Appl. Math. and Comp. Sci., 1997, Vol.7, No.3, 611–637

FUZZY CONTROL OF ROBOTIC MANIPULATORS

MENG J. ER*, NIKOS E. MASTORAKIS**

This paper presents an Intelligent Control Strategy for an n-degree-of-freedom (d.o.f.) robotic manipulator. It covers the design and simulation study of a Fuzzy Controller (FC) for the robotic manipulator with a view of tracking a predetermined trajectory of motion in the joint space. An industrial robotic manipulator, Adept One Robot, was used to evaluate the effectiveness of the proposed scheme. The Adept One Robot was simulated as a three-axis manipulator with the dynamics of the tool (fourth link) neglected and the mass of the load incorporated into the mass of the third link. The overall performance of the control system under various conditions, namely variation in payload, variations in coefficients of static, dynamic and viscous friction and various trajectories were studied and a comparison was made with the Neural Network Controller (NNC) of (Er, 1996) and Computed Torque Controllers 1 and 2 of (Craig, 1989) which are designed assuming full knowledge and partial knowledge of the robot dynamics, respectively. The FC was shown to be robust and able to overcome the drawbacks of the NNC and the two computed torque controllers.

1. Introduction

The most common controller used in present robotic control is the Proportional Derivative (PD) controller as it does not require any detailed knowledge of the manipulator inertia tensor, the Coriolis and centrifugal coupling forces and coefficients of friction. To enhance accuracy, computed torque and resolved acceleration control methods are employed. However, these two methods are model-based control schemes and the robot parameters must be known accurately. The computed torque method is an approach that makes direct use of the complete dynamic model of the manipulator to cancel not only the effects of gravity, but also the manipulator inertia tensor, the Coriolis and centrifugal forces, and coefficients of friction. For such schemes, we need to know the dynamic model of the manipulator. However, due to the complexity and nonlinearity of the dynamics, it is difficult to estimate the parameters of the robot manipulator such as link lengths, link masses, centres of gravity, coefficients of viscous friction, etc.

In oder to overcome these difficulties, considerable research on intelligent control systems with human-like inference and adaptation abilities has been conducted. Basically, there are two representative approaches to design intelligent control systems.

* School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, e-mail: emjer@ntu.edu.sg.

** Hellenic Naval Academy, Terma Hatzikyriakou, 18539, Piraeus, Greece, e-mail: mastor@softlab.ntua.gr.

One approach is to use neural networks which have distinct learning and adaptation capabilities. The other appproach is to apply fuzzy logic control theory which can emulate human thinking.

The neural network controller with a nonlinear multi-layer network and adaptable weights can be considered as a special form of the adaptive control scheme. In the neural network, knowledge acquisition can be automatically accomplished by learning the input-output relation. The neural network control is distinguished by high parallelism, fault-tolerance, adaptive and learning abilities. The neural-network-based approach is independent of the precise modelling of the robot dynamics due to its learning ability which enables it to learn the inverse dynamics of the robot arm. In this way, inaccurate modelling estimation may be avoided (Atkeson and Reinkensmeyer, 1988; Atkeson *et al.*, 1985; Bhat *et al.*, 1990; Chen, 1990; Er, 1996; Goldberg and Pearlmutter, 1988; Guez and Ahmad, 1988; Horne and Jamshidi, 1988; Karakasoglu *et al.*, 1993; Kawato *et al.*, 1987; 1988a; 1988b; Khosla and Kanade, 1985; Kung and Hwang, 1989; Kuperstein and Wang, 1990; Miller *et al.*, 1990; Miyamoto *et al.*, 1988; Narendra and Parthasarathy, 1989; 1990; Nyugen and Widrow, 1990; Psaltis *et al.*, 1988; Setoyama *et al.*, 1987).

In fuzzy control, a precise mathematical model of the plant may not be required. Also, the Fuzzy Controller (FC) possesses strengths of logic control, linguistic control, parallelism relaxation, flexibility and robustness. The theory of fuzzy sets as an extension of the traditional set theory was introduced in (Zadeh, 1965; 1973), together with fuzzy logic to manipulate fuzzy sets. A fuzzy set allows for degrees of membership to a set. A membership function defines the grade of membership in a fuzzy set for all possible members and is typically expressed as a mathematical function or a set of discrete digital numbers. This representation allows human observations, expressions and expert knowledge to be more precisely modelled.

Recently, there has been great resurgence of interest in combining neural networks and fuzzy logic in robotic control problems. A number of interesting results regarding using neural networks and fuzzy logic to create intelligent systems have been reported (Hasegawa *et al.*, 1993; Horikawa *et al.*, 1992; Mizukami and Fuke, 1991; Psaltis *et al.*, 1988; Watanabe *et al.*, 1996). Horikawa *et al.* derived a Fuzzy Neural Network (FNN) that realizes fuzzy reasoning within a multi-layered hierarchical neural network with sigmoidal functions as unit functions. The control rules can be identified by the data gathered from the plant which is manipulated by an expert and the parameters associated with the antecedence and consequence can also be fine tuned. The application of this method can also be found in (Hasegawa *et al.*, 1993). Note, however, that the method of (Horikawa *et al.*, 1992) cannot be applied to the case when there are no pattern control data because the generalized learning architecture of (Psaltis *et al.*, 1988) is utilized to train the neural network. Moreover, their method requires intermediate layers to generate the membership function in the antecedence because a pseudo-trapezoidal membership function is constructed by summing two sigmoid unit functions with different signs. Hence, the method is called a Fuzzy Sigmoidal Neural Network (FSNN). This fact also causes the number of units in the corresponding intermediate layer to grow as the number of fuzzy labels increases. The fuzzy arctangential neural network (FANN) studied in (Mizukami and

Fuke, 1991), in which an arctangential function is used as a unit function, suffers from the same problem. In (Watanabe *et al.*, 1996), a Fuzzy-Gaussian Neural Network (FGNN) controller that uses a Gaussian function as a unit function, which can solve the above problem, is described. A specialized learning architecture is used so that the membership function can be tuned without using an expert's manipulated data. Tracking control for the speed and azimuth of a mobile robot driven by two independent wheels using the FGNN controller is demonstrated. The effectiveness of the proposed method is illustrated by performing simulation studies of a circular and square trajectory tracking control. Following a different approach, Kim *et al.* propose an adaptive fuzzy control scheme. The proposed fuzzy control system, which consists of the Fuzzy-Neural Controller (FNC) and a Model Neural Network (MNN), has two important characteristics of adaptation and learning. In the FNC, the antecedence and consequence of a fuzzy rule are constructed by a clustering method and a multi-layer neural network. In the MNN, a multi-layer neural network is utilized to identify the unknown dynamics of the manipulator. The effectiveness of the proposed scheme was demonstrated by computer simulations of a cart-pole and a two-degree-of-freedom (d.o.f.) robotic manipulator.

In this paper, another approach to using neural networks and fuzzy logic is adopted to control an n-d.o.f. robotic manipulator. Our approach is different from the other approaches in that the design is carried out in two stages. First, the Neural Network Controller (NNC) of (Er, 1996) is used to control a robotic manipulator. Next, based on the performance of the NNC, an FC which incorporates experts' experience into fuzzy rules is designed. A comparison is then made with the NNC of (Er, 1996) and the Computed Torque Controllers 1 and 2 of (Craig, 1989) which are designed assuming full knowledge and partial knowledge of the robot dynamics, respectively. Simulation studies were carried out to evaluate the effectiveness and robustness of the proposed controller. They show that the proposed method outperforms the NNC and the two computed torque controllers.

2. Review of Neural Network Controllers

In this section, the operation of the NNC of (Er, 1996) is reviewed. There are basically two stages in controlling the Adept One Robot. The first stage consists in off-line training or pre-training, where the neural network learns the inverse dynamic model of the robot from a set of training data. The second stage is the control phase which involves real-time control and on-line training of the neural-network-based controller.

