Proceedings of DETC'99
1999 ASME Design Engineering Technical Conferences
September 12-15, 1999, Las Vegas, Nevada, USA

DETC99/DAC-8659

MODEL REFERENCE LEARNING CONTROL FOR RIGID ROBOTS

Guruprasad K. R.
Robotics & CAD Laboratory
Department of Mechanical Engineering
Indian Institute of Science
Bangalore 560 012, INDIA
Email: guru@mecheng.iisc.ernet.in

Ashitava Ghosal
Robotics & CAD Laboratory
Department of Mechanical Engineering
Indian Institute of Science
Bangalore 560 012, INDIA
Email: asitava@mecheng.iisc.ernet.in

ABSTRACT

The equations of motion of a rigid robot are often known only approximately, as some of the parameters are not known exactly and there are also unmodelled nonlinearities. Most adaptive control schemes can estimate the parameters if the structure of the equations is known, but are not very useful if structure itself is not known.

In this paper we propose a model reference learning control scheme using Adaptive Network based Fuzzy Inference System (ANFIS) for control of rigid robots whose model may have parametric and structural uncertainties. The approximate model of a robot, which may differ *very significantly* from the actual robot in parametric values and structure, is used as a reference plant and a nonlinear model based controller is designed based on this model. The ANFIS corrector provides an additional correction to control input as a function of the present and desired states of the plant. The error between states of plant and that of reference plant is used to tune the ANFIS corrector.

The proposed control scheme has been implemented for a two-degree-of-freedom serial rigid robot. The results of the simulation experiments carried out show that the proposed control scheme can learn to control the unmodelled dynamics. The ANFIS controller is shown to give improved performance for parameter as well as structural uncertainties.

INTRODUCTION

Control of dynamical systems modeled as linear systems is very well developed and theories and tools are available for design of linear systems. When the transfer function (or state space realization) is not known exactly, adaptive controllers like model reference adaptive control (MRAC) (Sastry, 1984) are available.

When the dynamical systems are modeled as nonlinear, design of controllers is much more difficult. In recent years, based on new geometric methods and for a class of nonlinear systems called feedback linearizable systems, nonlinear controllers have been designed which guarantee desired performance. The model based control scheme (Craig, 1989) used for robot control is an example of such controllers.

In designing controllers for a nonlinear plant, using the techniques of feedback linearisation, accurate knowledge of dynamic equations modeling the system is very useful. Unfortunately for most of the practical systems, the model may not be known exactly. In such cases one can go for system identification techniques like Kalman filtering (Franklin, et.al, 1990; Astrom, et. al., 1984), or advanced methods like use of artificial neural networks. Adaptive controllers which estimate the unknown parameters online have also been used for rigid robots (Craig, 1986).

Most of the above mentioned controllers assume *a priori* knowledge of the structure of the dynamic equations and can handle parametric uncertainties. In this paper, we use the Adaptive-Neuro Fuzzy Inference System (ANFIS) proposed by Jang (Jang, et. al., 1993) to control rigid robots which have both parametric and structural uncertainties. There are other intelligent controllers proposed by researchers such as a Fuzzy-Gaussian Neural Networks (FGNN) (Watanabe, et. al., 1996) and the work by Chen and Gill (Chen, et. al., 1996), however, Jang and Sun (Jang, et. al., 1993) show that ANFIS performance is better than other prediction methods, including artificial neural networks, in terms of quicker learning and minimization of error.

This paper differs from the existing works using ANFIS in the sense that we propose a *nonlinear model reference learning control scheme* using ANFIS which takes care of the *unstructured uncertainty* in a dynamical system. We also show that the ANFIS based controller can handle both parametric and structural uncertainties. An approximate model, which may differ significantly from the actual robot in terms of parametric values and structure, is used as the reference plant. Based on the reference plant model, we design a classical nonlinear controller and the ANFIS is used as a *corrector* which compensates for unknown dynamics of the plant.

The paper is organized as follows: In section 2, we describe, in general, the model reference learning control scheme using ANFIS. In section 3, we give the details of the implementation on a 2R robot. In section 4, we present the simulation results and in section 5, we present the conclusions.

MODEL REFERENCE LEARNING CONTROL USING ANFIS

The block diagram of a *Model reference learning controller using ANFIS* is shown in figure 1. The *plant* is a nonlinear dynamical system for which the dynamic equations in state space form can be written as

$$\dot{X} = f(X, U) \tag{1}$$

where, X is the state vector and U is the input vector.

We consider the case where the dynamics (1) of plant are known only approximately, both in *parameters and structure*. The model of the plant used for the *reference plant* is given by

$$\dot{X} = \hat{f}(X, U) \tag{2}$$

where $\hat{f}$ denotes the approximate model of the plant.

The *controller* block is a nonlinear controller designed based on reference plant dynamic equations (2) and is in the form

$$U' = g(X, X_d) \tag{3}$$

where X_d is the desired state. The compensation for unmodelled dynamics is added as a correction to the control input, U', by the *ANFIS corrector* and we denote the compensation by δU which is of the form

$$\delta U = \delta g(X, X_d) \tag{4}$$

The total control input to the plant is given by

$$U = U' + \delta U \tag{5}$$

The ANFIS corrector

The added correction, δU, is such that with the total control U, the plant tries to follow the response of reference plant. Inputs to the *ANFIS corrector* are feedback states X and the desired states X_d. Alternatively, error between X and X_d can also be used as input instead of X_d. The output is the correction δU. The *ANFIS corrector* architecture can be designed based on the approximate plant structure, i.e., based on number of states and size of vector U. Further, the knowledge of the plant can be exploited in optimizing the network.

