Neuro Feedback Linearization in the Control of Robotic Manipulators

Ngoo May Jin

NATIONAL UNIVERSITY OF SINGAPORE

2004

Neuro Feedback Linearization in the Control of Robotic Manipulators

Ngoo May Jin

(B.Eng.(Hons), M.Sc.)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgement

I thankfully express my gratitude to Prof Poo Aun Neow and Prof Chen Chao Yu Peter for supervising my dissertation.

Table of Contents

Chapters

Summary

This thesis investigates the trajectory-tracking performance of a robotic system under different control techniques, in particular the computed-torque control technique and state feedback linearization. A neural network control approach based on the state feedback linearization technique is also proposed and studied.

A two-link manipulator has highly nonlinear dynamic characteristics which are not easily controlled using conventional control approaches. Several model-based control approaches are available which compensates for these non-linear dynamics. However, the performance of such model-based approaches depends highly upon an accurate apriori knowledge of the robot's dynamic model which, in most cases, is difficult if not impossible to obtain.

Neural networks are used in the control schemes here, and they have been found to be able to model the manipulator's nonlinear dynamics. The advantage of using neural networks, when they can be trained using only the measured input-output data from the system-under-control, is the elimination of the need for an accurate dynamic model for good control performance.

Performance studies on the computed torque and neuro computed torque control schemes were first carried out. The neuro computed torque control scheme was found to have extremely good performance, almost matching the computed-torque's theoretically perfect tracking performance.

A nonlinear state feedback control scheme was then investigated. This control approach simplifies the system by compensating for the non-linear dynamics, essentially reducing the robot model to a linear system and thus amenable to control by known linear control schemes. The traditional linear approximation approach is not used here since, using this, reasonable performance is achievable over only a small range of state variables. The nonlinear state feedback linearization approach used here allows for operation over the entire operational range of the state variables.

Using simulations, the trajectory-tracking performance of this non-linear state feedback linearization approach was compared with that for the computed torque control approach. The computed torque control method is conventionally used to linearize a certain class of systems. The performance of the designed nonlinear feedback law in the present work was found to be comparable to that of the computed torque method.

Based on the non-linear state feedback linearization approach, a neural network control approach was developed. In this approach, the neural network controller was trained using only measured input-output data, thus eliminating the need for an accurate model of the system-under-control for good control performance. The performance of this neural network controller was found, through simulation studies, to be comparable to the non-linear controller designed assuming a perfect knowledge of the robot's dynamic model.

The main contribution of this dissertation is the application of the nonlinear state feedback controller for the control of a two-link robotic manipulator and the development of a neural-network controller based on this model-based approach. In this thesis, a nonlinear state feedback control law has been derived mathematically. This feedback law is applied to a two link robotic manipulator in order that the robot's closed loop system can be made linear. The current simulation work using the developed feedback law contributes towards the application of linearization techniques on nonlinear multi-link robotic system. Based on mathematical analysis and an experimental study, the proposed controller has been shown to give good tracking performance and stability. Simulation studies compare the trajectory-tracking performance of this approach to the more developed computed-torque control approach and its neural network equivalent.

List of Figures

Chapter 1 Introduction

1.1 Overview

Robotic manipulators are now being used widely, both in industry, for medical care and in the home. Accurate trajectory tracking are required in many such applications. In this dissertation, the control of a robotic manipulator for trajectory tracking is investigated.

The analysis and synthesis of the control system are well established for linear time-invariant systems [Ogata, 1970]. For a system with slow time-varying property, the adaptive control technique has proved to be a sensible solution [Narendra, et al., 1989]. However, for a non-linear system such as a robotic manipulator, control system design is typically handled on a case-by-case basis. Feedback linearization is a popular choice for deterministic system [Sidori, 1989]. However, feedback linearization implies a model-based control strategy in which its control performance is inherently sensitive to modeling accuracy [Zhu, et al., 1992]. In recent years, incorporating neural networks proved to be a popular method for the control of systems with significant nonlinearity, especially for the case that the plant nonlinearity is unknown [Hunt, et al., 1992].

The problem of controlling robotic manipulators is a challenging one as the dynamics of a robotic manipulator is highly non-linear. In addition, unmodeled dynamics, and environmental changes and unmeasurable disturbances during operation are just some of the uncertainties that prompt further research into better and more intelligent control schemes. Neural networks and feedback linearization techniques are the control techniques being investigated and applied in the work presented here. Feedback linearization is used to compensate for the non-linearities in the robot's dynamics. The resultant controllers designed are model-based and their control performance highly dependent upon an accurate knowledge of the robot's dynamic model. However, the latter is difficult, if not impossible to achieve. Furthermore, the dynamic model of the robot may change during operation, an example of which is when it picks up a payload thus changing its mass properties. Neural networks, with their abilities to be trained to approximate models, are used to avoid the need to have *apriori* knowledge of the plant's dynamic model.

In the work presented here, the trajectory-tracking performance of the computed torque control method, applied to a two-link robotic manipulator is compared with that obtained for a designed neural computed torque method. Next, a state feedback approach for the linearization of a class of non-affine non-linear systems was investigated and the mathematical analysis carried out for application to the same two-link robotic manipulator. Based on this linearization approach, a PD (proportional plus derivative) controller is designed and the trajectory-tracking performance of the controller determined and compared with that obtained for the computed torque approach. Based on this non-linear state feedback linearization approach, a neural network-based controller, together with the necessary training procedure, is designed. The advantage of this neural network controller is that it can be implemented using only measured plant input-output data and still achieve good control performance without the need to have any knowledge of the plant's dynamic model. With the neural network controller, an approach utilizing on-line re-training of the neural network controller can be implemented. This latter approach will be able to adapt and will be able to maintain good control performance in the face of environmental, modeling and operational uncertainties and changes during operation.

1.2 Thesis contributions

The main contributions of the work presented here are summarized below:

[1] Simulation work using the developed feedback law contributes towards the application of linearization techniques on nonlinear multi-link robotic systems. Current research work by others [Taware A, et al., 2003] focuses on development of feedback linearization for scalar output functions, for example, a one link robot system. Moreover simulation studies for scalar functions are rare in present research literature. Hence the motivation here will be to make use of the symbolic capability a program such as Matlab to do complex symbolic computations for two link systems. The new thing is that the current simulation work is done on vectored output functions, as illustrated by a two-link robot system. Accurate tracking results obtained illustrate the validity of the developed controller formulations. As the resulting controller is based on feedback from positions and speeds of the manipulator link, only conventional position and velocity sensors are required. Therefore this controller is practically viable. A neuro-feedback linearized controller is also simulated with good regression tracking results.

[2] Implementation of computed torque and neural computed control for a two-link robotic manipulator and simulation studies. The results are used as reference plots for the feedback linearization simulations.

[3] Investigation into a non-linear state feedback approach for the linearization of a class of non-affine non-linear systems and its implementation on a two-link robotic manipulator. Simulation results show that this control approach has a control performance comparable to that obtained by the computed torque approach.

[4] Development of a neural network control approach based on the non-linear state feedback method in (1) above. This neural network approach allows the neural network controller to be trained from actual measured plant input-output data. As such, an accurate apriori knowledge of the plant's dynamic model is not necessary and, because of the use of actual plant input-output data, the neural network controller is assured, assuming proper and adequate training, of being able to map the plant's actual dynamics accurately, thus achieveing good control performance. With on-line retraining, this neural control approach can be adaptive to operational changes and uncertainties.

1.3 Historical development background

There have been tremendous developments in nonlinear control theory over the last few decades. One such important nonlinear control technique is the feedback linearization technique [Marino et al., 1995]. Feedback linearization was first developed in the 1970s. This technique helps to transform a nonlinear system into a controllable linear system by means of static state feedback and nonlinear transformations. The feedback linearization problem was studied and became very important because of its potential use in industrial systems. Standard and well-established linear control theory and controller design approaches can be readily employed once a nonlinear system has been feedback linearized. On top of that, systems with multiple inputs and multiple outputs can also be linearized and decoupled, thereby allowing for the effective use of single loops with linear controllers.

1.4 The pitfalls of linear control

The common engineering practice assumes that a system be described by a set of linear differential equations

$$\dot{x} = Ax + Bu \tag{1.1}$$

where *x(t)* = state of the system,

A, *B* = time invariant matrices defining the properties of the system, and

u(*t*) = control effort

Assuming that (1.1) accurately describes the system behaviour, researchers and control practitioners can use well-developed techniques and properties derived from linear control theory for the analysis of the system and the design of appropriate controllers. These properties include

(i) a unique equilibrium point with a nonsingular matrix A,

(ii) a stable equilibrium point if the eigenvalues of A have negative real roots, and

(iii) possible analytic solutions of the linear differential equation.

The transient response can also be explicitly determined.

For cases where the control input $u(t)$ is present, properties include

(i) superposition,

(ii) asymptotic stability of the unforced system also ensures bounded input bounded output stability of the forced system, and

(iii) a sinusoidal input leading to a sinusoidal output of the same frequency.

Though linear system properties allows the use of good well-known design and analytical tools to achieve good control performance, any significant nonlinear characteristics in the system's behaviour may make approaches based on linear system theory inapplicable . Non-linear systems are much more complex and, in general, difficult to handle. If the nonlinear behaviours were to be neglected and linear system tools are used, the resulting control designs can have significantly degraded control performance with unpredictable stability characteristics. These are the limitations and pitfalls experienced by linear systems theory as they have difficulty encapsulating and compensating for the non-linear effects.

1.5 The need for nonlinear control techniques

All physical systems exhibit non-linear behavior, some more so than others. In a nonlinear system, the relationship between controlled and manipulated variables depends on the operating conditions. In such systems, linear control techniques may be applied in certain situations with satisfactory results where the nonlinearities are mild, or when the operating conditions do not change much. In the latter case, linearization around the locality of the operating point works quite well.

For many industrial systems with highly nonlinear behaviour, linear control techniques cannot be satisfactorily applied, particularly in cases where the systems operate over a wide range of operating conditions. Conventional linear controllers are sometimes used to control these highly nonlinear processes, but these controllers need to be tuned in a conservative manner in order to avoid unstable behaviour. The drawback in such an approach is that control performance can be seriously degraded, performing far from optimum conditions. Hence there is a need to use more sophisticated control techniques which will use information about the nonlinearities of the controlled system to achieve near-optimal control performance over the system's entire operational range.

1.6 Towards nonlinear control

Traditionally, nonlinear control systems are approached by taking linear approximations about equilibrium points x_{ei} that corresponds to constant inputs u_i [Marino et al., 1995].

Consider the nonlinear state space system described by the following equations

$$\dot{x} = f(x,u) \tag{1.2}$$

$$y = h(x) \tag{1.3}$$

where *x* is the system state, *u* the control effort or input, and *y* the system's output.

The small deviations $\dot{\xi}_i, \xi_i$ and v_i are

$$\dot{\xi}_i = \dot{x} - \dot{x}_{ei} , \tag{1.4}$$

$$\xi_i = x - x_{ei} \tag{1.5}$$

and $v_i = u - u_i$. (1.6)

where x_{ei} is the equilibrium state for a constant control input u_i. $\dot{x}_{ei}$, x_{ei} and u_i are the known nominal solutions.

The locally linearized model is given by

$$\dot{\xi}_i = F_i \xi_i + G_i v_i \tag{1.7}$$

$$y_i = H_i \xi_i \tag{1.8}$$

The Jacobian matrices evaluated at nominal solutions $x_{ei}, \dot{x}_{ei}$ and u_i are

$$F_i = \frac{\partial f}{\partial x}(x_{ei}, u_i) \tag{1.9}$$

$$G_i = \frac{\partial f}{\partial u}(x_{ei}, u_i) \tag{1.10}$$

and $H_i = \dfrac{\partial h}{\partial x}(x_{ei})$. (1.11)

At the equilibrium points,

$$f(x_{ei}, u_i) = 0 \tag{1.12}$$

$$h(x_{ei}) = 0 \tag{1.13}$$

This approach faces critical transition problems when one moves from one solution point to another. Maintenance of good performance and stability is difficult over wide ranges of variations of state variables.

1.7 Background of nonlinear control

During the seventies, nonlinear controllability and observability was initially studied using differential geometric tools. These studies led to the development of the nonlinear feedback control design theory [Schwarz 2000]. In practice, significant nonlinearities such as the centripetal, Coriolis and inertial forces could be exactly modeled using well-known physical laws. Engineers can then design nonlinear control algorithms that could better meet specifications which could not be met by means of linear control techniques. An example of such algorithms is the computed torque algorithm for high speed rigid-link robots in 1976. These algorithms mainly made use of nonlinear changes of state coordinates and of nonlinear state feedback's nonlinearity cancellation to make the closed loop system linear [Khalil 2003]. Nonlinear controls can outperform linear controls designed on the basis of linear approximations because nonlinear control algorithms can use all of the information contained in nonlinear models.

1.8 Model-based control

The nonlinear system's dynamic behaviour and information are represented by a set of nonlinear differential equations. With the design of the controller or control algorithm dependent on the dynamic model of the plant, feedback linearization and many other nonlinear control techniques become model-based in nature. If the nonlinear plant model can be obtained, a physical-based model will be derived from physical principles such as energy, force or momentum balance equations. Such models have the advantage of being applicable over the whole range of operating conditions. However, physical-based models are not always available or known, and even if so, the determination of accurate values of the parameters are often difficult. There are also costs and engineering efforts associated with the determination of these models. One solution could be to obtain the empirical dynamic model from measured input-output data using system identification techniques. There has been growing interest in the development of nonlinear dynamic models from input-output data.

Any model-based control design method will be prone to sensitivities to modelling errors. Models used for control system design cannot be infinitely precise and significant control performance degradation can result from errors in the model used for the design. Hence another possible solution to this problem is to obtain nonlinear empirical models from neural networks. Neural models are capable of being trained to map nonlinear dynamics, and this makes them a promising tool for nonlinear system modelling.

Chapter 2 Literature Review

- A Survey of Tracking Control techniques for Robots

2.1 Introduction

In the eighties, there were two different approaches to the control of uncertain systems. The first approach is that of adaptive control, and the second approach is that of robust control [Zhou 1998].

For the adaptive control approach, the designed controller adapts to the uncertain and/or changing parameters of the system. The “best” controller is thus obtained after learning or identifying the parameters of the system-under-control. Hence the adaptive controller can be applied to a wide range of uncertainties. For the robust control approach, the controller adopts a fixed structure. Such control structures give acceptable performance for a system with a specified uncertainty set. But they are simpler to implement, and there is no need to spend time on the tuning of the controllers. In the nineties, researchers have tried to merge the two approaches so that certain adaptive controllers can be robustified. In this way, the good qualities of both approaches can be combined.

2.2 Robust control

The robust control technique was applied to a nonlinear robotic system by Spong [Spong 1989, 2002] in 1992. The Lyapunov-based theory of guaranteed stability for uncertain systems is used to design the robust controller. The derived controller is innovative because the law depends on the inertia parameters of the robot, wheareas earlier controllers relied on the reference trajectory, manipulator state vectors and the inertia parameters. The controller was based on the adaptive control algorithm developed by Slotine and Li [Slotine, et al., 1998] in 1988. The closed loop system is globally convergent with the position tracking errors converging to zero and the parameter estimates remaining bounded. During the position tracking simulation, errors obtained after two seconds are –2.17E-4 for the first link position, and 2.28E-4 for the second link position. Such error records are considered small, and it further demonstrates that the adaptive controller is able to achieve global convergence. Such controllers are useful in robots that involve grinding operations with end-point force feedback. This is because in such environments, uncertainty is small, and robustness to disturbances and unmodeled dynamics are of importance.

2.3 Adaptive control

In 1995, Rafizadeh and Perz [Rafizadeh, et al., 1995]] applied robust and adaptive control techniques for their simulation studies on trajectory control of the Puma 560 robot model. Although perfect state convergence is achieved the tuning of Craig's adaptive controller is manual and also very time consuming. The parameters also tend to saturate within bounds of 0.01.

Parameter adaptive control is also used by other researchers. They used a gradient parameter update law, in addition to a tracking control law, as asymptotic exact cancellation of nonlinear terms are needed. Since exact cancellation of nonlinear terms is not possible, exactly linearizing control law implementations are difficult. The current work on the derivation and implementation on the feedback control law did not involve any use of adaptive parameter update laws. The linearizing feedback control law is in itself sufficient to give good tracking results.

2.4 Feedback linearization control

Design of nonlinear state feedback control began in the early eighties for certain simple classes of single-input-single-output nonlinear systems. Feedback linearizable and input-output linearizable systems are two common areas studied at that time.

For feedback linearizable systems, the state space equations are made linear in certain state coordinates via state feedback. Once the non-linear system has been linearized, conventional linear control design methods, such as the pole placement method, can be used.

For input-output linearizable systems, the input-output dynamics are linearized using state feedback controllers that may make certain dynamics unobservable from the output. The zero-pole cancellation technique is used.

Both methods have a reliance on exact cancellation of possible nonlinear terms containing uncertain parameters.

Since 1987, feedback linearizable system control design for uncertain parameters was done by Sastry and Isidori [Sastry, et al., 1989]. They used parameter adaptation to robustify the exact cancellation of nonlinear terms. This is because the two methods mentioned above suffered from the assumption that the model dynamics are certain.

But if the model is to contain uncertain nonlinear terms, exact cancellation of nonlinear terms is not possible. Hence parameter adaptive control filled in the weakness of the early methods.

Taware and Gao developed linearized feedback laws for a single-link manipulator arm system[Taware, et al., 2003] in 2003. They addressed the control problems involved for simple nonlinear system models, and it was noted that simulation work was not carried out for the verification of their developed controller laws. They proved the asymptotical stability of simple nonlinear systems under their developed controller laws.

Simulation was carried for flexible two-link joint robots by Berger [Berger, et al.,1992] in 1992. Trajectory tracking results are obtained for parametric errors of up to 50%.

In the early twentieth century, the availability of powerful computational microprocessors encouraged researchers to carry out simulation and testing of innovative nonlinear control algorithms for robotic applications.

2.5 Neural network control

Neural network controllers for robot manipulators are ‘model-free’. Hence they are a good alternative to robust and adaptive control techniques. Such controllers can be made to learn on-line the systems that they control.

Various robot control schemes have been developed in the literature. Two such control schemes will be investigated and their simulation results will be discussed.

Kim and Lewis developed a robust neural network output feedback scheme for closed loop output feedback control [Kim, et al.,1999]. Joint velocity measurements are not needed for their scheme. The weights of the neural network controller are tuned online, and off-line learning is not required. Exact knowledge of robot dynamics is also not required. Simulation results of their proposed scheme showed that their neural network controller is capable of overcoming uncertainties. They compared their results with a proportional derivative (PD) controller. The PD controller shows that there are oscillatory behaviours in the tracking errors. By comparison, the neural controller can minimize errors even when the end-effector's mass has been changed.

Sliding mode neural network (SMNN) controllers are also used for tracking control of robots. For the SMNN controller developed by Wai [Wai 2002], the tracking errors converge quickly. High precision control is the desired aim, and asymptotic stability of the control system is to be guaranteed since the adaptive learning algorithms in the SMNN control system are derived from Lyapunov stability analysis.

Flexible link manipulators are also used for position tracking simulations under neural network based controllers. Talebi and Patel [Talebi, et al.,2000] developed several neural network schemes. These schemes are simulated and tested experimentally on a single flexible link test-bed. Their networks are trained online, and offline training is not needed. Their experimental results demonstrate the advantages of neural network controllers over model-based PD controllers.

Static neural networks have been used for many research simulations and experiments in the literature. It is a challenge to incorporate dynamic neural networks into neural controllers for robot tracking control. Sun and Li [Sun, et al., 2002] developed dynamic neural network(DNN) adaptive controllers for robot manipulators with unknown nonlinear dynamics. Their simulation results show that the performance of the DNN controller is better than that of the static neural network(SNN) based controller.