2.1. Off-Line Training

In off-line training, an inverse dynamic model of the robot is determined. This is depicted in Fig. 1. The manipulator receives the desired input torque, $\tau_d(t)$, and outputs the resulting trajectory, $\theta(t)$. The inverse dynamic model (neural network) is set in an opposite fashion to that of the manipulator, i.e. the input and output of the model are the output and input of the robot, respectively. The error signal, $e(t)$, which is the difference between the desired and estimated torques, is then minimized

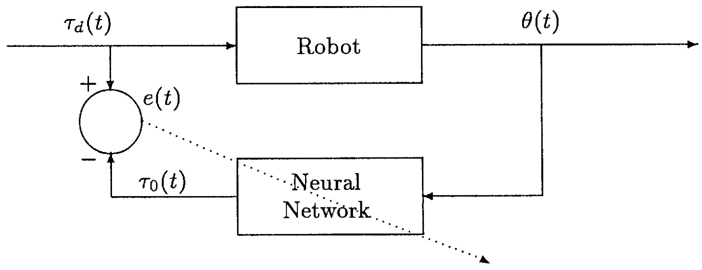


Fig. 1. Direct inverse modelling of a robotic manipulator.

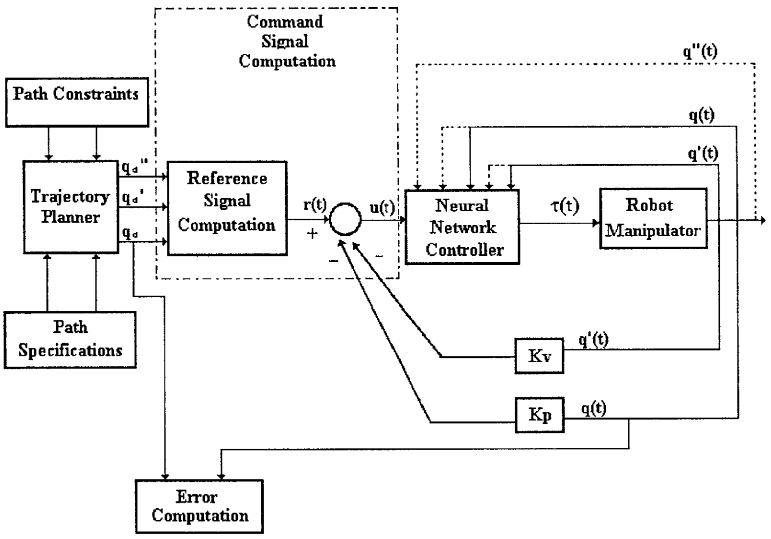


Fig. 2. Neural network controller.

to train the network. In (Er, 1996), back-propagation with variations in its weight adjustments is used as the training algorithm.

The inverse dynamics model of the robotic manipulator is modelled by two neural networks, namely NNC12 and NNC3 which denote neural network controllers for Links 1 and 2, and Link 3, respectively. The reason why two separate neural network structures are used is that the inputs of NNC12 and NNC3 are different. Note that the structure uses a dummy input of -1 and a bias neuron of -1 in the hidden layer. This is to augment the input vector and the hidden layer by a fixed bias component.

2.2. On-Line Control

Figure 2 shows the block diagram of the NNC of (Er, 1996) incorporating a PD control strategy. Solid lines indicate signal flows when the neural network is executing the control action while the dotted lines represent the information flow during on-line training. As mentioned before, the NNC has two modes of operation, namely the control mode and the on-line training mode.

In the sequel, real-time operation of the NNC is described. As depicted in Fig. 2, a pre-defined trajectory is first fed into the system as the desired motion. It then goes through the reference signal computation block and the PD architecture. The reference signal $r(t)$ is generated by the computed torque algorithm given by

$$r(t) = q_d''(t) + K_v q_d'(t) + K_p q_d(t) \tag{1}$$

where q_d, q_d' and q_d'' denote the desired trajectory, angular velocity and angular acceleration, respectively. The command signal $u(t)$ is generated as follows:

$$u(t) = r(t) - K_v q'(t) - K_p q(t) \tag{2}$$

where $q(t)$ and $q'(t)$ denote the actual trajectory and angular velocity, respectively. The command signal is approximately equal to the acceleration generated by the desired trajectory in the case where the robot dynamics are perfectly modelled. This command signal together with the robot variables q_0 and q_0' generate the control mode operation, producing an output torque $c\tau_1$ which, in turn, generates a new set of robot variables, q_1, q_1' and q_1''. However, as mentioned before, the on-line learning mode runs concurrently with the control mode. At the time moment when the command signal is computed by the neural network to generate the output torque $c\tau_1$, the network also learns and generates a new learning torque τ_1 based on the previous robot variables q_0, q_0' and q_0''.

At the next time instant, the control signal will generate an output torque $c\tau_2$ and the learning signal will generate a learning torque τ_2. Note that the learning torque τ_2 is actually generated based on the new robot variables at the previous instant, i.e. q_1, q_1', q_1'' which are derived from an input torque $c\tau_1$ of the robot. This makes $c\tau_1$ the desired torque and τ_2 the actual torque due to the neural network giving an error $e_1 = c\tau_1 - \tau_2$. This error signal is then fed back into the neural network to update the connection weights at the following instant and the process repeats in the same fashion.

It is important to note that the error signal is found only at the end of the second instant, i.e. after the desired and actual torques have been computed. Weight updating, therefore, only starts at the third instant. This is illustrated in Table 1.

Table 1. Time history of torque computation and weight update.

Time t (i ms)	Desired torque (due to command signal)	Actual torque (due to learning signal)	Computed error $e = c\tau_{i-1} - \tau_i$
1 ms	$c\tau_1$	τ_1	$-\tau_1$
2 ms	$c\tau_2$	τ_2	$c\tau_1 - \tau_2$
3 ms	$c\tau_3$	τ_3	$c\tau_2 - \tau_3$
⋮	⋮	⋮	⋮

Weight updating starts at 3 ms ↓

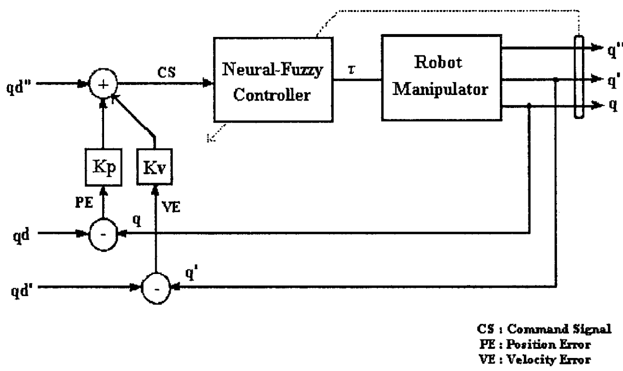


Fig. 3. Overall block diagram of robotic manipulator control system.

3. Design of a Fuzzy Controller

The general block diagram of the overall control system of the robotic manipulator is shown in Fig. 3. The chosen inputs are the command signal (c), position error (p) and velocity error (v). The output is the control torque (τ) for all the three joints. The data for the three crisp inputs c, p and v are extracted by simulating the NNC of (Er, 1996) without any change in parameters from 0 to 1.5 s in steps of 1 ms. These data are supposed to be accurate since the neural network is well-trained. It is then arranged in ascending torque values for the fuzzification stage. The range of values for c, p, v and τ for each joint can be estimated and are tabulated in Table 2.

Table 2. Universe of discourse.

Joint #	Joint 1		Joint 2		Joint 3	
Fuzzy I/0	Lower limit	Upper limit	Lower limit	Upper limit	Lower limit	Upper limit
c	−5.6829	6.0562	−5.3052	7.0146	−0.8573	0.82
p	−3.01E−04	1.29E−03	−1.63E−03	2.23E−03	−1.25E−04	0.864E−04
v	−0.0132	0.0268	−0.0584	0.0817	−0.918E−03	1.7E−03
τ	−45.326	48.7527	−7.88081	6.0706	−32.2	−27.0

3.1. Input Membership

The number of input Membership Functions (MF's) is now determined. Using Joint 1 as an illustration, the input MF's to represent the three fuzzy inputs (c, p and v) are triangular with 1/3 overlapping. As shown in Fig. 4, the two extreme ends are represented by a flat response to indicate that the output is saturated at those points. As for Joints 2 and 3, the shapes are similar except for the Universe of Discourse. For such an MF, each input value has a possibility to fall into at least one and at most two different labels. The overlap ratio r_a and the overlap robustness r_b are calculated as follows:

$$r_a = \frac{a}{5a} = 0.2$$

$$r_b = \frac{\text{area of overlap region}}{\text{total area}} = \frac{\int_L^U (\mu_{NS} \oplus \mu_{PS})}{2(U-L)} = \frac{(1/3 + 1/3)(U-L)}{2(U-L)}$$
$$= 0.3333$$

Since the safety regions are $0.2 \leq r_a \leq 0.6$ and $0.3 \leq r_b \leq 0.7$, the system is regarded to be relatively stable. Note that the inputs will have fuzzy values known as grades. The total grades from one or two labels for each input should be less than 1 since the overlap ratio is only 1/3. For $r_a = 0.5$, the grades for each input values will always sum up to 1.