Learning of ANFIS corrector

The *ANFIS corrector* has to learn the function δg as given in equation (4). For learning, we use a training data set[1]. A training data set is the desired input-output pair of the network. The training data is derived from the input (X, X_d) and the error E. To compute δU for given X, we can use the *Jacobian, J*, of the plant. If the Jacobian is known, we can write

$$\dot{U} = J\dot{X}$$

which can be approximated as (assuming constant time step)

$$\Delta U = J \Delta X \tag{6}$$

Equation (6) gives us a transformation from ΔX to ΔU. In other words, for a required correction in X, we can find correction for U. The correction ΔU should be such that the states of the plant should be as close as possible to that of reference plant. Thus the correction required at output of the plant is E. Hence we can rewrite (6) as

[1]The network can be trained both online and off-line. Initially the *ANFIS Corrector* may be trained off-line. Online learning can handle variation of the plant parameters while being controlled.

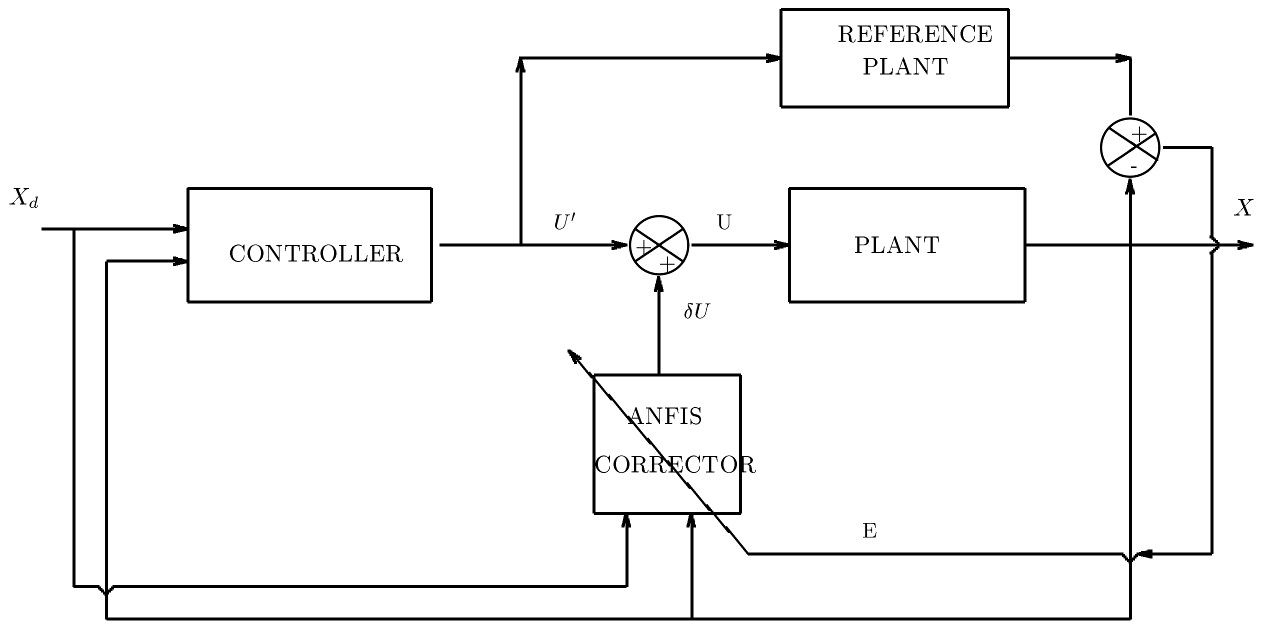

Figure 1. Block Diagram of Proposed Control Scheme

$$\Delta U = JE$$

We do not know the Jacobian, J, *a priori* as the plant dynamics is not known exactly. One option is to compute J online from samples of input and corresponding output of the plant. Once sufficient number of samples are taken, we can find J by using the pseudo inverse as

$$J(X,\dot{X}) = [\Delta U_1 \Delta U_2 ... \Delta U_n]^T A^{\#}$$

where, $A = [E_1 E_2 ... E_n]^T$ and $A^{\#}$ denotes the pseudo-inverse of A. The Jacobian, J, has to be computed for each input X. If we sample X and E at a time interval of T, then we can get a training data for time nT. The time interval, T, should be kept at minimum possible value so that the computed J is sufficiently accurate. Most mechanical systems are not very fast and hence we can compute J to sufficient accuracy with a realizable T. Alternatively we can use an Artificial Neural Networks (ANN) for the purpose of computing J online.

Once we have the training data for the network, *back-propagation (BP)* algorithm (using gradient descent (GD) method) can be used for training the network. If consequent layer is linear, i.e. first order Sugeno model (Kosko, 1994) is used for FIS, hybrid learning method (jang. et. al., 1995) can be used. In hybrid method both GD and least square estimation (LSE) are used.

IMPLEMENTATION FOR A 2-R ROBOT

Robot dynamic equations are nonlinear and coupled and many of the parameters like inertia, location of center of gravity etc. will not be know exactly. In addition, there will be unmodelled nonlinearities like, nonlinear friction, back-lash etc. In spite of the uncertainties in parameters and structure, we are required to design a tracking controller for robots. In this section we discuss the implementation details of *model reference learning control scheme* for a 2-R robot.

