REPETITIVE LEARNING CONTROL OF ROBOTIC MANIPULATORS

BY

JIANGUO (JAMES) FU

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

May, 1993

REPETITIVE LEARNING CONTROL OF ROBOTIC MANIPULATORS

In Memory of My Mother

and Dedicated to

My Wife and My Daughter

DOCTOR OF PHILOSOPHY (1993) MCMASTER UNIVERSITY

(Electrical & Computer Engineering) Hamilton, Ontario

TITLE: Repetitive Learning Control of Robotic Manipulators

AUTHOR: Jianguo (James) Fu, B. Sc. (Wuhan Univ. of Engineering)

M. Sc. (Beijing Institute of Technology)

SUPERVISOR: Naresh K. Sinha

Professor of Electrical & Computer Engineering

B. Sc. Eng. (Banaras)

Ph. D. (Manchester)

NUMBER OF PAGES: CLXXIV, 174

ABSTRACT

This thesis deals with the repetitive learning control of robotic manipulators. The research topic is motivated by the fact that industrial robots usually perform repetitive tasks. Unlike other conventional controllers, the repetitive learning controller can improve the performance of a robot as it repeats the same task. The proposed control strategy has advantages of easy implementation, low cost and better performance. A critical survey of the state of the art in repetitive learning control of robots is presented. This thesis is divided into two major parts: a two operational modes based approach and a neural network based scheme.

In the first part of the thesis, a theoretical framework has been developed for designing controllers based on the proposed two operational modes. The repetitive operations of a robot can be divided into two operational modes: a single operational mode and a repetitive operational mode. Based on this, the author proposes repetitive learning schemes for motion control of both rigid and flexible joints robots, as well as for control of constrained robots with rigid joints. The designed controllers converge in both operational modes. In the single operational mode, the controllers behave like adaptive controllers, while similar to a betterment process in the repetitive operational mode. These conclusions are firmly supported by simulation studies.

In the second part of this thesis, special attention is paid to schemes based on neural networks. The author's own modification of back-propagation neural networks is first established in the thesis. It is then applied to motion control of both rigid and flexible joints robots, as well as force control of rigid robots. The neural networks play the role of an approximate inverse dynamic model of a robot and are then parallelled with a conventional PD controller to achieve control goals. Very promising results have been reported in this thesis.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Professor N.K. Sinha for his expert guidance and constant encouragement throughout this work. I also thank Professor M. Elbestawi, Professor S.S. Haykin and Professor S. Poehlman, members of my supervisory committee for their continuing interest and support.

I would like to thank Dr. Yuexin Han, Mr. Zhiqiang Li and all my present and past control group colleagues for their participation in several of my presentations , valuable suggestions and inspiring discussions with me.

The financial assistance provided by the Department of Electrical and Computer Engineering, McMaster University, is gratefully acknowledged.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

CHAPTER ONE INTRODUCTION

1.1 Background

A robotic manipulator is a very complicated nonlinear system. It is difficult to determine its precise mathematical model. Due to the model uncertainties, many strategies have been developed for its control. Most of these use either a robust controller or an adaptive controller. These are feedback control schemes based on the inverse dynamics of the manipulator. They have achieved very good results. However, they suffer from the following drawbacks:

1) requirement of heavy on-line computation;

2) inability to handle large uncertainties;

3) operation errors may be repeated from cycle to cycle.

To overcome these difficulties, an iterative learning control scheme, or a repetitive learning control scheme,[1] for robotic manipulators has been studied in recent years. Iterative learning control can be defined as "Any control scheme that improves the performance of the device being controlled as actions are repeated, and to do so without the necessity of a parametric model of systems" [27]. Basically, the iterative learning controller is a kind of adaptive feedforward controller, which

---

[1]. In this thesis, an iterative learning controller is the same as a repetitive learning controller. As in literature, these two terms appear interchangeable.

modifies the control inputs by making use of the previous operational data, such as input signals, position and velocity errors. This signal-learning approach, as opposed to parameter-adaptation approach, is a fundamental difference between iterative learning control and adaptive control. Also, iterative learning control is implemented off-line while adaptive control is realized on-line. Thus, iterative learning control saves valuable on-line computation time. One disadvantage of iterative learning control is that it requires repetitive operations of the robotic manipulator. Fortunately, most industrial robots perform repetitive tasks.

Figure 1.1 is a schematic representation of iterative learning control. The block F.C. represents the feedback controller and the block L.C. represents the learning controller. The reference signal r(t) is the desired trajectory of the robot.

All signals are indexed by k, with k beginning from 1. For instance, in the (k+1)th operation, the total control input to robot U^{k+1} is composed of two parts:

a) a feedback control input $U_{fb}{}^{k+1}$, which could be produced by a PID controller or an adaptive controller.

b) a feedfoward control signal $U_{ff}{}^{K+1}$, generated by adding the control signal U^k used in the previous trial to the output of the learning operator. The learning operator maps the error signals obtained from previous trial into an additional control signal. Most learning operators are of the PD type and convergence can be guaranteed by carefully choosing the gains of the learning

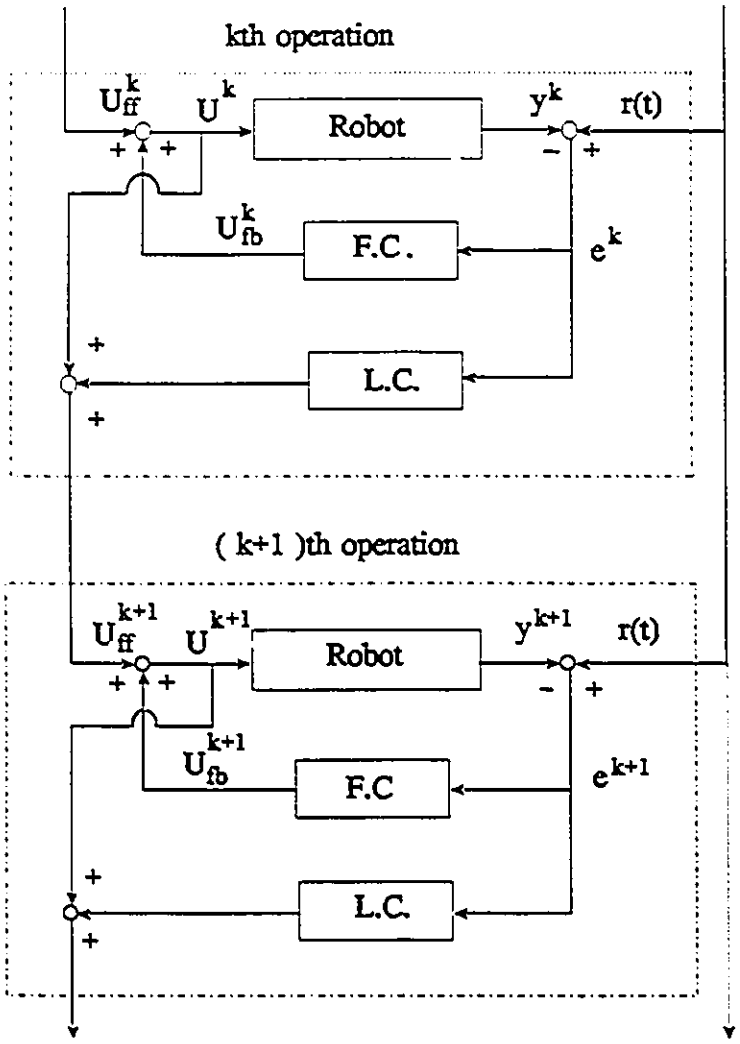

Figure 1.1 Diagram of Iterative Learning

operator.

Arimoto and his research group are one of the pioneers in studying iterative learning control of robotic manipulators [24] [25]. They developed a PD

type of learning control scheme based on experience from previous trials. The tracking errors can be reduced to zero as the number of trials increase if some restrictive conditions are satisfied. Almost at the same time as Arimoto's group, Aicardi [32] and Bondi's research groups [26] studied iterative learning control from the high gain learning point of view. They concluded that convergence can be achieved if the learning gains are sufficiently large. However, they could not specify how large the gains should be.

The main idea of iterative learning control is to improve the performance of the device as control actions are repeated, and to do so without needing a parametric model of the systems. As a matter of fact, convergence of an iterative learning scheme may not guarantee satisfactory performance and the controller may have a very slow convergence rate, as reported in reference [19]. A practical iterative learning controller should utilize an approximate model of the controlled system in order to achieve better performance and faster rate of convergence. To use the available knowledge of the controlled robots, both linearized models [56][66] and nonlinear models [23][38][60] are used by different researchers.

In order to deal with issues of implementation on a digital computer, various discrete time algorithms have been proposed in literature[52][53][54]. Based on the discrete model of a robot, many researchers have presented local optimization approaches to derive learning laws.

The various approaches mentioned above are mainly concentrated on motion control of robots. Based on previous investigations, iterative learning control needs considerable improvement in the following areas:

1) rate of convergence

2) robustness

3) applications to constrained robots and flexible joints robots.

The resurgence of neural networks has inspired the interest of the control community because of its high potential to solve control problems of complicated nonlinear systems. Back-propagation neural networks are dominantly used in the control field. Many researchers are engaged in the study of neural controllers for robotic manipulators [17][21][22][28][78][85]. They have made much progress in employing back-propagation neural networks. Except reference [28], the rest of the neural controllers are implemented on-line as feedback controllers. Thus, this is difficult to implement in real time since the neural controller is composed of a huge number of neurons. To overcome this problem, the author has proposed a method to combine the idea of iterative learning scheme with that of neural network. This idea is adopted to design a new type of learning scheme for control of a robot.

1.2 Organization and Contributions of the Thesis

The principal objective of this thesis is to study repetitive learning control of robotic manipulators. Two distinct approaches have been explored:

1) Two Operational Modes Based Approach: The operation of a robot is classified into two operational modes, the single operational mode and repetitive operational mode. This idea is explained in detail in Chapter 3. Then controller design methods have been developed. These include motion control of both rigid and flexible joints robots as well as control of constrained rigid robots. A novel theoretical frame has been established and satisfactory results are achieved.

2) Neural Network Based Scheme: In this part, a modification of back-propagation neural networks is first introduced. It is then applied to the control of robots. A broad scope of applications have been intensively investigated for robots. Simulation results suggest that the neural networks based scheme is quite promising.

A critical review of the state of the art in repetitive learning control of robots is presented in Chapter 2. The literature is classified according to the corresponding control strategies and learning laws. The merits and drawbacks of each approach and how different authors have applied it to the problem under consideration are stressed. The material presented in Chapter 2 also provides an adequate background for some of the studies presented in the subsequent chapters.

In Chapter 3, it is first emphasized that most industrial robots perform repetitive tasks in the form of two operational modes. Then, it is explained that the existing controllers are all one operational mode based schemes, for example, adaptive controllers and so-called iterative learning controllers. The studies in the

subsequent chapters of part one are carried out based on the idea proposed here.

Motion control of robotic manipulators has been studied by using a repetitive learning method in chapter 4. A controller designing method has been developed in the theory of two operational modes. The proposed controllers converge in both operational modes. A simulation study has demonstrated its effectiveness.

In Chapter 5, a model of a constrained robot is first investigated. Through proper nonlinear transformation, the movement of a constrained robot can be decomposed into constrained movement in one direction and free one in other directions. Using this model, the results obtained in Chapter 4 are extended to the case of a constrained robot.

A new controller is developed in Chapter 6 for a flexible-joint robot. The knowledge of joint stiffness can be assumed unavailable. It is treated as identified parameters in the new adaptive controller. Also, measurement of accelerations is not required in the proposed control scheme.

In Part Two, studies are devoted to neural network based repetitive learning controllers. A modification of back-propagation neural networks is first studied in Chapter 7. The presented neural network has a much faster rate of learning speed and good stability. Also, different training strategies are fully investigated and compared. Based on these, a training method is selected, which is most suitable for applications in this thesis.

In Chapter 8, motion control of robotic manipulators is studied using neural networks. The employed neural networks are trained as an approximate model of the robot in a neighbourhood of desired trajectory. This training method is also used in the subsequent chapters. The learning process gradually transfers the control action from a feedback PD controller to the feedforward neural controller. Meanwhile, the control performance is improved greatly. Promising simulation results are reported in this chapter also. Studies suggest that the neural network based approach is an ideal method for motion control of robots.

A study similar to that Chapter 8 is conducted in Chapter 9 for force control of robots. The proposed repetitive learning controller has achieved a satisfactory result for force control of robots, a more challenging job. However, in the author's opinion, the neural network based approach is more suitable for the motion control than the force control.

Motion control of a flexible-joint robot has been studied in Chapter 10 by using neural network based repetitive learning schemes. A teaching controller is first introduced. The operational data obtained are used to train the neural networks. The proposed control scheme does not require the measurement of accelerations. It needs only the measurement of position variables and velocity variables associated with both link part and motor part. Results of simulation have demonstrated that the proposed control approach is very effective. Moreover, it is difficult for conventional controllers to be equally effective.

In Chapter 11, some suggestions for future research have been made.

Th following contributions have been made in this thesis:

1. In the first part of the thesis, the author has proposed a new design method based on the theory of two operational modes. For a repetitive process, the operation can be divided into two modes, the single operational mode and the repetitive operation mode. Based on this decomposition, the author further suggests that the controller design of a repetitive process should be carried out in a fashion of two operational modes. A desired repetitive learning controller must have the convergence feature in both operational modes.

2. A new controller is proposed for motion control and force control of a robot performing repetitive tasks based on the proposed two operational modes. It has been theoretically verified that the proposed controller converges in both modes.

3. A novel scheme is presented in this thesis for motion control of flexible joint robots. The control scheme does not require the knowledge of joint stiffness. However, it is required in the previous approaches. First, a controller design is developed by assuming that dynamics of a flexible-joint robot is known. Based on this, adaptive version is studied when only partial knowledge of a robot is available. This adaptive controller has been proven to be globally convergent. The unknown joint stiffness is treated as an identified parameter and is updated on-line.

4. In previous work, repetitive learning controllers for constraint robots

do not ensure stability. This thesis has presented a new controller whose stability is guaranteed.

5. An effective modification of back-propagation neural network is presented in the thesis. It has a much faster rate of learning speed and better robustness. The developed neural networks are successfully applied to the control of a robot in various applications.

6. This thesis is among the first to study a repetitive learning scheme for motion and force control of rigid robots, as well as motion control of flexible joints robots, using neural networks. The proposed scheme has advantages of little on-line computation and faster performance improving rate.

CHAPTER TWO REPETITIVE LEARNING CONTROL OF ROBOTIC MANIPULATORS: A REVIEW

2.1 Introduction

Most industrial robots perform repetitive tasks in the present industrial world. Unfortunately, the robots repeat the tracking errors also as they repeat operations. To address this kind of issue, repetitive learning control of robots has been studied as an important problem in the last decade. It was first introduced in Japan by Arimoto' pioneering work [24][25]. The ultimate goal of this learning scheme is to allow the robot to have a betterment ability as the robot performs the same task, just like human beings. Since then, various schemes have been proposed in this field, including neural network based approaches. In this chapter, the author will give a detailed survey of the perspective and the state of art of repetitive learning control technology. Their applications to robotic control is also discussed.

2.2 A Type of PD Repetitive Learning Controller

In this section, a class of PD type repetitive learning controllers will be investigated. This class of repetitive learning controller is among the first, called first generation algorithms[24][25].

Consider a class of nonlinear multi-input multi-output dynamic system

described by

$$\begin{cases} \dot{x} = f(t,x) + Bu \\ y = Cx \end{cases} \tag{2.1}$$

where $x, f, \in R^n$, $u, y \in R^n$, $C \in R^{n\times n}$, $B \in R^{n\times n}$.

Assume that the vector-valued function f satisfies a Lipschitz continuous condition

$$\| f(t,x_1) - f(t,x_2) \| \le \alpha(t) \| x_1 - x_2 \| \tag{2.2}$$

for any $t \in [0, T]$ and any pair (x_1, x_2) in a certain domain $\subset R^n \times R^n$, where $\alpha(t)$ is a piecewise function defined in [0, T]. A repetitive learning controller for this system is composed of the following equations:

$$\begin{cases} u_{k+1} = u_k + \Gamma \dot{e}_k(t) \\ e_k(t) = y_d(t) - y_k(t) \\ y_k(t) = Cx_k(t) \\ x_k(t) = x^0 + \int_0^t [\, f(\tau, x_k(\tau)) + Bu_k(\tau)]\, d\tau \end{cases} \tag{2.3}$$

where it is implicitly assumed that $x_k(0) = x^0$, that is, the initial condition for system equations is always the same in every operational trial.

Further assume that the following three conditions are satisfied simultaneously:

1. $\| I_n - CB\Gamma \| \le 1$ for all $t \in [0, T]$.
2. $u_0(t)$ and $y_d(t)$ are continuously differentiable on [0, T].

3. $y_d(0) = Cx^o$.

Then the repetitive learning controller described by equations from (2.1) to (2.3) is convergent in the sense that there are positive constants λ and ρ such that

$$\|\dot{e}_{k+1}\|_\lambda \leq \rho \|\dot{e}_k\|_\lambda \quad 0 \leq \rho < 1 \tag{2.4}$$

for all $k > 0$. Hence the following conclusions hold as $k \to \infty$

$\dot{y}_k(t) \to \dot{y}_d(t)$ uniformly in t ϵ [0, T],

$y_k(t) \to y_d(t)$ uniformly in t ϵ [0, T].

Notice from the repetitive control law described in (2.3) that this scheme is an open-loop architecture. The state feedback in the current operation is not used, only the errors in the previous operation. This is one of main shortcomings in this class of repetitive learning controllers.

2.3 A High Learning Gain Approach [26][32]

Consider the dynamic equations of a manipulator relating the vector $q \in R^n$ of the chosen coordinates with vector $u \in R^n$, the corresponding torque, is given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{2.5}$$

where matrices $M(q)$, $C(q, \dot{q})$ and vector $g(q)$ are continuous and locally Lipschitz functions of their arguments. These matrices are supposed to be completely unknown with exception that the value of $g(q_0)$ at the initial time is assumed known.

Let us introduce the following control law

$$u = m + Q - K\ddot{q} - L\dot{q} - pq \tag{2.6}$$

where m represents the vector of the effective driving signals, Q another constant vector of external signals, while matrices P, L, K are the position, velocity and acceleration gains, respectively.

The driving signal for the initial trial is given by

$$m_0 = K\ddot{q}_d + L\dot{q}_d + Pq_d \tag{2.7}$$

Assume that the first trial of movement has been carried out using $m_0(t)$ starting at the rest position $q_0(0) = q_d(0)$ (this requires $Q = C[q_d(0)]$ and explains the assumption about the knowledge of $C(q_0)$). Then the corresponding trajectory errors, $e_0(t) \triangleq q_0(t) - q_d(t)$, $\dot{e}_0(t)$ and $\ddot{e}_0(t)$, for $t \in [0, T]$, are bounded. They are recorded by suitable devices. Then the proposed learning procedure suggests repeating a trial, starting from the same initial rest position $q_d(0)$ and using the new driving signal

$$m_1 = m_0 - \delta m_1 \tag{2.8}$$

where the correcting term is given by

$$\delta m_1 = K\ddot{e}_0 + L\dot{e}_0 + Pe_0 \tag{2.9}$$

The same procedure is in turn applied for all successive trials (provided the errors are bounded), thus implying that the driving signal applied at the general ith ($i > 0$) trial takes on the form

$$\begin{aligned} m_i &= m_{i-1} - \delta m_i = m_0 - \sum_{k=1}^{i} \delta m_k \\ &= m_0 - \sum_{k=1}^{i} (K\ddot{e}_{k-1} + L\dot{e}_{k-1} + Pe_{k-1}) \end{aligned} \tag{2.10}$$

If the feedback matrices K, L and P are selected as the following forms

$$K = \alpha \bar{K}, \quad L = \alpha \bar{L} \quad P = \alpha \bar{P} \tag{2.11}$$

where $\alpha > 0$. Further α is selected sufficiently large, then the following conclusion holds

$$\lim_{i \to \infty} \| (e_i(t), \dot{e}_i(t), \ddot{e}_i(t) \| = 0 \tag{2.12}$$

Obviously, the requirement of acceleration measurement is a drawback when considering implementation issue. To overcome this problem, a similar method is presented in [32] that does not require on-line acceleration feedback. This is achieved by using on-line identification of inertial matrix $M(q)$.

2.4 Linearized Models Based Methods [56][66]

Consider a robotic manipulators given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u(t) \tag{2.13}$$

Linearizing the above system along the desired trajectory for $t \in [0, T]$ at the ith iteration, the following linear time-varying system is obtained:

$$C_d[\ddot{q}_i(t)-\ddot{q}_d(t)]+E_d[\dot{q}_i(t)-\dot{q}_d(t)]+F_d[q_i(t)-q_d(t)]=u_i-S_d \qquad (2.14)$$

where

$$\begin{aligned} C_d &= M(q_d) \\ E_d &= \frac{\partial C}{\partial \dot{q}(t)}\Big|_{(q_d(t),\dot{q}_d(t))} \\ F_d &= \frac{\partial M}{\partial q(t)}\Big|_{q_d(t)}\ddot{q}_d+\frac{\partial C}{\partial q(t)}\Big|_{(q_d(t),\dot{q}_d(t))}+\frac{\partial g}{\partial q(t)}\Big|_{(q_d(t),\dot{q}_d(t))} \\ S_d &= M(q_d(t))\ddot{q}_d(t)+C(q_d(t),\dot{q}_d(t))\dot{q}_d+g(q_d(t)) \end{aligned} \qquad 2.15$$

$u_i(t)$ is the input torque at the ith iteration.

To make the robot system track $q_d(t)$ for $t \in [0, T]$, we construct a controller as follows:

$$u_i = T_i + H_i \qquad (2.16)$$

where $T_i = K(q_d - q_i) + L(\dot{q}_d - \dot{q}_i)$. K and L are positive definite matrices, which account for position and velocity feedback gains. Thus, T_i represents a conventional PD controller. H_i is the predicted feedforward control input to be computed at each iteration by a learning rule. Applying the input (2.16) to system (2.14), we obtain the error equation

$$C_d\ddot{e}_i(t)+(E_d+L)\dot{e}_i(t)+(F_d+K)e_i(t)=S_d-H_i \qquad (2.17)$$

Then a learning rule is derived for updating H_i such that H_i converges to an unknown value S_d. Consider the following index for $t \in [0, T]$:

$$J_i = \frac{1}{2}\sum_{i=1}^{\infty} \| S_d(t) - H_i(t) \| \tag{2.18}$$

Applying the gradient descent rule, we obtain

$$H_{i+1} = H_i - ß \frac{\partial J_i}{\partial H_i} = H_i + ß (S_d - H_i) \tag{2.19}$$

where ß is a positive constant often called a training factor. ß is selected such that $0 < ß < 2$ to guarantee the convergence of H_i. Since S_d is not known, the above equation cannot be used directly as a learning rule. Instead, replacing the unknown term S_d - H_i by the available quantity T_i in (2.16), we obtain

$$H_{i+1} = H_i + ß T_i \tag{2.20}$$

If K and L are sufficiently large, then T_i is about the same S_d - H_i. Roughly speaking, the learning rule can be considered as searching algorithm for the unknown desired input torque S_d in which the error T_i from the PD controller is used to update H_i. Assume that $q_i(0) = q(0)$ and $\dot{q}_i(0) = \dot{q}_d(0)$ for all $i > 0$. In practice, ß is usually selected to be less than one due to sensitivity considerations and $H_1(t) = 0$ is chosen for all $t \in [0, T]$.

With the control input (2.16) and learning rule (2.20), the following conclusions hold for all $t \in [0, T]$:

$$\lim_{i\to\infty} e_i(t) = 0, \qquad \lim_{i\to\infty} H_i = S_d \tag{2.21}$$

provided that K and L are selected sufficiently large such that

$$\begin{aligned} l_a &\triangleq \lambda_{\min}[(2-\beta)L] - (2a\|C_d\| + \|\dot{C}_d\|) > 0 \\ l_b &\triangleq \lambda_{\min}[(2-\beta)L] - \frac{2}{a}\|F_d\| > 0 \\ \sqrt{l_a l_b} &> \frac{1}{a}\|F_d\| \end{aligned} \tag{2.22}$$

where λ_{min} (A) denotes the minimum eigenvalue of A and $K=aI$. a is a positive constant.

2.5 Discrete Time Algorithms [52]

To consider the issue of implementation of learning algorithms on a digital computer, some discrete time approaches have been proposed.

Assume that the control system with sampling intervals given as ΔT. The time interval [0, T] is partitioned into N intervals given by $\{t_0, t_1, \ldots, t_N\}$, where $t_k = k\,\Delta T$. The measurements of $y(t)$ are sampled at $t_0, t_1, \ldots, t_N$. The control is held constant through the interval $[t_k, t_{k+1})$. Suppose the state space equation of a robot can be approximated by the following difference equation:

$$\begin{aligned} x_{k+1} &= \Psi(x_k) + \Phi(x_k)u_k \\ y_k &= g(x_k) \end{aligned} \tag{2.23}$$

where $x_k = x(t_k)$ etc. Clearly, the input u_k can only affect the outputs $y_{k+1}, y_{k+2}, \ldots$.

The approximation $\dot{e}(t_k) = [\, e(t_{k+1}) - e(t_{k-1}) \,] / 2\Delta T$ is valid for small ΔT. Hence , the iterative learning law is of following discrete-time form:

$$u^{i+1}(t_k) = u^i(t_k) + L(y^i(t_k)) \frac{e^i(t_{k+1}) - e^i(t_{k-1})}{2} \tag{2.24}$$

with the convergence condition, given $e^0 = 0$ and

$$\| I - L(y_k^i) G(x_k) \Phi(x_k) \| \le \rho < 1 \tag{2.25}$$

where $L(\cdot)$ is called a learning operator, either linear or nonlinear, and

$$G(x) = \frac{\partial g(x)}{\partial x} \tag{2.26}$$

is the output Jacobian matrix.

Remark: The central difference $\dot{e}(t_k) \approx (e_{k+1} - e_{k-1}) / 2\Delta T$ is a quadratic approximation, while the more common forward difference $\dot{e} \approx (e_{k+1} - e_k) / \Delta T$ is a less accurate linear approach.

Based on this discrete model of a robot, various local optimization algorithms are presented.

Define a criterion function to be minimized

$$J(u_k) = \frac{1}{2} e_{k+1}^T Q e_{k+1} \tag{2.27}$$

where Q is a positive definite weight matrix. The first and second partial derivative

of V with respect to u_k becomes

$$\nabla_u(J_k) = -[G(x_k)\Phi(x_k)]^T Q e_{k+1} \tag{2.28}$$

$$\nabla_{uu}(J_k) = [G(x_k)\Phi(x_k)]^T Q G(x_k)\Phi(x_k) \tag{2.29}$$

Higher order derivatives vanish.

Since the inertial matrix $M(q)$ is positive definite, the matrix $\Phi(x_k)$ has full rank for all x_k. The physical interpretation is obvious: every non-zero control input vector makes a non-zero contribution to the state vector at the next instant: $\Phi(x_k)u_k = 0 \leftarrow u_k = 0$. The following important result indicates that the most gradient based optimization methods will converge.

The criterion function $J(u_k)$ is convex for all $G(x)$ with full rank. That is, the Hessian matrix $\nabla_{uu}J_k$ is positive definite. For example, Newton's method gives the learning law

$$u_k^{i+1} = u_k^i - [\nabla_{uu}J(x_k^i)]^{-1}\nabla_u J(x_k^i) = u_k + L_k^i e_{k+1}^i \tag{2.30}$$

which satisfies (2.25) with $\rho = 0$.

However, the local analysis tells us little about the global feature of the algorithms. Also, it requires the complete knowledge of the robotic systems. This is practically impossible. Hence, the local optimization algorithms are unrealistic.

2.6 A Nonlinear Model Based Repetitive Technique [23][38]

Consider a robotic manipulator for repetitive tasks. By a repetitive task, we mean that the robot is required to execute the same motion over and over again within a fixed period. Therefore, it is assumed that the desired trajectory signals, $x_d(t)$, $\dot{x}_d(t)$, $\ddot{x}_d(t)$ are periodic and bounded. Namely,

$$x_d(t+T)=x_d(t) \quad \dot{x}_d(t+T)=\dot{x}_d(t) \quad \ddot{x}_d(t+T)=\ddot{x}_d(t) \tag{2.31}$$

where T is the period. Define

$$W(v_1,v_2,v_3,v_4)=M(v_1)v_4+C(v_1,v_2)v_3+g(v_1) \tag{2.32}$$

Matrices M, C, g are given in (2.5). Then $W(x_d(t), \dot{x}_d(t), \dot{x}_d(t), \ddot{x}_d(t))\theta$ is also periodic. We shall denote this periodic term by $w_r(t)$ and assume its lower and upper bounds are known for all $t \in [0, T]$.

$$\begin{aligned} w_r(t) &= W(x_d,\dot{x}_d,\dot{x}_d,\ddot{x}_d)\theta \\ w_r(t+T) &= w_r(t) \end{aligned} \tag{2.33}$$

and

$$\underline{w}_{ri} < w_{ri} < \bar{w}_{ri}, \quad \text{for all } t \tag{2.34}$$

where $\bar{w}_{ri}$ and $\underline{w}_{ri}$ are the known upper and lower bounds of $w_{ri}(t)$, and

$$\max\left\{\|w_r-\underline{w}_r\|, \|\bar{w}_r-w_r\|\right\} < r_w \tag{2.35}$$

The repetitive learning control law is given by

$$u(t) = \hat{w}_r(t) - K_d r - K_p e - q_n(r,e) \tag{2.36}$$

where the term $\hat{w}_r(t)$ is an estimate of the periodic function $w_r(t)$. $r(t) = \dot{e}(t) + \lambda e(t)$ and $e(t) = x(t) - x_d(t)$. K_d, K_p and λ are positive diagonal matrices. The nonlinear term q_n is given by

$$q_n(r,e) = \sigma_n r \|e\|^2, \quad \sigma_n > 0 \tag{2.37}$$

The following learning algorithm is proposed for updating the estimate $\hat{w}_r(t)$:

$$\begin{aligned} \hat{w}_r^*(t) &= \hat{w}_r(t-T) - K_L r(t), \quad K_L > 0 \\ \hat{w}_r(t) &= \pi[(\hat{w}_r^*(t)] \end{aligned} \tag{2.38}$$

where $\pi(\bullet)$ is the projection given by the following equation:

$$\pi(\hat{w}_{ri}^*) = \begin{cases} \hat{w}_{ri}^*(t) & \text{if } \underline{w}_{ri}(t) - \delta \le \hat{w}_{ri}^*(t) \le \overline{w}_{ri}(t) + \delta \\ \underline{w}_{ri}(t) & \text{if } \hat{w}_{ri}^*(t) < \underline{w}_{ri}(t) - \delta \\ \overline{w}_{ri}(t) & \text{if } \hat{w}_{ri}^*(t) > \overline{w}_{ri}(t) + \delta \end{cases} \tag{2.39}$$

and $\delta > 0$ is a small arbitrary constant.

With the above assumptions and the repetitive learning control law, the following conclusions hold:

$$\lim_{t\to\infty}[x(t) - x_d(t)] = 0, \qquad \lim_{t\to\infty}[\dot{x}(t) - \dot{x}_d(t)] = 0 \tag{2.40}$$

provided that matrices K_d, K_p, σ_n are selected sufficiently large.

This repetitive learning algorithm is nonlinear model based. It has advantages over previous algorithms. Part of this thesis is based on this work.

2.7 Neural Network Based Approaches

Model uncertainty is a serious problem facing control engineers. Often, it is impossible to represent, adequately, system characteristics such as nonlinearity, time delay, saturation, time-varying parameters, and overall complexity. The resurgence of neural networks has attracted special attention of control community, because neural networks have the potential to solve many problems where traditional analytic approaches have failed.[69][70][71][72]

The "artificial" neural networks can be regarded as massively parallel interconnected networks of simple (usually) adaptive elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same way as the biological nervous systems do. Therefore, a neural network is a parallel processing architecture in which a large number of processing units (or neurons) are interconnected and the knowledge is represented by the strengths of connections between the units. The connection strengths are achieved through a learning process. The knowledge is usually distributed over a large number of connections so that the operation of these networks degrades gracefully if part of the connections are destroyed due to malfunctioning. The parallel structure of a neural

network provides it with striking nonlinear mapping ability. This ability is what control engineers are looking for.

In the control field, back-propagation neural networks are most prevalent because they have the capability to "learn" system characteristics through nonlinear mapping[21][22][28]. Basically, a back-propagation neural network is a multi-layered network consisting of an input layer, an output layer, and at least one hidden layer with nonlinear processing units. The nonlinear processing units, or neurons, sum incoming signals and generate output signals according to some predefined functions. The neurons are connected by terms of variable weights.

A lot of investigations have been conducted in applications of back-propagation neural network to the control of robots. Here, a brief description of two typical control schemes is introduced in foregoing paragraphs.

An adaptive control architecture is depicted in figure 2.1, where an indirect learning scheme is used.

In this architecture, the neural network is presented with a desired trajectory of a robot fed into the input layer of the neural network. Using its connection weights, at the output layer, the network produces an output that is used as a control input for driving the robot. During the control process, the network learns the inverse dynamics of the unknown robot as a feedforward controller. The actual trajectory of the robot is measured and fed into the same network (although two different blocks are shown in the figure 2.1, with connection weights representing

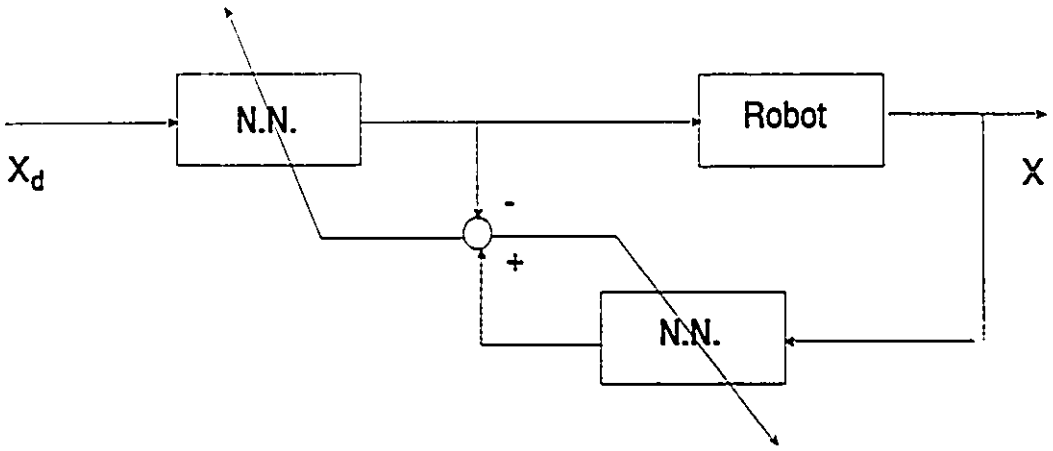

Figure 2.1 Adaptive Structure with an Indirect Learning Scheme

the inverse dynamic model of the robot, in a forward manner for estimating a required control input in order to achieve the actual trajectory of the robot according to the currently learned inverse dynamic model. When a learning process is initiated, the initial neural network might not accurately represents the inverse dynamics of the robot. The discrepancy between the actual input to the robot and an estimated input obtained by the above procedure is used to adjust the connection weights according to error back-propagation method.

Another control configuration is Kawato's [28] well-known feedforward and feedback training scheme described in figure 2.2.

An interesting point here is that feedback control signals are propagated back into neural networks in a feedforward path. This neural learning controller is trained as to make the output of the feedback controller zero.

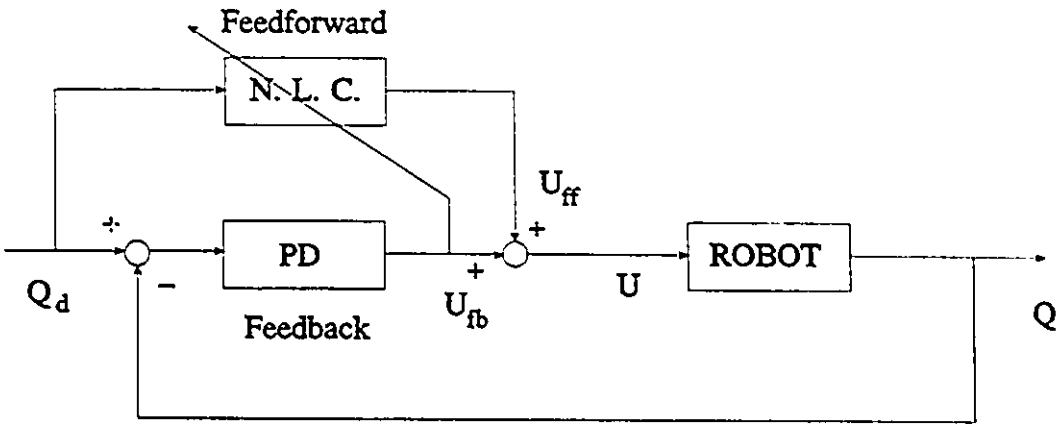

Figure 2.2 Scheme of Neural Learning Controller

Therefore, the control of the system is transferred from the feedback PD controller to the feedforward neural learning controller. This scheme has the advantage of making the system stable at the beginning of neural network training because of the robustness of the PD feedback controller. This training scheme does not use the output error signal. Unlike some other training configurations, the output error signal has to propagate through the robot into the neural networks. Since the robot dynamics is assumed unknown, the backpropagation algorithm cannot be used in the latter case.

