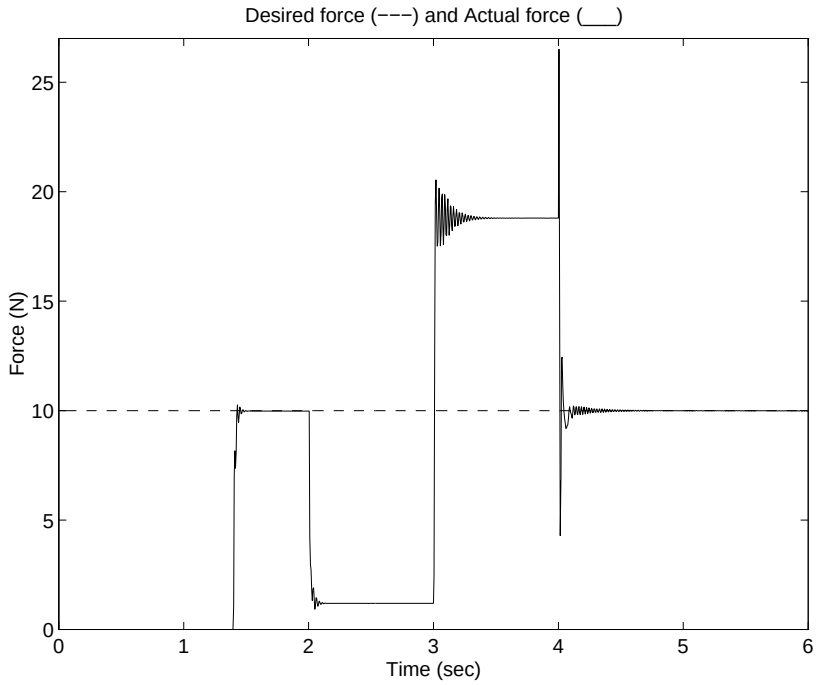


Figure 8.10: Force Tracking when $\dot{\boldsymbol{x}}_{\boldsymbol{e}} = \mathbf{0}$