Dynamics of 2-R robot

The dynamic equations of a robot can be derived using the Lagrangian formulation (Craig, 1989). The dynamic equations can be represented in matrix form as

$$\tau = \mathbf{M}(\theta)\ddot{\theta} + \mathbf{C}(\theta,\dot{\theta}) \qquad (7)$$

where, $\theta(t)$ is the $n \times 1$ vector of joint angles, $\mathbf{M}(\theta)$ is the $n \times n$ mass matrix which is positive definite and symmetric, $\mathbf{C}(\theta,\dot{\theta})$ is the $n \times 1$ vector of Coriolis, centrifugal, and gravity torques, and τ is the vector of joint input torques.

In this paper we consider a two link robot with rotary joints (2R) as shown schematically in figure 2. For a 2R robot, the components of M and C in equation (7) can be written as

$$\begin{aligned}
M_{11} &= m_1 r_1^2 + I_1 + I_2 + (m_2 + m_p + m_t) l_1^2 + \\
&\quad m_2 r_2^2 + m_p l_2^2 + 2 l_1 (r_2 m_2 + m_p l_2) cos(\theta_2), \\
M_{12} &= M_{21} = I_2 + m_2 r_2^2 + m_p l_2^2 + \\
&\quad l_1 (m_2 r_2 + m_p l_2) cos(\theta_2), \\
M_{22} &= I_2 + m_2 r_2^2 + m_p l_2^2, \\
C_1 &= -(m_2 r_2 + m_p l_2) l_1 sin(\theta_2) \dot{\theta}_2 (2\dot{\theta}_1 + \dot{\theta}_2) + \\
&\quad m_1 g r_1 cos(\theta_1) + (m_2 + m_p + m_t) g l_1 cos(\theta_1) + \\
&\quad cos(\theta_1 + \theta_2) g (m_2 r_2 + m_P l_2) + f_1(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2), \\
C_2 &= (m_2 r_2 + m_p l_2)(l_1 sin(\theta_2) \dot{\theta} 1^2 + \\
&\quad g cos(\theta_1 + \theta_2)) + f_2(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)
\end{aligned}$$

where f_1 and f_2 represent unmodelled nonlinearities, m_i, l_i, I_i and r_i are the mass, length, inertia and location of the center of gravity of link i respectively and m_p & m_t are payload and motor masses respectively.

As the model (7) may not be known exactly, we consider the reference plant model as

$$\hat{\tau} = \hat{\mathbf{M}}(\theta)\ddot{\theta} + \hat{\mathbf{C}}(\theta, \dot{\theta}) \tag{8}$$

where $\hat{M}$ and $\hat{C}$ are estimates of M and C respectively. There could be differences in structural as well as parameter values between M and C and their respective estimates.

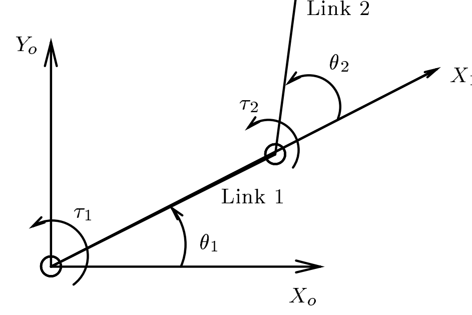

Figure 2. A schematic of a 2R planar rigid robot.

The controller

We use a model based controller designed based on reference plant dynamics (8) for the *controller* block of figure 1. This is of the form

$$\tau' = \hat{M}(\theta)\tau_{pd} + \hat{C}(\theta, \dot{\theta}) \tag{9}$$

where $\tau_{pd} = \ddot{\theta}_d + K_p e + K_v \dot{e}$.

If the plant were to be known exactly, the control law would be of the form

$$\tau = M(\theta)\tau_{pd} + C(\theta, \dot{\theta}) \tag{10}$$

ANFIS architecture

The difference between control inputs τ and τ' is taken care of by the ANFIS corrector. We can write the difference $\delta\tau = \tau - \tau'$ as

$$\delta\tau = \tilde{M}(\theta)\tau_{pd} + \tilde{C}(\theta, \dot{\theta}) \tag{11}$$

or in general

$$\delta\tau = f(\theta, \tau_{pd}) + g(\theta, \dot{\theta}) \tag{12}$$

Thus the ANFIS corrector needs to learn to approximate the functions $f()$ and $g()$ as given in (12). Inputs to the *ANFIS corrector* are θ, $\dot{\theta}$ and τ_{pd} and the output is $\delta\tau$. For a 2-R robot number of inputs and outputs are 6 and 2 respectively.

We can utilize the knowledge of the system to simplify the network. In the case of robot dynamics, functions of θ are always trigonometric functions, hence we can use $\sin(\theta)$ and $\cos(\theta)$ instead of θ. Even though this increases the number of inputs to the network by 2, the approximation characteristics improves. With this observation, if we partition each input space into n fuzzy sets then total number of rules will be n^8 [2]. In equation (12), there are two parts, f and g and considering the replacement of $\sin(\theta)$ and $\cos(\theta)$ for θ as input we have

$$\delta\tau = f(\sin(\theta), \cos(\theta), \tau_{pd}) + g(\sin(\theta), \cos(\theta), \dot{\theta})$$

[2]The number of partitions need not be same for all the inputs. If n_i represents number of fuzzy partitions in ith input, then number of rules in general is $\prod_{i=1}^{N} n_i$, where N is number of inputs.

This reduces the number of rules to $2n^6$. For $n = 2$, the number of rules are reduced from 256 to 128. The reduction in number of rules is more significant for larger n.

