DYNAMIC CONTROL OF ROBOTIC MANIPULATORS USING ARTIFICIAL NEURAL NETWORK

A DISSERTATION

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

of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

PRABU D

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

JUNE, 2004

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in this dissertation entitled, **"Dynamic Control of Robotic manipulators using Artificial Neural Network"**. in the partial fulfillment of the requirements for the award of degree of **Master of Technology** in the specialization ,**'System Engineering and Operations Research'**, submitted in the Department of the Electrical Engineering , Indian Institute of Technology, Roorkee, is an authentic record of my own work carried out under the supervision and guidance of **Dr.Rajendra Prasad, Associate Professor** and **Dr.Surendra Kumar, Assistant Professor**, Department of Electrical Engineering, Indian Institute of Technology, Roorkee.

I have not submitted the matter embodied in this dissertation for the award of any other degree.

[illegible]

Dated: 30.06.2004

PRABU.D

CERTIFICATE

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

Dated: 30.06.2004

[illegible]

Dr.Rajendra Prasad
Associate Professor,
Department of Electrical Engineering,
Indian Institute of Technology, Roorkee

[illegible]

Dr.Surendra Kumar
Assistant Professor,
Department of Electrical Engineering,
Indian Institute of Technology, Roorkee

ACKNOWLEDGEMENT

I am greatly indebted to my guides **Dr.Rajendra Prasad, Associate Professor** and **Dr.Surendra Kumar, Assistant Professor**, Department of Electrical Engineering, Indian Institute of Technology, Roorkee for their kind support and able guidance through my work. Their co-operative attitude and in-depth knowledge given to me has made my work possible.

Special and sincere thanks to **Dr. N.Sukavanam, Associate Professor,** Department of mathematics , Indian Institute of Technology who taught me Robotics & control and also the Graphics programming of Robot using C++, his support and encouragement has been a constant source of assurance , guidance strength, inspection and upliftment to me.

I acknowledge my sincere gratitude **Dr. G. Kronreif** (TU Vienna) for providing realistic robot data and other relevant technical details. **Horest Ecker, Institute for Machine Dynamics and Measurement,** email:hecker@email.tuwien.ac.at, Tel: +43-1-58801 5567, and Felix Breitenecker, SIMTECH, Vienna University of Technology, Wiedner hau(AUSTRIA) ,I sincerely appreciate his pronounced individualities, humanistic and warm personal approach, which has given me strength to carry out this research work on steady and smooth course. I humbly acknowledge a lifetime's gratitude to him.

My sincere thanks to **Prof Stephen Morgan,** University of Missouri Rolla, USA and **Prof Marian B Gorzalczany**, kielce University of Technology, Poland, for extending moral support and technical discussions at various issues.

My heartily gratitude to **Prof.M.K. Vasantha, Prof .A.K Pant and Dr. (Ms) Indra Gupta** for providing the excellent facilities of the lab, at all times, completely heartily, without, which it was impossible for me to complete this work.

I also wish to express my deep sense of gratitude to **all faculty members** of Department of Electrical Engineering, IIT Roorkee for their sincere advice and encourage during preparation of this thesis.

ABSTRACT

The aim of the thesis is to develop an efficient dynamic control such as point to point and continuous path control strategy for Robot manipulator using Artificial neural network (ANN) and Adaptive Neuro fuzzy inference system (ANFIS),owing to the advantage of their learning ability & unique characteristics, which enables to control Robot manipulators

In this work, a robot controller based on neural network is presented. This controller has been applied to a single link robot arm and three link SCARA (Selective Compliance Assembly Robot Arm) which has a highly nonlinear structure. The model based approaches for robot control (such as the computed torque technique) require high computational time and can result in a poor control performance, if the specific model-structure selected does not properly reflect all the dynamics. In addition, conventional-PD controller could not cope up with unmodeled dynamics. Moreover, Fuzzy logic can be used to map complex nonlinear relations by a set of IF-THEN rules. The membership functions are designed by intuitive human reasoning. This poses three problems. One, for different control application a new set of membership functions have to developed and second, latent stability problem and third, once these membership functions are developed and implemented there is no means of changing them. This means fuzzy logic lacks a learning function. Neural networks on the other hand self-organize the mapping relationship by learning. A dynamic model has been assumed here where a controller is associated with each joint and separate RBF (Radial Base Function) neural networks are used as assistants to the PD controllers in order to minimize tracking errors. All above features naturally allow one to consider investigating the feasibility of neural networks.

The proposed ANFIS methodology combines artificial neural networks with fuzzy logic. The fuzzy sets are used to formalize the level of human perception of the physical system. The neural networks, on the other hand, perform all the necessary computations and with regard to their learning capabilities, they enable an adaptation of the existing controller through its learning to the changes in the system behavior. In this work, the ANFIS (Adaptive Neuro-Fuzzy inference system) for the dynamic control (point-to-point as well as continuous path control) of the three-link SCARA manipulator is designed. This new method for control combines the advantages of neural networks (learning and adaptability) with the advantages of fuzzy logic (use of expert knowledge) to achieve the goal of robust control of robot dynamic systems. Simulation results show very good tracking performance.

In addition, a visual display of three-link SCARA manipulator is made by using C++. Further, in this work a practical approach to implement the Neuro -Fuzzy technique has been discussed for future extension.

CONTENTS

CHAPTER-1

1.1 INTRODUCTION

The very precise control of robot manipulator to track the desired trajectory is a very tedious job and almost unachievable to certain limit with the help of adaptive controllers. This task is achievable to certain limit with the help of adaptive controller but these also have their own limitation of assuming that the system parameters being controlled change relatively slow. With reference to the tasks assigned to an industrial robot, one important issue is to determine the motion of the joints and the end effectors of the robot. Therefore, the purpose of the robot arm control, as Fu et al [1]. wrote in one of the classical works on robotics, is to maintain the dynamic response of the manipulator in accordance with some prespecified performance criterion.

Among the early robots of the first generation, non-servo control techniques, such as bang-bang control and sequence control, were used. These robots move from one position to another under the control or limit switches, relays, or mechanical stops [19].

During the 1970s, a great deal of work was focused on including such internal state sensors as encoders, potentiometers, tachogenerators, etc., into the robot controller to facilitate manipulative operation [2, 3]. Since then, feedback control techniques have been applied for servoing robot manipulators.

Up till now, the majority of practical approaches to the industrial robot arm controller design use traditional techniques, such as PD or PID controllers, by treating each joint of the manipulator as a simple linear servomechanism. In designing these kinds of controllers, the non-linear, coupled and time-varying dynamics of the mechanical part of the robot manipulator system are completely ignored, or are dealt with as disturbances. These methods generally give satisfactory performance when the robot operates at a low speed. However, when the links are moving simultaneously and at a high speed, the non-linear coupling effects and the interaction forces between the manipulator links may degrade the performance of the overall system and increase the tracking errors. The disturbances and uncertainties in a task cycle may also reduce the tracking quality of robot manipulators.

Thus, these methods are only suitable for relatively slow manipulator motion and for limited-precision tasks [4].

The Computed Torque Control (CTC) is commonly used in the research community. The CTC control law has the ability to make the error asymptotically stable if the dynamics of the robot are exactly known [5] However, manipulators are subject to structured and/or unstructured uncertainty. Structured uncertainly is defined as the case of a correct dynamic model but with parameter uncertainty doe to tolerance variances in the manipulator link properties, unknown loads, inaccuracies in the torque constants of the actuators, and others. Unstructured uncertainty describes the case of unmodeled dynamics, which result from the presence of high-frequency modes in the manipulator, neglected time-delays and non-linear friction. It has been widely recognized that the tracking performance of the CTC method in high-speed operations is severely affected by the structured and unstructured uncertainties. To cope with the problem, some adaptive approaches have been proposed to maintain the tracking performance of the robotic manipulator in the presence of structured uncertainty [6]. Some other researchers have also tried to incorporate the neural network into the controller design and good results were reported [7, 8]. The Fuzzy logic can also be used to map complex nonlinear relations by a set of IF-THEN rules. The membership functions are designed by intuitive human reasoning. This poses three problems. One, for different control application a new set of membership functions have to developed and second, latent stability problem[21] and third, once these membership functions are developed and implemented there is no means of changing them. This means fuzzy logic lacks a learning function. An An ANFIS model [11] is an adaptive neural network which represents a particular type of fuzzy inference system. Three types of fuzzy inference systems can be represented by an ANFIS model.

1.Type 1: A fuzzy inference system whose overall output is the weighted average of each rule's crisp output. The output membership functions are monotonic functions.

2. Type2: A mamdani fuzzy inference system where the centroid defuzzification operator is replaced by a discrete version which calculates the approximate centroid of area.

3.A Sugeno-type fuzzy inference system whose output is a linear combination of the input variables plus a constant term.

In this thesis, another control strategy, namely, the Neuro Control & a special case of type 3 Neuro Fuzzy (Adaptive Neuro Fuzzy Inference System., i.e; ANFIS) control, which had been used in industrial process control previously, is proposed for robot control.The consequent output member of each rule is a constant. The output of the system is a weighted average of these constants. It can be represented by the network shown in fig 12.

It has been demonstrated that Independent joint control is used for the projection and execution of the trajectory tracking, where PIDs and neural networks are used as controllers. The neural networks are used as assistants to the PID controllers in order to minimize tracking errors. All above features naturally allow one to consider investigating the feasibility of neural network& ANFIS based controller in robot control.

1.2 Robot system

The aim of the robot simulation is to develop a complete mathematical representation of an open-loop robot system by incorporating actuator effects, gear backlash and dynamic equation with inertia, centrifugal and Coriolis, frictional and gravitational, and other uncertainties. After having been built up, this simulation model will be treated as the 'real robot'. All measurements and simulations through this thesis will be performed on this model.

1.2.1 SINGLE LINK MANPULATOR

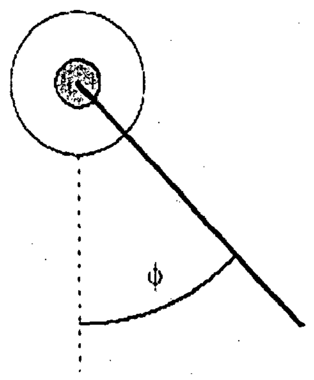


Fig: 1 Scheme of single link manipulator

The single link assumed to be a thin homogenous rod of mass m1 and length a1.In this case there is no velocity coupling terms due to coriolis and centrifugal force because there is only one axis .

1.2.2 THREE LINK SCARA MANIPULATOR

A SCARA (Selective Compliance Assembly Robot Arm) industrial robot is chosen as the prototype of the simulation model, because the dynamic characteristics of this kind of robots have been intensively researched. It is one of the best known robots to the research community.

The SCARA model chosen in this work has two revolute joints and one prismatic joint (in the configuration RRP) to position the wrist. However, for a SCARA robot, the axes of all three joints are vertical, as shown in fig 2 the first revolute joint swings the arm back and fourth about a base axis that can be thought of as a vertical shoulder axis. The second revolute joint swings the forearm back and fourth about a vertical elbow axis. Thus, the two revolute joints control motion in a horizontal plane. The vertical component of the motion is provided by the third joint, a prismatic joint which slides the wrist up and down.

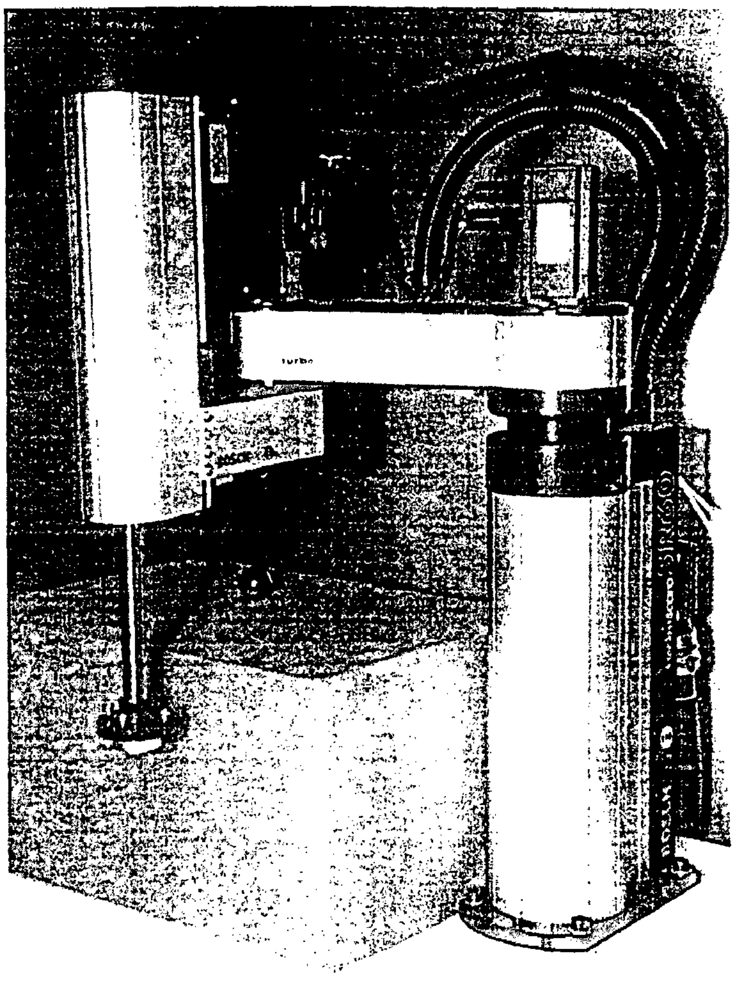


Fig 2: Picture of a three link SCARA Manipulator

CHAPTER -2

INTRODUCTION TO NEURAL NETWORK CONTROLLER

2.1 General

The science of artificial neural networks is based on the neuron. In order to understand the structure of artificial networks, the basic elements of the neuron should be understood. Neurons are the fundamental elements in the central nervous system. The diagram below (Fig.3) shows the components of a neuron. [5] A neuron is made up of 3 main parts - dendrites, cell body and axon. The dendrites receive signals coming from the neighboring neurons. The dendrites send their signals to the body of the cell. The cell body contains the nucleus of the neuron. If the sum of the received signals is greater than a threshold value, the neuron fires by sending an electrical pulse along the axon to the next neuron.

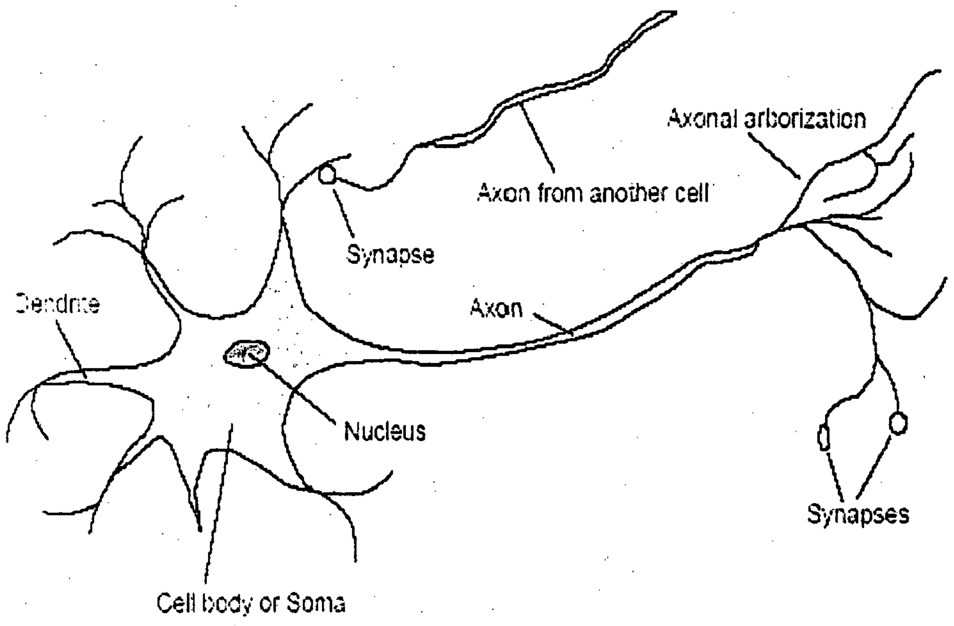


Fig 3: A Biological Neuron

The following model is based on the components of the biological neuron (Fig. 3). The inputs X0-X3 represent the dendrites. Each input is multiplied by weights W0- W3. The output of the neuron model, Y is a function, F of the summation of the input signals.

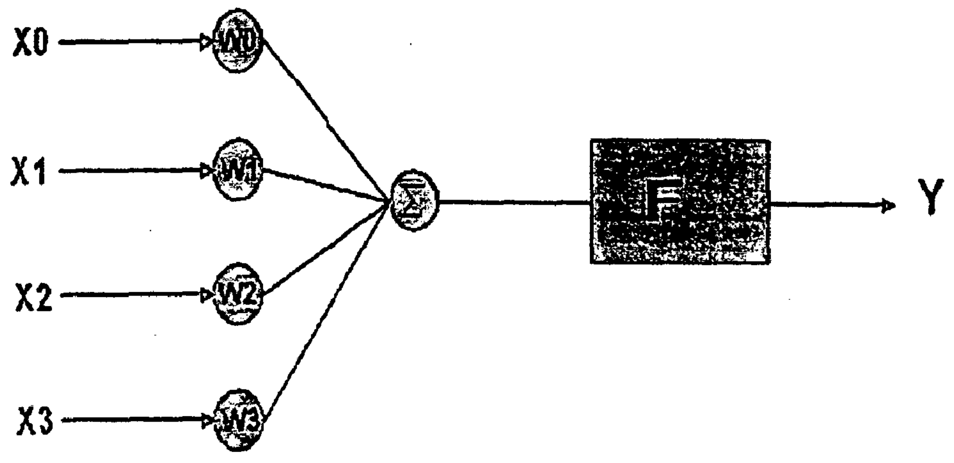


Fig 4: Scheme of neuron model

2.2 Advantages of ANN's

1. The main advantage of neural networks is that it is possible to train a neural network to perform a particular function by adjusting the values of connections (weights) between elements. For example, if we wanted to train a neuron model to approximate a specific function, the weights that multiply each input signal will be updated until the output from the neuron is similar to the function.

2. Neural networks are composed of elements operating in parallel. Parallel processing allows increased speed of calculation compared to slower sequential processing.

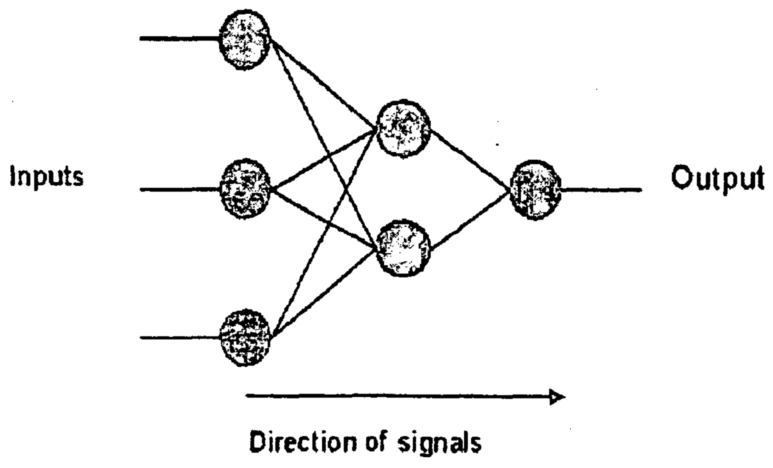


Fig 5: Scheme of neural network

3. Artificial neural networks (ANN) have memory. The memory in neural networks corresponds to the weights in the neurons. Neural networks can be trained offline and then transferred into a process where adaptive learning takes place. In our case, a neural network controller could be trained to control three link SCARA system in the simulink environment. After training, the network weights are set. The ANN is placed in a feedback loop with the actual process. The network will adapt the weights to improve performance as it controls the Robot system.

2.3 Neural network structures

The most common type of single layer feed forward network is the perceptron. Other types of single layer networks are based on the perceptron model. The details of the perceptron are shown in figure 6.

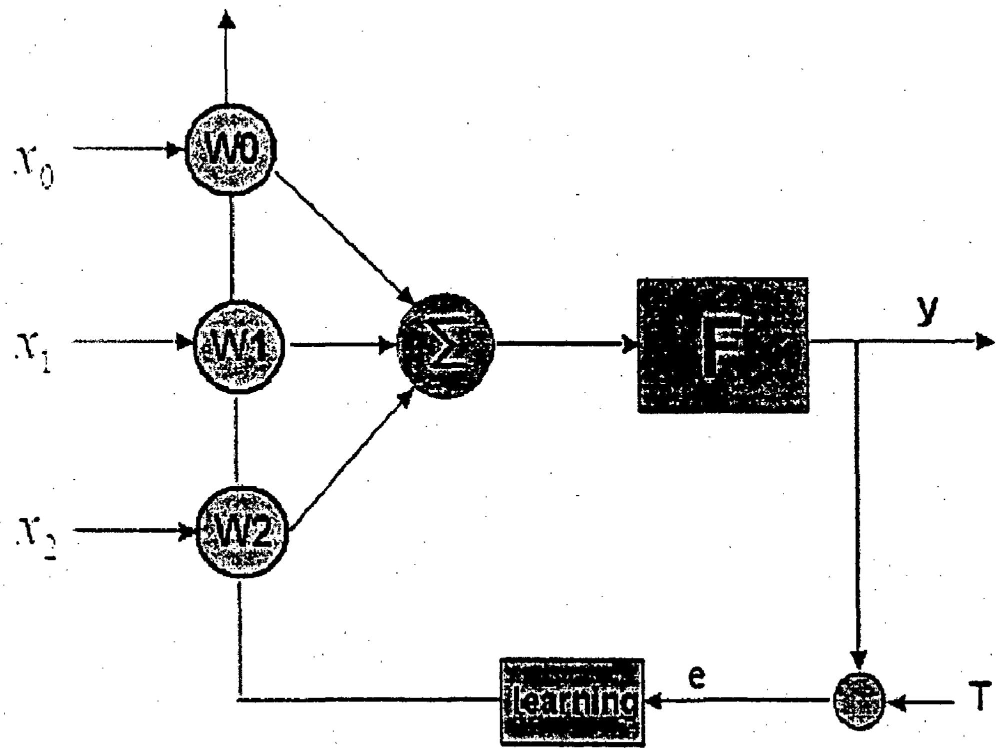
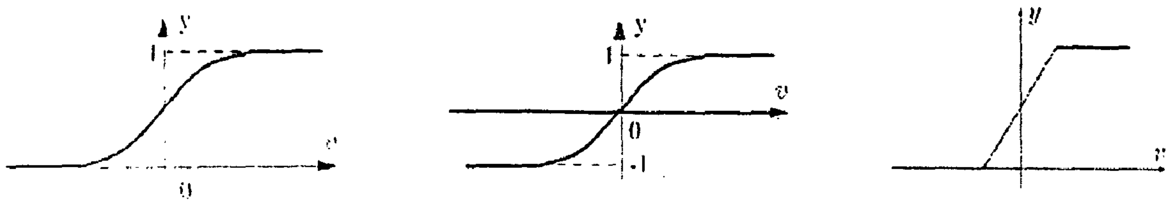


Fig 6: The learning scheme of neural network.

Inputs to the perceptron are individually weighted and then summed. The perceptron computes the output as a function F of the sum. The activation function, F is needed to introduce nonlinearities into the network. This makes multi-layer networks powerful in representing nonlinear functions.

There are 3 main types of activation function -tan-sigmoid, log-sigmoid and linear. [8] Different activation functions affect the performance of an ANN.



Log-sigmoid function Tan-sigmoid function Linear function

Fig 7: various activation function scheme of neural network.

The output from the perceptron is

$$y(k) = f(w^{T}[k].x[k]) \tag{1}$$

The weights are dynamically updated using the back propagation algorithm. The difference between the target output and the actual output (error) is calculated.

$$e(k) = T[k] - y[k] \tag{2}$$

The errors are back propagated through the layers and the weight changes are made. The formula for adjusting the weights is

$$w[k+1] = w[K] + \mu.e[k].x[k] \tag{3}$$

Once the weights are adjusted, the feed-forward process is repeated. The weights are adapted until the error between the target and actual output is low. The approximation of the function improves as the error decreases. Single-layer feed forward networks are useful when the data to be trained is linearly separable. If the data we are trying to model is not linearly separable or the function has complex mappings, the simple perceptron will have trouble trying to model the function adequately.

A key property of artificial neural networks is their ability to generate input-output maps which under mild assumptions can approximate any function to any degree of accuracy. This property has been exploited by a number of researchers to propose controllers and control strategies for a variety of applications [I, 7-9, 17] several types of artificial neural networks can be found in the literature. Some of them can be applied to a wide variety of problems, while others are targeted for special applications. This work considers the type of networks known as radial basis functions (RBF) neural networks.

2.4 Radial basis functions neural network controller

Radial basis functions neural networks belong to the class of multilayer feed forward neural networks. They are characterized by a hidden layer made up of a collection of locally tuned processing units. These units can be seen as independent kernel nodes which compute the distance between their corresponding centroid and the input they receive from the input units. The output of these kernel nodes is characterized by a nonlinear, radially symmetric activation function of that distance. Normally, the closer an input is to the center of the receptive field of one of these units, the stronger the response of the unit. The output layer is composed of linear units which are fully connected to the units in the hidden layer. In other words, each output unit performs a weighted sum of the responses it receives from each hidden unit. RBF neural networks are modeled by the following relation:

$$y(x) = \sum_{j=1}^{m} \omega_j g_j(\|x - c_j\|) \tag{4}$$

In this equation, gj corresponds to the jth hidden unit ω_j is the weight associated .With the j^{th} unit, x represents the input vector and c_j is the receptive-field center of the jth unit. Broomhead and Lowe [3] who used them in the prediction of chaotic time series first brought radial basis functions into the neural networks literature. Their work was influenced by previous theoretical developments on multivariable Interpolation reported by Powell [2] and Micchelli [34]. Since then, many other researchers have studied the learning ability and representational capacity of RBF neural networks e.g. see [2, 16]

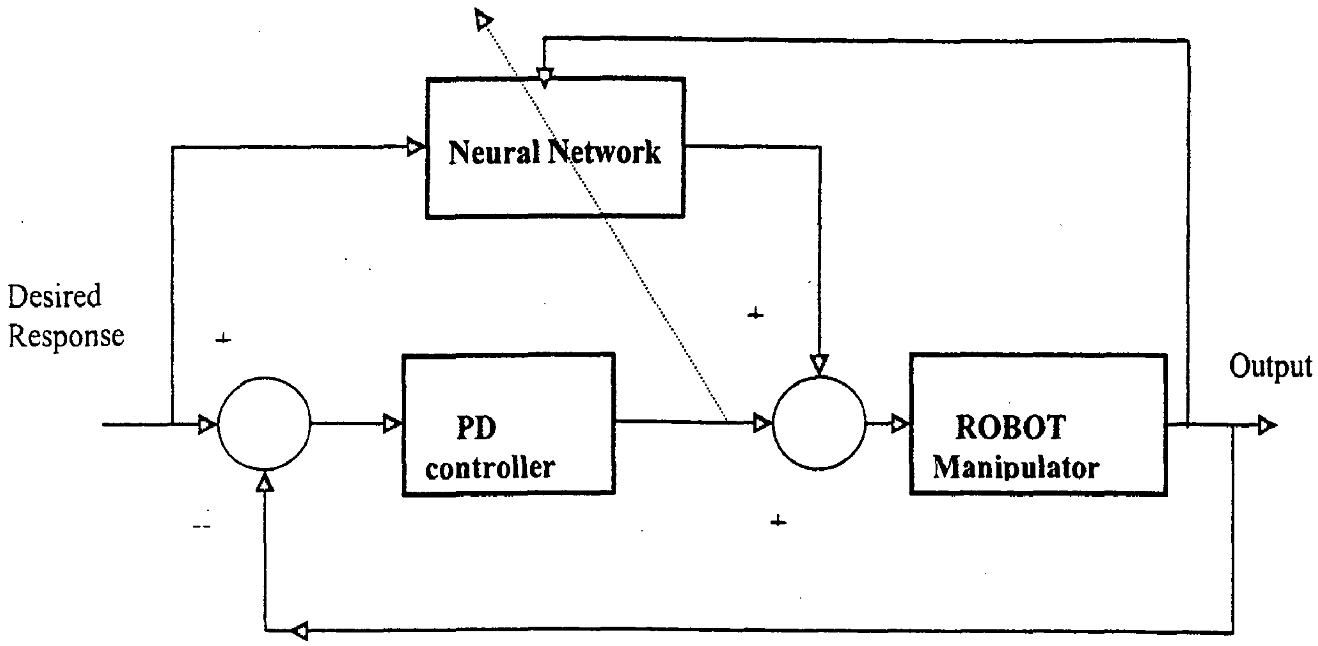


Fig 8: The control system design for Robot Manipulator with ANN.

It can be shown that the dynamic equations describing the behavior of robotic manipulators (controlled object) can be written in the following matrix form:

$$\ddot{q} = M^{-1}(q)f - M^{-1}(q)C(q,\dot{q}) - M^{-1}(q)G \qquad (5)$$

where f (t) is the vector of generalized (non-conservative) forces, q (t) the vector of generalized coordinates, M (q (t)) the inertia matrix, C (q (t), q (t)) the Coriolis and centrifugal force vector, and G (q (t)) the vector of potential energy terms representing the contributions to the generalized forces from the conservative forces acting on the system. Formally speaking, Eq. (2) represents the inverse dynamics of the arm: it is a mapping from link variables to input variables. The direct dynamics can easily be obtained from Eq. (2) considering that matrix M is always invertible. Therefore,

$$\ddot{q} = M^{-1}(q)f - M^{-1}(q)C(q,\dot{q}) - M-^{1}(q)G$$

$$\ddot{q} = [D](q,\dot{q},f) \qquad (6)$$

Eq. (4) represents the direct dynamics of the arm. [D] Represents a nonlinear transformation or a mapping from input variables to link variables. From this point of $f = [D]^{-1}(q,\dot{q},\ddot{q})$ view, the inverse dynamics can be represented by the following relation: To emphasize this, note that $[D]^{-1}$ denotes an inverse mapping, i.e. an inverse transformation of the direct dynamics. In this sense, $[D]^{-1}$ represents a nonlinear function (transformation) from link variables to the input variables. For convenience, Let us make use of the following state-variable notation:

$$x_1 = q \quad x_2 = \dot{q} \quad u = f \tag{7}$$

Then, u can be expressed in the following way

$$u = M(x_1)\dot{x}_2 + C(x_1, x_2) + G$$

$$u = [D]^{-1}(x_1, x_2, \dot{x}_2) \tag{8}$$

Where x_2 denotes the response of the system under the influence of the input u.

