ADAPTIVE CONTROL OF ROBOT ARM - A NEURAL NETWORK APPROACH

A DISSERTATION

submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF ENGINEERING

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operation Research)

By

A. EDWARD

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE – 247 667 (INDIA)

MARCH, 1997

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in this dissertation entitled **'ADAPTIVE CONTROL OF ROBOT ARM - A NEURAL NETWORK APPROACH'** in partial fulfillment of the requirements for the award of the degree of **MASTER OF ENGINEERING** in Electrical Engineering with specialization in **SYSTEM ENGINEERING & OPERATIONS RESEARCH** is an authentic record of my own work carried out during the period from August 1996 to March 1997 under the guidance of **Dr. A. K. PANT,** *Professor, Department of Electrical Engineering, University of Roorkee, Roorkee.*

The matter presented in this dissertation has not been submitted by me for any other degree.

[illegible]

DATED : 15 March, 1997 **(A. EDWARD)**

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

[illegible]

15 **March, 1997**

(Dr. A. K. PANT)
Professor
Department of Electrical Engineering
University of Roorkee
Roorkee - 247667 (U.P.)

ACKNOWLEDGEMENT

I feel immense pleasure in expressing my sincere gratitude to my guide Prof. A. K. Pant, Professor, Electrical Engineering Department, University of Roorkee, Roorkee, for his kind support and able guidance throughout my work. His cooperative attitude and indepth knowledge given to me has made my work possible.

I am also thankful to Prof. M. K. Vasantha, Professor, who, in the capacity of being our class O.C. have given me the complete freedom for doing my work in the lab at any time. My sincere thanks are also with Dr. Indra Gupta, Lab Incharge, Computer and Microprocessor Lab, who has provided me with one of the best computers available in the lab.

Last, but not the least, I am also grateful to my labmates Ms. Nidhi Saxena, Mr. Neeraj Dubey and Mr. Sanjay Jain whose presence have made a workable atmosphere in the lab, conducive for my work.

[illegible]

- A. Edward

ABSTRACT

The precise control of robot manipulator to track the desired trajectory is a very tedious job and almost unachievable with the help of conventional controller. This task is achievable to a certain limit with the help of adaptive controllers but these also have their own limitations of assuming that the systems parameters being controlled change relatively slowly. Many algorithms have been proposed from time to time to minimize this deficiencies.

Interfacing of neural network with the robot manipulator is one of the means of getting the rapid convergence of actual trajectory to the desired trajectory. In our work, we have tried to implement the neural network in the adaptive controller with the help of a neural network to control the robot arm. Results have been compared with the conventional P.D. controllers. A comparative study of neural network based controller without adaptive control and a neural network based controller with adaptive control is also given here.

CONTENTS

CHAPTER 1

INTRODUCTION

Adaptive Control of rigid robot manipulators has been the interest of many for several years. In joint space, one of the early applications of adaptive control to the manipulator problem has been done in early 80s. Some of the directions that have emerged for the adaptive control algorithms are the self-tuning regulator(STR) and the model-referenced adaptive control(MRAC). Because of the principal characteristics of model dependence in these methods, satisfactory results of non-linear systems are not available even though it can be able to give a convincing and well developed algorithms for linear systems[1].

The capability of trained neural networks for approximating arbitrary input-output mappings can find an important applications in devising simple procedures for the identification of unknown dynamical plants[2]. Hence it provides us the opportunity of identifying the dynamics of highly complex systems which does not require any model identification. This makes the neural network based methods more useful in the design of adaptive controllers. Furthermore, the computational features of neural networks convert into speed advantages in the identification and control computation at each step of implementation, when compared to the corresponding calculations required in a STR algorithm.

Considerable literature is available which gives the details of neural nets for system identification and identification-based control but only a few have given the use of NN in direct closed-loop controllers. The main problem which remain to be solved is the inability to guarantee satisfactory performance of the system[3].

In this thesis work, we have tried to tackle this deficiencies by considering a three layer neural network, where linearity in the parameter holds. This may be considered as a step in extending adaptive control theory to Neural Network control theory.

A NN controller structure derived using robot control techniques means that the NN weights are tuned on-line with no "learning phase" needed. The controller structure ensures good performance during the initial period if the NN weights are initialized at zero.

The controller is composed of a neural net incorporated into a dynamic system, where the structure comes from the filtered error notions standard in robot control. The basis functions for the NN controller is determined from the physics (Lagrangian dynamics) of general robot arms. It has been shown in [3] that the back propagation tuning techniques generally yields unbounded weights if the net cannot exactly reconstruct a certain non-linear control function or if there are bounded unmodeled disturbances in the robot dynamics.

In practice, the robot motion is basically defined by motion of its end effector, i.e., in robot control, the major concern is that the end effector motion tracks its desired motion, defined in the task space. The PID controllers which are easily implementable and which are fast enough compared to NN controllers lacks the guaranteed tracking performance. This necessitates the development of NN controllers for precise control in applications like welding etc. with sacrificing the CPU time.

The algorithm which has been used in this thesis work is a very simple one and it fulfills the requirement of a robust controller. The initial filtered tracking error of 80 % to 90 % are also been easily tracked.

This is the introduction of the algorithm of neural network based controller. The next step in our work is the development of neural network based adaptive controller. The algorithm for it is discussed in [2]. For the understanding of the adaptive controller and its concept, a brief review of it is given here. Our real motive is to find the superiority of this controller over all other controllers.

REVIEW OF ADAPTIVE CONTROL:

Traditionally, control systems have been designed based on a good understanding of the system to be controlled. When knowledge of the system is limited the relative modern issues of robust control, adaptive control and learning control become important.

One way to attempt to deal with poor knowledge of parameters in a control scheme is through techniques that are generally called *adaptive control.* Adaptive control is closely related to the problem of system identification; in fact, generally an adaptive control can be viewed as being composed of two parts:

(1) An identification portion which identifies parameters of the plant itself, or parameters that appear in the controller for the plant.

(2) A control law portion which implements a control law that is in some way a function of the parameters identified.

The central problem in the synthesis of adaptive controllers is to prove rigorously that the resulting overall system is stable.

One of the possibility of obtaining a satisfactory result of control design and sensitivity analysis is to automate the whole procedure which can be done by providing the regulator with algorithms for parameter estimation and control design. This leads/us to the so called Self Tuning Regulator (STR) and Adaptive Controller. This adaptive controller can handle the system with large parameter variations.

SELF TUNING CONTROL

One way of automating modelling and design is the following: Determine a suitable model structure. Estimate the parameters of the model recursively. Use the estimates to calculate the control law by a suitable design method. A block diagram of such a method is shown in fig. 1.1. The regulator obtained is called a Self Tuning Regulator because it has facility for tuning its own parameters. The regulator can be thought of as being composed of two loops. The inner loop consists of the process and an ordinary linear-feedback regulator. The parameter of the regulator are adjusted by the outer loop which is composed of a recursive parameter estimator and a design calculation. The design calculation box in fig. 1.1 represents an on-line solution to a design problem for a system with known parameters.

APPROACH TO ADAPTIVE CONTROL

The self tuning regulator could also be made to control a process with varying parameters i.e. an adaptive regulator can be achieved. To do this, it is necessary to change the algorithm so that the parameter estimator can track varying process parameters. There are many schemes for adaptive control that are closely related to self-tuner. One of them is described here. The starting point is an ordinary feedback-control loop with a process and a regulator with adjustable parameters. The schemes represent different ways to alter the regulator parameters in response to change in process and disturbance dynamics.

MODEL REFERENCE ADAPTIVE SYSTEMS(MRAS)

This scheme was originally developed for servo problem. The specifications of the reference model tells how the process output ideally should response to the command signal. The block diagram is shown in fig.1.2. Notice that the reference model is part of the control system. The regulator can be thought of as consisting of two loops. The inner loop is an ordinary control loop composed of the process and the regulator. The parameters of the regulator are adjusted by the outer loop in such a way that the error e between the model output y_m and the process output y becomes small. The outer loop thus also looks like a regulator loop. The key problem is to determine the adjustment mechanism so that a stable system, which brings the error to zero is obtained.

The following parameter adjustment mechanism called the MIT rule, was used in the original MRAS:

$$\frac{d\Theta}{dt} = -\alpha e \; \mathrm{grad}_{\Theta} e \qquad \ldots\ldots\ldots \qquad (1.1)$$

where e is the model error and the components of the vector Θ are the adjustable parameters. The number α is a parameter that determines the adaptation rate. Eq.(1.1) represents an adjustment mechanism which is composed of three parts: a

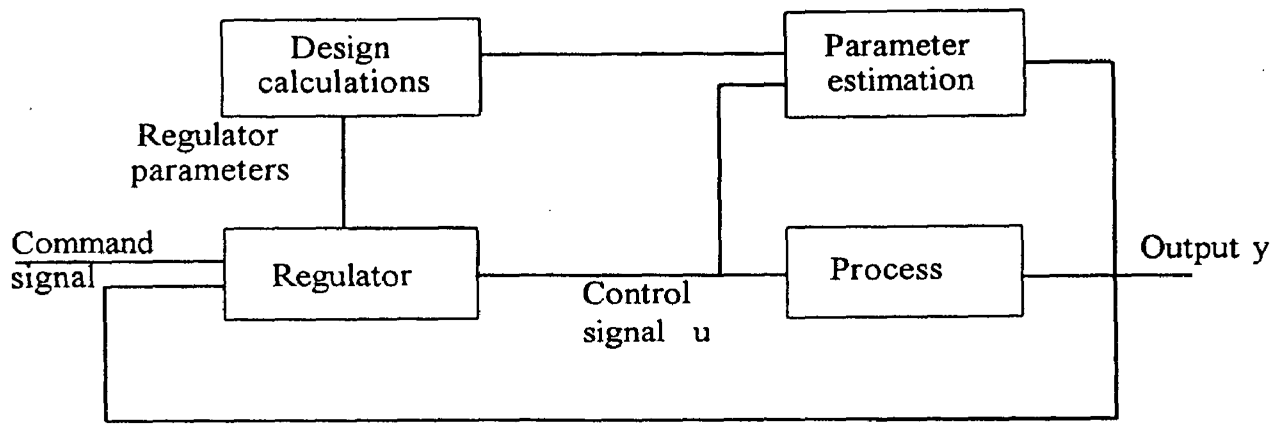


Fig. 1.1 Block diagram of a Self tuning regulator

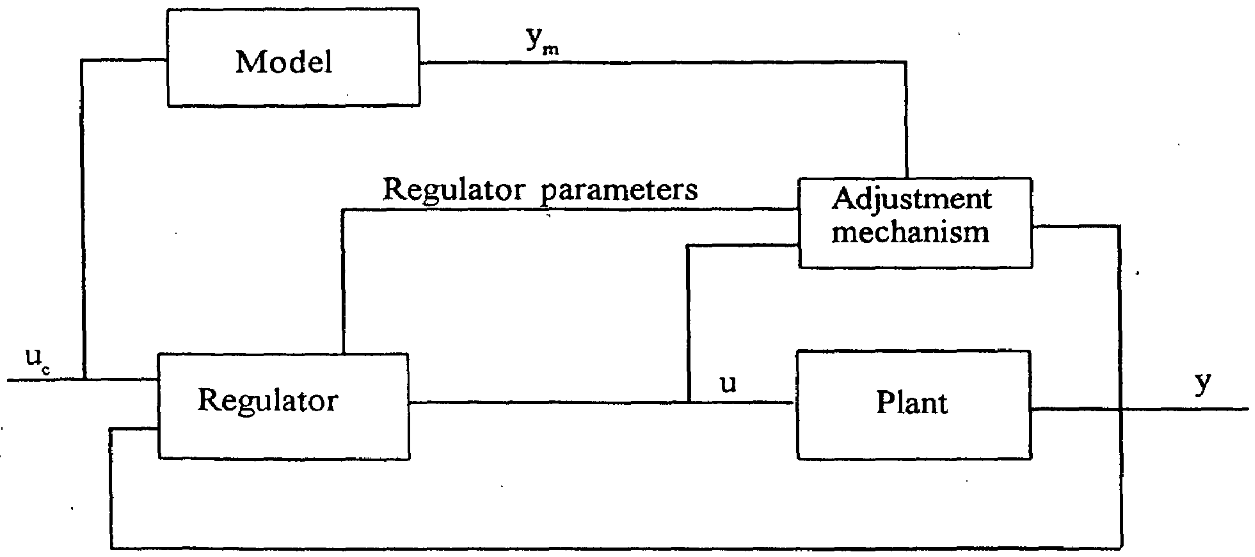


Fig. 1.2 Block diagram of Model-reference adaptive system

linear filter for computing the sensitivity derivatives from process inputs and outputs, a multiplier and an integrator.

The MIT will perform well if the parameter α is small. The allowable size depends on the magnitude of the reference signal. Consequently, it is not possible to give fixed limits that guarantee stability.

Hence, the advantages of the adaptive controller has been exploited and the disadvantages of it in the context of robot manipulator has been removed by the help of neural network. This provides us the guaranteed stability and the guaranteed tracking performance. In chapter 2, we discuss the robot arm dynamics which helps us in getting the mathematical model of it. In chapter 3, the adaptive control algorithm has been discussed. Chapter 4 is used to give the algorithm for the neural network based controller and the simple PD controller. The neural network based adaptive controller has been discussed in chapter 5. Chapter 6 is devoted to give the complete result analysis and conclusion of the whole work being done. The appendices are used to give the quick reference of the general method of deriving the dynamic equations of the robot arm. It also gives the mathematical model of the robot arm having revolute joints. The physical parameters for the robot arm used in this work are also given in it.

CHAPTER 2
ROBOT ARM DYNAMICS

2.1 INTRODUCTION:

Robot arm dynamics deals with the mathematical formulations of the equations of robot arm motion. The dynamic equations of motion of a manipulator are a set of mathematical equations describing the dynamic behaviour of the manipulator. In this chapter, a brief description of these equations are being given [4].

The actual dynamical model of a robot arm can be obtained from known physical laws of Newtonian mechanics and Lagrangian mechanics. This gives the dynamic equations of motion for the various articulated joints of the manipulator in terms of geometric and inertial parameters of the links. Here, the actual robot arm motion equations are developed systematically with the help of Lagrange-Euler formulations.

The derivation of dynamic model of a manipulator based on L-E formulation is simple and systematic. It is a set of second order coupled non-linear differential equations. It provides the state equations for robot dynamics. They can be used to solve the forward-dynamics problems and can also be used for the inverse dynamics problem, that is, given the desired trajectory or the desired generalized coordinates and their first two time derivatives, the generalized force or torque can be computed. These Lagrange-Euler formulation is used in our work to get the dynamic equations of the robot arm manipulator with two links.

2.2 LAGRANGE-EULER FORMULATION:

The general motion equation of a manipulator is conveniently expressed through the desired application of the Lagrange-Euler formulation. The algorithm is expressed by matrix operations. The derivation of the dynamic equation of an *n* degrees of freedom manipulator is based on the understanding of:

1) the 4x4 homogeneous coordinate transformation matrix ${}^{i-1}A_i$, which describes the spatial relationship between the i_{th} and $(i\text{-}1)_{th}$ link coordinate frames.

2) the Lagrange-Euler equation,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \qquad \text{......(2.1)}$$

$$i = 1,2,......,n$$

where L = Lagrangian function = kinetic energy K - potential energy P

K = total kinetic energy of the robot arm

P = total potential energy of the robot arm

q_i = generalized coordinate of the robot arm

$\dot{q}_i$ = first time derivative of the generalized coordinate q_i

τ_i = generalized force or torque applied to the system at joint i to drive link i.

Here, the generalized coordinates are being used as a convenient set of coordinates which completely describes the location (position and orientation) of a system with respect to a reference coordinate frame. We have taken the case of rotary joints, and therefore, $q_i=\theta_i$, the joint angle span of the joint.

2.2.1 KINETIC ENERGY OF ROBOT MANIPULATOR:

The total kinetic energy of a robot arm [7] is,

$$K = \sum_{i=1}^{n} K_i = \frac{1}{2} \sum_{i=1}^{n} \mathrm{Tr}\left[\sum_{p=1}^{i} \sum_{r=1}^{i} U_{ip} J_i U_{ir}^T \dot{q}_p \dot{q}_r \qquad \text{....(2.2)}$$

$$K = \frac{1}{2} \sum_{i=1}^{n} \sum_{p=1}^{i} \sum_{r=1}^{i} [\mathrm{Tr}(U_{ip} J_i U_{ir}^T) \dot{q}_i \dot{q}_r \qquad \text{.....(2.3)}$$

$$\text{where} \quad U_{ij} = \begin{cases} {}^0A_{j-1} Q_j {}^{j-1}A_i & \text{for } j <= i \\ 0 & \text{for } j > i \end{cases} \qquad \text{...(2.4)}$$

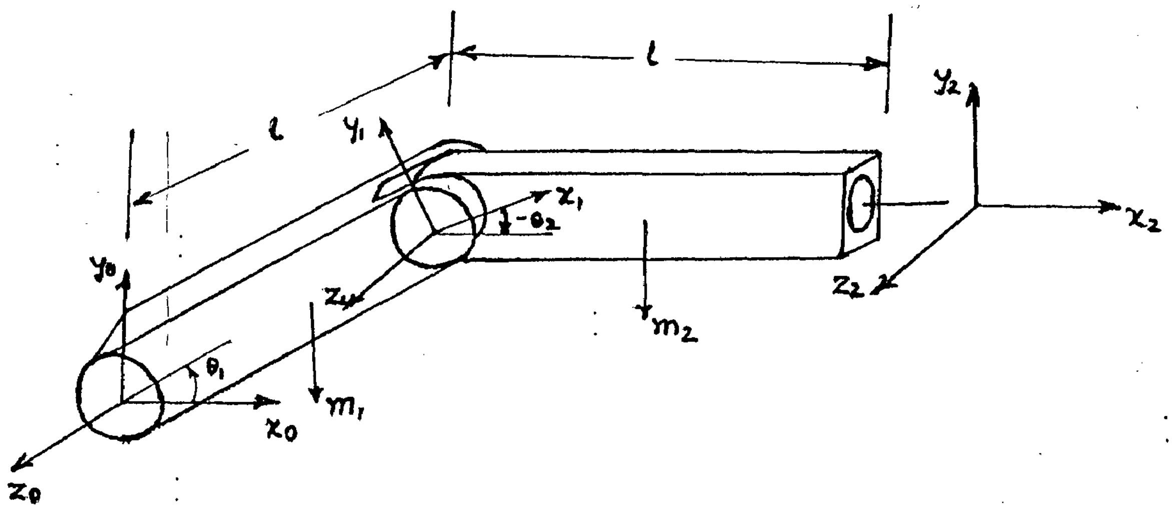


Fig. 2

$$Q_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{for revolute joints.} \qquad \text{....(2.5)}$$

$${}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{..(2.6)}$$

$$J_i = \begin{bmatrix} \frac{-I_{xx}+I_{yy}+I_{zz}}{2} & I_{xy} & Ixz & m_i x_i \\ I_{xy} & \frac{I_{xx}-I_{xy}+I_{zz}}{2} & I_{yz} & m_i y_i \\ I_{xz} & Iyz & \frac{I_{xx}+I_{yy}-I_{zz}}{2} & m_i z_i \\ m_i x_i & m_i y_i & m_i z_i & m_i \end{bmatrix} \qquad \text{..(2.7)}$$

All the non-zero elements in the matrix ${}^0A_i$ are a function of (θ_1 θ_2 ......θ_i) and α_i, a_i, d_i are known parameters and θ_i is the joint variable of joint i. I is the inertia tensor of the robot arm.

2.2.2 POTENTIAL ENERGY OF ROBOT MANIPULATOR:

Let the total potential energy of a robot arm be P and let each of its link potential energy be P_i. Then,

$$P_i = -m_i g\,{}^0\bar{r}_i = -m_i g({}^0A_i\,{}^i\bar{r}_i) \qquad \text{......(2.8)}$$

$$i = 1,2,\ldots\ldots,n$$

and the total potential energy of the robot arm can be obtained by summing all the potential energies in each link,

$$\text{Therefore,} \qquad P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} -m_i g({}^0A_i\,{}^i\bar{r}_i) \qquad \text{....(2.9)}$$

where $g = (g_x \; g_y \; g_z \; 0)$ is a gravity row vector expressed in the base coordinate system. For a level system, g is the acceleration due to gravity ($g = 9.8062$ m/sec^2).