The drawbacks of present training schemes, including Kawato's method, are that they depend heavily on-line updating of connected weights. As mentioned earlier, a neural network is a massively parallel connected architecture, the on-line learning is extremely time consuming. It is impractical for real time applications,

such as control of a robot. Thus, an off-line learning method is adopted in this thesis. It is believed in this thesis that neural network is most suitable to control a robot for repetitive tasks. The second part of the thesis will be devoted to investigating this issue.

2.8 Concluding Remarks

Relatively speaking, repetitive learning control for robots is quite a new area, although substantial work has been done by different researchers in the past decade. However, there are many unsolved issues. At present, iterative learning schemes are mainly directed to motion control of robots. Force control and flexible joints issues are rarely touched. It is the objective of this thesis to address these issues. The author's research work has given answers to some of these important problems.

PART ONE TWO OPERATIONAL MODES BASED METHODS

In this part, A fact is first clarified that a robot performs repetitive tasks in a form of two operational modes, a single operational mode and a repetitive operational mode. Adaptive controllers are designed in the domain of the single operational mode, in which all repetitive operations of the robot are treated as a single operation. Previous iterative learning controllers are designed in the domain of the repetitive operational mode, in which the information in each single operation is not utilized. Hence, both controllers are designed according to only one operational mode and suffer from some inherent drawbacks. For adaptive controllers, the tracking errors will be repeated also when a robot repeats its task, while previous iterative learning controllers have unsatisfactory performance in the first few trials.

To address this problem, a new approach is presented for design of controllers based on the proposed theory of two operational modes. The designed controllers converge in both operational modes.

CHAPTER THREE INTRODUCTION OF TWO OPERATIONAL MODES

3.1 Description of Two Operational Modes

A robotic manipulator is a very complicated nonlinear system. It is difficult to control because a precise description of its dynamics is not available. This is due to the existence of unknown characteristics inherent in mechanical structures such as friction, backlash and flexibility. Due to the model uncertainties, two major strategies, adaptive control schemes [1][2][4][6] and iterative learning controllers [24][25][26][27], have been developed for the control of robots. The former controllers are driven by an updated parameter scheme. The latter aim to reduce tracking errors to zero as a robot repeats its task without assuming a model of the robot. However, they suffer from some drawbacks. Adaptive controllers require heavy on-line computation and are unable to handle large uncertainties. Since most adaptive controllers do not guarantee that estimated parameters of robotic system converge to their true values, tracking errors may be repeated also as the robot repeats its operations. On the other hand, iterative learning controllers suffer from an extremely low rate of convergence. These two controllers are one operational mode based, either in the domain of single operational mode or repetitive operational mode. If we carefully examine operational process of a robot which performs a repetitive task, the repetitive operations take place in a form of two

operational modes. The figure below briefly describes a time history of trajectories when a robot executes a repetitive task.

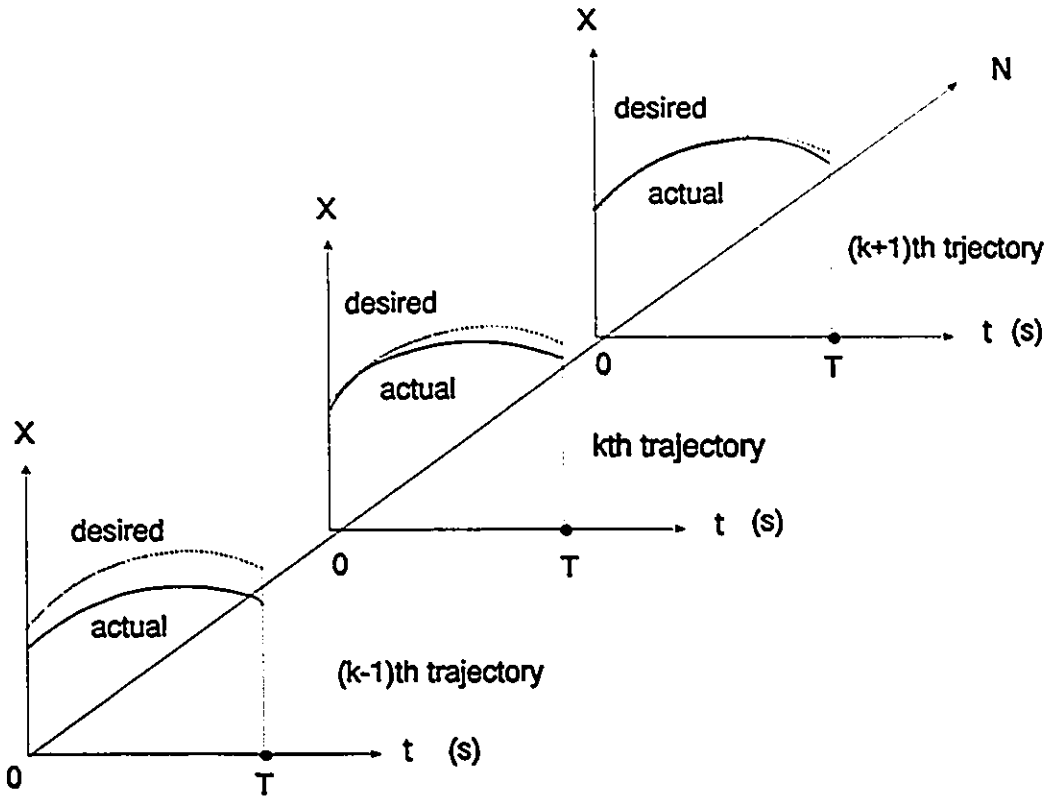

Figure 3.1

The repetitive operations of a robot can be decomposed into two operational modes: the single operational mode and the repetitive operational mode, as shown in the above figure. In the single operational mode, only the information of the present operation is utilized. In the repetitive operational mode, convergence of the controllers is not guaranteed in each individual operation. However, the

designed controllers, based on the proposed theory of two operational modes here, are ensured convergent in both modes. For convenience, we denote t as time in the single operational mode and N as the sequential number of repetitive operation in the repeated operational mode. Here t is measured in seconds within the single operation mode. N is the number of repetitions in the repetitive operational mode and its unit is an index k. As shown in figure 3.1, the repetitive operations of a robot are two operational modes based. In the single operational mode, the robot operates in a finite time domain of [0, T]. Adaptive controllers and other classical controllers are designed based on this single operational mode. Most adaptive controllers are assumed asymptotically convergent theoretically, which means as t approaches infinite, tracking errors tend to zero. Since operational time in the single operational mode is finite, adaptive controllers never achieve a zero tracking error practically. In addition, adaptive controllers cannot guarantee that the estimated parameters will converge to their true values. Thus, the tracking errors will be repeated also as a robot repeats its task. Iterative learning controllers are designed in the repetitive operational mode. It is well known that they have unsatisfactory performance in initial trials because they only guarantee convergence in repetitive operational mode. In other words, both adaptive controllers and iterative learning controllers are one operational mode based controllers. Thus, they have inherent drawbacks. Here, the author proposes a two operational modes based controller for robotic manipulators. This controller has convergence properties in both operational

modes. In the single operational mode, this controller is equivalent to an adaptive controller and in the repetitive operational mode, the controller is the same as an iterative learning controller. A state in the theory of two operational modes is denoted by X(t,k), or simply, $X(t_k)$, where t represents time in the single operational mode and k, the sequential number in the repetitive operational mode. In the proposed scheme, the controller signals are generated by using information from both operational modes. For example, control signals for the $(k+1)$th operation are given by the equations below:

$$U_{k+1} = f(U_k, e_k, e_{k+1}) \quad \text{where} \quad e_k = X_k - X_d; \quad e_{k+1} = X_{k+1} - X_d \tag{3.1}$$

This implies that the control signal in the present operation is derived from the operational data of previous operation and current state feedback. It is clearly shown that control signals are two operational modes based. The controller design should utilize the information in both operational modes.

A desired repetitive learning controller should have the following convergence property:

$$\begin{aligned} &\text{1. single operational mode} && \lim_{t\to\infty}[X(t,k)] = X_d(t) \\ &\text{2. repetitive operational mode} && \lim_{k\to\infty}[X(t,k)] = X_d(t) \end{aligned} \tag{3.2}$$

The state variables of a repetitive process are functions defined in a fashion of two operational mode.

CHAPTER FOUR MOTION CONTROL OF ROBOTIC MANIPULATORS

4.1 INTRODUCTION

Adaptive controllers and previous iterative learning controllers are single operational mode based controllers. However, most industrial robots perform repetitive tasks in two operational modes. To address this issue, this chapter proposes a new scheme for motion control of robots based on the theory of two operational modes. The proposed controller has been proven to be convergent in both operational modes. An example study of a two-link robot is also included in this chapter. As matter of a fact, the control issue of two operational modes is widely present in the industrial world. It is hoped that this work is able to intrigue further research on this topic among the control community.

This chapter is organized as follows: in section two, a controller for robotic manipulator is designed, which is convergent in the single operational mode. In section 3, a controller that converges in the repetitive operational mode is developed. An example study is given in section 4. Finally, some conclusions are given.

4.2 Design of Controller in Single Operational Mode

In this section, the author designs a controller convergent in the single

operational mode, which is mainly based on references [23] and [38].

Consider a robotic manipulator expressed by the following nonlinear dynamic model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u(t) \tag{4.1}$$

where $M(q)$ is the n×n symmetric and positive definite matrix (also called the generalized inertia matrix, $C(q, \dot{q})\dot{q}$ is the n dimensional vector due to coriolis and centripetal forces, $g(q)$ is the n dimensional vector due to gravitational forces. $u(t)$ is the n dimensional vector of joint torques supplied by the actuator. $q(t)$ is the n dimensional vector of joint positions.

The controller design will use the following properties[62].

$$\|M(q)\| \le \alpha_M \tag{4.2}$$

$$\|C(q,\dot{q})\| \le \alpha_C \|\dot{q}\| \tag{4.3}$$

$$\|g(q)\| \le \alpha_g \tag{4.4}$$

α_M, α_C and α_g are positive constants.

Since, $M(q)$ and $g(q)$ are bounded and continuous , they are both Lipschitz Functions,

$$\|M(q_1) - M(q_2)\| \le L_M \|q_1 - q_2\| \tag{4.5}$$

$$\|g(q_1) - g(q_2)\| \le L_g \|q_1 - q_2\| \tag{4.6}$$

where L_M and L_g are Lipschitz constants.

Also, the following properties are used to design the controller. From the derivation of dynamic equation (4.1) [8], we have

$$C(x_1,x_2)x_3=\begin{bmatrix} x_2{}^T N^1(x_1)x_3 \\ \cdots\cdots \\ x_2{}^T N^i(x_1)x_3 \\ \cdots\cdots \\ x_2{}^T N^n(x)x_3 \end{bmatrix} \tag{4.7}$$

where x_1 is assumed an position variable and x_2 and x_3 are assumed velocity variables and

$$N^i(x_1)=\frac{1}{2}\left[\frac{\partial M_i}{\partial x_1}+(\frac{\partial M_i}{\partial x_1})^T-\frac{\partial M}{\partial x_{1i}}\right] \tag{4.8}$$

Since x_1 will be assumed an position variable and $N^i(x_1)$ has the same form as $M(q)$, we have the following expression from eq.(4.2)

$$\|N^i(x_1)\| \le \frac{\alpha_N}{n} \quad \text{for} \quad i=1,\cdots,n. \tag{4.9}$$

where α_N is a positive constant. From eq.(4.5)

$$\|N^i(q_k)-N^i(q_d)\| \le \frac{L_N}{n}\|e_k\| \quad \text{for} \quad i=1,\cdots,n. \tag{4.10}$$

where L_N is the Lipschitz constant.

Having these properties, we begin to design the controller in the single operational mode. Assume that a robotic manipulator performs repetitive tasks

along a pre-defined desired trajectory in a fixed time period [0, T]. For the kth operation of the robot, we have

$$v_k = \dot{q}_d - \lambda(q_k - q_d) = \dot{q}_d - \lambda e_k \tag{4.11}$$

Define

$$r_k = \dot{q}_k - \dot{v}_k = \dot{e}_k + \lambda e_k \tag{4.12}$$

where λ is a positive diagonal matrix.

Also define:

$$S_k = M(q_k)\dot{v}_k + C(q_k, \dot{q}_K)v_k + g(q_k) \tag{4.13}$$

$$S_d = M(q_d)\ddot{q}_d + C(q_d, \dot{q}_d)\dot{q}_d + g(q_d) \tag{4.14}$$

In the following, we present an important lemma.

Lemma 4.1: with the above definitions, we have

$$\| S_k - S_d \| \leq \rho(\|r_k\| + \|r_k\|\|e_k\| + \|e_k\|^2 + \|e_k\|) \tag{4.15}$$

where ρ is a bounded, positive number.

Proof: It is obvious that

$$\|S_k - S_d\| \leq \|M(q_k)\dot{v}_k - M(q_d)\ddot{q}_d\| + \|C(q_k, \dot{q}_k)v_k - C(q_d, \dot{q}_d)\dot{q}_d\| + \|g(q_k) - g(q_d)\| \tag{4.16}$$

Using Schwartz Inequality and properties (4.2) - (4.6), we have

$$\|g(q_k) - q(q_d)\| \leq L_g \|q_k - q_d\| = L_g \|e_k\| \tag{4.18}$$

$$\begin{aligned}\|M(q_k)\dot{v}_k - M(q_d)\ddot{q}_d\| &= \|M(q_k)(\dot{v}_k - \ddot{q}_d) + [M(q_k) - M(q_d)]\ddot{q}_d\| \\ &\le \|M(q_k)\|\,\|\dot{v}_k - \ddot{q}_d\| + L_M\|e_k\|\,\|\ddot{q}_d\| \\ &\le \alpha_M\lambda\,\|r_k - \lambda e_k\| + L_M\|\ddot{q}_d\|\,\|e_k\| \\ &\le \alpha_M\|r_k\| + (\alpha_M\lambda + L_M\|\ddot{q}_d\|)\,\|e_k\|\end{aligned} \tag{4.17}$$

Also, we have

$$\begin{aligned}C(q_k,\dot{q}_k)v_k - C(q_d,\dot{q}_d)\dot{q}_d &= C(q_k,\dot{q}_k)v_k - C(q_k,\dot{q}_d)v_k \\ &\quad + C(q_k,\dot{q}_d)v_k - C(q_k,\dot{q}_d)\dot{q}_d \\ &\quad + C(q_k,\dot{q}_d)\dot{q}_d - C(q_d,\dot{q}_d)\dot{q}_d\end{aligned} \tag{4.19}$$

Using eq.(4.7) and (4.8), we have

$$\begin{aligned}\|C(q_k,\dot{q}_k)v_k - C(q_k,\dot{q}_d)v_k\| &\le \|\dot{q}_d - \dot{q}_k\|\Big(\sum_{i=1}^{n}\|N^i(q_k)\|\Big)\,\|v_k\| \\ &\le \|r_k - \lambda e_k\|\,\alpha_N\|\dot{q}_d - \lambda e_k\| \\ &\le \alpha_N\|\dot{q}_d\|\,\|r_k\| + \alpha_N\lambda\,\|r_k\|\,\|e_k\| + \alpha_N\lambda\,\|\dot{q}_d\|\,\|e_k\| + \alpha_N\lambda^2\,\|e_k\|^2\end{aligned} \tag{4.20}$$

$$\begin{aligned}\|C(q_k,\dot{q}_d)v_k - C(q_k,\dot{q}_d)\dot{q}_d\| &\le \|\dot{q}_d\|\Big(\sum_{i=1}^{n}\|N^i(q_d)\|\Big)\,\|v_k - \dot{q}_d\| \\ &\le \alpha_N\lambda\,\|\dot{q}_d\|\,\|e_k\|\end{aligned} \tag{4.21}$$

$$\begin{aligned}\|C(q_k,\dot{q}_d)\dot{q}_d - C(q_d,\dot{q}_d)\dot{q}_d\| &\le \|\dot{q}_d\|\Big(\sum_{i=1}^{n}\|N^i(q_k) - N^i(q_d)\|\Big)\,\|\dot{q}_d\| \\ &\le L_N\|\dot{q}_d\|^2\,\|e_k\|\end{aligned} \tag{4.22}$$

Combining inequalities from (4.17) to (4.22), we have the following expression

$$\| S_k - S_d \| \leq \rho (\| r_k \| + \| r_k \| \| e_k \| + \| e_k \|^2 + \| e_k \|) \tag{4.23}$$

whereρ is maximum value of constants on the right hand side of inequalities from (4.17) to (4.22).

Consider a control law

$$u_k = \hat{M}_k \dot{v}_k + \hat{C}_k v_k + \hat{g}_k - k_p r_k - f_k \tag{4.24}$$

where $\hat{M}_k$ and C_k represent the estimate of M_k, C_k, respectively.

$$f_k = \beta r_k \| e_k \|^2 \tag{4.25}$$

Define

$$S_k = \hat{M}_k \dot{v}_k + \hat{C}_k v_k + \hat{g}_k \tag{4.26}$$

The following algorithm is proposed to update the estimate S_k

$$\hat{S}_k = \hat{S}_{k-1} - K_L r_k \tag{4.27}$$

K_L is called learning the gain matrix, which is a positive definite, diagonal matrix.

Theorem 4.1: Assume that a robot performs repetitive tasks along a predefined trajectory within known period [0, T] and the control law given by eq.(4.24) and the updating law given by eq.(4.27) are applied to the robot. The following conclusions hold if K_L, k_p, are selected sufficiently large:

The error between the actual and desired trajectory is bounded in time interval [0, T]. Furthermore, tracking errors are asymptotically convergent to zeros

as time approaches infinite, i.e.,

$$\lim_{t\to\infty}[q_d(t)-q_k(t)]=0 \qquad \text{also} \qquad \lim_{t\to\infty}[\dot{q}_d(t)-\dot{q}_k(t)]=0 \tag{4.28}$$

Proof: Substituting the control law given by eq.(4.24) into eq.(4.1) and considering

$$\ddot{q}_k=\dot{v}_k+\dot{r}_k \tag{4.29}$$

we have the following expression

$$\begin{aligned} M_k\dot{r}_k+C_kr_k+k_pr_k&=[\hat{M}_k-M_k]\dot{v}_k+[\hat{C}_k-C_k]v_k+[\hat{g}_K-g_k]-f_k \\ &=\hat{S}_k-S_k-f_k=\bar{S}_k+(S_d-S_k)-f_k \end{aligned} \tag{4.30}$$

where $\bar{S}_k=\hat{S}_k-S_d$

Define:

$$\bar{S}_{k-1}=\hat{S}_{k-1}-S_d \tag{4.31}$$

Thus, from eq.(4.27), we have

$$\bar{S}_k=\bar{S}_{k-1}-K_Lr_k \tag{4.32}$$

Consider the following non-linear equations:

$$\begin{aligned} M_k\dot{r}_k + C_k r_k + k_p r_k &= \bar{S}_k + (S_d - S_k) - f_k \\ \bar{S}_k &= \bar{S}_{k-1} - K_L r_k \\ \dot{e}_k &= r_k - \lambda e_k \end{aligned} \tag{4.33}$$

where k_p , K_L , λ are positive definite, diagonal matrices:

Define a Lyapunov function candidate as

$$V = \frac{1}{2} r_k{}^T M_k r_k + \frac{1}{2} \int_{(k-1)T}^{kT} \bar{S}(t)^T K_L{}^{-1} \bar{S}(t) dt + e_k \lambda^T k_p e_k \tag{4.34}$$

where T is the period. Since the matrix M_k is positive definite, the above equation is a valid Lyapunov function.

Differentiating the above equation with respect to time, we obtain

$$\dot{V} = \frac{1}{2} r_k{}^T \dot{M}_k r_k + r_k{}^T M_k \dot{r}_k + \frac{1}{2} [\bar{S}_k{}^T K_L{}^{-1} \bar{S}_K - \bar{S}_{k-1}{}^T K_L{}^{-1} \bar{S}_{k-1}] + 2 e_k{}^T \lambda^T k_p \dot{e}_k \tag{4.35}$$

In view of eq.(4.30), eq.(4.35) can be rewritten as

$$\begin{aligned} \dot{V} &= \frac{1}{2} r_k{}^T \dot{M}_k r_k - r_k{}^T C_k r_k - r_k{}^T k_p r_k + r_k{}^T (\bar{S}_k - S_k - f_k) \\ &\quad + \frac{1}{2} [\bar{S}_k{}^T K_L{}^{-1} \bar{S}_k - \bar{S}_{k-1}{}^T K_L^{-1} \bar{S}_{k-1}] + 2 e_k{}^T \lambda^T k_p \dot{e}_k \\ &= \frac{1}{2} r_k{}^T [\dot{M}_k - 2C_k] r_k - r_k{}^T k_p r_k + r_k{}^T (\bar{S}_k + S_d - S_k - f_k) \\ &\quad + \frac{1}{2} [\bar{S}_k{}^T K_L^{-1} \bar{S}_k - \bar{S}_{k-1}^T K_L^{-1} \bar{S}_{k-1}] + 2 e_k{}^T \lambda^T k_p \dot{e}_k \\ &= -r_k{}^T k_p r_k + r_k{}^T (\bar{S}_k + S_d - S_k - f_k) \\ &\quad + \frac{1}{2} [\bar{S}_k{}^T K_L^{-1} \bar{S}_k - \bar{S}_{k-1}{}^T K_L^{-1} \bar{S}_{k-1}] + 2 e_k{}^T \lambda^T k_p \dot{e}_k \end{aligned} \tag{4.36}$$

The assumption that matrix [M - $2C$] is skew symmetric is used in the above proof.

Taking into account of the learning algorithm given by eq.(4.32), we have

$$\begin{aligned}\frac{1}{2}[\tilde{S}_k{}^T K_L^{-1}\tilde{S}_k-\tilde{S}_{k-1}{}^T K_L^{-1}\tilde{S}_{k-1}] &= \frac{1}{2}[(\tilde{S}_{k-1}-K_L r_k)^T K_L^{-1}\tilde{S}_k-\tilde{S}_{k-1}{}^T K_L^{-1}(\tilde{S}_k+K_L r_k)] \\ &= \frac{1}{2}[-r_k{}^T\tilde{S}_k-\tilde{S}_{k-1}{}^T r_k] \\ &= \frac{1}{2}[-r_k{}^T\tilde{S}_k-(\tilde{S}_k+K_L r_k)^T r_k] \\ &= -r_k{}^T\tilde{S}_k-\frac{1}{2}r_k{}^T K_L r_k\end{aligned} \tag{4.37}$$

Substituting the results into equation (4.36), we obtain

$$\begin{aligned}\dot{V} &= -r_k{}^T k_p r_k-\frac{1}{2}r_k{}^T K_L r_k+2e_k{}^T\lambda^T k_p\dot{e}_k+r_k{}^T(S_d-S_k)-r_k{}^T f_k \\ &= -\dot{e}_k{}^T k_p\dot{e}_k-e_k{}^T\lambda^T k_p\lambda e_k-\frac{1}{2}r_k{}^T K_L r_K+r_k{}^T(S_d-S_k)-\beta\|r_k\|^2\|e_k\|^2\end{aligned} \tag{4.38}$$

From Lemma 4.1, we have

$$\begin{aligned}r_k{}^T(S_k-S_d) &\le \|r_k{}^T\|\|S_k-S_d\| \le \rho(\|r_k\|^2+\|r_k\|^2\|e_k\|+\|r_k\|\|e_k\|^2+\|r_k\|\|e_k\|) \\ &\le \rho(\frac{5}{4}\|r_k\|^2+\frac{1}{4}\|e_k\|^2+2\|r_k\|^2\|e_k\|^2 \\ &\quad -\|e_k\|^2[\frac{1}{2}-\|r_k\|]^2-\|r_k\|^2[\frac{1}{2}-\|e_k\|]^2+\|r_k\|\|e_k\|)\end{aligned} \tag{4.39}$$

Substituting eq.(4.39) into eq.(4.38) and selecting $\beta>2\rho$, we have

$$\dot{V}\le -\dot{e}_k{}^T k_p\dot{e}_k-e_k{}^T\lambda^T k_p\lambda e_k-\frac{1}{2}r_k{}^T K_L r_k+\rho(\frac{5}{4}\|r_k\|^2+\frac{1}{4}\|e_k\|^2+\|r_k\|\|e_k\|) \tag{4.40}$$

Define

$$\lambda_{\min} = \min(\lambda^2 \min(\lambda_p), \min(\lambda_L)) \tag{4.41}$$

where λ_p are the eigen values of k_p and λ_L are eigen values of K_L.

Thus, we obtain

$$\begin{aligned}
\dot{V} &\le -\dot{e}_k{}^T k_p \dot{e}_k - \lambda_{\min}\|e_k\|^2 - \frac{1}{2}\lambda_{\min}\|r_k\|^2 + \rho\left(\frac{5}{4}\|r_k\|^2 + \frac{1}{4}\|e_k\|^2 + \|r_k\|\,\|e_k\|\right) \\
&\le -\dot{e}_k{}^T k_p \dot{e}_k - \lambda_{\min}\left(\frac{1}{2}\|e_k\| - \frac{1}{2}\|r_k\|\right)^2 - \frac{1}{2}\lambda_{\min}\|e_k\|\,\|r_k\| + \frac{3}{2}\rho\,\|e_k\|\,\|r_k\| \\
&\quad + \rho\left(\frac{1}{2}\|e_k\| - \frac{1}{2}\|r_k\|\right)^2 - \frac{1}{4}\lambda_{\min}\|r_k\|^2 + \rho\,\|r_k\|^2 - \frac{3}{4}\lambda_{\min}\|e_k\|^2 \le 0
\end{aligned} \tag{4.42}$$

provided that $\lambda_{\min}$ is selected sufficiently large such that

$$\lambda_{\min} \ge 4\rho \tag{4.43}$$

Thus, the trivial equilibrium of eq.(4.33), that is,

$$[e^T, \quad r^T, \quad S^T]^T = 0 \tag{4.44}$$

is globally stable in the sense of Lyapunov. Furthermore, from LaSalle's Theorem, the position error and velocity error converge to zero asymptotically. However, a robot runs only in a finite time interval. Thus, the conclusions are that the tracking errors remain bounded in the time interval [0, T] and we cannot expect zero tracking errors in the time interval [0, T].

4.3 Design of Controller In Repetitive Operational Mode

In the preceding section, we established a controller that converges in the single operational mode. Now, we use its conclusion to derive a controller convergent in the repetitive operational mode, summarized by following theorem.

Theorem 4.2: Assume that a robot performs repetitive tasks along a fixed trajectory within known period T and the control law given by eq.(4.24) and the updating law given by eq.(4.27) are applied to the robot. The following conclusions hold if k_p and K_L are selected sufficiently large:

$$q_k(t) - q_d(t) \to 0 \quad \text{also} \quad \dot{q}_k(t) - \dot{q}_d(t) \to 0 \qquad \text{as} \quad k \to \infty \tag{4.45}$$

Proof: For two consecutive operations, we have from (4.13)

$$\begin{aligned} M_k \dot{r}_k + C_k r_k &= u_k - S_k \\ M_{k-1} \dot{r}_{k-1} + C_{k-1} r_{k-1} &= u_{k-1} - S_{k-1} \end{aligned} \tag{4.46}$$

Considering control laws and learning laws, we have

$$\begin{aligned} u_k &= \hat{S}_k - f_k - k_p r_k \\ u_{k-1} &= \hat{S}_{k-1} - f_{k-1} - k_p r_{k-1} \\ \hat{S}_k &= \hat{S}_{k-1} - K_L r_k \end{aligned} \tag{4.47}$$

Thus, we obtain the following iterative learning law

$$u_k = u_{k-1} - k_p(r_k - r_{k-1}) - K_L r_k - (f_k - f_{k-1}) \tag{4.48}$$

From eq.(4.46), we have

$$M_k(\dot{r}_k-\dot{r}_{k-1})+(M_k-M_{k-1})r_{k-1}+C_k(r_k-r_{k-1})+(C_k-C_{k-1})r_{k-1}=u_k-u_{k-1}-(S_k-S_{k-1}) \tag{4.49}$$

Define

$$\Delta r_k=r_k-r_{k-1} \tag{4.50}$$

and

$$\Delta_k=(S_k-S_{k-1})+(f_k+f_{k-1})+(M_k-M_{k-1})\dot{r}_{k-1}+(C_k-C_{k-1})r_{k-1} \tag{4.51}$$

Therefore, eq.(4.49) can be rewritten as

$$M_k\Delta\dot{r}_k+C_k\Delta r_k+k_p\Delta r_k=-K_L r_k-\Delta_k \tag{4.52}$$

Define a Lyapunov Function

$$V_k(t)=\int_0^t r_k{}^T(\tau)K_L r_k{}^T(\tau)d\tau \tag{4.53}$$

For simplicity, we shall drop t in the following expressions.

$$\begin{aligned}\Delta V_k=V_k-V_{k-1}&=\int_o^t r_k{}^T K_L r_K d\tau-\int_0^t r_{k-1}{}^T K_L r_{k-1}d\tau\\&=-\int_0^t \Delta r_k{}^T K_L\Delta r_k d\tau+2\int_0^t \Delta r_k{}^T K_L r_k d\tau\end{aligned} \tag{4.54}$$

From eq.(4.52), we have

$$2\int_0^t \Delta r_k{}^T K_L r_k d\tau = -2\int_0^t \Delta r_k{}^T [M_k \Delta \dot{r}_k + C_k \Delta r_k + K_p \Delta r_k + \Delta_k] d\tau \qquad (4.55)$$

Applying integration by parts, we obtain

$$-2\int_0^t \Delta r_k{}^T M_k \Delta \dot{r}_k d\tau = -2\Delta r_k{}^T M_k \Delta r_k + 2\int_0^t \Delta r_k{}^T M_k \Delta \dot{r}_k d\tau + 2\int_0^t \Delta r_k{}^T \dot{M}_k \Delta r_k d\tau \qquad (4.56)$$

From eq. (4.52), we have

$$2\int_0^t \Delta r_k{}^T M_k \Delta \dot{r}_k d\tau = -2\int_0^t \Delta r_k{}^T K_L r_k d\tau - 2\int_0^t \Delta r_k{}^T k_p \Delta r_k d\tau - 2\int_0^t \Delta r_k{}^T [\Delta_k + C_k \Delta r_k] d\tau \qquad (4.57)$$

Substituting eq.(4.57) into eq.(4.56), then (4.56) into (4.55) and considering that $2\dot{M}_k - C_k$ is a skew symmetric matrix, we have

$$2\int_0^t \Delta r_k{}^T K_L r_k d\tau = -2\int_0^t \Delta r_k{}^T k_p \Delta r_k d\tau - \Delta r_k{}^T M_k \Delta r_k - \int_0^t \Delta r_k{}^T [\frac{3}{2} C_k \Delta r_k + 2\Delta_k] d\tau \qquad (4.58)$$

Thus, we have

$$\Delta V_k = -\Delta r_k{}^T M_k \Delta r_k - \int_0^t \Delta r_k{}^T K_L \Delta r_k d\tau - 2\int_0^t \Delta r_k{}^T k_p \Delta r_k d\tau - \int_0^t \Delta r_k{}^T [\frac{3}{2} C_k \Delta r_k + 2\Delta_k] d\tau \qquad (4.59)$$

Consider the Δ_k in eq.(4.51) and similar to Lemma 4.1, we have

$$\| S_k - S_{k-1} \| \le \rho^* (\| \Delta r_k \| + \| \Delta r_k \| \| \Delta e_k \| + \| \Delta e_k \|^2 + \| \Delta e_k \|) \qquad (4.60)$$

Also, similar to equations from (4.19) to (4.22), we obtain

Since we select λ_{min} defined by eq.(4.41) in according to eq.(4.43), we have position

$$(C_k - C_{k-1})r_{k-1} = C(q_k, \dot{q}_k)r_{k-1} - C(q_k, \dot{q}_{k-1})r_{k-1}$$
$$+ C(q_k, \dot{q}_{k-1})r_{k-1} - C(q_{k-1}, \dot{q}_{k-1})r_{k-1} \qquad (4.61)$$
$$\leq \alpha_N^* \| \Delta \dot{e}_k \| \| r_{k-1} \| + L_N^* \| \Delta e_k \| \| \dot{q}_{k-1} \| \| r_{k-1} \|$$

$$(M_k - M_{k-1})\dot{r}_{k-1} \leq L_M^* \| \Delta e_k \| \| \dot{r}_{k-1} \| \qquad (4.62)$$

$$f_k - f_{k-1} = \beta (r_k e_k^2 - r_{k-1} e_{k-1}^2) \qquad (4.63)$$

errors and velocity errors bounded in any kth operation (k=1,2,3,....). Also, $M_k(q_k)$ is bounded and invertible, thus we conclude that $\dot{r}_{k-1}$ is bounded also from eq.(4.1). Put equations from (4.60) to eq.(4.63) together, we have $\|\Delta_k\|$ is also bounded. C_k is also obviously bounded. Thus, we can select K_p and K_L sufficiently large to make $\Delta V_k \leq 0$. Thus we have $\Delta r_k \to 0$ as $k \to \infty$. Also, $\Delta e_k \to 0$ and $\Delta \dot{e}_k \to 0$. Thus, $\Delta_k \to 0$, which can be derived easily from the definition (4.51) of Δ_k and equations from (4.60) to (4.63). From eq.(4.52), we can see $r_k \to 0$ as $k \to \infty$. Finally, $e_k \to 0$ and $\dot{e}_k \to 0$, as $k \to \infty$.

4.4 A Case Study

The above controller is applied to motion control of two link planar robot. We assume that the robot has equivalent link length and has mass m_1=2.5 kg and m2=2 kg. We require the robot to following the following desired trajectories

$$q_1^d = \sin(\pi t) \qquad q_2^d = \cos(\pi t)$$

The following figure demonstrate our simulation results.