The figure 3 shows the architecture of *ANFIS* for 2-R robot. We partition each input into two fuzzy sets, namely - *negative* and *positive*. There are 5 layers in the network and we describe each of the layers in detail below. The notation $f_{l,i}$ indicates node function of i-th node in layer l.

Layer 1:

The nodes in this layer evaluate the membership functions, i.e., the fuzzification operation is carried out. The Gaussian membership functions (MF) are claimed to give better results compared to other MFs such as triangular, trapezoidal, and Cauchy functions (Lotfi, et. al., 1996) and hence we use Gaussian MFs.

There are 16 adaptive nodes in this layer. Each input node is connected to two nodes in this layer as shown in the figure 3. The node function is Gaussian and is given by

$$f_{1,i}(x) = e^{-(\frac{x-c}{a})^2}$$

where a and c are parameters of the node which constitute the premise parameter set.

The linguistic meanings that can be attached to these membership functions are *negative* and *positive*.

Layer 2:

There are 128 *fixed* nodes in this layer which multiply the incoming signals and send the product out. Each node output represents the firing strength of corresponding rule and we can write.

$$f_{2,i}(X) = \prod_{j=1}^{n} X_i$$

where X is set of all outputs of nodes in previous layer which are connected to the ith node of this layer.

It should be noted that for the first 64 nodes, the connectivity is restricted to first 6 inputs and next 64 nodes to the last 6 inputs. This is because of presence of two parts in the function (12) namely $f()$ and $g()$, $f()$ being function of $sin(\theta)$, $\cos(\theta)$ and τ_{pd} which constitute the inputs 1-6 and $g()$ being function of $sin(\theta)$, $\cos(\theta)$ and $\dot{\theta}$ which constitute input 2-8.

Layer 3:

There are 128 *fixed* nodes in this layer to compute the ratio of i-th rule's firing strength to sum of firing strengths of all rules, i.e., they compute the normalized firing strength of corresponding rule. The computation is of the form

$$f_{3,i} = \frac{f_{2,i}}{\sum_{j=1}^{128} f_{2,j}}$$

Layer 4:

We have 256 nodes in this layer. Each node in layer 3 is connected to two *adaptive* nodes of this layer. The node function is the consequent part of first order Sugeno model of fuzzy inference system multiplied by firing strength of corresponding rule, i.e., the nodes compute the outcome of the rule and multiply them by the normalized firing strength.

If $i \leq 128$, we have

$$f_{4,i} = w_n(\sum_{j=1}^{6} a_{i,j}x_j + a_{i,7})$$

for $i > 128$ we have

$$f_{4,i} = w_n(\sum_{j=3}^{8} a_{i,j-2}x_j + a_{i,7})$$

where, w_n is output of node of previous layer connected to the node under consideration, x_i is i-th input and $a_{i,j}$ is j-th parameter of i-th node. The odd numbered nodes compute the contribution of corresponding rule to τ_1 while even numbered nodes compute contribution to τ_2.

Layer 5:

This layer has 2 *fixed* nodes which add all the inputs. These nodes sum contribution of each rule to corresponding outputs namely τ_1 and τ_2 to get the final values.

$$f_{5,1} = \sum_{i=1}^{128} f_{4,2i-1}$$

$$f_{5,2} = \sum_{i=1}^{128} f_{4,2i}$$

The output of fifth layer is the output of the network.

Learning of the ANFIS corrector

The ANFIS corrector described in previous sections should be trained to approximate the function (12). A