3.2. Output Membership

Singleton MF's are used to represent the fuzzy output as shown in Fig. 5. This is because a singleton is easier to implement in software as it has a simple mathematical representation. The numerical value of the crisp output τ can be computed as follows:

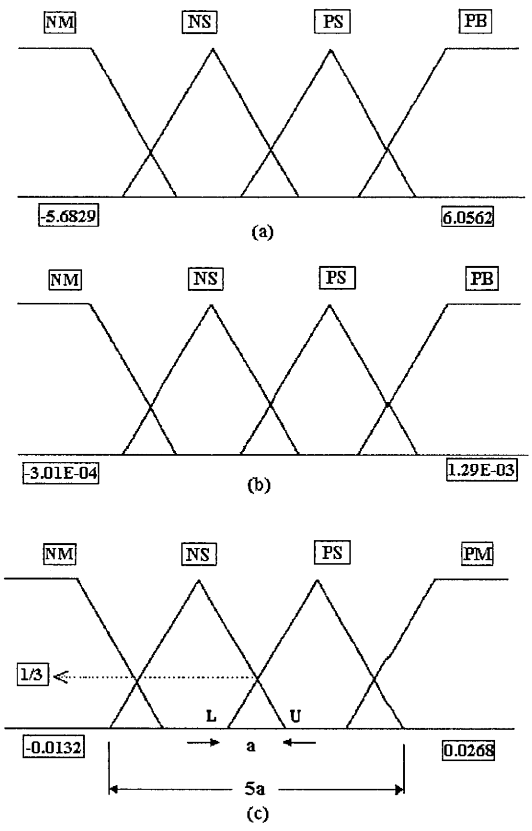
$$\tau = \frac{\sum_i (\text{fuzzy output})_i * (\text{singleton position on x-axis})}{\sum_i (\text{fuzzy output})_i} \tag{3}$$

For the example shown in Fig. 6, the numerical value of the crisp output is given by

$$\tau = \frac{(0.8)(-45) + (0.14)(-26.4) + (0.06)(-7.8)}{0.8 + 0.14 + 0.06} = -40.16\,\text{Nm} \tag{4}$$

3.3. Fuzzy Rules

The fuzzy rules are based on observations made about the neural network data. Consider the data marked for Joint 1 in Table 3. Using the membership functions of Fig. 4, the c value is found to fall into the NM label, the p value into the NM and NS labels and v into the NM and NS labels. As for the output torque, the value falls in both the NB and NM labels with reference to Fig. 5. Thus, it will be represented in the particular positions of the Fuzzy Associate Memory (FAM) as shown in Table 4. The corresponding FAM for the other two joints can also be represented using their respective set of data as shown in Tables 5 and 6. To facilitate software implementation, each label is represented by numbers as illustrated in Table 7.



(a) Command signal input
(b) Position error input
(c) Velocity error input

Fig. 4. Membership functions of Joint 1.

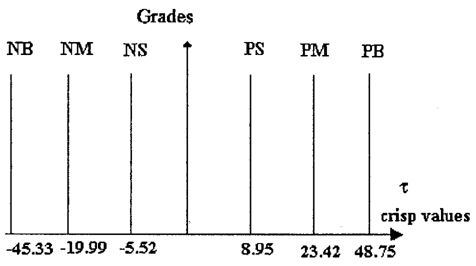


Fig. 5. Singleton MF for output torque of Joint 1.

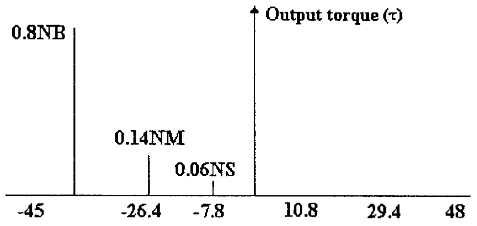


Fig. 6. Clipped output MF.

Table 3. Sample data extracted from Joint 1.

Time (s)	Torque (Nm)	Position error (rad)	Velocity error	Command signal
1.45	−30.7430	−9.80E−05	5.68E−03	−3.5790
1.29	−30.4725	9.00E−06	−2.84E−03	−3.7870
1.28	−27.4266	3.41E−05	−2.16E−03	−3.3790
1.46	−26.1632	−4.12E−05	5.69E−03	−2.9935

Using the set of data indicated in Table 3, the rules listed below were designed:

1. If c is NM and p is NM and v is NM, then τ is NB.
2. If c is NM and p is NM and v is NS, then τ is NM and NB.
3. If c is NM and p is NS and v is NM, then τ is NB.
4. If c is NM and p is NM and v is NS, then τ is NM and NB.

The max-min inference method is employed in the software implementation. Having obtained grades for the labels of each rule, the rules generated are of the following form:

1. If c is 0.6 NM and p is 0.4 NM and v is 0.35 NM, then τ is NB.
2. If c is 0.6 NM and p is 0.4 NM and v is 0.2 NS, then τ is NM and NB.
3. If c is 0.6 NM and p is 0.4 NS and v is 0.1 NM, then τ is NB.
4. If c is 0.6 NM and p is 0.4 NM and v is 0.2 NS, then τ is NM and NB.

The minimum, which is known as the rule strength, is MIN(c,p,v) for each particular rule and is given as:

$$\begin{array}{ll}
\text{MIN}[c,p,v] = 0.35\text{ NB} & \text{Rule 1}\\
\text{MIN}[c,p,v] = 0.20\text{ NM, NB} & \text{Rule 2}\\
\text{MIN}[c,p,v] = 0.10\text{ NB} & \text{Rule 3}\\
\text{MIN}[c,p,v] = 0.20\text{ NM, NB} & \text{Rule 4}
\end{array}$$

Table 4. FAM for Joint 1.

CS		Position Error			
NM 1		NM 1	NS 2	PS 3	PM 4
Velocity	NM 1	NB	NB NM	NB NS NM	NB NM
	NS 2	NB	NB	NB NS NM	NB NS NM
Error	PS 3	NB	NM	NB NS NM	NB NS NM
	PM 4	NB	NB	NB NS NM	NB NS NM

CS		Position Error			
NS 2		NM 1	NS 2	PS 3	PM 4
Velocity	NM 1	NB NM	NB NS NM PS	NB NS NM PS	NB NS NM PS
	NS 2	NB	NS NM PS	NS NM PS	NS NM PS
Error	PS 3	NB	NS NM	NS NM	NS NM PS
	PM 4	NB	NS NM	NS NM	NS NM

CS		Positon Error			
PS 3		NM 1	NS 2	PS 3	PM 4
Velocity	NM 1	NS PS	PM PS PB	PM PS PB	PB
	NS 2	NS PM PS	NS PM PS	PM PS	PM PS
Error	PS 3	PS	NS PS	PM PS	PM PB
	PM 4	PS	PM PS	PM PS	PM PB

CS		Positon Error			
PM 4		NM 1	NS 2	PS 3	PM 4
Velocity	NM 1	PM PB	PB	PB	PB
	NS 2	PB	PB	PB	PB
Error	PS 3	PB	PB	PB	PB
	PM 4	PB	PB	PB	PB

Table 5. FAM for Joint 2.

CS		Position Error			
NM 1		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	NB	NB	PM	PM
	NS 2	NB NM	NB NM	PB	PB
	PS 3	NB NM	NB	PM	PB
	PM 4	NB NM	NB	PM	PM

CS		Position Error			
NS 2		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	NB	NB	NS NM PS	NS NM
	NS 2	NB	PM	PM	PB
	PS 3	NB	PM	PM	PB
	PM 4	NB NS NM PS	PM	PM	PB

CS		Positon Error			
PS 3		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	NB	NB	PB	PB
	NS 2	NB	PM	PM	PB
	PS 3	NB	PB	PM	PB
	PM 4	PB	PB	PM	PB

CS		Positon Error			
PM 4		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	PB	PB	PB	PB
	NS 2	PB	PB	PB	PB
	PS 3	PB	PB	PB	PB
	PM 4	PB	PB	PB	PB

Table 6. FAM for Joint 3.