Intelligent optimal control techniques can also be combined with neural networks for trajectory tracking of robots. Kim solved the algebraic Riccati equations so that explicit solutions to the Hamilton-Jacobi-Bellman equation for optimal control of robotic systems may be solved [Kim, et al., 1999]. Their proposed neural adaptive learning scheme gives satisfactory tracking results. This scheme is robust and can adapt to changing system dynamics.

Experimental results by Gupta and Sinha [Gupta, et al., 2000] show that it is practically viable to combine neural networks and the PD controller for trajectory tracking. Their results also show that a neurocontroller still performs satisfactorily when there are uncertainties. Performance of conventional schemes deteriorates slightly when there are uncertainties that could not be included in the dynamic model.

Patino developed feedback adaptive neurocontrollers for trajectory tracking of robots. [Patino, et al., 2002] They combined feedforward neural networks with adaptive and robust control techniques. Their simulation studies on a PUMA 560 robot show that the control error converges asymptotically to a neighbourhood of zero. This is because they used a bank of off-line trained fixed neural networks instead of conventional backpropagation networks.

Experimental studies with neural control using conventional backpropagation algorithms were done on a PUMA 560 robot by Acosta [Acosta, et al., 1999]. The neural network controller was implemented on a computer and analog-to-digital (A/D) converters, digital-to-analog (D/A) converters and optical encoders were used for the issue and capture of torque values to and from the robot links. The neural controller gave better experimental results than the conventional PD controller. However, it was reported that the neural controller faced implementation difficulties. During startup, the robot exhibited erratic movements since the joint angles took on arbitrary initial values. Initial weight assignments were random, but a proposed solution was to assign values for the initial weights based on those found from previous experiments.

Chapter 3 Computed Torque and Neural Computed Torque Control

3.1 Summary

In this chapter, the theoretical background for two control approaches are discussed and developed. These are computed torque control and neuro-computed torque control.

Implementation issues in respect of a two-link robotic manipulator are discussed. In a subsequent chapter, simulation experiments are discussed and performance results presented.

3.2 Robot Dynamic Model

The dynamic model of a robot can be written as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{V}_m(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{\tau}_d = \boldsymbol{\tau} \tag{3. 1}$$

where

$\boldsymbol{M}(\boldsymbol{q})$ = inertia matrix,

$\boldsymbol{V}_m(\boldsymbol{q},\dot{\boldsymbol{q}})$ = Coriolis/centripetal matrix,

$\boldsymbol{F}(\dot{\boldsymbol{q}})$ = friction terms,

$\boldsymbol{G}(\boldsymbol{q})$ = gravity vector,

$\boldsymbol{\tau}_d$ = disturbances, and

$\boldsymbol{\tau}$ = torque control input.

Figure 3.1 shows a two-link robot manipulator.

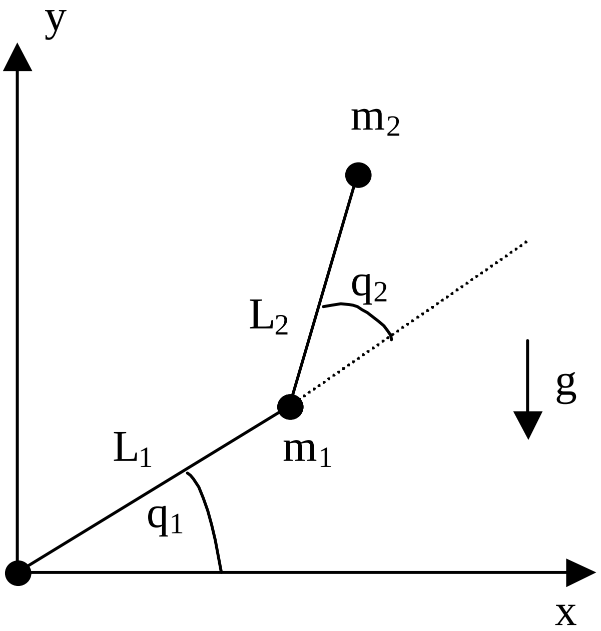

Figure 3. 1 Two-link robot manipulator model.

Assuming that the masses are point masses located at the ends of the links, the links have neligible masses, and neglecting friction, the dynamic model of the two-link manipulator shown in Figure 3.1 can be written as

$$\begin{aligned} \tau_1 = {} & [(m_1 + m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos(q_2)]\ddot{q}_1 \\ & + [m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)]\ddot{q}_2 \\ & - \dot{q}_2 m_2 l_1 l_2 \sin(q_2)\dot{q}_1 \\ & - (\dot{q}_1 + \dot{q}_2)m_2 l_1 l_2 \sin(q_2)\dot{q}_2 \\ & + (m_1 + m_2)g l_1 \cos(q_1) \\ & + m_2 g l_2 \cos(q_1 + q_2) \end{aligned} \tag{3. 2}$$

$$\begin{aligned} \tau_2 = {} & m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)]\ddot{q}_1 + m_2 l_2^2 \ddot{q}_2 \\ & + \dot{q}_1^2 m_2 l_1 l_2 \sin(q_2) \\ & + m_2 g l_2 \cos(q_1 + q_2) \end{aligned} \tag{3. 3}$$

where τ = Torque, m = mass, q = link angular position, g = gravitational acceleration and l = length of link, and the subscripts 1 and 2 refer to Link 1 and Link 2.

3.2.1 Summary of control problem

Assuming that the whole state $(q_1, \dot{q}_1, q_2, \dot{q}_2)$ is measured, a control law is needed to compute the values of τ such that $\boldsymbol{q}$ tracks a desired reference $\boldsymbol{q}_r(t)$.

3.3 Neural Networks- Backpropagation

Neural networks has the ability to learn the nonlinearities of a system and is able to do function approximation. The neural network is a vector-valued nonlinear function that provides a nonlinear mapping process from the input signal vector to the output signal vector. Learning by, or training of, a neural network is done by presenting it with training pairs of vectors of inputs and the corresponding desired outputs. Based on these training pairs, the neural network adjusts its internal weights in such a way as to approximate the function represented by the training pairs through a process known as back-propagation[Haykins, 1999].

3.3.1 Neural Network Architecture

A feedforward neural network is used in this work for the neuro-computed torque control scheme. Two neural sub-networks are used, one to generate each of the two control torques required. The first sub-network is used to generate the control torque for Link 1 while the second sub-network is used for that for Link 2. Three processing layers, including two hidden layers, are used for each sub-network with the first layer having 10 neurons, the second layer 5 neurons, and the third output layer one neuron.

When neural networks are used, the required nonlinear input-output mapping is assumed to have a functional relationship described by $\boldsymbol{d} = \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{d}$ is the output vector, and $\boldsymbol{x}$ is the input vector. The vector-valued function $\boldsymbol{f}$(.) is assumed to be unknown. A set of labeled examples $\varsigma = \{(\boldsymbol{x}_i, \boldsymbol{d}_i)\}_{i=1}^{N}$ are given so as to make up for the lack of knowledge in the function $\boldsymbol{f}$(.). $\boldsymbol{d}_i$ is the desired response. This set of labeled examples is used to train a neural network as a model of the system.

Figure 3,2 shows the architecture of a multi-layer perceptron, one form of feedforward neural network.

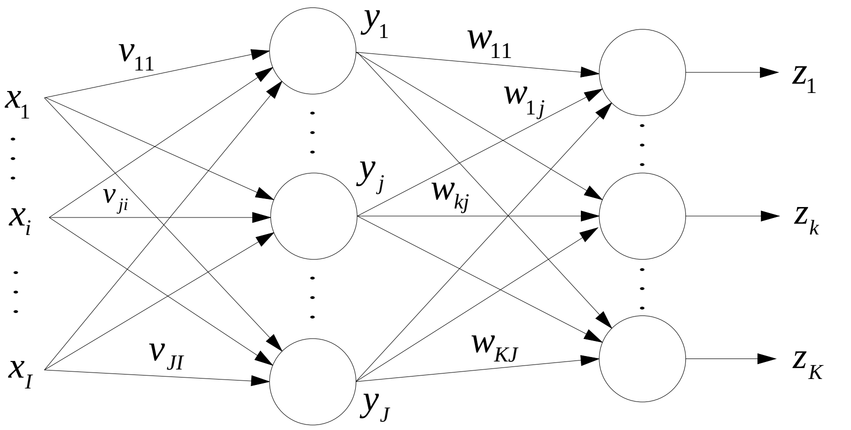

Figure 3.2 Architectural graph of a multiplayer perceptron with one hidden layer.

I, *J*, *K* are the number of nodes in the input, hidden, and output layer respectively. x_i, y_j, z_k are the outputs of the *i*th, *j*th and *k*th nodes of the input, hidden and output layers respectively. v_{ji} is the weight connecting the *i*th input node to the *j*th node in the hidden layer and w_{kj} is the weight connecting the output of the jth node in the hidden layer to the input of the kth node in the output layer.

The backpropagation algorithm used for training the neural network in the work described here is described as follows. This method is called error backpropagation because error signals are first computed at the outputs of the last layer of the network. These are then propagated backward through the network to compute the corresponding error signals at each of the outputs of the neurons in the hidden layer. These error signals are used to compute the necessary adjustments to the connecting weights in the neural networks. In this way, the neural network is trained by having its connecting weights adjusted. The error backpropagation procedure is described in details in the following sections.

A training pair comprises the input vector

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_I \end{bmatrix}^T \tag{3. 4}$$

together with the corresponding desired output value vector

$$\mathbf{d} = \begin{bmatrix} d_1 & d_2 & \dots & d_K \end{bmatrix}^T \tag{3. 5}$$

When presented with the input vector **x,** the first layer gives the output

$$\mathbf{y} = \mathbf{\Gamma}(\mathbf{vx}) = [y_1 \quad y_2 \dots y_J]^T \tag{3. 6}$$

and for the second layer, the output is

$$\mathbf{z} = \mathbf{\Gamma}(\mathbf{wy}) = [z_1 \quad z_2 \quad \dots \quad z_K]^T \tag{3. 7}$$

where **v** is the weight matrix between the first two layers, and **w** the weight matrix between the second and the third layers. In general, the computed output vector **z** will not be the same as the desired output vector **d**. The error at the *k*th neuron in the output layer is then given by

$$e_k = d_k - z_k \,. \tag{3. 8}$$

For a bipolar sigmoid activation function, the error signal vector at the *k*th neuron in the output layer is

$$\delta_{zk} = \frac{1}{2}(d_k - z_k)(1 - z_k^2) \,. \tag{3. 9}$$

The error signal at the output layer, given by Equation 3.9, is backpropagated to produce the error signals at the output of the *j*th neuron in the hidden layer which is given by

$$\delta_{yj} = (1 - y_j^2)\sum_{k=1}^{K}\delta_{zk} w_{kj} \tag{3. 10}$$

The weight increments are then given by

$$\Delta \mathbf{w} = \eta \boldsymbol{\delta}_z \mathbf{y}^T \tag{3. 11}$$

and

$$\Delta \boldsymbol{v} = \eta \boldsymbol{\delta}_y \boldsymbol{x}^T \,. \tag{3. 12}$$

The weights are updated with the weight incremental values, and the above described algorithm is repeated with a different set of training pair until the error in the output decrease to some specified value.

3.4 Computed torque control

Consider the robot model as given in Equation (3.1). This can be simplified as

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q},\dot{\boldsymbol{q}}) . \tag{3. 13}$$

with $\boldsymbol{H}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{V}_m(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{\tau}_d$

The control torque is computed as

$$\boldsymbol{\tau} = \hat{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{u} + \hat{\boldsymbol{H}}(\boldsymbol{q},\dot{\boldsymbol{q}}) \tag{3. 14}$$

where $\hat{\boldsymbol{M}}(\boldsymbol{q})$ and $\hat{\boldsymbol{H}}(\boldsymbol{q},\dot{\boldsymbol{q}})$ represents estimates of $\boldsymbol{M}(\boldsymbol{q})$ and $\boldsymbol{H}(\boldsymbol{q},\dot{\boldsymbol{q}})$ respectively.

The term ***u*** in Equation (3.12) is computed as

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}_d + k_v(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + k_p(\boldsymbol{q}_d - \boldsymbol{q}) \tag{3. 15}$$

where $\ddot{\boldsymbol{q}}_d$, $\dot{\boldsymbol{q}}_d$ and $\boldsymbol{q}_d$ are the desired or reference input values of acceleration, velocity and angular positions of the links respectively, and k_v and k_p are constants representing derivative and proportional gains of the PD controller. In practice, the joint positions are measured very accurately with position encoders. The joint velocity is usually measured using a tachogenerator, which may be subject to small noise disturbances.

Expansion of (3.15) gives

$$u_1 = \ddot{q}_{d_1} + k_v(\dot{q}_{d_1} - \dot{q}_1) + k_p(q_{d_1} - q_1) \tag{3. 16}$$

and

$$u_2 = \ddot{q}_{d_2} + k_v(\dot{q}_{d_2} - \dot{q}_2) + k_p(q_{d_2} - q_2) \tag{3. 17}$$

If perfect knowledge of the robot’s dynamic model is available, then

$$\hat{\boldsymbol{M}}(\boldsymbol{q}) = \boldsymbol{M}(\boldsymbol{q}) \tag{3. 18}$$

and

$$\hat{\boldsymbol{H}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{3. 19}$$

From Equations (3.13) to (3.17), we obtain the following

$$\ddot{\boldsymbol{q}} = \boldsymbol{u}\,. \tag{3. 20}$$

Using Equation (3.14) for the two-link manipulator given by Equations (3.2) and (3.3) gives

$$\begin{aligned}\tau_1 = {} & [(m_1 + m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos(q_2)]u_1 \\ & + [m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)]u_2 \\ & - \dot{q}_2 m_2 l_1 l_2 \sin(q_2)\dot{q}_1 \\ & - (\dot{q}_1 + \dot{q}_2) m_2 l_1 l_2 \sin(q_2)\dot{q}_2 \\ & + (m_1 + m_2) g l_1 \cos(q_1) \\ & + m_2 g l_2 \cos(q_1 + q_2)\end{aligned} \tag{3. 21}$$

$$\begin{aligned}\tau_2 = {} & m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)]u_1 + m_2 l_2^2 u_2 \\ & + \dot{q}_1^2 m_2 l_1 l_2 \sin(q_2) \\ & + m_2 g l_2 \cos(q_1 + q_2)\end{aligned} \tag{3. 22}$$

where u_1 and u_2 are obtained from Equations (3.16) and (3.17).

Substitution of (3.20) into (3.15) gives

$$\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_d + k_v(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + k_p(\boldsymbol{q}_d - \boldsymbol{q}) \tag{3. 23}$$

Rearranging (3.23) gives

$$\ddot{\boldsymbol{e}} + k_v\dot{\boldsymbol{e}} + k_p\boldsymbol{e} = 0 \tag{3. 24}$$

where $\boldsymbol{e} = \boldsymbol{q}_d - \boldsymbol{q}$ is the trajectory-tracking error.

We can re-write Equation (3.24) in the form

$$\ddot{\boldsymbol{e}} + 2\rho\omega_n\dot{\boldsymbol{e}} + \omega_n^2\boldsymbol{e} = 0 \tag{3. 25}$$

in which ω_n is the undamped natural frequency and ρ is the damping factor.

Comparison of (3.24) with (3.25) gives

$$k_v \equiv 2\rho\omega_n \tag{3. 26}$$

$$k_p \equiv \omega_n^2 \tag{3. 27}$$

Equation (3.24) is the error equation which states that if the initial error is zero, that is, $\dot{\boldsymbol{e}} = 0$ and $\boldsymbol{e}=0$, then the error $\boldsymbol{e}$ will be always zero, thereby giving perfect tracking. If there is some initial value of error, Equation (3.23) states that the error will tend to zero with time as long as k_v>0 and $k_p > 0$.

Figure 3.3 shows the schematic diagram illustrating the computed torque control method. The dotted box represents the regressive part that the neural computed torque control method aims to approximate. There is an analogy between the boxed section in Figure 3.3 and Figure 3.4.

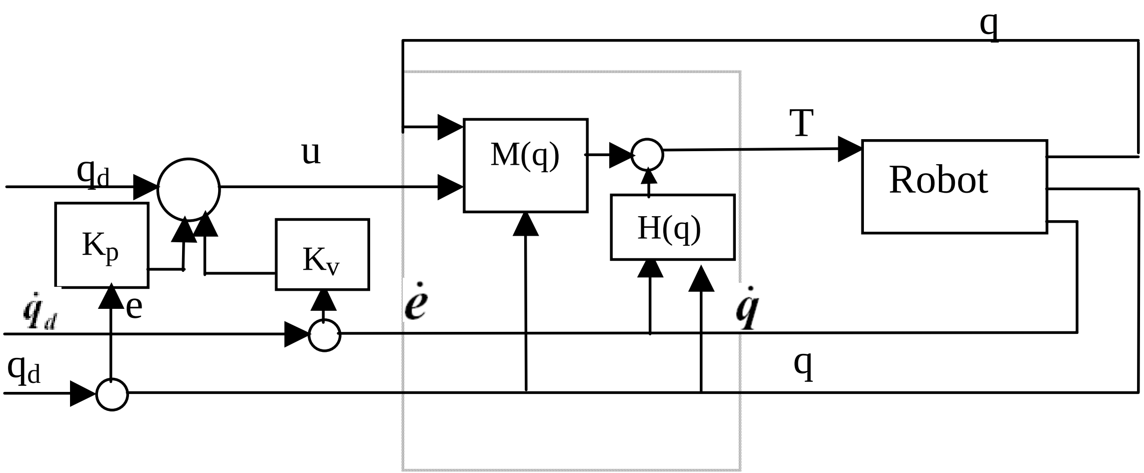

Figure 3.3 Computed torque control.

The algorithm of the discrete form of the computed torque controller, as implemented in a computer, is as follows:

(i) At sampling instant k, at time $t=t_k$, compute desired trajectory values of $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$ and $\ddot{\mathbf{q}}_d$.

(ii) Use the $\mathbf{q}$ and $\dot{\mathbf{q}}$ output at $t=t_k$ to compute the acceleration input, $\mathbf{u}$, of Equation (3.15) .

(iii) Compute torque values τ_1 and τ_2 from the dynamic Equations (3.21) and (3.22).

(iv) These control torques are then applied to the robot. For the simulation studies, ordinary differential equation (ODE) solvers are used to solve the nonlinear dynamic Equations (3.21) and (3.22). The Matlab™ ode45 solver is used to obtain the next time ($t=t_{k+1}$) step's $\mathbf{q}$ and $\dot{\mathbf{q}}$ from the dynamic Equations (3.21) and (3.22). $\mathbf{q}$ and $\dot{\mathbf{q}}$ are initialized to zero as initial conditions for the simulation for t=0.

(v) The new values of $\mathbf{q}$ and $\dot{\mathbf{q}}$ are then used and step (i) is repeated with $t=t_{k+1}$.

The loop is terminated once the desired simulation time has been reached.

The time response simulation uses the Runge-Kutta ODE integrator to compute the state trajectory x(t) by solving for $\dot{x}$.

3.5 Neural computed torque control

The computed torque control method suffers from the disadvantage that an accurate dynamic model of the robot needs to be known to achieve good control performance. However, this is not easily accomplished in practice.

In the neuro-computed torque control approach [Li *et al*, 1995], shown in Figure 3.4, a feedforward neural network is used and trained using the robot’s actual input-output data. This neural network controller essentially replaces that portion of Figure 3.3 enclosed by the dotted box. For the neural computed torque controller, the same algorithm as described in the previous section can be applied. However, in this case, the backpropagation neural network is used to generate the control torques instead of Equations (3.21) and (3.22). Hence the trained neural network is used in place of the model to predict the motor torque values once input, u, is given to it.

The robot’s model is still needed as target values before the algorithm is applied for purposes of training the neural network before the network is being used real-time during the algorithm loops of (i) to (v). The neural network uses scaled inputs and outputs.

For the first neural subnetwork of link 1, torque values of the first link are obtained from the network when the position, velocity and acceleration values of links 1 and 2 are fed as inputs. (Equation [(3.21)]) For the second neural subnetwork of link 2, torque values of the second link are obtained from the network when the position and acceleration values of links 1 and 2 are fed as inputs. Velocity values of link 1 are also fed as inputs for the second subnetwork (Equation [(3.22)]).