2.5 Controller design

Let denote the desired response of the system, i.e. the desired acceleration. Thus, a control signal can be formed by using the arm's inverse dynamics.

$$u = [D]^{-1}(x_1, x_2, \dot{x}_{2d})$$
$$u = M(x_1)\dot{x}_{2d} + C(x_1, x_2) + G \tag{9}$$

Then, equating Eqs. (7) and (10), it follows that

$$M(x_1)[\dot{x}_{2d} - \dot{x}_2] = 0 \tag{10}$$

This leads to the following equality:

$$\dot{x}_2 = \dot{x}_{2d} \tag{11}$$

Hence, the response of the system satisfies the desired performance. The important aspect to highlight here is that this error equation is a satisfied only if the inverse dynamics of the system are known precisely. In this thesis, we propose the use of RBF neural networks to approximate the inverse mapping $[D]^{-1}$ as closely as possible. From what has been said so far, let us Use the following control law:

$$u = [\hat{D}]^{-1}(q, \dot{q}, \ddot{q}_d) + K_v\dot{e} + K_p e \tag{12}$$

Where $[D]^{-1}$ is the neural network approximation of the actual inverse dynamics of the system. The last two terms on the right-hand side represent a servo feedback which is introduced to stabilize the system. K_v and K_p are constant gain matrices and $e = x_{1d} - x_1$ represents the tracking error. Now, making use of Eqs. (13) and (15), it can be shown that

$$k_v \dot{e}_i + k_p e_i = \tilde{d}_i^{-1}(q, \dot{q}, \ddot{q}_d) \tag{13}$$

Eq.(14) characterizes a linear decoupled system driven by the nonlinear vector Function $[\tilde{D}]^{-1}(q, \dot{q}, \ddot{q}_d)$.This function represents the error in the neural network approximation of the manipulator's inverse dynamics. It makes intuitive sense that instead of using one network to approximate the inverse dynamics of the whole arm, one ought to use a separate network for each joint of the manipulator. With this in mind and by using Eq.(14), the error equation for the ith joint of the manipulator is expressed as follows:

$$k_v \dot{e}_i + k_p e_i = \tilde{d}_i^{-1}(q, \dot{q}, \ddot{q}_d) \tag{14}$$

In this last equation, K_v and K_p are constant gains and $\tilde{d}_i^{-1}(.)$ the local approximation error of the RBF neural network assigned to the i^{th} joint. Now, letting

$$k_v \dot{e} + k_p e = \varepsilon_i, \tag{15}$$

Eq. (15) can be written as follows:

$$\varepsilon_i = \tilde{d}_i^{-1}(q, \dot{q}, \ddot{q}_d) \tag{16}$$

Noting that $\varepsilon_i = \varepsilon_i(t)$, one concludes that $\varepsilon_i(t)$ constitutes a measure of the tracking error that reflects the mismatch between the actual inverse dynamics of the system and its local neural network approximation. The question that remains to be answered at this point is how to update, or adjust, the network parameters on-line, so that the error measure $e(t)$ converges to $\varepsilon_i(t)$ as $t \longrightarrow \infty$. A gradient descent approach comes immediately to mind. To do that, let us define the following cost function:

$$J = \frac{1}{2}\varepsilon_i^2 \tag{17}$$

J must be minimized over the parameter space of the network. In doing so, let us derive the update law.

$$\dot{\omega}_j = -\alpha \nabla J$$

$$\dot{\omega}_j = -\alpha \frac{\partial \varepsilon_i}{\partial \omega_j} \tag{18}$$

$$\dot{\omega}_j = -\alpha \varepsilon_i \frac{\partial \hat{d}_i^{-1}}{\partial \omega_i}$$

Recalling that the output of the network is given by Eq. (1), the update law takes on the following form:

$$\dot{\omega}_j = -\alpha \varepsilon_i g_j(.) \tag{19}$$

In this update law, $a > 0$ is the adaptation or learning parameter. The subscript j denotes the jth weight of the RBF network. Therefore, the network's weights are updated according to a simple first-order differential equation involving the param- eter a, the current value obtained from a servo feedback loop, and the local response of the corresponding radial unit. It is important to point out that the above development depends on the underlying assumption that RBF networks are capable of approximating the inverse dynamics of the controlled system. It is known that RBF networks are capable of approximating any reasonable function. However, as it will be made evident shortly, that the fact by itself is not sufficient for stability. Stability imposes the additional requirement that the error in the approximation of the inverse dynamics has to remain bounded. Satisfying this requirement with carefully designed networks constitutes a nontrivial problem which still needs further research. A particularly Good example of research on this subject is the work by Sanner [47] who makes use of principles from sampling theory and Fourier analysis to develop networks that, under mild assumptions, are capable of uniformly approximating smooth functions on specified compact sets. Not surprisingly, it turns out that the condition on the bounded ness of the approximation error can be satisfied by carefully choosing the RBF network parameters, so as to guarantee a desired uniform approximation in a target set. This translates into selecting an appropriate variance, when the units in the networks are Gaussian units, and precisely placing the network units in the region of interest. For the purposes of this thesis, it will be assumed that the RBF networks in the proposed control system have been designed so as to guarantee that the error in the approximation of the inverse dynamics remains bounded.

2.6 Stability analysis

Consider the following proposition:

Proposition. *Given that the error in the approximation of the inverse dynamics* is *bounded, all states of the system will also remain bounded.*

Proof. Realizing that eq (14) represents an asymptotically stable linear system driven by the nonlinearity$[D]^{-1}$, the above proposition can be proven using the direct method of Lyapunov. To do that, let us use the following scalar Lyapunov function:

$$V(e)=\frac{1}{2}e^{T}e \tag{20}$$

The system itself is represented by the expression shown below.

$$\begin{aligned} K_{v}\dot{e}+K_{p}e&=[\tilde{D}]^{-1}(q,\dot{q},\ddot{q}) \\ \dot{e}+K_{v}^{-1}K_{p}e&=K_{v}^{-1}[\tilde{D}]^{-1}(q,\dot{q},\ddot{q}) \end{aligned} \tag{21}$$

Obviously, the function of Eq. (22) is positive definite. This satisfies the first condition of the Lyapunov theorem. To satisfy the second condition we must determine the circumstances under which V (e) is monotonically decreasing:

$$\begin{aligned} \dot{V}(e)&=e^{T}\dot{e} \\ \dot{V}(e)&=e^{T}\left[K_{v}^{-1}[\tilde{D}]^{-1}(.)-K_{v}^{-1}K_{p}e\right] \end{aligned} \tag{22}$$

$\dot{V}(e)$ must satisfy the condition that $\dot{V}(e)<0$. Hence,

$$e^{T}K_{v}^{-1}[\tilde{D}]^{-1}(\cdot)-e^{T}K_{v}^{-1}K_{p}e\leq 0, \tag{23}$$

$$\left\|K_{v}^{-1}\right\|\left\|[\tilde{D}]^{-1}(\cdot)\right\|\geq\left\|K_{v}^{-1}\right\|\left\|K_{p}\right\|\left\|e\right\|$$

But, $\left\|[\tilde{D}]^{-1}\right\|\leq\xi$; therefore,

$$\left\|e\right\|\leq\frac{\xi}{\lambda_{p}} \tag{24}$$

In this last expression,Ip represents the largest eigenvalue of lambda ***p***. Furthermore, from Eq. (24) and using Eq. (31) we can determine the bounds on 8. This is shown below.

$$\|\dot{e}\| \leq \left\|K_{\upsilon}^{-1}\right\| \left\|[\tilde{D}]^{-1}(\cdot)\right\| + - \left\|K_{\upsilon}^{-1}\right\| \left\|K_{p}\right\| \|e\|$$

$$\|\dot{e}\| \leq 2\lambda_{v}\xi \tag{25}$$

Where λ_{v} , represents the maximum eigenvalue of K_{v}^{-1} .

CHAPTER-3

INTRODUCTION TO NEURO FUZZY CONTROLLER FOR ROBOT SYSTEM

3.1 Introduction

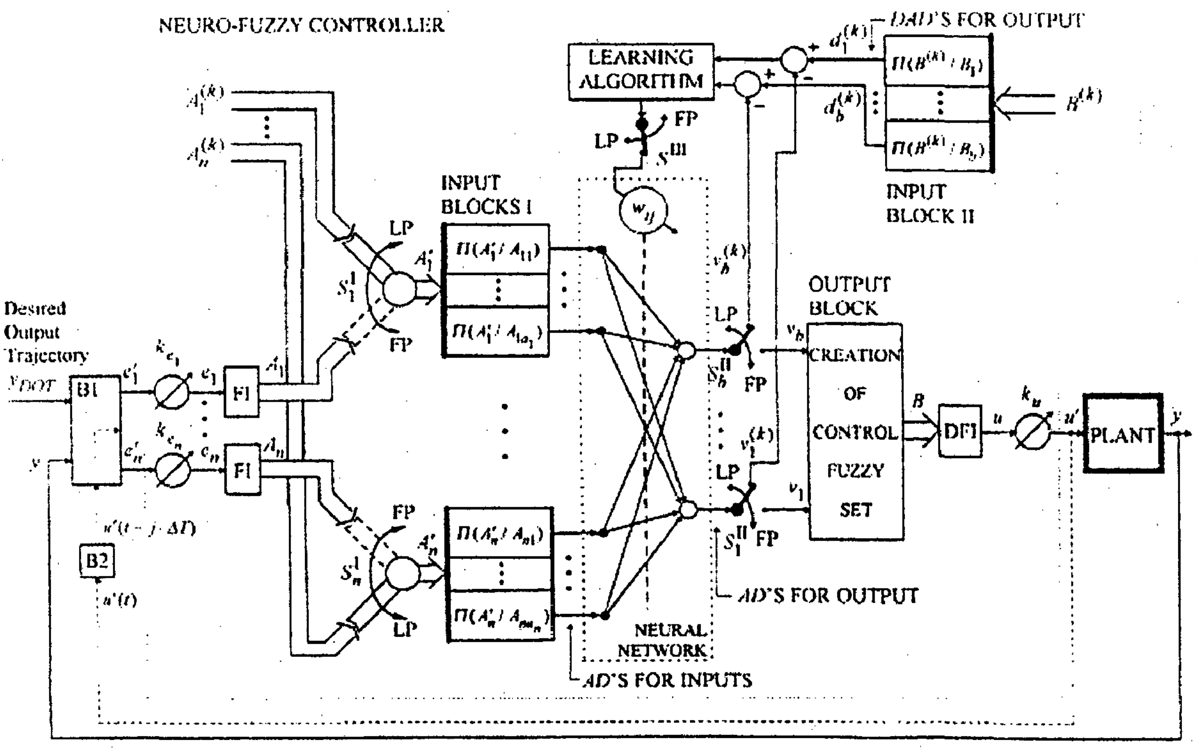


Fig 9: The general structure of the neuro fuzzy controller in closed loop mode.

3.2 The structure of the controller

A general structure of the proposed controller in a closed-loop control mode is presented in Fig. 1. When the 'switches' S_1^I,,,,,,,S_n^I,S_1^{II},...,S_b^{II},andS^{III} are in the Learning Phase (LP) positions, the control loop is open and the controller is in its learning phase during which it acquires and accumulates control knowledge. This knowledge is stored in an artificial neural network which is an important part of the neuro-fuzzy structure of the controller. The details concerning the learning phase are presented in the following section. After the learning process is completed, all switches can be put in the Functioning Phase (FP) positions. The control loop is then closed and the control process is being performed.

The general procedure for the construction of the proposed controller has **three** main stages:

1. The choice of the controller structure in terms of its inputs and outputs, and the definition of the so-called primary fuzzy sets for them,
2. The learning phase of the controller,
3. The assessment of the controller quality which corresponds to the functioning phase (that is, the decision making phase) of the controller. The controller presented in Fig. 1 has n inputs, e01; e02; . . . ; e0n and one output u0 (of course, our approach can be easily generalized for the case of multi-output controller). Since the controller output (the control signal) is usually predetermined, the choice of the controller inputs determines the controller dynamics and, therefore, has a significant influence on the quality of the controller. The choice of the controller inputs is performed by block B1 (see Fig. 1) on the basis of the plant output signal y (present and previous values), the desired output trajectory yDOT (also present and previous values), and sometimes on the basis of previous control actions $u'(t-\Delta T)$. Where $j \in \{1,2,...\}$ and ΔT is a sampling period. Block B2 in Fig. 1 is a delay unit supplying block B1 with the control signal u0 delayed by j sampling periods. The parameters ke1 ; ke2 ; . . . ; ken are the scaling factors for the controller inputs; similarly ku is the scaling factor for the control signal u. Blocks FI and DFI of Fig. 1 represent the fuzzification interface and the defuzzification interface, respectively, and they will be described in the following sections. In the case of a simple control system in which yDOT is a set-point value ySP and the controller has two inputs: the control error and the change of control error, the structure of block B1 is presented in Fig. 2. If we need an incremental controller, then the system of Fig. 1 - instead of output u (the control action) is characterized by output Δu (the change of the control action).In such a case, the final control action u0.t applied to the plant is of the form (see Fig. 3): $u'(t) = u'(t-\Delta T) + k_{\Delta u} . \Delta u(t)$, $t \geq 0, u'(t<0) = 0$ where $u'(t-j \cdot \Delta T)$. is a previous control and kDu is a scaling factor for the

controller. The details concerning the learning phase are presented in the following section. After the learning process is completed, all switches can be put in the Functioning Phase (FP) positions. The control loop is then closed and the control process is being performed.

The general procedure for the construction of the proposed controller has

three main stages:

1. The choice of the controller structure in terms of its inputs and outputs, and the definition of the so-called primary fuzzy sets for them,
2. The learning phase of the controller,
3. The assessment of the controller quality which corresponds to the functioning phase (that is, the decision making phase) of the controller. The controller presented in Fig. 1 has n inputs, e01; e02; . . . ; e0n and one output u0 (of course, our approach can be easily generalized for the case of multi-output controller). Since the controller output (the control signal) is usually predetermined, the choice of the controller inputs determines the controller dynamics and, therefore, has a significant influence on the quality of the controller. The choice of the controller inputs is performed by block B1 (see Fig. 1) on the basis of the plant output signal y (present and previous values), the desired output trajectory yDOT (also present and previous values), and sometimes on the basis of previous control actions $u'(t-\Delta T)$. Where $j \in \{1,2,...\}$ and ΔT is a sampling period. Block B2 in Fig. 1 is a delay unit supplying block B1 with the control signal u0 delayed by j sampling periods. The parameters ke1 ; ke2 ; . . . ; ken are the scaling factors for the controller inputs; similarly ku is the scaling factor for the control signal u. Blocks FI and DFI of Fig. 1 represent the fuzzification interface and the defuzzification interface, respectively, and they will be described in the following sections. In the case of a simple control system in which yDOT is a set-point value ySP and the controller has two inputs: the control error and the change of control error, the structure of block B1 is presented in Fig. 2. If we need an incremental controller, then the system of Fig. 1 - instead of output u (the control action) is characterized by output Δu (the change of the control action).In such a case, the final control action u0.t applied to the plant is of the form (see Fig. 3): $u'(t)=u'(t-\Delta T)+k_{\Delta u}.\Delta u(t)$, $t \geq 0, u'(t<0)=0$ where $u'(t\text{-}j\cdot\Delta T)$. is a previous control and kDu is a scaling factor for the

change of control $\Delta u(t)$.Once the controller structure, in terms of its inputs and outputs, is established, a collection of the primary fuzzy sets for each input and for the output of the controller must be defined. The primary fuzzy sets (fuzzy clusters or granulas) for a given input or output formally represents the aggregations of the masses of numerical data from the inputs and the output. These sets establish a perception level for the ordinary neural network which is an internal part of the proposed neuro-fuzzy scheme. All learning and inference processes are then carried out at this level. The primary fuzzy sets enable reasoning on a higher (semantic or linguistic) level than in the case of ordinary neural network. These

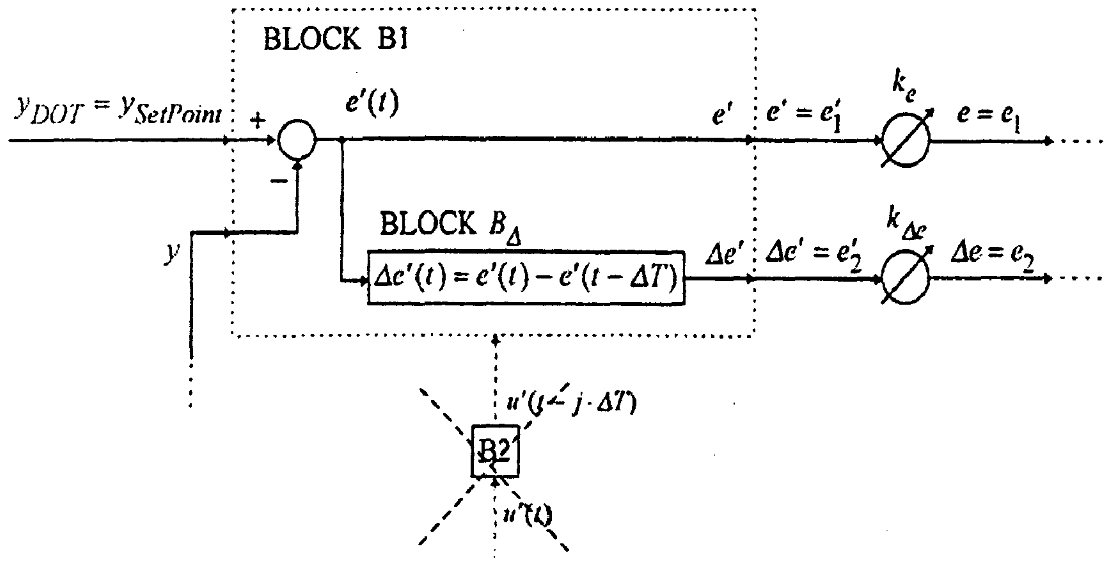


Fig 10: Internal representation of block B1 of neuro-fuzzy controller.

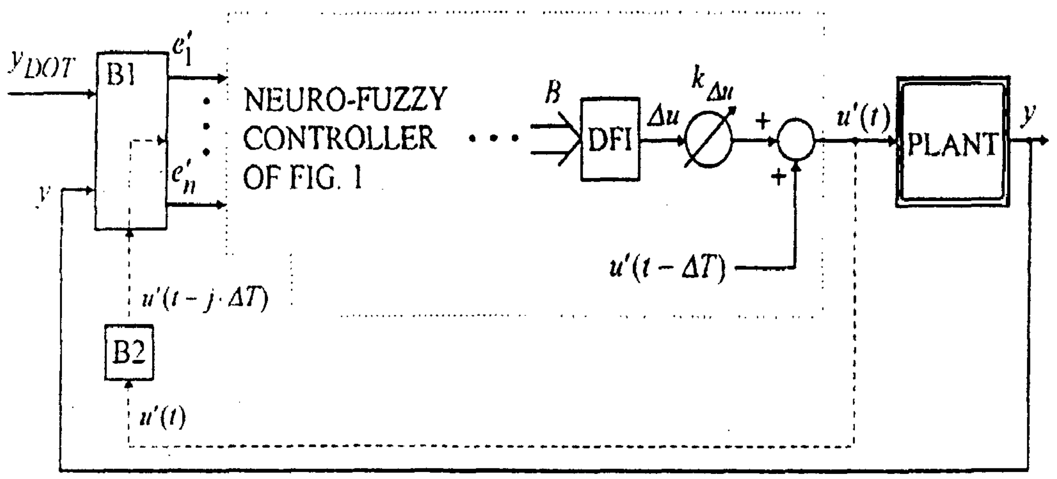


Fig 11:The representation of B1, B2 and DF block with the plant model.

sets can also be used as premises and consequences in the linguistic conditional rules describing the control strategy. If significant amounts of numerical data from the inputs and the output of the controller are available, then the determining of the primary fuzzy sets can be made with the use of the fuzzy clustering technique .However, in general, both the definition of primary fuzzy sets and the choice of model structure utilize a considerable amount of a priori knowledge and rely to significant extent on the 'engineering feel' of the plant to be controlled. For these reasons, this stage of controller design is difficult to be formalized. The second stage of the controller design is the learning of the neuro-fuzzystructure of the controller. The aim of this stage is to incorporate into the neuro-fuzzy system all available knowledge concerning the control strategy of a given plant the qualitative, linguistic, usually rule-based knowledge as well as the quantitative, numerical, nonfuzzy data and relations between them.

3.3 The learning phase of the controller

Consider a general case of the controller of Fig. 1 with n normalized (that is, After applying the scaling factors) inputs $e_1, e_2, .., e_n (e_i \in E_i, i = 1,2,.........,n)$. And one normalized output. $u(u \in U)$ Ei and U are the universes of discourse for fuzzy sets. For input $e_i (i = 1,2,......,n)$,, a collection $A_{i1}, A_{i2,..}, A_{iai} \in F(E_i)$of ai primary fuzzy sets is defined. F(Ei) denotes a family of all fuzzy sets defined on Ei. For output u, a collection $B_1, B_2,..., B_b \in F(U)$ of b primary fuzzy sets is determined. During the learning phase when the switches $S_1^I,,,,,,S_n^I, S_1^{II},...,S_b^{II}$, and$S^{III}$ of Fig. 1 are in the LP positions and the control loop is open, the neuro-fuzzy structure of the controller acquires and accumulates the control knowledge. A part of this knowledge (provided by a human expert) is usually formulated as a set of linguistic conditional rules of the type:

$$\begin{array}{l} (......ALSO) \\ IF(e_1 is A_1^{(k)}) AND(e_2 is A_2^{(k)}) AND.....AND(e_n is A_n^{(k)}) \\ THEN(u is B^{(k)}) \\ (ALSO....) k = 1,2,......., K, \end{array} \qquad (26)$$

where $A_i^{(k)}, i=1,2,\ldots,n$ and $B^{(k)}$.are the linguistic descriptions (like negative big, positive small, close to zero and so on) of the controller inputs e1;e2; . . . ; en and output u for the kth control rule. The symbols $A_i^{(k)}, B^{(k)}$. Also denote here fuzzy sets, which formally represent these descriptions, that is, $A_i^{(k)} \in F(E_i), i=1,2,\ldots\ldots,n$ and $B^{(k)} \in F(U)$ another part of the control knowledge, in a general case, has the form of the sets of the controller input/output measurements

$$(e_1^{(p)}, e_2^{(p)}, \ldots\ldots, e_n^{(p)}, u^{(p)}),\ p=1,2,\ldots\ldots,P, \tag{27}$$

where $e_i^{(p)} \in E_i, i=1,2,\ldots\ldots,n, u^{(p)} \in U$ and p is a number of (n+1)-element measurement sample. In order to underline the cause-effect relationship, data (3) can also be presented in a rule-like form:

(.....ALSO)

$$\text{IF}\left(e_1 \text{is} e_1^{(p)}\right)\text{AND}\left(e_2 \text{is} e_2^{(p)}\right)\text{AND}\ldots\ldots\ldots\text{AND}\left(e_n \text{is} e_n^{(p)}\right)$$

$$\text{THEN}(u \text{is} u^{(p)})$$

$$(\ldots\ldots\text{ALSO})p=1,2,\ldots\ldots,P. \tag{28}$$

$$(\ldots\ldots\ldots\text{ALSO})$$

Fuzzy sets $\tilde{e}_1^{(p)} \in F\left(E_i\right)$, i= 1, 2,…,n, $\tilde{u} \in F(U)$

which formally represent the measurements e.p. i ; u.p., have the form of fuzzy singletons, that is,

$$\mu_{e^{-(p)}}\left(e_i\right) = \{1 \text{for} e_i = e_i^p, 0 \text{for} e_i \neq e_i^p\} \tag{29}$$

where $\mu_{\tilde{e}_i^{(p)}}\left(e_i\right)$ denotes a membership function of the fuzzy set $\tilde{e}_1^p$ (analogously, the fuzzy singleton is defined for $u^{(p)}$. In this way, both the control knowledge (2) and the numerical control data (3) or (4), have a unified form of a fuzzy-set based representation.

In further considerations, for simplicity, we assume that description (2) ± with index k ranging from 1 to K+P -covers both knowledge (2) and data (4). As mentioned earlier, the collections of the primary fuzzy sets for the inputs and for the output, establish the perception level for the ordinary neural network which is an internal part of the proposed neuro-fuzzy controller. This implies that both the fuzzy and nonfuzzy (measurements) data which are to be processed by the neuro-fuzzy system, first must be 'transferred' to the perception level determined by the primary fuzzy sets for the inputs (see INPUT BLOCKS I in Fig. 1) and the output (see INPUT BLOCK II in Fig. 1). The representations of these transferred data are called activation degrees (ADs -for short) of the primary fuzzy sets for particular inputs. These ADs are then processed by the neural network, which generates at its output the Ads $\upsilon_1^{(k)},\dots\dots,\upsilon_b^{(k)}$ of the primary fuzzy sets for the output -see Fig. 1. During the learning phase of the controller design, these output ADs are compared with corresponding so-called desired activation degrees (DADs ± for short) $d_1^{(k)},\dots\dots,d_b^{(k)}$ which are determined for the output portion of the learning data. The errors between the DADs $d_j^{(k)}$ and the Ads $\upsilon_j^{(k)}, j=1,2,\dots\dots,m,$ are then processed by the learning algorithm (see Fig. 1) which adjusts the weights of the neural network in such a way as to reduce the errors to an acceptable level. As for the classical neural network of Fig. 1, we use a multilayer perceptron [17] because of its universal approximation properties [3, 4, 10, 11]. As far as these properties are concerned, the existing literature regarding the number of neurons in hidden layer(s) of the perceptron is, unfortunately, of limited practical usefulness, since it does not give practical indication as to the sufficient number of neurons in the hidden layer for a given problem. Therefore, usually the learning process is being conducted independently for several different numbers of nodes in the hidden layer in order to select the best solution. The back-propagation learning algorithm [17] is used as a learning technique for this network. All available learning data are processed repeatedly by the learning algorithm until the cost function is reduced to an acceptable value. The overall cost function Q which is being minimized during the learning process is a mean square error between DADs and ADs for outputs, that is

$$Q=\frac{1}{(K+P)b}\sum_{k=1}^{k+p}\sum_{j=1}^{b}\left(\upsilon_j^{(k)}-d_j^{(k)}\right)^2 \tag{30}$$

It remains yet to determine how to calculate the ADs and DADs. Consider the controller input ei with the collection of ai primary fuzzy sets $A_{ij}\in F(E_i), j=1,2,\dots.,a_i$ and assume that the fuzzy set $A_i' \in F(E_i)$. is to be represented in terms of these primary sets. The ADs of A_{ij} induced by A_i' can be calculated as the possibility measures $\prod(A_i'/A_{ij})$ of A_i' with respect to Aij:

$$\prod(A_i'/A_{ij}) = \sup\{\min[\mu_{A_i'}(e_i), \mu_{A_{ij}}(e_i)]\} \tag{31}$$

In a particular case of a nonfuzzy numerical data $e_i^o \in E_i$, the fuzzy set A_i' is reduced to a fuzzy singleton (cf. (5)) $\tilde{e}_i^o$, and then expression (7) has the form:

$$\prod(\tilde{e}_i^o/A_{ij}) = \mu_{A_{ij}}(e_i^o) \tag{32}$$

The DADs are calculated in an analogous way. These definitions of ADs and DADs have been incorporated into the structure of Fig. 1.

4. The functioning phase of the controller after the learning phase of the neural network is successfully completed, the switches $S_1^I,,,,,,,S_n^I, S_1^{II},...,S_b^{II}$, and S^{III} of Fig. 1 can be 'shifted' to the FP position, and then the control process starts. However, some additional problems concerning the output part of the controller remain to be solved. The output Ads υ_j j = 1; 2; . . . b; form a controller response expressed in terms of the perception level determined by the output primary fuzzy sets. This response must be 'retranslated' to the level of the output u of the controller. In other words, this response must be expressed in the form of a fuzzy set, say B (in Fig. 1, we call it a control fuzzy set), from the family F .U.. The fuzzy set B - according to our proposition - can be created as follows:

$$\mu_B(u) = \max\left\{\min\left[\mu_{B1}(u), \upsilon_1\right], \ldots\ldots \min\left[\mu_{Bb}(u), \upsilon_b\right]\right\} \tag{33}$$

The set B is a sum (max-operator) of particular primary fuzzy sets for the controller output u modified (min-operators) by corresponding activation degrees υ_j. This way of determining B was chosen because it is compatible to the way of performing the 'translation' in the opposite direction; that is, from the input-output level of the controller to the perception level determined by the primary fuzzy sets. The solution (9) is also similar to the way the output fuzzy set is generated by the compositional rule of inference [18]. Since the controller operates in a closed-loop control mode, it has to generate a nonfuzzy control signal u. For this reason, a defuzzification procedure (a defuzzification interface DFI, see Fig. 1) must be applied to the set B. In the existing literature, several defuzzification methods have been reported, e.g. a method which selects the nonfuzzy output, the membership function value of which corresponds to the maximum or a more effective a centre-of-area method which takes into account the entire shape of the membership function of B ([13] presents the details). Blocks FI of Fig. 1 represent the fuzzification interfaces. Each of them for a given nonfuzzy input $e_i^0 \in E_i$ generates its fuzzy-set representation in the form of a fuzzy singleton in an analogous way as in formula (5). Before the control process starts, we can also assess how the obtained controller (including its output part which has not participated in the learning phase) fits the control knowledge (2) and data (3). In the case of the nonfuzzy data (3), we can apply the following quality index:

$$q_1 = \frac{1}{P}\sum_{p=1}^{P}\left(u^{(p)} - u^0\right)^2 \tag{34}$$

where $u^{(p)}$ as defined in Section 3 (formula (3)) the pth sample of the output portion of control measurements ($p = 1, 2; \ldots ; P$) and u^0 is the controller response corresponding to . In the case of the fuzzy knowledge (2), the criterion of the good-mapping property [6] can be applied. The functioning phase of the neuro-fuzzy controller in the closed-loop control mode directly corresponds to the phase of testing of neural-network based systems. Testing is being performed with the use of the data that have not been utilized at the learning of the neuro-fuzzy system and, therefore, it enables to assess the generalizing properties of the neuro-fuzzy system obtained. The above controller design can be accomplished by using ANFIS Toolbox.

