'88 KACC 1988. 10. 21~22

An Application of the CMAC to Robot Control

Kwanghee Nam and Tae-yong Kuc
Department of Electrical Engineering, POSTECH
Pohang, P.O.Box 125, Kyungbuk 790-600, S. Korea

Abstract: An iterative learning control scheme is presented with the aid of CMAC module. By enforcing the role of linear controller with the introduction of velocity feedback, it becomes possible to make the trajectory error equation stable. One advantage of this control scheme is that it does not require acceleration feedback. Computer simulation results shows a good performance of the scheme even in the case where the gravity is not compensated.

1 Introduction

Since the dynamics of rotot manipulators is highly nonlinear, it is not easy with classical fixed gain controllers to accomodate the parameter changes resulting from nonlinear charaterisctics. As a nonlinearity compensating controller, computed torque method[3] was proposed. However, since the computed torque method requires apriori knowledge of robot dynamics, it is not flexible with respect to the varying environments such as payload changes. Various techniques of model following adaptive control[8] have been utilized to incompass such an inflexibility without a precise description of the dynamic model. A general drawback of adaptive approach is a large computational load for the real-time parameter identification.

Recently, iterative learning techniques[2,5,6] based on the repeatability of robot motion have been paid a special attention as a new control scheme. Apart form the simplicity and straightforwardness, the iterative learning methods are flexible because they don not require the exact dynamic model. Bondi et. al.[4] established the the uniform boundedness of the trajectory error for an iterative learning scheme by utilizing high gain feedback.

Miller et. al.[9] have proposed a feasible control scheme utilizing *Cerebellar Model Articulation Controller (CMAC)* which was developed by Albus[1]. They have used a learning scheme to adjust the values in the CMAC module online based observation of the robot input-output relationship, in order to form an approximate dynamic model of the robot in appropriate regions of the state space.

Following the work of Miller III et.al.[9], we also utilize the CMAC for a learning control of the robot motion, but in this paper the need for the acceleration measurement is eliminated because the possible acceleration measurement error may deteriorate the performance controller. In Section 2, we describe the dynamic model of a robot manipulator algon with a learning control strategy. In Section 3, it is illustrated the learning control scheme that is utilized in this paper. Finally, compuater simulation results for an example are shown in Section4.

2 Dynamics of a Robot Manipulator and CMAC

Generally, the dynamics of robot manipulators which have n degree of freedom is described by

$$D(q)\ddot{q} + B(q,\dot{q}) + G(q) = T, \tag{1}$$

where $D(q)$, $B(q,\dot{q})$, and $G(q)$ represent the inertia matrix, Coriolis plus centripetal forces and the gravity vector, respectively in the generalized coordinates $q \in R^n$, while $T \in R^n$ represents the vector of torques applied to each joint.

The control objective here is to make a manipulator track a given desired trajectory by iterative learning method. That is, by training the system through repetition of tracking, we are aiming at reducing trajectory error progressively.

By superscript i we denote the state and torques of i-th iteration. To distiguish from the state $(q_d, \dot{q}_d)$ of the desired trajectory, we denote by $(q_o, \dot{q}_o)$ the state of robot.

Linearizing the system (1) in i-th iteration along the desired trajectory $q_d(t)$, we obtain the following system:

$$C(t)(\ddot{q}_o^i - \ddot{q}_d) + E(t)(\dot{q}_o^i - \dot{q}_d) + F(t)(q_o^i - q_d) = T - S(t), \qquad (2)$$

where

$$C(t) = D(q_d), \quad E(t) = \frac{\partial B}{\partial \dot{q}}|_{(q_d,\dot{q}_d)},$$

$$F(t) = \frac{\partial D}{\partial q}|_{q_d}\ddot{q}_d + \frac{\partial B}{\partial q}|_{(q_d,\dot{q}_d)} + \frac{\partial G}{\partial q}|_{q_d},$$

and

$$S(t) = D(q_d)\ddot{q}_d + B(q_d,\dot{q}_d) + G(q_d).$$

We apply the following input to the system (1) at the i-th iteration:

$$T^i = K(q_d - q_o^i) + L(\dot{q}_d - \dot{q}_o^i) + H^i(t), \qquad (3)$$

where K and L are positive definite constant matrices, and $H^i(t)$ is an additional input vector to be adjusted so that the term $S(t)$ is canceled out in the error equation. Applying input (2) to the system, we obtain the equation for an error $e^i(t) = q_d(t) - q_o^i(t)$

$$D(t)\ddot{e}^i(t) + (E(t) + K)\dot{e}^i(t) + (F(t) + L)e^i(t) = S(t) - H^i(t). \qquad (4)$$

The goal is to let $e^i(t)$ go to zero as the iteration number i increases. For this purpose, we choose for K and L sufficiently large positive definte matrices so that the time-varing terms, $E(t)$ and $F(t)$ may be dominated. Then, the dynamics for the error equation (3) can be made stable. On the other hand, since the nonzero term $S(t)$ drives the error $e^i(t)$ away from zero, it should be canceled out by the term $H^i(t)$. However, since it is assumed that the exact dynamic model of the manipulator is unkown, $S(t)$ is not known initially. Hence, we corporate a learning mechanism so that $H^i(t)$ converges to $S(t)$ for each t with the number of iteration. If the trajectory error $e^i(t)$ is exactly zero, then the following must be satisfied:

$$T^i(t) = H^i(t) = S(t).$$

Hence, $S(t)$ can be interpreted as the torque required when moving along the desired trajectory. The proof of convergence of this sort has been establised in [7].