2.2.3 MOTION EQUATION OF A MANIPULATOR:

The Lagrangian equation of a manipulator L = K - P is given by,

$$L = \frac{1}{2}\sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{i} \left[Tr(U_{ij} J_i U_{ik}) \dot{q}_j \dot{q}_k \right] + \sum_{i=1}^{n} m_i g ({}^0A_i \, {}^i\bar{r}_i) \qquad ....(2.10)$$

Applying the Lagrange-Euler formulation to the lagrangian function of the robot arm eq.(2.10), eq.(2.1) yields the necessary generalized torques τ_i for joint i actuator to drive the link of the manipulator. In simpler matrix notation form, it is expressed as,

$$\tau_i = \sum_{k=1}^{n} D_{ik} \ddot{q}_k + \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm} \dot{q}_k \dot{q}_m + C_i \qquad ....(2.11)$$

$$i = 1,2,....,n$$

or in matrix form as,

$$\tau_i = D(q(t))\ddot{q}(t) + h(q(t),\dot{q}(t)) + C(q(t)) \qquad .....(2.12)$$

where,

$\tau(t)$ = nx1 generalized torque vector applied at joints $i = 1,2,.....,n$

$q(t)$ = an nx1 vector of joint variables of the robot arm

$\dot{q}(t)$ = an nx1 vector of joint velocity of the robot arm

$\ddot{q}(t)$ = an nx1 vector of the joint acceleration of the robot arm

$D(q)$ = an nxn inertial acceleration-related symmetric matrix whose elements are,

$$D_{ik} = \sum_{j=\max(i,k)}^{n} Tr(U_{jk} J_i U_{ji}^T) \qquad .....(2.13)$$

$$i,k = 1,2,\ldots\ldots.n$$

$h(q,\dot{q})$ = an nx1 non-linear Coriolis and centrifugal force vector whose elements are,

$$h(q,\dot{q}) = (h_1, h_2,\ldots\ldots,h_n)^T$$

where $$h_i = \sum_{k=1}^{n} \sum_{m=1}^{n} h_{ikm}\dot{q}_k\dot{q}_m \qquad \ldots..(2.14)$$

$$i = 1,2,\ldots\ldots\ldots.,n$$

and $$h_{ikm} = \sum_{j=\max(i,k,m)}^{n} Tr(U_{jkm}J_jU_{ji}^T) \qquad \ldots\ldots(2.15)$$

$$i,k,m = 1,2,\ldots\ldots\ldots n$$

$C(q)$ = an nx1 gravity loading force vector whose elements are,

$$C(q) = (c_1,c_2,\ldots\ldots\ldots,c_n)^T$$

where $$c_i = \sum_{j=1}^{n} (-m_jgU_{ji}{}^j\bar{r}_j) \qquad \ldots..(2.16)$$

$$i = 1,2,\ldots\ldots\ldots n$$

The motion equations of the robot arm with rotary joints are given in Appendix A.

2.3 A TWO LINK MANIPULATOR EXAMPLE:

Using the Lagrange-Euler equations of motion, an example is worked out here for a two link manipulator with revolute joints which is further being used in the simulation work of the thesis.

Assuming,

joint variables = θ_1, θ_2

mass of the manipulator links = m_1, m_2

link parameters = $\alpha_1 = \alpha_2 = 0; d_1 = d_2 = 0$

and $a_1 = a_2 = l$, the length of the arms.

the homogeneous coordinate transformation matrices ${}^{i-1}A_i$ $(i = 1,2)$ are obtained as,

$$^{0}A_{1} = \begin{bmatrix} C_1 & -S_1 & 0 & lC_1 \\ S_1 & C_1 & 0 & lS_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^{1}A_{2} = \begin{bmatrix} C_2 & -S_2 & 0 & lC_2 \\ S_2 & C_2 & 0 & lS_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{0}A_{2} = {}^{0}A_{1}{}^{1}A_{2} = \begin{bmatrix} C_{12} & -S_{12} & 0 & l(C_{12}+C_1) \\ S_{12} & C_{12} & 0 & l(S_{12}+S_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $C_i = \cos(\theta_i)$; $S_i = \sin(\theta_i)$; $C_{ij} = \cos(\theta_i+\theta_j)$;

$S_{ij} = \sin(\theta_i+\theta_j)$;

From the definition of the Q_i matrix, for the rotary joints, we have,

$$Q_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Then,

$$U_{11} = \frac{\partial^{0}A_1}{\partial\theta_1} = Q_1{}^{0}A_1 = \begin{bmatrix} -S_1 & -C_1 & 0 & -lS_1 \\ C_1 & -S_1 & 0 & lS_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Similarly, we calculate the value of $U_{21} = Q_1{}^{0}A_2$ and $U_{22} = {}^{0}A_1Q_2{}^{1}A_2$.

From eq.(2.7) assuming all the products of inertia are zero, the pseudo-inertia matrix J_i will be,

$$J_1 = \begin{bmatrix} 1/3m_1l^2 & 0 & 0 & -1/2m_1l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/2m_1l & 0 & 0 & m_1 \end{bmatrix} \qquad J_2 = \begin{bmatrix} 1/3m_2l^2 & 0 & 0 & -1/2m_2l \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/2m_2l & 0 & 0 & m_2 \end{bmatrix}$$

Then using eq.(2.13), we have,

$$D_{11} = Tr(U_{11}J_1U_{11}^T) + Tr(U_{21}J_2U_{21}^T)$$

$$= 1/3m_1l^2 + 4/3m_2l^2 + m_2C_2l^2$$

$$D_{12} = D_{21} = Tr(U_{22}J_2U_{21}^T)$$

$= 1/3m_2l^2 + 1/2m_2l^2C_2$

$D_{22} = Tr(U_{22}J_2U_{22}^T)$

$= 1/3m_2l^2S_{12}^2 + 1/3m_2l^2C_{12}^2 = 1/3m_2l^2.$

To derive the Coriolis and centrifugal terms, we use eq.(2.14). For i = 1, we have,

$h_1 = h_{111}\dot{\theta}_1^2 + h_{112}\dot{\theta}_1\dot{\theta}_2 + h_{121}\dot{\theta}_1\dot{\theta}_2 + h_{122}\dot{\theta}_2^2$

Using eq.(2.15), we can obtain the value of h_{ikm}. Therefore,

$h_1 = -1/2m_2S_2l^2\dot{\theta}_2^2 - m_2S_2l^2\dot{\theta}_1\dot{\theta}_2$

Similarly, for i = 2, we have,

$h_2 = h_{211}\dot{\theta}_1^2 + h_{212}\dot{\theta}_1\dot{\theta}_2 + h_{221}\dot{\theta}_1 + h_{222}\dot{\theta}_2^2$

$= 1/2m_2S_2l^2\dot{\theta}_1^2.$

Next, we use the eq.(2.16) to derive the gravity related terms,

$C_1 = -[m_1gU_{11}{}^1r_1 + m_2gU_{21}{}^2r_2]$

$= 1/2m_1glC_1 + 1/2m_2glC_{12} + m_1glC_1$

Similarly,

$C_2 = -m_2gU_{22}{}^2r_2$

$= 1/2m_2glC_{12}$

Finally, the Lagrange-Euler equations of motion for the two link manipulator are obtained as,

$\tau(t) = D(\theta)\ddot{\theta}(t) + h(\theta,\dot{\theta}) + C(\theta)$

CHAPTER 3
ADAPTIVE CONTROL OF ROBOT MANIPULATOR

3.1 INTRODUCTION:

The general control algorithms are sometimes felt inadequate because of the requirement of accurate modelling of the arm dynamics and neglection of changes of the load in task cycle which are significant for feedback control strategies. Hence the consideration of adaptive control techniques becomes significant for tracking the desired time-based trajectory as closely as possible over a wide range of motion and payloads.

Adaptive control methods are based on the assumptions that the parameters of the system being controlled do not change too rapidly in comparison to the system time constants [6]. In the robotic manipulators, the system parameters such as the moments of inertia, and the effect of gravity tend to change rapidly as the arm moves from one configuration to another. This is the reason, why the application of adaptive control methods enjoyed limited success in this field.

The basic idea behind adaptive control is that the controller gains gradually change as the parameters of the system being controlled evolve. It is also possible to change the control signal abruptly on the basis of the state of the system being controlled. Control system of this type are referred to as Variable-structured systems.

Among the most widely used methodologies are the self tuning regulator and the model reference adaptive control. It is being discussed in brief here.

3.2 MODEL REFERENCED ADAPTIVE CONTROL:

The concept of a model-referenced adaptive control [7] is based on selecting an appropriate reference model and adaptation algorithm which modifies the feedback gains to the actuators of the actual system. It is driven by the errors between the reference model outputs and the actual system outputs. A general control block diagram of MRAC is shown in fig.(3.1).

A simple model-referenced adaptive control for the control of mechanical manipulators has been proposed in [6]. A linear second order differential equation is selected as the reference model for each degree of freedom of the robot arm. The manipulator is controlled by adjusting the positions and velocity feedback gains to follow the model so that its closed loop performance characteristics closely match the set of desired performance characteristics in the reference model.

Defining the vector y(t) to represent the reference model response and the vector x(t) to represent the manipulator response, the joint i of the reference model can be described by,

$$a_i y_i(t) + b_i y_i(t) + y_i(t) = r_i(t) \qquad ....(3.1)$$

Assuming the manipulator is controlled by position and velocity feedback gains, and that the coupling terms are negligible, the manipulator dynamic equations for joint i can be written as,

$$\alpha_i(t) x_i(t) + \beta_i(t) x_i(t) + x_i(t) = r_i(t) \qquad ....(3.2)$$

where the system parameters $\alpha_i(t)$ and $\beta_i(t)$ are assumed to vary slowly with time.

Using the steepest descent method to minimize the system error which is the difference between the response of the actual system and the reference model, the system parameter adjustment mechanism is governed by,

$$\alpha_i(t) = \left[k_2{}^i e_i(t) + k_1{}^i e_i(t) + k_0{}^i e_i(t)\right]\left[k_2{}^i u_i(t) + k_1{}^i u_i(t) + k_0{}^i u_i(t)\right] \qquad ...(3.3)$$

$$\beta_i(t) = \left[k_2{}^i e_i(t) + k_1{}^i e_i(t) + k_0{}^i e_i(t)\right]\left[k_2{}^i w_i(t) + k_1{}^i w_i(t) + k_0{}^i w_i(t)\right] \qquad ...(3.4)$$

where $u_i(t)$ and $w_i(t)$ and their derivatives are obtained from the solutions of the following differential equations.

$$a_i u_i(t) + b_i u_i(t) + u_i(t) = y_i(t) \qquad ....(3.5)$$

$$a_i w_i(t) + b_i w_i(t) + w_i(t) = y_i(t) \qquad ......(3.6)$$

where $y_i(t)$ and $y_i(t)$ are the first two time derivatives of response of the reference model.

This closed loop adaptive system involves solving the reference model equation

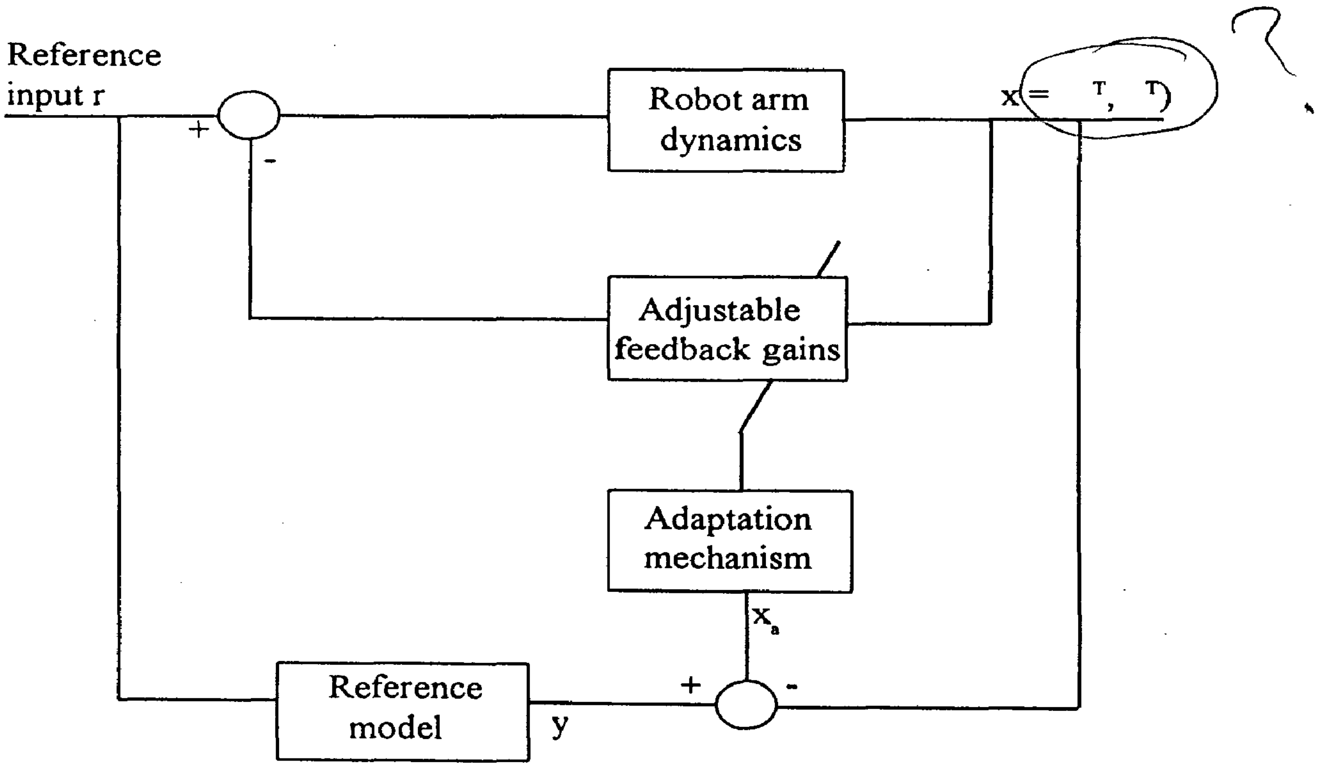


Fig. 3.1 Block diagram for the model-referenced adaptive control

for the given desired input; then the differential equations in eq.(3.5) and(3.6) are solved to yield $u_i(t)$ and $w_i(t)$ and their derivatives for eq.(3.3) and (3.4). Finally, solving the differential equations in eq.(3.3) and (3.4) yields $\alpha_i(t)$ and $\beta_i(t)$.

3.3 A NEW ADAPTIVE CONTROL ALGORITHM:

A new adaptive control algorithm for robot manipulator in task space is proposed in [4]. In this, the use of prediction error and tracking error to drive the parameter estimator is proposed, based on sliding mode and the Lyapunov-like methodology. Under the assumption that the Jacobian matrix is known, it has shown that the proposed adaptive control algorithm is globally stable and convergent.

In the absence of friction and other disturbances, the joint space dynamics of an n-link robot manipulator is written with the help of Lagrange-Euler equations,

$$\sum_j d_{kj}(q)q_j + \sum_{i,j} c_{ijk}(q)q_iq_j + g_k(q) = \tau_k \qquad ..(3.7)$$

$$k = 1,2,.....,n$$

where d_{ij} are coefficients of the inertia matrix $D(q)$, $g_k(q)$ are the gravitational forces and torques. The coefficients c_{ijk} of the Coriolis and centrifugal terms are defined as,

$$c_{ijk} = \frac{1}{2}\left[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} + \frac{\partial d_{ij}}{\partial q_k}\right] \qquad ...(3.8)$$

3.7 ?

and are known as Christoffel symbols. The eq.$\boxed{(3.6)}$ is written as,

$$D(q)q + C(q,q)q + G(q) = \tau \qquad ...(3.9)$$

where the k, j_{th} element of the matrix $C(q,q)$ is defined as,

$$= \sum_{i=1}^{n} c_{ijk}(q)q_i = \sum_{i=1}^{n} \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} + \frac{\partial d_{ij}}{\partial q_k}\right) q_i \qquad ...(3.10)$$

and the component of $G(q)$ is g_k.

Defining,

$$e = \underline{\overset{\wedge}{y_d} - \hat{y}} \qquad ?$$

and $\quad s = \dot{e} - \Lambda e$

where Λ is a diagonal matrix with positive elements, the adaptive controller is defined as,

$$\tau = D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) - K(\ddot{q}-b)$$

$$= Y(q,\dot{q},\ddot{q})\Theta - K(\ddot{q}-b) \qquad \text{......(3.11)}$$

with $b = w\,\dfrac{h_1 + h_2}{s^TPJw} \qquad \text{if } s^TPJw = 0$

where w is an arbitrary chosen vector $w \in R^n$, P is any positive definite matrix with proper dimension,

$$h_1 = s^TP(\Lambda\dot{e} + \ddot{y}_d + \dot{J}\dot{q})$$

$$h_2 = \frac{1}{2}\,\alpha s^TPs \qquad \alpha > 0.$$

$J = \dfrac{\partial f}{\partial q}$ is the Jacobian matrix and $f(q)$ is the end-effector position related to the joint space vector q.

If we choose parameter update law as,

$$\dot{\hat{\Theta}} = \Gamma^{-1}Y^TK^{-T}J^TPs - \beta\Gamma^{-T}Y^T\varepsilon$$

where $\Gamma = \Gamma^T > 0$, $\beta > 0$ and K is chosen as

$$K = \begin{cases} \hat{D}(q) & \text{if } \hat{D}(q) > 0 \\ kI & \text{otherwise.} \end{cases}$$

where $k > 0$, then the closed loop system is globally convergent and stable in the sense that tracking error e and its derivative will converge to zero when t approaches infinity.

CHAPTER - 4
NEURAL NETWORK BASED CONTROLLER

4.1 INTRODUCTION:

Artificial neural systems, or neural networks, are physical cellular systems which acquire, store and utilize experiential knowledge. This knowledge is in the form of stable states or mappings embedded in networks that can be recalled in response to the presentation of signals.

Various algorithms are available for the purpose of tuning of weights. These weights are being tuned in order to get the desired output for a given set of input values. Brief introduction of neural networks has been given in [1]. Since we are using the error back propagation tuning algorithm in this controller, it is being described in steps here.

2.2 ERROR BACK PROPAGATION TUNING ALGORITHM:

Fig ?

The learning of this error back-propagation training algorithm begins with the feedforward recall phase in which the single pattern vector z is given at the input, and the layers' responses y and o are computed. Then the error signal vector is computed and after that propagated towards the network input nodes. The KxJ weights are adjusted within the matrix W which is the weight matrix between hidden layer and the output layer. After that JxI weights are adjusted within the matrix V which are the weights between the input layers and the hidden layers. The final error value for the entire training cycle is calculated after each completed iteration. The learning procedure stops when the final error value below the upper bound E_{max} is obtained.

The algorithm of error back-propagation training is as follows:

Given are P training pairs

0's or 1's or analog value ?

$$\{z_1, d_1, z_2, d_2, \ldots\ldots, z_p, d_p\}$$

where z_i is (I x 1), d_i is (K x 1) and $i = 1,2,\ldots\ldots,p$. Note that the I_{th} component of each z_i is of value -1 since input vectors have been augmented. Size J-1

?

of the hidden layer having output y is selected. Note that J_{th} component of y is of value -1, since hidden layer outputs have been augmented: y is (J x 1) and o is (K x 1). ?

Step 1: $\eta > 0$, E_{max} chosen.

Weights W and V are initialized at small random values: W is (K x J), V is (J x I).

$q \leftarrow 1,\ p \leftarrow 1,\ E \leftarrow 0.$

Step 2: Training step starts here.

Input is presented and the layers' outputs computed.

$z \leftarrow z_p, \qquad d \leftarrow d_p$

$y_j = f(v_j^t z) \qquad$ for j = 1,2,.......,J

where v_j is a column vector, is the j'th row of V, and

$o_k = f(w_k^t y), \qquad$ for k = 1,2,.......,K

where w_k a column vector, is the k'th row of W

and $\quad f(x) = \frac{1}{1 + \exp(-x)}.$?

Step 3: Error value is computed.

$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E \qquad$ for k = 1,2,....,K

Step 4: Error signal vectors δ_o and δ_y of both layers are computed. Vector δ_o is (K x 1) and δ_y is (J x 1). The error signal terms of the output layer in this step are,

$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2) \quad$ for 1,2,........,K

The error signal terms of the hidden layer in this step are:

$\delta_{ij} = \frac{1}{2}(1 - y_j^2) \sum_{k=1}^{K} \delta_{ok} w_{kj} \qquad$ for j = 1,2,....,J

Step 5: Output layer weights are adjusted.

$w_{kj} \leftarrow w_{kj} + \eta \delta_{ok} y_{kj}, \qquad$ for k = 1,2,.....,K

and j = 1,2,.....,J

Step 6: Hidden layer weights are adjusted.

$v_{ji} \leftarrow v_{ji} + \eta \delta_{yj} z_i \qquad$ for j = 1,2,.....,J

and i = 1,2,.....,I

Step 7: If $p > P$ then,

$p \leftarrow p + 1, \quad q \leftarrow q + 1$ and go to step 2, otherwise go to step 8.

Step 8: The training cycle is completed.

For $E > E_{max}$, terminate the training session.

Output weights W, V, q, and E.

If $E > E_{max}$, then $E \leftarrow 0$, $p \leftarrow 1$ and initiate the new training cycle by going to step 2.

Here, it is observed that although network is non-linear in the feedforward mode, the error back-propagation is computed using the linearized activation function. The linearization is achieved by extracting the slope f'(net) at each neuron's operating point and using it for back-transmitted error signal scaling.