CS		Position Error			
NM 1		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	NB	NB	NB	NB
	NS 2	NB	NB	NB	NM
	PS 3	NB	NB	NB	NM
	PM 4	NB	NB NM	NS NM	NS NM

CS		Position Error			
NS 2		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	NB NM	NM	NS PS	PS
	NS 2	NS NM	NM	PS	PM PB
	PS 3	NS NM PS	NM	NS	PM
	PM 4	NS NM PS	NM	NS	PM PB

CS		Positon Error			
PS 3		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	NS PS	PM PS PB	PM PS PB	PB
	NS 2	NS PM PS	NS PM PS	PM PS	PM PS
	PS 3	PS	NS PS	PM PS	PM PB
	PM 4	PS	PM PS	PM PS	PM PB

CS		Positon Error			
PM 4		NM 1	NS 2	PS 3	PM 4
Velocity Error	NM 1	PM PS PB	PM PS PB	PM PB	PB
	NS 2	PM PS	PM PS	PB	PB
	PS 3	PM PS	PM PS	PM PS	PB
	PM 4	PM PS PB	PM PB	PB	PB

Table 7. Representation of labels.

Label	Input	Output
NB	–	1
NM	1	2
NS	2	3
PS	3	4
PM	4	5
PB	–	6

The maximum of each output label is evaluated as the maximum of the four rule strengths:

$$\text{MAX[NB,NM,NS,PS,PM,PB]} = [0.35,0.2,0,0,0,0]$$

This corresponds to the torques of 0.35 NB and 0.2 NM. The actual torque value is computed by defuzzification using the singleton method as discussed before.

3.4. Fine Tuning of Rules

After implementing FAM in software and after completion of defuzzification, simulation studies were carried out to observe whether the robot is able to track the desired path. Necessary corrections in the rules were made to offset the large initial error. The final rules are shown in Tables 4–6.

4. Simulation Studies

In this paper, the Adept One robot shown in Fig. 7 is chosen to evaluate the performance of the two control schemes. It is a SCARA (Selective Compliance Assembly Robot Arm) configuration with a nearly full-circle work space of radius 31.5 in and can carry a maximum payload of 13.2 lbs, moving it 1 in. up, 12 in. over, 1 in. down and back to its starting position in 1.3 s. It is unique in that it is the first commercial robot to implement a direct drive system for actuation. No gears or other mechanical power conversion devices are used. Instead, high-torque low-speed brushless DC motors are used to drive the joints directly. This eliminates gear friction and backlash and allows for clean, precise and high-speed operation.

The Adept One robot is a four-axis horizontal-jointed robot. Its link-coordinate system is shown in Fig. 8. The vector of joint variables is $q = [\theta_1\ \theta_2\ d_3\ \theta_4]^T$. The first two joint variables θ_1 and θ_2 are revolute variables which establish the horizontal component of the total position p. The third joint variable d_3 is a prismatic variable which determines the vertical component of p. Finally, the last joint variable θ_4 is a revolute variable which controls the tool orientation R. Using the D-H algorithm (Schilling, 1990), the link coordinate diagram is constructed and simplified kinematic parameters are listed in Table 8.

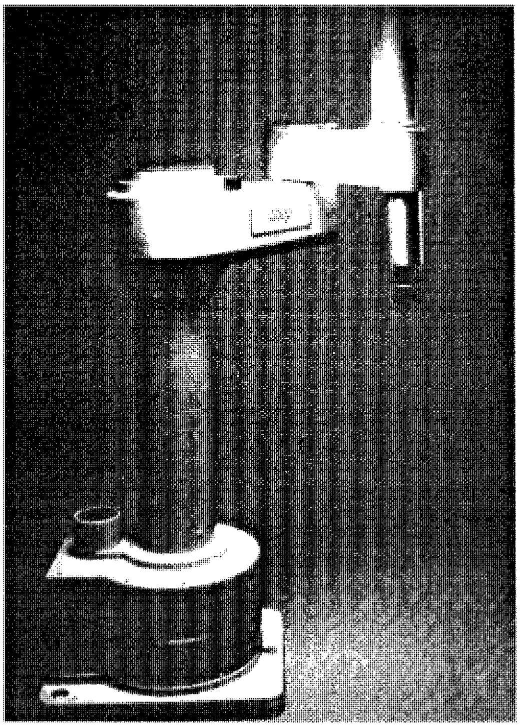


Fig. 7. Adept One robot.

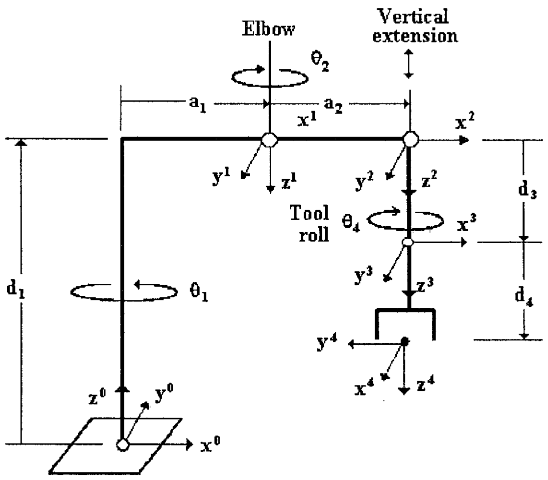


Fig. 8. Link coordinate diagram of Adept One robot.

Table 8. Kinematic parameters of the Adept One robot.

Axis	θ	d	a	α	Home
1	q_1	d_1	a_1	π	0
2	q_2	0	a_2	0	0
3	0	q_3	0	0	100
4	q_4	d_4	0	0	$\pi/2$

The first three axes of the robot position the tool-tip while the fourth axis orients the tool through a roll motion. To keep the dynamic model relatively simple, it is assumed that the mass of the fourth link and of any tool attached to it are sufficiently small in comparison with the masses of the other links and they can be ignored. This is a reasonable assumption as the links of the manipulator tend to become smaller and less massive as it proceeds from the base joint to the tool.

By using the Langrange-Euler algorithm (Schilling, 1990), a dynamic model of the three joints can be derived:

$$\begin{aligned} T_1 &= \left[\left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2\cos(\theta_2) + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{\theta}_1 \\ &\quad - \left[\left(\frac{m_2}{2} + m_3\right)a_1a_2\cos(\theta_2) + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{\theta}_2 \\ &\quad - a_1a_2\sin(\theta_2)\left[(m_2 + 2m_3)\dot{\theta}_1\dot{\theta}_2 - \left(\frac{m_2}{2} + m_3\right)\dot{\theta}_2{}^2\right] + b_1(\dot{\theta}_1) \end{aligned} \tag{5}$$

$$\begin{aligned} T_2 &= -\left[\left(\frac{m_2}{2} + m_3\right)a_1a_2\cos(\theta_2) + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{\theta}_1 \\ &\quad + \left(\frac{m_2}{3} + m_3\right)a_2^2\ddot{\theta}_2 + \left(\frac{m_2}{2} + m_3\right)a_1a_2\sin(\theta_2)\dot{\theta}_1^2 + b_2(\dot{\theta}_2) \end{aligned} \tag{6}$$

$$T_3 = m_3\ddot{d}_3 - gm_3 + b_3(\dot{d}_3) \tag{7}$$

where

$$b_1(\dot{\theta}_1) = b_1^v\dot{\theta}_1 + \operatorname{sgn}(\dot{\theta}_1)\left[b_1^d + (b_1^s - d_1^d)\exp\left(\frac{-|\dot{\theta}_1|}{\epsilon}\right)\right] \tag{8}$$

$$b_2(\dot{\theta}_2) = b_2^v\dot{\theta}_2 + \operatorname{sgn}(\dot{\theta}_2)\left[b_2^d + (b_2^s - d_2^d)\exp\left(\frac{-|\dot{\theta}_2|}{\epsilon}\right)\right] \tag{9}$$

$$b_3(\dot{d}_3) = b_3^v\dot{d}_3 + \operatorname{sgn}(\dot{d}_3)\left[b_3^d + (b_3^s - d_3^d)\exp\left(\frac{-|\dot{d}_3|}{\epsilon}\right)\right] \tag{10}$$

T_1, T_2 and T_3 being the control torques at Joints 1, 2 and 3, respectively. The terms b_i^v, b_i^d, b_i^s, ϵ and g denote the coefficient of viscous friction, coefficient of dynamic friction, coefficient of static friction, friction parameter and constant of gravity ($= 9.8\,\mathrm{ms}^{-2}$), respectively, and a_1, a_2 and d_3 are as indicated in Fig. 8.