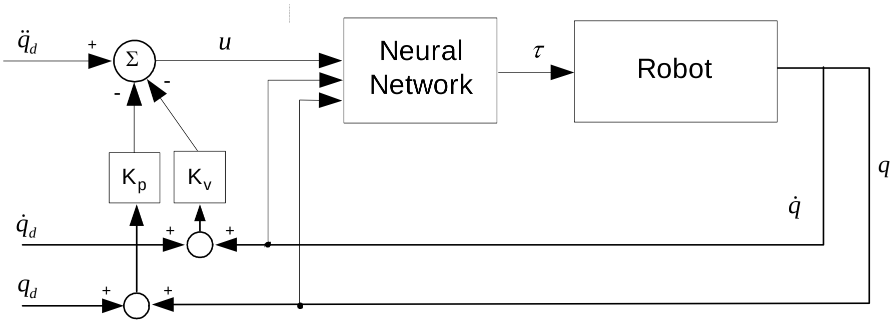

Figure 3.4 Neural computed torque control.

Measured input-output training data from the plant is obtained from experiment as shown in Fig. 3.5. An excitation function generator generates a trajectory as input to the plant. Both the sequence for the input and the output of the plant, τ_k and q_k respectively, are then measured at each sampling instant and these sequences are then used to form training sets as given by Eqn. (3.21) and (3.22). Values of $\dot{q}_k$ and $\ddot{q}_k$ are estimated from values of q_k using the backward difference. Training of the controller is then done with the training data sets obtained. In the work done here, both the inputs and outputs to the neural network are scaled so that their values are in the approximate range of ± 1. This is done for better training and performance of the neural network [Haykins, 1999].

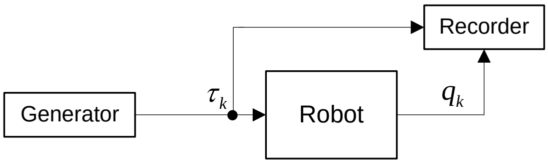

Figure 3.5. Generating input-output training data.

4 Nonlinear Feedback Linearization

4.1Mathematical preliminaries for feedback linearization

Consider the time-invariant, non-affine non-linear continuous-time system written in the state-space form $\dot{x} = f(x) + ug(x)$, with an open subset $x \subseteq \Re^n$, and the control-value space $U = \Re$.

Suppose $\sum$ and $\tilde{\sum}$ are two systems of the above form and suppose that O and $\tilde{\text{O}}$ are open subsets of the state spaces x and $\tilde{x}$ respectively. Then $(\sum, O)$ is feedback equivalent to $(\tilde{\sum}, \tilde{O})$ if there exists a diffeomorphism $T : O \rightarrow \tilde{O}$, or *T(x)*, and smooth maps $\alpha, \beta : O \rightarrow \Re$ where $\beta(x) \neq 0$ for all $x \in O$, such that for each $x \in O$, $T_*(x)(f(x) + \alpha(x)g(x)) = \tilde{f}(T(x))$ and $\beta(x)T_*(x)g(x) = \tilde{g}(T(x))$ where *f,g* and $\tilde{f}, \tilde{g}$ are the vector fields associated with their respective systems [Sontag 1998].

Since T is a diffeomorphism, x and $\tilde{x}$ need to have the same dimension. The previous equalities $T_*(x)(f(x) + \alpha(x)g(x)) = \tilde{f}(T(x))$ and $\beta(x)T_*(x)g(x) = \tilde{g}(T(x))$ may be expressed in the following equivalent form $T_*\big(f(x) + ug(x)\big) = \tilde{f}\big(T(x)\big) + \dfrac{1}{\beta(x)}\big(u - \alpha(x)\big)\tilde{g}\big(T(x)\big) \quad \forall x \in O, \forall u \in R^m$.

A change of variables $(x,u) \mapsto (z,v) := \left(T(x), \dfrac{1}{\beta(x)}[u - \alpha(x)] \right)$ provides a diffeomorphism between $O \times \Re$ and $W \times \Re$ where the inverse is

$$(z,v) \mapsto (x,u) := \left(T^{-1}(z), \alpha\left(T^{-1}(z)\right) + \beta\left(T^{-1}(z)\right)v\right).$$

The solutions of $\dot{x} = f(x) + ug(x)$ are transformed into solutions of $\dot{z} = \tilde{f}(z) + v\tilde{g}(z)$. These solutions correspond to the input $v = \dfrac{1}{\beta(x)}\left[u - \alpha(x)\right]$. $u = k(x,v) = \alpha(x) + \beta(x)v$ may be viewed as a feedback law that closes the loop about the system with v as the new input. Therefore the closed-loop system $\dot{x} = f(x) + k(x,v)g(x)$ transforms into the new system $\dot{z} = \tilde{f}(z) + v\tilde{g}(z)$ under the change of variables *z=T(x).*

4.2 Theoretical development results

The robot model being used here will be the same as the two-link model described in Section 3.1. Subscript 1 denotes the inner link, and subscript 2 denotes the outermost free link.

The joint variables are q_1 and q_2.

$$\mathbf{q} = [q_1 \ \ q_2]^T \tag{4.1}$$

The torques to the robot's actuator motors are τ_1 and τ_2.

$$\boldsymbol{\tau} = [\tau_1 \ \ \tau_2]^T \tag{4.2}$$

The dynamic model of the robot is

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}_m(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_d = \boldsymbol{\tau} \tag{4.3}$$

where

$\mathbf{M}(\mathbf{q})$ = inertia matrix,

$\mathbf{V}_m(\mathbf{q},\dot{\mathbf{q}})$ = Coriolis/centripetal matrix,

$\mathbf{F}(\dot{\mathbf{q}})$ = friction terms,

$\mathbf{G}(\mathbf{q})$ = gravity vector,

$\boldsymbol{\tau}_d$ = disturbances, and

$\boldsymbol{\tau}$ = torque control input.

The robot's arm dynamic equations are obtained from Lagrange's equations (Appendix A.1). They are re-written in matrix-vector form as follows:

$$\begin{aligned}
&\begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos q_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos q_2 & m_2 l_2^2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \\
&+ \begin{bmatrix} -m_2 l_1 l_2 (2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2) \sin q_2 \\ m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 \end{bmatrix} \\
&+ \begin{bmatrix} (m_1+m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1+q_2) \\ m_2 g l_2 \cos(q_1+q_2) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}
\end{aligned} \tag{4.4}$$

Consolidation of the terms gives the robot dynamic equations in a standard form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}\,. \tag{4.5}$$

with

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos q_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos q_2 & m_2 l_2^2 \end{bmatrix} \tag{4.6}$$

$$\mathbf{V(q,\dot{q})} = \begin{bmatrix} -m_2 l_1 l_2 (2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 \\ m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 \end{bmatrix} \tag{4.7}$$

$$\mathbf{G(q,\dot{q})} = \begin{bmatrix} (m_1 + m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1 + q_2) \\ m_2 g l_2 \cos(q_1 + q_2) \end{bmatrix} \tag{4.8}$$

Acceleration is made the subject of the standard dynamic equation as follows

$$\mathbf{M(q)\ddot{q}} + \mathbf{V(q,\dot{q})} + \mathbf{G(q)} = \boldsymbol{\tau}$$

$$\Rightarrow \quad \mathbf{\ddot{q}} = \mathbf{M}^{-1}[\boldsymbol{\tau} - \mathbf{V(q,\dot{q})} - \mathbf{G(q)}] \tag{4.9}$$

The state of the system is defined as

$$\mathbf{x} \equiv [\mathbf{q}^T \quad \mathbf{\dot{q}}^T]^T . \tag{4.10}$$

It is chosen that the state terms be

$$\begin{aligned} x_1 &= q_1 \\ x_2 &= q_2 \\ x_3 &= \dot{q}_1 = \dot{x}_1 \\ x_4 &= \dot{q}_2 = \dot{x}_2 \end{aligned} \quad . \tag{4.11}$$

The robot model system is then expressed in partial state-space form as

$$\mathbf{\dot{x}} = \mathbf{f(x)} + \mathbf{g(x)u} . \tag{4.12}$$

with

LHS:
$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \tag{4.13}$$

RHS:
$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{M}^{-1}(\mathbf{q})[\mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q})] \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}(\mathbf{q}) \end{bmatrix} \boldsymbol{\tau} \tag{4.14}$$

The third and fourth terms of the Equation (4.33) is $\ddot{\mathbf{q}}$. This acceleration has its derivation origins from the earlier Equation (4.29). This is reflected in the second vector row term of Equation (4.34).

The terms of the state-space form are

$$\mathbf{f(x)} = \begin{bmatrix} \dot{\mathbf{q}} \\ -\mathbf{M}^{-1}(\mathbf{q})[\mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q})] \end{bmatrix} \tag{4.15}$$

$$\mathbf{g(x)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}(\mathbf{q}) \end{bmatrix} \tag{4.16}$$

$$u = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{4.17}$$

The feedback controller formula needs to be derived, and it will have the general form

$$\mathbf{u} = \boldsymbol{\alpha}(\mathbf{x}) + \boldsymbol{\beta}(\mathbf{x})\mathbf{v}\,. \tag{4.18}$$

4.3 Results of the derivation of the nonlinear feedback control law

The nonlinear robot system expressed in partial *state space* form [Taware A, et al., 2003] is

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{4.19}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}). \tag{4.20}$$

Some of the mathematical preliminaries of section 4.1 are used in section 4.3.

The output of the system is $\mathbf{y} = \mathbf{Ch(x)}$. Since trajectory tracking is the aim here, the output of the system is fed as input to the system. Therefore **C=I**, the identity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. When **C**=**I**, **y**=**Ch**(**x**) gives **y**=**h**(**x**). Hence **y=h(x)**.

For the present system, $\dot{\mathbf{x}}$ and $\mathbf{f}(\mathbf{x})$ are 4×1 vectors. $\mathbf{g}(\mathbf{x})$ is a 4×2 matrix, and **u** is a 2×1 vector. **h(x)** is a 2×1 vector, where

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{4.21}$$

Assume that the system has a starting state $\mathbf{x}(t_0)$ at time t_0. The output $\mathbf{y}(t)$ and its derivatives $\mathbf{y}^{(k)}(t)$ needs to be calculated.

$$\begin{aligned} \mathbf{y}(t_0) &= \mathbf{h}(\mathbf{x}(t_0)) \\ &= \mathbf{h}(\mathbf{x}_0) \end{aligned} \tag{4.22}$$

Differentiating once with respect to time gives

$$\begin{aligned}\mathbf{y}^{(1)}(t) &= \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\frac{d\mathbf{x}}{dt} \\ &= \frac{\partial \mathbf{h}}{\partial \mathbf{x}}[(\mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))\mathbf{u}(t)] \\ &= \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\mathbf{f}(\mathbf{x}(t)) + \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{g}(\mathbf{x}(t))\mathbf{u}(t) \\ &= L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t)) + L_{\mathbf{g}}\mathbf{h}(\mathbf{x}(t))\mathbf{u}(t)\end{aligned} \quad . \tag{4.23}$$

The mathematical notation used here is the Lie derivative of **h(x)** along **f(x)** being defined as $L_{\mathbf{f}}\mathbf{h}(\mathbf{x}) = \left[\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right]^T . \mathbf{f}(\mathbf{x})$.

For the present robot model, it has been calculated that

$$L_{\mathbf{g}}\mathbf{h}(\mathbf{x}(t)) = 0 \quad . \tag{4.24}$$

The first derivative term is then reduced to

$$\therefore \mathbf{y}^{(1)}(t) = L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t)) \quad . \tag{4.25}$$

The first derivative in Equation (4.25) is further differentiated to give

$$\begin{aligned}\mathbf{y}^{(2)}(t) &= \frac{\partial L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}.\frac{d\mathbf{x}}{dt} \\ &= \frac{\partial L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}.[\mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))\mathbf{u}(t)] \\ &= \frac{\partial L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}.\mathbf{f}(\mathbf{x}(t)) + \frac{\partial L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))}{\partial \mathbf{x}}.\mathbf{g}(\mathbf{x}(t))\mathbf{u}(t) \\ &= L_{\mathbf{f}}(L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t)) + L_{\mathbf{g}}(L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t)))\mathbf{u}(t) \\ &= L_{\mathbf{f}}^2\mathbf{h}(\mathbf{x}(t)) + L_{\mathbf{g}}L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))\mathbf{u}(t)\end{aligned} \qquad (4.26)$$

The repeated Lie derivative - first along **f(x)** and then along **g(x)** - is defined

as $L_g L_f h(\mathrm{x}) = \frac{\partial(L_f h(\mathrm{x}))}{\partial \mathrm{x}} \mathrm{g}(\mathrm{x})$.

The second derivative equation $\mathbf{y}^{(2)}(t) = L_{\mathbf{f}}^2\mathbf{h}(\mathbf{x}(t)) + L_{\mathbf{g}}L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))\mathbf{u}(t)$ is rearranged such that **u(*t*)** is the subject of the equation.

$$\mathbf{u}(t) = \frac{\mathbf{y}^{(2)}(t) - L_{\mathbf{f}}^2\mathbf{h}(\mathbf{x}(t))}{L_{\mathbf{g}}L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))} \qquad (4.27)$$

This controller yields the linear system

$$\ddot{\mathbf{y}} = \mathbf{v}\,. \qquad (4.28)$$

The linearized feedback control law, ***u(t)***, for the nonlinear robot system is thus derived.

The block diagram illustrating the approach is shown in Figure 4.1. Here, state feedback transforms the nonlinear robot system into a linear and controllable system.

The state $\boldsymbol{x}$ is fed back to generate control $\boldsymbol{u}$ such that there is a linear relationship between $\boldsymbol{y}$ and $\boldsymbol{v}$ as represented by Equation 4.28.

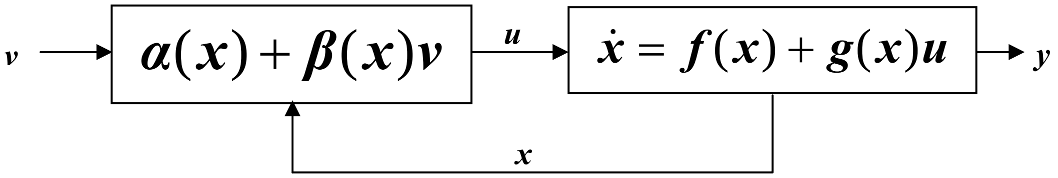
$\boldsymbol{v}$
$\boldsymbol{\alpha}(\boldsymbol{x})+\boldsymbol{\beta}(\boldsymbol{x})\boldsymbol{v}$
$\boldsymbol{u}$
$\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x})+\boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}$
$\boldsymbol{y}$
$\boldsymbol{x}$

Figure 4.1 State feedback

Substitution of (4.28) into (4.27) gives

$$\boldsymbol{u}(t)=\frac{\boldsymbol{v}-L_f^2\boldsymbol{h}(\boldsymbol{x}(t))}{L_g L_f\boldsymbol{h}(\boldsymbol{x}(t))} \quad . \tag{4.29}$$

This feedback law has the initial desired general form of Equation (4.30).

$$\boldsymbol{u}(t)=\boldsymbol{\alpha}(\boldsymbol{x})+\boldsymbol{\beta}(\boldsymbol{x})\boldsymbol{v}(t) \tag{4.30}$$

With reference to the general form of the control law, the corresponding analogous terms are as follows.

$$v(t)\equiv\mathbf{y}^{(2)}(t) \qquad \text{(Comparing (4.27) and 4.29))} \tag{4.31}$$

$$\beta(\mathbf{x})\equiv\frac{1}{L_{\mathbf{g}}L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))} \tag{4.32}$$

$$\alpha(\mathbf{x})\equiv\frac{L_{\mathbf{f}}^2\mathbf{h}(\mathbf{x}(t))}{L_{\mathbf{g}}L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))} \tag{4.33}$$

$\beta(\mathbf{x})$ is assumed to be nonzero for all **x**.

In general, if γ is the smallest integer such that $L_{\mathbf{g}}L_{\mathbf{f}}^{i}\mathbf{h} \equiv 0$ for i =0,…, $\gamma - 2$ and $L_{\mathbf{g}}L_{\mathbf{f}}^{\gamma-1}\mathbf{h} \neq 0\, \forall x \in \Re^{n}$, the control law

$$u = \frac{1}{L_{g}L_{f}^{\gamma-1}h}(-L_{f}^{\gamma}h + v) \tag{4.34}$$

gives $y^{(\gamma)} = v$. (4.35)

The *k*-multiple Lie derivative of **h(x)** along **f(x)** is defined as the recursive relationship $L_{f}^{k}h(\mathbf{x}) = \left[\frac{\partial}{\partial \mathbf{x}}L_{f}^{k-1}h(\mathbf{x})\right].\mathbf{f}(\mathbf{x})$ (4.36)

with $L_{f}^{o}h(\mathbf{x}) = h(\mathbf{x})$. (Appendix A.2) (4.37)

The derived control law is fed into the robot system equation.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{4.38}$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\frac{\mathbf{y}^{(2)}(t) - L_{\mathbf{f}}^{2}\mathbf{h}(\mathbf{x}(t))}{L_{\mathbf{g}}L_{\mathbf{f}}\mathbf{h}(\mathbf{x}(t))} \tag{4.39}$$

The calculated value of control law, **u**, is allocated to the torque, τ, of the robot.

The error of the trajectory tracking is defined as

$$\mathbf{e}(t) = \mathbf{r}(t) - \mathbf{x}(t), \tag{4.40}$$

where **r***(t)* is the input to the system, and **y**(*t*) = **x***(t)* is the output of the system. It is the difference between the real output, **y***(t)*, and the reference output, **r***(t)*.

Differentiating the error once gives

$$\dot{\mathbf{e}}(t) = \dot{\mathbf{r}}(t) - \dot{\mathbf{x}}(t) \qquad (4.41)$$

Input v was chosen as a PD controller The external reference input *v* is chosen to contain proportional integral control tuning parameters.

$$\mathbf{v}(t) = k_D \dot{\mathbf{e}}(t) + k_p \mathbf{e}(t) \qquad (4.42)$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \frac{\mathbf{v}(t) - L_{\mathbf{f}}^2 \mathbf{h}(\mathbf{x}(t))}{L_{\mathbf{g}} L_{\mathbf{f}} \mathbf{h}(\mathbf{x}(t))} \qquad (4.43)$$

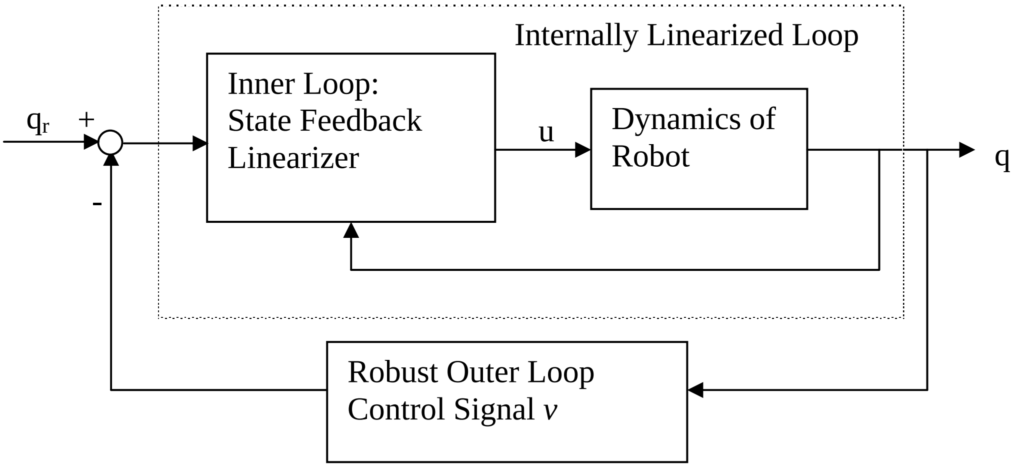
Internally Linearized Loop
Inner Loop: State Feedback Linearizer
qr
+
-
u
Dynamics of Robot
q
Robust Outer Loop Control Signal *v*

Figure 4.2 The general feedback linearization scheme.