3.4 Adaptive Neuro Fuzzy Inference System (ANFIS)

3.4.1 Introduction

The acronym ANFIS derives its name from *adaptive neuro-fuzzy inference system*. Using a given input/output data set, the toolbox function anfis constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a back propagation algorithm alone, or in combination with a least squares type of method. This allows our fuzzy systems to learn from the data they are modeling.

ANFIS (Adaptive Neuro Fuzzy Inference System) is an architecture, which is functionally equivalent to a Sugeno type fuzzy rule base (Jang [10], Sun & Mizutani, 1997; Jang & Sun, 1995). Under certain minor constraints the ANFIS architecture is also equivalent to a radial basis function network. Loosely speaking ANFIS is a method for tuning an existing rule base with a learning algorithm based on a collection of training data. This allows the rule base to adapt. The network in Figure. 8 may be extended by assigning a linear function to the output weight of each neuron, $w_k = a_k^T u + b_k$, k=1, 2,.......,K where $a_k \in R^m$ is a parameter vector and en is a scalar parameter. The network is then equivalent to a first order Sugeno type fuzzy rule base (Takagi and Sugeno, 1985). The requirements for the radial basis function network to be equivalent to a fuzzy rule base is summarised in the following (Jang et al., 1997)

1. Both must use the same aggregation method (weighted average or weighted sum) to derive their overall outputs.
2. The number of activation functions must be equal to the number of fuzzy if-then rules.
3. When there are several inputs in the rule base, each activation function must be equal to a composite input membership function. One way to achieve this is to employ Gaussian membership functions with the same variance in the rule base, and apply product for the DQG operation. The multiplication of the Gaussian membership functions becomes a multi-dimensional Gaussian radial basis function.
4. Corresponding activation functions and fuzzy rules should have the same functions on the output side of the neurons and rules respectively.

If the training data are contained in a small region of the input space, the centers of the neurons in the hidden layer can be concentrated within the region and sparsely cover the remaining area. Thus, only a local model will be formed and if the test data lie outside the region, the performance of the network will be poor. On the other hand, if one distributes the basis function centres evenly throughout the input space, the number of neurons depends exponentially on the dimension of the input space.

3.4.2 ANFIS architecture and learning algorithm

Without loss of generality we assume two inputs, u_1 and u_2 and one output, y. Assume for now a first order Sugeno type of rule base with the following two rules

If u_1 is A_1 and u_2 is B_1 **then** $y_1 = c_{11}u_1 + c_{12}u_2 + c_{10}$

If u_1 is A_2 and u_2 is B_2 **then** $y_2 = c_{21}u_1 + c_{22}u_2 + c_{20}$

Incidentally, this fuzzy controller could interpolate between two linear controllers depending on the current state. If the firing strengths of the rules are α_1 and α_2 respectively, for two particular values of the inputs u_1 and u_2 then the output is computed as a weighted average

$$y = \frac{\alpha_1 y_1 + \alpha_2 y_2}{\alpha_1 + \alpha_2} = \bar{\alpha}_1 y_1 + \bar{\alpha}_2 y_2$$

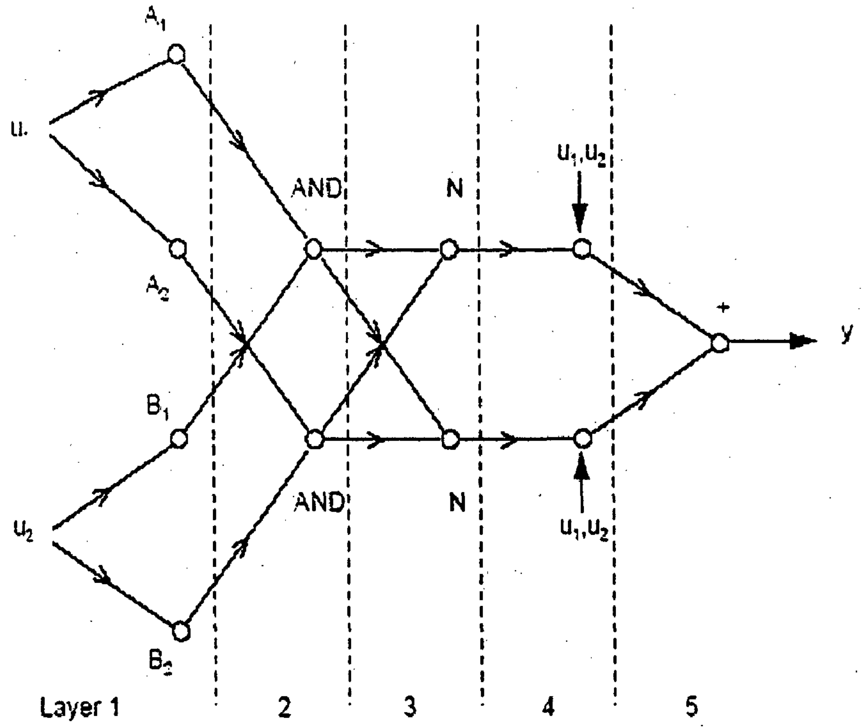


Fig 12: The layer structure of the ANFIS network.

The corresponding ANFIS network is shown in Figure.12. A description of the layers in the network follows.

1. Each neuron i in layer 1 is adaptive with a parametric activation function. Its output is the grade of membership to which the given input satisfies the membership function, i.e. ,. An example of a membership function is the generalised bell function

$$\mu(x)=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{35}$$

where $\{a,b,c\}$ is the parameter set. As the values of the parameters change, the shape of the bell-shaped function varies. Parameters in that layer are called premise parameter

2. Every node in layer 2 is a fixed node, whose output is the product of all incoming signals. In general, any other fuzzy AND operation can be used. Each node output represents the firing strength α_i of the i th rule.

3 Every node in layer 3 is a fixed node which calculates the ratio of the Ith rule's firing strength relative to the

sum of all rule's firing strengths,

$$\bar{\alpha}_i=\frac{\alpha_i}{\alpha_i+\alpha_2},i=1,2 \tag{36}$$

The result is a normalized firing strength

4. Every node in layer 4 is an adaptive node with a node output $\bar{\alpha}_i y_i=\bar{\alpha}_i(c_{i1}u_1+c_{i2}u_2+c_{i0})$ i=1, 2 .Where $\bar{\alpha}_i$ is the normalized firing strength from layer 3 and $\{c_{i1},c_{i2},c_{i0}\}$ is the parameter set of this node. Parameters in this layer are called consequent parameters. 5. Every node in layer 5 is a fixed node, which sums all incoming signals. It is straightforward to generalise the ANFIS architecture in figure.12 to a rule base with more than two rules.

3.4.3 The ANFIS learning algorithm

When the premise parameters are fixed, the overall output is a linear combination of the consequent parameters. In symbols, the output y can be written as

$$y=\frac{\alpha_1}{\alpha_1+\alpha_2}y_1+\frac{\alpha_2}{\alpha_1+\alpha_2}y_2$$
$$y=\bar{\alpha}_1(c_{11}u_1+c_{12}u_2+c_{10})+\bar{\alpha}_2(c_{21}u_1+c_{22}u_2+c_{20}) \qquad (37)$$
$$y=(\bar{\alpha}_1u_1)c_{11}+(\bar{\alpha}_1u_2)c_{12}+\bar{\alpha}_1c_{10}+(\bar{\alpha}_2u_2)c_{21}+(\bar{\alpha}_2u_2)c_{22}+\bar{\alpha}_2c_{20}$$

which is linear in the consequent parameters c_{ij}(i =1, 2 ...j = 0, 1, 2...) hybrid algorithm adjusts the consequent parameters c_{ij} in a forward pass and the premise parameters $\{a_i, b_i, c_i\}$ in a backward pass (Jang et al., 1997)[10]. In the forward, pass the network inputs propagate forward until layer 4, where the consequent parameters are identified by the least-squares method. In the backward pass, the error signals propagate backwards and the premise parameters are updated by gradient descent. Because the update rules for the premise and consequent, parameters are decoupled in

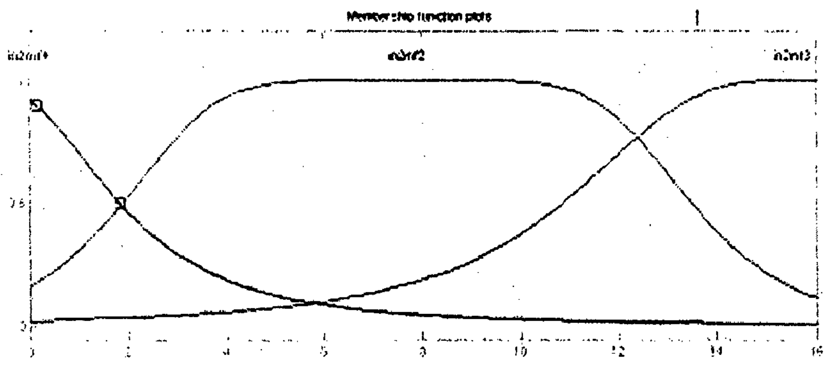


Figure 13: the membership functions before learning

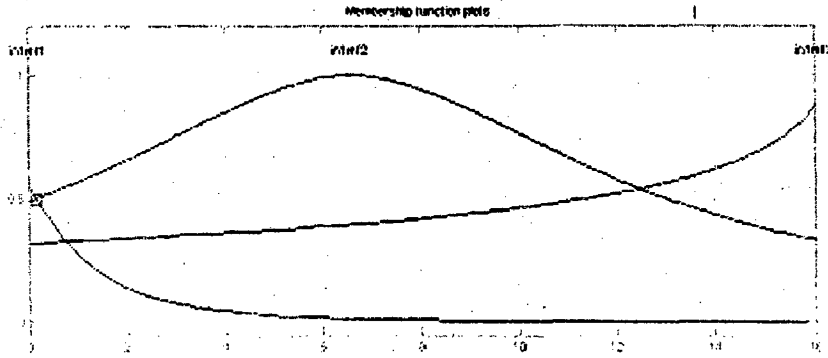


Figure 14: the membership functions after learning.

the hybrid learning rule, a computational speedup may be possible by using variants of the gradient method or other optimisation techniques on the premise parameters. Since ANFIS and radial basis function networks (RBFNs) are functionally equivalent, a variety of learning methods can be used for both of them [19]. Figure 13 and 14 shows the membership function of the input before training and after training.

3.5 FIS Structure and Parameter Adjustment

A network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map. The parameters associated with the membership functions will change through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure (usually defined by the sum of the squared difference between actual and desired outputs). Anfis uses either back propagation or a combination of least squares estimation and back propagation for membership function parameter estimation.

3.6 Validation

The modeling approach used by anfis is similar to many system identification techniques. First, we hypothesize a parameterized model structure (relating inputs to membership functions to rules to outputs to membership functions, and so on). Next, we collect input/output data in a form that will be usable by anfis for training. We can then use anfis to *train* the FIS model to emulate the training data presented to it by modifying the membership function parameters according to a chosen error criterion. In general, this type of modeling works well if the training data presented to anfis for training (estimating) membership function parameters is fully representative of the features of the data that the trained FIS is intended to model. This is not always the case, however. In some cases, data is collected using noisy measurements, and the training data cannot be representative of all the features of the data that will be presented to the model. This is where *model validation* comes into play.

3.7 Model Validation Using Checking and Testing Data Sets

Model validation is the process by which the input vectors from input/output data sets on which the FIS was not trained, are presented to the trained FIS model, to see how well the FIS model predicts the corresponding data set output values. This is accomplished with the ANFIS Editor GUI using the so-called *testing data set*, and its use is described in a subsection that follows. We can also use another type of data set for model validation in anfis. This other type of validation data set is referred to as the *checking data set* and this set is used to control the potential for the model over fitting the data. When checking data is presented to anfis as well as training data, the FIS model is selected to have parameters associated with the minimum checking data model error.

One problem with model validation for models constructed using adaptive techniques is selecting a data set that is both representative of the data the trained model is intended to emulate, yet sufficiently distinct from the training data set so as not to render the validation process trivial. If we have collected a large amount of data, hopefully this data contains all the necessary representative features, so the process of selecting a data set for checking or testing purposes is made easier. However, if we expect to be presenting noisy measurements to the model, it's possible the training data set does not include all of the representative features we want to model.

The basic idea behind using a checking data set for model validation is that after a certain point in the training, the model begins over fitting the training data set. In principle, the model error for the checking data set tends to decrease as the training takes place up to the point that over fitting begins, and then the model error for the checking data suddenly increases.

3.8 Constraints of anfis

Anfis is much more complex than the fuzzy inference systems discussed so far, and is not available for all of the fuzzy inference system options. Specifically, anfis only supports Sugeno-type systems, and these must have the following properties:

•Be first or zeroth order Sugeno-type systems.

•Have a single output, obtained using weighted average defuzzification. All output membership functions must be the same type and be either linear or constant.
•Have no rule sharing. Different rules cannot share the same output membership function, namely the number of output membership functions must be equal to the number of rules.
•Have unity weight for each rule
An error occurs if the FIS structure does not comply with these constraints. Moreover, anfis cannot accept all the customization options that basic fuzzy inference allows. That is, we cannot make our own membership functions and defuzzification functions; we must use the ones provided.

3.9 ANFIS Editor GUI

The basic structure of the type of fuzzy inference system that we've seen thus far is a model that maps input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single-valued output or a decision associated with the output. We have only considered membership functions that have been fixed, and somewhat arbitrarily chosen. Also, we've only applied fuzzy inference to modeling systems whose rule structure is essentially predetermined by the user's interpretation of the characteristics of the variables in the model.
In this section we discuss the use of the function anfis and the ANFIS Editor GUI in the Fuzzy Logic Toolbox. These tools apply fuzzy inference techniques to data modeling. As we have seen from the other fuzzy inference GUIs, the shape of the membership functions depends on parameters, and changing these parameters will change the shape of the membership function. Instead of just looking at the data to choose the membership function parameters, we will see how membership function parameters can be chosen automatically using these Fuzzy Logic Toolbox applications.

Suppose we want to apply fuzzy inference to a system for which we already have a collection of input/output data that we would like to use for modeling, model-following, or some similar scenario. We don't necessarily have a predetermined model structure based on characteristics of variables in the system. There will be some modeling situations in which we can't just look at the data and discern what the membership functions should look like. Rather than choosing the parameters associated with a given membership function arbitrarily, these parameters could be chosen so as to tailor the membership functions to the input/output data in order to account for these types of variations in the data values. This is where the so-called *neuro-adaptive*learning techniques incorporated into anfis in the Fuzzy Logic Toolbox can help.

The basic idea behind these neuro-adaptive learning techniques is very simple. These techniques provide a method for the fuzzy modeling procedure to *learn* information about a data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. This learning method works similarly to that of neural networks. The Fuzzy Logic Toolbox function that accomplishes this membership function parameter adjustment is called anfis..

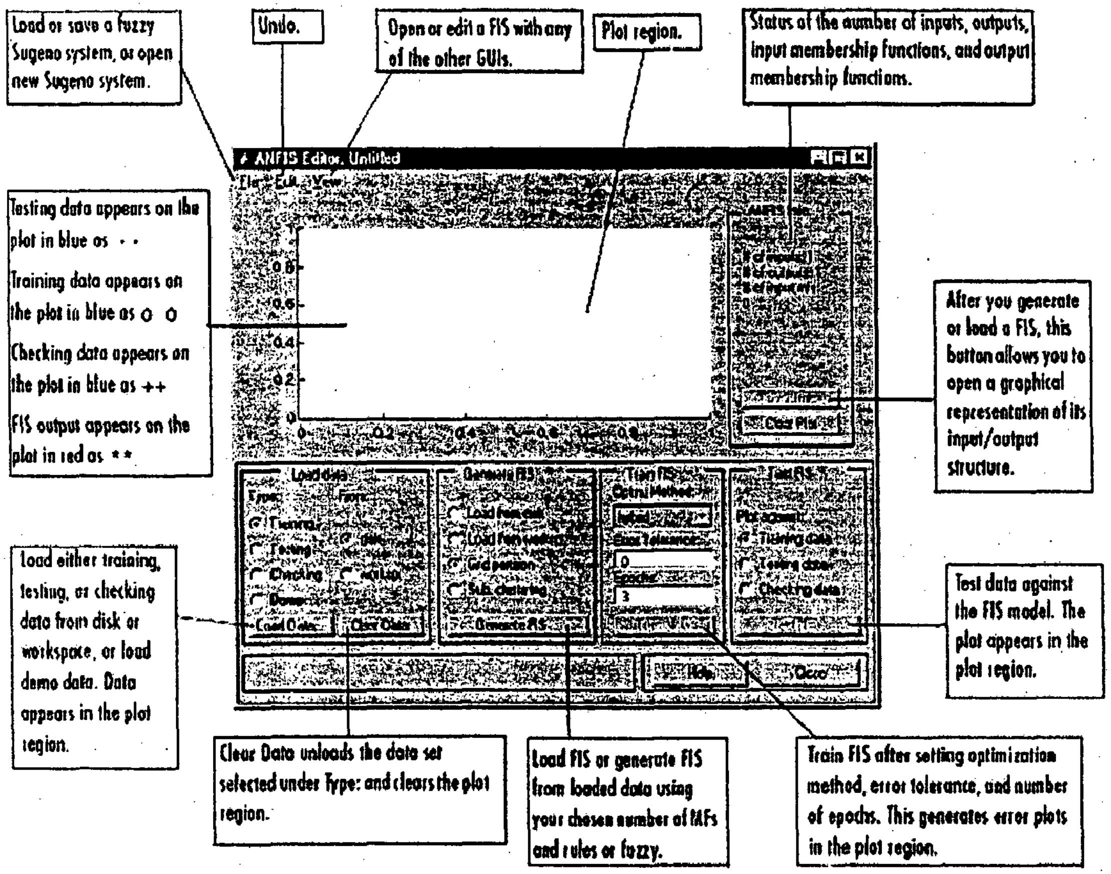

Load or save a fuzzy Sugeno system, or open new Sugeno system.
Undo.
Open or edit a FIS with any of the other GUIs.
Plot region.
Status of the number of inputs, outputs, input membership functions, and output membership functions.
ANFIS Editor: Untitled
Testing data appears on the plot in blue as · ·
Training data appears on the plot in blue as o o
Checking data appears on the plot in blue as ++
FIS output appears on the plot in red as **
After you generate or load a FIS, this button allows you to open a graphical representation of its input/output structure.
Load either training, testing, or checking data from disk or workspace, or load demo data. Data appears in the plot region.
Test data against the FIS model. The plot appears in the plot region.
Clear Data unloads the data set selected under Type: and clears the plot region.
Load FIS or generate FIS from loaded data using your chosen number of MFs and rules or fuzzy.
Train FIS after setting optimization method, error tolerance, and number of epochs. This generates error plots in the plot region.

Fig 15 shows the Graphic user interface of ANFIS Editor.

By this GUI(graphic user interface):

•Load data (training, testing, and checking) by selecting appropriate radio buttons in the **Load data** portion of the GUI and then clicking Load Data... The loaded data is plotted on the plot region.

•Generate an initial FIS model or load an initial FIS model using the options in the **Generate FIS** portion of the GUI

•View the FIS model structure once an initial FIS has been generated or loaded by clicking the **Structure** button

•Choose the FIS model parameter optimization method: backpropagation or a mixture of back propagation and least squares (hybrid method)

•Choose the number of training epochs and the training error tolerance

•Train the FIS model by clicking the **Train** now button This training adjusts the membership function parameters and plots the training (and/or checking data) error plot(s) in the plot region.

•View the FIS model output versus the training, checking, or testing data output by clicking the **Test Now** button

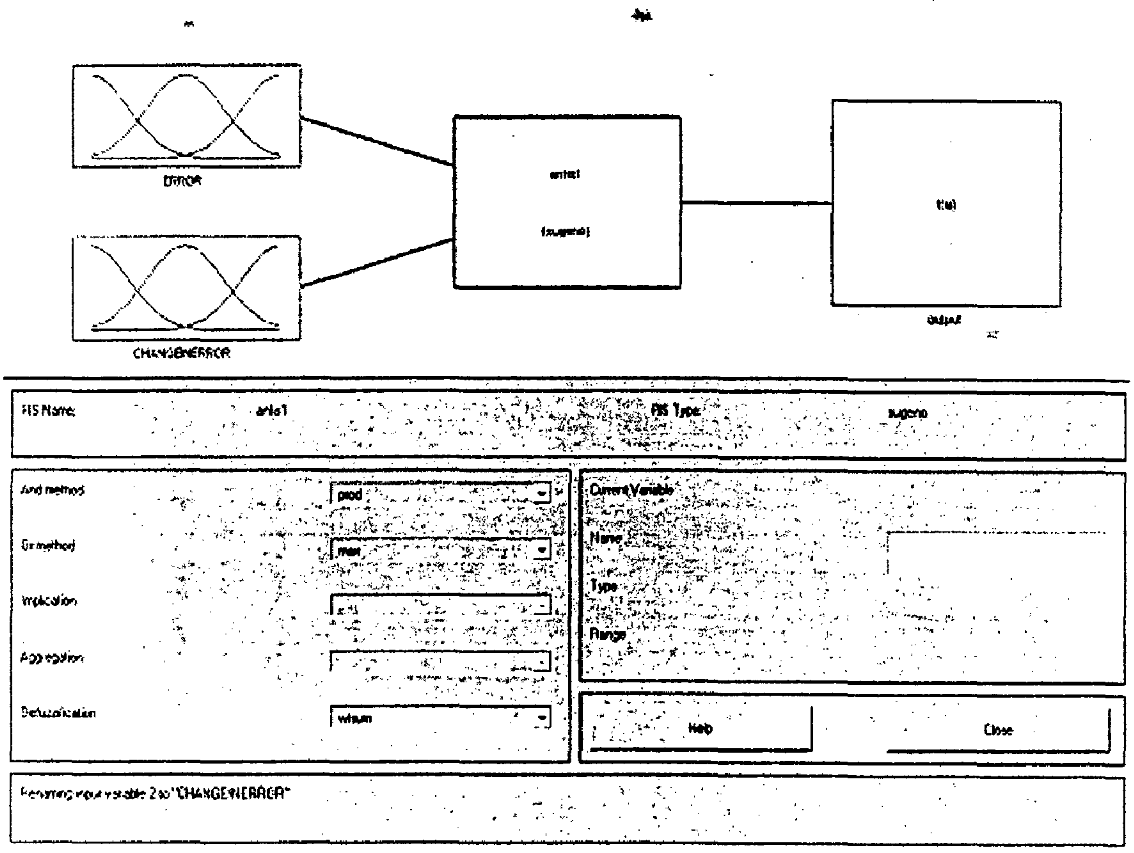
This function plots the test data against the FIS output in the plot region. We can also use the ANFIS Editor GUI menu bar to load FIS training initializations, save the trained FIS, open a new Sugeno system, or open any of the other GUIs to interpret the trained FIS model. In this work the ANFIS Editor is used for designing the controller for single link manipulator and three link SCARA manipulator.The training data(such as error, change in error(input sgnal) and control signal(output signal))is collected from the conventional controller model as shown in fig 36. In this section, the tracking and adaptability features of the ANFIS control applied to a three-link SCARA manipulator are tested using simulation. Figure. 17 and 18 show the architecture of the fuzzy system with the ANFIS approach .The ANFIS methodology is used to estimate the parameters of the membership functions and the consequent functions. The nine rules are used to model the fuzzy part of the ANFIS controller as shown in Figure 18, 19 and 20 show the three membership functions for each linguistic variable.

The fuzzy rules generated by the ANFIS method are shown in Figure.21 .Figure. 23 and 24 show the results of applying the ANFIS methodology with the training data &

trained results. The ANFIS structure is trained for 50 epochs and the performance (MSE) is found to be 0.0064759. Figure.22 the fuzzy rule viewer of MATLAB, which shows the use of the fuzzy system for calculating the output of the model for specific input values. Figure. 37 and 43. shows the response of the point to point control to a sequence of step (constant) input signals (we use 400 samples) .The figure.31, 39 and 41 show the simulation model of three-link SCARA Manipulator Figure. the response of the three-link SCARA manipulator for continuous path control with the sine wave trajectory.

Finally, the figure.42, 45 and 47, 49,51 show the plot of the difference between both the conventional PD controller and the signal by the ANFIS. The results are compared with a classical PD controller and with an ANFIS (sugeno) controller, to measure how much the adaptive fuzzy approach could improve the performance. Of course, the neuro-fuzzy controller (designed with ANFIS) was better in tracking and adaptability than the other controllers. Another advantage of this method over classical quantitative controllers is that it does not require a fixed sampling time. Therefore, the proposed design confirms the fact that ANFIS control is relevant to the control fast of non-linear processes such as robot manipulator controls where quantitative methods are not always appropriate. From the response shown in figure 42, 45,47, ,49, ,51,52, and 53 it is very clear that ANFIS controller gives no tracking error,i.e the response of the desired trajectory is almost superimposed with the actual one, Thus the ANFIS controller gave the best results when compare to conventional PD controller.

In this work, the feasibility of ANFIS control for a three-link SCARA manipulator has been proved and illustrated by simulation. The best parameters for the fuzzy controller were determined by using the ANFIS methodology and by using simulations of the SCARA manipulator dynamics. A simulation tool (i.e., Fuzzy logic toolbox (ANFIS)) was used to validate experimentally the tracking ability and the insensibility to plant parameter changes. The ANFIS controller presented very interesting tracking features and was able to respond to different dynamic conditions. In addition, the fuzzy control computation is very inexpensive, and this regulator could be used for the control of machine tools and robotics manipulators [11] without significantly increasing the cost of the drive. The only extra cost is for the optical encoder. Another advantage of this method over classical quantitative controllers is that it does not require a fixed sampling time. Therefore, the proposed design confirms the fact that fuzzy control is relevant to the fast control of non-linear processes such as SCARA manipulator control where quantitative methods are not always appropriate. Thus, the results obtained using the ANFIS controllers are encouraging when compared to conventional PD controller



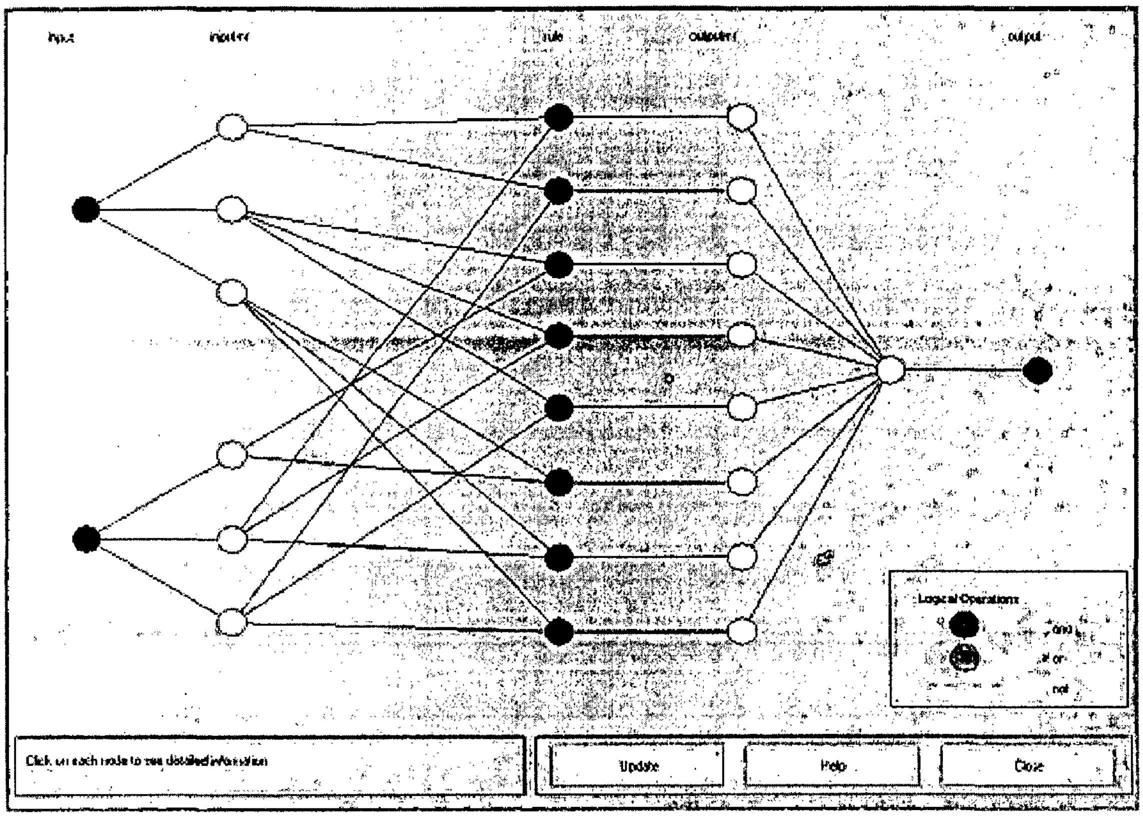
FIS Name: anfis1 FIS Type: sugeno

And method: prod
Or method: max
Implication: [illegible]
Aggregation: [illegible]
Defuzzification: wtaver

Current Variable
Name
Type
Range

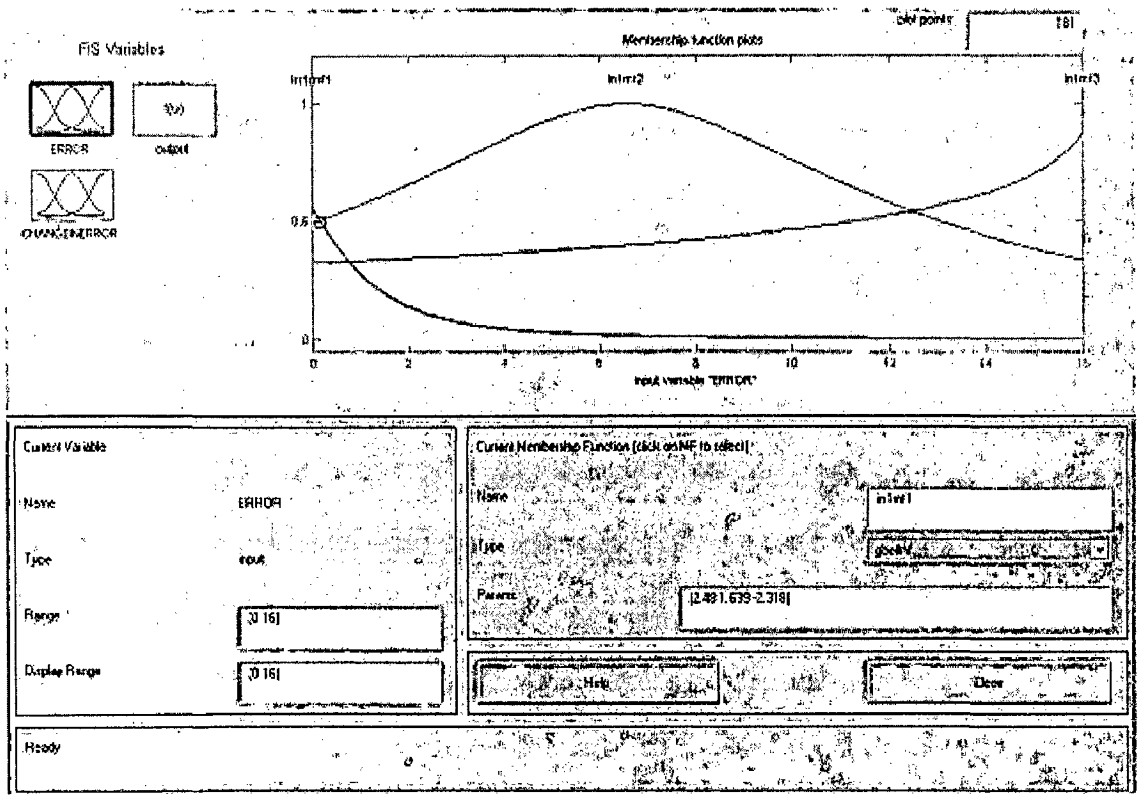
Renaming input variable 2 to "CHANGEINERROR"

Fig 16: Architecture of NEURO FUZZY (sugeno) inference system.