Simulation studies were carried out for all control schemes. The manipulator was commanded to move from the initial position $q_{\text{initial}} = [0^\circ \;\; 0^\circ \;\; 0\,\mathrm{m}]^T$ to the final position $q_{\text{final}} = [60^\circ \;\; 45^\circ \;\; 0.15\,\mathrm{m}]^T$. The sampling time was chosen to be 1 ms. The design specifications are that the final position errors for the first, second and third joint must be less than 52.3E-3 rad, 39.27E-3 rad and 7.5 mm, respectively. These amount to less than 5% of the desired destinations.

The simulations were carried out for the following cases:

1. An ideal situation where there are no payload and no variations in the coefficients of static, dynamic and viscous friction.
2. The payload was increased from 0 kg to 3 kg which is approximately half of the maximum load the Adept One Robot can carry while the other parameters were kept constant at zero.
3. The coefficient of friction was increased from 0 to a certain value for a particular joint while the other two coefficients of friction and the payload were kept at zero. The payload mass as well as all the coefficients of friction for the two other joints were maintained at zero.
4. Three trajectories, Paths A, B and C which represent the original trajectory, the trajectory for which the speed of the robotic arm is reduced by half and the trajectory for which the speed of the robotic arm is doubled were created to test the robustness of the FC. The FC was also subjected to a sine function trajectory to test whether the trajectory is invariant.
5. The controllers were subjected to combined variations of the payload and coefficients of static, dynamic and viscous friction.

5. Simulation Results

For brevity, only simulation results of Joint 1 for Cases 1, 2 and 4 will be presented.

5.1. Ideal Situation

Simulation results for the Computed Torque Controller 1 (CTC1), Computed Torque Controller 2 (CTC2), NNC and FC are shown in Fig. 9. The end-point and total errors for the four controllers are listed in Table 9.

Table 9. End-point and total tracking errors of Adept One robot in ideal situation.

Joint	Computed-Torque Controller 1		Computed-Torque Controller 2	
	End-point error	Total error	End-point error	Total error
1	8.34465E−05	7.87323E−02	−1.28837E−02	6.58893
2	1.66893E−05	7.16087E−02	−5.43880E−03	7.28812
3	1.28001E−05	3.09677E−02	−5.18996E−03	6.87389
Joint	**Neural-network controller**		**Fuzzy controller**	
	End-point error	**Total error**	**End-point error**	**Total error**
1	−2.96593E−04	1.18422	1.2815E−04	0.488114
2	−1.63269E−03	3.19699	1.99974E−04	0.507885
3	−3.27677E−05	4.8721E−02	−4.88758E−06	3.53905E−02

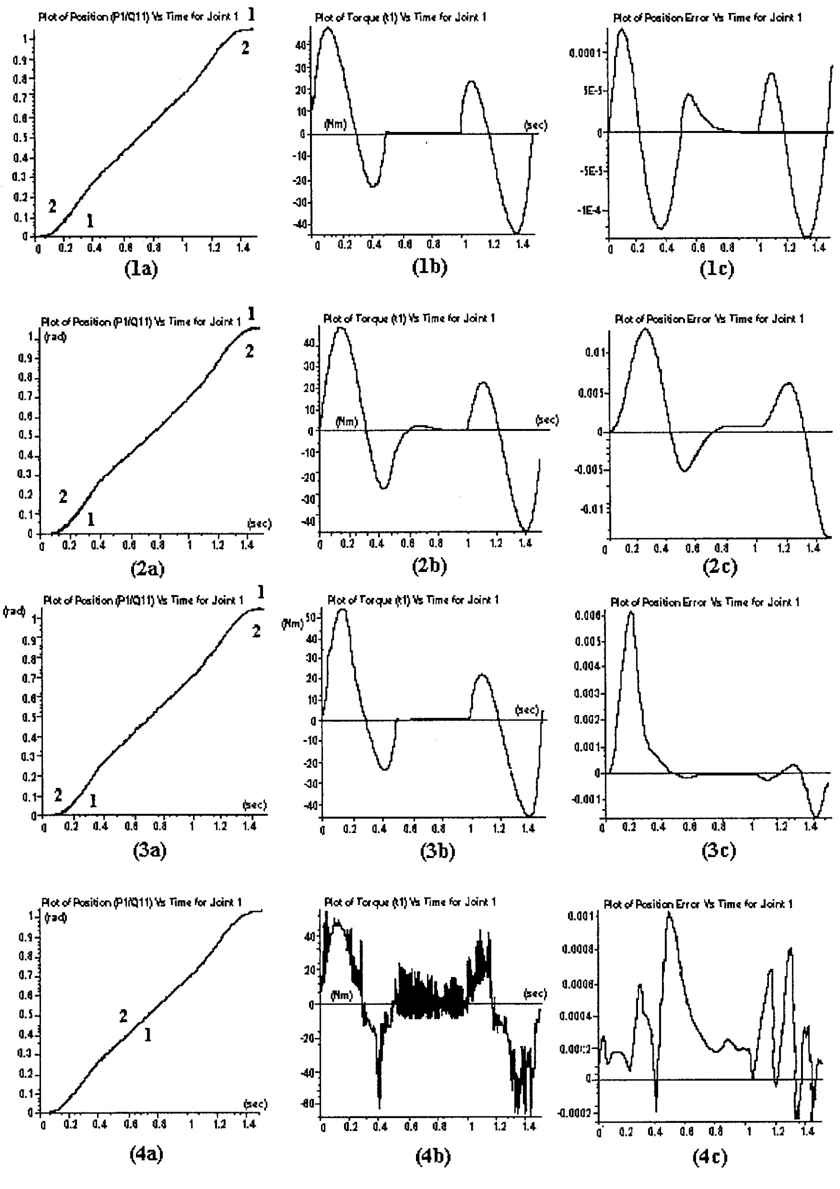
The following observations can be made about the simulation results:

1. The results are in general satisfactory. However, the CTC2 which is designed with partial knowledge of the parameters of the robot manipulator does not yield as accurate results as the other controllers. The controller exhibited no convergence to the desired trajectory for the third joint. The CTC1 is designed using the inverse dynamics of the robot manipulator with exact parameters. Thus, it is expected to achieve the most accurate result when compared with the other controllers in terms of both end-point and total tracking errors.
2. Generally speaking, the FC performed well except for its control torque which oscillated at a high frequency of 1 kHz. This is not acceptable in practice because the motors could be overloaded and the manipulator would oscillate. The high frequency oscillations of the control torque can be eliminated by using MAX-PROD in place of MAX-MIN in the fuzzy rules. The improvement is illustrated in Fig. 10.

3 Since the FC is designed using the data obtained from the NNC, it is expected that the shape of the control torque is quite identical to that of the latter.

5.2. Variations of the Payload Mass

The payload mass was increased from 0 to 3 kg while the coefficients of static, dynamic and viscous friction were kept at zero. The simulation results for the computed-torque controllers, NNC and FC are shown in Fig. 11. The end-point errors and total errors for the four controllers for Joint 1 are listed in Table 10.



(1) Computed torque controller 1
(2) Computed torque controller 2
(3) Neural-network controller
(4) Fuzzy controller

Legend : 1 - Actual
2 - Desired

Fig. 9. Plot of (a) actual and desired trajectory, (b) control torque, and (c) position error of Joint 1.