Figure 4.2 illustrates the control of the robot using state feedback linearization. The inner state feedback loop is used to linearize the non-linear dynamics of the robot so the well-known linear control principles can be used with the outer feedback loop.

Tracking control here aims to produce an output that converges to the prescribed reference function. The reference profile used for simulation in this work is the quintic polynomial profile.

The robot dynamic model equations are rearranged such that acceleration is made the subject of the Equations (4.54) and (4.55). This is to allow the ordinary differential equation (ode) computer routines to integrate on the acceleration term so that velocity values can be obtained. The ode routines are based on the Runge-Kutta formulations. Two state vectors are passed to the ode routines- the position vector and the velocity vector. Other than integrating the acceleration term of Equations (4.54) and (4.55), the ode function also integrates the velocity term in the state vector that is being fed in so that future position values can be obtained. The future values of the velocity and the position vectors obtained after integration are collected and then fed back into the controller to generate future values of the control torques.

$$\begin{aligned}\tau_1 = {} & [(m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2]\ddot{q}_1 + [m_2 l_2^2 + m_2 l_1 l_2 \cos q_2]\ddot{q}_2 \\ & - m_2 l_1 l_2 (2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 + (m_1+m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1+q_2)\end{aligned} \tag{4.44}$$

$$\tau_2 = [m_2 l_2^2 + m_2 l_1 l_2 \cos q_2]\ddot{q}_1 + m_2 l_2^2 \ddot{q}_2 + m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 + m_2 g l_2 \cos(q_1+q_2) \tag{4.45}$$

$$A1 = [(m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2] \tag{4.46}$$

$$B1 = [m_2 l_2^2 + m_2 l_1 l_2 \cos q_2] \tag{4.47}$$

$$C1 = -m_2 l_1 l_2 (2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 + (m_1+m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1+q_2) \tag{4.48}$$

$$A2 = [m_2 l_2^2 + m_2 l_1 l_2 \cos q_2] \tag{4.49}$$

$$B2 = m_2 l_2^2 \ddot{q}_2 + m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 \tag{4.50}$$

$$C2 = m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 + m_2 g l_2 \cos(q_1 + q_2) \tag{4.51}$$

$$\tau_1 = A1\,\ddot{q}_1 + B1\,\ddot{q}_1 + C1 \tag{4.52}$$

$$\tau_2 = A2\,\ddot{q}_2 + B2\,\ddot{q}_2 + C2 \tag{4.53}$$

Acceleration is made the subject of the equation.

$$\ddot{q}_1 = \frac{-(-\tau_2 \times B1 + B2 \times \tau_1 - B2 \times C1 + C2 \times B1)}{-A1 \times B2 + B1 * A2} \tag{4.54}$$

$$\ddot{q}_2 = \frac{\tau_1 \times A2 - A1 \times \tau_2 + A1 \times C2 - C1 \times A2}{-A1 \times B2 + B1 \times A2} \tag{4.55}$$

<u>Summarized Algorithm (with reference to Figure 4.3):</u>

For t=0 to 2 sec

- *e(t)=r(t)-y(t)*
- $\dot{e}(t) = \dot{r}(t) - \dot{x}(t)$
- $v(t) = k_D \dot{e}(t) + k_p e(t)$
- *u(t)=α(x)+β(x)v(t) (α,β from symbolic formulas)*
- *$\dot{x}$ (t+1)=f(x)+g(x)u(t) (f, g from symbolic formulas)*

  *(*assume that *u(t)* does not change between *t* and *t+1)*
- *assign u(t) to be the control torque value*

- *x(t) contains q and* $\dot{q}$ values. *Feed x(t) into function ode45 to get x(t+1)* . After evaluation of x(t+1), assign y(t+1)=x(t+1) since from y=h(x))
- *Update the state value x(t) for the next loop*

End

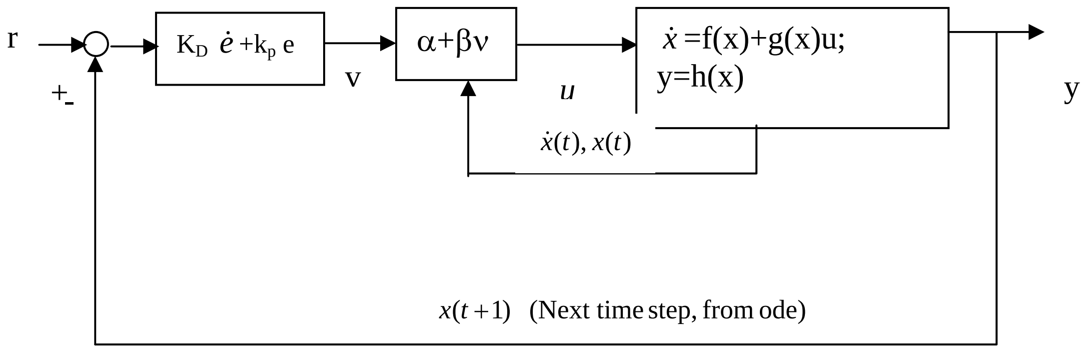

Figure 4.3 PD control with state feedback linearization

4.4 Controller results

For the present two link robot system, the following formulations are obtained for numerical simulation.

$$L_g h = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.56}$$

$$L_f h = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{4.57}$$

$$L_g L_f h = \begin{bmatrix} \dfrac{1}{l_1^2(-m_1 - m_2 + m_2 \cos^2 q_2)} & \dfrac{(l_2 + l_1 \cos q_2)}{l_2 l_1^2(-m_1 - m_2 + m_2 \cos^2 q_2)} \\ \dfrac{-l_2 + l_1 \cos q_2}{l_2 l_1^2(-m_1 - m_2 + m_2 \cos^2 q_2)} & \dfrac{l_1^2 m_1 + l_1^2 m_2 + m_2 l_2^2 + 2 m_2 l_1 l_2 \cos q_2}{m_2 l_2^2 l_1^2(-m_1 - m_2 + m_2 \cos^2 q_2)} \end{bmatrix} \tag{4.58}$$

$L_f^2 h(1) =$

$$\frac{-m_2 l_1 l_2 (2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2)\sin q_2 + (m_1 + m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1 + q_2)}{l_1^2(-m_1 - m_2 + m_2 \cos^2 q_2)}$$

$$-\frac{(l_2 + l_1 \cos q_2)(m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 + m_2 g l_2 \cos(q_1 + q_2)}{l_2 l_1^2 (-m_1 - m_2 + m_2 \cos^2 q_2)} \tag{4.59}$$

$L_f^2 h(2) =$

$$\frac{-l_2 + l_1 \cos q_2}{l_2 l_1^2 (-m_1 - m_2 + m_2 \cos^2 q_2)} \times \Big(\Big(-m_2 l_1 l_2 (2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2 \Big)\sin q_2 + (m_1 + m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1 + q_2))\Big)$$

$$+\frac{l_1^2 m_1 + l_1^2 m_2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2}{m_2 l_2^2 l_1^2 (-m_1 - m_2 + m_2 \cos^2 q_2)} \times \Big(m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 + m_2 g l_2 \cos(q_1 + q_2 \Big) \tag{4.60}$$

$$L_f^2 h = \begin{bmatrix} L_f^2 h(1) \\ L_f^2 h(2) \end{bmatrix} \tag{4.61}$$

$$\begin{aligned} u(1) &= (-l_1^2m_1-l_1^2m_2-m_2l_2^2-2m_2l_1l_2\cos q_2)\begin{pmatrix} v_1-\dfrac{-m_2l_1l_2(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)\sin q_2+(m_1+m_2)gl_1\cos q_1+m_2gl_2\cos(q_1+q_2)}{l_1^2(-m_1-m_2+m_2\cos^2 q_2)} \\ +\dfrac{(l_2+l_1\cos q_2)(m_2l_1l_2\dot{q}_1^2\sin q_2+m_2gl_2\cos(q_1+q_2)}{l_2l_1^2(-m_1-m_2+m_2\cos^2 q_2)} \end{pmatrix} \\ & -l_2(l_2+l_1\cos q_2)m_2\begin{bmatrix} v_2+\dfrac{(l_2+l_1\cos q_2)-m_2l_1l_2(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)\sin q_2+(m_1+m_2)gl_1\cos q_1+m_2gl_2\cos(q_1+q_2)}{l_2l_1^2(-m_1-m_2+m_2\cos^2 q_2)}- \\ \dfrac{(l_1^2m_1+l_1^2m_2+m_2l_2^2+2m_2l_1l_2\cos q_2)(m_2l_1l_2\dot{q}_1^2\sin q_2+m_2gl_2\cos(q_1+q_2)}{m_2l_2^2l_1^2(-m_1-m_2+m_2\cos^2 q_2)} \end{bmatrix} \end{aligned} \tag{4.62}$$

$$\begin{aligned} u(2) &= -l_2(l_2+l_1\cos q_2)m_2\ (\ v_1-\frac{-m_2l_1l_2(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)\sin q_2+(m_1+m_2)gl_1\cos q_1+m_2gl_2\cos(q_1+q_2)}{l_1^2(-m_1-m_2+m_2\cos^2 q_2)} \\ &\quad +\frac{l_2+l_1\cos q_2\Bigl(m_2l_1l_2\dot{q}_1^2\sin q_2+m_2gl_2\cos(q_1+q_2)\Bigr)}{l_2l_1^2(-m_1-m_2+m_2\cos^2 q_2)} \\ &\quad -m_2l_2^2(v_2+\frac{(l_2+l_1\cos q_2)\Bigl(m_2l_1l_2(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)\sin q_2+(m_1+m_2)gl_1\cos q_1+m_2gl_2\cos(q_1+q_2)\Bigr)}{l_2l_1^2(-m_1-m_2+m_2\cos^2 q_2)} \\ &\quad -\frac{(l_1^2m_1+l_1^2m_2+m_2l_2^2+2m_2l_1l_2\cos q_2)(m_2l_1l_2\dot{q}_1^2\sin q_2+m_2gl_2\cos(q_1+q_2)}{m_2l_2^2l_1^2(-m_1-m_2\cos^2 q_2)}\quad) \end{aligned} \tag{4.63}$$

$$\boldsymbol{u}=\begin{bmatrix} u(1) \\ u(2) \end{bmatrix} \tag{4.64}$$

The novel *controller u,* is thus obtained for feedback linearization of the robot system.

4.5 Neuro-Feedback Linearisation

Simulation studies on the same robot model are carried out using the neuro-feedback linearization controller. The neural network requires data for training. To get the data points for training, a sine wave is used as the reference trajectory input. This sine curve has some random noise introduced into it. The input is then fed into a PD controller so as to obtain the torque for the robot. This torque value is then fed into the robot plant. The next time step value of the robot's position and velocity are obtained from the plant's ode function. This updated position and velocity value is fed back to the beginning of the loop to be compared with the reference input's value. The error obtained due to the slight discrepancy between the reference and feedback position and velocity values are needed for calculation of the torque from the PD controller. Ten thousand data points for each attribute are saved. Six attributes are saved for the neural network. They are the position, velocity and torque values of both links. During the generation of data points, the output of the PD controller *v*, is taken to be equal to the torque value.

During neural network training, the position, velocity and the external reference input *v* are the inputs to the neural network. The torque output values are provided for as teaching values for the training of the neural network.

During simulation, the controller u of (Equations 4.62 to 4.64) is not used. In its place, the trained neural network is used in each loop of the simulation to give the output torque value when the input position, velocity and reference input values are fed into the network. Hence the neural network serves as a nonlinear function that maps the input signal vector consisting of $\boldsymbol{v}$, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, to the output torque signal vector τ.

Chapter 5 Discussion of Simulation Results

5.1 Computed torque and neuro computed torque control

Simulations studies were performed for the computed torque and the neuro computed torque control schemes discussed in Chapter 3. The controlled plant is the two-link robotic manipulator as described in Section 3.1. During the computed torque and neuro computed torque control simulation, the two links of the robot model were controlled to follow a quintic polynomial trajectory for 2 seconds. The task trajectory is chosen as

$$q_d(t) = (10\frac{q_f}{t_f^3} - 15\frac{q_f}{t_f^4}t + 6\frac{q_f}{t_f^5}t^2)t^3 \tag{5.1}$$

where t_f = total move time, q_d=desired position, q_f=final position and t= time.

Before a robot link can be controlled, the desired path for performing a task needs to be known. Such paths are user defined. In the simulations here, the path is chosen to be that of the commonly used quintic polynomial. Other paths such as the cubic polynomial, sine and cosine curves may also be used. For the motion control problem here, the ultimate control objective is to ensure that the robot moves along a prescribed desired trajectory.

The developed robot controllers are simulated on a computer to verify the effectiveness of the proposed control schemes. Computer simulations help to verify the viability of the controller design. The computer-controlled system is assumed to behave as a continuous time system if the sampling period is sufficiently small. In the simulations done here, a sampling time of 1 s is used throughout. During simulation, the links are made to follow the desired quintic trajectory in 2 seconds over a distance of 1rad. Two trajectory tracking control methods, the computed torque control method (CTC) and the neuro computed torque control method (NCTC), were applied on a robot model. The two method's simulation performance are compared and discussed.

The following parameters for the two-link manipulator were used in the simulation:
m_1=2kg, m_2=3kg, l_1=1m, l_2=1.5m and g=9.81m/s^2.

From Figure 5.1, for the first link, the CTC method gives errors which are smaller than that of the NCTC method. The CTC errors hover closely around zero. The maximum error of NCTC is a small value of 4.5×10^{-3}rad.

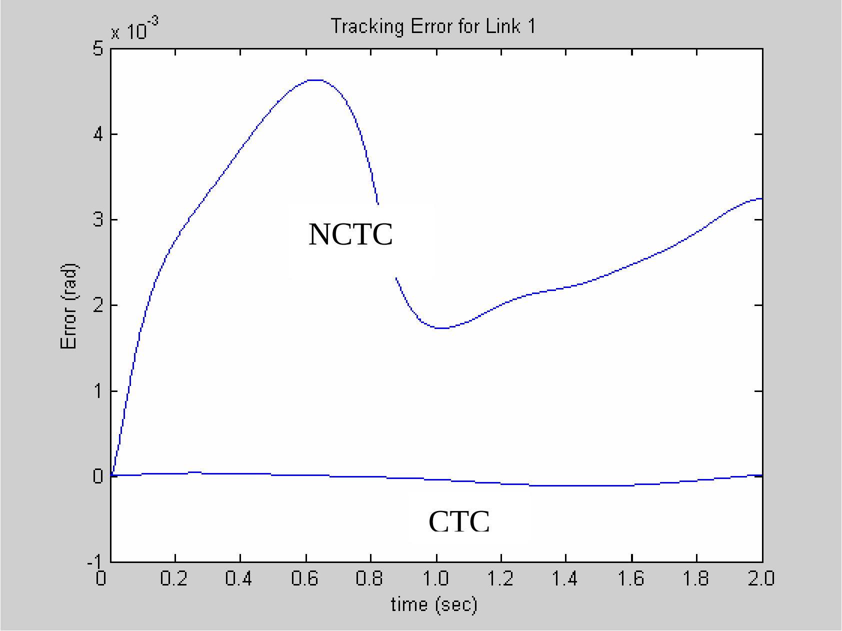

Figure 5.1 Time history of position error of link 1 with neural CTC and CTC scheme.

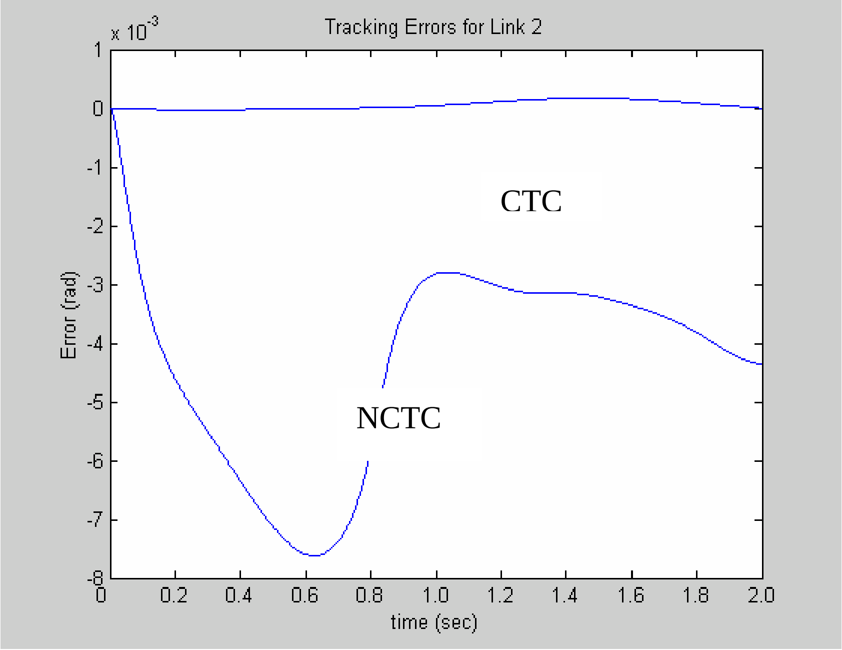

Figure 5.2 Time history of position error of link 2 with neural CTC and CTC scheme.

From Figure 5.2, link two’s performance is similar to that of link one’s performance. From the plot, it is seen that the CTC method gives smaller errors than errors of the NCTC method. The maximum error of NCTC is a small value of 4.5×10^{-3}rad.

In general, tracking error stayed in the range of 10^{-3} rad for the NCTC method, whilst tracking error stayed in the range of 10^{-6} rad for the CTC method. Theoretically, for a continuous system, the CTC should give perfect tracking control, meaning that tracking should always stay at zero. In the simulation, the small tracking errors, on the order of 10^{-5}, are due to two main factors, round-off errors in computation in the digital computer and the introduction of sampling (a sampling period of 0.001 s was used) with a zero-order hold for the control torques. The NCTC method also performed very well with maximum tracking errors on the order of 10^{-3}. This showed that the neural network can accurately map the dynamic model of the robot through training.

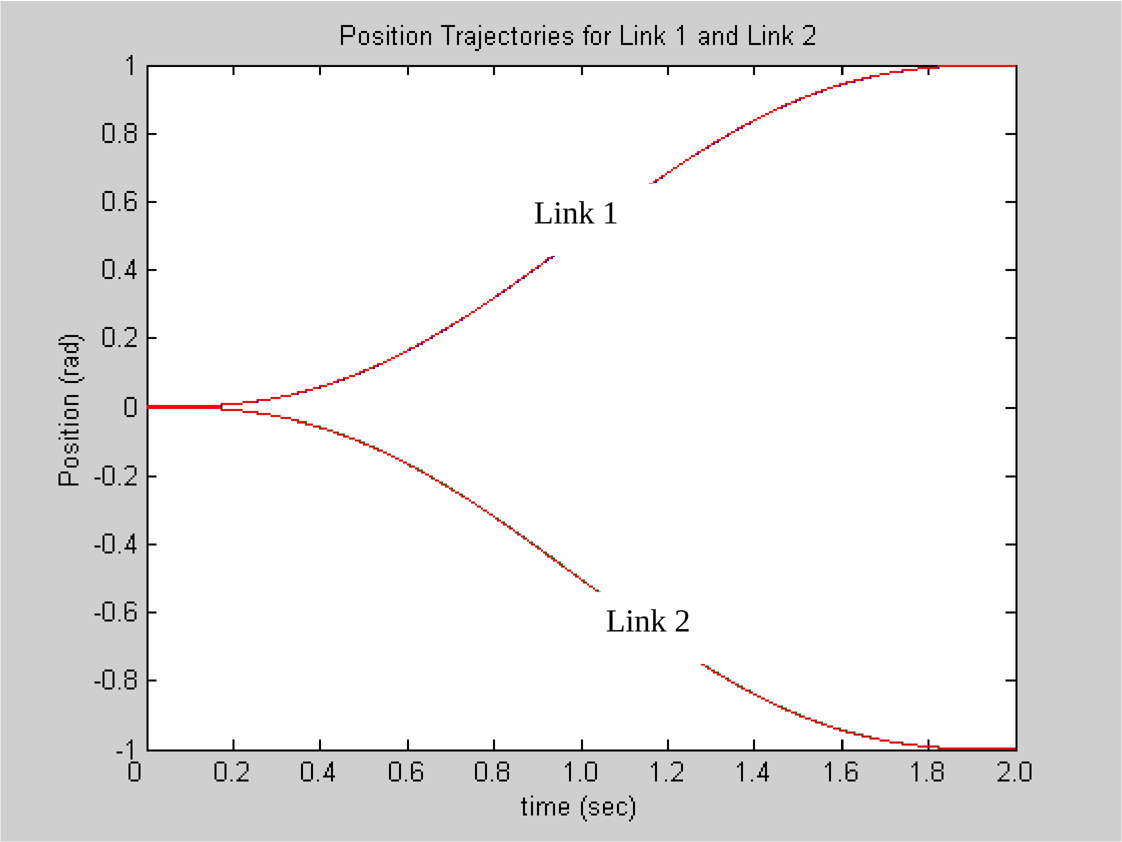

Figure 5.3 Time history of position of links 1,2 with neural-CTC and CTC scheme.