The f'(net) is given by the sigmoidal function,

$$f'(net) = \frac{1}{1 + \exp(-net)}$$

4.3 ROBOT ARM DYNAMICS:

The dynamic equations of motion of an n-link robot manipulator is expressed in Lagrange form as

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) = \tau \qquad ....(4.1)$$

where q(t) is the joint variable vector, M(q) is the inertia matrix, $V_m(q,\dot{q})$ is the Coriolis and centrifugal matrix, G(q) is the gravity vector and $F(\dot{q})$ is the friction. The $\tau(t)$ is the control input torque.

Taking the desired arm trajectory as $q_d(t)$, the tracking error is given by,

$$e = q_d(t) - q(t) \qquad ....(4.2)$$

and the filtered tracking error is given by,

$$r = \dot{e} + \Lambda e \qquad .... (4.3)$$

where $\Lambda = \Lambda^T > 0$

Now, the arm dynamics in terms of filtered tracking error may be written as,

$$M\dot{r} = -V_m r - \tau + f \qquad .... (4.4)$$

where the non-linear robot function f(x) is,

$$f(x) = M(q)(\ddot{q}_d - \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d - \Lambda e) + G(q) + F(\dot{q}) \qquad \ldots(4.5)$$

The control input torque is defined as,

$$\tau_0 = \hat{f} + K_v r \qquad \ldots\ldots(4.6)$$

where $\hat{f}(x)$ is an estimate of f(x) and $K_v = K_v^T > 0$. Hence the closed loop system becomes,

$$M\dot{r} = -(K_v + V_m)r + \tilde{f} \qquad \ldots\ldots.(4.7)$$

where the functional estimation error is,given by,

$$\tilde{f}(x) = f(x) - \hat{f}(x) \qquad \ldots\ldots(4.8)$$

THe control τ_0 incorporates a proportional-plus-derivative (PD) term as

$$K_v r = K_v(\dot{e} + \Lambda e).$$

4.4 NEURAL NETWORK BASED CONTROLLER:

The three layer neural net structure which have been considered here is shown in fig.(4.1). Given $x \in R^{n1}$ where R^n denote the real n-vectors, the net output is given by,

$$y_i = \sum_{j=1}^{N_2} [w_{ij}\sigma [\sum_{k=1}^{N_1} v_{jk}x_k + \theta_j] + \theta_{wi}] \qquad \ldots\ldots(4.9)$$

where $\sigma(.)$ is the activation function, V_{jk} is the first to second layer interconnection weights and w_{ij} is the second to third layer interconnection weights. The θ_{vm}, θ_{wm}, $m = 1,2,\ldots$ are the threshold offsets and N_2 is the number of hidden layer neurons.

Equation (4.1) can be expressed in matrix format by defining $x = [x_0,x_1,\ldots\ldots..,x_{N1}]^T$, $y = [y_0,y_1,\ldots\ldots..,y_{N3}]^T$ and weight matrices $W^T = [w_{ij}]$, $V^T = [v_{jk}]$. Including x_0 in x, includes the threshold vector $[\theta_{v1},\theta_{v2},\ldots\ldots..\theta_{vN_2}]^T$ as the first column of V^T so that V^T contains both the weights and the thresholds of the first to second layer connections. Then,

$$y = W^T\sigma(V^Tx) \qquad \ldots\ldots(4.10)$$

Typical selection of the activation function $\sigma(.)$ with $z \in R$ is,

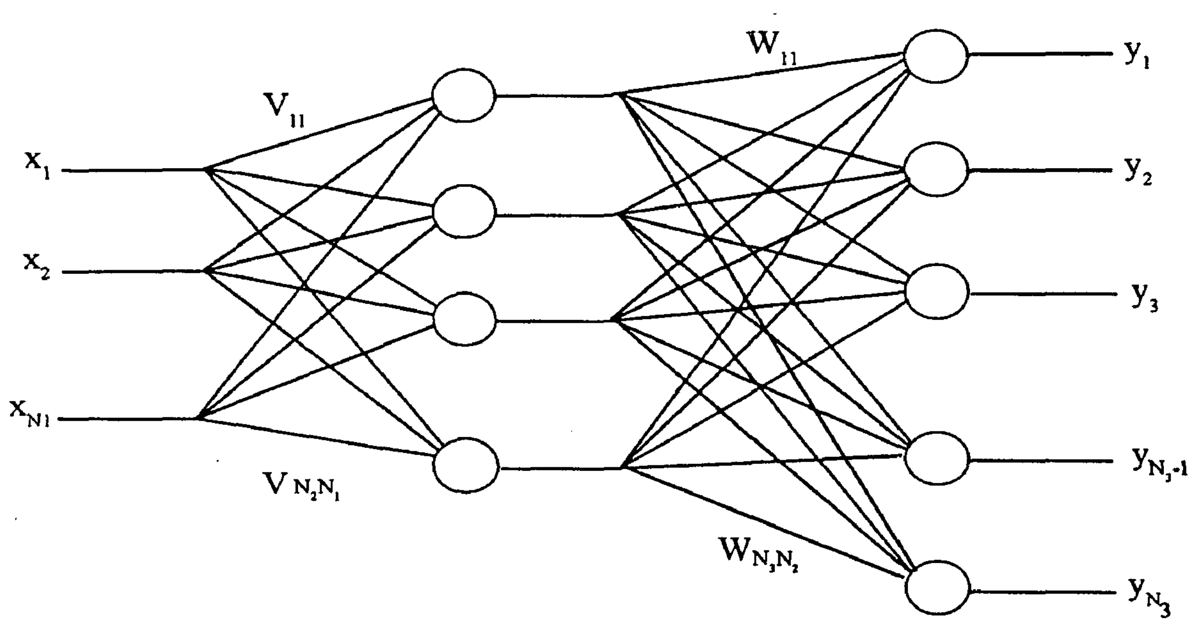


Fig. 4.1 Three layer neural net structure

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \qquad \text{.........(4.11)}$$

which is a sigmoidal function.

As a first to bridging the gap between adaptive control and the NN control, the case of fixed V is taken, This makes the NN linear in the parameters. Defining,

$$\phi(x) = \sigma(V^T x) \qquad \text{........(4.12)}$$

the net output is given by,

$$y = W^T \phi(x) \qquad \text{........(4.13)}$$

Assuming the existence of constant ideal weights W, THe robot function eq.(4.5) is written as,

$$f(x) = W^T \phi(x) + \varepsilon(x) \qquad \text{........(4.14)}$$

where $\phi(x)$ is the basis function as defined earlier and $\varepsilon(x)$ is a known value of bounded error.

Defining the NN functional estimate by,

$$f(x) = \hat{W}^T \phi(x) \qquad \text{.........(4.15)}$$

with $\hat{W}$ the current value of the NN weights provided by the tuning algorithm, the weight deviation or weight estimation error is given by,

$$\tilde{W} = W - \hat{W} \qquad \text{..........(4.16)}$$

Here, the control input is selected as,

$$\tau = \hat{W}^T \phi(x) + K_v r \qquad \text{..........(4.17)}$$

From above, the closed loop filtered error dynamics become,

$$M\dot{r} = -(K_v + V_m)r + \tilde{W}^T \phi(x) \qquad \text{...........(4.18)}$$

Solving eq.(4.18) gives the value of the filtered tracking error r. Using eq.(4.3), we get the value of tracking error e.

THe filtered tracking error is minimized using the neural net structure which further results in the minimization of the tracking error e. After executing the process, we get the value of the actual trajectory q(t) by eq.(4.2) as,

$$q(t) = q_d(t) - e \qquad \text{.......(4.19)}$$

The value of $\hat{W}$ is obtained by using the back propagation tuning algorithm. Since the control input for eq.(4.1) is given by eq.(4.17), the weight tuning is provided by,

$$\hat{W} = F\phi r^T \qquad \text{........(4.20)}$$

and the dynamics of $\tilde{W}$ is given [1] as,

$$\tilde{W} = -F\phi r^T \qquad \text{........(4.21)}$$

where $F = F^T > 0$ is a constant matrix. The tracking error r(t) goes to zero with time and the weight estimation W is bounded.

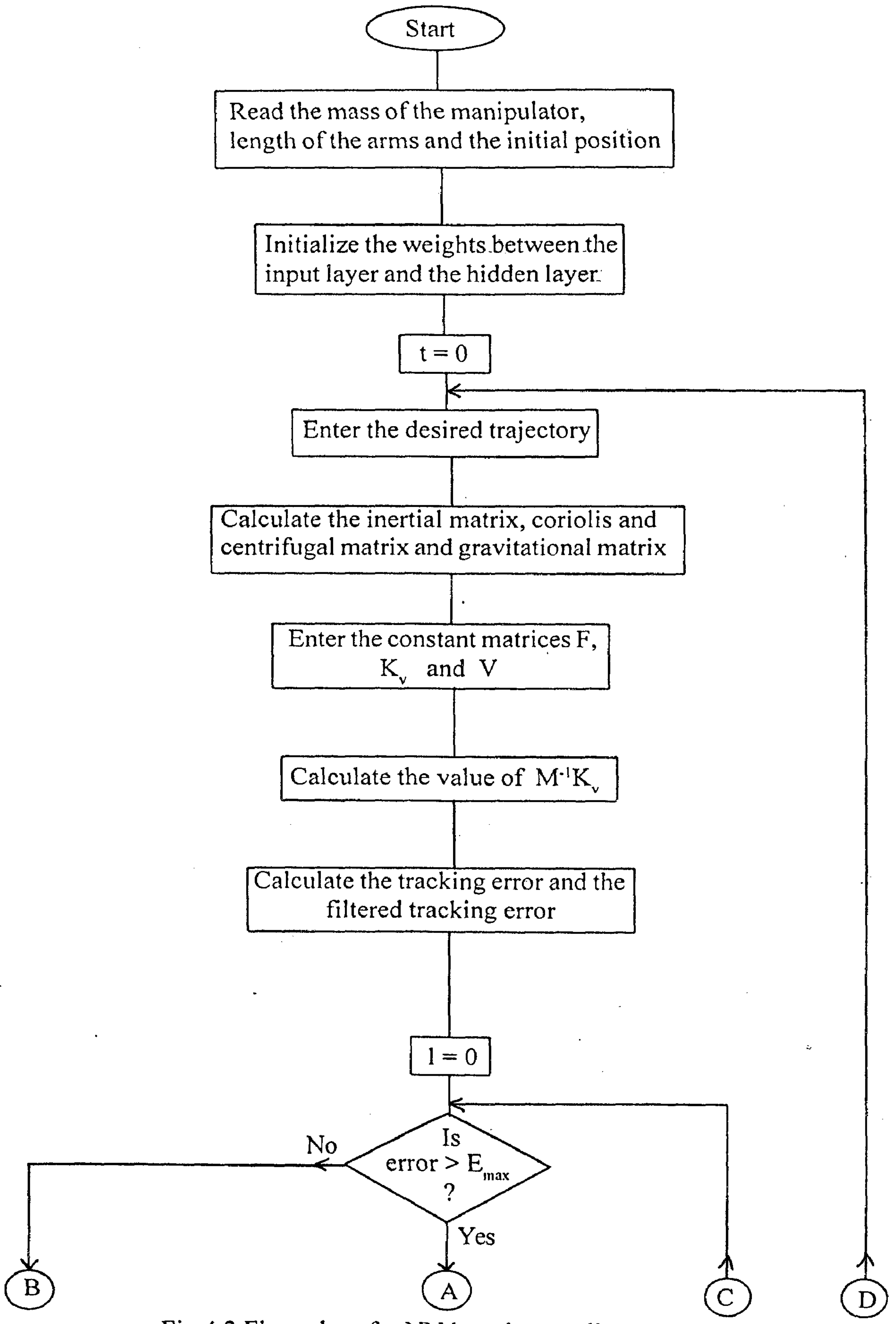
4.5 PROCEDURE AND SOFTWARE USED:

The software for the neural network based controller is written in "C" language in a DOS environment. The flow-chart of the complete algorithm is given in fig.4.2. The details of the procedure according to the flow-chart is given below:

Five input nodes are taken in this neural net controller as per the requirements. It is for the displacement, velocity and acceleration of the robot links. The other two inputs are the tracking error and its first time derivative. These five variables are used in eq.(4.5) and hence are taken. Three hidden layer nodes and two output nodes for the two links of the robot arm are also taken.

The weights between the input layer and the hidden layer are randomly chosen and kept constant. Keeping it constant without tuning is according to the algorithm given which is already discussed earlier. Then the inertial matrix, Coriolis and centrifugal matrix, and gravitational matrix is calculated. Here, the zero friction is assumed. The external disturbance is also taken as zero. After these calculations, the error between the actual position and the desired psition is calculated. Using this error value, the filtered tracking error is calculated. If this tracking error is not less than the maximum permitted error then the tuning process has been done The value of weight estimation error is calculated by using eq.(4.21) and it is used in eq.(4.18) to minimize the value of filtered tracking error. The value of error is then calculated using eq.(4.19).

This is how the desired trajectory is tracked. The physical parameters of the robot arm which is used here is given in Appendix C and the results are discussed in



Start
Read the mass of the manipulator, length of the arms and the initial position
Initialize the weights between the input layer and the hidden layer.
t = 0
Enter the desired trajectory
Calculate the inertial matrix, coriolis and centrifugal matrix and gravitational matrix
Enter the constant matrices F, K_v and V
Calculate the value of M^-1K_v
Calculate the tracking error and the filtered tracking error
l = 0
Is error > E_max ?
No
Yes
B
A
C
D

Fig 4.2 Flow-chart for NN based controller

Continue...

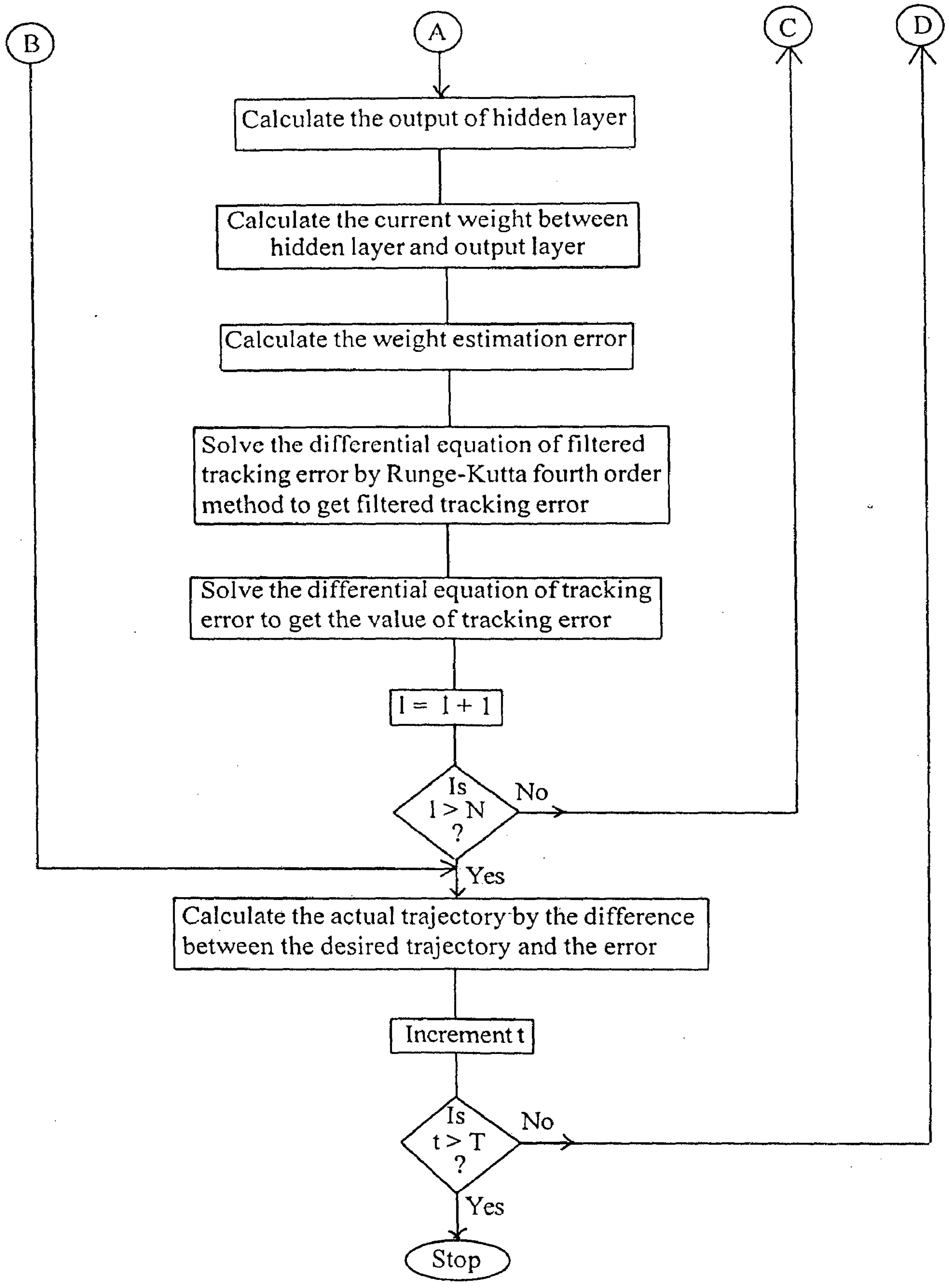

A
B
C
D
Calculate the output of hidden layer
Calculate the current weight between hidden layer and output layer
Calculate the weight estimation error
Solve the differential equation of filtered tracking error by Runge-Kutta fourth order method to get filtered tracking error
Solve the differential equation of tracking error to get the value of tracking error
I = I + 1
Is I > N ?
No
Yes
Calculate the actual trajectory by the difference between the desired trajectory and the error
Increment t
Is t > T ?
No
Yes
Stop

Fig 4.2 Flow-chart for NN based controller

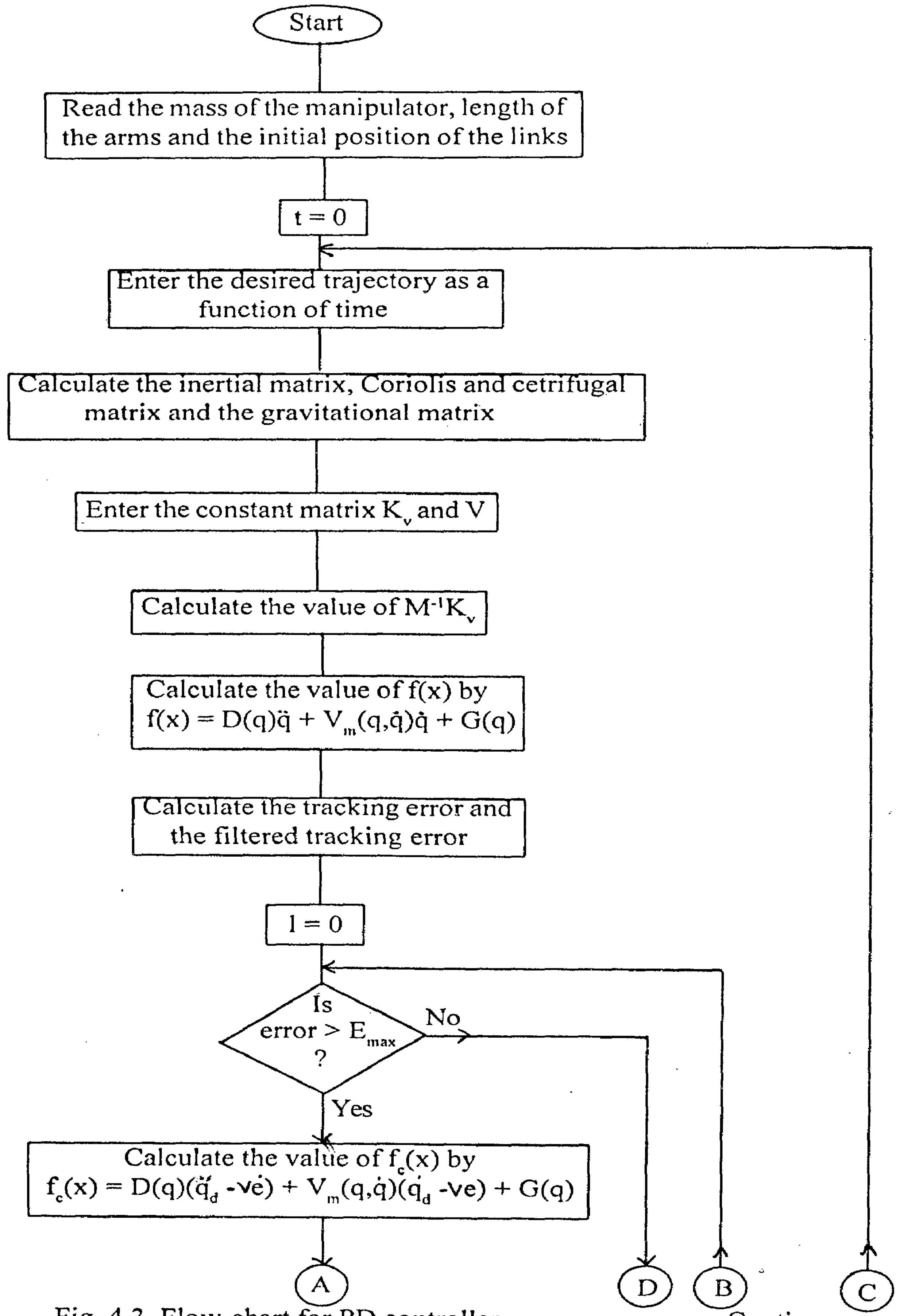

Start
Read the mass of the manipulator, length of the arms and the initial position of the links
t = 0
Enter the desired trajectory as a function of time
Calculate the inertial matrix, Coriolis and cetrifugal matrix and the gravitational matrix
Enter the constant matrix K_v and V
Calculate the value of $M^{-1}K_v$
Calculate the value of f(x) by $f(x) = D(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q)$
Calculate the tracking error and the filtered tracking error
l = 0
Is error > E_{max} ?
No
Yes
Calculate the value of $f_c(x)$ by $f_c(x) = D(q)(\ddot{q}_d - v\dot{e}) + V_m(q,\dot{q})(\dot{q}_d - ve) + G(q)$
A
D
B
C

Fig. 4.3 Flow-chart for PD controller

Continue...