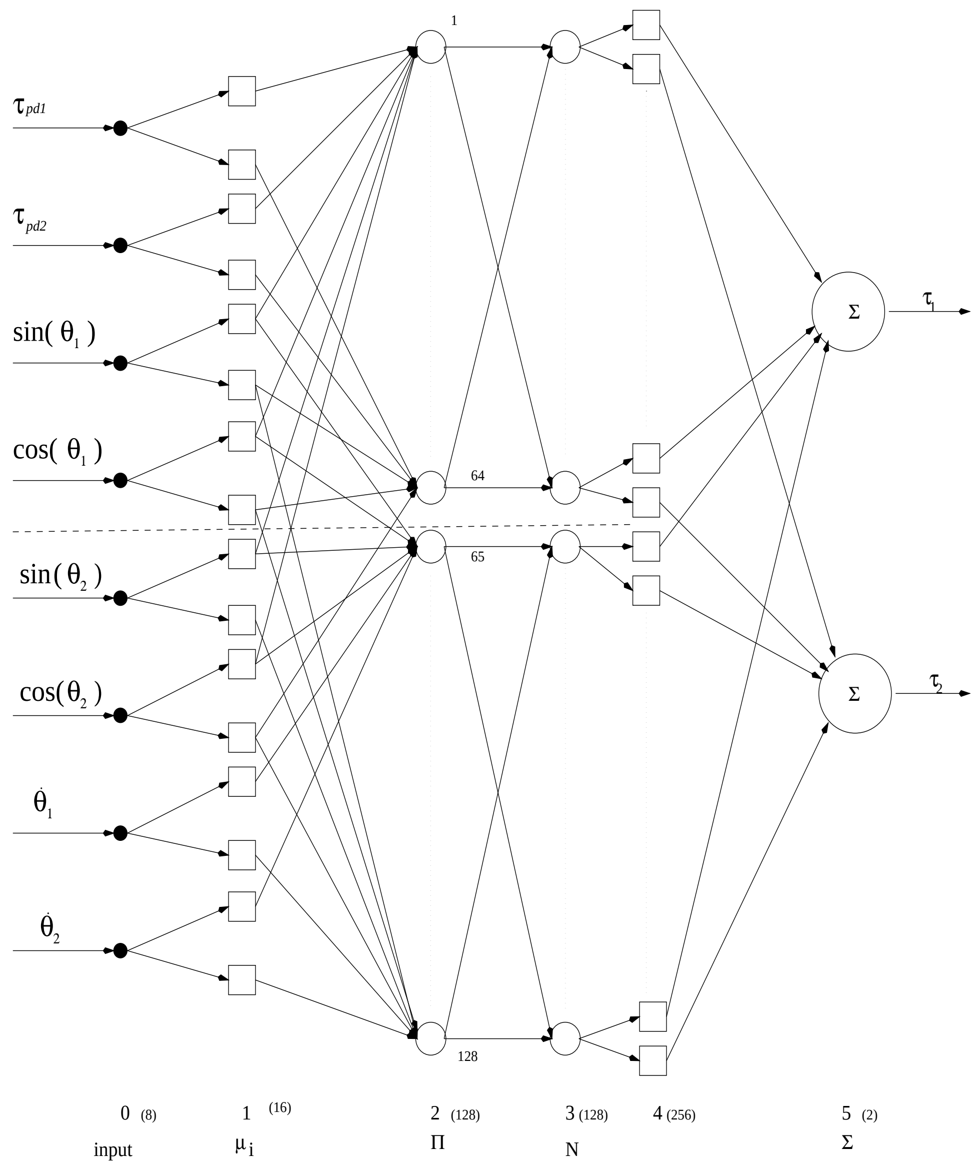

Figure 3. Architecture of ANFIS

training data set and a learning algorithm is required for the purpose. The process of training data set generation and the learning algorithm is discussed next.

A. Training Data: For learning, we need to generate a training data which consists of desired input-output values of the network. For online learning, this can be achieved by observing error in states E, for present feedback and desired states. The inputs are $sin(\theta)$, $cos(\theta)$, $\dot{\theta}$ and τ_{pd} which can be computed directly from feedback and desired values of the states. The output $\delta\tau$ can be computed from E using online computed Jacobian, J, as discussed in section 2. For off-line learning, a set of training data is required, i.e., we need to have a set of desired input-output values of network.

For the purpose of simulation, we compute $\delta\tau$ directly, by computing difference between two control laws (9) and (10) as given by equation (11). The figure 4 illustrates the training data set generation procedure adopted in this work. The blocks *controller1* and *controller2* are implementation of control laws (9) and (10) respectively. As both controller blocks receive same inputs, the difference in their outputs is nothing but $\delta\tau$, the desired output of ANFIS corrector. By simulating 4 for different trajectories and storing the values of desired and actual states and $\delta\tau$, we get the training data set. The trajectories should be such that input space

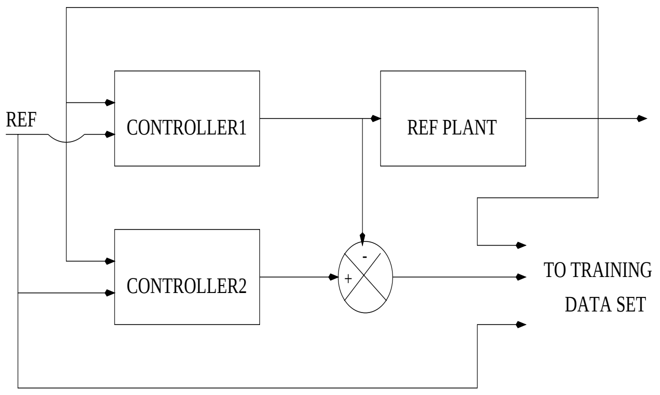

Figure 4. Training Data Set generation

is spanned uniformly. We have used sinusoidal trajectories for the purpose.

B. Learning Algorithm: Once the training data is available, we can train the network using a learning algorithm. The nodes in layer 5 being linear, hybrid learning method (jang. et. al., 1995) can be used. LSE can be used to tune the parameters of nodes in layer 5 and GD can be used to tune parameters in layer 2. The use of LSE in this case involves multiplications of large matrices of the size 896×896. This is computationally very expensive and hence we have used only the gradient descent based backpropagation algorithm for learning.

Extension to higher degree-of-freedom robots

With the increase in the number of degrees-of-freedom in a robot, the number of rules will clearly increase and with it the computational complexity. As discussed earlier, the number of rules are $2n^6$ for an n degree-of-freedom robot – the number of rules for a six degree-of-freedom robot will be 93312. It may be noted that each rule is independent and can be computed in parallel. In this algorithm, the number of layers is always 5 and is independent of n. Hence there is no increase in complexity due to the increase in degrees-of-freedom as far as the number of layers is concerned. In a parallel implementation, the computational time is independent of n. Although, the results shown in this paper were obtained by a software implementation, for actual applications in control of multi-degree-of-freedom robots, the ANFIS corrector should be implemented in hardware using VLSI technology.