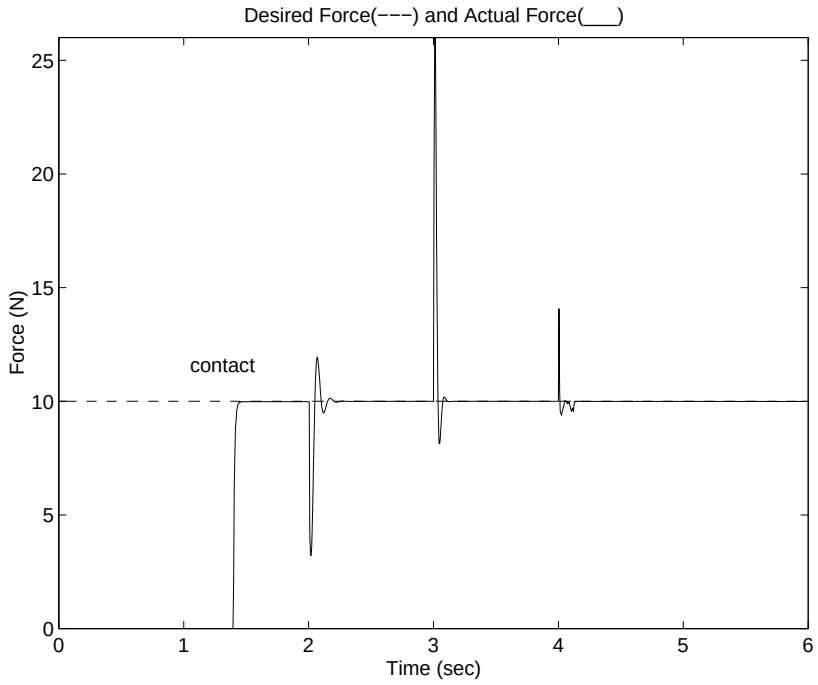


Figure 8.11: Force Tracking with the adaptive law

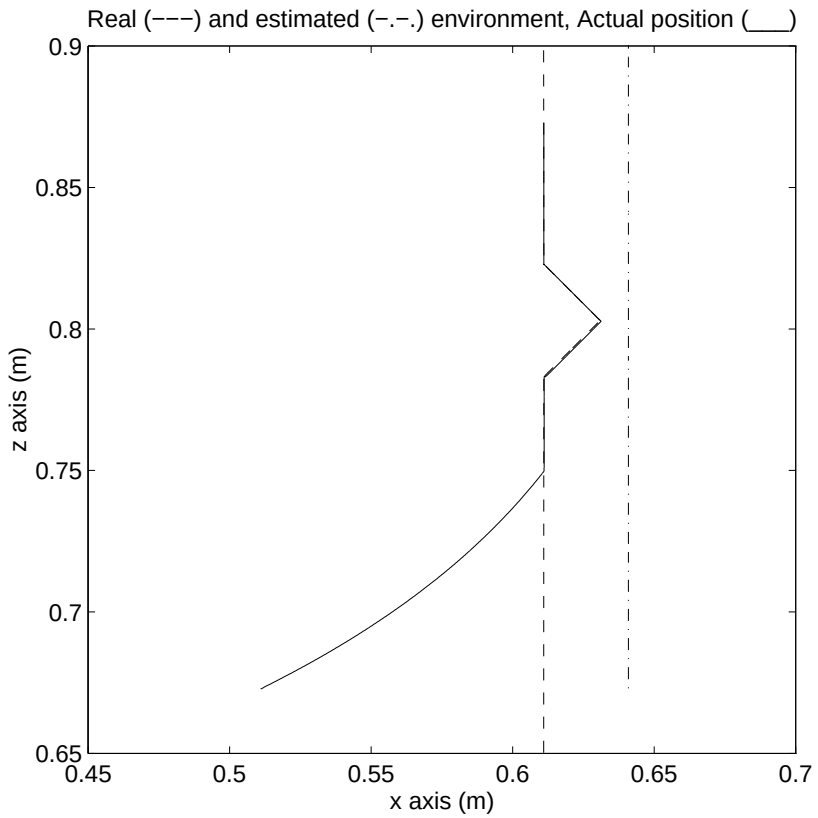


Figure 8.12: Position Tracking with the adaptive law

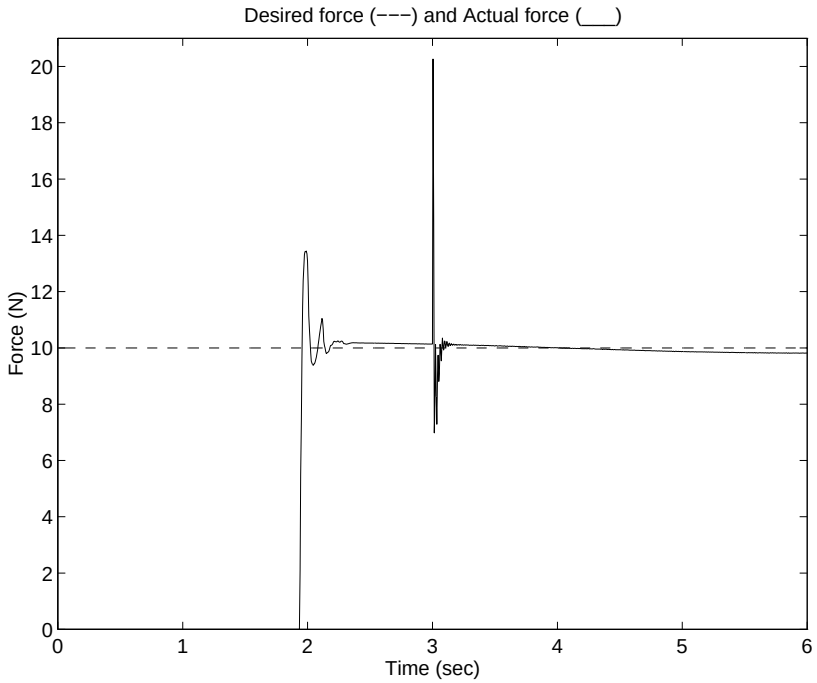


Figure 8.13: Force Tracking with the adaptive law

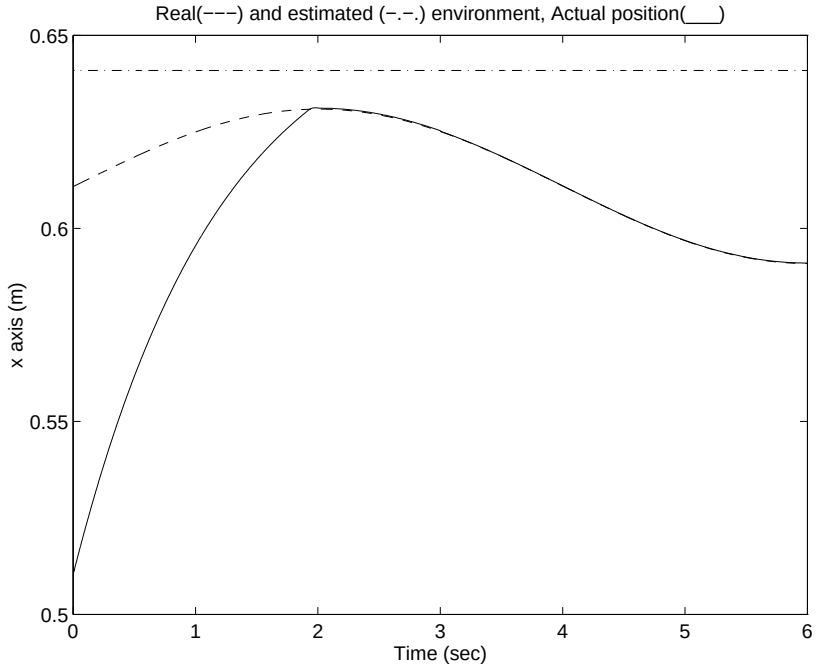


Figure 8.14: Position Tracking with the adaptive law

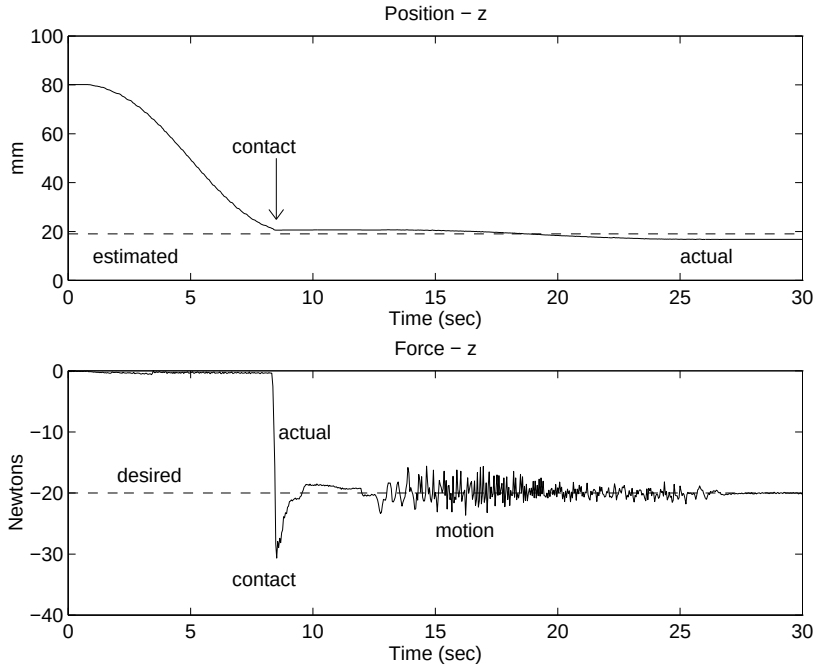


Figure 8.15: Experiment - Flat Surface

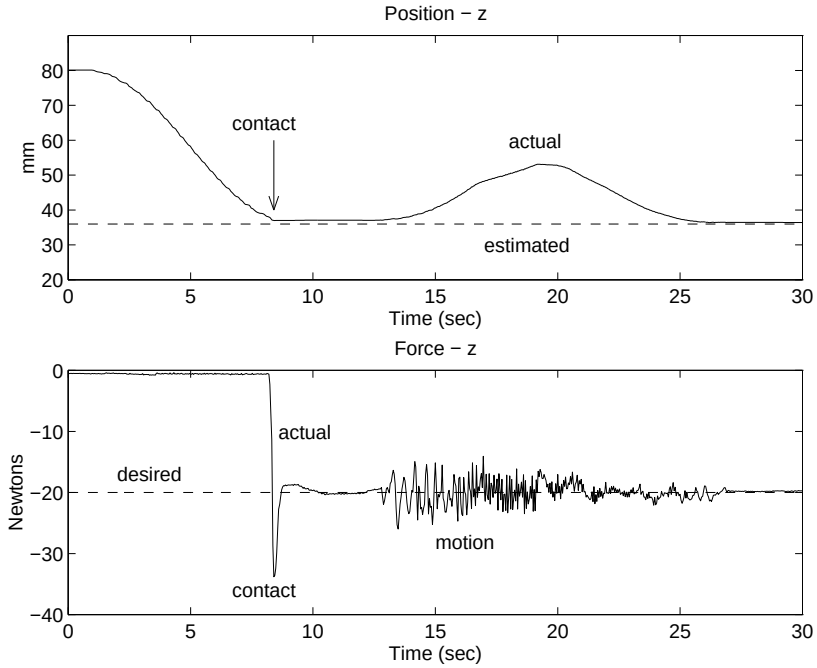


Figure 8.16: Experiment - Curved Surface

Conclusions

In this thesis, the feasibility studies of using neural network as a controller for robot manipulators have been investigated. Neural network controllers are used as auxiliary compensators that compensate for any nonlinear uncertainties in robot dynamics when partial robot dynamic equations are available. As a nonlinear adaptive element neural network controller has shown effective compensation in joint position control of robot manipulators. Based on on-line control strategy two main control schemes have been proposed : one is an auxiliary control type whose neural compensator compensates at torque or control input and the other is an inverse control type whose controller compensates at the reference trajectory. Performances of two schemes are compared in Cartesian space. It is found that an inverse control type that compensates at reference trajectory not only performs better but is robust to feedback controller gain variations. The reason of the better performance of compensating at input trajectory rather than at torque is that the magnitude of the neural network input signal level is smaller.

More interestingly, when no knowledge of robot dynamic equations is given neural network also performs very effectively with a PD feedback controller. In order for robots to perform accurate position control in non-model based neural network has to compensate for both robot dynamics and uncertainties.

A summary for research achievements on position control of robot manipulators using NN is as follows:

- Model Based Approach
  - Partial dynamic model is required.
  - Feedforward and feedback compensation methods were compared.
  - Prefilter type NN control scheme has been proposed.
  - Reference compensation technique has been proposed.
  - The RCT scheme has the best performance as well as simpler implementation.
- Non-model Based Approach
  - Dynamic model is not required at all.
  - Comparison studies between FEL and RCT has been conducted.
  - The performance of RCT is much better.
  - The RCT scheme has the robustness to feedback controller gain variations.
- Cartesian Space Approach
  - The performance of compensating at input trajectory is the best.
  - The lower the signal level of NN output is the better the performance is.

Finally, neural network was applied to force control of robot manipulator. In order for a single neural network to compensate for all the uncertainties such as uncertainties in robot dynamics, uncertainties in environment position and stiffness new impedance function and new training signal have been developed.