Click on each node to see detailed information

Fig17: Neuro fuzzy structure of ANFIS controller



Current Variable

Name ERROR

Type input

Range [0 16]

Display Range [0 16]

Current Membership Function (click on MF to select)

Name in1mf1

Type gbellmf

Params [2.481 1.639 -2.318]

Ready

Fig 18: Input membership function after training

4.2 Dynamic model of Single link Manipulator:

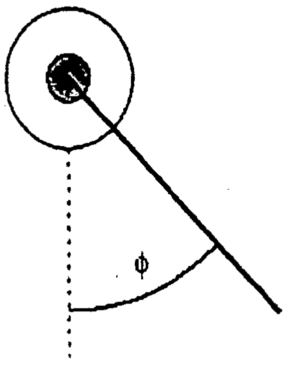


Fig 25:The scheme of single link manipulator

The single link assumed to be a thin homogenous rod of mass m and length l.In this case there is no velocity coupling terms due to coriolis and centrifugal force because there is only one axis

The equation of motion for the arm is

$$\frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u \tag{38}$$

Where ϕ is the angle of the arm, and u is the torque supplied by the DC motor. The objective is to train the controller so that the arm tracks the reference trajectory.

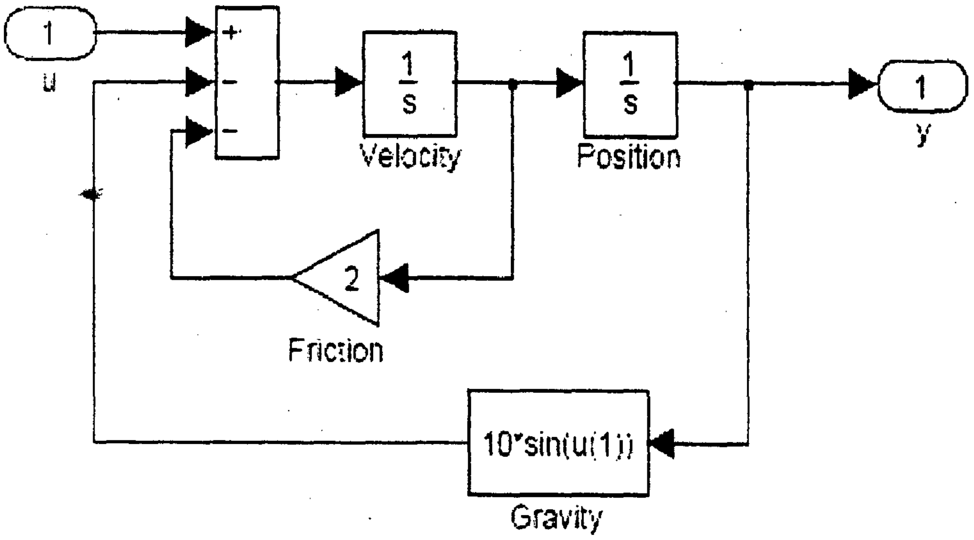
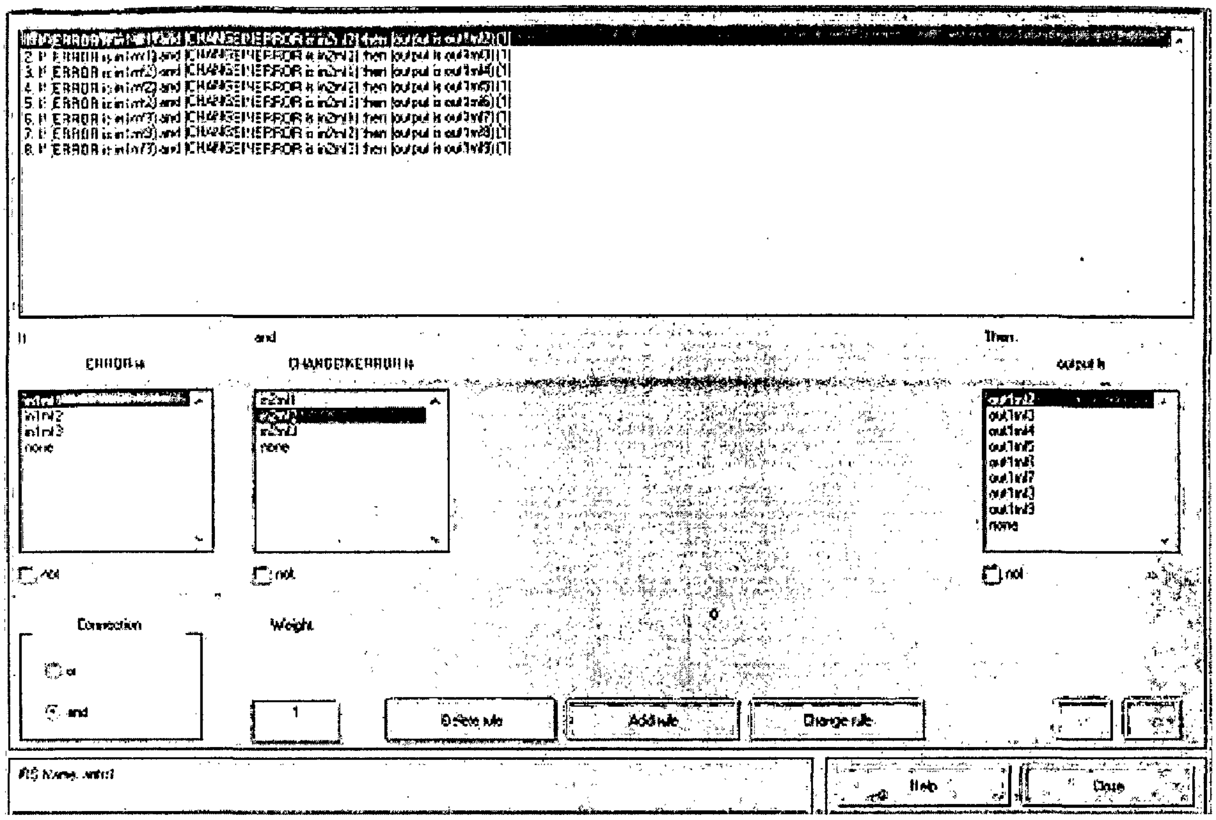


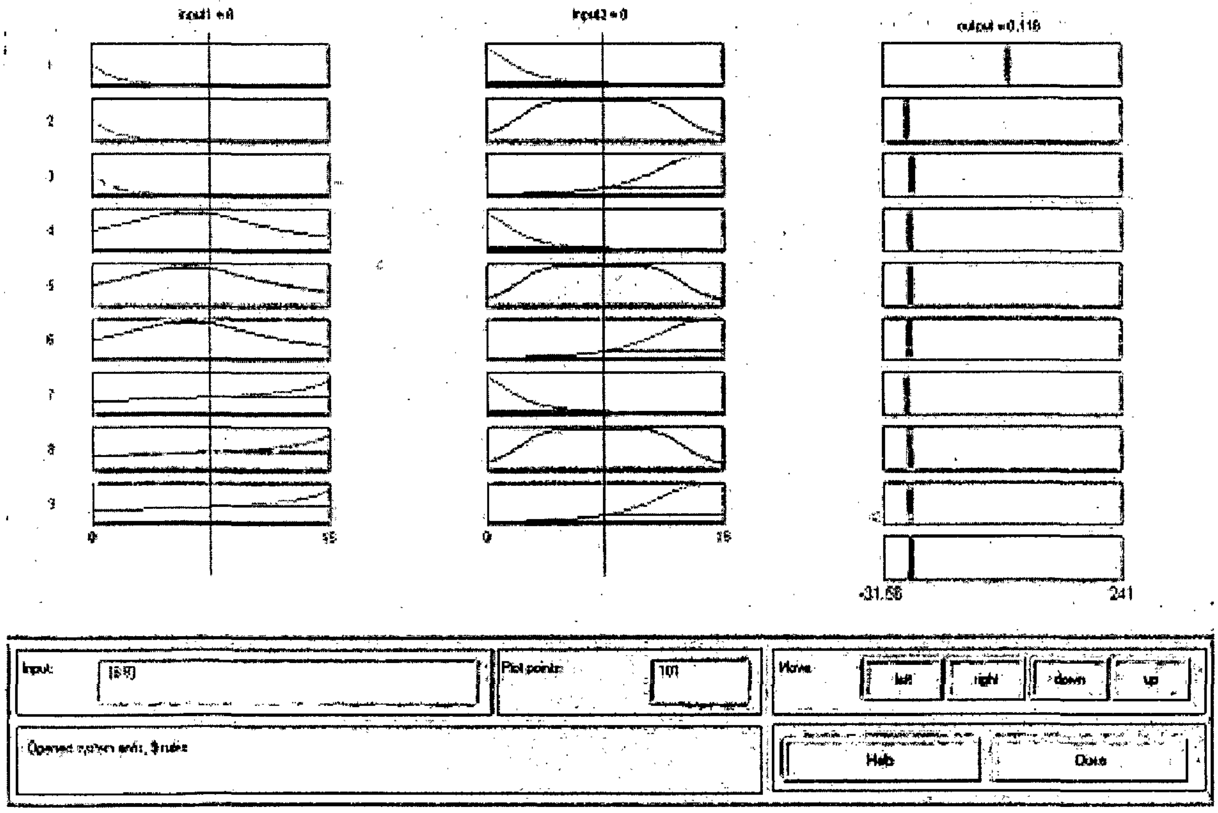
Fig 26: Simulink Model of Single Link Manipulator.



1. If (ERROR is [illegible]) and (CHANGEINERROR is [illegible]) then (output is out1mf2) (1)
2. If (ERROR is in1mf1) and (CHANGEINERROR is in2mf2) then (output is out1mf3) (1)
3. If (ERROR is in1mf2) and (CHANGEINERROR is in2mf1) then (output is out1mf4) (1)
4. If (ERROR is in1mf2) and (CHANGEINERROR is in2mf2) then (output is out1mf5) (1)
5. If (ERROR is in1mf2) and (CHANGEINERROR is in2mf3) then (output is out1mf6) (1)
6. If (ERROR is in1mf3) and (CHANGEINERROR is in2mf1) then (output is out1mf7) (1)
7. If (ERROR is in1mf3) and (CHANGEINERROR is in2mf2) then (output is out1mf8) (1)
8. If (ERROR is in1mf3) and (CHANGEINERROR is in2mf3) then (output is out1mf9) (1)

FIS Name: anfis1

Fig 21: Representation of rule base of ANFIS Controller.

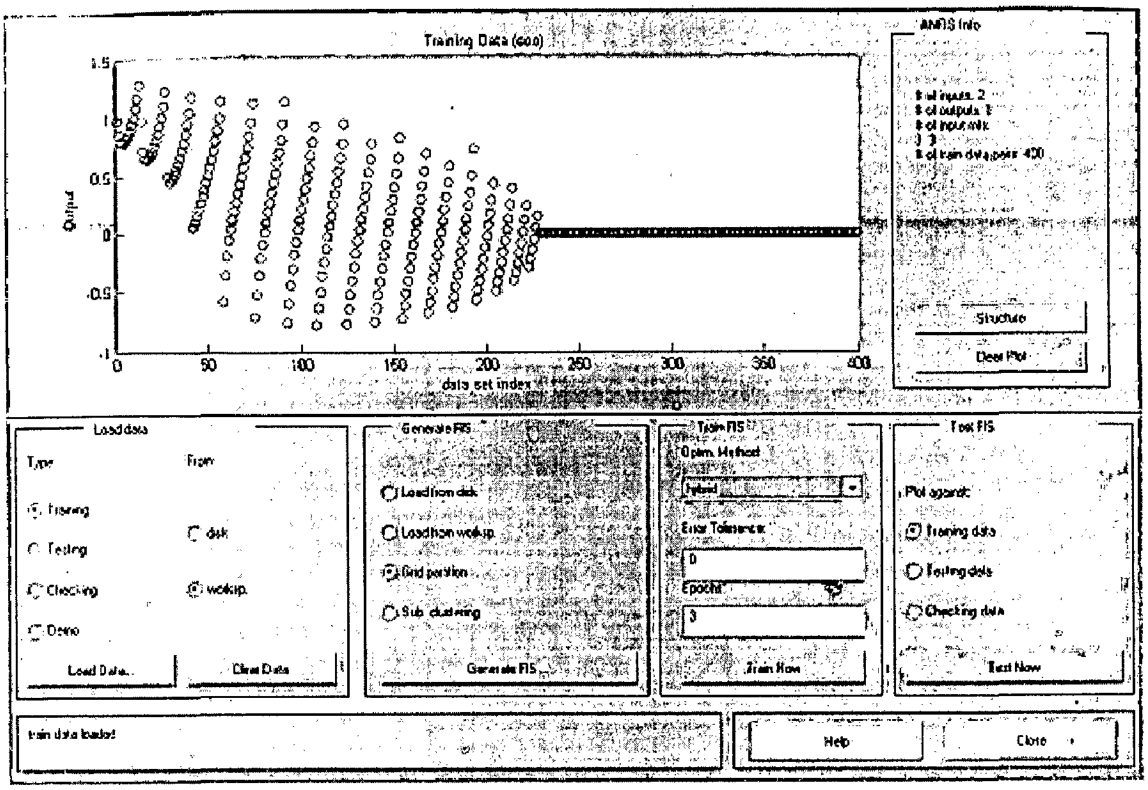


Input: [8 8]

Plot points: 101

Opened system anfis, 9 rules

Fig 22: Rule viewer of ANFIS structure.

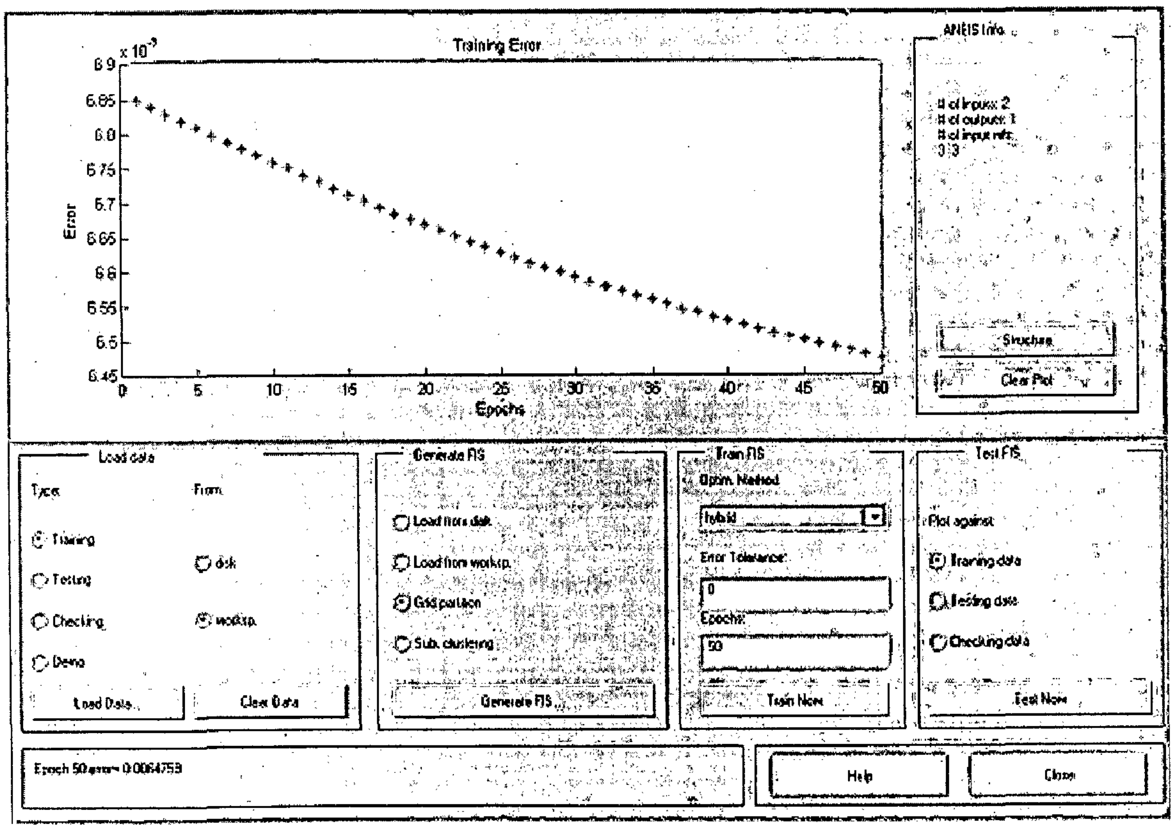


# of inputs: 2
# of outputs: 1
# of input mfs:
3 3
# of train data pairs: 400

Type: [✓ Training] Testing Checking Demo
From: disk [✓ worksp.]
[✓ Grid partition]
Optim. Method: hybrid
Error Tolerance: 0
Epochs: 3
Plot against: [✓ Training data]

train data loaded

Fig 23: Loading of Training data



# of inputs: 2
# of outputs: 1
# of input mfs:
3 3

Type: [✓ Training] Testing Checking Demo
From: disk [✓ worksp.]
[✓ Grid partition]
Optim. Method: hybrid
Error Tolerance: 0
Epochs: 50
Plot against: [✓ Training data]

Epoch 50:error= 0.0064753

Fig 24: Training when error tolerance is chosen to be 0 and number of epochs is limited to 50.

CHAPTER- 4

DYNAMICS OF ROBOT MANIPULATOR

4.1 Introduction

The manipulator system is a classic control problem that is used industries around the world. It is a suitable process to test prototype controllers due to its high non-linearities and lack of stability. In this chapter, the dynamical equations of the system will be derived, the model will be developed in simulink and basic controllers will be developed. The aim of developing a Robot system in simulink is that the developed model will have the same characteristics as the actual process. It will be possible to test each of the prototype controllers in the simulink environment. Before the robot model can be developed in simulink, the system dynamical equations will be derived using 'Lagrange Equations'. [1] The Lagrange equations are one of many methods of determining the system equations. Using this method, it is possible to derive dynamical system equations for a complicated mechanical system such as the Robot manipulator. The Lagrange equations use the kinetic and potential energy in the system to determine the dynamical equations of the robot system.

In this work, two kinds of robot systems are considered viz, Single link robot manipulator and Three link SCARA manipulator. The **SCARA** model chosen in this work has two revolute joints and one prismatic joint (in the configuration RRP) to position the wrist. However, for a SCARA robot, the axes of all three joints are vertical, as shown in fig 27 the first revolute joint swings the arm back and fourth about a base axis that can also be thought of as a vertical shoulder axis. The second revolute joint swings the forearm back and fourth about a vertical elbow axis. Thus, the two revolute joints control motion in a horizontal plane. The vertical component of the motion is provided by the third joint, a prismatic joint which slides the wrist up and down.

4.2 Dynamic model of Single link Manipulator:

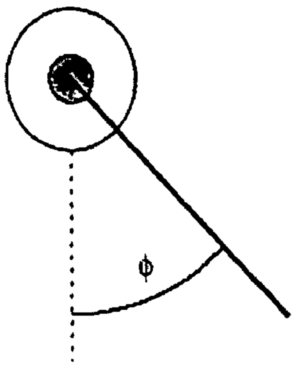


Fig 25:The scheme of single link manipulator

The single link assumed to be a thin homogenous rod of mass m and length l.In this case there is no velocity coupling terms due to coriolis and centrifugal force because there is only one axis

The equation of motion for the arm is

$$\frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} - u \tag{38}$$

Where ϕ is the angle of the arm, and u is the torque supplied by the DC motor. The objective is to train the controller so that the arm tracks the reference trajectory.

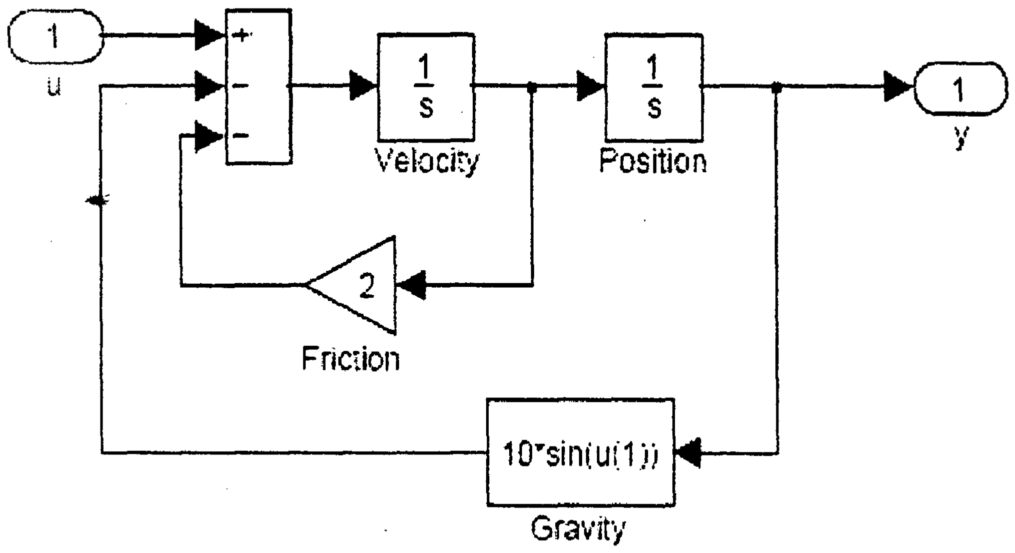


Fig 26: Simulink Model of Single Link Manipulator.

4.3 Dynamic model Of Three link SCARA Manipulator

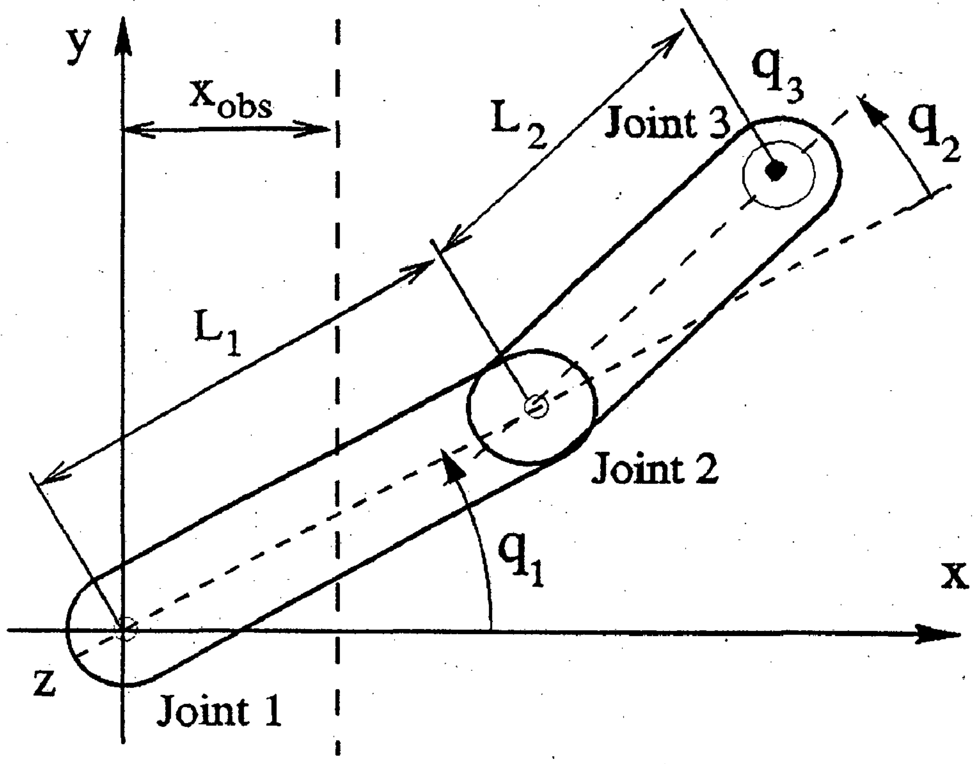


Fig 27: The arrangement of three-link SCARA manipulator in three-dimensional co-ordinates.

4.3.1Background: Mechanical and mechatronic system often result in an implicit second order model description of the type

$$M(\vec{q})\ddot{\vec{q}} = \vec{g}(\dot{\vec{q}}, \vec{q}, \vec{u}, t) \tag{39}$$

with a state-dependent mass matrix M, an acceleration vector $\ddot{\vec{q}}$ and a generalized force function$\vec{g}$. Simulators often impose restrictions for this type of model descriptions. Only a few simulators accept the description as given above, some allow a description as an implicit first order system.

$$A(\vec{z})\dot{\vec{z}} = \vec{h}(\vec{z}, \vec{u}, t)$$

and some require the explicit description given by

$$\dot{\vec{z}} = \vec{f}(\vec{z}, \vec{u}) = A(\vec{z})^{-1}\vec{h}(\vec{z}, \vec{u}, t) \tag{40}$$

The symbolic derivation of the explicit form is only possible with reasonable effort for very small systems or systems with a simple-structured mass matrix. Therefore it is common practice to carry out the inversion of the mass matrix numerically.

Another interesting question is, whether a simulator that permits implicit descriptions breaks the implicit loop before integrating the states or uses an implicit integration scheme to solve the system directly. Few simulators offer so-called DAE solvers for the second method, sometimes with restrictions with respect to other features like linearization, event handling etc. In general, advanced features like implicit description, DAE solvers, algebraic loop solvers etc. result in higher computation times and in some computational overhead. Therefore it has to be checked whether it is worth to use such a tool or to work "conventionally" by setting up an explicit system description. In order to investigate this class of problems, a model for a SCARA robot (Selective Compliance Assembly Robot Arm) as shown on the title page of this SNE issue was chosen. Fig. 27

4.4 Mechanical system of three link SCARA (Task a)

A three-axis SCARA robot as indicated in Fig. 1 is investigated. This robot type has two vertical revolute joints and one vertical prismatic joint. The axes of all three joints are vertical (parallel to z-axis in fig 27). The joint vector $\vec{q}$ consists of the joint angle q_1 and q_2 and the joint distance q_3.

$$\vec{q} = (q_1, q_2, q_3)^T, \quad \dot{\vec{q}} = \frac{d\vec{q}}{dt}, \quad \ddot{\vec{q}} = \frac{d\dot{\vec{q}}}{dt} \tag{41}$$

The equations of motion of can be written in the following compact form

$m(q)\ \ddot{q} + h(q,\ \dot{q})\dot{q} + g(q)\ = \tau$

(Or)

$$m(q)\ddot{q} = \vec{b} \text{ (Implicit equation)} \tag{42}$$

The mass matrix M is block –diagonal and can be easily inverted symbolically.

$$M = \begin{bmatrix} ma_{11} & ma_{12} & 0 \\ ma_{21} & ma_{22} & 0 \\ 0 & 0 & ma_{33} \end{bmatrix} \tag{43}$$

Several elements of M depend on the joint variable q_2

$$\begin{aligned} ma_{11} &= \Theta_1 + 2\Theta_2\cos(q_2) + \Theta_3, \\ ma_{12} &= \Theta_2\cos(q_2) + \Theta_3, \\ ma_{21} &= ma_{12}, ma_{22} = \Theta_3 \\ ma_{33} &= m_{3L} + \Theta_{3mot}u_3^2. \end{aligned} \tag{44}$$

The calculation of the moments of inertia Θ_i is based on the assumption that the two physical links are rods of mass m_1, m_2 with homogeneous mass distribution along the length L_1, L_2. The stator mass of the vertical drive motor is m_{3A}, the moment of inertia of the rotating parts is Θ_{3mot} and the mass of the load is m_{3L}.

$$\begin{aligned} &\Theta_1 = \left(\frac{m_1}{3} + m_2 + m_3\right)L_1^2, \Theta_2 = \left(\frac{m_2}{2} + m_3\right)L_1L_2, \\ &\Theta_3 = \left(\frac{m_2}{3} + m_3\right)L_2^2, m_{3A} + m_{3L} \end{aligned} \tag{45}$$

The right-hand side of the dynamic equation is

$$\begin{aligned} &\vec{b} = (b_1, b_2, b_3)^T, \\ &b_1 = T_1 + \Theta_2\left(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2\right)\sin(q_2), \\ &b_2 = T_2 - \Theta_2\dot{q}_1^2\sin(q_2), b_3 = T_3 - m_{3L}g \end{aligned} \tag{46}$$

with the joint torques $T_1(t)$, $T_2(t)$ and the joint force $T_3(t)$. Numerical data for the geometric and mass parameters of the SCARA robot are given below:

$$\begin{aligned} &m_1 = 8\text{kg}, L_1 = 0.4\text{m}, g = 9.81\text{m/s}^2, \\ &m_2 = 6\text{kg}, L_2 = 0.3\text{m}, u_3 = 1047\text{m}^{-1}, \\ &m_{3A} = 2.5\text{kg}, m_{3L} = 0.5\text{kg}, \Theta_{3mot} = 91.10^{-6}\text{kgm}^2 \end{aligned}$$

4.5 Servo Motor and PD-Control for three link SCARA (Task b)

The electrical relationship of the measure of a robot servo motor is given by a first order differential equation.

$$\dot{I}_i = \frac{(U_{ai} - k_{Ti}u_i\dot{q}_i - R_{ai}I_{ai})}{L_{ai}}, i = 1,2,3 \tag{47}$$

$$I_{ai} = [-I_i^{max} \le I_i \le I_i^{max}], i = 1,2,3 \tag{48}$$

where $U_{ai}(t)$ is the applied armature voltage. The resulting armature current I_i is limited to maximum value I_i^{max} that can be calculated from the maximum permitted torque T_i^{max}

$$I_i^{max} = T_i^{max}\left(\frac{\sqrt{3}}{2}k_{Ti}\right)^{-1}, \quad i = 1,2,3 \tag{49}$$

The joint torque (force) T_i of a motor is proportional to the armature current I_{ai} and given by

$$T_i = u_i \frac{\sqrt{3}}{2} k_{Ti} I_{ai}, \qquad i = 1,2,3 \tag{50}$$

Numerical values for the motor constant k_{Ti}, the gear ratio u_i, the resistance R_i and the inductance L_i for each motor are given below. Note that u_3 includes the transformation from the rotational to the linear motion and is not dimensionless.