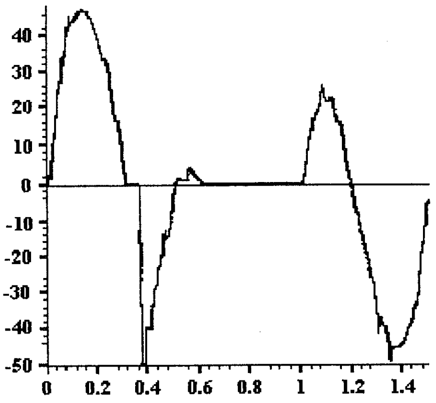


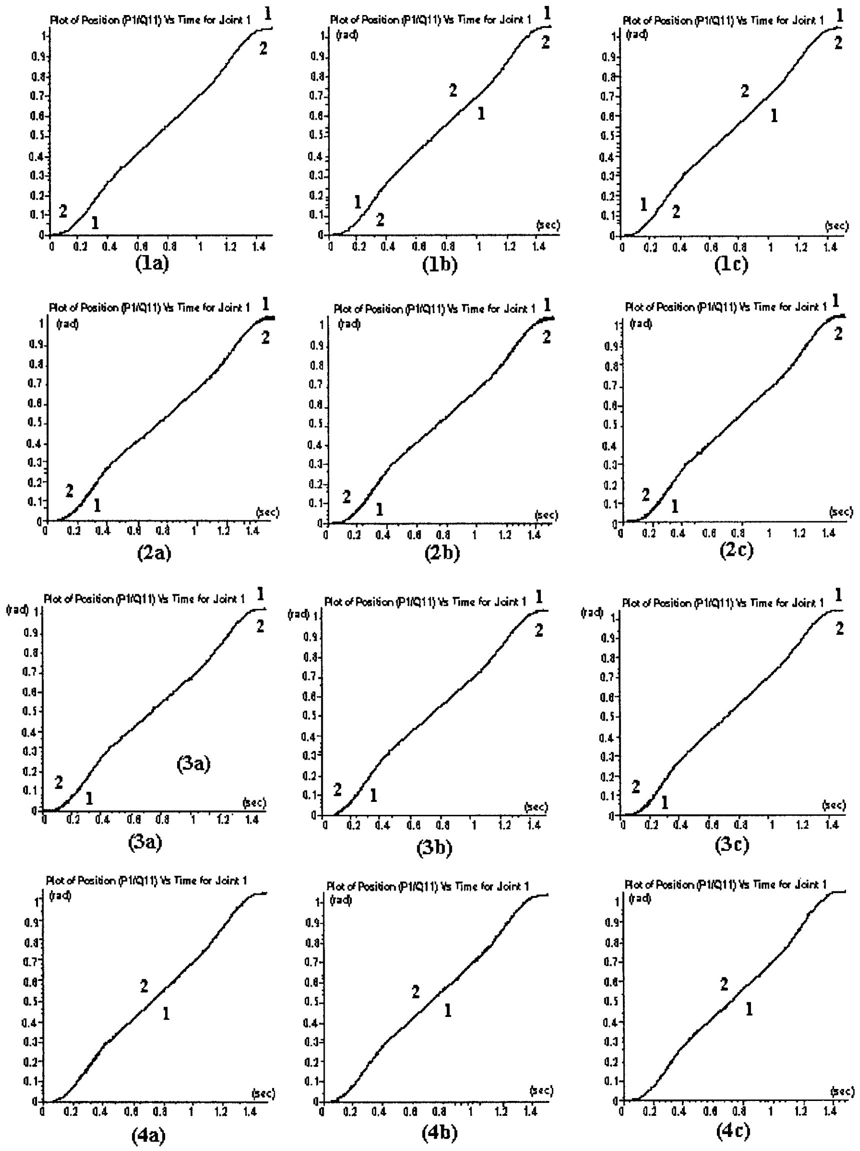
Fig. 10. Control torque obtained using MAX-PROD.

Table 10. End-point and total tracking errors of Adept One robot in payload variation for Joint 1.

Load (Kg)	Computed-Torque Controller 1		Computed-Torque Controller 2	
	End-point error	Total error	End-point error	Total error
0.0 *	8.34465E−05	7.87323E−02	−1.28837E−02	6.58893
0.5 *	−1.56164E−04	0.176075	−1.34917E−02	6.54295
1.0 *	−3.92914E−04	0.318514	−1.40797E−02	6.49585
3.0	−1.31643E−03	0.929688	−1.63026E−02	6.42230
Load (Kg)	**Neural-network controller**		**Fuzzy controller**	
	End-point error	Total error	End-point error	Total error
0.0 *	−2.96593E−04	1.18422	1.28150E−04	0.488114
0.5 *	−6.41108E−04	1.83899	4.01497E−04	0.485443
1.0 *	−1.23668E−03	2.74246	4.07100E−04	0.514085
3.0	−3.59237E−03	7.34395	6.49571E−04	0.609143

The following observations can be made regarding the simulation results:

1. Both the end-point and total tracking errors increased when the payload was increased from 0 to 3 kg. The third joint experienced a large increase in both errors. This is due to the effect of the load on its movement by the gravity term.
2. From Fig. 11 it can be seen that for Joint 1 both computed torque controllers are capable of tracking the trajectory. However, the third joint failed to converge to the desired trajectory as the load was increased. The NNC experienced a



(1) Computed torque controller 1
(2) Computed torque controller 2
(3) Neural-network controller
(4) Fuzzy controller

Legend : 1 - Actual
2 - Desired

Fig. 11. Plot of actual and desired trajectories for Joint 1 under payload variation.

convergence problem during the initial stage, but managed to adapt to track the trajectory after some time.

3. Comparing with the other controllers, the FC achieved reasonably good results and revealed no convergence problems for all three joints regardless of the payload increase. This is particularly so when the load was increased to 3 kg. As seen from Table 10, only the FC was able to achieve the specification when the load was 3 kg.

5.3. Variations of the Trajectory

The performance of the FC was studied and compared with the other three controllers in terms of the tracking errors of the robot manipulator for three different trajectories, namely Paths A, B and C. They represent the original trajectory, the trajectory whose end speed is half of that of Path A, and the trajectory whose end speed is twice as large as that of Path A, respectively. The FC was also subjected to a sine function to examine its adaptive capabilities. The payload mass and coefficients of static, dynamic and viscous friction were set to zero in this simulation study.

Simulation results for the CTC1, CTC2, NNC and FC are shown in Figs. 12 and 13. The end-point and total errors for the four controllers for Joint 1 are tabulated in Table 11.

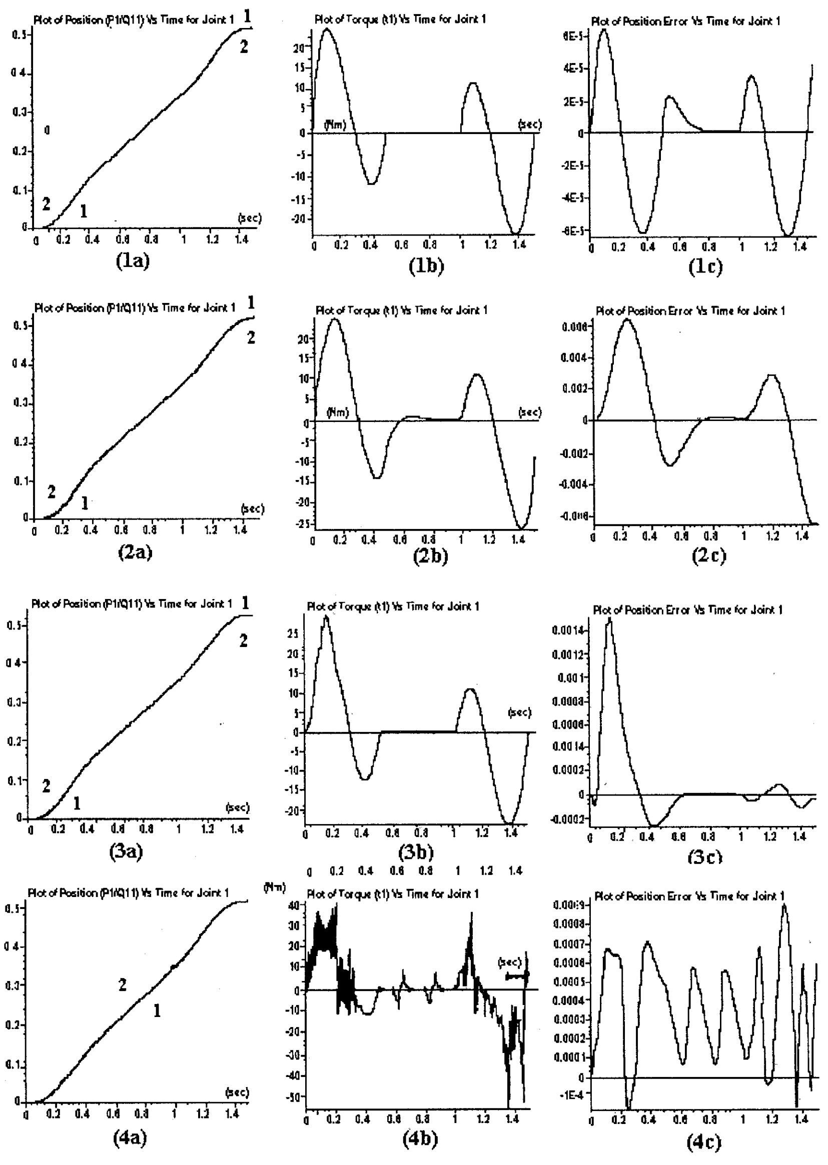
Table 11. End-point and total tracking errors of Adept One robot under different tracking trajectories for Joint 1.