SIMULATION RESULTS

We present some of the results of simulation experiments in this section. We first show some representative

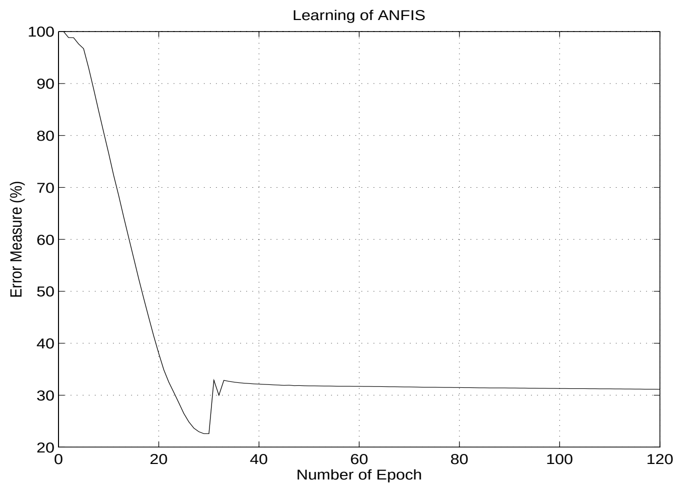

Figure 5. The history of error measure during learning process

results on learning of ANFIS and then give results for the 2R robot.

Learning of ANFIS

As discussed earlier, we use backpropagation learning algorithm to estimate both premise and consequent parameters. To ensure convergence of the algorithm, we chose the step size of gradient descent adaptively. We use following heuristic approach (Jang, 1993)

- if the error measure increases, decrease the step size by 20%
- if the error measure is decreasing continuously over 3 epochs, then increase the step size by 10%.

Figures 5 and 6 show the error measure history and figure 7 shows how the step size varies adaptively during a learning process.

It was observed that the learning algorithm can converge to different solutions with different initial conditions, and/or different ways in which the step size is chosen. This can be attributed to the nature of the problem in hand, the minimization of a nonlinear function with its associated local minima. When we used a very small step size, not only the algorithm converged slowly, but also it got trapped in a local minimum. If we used a large step size, the solution would diverge. Thus adaptive selection of the step size was used.

Control of the 2R Robot

The parameters of the 2R robot (8) chosen are give in table 1

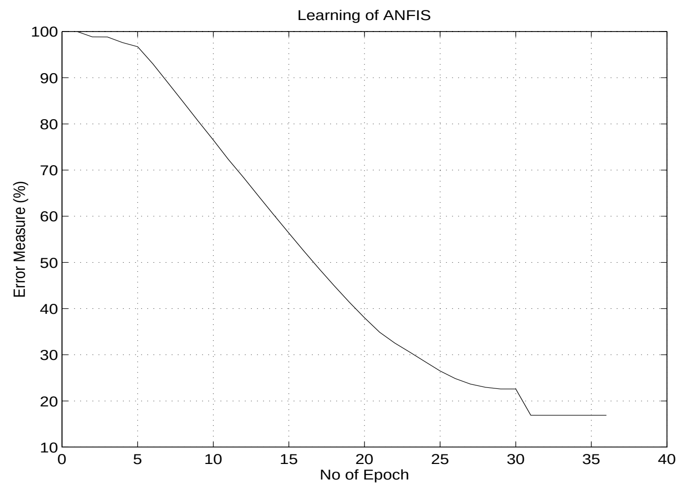

Figure 6. The history of error, after restarting the learning process where the error was minimum in previous case with different step size

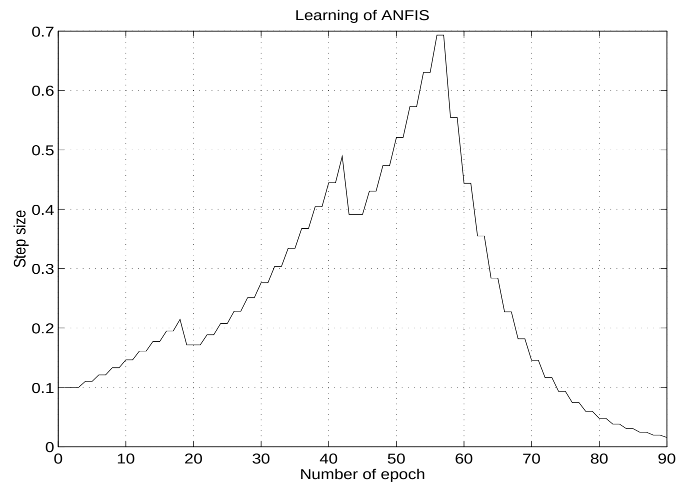

Figure 7. The history of step size during learning process

The simulations were carried out for the following trajectories:

- Step Input:
  - $\theta_{d1} = \theta_{d2} = 0.5$rad
- Sinusoidal:
  - Joint 1. $\theta_{d2}(t) = sin(0.1t + \pi/2)$
  - Joint 2 $\theta_{d2}(t) = sin(0.1t - \pi/2)$
- Cartesian Trajectory:
  - $t = 0$s: $x = 1.45$m; $y = 0$m;
  - $t = 5$s: $x = 1.45$m; $y = 0.5$m;
  - $t = 10$s: $x = 1.45$m; $y = 0$m; The intermediate points were generated using a cubic.

We also performed the simulations for two models.

- **Case 1**: When the reference plant is a bad estimation of the actual plant.
  The parameters of the reference plant (8) are listed in table 2
  It may be noted that, we have chosen purposefully bad estimates of the plant to demonstrate that the control scheme performs acceptably even for fairly "bad" estimates of the plant. For "very good" estimate, there may not be significant improvement over the conventional model based control scheme.
- **Case 2**: Linear Reference Plant. The reference plant model was chosen as linear with decoupled equations.

$$\tau_i = J_i\ddot{\theta}_i + c_i\dot{\theta}_i \quad i = 1, 2 \tag{13}$$

The numerical values chosen for the simulations are $J_1 = J_2 = 10$ and $c_1 = c_2 = 1$.