Studies on neural network application to control robot manipulators have shown that neural network is one good candidate for a nonlinear controller.

As an alternative solution to NN robust position control has been used to compensate for uncertainties in robot dynamics. So new impedance function was used with robust position control and implemented on PUMA 560. It has demonstrated effective force tracking under unknown environment characteristic.

A summary for research achievements on force control of robot manipulators.

- Identification method of environment stiffness has been developed.
- Non linear impedance function has been developed.
- Impedance control using NN was developed.
  - Torque-based approach
  - Position-based approach
- New impedance function that has the capability of tracking desired force under unknown flat environment was developed.
- Simple robust adaptive impedance function with robust position control that

provides a solution for all the problems in force control. It compensates for robot dynamic uncertainties and it has force tracking capability under any unknown surface and unknown stiffness of environment. Experimental works have proven its performance on PUMA 560 robot manipulators.

Future Researches

In this thesis, many parts require more studies and developments. In future, the following extended researches will be investigated:

- Practical implementation of neural network control on robot manipulators
- Develop a unified neural network controller on PC for general purpose
- Apply proposed NN control to other system such as mobile robot, flexible robot, and crane control, etc.
- Theoretical investigations inside neural network processing
- Stability analysis of NN control system
- Combine neural network with fuzzy control algorithm or generic algorithm to improve performances in position and force control of robot manipulators.
- Application of the proposed force control to manufacturing process such as grinding, deburring, and polishing, etc.
- Man-machine interface using neural networks

Appendix A

Adaptive Robot Control

Here, one adaptive control of robot manipulator proposed by ***Craig et. al*** is presented. The nonlinear parameters such as link mass, payload, and friction are estimated on-line. From the robot position control closed loop equation we can rewrite the error equation as

$$\ddot{\boldsymbol{E}} + \boldsymbol{K_D}\dot{\boldsymbol{E}} + \boldsymbol{K_P}\boldsymbol{E} = \hat{\boldsymbol{D}}^{-1}\boldsymbol{W}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{\Phi} \tag{A.1}$$

where $\boldsymbol{W}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{\Delta D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{\Delta H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\tau_f}(\dot{\boldsymbol{q}})$ and $\boldsymbol{\Phi}$ is a parameter error vector.