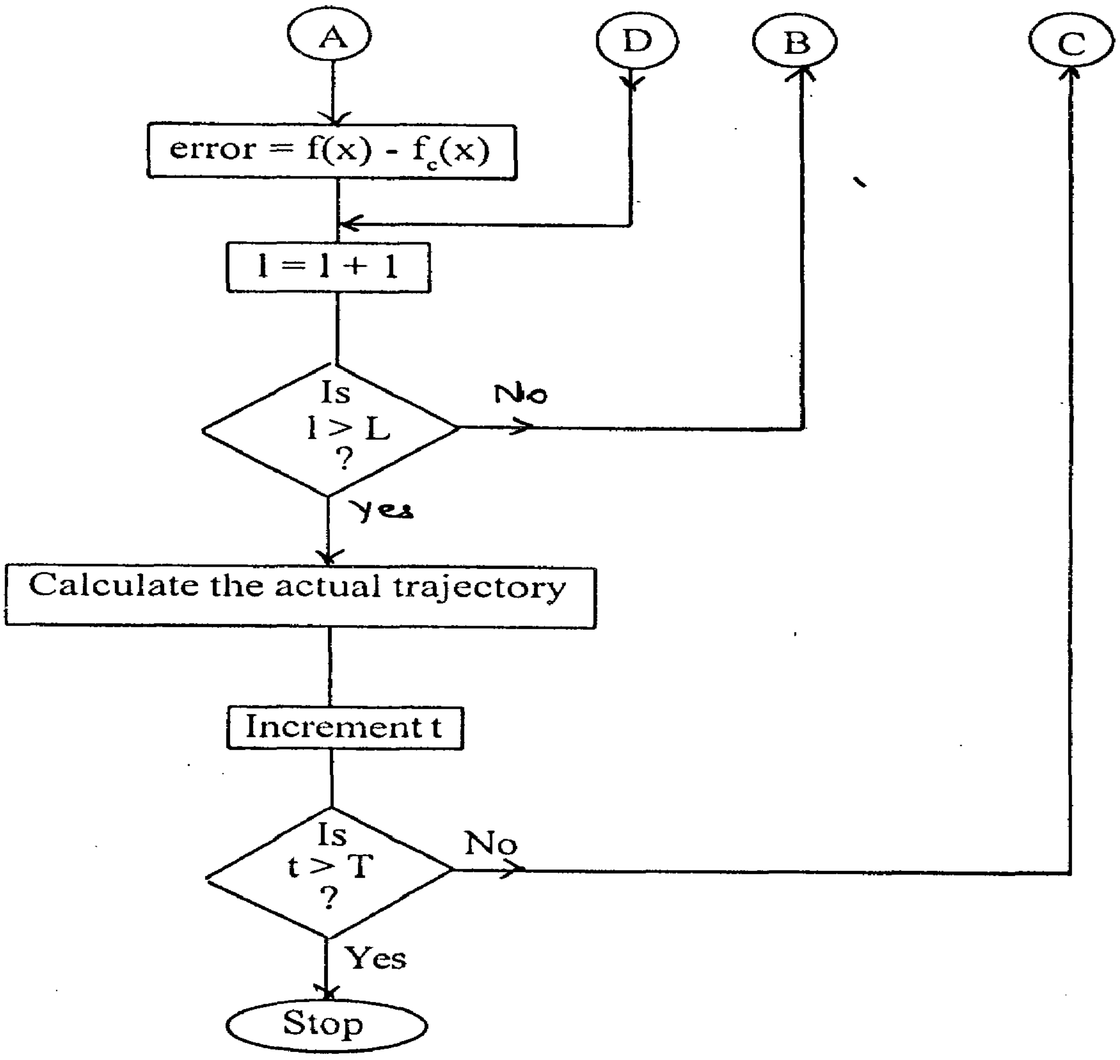


Fig. 4.3 Flow-chart of PD controller

chapter 6.

4.6 PD CONTROLLER:

For the comparison purpose, a PD controller has also been developed using the robot arm dynamics described in section 4.3. The flow-chart of this algorithm is given in fig.4.3 and its results are discussed in chapter 6.

CHAPTER - 5
NEURAL NETWORK BASED ADAPTIVE CONTROLLER

5.1 INTRODUCTION:

An efficient implementation of a neural network based strategy for the on-line adaptive control of the robot manipulator centers on the rapidity of the convergence of the training scheme used for learning the system dynamics. For facilitating this requirement, a new multilayer neural network structure proposed by [2] is used here which includes dynamical nodes in the hidden layer.

The procedure employed, uses three-layer neural network structure with a hidden layer of dynamical nodes together with a simple distributed updating rule. Since a very different updating rule is being used, the manipulator dynamics and control problem is discussed in brief here again according to this new algorithm. It is given in section 5.2. THe three layer neural network structure and its learning scheme is discussed in section 5.3 and section 5.4 discusses the implementation of this neural network in the manipulator dynamics.

5.2 MANIPULATOR DYNAMICS AND THE CONTROL PROBLEM:

In this section, the introduction of the model for the multijointed manipulator which is used for illustrating the control design, is given.

The equation of motion for a general n-degrees of freedom rigid manipulator as derived from Lagrange formulation is expressed [5] in the form,

$$u_i = \sum_{j=1}^{n} H_{ij}(y)\ddot{y}_j + \sum_{j=1}^{n}\sum_{l=1}^{n} C_{ijl}(y)\dot{y}_j\dot{y}_l + G_i(y) \qquad \ldots(5.1)$$

$$i = 1,2,\ldots\ldots,n$$

or in matrix form is expressed as,

$$u = H(y)\ddot{y} + C(y,\dot{y}) + G(y) \qquad \ldots\ldots(5.2)$$

where $y = [y_1,y_2,\ldots\ldots,y_n]^T \in R^n$ is the generalized coordinate vector, n is the number of joints, $u = [u_1,u_2,\ldots.,u_n]^T \in R^n$ is the vector of external forces, $H(y) \in R^{nxn}$ is the positive definite and symmetric inertia matrix, $c(y,\dot{y}) \in R^n$ is the coriolis and centripetal forces vector and $G(y) \in R^n$ is the gravity forces vector.

A discrete method of solving the non-linear dynamics for the purpose of controller design has been given in [2]. This discretized model is given by,

$$y_i(k+1) = \phi_i(y_a(k), \dot{y}_b(k), u_a(k)) + \psi_i(y_c(k), \dot{y}_d(k), u_c(k))u_i(k) \qquad .....(5.3)$$

where, $y_a(k) = [y(k)y(k-1)..........y(k-m_1)]^T$

$\dot{y}_b(k) = [\dot{y}(k)\dot{y}(k-1)..........\dot{y}(k-m_2)]^T$

$u_a(k) = [u(k-1)u(k-2)........u(k-m_3)]^T$

$y_c(k) = [y(k)y(k-1)..........y(k-m_4)]^T$

$\dot{y}_d(k) = [\dot{y}(k)\dot{y}(k-1)..........\dot{y}(k-m_5)]^T$

and $u_c(k) = [u(k-1)u(k-2)........u(k-m_6)]^T$ .....(5.4)

$\phi(.)$ and $\psi(.)$ are the continuous non-linear functions of the arguments u, y and $\dot{y}$. m_1, m_2, m_3, m_4, m_5 and m_6 are appropriate positive integers and is selected according to the requirement of the number of delayed signals given to the controller. The more the number of delayed signals, the more will be the adaptation of the controller according to the varying parameters.

The task is to evaluate the controls $u_i(k)$, $i = 1,2,.....n$ to be applied to individual joints such that it can follow the desired trajectory motion $y_{di}(k)$ of each joint. The objective is to employ the neural-network based approach for the identification of the unknown function $\phi(.)$ and $\psi(.)$ at each time step in order to facilitate the computation of required controls.

5.3 THREE LAYER NEURAL NETWORK WITH LEARNING SCHEME:

Here, the consideration of a three layer network [2] is taken with a hidden layer comprising of dynamical nodes for approximating the mapping between the input vector $z(k) = [z_1(k), z_2(k), ..., z_p(k)]^T$ and the corresponding scalar output $y(k)$.

The dynamics of the network architecture shown in fig. 5.1 are described by,

$$\dot{x} = -x + Wg(x) + Bz(k) \qquad ....(5.4)$$

and $$y(k) = h^T x^* \qquad ....(5.5)$$

where $x \in R^q$ and g(.) is a vector valued function with sigmoidal elements. W $\in$

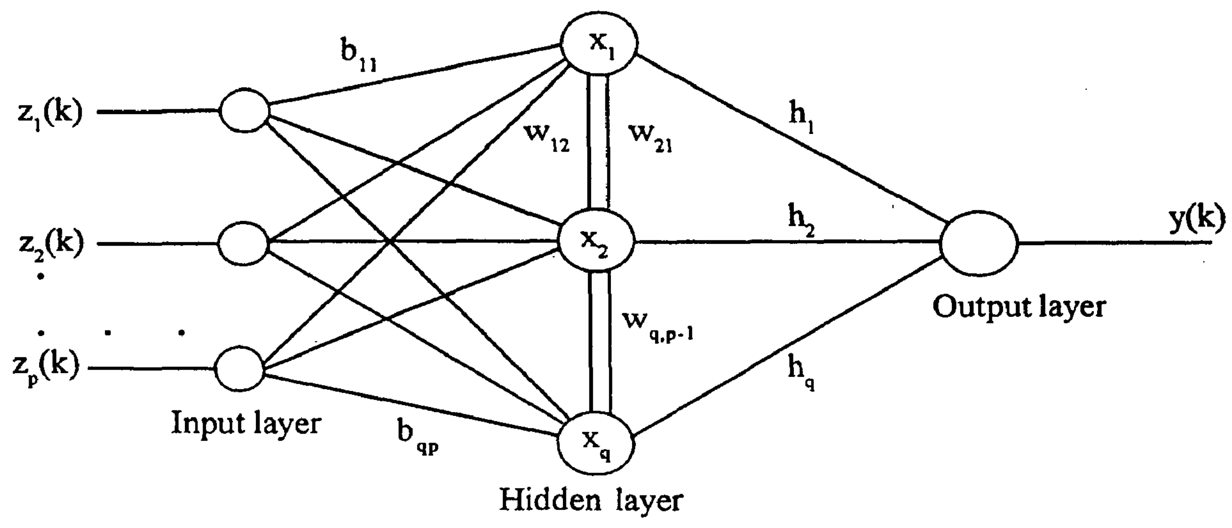


Fig. 5.1 Architecture of the dynamic neural network

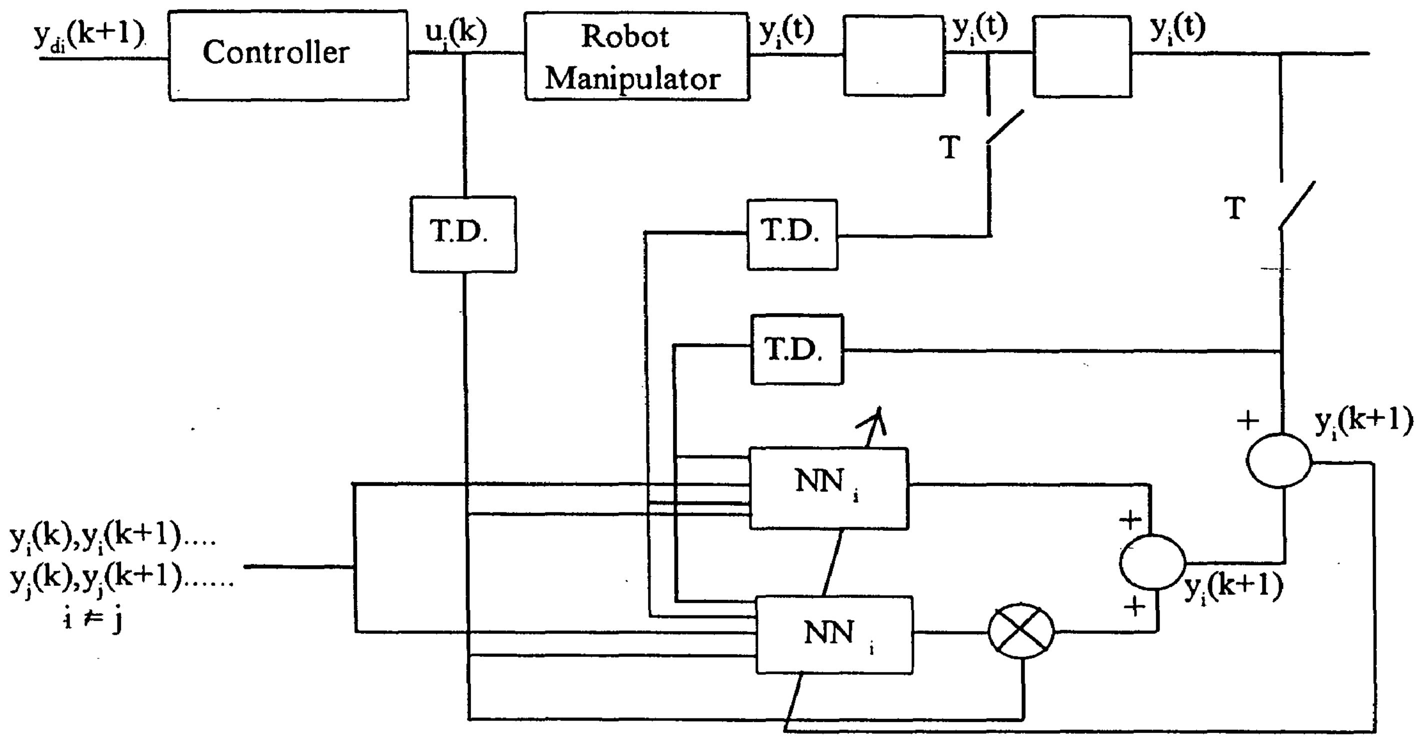


Fig. 5.2 Identification and control architecture

R^{qxq}, $B \in R^{qxp}$ and $h \in R^q$ specify the interconnection weights. In eq.(5.5), x^* denotes the stable equilibrium of eq.(5.4) for the input pattern presentation z(k) at instant k. Under the attainment of steady state conditions, (5.4) and (5.5) is replaced by,

$$x^* = Wg(x^*) + Bz(k) \quad \text{.......(5.6)}$$

$$y(k) = h^T x^* \quad \text{.......(5.7)}$$

or equivalently,

$$x_i^* = \sum_{j=1}^{q} w_{ij} g_j(x_j^*) + \sum_{l=1}^{p} b_{il} z_l(k) \quad \text{...(5.8)}$$

$$y(k) = \sum_{i=1}^{q} h_i x_i^* \quad \text{...(5.9)}$$

where x_i^*, w_{ij}, b_{il}, z_l,and h_i are elements of $x^* \in R^q$, $W \in R^{qxq}$, $g \in R^q$, $B \in R^{qxp}$, $z \in R^p$ and $h \in R^q$ respectively.

The training scheme for minimizing the output error

$$E(k) = \left[y_d(k) - y(k)\right]^2 = \left[y_d(k) - h^T x^*\right]^2 \quad \text{...(5.10)}$$

where $y_d(k)$ is the desired output, is obtained in the form,

$$\left.\begin{aligned} h_i(k+1) &= h_i(k) + \mu_1 \left[y_d(k) - y(k)\right] x_i^* \\ &\qquad i = 1,2,\ldots\ldots, q \\ w_{ij}(k+1) &= w_{ij}(k) + \mu_2 h_i(k) \left[y_d(k) - y(k)\right] g_j(x_j^*) \\ &\qquad i,j = 1,2,\ldots\ldots,q \\ b_{ij}(k+1) &= b_{ij}(k) + \mu_3 h_i(k) \left[y_d(k) - y(k)\right] z_j(k) \\ &\qquad i = 1,2,\ldots,q,\ j = 1,2,\ldots,p \end{aligned}\right\} \quad \text{...(5.11)}$$

where μ_1, μ_2 and μ_3 are appropriately selected updating parameters.

5.4 IDENTIFICATION AND CONTROL ARCHITECTURE:

The objective in this section is to employ the dynamical network given by (5.4) and (5.5) together with the training scheme given by (5.11) for the identification of the functions $\phi_i(.)$ and $\psi_i(.)$ in the manipulator dynamics specified by (5.3) in order to compute the required control $u_i(k)$ for the i_{th} joint.

If the function $\phi_i(.)$ and $\psi_i(.)$ are exactly known, then for tracking the desired output $y_{di}(k+1)$, the required control $u_i(k)$ is computed as,

$$u_i(k) = \frac{y_{di}(k) - \phi_i(k)}{\psi_i(k)} \qquad \text{....(5.12)}$$

However, since there is no prior information of the value of $\phi_i(k)$ and $\psi_i(k)$, a neural network based identification of these parameters are performed by using the model,

$$\hat{y}_i(k+1) = \hat{\phi}_i(.) + \hat{\psi}_i(.)u_i(k) \qquad \text{.....(5.13)}$$

and the architecture shown in fig.(5.2).

NN_{ϕ_i} and NN_{ψ_i} are two distinct neural networks of the type described in (5.4) and (5.5) with adjustable parameters (interconnection weights) used to obtain the approximation of ϕ_i and ψ_i by the function $\hat{\phi}_i$ and $\hat{\psi}_i$ respectively. Employing the learning algorithm specified by (5.11) in order to minimize the error

$$E_i(k+1) = \left[y_i(k+1) - \hat{y}_i(k+1)\right]^2 \qquad \text{...(5.14)}$$

NN_{ϕ_i} and NN_{ψ_i} are trained closely to approximate $\phi_i(.)$ and $\psi_i(.)$ by $\phi_i(.)$ and $\psi_i(.)$ respectively. The control signal $u_i(k)$ is then obtained from,

$$u_i(k) = \frac{y_{di}(k+1) - \hat{\phi}_i(.)}{\hat{\psi}_i(.)} \qquad \text{...(5.15)}$$

for tracking the desired signal $y_{di}(k+1)$.

The blocks that are marked T.D. in fig.(5.2) are tapped delays which give appropriately delayed versions of the signals $u_i(k)$, $y_i(k)$ and $\hat{y}_i(k)$ in order to provide inputs to the the two neural networks.