For a learning scheme, we utilize CMAC which was originally proposed by Albus[1]. In the CMAC module, memory is used as associative neural elements, whose contents are adjusted so as to produce the correct output. CMAC is basically a table look-up techniques for representing complex nonlinear functions utilizing the concept of continuity. That is, since the output values of a continuous function for neighboring input points are not so much different, one may be able to save the size of memory in storing the output values of the continuous function by sharing the common portion of the data. CMAC realizes this idea through a content addrresing technique. For more specific illustration of CMAC, refer to [9,1].

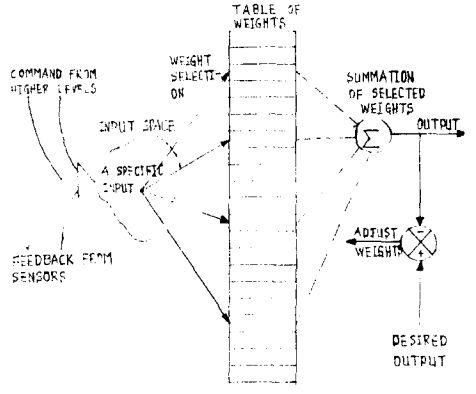

Figure 1. The CMAC module block diagram

3 A Learning Control Scheme

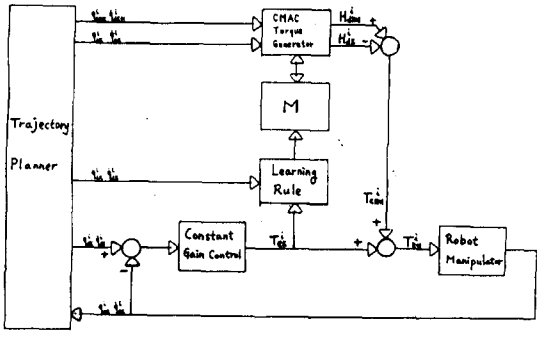

Figure 2. Learning control scheme of a manipulator with CMAC

Figure 2 illustrates a block diagram of a CMAC-based learning control scheme utilized in this work. It consists of three major parts: Firstly, it has a position and velocity feedback controller in which the gain matrices K and L are suitably chosen so that the error equation (4) is stable. No-

tice again that it is basically a high gain controller for the stability. Secondly, it has a CMAC learning algorithm with which the stored values *(often called weights)* in the look-up table *(often called neural memory)* are updated for the next iteration. Specifically, the weights are corrected by the proportional amount of errors in the position and velocity. Finally, it has a CMAC torque generator where the torques H^i_{dk+1}, H^i_{ok} are read out from the memory corresponding to the desired and actual states. That is, the torques H^i_{dk+1}, H^i_{ok} are generated by summing the weights in the neural memory which are addressed by the content addressable hash function with inputs $(q^i_{dk+1}, \dot{q}^i_{dk+1})$, $(q^i_{ok}, \dot{q}^i_{ok})$.

The trajectory planner has a function generator inside which produce the desired state for each step. The two torques H^i_{dk+1}, H^i_{ok} from the CMAC torque generator are subtracted to make a compensating torque T^i_e, i.e., the compensating torque at the $(k+1)-$th step is given by

$$T^i_{ck+1} = H^i_{dk+1} - H^i_{ok}. \tag{5}$$

This torque corresponds to the torque needed for moving the manipulator to the desired trajectory at the $(k+1)$-th step, if not for the position and velocity errors. This compensating torque is added to the torque T^i_{ek} from the linear controller, resulting in the total torque T^i_{k+1} which is applied to the system. When the control system is not sufficiently trained at the initial stage of learning, the manipulator would not follow the desired trajectory exactly. In this case, the whole scheme depends more on the linear controller, because the error has not been vanished. However, if the manipulator follows the desired trajectory after training, the function of linear controller is not activated at all since the error $e(t)$ is equal to zero. In other word, after the perfect training, it turns out to be that

$$D(q_d)\ddot{q}_d + B(q_d, \dot{q}_d) + G(q_d) = H_d, \tag{6}$$

since $T_k = H_{dk}$ for each step.