In order to derive the adaptive law we set the Lyapunov as

$$\boldsymbol{L} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{X} + \boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Phi} \tag{A.2}$$

The derivative of (B.2) yields

$$\dot{\boldsymbol{L}} = \boldsymbol{x}^{\boldsymbol{T}}(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}) + \boldsymbol{B}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{X} + \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B} + 2\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Lambda}^{-1}\dot{\boldsymbol{\Phi}}$$

$$= \quad -\boldsymbol{X Q X^T} + \boldsymbol{2\Phi^T(W^T \hat{D}^{-1} S} \tag{A.3}$$

where $\boldsymbol{S = \dot{E} + \alpha E.}$ In order to have negative definite of $\dot{\boldsymbol{L}}$ $\dot{\boldsymbol{\Phi}}$ is chosen as

$$\dot{\boldsymbol{\Phi}} = -\boldsymbol{\Lambda W^T \hat{D}^{-1} S} \tag{A.4}$$

Then $\dot{\boldsymbol{L}} < \boldsymbol{0}$ which satisfy the global stability. Since the $\boldsymbol{\Phi}$ is a parameter error such that $\boldsymbol{\Phi = \Theta - \hat{\Theta}}$ the parameter updating law can be obtained as follows:

$$\dot{\hat{\boldsymbol{\Theta}}} = \boldsymbol{\Lambda W^T \hat{D}^{-1} S} \tag{A.5}$$

Based on our robot model parameters to estimate are

$$\boldsymbol{\Theta = [M_1 M_2 M_3 K_{11} K_{12} K_{13} K_{21} K_{22} K_{23}]} \tag{A.6}$$

The corresponding $\boldsymbol{W}$ matrix is

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w_{11}} & \boldsymbol{w_{12}} & \boldsymbol{w_{13}} & \boldsymbol{w_{14}} & \boldsymbol{w_{15}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{w_{21}} & \boldsymbol{w_{22}} & \boldsymbol{w_{23}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{w_{26}} & \boldsymbol{w_{27}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{w_{31}} & \boldsymbol{w_{32}} & \boldsymbol{w_{33}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{w_{38}} & \boldsymbol{w_{39}} \end{bmatrix} \tag{A.7}$$

where $\boldsymbol{w_{11} = 0}$

$\boldsymbol{w_{12} = \frac{1}{3} L_2^2 C_2^2 \ddot{q}_1 - \frac{2}{3} L_2^2 C_2 S_2 \dot{q}_1 \dot{q}_2 + \frac{1}{2} L^2 g C_2}$

$\boldsymbol{w_{13} = (\frac{1}{3} L_2^2 C_{23}^2 + L_2^2 C_2^2 + L_2 L_3 C_2 C_{23}) \ddot{q}_1 + (-\frac{2}{3} L_3^2 S_{23} C_{23} - L_2 L_3 C_2 S_{23} -}$

$\boldsymbol{L_2 L_3 S_2 C_{23} - 2 L_2^2 S_2 C_2) \dot{q}_1 \dot{q}_2 + (-\frac{2}{3} L_3^2 S_{23} C_{23} - L_2 L_3 C_2 S_{23}) \dot{q}_1 \dot{q}_3 + \frac{1}{2} L_3 g C_{23}}$

$\boldsymbol{w_{14} = \dot{q}_1}$

$\boldsymbol{w_{15} = sqn(\dot{q}_1)}$

$\boldsymbol{w_{21} = 0}$

$$w_{22} = \tfrac{1}{3}L_2^2\ddot{q}_2 + \tfrac{1}{3}L_2^2C_2S_2\dot{q}_1^2 + \tfrac{1}{2}L_2gC_2i$$

$$w_{23} = (\tfrac{1}{3}L_3^2+L_2L_3C_3+L_2^2)\ddot{q}_2+(\tfrac{1}{3}L_3^2+\tfrac{1}{2}L_2L_3C_3)\ddot{q}_3+(\tfrac{1}{3}L_3^2S_{23}C_{23}+\tfrac{1}{2}L_2L_3(C_2S_{23}+$$

$$S_2C_{23}) + L_2^2C_2S_2)\dot{q}_1^2 - L_2L_3S_3\dot{q}_2^2\dot{q}_3^2 - \tfrac{1}{2}L_2L_3S_3\dot{q}_3^2 + g(\tfrac{1}{2}L_2C_{23} + L_2C_2)$$

$$w_{26} = \dot{q}_2$$

$$w_{27} = sgn(\dot{q}_2)$$

$$w_{31} = 0$$

$$w_{32} = 0$$

$$w_{33} = (\tfrac{1}{3}L_3^2+\tfrac{1}{2}L_2L_3C_3)\ddot{q}_2+\tfrac{1}{3}L_3^2\ddot{q}_3+(\tfrac{1}{3}L_3^2C_{23}S_{23}+\tfrac{1}{2}L^2L_3C_2S_{23})\dot{q}_1^2+\tfrac{1}{2}L_2L_3S_3\dot{q}_2^2+$$

$$\tfrac{1}{2}L_3gC_{23}$$

$$w_{38} = \dot{q}_3$$

$$w_{39} = sgn(\dot{q}_3)$$

Appendix B

Three-Link Robot Manipulator Model

The robot parameters used for simulations:

Table B.1: Robot Model Parameters

	Link length($\boldsymbol{m}$)	Link mass($\boldsymbol{Kg}$)
Joint **1**	**0.673**	**30.0**
Joint **2**	**0.432**	**15.91**
Joint **3**	**0.432**	**11.36**

For simplicity, we used simple notations for joint angles.

$$\begin{aligned} \boldsymbol{S_1} &= \boldsymbol{sin}(q_1), C_1 = \boldsymbol{cos}(q_1), \\ \boldsymbol{S_{12}} &= \boldsymbol{sin}(q_1 + q_2), C_{12} = \boldsymbol{cos}(q_1 + q_2), \\ \boldsymbol{S_{23}} &= \boldsymbol{sin}(q_2 + q_3), C_{23} = \boldsymbol{cos}(q_2 + q_3) \end{aligned} \tag{B.1}$$