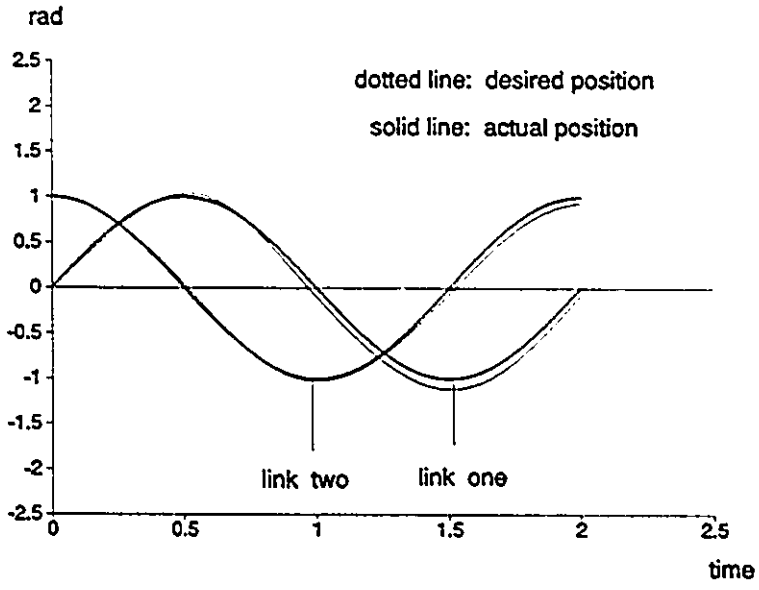

Figure 4.1 the first operation

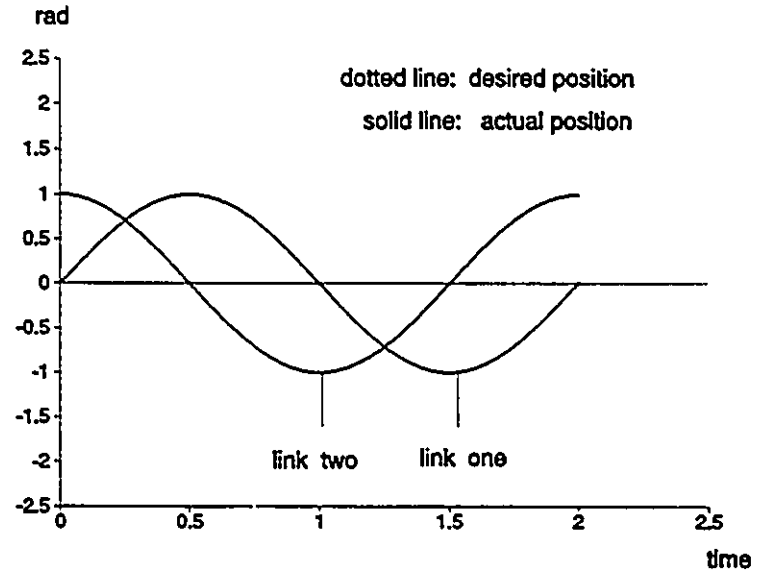

Figure 4.2 the tenth operation

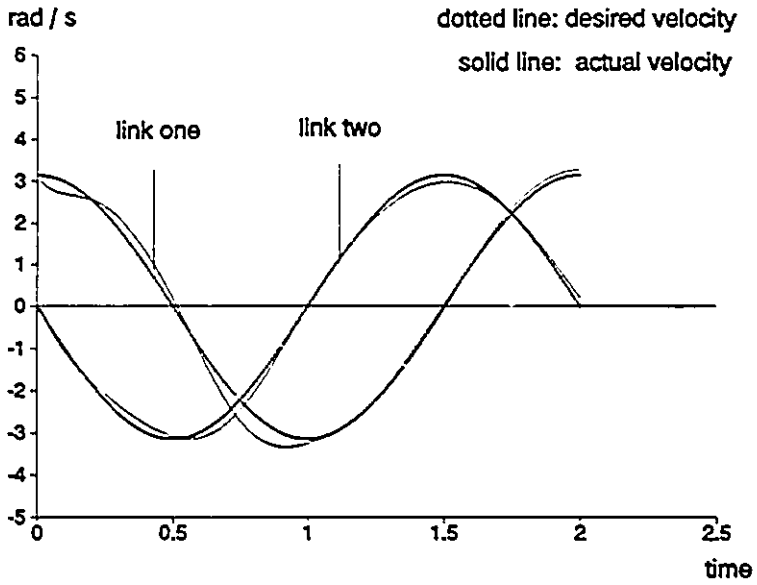

Figure 4.3 the first operation

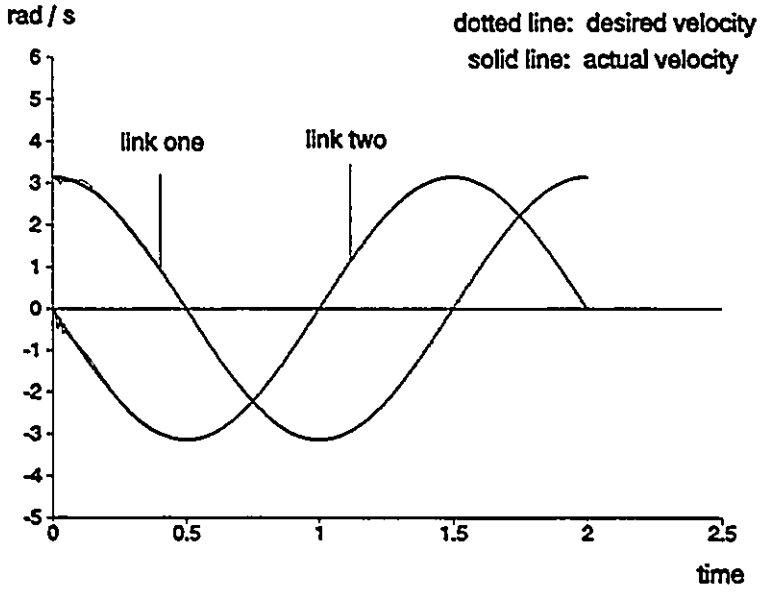

Figure 4.4 the tenth operation

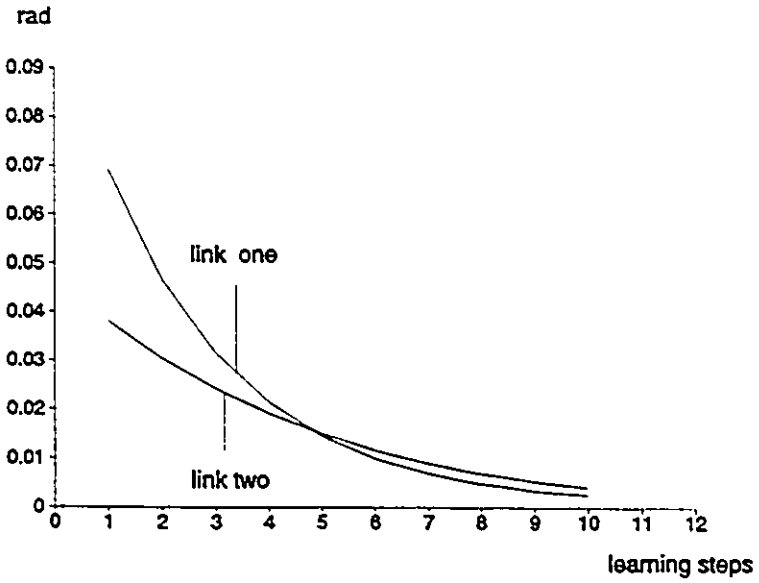

Figure 4.5 position errors

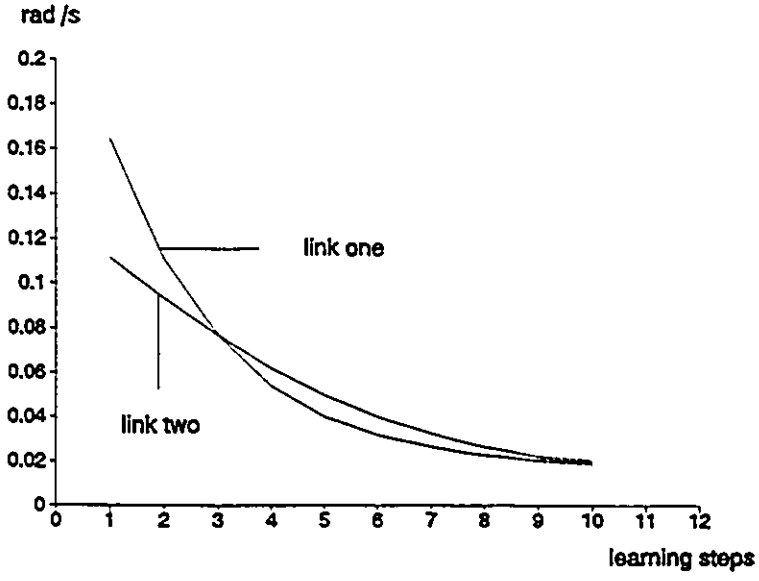

Figure 4.6 velocity errors

5.5 Concluding Remarks

Based on the analysis in the preceding chapter, the repetitive processes of a robot can be divided into two operational modes, a single operational mode and a repetitive operational mode. We design a motion controller for a robot using the proposed theory of the two operational modes. The designed controller has achieved the goal that the tracking errors are asymptotically convergent in both operational modes. In a single operational mode, the designed controller is equivalent to an adaptive controller and an iterative learning controller in the repetitive operational mode. The proposed controller has the advantages of both adaptive controllers and iterative learning controllers, while avoiding their drawbacks.

CHAPTER FIVE CONTROL OF CONSTRAINED ROBOTIC MANIPULATORS

5.1 Introduction

Constrained robot control, as opposed to pure motion control in free space, is concerned with control of a robot when the end effector interacts mechanically with the environment. It is widely recognized that constrained robot control is essential for automation of many industrial tasks such as contour following, deburring, grinding and assembling. For example, if the robot is to turn a crank without pulling on its pivot, its motion should comply with the geometric constraints of the crank. To address this problem, many control strategies have been proposed, such as adaptive control and robust control. However, repetitive learning control of constrained robots has scarcely been studied. This issue is investigated in this chapter.

Consider an n-link rigid constrained robot described in joint space as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u + f \tag{5.1}$$

Where $M(q) \in R^n$ is the inertial matrix, which is symmetric and positive definite

$C(q, \dot{q}) \dot{q} \in R^n$ is a term representing coriolis and centrifugal torques,

$g(q) \in R^n$ is the vector of gravitational torque

$q \in R^n$ is joint angle.

$u \in R^n$ is applied input torque

$f \in R^n$ is the constraint force in joint space

The following is McClamroch's Transformation ofZ Constrained Dynamics[29] . Let $p \in R^n$ denote the generalized position vector of the end-effector in Cartesian Space. Assuming that the constraints imposed are holonomic, the algebraic equation for the constraint surface can be written as:

$$\theta(p) = 0 \tag{5.2}$$

where the mapping $\theta: R^n \rightarrow R^m$ is twice differentiable.

If we assume that the vector p can be expressed in joint space according to the forward kinematic equation

$$p = H(q) \tag{5.3}$$

with mapping H invertible, then the constraint equation in the joint space can be written as:

$$\phi(q) = \theta(H(q)) = 0 \tag{5.4}$$

Also, if we define

$$J_1(q) = \frac{\partial \phi(q)}{\partial q} \tag{5.5}$$

which is the Jacobian matrix of constraint mapping (5.4). Then, since $\phi(q) = 0$ is identically satisfied, it is easily seen that $J_1(q)\ \dot{q} = 0$. Thus the effect of the constraints

on the end-effector can be viewed as restricting the robot dynamics to the manifold by

$$S = [(q, \dot{q}):\ \phi(q) = 0,\ J_1(q)\dot{q} = 0] \tag{5.6}$$

rather than the space R^{2n}.

When the end-effector is moving along the constraint surface, the constraint force in joint space is given by

$$f = J_1^T(q)\lambda \tag{5.7}$$

where $\lambda \in R_m$ is the associated Lagrange multiplier.

Defining the vector partition

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{5.8}$$

where $q_1 \in R_m$ and $q_2 \in R_{n-m}$ (the components of the vector q may be reordered). By implicit function theorem, the constraint equation (5.4) can be expressed explicitly as:

$$q_1 = \Omega(q_2) \tag{5.9}$$

McClamroach and Wang [29] defined a nonlinear mapping $X: R^n \rightarrow R^n$ as:

$$x = X(q) = \begin{bmatrix} q_1 - \Omega(q_2) \\ q_2 \end{bmatrix} \tag{5.10}$$

which is differentiable and has a differentiable inverse transformation

$$q = Q(x) = \begin{bmatrix} x_1 + \Omega(x_2) \\ x_2 \end{bmatrix} \tag{5.11}$$

Define

$$\frac{\partial Q(x)}{\partial x} = \begin{bmatrix} I_M & \dfrac{\partial \Omega(x_2)}{\partial x_2} \\ 0 & I_{n-m} \end{bmatrix} \tag{5.12}$$

which is the nxn Jacobian matrix of the transformation (5.11). Define

$$T(x) = \begin{bmatrix} I_m & a^T \\ 0 & I_{n-m} \end{bmatrix} \tag{5.13}$$

$$\begin{aligned} &\frac{\partial \phi(q)}{\partial q_1}\frac{\partial \Omega(q_2)}{\partial q_2}\dot{q}_2 + \frac{\partial \phi(q)}{\partial q_2}\dot{q}_2 = 0 \\ &a^T = \frac{\partial \Omega(q_2)}{\partial q_2} = -\left[\frac{\partial \phi}{\partial q_1}\right]^{-1}\frac{\partial \phi}{\partial q_2} \end{aligned} \tag{5.14}$$

From equations (5.11) and (5.12), we have

$$\dot{q} = T\dot{x} \tag{5.15}$$

Furthermore

$$\ddot{q} = T\ddot{x} + \dot{T}\dot{x} \tag{5.16}$$

Substitute (5.15) and (5.16) into (5.1), the dynamic motion equation of constrained robot can be expressed in the transformed coordinate as

$$A(x)\ddot{x} + B(x,\dot{x})\dot{x} + G(x) = T^T(u + f) \tag{5.17}$$

where

$$\begin{aligned} A &= T^T M T \\ B &= T^T C T + T^T M \dot{T} \\ G &= T^T g \end{aligned} \tag{5.18}$$

Introducing the partition matrix

$$E = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} = \begin{bmatrix} I_m & 0 \\ 0 & I_{n-m} \end{bmatrix} \tag{5.19}$$

then

$$\begin{aligned} E_2 T^T f = E_2 T^T J_1^T(q)\lambda &= \begin{bmatrix} a & I_{n-m} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial q_1} & \frac{\partial \phi}{\partial q_2} \end{bmatrix}^T \lambda \\ &= \left[\frac{\partial \phi}{\partial q_1} \frac{\partial \Omega(q_2)}{\partial q_2} + \frac{\partial \phi}{\partial q_2} \right]^T \dot{q}_2 \lambda = 0 \end{aligned} \tag{5.20}$$

due to equation (5.14).

Therefore, in the transformed coordinates, the dynamic model of robots, when restricting to the constraint manifold, can be expressed in a so-called

reduced form as:

$$E_1[A(x)\ddot{x} + B(x,\dot{x})\dot{x} + G(x)] = E_1[T^T(u + f)] \tag{5.21}$$

$$E_2[A(x)\ddot{x} + B(x,\dot{x})\dot{x} + G(x)] = E_2 T^T u \tag{5.22}$$

with the m-dimensional constraint equation

$$x_1 = 0 \tag{5.23}$$

Remark: The equations (5.21) to (5.23) express the inherent structure of the dynamics of the constrained robot. The external constraint forces affect the motion of the robot in m directions via eq. (5.21), while eq.(5.22) represents the free motion equation of the robot on the constraint surface.

Property 1:

Motion equation (5.17) is still linear in term of suitable selected parameters, after applying non-linear transformation, i.e.

$$A(x)\ddot{x} + B(x,\dot{x})\dot{x} + G(x) = Y\theta = T^T(u + f) \tag{5.24}$$

Proof: using equations (5.15) to (5.19), we have

$$\begin{aligned} A\ddot{x}+B\dot{x}+G &= T^T M T\ddot{x}+[T^T C T+T^T M\dot{T}]\dot{x}+T^T g \\ &= T^T[M(T\ddot{x}+\dot{T}\dot{x})+C T\dot{x}+g] \\ &= T^T[Mv_1+Cv_2+g] \\ &= T^T Y^*(q,\dot{q},v_1,v_2)\theta \\ &= Y(q,\dot{q},v_1,v_2)\theta \end{aligned} \tag{5.25}$$

where

$$v_1 = T\ddot{x}+\dot{T}\dot{x} \qquad v_2 = T\dot{x} \qquad Y = T^T Y^* \tag{5.26}$$

Property 2. The skew symmetry is preserved for the matrix $\dot{A}$ - 2B in equation (5.17).

proof:

$$\begin{aligned} \dot{A}-2B &= \dot{T}^T M T+T^T M\dot{T}+T^T\dot{M}T-2T^T C T-2T^T M\dot{T} \\ &= \dot{T}^T M T+T^T\dot{M}T-2T^T C T-T^T M\dot{T} \\ &= T^T[\dot{M}-2C]T+[\dot{T}MT-T^T M\dot{T}] \end{aligned} \tag{5.27}$$

Since

$$[\dot{T}^T M T-T^T M\dot{T}]^T = -[\dot{T}^T M T-T^T M\dot{T}] \tag{5.28}$$

and $\dot{M}$ - 2C is a skew symmetric matrix, the matrix $\dot{A}$ - 2B is also skew symmetric.

5.2. Controller Design Within the Single Operational Mode

Based on the transformed form of the robot model (5.17), we can design a controller in the single operational mode. Let x_d be the desired trajectory

of motion in the transformed coordinate as used in eq.(5.17) and f_d be the desired contact force normal to the constraint surface

Define the following according to reference [20]

$$e_m = x - x_d \tag{5.29}$$

$$e_f = \int_o^t (f - f_d)\, d\tau \tag{5.30}$$

$$v = \dot{x}_d - \Lambda_1 e_m \tag{5.31}$$

$$r = \dot{x} - v - \Lambda_2 e_f \tag{5.32}$$

where Λ_1 and Λ_2 are tunable positive definite matrices. In eq.(5.29) and (5.30), e_m is the motion tracking error and e_f is the accumulated force tracking error. The designed controller is composed of a motion control law u_m and force control law u_f. They are given by

$$T^T u_m = \hat{A}\dot{v} + \hat{B}v + \hat{G} - K_D r = Y_1\hat{\theta} - K_D r \tag{5.33}$$

$$T^T u_f = \hat{A}\Lambda_2\dot{e}_f + \hat{B}\Lambda_2 e_f - T^T f = Y_2\hat{\theta} - T^T f \tag{5.34}$$

Now, we consider the kth operation. Substitute the above control law into dynamic equation (5.17), we have

$$\begin{aligned} A_k\dot{r}_k + B_k r_k + K_D r_k &= (\hat{A}_k - A_k)\dot{v}_k + (\hat{B}_k - B_k)v_k + (\hat{G}_k - G_k) \\ &\quad + (\hat{A}_k - A_k)\Lambda_2\dot{e}_{f,k} + (\hat{C}_K - C_k)\Lambda_2 e_{f,k} \\ &= Y_{1,k}\tilde{\theta}_k + Y_{2,k}\tilde{\theta}_k = Y_k\tilde{\theta}_k \end{aligned} \tag{5.35}$$

where

$$\tilde{\theta}_k = \hat{\theta}_k - \theta \tag{5.36}$$

θ is true parameter of the robot, which is a constant vector. $\hat{\theta}_k$ is the estimated parameters at the kth operation.

Also, introduce the following parameter learning law:

$$\hat{\theta}_k = \hat{\theta}_{k-1} - K_L Y_k{}^T r_k \tag{5.37}$$

K_L is a positive diagonal matrix. Subtracting θ from both sides of eq.(37), we have

$$\tilde{\theta}_k = \tilde{\theta}_{k-1} - K_L Y_k{}^T r_k \tag{5.38}$$

Theorem 5.1: Assume that a robot performs repetitive tasks along a fixed trajectory within known period T and the control law given by eq.(5.33) -(5.34) and the updating law given by eq.(5.38) are applied to the robot. The following conclusions hold if K_D and K_L are selected sufficiently large:

$$q_k(t) \to q_d(t) \quad \text{also} \quad f_k(t) - f_d(t) \to 0 \qquad \text{as} \quad t \to \infty \tag{5.39}$$

Proof: Define a Lyapunov function as

Then differentiating V_k with respect to time t, we have

Substitute eq.(5.35) into (5.41). Since $\dot{A}$ - $2B$ is a skew symmetric matrix, we obtain

$$V_k = \frac{1}{2} r_k{}^T A_k r_k + \frac{1}{2} \int_{(K-1)T}^{KT} \tilde{\theta}^T K_L^{-1} \tilde{\theta} dt \tag{5.40}$$

$$\dot{V}_k = \frac{1}{2} r_k{}^T \dot{A}_k r_k + r_k{}^T A_k \dot{r}_k + \frac{1}{2} (\tilde{\theta}_k{}^T K_L^{-1} \tilde{\theta}_k - \tilde{\theta}_{k-1}{}^T K_L^{-1} \tilde{\theta}_{k-1}) \tag{5.41}$$

$$\dot{V}_k = -r_k{}^T K_D r_k + r_k{}^T Y_k \tilde{\theta}_k + \frac{1}{2} (\tilde{\theta}_k{}^T K_L^{-1} \tilde{\theta}_k - \tilde{\theta}_{k-1}{}^T K_L^{-1} \tilde{\theta}_{k-1}) \tag{5.42}$$

since

$$\begin{aligned} \tilde{\theta}_k^T K_L^{-1} \tilde{\theta}_k - \tilde{\theta}_{k-1}^T K_L^{-1} \tilde{\theta}_{k-1} &= \tilde{\theta}_k^T K_L^{-1} [\tilde{\theta}_{k-1} - K_L Y_k^T r_k] - [\tilde{\theta}_k + K_L Y_k^T r_k]^T K_L^{-1} \tilde{\theta}_{k-1} \\ &= -\tilde{\theta}_k^T Y_k^T r_k - \tilde{\theta}_{k-1}^T Y_k^T r_k \\ &= -\tilde{\theta}_k^T Y_k^T r_k - [\tilde{\theta}_k + K_L Y_k^T r_k]^T Y_k^T r_k \\ &= -2\tilde{\theta}_k^T Y_k^T r_k - [Y_k^T r_k]^T K_L [Y_k^T r_k] \end{aligned} \tag{5.43}$$

Substituting the above equation into eq.(5.42), we obtain

$$\dot{V} = -r_k^T K_D r_k - [Y_k^T r_k]^T K_L [Y_k^T r_k] \le 0 \tag{5.44}$$

Thus, we have $r_k \to 0$ as $t \to \infty$. Further $e_{m,k} \to 0$ and $f_k \to f_d$ as $t \to \infty$. Since the operation time interval T is limited, the above error signals remain bounded in the time interval [0, T]. Based on this conclusion, we shall demonstrate that the controller is convergent in the repetitive operational mode provided that K_L and K_D are selected sufficiently large.

5.3 Controller Design In Repetitive Operational Mode

In the preceding section, we established a controller that converges in the single operational mode. Now, we use this conclusion to derive a controller convergent in the repetitive operational mode, summarized by following theorem.

Theorem 5.2: Assume that a robot performs repetitive tasks along a fixed trajectory within known period T and the control law given by eq.(5.33) - (5.34) and the updating law given by eq.(5.38) are applied to the robot. The following conclusions hold if K_D and K_L are selected sufficiently large:

$$q_k(t) \to q_d(t) \quad \text{also} \quad f_k(t) - f_d(t) \to 0 \qquad \text{as} \quad k \to \infty \tag{5.45}$$

Proof: For two consecutive operations, we have

$$\begin{aligned} A_k \dot{r}_k + B_k r_k + K_D r_k &= Y_k \tilde{\theta}_k \\ A_{k-1} \dot{r}_{k-1} + B_{k-1} r_{k-1} + K_D r_{k-1} &= Y_{k-1} \tilde{\theta}_{k-1} \end{aligned} \tag{5.46}$$

Considering control laws (5.33) - (5.34) and learning law (5.38), we have

$$\begin{aligned} A_k(\dot{r}_k - \dot{r}_{k-1}) + (A_k - A_{k-1})\dot{r}_{k-1} + B_k(r_k - r_{k-1}) + (B_k - B_{k-1})r_{k-1} + K_D(r_k - r_{k-1}) \\ = Y_k \tilde{\theta}_k - Y_{k-1} \tilde{\theta}_{k-1} \end{aligned} \tag{5.47}$$

Consider that

$$\begin{aligned} Y_k \tilde{\theta}_k - Y_{k-1} \tilde{\theta}_{k-1} &= Y_k(\tilde{\theta}_{k-1} - K_L Y_k^T r_k) - Y_{k-1} \tilde{\theta}_{k-1} \\ &= -[Y_k K_L Y_k^T] r_k + (Y_k - Y_{k-1}) \tilde{\theta}_{k-1} \end{aligned} \tag{5.48}$$

Define

$$\Delta r_k = r_k - r_{k-1} \tag{5.49}$$

and

$$\Delta_k = -(Y_k - Y_{k-1})\tilde{\theta}_{k-1} + (A_k - A_{k-1})\dot{r}_{k-1} + (B_k - B_{k-1})r_{k-1} \tag{5.50}$$

Therefore, eq.(5.47) can be rewritten as

$$A_k \Delta \dot{r}_k + B_k \Delta r_k + K_D \Delta r_k = -K_L^* r_k - \Delta_k \tag{5.51}$$

where

$$K_L^* = Y_k K_L Y_k^T \tag{5.52}$$

Since K_L is positive diagonal matrix, it is obvious that K_L^* is positive definite also.

Define a Lyapunov Function

$$V_k(t) = \int_0^t r_k^T(\tau) K_L^* r_k^T(\tau) d\tau \tag{5.53}$$

For simplicity, we shall drop t in the following expressions.

$$\begin{aligned} \Delta V_k = V_k - V_{k-1} &= \int_o^t r_k^T K_L^* r_K d\tau - \int_0^t r_{k-1}^T K_L^* r_{k-1} d\tau \\ &= -\int_0^t \Delta r_k^T K_L^* \Delta r_k d\tau + 2\int_0^t \Delta r_k^T K_L^* r_k d\tau \end{aligned} \tag{5.54}$$

From eq.(5.51), we have

$$2\int_0^t \Delta r_k{}^T K_L^* r_k d\tau = -2\int_0^t \Delta r_k{}^T [A_k \Delta \dot{r}_k + B_k \Delta r_k + K_D \Delta r_k + \Delta_k] d\tau \tag{5.55}$$

Applying integration by parts, we obtain

$$-2\int_0^t \Delta r_k{}^T A_k \Delta \dot{r}_k d\tau = -2\Delta r_k{}^T A_k \Delta r_k + 2\int_0^t \Delta r_k{}^T A_k \Delta \dot{r}_k d\tau + 2\int_0^t \Delta r_k{}^T \dot{A}_k \Delta r_k d\tau \tag{5.56}$$

From eq.(5.51), we have

$$2\int_0^t \Delta r_k{}^T A_k \Delta \dot{r}_k d\tau = -2\int_0^t \Delta r_k{}^T K_L^* r_k d\tau - 2\int_0^t \Delta r_k{}^T k_D \Delta r_k d\tau - 2\int_0^t \Delta r_k{}^T [\Delta_k + B_k \Delta r_k] d\tau \tag{5.57}$$

Substituting eq.(5.57) into eq.(5.56), then (5.56) into (5.55) and Considering that $\dot{A}_k$ $-2B_k$ is a skew symmetric matrix, we have

$$2\int_0^t \Delta r_k{}^T K_L^* r_k d\tau = -2\int_0^t \Delta r_k{}^T k_D \Delta r_k d\tau - \Delta r_k{}^T A_k \Delta r_k - \int_0^t \Delta r_k{}^T [\frac{2}{3} B_k \Delta r_k + 2\Delta_k] d\tau \tag{5.58}$$

Thus, we have

$$\Delta V_k = -\Delta r_k{}^T A_k \Delta r_k - \int_0^t \Delta r_k{}^T K_L^* \Delta r_k d\tau - 2\int_0^t \Delta r_k{}^T k_D \Delta r_k d\tau - \int_0^t \Delta r_k{}^T [\frac{2}{3} B_k \Delta r_k + 2\Delta_k] d\tau \tag{5.59}$$

Since the control input (5.33) - (5.34) and learning law (5.38) are applied to the eq.(5.1), we have position errors and velocity errors bounded in any kth operation (k=1,2,3,....). Also, $A_{k-1}(q_{k-1})$ is bounded and invertible, thus we conclude that $\dot{r}_{k-1}$ is also bounded from eq.(5.17). From eq.(5.50), we also have $\|\Delta_k\|$ is bounded. B_k is

also obviously bounded. Thus, we can select K_D and K_L^* sufficiently large to make $\Delta V_k \leq 0$. Thus we have $\Delta r_k \to 0$ as $k \to \infty$. Also, $\Delta e_{m,k} \to 0$, $\Delta \dot{e}_{m,k} \to 0$ and $\Delta e_{f,k} \to 0$. Thus, $\Delta_k \to 0$, which can be derived easily from the definition (5.50) of Δ_k. From eq.(5.51), we can see $r_k \to 0$ as $k \to \infty$. Finally, $e_{m,k} \to 0$, $\dot{e}_{m,k} \to 0$ and $e_{f,k} \to 0$, as $k \to \infty$. Thus, $f_k \to f_d$, as $k \to \infty$.

5.4 A Case Study

A two-link robotic manipulator with a circular path constraint, as given in [20], is used to verify the validity of the control approach described in this chapter. The dynamic model of the robot is expressed in [20].

Define the following parameter:

$$p_1 = m_1 L^2 \quad p_2 = m_2 L^2 \quad p_3 = m_1 L \quad p_4 = m_2 L \tag{5.60}$$

Also define

$$\dot{z} = \dot{v} + \Lambda_2 \dot{e}_f \qquad z = v + \Lambda_2 e_f$$

The constraint is a circle in the work space (the x - y plane) whose centre coincides with the axis of rotation of the first link. The constraint surface is expressed mathematically as

$$\phi(p) = x^2 + y^2 - r^2 = 0 \qquad p = [x \quad y]^T \tag{5.62}$$

The transformation from work space to joint space is given by

$$H(q)=\begin{bmatrix} L\cos(q_1)+L\cos(q_1+q_2) \\ L\sin(q_1)+L\sin(q_1+q_2) \end{bmatrix} \tag{5.63}$$

The constraint, when expressed in terms of joint space, is

$$\phi(q)=2L^2+2L^2\cos(q_2)-r^2=0 \tag{5.64}$$

which has a unique constant solution for q_2

$$q_2=\cos^{-1}\left(\frac{r^2-2L^2}{2L^2}\right)=q_2^* \tag{5.65}$$

The Jacobian matrix of (5.5) is

$$J_1(q)=\begin{bmatrix} 0 \\ -2L^2\sin(q_2) \end{bmatrix} \tag{5.66}$$

Therefore, the matrix defined in (5.13) is

$$T(x)=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.67}$$

Then, the elements of matrix Y defined in eq.(5.25) are expressed as

$$Y_{11}=\frac{1}{3}\dot{z};\quad Y_{13}=\frac{1}{2}g\cos(q_1);\quad Y_{14}=\frac{1}{2}gl\cos(q_1+q_2)+g\cos(q_1)$$

$$Y_{12}=\frac{4}{3}\dot{z}_1+\cos(q_2)\dot{z}_1+\frac{1}{3}\dot{z}_2+\frac{1}{2}\cos(q_2)\dot{z}_2-\frac{1}{2}\sin(q_2)\dot{q}_2 z_1-\frac{1}{2}\sin(q_2)(\dot{q}_1+\dot{q}_2)z_2 \quad (5.68)$$

$$Y_{21}=0;\quad Y_{23}=0;\quad Y_{24}=\frac{1}{2}g\cos(q_1+q_2);$$

$$Y_{22}=[\frac{1}{3}+\frac{1}{2}\cos(q_2)]\dot{z}_1+\frac{1}{3}\dot{z}_2+\frac{1}{2}g\cos(q_1+q_2)$$

The above two-link robot is required to follow the defined circle and exert 10 Newtons of force to contacted work piece. During simulation study, estimated parameters are used in the first trial with control parameters kp=[10 5], Λ_1=[10 5] and Λ_2=[0.5 0.5]. The simulated results are shown in figures from 5.1 to 5.3. We can see that force, velocity and position converge along T_x time direction. However, the transient force response is not satisfactory for its huge initial errors. Then we use learning laws to improve the transient performance of force response. Figures from 4 to 6 demonstrate the results after 10 learning steps. It is clearly shown that the transient force errors, velocity errors and position errors are only about 10% of those in the first trial. Figure 7 shows the performance index, average force error in each trial, vs learning steps.

Notes: In following figures, all dotted lines represent link one and link two respectively.

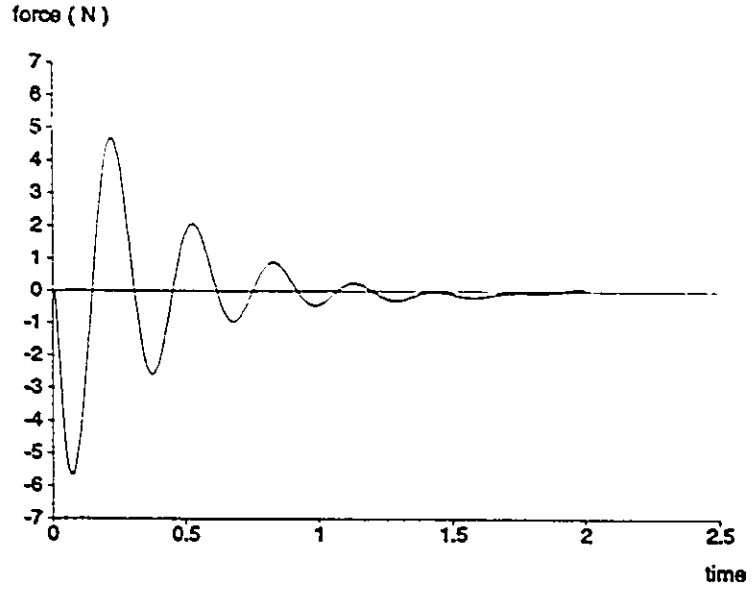

Figure 5.1 Force Error in the First Trial

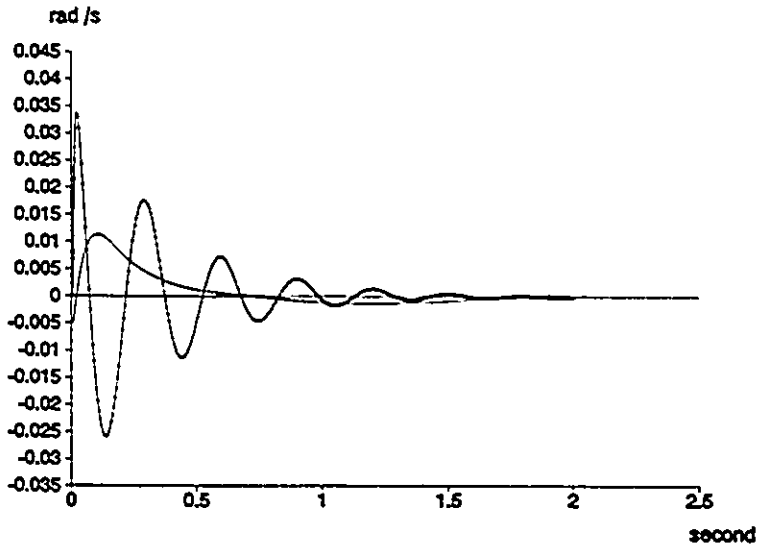

Figure 5.2 Velocity Errors in the First trial

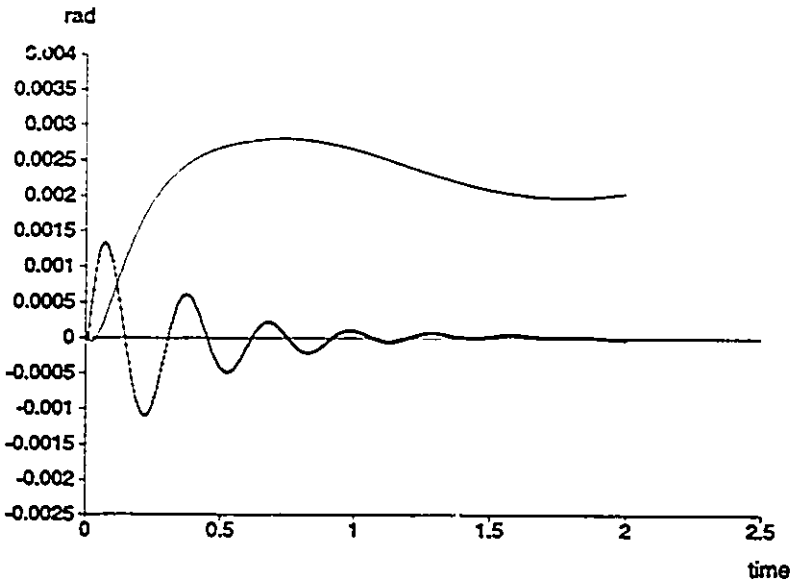

Figure 5.3 Position Errors in the first Trial

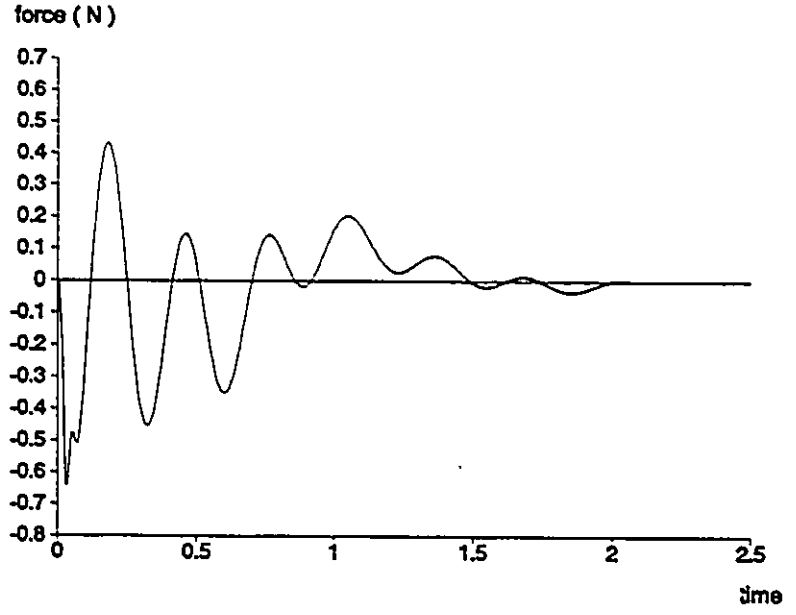

Figure 5.4 Force Error in the Tenth Trial

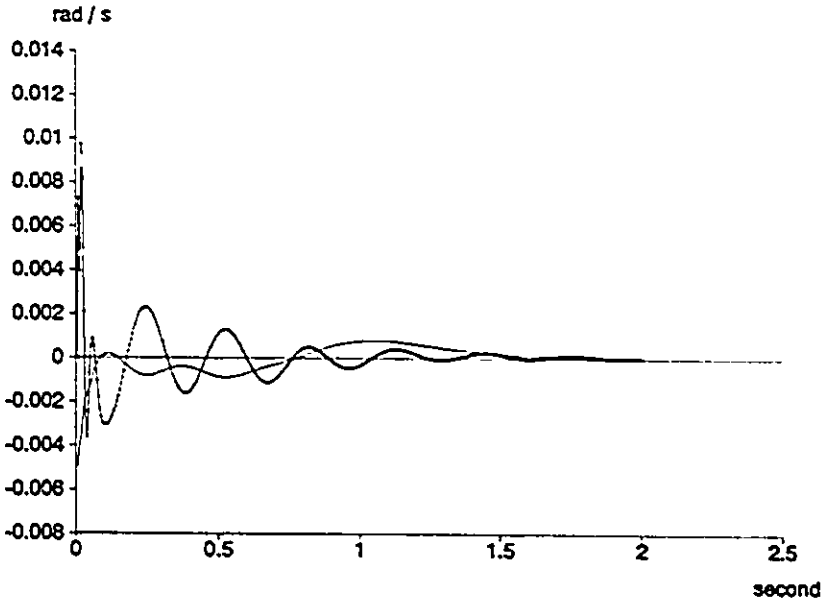

Figure 5.5 Velocity Errors in the Tenth Trial

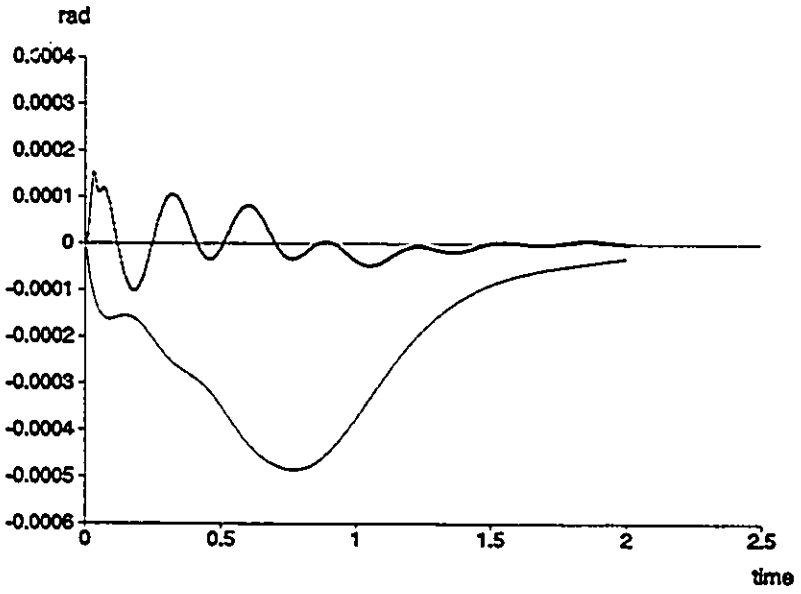

Figure 5.6 Position Error in the Tenth Trial

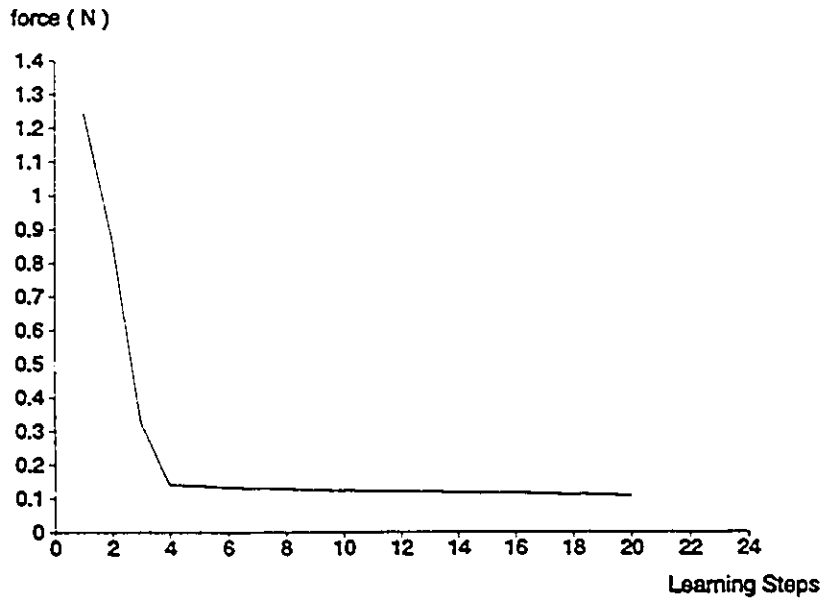

Figure 5.7 Performance Index (Average Force Error)

5.5 Concluding Remarks

In this chapter, we extended the conclusions of chapter 4 on motion control to the case of force control. However, this is not a simple extension. First, we introduced a unique nonlinear transformation to decompose the task into pure motion control and pure force control. Based on this decomposition, we design the force controller for a robot. The proposed controller has outstanding features that the convergence of tracking error, associated with motion and force, are ensured. One of the advantages of the designed controller is the ability to improve the initial errors dramatically after a few trials. This is seldom achieved by conventional controllers.