$k_{T1} = 0.4Vs$,	$k_{T2} = 0.25Vs$,	$k_{t3} = 0.4Vs$,
$R_{a1} = 3.9Ohm$,	$R_{a2} = 50$ Ohm,	$R_{a3} = 40$ Ohm,
$L_{a1} = 7.3mH$,	$L_{a2} = 25mH$,	$L_{a3} = 25mH$,
$u_1 = 130$,	$u_2 = 100$,	$u_3 = 1047m^{-1}$
$T_1^{max} = 2.3Nm$,	$T_2^{max} = 0.6Nm$,	$T_3^{max} = 0.6Nm$

In order to control the point –to-point motion of the robot a rather primitive single –axis PD-control is employed. For a given target joint position vector $\hat{\vec{q}}$ position errors $(\hat{q}_i - q_i)$ can be calculated. From the position errors and the joint velocities $\dot{q}_i$ the control voltage U_{ai} is determined by

$$\begin{aligned} U_i &= P_i(\hat{q}_i - q_i) - D_i\dot{q}_i, i = 1,2,3 \\ U_{ai} &= [-U_i^{max} \le U_i \le U_i^{max}], i = 1,2,3. \end{aligned} \tag{51}$$

Proportional gains P_i and derivative gains D_i are given for each controller. In regular operation mode the armature voltage shall be limited by U_{ireg}^{max}. However, in an emergency situation $U_{i\max}^{max}$ may be.

$$P_1 = 1000V, \quad P_2 = 1000V, \quad P_3 = 5000V,$$
$$D_1 = 10Vs, \quad D_2 = 25Vs, \quad D_3 = 10Vs,$$
$$U_{1reg}^{max} = 100V, \quad U_{2reg}^{max} = 75V, \quad U_{3reg}^{max} = 90V,$$
$$U_{1\max}^{max} = 230V, \quad U_{2\max}^{max} = 230V, \quad U_{3\max}^{max} = 230V$$

Task a) Modelling method. There are several ways to formulate and implement the model, depending on the simulator's features, e.g.

i) "manual" symbolic manipulations for setting up explicit model equations, implementation of the explicit model description,

ii) derivation of explicit equation using software for symbolic calculations, implementation of the explicit model description,

iii) using special features of the simulator for deriving and simulating the equations (mechatronic modules, etc.),

iv) Implementation of the implicit equations, using algebraic loop breaking features of the simulator.

v) Implementation of the implicit equations, using **Matlab 6.5/simulink**.

The simulator's features for this type of models should be sketched briefly by giving (parts of) the model description of at least one (but preferably of some) of the above given methods. In case of alternative modeling approaches the effectiveness should be compared, taking into account preparation time, necessary knowledge for certain alternatives, etc.

Task b) Simulation of a point-to-point motion, controlled by a single axis PD control shall be performed. No obstacle is present for this task.

Initial values at $t = 0$:

$$q_1 = q_2 = q_3 = 0, \quad \dot{q}_1 = \dot{q}_2 = \dot{q}_3 = 0$$

Target (terminal) values at $\hat{t}$:

$q_1 = 2, q_2 = 1, q_3 = 0.3,$

As results graphs of the joint positions is plotted.

4.6 Simulink model of three link SCARA manipulator.

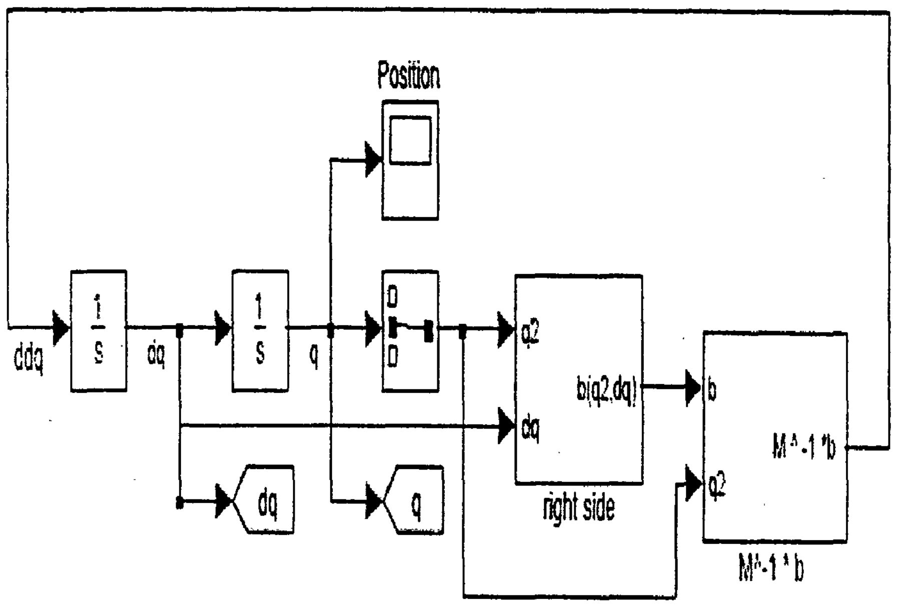


Fig 28:The simulink model of three link SCARA Manipulator.

The dynamic equation of the robot is given by

$$m(q)\ddot{q} + h(q, \dot{q})\dot{q} + g(q) = \tau$$

(OR)

$$m(q)\ddot{\bar{q}} = \bar{b} \qquad (50)$$

The overall dynamic model of a three-link SCARA robot without controller and actuator dynamics is designed by using simulink tool of MATLAB 6. as follows,

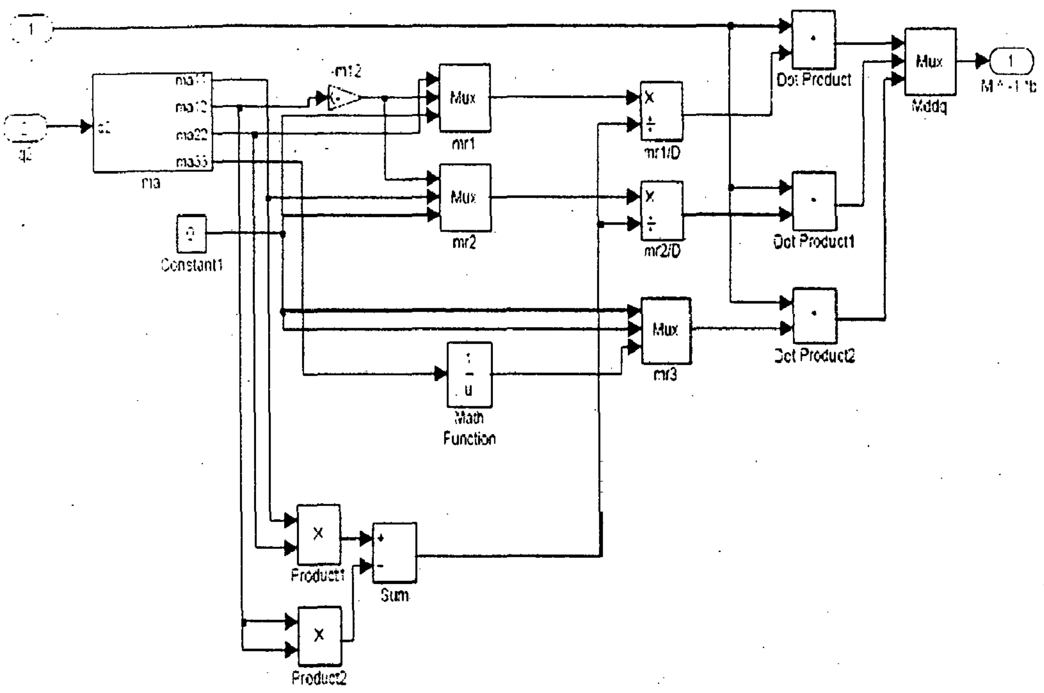


Fig 29: Simulink representation of Mass matrix of the three link SCARA

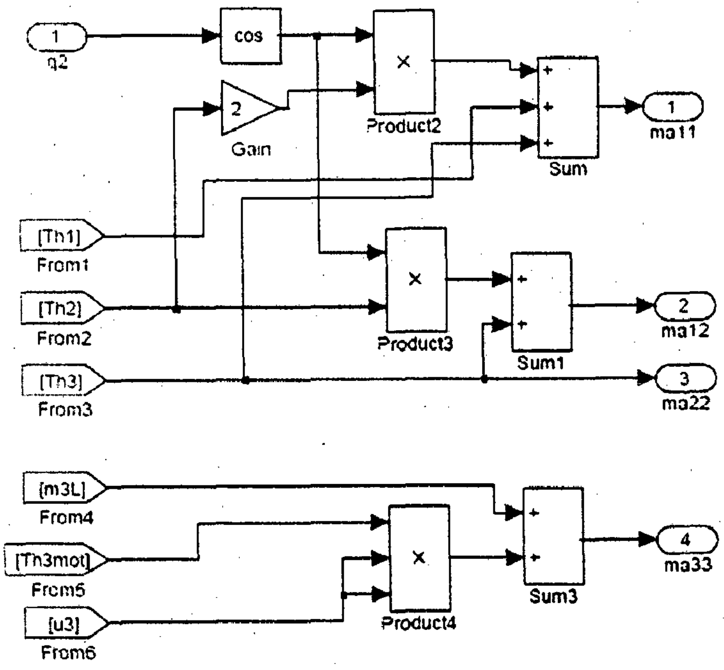
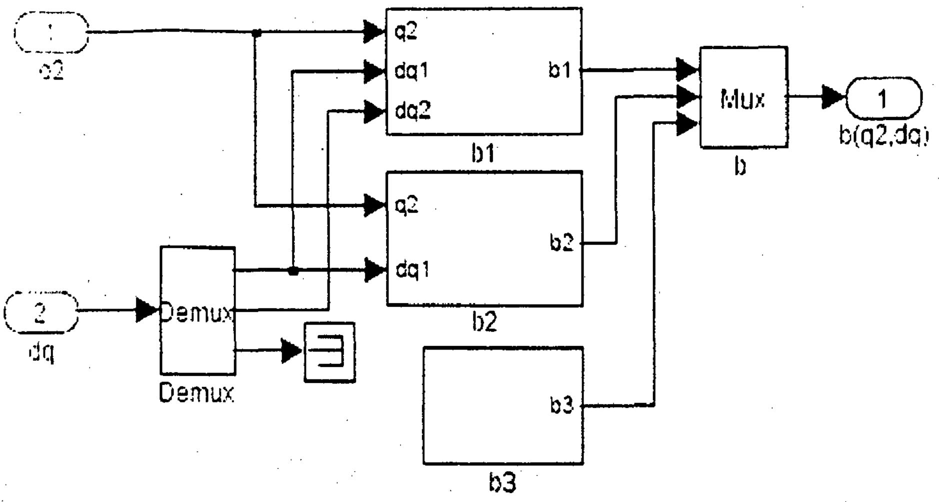
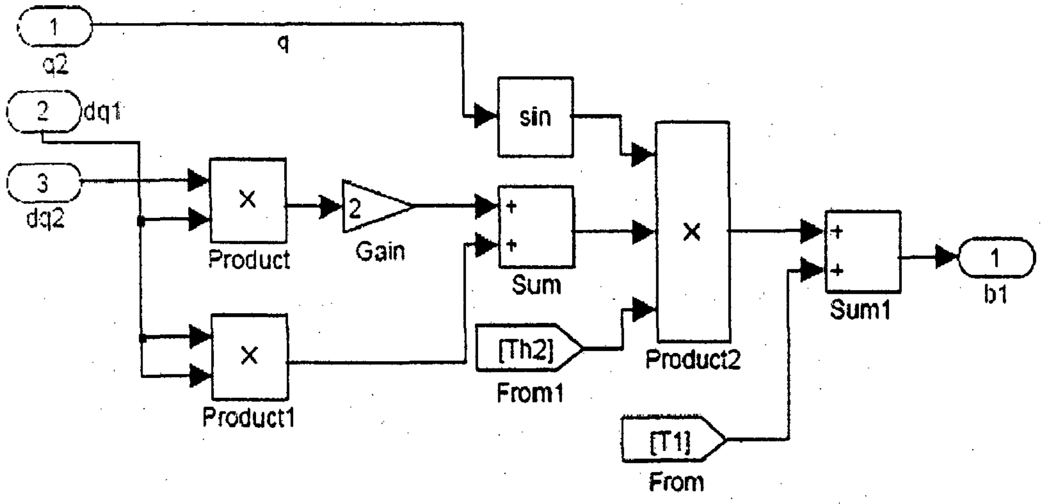


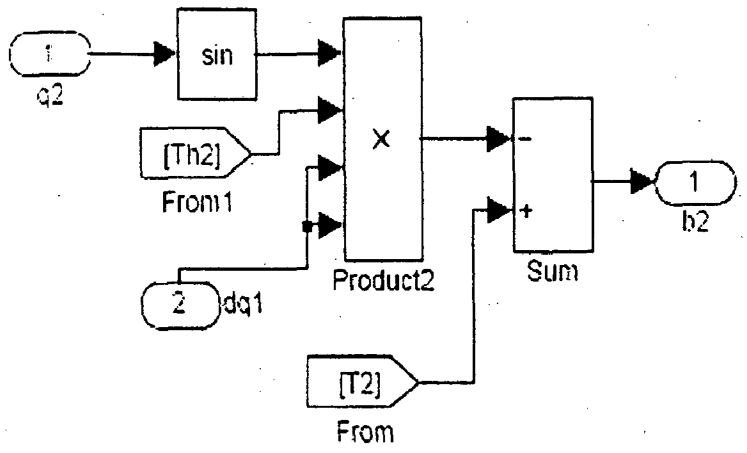
Fig 30: Simulink representation of block ma of Mass matrix.

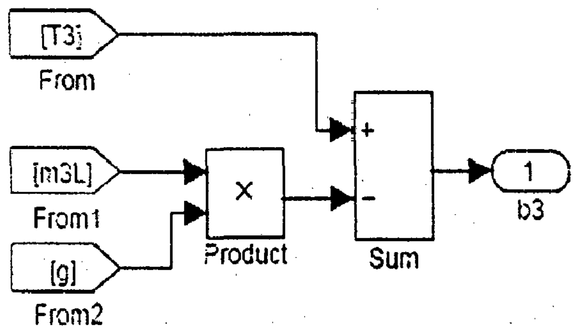


$$\vec{b} = (b_1, b_2, b_3)^T,$$
$$b_1 = T_1 + \Theta_2(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin(q_2),$$
$$b_2 = T_2 - \Theta_2\,\dot{q}_1^2\sin(q_2),\ b_3 = T_3 - m_{3L}\,g$$

Fig 31: Simulink representation of b (right side) block of Dynamic equations.







$$\vec{b} = (b_1, b_2, b_3)^T,$$
$$b_1 = T_1 + \Theta_2(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2)\sin(q_2),$$
$$b_2 = T_2 - \Theta_2\,\dot{q}_1^2\sin(q_2),\ b_3 = T_3 - m_{3L}\,g$$

Fig 32: Simulink models of b vector equations.

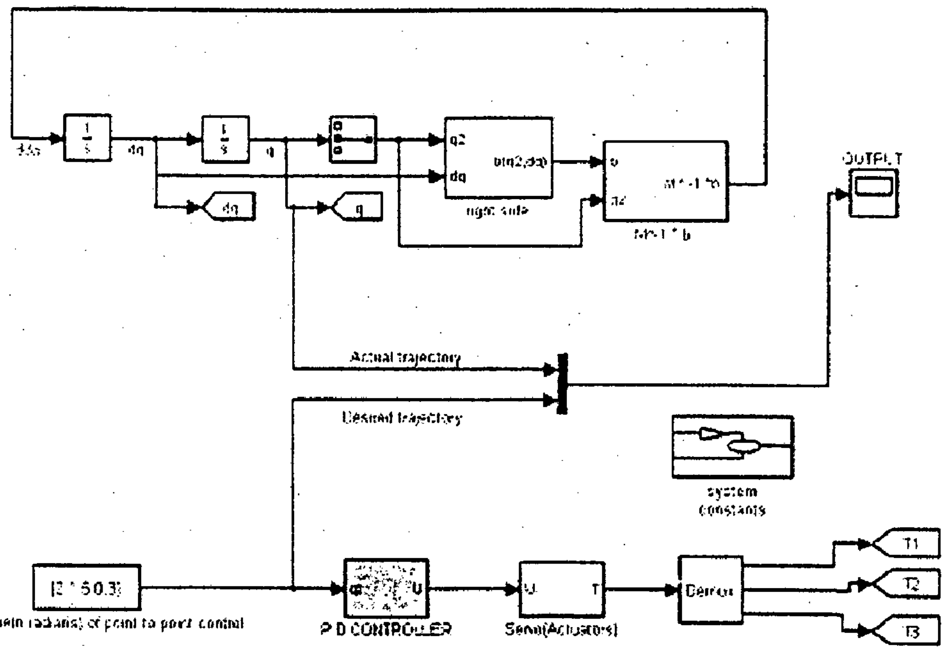
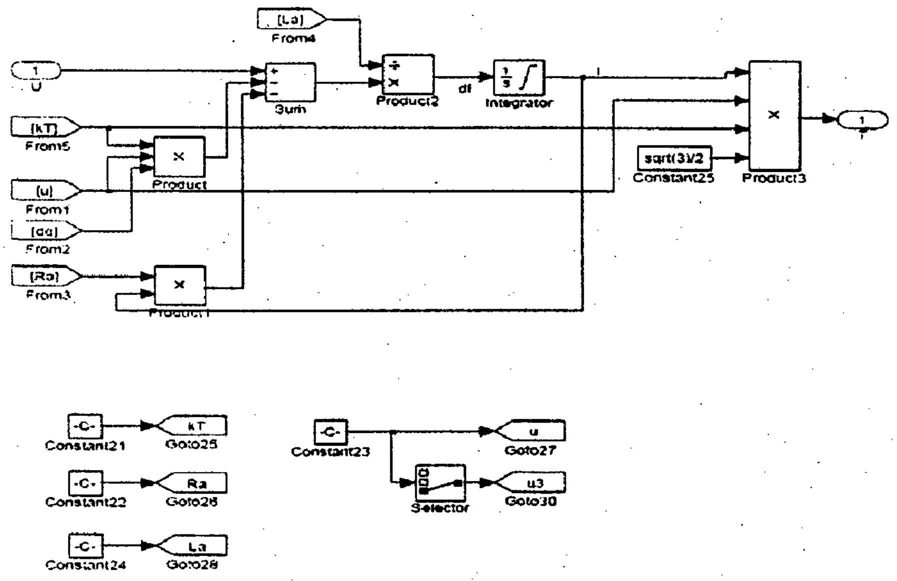


Fig 33: Simulink Model of Three Link SCARA Manipulator with controller and actuator.

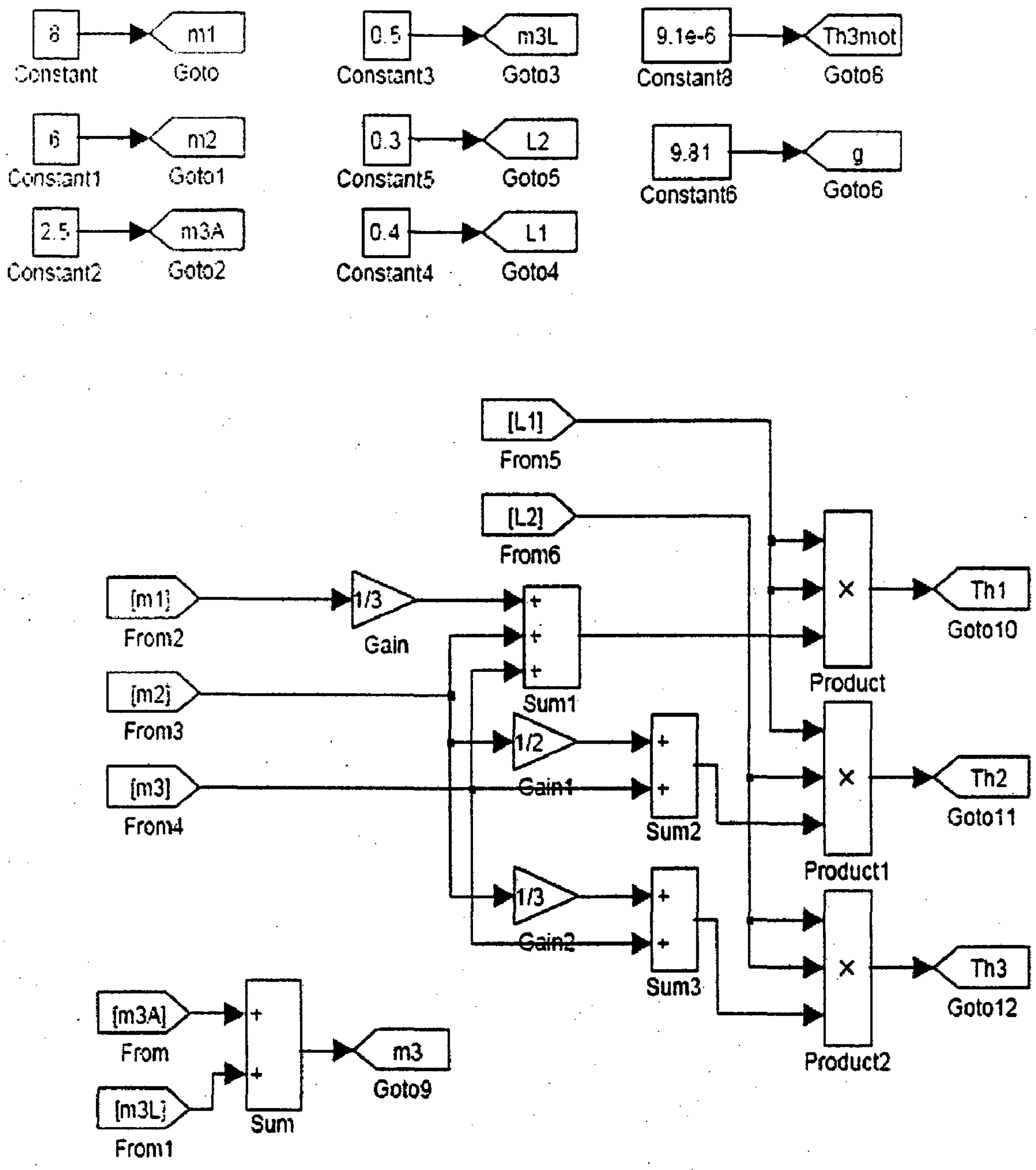


$$\dot{I}_i = \frac{(U_{ai} - k_{Ti} u_i \dot{q}_i - R_{ai} I_{ai})}{L_{ai}}, \quad i = 1,2,3$$

$$I_{ai} = [-I_i^{\max} \le I_i \le I_i^{\max}], \quad i = 1,2,3$$

$$I_i^{\max} = T_i^{\max} (\frac{\sqrt{3}}{2} k_{Ti})^{-1}, \quad i = 1,2,3$$

Fig 34: Simulink model of Servo motor dynamics



$$\Theta_1 = (\frac{m_1}{3} + m_2 + m_3) L_1^2, \quad \Theta_2 = (\frac{m_2}{2} + m_3) L_1 L_2,$$

$$\Theta_3 = (\frac{m_2}{3} + m_3) L_2^2, \quad m_3 = m_{3A} + m_{3L}$$

Fig 35: Simulink model of Moment of inertia of three link SCARA Manipulator and Realistic data's

CHAPTER 5

DYNAMIC CONTROL OF SINGLE LINK MANIPULATOR:

5.1 Point to point control

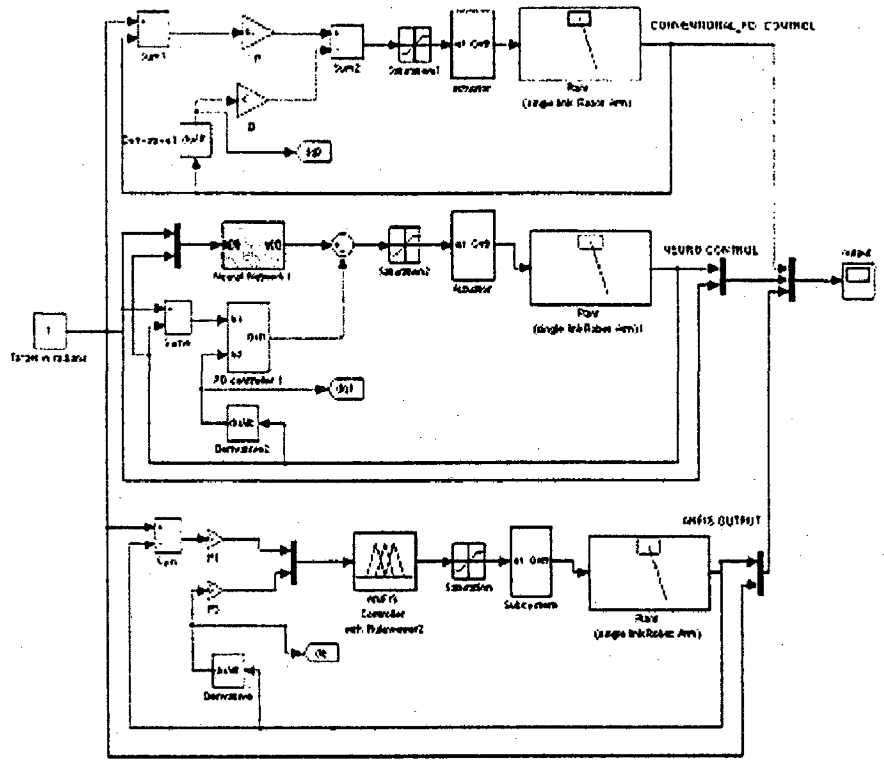


Fig 36: shows the simulation of single link manipulator using conventional PD, ANN and ANFIS controller.

The figure 34 shows the simulation arrangement of single link manipulator with three type of controller namely, Conventional PD, Neuro control and ANFIS control. This model is developed by using simulink, neural network toolbox and ANFIS (fuzzy logic Toolbox). The target value one radian is chosen to be the desired point to reach by the manipulator from initial point zero. All the three output of the controllers are connected to multiplexer to have a visual display of three-controller output.

5.2 Response of single link manipulator

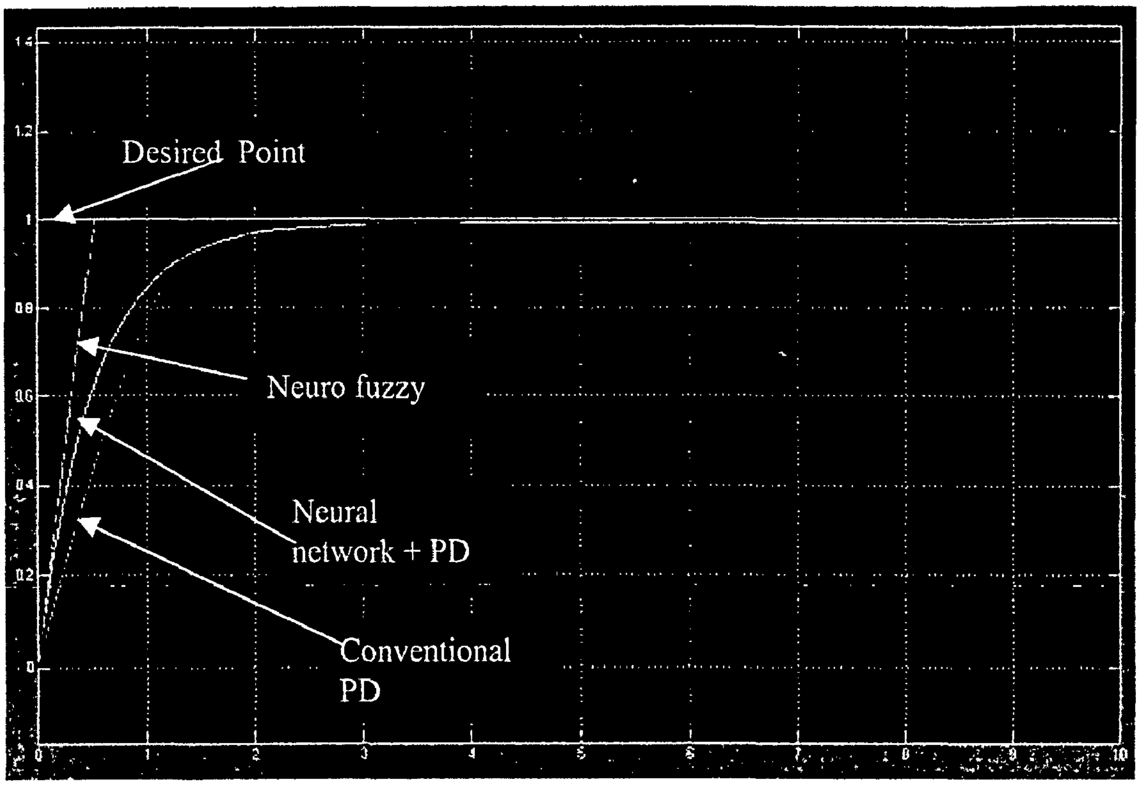


Fig: 37 shows the response of single link manipulator with Conventional PD, ANN and ANFIS controller.