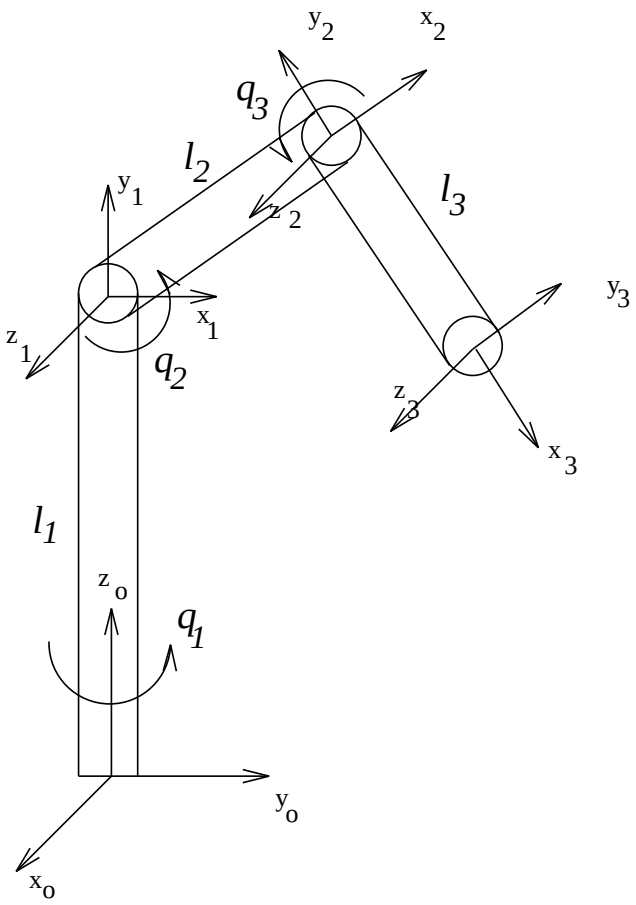


Figure B.1: Three-Link Rotary Robot Manipulator

The forward kinematic equations for three-link robot are

$$\begin{aligned} x &= C_1(L_3C_{23} + L_2C_2) \\ y &= S_1(L_3C_{23} + L_2C_2) \\ z &= L_1 + L_2S_2 - L_3S_{23} \end{aligned} \tag{B.2}$$

The inverse kinematic equations for three-link robot are

$$a = \frac{x^2 + y^2 + (z - l_1)^2 - l_2^2 - l_3^2}{2l_2l_3} \tag{B.3}$$

$$\begin{aligned} \theta_3 &= atan2(\sqrt{1 - a^2}, a) \\ b &= atan2(l_3sin(\theta_3), l_2 + l_3cos(\theta_3)) \\ \theta_2 &= atan2(z - l_1, \sqrt{y^2 + x^2}) + b, \qquad \theta_3 < 0 \\ \theta_2 &= atan2(z - l_1, \sqrt{y^2 + x^2}) - b, \qquad \theta_3 > 0 \\ \theta_1 &= atan2(y, x) \end{aligned} \tag{B.4}$$

The Jacobian matrix is

$$J = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{bmatrix} \tag{B.5}$$

where

$$\begin{aligned} j_{11} &= -(L_2cos(q_2) + L_3cos(q_2 + q_3))sin(q_1) \\ j_{12} &= -(L_2sin(q_2) + L_3sin(q_2 + q_3))cos(q_1) \end{aligned}$$

$$
\begin{aligned}
j_{13} &= -L_3 cos(q_1) sin(q_2 + q_3) \\
j_{21} &= (L_3 cos(q_2) + L_3 cos(q_2 + q_3)) cos(q_1) \\
j_{22} &= -(L_2 sin(q_2) + L_3 sin(q_2 + q_3)) sin(q_1) \\
j_{23} &= -L_3 sin(q_1) sin(q_2 + q_3) \\
j_{31} &= 0.0 \\
j_{32} &= (L_2 cos(q_2) + L_3 cos(q_2 + q_3)) \\
j_{33} &= L_3 cos(q_2 + q_3) \qquad \text{(B.6)}
\end{aligned}
$$

The derivative of Jacobian $\dot{J}$ is

$$
\begin{aligned}
\dot{j}_{11} &= -(L_2 cos(q_2) + L_3 cos(q_2 + q_3)) cos(q_1) \dot{q}_1 \qquad \text{(B.7)} \\
&\quad + sin(q_1)(L_2 sin(q_2) \dot{q}_2 + L_3 sin(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)) \qquad \text{(B.8)} \\
\dot{j}_{12} &= (L_2 sin(q_2) + L_3 sin(q_2 + q_3)) sin(q_1) \dot{q}_1 \qquad \text{(B.9)} \\
&\quad - cos(q_1)(L_2 cos(q_2) \dot{q}_2 + L_3 cos(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)) \qquad \text{(B.10)} \\
\dot{j}_{13} &= L_3 sin(q_2 + q_3) sin(q_1) \dot{q}_1 - L_3 cos(q_1) cos(q_2 + q_3)(\dot{q}_2 + \dot{q}_3) \qquad \text{(B.11)} \\
\dot{j}_{21} &= -(L_2 cos(q_2) + L_3 cos(q_2 + q_3)) sin(q_1) \dot{q}_1 \qquad \text{(B.12)} \\
&\quad - cos(q_1)(L_2 sin(q_2) \dot{q}_2 + L_3 sin(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)) \qquad \text{(B.13)} \\
\dot{j}_{22} &= -(L_2 sin(q_2) + L_3 sin(q_2 + q_3)) cos(q_1) \dot{q}_1 \qquad \text{(B.14)} \\
&\quad - sin(q_1)(L_2 cos(q_2) \dot{q}_2 + L_3 cos(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)) \qquad \text{(B.15)} \\
\dot{j}_{23} &= -L_3 sin(q_2 + q_3) cos(q_1) \dot{q}_1 - L_3 sin(q_1) cos(q_2 + q_3)(\dot{q}_2 + \dot{q}_3) \qquad \text{(B.16)}
\end{aligned}
$$

$$\dot{j}_{31} = 0.0 \tag{B.17}$$

$$\dot{j}_{32} = -L_2 sin(q_2)\dot{q}_2 - L_3 sin(q_2+q_3)(\dot{q}_2+\dot{q}_3) \tag{B.18}$$

$$\dot{j}_{33} = -L_3 sin(q_2+q_3)(\dot{q}_2+\dot{q}_3) \tag{B.19}$$

(B.20)