5.5 PROCEDURE AND SOFTWARE USED:

The software for this controller is designed in "C" language in a DOS environment. The complete flow-chart of the implemented algorithm is shown in fig.5.3. The complete procedure and the explanation of the flow-chart is given as follows:

The value of m_1, m_2, m_3, m_4, m_5 and m_6 are chosen as 2, 2, 0, 2, 2 and 1 respectively. Hence, according to eq.(5.3), seven input nodes are taken for the input

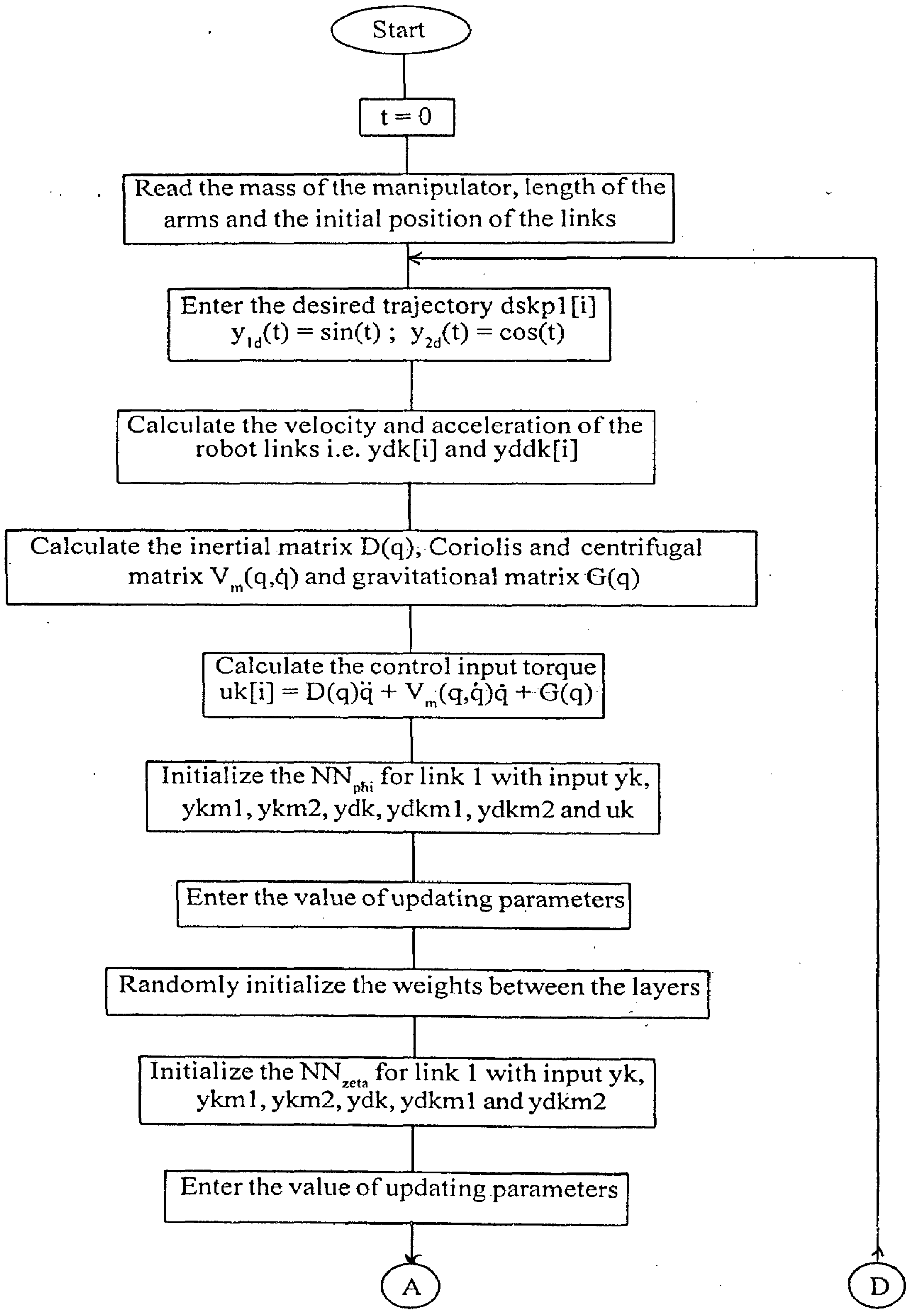

Start
t = 0
Read the mass of the manipulator, length of the arms and the initial position of the links
Enter the desired trajectory dskp1[i] $y_{1d}(t) = \sin(t)$; $y_{2d}(t) = \cos(t)$
Calculate the velocity and acceleration of the robot links i.e. ydk[i] and yddk[i]
Calculate the inertial matrix D(q), Coriolis and centrifugal matrix $V_m(q,\dot{q})$ and gravitational matrix G(q)
Calculate the control input torque $uk[i] = D(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q)$
Initialize the NN_{phi} for link 1 with input yk, ykm1, ykm2, ydk, ydkm1, ydkm2 and uk
Enter the value of updating parameters
Randomly initialize the weights between the layers
Initialize the NN_{zeta} for link 1 with input yk, ykm1, ykm2, ydk, ydkm1 and ydkm2
Enter the value of updating parameters
A
D

Fig. 5.3 Flow-chart for NN based adaptive controller

Continue...

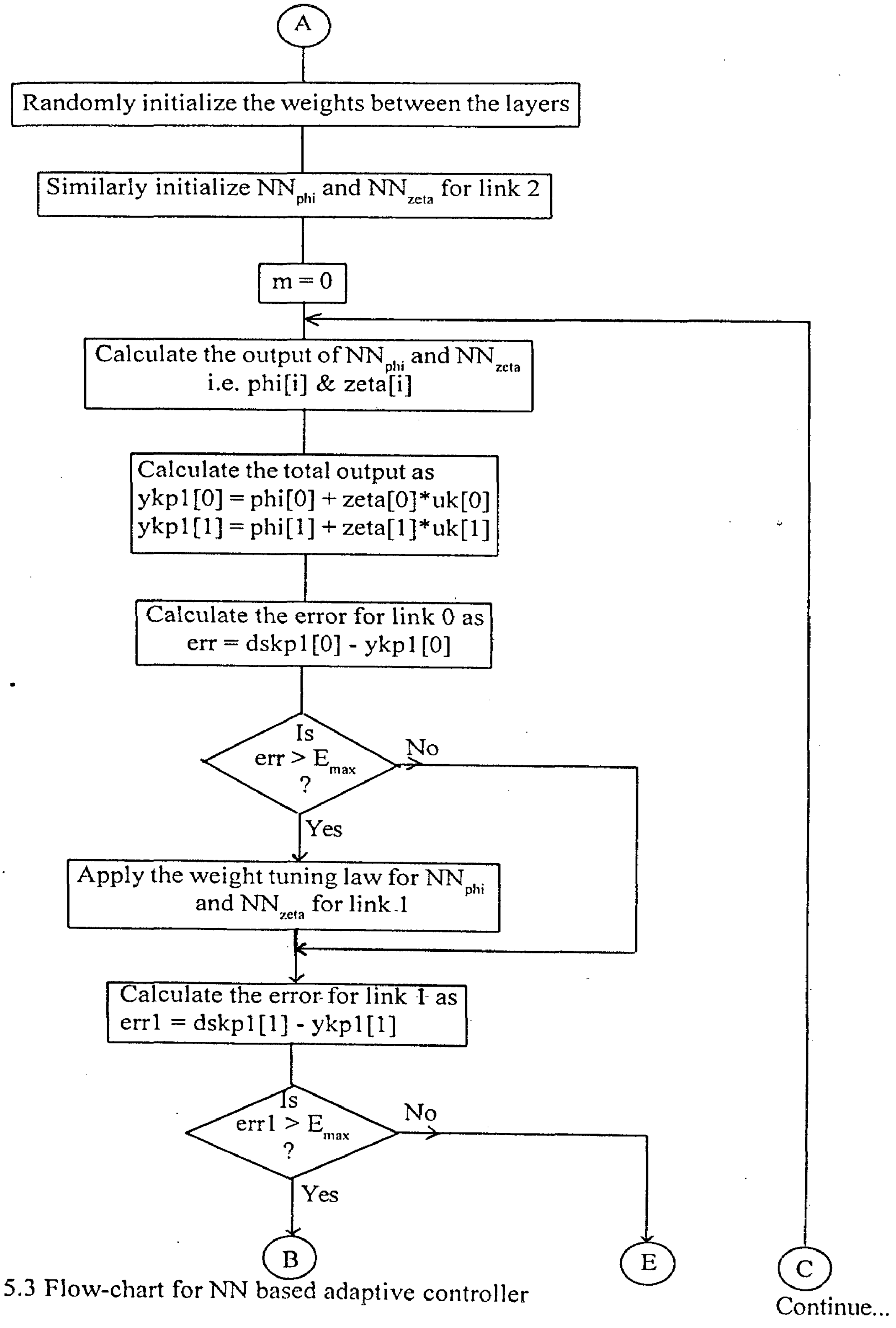
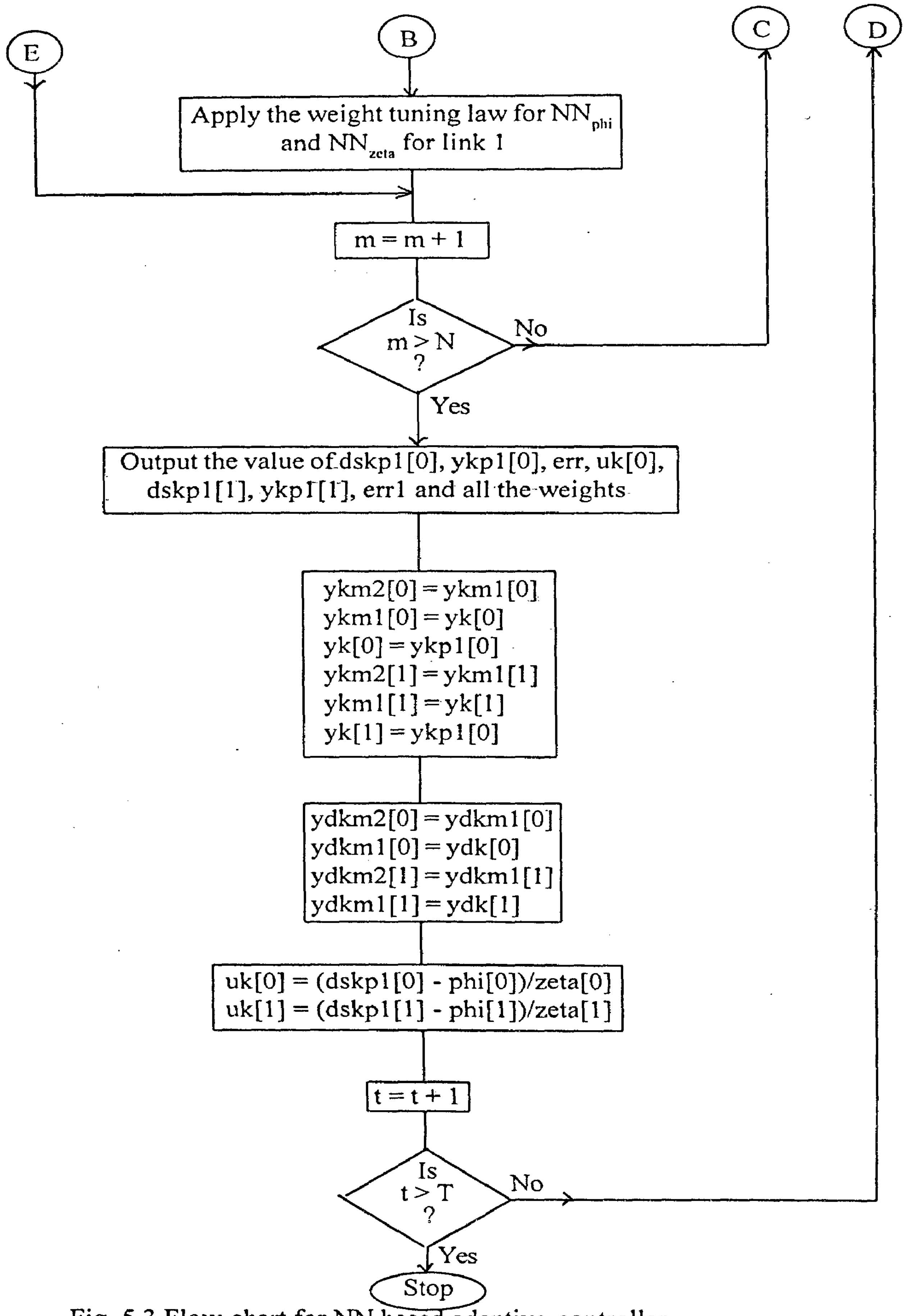

A
Randomly initialize the weights between the layers
Similarly initialize NN_{phi} and NN_{zeta} for link 2
m = 0
Calculate the output of NN_{phi} and NN_{zeta} i.e. phi[i] & zeta[i]
Calculate the total output as ykp1[0] = phi[0] + zeta[0]*uk[0] ykp1[1] = phi[1] + zeta[1]*uk[1]
Calculate the error for link 0 as err = dskp1[0] - ykp1[0]
Is err > E_{max} ?
No
Yes
Apply the weight tuning law for NN_{phi} and NN_{zeta} for link 1
Calculate the error for link 1 as err1 = dskp1[1] - ykp1[1]
Is err1 > E_{max} ?
No
Yes
B
E
C
Continue...

Fig. 5.3 Flow-chart for NN based adaptive controller


E
B
C
D
Apply the weight tuning law for NN_phi and NN_zeta for link 1
m = m + 1
Is m > N ?
No
Yes
Output the value of dskp1[0], ykp1[0], err, uk[0], dskp1[1], ykp1[1], err1 and all the weights
ykm2[0] = ykm1[0]
ykm1[0] = yk[0]
yk[0] = ykp1[0]
ykm2[1] = ykm1[1]
ykm1[1] = yk[1]
yk[1] = ykp1[0]
ydkm2[0] = ydkm1[0]
ydkm1[0] = ydk[0]
ydkm2[1] = ydkm1[1]
ydkm1[1] = ydk[1]
uk[0] = (dskp1[0] - phi[0])/zeta[0]
uk[1] = (dskp1[1] - phi[1])/zeta[1]
t = t + 1
Is t > T ?
No
Yes
Stop
Fig. 5.3 Flow-chart for NN based adaptive controller

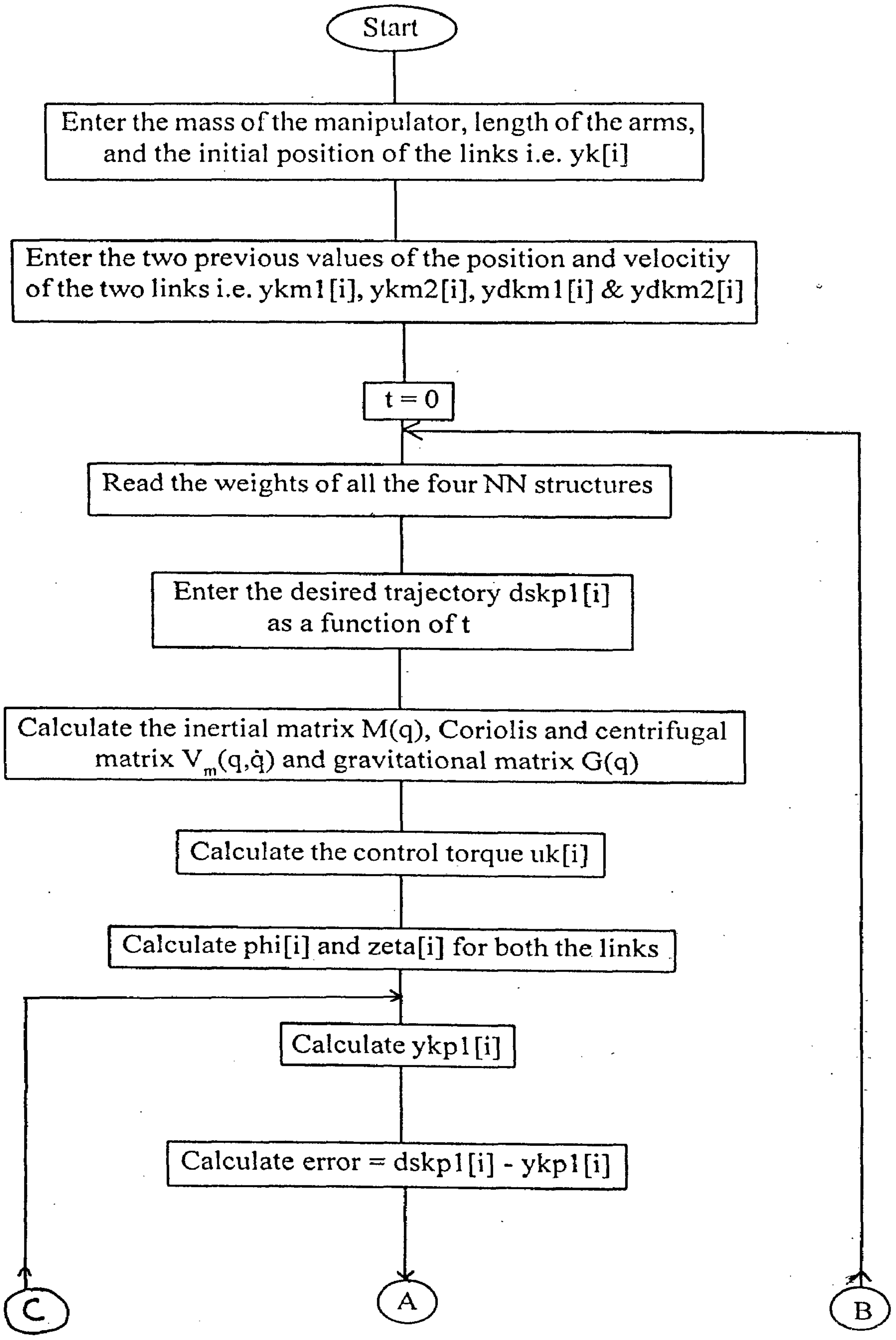

Start
Enter the mass of the manipulator, length of the arms, and the initial position of the links i.e. yk[i]
Enter the two previous values of the position and velocitiy of the two links i.e. ykm1 [i], ykm2[i], ydkm1 [i] & ydkm2[i]
t = 0
Read the weights of all the four NN structures
Enter the desired trajectory dskp1 [i] as a function of t
Calculate the inertial matrix M(q), Coriolis and centrifugal matrix V_m(q,q̇) and gravitational matrix G(q)
Calculate the control torque uk[i]
Calculate phi[i] and zeta[i] for both the links
Calculate ykp1 [i]
Calculate error = dskp1 [i] - ykp1 [i]
C
A
B

Fig. 5.4 Flow-chart for NN based adaptive control in an on-line system

Continue...

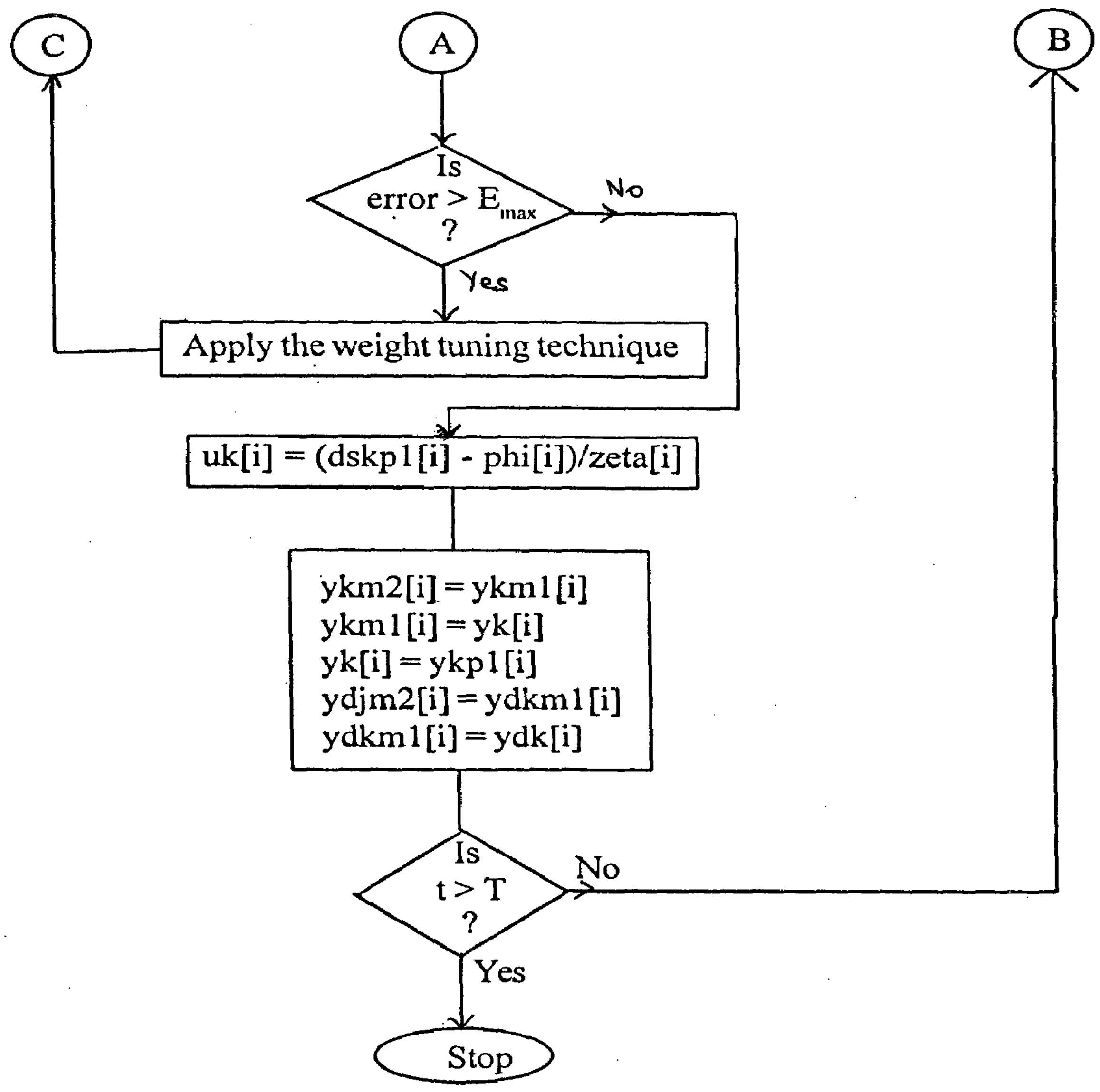


Fig. 5.4 Flow-chart for NN based adaptive control in an on-line system

layer z(k). Two hidden layer nodes x_1 and x_2 are the members of the hidden layer vector x(k) and there is one scalar node y(k).

Four identical neural net structure are made for the approximation of NN_{ϕ_i} and NN_{ψ_i} i.e. NN_{ϕ_0} and NN_{ψ_0} for link 1 and NN_{ϕ_1} and NN_{ψ_1} for link 2. Initially all the weights are randomly chosen between all the layers in the network structure. Then the output of these structures are calculated. The actual output is computed by using eq.(5.13). Difference between the desired value and the actual value of the NN becomes the error value E. This is used for the tuning of the network structure. These network structures are tuned till the error E become less than the maximum permitted error E_{max}.