Simulation studies were carried out to evaluate the performance of the NN controller. They were carried out in two parts. The first part is to evaluate the trajectory tracking capability of the controller for single link manipulator. The second part is to compare the performance of the controller with a conventional Proportional Derivative (PD) controller. Figure 33 shows the simulation result of single-link manipulator with neurocontroller for a sine wave joint angle trajectory. The solid line represents the desired input and the dotted line represents actual output. From Figs.33 and 35 it can be seen that the difference between actual joint angle trajectory and desired trajectory is almost zero. As expected, the network was found to be fully trained. Significant improvement can be observed as the broken line closely follows the solid line from this it is concluded that the errors in the links are almost minimum

Continuous point control

5.3 Simulation diagram of single link manipulator

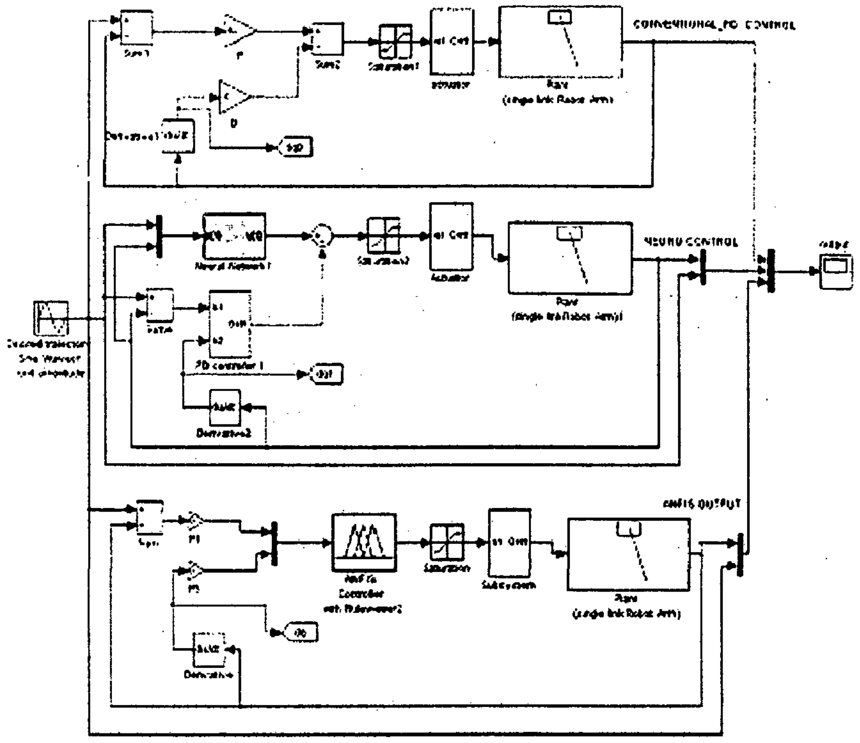


Fig 38: Simulation of single link manipulator using Conventional PD, ANN and ANFIS Controller

. The figure 38 shows the simulation arrangement of single link manipulator with three type of controller namely, Conventional PD, Neuro control and ANFIS control. This model is developed by using simulink, neural network toolbox and ANFIS (fuzzy logic Toolbox). A PD controller does not use the dynamic model of the controlled robot manipulator and is there for easy to implement, simple to compute, and also robust against dynamic uncertainties of the robot manipulator the disadvantage, on the other hand , is its relatively poor tracking performance in comparison with controller using accurate robot dynamic models such as computed torque controller. The sine wave of unit amplitude is chosen as desired trajectory of the manipulator from initial point zero. All the three output of the controllers are connected to multiplexer to have a visual display of three-controller output.

Hence, from the result, the Neuro fuzzy output is superior when compare to other two controllers.

5.4 Response of single link manipulator

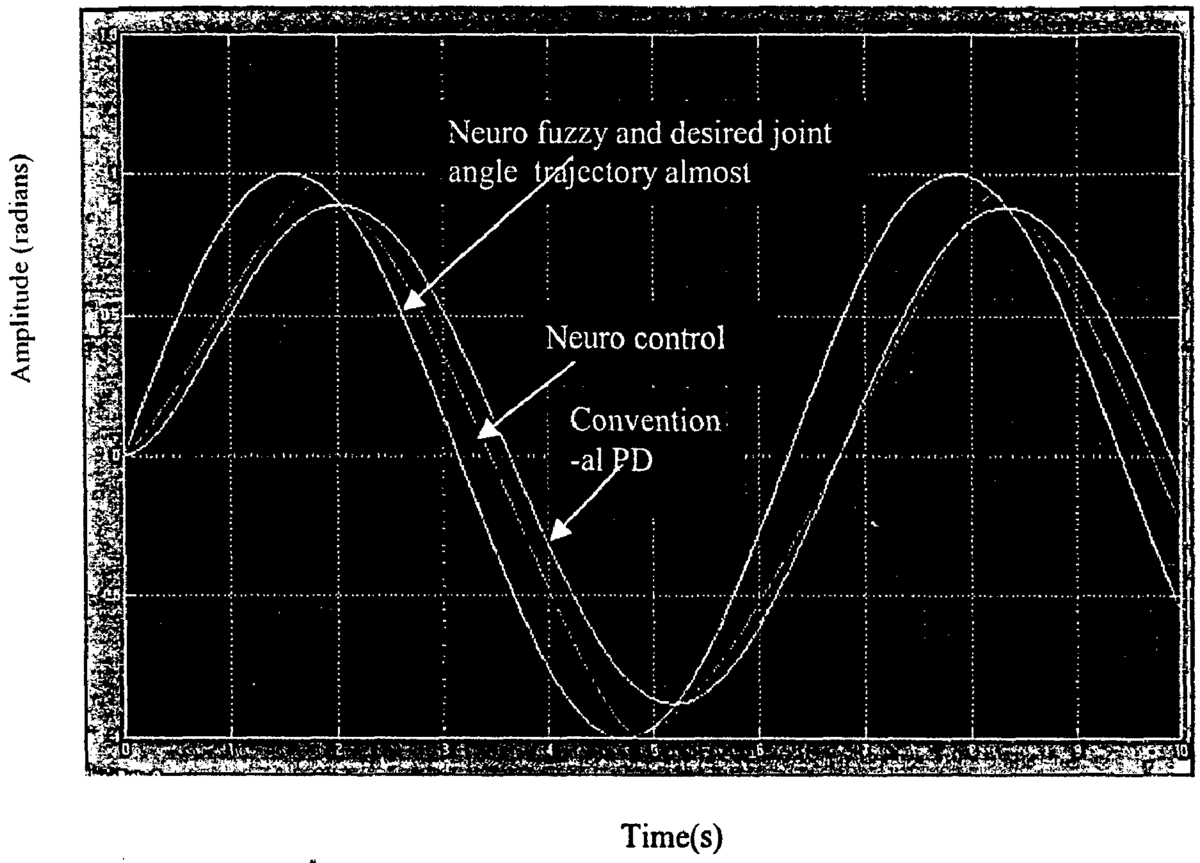


Fig 39: Response of joint angle trajectory (sine) of single link manipulator with conventional PD, ANN and ANFIS controller.

Hence, the response of ANFIS controller is almost very near to the desired joint angle trajectory. This is due to the genesis of ANFIS and also the Adaptive capabilities and learning ability of it. From the figure, it infers that the neuro-fuzzy controller response is faster in comparison with the neuro controller; moreover, the desired trajectory is almost coincident with the actual trajectory. Whereas the PD control could not able to follow very exactly because of no proper method to choose the optimal value of proportional and derivative constant.

CHAPTER-6

DYNAMIC CONTROL OF THREE-LINK SCARA MANIPULATOR:

Point to point control

In point to point control method, the end effector of the robot moves to a sequence of discrete point in the workspace. The path between the points is not explicitly controlled by the user. Point to point motion is useful for operations, which are discrete in nature .for example, spot welding, pick-and-place, loading, and unloading, is an application for which point-to-point motion of the tool is all that is required [18]. The Figure shown below is the simulation model of the Three link SCARA manipulator with PD controller, this controller can perform very well under normal condition, But in case of parameter variation and other uncertainties it can not do well its operation and hence it detroites the over all system performance

6.1 Simulation model of three-link SCARA manipulator with conventional controller

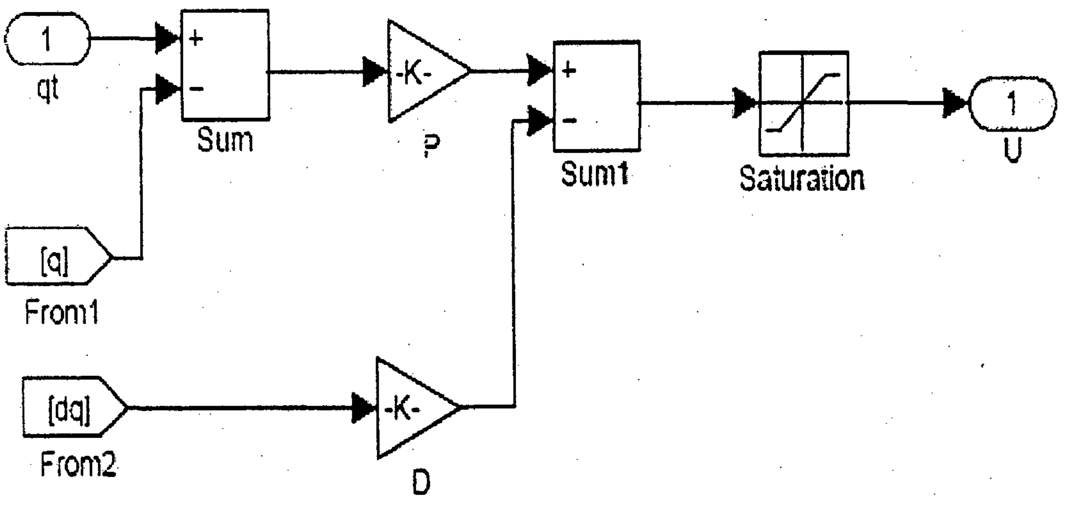


Fig 40: The internal block of PD controller of a three-link SCARA manipulator.

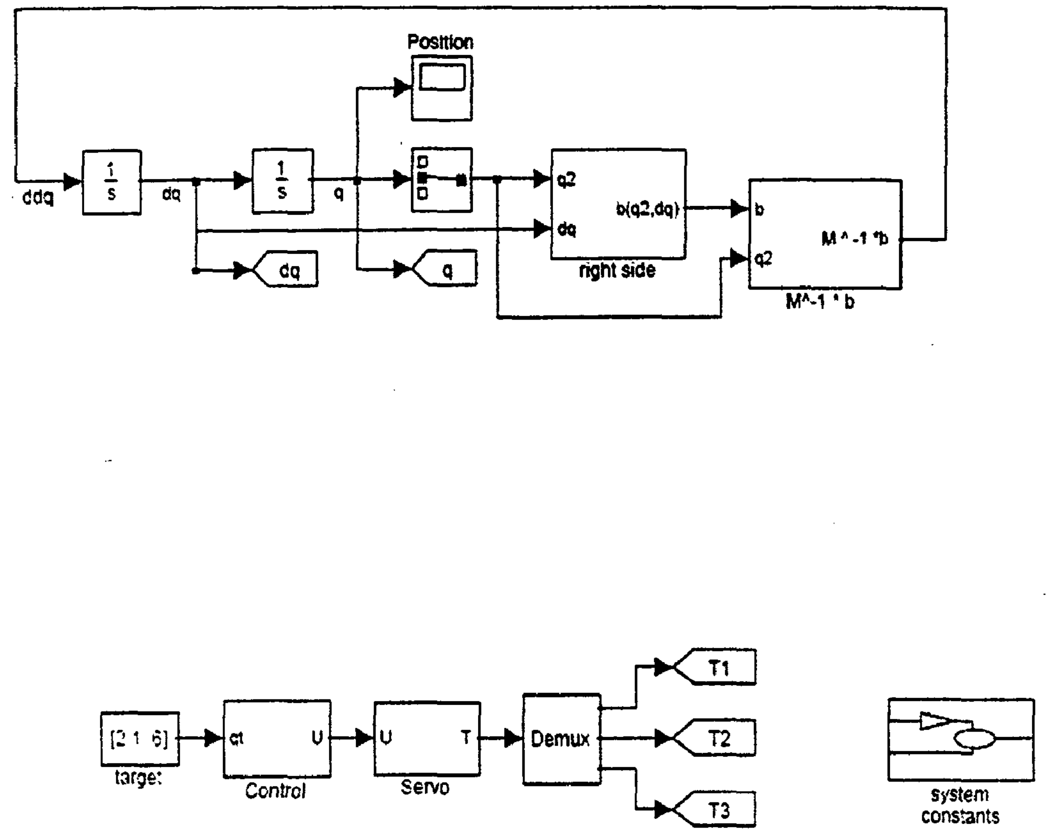


Fig 41: Simulation model of a three-link SCARA Manipulator with conventional PD Controller.

The Figure 41 shows the simulation model of a three-link SCARA manipulator for point to point control .This model is developed by using the dynamic equations and Simulink package of MATLAB6.The initial point is assumed to be zero and the final joint angles are 2, 1, 0.6, .Control section is a PD controller internally it has summer and other terminals from the output of the dynamic model as shown in fig 40.The PD controller is tuned for achieving the desired output, but due to lack of intelligence like Neuro-Fuzzy and ANN is could not able to cope up with the structured uncertainties and parameter variations. The output of the PD control goes to the servo motor control and actuates the joints of the SCARA manipulator to follow the desired point to point, from the initial value.

6.2 Response of Three link SCARA Manipulator

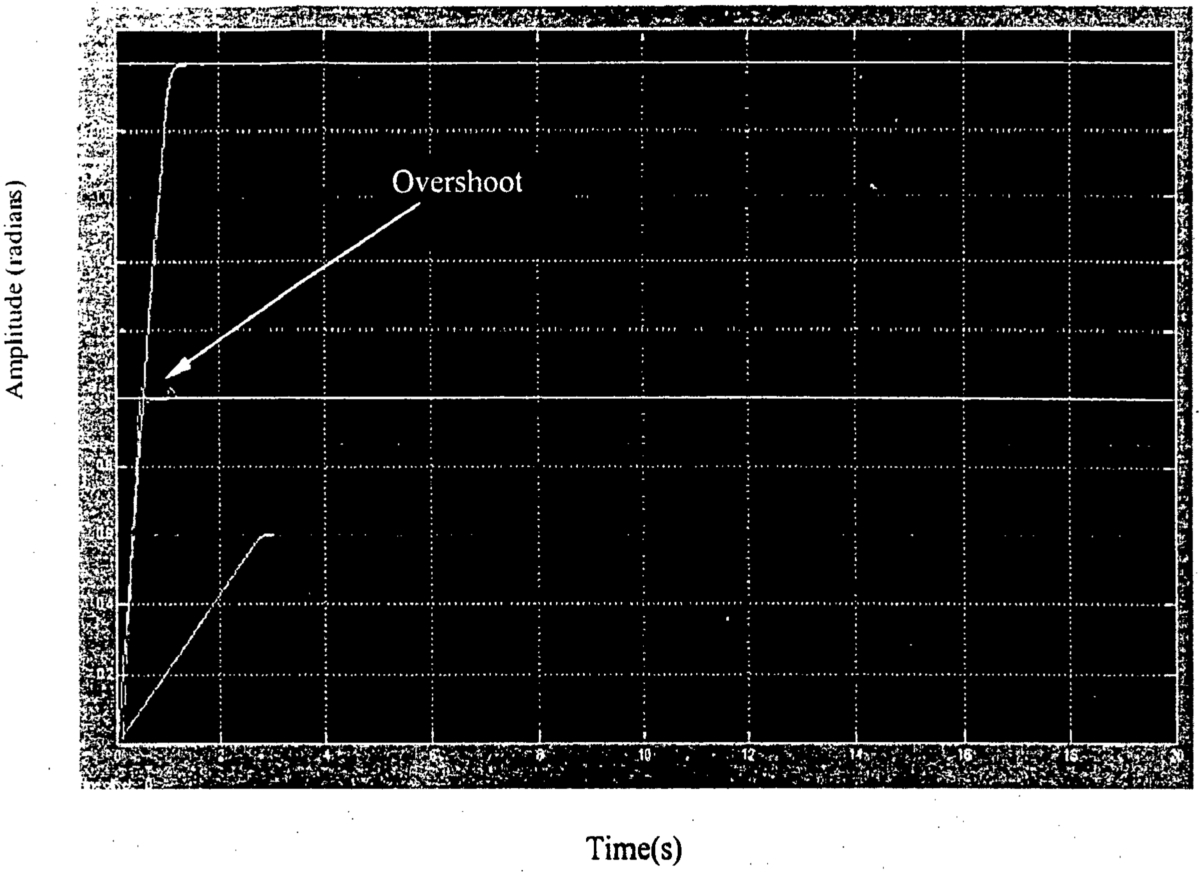


Fig 42: Simulation results of three link SCARA Manipulator with conventional PD Controller. With initial point (0,0,0) to target point (2, 1, 0.6).

The Figure shows the response of the point-to-point control of SCARA manipulator the link two has some initial transient due to sudden change in inertia and momentary actuating signals to the joints this effect can be overcome by using neuro control or ANFIS control. The sine trajectory is chosen for the joint 1 and joint 2 and a step of 0.6 is chosen for prismatic joint a desired input, to evaluate the performance under dynamic condition.

6.3 Training results of ANN

The neural network tool box provide a set of blocks we can build in simulink or which can be used by the function **gensim** to generate the simulink version of any network we have created in Matlab. Here the single layer neural network is trained (back propagation algorithm) for 10000 epochs, with 16 hidden neurons and learning rate of 0.1 .the performance indices (mean square error) after training is found to be 0.000111.

This index is very good for this application. Hence, the generated simulink block from this program is used in the simulation model for evaluating the performance of the robot system.

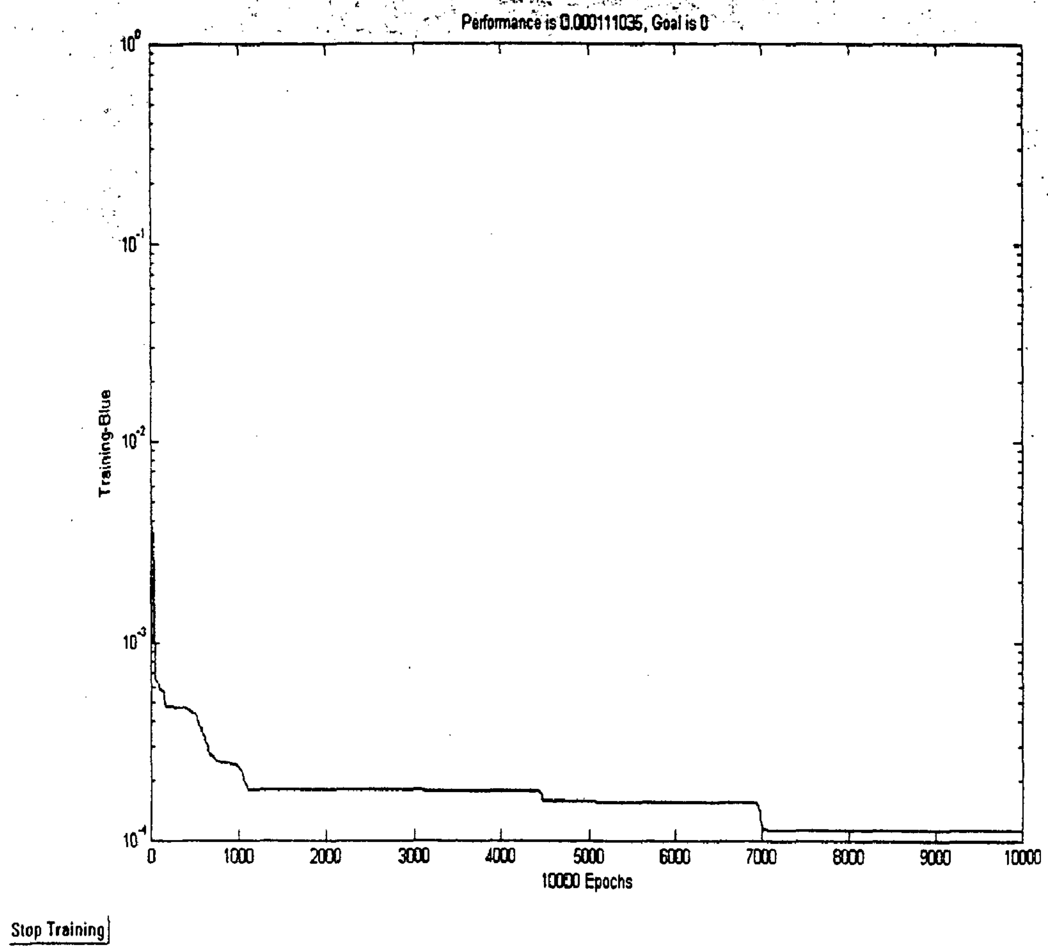


Fig 43: The training results of neural network.

6.4 Simulation of Three link SCARA manipulator with Neuro controller

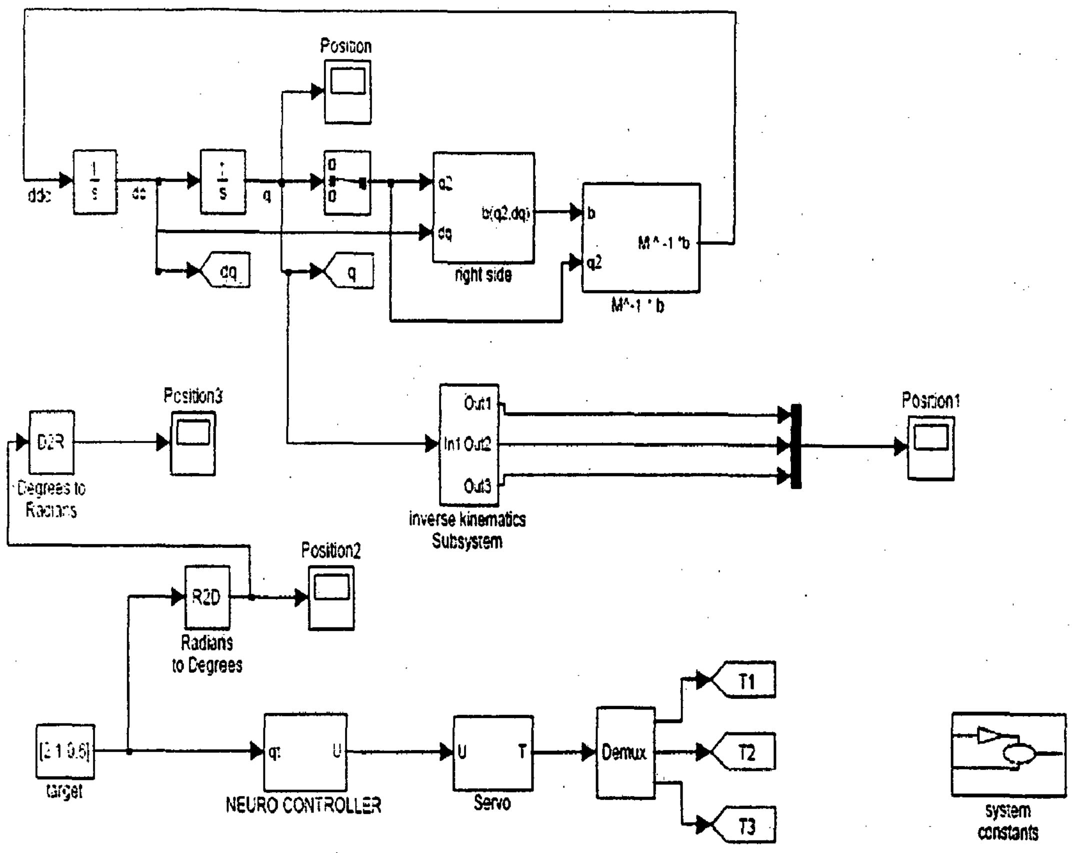


Fig 44: The simulink model of neuro controller for a three link SCARA Manipulator

The figure shows the simulation model of three-link SCARA manipulator using neural network. The trained ANN network is obtained in the form of simulink block is used as a neuro controller for each link of the manipulator .The output of the neuro network gives the control signal to the robot actuators and also neural network act as a nonlinear compensator during uncertainties .The neural network is trained for the given trajectories after attaining the desired performance index .the neural network will work as a assisting member with conventional PD controller during uncertainties and parameter variation. The tracking performance of this manipulator is evaluated by using its response .From the response it concludes that the neural network follows almost closely to the desired value. Figure 44 shows the response of three link SCARA manipulator with Neuro control from the graph it infers that the neural network has good tracking performance, but the problem is to have judicious data for training with properly chosen training parameters .Simulation result of three-link SCARA manipulator with neurocontroller for joint angles

6.5 Response of Three link SCARA manipulator with Neuro controller of Point to Point control

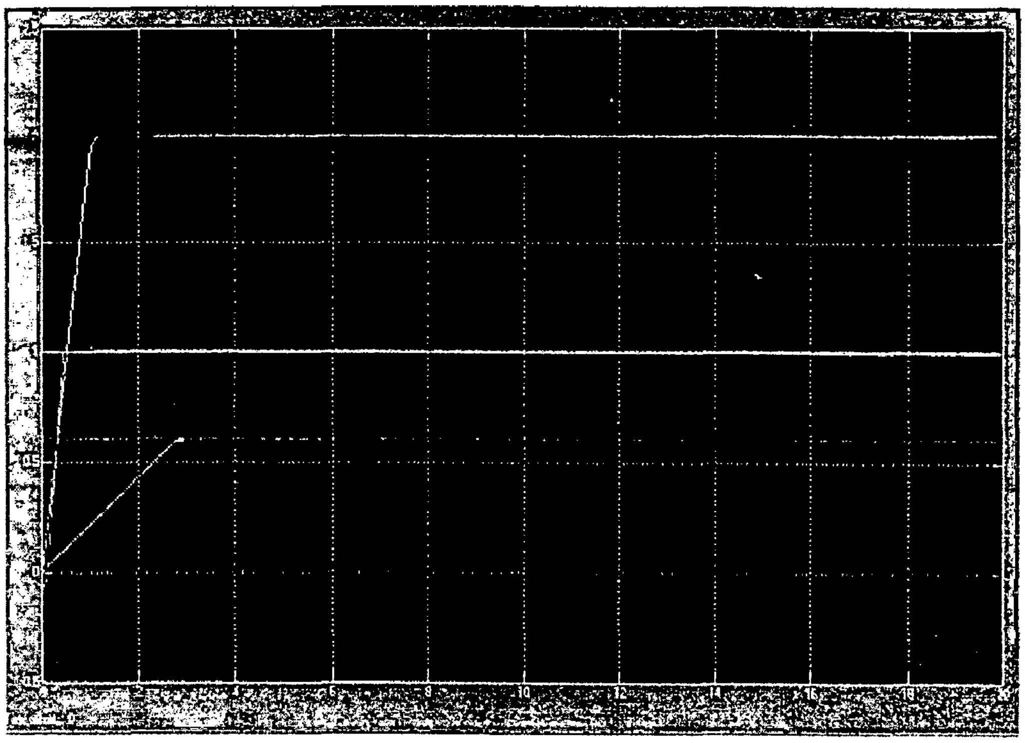


Fig 45: The response of point-to-point control of three link SCARA Manipulator with ANN.

The figure shows the response of the three link SCARA manipulator for point to point control with initial values for three joint are (0,0,0) and the target value is (2,1,0.6) radians the trained neural network try to use the recall phase(generalization and memorization ability) and it produce a desirable control signal .this signal actuates the servomotors of the joints .from the response it is very clear that neural network has good tracking ability ,but it requires a judicious data and training parameter to obtain the satisfactory . The Fuzzy logic can also be used to map complex nonlinear relations by a set of IF-THEN rules. The membership functions are designed by intuitive human reasoning. This poses three problems. One, for different control application a new set of membership functions have to developed and second, latent stability problem[21] and third, once these membership functions are developed and implemented there is no means of changing them. This means fuzzy logic lacks a learning function Therefore this work is again focused on the area of neural and fuzzy logic together to overcome the drawbacks of neural network as well as fuzzy logic.