The dynamic equations are represented as the following:

$$\begin{aligned}
\tau_1 &= (\frac{1}{3}M_2L_2^2C_2^2 + \frac{1}{3}M_3L_3^2C_{23}^2 + M_3L_2^2C_2^2 + M_3L_2L_3C_2C_{23})\ddot{q}_1 \\
&+ (-\frac{2}{3}M_2L_2^2C_2S_2 - \frac{2}{3}M_3L_3^2C_{23}S_{23})\dot{q}_1\dot{q}_2 \\
&- (M_3L_2L_3(C_2S_{23}+S_2C_{23}) - 2M_3L_2^2C_2S_2)\dot{q}_1\dot{q}_2 \\
&+ (-\frac{2}{3}M_3L_3^2C_{23}S_{23} - M_3L_2L_3C_2S_{23})\dot{q}_1\dot{q}_3 \qquad \text{(B.21)} \\
\tau_2 &= (\frac{1}{3}M_2L_2^2 + \frac{1}{3}M_3L_3^2 + M_3L_2L_3C_3 + M_3L_2^2)\ddot{q}_2 + (\frac{1}{3}M_3L_3^2 + \frac{1}{2}M_3L_2L_3C_3)\ddot{q}_3 \\
&+ \frac{1}{3}M_2L_2^2C_2S_2 + \frac{1}{3}M_3L_2L_3C_{23}S_{23} + \frac{1}{2}M_3L_2L_3(S_2C_{23}+C_2S_{23}) \\
&+ M_3L_2^2C_2S_2)\dot{q}_1^2 - M_3L_2L_3S_3\dot{q}_2\dot{q}_3 - \frac{1}{2}M_3L_2L_3S_3\ddot{q}_3^2 + \frac{1}{2}M_3L_2gC_2 \\
&+ M_3g(\frac{1}{2}L_3C_{23} + L_2C_2) \qquad \text{(B.22)} \\
\tau_3 &= (\frac{1}{3}M_3L_3^2 + \frac{1}{2}M_3L_2L_3C_3)\ddot{q}_2 + \frac{1}{3}M_3L_2L_3^2\ddot{q}_3 \\
&+ (\frac{1}{3}M_3L_3^2C_{23}S_{23} + \frac{1}{2}M_3L_2L_3C_2S_{23})\ddot{q}_1^2 \\
&+ \frac{1}{2}M_3L_2L_3S_3\dot{q}_2^2 - \frac{1}{2}M_3L_3gC_{23} \qquad \text{(B.23)}
\end{aligned}$$

Bibliography

[1] J. J. Craig, *Introduction to Robotics:mechanics and control*, Addison Wesley, 1986.

[2] Mark W. Spong and M. Vidyasagar, *Robot Dynamics and Control*, John Wiley and Sons, 1989.

[3] H. Asada and J. J. Slotine, *Robot Analysis and Control*, John Wiley and Sons, 1991.

[4] J. J. Craig, Ping Hsu, and S. S. Sastry, "Adaptive control of mechanical manipulators", *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 190–195, 1986.

[5] T.C. Hsia, "Adaptive control of robot manipulator - a review", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 183–190, 1986.

[6] T.C. Hsia, "Decoupled robust joint control of robot manipulators", *Proc. of 2nd IEEE Workshop on Advanced Motion Control*, pp. 1–8, Nagoya, Japan, 1992.

[7] C. Abdallah, D. Dawson, P. Dorato, and M. Jamshidi, "Survey of robust control for rigid robots", *IEEE Control System Magazine*, pp. 24–30, 1991.

[8] J.J. Slotine and W. Li, *Applied Nonlinear Control*, Prentice Hall, 1991.

[9] K. Narendra and K. Parthasarathy, *Stable Adaptive Systems*, Prentice Hall, 1989.

[10] T.C. Hsia, "A new technique for robust control of servo systems", *IEEE Transactions on Industrial Electronics*, vol. 36, pp. 1–7, 1989.

[11] T.C. Hsia and L. S. Gao, "Robot manipulator control using decentralized linear time-invariant time delayed joint controllers", *IEEE Conference on Robotics and Automation*, pp. 2070–2075, 1990.

[12] T.C. Hsia, T. A. Lasky, and Z. Y. Guo, "Robust independent joint controller design for industrial robot manipulators", *IEEE Transactions on Industrial Electronics*, vol. 38, pp. 21–25, 1991.

[13] T.C. Hsia, "Simple robust schemes for cartesian space control of robot manipulators", *International Journal of Robotics and Automation*, pp. 167–174, 1994.

[14] T.C. Hsia and S. Jung, "A simple alternative to neural network control scheme for robot manipulator", *IEEE Transactions on Industrial Electronics*, pp. 414–416, August, 1995.

[15] D. Psaltis, A. Sideris, and A. Yamamura, "A multilayered neural network controllers", *IEEE Control System Magazine*, pp. 17–21, 1986.

[16] V.C. Chen and Y. H Pao, "Learning control with neural network", *IEEE Magazine*, pp. 1448–1453, 1989.

[17] W. Li and J.J. Slotine, "Neural network control of unknown nonlinear systems", *Proc. of American Control Conference*, pp. 1136–1141, 1989.

[18] F.C. Chen, "Back propagation neural networks for nonlinear self-tuning adaptive control", *IEEE Control System Magazine*, vol. 10, pp. 44–48, 1990.

[19] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks*, vol. 1, pp. 4–27, 1990.

[20] K. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks", *IEEE Trans. on Neural Networks*, vol. 2, pp. 252–262, 1991.

[21] K. Berns, R. Dillmann, and R. Hofstetter, "An application of a backpropagation network for the control of a tracking behavior", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 2426–2431, 1991.