Figure 5.3 shows the quintic polynomial trajectory profiles of both Link 1 and Link 2. The actual paths and desired paths closely coincide with one another. The same curves are obtained for both neural CTC and CTC schemes as the tracking errors in both cases are very small. In both control schemes, it is noted that the desired and actual trajectories for both links coincides with each other.

The change in trajectory profiles due to changes in mass is carried out. A weight increase of the robot link can indicate a pickup of load by the arm, and a weight decrease can also indicate a release of load by the arm. For the simulation here, link one has an increase of mass of 0.5kg (from 2 to 2.5), and link two has a decrease of mass of 2kg (from 3 to 1). From figure 5.4, the trajectory profile of the system under computed torque controller shows that there was some deviation from the profile to be tracked. The deviation seemed to be more obvious during the initial pick or drop event. The deviation of the first link is smaller since it only gained 0.5kg. The deviation of the first link is more marked as it has lost 2kg. Deviation profiles can be seen when figure 5.4 was compared with figure 5.3.

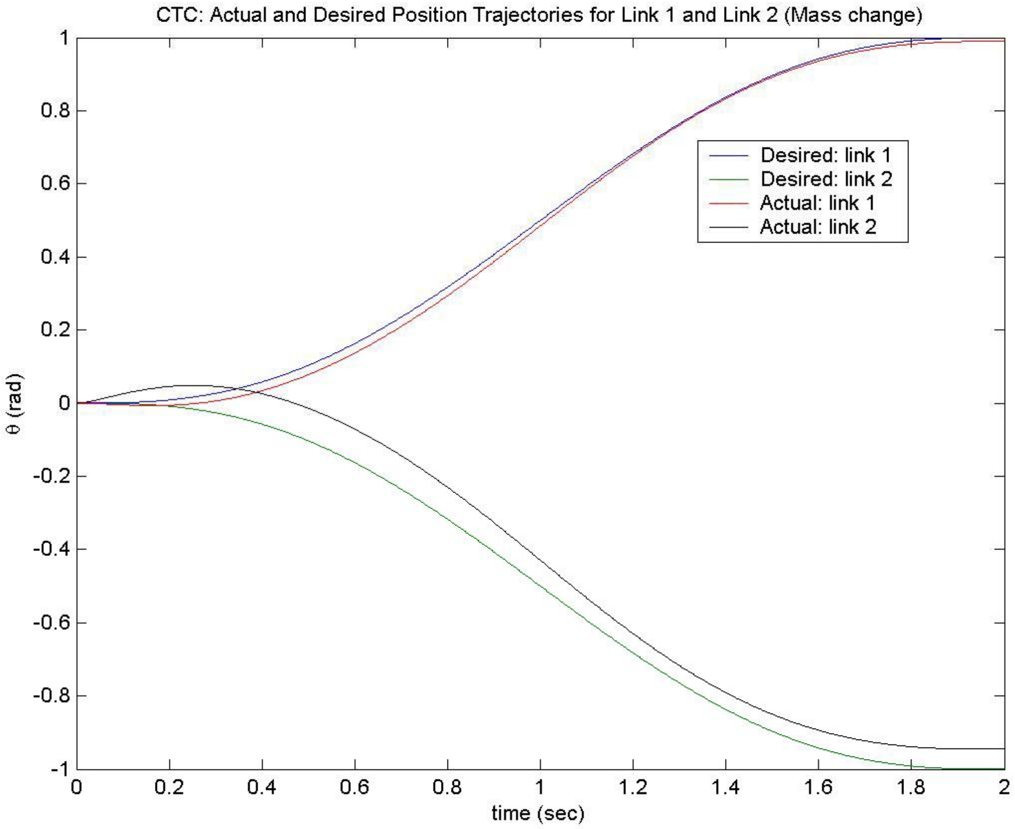

Figure 5.4 Time history of both link under CTC scheme with link mass change

From figure 5.5, it is seen that error of the first link reached a peak of 0.027 at 0.6 seconds. The error level then goes down smoothly to a low value of 0.005. It then settles at a constant value of 0.01 by 2 seconds.

For the second link, the error reached a peak of 0.09 at 0.6 seconds, as seen from figure 5.6. The error level settles at a constant value of 0.065 by the end of two seconds.

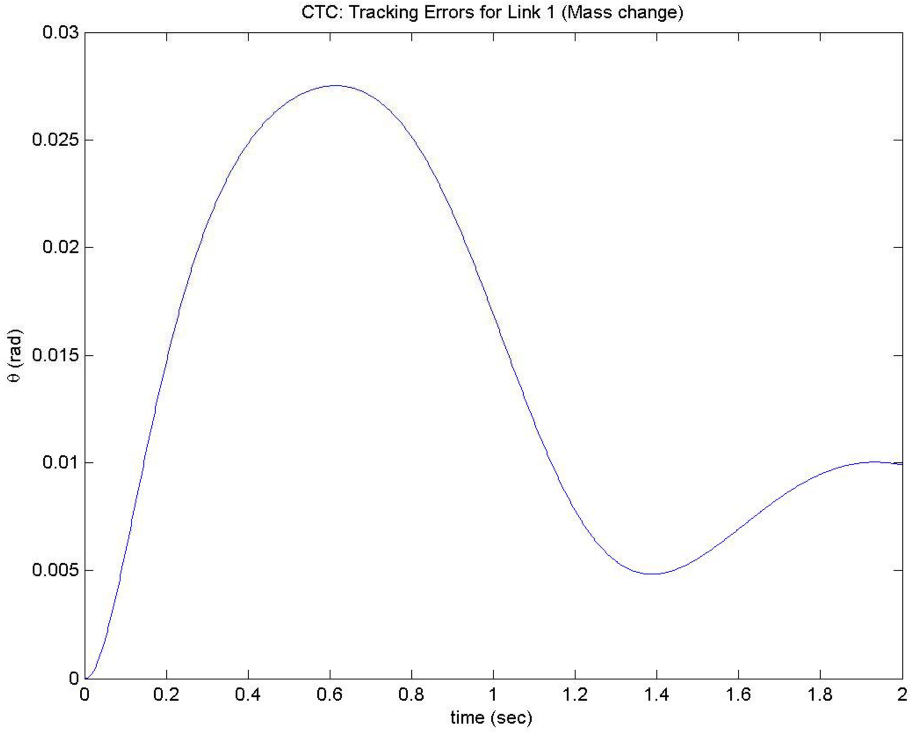

Figure 5.5 Time history of position error of link 1 under CTC scheme with mass change

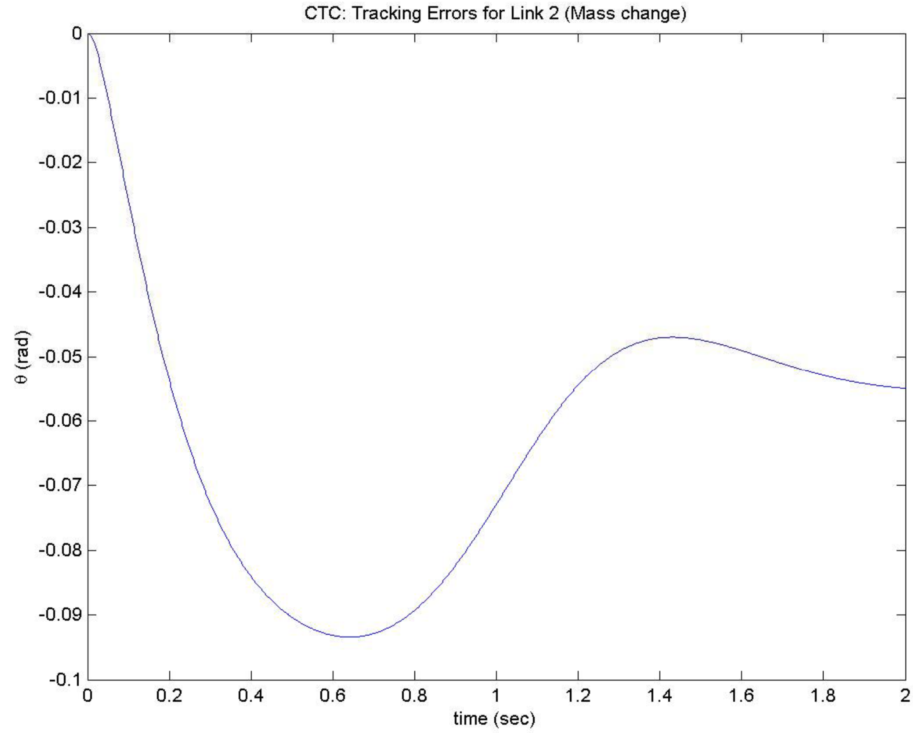

Figure 5.6 Time history of position error of link 2 under CTC scheme with mass change

From figure 5.7, the trajectory profile of the system is seen after training on new mass change data under the neural computed torque controller. It shows that there is some slight deviation from the profile to be tracked. The deviation also seems to be more obvious during the initial drop event for the second link. For the first link, the path is still well tracked.

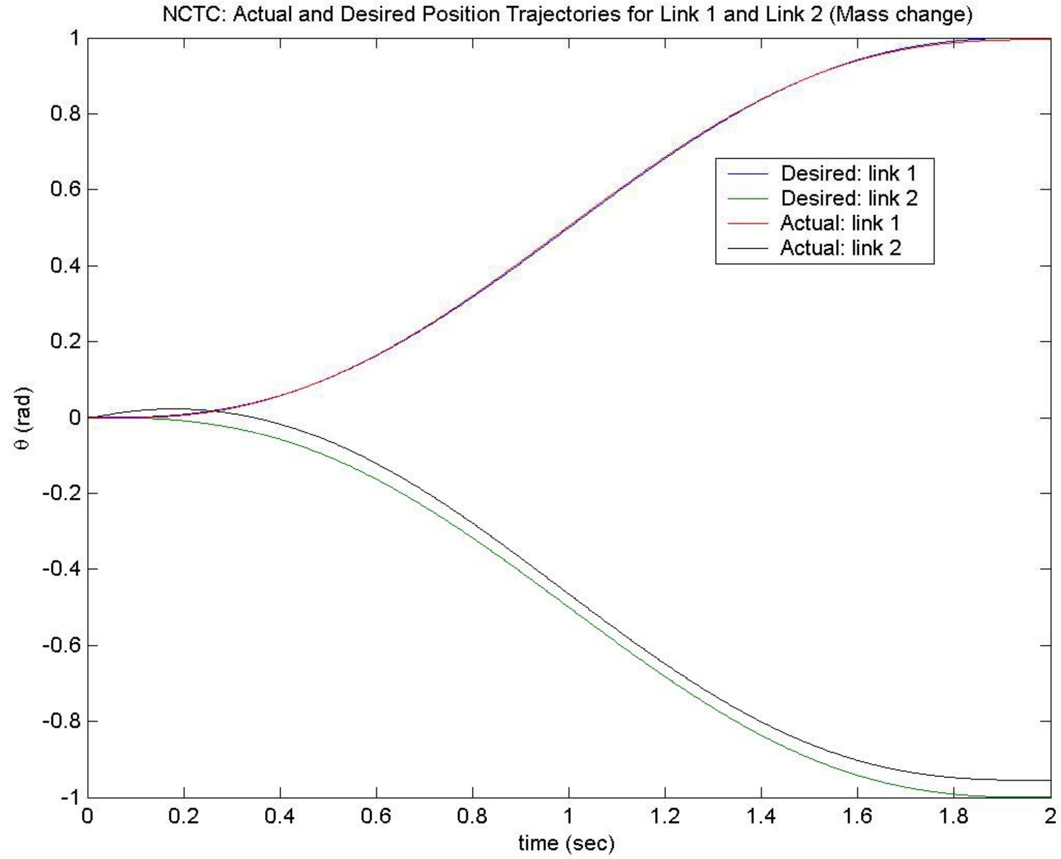

Figure 5.7 Time history of both link under NCTC scheme with link mass change

From figure 5.8, it is seen that error of the first link has a maximum of 0.005 at the end of two seconds. For the second link in figure 5.9, the error level settles at a constant value of 0.045 by the end of two seconds. After the neural network is trained on the new mass change data, the maximum error values of the NCTC scheme are less than the maximum error values of the CTC.

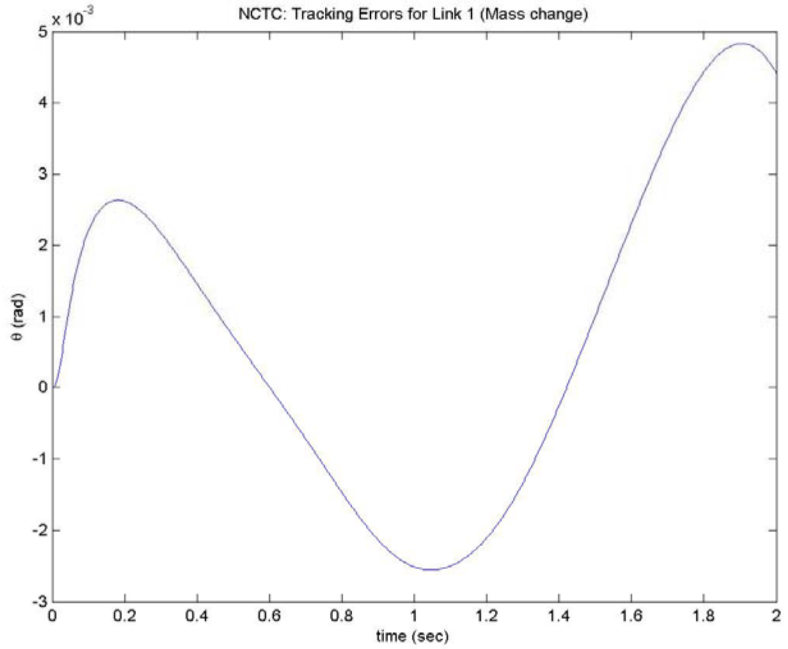

Figure 5.8 Time history of position error of link 1 under NCTC scheme with mass change

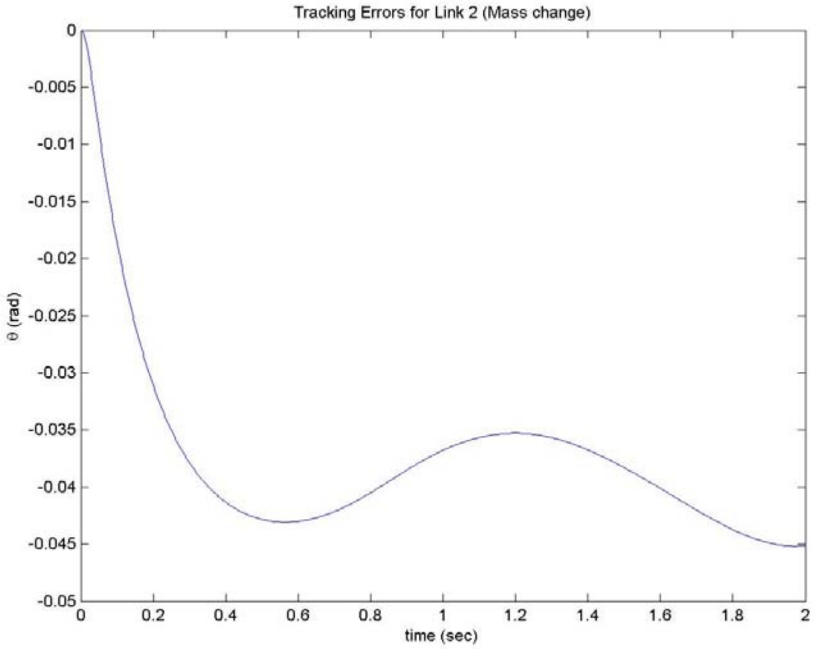

Figure 5.9 Time history of position error of link 2 under NCTC scheme with mass change

5.2 Simulation results of the designed feedback linearized law.

Simulation studies are carried out using the designed feedback linearized law of (4.81-82) on the two-link manipulator. When the feedback linearized law was applied to the robot model, the position error profile obtained is seen in Figure 5.10. From figure 5.10, the error of both links is acceptably low, the maximum error being at most 0.015 rad. The error is cyclic in nature.

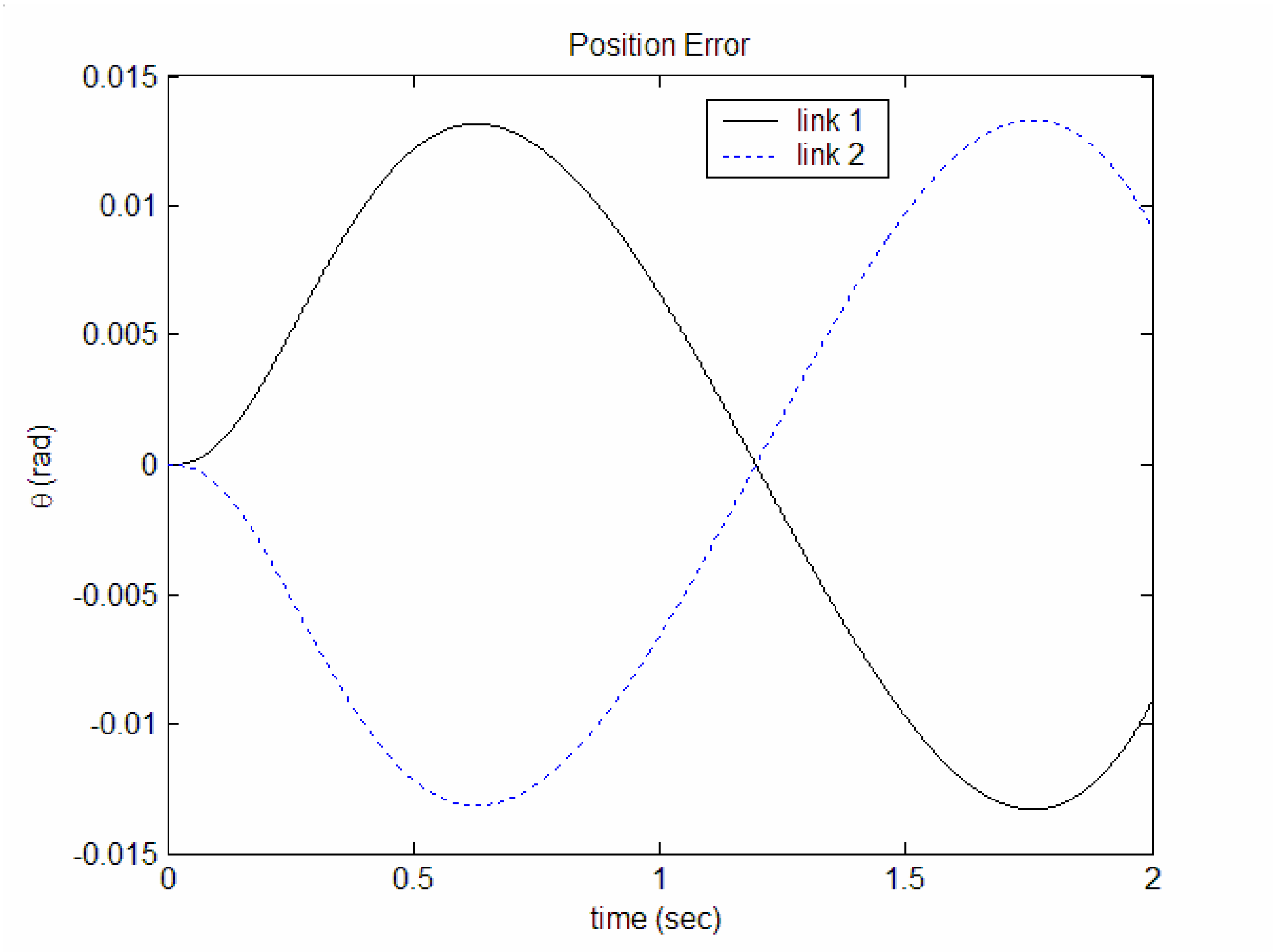

Figure 5.10 Time history of position error of link 1 and 2 with feedback linearized law.