CHAPTER SIX MOTION CONTROL OF FLEXIBLE-JOINTS ROBOTS

6.1 Introduction

In this chapter, an approach is presented for motion control of robotic manipulators with flexible joints. Compared with the large volume of literature available on the control of rigid robots, relatively little has been published on the control of flexible joint robots. On the other hand, experiments indicate that the joint flexibility should be considered in both modelling and control if high performance is to be achieved. Due to the increased complexity of the dynamics of flexible joint robots, some nice features associated with rigid robots, have been lost. Thus, adaptive controllers for rigid robots cannot be simply extended to flexible joint robots. In this chapter, a new adaptive control strategy is presented for flexible joint robots. The controller is asymptotically stable and prior knowledge of joint flexibility need not be assumed. Measurement of acceleration is not needed.

In this chapter, section 2 discusses an effective controller design for a flexible-joint robot. Section 3 develops a new adaptive controller for flexible joint robots. Section 4 presents a simulation study on a single link robot with flexible joints. Finally, section 5 gives some concluding remarks and some open questions.

6.2 A New Controller for Flexible-joints robots

Case 1: A Control Singularity Case Study

Consider a flexible joint robotic manipulator defined by the following equations [3]:

$$M(q_1)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) = 0 \tag{6.1}$$

$$J\ddot{q}_2 - K(q_1 - q_2) = u \tag{6.2}$$

where $q_1 \in R^n$: link angles;

$q_2 \in R^n$: motor angles;

$K \in R^{n\times n}$: a positive diagonal matrix, representing the joint stiffness;

$M(q_1) \in R^{n\times n}$: inertial matrix, which is positive;

$C(q_1, \dot{q}_1)\dot{q}_1 \in R^n$: representing the coriolis and centrifugal terms;

$g(q_1) \in R^n$: representing the gravitational terms;

$u \in R^n$: input torque;

$J \in R^{n\times n}$: actuator inertial matrix, which is positive and diagonal .

The controller design will use the following established properties.

P1: The initial matrix $M(q_1)$ is positive definite.

P2: The constant parameters of interest (1.e., link masses, moments of inertia, etc) in each term of (6.1) appear as coefficients of known functions of the

generalized coordinates.

P3: $C(q_1, \dot{q}_1)$ can be chosen such that, the matrix $\dot{M}(q_1)$-$2C(q_1, \dot{q}_1)$ is skew symmetric.

First, we introduce a controller design for flexible joint robots. Even though it fails on singularity cases, the controller provides valuable hints on how to design an effective controller.

First, we define

$$\begin{aligned} v_1 &= \dot{q}_{1d} - \lambda_1(q_1 - q_{1d}) = \dot{q}_{1d} - \lambda_1 e_1 \\ r_1 &= \dot{q}_1 - v_1 = \dot{e}_1 + \lambda_1 e_1 \end{aligned} \tag{6.3}$$

$$\begin{aligned} v_2 &= \dot{q}_{1d} - \lambda_2(q_2 - q_{1d}) = \dot{q}_{1d} - \lambda_2 e_2 \\ r_2 &= \dot{q}_2 - v_2 = \dot{e}_2 + \lambda_2 e_2 \end{aligned} \tag{6.4}$$

where λ_1 and λ_2 $\epsilon R^{n\times n}$ are positive diagonal matrices.

Thus, from eq.(6.1), we have

$$M\dot{r}_1 + Cr_1 = -K(q_1 - q_2) - M\dot{v}_1 - Cv_1 - g(q_1) \tag{6.5}$$

Define

$$M\dot{v}_1 + Cv_1 + g(q_1) = Y_1\theta_1 \tag{6.6}$$

where $Y_1 \epsilon R^{n\times m}$ is a known function of the generalized coordinates associated with the link and does not include acceleration items. $\theta_1 \epsilon R^m$ is a constant parameter vector of interest associated with the link. From eq.(6.2), we have

$$J\dot{r}_2 = u + K(q_1 - q_2) - J\dot{v}_2 \tag{6.7}$$

Consider the following expression

$$V = \frac{1}{2} r_1{}^T M r_1 + \frac{1}{2} r_2{}^T J r_2 \tag{6.8}$$

Since M and J are positive matrices, V is an eligible Lyapunov function. Differentiating with respect to time, we obtain

$$\begin{aligned} \dot{V} &= \frac{1}{2} r_1{}^T \dot{M} r_1 + r_1{}^T M \dot{r}_1 + r_2{}^T J \dot{r}_2 \\ &= \frac{1}{2} r_1{}^T \dot{M} r_1 + r_1{}^T [-K(q_1 - q_2) - Y_1\theta_1 - Cr_1] + r_2{}^T [u + K(q_1 - q_2) - J\dot{v}_2] \\ &= -r_1{}^T K(q_1 - q_2) - r_1{}^T Y_1\theta_1 + r_2{}^T u + r_2{}^T K(q_1 - q_2) - r_2{}^T J\dot{v}_2 \end{aligned} \tag{6.9}$$

while the property that $\dot{M} - 2C$ is skew symmetric is used in the above processing.

Select $\lambda_1 = \lambda_2 = \lambda$ and consider that

$$\begin{aligned} -(r_1{}^T - r_2{}^T) K(q_1 - q_1) &= -[\dot{q}_1 - \dot{q}_2 + \lambda(q_1 - q_2)]^T K(q_1 - q_2) \\ &= -(q_1 - q_2)^T \lambda^T K(q_1 - q_2) - (\dot{q}_1 - \dot{q}_2)^T K(q_1 - q_2) \\ &= -(q_1 - q_2)^T \lambda^T K(q_1 - q_2) - (\dot{q}_1 - \dot{q}_2)^T K(\dot{q}_1 - \dot{q}_2) \\ &\quad -(\dot{q}_1 - \dot{q}_2)^T K[(q_1 - q_2) - (\dot{q}_1 - \dot{q}_2)] \end{aligned} \tag{6.10}$$

Define

$$\tilde{q} = q_1 - q_2 \qquad \dot{\tilde{q}} = \dot{q}_1 - \dot{q}_2 \tag{6.11}$$

Thus, we have

$$\dot{V} = -\tilde{q}^T \lambda^T K \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} - \dot{\tilde{q}}^T k(\tilde{q} - \dot{\tilde{q}}) + r_2^T (u - J\dot{v}_2) - r_1^T Y_1 \theta_1 \tag{6.12}$$

Select the control torque as

$$u = -K_d r_2 + J\dot{v}_2 + \Delta u \tag{6.13}$$

where $K_d \epsilon R^{n \times n}$ is a positive diagonal matrix. Δu is determined by the following formula

$$r_2^T \Delta u = r_1^T Y_1 \theta_1 + \dot{\tilde{q}}^T K(\tilde{q} - \dot{\tilde{q}}) \tag{6.14}$$

Therefore, we obtain

$$\dot{V} = -\tilde{q}^T \lambda^T K \tilde{q} - \dot{\tilde{q}}^T K \dot{\tilde{q}} - r_2^T K_p r_2 \tag{6.15}$$

From Lasalle's Theorem [8], we have the following conclusion:as t→∞, $r_2 \to 0$, $\tilde{q} \to 0$ and $\dot{\tilde{q}} \to 0$. Finally, we have $q_1 \to q_1^d$ and $\dot{q}_1 \to \dot{q}_1^d$.

However, for the controller singularity case, $r_2 = 0$ in eq.(6.14), we can not come to the above conclusions, because $\dot{V}$ in eq.(6.15) may become positive. Thus stability of the controller cannot be guaranteed.

Case 2: A New Controller Design Without Singularity

To overcome the control singularity mentioned in above section, we shall redesign the controller. First, we define.

$$v_\varepsilon = v_2 - \varepsilon \tag{6.16}$$

$$\dot{v}_\varepsilon = \dot{v}_2 - \dot{\varepsilon} \tag{6.17}$$

$$r_\varepsilon = \dot{q}_2 - v_\varepsilon = r_2 + \varepsilon \tag{6.18}$$

where

$$\varepsilon_i = \alpha_i sgn(r_{2i}) \int_{t_1}^{t} \|r_{1i}\| d\tau \qquad \alpha_i > 0 \tag{6.19}$$

$$t_1 = \begin{cases} 0 & 0<t\le\delta \\ t-\delta & t>\delta \end{cases} \tag{6.20}$$

where δ is a small positive constant.

Also, we define

$$sgn(r_{2i}) = \begin{cases} 1 & r_{2i}\ge 0 \\ -1 & r_{2i}<0 \end{cases} \tag{6.21}$$

Since the derivative of $sign(r_{2i})$ does not exist at $r_{2i}=0$, we define

$$\frac{d[sgn(r_{2i})]}{dt} = 0 \qquad \text{at} \quad r_{2i} = 0 \tag{6.22}$$

Thus, we have

$$\dot{\varepsilon} = \alpha_i sgn(r_{2i}) [\|r_{1i}(t)\| - \|r_{1i}(t_1)\|] \tag{6.23}$$

Putting the above definitions together, we have the following important lemma:

Lemma 6.1: If $r_\varepsilon(t) = 0$, defined by eq.(6.18), then we have $r_2(t) = 0$ and

$r_1(\bullet) = 0$ in the time interval of $[t_1, t] = 0$.

This can be easily justified, since r_2 and ε have the same sign.

Now, we are able to design a new controller to avoid the control singularity by using Lemma 6.1.

Theorem 6.1. The system described by eq.(6.1) and eq.(6.2) is globally stable, i.e., as $t \to \infty$, $q_1 \to q_1^d$ and $\dot{q}_1 \to \dot{q}_1^d$; under control of the following input

$$u = -r_\varepsilon^T K_d r_\varepsilon + J\dot{v}_\varepsilon - K(q_1 - q_2) + \Delta u \tag{6.24}$$

where $\Delta u = [\Delta u_1, ..., \Delta u_n]^T$ is determined as

$$\Delta u_i = \begin{cases} \dfrac{r_{1i}[K_i(q_{1i} - q_{2i}) + Y_{1i}\theta_1]}{r_{\varepsilon_i}} & r_{\varepsilon_i \neq 0} \\ 0 & r_{\varepsilon_i = 0} \end{cases} \tag{6.25}$$

and Y_{1i} denotes the ith row of matrix Y_1.

Proof: Combining eq.(6.2) and eq.(6.18), we have

$$J\dot{r}_\varepsilon = u + K(q_1 - q_2) - J\dot{v}_\varepsilon \tag{6.26}$$

Define a new Lyapunov function as

$$V = \frac{1}{2} r_1^T M r_1 + \frac{1}{2} r_\varepsilon^T J r_\varepsilon \tag{6.27}$$

Differentiating V with respect to time, we obtain

$$\begin{aligned}\dot{V} &= \frac{1}{2} r_1^T \dot{M} r_1 + r_1^T M \dot{r}_1 + r_\varepsilon^T J \dot{r}_\varepsilon \\ &= \frac{1}{2} r_1^T \dot{M} r_1 + r_1^T [-K(q_1 - q_2) - Y_1 \theta_1 - C r_1] + r_\varepsilon^T [u + K(q_1 - q_2) - J \dot{v}_\varepsilon] \\ &= -r_1^T [K(q_1 - q_2) + Y_1 \theta_1] + r_\varepsilon^T [u + K(q_1 - q_2) - J \dot{v}_\varepsilon]\end{aligned} \tag{6.28}$$

Select control law as

$$u = -K_d r_\varepsilon + J \dot{v}_\varepsilon - K(q_1 - q_2) + \Delta u \tag{6.29}$$

where $K_d = diag[K_{di}] \in R^{n \times n}$ is positive and Δu will be determined later.

Thus we have

$$\begin{aligned}\dot{V} &= -r_\varepsilon^T K_d r_\varepsilon - r_1^T [K(q_1 - q_2) + Y_1 \theta_1] + r_\varepsilon^T \Delta u \\ &= -\sum_{i=1}^{n} K_{di} r_{\varepsilon_i}^2 - \sum_{i=1}^{n} r_{1i} [K_i (q_{1i} - q_{2i}) + Y_{1i} \theta_1] + \sum_{i=1}^{n} r_{\varepsilon_i} \Delta u_i = \sum_{i=1}^{n} \dot{V}_i\end{aligned} \tag{30}$$

where

$$\dot{V}_i = -K_{di} r_{\varepsilon_i}^2 - r_{1i} [K_i (q_{1i} - q_{2i}) + Y_{1i} \theta_1] + r_{\varepsilon_i} \Delta u_i \tag{6.31}$$

Now, we consider two different cases:

a) if $r_{\varepsilon i} \neq 0$, we select

$$\Delta u_i = \frac{r_{1i} [K_i (q_{1i} - q_{2i}) + Y_{1i} \theta_1]}{r_{\varepsilon_i}} \tag{6.32}$$

Then, $\dot{V}_i = -K_{di} r_{\varepsilon i}^2 < 0$.

b) if $r_{ei}=0$, from Lemma 6.1 we have $r_{1i}=0$. Thus, select $\Delta u_i=0$, then $\dot{V}_i=0$.

Consider a) and b) together, we have

$$\dot{V}=\sum_{i=1}^{n}\dot{V}_i=-r_e{}^T K_d r_e \le 0 \tag{6.33}$$

Further, from Lasalle Theorem [8], we come to the following conclusions:as $t\to\infty$, $r_e\to 0$. From Lemma 6.1, we obtain $r_2\to 0$ and $r_1\to 0$. Finally, we have $q_1\to q_1^d$ and $\dot{q}_1\to\dot{q}_1^d$.

Notes: a) Δu_i is continuous at $r_e=0$. It can be explained as below: Assuming that $r_{ei}\to 0$ as $t\to t^*$, then from Lemma 1, we have $r_{1i}=0$ in the time interval of $[t_1, t^*]$. Since r_{1i} is continuous, we have $r_{1i}(t^*)=0$. Thus, $r_{1i}\equiv 0$ in the time interval of $[t_1, t^*]$. and we have $\Delta u_i\to 0$ as $r_{ei}\to 0$.

b) The conclusions appear contradictory as $q_1\to q_1^d$ and $\dot{q}_1\to\dot{q}_1^d$;$q_2\to q_1^d$ and $\dot{q}_2\to\dot{q}_1^d$. This leads to $q_1=q_2$ and leaves link variables uncontrollable. However, we can chose control parameters in such a way that link variables converge faster and allow motor variables to have some tracking errors. For example, we can select $\lambda_2=0$, thus only motor velocity feedback is used. This is similar to the corrective method used in reference [4]. However, in our simulation study, we found out that choosing λ_2 as a small number has advantages in some occasions.

6.3 An Adaptive Version

In the above sections, we have derived a control law that makes a flexible-joint robot globally asymptotically stable with parameters exactly known.

However, we should not expect that we have identified the exact parameters of the flexible-joint robot in practice. Thus, we need to develop an adaptive version of the above control law. With adaptation of unknown parameters, we can achieve the stability results.

Now, we define

$$J\dot{v}_\varepsilon = Y_2\theta_2 \tag{6.34}$$

where Y_2 and θ_2 are expressed by

$$Y_2 = diag[\dot{v}_{\varepsilon_i}] \qquad \theta_2 = [J_1 \cdots J_i \cdots J_n]^T \tag{6.35}$$

Also, we define

$$K(q_1 - q_2) = Y_3\theta_3 \tag{6.36}$$

where

$$Y_3 = diag[q_{1i} - q_{2i}] \qquad \theta_3 = [K_1 \cdots K_i \cdots K_n]^T \tag{6.37}$$

Denote $\hat{\theta}_1$, $\hat{\theta}_2$, $\hat{\theta}_3$ as the corresponding estimated parameters of the true values θ_1, θ_2 and θ_3, where θ_1 is described by eq.(6.6).

Theorem 6.2: The flexible-joint robotic system with unknown parameters described by eq.(6.1) and eq.(6.2) is globally stable with control law

$$u = -K_d r_\varepsilon + Y_2\hat{\theta}_2 - Y_3\hat{\theta}_3 + \Delta u \tag{6.38}$$

where $\Delta u = [\Delta u_1, \ldots, \Delta u_n]^T$ is determined as

$$\Delta u_i = \begin{cases} \dfrac{r_{1i}[Y_{3i}\hat{\theta}_{3i} + Y_{1i}\hat{\theta}_1]}{r_{\varepsilon_i}} & r_{\varepsilon_i \neq 0} \\ 0 & r_{\varepsilon_i = 0} \end{cases} \tag{6.39}$$

and with parameter estimation laws:

$$\begin{cases} \dot{\hat{\theta}}_1 = -\Gamma_1 Y_1^T r_1 \\ \dot{\hat{\theta}}_2 = -\Gamma_2 Y_2^T r_\varepsilon \\ \dot{\hat{\theta}}_3 = -\Gamma_3 (Y_3^T r_1 - Y_3^T r_\varepsilon) \end{cases} \tag{6.40}$$

Proof: Define a Lyapunov Function

$$\dot{V} = \frac{1}{2} r_1^T M r_1 + r_\varepsilon^T J r_\varepsilon + \frac{1}{2}\tilde{\theta}_1^T \Gamma_1^{-1} \tilde{\theta}_1 + \frac{1}{2}\tilde{\theta}_2^T \Gamma_2^{-1} \tilde{\theta}_2 + \frac{1}{2}\tilde{\theta}_3^T \Gamma_3^{-1} \tilde{\theta}_3 \tag{6.41}$$

where $\tilde{\theta}_i = \hat{\theta}_i - \theta_i$ (i=1,2,3) and Γ_i (i=1,2,3) is a positive diagonal matrix.

Differentiate V with respect to time, we obtain

$$\begin{aligned} \dot{V} = & -r_1^T [Y_1\theta_1 + Y_3\theta_3] + r_\varepsilon^T [u + Y_3\theta_3 - Y_2\theta_2] \\ & + \dot{\tilde{\theta}}_1 \Gamma_1^{-1} \tilde{\theta}_1 + \dot{\tilde{\theta}}_2 \Gamma_2^{-1} \tilde{\theta}_2 + \dot{\tilde{\theta}}_3 \Gamma_3^{-1} \tilde{\theta}_3 \end{aligned} \tag{6.42}$$

Applying eq.(6.38) to the above expression, we have

$$\dot{V} = -r_\varepsilon^T K_d r_\varepsilon - r_1^T [Y_1\theta_1 + Y_3\theta_3] + r_\varepsilon^T [Y_3\tilde{\theta}_3 - Y_2\tilde{\theta}_2 + \Delta u] + \dot{\tilde{\theta}}_1 \Gamma_1^{-1} \tilde{\theta}_1 + \dot{\tilde{\theta}}_2 \Gamma_2^{-1} \tilde{\theta}_2 + \dot{\tilde{\theta}}_3 \Gamma_3^{-1} \tilde{\theta}_3 \tag{6.43}$$

Now, apply eq.(6.39), we have

$$\dot{V} = -r_\varepsilon^T K_d r_\varepsilon + (\dot{\tilde{\theta}}_1 \Gamma_1^{-1} + Y_1^T r_1)\tilde{\theta}_1 + (\dot{\tilde{\theta}}_2 \Gamma_2^{-1} + Y_2^T r_\varepsilon)\tilde{\theta}_2 + [\dot{\tilde{\theta}}_3 \Gamma_3^{-1} + Y_3^T (r_1 - r_\varepsilon)]\tilde{\theta}_3 \tag{6.44}$$

Substituting the adaptation law eq.(6.40) into above equation, notice that $r_{\varepsilon i} = 0$, then $r_{1i} = 0$ **(Lemma 6.1)** and $\dot{V}_i = 0$. Thus, we have

$$\dot{V} = -r_\varepsilon^T K_d r_\varepsilon \le 0 \tag{6.45}$$

Further, from Lasalle's Theorem [8], we have the following conclusions:as $t \to \infty$, $r_\varepsilon \to 0$. From Lemma 6.1, we obtain $r_2 \to 0$ and $r_1 \to 0$. Finally, we have $q_1 \to q_1^d$ and $\dot{q}_1 \to \dot{q}_1^d$.

6.4 Simulation Studies

Consider a one link flexible joint robot express by

$$\begin{aligned} I\ddot{q}_1 + MgL\sin(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 - K(q_1 - q_2) &= u \end{aligned} \tag{6.46}$$

In our simulation, the robot is required to follow the following 5th order polynomial trajectory in the time interval of 2.5 seconds.

$$q^d(t) = 1.25t^3 - 0.9375t^4 + 0.1875t^5 \qquad (6.47)$$

The true parameters of the manipulator and its modelled values are shown as below

Parameters	True Values	Estimated Values
Link inertia, I	0.031	0.025
Rotor inertia, J	0.004	0.005
Nominal Load, MgL	0.8	0.6
Joint stiffness, K	50	60

Parameter associated with the adaptive controller.

1. Adaptive gains: $\Gamma_1 = [2\ \ 20]$; $\Gamma_2 = 0.04$; $\Gamma_3 = 10000$.
2. Interval of integration: $\delta = 80$ sampling interval. $\alpha = 10$;
3. Feedback gain: $K_d = 5$; $\lambda_1 = 2.5$; $\lambda_2 = 1.5$.
4. Sampling interval $\Delta t = 0.001$s.

The small sampling interval was chosen to make the integration more accurate. The algorithm itself does not require such a high sampling frequency. Figures 1 to 8 show the simulation results. These figures demonstrate that the proposed adaptive controller is very promising. We can see from the simulation results that link position and velocity have high tracking precision but the corresponding motor variables have relatively large tracking errors.

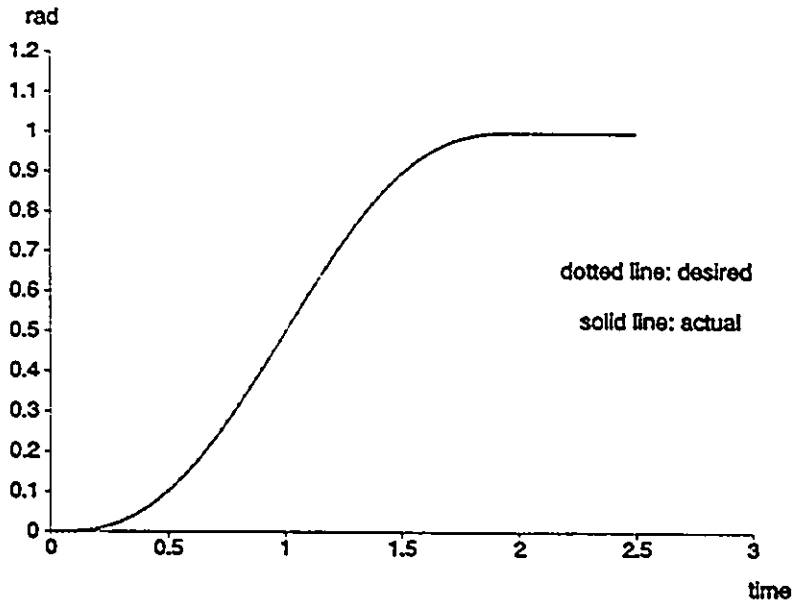

Figure 6.1 Position of Link

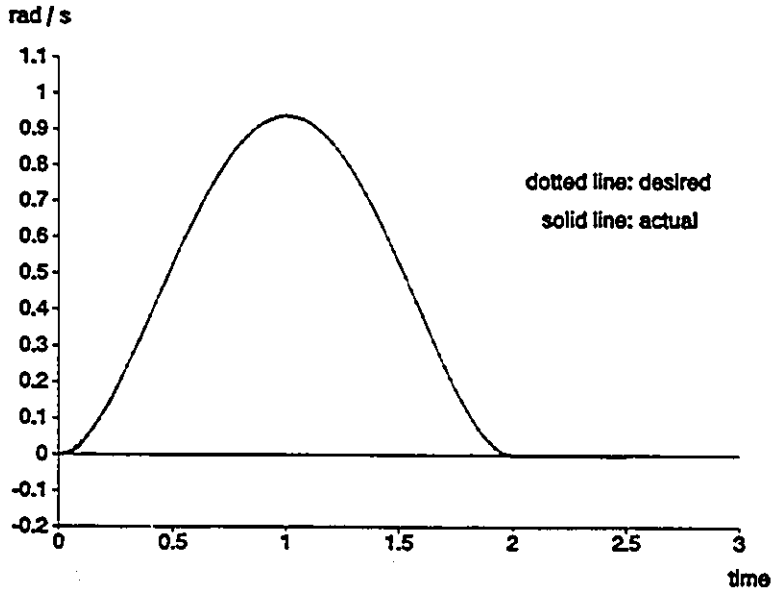

Figure 6.2 Velocity of Link

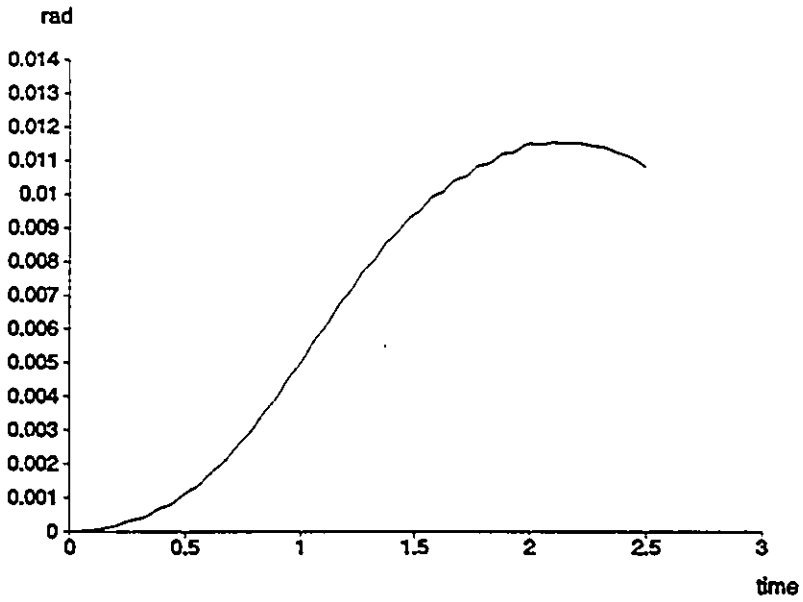

Figure 6.3 Position Error of Motor

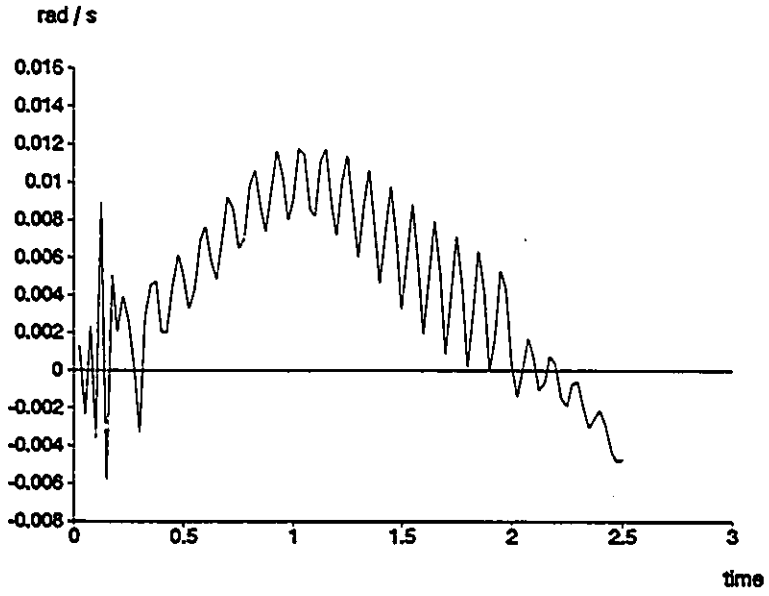

Figure 6.4 Velocity Error of Motor

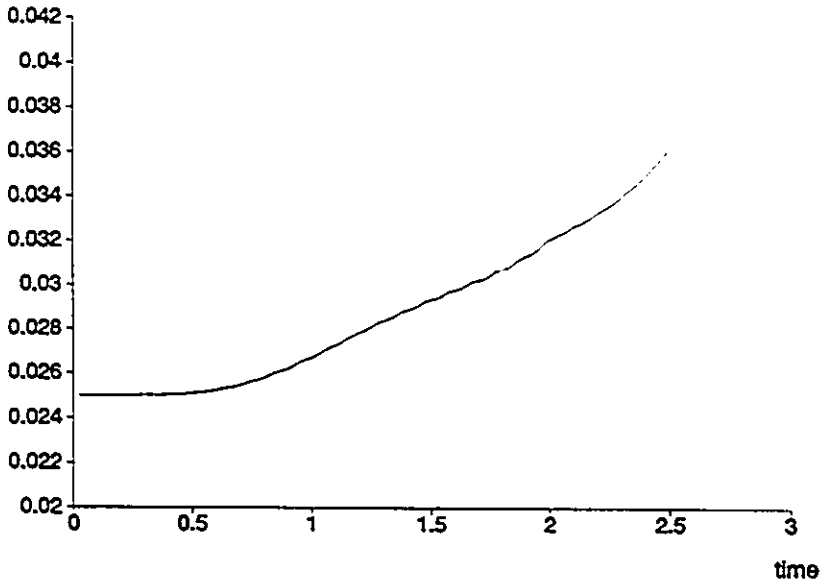

Figure 6.5 Parameter Estimation of I

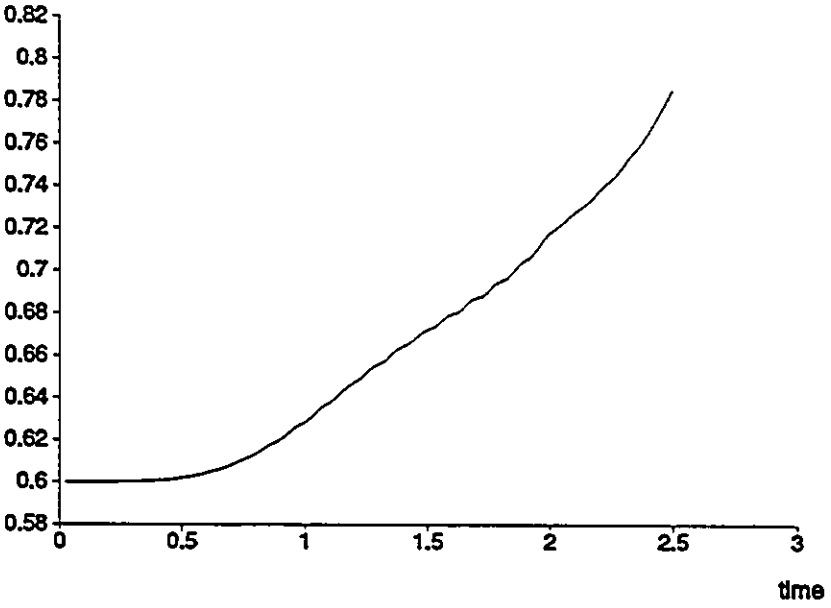

Figure 6.6 Parameter Estimation of mgl

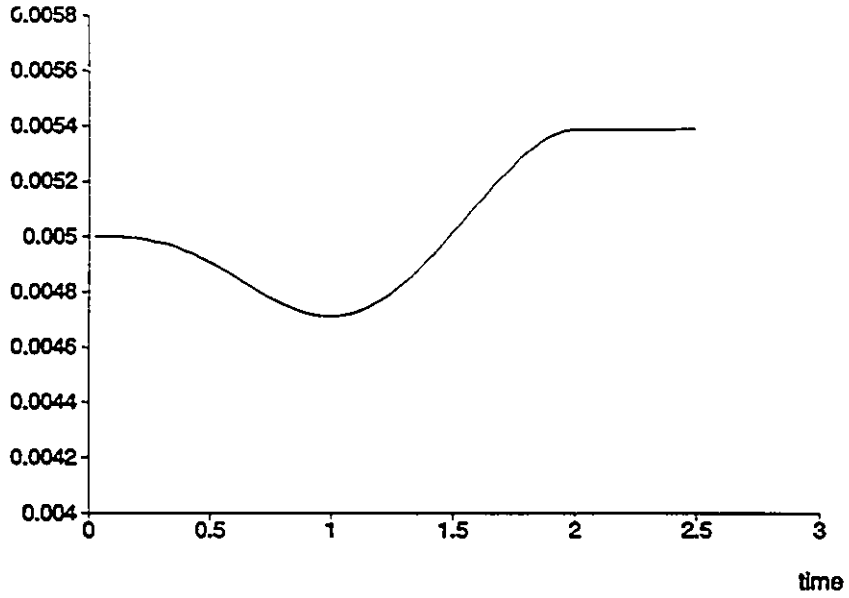

Figure 6.7 Parameter Estimation of J

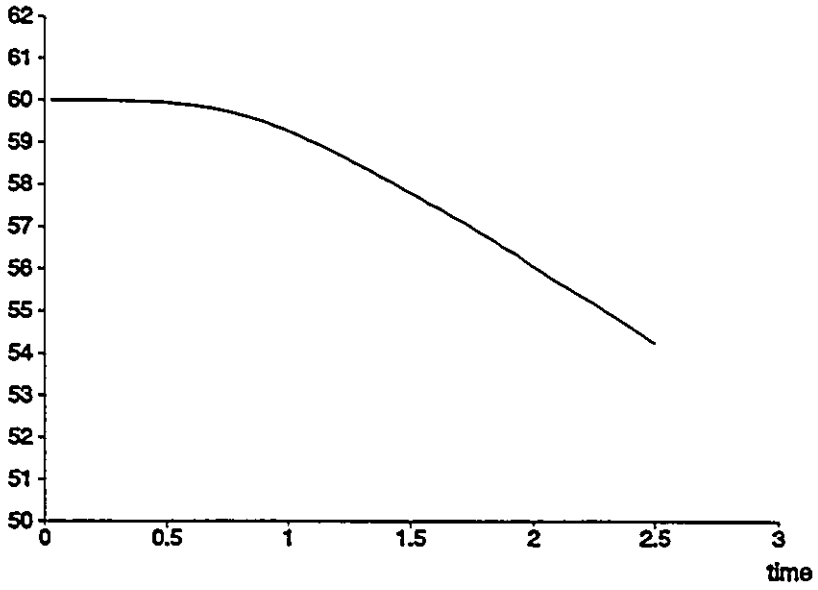

Figure 6.8 Parameter Estimation of Stiffness K

6.5 Concluding Remarks

In this chapter, a new control strategy for flexible joint robots has been presented. Theoretical results and simulation studies have demonstrated that the link variables (position and velocity) are convergent while the corresponding motor variable retain some errors by appropriately selecting control parameters. In our control algorithm, no measurement of acceleration is needed. When the motor position feedback is set to zero, i.e. $\lambda_2=0$, the adaptive controller is almost identical to the corrective control approach described in reference [4]. Also, there are some remaining issues to be resolved. One of them is to find a continuous ε_i subject to the condition that its derivative does not include the derivative of r_1.