Path	Computed-Torque Controller 1		Computed-Torque Controller 2	
	End-point error	Total error	End-point error	Total error
A	4.29750E−05	3.92004E−02	−6.45053E−03	3.22769
B	8.34465E−05	7.87323E−02	−1.28837E−02	6.58893
C	1.75953E−04	0.152033	−2.66454E−02	13.56030

Path	Neural-network controller		Fuzzy controller	
	End-point error	Total error	End-point error	Total error
A	−3.05176E−05	0.27312	6.0451E−04	0.560862
B	−2.96593E−04	1.18422	1.2815E−04	0.488114
C	−3.87697E−02	67.88790	−7.53403E−03	5.488560

The following conclusions can be drawn from the simulation results:

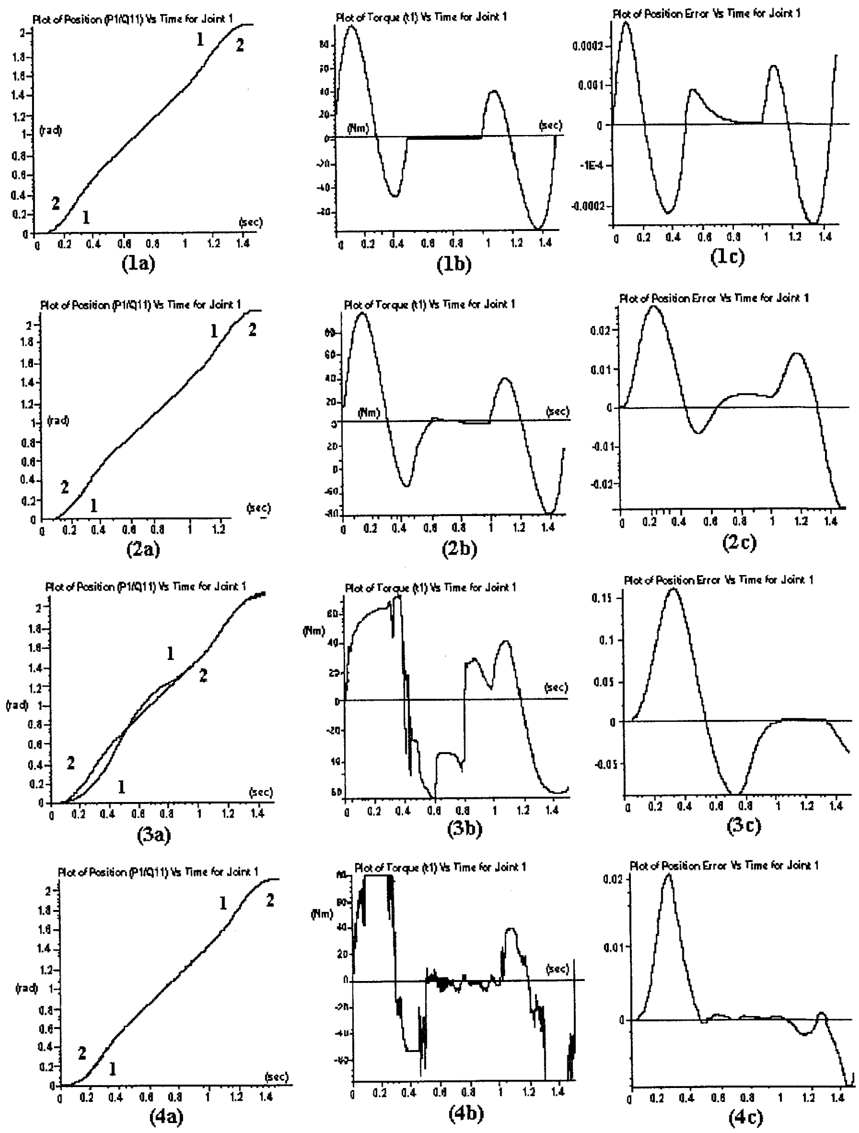
1. The performance of the controllers improved when Path B was used. Since the speed of Path B is half of that of Path A, it is expected that the performance is better. All the four controllers exhibited convergence to the desired trajectories except for CTC2.



(1) Computed torque controller 1
(2) Computed torque controller 2
(3) Neural-network controller
(4) Fuzzy controller

Legend : 1 - Actual
2 - Desired

Fig. 12. Plot of (a) actual and desired trajectory, (b) control torque, and (c) position error of Joint 1 for Path B.



(1) Computed torque controller 1
(2) Computed torque controller 2
(3) Neural-network controller
(4) Fuzzy controller

Legend : 1 - Actual
2 - Desired

Fig. 13. Plot of (a) actual and desired trajectory, (b) control torque, and (c) position error of Joint 1 for Path C.

2. The performances of the controllers deteriorated when the trajectory was changed to Path C. As the speed for Path C was twice as large as that of Path A, the control torque was observed to be higher for all controllers. In the case of Path C, the CTC2 performed even poorer, whereas the second and third joints exhibited more instances of non-convergence to the desired trajectory. The NNC only experienced an initial problem of convergence for the first and second joints.

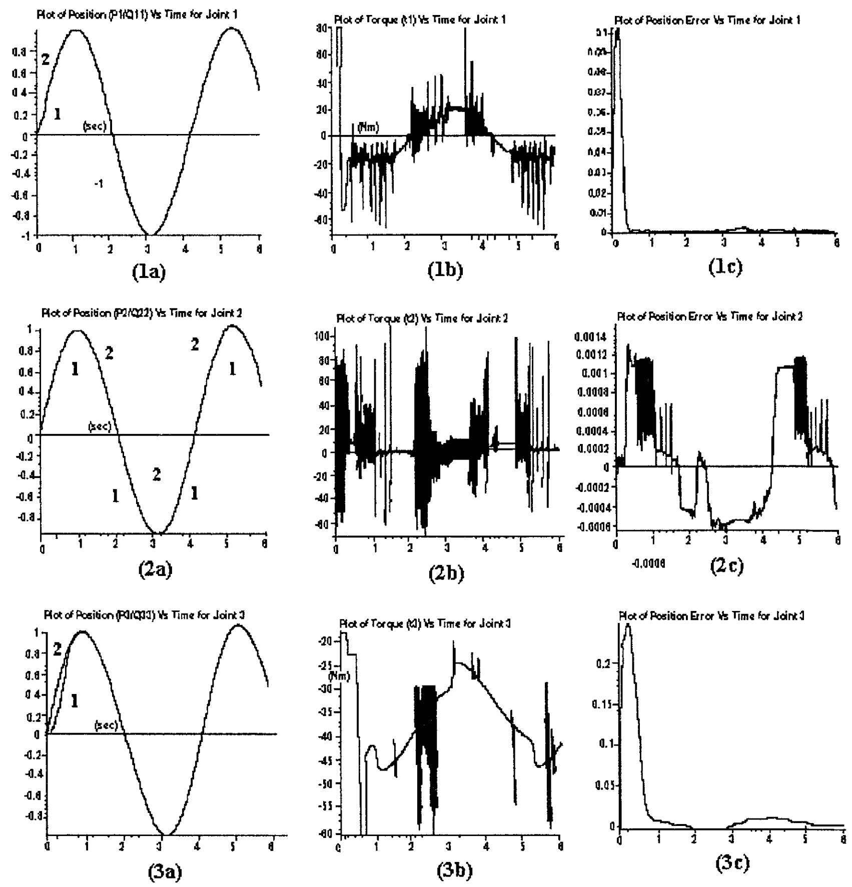
3. When the FC was subjected to a sine function trajectory, it was able to track well except at the initial stage. The plots of actual and desired trajectories, control torques and position errors for the joints of the manipulator are shown in Fig. 14. End-point and total position errors are listed in Table 12. Owing to the fact that the FC was not redesigned for the sine trajectory, the performance is considered satisfactory. It is exptected that the performance would be much better if the FC was redesigned for the sine trajectory.