For the first link, the actual path of the robot follows closely to that of the reference trajectory path. At about 1.25 seconds, the actual path is seen to coincide with that of the reference trajectory's path.

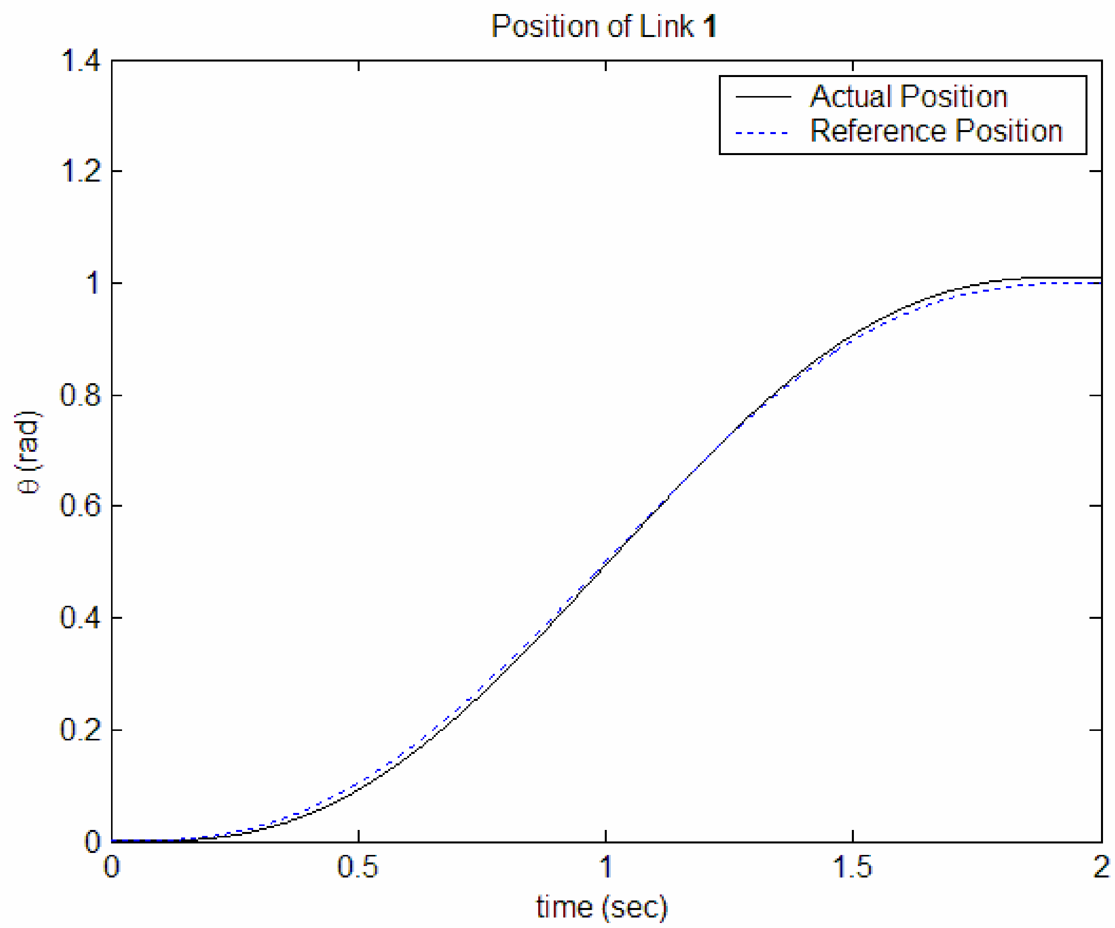

Figure 5.11 Time history of link one's position with feedback linearized law.

For the second link, the actual path of the robot also follows closely to that of the reference trajectory path. At about 1.20 seconds, the actual path is seen to coincide with that of the reference trajectory's path.

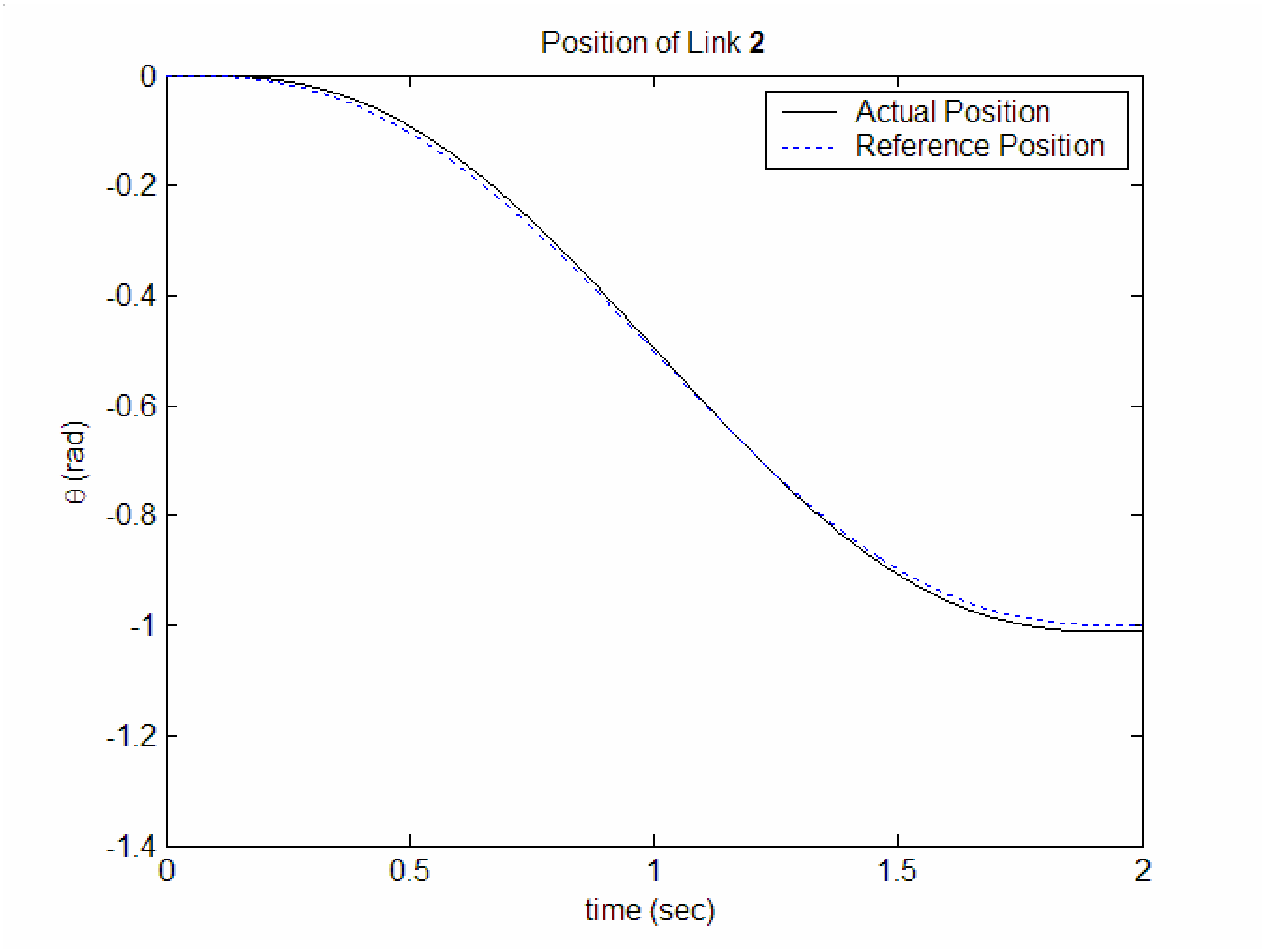

Figure 5.12 Time history of link two's position with feedback linearized law.

5.3 Neuro-Feedback linearisation

From figure 5.15, the peak value of error is 0.008 for the first link, and 0.027 for the second link. Link 1's error is more stabilized, and it lies between the range of -0.005 and -0.008.

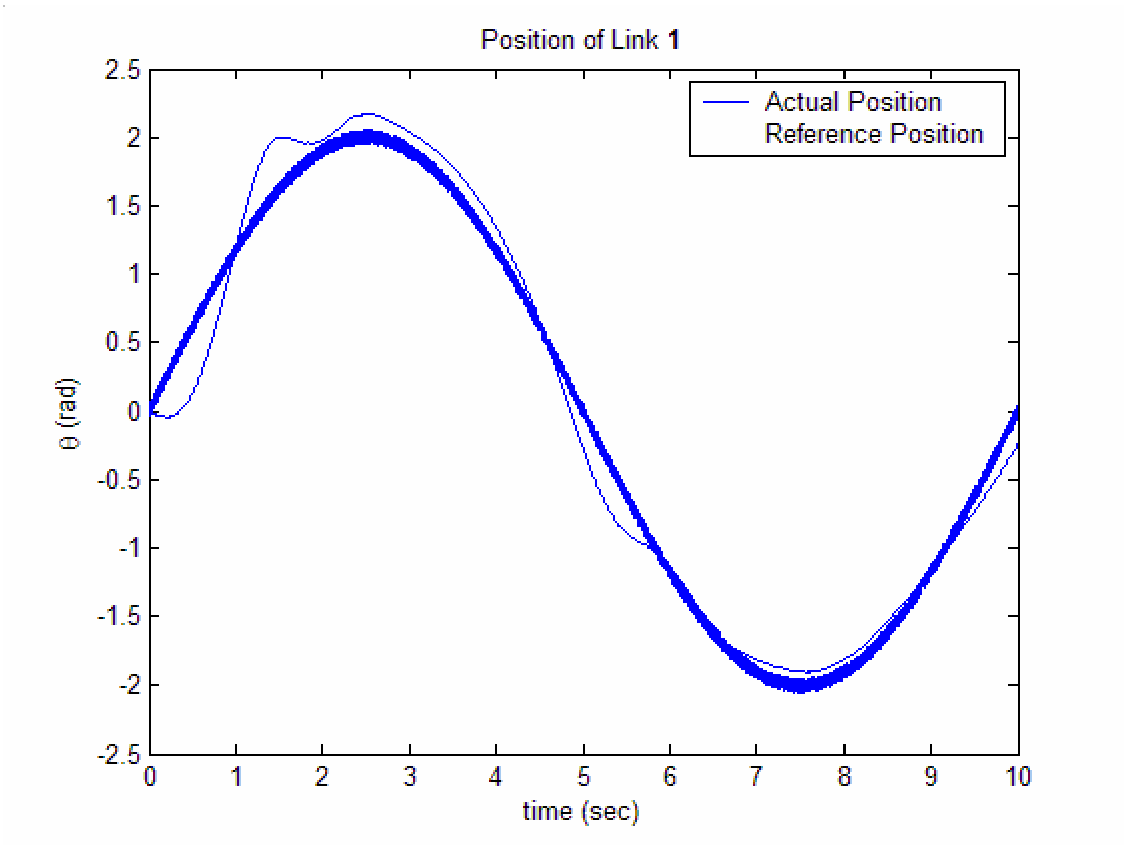

Figure 5.13 Link 1's reference points for neural network training.

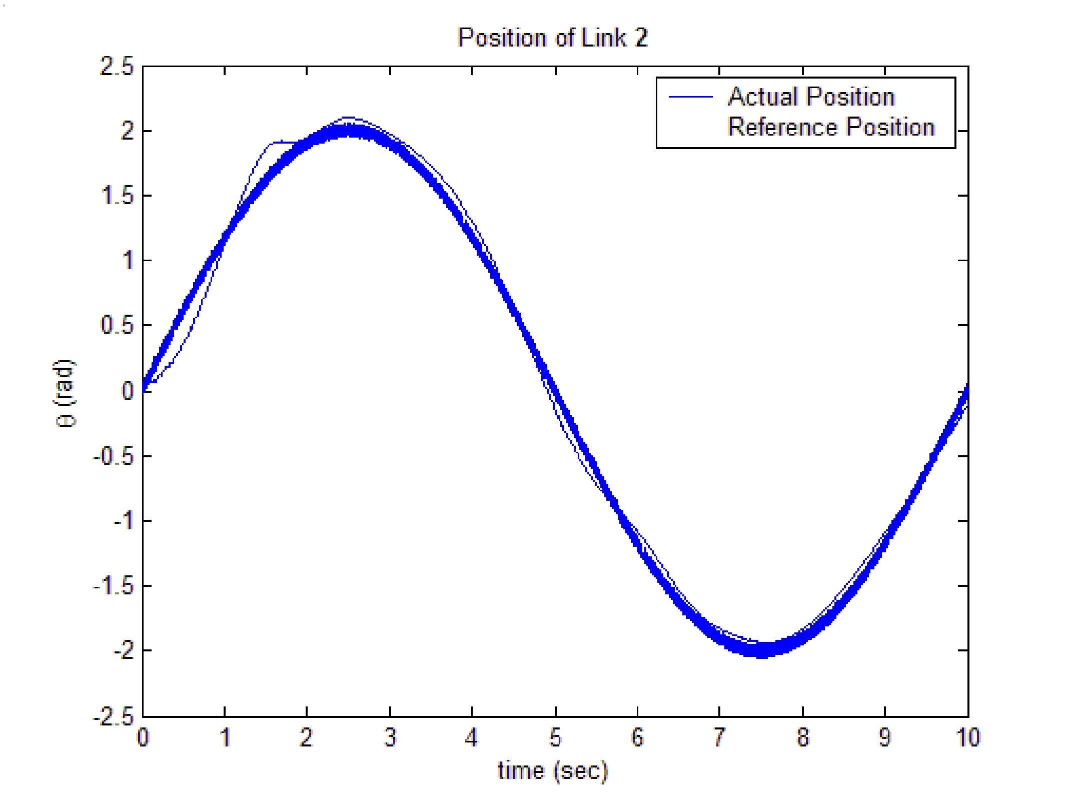

Figure 5.14 Link 2’s reference points for neural network training.

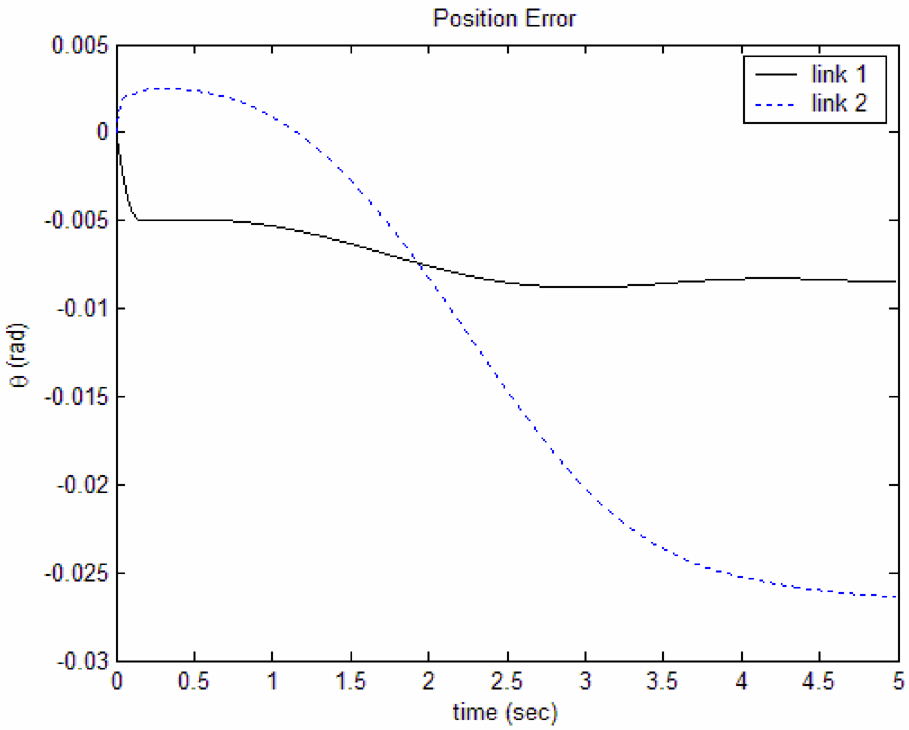

Figure 5.15 Time history of position error of link 1 with neuro-feedback linearized law.

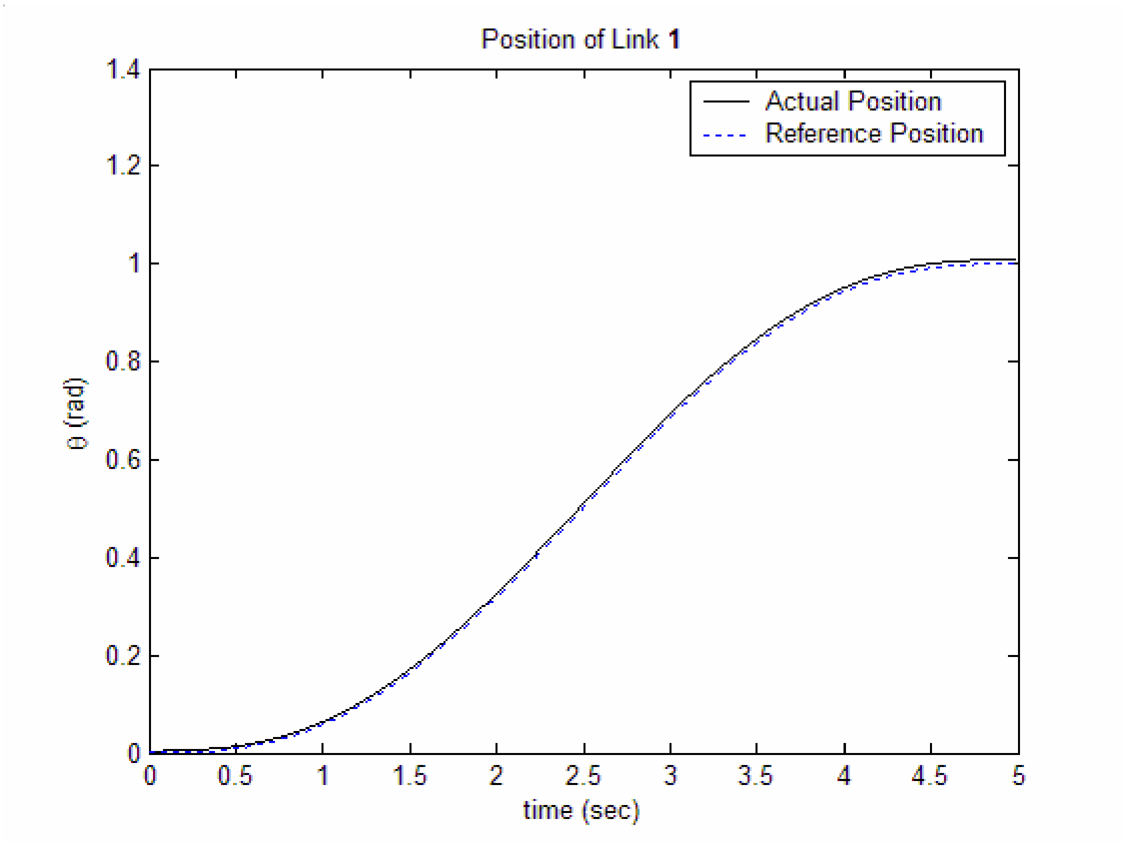

Figure 5.16 Time history of link one's position with neuro-feedback linearized law.