PART TWO NEURAL NETWORKS BASED LEARNING APPROACH

In this part, neural networks are applied to repetitive learning control of robotic manipulators. This is a relatively new research area. In the author's view, a neural network is most suitable to this application since it possesses an exceptional repetitive learning ability. However, few researchers are paying attention to this area. Part two of this thesis shall address this issue.

First, the author proposes his own modification of a back-propagation neural network. Compared with previous approaches, the training schemes proposed here have been demonstrated to have a much faster rate of learning and can be easily stabilized. Based on this, neural learning controllers are developed and serve purposes very well. This will be shown in the following chapters.

This part is not devoted solely to research on neural networks. The ultimate goal of this part is to develop an appropriate neural network that can accommodate various needs of robotic controllers. The emphasis of the following neural network research is its applicability to robots.

CHAPTER SEVEN A MODIFICATION OF BACK-PROPAGATION NEURAL NETWORKS

7.1 Introduction

In this chapter, a basic back-propagation neural network is introduced and its shortcomings are investigated. Based on this analysis, the author's own modification of a back-propagation neural network is presented in this chapter. The modified neural network will serve the control purposes studied later.

7.2 A Basic Back-propagation Neural Network

A brief description of the multilayered neural networks used in this part of the thesis is given. Figure 7.1 shows the topological description of the back-propagation neural networks, where the following notation is used :

m: total number of layers including the input and output layers;

N_n: total number of neural nodes in the nth layer;

$Y_i^n(k)$: output of node (n, i) at time k;

(n, i): the ith neuron node in the nth layer;

$W_{i,j}^{n-1}(k)$: connected weight from the node (n-1, j) to node (n, i)

The operation of the node (n,i) in figure 7.2 is expressed by

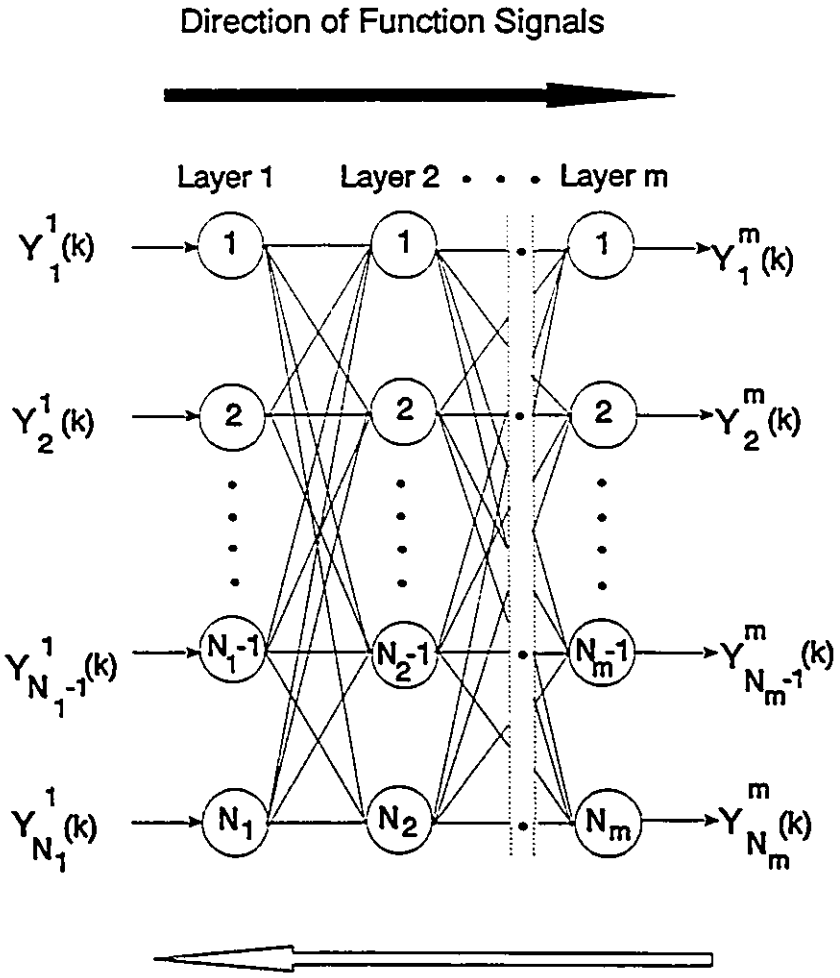

Figure 7.1

$$x_i^n(k) = \sum_{j=1}^{N_{n-1}} W_{ij}^{n-1}(k) \times Y_j^{n-1}(k) \tag{7.1}$$

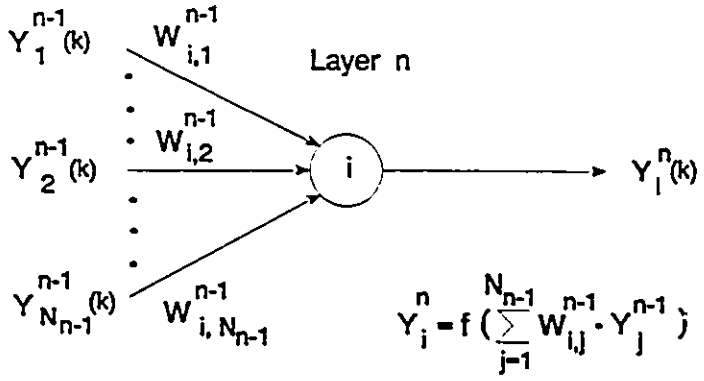

Figure 7.2

$$Y_i^n(k) = f\,[x_i^n(k)] \tag{7.2}$$

In this chapter, the function f associated with hidden layers is assumed to be a sigmoid function

$$f(x) = \frac{1}{1+e^{-x}} \tag{7.3}$$

Consider the following performance index

$$J(l) = [Y_s(l) - Y(l)]^T [Y_s(l) - Y(l)] = e^T(l)e(l) \tag{7.4}$$

where

l: denoting the sequential number of the training data.

$Y(l)$: $N_m \times 1$ vector, outputs of the neural network, as shown in figure 7.2

$Y(l) = [Y_1^m(l) \quad Y_2^m(l) \ldots\ldots Y_{Nm}^m(l)]$;

$Y_s(l)$: $N_m \times 1$ vector, outputs of the real systems.

N_m: number of the output nodes

$e(l)$: $N_m \times 1$ vector, representing the error between the outputs of neural network and the outputs of real systems.

The training procedure of back-propagation neural network is to minimize the performance index (7.4). A typical approach is using the gradient decent method as follows:

$$W_{i,j}^n(l+1) = W_{i,j}^n(l) + {}^n g_{i,j}^n(l) \tag{7.5}$$

where ^{n} is called the learning gain and

$$\begin{aligned} g_{ij}^n(l) &= -\frac{\partial[e^T(l)e(l)]}{\partial W_{ij}^n(l)} \\ &= -e^T(l)\frac{\partial e(l)}{\partial W_{i,j}^n(l)} = e^T(l)\frac{\partial Y(l)}{\partial W_{i,j}^n(l)} \end{aligned} \tag{7.6}$$

From the reference [22], it is obtained

$$\frac{\partial Y(l)}{\partial W_{i,j}^n(l)} = \Delta_i^n(l)x_j^n(l) \tag{7.7}$$

where

$$\Delta_i^n(l) = f'[X_i^{n+1}(l)] \sum_{p=1}^{N_n+2} W_{p,i}^{n+1}(l)\,\Delta_p^{n+1}(l) \qquad (7.8)$$

$$1 \le n \le M-2$$

with the initial condition

$$\Delta_i^{M-1}(l) = [0,\cdots,0,\underset{i}{1},0,\cdots,0]^T \qquad (7.9)$$

7.3 A Modification of Basic Back-propagation Neural Network

In the preceding section, the basic back-propagation algorithm described in reference [9] is introduced. It is widely believed that it has a very slow rate of convergence and that convergence cannot be guaranteed. During the learning process, updating of the connected weights in the neural network is usually based on the minimization of errors over a single input-output training sample, not over the whole training data. It is believed that this results in rather inefficient training of the original back-propagation neural network, which restricts its practical application.

To overcome this problem, a modified back-propagation algorithm is presented in this chapter. In the modified scheme, the input layer and hidden layers are the same, but output layer is different. The function associated with the output layer is a linear function: $f(x)=x$. Since the functions associated with neurons in the output layer are linear, the Linear Least Squares Learning Algorithm can be applied to the training of the output layer. This is expected to have a much faster rate of

convergence than the gradient method. The training of the hidden layers still utilizes the gradient method. However, the gradient used here is with respect to the weights based on the whole training samples rather than a single training sample. Iterative computation of the gradient is studied in order to save computational effort and time in the presence of a large number of training samples.

7.3.1 Training of the hidden layers - gradient method

Assume that there are L training data available. The objective of the neural network training is to minimize the following performance index.

$$J(L)=\sum_{l=1}^{L}[Y_s(l)-Y(l)]^T[Y_s(l)-Y(l)]=\sum_{l=1}^{L}e^T(l)e(l) \tag{7.10}$$

where

L: the number of total training data

l: denoting the sequential number of the training data, $l=1,2,\ldots,L$.

$Y(l)$: $N_m \times 1$ vector, outputs of the neural network,as show in figure 2

$Y(l)=[Y_1^m(l)\ \ Y_2^m(l)\ \ldots\ Y_{Nm}^m(l)]$;

$Y_s(l)$: $N_m \times 1$ vector, outputs of the real systems.

N_m: number of the output nodes

$e(l)$: $N_m \times 1$ vector, representing the error between the outputs of neural network and the outputs of real systems.

Note that the index $J(L)$ is different from that used in the back-

propagation algorithm described in reference [9]. Here, the index is the sum of the square of the errors over the total training samples. Direct computation of the gradient is quite time consuming where there exist tens of thousands of training data. Therefore, it is important to find an efficient computing technique to derive the gradient. The ideal approach is to develop an iterative method to obtain the gradient rather than to compute it directly.

Let $1<k\leq L$, from eq.(7.10) we have

$$J(k)=\sum_{l=1}^{k} e^{T}(l)\,e(l) \tag{7.11}$$

Using the original back-propagation algorithm described in reference [9], we have

$$W_{i,j}^{n}(k+1)=W_{i,j}^{n}(k)+\eta\,\frac{\partial J(k)}{\partial W_{i,j}^{n}(k)} \tag{7.12}$$

where η is the learning rate. From eq.(7.11), we obtain

$$J(k)=\sum_{l=1}^{k} e^{T}(l)\,e(l)=e^{T}(k)\,e(k)+J(k-1) \tag{7.13}$$

Let

$$g_{i,j}^{n}(k)=\frac{\partial[e^{T}(k)\,e(k)]}{\partial W_{i,j}^{n}(k)} \tag{7.14}$$

Since the computing method for $g(k)$ is described in detail in reference [9], we shall omit it here. From equation (7.13), we can obtain

$$\frac{\partial J(k)}{\partial W_{i,j}^{n}(k)} = g_{i,j}^{n}(k) + \frac{\partial J(k-1)}{\partial W_{i,j}^{n}(k)} \tag{7.15}$$

Although the partial differential term on the right hand side of the equation is unknown at the *k*th iteration step, by using Taylor's formula, we have the following approximate expression:

$$\frac{\partial J(k-1)}{\partial W_{i,j}^{n}(k)} = \frac{\partial J(k-1)}{\partial W_{i,j}^{n}(k-1)} + \frac{\partial^2 J(k-1)}{\partial^2 W_{i,j}^{n}(k-1)}[W_{i,j}^{n}(k) - W_{i,j}^{n}(k-1)] \tag{7.16}$$

and the second-order partial derivative can be estimated by the following equation:

$$\frac{\partial^2 J(k-1)}{\partial^2 W_{i,j}{}^{n}(k-1)} = \frac{\dfrac{\partial J(k-1)}{\partial W_{i,j}^{n}} - \dfrac{\partial J(k-2)}{\partial W_{i,j}^{n}}}{W_{i,j}^{n}(k-1) - W_{i,j}^{n}(k-2)} \tag{7.17}$$

where j=1, ... , N.

Substituting eq.(7.15) into eq.(7.12), we have

$$W_{i,j}^{n}(k+1) = W_{i,j}^{n}(k) + {}^{n}g_{i,j}^{n}(k) + {}^{n}\frac{\partial J(k-1)}{\partial W_{i,j}^{n}(k)} \tag{7.18}$$

Comparing with the back-propagation algorithm with additional momentum term described in reference [9], which is expressed by

$$W_{i,j}^{n}(k+1) = W_{i,j}{}^{n}(k) + \eta g_{i,j}^{n}(k) + \alpha[W_{i,j}^{n}(k) - W_{i,j}^{n}(k-1)] \tag{7.19}$$

where *g(k)* is defined by equation (7.14), we find that the additional momentum term in equation (7.19) is trying to accomplish the same tasks that equation (7.18) has

achieved. Eq.(7.19) is a rough approximation of eq.(7.18). From this point of view, it is believed that the proposed training algorithm in this paper is better than that of reference [9]. Our later simulations will confirm this.

Now, we discuss the selection of an important parameter η, the rate of learning. The smaller we make the learning rate , the smaller the changes to the weights in the network will be, and therefore the better will be the approximation. This improvement, however, is obtained at the cost of a slower rate of learning. If, on the other hand, we make the learning rate very large so as to speed up the training process, the resulting large changes in the weights may assume such a form that the network becomes unstable(i.e.,oscillatory). A simple method of increasing the learning rate and yet avoiding the danger of instability is to select the learning rate as:

$$\eta = \frac{\alpha / k}{\| \frac{\partial J(k)}{\partial W_{i,j}^{n}} \|} \tag{7.20}$$

where $\| \frac{\partial J(k)}{\partial W_{i,j}^{n}} \|$ is the Euclidean norm of the vector $\frac{\partial J(k)}{\partial W_{i,j}^{n}}$

and α is a very small positive number.

7.3.2 Training of the output layer - Iterative Least Squares Algorithm

Consider an output neuron node j $(j=1,2, \ldots, N_m)$ and define

$$W_j = [W_{1,j}^{m-1}, \ldots\ldots, W_{i,j}^{m-1}, \ldots\ldots, W_{N_{m-1},j}^{m-1}]^T \tag{7.21}$$

$$h_j = [Y_1^{m-1}, \ldots\ldots, Y_i^{m-1}, \ldots\ldots, Y_{N_{m-1}}^{m-1}]^T \tag{7.22}$$

Thus we have

$$Y_j^m = \sum_{i=1}^{N_{m-1}} W_{i,j}^{m-1} \cdot Y_i^{m-1} = h_j^T \cdot W_j \tag{7.23}$$

The training process of the output weights W_j associated with neuron j can be regarded as adjusting the weights to minimize the following performance index.

$$J_j(L) = \sum_{l=1}^{L} [Y_{sj}(l) - Y_j^m(l)]^2 = \sum_{l=1}^{L} [Y_{sj}(l) - h_j(l)^T W_j(l)]^2 \qquad j=1,2,\ldots\ldots,N_m \tag{7.24}$$

$Y_{sj}(l)$: the jth output of the real system.

We can obtain the optimal weights by directly minimizing the performance index in equation (7.24). However, a better approach is to utilize an iterative algorithm. The iterative Least Square Algorithm is the most suitable since there exists large amount of training data making the direct method impractical.

According to reference [10], it is quite straightforward to obtain the updating formula of Iterative Least Squares Algorithm for the weights in the output layer

$$
\begin{aligned}
W_j(k+1) &= W_j(k) + K_j(k+1)\,[Y_{sj}(k+1) - h_j^T(k+1)W_j(k)] \\
K_j(k+1) &= P_j(k)h_j(k+1)\,[h_j^T(k+1)P_j(k)h_j(k+1) + \mu]^{-1} \\
P_j(k+1) &= \frac{1}{\mu}\,[I - K_j(k+1)h_j^T(k+1)]\,P_j(k)
\end{aligned}
\tag{7.25}
$$

where $j=1,2,.....,N_m$ indicates the index number of the output neuron nodes.

$K_j(k+1)$: least square gain matrix.

μ : forgetting factor:

To start the iterative least squares algorithm, we have to select the initial values of matrix P_j and weights W_j. A simple method is to choose

$$P_j(0) = \gamma^2 I, \qquad W_j(0) = \epsilon \tag{7.26}$$

where γ is very large number and ϵ is a very small vector.

7.3.3 Summary of the training procedures

Combining the training procedures of the hidden layers and the output layer, we derive the modified training algorithm for the backpropagation neural network.

Step 1: Set the initial values, $W_j(0)$ and $P_j(0)$; $j=1,2, ... ,N_m$

Step 2: Present the network with input vectors and output response vectors of the systems for iteration $k=1,2,...,L$;

Step 3: Train the output layer using the iterative Linear Least Squares Algorithm;

Step 4: Train the hidden layers using the modified gradient algorithm;

Step 5: Repeat the computation by going back to step 2 until satisfactory results are achieved.

7.4 Concluding Remarks

In this chapter, a modification of a back-propagation neural network has been presented. This is the milestone for the following study of neural controllers. Compared with the basic-propagation neural network and other modifications, the approach proposed here has demonstrated superior stability and a faster rate of convergence. Based on these features, better controllers are able to be developed for robotic manipulators. The subsequent chapters will study these issues in more detail.

CHAPTER EIGHT MOTION CONTROL OF ROBOTIC MANIPULATORS

8.1 Introduction

In this chapter, repetitive learning control using neural networks has been studied. Simulations of a two-link robot have demonstrated that the proposed control scheme for robotic manipulators can greatly reduce tracking errors. The author's modification of back-propagation algorithm presented in the preceding chapter is employed in the neural network, resulting in a much faster learning rate. The results of simulation have also shown that the proposed repetitive learning controller has a faster rate of convergence and better robustness.

8.2 Discussion of A Learning Control Scheme

Most present neural learning controllers are implemented on-line in a feedback loop. Since neural controllers are multilayered networks consisting of large numbers of neurons, they need much on-line computation which causes problems in a real-time implementation. Here, we use the idea of the iterative learning controller described in reference [24] [25] and the learning capability of neural networks to design a repetitive learning controller, which is motivated by the fact that industrial robots usually perform repetitive tasks.

In the robotic control community, a general learning of neural network

is favoured, in which a neural network is aimed to be trained as an inverse dynamic model of the robot in the entire state space. However, there has been no successful report in literature up to now. There are two major problems facing the general learning of a neural network for robots. One is how to select the training input-out pairs in order to excite all modes of the robotic system and another is concerned with the stability of the training algorithm for the neural network. Due to these unsolved problems, there are some feelings in the robotic control community that the neural controller is not as helpful as expected. However, it is believed that the neural model of robotic manipulator is much superior to linearized models. In this chapter, we manage to solve the problem of stability of the back-propagation neural network by proposing the modified algorithm. Although a strictly theoretical verification is not given, a lot of simulations studied by the author have shown that the proposed modified back-propagation algorithm is easily stabilized by properly choosing the learning rate according to eq.(7.20), resulting in a very fast learning speed. A satisfactory precision can be achieved after a couple of training cycles.

When we apply a neural network to the control of robots, two important things should be taken into consideration. One is that most industrial robots execute tasks along a fixed trajectory. Secondly, a trajectory close to the desired is easily obtained by using a computed torque controller or a PD controller. Thus, our approach is to train the neural network as an inverse dynamic model of the robot in the neighbourhood of the desired trajectory based on the experience

from the realized trajectory. After each operation of the robot, the learning process is repeated and as the number of repeated operations increases, the trajectory tracking errors can be reduced to almost zero. The control system studied is depicted in the following figure

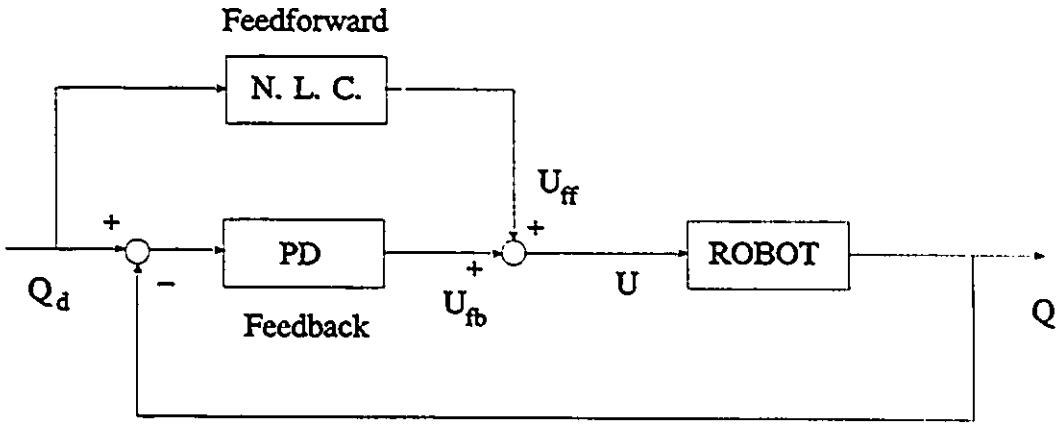

Figure 8.1 Scheme of Neural Learning Controller

The above controller consists of two major parts: a feedback P.D. controller and a feedforward neural learning controller (N.L.C.), which is an approximate inverse dynamic model of the robot in the neighbourhood of the desired trajectory. At the initial control stage, the feedback controller plays a role in making the whole system stable and contributes a relatively larger portion of control inputs to the robot. However, as the learning process continues, the dominant control input to the robot will be shifted to the feed-forward controller, i.e. the neural learning controller, and the feedback controller only plays a function to

suppress the disturbances. We should point out that the weights of the neural controller are fixed during the control period. After each operation of the robot, the neural network will be retrained with the operational data obtained from last trial. Basically, the neural network will be trained as an inverse dynamic model of the robot. The training scheme can be described by figure 8.2.

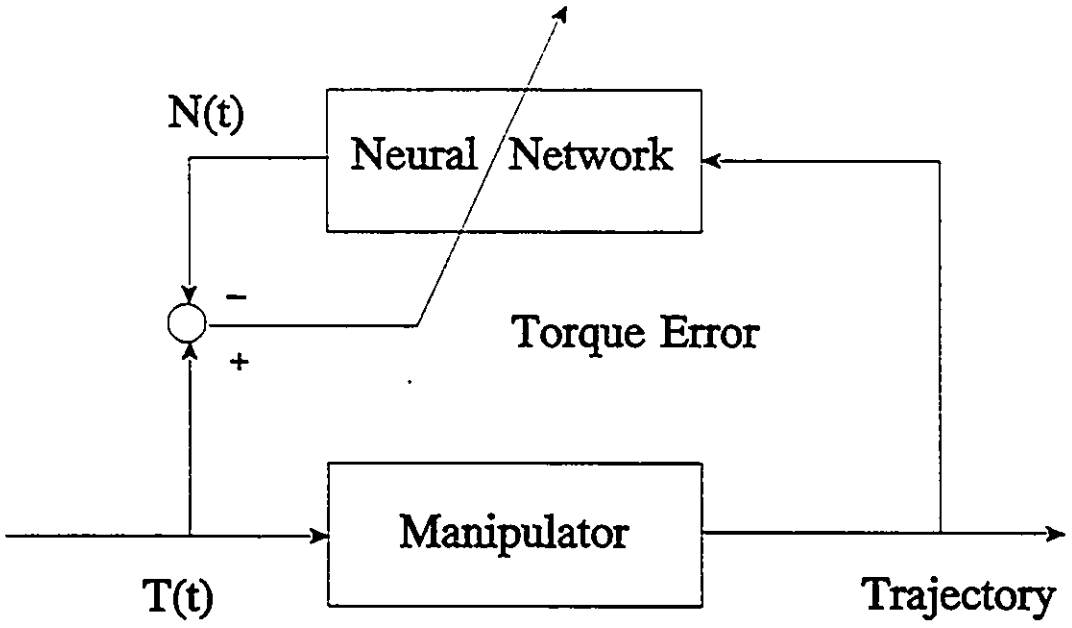

Figure 8.2 training scheme as an inverse dynamic model

During the training process, the output of the robot, i.e. trajectory information, will be fed into the input layer of the neural network. The weights of the neural network are adjusted based on the errors between the input torque of

manipulator T(t) and the output of neural network N(t) by using the modified back-propagation algorithm of the neural network. We point out here that the actual trajectory from the previous operation is fed into the input layer of the neural network during the training process while the desired trajectory is presented in the control process.

8.3 A Teaching Controller

As we know, a teacher is needed to start the training of a back-propagation neural network. Here, we choose an inverse dynamic controller of robots as the teacher of the training process. First, we are going to give a brief review of inverse dynamic control.

Consider a robotic manipulator expressed by the following nonlinear dynamic model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u(t) \tag{8.1}$$

where $M(q)$ is the $n \times n$ symmetric and positive definite matrix (also called the generalized inertia matrix), $C(q, \dot{q})$ is the $n \times 1$ vector due to coriolis and centripetal forces, $g(q)$ is the n×1 vector due to gravitational forces. $u(t)$ is the $n \times 1$ vector of joint torques supplied by actuator. $q(t)$ is the vector of joint positions.

For simplicity, we rewrite the above equation as

$$M(q)\ddot{q} + h(q,\dot{q}) = u(t) \tag{8.2}$$

The idea of inverse dynamic control is to seek a nonlinear feedback control law

$$u = M(q)v + h(q,\dot{q}) \tag{8.3}$$

which results in a linear closed loop system. Since M is invertible, the combined system reduces to a double integrator system

$$\ddot{q} = v \tag{8.4}$$

The term v represents a new input to the system which is yet to be chosen. Since system eq.(8.4) is a simple linear second order system, the obvious choice is to set

$$v = -k_p(q - q_d) - k_v(\dot{q} - \dot{q}_d) + \ddot{q}_d \tag{8.5}$$

where k_p and k_v are diagonal matrices with diagonal elements consisting of position and velocity gains, respectively.

Then the tracking error $e(t) = q - q_d$ satisfies

$$\ddot{e}(t) + k_v\dot{e}(t) + k_p e(t) = 0 \tag{8.6}$$

An obvious choice for the gain matrix k_p and k_v is

$$\begin{aligned} k_p &= diag(\omega_1^2, \cdots\cdots, \omega_n^2) \\ k_v &= diag(2\omega_1, \cdots\cdots, 2\omega_n) \end{aligned} \tag{8.7}$$

This results in a closed loop system which is globally decoupled, with each joint response equal to the response of a critically damped linear second order system with natural frequency ω.

Since the exact parameters of the robot are unknown, instead of

eq.(8.3), the nonlinear control law is actually of the form

$$u(t) = \hat{M}(q)v + \hat{h}(q,\dot{q}) \tag{8.8}$$

where $\hat{M}(q)$ and $\hat{h}(q, \dot{q})$ represent nominal or computed version of $M(q)$, $h(q, \dot{q})$, respectively. The above control law is not exactly the inverse dynamic control law. However, we can obtain a trajectory close to the desired by applying the above control scheme to a robot. This is good enough to start the learning process of the neural controller. Notice that the inverse dynamic controller is applied only in the first operation. After that, the controller implementation is composed of the neural learning controller together with a feedback PD controller. The iterative learning controller with neural network can improve the tracking precision by itself as the number of trials increases.

8.4 Simulation Studies

In this section, the proposed iterative learning controller with neural network is applied to the simulation of a two-link robot. The simulated robotic manipulator is depicted by the following figure.

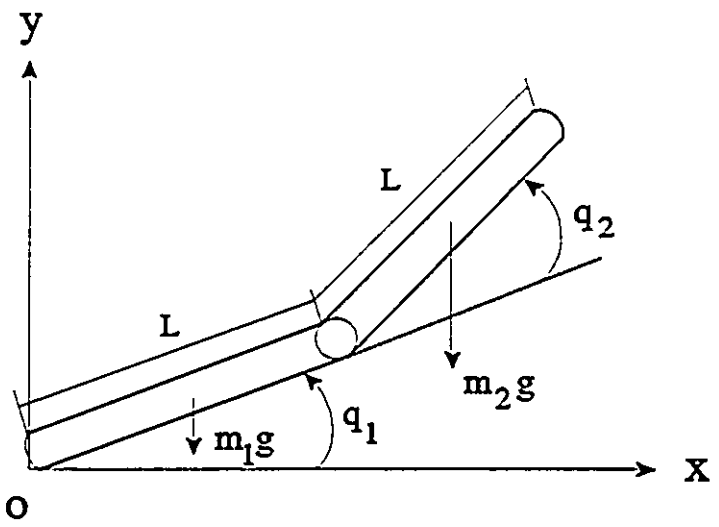

Figure 8.3 configuration of a two-link robot

whose dynamic equation is governed by

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2c_2l^2 & \frac{1}{3}m_2l^2 + \frac{1}{2}m_2c_2l^2 \\ \frac{1}{3}m_2l^2 + \frac{1}{2}m_2c_2l^2 & \frac{1}{3}m_2l^2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}m_2s_2l^2\dot{q}_2^2 - m_2s_2l^2\dot{q}_1\dot{q}_2 \\ \frac{1}{2}m_2s_2l^2\dot{q}_1^2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}m_1glc_1 + \frac{1}{2}m_2glc_{12} + m_2glc_1 \\ \frac{1}{2}m_2glc_{12} \end{bmatrix} \tag{8.9}$$

where

$$\begin{aligned} c_1 &= \cos(q_1) \quad & c_2 &= \cos(q_2) \quad & c_{12} &= \cos(q_1 + q_2) \\ s_1 &= \sin(q_1) \quad & s_2 &= \sin(q_2) \end{aligned} \tag{8.10}$$

The desired trajectory of the robot is expressed as

$$\begin{bmatrix} q_1{}^d(t) \\ q_2{}^d(t) \end{bmatrix} = \begin{bmatrix} 0.5t^2 + 2\sin(2.5t) \\ t + \sin(5t) \end{bmatrix} \tag{8.11}$$

The neural network employed in the simulation consists of an input layer with six neuron nodes, the first hidden layer with 25 neuron nodes, the second hidden layer with 35 neuron nodes and an output layer with 2 neuron nodes, which is symbolized as $N_{6,25,35,2}$.

Since the exact model of the robot is not available, in our simulation, we assume that the parameters of the robotic model are different from the true values as shown in Table 8.1

Table 8.1 **Input Data For Simulation**

Parameters	True Values	Modeled Values
Mass of Link 1	2kg	1.7kg
Mass of Link 2	2.5kg	3.1kg
Length of Link	0.6m	0.8m

As discussed before, we first employ an inverse dynamic control scheme based on an approximate model to bring the robot into the neighbourhood of the desired trajectory. Starting from this point, we proceed with our learning control. The resulting trajectories realized by the inverse dynamic control scheme, based on an approximate model, are shown in figures 8.4 and 8.5. The dotted line represents

the desired trajectory and the solid line represents the realized trajectory. We observe that the performance is not good because there is a significant difference between the modelled parameters of the robot and the true parameters of the robot. However, this is good enough to begin our learning process, in which the learning controller implemented with the neural network is able to improve the control performance by itself and to achieve satisfactory tracking precision after a few trials. Using the operational data obtained from the results of the inverse dynamic control scheme, we train the neural network as an approximate model of the robot as described in figure 8.2. Figure 8.6 displays the training results after two lessons. The dotted line shows the control torque generated by the inverse dynamic controller and the solid line depicts the torque learned by the neural network. These two lines are almost the same showing that the training results are very good.

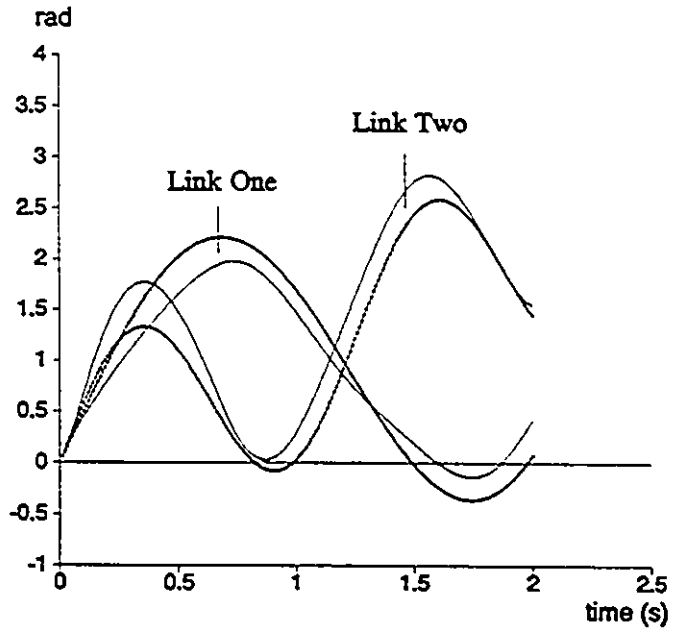

Figure 8.4 trajectory of position (inverse dynamic scheme)

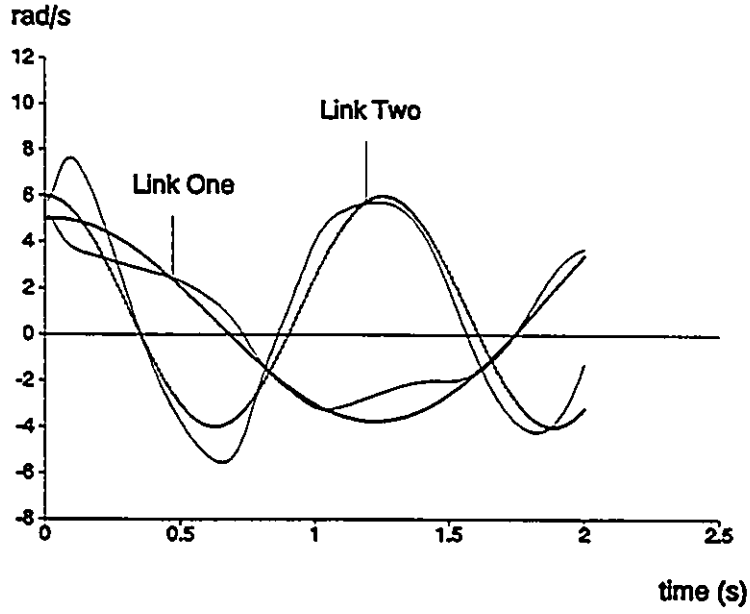

Figure 8.5 trajectory of velocity (inverse dynamic scheme)

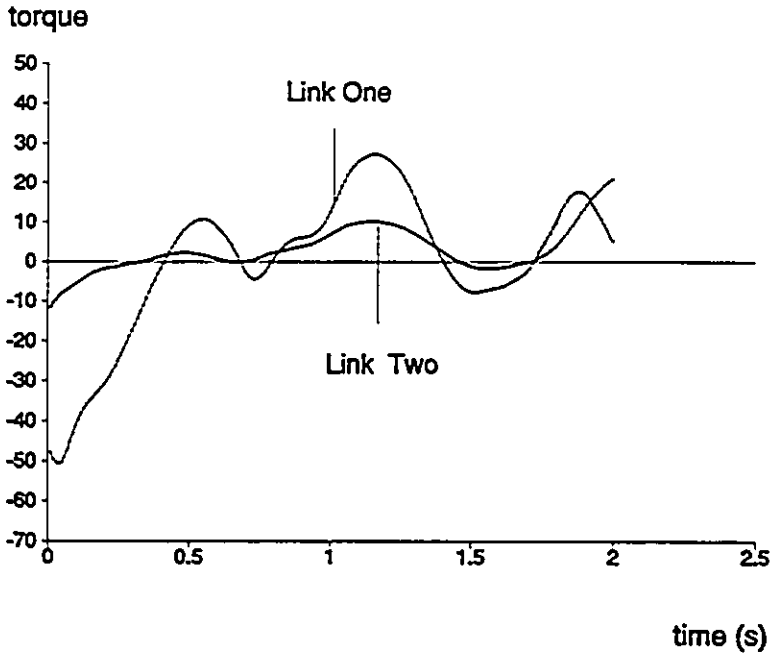

Figure 8.6 training results of the neural network

After finishing the training of the neural network, we implement the repetitive learning controller with a neural network as shown in figure 8.1. Note here that the control signals generated by the neural controller are calculated beforehand off-line and only the feedback PD controller requires on-line computation. In all following figures, dotted lines represent the desired trajectory and solid lines represent the actual trajectory. Except as indicated, horizontal axis is time and vertical axis is radian for position and radian per second for velocity respectively. Figures 8.7 and 8.8 show the simulation results of the first trial. Figures 8.9 and 8.10 display the simulation results of the third trial while figures 8.11 and

8.12 depict the simulation results of the ninth trial. We observe that there is a slight velocity error which is caused by the factor that the desired acceleration is used to train neural network, since we assume the actual acceleration is not available. We can see from these figures that there is a significant improvement of tracking precision as the number of trials increases. Figures 8.13 and 8.14 present the absolute sums of the position tracking errors and the absolute sums of the velocity tracking errors respectively. At the initial stage, the tracking performance improves very quickly, but this rate slows down as time advances. This familiar phenomenon is also observed in classical controllers. Although we assume that most industrial robots are performing repetitive tasks, it is usual that the robotic systems are subject to noise and payload variations arising from the fact that the standard articles handled by the robot may have slightly different mass due to the problems produced in the manufacturing process. Thus, a key problem is how the iterative learning controllers deal with these uncertainties. As expected, our simulations have shown that the iterative learning controller with neural networks has better robustness and is less sensitive to noise than the common iterative learning controllers. Figures 8.15 and 8.16 show the simulation results with 10% variation from the standard payload used in the training process . Figures 8.17 and 8.18 describe the simulation results in the presence of 10% velocity measurement noise and 5% position measurement noise. Figures 8.19 and 8.20 depict the measured position and velocity. We have claimed before that the control action will be gradually shifted from the feedback

controller to the feed-forward neural controller as the learning process takes place. This is confirmed in our simulation results as shown in figures 8.21 and 8.22. The dotted lines describe feedback control signal U_{fb} while the solid lines indicate feedforward control signal U_{ff}.