[22] T. Fukuda and T. Shibata, "Theory and applications of neural networks for industrial control systems", *IEEE Trans. on Industrial Electronics*, vol. 39, pp. 472–489, 1992.

[23] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for nonlinear system control", *IJCNN*, pp. 315–319, 1992.

[24] C.C. Ku and K.Y. Lee, "System identification and control using diagonal recurrent neural networks", *Proc. of American Control Conference*, pp. 545–549, 1992.

[25] K. Narendra and K. Parthasarathy, "Control of nonlinear dynamical systems using neural networks: Controllability and stabilization", *IEEE Trans. on Neural Networks*, vol. 4, pp. 192–206, 1993.

[26] J.G. Kuschewski, S. Hui, and S.H. Zak, "Application of feedforward neural networks to dynamical system identification and control", *IEEE Trans. on Control Systems Technology*, vol. 1, pp. 37–49, 1993.

[27] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks", *IEEE Trans. on Neural Networks*, vol. 5, pp. 279–297, 1994.

[28] W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*, 1991.

[29] W. T. Miller III, F. H. Glanz, and L. G. Kraft III, "Application of a general learning algorithm to the control of robotic manipulators", *The international journal of Robotics Research*, vol. 6, pp. 84–98, 1990.

[30] M. Kawato, K. Kurukkawa, and R. Suzuki, "A hierarchical neural network model for learning of voluntary movement", *Biological Cybernetics*, pp. 169–185, 1987.

[31] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback error learning neural network for trajectory control of a robotic manipulator", *IEEE Trans. on Neural Networks*, vol. 1, pp. 251–265, 1988.

[32] D. Katic, "Using neural network model for learning control of manipulation robot", *Proc. of Intelligent Autonomous Systems*, pp. 424–433, 1989.

[33] D. A. Handelman, S. H. Lane, and J. J. Gelfand, "Integrating neural networks and knowledge-based systems for robotic control", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 1454–1460, 1989.

[34] H. Gomi and M. Kawato, "Learning control for a closed loop system using feedback error learning", *Proc. of the IEEE International Conference on Decision and Control*, pp. 3289–3294, 1990.

[35] T. Yabuta and T. Yamada, "Possibility of neural networks controller for robot manipulator", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 1686–1691, 1990.

[36] B. Horne, M. Jamshidi, and N. Vadiee, "Neural networks in robotics : A survey", *Journal of Intelligent and Robotic Systems*, vol. 3, pp. 61–66, 1990.

[37] W. T. Miller III, R. P. Hewes, F. H. Glanz, and L. G. Kraft III, "Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller", *IEEE Trans. on Robotics and Automation*, vol. 6, pp. 1–9, 1990.

[38] S. Liu and H. Asada, "Teaching and learning of compliance using neural nets:representation and generation of nonlinear compliance", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 1237–1244, 1990.

[39] J. J. Helferty and S. Biswas, "Neuromorphic control of robotics manipulators", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 2436–2441, 1991.

[40] T. Yabuta and T. Yamada, "Learning control using neural networks", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 740–745, 1991.

[41] D. F. Bassi and G. A. Bekey, "Decomposition of neural network models of robot dynamics : a feasibility study", *Simulation and AI*, pp. 8–13, 1991.

[42] T. Ozaky, T. Suzuki, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Trajectory control of robotic manipulators using neural networks", *IEEE Trans. on Industrial Electronics*, vol. 38, pp. 195–202, 1991.

[43] A. Ishiguro, T. Furuhashii, S. Okuma, and Y. Uchikawa, "A neural network compensator for uncertainties of robot manipulator", *IEEE Trans. on Industrial Electronics*, vol. 39, pp. 61–66, December, 1992.

[44] N.S. Gehlot and P.J. Alsina, "A comparison control strategies of robotic manipulators using neural networks", *Proc. of International Conference on Industrial Electronics*, pp. 688–693, 1992.

[45] M. Kemal Ciliz and Can Isik, "Stability and convergence of neurogic model based robotic controllers", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 2051–2056, 1992.

[46] D. Katic and M. Vukobratovic, "Decomposed connectionist architecture for fast and robust learning of robot dynamics", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 2064–2069, 1992.

[47] S. Liu and H. Asada, "Teaching and learning of deburring robots using neural networks", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 339–345, 1993.

[48] F.L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance", *IEEE Symposium on Intelligent Control*, pp. 225–231, 1993.

[49] J. M. Tao and J. Y. S. Luh, "Application of neural network with real-time training to robust position/force control of multiple robots", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 142–148, 1993.

[50] A. Y. Zomaya and T. M. Nabhan, "Centrallized and decentralized neuro-adaptive robot controllers", *IEEE J of Ocean Engineering*, vol. 6, pp. 223–244, 1993.

[51] R. Carelli, E. F. Camacho, and D. Patino, "A neural network based feedforward adaptive controller for robots", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, pp. 1281–1287, 1995.

[52] S. P. Chan, "A neural network compensator for uncertainties in robotic assembly", *Journal of Intelligent and Robotic Systems*, vol. 13, pp. 127–141, 1995.

[53] J. Yuh, "An intelligent control system for remotely operated vehicles", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 20, pp. 1475–1483, 1990.

[54] K. P. Venugopal, R. Sudhakar, and A. S. Pandya, "On-line learning control of autonomous underwater vehicles using feedforward neural networks", *Journal of Oceanic Engineering*, vol. 17, pp. 308–319, 1992.

[55] J. Yuh, "A neural net controller for underwater robotic vehicles", *IEEE J of Ocean Engineering*, vol. 15, pp. 161–166, 1990.

[56] J. Yuh and R. Lakshmi, "An intelligent control system for remotely operated vehicles", *IEEE J of Ocean Engineering*, vol. 18, pp. 55–62, 1993.

[57] R. M. H. Cheng, J. W. Xiao, and S. LeQuoc, "Neuromorphic controller for agv steering", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 2057–2062, 1992.

[58] R. Fierror and F.L. Lewis, "Control of a nonholonomic robot using neural networks", *IEEE Symposium on Intelligent Control*, pp. 415–421, 1995.