For the first link, the actual path of the robot follows reasonably close to that of the reference trajectory path.

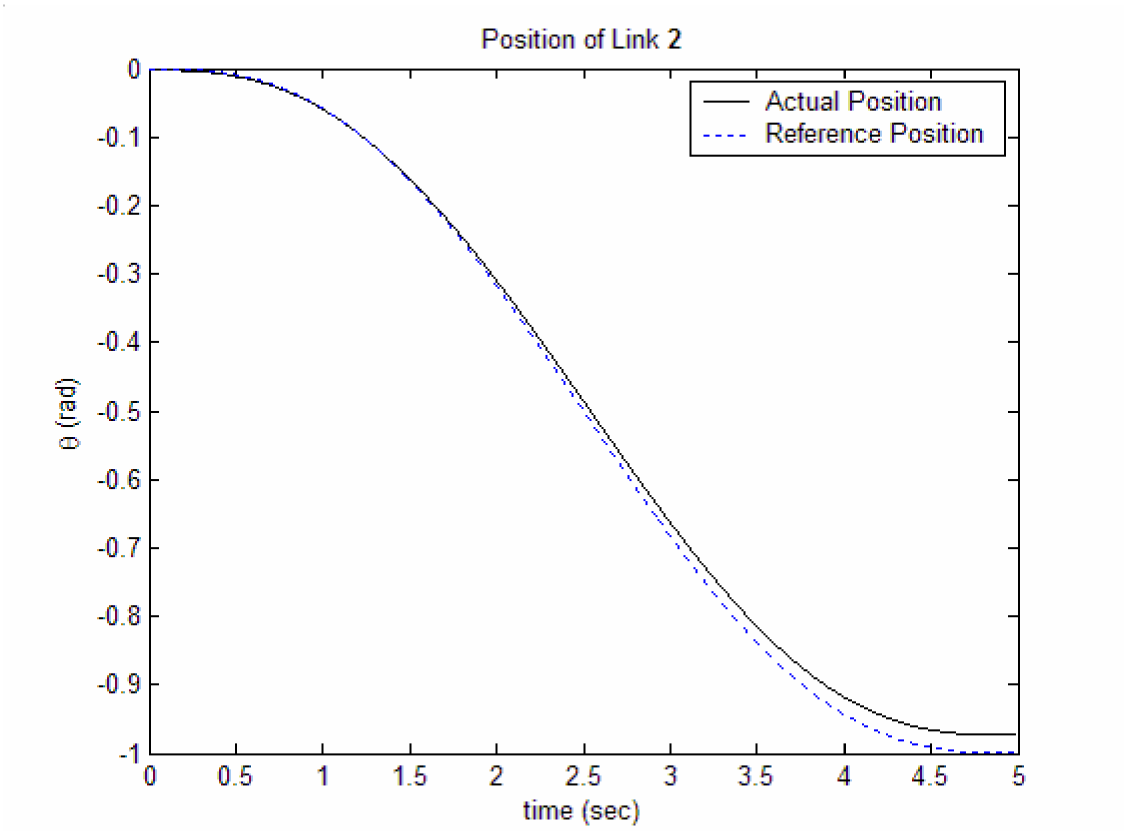

Figure 5.17 Time history of link two's position with neuro-feedback linearized law.

For the second link, the actual path of the robot follows the reference trajectory path with slight deviations after 2 second. Tracking is more accurate before t=2sec.

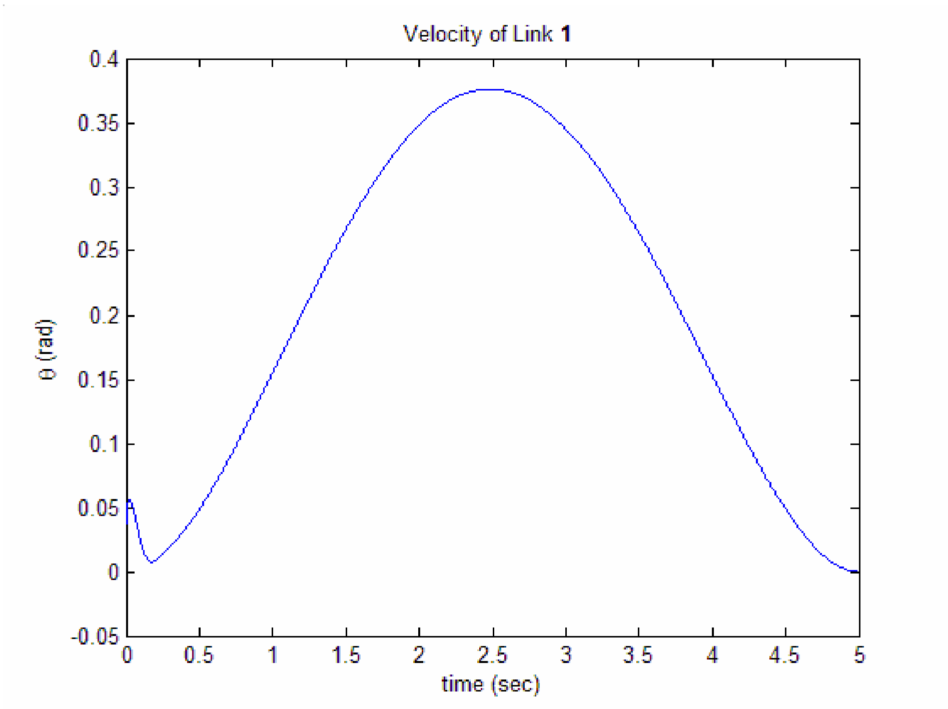

Figure 5.18 Time history of link one's velocity with neuro-feedback linearized law.

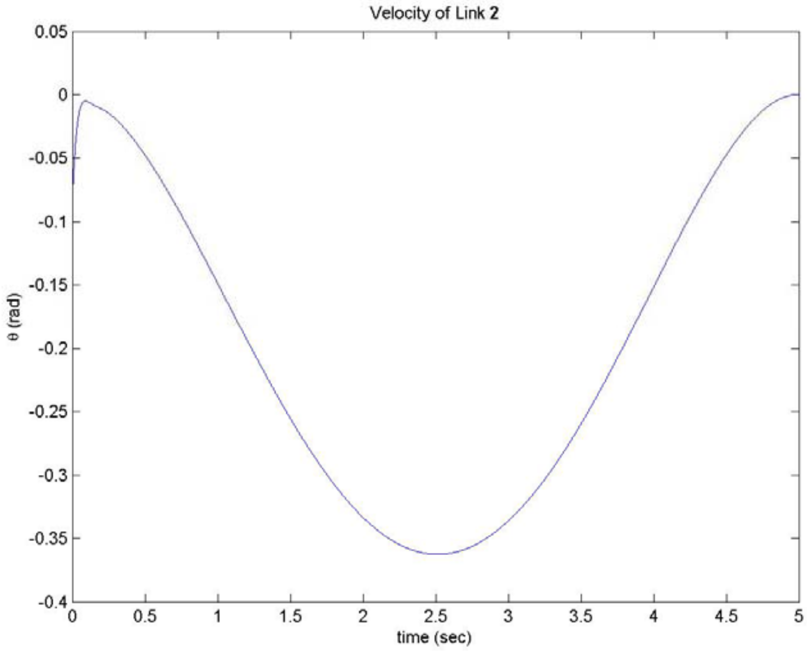

Figure 5.19 Time history of link two's velocity with neuro-feedback linearized law.

5.4 Conclusion

Performance comparisons are made between the neuro-computed torque control method and the conventional computed torque control method. A two-link manipulator model has been used for simulation studies. The backpropagation neural network in the neural-CTC control method has been found to give excellent tracking control results. It is shown that the neural-CTC method can learn a robot’s nonlinear dynamic behaviour very well.

The feedback linearization technique is applied to the nonlinear two-link robot model. An inner loop control is added so that an inner linearized block of control system can be generated. The results of position tracking control are validated by simulation of a two-link robot model. Good tracking results are obtained with the feedback linearized controller. The tracking performance with the designed linearized feedback law also gave results comparable to that of NCTC and CTC methods. The linearized feedback law simulation consumes the least amount of computation time, and tracking results are still good, though not as good as that of NCTC. The neuro-linearized feedback controller tracks the trajectory reasonably well. Since it is a novel controller, there is potential in using it experimentally if adaptations to link weight changes are needed. The time taken for simulation for NCTC is the longest, but the NCTC gives better tracking results, and is model free.

Feedback linearization has the advantage of allowing for the use of linear techniques to achieve desired closed loop control specifications for nonlinear full dynamic robot descriptions. Feedback linearization is also robust to parameter uncertainty. The disadvantage of feedback linearization is that when the dynamic model becomes more detailed as the number of robot links increases, computational complexity results. This limitation may be overcome with the current availability of fast computers. From the results of the current work, the proposed feedback linearized controller has been shown to have potential for controlling nonlinear multi-linked robotic systems real-time with good tracking results.

6 References

[1] Abdallah C, Dawson D., Dorato P. and Jamshidi M. *Survey of Robust Control for Rigid Robots*. IEEE Control Systems, pp. 24-30. 1991.

[2] Abdelhameed M. M. *Adaptive Neural Network Based Controller for Robots.* Mechatronics, Vol. 9, pp. 147-162. 1999.

[3] Acosta L., Marichal G. N., Moreno L., Rodrigo J. J., Hamilton A. and Mendez J. A. *A Robotic System based on Neural Network Controllers.* Artificial Intelligence in Engineering, Vol. 13, pp. 393-398. 1999.

[4] Astrom K. J. and Wittenmark B. *Computer-Controlled Systems: Theory and Design.* 3rd Ed. Prentice Hall. 1997.

[5] Astrom K.J. and Hagglund T. *PID Controller: Theory, Design, and Tuning*. Research Triangle Park, NC, USA. 1995.

[6] Astrom K.J. and Wittenmark B. *Adaptive Control*. New York: Addison-Wesley. 1995.

[7] Barambones O., Etxebarria V. *Robust Neural Control for Robotic Manipulators*. Automatica, Vol. 38, pp. 235-242. 2002.

[8] Berger R. M. and Elmaraghy H. A. *Feedback Linearization Control of Flexible Joint Robots*. Robotics and Computer Integrated Manufacturing, Vol. 9, No. 3, pp. 239-246, 1992.

[9] Berghuis H. and Nijmeijer H. *A Passivity Approach to Controller-Observer Design for Robots*. IEEE Transactions on Robotics and Automation, Vol. 9, No. 6, pp. 740-754, December 1993.

[10] Berghuis H., Nijmeijer H., Lohnberg P. *An Addendum on 'Robust Control of Robots by the Computed Torque Method*. Systems and Control Letters, Vol. 18, pp. 403-407. 1992.

[11] Cabrera J. B. D and Narendra K. S. *Issues in the Application of Neural Networks for Tracking Based on Inverse Control*. IEEE Transactions on Automatic Control, Vol. 44, No. 11, pp. 2007-2027. November 1999.

[12] Caccavale F, Natale C., Siciliano B. and Villani L. *Resolved-Acceleration Control of Robot Manipulators: A Critical Review with Experiments.* Robotica, Vol. 16, pp. 565-573, 1998.

[13] Canudas C. D. W. and Slotine J. J. E. *Sliding Observers for Robotic Manipulators.* Automatica, Vol. 27, No. 5, pp. 859-864. 1991.

[14] Canudas C. d. W., Siciliano B. and Bastin G. *Theory of Robot Control*. Springer. 1996.

[15] Cavallo A., Setola R. and Vasca F. *Using Matlab: Simulink and Control System Toolbox: A Practical Approach*. Europe: Prentice Hall. 1996.

[16] Chang, C.-C., & Lin, C.-J. *LIBSVM: Introduction and benchmarks* (Tech. Rep.). Taipei, Taiwan: Department of Computer Science and Information Engineering, National Taiwan University, 2000.

[17] Chang, C.-C., & Lin, C.-J. *LIBSVM: A library for support vector machines* [Computer Software]. Available on-line: http://www.csie.ntu.edu.tw/~cjlin/libsvm , 2001.

[18] Chang, C.-C., & Lin, C.-J. *Training ν-support vector classifiers: Theory and algorithms. Neural Computation, 13*(9), 2119–2147, , 2001.

[19] Cheng X. P. and Patel R. V. *Neural Network Based Tracking Control of a Flexible Macro-micro Manipulator System*. Neural Networks, Vol. 16, pp. 271-286. 2003.

[20] Chiavarini S., Siciliano B. and Villani L. *An Adaptive Force/Position Control Scheme for Robot Manipulators.* Applied Mathematics and Computer Science, Vol. 7, No. 2, pp. 293-303, 1997.

[21] Colbaugh R., Glass K., and Seraji H. *Adaptive Tracking Control of Manipulators: Theory and Experiments*. Robotics and Computer-Integrated Manufacturing, Vol. 12, No. 3, pp. 209-216. 1996.

[22] Corke P. *A Robotics Toolbox for Matlab*. IEEE Robotics and Automation Magazine, pp. 24-32. March 1996

[23] Efe M. O. and Kaynak O. *A Comparative Study of Neural Network Structures in Identification of Nonlinear Systems*. Mechatronics, Vol. 9, pp. 287-300. 1999.

[24] Ertugrul M., Kaynak O. *Neuro Sliding Mode Control of Robotic Manipulators. Mechatronics*. Vol. 10, pp. 239-263, 2000.

[25] Fu K. S., Gonzalez R. C. and Lee C. S. G. *Robotics: Control, Sensing, Vision and Intelligence*. McGraw Hill. 1987.

[26] Ge S. S., Hang C. C., Lee T. H. and Zhang T. *Stable Adaptive Neural Network Control.* Kluwer Academic Publishers. 2002.

[27] Ge S. S., Huang L. and Lee T. H. *Model-Based and Neural-Network-based Adaptive Control of two Robotic Arms Manipulating an Object with Relative Motion.* International Journal of Systems Science, Vol. 32, No. 1, pp. 9-23. 2001.

[28] Ge S. S., Lee T. H. and Harris C. J. *Adaptive Neural Network Control of Robotic Manipulators*. World Scientific Series in Robotics and Intelligent Systems- Vol. 19. 1998.

[29] Goodwin G. C. and Sin K. S. *Adaptive Filtering Prediction and Control*. Prentice Hall. 1984.

[30] Gupta M. M., Jin L. and Homma N. *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. New Jersey: John Wiley & Sons. 2003.

[31] Gupta P. and Sinha N. K. *Intelligent Control of Robotic Manipulators: Experimental Study using Neural Networks*. Mechatronics, Vol. 10, pp. 289-305, 2000.

[32] Gutierrez L.B., Lewis F.L. and Lowe J.A. *Implementation of a neural network tracking controller for a single flexible link: comparison with PD and PID controller*. IEEE Trans. Ind. Electron, Vol 45, p. 307. 1998.

[33] Hagan M. T. and Demuth H. B. *Neural Networks for Control*. Proceedings of the American Control Conference, June 1999, San Diego, California, pp. 1642-1656.

[34] Hagan M. T., Demuth H. B. and Beale M. *Neural Network Design*. PWS Publishing Company. 1996.

[35] Hagan M. T., Orlando D. J., Schultz R. *Training Recurrent Networks for Filtering and Control.* In Recurrent Neural Networks: Design and Applications. Pp. 325-354, Chapter 12, New York: CRC Press, 2001

[36] Haykins S. *Neural Networks: A Comprehensive Foundation*. 2nd Ed. USA: Prentice Hall. 1999.

[37] Holt K. and Desrochers A. A. *Disturbance Rejection for Space-Based Manipulators*. Journal of Robotic Systems, *16(5)*, pp. 285-299. 1999.

[38] Huijberts H. J. C. *Dynamic Feedback in Nonlinear Synthesis Problems*. Netherlands: Centrum voor Wiskunde en Informatica CWI Tract, Stichting Mathematische Centrum Amsterdam. 1994

[39] Hunt, K. J., Sbarbaro, D., Zbikowski, R. and Gawthrop, P. J. *Neural networks for control systems —a survey Automatica*, 1992, **28**, (6), pp. 1083-1112, 1992.

[40] Isidori A. and Byrnes C. *Output Regulation of Nonlinear Systems*. IEEE Transactions on Automatic Control, Vol. 35, No. 2, pp. 131-140, February 1990.

[41] Isidori A. Krener A. J., Giorgi C. G. and Monaco S. *Nonlinear Decoupling via Feedback: A Differential Geometric Approach.* IEEE Transactions on Automatic Control, Vol. AC-26, No. 2, pp. 331-345. April 1981.

[42] Isidori A. *Nonlinear Control Systems*. London: Springer, Third Ed., 1995.

[43] Johansson R., Robertsson A., Nilsson K., Verhaegen M. *State-Space System Identification of Robot Manipulator Dynamics*, Mechatronics, Vol. 10, pp. 403-418. 2000.

[44] Jungbeck M. and Cerqueira J. J. F. *Comments on "Intelligent Optimal Control of Robotica Manipulator using Neural Networks*. Automatica, Vol. 38, pp. 745, 2002.

[45] Karlik B., Aydin S. *An Improved Approach to the Solution of Inverse Kinematics Problems for Robot Manipulators*. Engineering Applications of Artificial Intelligence, *13*, pp. 159-164, 2000.

[46] Khalil H. K. *Nonlinear Systems*. Prentice Hall 2002.

[47] Khalil H. K. *Adaptive Output Feedback Control of Nonlinear Systems Represented by Input-Output Models*. IEEE Transactions on Automatic Control, Vol. 41, No. 2, pp. 177-188, February 1996.

[48] Khalil W. and Dombre E. *Modeling, Identification and Control of Robots*. London: Hermes Penton. 2002.

[49] Kim Y. H. and Lewis F. L. *Neural Network Output Feedback Control of Robot Manipulators*. IEEE Transactions on Robotics and Automation, Vol. 15, No. 2, pp. 301-309. April 1999.

[50] Kim Y. H. and Lewis. *High-Level Feedback Control with Neural Networks*. World Scientific Series in Robotics and Intelligent Systems- Vol. 21, Singapore: World Scientific. 1998.

[51] Kim Y. H., Lewis F. L., Dawson D. M. *Intelligent Optimal Control of Robotic Manipulators using Neural Networks*. Automatica, Vol. 36, pp. 1355-1364. 2000.

[52] Koivo A.J. *Fundamentals for Control of Robotic Manipulators*. New York: Wiley. 1989.

[53] Kuschewski J. G, Hui S. and Zak S. H. *Application of Feedforward Neural Networks to Dynamical System Identification and Control*. IEEE Transactions on Control Systems Technology, Vol. 1., No. 1, pp. 37-49. March 1993.

[54] Kwatney H. F. and Blankenship G. L. *Nonlinear Control and Analytical Mechanics: A Computatinal Approach.* Berlin: Birkhauser. 2001.

[55] Levin A. U. and Narendra K. S. *Control of Nonlinear Dynamical Systems Using Neural Networks- Part 2: Observability, Identification, and Control*. IEEE Transactions on Neural Networks, Vol. 7, No. 1, pp. 30-42. January 1996.

[56] Levin A. U. and Narendra K. S. *Identification of Nonlinear Dynamical Systems Using Neural Networks*. In Neural Systems for Control, ed by Omidvar O. and Elliott D. L., Chapter 6, pp.129-159. Academic Press. 1997.

[57] Lewis F. L. *Intelligent Control: Neural Network Control of Robot Manipulators*. IEEE Expert, pp. 64-75. 1996.