Table 12. End-point and total tracking errors of Adept One robot under sine function trajectories.

Joint	Fuzzy controller	
	End-point error	Total error
1	1.05381E−04	27.91250
2	−4.29898E−04	2.91946
3	−5.06508E−03	134.32800

6. Conclusions

In this paper, an FC for a three-link manipulator has been proposed and simulated. Simulation results have shown that the performance of the proposed controller is better than that of the NNC of (Er, 1996) and those of the Computed Torque Controllers of (Craig, 1989). Furthermore, the controller is shown to be robust. This is demonstrated via simulation studies under the ideal situation, variations of the payload and coefficients of static, dynamic and viscous friction, changes in the trajectory and combined variations of the above. However, it was found that the control torque is highly oscillatory when the MAX-MIN function are used in the fuzzy rules. These high oscillations could be eliminated by using MAX-PROD in the fuzzy rules. In summary, another approach of using neural networks and fuzzy logic to control an n-degree-of-freedom robotic manipulator has been demonstrated via simulation studies on the the Adept One Robot.



Legend : 1 - Actual
2 - Desired

Fig. 14. Plot of (a) actual and desired trajectory, (b) control torque, and (c) position error for fuzzy controller under sine trajectory.

References

Atkeson C.G. and Reinkensmeyer D.J. (1988): *Using associative content-addressable memories to control robots.* — Proc. 27th IEEE CDC, USA, Vol.1, pp.792–797.

Atkeson C.G., An C.H. and Hollerbach J.M. (1985): *Rigid body load identification for manipulators.* — Proc. 24th IEEE CDC, USA, pp.996–1002.

Bhat N.V., Mindelman P.A., McAvoy T. and Wang N.S. (1990): *Modelling chemical process systems via neural computation.* — IEEE Contr. Syst. Mag., USA, Vol.10, No.3, pp.24–31.

Chen F.C. (1990): *Backpropagation neural networks for nonlinear self-tuning adaptive control.* — IEEE Contr. Syst. Mag., USA, Vol.10, No.3, pp.44–48.

Craig J.J (1989): *Introduction to Robotics: Mechanics and Control.* — Reading, Mass: Addison-Wesley.

Er M.J. (1996): *Neural network control of Adept One SCARA.* — The EEE Journal, Vol.8, No.1, pp.91–95.

Goldberg K. and Pearlmutter B. (1988): *Using a neural network to learn the dynamics of the CMU Direct-Drive Arm II.* — Techn. Rep. CMU-CS-88-160, Carnegie Mellon University.

Guez A. and Ahmad Z. (1988): *Solution to the inverse kinematics problem in robotics by neural network.* — Proc. IEEE Int. Conf. *Neural Networks*, USA, Vol.2, pp.617–624.

Hasegawa T., Horikawa S., Furuhashi T. and Uchikawa Y. (1993): *An application of fuzzy neural networks to design of adaptive fuzzy controllers.* — Proc. Int. Joint Conf. *Neural Networks*, Nagoya, Japan, pp.1761–1764.

Horikawa S., Furuhashi T. and Uchikawa Y. (1992): *On fuzzy modelling using fuzzy neural networks with the backpropogation algorithm.* — IEEE Trans. Neural Networks, Vol.3, No.5, pp.801–806.

Horne B. and Jamshidi M. (1988): *A connection network for robotic gripper control.* — Proc. 27th CDC, USA, Vol.2, pp.1070–1074.

Karakasoglu A., Sudharsanan S.I. and Sundareshan M.K. (1993): *Identification and decentralized adaptive control using dynamical neural networks with application to robot manipulators.* — IEEE Trans. Neural Networks, Vol.4, No.6.

Kawato M., Furukawa K. and Suzuki R. (1987): *A hierarchical neural-network model for control and learning of voluntary movement.* — Biol. Cybern., Vol.57, pp.169–185.

Kawato M., Setoyama T. and Suzuki R. (1988a): *Feedback error learning of movement by multilayer neural network.* — Proc. Int. *Neural Networks Society First Annual Meeting*, USA, Vol.1, No.1.

Kawato M., Uno Y., Isobe M. and Suzuki R. (1988b): *Hierarchical neural network model for voluntary movement with application to robotics.* — IEEE Contr. Syst. Mag., Japan, Vol.8, No.2, pp.8–15.

Khosla P.K. and Kanade T. (1985): *Parameter identification of robot dynamics.* — Proc. 24th CDC, USA, pp.1754–1760.

Kim S.H., Kim Y.H., Sim K.B. and Jeon H.T. (1993): *On development of an adaptive neural-fuzzy control system.* — Proc. 1993 IEEE/RSJ Int. Conf. *Intelligent Robots and Systems*, Yokohama, Japan, pp.950–957.

Kung S.Y. and Hwang J.N. (1989): *Neural network architectures for robotic applications.* — IEEE Trans. Robot. Automat., Vol.5, No.2, pp.641–657.

Kuperstein M. and Wang J. (1990): *Neural controller for adaptive movements with unforeseen payloads.* — IEEE Trans. Neural Networks, Vol.1, No.1, pp.137–142.

(1990): *Real time dynamic control of an industrial manipulator using a neural network based learning controller.* — IEEE Trans. Robot. Automat., USA, Vol.6, No.1, pp.1–9.

Miyamoto H., Kawato M., Setoyama T. and Suzuki R. (1988): *Feedback error learning neural network for trajectory control of a robotic manipulator.* — Neural Networks, Japan, Vol.1, No.3, pp.251–265.

Mizukami K. and Fuke H. (1991): *On learning of fuzzy controller with neural network.* — Proc. 1st FAN Symp., Osaka, Japan, pp.263–266.

Narendra K.S. and Parthasarathy K. (1989): *Neural networks in dynamical systems.* — Proc. *SPIE Vol.1196, Int. Control and Adaptive Syst.*, USA, pp.230–241.

Narendra K.S. and Parthasarathy K. (1990): *Identification and control of dynamical systems using neural networks.* — IEEE Trans. Neural Networks, USA, Vol.1, No.1, pp.4–27.

Nyugen D.H. and Widrow B. (1990): *Neural networks for self-learning control systems.* — IEEE Cont. Syst. Mag., USA, Vol.10, No.3, pp.18–23.

Schilling R.J. (1990): *Fundamentals of Robotics—-Analysis and Control.* — New Jersey: Prentice Hall.

Setoyama T., Kawato M. and Suzuki R. (1987): *Manipulator control inverse-dynamic model learned in multi-layered neural network.* — Japan IEICE Techn. Report., Vol.MBE87–135, pp.249–256.

Psaltis D., Sideris A. and Yammamura A. (1988): *A multilayered neural network controller.* — IEEE Contr. Syst. Mag., USA, Vol.8, No.2, pp.17–21.

Watanabe K., Tang J., Nakamura M., Koga S. and Fukuda T. (1996): *A fuzzy-Gaussian neural network and its application to mobile robot control.* — IEEE Trans. Contr. Syst. Techn., Vol.4, No.2, pp.193–199.

Zadeh L.A. (1965): *Fuzzy sets.* — Information and Control, Vol.8, pp.338–353.

Zadeh L.A. (1973): *Outline of a new approach to the analysis of complex systems and decision processes.* — IEEE Trans. Syst. Man Cybern., Vol.SMC–3, No.1, pp.28–44.

Received: 10 October 1996
Revised: 25 February 1997