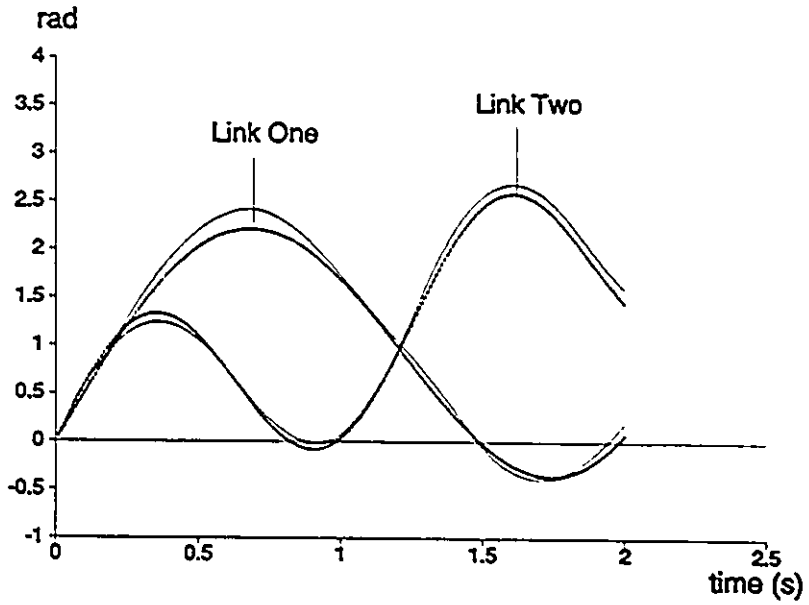

Figure 8.7 trajectory of position (trial 1)

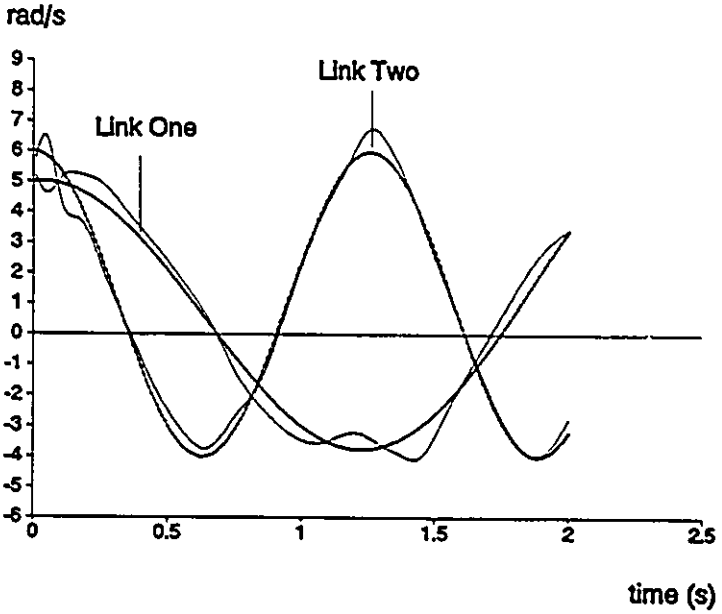

Figure 8.8 trajectory of velocity (trial 1)

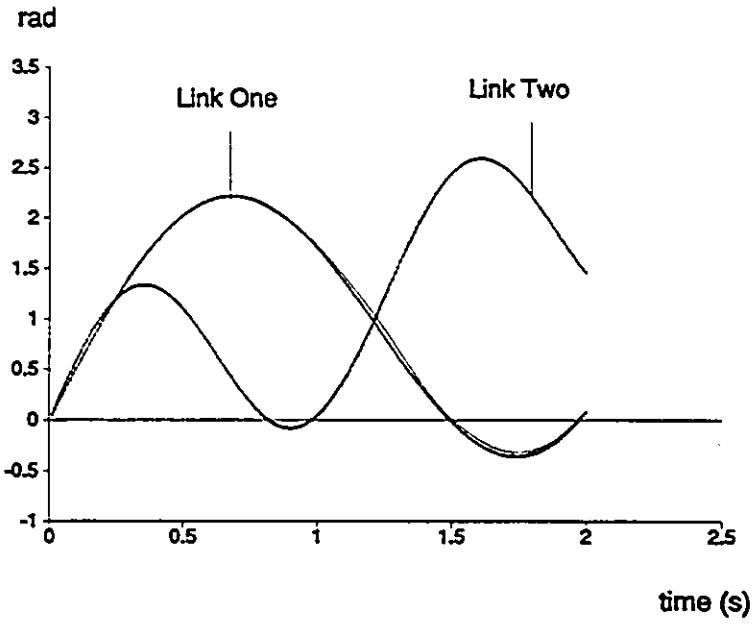

Figure 8.9 trajectory of position (trial 3)

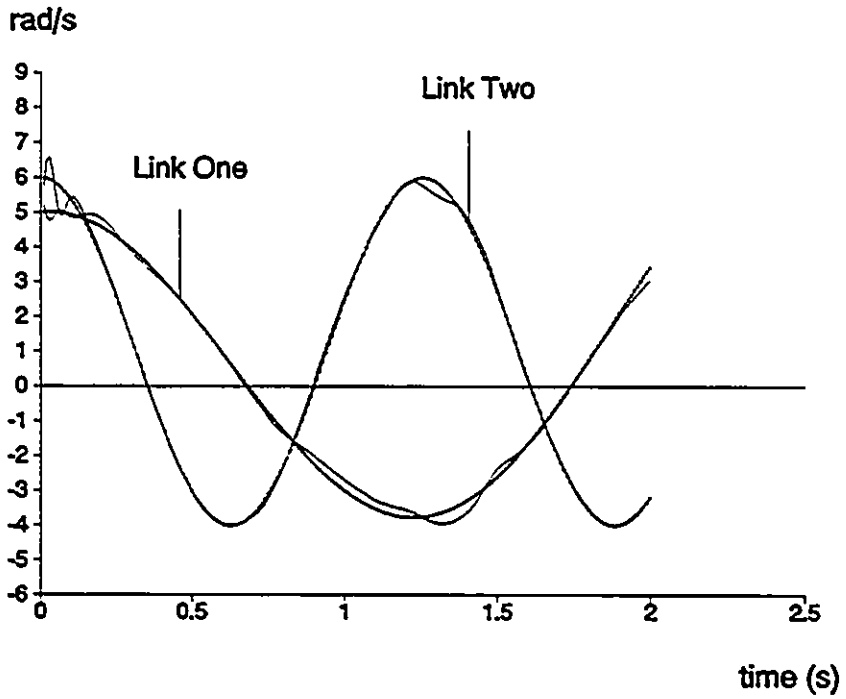

Figure 8.10 trajectory of velocity (trial 3)

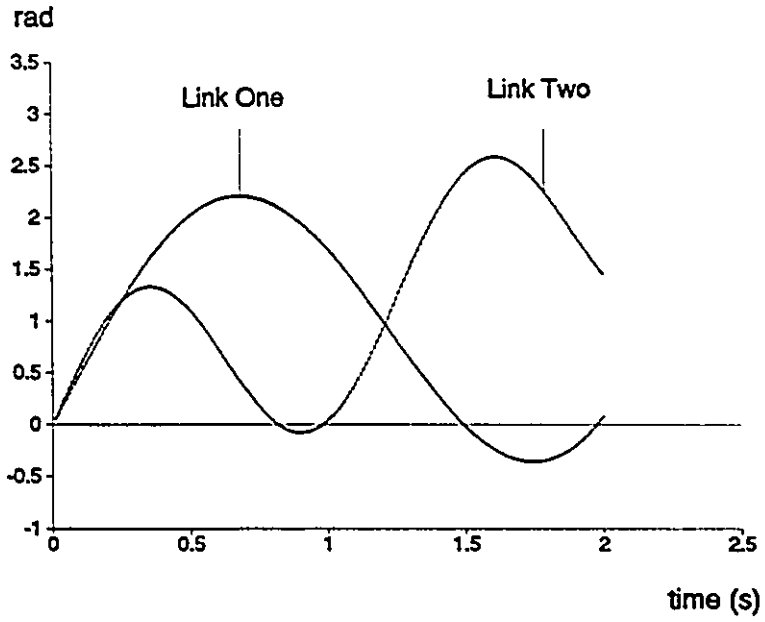

Figure 8.11 trajectory of position (trial 9)

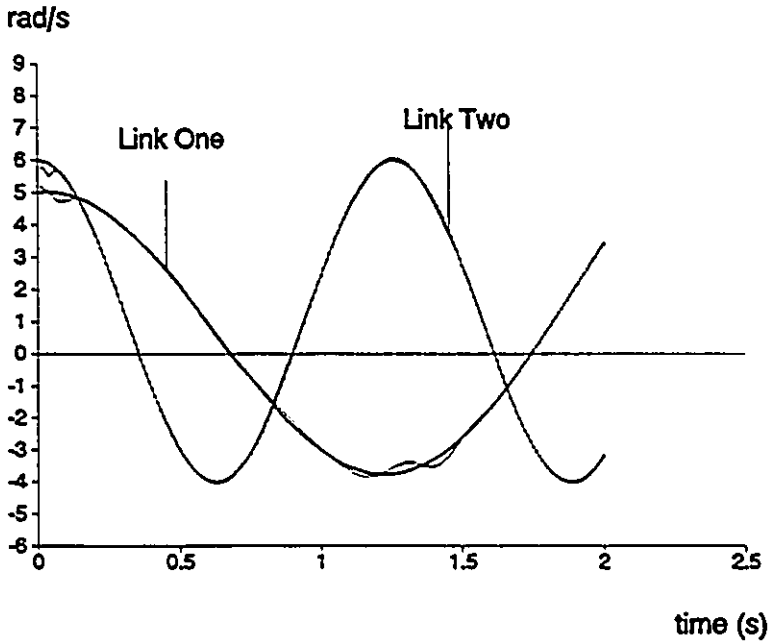

Figure 8.12 trajectory of velocity (trial 9)

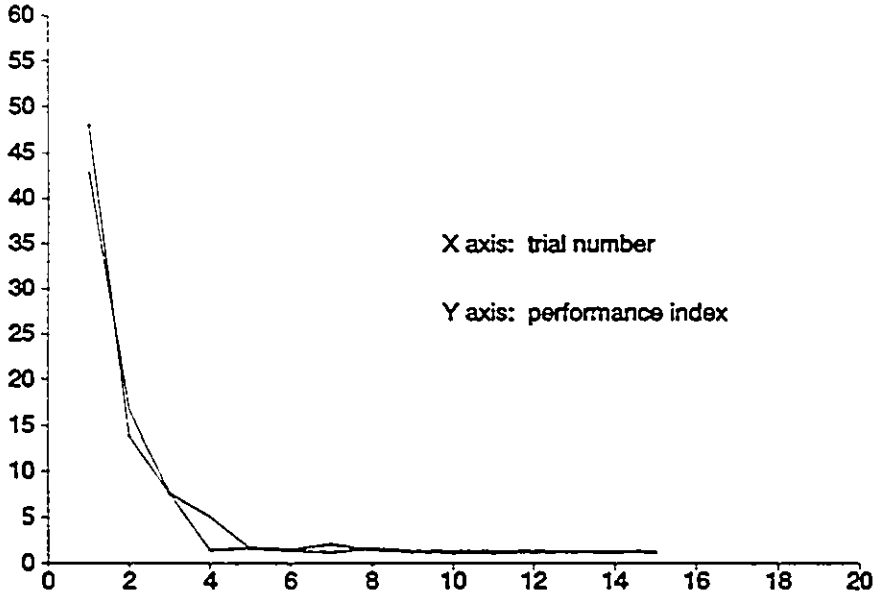

Figure 8.13 The absolute sums of position errors over trial number

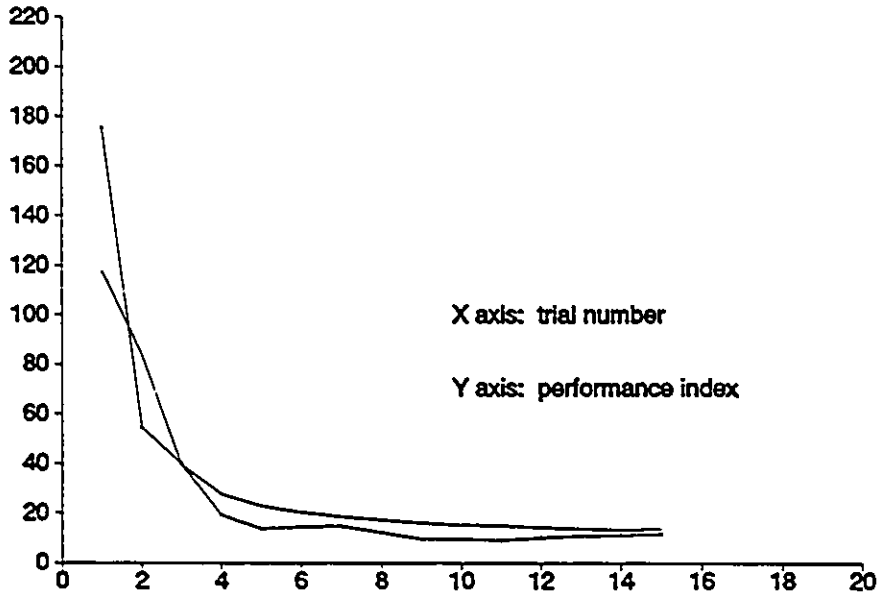

Figure 8.14 The absolute sums of velocity errors over trial number

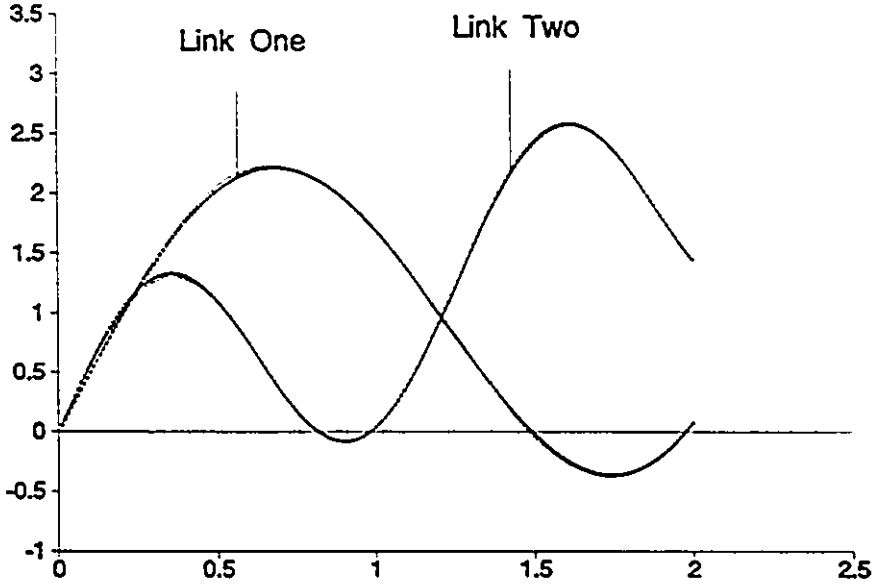

Figure 8.15 trajectory of position with 10% payload variation

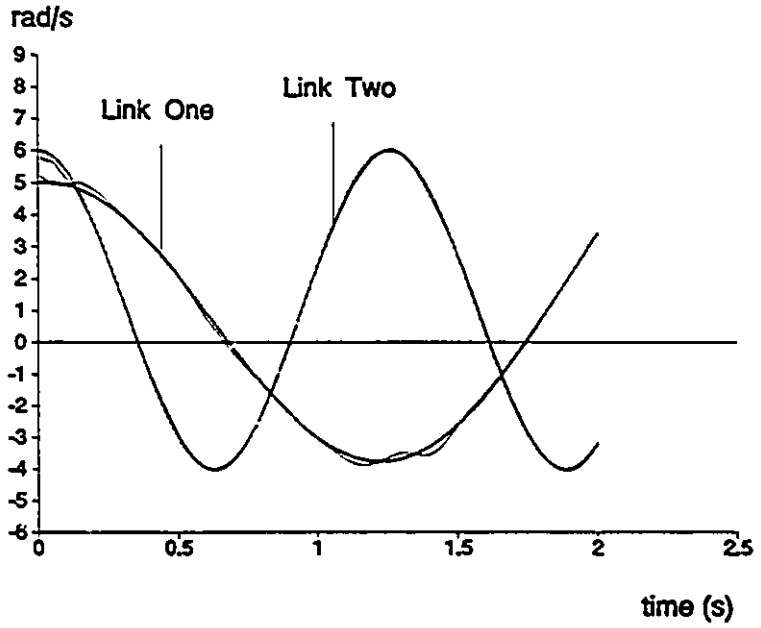

Figure 8.16 trajectory of velocity with 10% payload variation

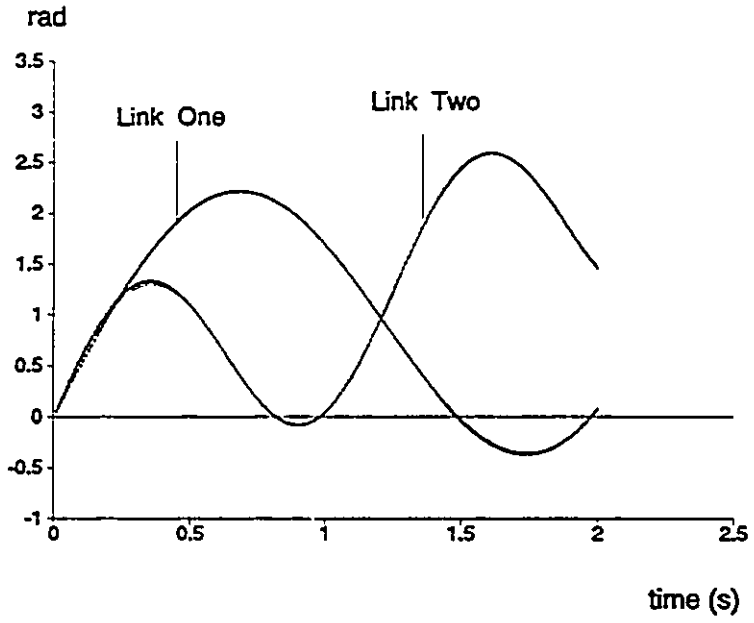

Figure 8.17 trajectory of position in the presence of noise

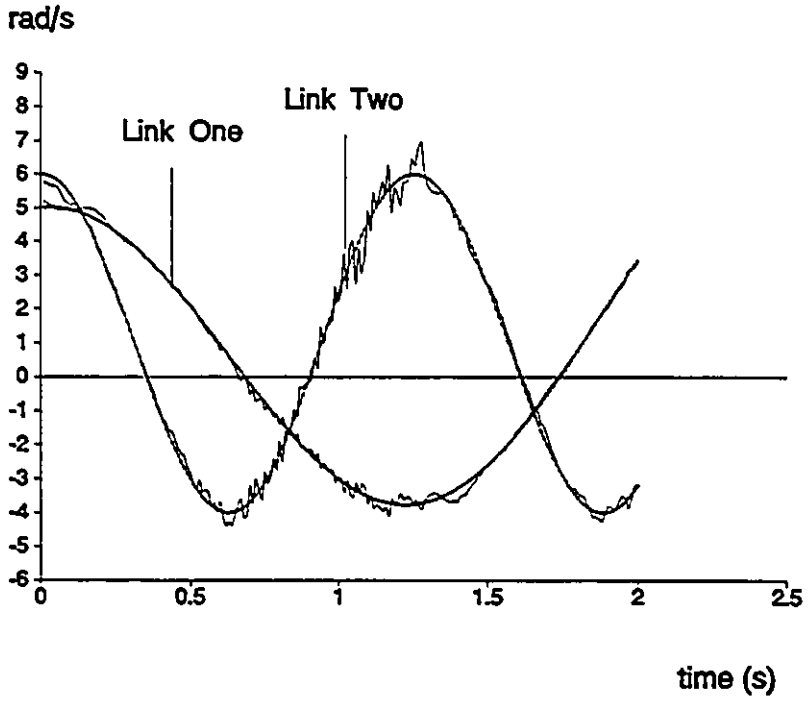

Figure 8.18 trajectory of velocity in the presence of noise

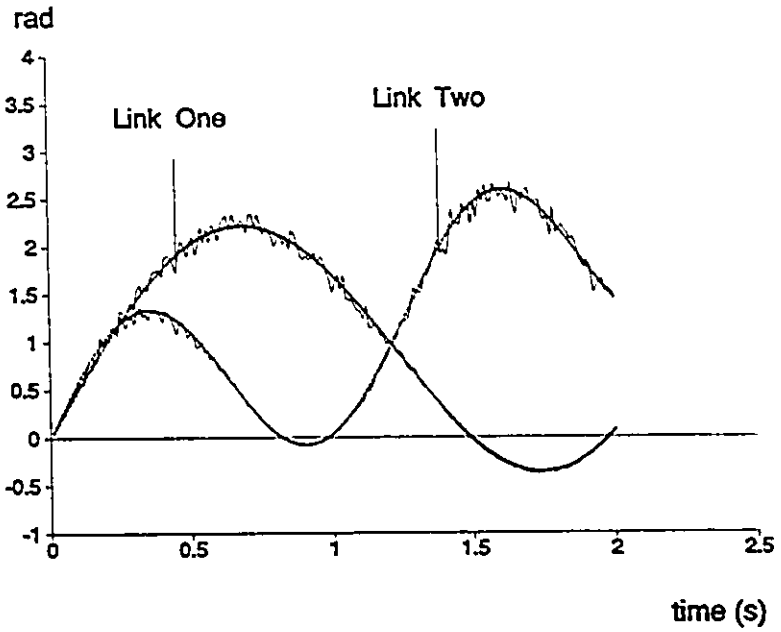

Figure 8.19 measured position

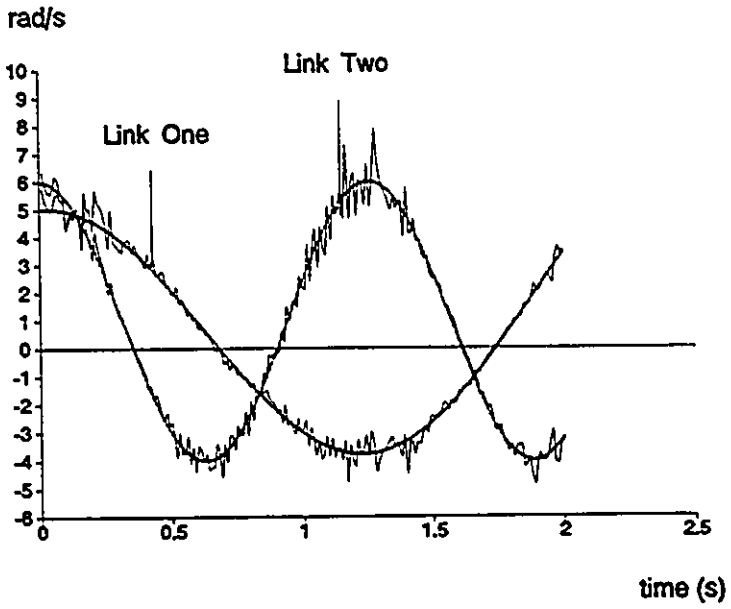

Figure 8.20 measured velocity

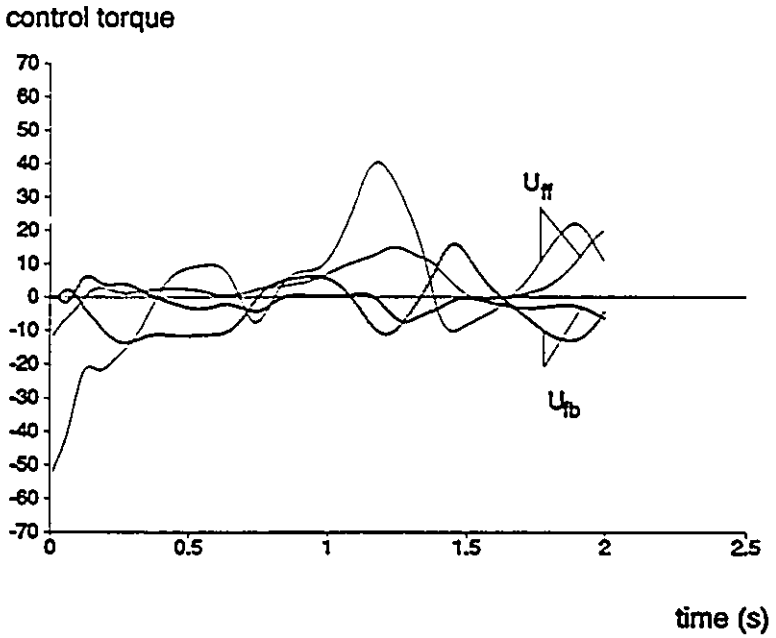

Figure 8.21 the first trial

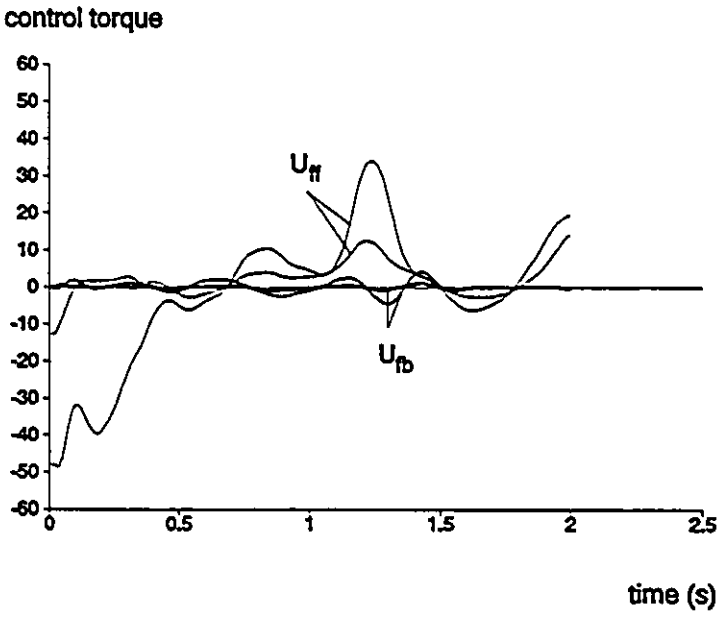

Figure 8.22 the ninth trial

8.5 Concluding Remarks

In this chapter, a repetitive neural learning controller for a robot is presented by using the modified backpropagation algorithm. A case study on a two-link robot has demonstrated that the proposed neural learning controller is very promising, having achieved satisfactory performance after a few trials. Generally, the developed iterative neural learning controller has a faster rate of convergence and better robustness than the common iterative learning controller. The learning process occurs between two consecutive operations of the robot and the neural learning control is implemented as a feedforward controller. The required on-line computation is only for the PD feedback controller. Thus, this avoids the problem of heavy on-line computation required by adaptive controllers and robust controllers. Furthermore, the proposed learning controller has the potential to handle the effects of flexibility, backlash and friction on the robot. It is very difficult for other controllers to deal with these problems. However, there exists the unsolved problem that there is no strictly theoretical verification for the convergence of the proposed iterative learning controller. The neural learning controller needs retraining when the robotic trajectory has been changed. This is one drawback of it compared with adaptive controllers and robust controllers. The future work should address the issues above mentioned and apply the proposed iterative learning controller to the practical applications of robotic manipulators in various aspects.

CHAPTER NINE FORCE CONTROL OF ROBOTIC MANIPULATORS

9.1 Introduction

In this chapter, an iterative learning controller using neural networks has been studied for force control of robotic manipulators. The proposed learning controller is composed of two parts: a feedforward neural learning controller, which is trained as the inverse dynamic model of robotic arms and a PD type compensator in feedback control loop, which is used to stabilize the system and reject disturbances. A modification of the original back-propagation algorithm [34] is employed in the neural network, resulting in a much faster learning rate. Simulations of a two-link robot have demonstrated that the proposed control scheme for robotic manipulators can greatly reduce force errors after a few learning steps. The results of simulation have also shown that the proposed iterative learning controller has a faster rate of convergence and better robustness.

9.2 A Study of Controller Configuration

Most existing neural learning controllers are implemented on-line in a feedback loop. Since neural controllers are multilayered networks consisting of large number of neurons, they need much on-line computation, making real-time implementation difficult. Here, we use the idea of the iterative learning controller

described in reference [24] [25] and the learning capability of neural networks to design an iterative learning controller. This is motivated by the fact that industrial robots usually perform repetitive tasks.

First, we briefly introduce a feedforward control scheme for linear systems to track time varying trajectories and reject time varying disturbances. Later, we shall extend this idea to the control of non-linear robotic manipulators.

Suppose that *r(t)* is an arbitrary reference trajectory and consider the block diagram of figure 9.1, where *G(s)* represents the forward transfer function of a given system and H(s) is the compensator transfer function. A feedforward control scheme consisting of adding a feedforward path of transfer function *F(s)*, as shown in figure 9.1.

Let each of the transfer functions be represented as ratios of polynomials

$$G(s)=\frac{q(s)}{p(s)} \qquad H(s)=\frac{c(s)}{d(s)} \qquad F(s)=\frac{a(s)}{b(s)} \tag{9.1}$$

We assume that the *G(s)* is strictly proper and H(s) is proper. If there is no disturbance, i.e. $D(s)=0$, the closed loop system transfer function $T(s)=Y(s)/R(s)$ is given by

$$T(s)=\frac{q(s)[c(s)b(s)+a(s)d(s)]}{b(s)[p(s)d(s)+q(s)c(s)]} \tag{9.2}$$

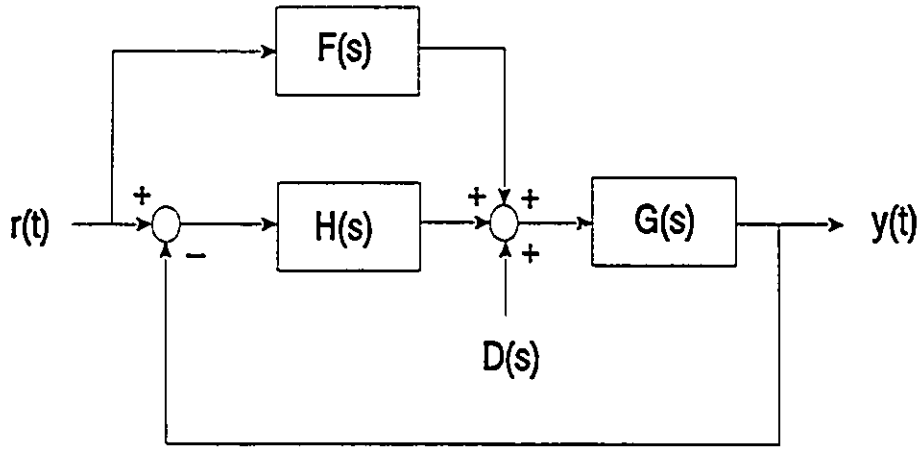

Figure 9.1

The characteristic polynomial of the closed loop system is then $b(s)[p(s)d(s)+q(s)c(s)]$. For stability of the closed loop system, therefore, we require that the compensator $H(s)$ and the feedforward transfer function $F(S)$ be chosen so that the polynomials $p(s)d(s)+q(s)c(s)$ and $b(s)$ are Hurwitz. This implies that, in addition to the stability of the closed loop system, the feedforward transfer function $F(s)$ must also be stable.

If we choose the feedforward transfer function $F(s)$ equal to $1/G(s)$, the inverse of the forward plant, then the transfer function of the closed loop system becomes

$$b(s)[p(s)d(s)+q(s)c(s)]Y(s)=q(s)[c(s)b(s)+a(s)d(s)]R(s) \tag{9.3}$$

or in terms of the tracking error $E(s)=R(s)-Y(s)$

$$q(s)[p(s)d(s)+q(s)c(s)]E(s)=0 \tag{9.4}$$

Thus, assuming stability, the output $y(t)$ will track any reference trajectory r(t). Note that we can choose $F(s)$ in this manner provided that the numerator $q(s)$ of the forward plant is Hurwitz, that is, as long as all zeroes of the forward plant are in the left half plane. Such systems are called minimum phase.

If there is a disturbance $D(s)$ entering the system as shown in figure 9.1, then it is easily shown that the tracking errors $E(s)$ is given by

$$E(s)=\frac{q(s)d(s)}{p(s)d(s)+q(s)c(s)}D(s) \tag{9.5}$$

We have shown that, if the forward transfer function is selected as the inverse of the plant, in the absence of disturbance the closed loop system will track any desired trajectory r(t) provided that the closed loop system is stable. The steady state error is thus only due to the disturbance.

Motivated by this simple idea, we propose an iterative neural learning scheme for force control of robots. Since the system is very complicated and its precise model is unavailable, we shall develop a model of the robot by using neural networks, which learns the inverse dynamics of the robot. The learning controller will assume the function of the feedforward controller described in figure 9.2.

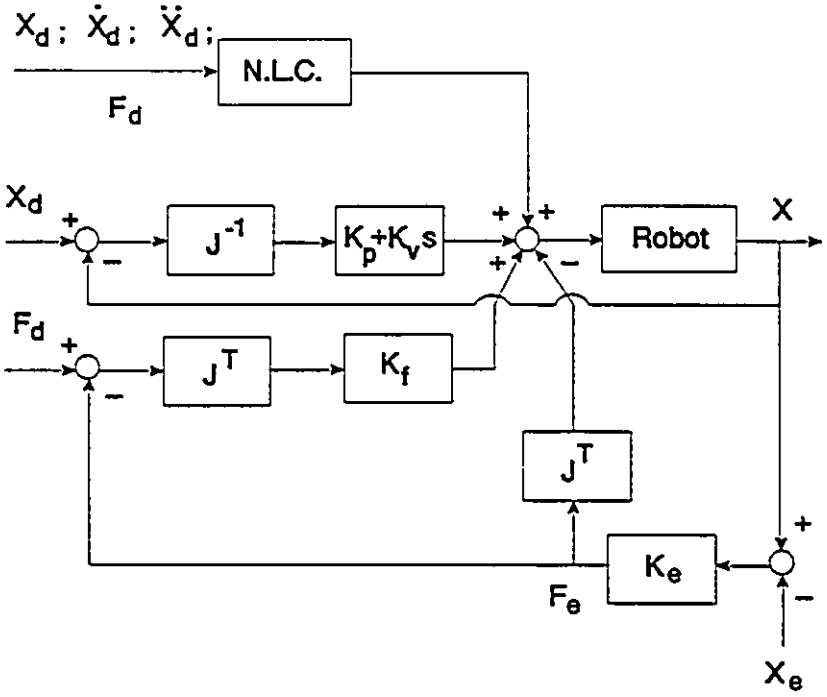

Figure 9.2

where N.L.C. represents the neural learning controller and

J: Jacobin matrix of robot;

K_e: stiffness of environment;

X_e: position of environment surface

K_p: position feedback gains;

K_v: velocity feedback gains;

K_f: force feedback gains.