[59] H.C. Tseng and V.H. Hwang, "Neural network for nonlinear servomechanism", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 2414–2417, 1991.

[60] T.S. Low, T.H. Lee, and H.K. Lim, "A methodology for neural network training for control of drives with nonlinearities", *IEEE Trans. on Industrial Electronics*, vol. 40, pp. 243–249, 1993.

[61] Y. S. Kung, C. M. Liaw, and M. S. Ouyang, "Adaptive speed control for induction motor drives using neural networks", *IEEE Transactions on Industrial Electronics*, vol. 42, pp. 25–32, 1995.

[62] M. Godoy and B.K. Bose, "Neural network based estimation of feedback signals for a vector controlled induction motor drive", *IEEE Trans. on Industry Application*, vol. 31, 1995.

[63] G. Cybenko, "Approximation by superposition of sigmoidal function", *Mathematics of Control signals and systems*, vol. 2, pp. 303–314, 1989.

[64] K. Hornik, M. Stinchcomb, and H. White, "Multilayer feedforward networks are universal approximator", *Neural Networks*, vol. 2, pp. 359–366, 1989.

[65] L. G. Kraft III and D. P. Campagna, "A summary comparison of cmac neural network and traditional adaptive control systems", *Neural Networks for Control, The MIT Press*, pp. 143–169, 1990.

[66] S. Jung and T.C. Hsia, "Neural network inverse control of robot manipulator", *Proc. of International Conference on Neural Information Processing*, pp. 603–608, Seoul, 1994.

[67] S. Jung and T.C. Hsia, "On-line neural network control of robot manipulators", *Proc. of International Conference on Neural Information Processing*, pp. 1663–1668, Seoul, 1994.

[68] S. Jung and T. C. Hsia, "A new neural network control technique for robot manipulators", *Robotica*, vol. 13, pp. 477–484, 1995.

[69] S. Jung and T.C. Hsia, "A study on new neural network schemes for robot manipulator control", *Robotica*, vol. 14, pp. 7–16, 1996.

[70] W. S. McCullogh and W. H. Pitts, "A logical calculus for the ideas imminent in nervous activity", *Bulletin of Math. Biophisics*, vol. 5, pp. 115–133, 1943.

[71] F. Rosenbratt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanism*, Spartan Books, 1962.

[72] M. Minsky and S. Papert, *Perceptrons*, The MIT Press, 1969.

[73] P. J. Werbos, *Beyond Regression: New tools for prediction and analysis in the behavioral sciences*, Ph.D Thesis, Harvard University, 1974.

[74] D. B. Parker, *Learning Logic*, Tech. report TR-47, center for Computational Research in Economics and Management Science, MIT, 1985.

[75] D. E. Rumelhart, G.E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*, The MIT Press, 1986.

[76] T.C. Hsia, *System Identification*, Lexington Books, Lexington, Mass, 1977.

[77] T.C. Hsia, "Robustness analysis of pd controller with approximate gravity compensation for robot manipulator control", *Journal of Robotic System*, 1994.

[78] S.M. Shahruz, G. Langari, and M. Tomizuka, "Design of robust pd-type control laws for robotic manipulators with parametric uncertainties", *Technical Report ESRC 91-3,Engineering Systems Research Center, University of California,Berkeley*, 1991.

[79] R. C. Dorf and R. H. Bishop, *Modern Control Systems, 7th edition*, Addison Wesley, 1995.

[80] Z. Mao and T. C. Hsia, "Stability bounds of momentum coefficient and learning rate in back-propagation algorithm", *Proc. of European Symposium on Artificial Neural Network*, 1994.

[81] S. Jung, "On tracking performance of a robot manipulator under force control", *Master Thesis at University of California, Davis*, June, 1991.

[82] M. T. Mason, "Compliance and force control of computer controlled manipulators", *IEEE Transactions on System, Man, and Cybernectics*, pp. 449–455, 1981.

[83] N. Hogan, "Impedance control : An approach to manipulator, part i, ii, iii", *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 3, pp. 1–24, 1985.

[84] R. Colbaugh, H. Seraji, and K. Glass, "Direct adaptive impedance control of robot manipulators", *Journal of Robotic Systems*, vol. 10, pp. 217–248, 1993.

[85] T. Lasky and T.C. Hsia, "On force-tracking impedance control of robot manipulators", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 274–280, 1991.

[86] H. Seraji, "Adaptive admittance control : An approach to explicit force control in compliant motion", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 2705–2712, 1994.

[87] D. A. Lawrence, "Impedance control stability properties in common implementations", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 1185–1190, 1988.

[88] S. Jung and T. C. Hsia, "Neural network techniques for robust force control of robot manipulators", *Proc. of IEEE Symposium on Intelligent Control*, pp. 111–116, Monterey, August, 1995.

[89] B. W. Drake and T. C. Hsia, "Implementation of a unified robot kinematics and inverse dynamics algorithm on a DSP chip", *IEEE Transactions on Industrial Electronics*, vol. 40, pp. 190–195, 1993.

[90] Sukhan Lee and Hahk Sung Lee, "Intelligent control of manipulators interfacing with an uncertain environment based on generalized impedance", *Proc. of IEEE Symposium on Intelligent Control*, pp. 61–66, 1991.

[91] H. Seraji and R. Colbaugh, "Force tracking in impedance control", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 499–506, 1993.

[92] R. Colbaugh and A. Engelmann, "Adaptive compliant motion of manipulators : Theory and experiments", *Proc. of IEEE International Conference on Robotics and Automations*, pp. 2719–2726, 1994.

[93] S. Jung, T. C. Hsia, and R. G. Bonitz, "On force tracking impedance control with unknown environment stiffness", *Proc. of IASTED International Conference on Robotics and Manufacturing*, pp. 181–184, Cancun, June, 1995.

[94] R.G. Bonitz and T.C. Hsia, "Internal force-based impedance control for cooperating manipulators", to appear in *IEEE Transactions on Robotics and Automation*.