6.6 Simulation of Three link SCARA manipulator with Neuro-Fuzzy controller.

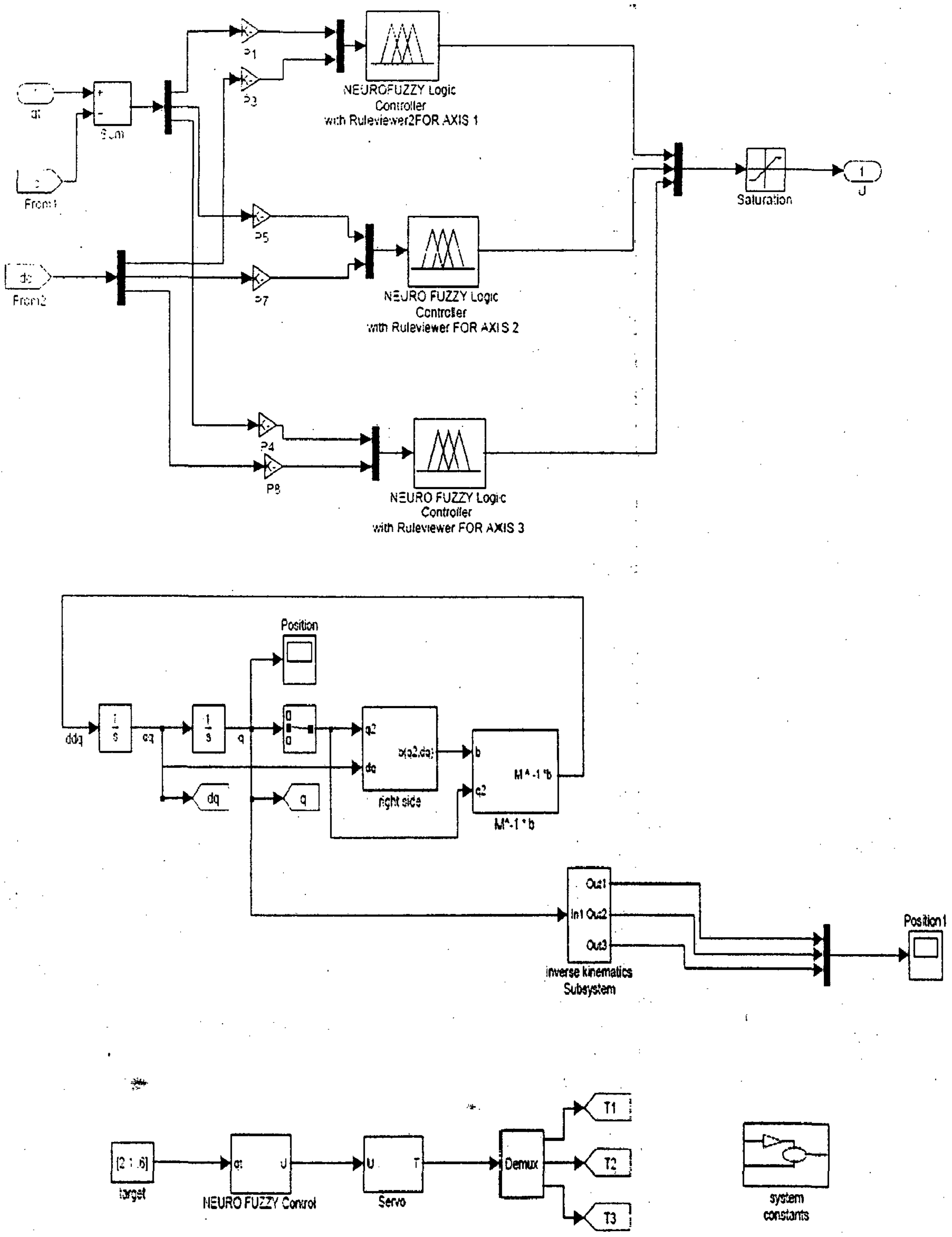

P1
P3
NEUROFUZZY Logic Controller with Ruleviewer2FOR AXIS 1
Sum
From1
From2
P5
P7
NEURO FUZZY Logic Controller with Ruleviewer FOR AXIS 2
P4
P8
NEURO FUZZY Logic Controller with Ruleviewer FOR AXIS 3
Saturation
1
u
Position
ddq
dq
q
right side
M^-1*b
b
q2
Out1
In1 Out2
Out3
inverse kinematics Subsystem
Position1
target
NEURO FUZZY Control
Servo
Demux
T1
T2
T3
system constants

Fig 46: Simulink Model of Neuro Fuzzy controller for three-link SCARA manipulator

6.7 Response of Three link SCARA manipulator with Neuro-Fuzzy controller of Point to Point control

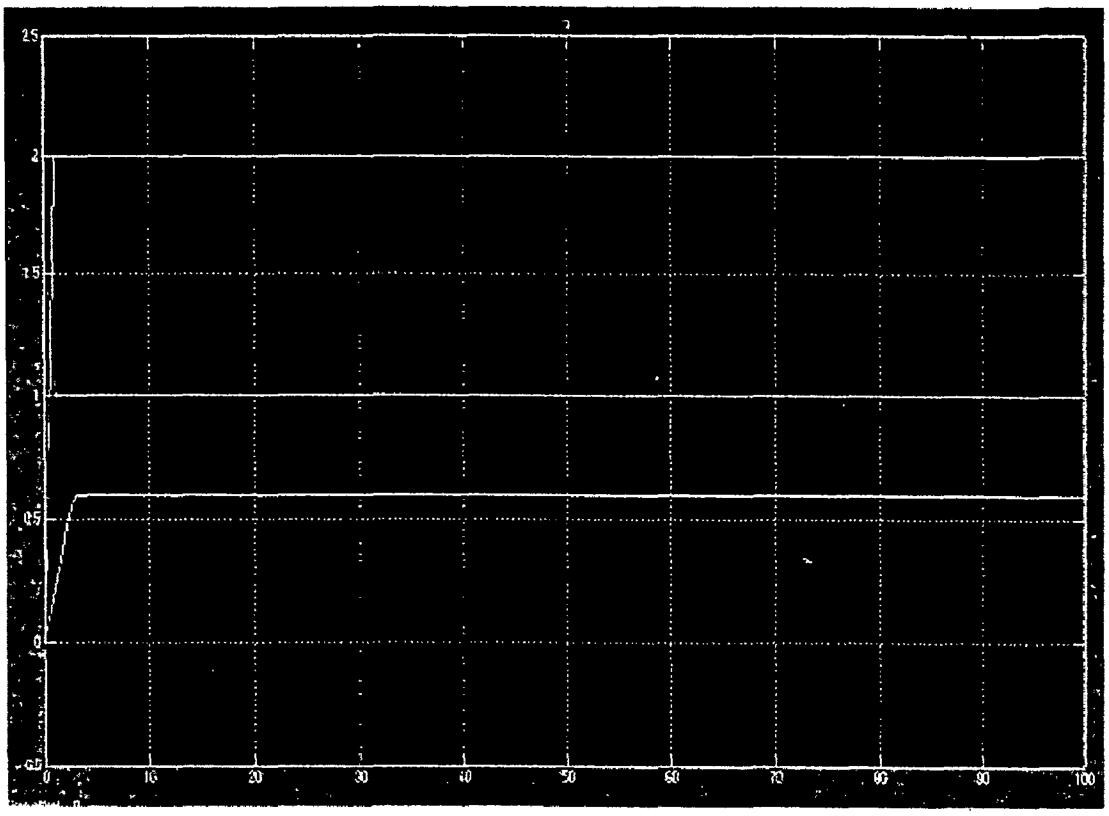


Fig 47: The response of point to point control (2, 1, and 0.6) of three link SCARA manipulator with Neuro Fuzzy controller.

From the graph it infers that the ANFIS control has good tracking performance, which combines the advantage of two methodologies namely, Fuzzy logic and neural network he solid line represents the desired input and the dotted line represents actual output. From Fig 47.It can be seen that the difference between actual joint angle trajectory and desired trajectory is almost coincident in the graph. As expected, the ANFIS network was found to be fully trained. Significant improvement can be observed as the broken line closely follows the solid line; this is due to the function approximation capabilities and adaptability features of ANFIS network. The ANFIS network provides the desirable torque signal to cope with uncertainties .Thus it makes the joint angles to follow the desired joint angles.

6.8 Simulation of Three link SCARA manipulator with conventional controller

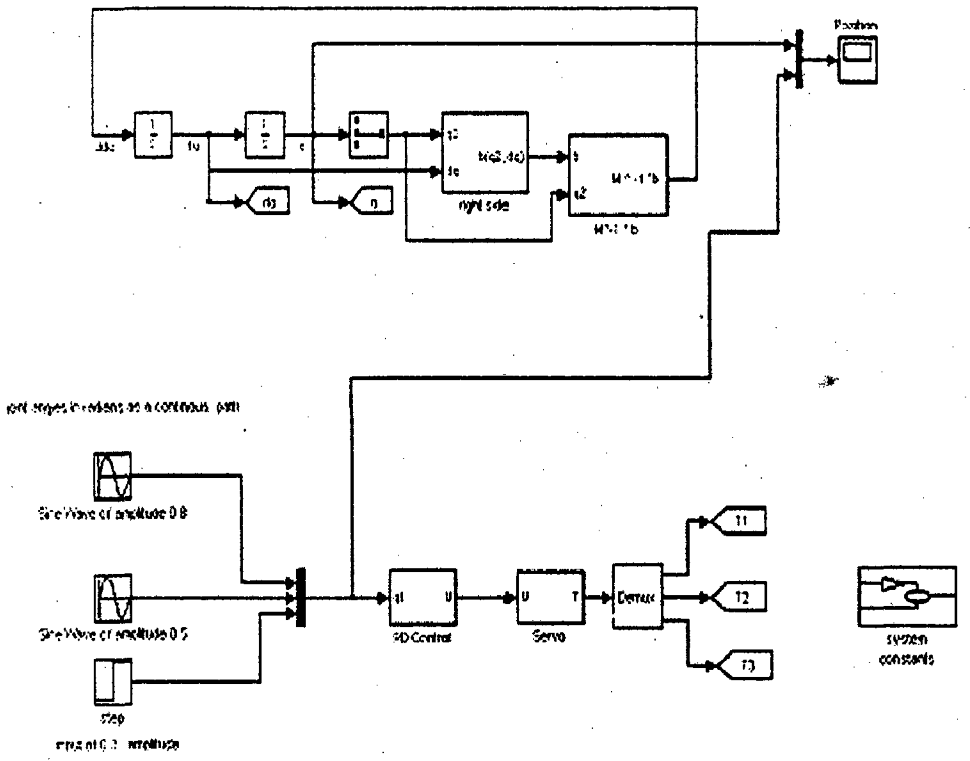


Fig 48: Simulink Model of PD controller for three link SCARA manipulator.

The figure 48 shows the simulation arrangement of three link SCARA manipulator with conventional PD control of values as follows

$$
\begin{aligned}
&P_1 = 1000V, && P_2 = 1000V, && P_3 = 5000V,\\
&D_1 = 10Vs, && D_2 = 25Vs, && D_3 = 10Vs,\\
&U_{1reg}^{\max} = 100V, && U_{2reg}^{\max} = 75V, && U_{3reg}^{\max} = 90V,\\
&U_{1\max}^{\max} = 230V, && U_{2\max}^{\max} = 230V, && U_{3\max}^{\max} = 230V
\end{aligned}
$$

6.9 Response of Three link SCARA manipulator with conventional controller

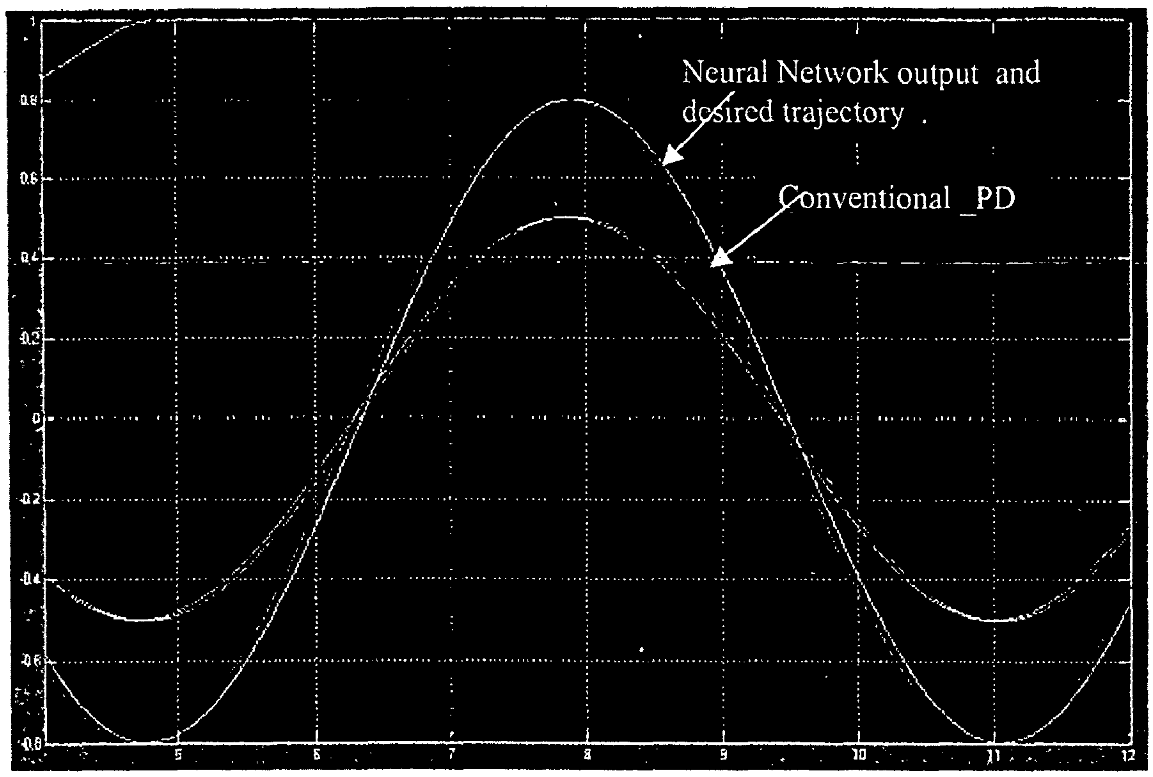


Fig 49:Shows the joint angle trajectory response of SCARA Manipulator with Conventional PD controller.

This figure 49 shows the response of continuous path control of three link SCARA manipulator with joint angle 0.8sin(t), 0.5sin(t) and the joint distance 0.3m for the third link. The PD control shows some deviation from the desired trajectory, this because of the selection of PD vaules does not depend on the dynamic model, hence due to approximation in the dynamic model lead to the deviation. Hence, this problem can be overcome by using the intelligent control.

6.10 Simulation of Three link SCARA manipulator with Neuro controller

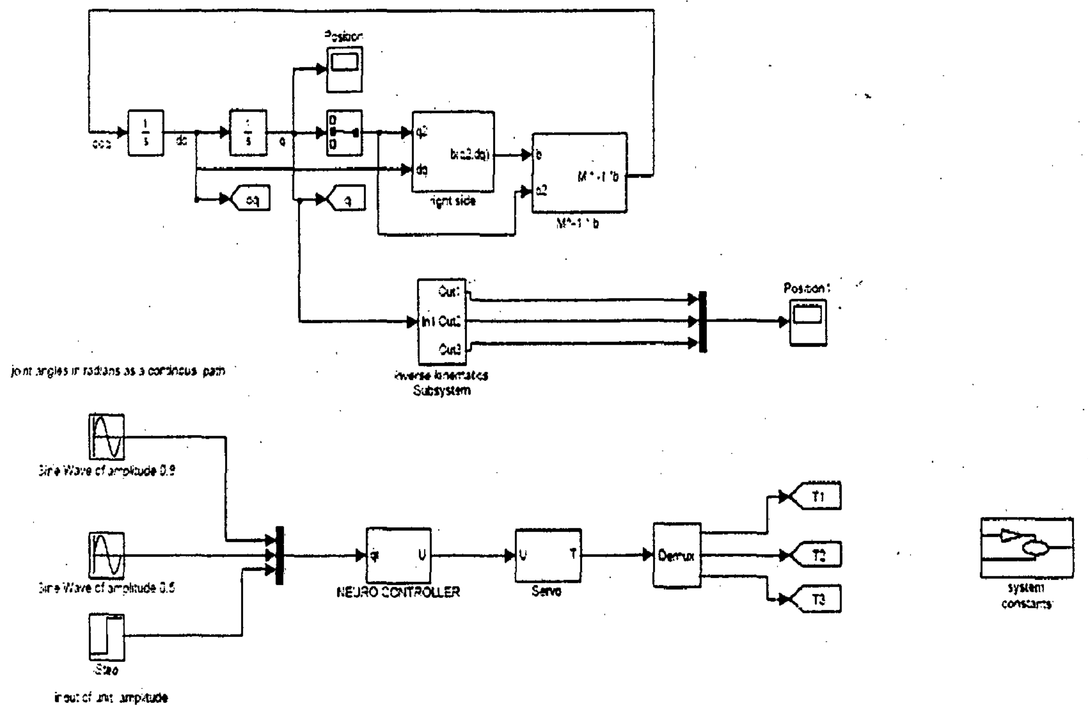


Fig 50: Shows the simulink model of Neuro controller for three link SCARA Manipulator.

The figure 50 shows the simulink model of SCARA manipulator with Neuro control .The neural network is trained for the given trajectories after attaining the desired performance index .the neural network will work as a assisting member with conventional PD controller during uncertainties and parameter variation. The tracking performance of this manipulator is evaluated by using its response .From the response it concludes that the neural network follows almost closely to the desired trajectory. Figure 51 shows the response of three link SCARA manipulator with Neuro control from the graph it infers that the neural network has good tracking performance, but the problem is to have judicious data for training with properly chosen training parameters .Simulation result of three-link SCARA manipulator with neurocontroller for a sine wave joint angle trajectory.

6.11 Response of Three link SCARA manipulator with Neuro controller

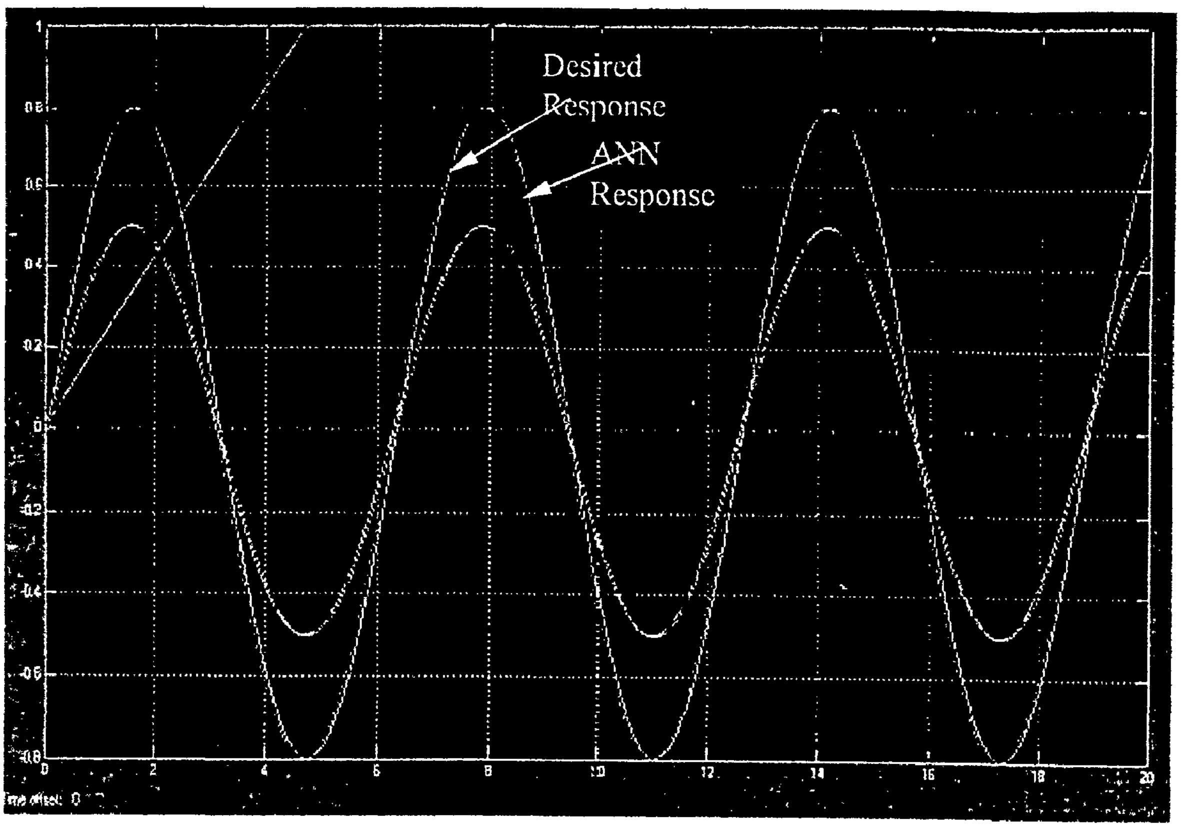
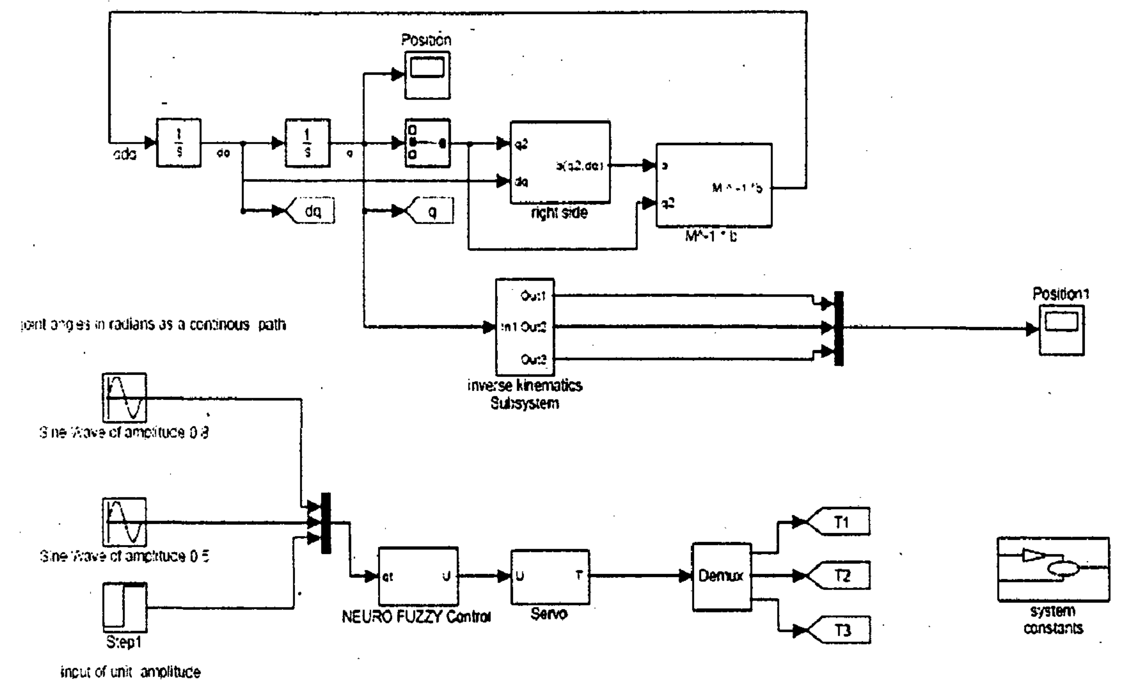


Fig 51: Response of joint angle trajectory of three-link SCARA manipulator with neuro controller

The solid line represents the desired input and the dotted line represents actual output. From Fig 51 it can be seen that the difference between actual joint angle trajectory and desired trajectory is almost merged in the graph. As expected, the network was found to be fully trained. Significant improvement can be observed as the broken line closely follows the solid line; this is due to the function approximation capabilities of RBF neural network in conjunction with the conventional PD controller. The RBF neural network provides the desirable torque signal to cope with uncertainties .Thus it makes the joint angles to follow the desired joint angle trajectory.

6.12 Simulation of Three link SCARA manipulator with Neuro-Fuzzy controller



Position
ddq
dq
q
dq
q
right side
M^-1 * b
Position1
joint angles in radians as a continous path
inverse kinematics Subsystem
Sine Wave of amplitude 0.8
Sine Wave of amplitude 0.5
Step1
input of unit amplitude
NEURO FUZZY Control
Servo
Demux
T1
T2
T3
system constants

Fig 52: Show the simulink model of SCARA Manipulator with Neuro fuzzy controller.

Figure 52 shows the feasibility of ANFIS control for a three-link SCARA manipulator has been proved and illustrated by simulation. The best parameters for the fuzzy controller were determined by using the ANFIS methodology and by using simulations of the SCARA manipulator dynamics. A simulation tool (i.e., Fuzzy logic toolbox (ANFIS)) was used to validate experimentally the tracking ability and the insensibility to plant parameter changes. The ANFIS controller presented very interesting tracking features and was able to respond to different dynamic conditions. In addition, the fuzzy control computation is very inexpensive, and this regulator could be used for the control of machine tools and robotics manipulators [11] without significantly increasing the cost of the drive. The only extra cost is for the optical encoder. Another advantage of this method over classical quantitative controllers is that it does not require a fixed sampling time. Therefore, the proposed design confirms the fact that fuzzy control is relevant to the fast control of non-linear processes such as SCARA manipulator control where quantitative methods are not always appropriate. Thus, the results obtained using the ANFIS controllers are encouraging when compared to conventional PD controller.

6.13 Response of Three link SCARA manipulator with Neuro-Fuzzy controller

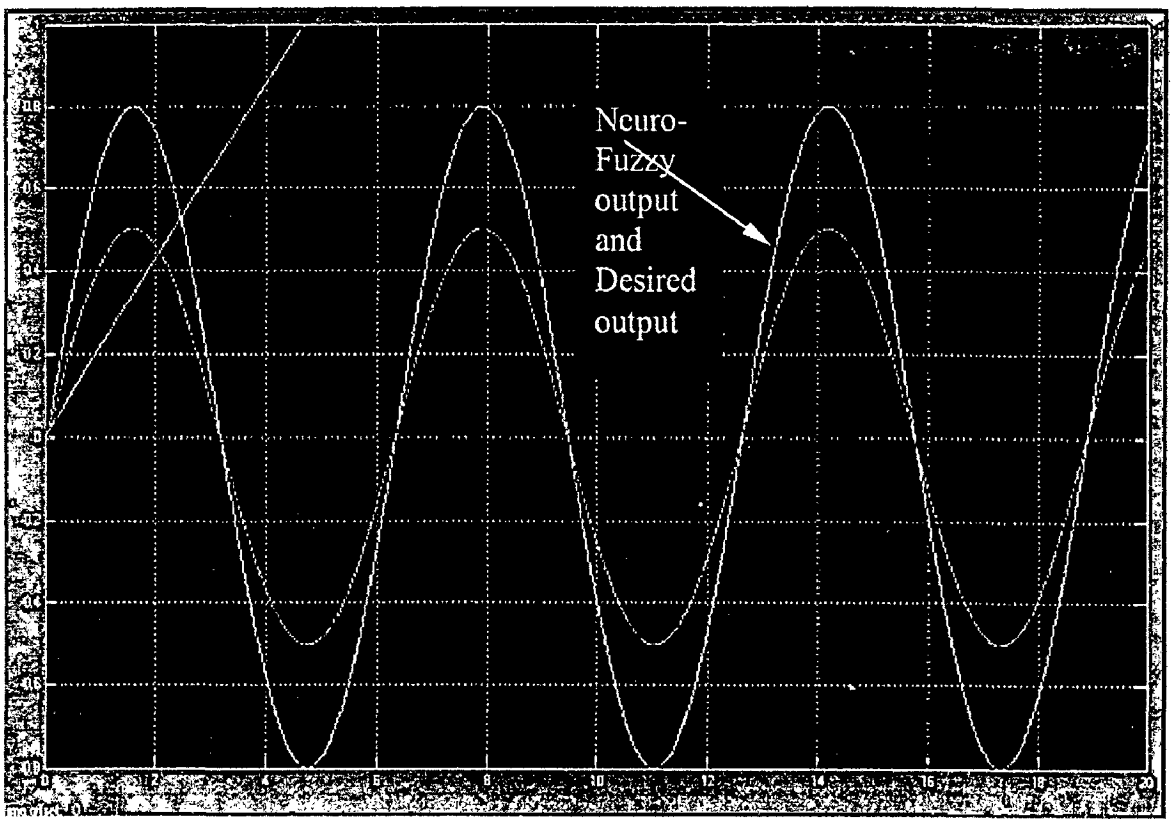


Fig 53: Show the joint angle trajectory response of the Three-Link SCARA manipulator with Neuro Fuzzy controller.

From the graph it infers that the ANFIS control has good tracking performance, which combines the advantage of two methodologies namely, Fuzzy logic and neural network he solid line represents the desired input and the dotted line represents actual output. From Fig 53 it can be seen that the difference between actual joint angle trajectory and desired trajectory is almost coincident in the graph. As expected, the ANFIS network was found to be fully trained. Significant improvement can be observed as the broken line closely follows the solid line; this is due to the function approximation capabilities and adaptability features of ANFIS network. The ANFIS network provides the desirable torque signal to cope with uncertainties .Thus it makes the joint angles to follow the desired joint angle trajectory.

CHAPTER-7

THE VISUAL DISPLAY OF THREE-LINK SCARA MANIPULATOR

7.1 Animation

Animation is the art of moving an object on the computer screen. Twining is the art of changing the shape of an image. We might have seen a lot of animation in cartoon films and video games, Twining is very common in advertisements. In this chapter we will discuss, in brief, how animation and twining are possible in computer graphics.

Animation is an optical illusion. We can not give continuous motion to an image. But we can flash a number of images quite rapidly in a certain manner so that the viewer will get the impression of a moving image. Suppose an image is placed a some position of the screen, shown for a fraction of a second, erased rapidly and redrawn at a new location slightly shifted from its original location. Human brain require time to register the change in position of the image .If the process is completed fast enough, because of persistence of vision, the human brain will register the shift in position of the image as motion .This is the principle of all animations.

7.2 The steps for producing animation is given below

1. Generate the image at some location X
2. Wait for some time (a few mill seconds)
3. Erase the image from the current location.
4. Redraw the image at a new location slightly from the original location
5. Repeat step 2 through 4 as long as required.

For animation we will use function **delay ( )** defined in **header** file dos.h .the function may be called as delay (k).

Using the inverse kinematics of SCARA manipulator , and perspective projection of the frame of the SCARA manipulator in three dimensional co-ordinate enables to develop the visual display of SCARA Manipulator in C++.[18]

CHAPTER-8

A PRATICAL APPROACH TO IMPLEMENT A NEURO FUZZY TECHNIQUE.

8.1 Introduction

Fuzzy logic can be used to map complex nonlinear relations by a set of IF-THEN rules. The membership functions are designed by intuitive human reasoning. This poses two problems, one for different control application a new set of membership functions have to developed and second, once these membership functions are developed and implemented there is no means of changing them. This means fuzzy logic lacks a learning function. Neural network on the other hand self-organizes the mapping relationship by learning. So by integrating neural networks and fuzzy logic it is possible to overcome these problems. Implementing this algorithm on a TMS320 DSP chip is discussed. Controlling a three link manipulator is taken as an example to evaluate the controller.

The object of this work is to develop an algorithm to implement such a type of controller for a specific control problem. In order to do such a task, it is necessary to have a large computing power. The TMS320C30 digital signal processor from Texas Instrument [24], with its powerful instruction sets, high-speed number crunching capabilities, innovative architectures, is ideally suited for such an application. There are commercially available boards based on TMS320C30 chips, which can be installed on a PC. A board form DSP Research [109] has been utilized for this purpose. The software for the control algorithm can be developed in C-language and can be compiled and down loaded to the DSP board. O'INCA 1131 software is used to train the neural network. A three link SCARA robot is used as an object system. A model of a three link SCARA manipulator is used to test the controller. The controller objective is to move the end-effector of the three link SCARA manipulator to the desired position with least oscillation and error. To perform this, a two-level hierarchical controller structure is used as discussed in [21]. This structure has two fuzzy controllers, one a feature extractor which is a higher level and another is the low level controller producing the desired torque. The NN can be used to adapt both the fuzzy controller. However here we propose only adapting the lower level controller.