The controller consists of two major parts: a feedback P.D. position controller, a P type feedback force controller and a feedforward neural learning

controller (N.L.C.), which is an approximate inverse dynamic model of the robot in the neighbourhood of the desired trajectory. The differences between the true model of the robot and the approximate neural network model may be regarded as disturbances, as shown in figure 9.1. At the initial stage, the feedback controller plays a role in making the whole system stable and contributes a relatively larger portion of control inputs to the robot. However, as the learning process continues, the dominant control input to the robot will be shifted to the feed-forward neural controller, because the neural network model of the robot gets closer to the true model of the robot. We should point out that the weights of the neural controller are fixed during the control period. After each operation of the robot, the neural network will be retrained with the operational data obtained from last trial.

Basically, the neural network will be trained as an inverse dynamic model of the robot. In the robotic control community, a type of general learning of neural network is favoured, in which a neural network is aimed to be trained as an inverse dynamic model of the robot in the entire state space. However, there is no successful report in the literature up to now. There are two major problems facing the general learning of neural network for robot. One is how to select the training input-out pairs in order to excite all modes of robotic system and another is concerned with the stability of the training algorithm for the neural network. Due to these unsolved problems, there are some feelings in the robotic control community that the neural controller is not as helpful as expected. However, it is believed that

the neural model of a robotic manipulator is much superior to linearized models. In this paper, we manage to solve the problem of stability of the back-propagation neural network by proposing the modified algorithm. Although a strictly theoretical verification is not given, many simulations studied by the author have shown that the proposed modified back-propagation algorithm is easily stabilized by correctly choosing the learning rate according to eq.(7.20), resulting in a very fast learning speed. A satisfactory precision can be achieved after a couple of training cycles. The training scheme of the neural network can be described by figure 9.3.

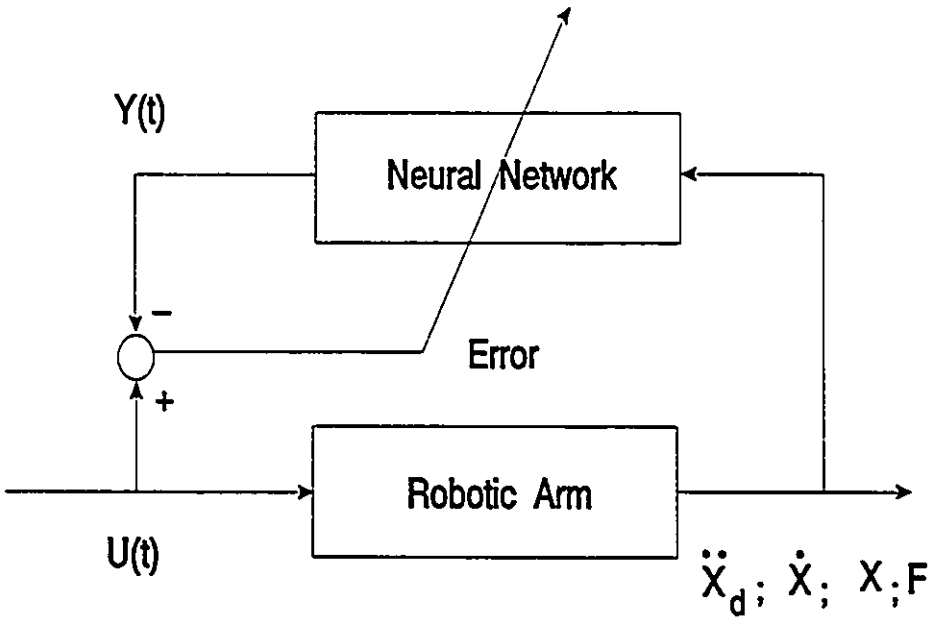

Figure 9.3

During the training process, the output of the robot, i.e. position, velocity, force and desired acceleration, will be fed into the input layer of the neural network. Notice that no measured acceleration is needed in the training of the neural network. The weights of the neural network are adjusted based on the errors between the input torque of manipulator U(t) and the output of neural network Y(t) by using the modified back-propagation algorithm of the neural network. We point out here that the actual trajectory from the previous operation is fed into the input layer of the neural network during training process while the desired trajectory is presented in the control process.

When we apply neural networks to the control of robots, two important things should be taken into consideration. One is that most industrial robots execute tasks along a fixed trajectory. Secondly, a trajectory close to the desired is easily obtained by using a computed torque controller or a PD controller. Thus, our approach is to train the neural network to determine an inverse dynamic model of the robot in the neighbourhood of the desired trajectory based on the experience of the realized trajectory. After each operation of the robot, the learning process is repeated and as the number of repeated operations increases, the trajectory tracking errors can be reduced to almost zero.

9.3 A Teaching Controller

As we know, a teacher is needed to start the training of back-

propagation neural network. Here, we choose an inverse dynamic controller of robot as the teacher of the training process. First, we give a brief review of inverse dynamic control. The idea of inverse dynamic control is to seek a nonlinear feedback control law, which results in a linear closed loop system.

Consider a robotic manipulator expressed by the following nonlinear dynamic model

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u(t) - J^T F_e \tag{9.6}$$

where $M(q)$ is the $n \times n$ symmetric and positive definite matrix (also called the generalized inertia matrix), $C(q, \dot{q})$ is the $n \times 1$ vector due to Coriolis and centripetal forces, $g(q)$ is the n×1 vector due to gravitational forces. $u(t)$ is the $n \times 1$ vector of joint torques supplied by actuator. $q(t)$ is the vector of joint positions. J^T is the transpose of the Jacobian Matrix and Fe is the force exerted on the end effector of the robot by contacted environment.

For simplicity, we rewrite the above equation as

$$M(q)\ddot{q} + h(q,\dot{q}) = u(t) - J^T F_e \tag{9.7}$$

Also, we can rewrite these equations directly in terms of the end-effector coordinates as follows. The relationship between the end-effector coordinates and the vector of joint coordinates q is given as

$$X = f(q) \tag{9.8}$$

where the function f represents the forward kinematic equation. In general we can only determine such a function f uniquely in a region free of kinematic singularities. Taking the first and second derivatives of eq.(9.8) yields

$$\dot{X} = J(q)\dot{q} \tag{9.9}$$

$$\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q} \tag{9.10}$$

where $J(q)$ is the Jacobin matrix of the robot. From eq.(9.7), we have

$$\ddot{q} = M^{-1}(q)[u(t) - h(q,\dot{q}) - J^T F_e] \tag{9.11}$$

Substituting the above equation into (9.10) and suppressing arguments for brevity, we obtain

$$\ddot{X} = JM^{-1}(u - h - J^T F_e) + \dot{J}\dot{q} \tag{9.12}$$

Eq.(9.12) can be rewritten concisely as

$$D(X)\ddot{X} + H(X,\dot{X}) = F - F_e \tag{9.13}$$

where $F = (J^T)^{-1}u$ is the input expressed in task space and

$$D(X) := (J^T)^{-1}MJ^{-1}; \qquad H(X,\dot{X}) = (J^T)^{-1}h - D(X)(J^T)^{-1}\dot{J}J^{-1}\dot{X} \tag{9.14}$$

Now, to obtain the double integrator system in task space

$$\ddot{X} = f \tag{9.15}$$

it is easy to see that the control F should be chosen according to

$$F = F_c + D(X)f + H(X,\dot{X})$$

Since the parameters of robot dynamics are partially unknown, the above inverse dynamic control law in task space cannot result in a linearized system in an actual case. However, this scheme is still a good teacher for the neural controller. It is expected that the approximate inverse dynamic control can bring the robot to the neighbourhood of the desired trajectory. This is sufficient to start the learning of the neural controller. The iterative learning controller with neural network can improve the tracking precision by itself as the number of learning times increases.

9.4 A Simulation Study

The proposed repetitive learning controller with neural network is applied to the simulation of a two-link planar robot. The simulated robotic manipulator is depicted by the following figure.

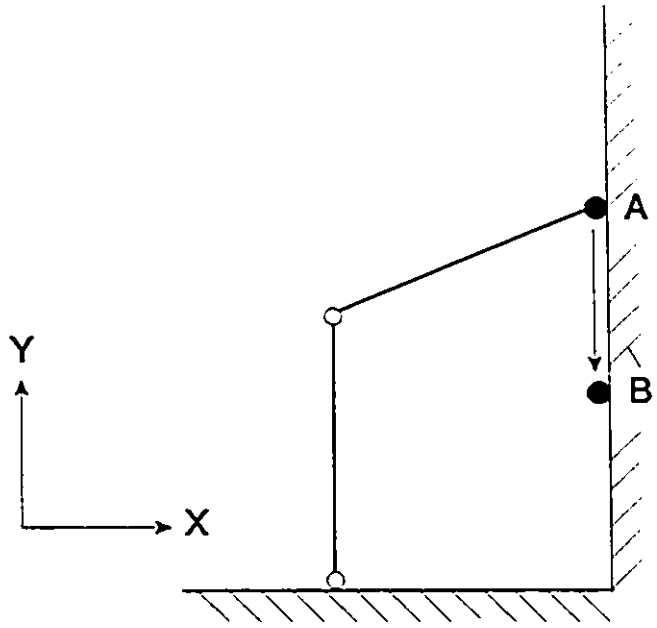

Figure 9.4 configuration of a two-link robot

Assuming that the length of the links of the robot is equivalent, then its dynamic equation is given by

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3}m_1l^2 + \frac{4}{3}m_2l^2 + m_2c_2l^2 & \frac{1}{3}m_2l^2 + \frac{1}{2}m_2c_2l^2 \\ \frac{1}{3}m_2l^2 + \frac{1}{2}m_2c_2l^2 & \frac{1}{3}m_2l^2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}m_2s_2l^2\dot{q}_2^2 - m_2s_2l^2\dot{q}_1\dot{q}_2 \\ \frac{1}{2}m_2s_2l^2\dot{q}_1^2 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}m_1glc_1 + \frac{1}{2}m_2glc_{12} + m_2glc_1 \\ \frac{1}{2}m_2glc_{12} \end{bmatrix} + J^T F_e \tag{9.17}$$

where

$$\begin{array}{lll} c_1 = \cos(q_1) & c_2 = \cos(q_2) & c_{12} = \cos(q_1 + q_2) \\ s_1 = \sin(q_1) & s_2 = \sin(q_2) & \end{array} \tag{9.18}$$

The Jacobian matrix of the robot is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} lc_{12} + lc_1 \\ ls_{12} + ls_1 \end{bmatrix}$$

The environment is modelled as

$$F_e = \begin{cases} K_e(x - X_e) & if \quad x > X_e \\ 0 & \text{otherwise} \end{cases} \tag{9.20}$$

where K_e is the stiffness of the environment and X_e is the position of environment surface.

In our simulation, we require that the robot move from point A to B along the y direction and exert 10 Newtons force on environment along the x direction at the same time. The neural network employed in the simulation consists of an input layer with seven neuron nodes , the first hidden layer with 25 neuron nodes, the second hidden layer with 35 neuron nodes and an output layer with 2 neuron nodes, which is symbolized as $N_{7,25,35,2}$.

Since the exact model of the robot is not available, in our simulation , we assume that the parameters of the robotic model are different from the true values as shown in table 9.1

Table 9.1 **Input Data For Simulation**

Parameters	True Values	Modeled Values
Mass of Link 1	2kg	1.7kg
Mass of Link 2	2.5kg	2.0kg
Length of Link	0.6m	0.6m

As discussed before, we employ an inverse dynamic control scheme in task space based on an approximate model to bring the robot into the neighbourhood of the desired trajectory. The resulting force trajectory is shown in figure 9.5. We observe that the performance is not good because there is a significant difference between the modelled parameters and the true parameters of the robot. However, this is sufficient to start training for learning process, since the neural learning controllers are able to improve the control performance by themselves and to achieve satisfactory performance after a few learning steps. Figure 9.6 shows the control torque history of the teacher. Using the operational data obtained from the results of the teacher, we train the neural network as an approximate model of the robot as shown in figure 9.3. Figures 9.7 and 9.8 are the training results of the neural network after one lesson. Figures 9.9 and 9.10 are the training results of the neural network after 5 lessons. These figures demonstrate that the modified back-propagation algorithm has a very fast rate of convergence and many simulations conducted by the author have proven the training is always convergent. After finishing the training of

the neural network, we implement the neural learning controller as described in figure 9.2. During the control stage, all weights of the neural network are fixed. The feedforward control signals are calculated beforehand, where the inputs are the desired position, velocity, acceleration and force. Only the PD position controller and P type force controller require on-line computation. Figure 9.11 represents the force exerted on the contacted environment by the end effector of the robot after first trial. Figure 9.12 is the result of the second trial and figure 9.13 is the result of the 7th trial. The simulation results demonstrate that the proposed neural learning scheme is an effective approach for force control of a robot, which can achieve satisfactory results in a few leaning steps.

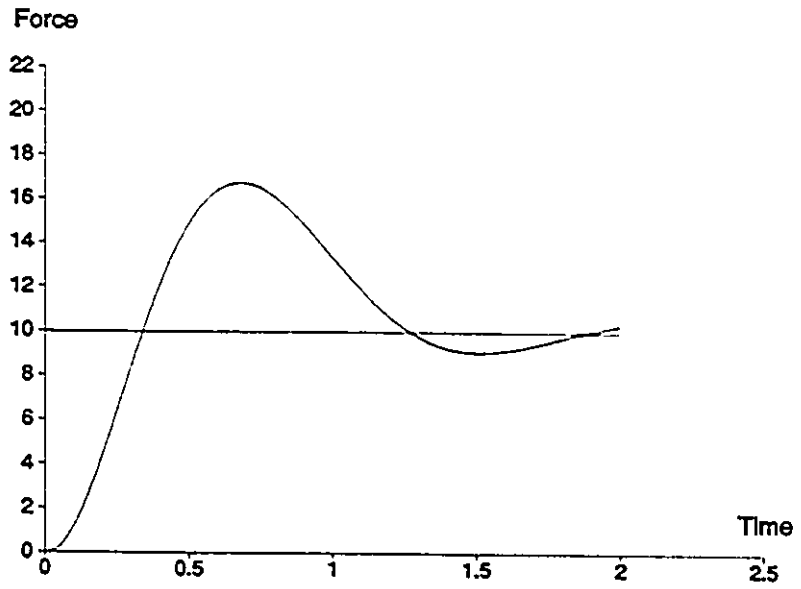

Figure 9.5 Contact Force by Teacher

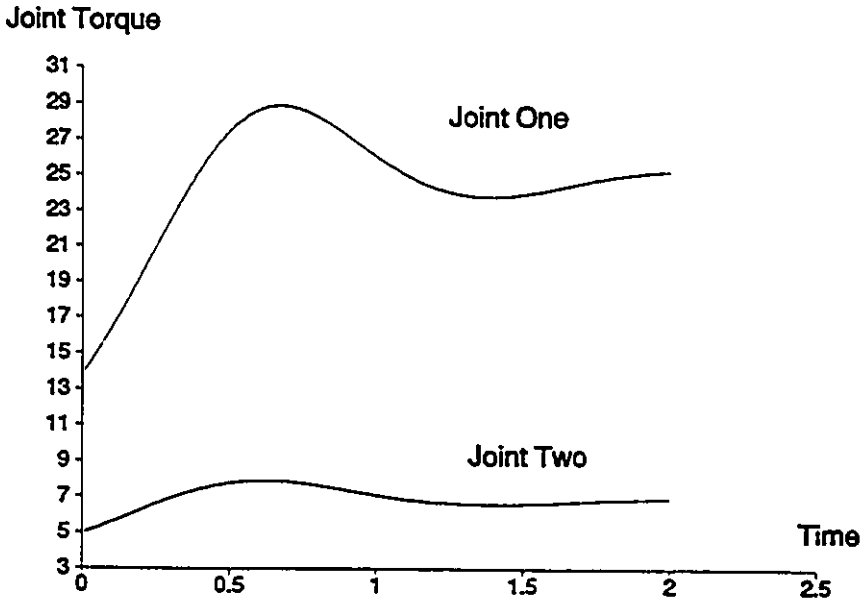

Figure 9.6 Control Torque History of Teacher

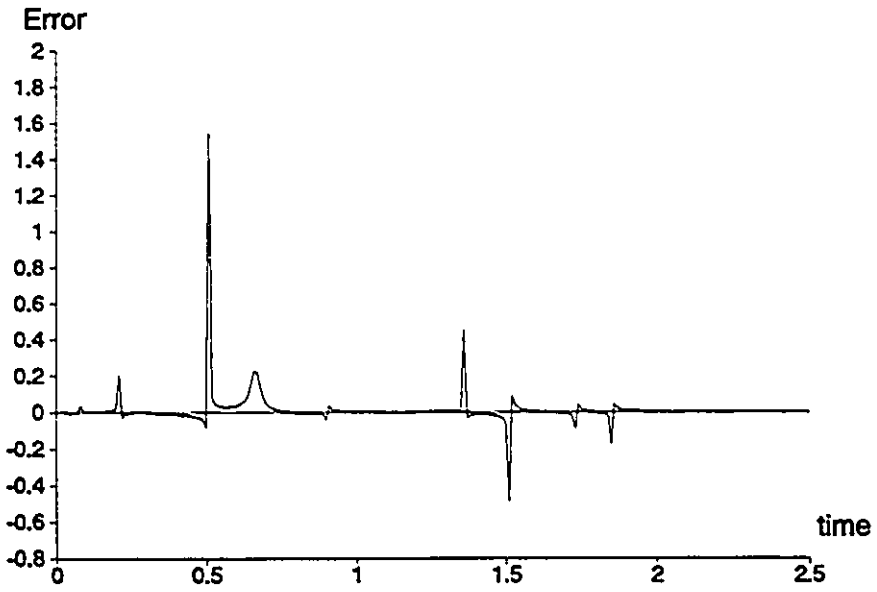

Figure 9.7 Training Results of Joint One (after one lesson)

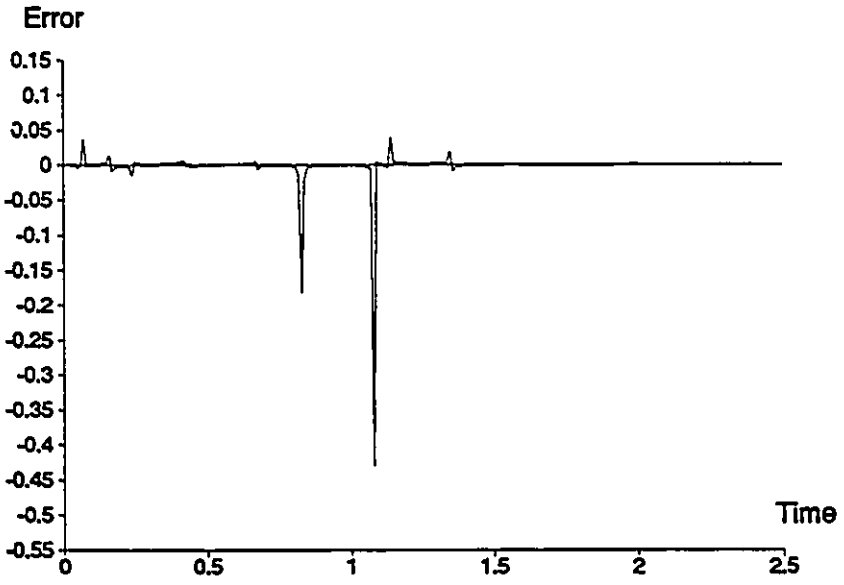

Figure 9.8 Training Results of Joint Two (after one lesson)

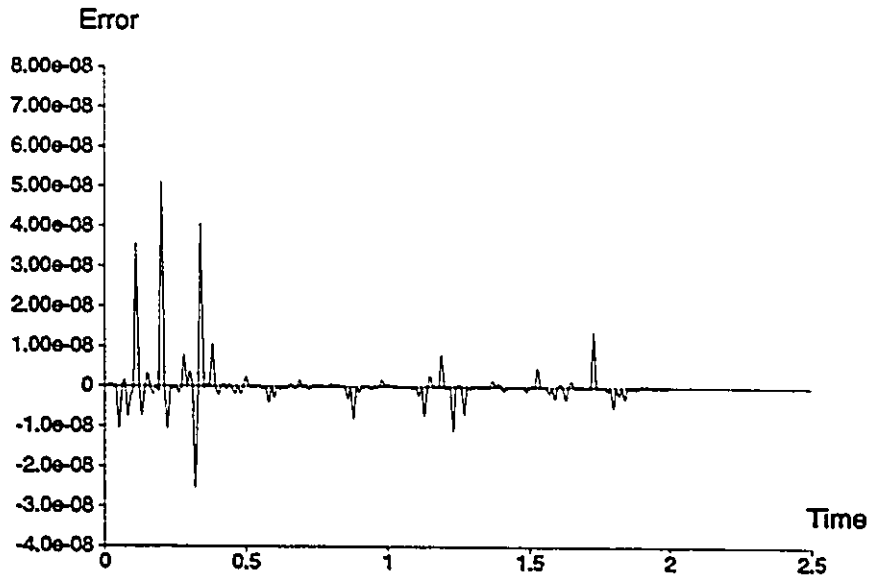

Figure 9.9 Training Results of Joint One (after 5 lessons)

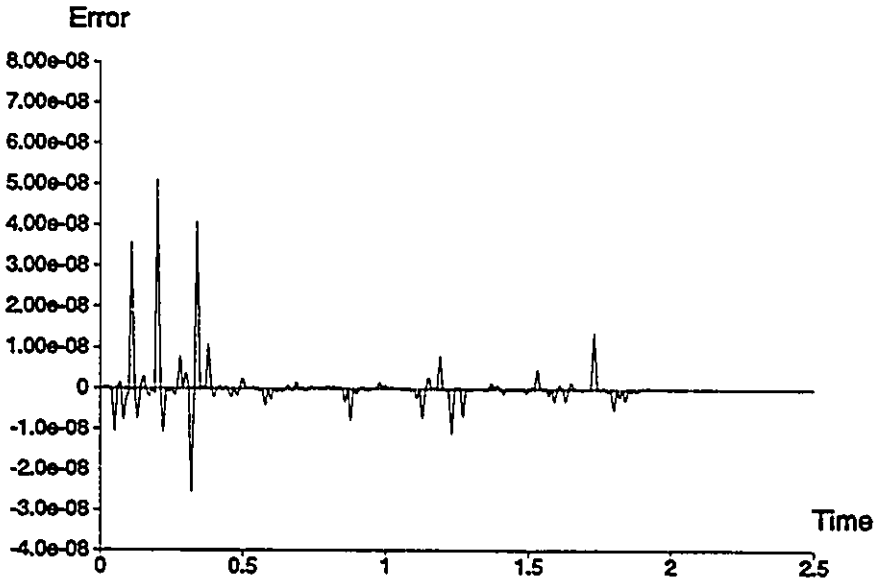

Figure 9.10 Training Result of Joint Two (after 5 lessons)

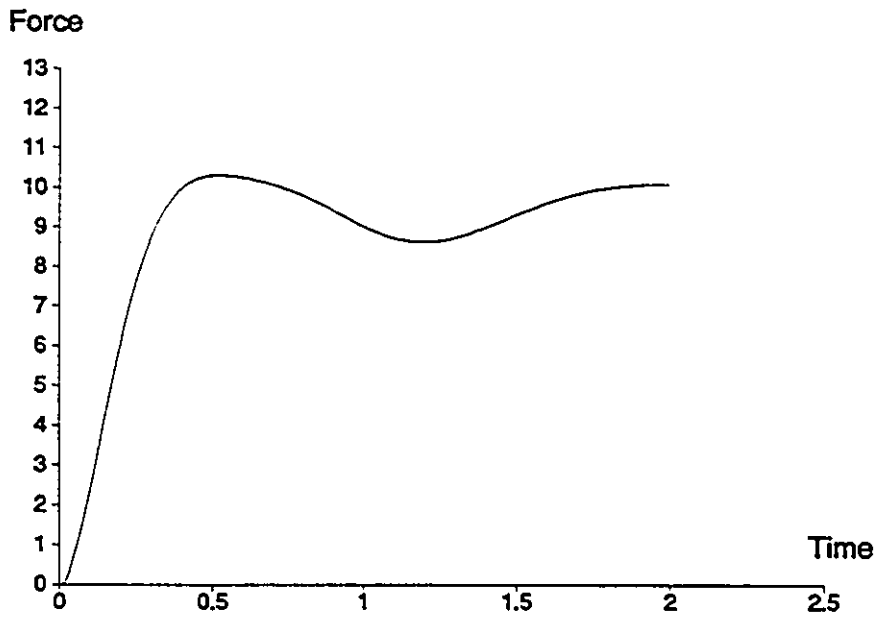

Figure 9.11 Contact Force (trial one)

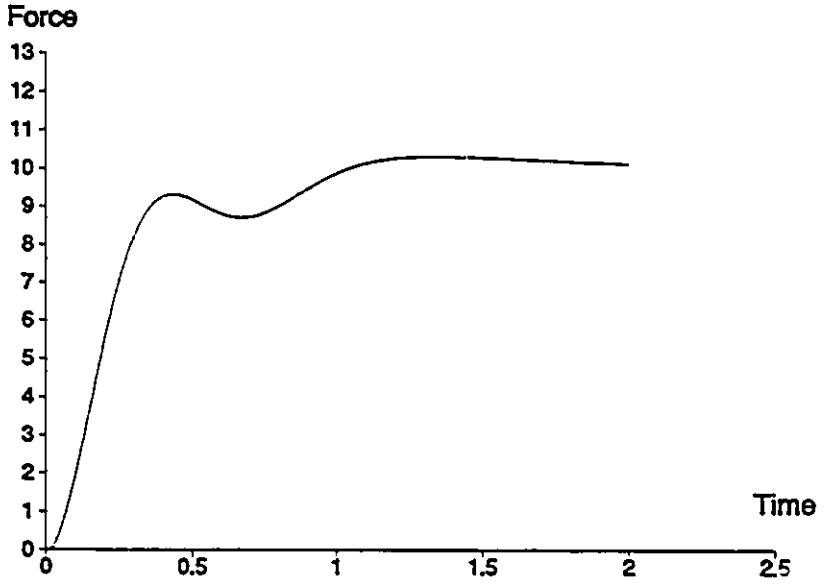

Figure 9.12 Contact Force (trial two)

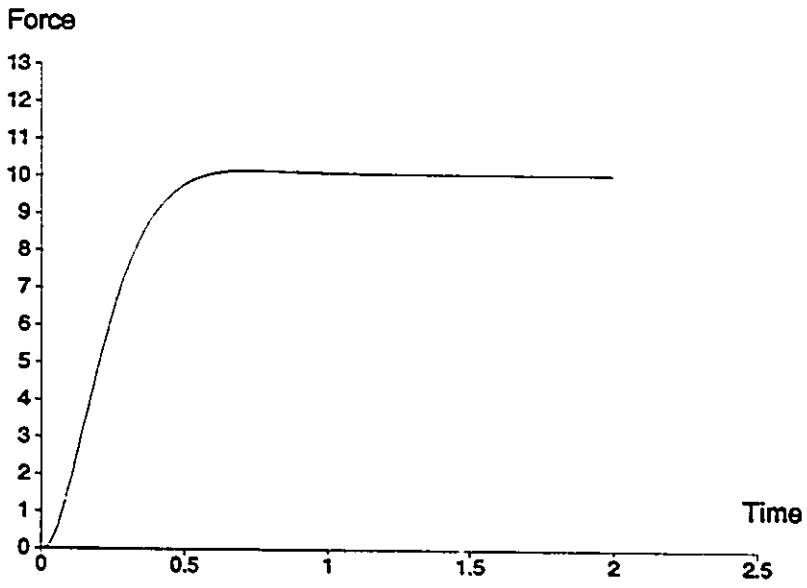

Figure 9.13 Contact Force (trial seven)

9.5 CONCLUSIONS

In this chapter, we have presented an iterative neural learning scheme for force control by using the author's modification of the backpropagation algorithm. A case study of a two-link robot has demonstrated that the proposed neural learning controller is very promising and achieved satisfactory performance after a few trials. Generally, the proposed iterative neural learning controller has a faster rate of convergence and better robustness than the common iterative learning controller. The learning process occurs between two consecutive operations of the robot and the neural learning control is implemented as a feedforward controller. The required on-line computation is only for the PD feedback controller. This avoids the problem of

heavy on-line computation required by adaptive controllers and robust controllers. Furthermore, the proposed learning controller has the potential to handle the effects of flexibility, backlash and friction in the robot. It is difficult for other controllers to deal with these problems. In practice, we can develop an approximate model of the robot. Then we simulate this robot on a computer by using the inverse dynamic control scheme. The neural network is trained by using the simulation data. After this, the learning scheme described in figure 9.2 will put into the actual robot. It is expected that more precise the model of the robot, the faster will be the learning process. However, there exists the unsolved problem that there is no strictly theoretical verification of the convergence of the proposed iterative learning controller. The neural learning controller needs retraining when the robotic trajectory has been changed. This is one drawback of the proposed approach compared with adaptive controllers and robust controllers. Also, when the stiffness of the robot is very high, the neural learning controller failed in our study. Thus, there is a trade-off in selecting adaptive controller or iterative neural learning controller. An adaptive controller has better control precision, but the controller needs more on-line computation, which is more complicated and expensive.

CHAPTER TEN MOTION CONTROL OF FLEXIBLE-JOINT ROBOTS

10.1 Introduction

Due to the increased complexity of the dynamics of robots with joint flexibility, many conventional robot control strategies are incapable to solve this problem effectively. To overcome this difficulty, a neural controller based on iterative learning has been proposed for the control of a flexible joint robot in this chapter. The controller design is based on the fact that most industrial robots perform repetitive tasks. The neural network for this purpose is trained as an inverse dynamic model of a flexible joint robot and is implemented as a feedforward controller. A modification of the back-propagation neural network [34] is employed in this study and result in a much better learning efficiency. A case study of a two-link flexible joint robot has demonstrated that the proposed iterative neural learning controller can reduce the tracking errors after a few learning steps. Also, the proposed neural learning has advantages of easy implementation and requiring very little on-line computation.

In this chapter, we consider motion control of robotic manipulators with flexible joints. Compared with the large volume of literature available on the control of rigid robots, relatively little has been published on the control of flexible-joint robots. On the other hand, experiments indicate that joint flexibility should

be considered in both modelling and control of robots if high performance is to be achieved. Due to the increased complexity of the dynamics of flexible joint robots, nice features, associated with rigid robots, such as independent control input for each degree of freedom and the passivity, have been lost. To overcome these problems, various control strategies have been proposed. One approach is based on feedback linearization. The drawbacks of this approach is the required measurement of link acceleration and jerk. Another approach is based on the concept of integral manifold. In this case, the flexible-joint robot dynamics are restricted to a suitable integral manifold in state space and a reduced-order model can be derived. This reduced model can be linearized by non-linear feedback using only position and velocity information. The main drawback of this approach is its lack of robustness to parametric uncertainties.

The recent resurgence of neural networks has inspired the interest of the control community because of its high potential to solve control problems of complicated nonlinear systems, such as robotic manipulators. Some researchers have made considerable progress in applying neural networks to the control of robotic manipulators. In particular, reference [37] has applied neural networks to the control of flexible-joint robots. However, like other neural learning controllers, they are configured on-line in a feedback loop. This is difficult to implement in real time since the neural controller is composed of a huge number of neurons.

To overcome this problem, we proposed a neural controller based on

iterative learning for control of flexible-joint robots in this chapter. The proposed learning scheme requires little on-line computation and achieves satisfactory control precision after a few learning steps.

This chapter is organized as follows: In section 2, a repetitive learning scheme using neural network is studied. A teacher controller is investigated in section 3. Simulation results will be presented in section 4. Finally, some concluding remarks are given.

10.2 A Control Technique

Consider a robotic manipulator with flexible joints expressed by the following nonlinear dynamic model [3]

$$\begin{aligned} M(q_1)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) + K(q_1 - q_2) &= 0 \\ J\ddot{q}_2 - K(q_1 - q_2) &= u(t) \end{aligned} \tag{10.1}$$

where $M(q_1)$ is the $n \times n$ symmetric and positive definite matrix (also called the generalized inertia matrix), $C(q_1, \dot{q}_1)$ is the $n \times 1$ vector due to Coriolis and centripetal forces, $g(q_1)$ is the n×1 vector due to gravitational forces. $u(t)$ is the $n \times 1$ vector of joint torques supplied by actuator. The vectors $q_1(t)$ and $q_2(t)$ represent the link angels and motor angles. K is a diagonal matrix consisting of stiffness constants.

Experimental investigations of industrial manipulators indicate that joint flexibility contributes significantly to the overall dynamics of the systems. The

increased complexity of dynamics of flexible joint manipulator makes it much more difficult to control. Here, we use the idea of the iterative learning controller described in reference [24] [25] and the learning capability of neural networks to design an iterative learning controller, which is described by figure 10.1

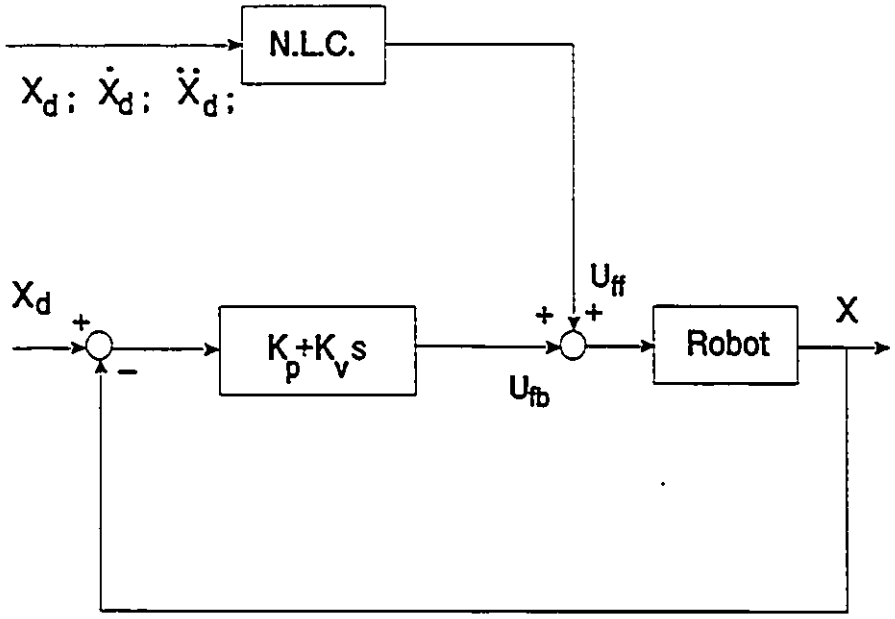

Figure 10.1

where N.L.C. the neural learning controller;

K_p: position feedback gains;

K_v: velocity feedback gains.

The controller consists of two major parts: a feedback P.D. controller and a feedforward neural learning controller (N.L.C.), which is an approximate inverse dynamic model of the robot in the neighbourhood of the desired trajectory.

The differences between true model of the robot and the approximate neural network model are regarded as disturbances. At the initial control stage, the feedback controller plays a role in making the whole system stable and contributes a relatively larger portion of control inputs to the robot. However, as the learning process continues, the dominant control input to robot will be shifted to the feed-forward neural controller, because the neural network model of the robot is getting closer tc the true model of the robot. We should point out that the weights of the neural controller are fixed during the control period. After each operation of the robot, the neural network will be retrained with the operational data obtained from last trial. Basically, the neural network will be trained as an inverse dynamic model of the robot. The training scheme of the neural network can be described by figure 10.2.