It may be noted that Case 1 is a bad estimation but the structure of the reference plant is same as that of the plant. However, in Case 2, the structure is also different. In fact, for Case 2, the model based controller is similar to a PD controller with an additional damping indicated by c_1 and c_2, and $\ddot{\theta}_{di}$, is scaled by a factor of J_i (see equation 10).

Some of the results of simulation experiments have been shown in figures 8 -13.

Table 1. Physical parameters of the 2R robot for simulation of Model Reference Learning Control using ANFIS

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
1	0.7	75.15	0.26	36.30
2	0.9	46.15	0.66	31.14

Table 2. Physical parameters of the reference plant

Link	Length (m)	Mass (kg)	C.G. (m)	Inertia (kgm^2)
1	0.7	2.15	0.27	1.30
2	0.9	0.25	0.56	31.64

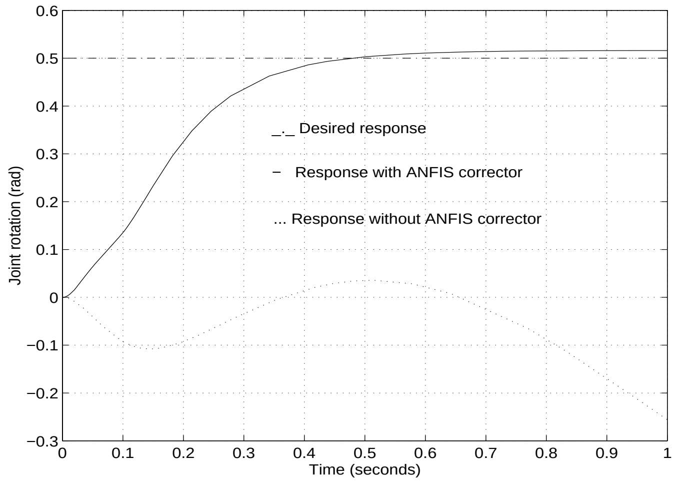

Figure 8. The response of first joint to step input when the reference model is nonlinear (case 1).

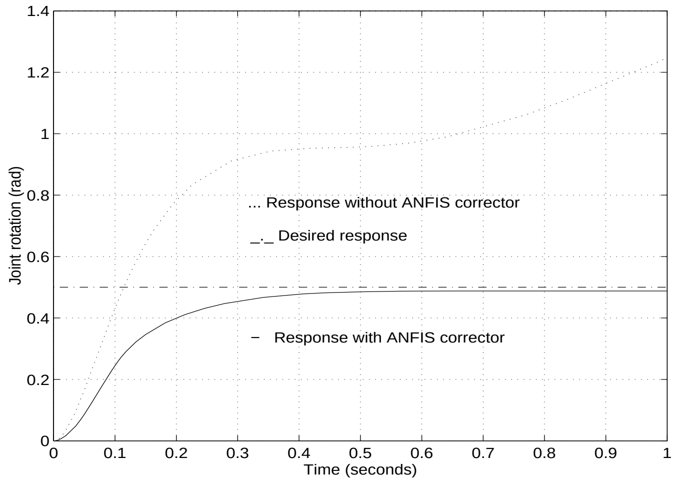

Figure 9. The response of second joint to step input when the reference model is nonlinear (case 1).

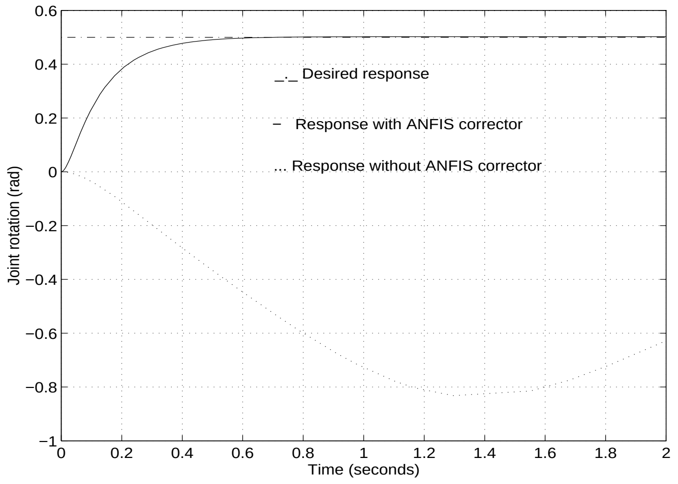

Figure 10. The response of first joint to step input when the reference model is linear (case 2).

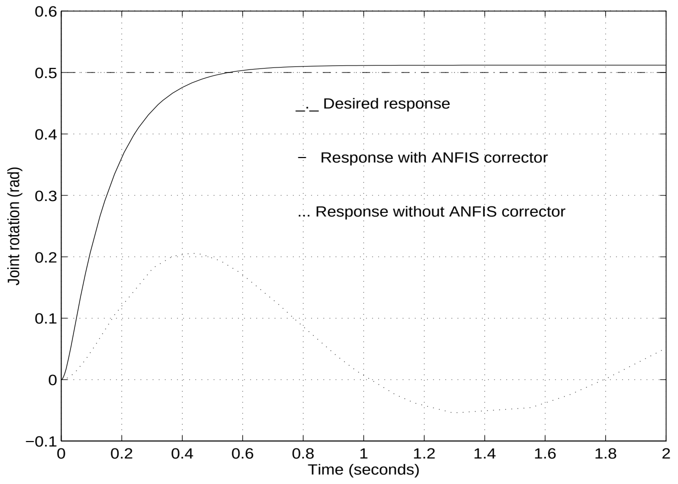

Figure 11. The response of second joint to step input when the reference model is linear (case 2).