8.2 Self-organizing Neuro-Fuzzy Controller

The learning capability of the neural network can be made use of in designing the fuzzy controller. The self-organizing fuzzy controller is one such combination of a neural network and a fuzzy controller. Figure 54 shows a schematic diagram of the system forming the self-organizing fuzzy controller. The aim of this system is to automatically form the fuzzy controller. It uses two neural networks of the back propagation learning type, NN1 and NN2. NN1 acts as a classifier of the dynamic responses of the object system being controlled (robot system). NN2, set in judge- ment mechanism 2, has knowledge of the dynamic characteristics of the object system. Judgment mechanism 2 has a self-tuning mechanism to automatically

determine the normalizing values of the membership functions to control the object system adequately. In this particular case NN1 classifies the error in the trajectory of the end effector to several typical pat- terns such as a similar pattern to the desired response or an oscillating and diverging pattern or an oscillating and slowly converging pattern or any other pattern. The result of the classification is sent to judgment mechanism 1. NN2 is made to learn the dynamic characteristics of the object system through pairs of

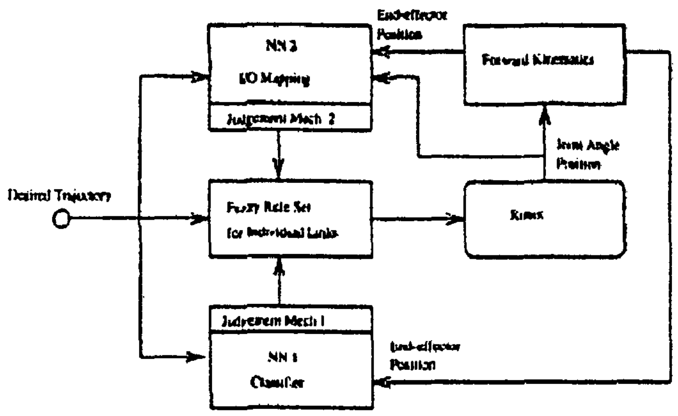


Fig 54: Shows the schematic of self-organizing fuzzy controller.

input and system response. NN2 can then be used to simulate the object system in cases where it is too risky to control the object system with an incomplete fuzzy controller. The judgment mechanism 1 decides whether a next fuzzy rule has to be formed or not and whether tuning operation is needed or not for the scaling of member- ship functions at the currently established rules. Next the labels of the fuzzy variables in the antecedent part of the fuzzy control rule are determined from the information of the dynamic response of the object system and the rule is integrated using the fuzzy variable in the consequent part whose label is calculated in the judgment mechanism 2. If there are no rules at the initial state in the fuzzy controller, then the judgment mechanism sets up some initial control value to control the object system. The object system is controlled by the established fuzzy rules and the information of the dynamic response is sent to NN1. The self tuning operation of the normalizing values for currently established fuzzy control rules is repeated until the dynamic response is classified to desired response by NN1. If the dynamic response of the object system is classified to the desired response by NN1, the judgment mechanism 1 decides to form a next new fuzzy control rule. The judgment mechanism 2 mainly plays a role in determining the quantities for the label of the fuzzy variable in the consequent part and in tuning values for the fuzzy variables in both the antecedent and consequent parts.

8.3 Hardware and Software

A fuzzy controller consists of mainly three parts namely, a ***fuzzification*** of the input signals, an ***inferencing*** mechanism and a ***defuzzification*** process to produce the crisp output. There are several softwares available in the market which can be implemented for developing a fuzzy controller. However, in order to make it adaptive fuzzy controller, it is necessary to develop the entire code, so as to have mechanism to update the rules or tune the membership functions. The block diagram of the implementation scheme is shown in Figure 55. A Texas Instruments TMS320C30 DSP chip is used for real-time control. The TMS320C30 [108] DSP is a third generation signal processing chip from Texas Instruments. The chip architectures, featuring multiple buses, provide a high degree of parallelism. It has a rich instruction set, with many special digital signal processing instructions. It can perform parallel multiply and add operations on integer or floating point data in a single clock cycle.

The TIGER 30 board [25] from DSP research incorporates a TMS320C30 running at its maximum clock speed of 40 MHz, yielding a maximum throughput of 40 MFLOPS. The TIGER 30 Development Systems includes an optimizing C compiler, supplied by Texas Instruments. The C Compiler generates efficient object code, using the parallel instructions of the TMS320C30. The output of the compiler is TMS320C30 assembly language source code, making it easy to optimize critical sections of the code. Also included is a debugging environment, which supports multiple source files and multiple watch variables, including auto-variables. The TM320C30 has two memory buses, the primary bus and the expansion bus. On the primary bus the TIGER 30 board has up to 1 M Words of static RAM on the board. On the expansion bus it has up to 8kWords of memory on the main board. The TIGER 30 is IBM PC compatible. The data can be transferred in either 16 bit or 8 bit mode. The memory on the primary bus is dual ported with host PC. Transfers can be either under CPU, interrupt or DMA control, or any combination of the three. The host interface is designed such that multiple TIGER 30 boards can reside on the bus for multi-processing applications. The TMS320C30 has two on-chip serial channels for 8/16/32 bit transfers of up to 8.3 Mbps. The TIGER 30 board has two TLC32044 AID-D/A converters. It features 14 bit resolution, a band pass switched-capacitor anti-aliasing filter and a low-pass switched capacitor output reconstruction filter. It has variable sampling rates up to 19.2 kHz. The software is developed in C language. Only the real-time component of the software is compiled using the C30 optimizing compiler. This includes all the components of the fuzzy controller. The adaptation and rule update is done on the PC. TIGER 30 board allows the PC to directly access the primary memory of the TMS320C30 bus and visa verse. This allows easy data transfer. If there are any changes in the membership function or changes in the rule base, a memory swap is done which updates the program running on the DSP board. Likewise all the parameter from the controlled system can be transferred to the PC.

As described in Section 2 there are two neural nets NN1 and NN2 which is used to adapt the fuzzy controller. NN2 is basically used to map the input-output relation of the controlled system. The NN1 is the one which maps the temporal response of the system out- put. A multilayer perceptron is used to perform this function. O'INCA [107] software is used to develop and train this neural net. The perceptron consists of two hidden layer consisting of

10 neurons. The input layer consists of 20 inputs corresponding to the samples of the temporal data of the output. This NN has a bi- nary output consisting of 3 layers. The binary number can map up to 8 different patterns. Initially this network is made to learn signals such as oscillating and

diverging pattern, an oscillating and slowly converging pattern, an asymptotically and slowly converging pattern etc. This is done by feeding the network with standard signals such as for converging oscillatory pattern, e^{-at} for asymptotically converging pattern, $\sin(\omega t+\theta)e^{-at}$ for diverging pattern and like wise. Here ω, a and θ are constants which are varied. The neural net classifies these signals irrespective of the values of the constants.

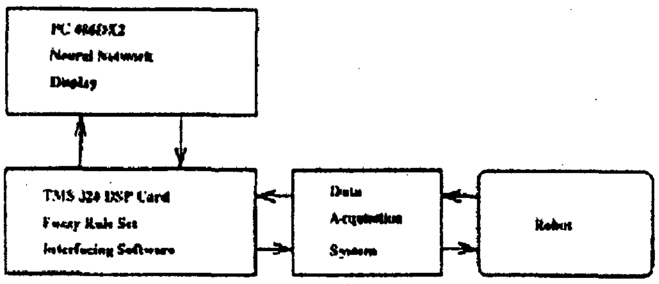


Fig: 55 Implementation scheme of Neuro fuzzy technique.

Hence, Fuzzy mathematics has provided a range of mathematical tools to formalize ill-defined descriptions in the form of linguistically stated IF-THEN rules into mathematical equations that can be implemented on digital computers. Integration of neural networks and fuzzy logic is a new direction in intelligent control engineering. A self-organizing fuzzy controller is pro- posed which can automatically tune the fuzzy rule base and also create new fuzzy rules by the help of neural networks. The learning ability of the neural network can be utilized, to optimize an adaptive fuzzy

controller. An adaptive fuzzy controller is developed which can be run for a real-time system incorporating a TMS320C30 digital signal processing chip. The controller has two parts, one which can be real-time re- sides in the DSP board and the other part consisting of adaptive mechanism runs on the PC processor. The adaptation of the fuzzy controller is carried out by classifying the temporal history of the controlled variable into different patterns. These patterns initiate changes in the components of the fuzzy controller in order to make the system perform in the desired manner.

A Three link manipulator model is used to evaluate the controller. A hierarchical structure fuzzy controller is used to control the flexible link. There are two level of hierarchy, the higher level basically extracts the feature of the manipulator link and the lower level controller produces the desired torque. The adaptation of the fuzzy controller is carried out at this level [12].

CHAPTER- 9

CONCLUSIONS AND FUTURE SCOPE OF THE WORK

9.1Conclusions

The study compares conventional PD control scheme, neuro control scheme and Neuro fuzzy control scheme. From the results of the first part of the work it has been found that the conventional control scheme using PD controller is simplest to be implemented, but it cannot cope with uncertainties in the robot dynamics.

From the second part of the work it was found that in the case of neural network based controller is designed. It has good tracking performance .Only difficulties are prior training and judicious parameters are essential for the success of neural network based controller.

Third part of the work in which ANFIS (Adaptive Neuro-Fuzzy inference system) is designed. The feasibility of ANFIS control for a three-link SCARA manipulator has been prov.ed and illustrated by simulation. The best parameters for the fuzzy controller were determined by using the ANFIS methodology and by using simulations of the SCARA manipulator dynamics. A simulation tool (i.e., Fuzzy logic toolbox (ANFIS)) was used to validate experimentally the tracking ability and the insensibility to plant parameter changes. The ANFIS controller presented very interesting tracking features and was able to respond to different dynamic conditions. In addition, the fuzzy control computation is very inexpensive, and this regulator could be used for the control of machine tools and robotics manipulators [106] without significantly increasing the cost of the drive. The only extra cost is for the optical encoder. Another advantage of this method over classical quantitative controllers is that it does not require a fixed sampling time. Therefore, the proposed design confirms the fact that fuzzy control is relevant to the fast control of non-linear processes such as SCARA manipulator control where quantitative methods are not always appropriate. Thus, the results obtained using the ANFIS controllers are encouraging when compared to conventional PD controller.

Finally, Visual display of three link SCARA Manipulator is made by using C++.

9.2 Future scope

The neural network based controller & ANFIS controller , which is developed in this work is only software , which has not been interfaced with the physical robotic arm .Taking the idea from this controller, it is possible to develop practical setup(as mentioned in chapter-8), which can be interfaced with the robot arm and can control it on-line. Practical work of this type can be done now with the recent data acquisition
of another robotic arm set up in the department.

Since the required torque calculated from the neural network controller is not smooth, improved performance is expected by the use of fuzzy logic. Due to lack of time it was not possible to implement this. So, taking the idea of ANFIS controller discussed in chapter three, software can be made, and then the practical work over it can be done.

With the help of this practical work, we can confirm the real supremacy of neuro-fuzzy controller over other conventional controller .It can be done by doing a comparative study using both the controller in turn to control the robotic arm.

REFERENCES

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

[2]. Inoue, H., "Force Feedback in Precise Assembly Tasks", MIT Artificial Intelligence Laboratory Memo 308, MIT, Cambridge, Mass., 1974

[3]. Will. P. and Grossnlan. D. "An Experimental System for Computer Controlled Mechanical Assembly)." IEEE Trans. EICYY. Devices, Vol. 29, 1975. pp. 42-48

[4]. Sciavicco. L. and Siciliano. B., Modelling and Control of Robot Manipulators. McGraw-Hill Companies. Inc., 1996

[5]. Paul, R.C... "Modellings, trajectory, Cancelation and Servoing of computerControlled Arm." A.I. Memo 177, Stanford Artificial Intelligence Lab., Stanford University. California. 1972

[6].Dubowsky. S. and Desforges, D.T. "The application of Model Referenced Adaptive control to Robotic Manipulators." Transactions of ASME, Journal of Dynamic System. Measurement and Control. Vol. 101, 1979, pp. 193-200.

[7]. Jezernik, K., Rodic, M., Safaric, R., & Lurk, B., "Neural Network Sliding Mode Robot Control." Robotica, Vol. 15, Part 1, 1997, pp 23-30

[8]. Li. Q., Poo. AN.. Lim. C.M. and Ang., M., "Neural-based Adaptive Internal Model Control for Robot Manipulators." IEEE International Conference on Neural Networks, 1905

[9]. R.J. Schilling, Fundamentals of Robotics, Prentice-Hall, 1990

[10].Acknowledgement: The authors thank **Dr. G. Kronreif** (TU Vienna) for providing realistic robot data. **Horest Ecker, Institute for Machine Dynamics and Measurement,** email:hecker@email.tuwien.ac.at, Tel: +43-1-58801 5567, and Felix Breitenecker, SIMTECH, Vienna University of Technology, Wiedner hau

[11]. Jorge I. Arciniegas, et al, "Neural-networks-Based adaptive control of flexible robotic arms", Journal of NeuroComputing, vol.1 issue 7, April 1997.

[12]. kishan kumar kumbla ,Mohammed .R, Akbarzadeh.T and Mohammad Jamshidi, "TMS320 DSP Based neuro-fuzzy controller", IEEE,Vol. 1, Issue 3, 1995

[13]. S. Nakanshi, T. Takagi, K. Unehara and Y. Gotoh, "Self-Organizing Fuzzy Controllers and Neural Networks", Proceedings of the International Conference on Fuzzy Logic & Neural Networks, Vol.1 pp 183-186, Japan, July 1990.

[14]. Kiyohiko Uehara, Masayuki Fujise, "Learning of Fuzzy-Inference Criteria with Artificial Neural Network", Proceedings of the International Conference on fizzy Logic & Neural Networks, Vol.1 pp 187-191, Japan, July 1990.

[15].Shinichi Horikawa, Takeshi Furuhashi, Shigeru Okuma and Yoshiki Uchikawa, "A Fuzzy Controller using a Neural Network and its capability to learn Expert's Control rules", Proceedings of the International Conference on Fuzzy Logic & Neural Networks, Vol. 1 pp 103-106, Japan, July 1990.

[16].Hideyuki Takagi and Isao Hayashi, "NN-Driven Fuzzy Reasoning", International Journal of Approximate Reasoning 1991, 5, pp 191-212. [7] Hisao Ishibuchi, Ryosuke Fujika and Hideo Tanaka, "Neural Networks that Learn fromFuzzy If-Then Rules, IEEE Transactions on Fuzzy Systems, Vol.1 No. 2, May 1993.

[17]. Hisao Ishibuchi, Ryosuke Fujika and Hideo Tanaka, "Neural Networks that Learn from Fuzzy If-Then Rules , IEEE Transactions on Fuzzy Systems, Vol.1 No. 2, May 1993.

[18].P.B. Mahapatra, 'GRAPHICS PROGRAMMING IN C++, K .B Publishing Co. (P) LTD.

[19].Shinichi Horikawa, Takeshi Furuhashi, and Yoshiki Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back- Propagation Algorithm", IEEE Transactions on Neural Networks, Vo1.3, No.5, September 1992.

[20].Sujeet Shenoi, Kaveh Ashenayi and Marc Timmerman, "Implementation of a Learning Fuzzy Controller", IEEE Control Systems, Vol. 15 No. 3 June 1995.

[21].M.-R. Akbarzadeh-T., M. Jamshidi, and N. Vadiee, UA Hierarchical Fuzzy Controller Using Line-Curvature Feature Extraction for a Single Flexible arm, "Proc. of the 1994 IEEE Conference on fuzzy Systems, FUZZ'94, Orlando, F1.994.

[22].S. Botros, C. Atkeson, Generalization properties of radial basis functions, in: R. Lippman, J. Moody, D. Touretzky (Eds.), Neural Information Processing, vol. 3, Morgan Kaufmann, Los Altos, CA, 1991. pp. 707-713.

[23] D. Broomhead, D. Lowe, Multivariable interpolation and adaptive networks, Complex Systems 2. (1988) 321-355.

[24] F.-C. Chen, H.K. Khalil, Adaptive control of nonlinear systems using neural networks, Int. J. Control. 55 (6) (1992) 1299-1317.

[25] F.-C. Chen, C.-C. Liu, Adaptively controlling nonlinear continuous-time systems using neural networks, IEEE Trans. Automat. Control 39 (6) (1994) 1306-1310.

[26] X. Cui, K.G. Shin, Direct control and co-ordination using neural networks, IEEE Trans. Systems Man Cybernet. 23 (3) (1993).

[27] R. Hampo, K. Marko, Neural network architectures for active suspension control, Proc. Int. Joint Conf. Neural Networks 2 (1991) 765-770.

[28] M. Jordan, D. Rumelhart, Forward models: supervised learning with a distal teacher, Tech. Report MIT Center for Cognitive Science, Occasional Paper No. 40, 1989.

[29] M. Kawato, Computational schemes and neural network models for formation and control of multijoint arm trajectory, in: T. Miller, R. Sutton, P. Werbos (Eds.), Neural Networks for Control, MIT Press, Cambridge, MA, 1990.

[30] F.L. Lewis, K. Liu, A. Ye, Neural net robot controller with guaranteed tracking performance, Proc. IEEE Int. Symp. Intelligent Control, August 1993, pp. 225-231.

[31] F.L. Lewis, K. Liu, A. Ye, Neural net robot controller: structure and stability proofs, Proc. IEEE Conf. Decision and Control, San Antonio, December 1993, pp. 2785-2791.

[32] F.L. Lewis, K. Liu, A. Ye, Neural net robot controller with guaranteed tracking performance, IEEE Trans. Neural Networks 6 (3) (1995) 703.

[33] C.-C. Liu, F.-C. Chen, Adaptive control of non-linear continuous-time systems using neural networks: general relative degree and MIMO cases, Int. J. Control 58 (2) (1993) 317-335.

[34] C. Micchelli, Interpolation of scattered data: distance matrices and conditionally positive definite functions, Constr. Approx. 2 (1986) 11-22.

[35] W.T. Miller, R.S. Sutton, P.J. Werbos. Neural Networks for Control, MIT Press, Cambridge, MA, 1991.

[36] J. Moody, C. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (1989) 281-294.

[37] K. Narendra, S. Mukhopadhyay, Intelligent control using neural networks, IEEE Control Systems Magn. (1992) 11.- 18.

[38] D. Nguyen, B. Widrow, Neural networks for self-learning control systems, Int. J. Control 54 (6) (1991) 1439-1451.

[39] T. Poggio, F. Girosi, Regularization algorithms for learning that are equivalent to multilayer networks, Science 247 (1990) 978-982.

[40] M.M. Polycarpou, P.A. Ioannou, Idenfication and control using neural models: design and stability anaysis, Tech. Report 91-09-01. Dept. Elect. Eng. Sys., University of South Carolina, September 1991.

[41] M.M. Polycarpou, P.A. Ioannou. Neural networks as on-line approximator of nonlinear systems, IEEE Conf. Decision and Control, Tucson, December 1992, pp. 7-12.

[42] M. Powell, Radial basis functions for multivariable interpolation: a review, in: J. Mason, M. Cox (Eds.), Algorithms for Approximation, Clarendon Press, Oxford, 1987.

[43] G.A. Rovithakis, M.A. Christodoulou. Adaptive control of unknown plants using dynamical neural networks, IEEE Trans. Systems Man Cybernet., 1994, to appear.

[44] N. Sadegh, Nonlinear identification and control via neural networks, Control Systems with Inexact Dynamics Models, DSC-vol. 33, ASME Winter Annual Meeting, 1991.

[45] N. Sadegh, A perceptron network for functional identification and control of nonlinear systems, IEEE Trans. Neural Networks 4 (6) (1993) 982-988.

[46] R.M. Sanncr, J.-J. E. Slotine, Stablc adaptivc control and recursive identification using radial Gaussian networks, Proc. IEEE Conf. Decision and Control, Brighton, 1991.

[47] R. Sanner, J. Slotine, Gaussian networks for direct adaptive control, IEEE Trans. Neural Networks 3 (4) (1992) 837-863.

[48] D.A. White, D.A. Sofge, Handbook of Intelligent Control, Van Nostrand Reinhold, New York, 1992.

[49] M. Vandergrift, F.L. Lewis, S. Zhu, Flexible-link robot arm control by a feedback lineariza- tion/singular perturbation approach, J. Robotic Systems 11 (7) (1994) 591-603.

[50] A. Ye, F.L. Lewis, A neural network controller for feedback linearization, Proc. IEEE Conf. Decision and Control, December 1994, pp. 2494- 2499.

[51] A. Ye. M. Vandergrift, F.L. Lewis, A neural network controller for flexible-link robots, Proc. IEEE Int. Symp. Intelligent Control, 1994, pp. 63-68.

[52] Albus, J. S. 1975. A new approach to manipulator control: Cerebellar model articulation controller (CMAC). Trans.ASME J. Dynamic Systems, Measurement and Control, September, pp. 220–223.

[53] Chen, P.C.Y., Mills, J. K., and Smith, K. C. 1996. Performance improvement of robot continuous-path operation through iterative learning using neural networks. Machine Learning Journal 23:191–220.

[54]Chen, P.C.Y., Ogilvie, A., Clark, C., Zhou, K., and Mills, J. K. 1998. Development of a neural network module for improving the performance of a commercial robot. World Automation Conference, Alaska, May, pp. 288–293.

[55]Frank, W. P. 1976. Introduction to Sensitivity Theory. San Diego: Academic Press. Graham, D.P.W., and D'Eleuterio, G.M.T. 1990. MOVE—A neural network paradigm for robotic control. 6th CASI Conference on Astronautics, Ottawa, Canada, October 21, pp. 588–593.

[56]Graham, D.P.W., and D'Eleuterio, G.M.T. 1991. Robotic control using a modular architecture of cooperative artifi-cial neural networks. In *Artificial Neural Networks*, vol. 1,ed. T. Kohonen, 365–370.

[57]Jacobs, R., and Jordan, M. I. 1993. Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man, and Cybernetics* 3(2):337–345.

[58]Jacobs, R., Jordan, M. I., Nowlan, S. J., and Hinton, G.E. 1991. Adaptive mixtures of local experts. *Neural Computation* 3(1):79–87.

[59]Liu, H. T., and Mills, J. K. 1998. ROBOTOOL: A simulation software for robot performance analysis. *Proceedings of the Canadian Society for Mechanical Engineering Forum*, Toronto, Ontario, June, pp. 1044–1049.

[60]Rueckl, J. G., Cave, K. R., and Kosslyn, S. M. 1989. Why and "what" and "where" processed by separate cortical visual systems? A computational investigation. *Journal of* Cognitive Neuroscience 1:171–186.

[61]Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning representations by

back-propagating error. *Nature (London)* 323:533–536.

[62] Y. Xu, B. Brown, S. Aoki, and T. Kanade, "Mobility and manipulation of a light-weight space robot," *IEEE Int. Con$ Intell. Robots Syst.,* 1992.

[63] H. Ueno, Y. Xu, B. Brown, M. Ueno, and T. Kanade, "On control and planning of a flexible space manipulator," *IEEE Int. Con$ Syst. Eng.,* 1990.

[64] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error- learning neural network for trajectory control of a robotic manipulator," *Neural Net.,* vol. 11, pp. 251-265, 1988.

[65] J. McClelland, D. Rumelhart, and PDP Research Group, *Parallel Distrib uted Processing,* vol. I. Cambridge, MA: M.I.T. Press, 1986.

[66] K. Wilhelmsen and N. Cotter, "Neural network based controllers for a single-degree-of-freedom robotic arm," *Int. J. Con5 Neural Net..* 1990.

[67] K.S. Narendra and K. Parthasarathy, "Identification and control of dy namical systems using neural networks," *IEEE Trans. Neural Net.,* vol. 1, no. 1, 1992.

[68] F. L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Networks*, vol. 6, pp. 225–231, May 1995.

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

[70] C. M. Kwan and F. L. Lewis, "Robust backstepping control of nonlinear systems using neural networks," *IEEE Trans. Syst., Man, Cybern. A*, vol. 30, pp. 753–766, Nov. 2000.

[71] Robust backstepping control of induction motors using neural networks," *IEEE Trans. Neural Networks*, vol. 11, pp. 1178–1187, Sept. 2000.

[72] I. Kanellakopoulos, P. V. Kokotovic, and A. S. Morse, "Systematic design of adaptive controllers for feedback linearizable systems," *IEEE Trans. Automat. Contr.*, vol. 36, pp. 1241–1253, Nov. 1991.

[73] P. V. Kokotovic, "The joy of feedback: Nonlinear and adaptive," IEEE Contr. Syst. Mag., vol. 12, pp. 7–17, June 1992.

[74] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," IEEE Trans. Inform. Theory, vol. 39, pp. 930–945, May 1993.

[75] G. Cybenko, "Approximation by superpositions of a sigmoidal function," Math.

Control, Signal Syst., vol. 2, no. 4, pp. 303–314, 1989.

[76] N. Sadegh, “A perceptron network for functional identification an control of nonlinear systems,” IEEE Trans. Neural Networks, vol. 4, pp. 982–988, Nov. 1993.

[77] B. Igelnik and Y. H. Pao, “Stochastic choice of basis functions in adaptive function approximation and the functional-link net,” IEEE Trans. Neural Networks, vol. 6, pp. 1320–1329, Nov. 1995.

[78] M. M. Polycarpou, “Stable adaptive neural control scheme for nonlinear systems,” IEEE Trans. Automat. Contr., vol. 41, pp. 447–451, Mar. 1996.

[79] M. M. Polycarpou and M. Mears, “Stable adaptive tracking of uncertain systmes using nonlinearly parameterized on-line approximators,” Int. J. Control, vol. 70, no. 3, pp. 363–384, 1998.

[80] Y. Zhang, P.-Y. Peng, and Z.-P. Jiang, “Stable neural controller design for unknown nonlinear systems using backstepping,” IEEE Trans. Neural Networks, vol. 11, pp. 1347–1360, Nov. 2000.

[81] S. He, K. Reif, and R. Unbehauen, “A neural approach for control of nonlinear systems with feedback linearization,” IEEE Trans. Neural Networks, vol. 9, pp. 1409–1421, Nov. 1998.

[82] Spong MW, Vidyasagar M. Robust linear compensator design for nonlinear robotic control. IEEE J Rob Autom 1987;RA-3:345–51.

[83] Hornik K, Stinchcombe M, White H. Multilayered feedforward networks are universal approximators. Neural Networks 1989;2:359–66.

[84] Powell MJD. Radial basis function approximations to polynomials. Proceedings 12th Biennial Numerical Analysis Conference, 1987.

[85] Miller WT, Glanz FH, Kraft III LG. Application of a general learning algorithm to the control of robotic manipulator. Int J Rob Res 1987:84–98.

[86] Goldberg KY, Pearlmutter BA. Using backpropagation with temporal windows to learn the dynamics of the cmu direct drive arm ii. In: D.S. Touretzky, editor. Advances in Neural Inf Processing Systems, 1989.

[87] Wang D, Bao P. Enhancing the estimation of plant jacobian for adaptive neural inverse control. Neurocomputing 2000;34:99–115.

[88] Narendra KS, Parthasarathy K. Identification and control of dynamic systems using neural networks. IEEE Trans Neural Networks 1990;1(1):4–27.

[89] Jordan MI. Supervised learning and systems with excess degree of freedom. COINS Tech. Rep. no. 88-27, 1988.

[90] Nguyen D, Widrow B. Neural networks for self-learning control systems. Int J Control 1991;54(6):1439–51.

[91] Miyamoto H, Kawato M, Setoyama T, Suzuki R. Feedback error learning neural networks for trajectory control of a robotic manipulator. Neural Networks 1988; 1:251–65.

[92] Gomi H, Kawato M. Neural network model control for a closed loop system using feedback-error learning. Neural Networks 1993;6(7):933–46.

[93] Hwang JN, Choi JJ, Oh S, Marks II RJ. Query based learning applied to partially trained multilayered perceptrons. IEEE Trans Neural Networks 1991;2(1):131–6.

[94] Hoskins DA, Hwang JN, Vagners J. Iterative inversion of neural networks and its application to adaptive control of nonlinear systems. IEEE Trans Neural Networks 1992;3(2):292–301.

[95] Behera L, Gopal M, Chaudhury S. On inversion of rbf network and its application to adaptive control of nonlinear systems. Proc IEE Control Theory Appl 1995;143(6).

[96] Behera L, Gopal M, Chaudhury S. On adaptive trajectory tracking of a robot manipulator using inversion of its neural emulator. IEEE Trans Neural Networks 1996;7(6):1401–14.

[97] Poznyak AS, Sanchez EN, Wu W, Perez JP. Nonlinear adaptive trajectory tracking using dynamic neural networks. IEEE Trans Neural Networks 1999;10(6):1402–11.

[98] Ahmed MS. Neural net based mrac for a class of nonlinear plants. Neural Networks 2000;13:111–24.

[99] Jagannathan S, Lewis FL. Multilayer discrete-time neural net controller with guaranteed performance. IEEE Trans Neural Networks 1996;7(1):107–30.

[100] Kwan C, Lewis FL, Dawson DM. Robust neural-network control of rigid-link electrically driven robots. IEEE Trans Neural Networks 1998;9(4):581–8.

[101] Choi JY, Farrell JA. Nonlinear adaptive control using networks of piecewise linear

approximators. IEEE Trans Neural Networks 2000;11(2):390–401.

[102] Poggio TA, Girosi F. Networks for approximation and learning. Proc IEEE 1990;78(9):1481–97.

[103] Sunil Elanayar VT, Shin YC. Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. IEEE Trans Neural Networks 1994;5(4):594–603.

[104] Chen S, Cowans CFN, Billings SA, Grant PM. Parallel recursive prediction error algorithm for training layered neural networks. Int J Control 1990;51(6):1215–28.

[105] Iiguni Y, Sakai H, Tokumaru H. A real-time learning algorithm for a multilayered network based on extended kalman filter. IEEE Trans Signal Process 1992:40.

[106].Marian B Gorzalczany "On some idea of a neuro fuzzy controller ", Journal of Information Science, Elsevier, 1999.

[107].O'INCA Design Framework, User's Manual, Intelligent Machines Inc.

[108]. Texas Instruments, TMS320C3x User's Guide.

[109]. DSP Research, TIGER 30 Reference Manual.

[110]. http//:192.168.121.13

[111] www.mathworks.com.

LIST OF FIGURES