After the tuning process is completed, all the weights are stored in order to use it in another software which moves the robot on-line. The flow-chart of this software is given in fig.(5.4).

The physical parameters of the robot arm which are taken for the analysis are given in Appendix C and the results are discussed in chapter 6.

CHAPTER - 6
RESULT ANALYSIS, CONCLUSIONS AND FUTURE SCOPE OF WORK

6.1 INTRODUCTION:

In this chapter, the results of the PD controller, neural network based controller and the neural network based adaptive controller are discussed. A comparative study of all these controllers are also given.

A sine wave and a cosine wave are taken as the desired trajectory for link 1 and link 2 respectively. In all the graph that has been given, the desired trajectory is shown by the solid line and the actual trajectory obtained by the controller is shown by the dotted (or dashed) line. The physical parameters of the robot arm that has been taken for the analysis is given in Appendix C. These parameters are kept the same for all the controllers to facilitate the comparison purpose.

MATHEMATICAL COMPUTATIONS:

The dynamic equations of motion of the robot manipulator in matrix form is written as,

[illegible]

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}a_m & b_m\\ c_m & d_m\end{bmatrix}\begin{bmatrix}q_1\\ q_2\end{bmatrix} + \begin{bmatrix}e_m & f_m\\ g_m & h_m\end{bmatrix}\begin{bmatrix}q_1\\ q_2\end{bmatrix} + \begin{bmatrix}i_m\\ j_m\end{bmatrix}$$

The coefficients of the inertial matrix, Coriolis and centrifugal matrix and the gravitational matrix as referred to Appendix C are as follows:

a_m = 4.999805

b_m = 1.666569

c_m = 1.566569

d_m = 0.666667

e_m = -3.350923

f_m = -1.675462

g_m = 0.2094327

h_m = 0.0

$i_m = 34.29875$

$j_m = 9.798791$

Hence, the complete dynamic equation is given as,

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 4.999805 & 1.666569 \\ 1.666569 & 0.666667 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} -3.350923 & -1.675462 \\ 0.2094327 & 0.0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} 34.29875 \\ 9.798791 \end{bmatrix}$$

6.2 PD CONTROLLER:

The PD controller is designed by using the algorithm discussed in chapter 4. The physical parameters taken for the consideration are as given in Appendix C. The desired trajectory is taken as,

$q_{1d}(t) = \sin(t)$

$q_{2d}(t) = \cos(t)$

The desired trajectory and the tracked actual trajectory is shown in fig.6.1. Even though it seems that the actual trajectory overlaps the desired trajectory, it isn't so. The actual trajectory deviates from the desired one by an error in the range of 0.01 for link 1 and 0.02 for link 2. The graph showing the error in these trajectories are shown in fig.6.2(a&b). Since the PD controller is only able to decrease the error upto a certain limit, almost constant error is obtained throughout the robot motion.

6.3 NEURAL NETWORK BASED CONTROLLER:

The algorithm to design this controller is also given in chapter 4. The physical parameters of the robot arm used here is the same as has been used in the PD controller. The desired trajectory of the links are also kept the same.

The tracked actual trajectory and the desired trajectory in this case is shown in fig.6.3. By viewing the graph, it is analyzed that this controller is tracking the desired trajectory more closely than the PD controller. The error in this case is in the range of 0.005 for link 1 and 0.007 for link 2 which is lower than that of the previous controller. This shows that this controller is better than the PD controller for the more precise work. The graph showing the deviation of the actual trajectory

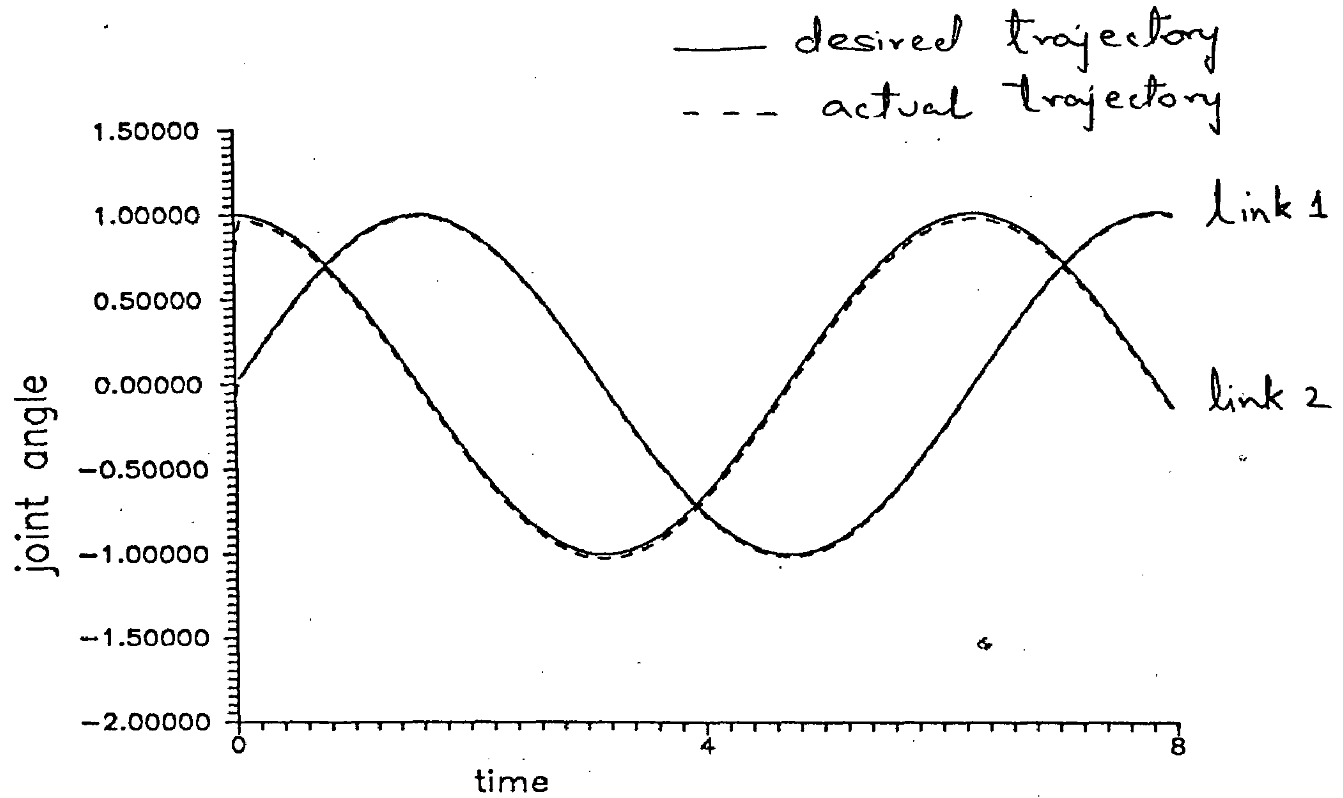


Fig.6.1 Trajectory of links
in the case of PD controller

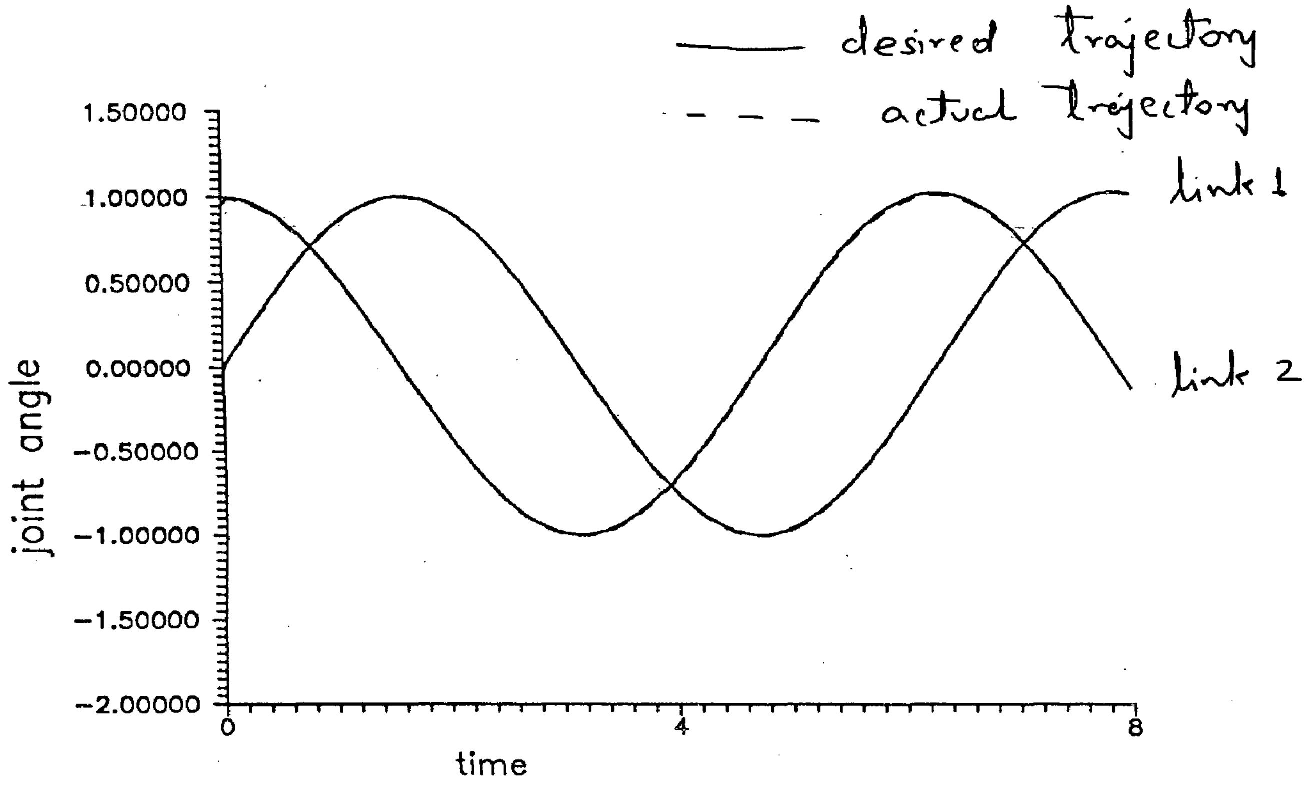


Fig.6.3 Trajectory of links
in the case of NN based controller

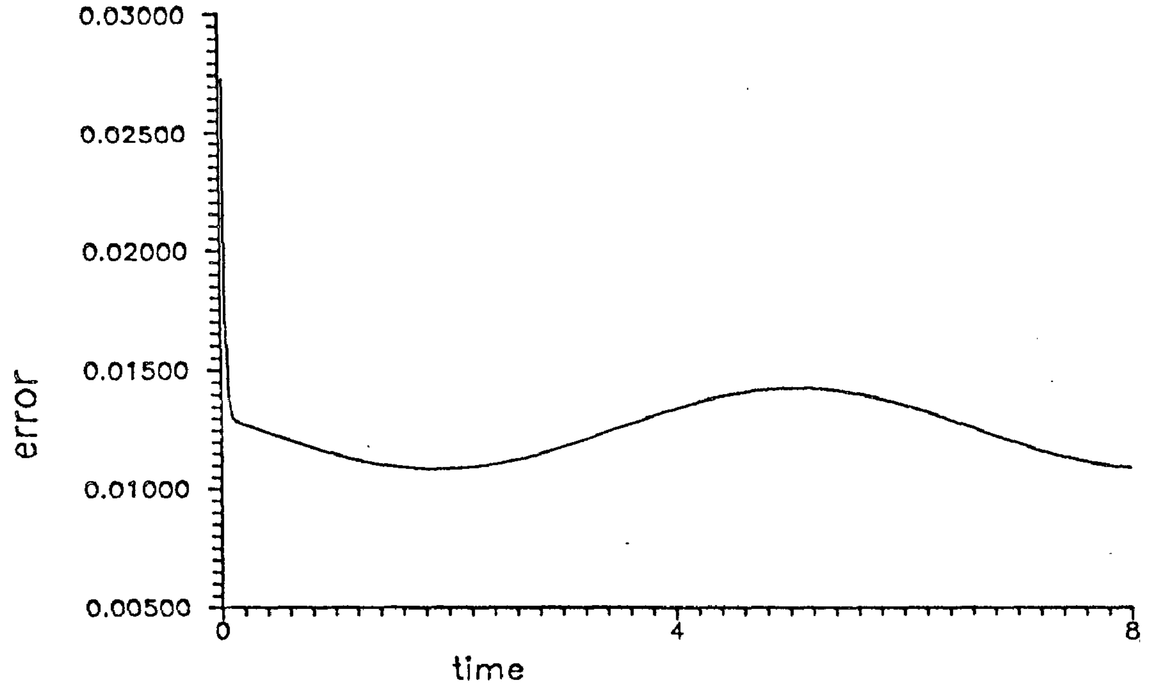


Fig.6.2 a Error in link 1
in the case of PD controller

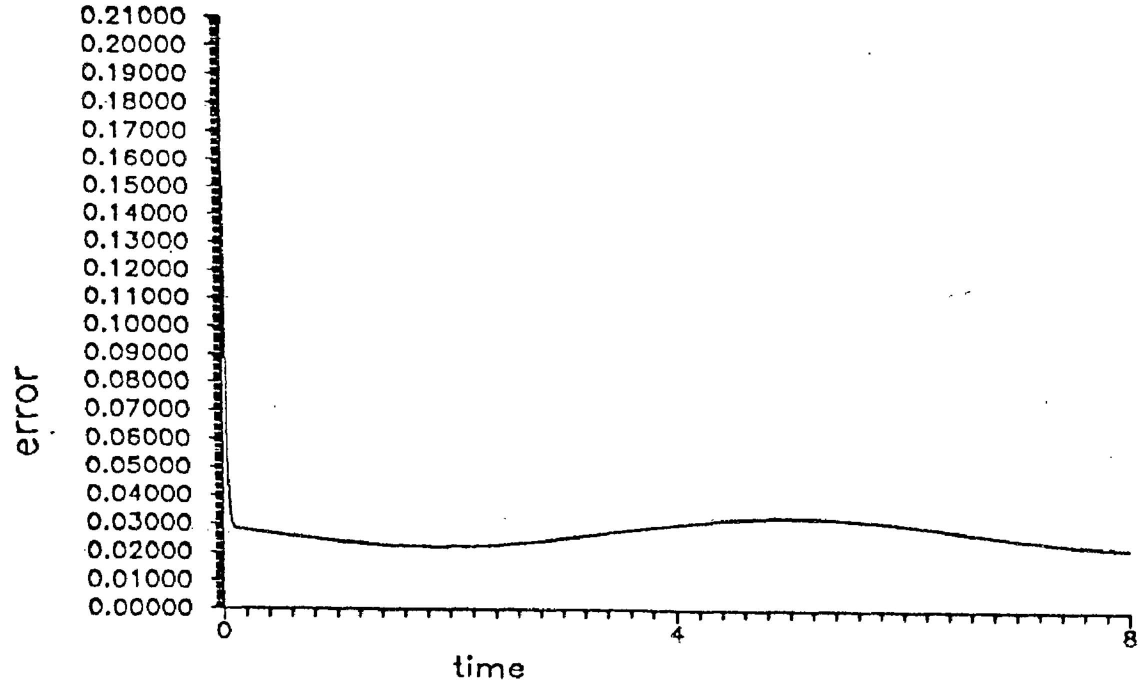


Fig.6.2 b Error in link 2
in the case of PD controller

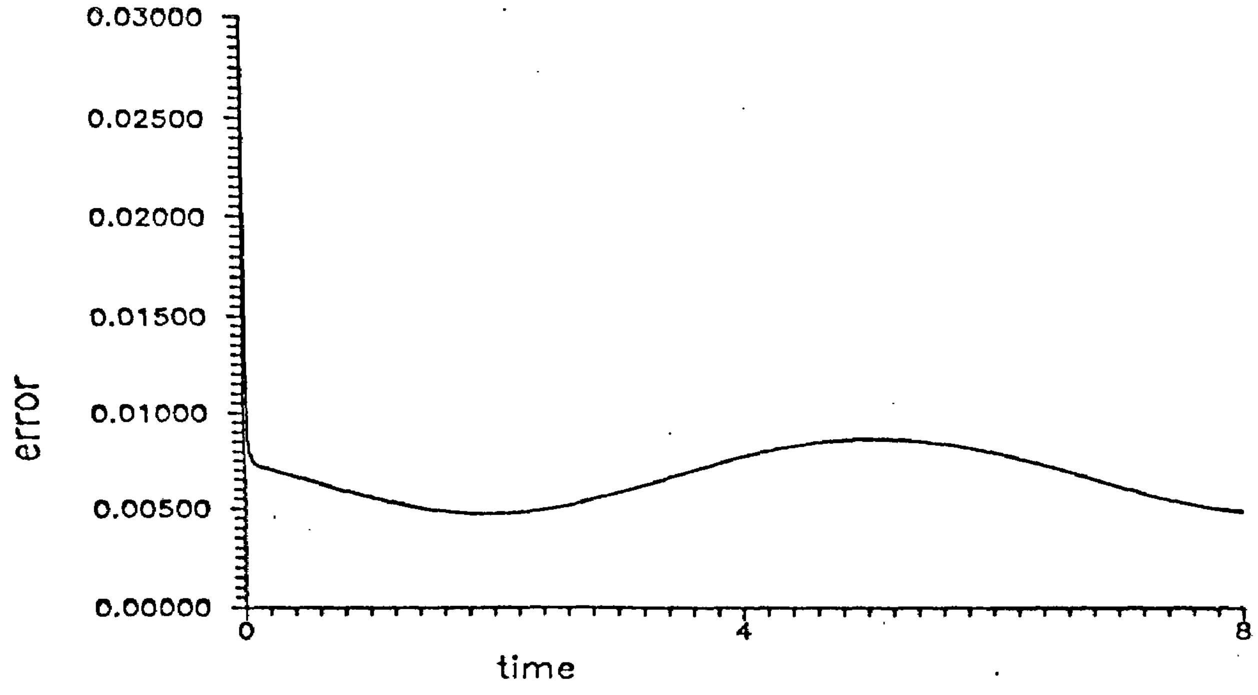


Fig.6.4 a Error in link 1
in the case of NN based controller

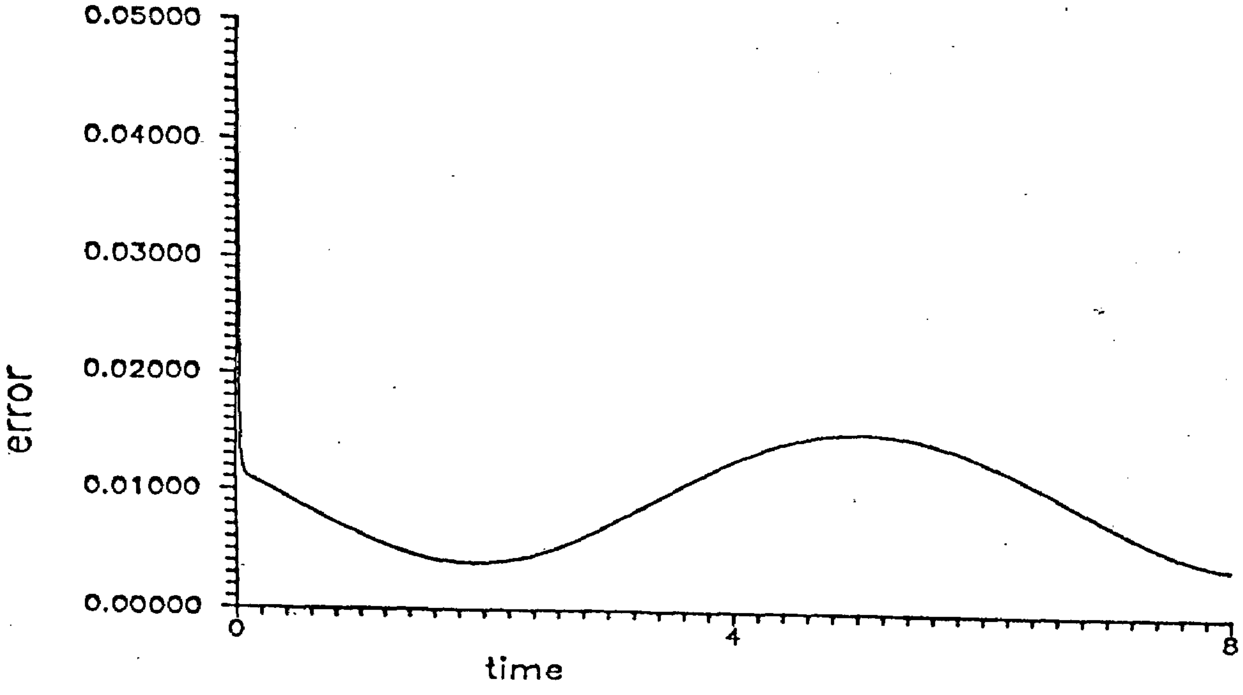


Fig.6.4 b Error in link 2
in the case of NN based controller

from the desired one is shown in fig.6.4(a&b) for both the links.