[58] Lewis F. L., Jagannathan S. and Yesildirek A. *Neural Network Control of Robot Arms and Nonlinear Systems*. In Neural Systems for Control, ed by Omidvar O. and Elliott D. L.,Chapter 7, pp.161-211. Academic Press. 1997.

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

[60] Lewis F. L., Yesildirek A. and Liu K. *Multilayer Neural-Net Robot Controller with Guaranteed Tracking Performance*. IEEE Transactions on Neural Networks, Vol. 7, No. 2, pp. 388-399. March 1996.

[61] Lewis F.L., Abdallah C.T. and Dawson D.M. *Control of Robot Manipulators*. New York: Macmillan. 1993.

[62] Li Q., Poo A. N. and Ang M. *An Enhanced Computed-Torque Control Scheme for Robot Manipulators with a Neuro-Compensator*. Proceeding of the IEEE International Conference on Systems, Man & Cybernetics, 22-25 October 1995, Vancouver, Canada.

[63] Lin, C.-J. Formulations of support vector machines: A note from an optimization point of view. *Neural Computation, 13*(2), 307–317, 2001.

[64] Lucibello P. *Some Notes on Output Regulation of Nonlinear Systems*. In Proceedings of the 32nd Conference on Decision and Control, San Antonio, Texas, pp. 3556-3561, December 1993.

[65] Marino R. *Adaptive Control of Nonlinear Systems: Basic Results and Applications*. A. Rev. Control, Vol. 21, pp. 55-66. 1997.

[66] Marino R. and Tomei P. *Nonlinear Control Design: Geometric, Adaptive and Robust*. UK: Prentice Hall. 1995.

[67] Marino R. and Tomei P. *Nonlinear Output Feedback Tracking with Almost Disturbance Decoupling*. IEEE Transactions on Automatic Control, Vol. 44, No. 1, pp.18-28. 1999.

[68] Marino R., Tomei P., Kanellakopoulos I. And Kokotovic P. V. *Adaptive Tracking for a Class of Feedback Linearizable Systems.* Vol. 39, No. 6, pp. 1314-1319, June 1994.

[69] Miller W. T., Sutton R. S. and Werbos P. J. (Eds.) *Neural Networks for Control and Systems*. Peter Peregrinus Ltd. 1992.

[70] Moallem M., Khorasani K., Patel R. V. *An Integral Manifold Approach for Tip-Position Tracking of Flexible Multi-Link Manipulators*. IEEE Transactions on Robotics and Automation, Vol. 13, No. 6, pp. 823-836, 1997.

[71] Moallem M., Patel R. V. and Khorasani K. *Experimental Results for Nonlinear Decoupling Control of Flexible Multi-Link Manipulators*. In Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997, pp. 3142-3147.

[72] Moallem M., Patel R. V., Khorasani K. *An Observer-Based Inverse Dynamics Control Strategy for Flexible Multi-Link Manipulators*. In Proceedings of the 35th Conference on Decision and Control Kobe, Japan, December 1996, pp. 4112-4117.

[73] Moallem M., Patel R. V., Khorasani K. *Nonlinear Tip-Position Tracking Control of a Flexible-Link Manipulator: Theory and Experiments*. Automatica, Vol. 37, pp. 1825-1834. 2001.

[74] Moya M. M. *Robot Control Systems: A Survey*. Robotics, Vol. 3, pp. 329-351. 1987.

[75] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machinesn (and other kernel-based learning methods).* Cambridge University Press, 2000.

[76] Narendra K. S. and Annaswamy A. M. *Stable Adaptive Systems*. Prentice Hall. 1989.

[77] Narendra K. S. and Parthasarathy K. *Identification and Control of Dynamical Systems using Neural Networks*. IEEE Transactions on Neural Networks. Vol. 1, No. 1, pp. 4-27. March 1990.

[78] Narendra, K. S. and Mukhopadhyay S. *Adaptive Control Using Neural Networks and Approximate Models.* IEEE Transactions on Neural Networks, Vol. 8, No. 3, pp. 475-485. May 1997.

[79] Nguyen D. H. and Widrow B. *Neural Networks for Self-Learning Control Systems*. IEEE Control Systems Magazine, pp. 18-23. 1990.

[80] Ogata, K. *Modern Control Engineering*, Prentice-Hall, Englewood Cliffs, N. J., 1970.

[81] Ortega R. and Spong M. W. *Adaptive Motion Control of Rigid Robots: A Tutorial*. Automatica, Vol. 25, No. 6, pp.877-888. 1989.

[82] Patarinski S. P. and Botev R. G. *Robot Force Control: A Review*. Mechatronics, Vol. 3, No. 4, pp. 377-398. 1993.

[83] Patino H. D., Carelli R. and Kuchen B. R. *Neural Networks for Advanced Control of Robot Manipulators*. IEEE Transactions on Neural Networks, Vol. 13, No. 2, pp. 343-354. March 2002.

[84] Poo A. N., Hong G. S., Xi W. Y and Chan K. *Experimental Studies on a Neural-network Based Controller for a single rigid link manipulator*. In International Symposium on Intelligent Automation and Control, May 1998, Alaska, USA.

[85] Poznyak A., Sanchez E. N. and Yu W. *Differential Neural Networks for Robust Nonlinear Control: Identification, State Estimation and Trajectory Tracking*. Singapore: World Scientific. 2001

[86] Qu Z. H., Dawson D. M. and Dorsey J. F. *Exponentially Stable Trajectory Following of Robotic Manipulators Under a Class of Adaptive Controls*. Automatica, Vol. 28, No. 3, pp. 579-586. 1992.

[87] Qu Z. H., Dorsey J. F., Zhang X. F. and Dawson D. M. *Robust Control of Robots by the Computed Torque Law*. Systems and Control Letters, Vol. 16, pp. 25-32. 1991.

[88] Rafizadeh H. and Perez R. *Comparison of Adaptive Controllers Applied to the Robotic Manipulators*. Journal of the Franklin Institute, Vol. 332B, No. 4, pp. 403-417. 1995.

[89] Ramirez J. A., Cervantes I. and Kelly R. PID *Regulation of Robot Manipulators: Stability and Performance*. Systems and Control Letters, Vol. 41, pp. 73-83. 2000.

[90] Saad D. *On-Line Learning in Neural Networks*. Publications of the Newton Institute, Cambridge University Press. 1998.

[91] Sastry S. and Bodson. *Adaptive Control of a Class of Nonlinear Systems*. In Adaptive Control: Stability, Convergence, and Robustness. Chapter 7, pp. 294-323, Prentice-Hall. 1994.

[92] Sastry S. S. and Isidori A. *Adaptive Control of Linearizable Systems*. IEEE Transactions on Automatic Control, Vol. 34, No. 11, pp. 1123-1131. November 1989.

[93] Schilling R.J. *Fundamentals of Robotics: Analysis and Control*. Prentice-Hall, Englewood Cliffs, NJ. 1998.

[94] Schölkopf, B. *Support vector learning*. Doctoral dissertation, Munich: R. Oldenbourg Verlag, 1997.

[95] Schölkopf, B., Burges, C. J. C., & Smola, A. J. *Advances in kernel methods—Support vector learning*. Cambridge, MA: MIT Press, 1999.

[96] Schölkopf, B., Smola, A. J. *Learning with Kernels- Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA: MIT Press, 2002.

[97] Schölkopf, B., Smola, A. J., & Williamson, R. Shrinking the tube: A new support vector regression algorithm. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in neural information processing systems, 11*. Cambridge, MA: MIT Press, 1999.

[98] Schölkopf, B., Smola, A., Williamson, R. C., & Bartlett, P. L. *New support vector algorithms*. Neural Computation, *12*, 1207–1245, 2000.

[99] Schwartz C. A. and Mareels I. M. Y. *Comments on "Adaptive Control of Linearizable Systems"*. IEEE Transactions on Automatic Control, Vol. 37, No. 5, pp. 698-701, May 1992.

[100] Schwarz H. *Systems Theory of Nonlinear Control-An Introduction*. Germany: Shaker Verlag, 2000.

[101] Sciavicco L. and Sciliano B. *Modeling and Control of Robot Manipulators*. 2nd Ed., Springer, 2002.

[102] Shampine L. F. and Reichelt M. W. *The Matlab ODE Suite*. SIAM J. Sci. Comput., Vol. 18, No. 1, pp. 1-22. January 1997.

[103] Shampine L. F., Reichelt M. W. and Kierzenka J. A. *Solving Index-I DAEs in Matlab and Simulink.* SIAM Review, Vol. 41, No. 3, pp. 538-552. 1999.

[104] Siciliano B. *Robot Control*. In Perspectives in Control Engineering: Technologies, Applications and New Directions, ed by Samad R., pp. 442-461. IEEE Press, Piscataway, NJ. 2000.

[105] Slotine J.J.E. and Li W. *Applied Nonlinear Control*. Prentice-Hall, Englewood Cliffs, NJ. 1991.

[106] Slotine J.J.E. and Li W.P. *Adaptive Manipulator Control: A Case Study*. IEEE Transactions on Automatic Control, Vol. 33, No. 11. pp. 995-1003. November 1988.

[107] Smola, A. J. *Regression Estimation with Support Vector Learning Machines.* Diplomingenieur der Physik Dissertation, Technische Universität, 1996

[108] Smola, A. J. *Learning with kernels*. Doctoral dissertation, Technische Universität Berlin. Also: *GMD Research Series No. 25*, Birlinghoven, Germany, 1998

[109] Smola, A. J., Schölkopf, B. *A tutorial on Support Vector Regression.* NeuroCOLT2 Technical Report Series NC2-TR-1998-030, 1998.

[110] Sontag E. D. *Mathematical Control Theory: Deterministic Finite Dimensional Systems.* 2nd Ed., New York: Springer. 1998.

[111] Spong M. W. *Adaptive Control of Flexible Joint Manipulators*, Systems & Control Letters, Vol. 13, pp. 15-21. 1989.

[112] Spong M. W. On Feedback *Linearization of Robot Manipulators and Riemannian Curvature*. In Essays on Mathematical Robotics. The IMA Volumes in Mathematics and its Applications, Vol. 104, Baillieul J., Sastry S. S. and Sussmann H. J. (eds.), pp.185-202, New York: Springer-Verlag, 1998.

[113] Spong M. W. *On Robust Control of Robot Manipulators*. IEEE Transactions on Automatic Control, Vol. 37, No. 11, pp. 1782-1786. November 1992.

[114] Spong M. W., Ortega R. and Kelly R. *Comments on "Adaptive Manipulator Control: A Case Study."* IEEE Transactions on Automatic Control, Vol. 3. No. 6, pp. 761-762. June 1990.

[115] Sun F. C., Li H. X. and Li L. *Robot Discrete Adaptive Control based on Dynamic Inversion using Dynamical Neural networks*. Automatica, Vol. 38, pp. 1977-1983. 2002.

[116] Sun F. C., Sun Z. Q. and Woo P. Y. N. *Neural Network-Based Adaptive Controller Design of Robotic Manipulators with an Observer*. IEEE Transactions on Neural Networks, Vol. 12, No. 1, January 2001.

[117] Talebi H. A., Khorasani K. and Patel R. V. *Neural Network based Control Schemes for Flexible-link Manipulators: Simulations and Experiments*. Neural Networks, Vol. 11, pp. 1357-1377. 1998.

[118] Talebi H. A., Patel R. V. and Asmer H. *Neural Network Based Dynamic Modeling of Flexible-Link Manipulators with application to the SSRMS*. Journal of Robotic Systems, *17(7)*, pp. 385-401. 2000.

[119] Taware A. and Tao G. *Control of Sandwich Nonlinear Systems*. Lecture Notes in Control and Information Sciences, Vol. 288, Heidelberg, Berlin: Springer-Verlag. 2003.

[120] Terra M. H. and Tinos R. *Fault Detection and Isolation in Robotic Manipulators via Neural Networks: A Comparison among three Architectures for Residual Analysis*. Journal of Robotic Systems, *18(7)*, pp. 357-374. 2001.

[121] Tomei P. *Tracking Control of Flexible Joint Robots with Uncertain Parameters and Disturbances*. IEEE Transactions on Automatic Control Vol. 39, No. 5, pp. 1067-1072. 1994.

[122] Vapnik, V. *Statistical learning theory***.** New York: Wiley, 1998.

[123] Villani L., Natale C., Siciliano B., Canudas C. d. W. *An Experimental Study of Adaptive Force/ Position Control Algorithms for an Industrial Robot*. IEEE Transactions on Control Systems Technology, Vol. 8, pp. 777-786, September 2000.

[124] Wai R. J. *Tracking Control based on Neural Network strategy for Robot Manipulator*. Neurocomputing, Vol. 51, pp. 425-445, April 2002.

[125] Widrow B. and Lehr M. A. *30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation*. Proceedings of the IEEE, Vol. 78, No. 9, pp. 1415-1442. September 1990.

[126] Wilkie J., Johnson M. and Katebi R. *Control Engineering: An Introductory Course*. Palgrave. 2002.

[127] Xi W. Y., Poo A. N. and Ang M. *On the Generation of Suitable Data Sets for the Off-line Training of Neural-network Controllers Using Actual Plant Data*. International Symposium on Intelligent Automation and Control, May 1998, Alaska, USA.

[128] Yu J. S., Quick G., Paulicks W., Muller P. C. and Berteit W. *Transputer-based Robot Control System for six-joint Robot Manipulators.* Robotics and Computer-Integrated Manufacturing, Vol. 14, pp. 111-119, 1998.

[129] Zhou K. and Doyle J. C. *Essentials of Robust Control*. USA: Prentice Hall. 1998.

[130] Zhu, H. A., Teo, C. L., Hong, G. S., and Poo, A. N *An enhanced scheme for the model-based control of robot manipulators.* International Journal of Control, **56**, (6), pp. 1243-1261, 1992.

Appendix

A.1

Lagrange's equation of motion is

$$L = \boldsymbol{K} - \boldsymbol{P} \tag{A.1}$$

where *K*= kinetic energy and *P*= potential energy.

The kinetic energy is

$$K_1 = \frac{1}{2} m_1 l_1^2 \dot{q}_1^2 \tag{A.2}$$

and the potential energy is

$$P_1 = m_1 g l_1 \sin q_1 . \tag{A.3}$$

The positions are

$$x_2 = l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \tag{A.4}$$

$$y_2 = l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \qquad , \tag{A.5}$$

and the velocities are

$$\dot{x}_2 = -l_1 \dot{q}_1 \sin q_1 - l_2 (\dot{q}_1 + \dot{q}_2) \sin(q_1 + q_2) \tag{A.6}$$

$$\dot{y}_2 = l_1 \dot{q}_1 \cos q_1 + l_2 (\dot{q}_1 + \dot{q}_2) \cos(q_1 + q_2) . \tag{A.7}$$

The square of velocity is

$$\begin{aligned} v_2^2 &= \dot{x}_2^2 + \dot{y}_2^2 \\ &= l_1^2 \dot{q}_1^2 + l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2 l_1 l_2 (\dot{q}_1^2 + \dot{q}_1 \dot{q}_2) \cos q_2 \end{aligned} . \tag{A.8}$$

The kinetic energy of the second link is

$$\begin{aligned} K_2 &= \frac{1}{2} m_2 v_2^2 \\ &= \frac{1}{2} m_2 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + m_2 l_1 l_2 (\dot{q}_1^2 + \dot{q}_1 \dot{q}_2) \cos q_2 \end{aligned} . \tag{A.9}$$

The potential energy of the second link is

$$\begin{aligned} P_2 &= m_2 g y_2 \\ &= m_2 g [a_1 \sin q_1 + a_2 \sin(q_1 + q_2) \end{aligned} \tag{A.10}$$

Combining kinetic and potential energy results in Lagrangian's equation of motion.

The Lagrangian is thus

$$\begin{aligned} L &= \boldsymbol{K} - \boldsymbol{P} \\ &= K_1 + K_2 - P_1 - P_2 \\ &= \frac{1}{2}(m_1 + m_2) l_1^2 \dot{q}^2 + \frac{1}{2} m_2 l_2^2 (\dot{q}_1 + \dot{q}_2)^2 + m_2 l_1 l_2 (\dot{q}_1^2 + \dot{q}_1 \dot{q}_2) \cos q_2 \\ &\quad - (m_1 + m_2) g l_1 \sin q_1 - m_2 g l_2 \sin(q_1 + q_2) \end{aligned} . \tag{A.11}$$

Differentiating the Lagrangian, the following equations are obtained.

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} = \boldsymbol{\tau} \tag{A.12}$$

$$\frac{\partial L}{\partial \dot{q}_1} = (m_1 + m_2) l_1^2 \dot{q}_1 + m_2 l_2^2 (\dot{q}_1 + \dot{q}_2) + m_2 l_1 l_2 (2\dot{q}_1 + \dot{q}_2) \cos q_2 \tag{A.13}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_1} = (m_1 + m_2)l_1^2\ddot{q}_1 + m_2 l_2^2(\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 (2\ddot{q}_1 + \ddot{q}_2)\cos q_2 - m_2 l_1 l_2 (2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 \tag{A.14}$$

$$\frac{\partial L}{\partial q_1} = -(m_1 + m_2) g l_1 \cos q_1 - m_2 g l_2 \cos(q_1 + q_2) \tag{A.15}$$

$$\frac{\partial L}{\partial \dot{q}_2} = m_2 l_2^2(\dot{q}_1 + \dot{q}_2) + m_2 l_1 l_2 \dot{q}_1 \cos q_2 \tag{A.16}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_2} = m_2 l_2^2(\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 \ddot{q}_1 \cos q_2 - m_2 l_1 l_2 \dot{q}_1\dot{q}_2 \sin q_2 \tag{A.17}$$

$$\frac{\partial L}{\partial q_2} = -m_2 l_1 l_2 (\dot{q}_1^2 + \dot{q}_1\dot{q}_2)\sin q_2 - m_2 g l_2 \cos(q_1 q_2) \tag{A.18}$$

The robot arm dynamics obtained from Lagrange’s equation are

$$\begin{aligned}\tau_1 = {} & [(m_1 + m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos q_2]\ddot{q}_1 + [m_2 l_2^2 + m_2 l_1 l_2 \cos q_2]\ddot{q}_2 \\ & - m_2 l_1 l_2 (2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin q_2 + (m_1 + m_2) g l_1 \cos q_1 + m_2 g l_2 \cos(q_1 + q_2)\end{aligned} \tag{A.19}$$

and

$$\tau_2 = [m_2 l_2^2 + m_2 l_1 l_2 \cos q_2]\ddot{q}_1 + m_2 l_2^2 \ddot{q}_2 + m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 + m_2 g l_2 \cos(q_1 + q_2)\,. \tag{A.20}$$

A.2

During the differentiation of $\frac{\partial}{\partial \mathbf{x}} L_f^{k-1} h(\mathbf{x})$, the Jacobian matrix (4.58) is used. Let

$L_f^{k-1}h(\boldsymbol{x})$ =λ(x) where $\lambda(x) \in \Re^m$ for each $\boldsymbol{x} \in \Re^n$.

$$
\begin{aligned}
\boldsymbol{\lambda}_x(\boldsymbol{x}) &= \frac{\partial}{\partial \boldsymbol{x}} \lambda(\boldsymbol{x}) \\
&= \begin{bmatrix}
\frac{\partial}{\partial x_1}\lambda_1 & \frac{\partial}{\partial x_2}\lambda_1 & \cdots & \frac{\partial}{\partial x_n}\lambda_1 \\
\frac{\partial}{\partial x_1}\lambda_2 & \frac{\partial}{\partial x_2}\lambda_2 & \cdots & \frac{\partial}{\partial x_n}\lambda_2 \\
\cdots & \cdots & \cdots & \cdots \\
\frac{\partial}{\partial x_1}\lambda_m & \frac{\partial}{\partial x_2}\lambda_m & \cdots & \frac{\partial}{\partial x_n}\lambda_m
\end{bmatrix}.
\end{aligned}
\tag{A.21}
$$