During the training process, the output of the robot, i.e. measurement of link angles, velocity angels of both link and motor as well as desired acceleration, will be fed into the input layer of the neural network. Notice that no measured acceleration is needed in the training of the neural network. The weights of the neural network are adjusted based on the errors between the input torque of manipulator U(t) and the output of neural network Y(t) by using modified back-propagation algorithm of the neural network. We point out here that the actual trajectory from the previous operation is fed into the input layer of the neural network during training process while the desired trajectory is presented in the

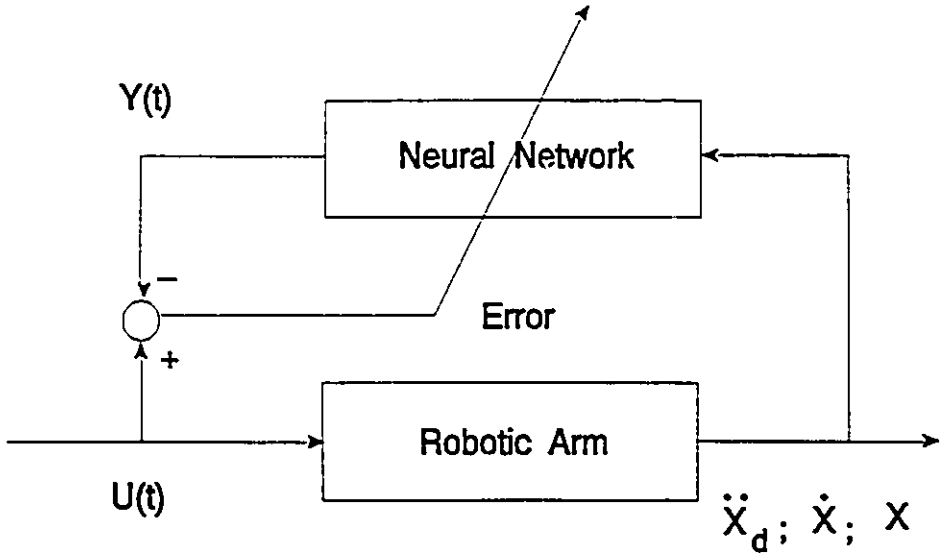

Figure 10.2

control process.

When we apply neural network to the control of robots, two important things should be taken into consideration. One is that most industrial robots execute tasks along a fixed trajectory. Secondly, a trajectory close to the desired is easily obtained by using a simple controller. Thus, our approach is to train the neural network as an inverse dynamic model of the robot in the neighbourhood of the desired trajectory based on the experience from the realized trajectory. After each operation of the robot, the learning process is repeated and as the number of repeated operations increases, the trajectory tracking errors can be reduced to almost

zero.

10.3 A Teaching Controller

As we know, a teacher is needed to start the training of a back-propagation neural network. Here, we choose a control scheme introduced in reference [4] as the teacher, which is a modified version of inverse dynamic scheme for rigid robot.

Consider a flexible-joint robot expressed by eq.(10.1). The proposed controller is described as

$$u(t) = u_r(t) + K_v(\dot{q}_1 - \dot{q}_2) \tag{10.2}$$

where $u_r(t)$ is the control input derived based on the inverse dynamic controller for rigid robot and K_v is a constant diagonal matrix. $u_r(t)$ is given by

$$u_r = (\hat{M}(q) + \hat{J})\dot{v} + \hat{C}(q,\dot{q})v + \hat{g}(q) \tag{10.3}$$

Following shows why the simple modification of a rigid controller achieves the desired result.

Substitute the control law (10.2) into (10.1) and define $e = q_2 - q_1$. The result is

$$J\ddot{e} + K_v\dot{e} + Ke = u_r - J\ddot{q}_1 \tag{10.4}$$

Next, define the variable $z = Ke$, and rewriting eq.(10.4) as

$$J\ddot{z} + K_v\dot{z} + Kz = K(u_r - J\ddot{q}_1) \tag{10.5}$$

Assuming K is large relative to the other parameters in the system. More specifically, K is on the order of $1/\varepsilon^2$, where ε is a small parameter. Thus, we may write

$$K = K_1/\varepsilon^2 \qquad K_v = K_2/\varepsilon \tag{10.6}$$

where K_1 and K_2 are on the order of 1. Thus, eq.(10.5) can be written as

$$\varepsilon^2 J\ddot{z} + \varepsilon K_2\dot{z} + K_1 z = K_1(u_r - J\ddot{q}_1) \tag{10.7}$$

Thus, we have

$$\begin{aligned} M(q_1)\ddot{q}_1 + C(q_1,\dot{q}_1)\dot{q}_1 + g(q_1) &= z \\ \varepsilon^2 J\ddot{z} + \varepsilon K_2\dot{z} + K_1 z &= K_1(u_r - J\ddot{q}_1) \end{aligned} \tag{10.8}$$

The above system is singularly perturbed. The variables of z and $\dot{z}$ have the interpretation of "fast" variable. whereas the link variable q_1 and $\dot{q}_1$ are "slow" variables. Using standard results from the singular perturbation theory described in reference [30], we may approximate the system of eq.(10.8) by using quasi-steady-state system and a boundary-layer system.

Since parameters of robot dynamics are partially unknown, the above modified inverse dynamic control law cannot result in a linearized system in practice. However, this scheme is still a good teacher for the neural controller. It is expected that even the approximate inverse dynamic control can bring the robot to the

neighbourhood of the desired trajectory. This is good enough to start the learning of the neural controller. The iterative learning controller with neural network can improve the tracking precision by itself as the number of learning steps increases. Our simulation studies confirm it.

10.4 Simulation Studies

In this section, we apply the proposed iterative learning scheme to the control of a two-link planar manipulator with joint flexibility. In our simulation, we assume that the two links of the manipulator have the same length, with mass m_1=2.0 kg and m_2=2.5 kg respectively. The stiffness constant K=1000. The joints of simulated manipulator are required to move along the following desired trajectories

$$\begin{aligned} &\text{Joint one:} \quad q^1(t) = 3t^2 - 2t^3 + 0.375t^4 \\ &\text{Joint two:} \quad q^2(t) = 4.5t^2 - 3t^3 + 0.5625t^4 \end{aligned} \tag{10.9}$$

The running time is two seconds.

The neural network employed in the simulation consists of an input layer with eight neuron nodes , the first hidden layer with 20 neuron nodes, the second hidden layer with 40 neuron nodes and an output layer with 2 neuron nodes. Thus it is symbolized as $N_{8,20,40,2}$.

Since the exact model of the robot is not available, in our simulation. We assume that the parameters of the model are different from the true values,

while m_2 equals 3.0 kg.

As discussed before, we employ a modified inverse dynamic control scheme in joint space based on an approximate model of the flexible joint robot to bring the robot into the neighbourhood of desired trajectory. The resulting trajectory are shown in figure 10.3. In the following figures, the solid lines represent desired trajectories and the dotted lines represent actual trajectories. We observe that the performance is not good because there is a significant difference between the modelled parameters of the robot and the true parameters of the robot. However, this is sufficient to start training for learning process, in which the neural learning controllers are able to improve the control performance by themselves and to achieve satisfactory performance after a few learning steps. Using the operational data obtained from the results of the teacher, we train the neural network as an approximate model of the robot as shown in figure 10.2. Figure 10.4 is the training results of neural network after three lessons. Figure 10.5 shows the curves of learning performance vs training times. These figures demonstrate that the modified back-propagation algorithm has a very fast rate of convergence and many simulations conducted by the author have shown that the training is always convergent. After finishing the training of the neural network, we implement the neural learning controller as described in figure 10.1. During the control stage, all weights of the neural network are fixed. The feedforward control signals are calculated beforehand, where the inputs are desired position, velocity, acceleration. Only the PD controller

requires the on-line computation. Figure 10.6 represents the control results after the second trial. Figure 10.7 is the result of the eighth trial. Figure 10.8 shows the convergence rate of the learning controller. We find that the satisfactory control precision can be achieved after a few learning steps. This shows that the proposed neural learning scheme is an effective approach for control of flexible joint robotic manipulators.

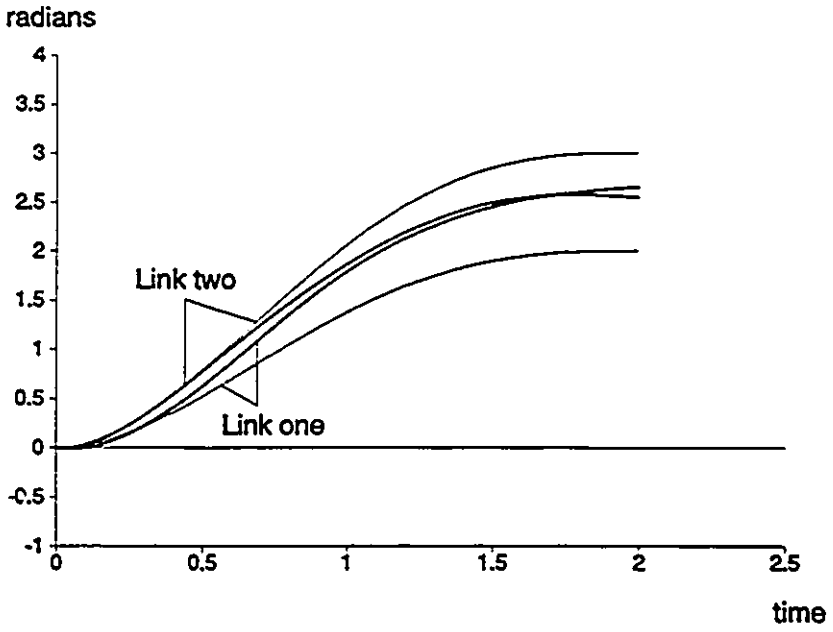

Figure 10.3 control results of the teacher

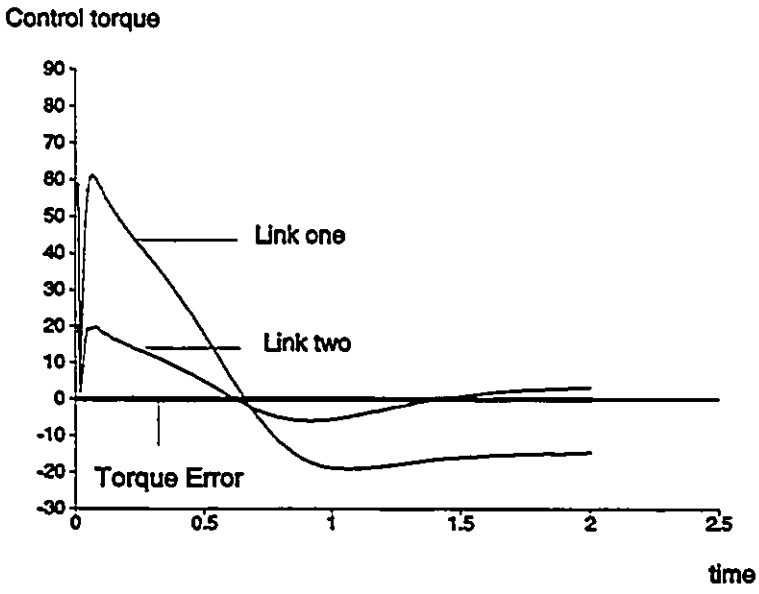

Figure 10.4 joint control torque history

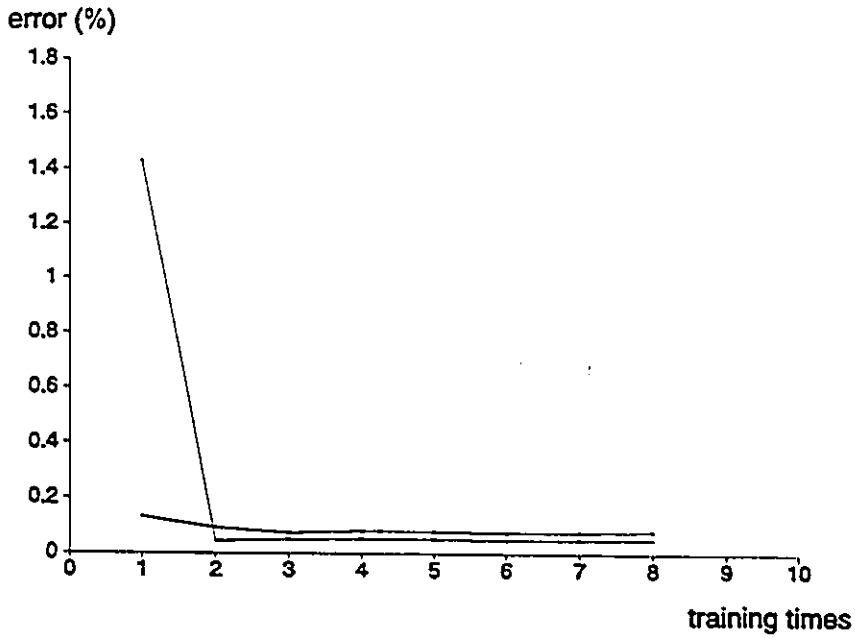

Figure 10.5 training results

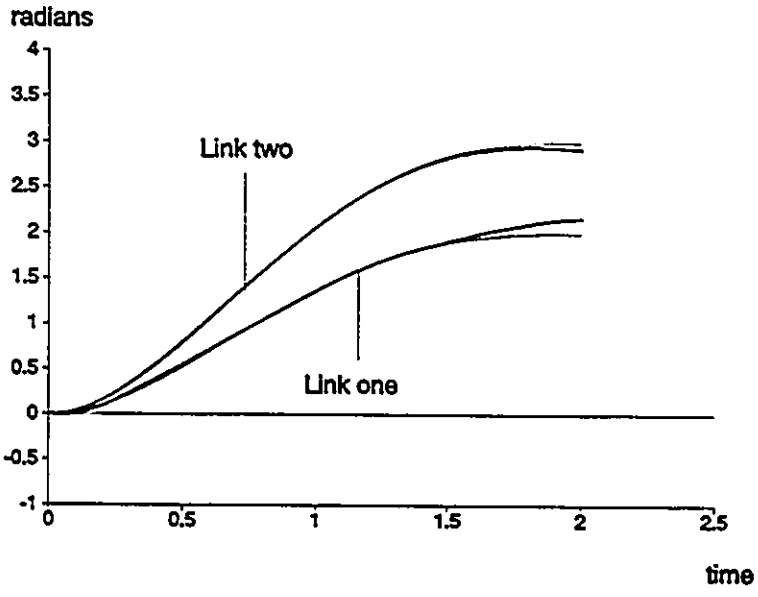

Figure 10.6 results after two learning steps

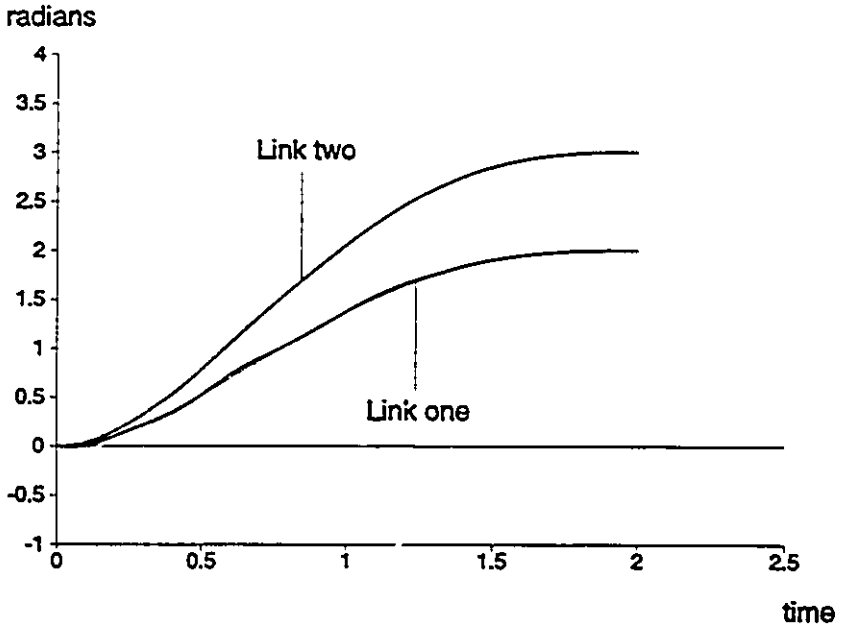

Figure 10.7 results after eight learning steps

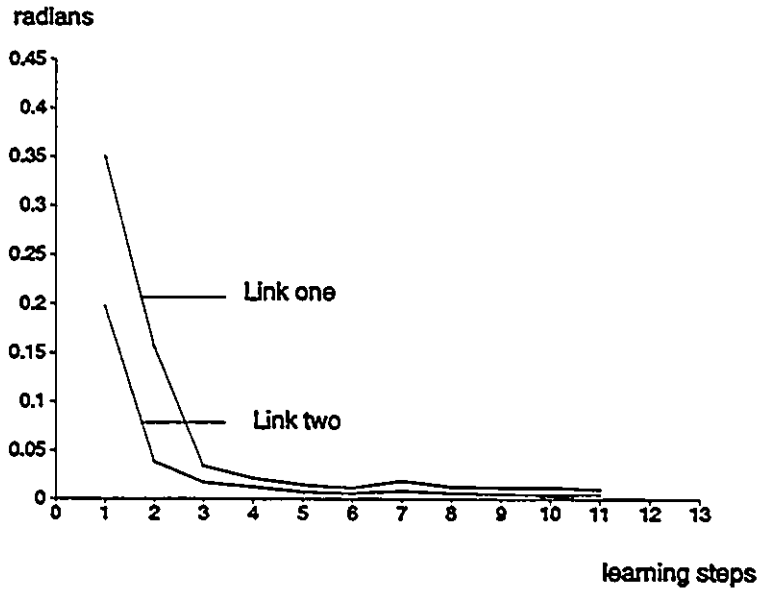

Figure 10.8 tracking errors vs learning steps

10.5 CONCLUSIONS

In this chapter, we have presented an iterative neural learning scheme for control of flexible joint robot by using the author's modified backpropagation algorithm. A case study on a two-link robot has demonstrated that the proposed neural learning controller is very promising and achieved satisfactory performance after a few trials. Generally, the proposed iterative neural learning controller has a faster rate of convergence and better robustness than the common iterative learning controller. The learning process occurs between two consecutive operations of the robot and the neural learning control is implemented as a feedforward controller. The required on-line computation is only for the PD feedback controller. Thus, this avoids the problem of heavy on-line computation required by adaptive controllers and robust controllers. In practice, we can develop an approximate model of the robot. Then we simulate this robot on a computer by using the inverse dynamic control scheme. The neural network is trained by using the simulation data. After this, the learning scheme described in figure 10.2 will put into the actual robot. It is expected that more precise the model of the robot, the faster will be the learning process. However, the neural learning controller needs retraining when the robotic trajectory has been changed. This is one drawback of the proposed approach compared with adaptive controllers and robust controllers. Our future work will be directed into experiments of real robotic manipulators using the learning controller proposed in this paper and more strict theoretical analysis.

CHAPTER ELEVEN CONCLUSIONS

In the previous chapters of this thesis, the author has described his research work in repetitive learning control of robotic manipulators. The work is significant to the development of advanced industrial robots, where high speed and high control precision are required. The objective of this thesis is to bring the low cost and high performance industrial robot into reality. In the proposed repetitive control scheme of this thesis, very little on-line computation is required. It can boost the performance of existing industrial robots. Also, almost no extra cost is needed. It is the author's hope that the work done in this thesis will attract the attentions of the industrial world.

It is well known that most industrial robots perform repetitive tasks. The drawbacks are that the tracking errors are also repeated. An intelligent robot should have the capability to learn from previous operations and improve performance by itself. The research in this thesis ensures that a robot has this desirable feature.

The work of this thesis has laid a theoretical foundation for further studies on the repetitive learning control of robotic manipulators. In this thesis, both neural networks and two operational modes theory based approaches have been found effective control techniques for a robot performing repetitive tasks. Which

approach is better. In the author's opinion, both methods have their advantages and disadvantages. On the side of two operational modes, all conclusions are mathematically elegant and more systematic. However, the controllers have to be designed separately for different applications. This requires additional work in the design process. As to the neural network based method, the control schemes are simple and unified to a large extent. For example, motion control and force control can be implemented on the same structure of a neural network. The extra work needed is to change the input and outputs nodes, but it is more or less heuristic. Generally, a neural network based approach can achieve satisfactory results by appropriate training of neural networks. The neural network based method needs more work and more time before its behaviour is fully understood. Although, it is an effective alternative approach to the conventional controllers, it cannot totally replace the conventional controllers. By comparison, conventional learning controllers are more appropriate for force control applications, while neural networks based learning controllers have advantages in the area of applications to flexible joint robots. Better controllers are usually achieved by combining the benefits of these two controllers and suppressing their shortcomings.

The two operational modes based method has an outstanding feature. The designed controller is a combination of an adaptive controller and an iterative learning controller. In the single operational mode, it is an adaptive controller, while

an iterative learning controller in the repetitive operational modes. This unique approach ensures that a robot has a good performance in the initial trials while improving the over-all control precision as repetitive operations go on. This gives a robot a good start-up performance and avoids down-grade of precision as the robot ages.

The author acknowledges that neural network is an exploding research area and advances very quickly. Most of work regarding the neural network part in this thesis was done nearly two years ago. It is not surprising that the approaches proposed here may not be the best compared with the latest results. However, the neural network employed in this research work is simple yet quite effective. Also, emphasis of this work is laid on the application of neural networks to control of a robot, rather than on research in the theory of neural networks. A neural network here provides an alternative approach to implement the repetitive learning controllers of robots. As demonstrated in this research work, neural network based techniques are an integral part of a class of effective repetitive learning controllers.

This thesis is mainly devoted to the theoretical work of repetitive learning control of robots. Most of the conclusions have been mathematically verified and supported by many computer simulation studies. Thus, the author is quite confident that the proposed schemes are able to work very well in practical robots, because the proposed schemes are simple and require little computation. It is to be

expected that simulation results are obtained in a more ideal situation than the practical ones. Thus, it is the author's sincere hope that this research work can attract special attentions of the industrial robotic field and find more and more applications for this research work. As a matter of fact, the research results can enhance the performance of the existing industrial robot with few costs.

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

11.1 Suggestions for Further Research

1. In the theory of two operational modes, the feedback gains have to be selected sufficiently large, but specific values have not been given. Further work should address this issue. Intuitively, the selection of these feedback gains is dependent on the present tracking errors and tracking errors from the previous operations. A possible solution is to update on line the feedback gains in order to remove the restriction mentioned above. However, this contradicts the assumptions of repetitive learning controller that little on-line computation should be

involved.

2. In the neural network part, a lot of simulation studies suggest that the author's modification of back-propagation neural networks are quite effective and easily stabilized, but a strict mathematical proof is not given. The convergence of neural network is still an open problem. Also, some partial results are reported recently, but they provide little help for practical training of the neural networks. In the author's view, much research is needed before the potential of neural networks is fully utilized.

3. The emphasis of further research is to apply the proposed theory to industrial robots.

REFERENCES

[1] R. Ortega and M.W. Spong, "Adaptive Motion Control of Rigid Robots: A Tutorial," Proceedings of the 27th Conference on Decision and Control, pp. 1575-1584, December, 1988, Austin, Texas.

[2] J.J.E Slotine and W. Li, " Adaptive Manipulator Control: A Case Study," IEEE Trans. Automatical Control, vol. 33, No. 11, pp. 995-1003,Nov. 1989.

[3] M.W. Spong, "Modelling and Control of Elastic Joint Robots," ASME J.Dynam. Sys. Meas. Contr., vol. 109. pp 310-319, 1987.

[4] M.W. Spong, "Adaptive Control of Flexible Joint Manipulators", IEEE Control Systems Magezine, pp 9-13, Sep., 1989.

[5] M.W. Spong," On the Force Control problem for Flexible Joint manipulators", IEEE Trans. Automat. Contr., vol. 34, No. 1,pp 107-111 Jan. 1989.

[6] R. Lozano, etc," Adaptive Control of Robot Manipulators with Flexible Joints", IEEE Trans. Automat. Contr., Vol. 37, No. 2, pp 174-181, Feb. 1992.

[7]. S. Nicosia and P. Tomei, "A New Approach to Control Elastic Joint Robots with Application to Adaptive Control", IEEE Proceedings of the t Conf. Decision and Control, pp 343-347, Dec. 1991.

[8] M.W. Spong and M. Vidysagar, Robot Dynamics and Control, John Willey &Sons, 1989

[9]. J.G. Fu and N.K Sinha," Iterative Learning Control of Flexible Joint Robots with Neural Networks", IFAC Symposium on Intelligent Components and Instruments for Control Applications, pp 313-318, May 1992, Spain.

[10]. J.G. Fu and N.K. Sinha, " A Learning Scheme for Force Control of Robotic Manipulators Using Neural Networks", IEEE Singapore International Conference on Intelligent Control and Instrumentation, pp 398-405,Jan.,1992.

[11]. J.G. Fu and N.K. Sinha, " An Iterative Learning Scheme for Motion Control of Robotic Manipulators Using Neural Network: A Case Study", to appear in Journal of Intelligent and Robotic Systems.

[12]. J.G. Fu, " Division by Polynomial Matrix and Its Application to Multivariable Linear Systems", Canadian Conference on Electrical and Computer Engineering, 1990, Ottawa, Canada

[13]. J.G. Fu and J.B. Cai, " Fuzzy Identification of Systems and its Applications to Process Control", 8th IFAC Symposium on Identification and System Parameter Estimation', 1988, Beijing, China

[14]. J.G. Fu and N.K. sinha, " Adaptive Learning Control of Robotic Manipulators", Considered Submitted to IEEE Transactions on Systems, Man and Cybernetics.

[15]. J.G. Fu and N.K. Sinha, " A New Controller Design of Constraint Robots Based on a Two-operational Modes", Considered Submitted to IEEE Transactions on Robotics and Automation.

[16] R. Middleton & G. Goodwin, " A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task", The International Journal of Robotic Research", Oct., 1989, pp. 67-74

[17]. M. Cohen and T. Flash, " Learning Impedance Parameters for Robot Control Using an Associative Search Network", IEEE Transactions On Robotics and Automation, Vol. 7, NO. 3, June 1991, pp. 382-389.

[18]. K. Narenda etc., " Identification and Control of Dynamic Systems Using Neural Networks", IEEE Transactions on Neural Network, Vol. 1, No. 1, March 1990, pp. 4-27.

[19]. Chae H. An, (1988), Model-based Control of A Robot manipulator, MIT Press.

[20]. Y. Han etc. " Adaptive Control of Constrained Robots", Second Workshop on Military Robotic Applications", August 1989, Kingston, Ont., Canada, pp. 298-303.

[21]. B. Horne etc., " Neural Networks in Robotics: A Survey", Journal of Intelligent and Robotic Systems 3: pp. 51-66, 1990

[22]. Y. Iiguni etc. " A Nonlinear Regulator Design in the Presence of System Uncertainties Using Multilayered Neural Networks", IEEE Transactions on Neural Networks, Vol. 2, No. 4, July 1991, pp.410-417.

[23]. W. Messner, " A New Adaptive Learning Rule", IEEE Transactions on Automatic Control", Vol. 36, No. 2, February 1991, pp. 188-197

[24] Arimoto S., "Bettering Operation of Dynamic Systems by Learning: A New Control Theory for Servomechanism or Mechanics Systems", in Proceedings of Conference on Decision and Control, pp. 1064-1069, Las Vegas, NV., 1984

[25] Arimoto S., "Robustness of Learning Control for Robotic Manipulators", in IEEE International Conference on Robotics and Automation, pp. 1528-1533, 1990.

[26] Bondi Paola, "On the Iterative Learning Control Theory for Robotic Manipulator", in IEEE Journal of Robotics and Automation, Vol.4, 14-21, 1988.

[27] Craig J.J., Adaptive Control of Mechanical Manipulators, Adison-Wesley, MA. 1988

[28] Kawato M., "Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics", IEEE Control Systems Magazine, pp. 9-15, 1988.

[29] McClamroch and D. Wang, "Feedback Stabilization and Tracking of Constrained Robots", IEEE Transaction on Automatic Control, vol 33, No. 5, 1988.

[30] P.V. Kokotoviv, H. K. Khalil, and J. O'Reilly, Singular Perturbation Methods in Control: Analysis and Design, London: Academic Press, 1986.

[31] M.W. Spong, "Comments on ' Adaptive Manipulator Control: A Case Study'", in IEEE Transactions on Automatic Control, pp.762-763, 1990.

[32] Aicardi M., "Combined Learning and Identification Control Techniques in the Control Manipulator", in Proceedings of the 28th Conference on Control and Decision, pp. 1651-1656, 1989.

[33] You-Liang Gu, "On Nonlinear Systems Invertibility and Learning Approaches by Neural Networks", 1990 American Control Conference, pp. 3013-3017.

[34] D.F. Rumelhart, Parallel Distributed Processing, Vol.1, Foundations, MIT. 1986

[35] N.K. Sinha and B. Kuszta, 1983, Modelling and Identification of Dynamic Systems, Van Nostrand Reinhold.

[36] T.Yabuta, "Possibility of Neural Networks Controller for Robot Manipulator", in 1990 IEEE International Conference on Robotics And Automation, pp. 1686-1691.

[37] V. Zeman, "A Neural Network Based Control Strategy for Flexible Joint Manipulators", in 1989 IEEE International Conference on Robotics and Automation, pp. 1759-1764.

[38] N. Sadegh, "A Unified Approach to the Design of Adaptive and Repetitive Controllers for Robotic Manipulators," in Journal of Dynamic Systems, Measurement, and Control, December, pp. 618-629, 1990.

[39] Daniel E. Whitney, "Historical Perspective and State of the Art in Robot Force Control", The International Journal of Robotics Research, Vol. 6, 1987, pp. 3-14.

[40] S. Kawamura, "Hybrid Position / Force Control of Robot Manipulators Based on Learning Method", 1985 ICAR, pp. 235-242.

[41] R. Su & N. Kermiche, "A Learning Scheme for Open-loop and Closed-loop Control", 1989 IEEE International Conference On Robotics And Automation, pp. 523-528.

[42] R. Su & N. Kermiche, "Learning Control For A Class of Nonlinear Systems", 1989 IEEE International Conference on Robotics and Automation", pp. 582-585.

[43] K. Moore, "Iterative Learning For Trajectory Control", Proceedings Of the 28th Conference on decision and Control", Tampa, Florida, Dec. 1989, pp. 860-865.

[44] Z. Geng, "Learning Control System Design Based on 2-D Theory: An Application to Parallel Link Ma nipulation", 1990 IEEE International Conference on Robotics and Automation", pp. 1510-1515.

[45] "D. W. Daws, "On the Learning Control of A Robot Manipulator", 1989 IEEE International Conference on Robotics and Automation, pp. 2632-2634.

[46]. L. Hidge, "Stability Analysis For Learning Systems", 1989 American Control Conference, pp. 85-87.

[47] S. Kawamura, "Realization of Robot Motion Based On a Learning Method", IEEE Transactions on SMC., vol 18, No.1, Jan./Feb., 1988, pp. 126-133.

[48] S. Hara, "Synthesis of Repetitive Control Systems and Its Application", Proceedings of the 24th Conference on Decision and control", Ft. Lauderale,FL., Dec., 1985, pp.1387-1392.

[49] T. Mita , "Iterative Control and Its Application to Motion Control of Robot Arm: A Direct Approach to Servo-problems", Proceedings of the 24th Conference on Decision and Control, Lauderale, FL., Dec., 1985, pp. 1393-1398.

[50] E. Lunde,"Practical Trajectory Learning Algorithms for Robot Manipulators", 1990 IEEE International Conference on Robotics and Automation, pp. 1516-1521.

[51] A. Deluca., "A Frequency Domain Approach to Learning Control:Implementation for a Robot Manipulator", 1989 IEEE International Conference on Robotics and Automation, pp. 66-71.

[52] M. Togai, "Analysis and Design of an Optimal Learning Control Scheme for Industrial Robots: A Discrete System Approach", Proceedings of the 24th Conference on Decision and Control", Ft. Lauderale,FL., Dec., 1985, pp.1399-1404.

[53] E. G. Harokopos, "Optimal Learning Control of Mechanical Manipulators in Repetitive Motions", 1986 IEEE International Conference on Robotics and Automation, pp. 396-401.

[54]. M. Togai, "Learning Control and Its Optimality: Analysis and Its Application to Controlling Industrial Robots", 1986 IEEE International Conference on Robotics and Automation, pp. 248-253.

[55] F. Miyazaki., "Learning Control Scheme for A Class of Robot Systems with Elacity", Proceedings of 25th Conference on Decision and Control, Athens,

Greece, December 1986, pp. 74-79.

[56] S. OH, "An Iterative Learning Control Method with Application for the Robot Manipulator", IEEE Journal of Robotics and Automation, Vol. 4, No. 5, Oct., 1988, pp. 508-514.

[57] John T. Wen., "New Class of Control Laws for Robotic Manipulates Part 1: Non-adaptive Case", INT. J. Control, 1988, vol.47, No. 5, pp. 1361-1385.

[58] John T. Wen , "New Class of Control Laws for Robotic Manipulates Part 2: Adaptive Case", INT. J. Control, 1988, vol.47, No. 5, pp. 1387-1406.

[59] You-liang Gu , "Learning Control in Robotic Systems", Journal of Intelligent and Robotic Systems 2: 198, pp. 297-305.

[60] John E. Hauser, "Learning Control for a Class of Nonlinear Systems", Proceedings of the 26th Conference on Decision and Control, Los Angeles, CA., 1987, pp. 859-860.

[61] John S. Bay, "Dynamics of a Learning Controller for Surface Tracking Robots on Unknown Surface", IEEE Transactions on Automatic Control, vol. 35, no. 9, Sep., 1990, pp. 1051-1054.

[62] R. H. Middleton, "Adaptive Control for Robot Manipulators using Discrete Time Identification", IEEE Transactions on Automatic Control, vol. 35, no. 4, May 1990, pp. 633-637.

[63] Z. Geng., "An Adaptive Learning Control Approach", Proceedings of the t Conference on Decision and Control, Brighton, England, Dec. 1991, pp 1221-1222.

[64] F. Pourboghrat, "A Learning Controller for Robotic Manipulators", Proceedings of the 27th Conference on Decision and Control, Austin, Texas, Dec. 1988, pp. 604-609.

[65] Mark D. Peek, "Parameter learning for Performance Adaptation", IEEE Control Systems Magazine, December 1990, pp. 3-11.

[66] T. Kuc, "An Iterative learning Control of Robot Manipulators", IEEE Transactions on Robotics and Automation, vol. 7, no. 6, Dec. 1991, pp. 835-841.

[67] Michael B. Leahy, "Neural Network Payload Estimation for Adaptive Robot Control", IEEE Transaction on Neural Networks. vol. 2, no. 1, Jan. 1991, pp. 93-100.

[68] M. Tenorio, "Self-organizing Network for Optimum Supervised Learning", IEEE Transaction on Neural Networks. vol. 1, no. 1, March 1990, pp. 100-110.

[69] Bill Horne, " Neural Networks in Robotics: A Survey", Journal of Intelligent and Robotic Systems 3: 1990, pp. 51-66.

[70] K. Watanabe, "Learning Algorithms for Neural Networks with the Kalman Filters", Journal of Intelligent and Robotic Systems 3: 1990, pp. 305-319.

[71] S. Chen, " Linear System Identification Using Neural Networks", Int. J. Control, vol. 51, no. 6, 1990, pp. 1191-1214.

[72] S. Chen, "Parallel Recursive Prediction Error Algorithm for Training Layered Neural Networks", Int. J. Control, vol. 51, no. 6, 1990, pp. 1215-1228.

[73] John K. Kruschke, "Benefits of Gain: Speed Learning and Minimal Hidden Layers in Back-Propagation Neural Networks", IEEE Transactions on SMC., vol. 21, no. 1, Jan./Feb. 1991, pp. 273-280.

[74] Lasse Holmstrom, " Using Additive Noise in Back-propagation Training", IEEE Transactions on Neural Networks, vol.3, no. 1, Jan. 1991, pp. 24-37.

[75] Moshe Cohen, "Learning Impedance Parameters for Robot Control Using an Associative Search Network", IEEE Transactions on Robotics and Automation, vol. 7, no. 3, June 1991, pp. 382-389.

[76] Derrick H. Nguyen, "Neural Networks for Self-learning Control Systems", IEEE Control Systems Magazine, 1990, pp. 18-23.

[77] Ming-shong Lan, "Adaptive Control of Unknown Dynamic Systems Via Neural Network Approach", 1989 American Control Conference, pp. 910-915.

[78] Huan Liu, "Neural Network Architecture for Robot Hand Control", IEEE Control Systems Magazine, 1989, pp. 38-42.

[79] "Naveen V. Bhat, "Modelling Chemical Process Systems Via Neural Computations", IEEE Control Systems Magazine, 1990 , pp. 24-29.

[80] Paul J. Werbos, "Back-Propagation Through Time: What It Does and How to Do it", Proceedings of IEEE, vol. 78, no. 10, Oct. 1990, pp. 1550-1560.

[81] Fu-chuang Chen, "Back-propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control", IEEE Control Systems Magazine, 1990, pp. 44-48.

[82] Demetri Psaltis, "A Multilayred Neural Network Controller", IEEE Control Systems Magazine, 1988, pp. 17-21.

[83] W. Miller, "Real-time Dynamic Control of an Industrial Manipulator Using a Neural Network Based Learning Controller", IEEE Transactions on Robotics and Automation, vol. 6 no. 1, Feb. 1990, pp. 1-9.

[84] Victor C. Chen and Yoh-Han Pao, "Learning Control With Neural Networks", IEEE International Conference on Robotics and Automation, 1989, pp. 1448-1453.

[85] Tae-young Kuc, "CMAC Based Iterative Learning Control of Robot Manipulators", Proceedings of the 28th Conference on Decision and Control, Tampa, FL. Dec. 1989, pp. 2613-2618.

[86] M.A. Johnson, "Adaptive Model-based Neural Network Control", IEEE International Conference on Robotics and Automation, 1990, pp. 1704-1709.