Fig.6.5 shows the comparative study of PD controller and the NN based controller. It shows the error in the case of both the controllers for tracking the desired trajectory. It is clearly shown that the NN based controller is having less error than the PD controller.

6.4 NEURAL NETWORK BASED ADAPTIVE CONTROLLER:

The algorithm for the design of this controller is given in chapter 5. The physical parameters of the robot arm is again kept the same as that for the two previous controllers. In it, initially the value of y(k-2), y(k-1), y_d(k-2) and y_d(k-1) are kept at zero. All the calculations are done according to the algorithm given. The desired trajectory and the tracked actual trajectory is shown in fig.6.6. This figure shows the representative joint trajectories used for training and also the output of the neural network after training is completed (shown by dotted lines).

In this graph, it is noted that the actual trajectories overlaps the desired trajectories. Here, the deviation of the tracked actual trajectories is in the range from -0.00001 to 0.00001.(means the error is almost zero in this case). This deviation of the actual trajectory from the desired one is shown in fig.6.7(a&b).

A comparative study of all the three controllers is shown in fig.6.8. In it, it is noticed that the error in the case of PD controller is greater than that of the NN based controller and the error of both these controllers are very much higher than the NN based adaptive controller. This shows the superiority of the NN based adaptive controller over all other controllers.

Fig. 9 shows the case when the payload is increased after robot has tracked for 4 secs.

Because of the limitations of the computer being used, the error in the case of on-line NN based adaptive control is little more than the off-line one. This is shown in fig. 10 and fig. 11. Fig. 10 is for link 1 and fig. 11 is for link 2. This error is due to the truncation of the values when it takes it from the data file.

6.5 CONCLUSIONS:

It is confirmed that the neural network based controller for the robot manipulator is very precise and fast in comparison to all other controllers. It is also been noted that the inclusion of the neural network in highly complex dynamics helps us in obtaining a more precise work than the other controllers without it. It gives us the guaranteed tracking performance.

It is also been noted that as the number of iterations in minimizing the filtered tracking is increased, more and more accuracy is obtained.

6.6 FUTURE SCOPE OF WORK:

The neural network based adaptive controller which is developed in this work is only a software which is not been interfaced with the physical robot arm. Taking the idea from this controller, it is possible to develop a software which can be interfaced with the robot arm and can control it on-line. A practical work over it can be done.

With the help of this practical work, we can confirm the real supremacy of NN based adaptive controller over other conventional controllers It can be done by doing a comparative study using both the controllers in turn to control the robot manipulator.

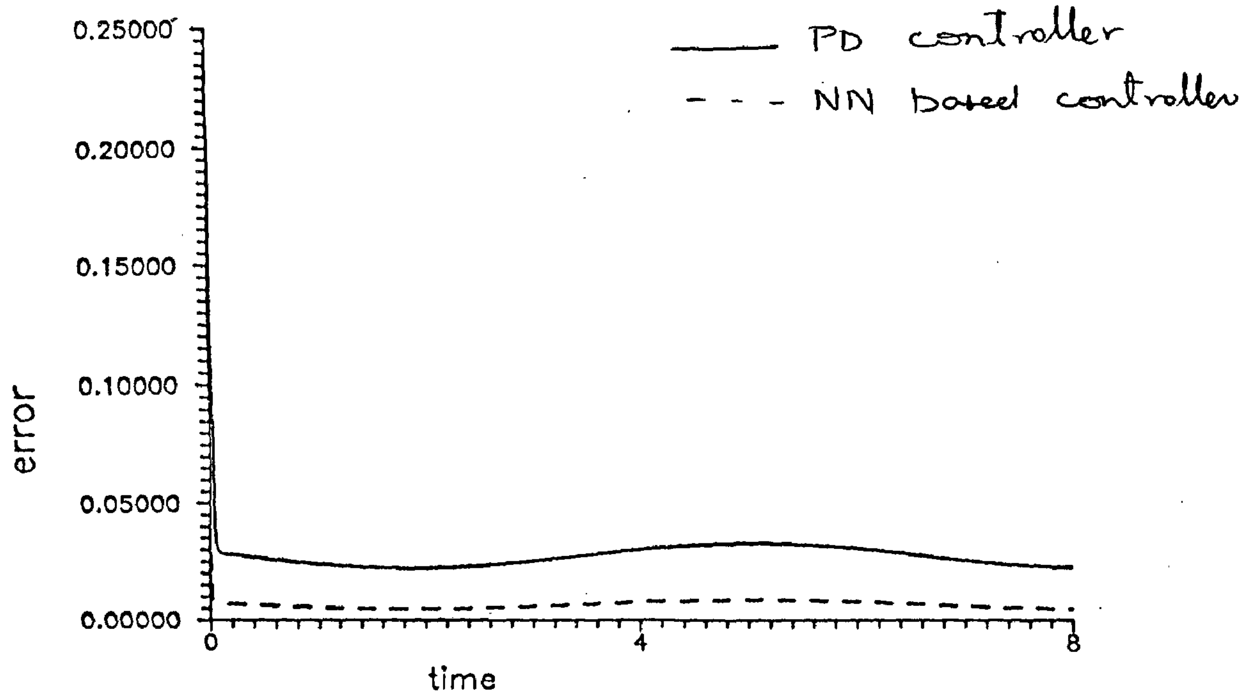


Fig.6.5 Error in links in the case of
PD and NN based adaptive controller

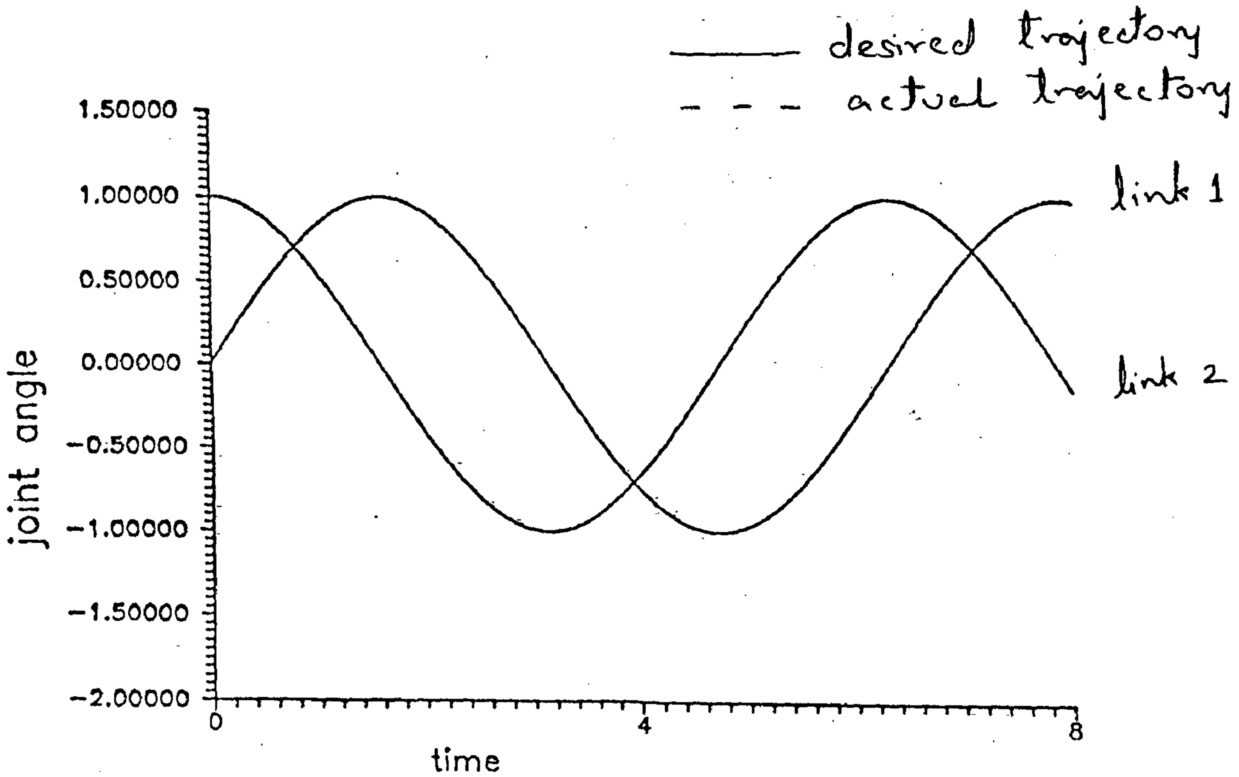
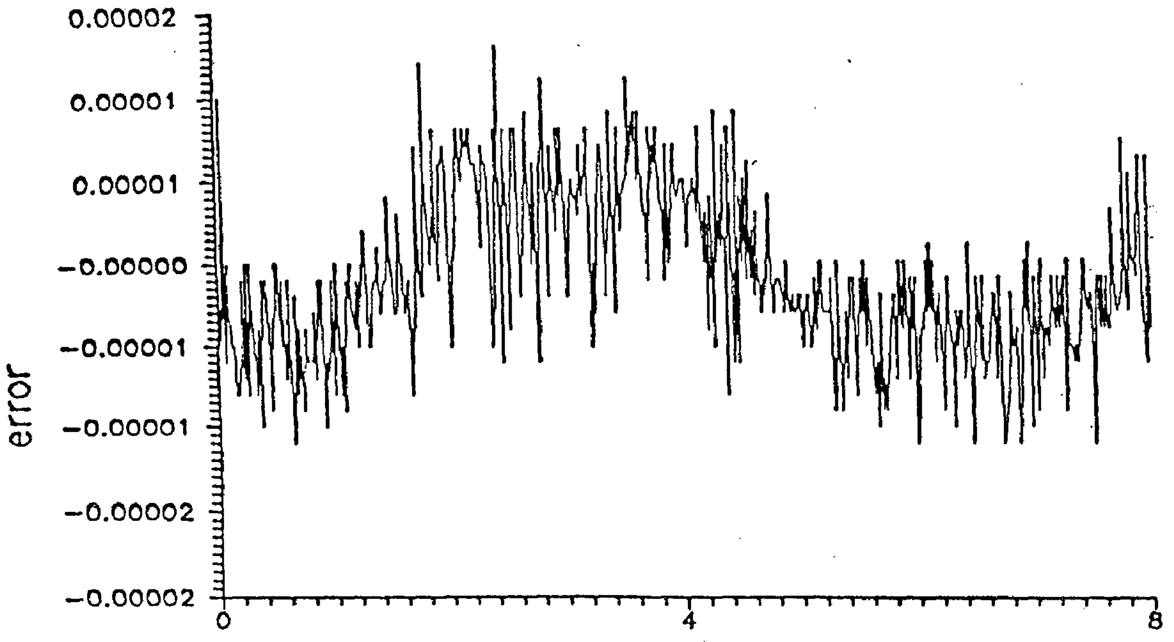
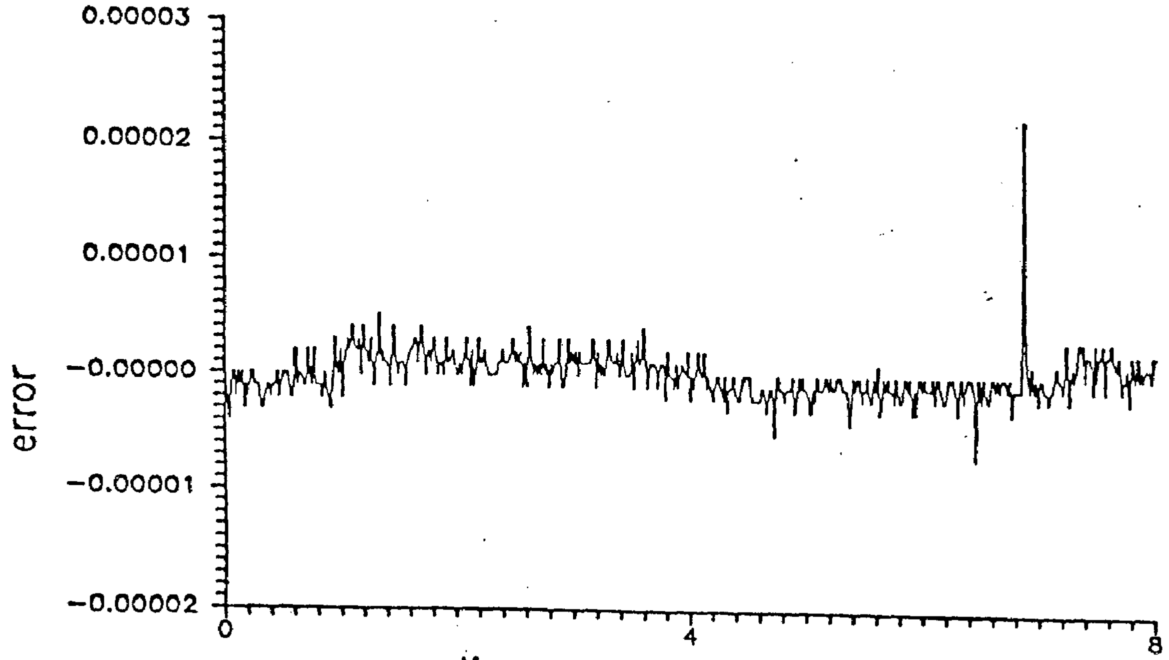