CONCLUSION

In this paper, we have presented a model reference learning control using ANFIS. The control scheme has been implemented for a 2R rigid robot with the model containing uncertainty in structure and parameters. Two cases of reference plants were considered. First, the model is a bad estimation of the plant but with same structure as the actual robot, and second, the reference plant is assumed to be linear with decoupled equations. Simulations were carried on for various trajectories such as step, sinusoidal and a cubic Cartesian trajectory. It was found that the controller could track the desired trajectories well even when the reference model is linear.

REFERENCES

J. J. Craig , 1989. *"Introduction to Robotics, Mechanics and Control"*, Addison-Wesley.

Bart Kosko,1994. "Neural Networks and Fuzzy Systems A Dynamical approach To Machine Intelligence" *Prentice-Hall of India.*

A. Lotfi and A. C. Tsoi, "Learning Fuzzy Inference Systems Using an Adaptive Function Scheme", *IEEE Transactions on System Man, and Cybernetics, Vol. 26 No. 2. April 1996.* pp. 326-331

S. Shankar Sastry, "Model-reference adaptive control - stability, Parameter Convergence, and Robustness," IMA Journal of Math. Control and Information, 1984 pp. 27-66

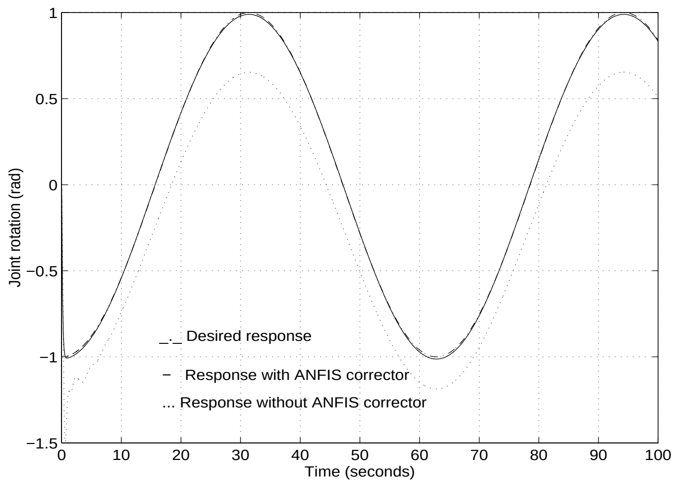

Figure 12. The response of second joint to sinusoidal input when the reference model is linear (case 2).

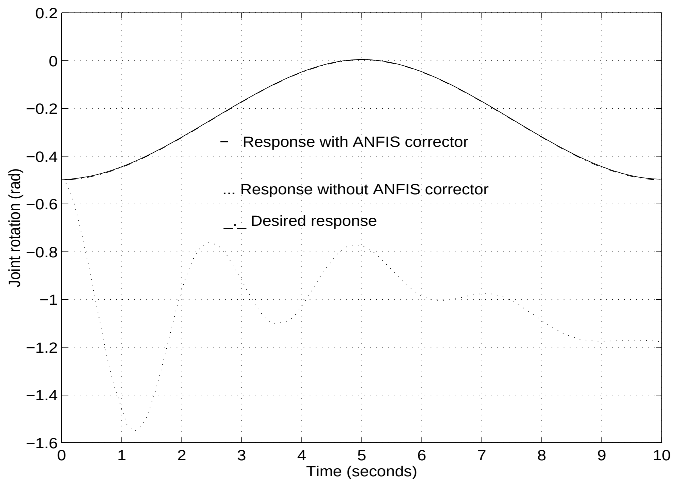

Figure 13. The response of first joint to Cartesian trajectory input when the reference model is linear (case 2).

Gene F. Franklin, J. David Powell, and Michael C. Workman,"Digital Control of Dynamical Systems" II edition, Addison Wesley 1990

Kaul J. Astrom, B. Wittenmark, "Computer Controlled Systems: Theory and Design,". Prentice-Hall, Inc, Englewood Cliffs 1984.

J. J. Craig, "Adaptive Control of Mechanical Manipulators", Ph.D. Thesis. 1986

Jyh-Shing Roger Jang and Chuen-Tsai Sun, 1995. "Neuro-Fuzzy Modeling and Control," *Proceedings of IEEE, March 1995.*

Jyh-Shing Roger Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 23, No. 3, 1993. pp. 665-685

Jyh-Shing Roger Jang and C. T. Sun, "Predicting chaotic time series with fuzzy if-then rules,"*Proc of IEEE international conference on fuzzy systems,* 1993.

Kiego Watanabe, et al., "A Fuzzy-Gaussian Neural Network and Its Application to Mobile Robot Control," *IEEE Trans. on Control Systems technology,* Vol. 4, No. 2, 1996. pp. 193-199

Y-M. Chen, K. F. Gill, "A fuzzy controller design for use with a non-linear system," *Proc. IMechE*, Vol. 210, 1996. pp. 141-150

Tang-Kai Yin and C. S. George Lee, "Fuzzy Model-Reference Adaptive Control," *IEEE Trans. on Systems, Man, and Cybernetics,* Vol. 25, No. 12, Dec. 1995. pp. 1606-1615