Fig.6.6 Trajectory of links
in the case of NN based adaptive controll



time

Fig.6.7 a Error in link 1
in the case of NN based adaptive controll



time

Fig.6.7 b Error in link 2
in the case of NN based adaptive controll

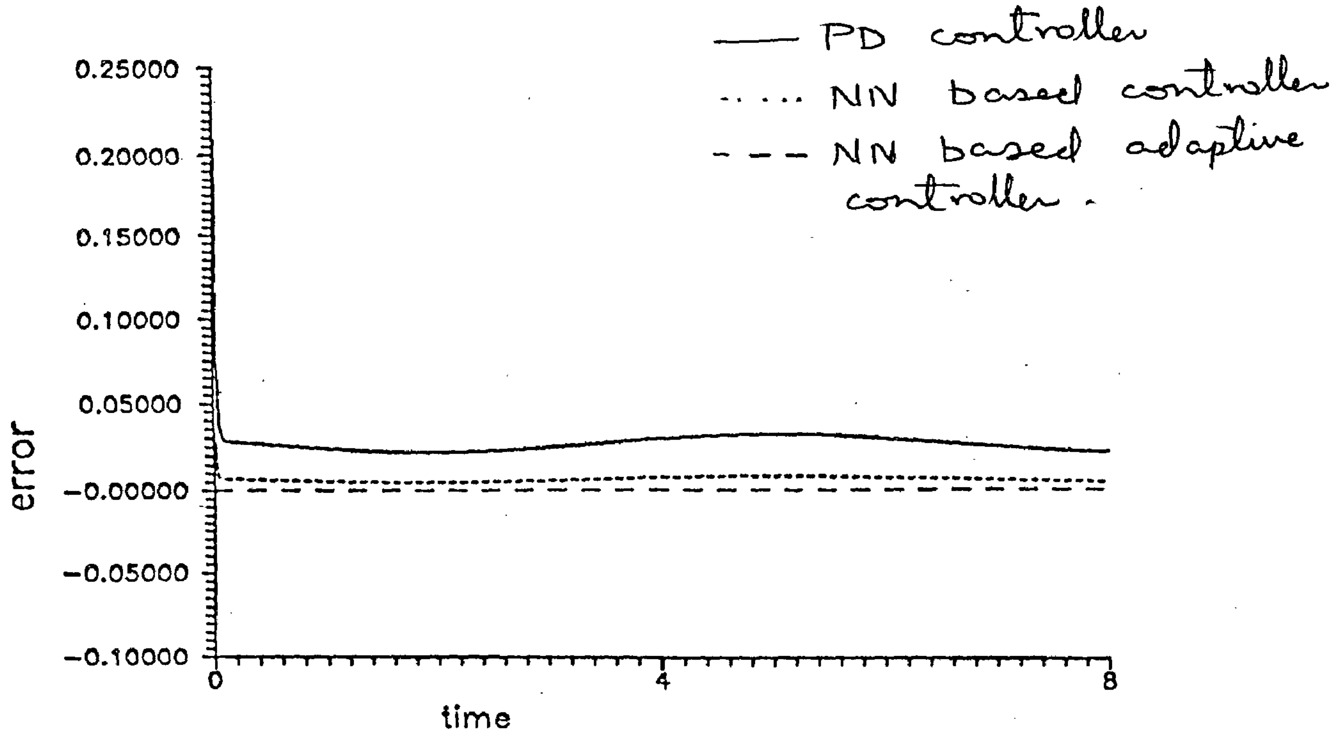


Fig.6.8 Error in links in the case of PD, NN based and NN based adaptive cor

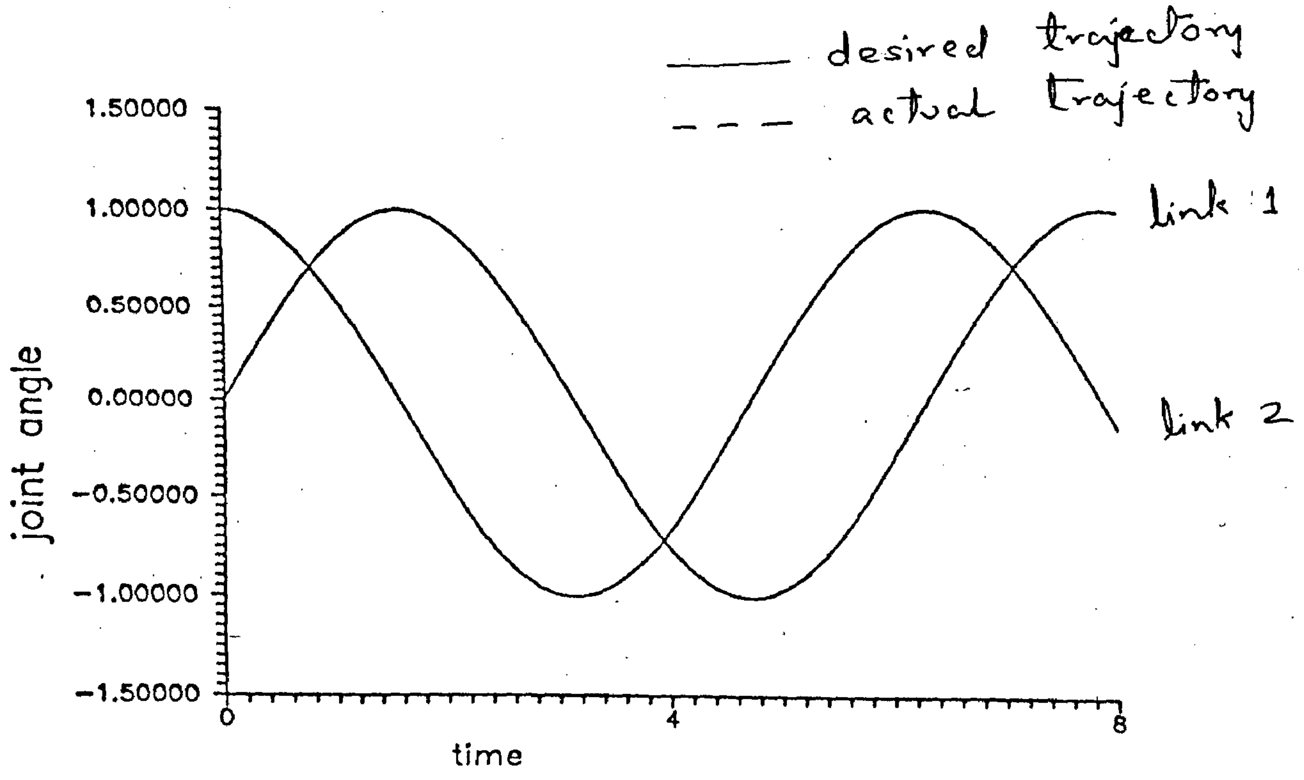


Fig.6.9

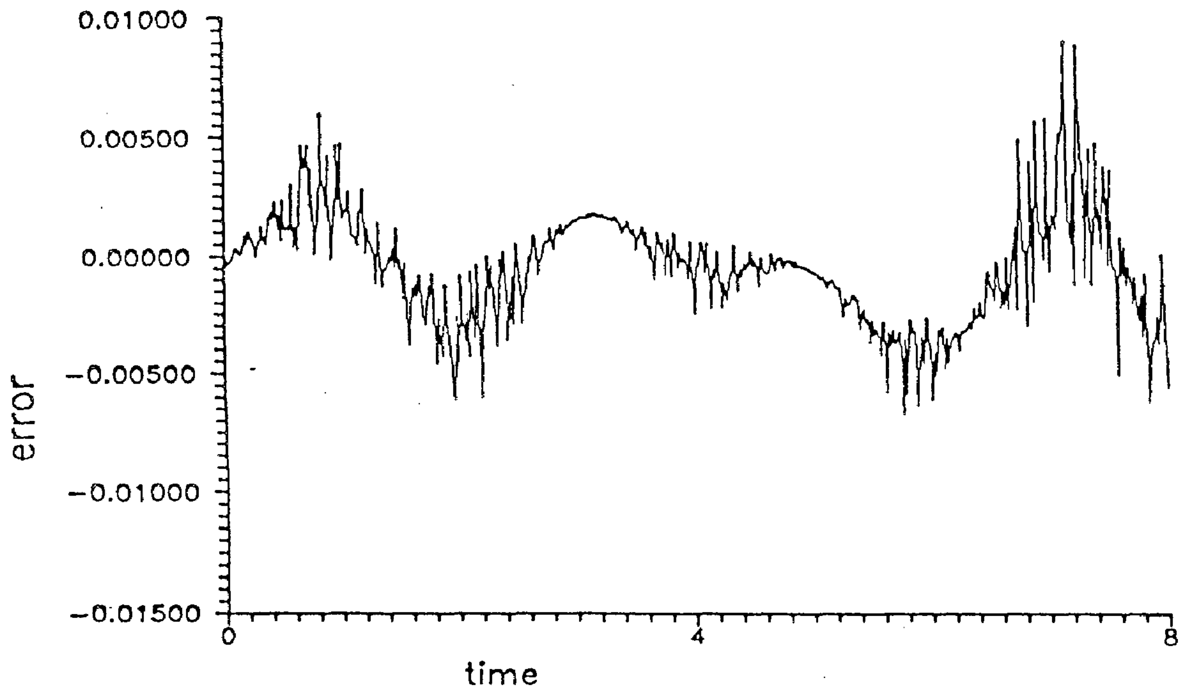


Fig.6.10

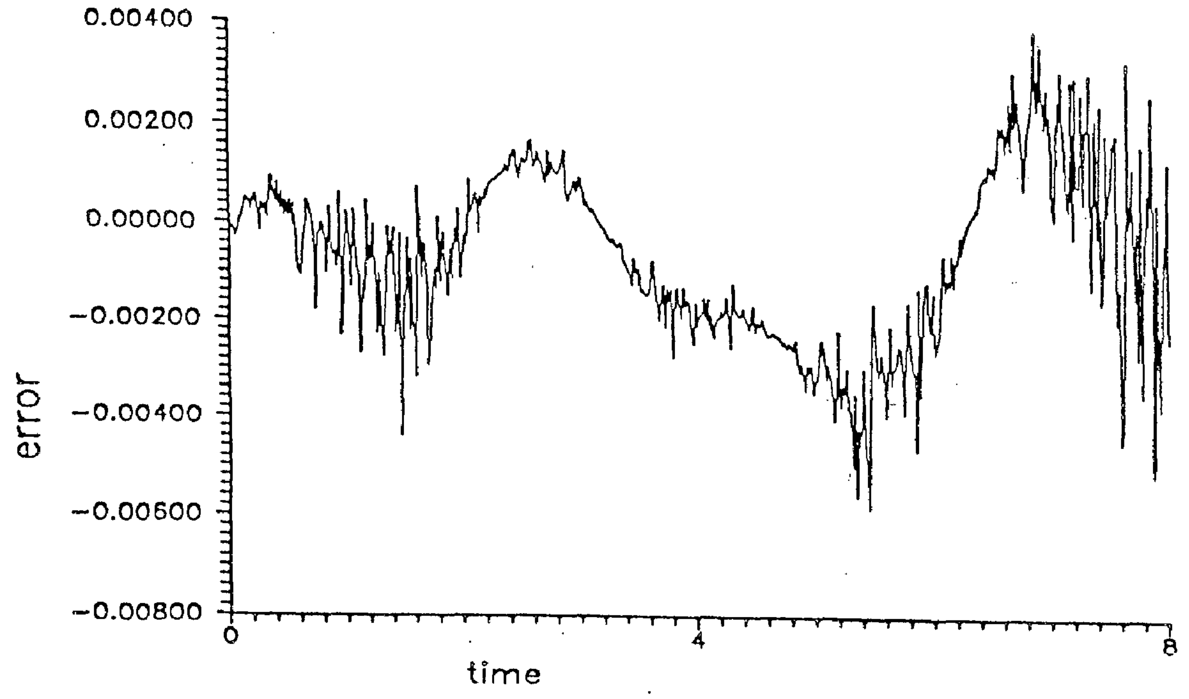


Fig.6.11

REFERENCES

1. **Gang Feng,**"A new adaptive control algorithm for Robot manipulator in task space", IEEE transactions on Robotics and Automation, Vol. 11, No. 3, June 1995.
2. **Ahmet Karakasoglu, Subramania I. Sudharsanan** and **Malur K. Sundareshan,**"Identification and Decentralized Adaptive Control using Dynamical Neural networks with applications to Robot manipulators",IEEE transactions on neural networks, Vol. 4, No. 6, November 1993.
3. **Frank L. Lewis, Kai Liu** and **Aydin Yesildirek,**"Neural net Robot controller with guranteed tracking performance", IEEE transactions on neural networks, Vol. 6, No. 3, 1995.
4. **K. S. Fu, R. C. Gonzalez** and **C. S. G. Lee,**"Robotics: Control, Sensing, Vision and Intelligence", McGraw Hill International edition, 1987.
5. **Robert J. Schilling,**" Fundamentals of Robotics: Analysis and control", Prentice Hall, 1990.
6. **S. Dubowsky** and **D. T. DesForges,**"The Application of Model-Referenced Adaptive Control to Robotic Manipulators", ASME journal of Dynamic Systems, Measurement and Control", Vol. 101, September 1979, pp 193-200.
7. **Rogelio Lozano** and **Bernard Brogliato,**"Adaptive Control of Robot Manipulator with flexible joints", IEEE transactions on Automatic Control, Vol. 37, No. 2, February 1992.
8. **John J. Craig,**"Adaptive Control of Mechanical manipulators", Addison-Wesley Publishing Company, 1988.
9. **Karl J. Astrom** and **Bjorn Wittenmark,**"Computer Controlled Systems: Theory and Design", Prentice Hall of India Pvt. Ltd., 1994.
10. **Richard P. Paul,**"Robot Manipulators: Mathematics, Programming and Control",The MIT Press, 1981.

APPENDIX - A

The acceleration related symmetric matrix D(⊖) is given by,

$$D(\ominus) = \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} & D_{15} & D_{16} \\ D_{12} & D_{22} & D_{23} & D_{24} & D_{25} & D_{26} \\ D_{13} & D_{23} & D_{33} & D_{34} & D_{35} & D_{36} \\ D_{14} & D_{24} & D_{34} & D_{44} & D_{45} & D_{46} \\ D_{15} & D_{25} & D_{35} & D_{45} & D_{55} & D_{56} \\ D_{16} & D_{26} & D_{36} & D_{46} & D_{56} & D_{66} \end{bmatrix}$$

$D_{11} = Tr(U_{11}J_1U_{11}^T) + Tr(U_{21}J_2U_{21}^T) + Tr(U_{31}J_3U_{31}^T) + Tr(U_{41}J_4U_{41}^T) + Tr(U_{51}J_5U_{51}^T) + Tr(U_{61}J_6U_{61}^T)$

$D_{12} = D_{21} = Tr(U_{22}J_2U_{21}^T) + Tr(U_{32}J_3U_{31}^T) + Tr(U_{42}J_4U_{41}^T) + Tr(U_{52}J_5U_{51}^T) + Tr(U_{62}J_6U_{61}^T)$

$D_{13} = D_{31} = Tr(U_{33}J_3U_{31}^T) + Tr(U_{43}J_4U_{41}^T) + Tr(U_{53}J_5U_{51}^T) + Tr(U_{63}J_6U_{61}^T)$

$D_{14} = D_{41} = Tr(U_{44}J_4U_{41}^T) + Tr(U_{54}J_5U_{51}^T) + Tr(U_{64}J_6U_{61}^T)$

$D_{15} = D_{51} = Tr(U_{55}J_5U_{51}^T) + Tr(U_{65}J_6U_{61}^T)$

$D_{16} = D_{61} = Tr(U_{66}J_6U_{61}^T)$

$D_{22} = Tr(U_{22}J_2U_{22}^T) + Tr(U_{32}J_3U_{32}^T) + Tr(U_{42}J_4U_{42}^T) + Tr(U_{52}J_5U_{52}^T) + Tr(U_{62}J_6U_{62}^T)$

$D_{23} = D_{32} = Tr(U_{33}J_3U_{32}^T) + Tr(U_{43}J_4U_{42}^T) + Tr(U_{53}J_5U_{52}^T) + Tr(U_{63}J_6U_{62}^T)$

$D_{24} = D_{42} = Tr(U_{44}J_4U_{42}^T) + Tr(U_{54}J_5U_{52}^T) + Tr(U_{64}J_6U_{62}^T)$

$D_{25} = D_{52} = Tr(U_{55}J_5U_{52}^T) + Tr(U_{65}J_6U_{62}^T)$

$D_{26} = D_{62} = Tr(U_{66}J_6U_{62}^T)$

$D_{33} = Tr(U_{33}J_3U_{33}^T) + Tr(U_{43}J_4U_{43}^T) + Tr(U_{53}J_5U_{53}^T) + Tr(U_{63}J_6U_{63}^T)$

$D_{34} = D_{43} = Tr(U_{44}J_4U_{43}^T) + Tr(U_{54}J_5U_{53}^T) + Tr(U_{64}J_6U_{63}^T)$

$D_{35} = D_{53} = Tr(U_{55}J_5U_{53}^T) + Tr(U_{65}J_6U_{63}^T)$

$D_{36} = D_{63} = Tr(U_{66}J_6U_{63}^T)$

$D_{44} = Tr(U_{44}J_4U_{44}^T) + Tr(U_{54}J_5U_{54}^T) + Tr(U_{64}J_6U_{64}^T)$

$D_{45} = D_{54} = Tr(U_{55}J_5U_{54}^T) + Tr(U_{65}J_6U_{64}^T)$

$D_{46} = D_{64} = Tr(U_{66}J_6U_{64}^T)$

$D_{55} = Tr(U_{55}J_5U_{55}^T) + Tr(U_{65}J_6U_{65}^T)$

$D_{56} = D_{65} = Tr(U_{66}J_6U_{66}^T)$

$D_{66} = Tr(U_{66}J_6U_{66}^T)$

The Coriolis and centrifugal terms:

The velocity related coefficients in the Coriolis and centrifugal terms in eq.(2.14) and (2.15) are expressed by a 6x6 symmetric matrix denoted by $H_{i,v}$ and defined in the following way:

$$H_{i,v} = \begin{bmatrix} h_{i11} & h_{i12} & h_{i13} & h_{i14} & h_{i15} & h_{i16} \\ h_{i12} & h_{i22} & h_{i23} & h_{i24} & h_{i25} & h_{i26} \\ h_{i13} & h_{i23} & h_{i33} & h_{i34} & h_{i35} & h_{i36} \\ h_{i14} & h_{i24} & h_{i34} & h_{i44} & h_{i45} & h_{i46} \\ h_{i15} & h_{i25} & h_{i35} & h_{i45} & h_{i55} & h_{i56} \\ h_{i16} & h_{i26} & h_{i36} & h_{i46} & h_{i56} & h_{i66} \end{bmatrix} \qquad i = 1,2,\ldots,6$$

Let the velocity of the six joint variables be expressed by a six-dimensional column vector denoted by θ:

$$\theta(t) = \left[\theta_1(t),\theta_2(t),\theta_3(t),\theta_4(t),\theta_5(t),\theta_6(t)\right]^T$$

Then, eq.(2.14) can be expressed in the following compact matrix-vector product form:

$$h_i = \dot{\theta}^T H_{i,v} \dot{\theta}$$

where the subscript i refers to the joint $(i = 1,2,\ldots,6)$ at which the velocity induced torques or forces are felt.

The Gravity terms:

From eq(2.16) we have,

$c(\theta) = (c_1,c_2,c_3,c_4,c_5,c_6)^T$

where,

$$c_1 = -(m_1 g U_{11} {}^1r_1 + m_2 g U_{21} {}^2r_2 + m_3 g U_{31} {}^3r_3 + m_4 g U_{41} {}^4r_4 + m_5 g U_{51} {}^5r_5 + m_6 g U_{61} {}^6r_6)$$
$$c_2 = -(m_2 g U_{22} {}^2r_2 + m_3 g U_{32} {}^3r_3 + m_4 g U_{42} {}^4r_4 + m_5 g U_{52} {}^5r_5 + m_6 g U_{62} {}^6r_6)$$
$$c_3 = -(m_3 g U_{33} {}^3r_3 + m_4 g U_{43} {}^4r_4 + m_5 g U_{53} {}^5r_5 + m_6 g U_{63} {}^6r_6)$$
$$c_4 = -(m_4 g U_{44} {}^4r_4 + m_5 g U_{54} {}^5r_5 + m_6 g U_{64} {}^6r_6)$$
$$c_5 = -(m_5 g U_{55} {}^5r_5 + m_6 g U_{65} {}^6r_6)$$
$$c_6 = -m_6 g U_{66} {}^6r_6$$

The coefficients c_i represents the gravity loading terms due to the links and is defined by eq.(2.13). The coefficient D_{ik} is related to the acceleration of the joint variables and is defined by eq.(2.16). The coefficients h_{ikm} is related to the velocity of the joint variables and is defined by eq.(2.15).

APPENDIX - B

The Lagrangian L is defined as the difference between the kinetic energy K and the potential energy P of the system,

$$L = K - P \qquad ....(1)$$

The figure of the robot manipulator used for the derivation is given in fig.(1). The kinetic energy of mass m_1 is written as,

$$K_1 = \frac{1}{2} m_1 d_1^2 \dot{\theta}_1^2 \qquad .....(2)$$

The potential energy is related to the vertical height of the mass expressed by the y coordinates and may be written directly as,

$$P_1 = -m_1 g d_1 \cos(\theta_1) \qquad .....(3)$$

In the case of the second mass m_2, the expression for the cartesian position coordinates are written first, and then differentiating them in order to obtain the velocity. Thus,

$$x_2 = d_1 \sin(\theta_1) + d_2 \sin(\theta_1 + \theta_2) \qquad ......(4)$$

$$y_2 = -d_1 \cos(\theta_1) - d_2 \cos(\theta_1 + \theta_2) \qquad .....(5)$$

The cartesian component of the velocity are then,

$$\dot{x}_2 = d_1 \cos(\theta_1)\dot{\theta}_1 + d_2 \cos(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \qquad .....(6)$$

$$\dot{y}_2 = d_1 \sin(\theta_1)\dot{\theta}_1 + d_2 \sin(\theta_1 + \theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \qquad ....(7)$$

and the kinetic energy is,

$$K_2 = \frac{1}{2} m_2 d_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 d_2 \left[\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right]$$
$$+ m_2 d_1 d_2 \cos(\theta_2)(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2) \qquad ...(8)$$

and the potential energy is,

$$P_2 = -m_2 g d_1 \cos(\theta_1) - m_2 g d_2 \cos(\theta_1 + \theta_2) \qquad ...(9)$$

Therefore,

$$L = \frac{1}{2}(m_1 + m_2) d_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 d_2^2 \left[\dot{\theta}_1^2 + 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right] + m_2 d_1 d_2 \cos(\theta_2)(\dot{\theta}_1^2 + \dot{\theta}_1\dot{\theta}_2)$$
$$+ (m_1 + m_2) g d_1 \cos(\theta_1) + m_2 g d_2 \cos(\theta_1 + \theta_2) \qquad ...(10)$$

Differentiating the Lagrangian L with respect to angular velocity $\dot{\theta}$ and taking the time derivative, we get,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_1} = \left[(m_1 + m_2)d_1^2 + m_2d_2^2 + 2m_2d_1d_2\cos(\theta_2)\right]\ddot{\theta}_1$$
$$+\left[m_2d_2^2 + m_2d_1d_2\cos(\theta_2)\right]\ddot{\theta}_2 - 2m_2d_1d_2\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2$$
$$- m_2d_1d_2\sin(\theta_2)\dot{\theta}_2^2 \qquad ...(11)$$

and differentiating the Lagrangian with respect to the angular displacement θ, we get,

$$\frac{\partial L}{\partial \theta_1} = -(m_1 + m_2)gd_1\sin(\theta_1) - m_2gd_2\sin(\theta_1 + \theta_2) \qquad ...(12)$$

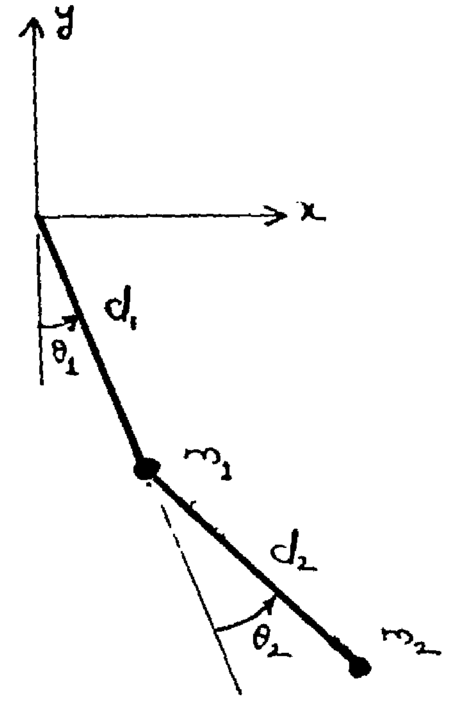
Combining eq.(11) and eq.(12) we get the value of torque at link l_1 as,

$$T_1 = \left[(m_1 + m_2)d_1^2 + m_2d_2^2 + 2m_2d_1d_2\cos(\theta_2)\right]\ddot{\theta}_1 + \left[m_2d_2^2 + m_2d_1d_2\cos(\theta_2)\right]\ddot{\theta}_2$$
$$- 2m_2d_1d_2\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 - m_2d_1d_2\sin(\theta_2)\dot{\theta}_2^2 + (m_1 + m_2)gd_1\sin(\theta_1)$$
$$+ m_2d_2g\sin(\theta_1 + \theta_2) \qquad .....(13)$$

To obtain the equation for the torque at joint 2, differentiate the Lagrangian L with respect to θ_2 and $\dot{\theta}_2$ and the apply the Lagrange-Euler equation. Doing this we get,

$$T_2 = \left[m_2d_2^2 + m_2d_1d_2\cos(\theta_2)\right]\ddot{\theta}_1 + m_2d_2^2\ddot{\theta}_2 - 2m_2d_1d_2\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 -$$
$$m_2d_1d_2\sin(\theta_2)\dot{\theta}_1^2 + m_2gd_2\sin(\theta_1 + \theta_2) \qquad ......(14)$$

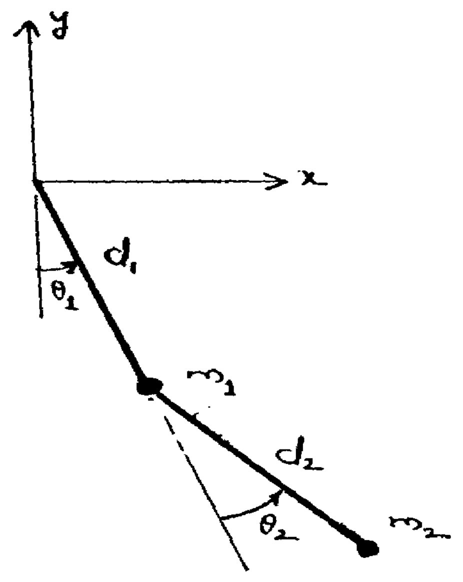
The coefficients of $\ddot{\theta}_i$ are known as inertial terms because an inertial term at joint i causes a torque at joint i equal to $D_{ii}\ddot{\theta}_i$. The coefficients of $\dot{\theta}_j^2$ is known as the centripetal force acting at joint i due to velocity at joint j and the combination of the terms of $\dot{\theta}_j\dot{\theta}_k$ are known as Coriolis force acting at joint i due to the velocities at joint j and joint k and the remaining terms are the gravitational terms.



A two link manipulator

ERRATA

Page no.	Line no	Corrected Statement
28	16	"V_{jk}" is"v_{jk}"
28	20	"Equation 4.1" is "Equation 4.9"
33	26	equa[illegible]on no. "5.4" is equation no. "
	27	equation no. "5.5" is equation no.
35	2, 3, 22	equation no. "5.4" is equation n
35	3,22	equation no. "5.5" is equation
36	10	"function ϕ_i and ψ_i" is "functi
42	15	"Initially, when the robot ar position the coefficients of



A two link manipulator

Page no.	Line no	Corrected Statement
28	16	"V_{jk}" is"v_{jK}"
28	20	"Equation 4.1" is "Equation 4.9"
33	26	equation no. "5.4" is equation no. "5.5 i"
	27	equation no. "5.5" is equation no. "5.5 ii"
35	2, 3, 22	equation no. "5.4" is equation no. "5.5 i"
35	3,22	equation no. "5.5" is equation no. "5.5 ii"
36	10	"function ϕ_i and ψ_i" is "function $\hat{\phi}_i$ and $\hat{\psi}_i$"
42	15	"Initially, when the robot arm is in the rest position the coefficients of ....."

APPENDIX - C

1. Mass of link 1 = 2.0 kg.
2. Mass of link 2 = 1.0 kg.
3. Length of link 1 = 1.0 m.
4. Length of link 2 = 1.0 m.
5. Initial position of link 1 = 6.0^0
6. Initial position of link 2 = 37.0^0

Constant matrix

$$F = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

and

$$\wedge = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$