On the other hand, by the CMAC learning rule, the torque H^i_{dk} for the desired trajectory at the $k-$th step are updated for the $(i+1)$-th iteration as follows:

$$H^{i+1}_{dk} = H^i_{dk} + \beta T^i_{ek}, \tag{7}$$

where β is a training factor between 0 and 1. With the above update rule, the sequence $\{H^i_{dk}\}_i$ converge to a value which is required for moving the manipulator one step ahead on the desired trajectory.

Miller III et.al.[9] also utilized the CMAC module in the process of learning the inverse dynamics the system through input-output data base. Under the assumption that the acceleration term is available, they treated the dynamics of a robot just as an algebraic equation and obtained the inverse mapping relationship between the input-output data. However, the intrinsic time delay of acceleration sensors and the noise sensitivity of differentiation of signals make it difficult to utilize acceleration measurements in the contoller.

However, in our approach we do not assume that the acceleration is available, because it is practically very hard to obtain the accurate values of acceleration due to the intrinsic time delay of acceleration sensors and the noise sensitivity. We take more of dynamics into account by merging the concept of classical feedback control.

4 A Simulation Example

We consider a two-axis robot manipulator shown in Figure 3 as an example.

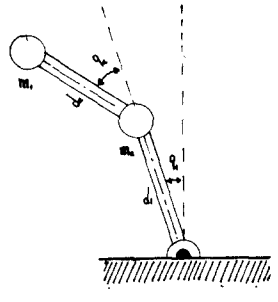

Figure 3. Schematic diagram of two-axis robot manipulator
used in the dynamic simulation

The dynamic model of the above manipulator is given by

$$\begin{aligned} T_1 &= [(m_1+m_2)d_1^2 + m_2 d_2^2 + 2m_2 d_1 d_2 \cos q_2]\ddot{q}_1 \\ &+ (m_2 d_2^2 + m_2 d_1 d_2 \cos q_2)\ddot{q}_2 \\ &- m_2 d_1 d_2 \dot{q}_2^2 \sin q_2 - 2m_2 d_1 d_2 \dot{q}_1 \dot{q}_2 \sin q_2 \\ &+ (m_1+m_2) g d_1 \sin q_1 + m_2 g d_2 \sin(q_1+q_2) \end{aligned} \tag{8}$$

$$\begin{aligned} T_2 &= [m_2 d_2^2 + m_2 d_1 d_2 \cos q_2]\ddot{q}_1 + m_2 d_2^2 \ddot{q}_2 \\ &- m_2 d_1 d_2 \dot{q}_1 \dot{q}_2 \sin q_2 + m_2 g d_2 \sin(q_1+q_2), \end{aligned} \tag{9}$$

where m_i, d_i and q_i represent the mass, length and joint

angle of the link i for $i = 1, 2$. We let $m_1 = 10Kg$, $m_2 = 10Kg$, $d_1 = 1.0m$ and $d_2 = 0.8m$. For a desired trajectory, we chose

$$q_1(t) = (\pi/3)\cos(2\pi/5)t, \qquad 0 \le t \le 5$$

$$q_2(t) = -(\pi/2)\cos(2\pi/5)t - \pi/4, \qquad 0 \le t \le 5$$

with the following initial and final conditions:

$$q_1(0) = q_1(5) = \pi/3, \quad q_2(0) = q_2(5) = -3\pi/4,$$

$$\dot{q}_1(0) = \dot{q}_2(0) = \dot{q}_1(5) = \dot{q}_2(5) = 0.0\,(rad/sec)$$

Further, we chose $\beta = 0.6$, $5msec$ for the step size, and the following for the gain matrices:

$$K = \begin{bmatrix} 150 & 0 \\ 0 & 150 \end{bmatrix}, \qquad L = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}.$$

The simulation program was written in language C and was run on VAX-8800. In Figure 4 the actual trajectories of the system with only a linear feedback controller is shown compared with the desired trajectory. Figure 5,6,7 show that the actual trajectories converge to the desired one as the iteration number increases from 1 to 9. Figure 8,9,10 show the velocity profiles corresponding to Figure 5,6,7. In Figure 11, the sums of squared position and velocity errors of each joint are plotted versus iteration numbers. One can notice from Figure 11 that the position erros vanish after iteration number 5, while the velocity errors become negligible after number 6. Figure 12 shows $\sum_k \{T_{ck}^i\}^2$ and $\sum_k \{T_{ck}^i\}^2$ for each joint are plotted versus iteration numbers. Figure 12 shows the evidence of the transition of control schemes. That is, at the early stage of iteration the torques from linear controller T_e are much larger than the compensating torque T_c from the neural memory. But, since the position and velocity errors decreas as the iteration number increases, the torque T_e becomes negligible while the compensating torque T_c grows high. In other words, the compensating torque T_c becomes the desired one needed for the tracking of the desired trajectory. Finally, Figure 13 show the actual trajectory of the system with only a linear feedback controller when the gravitational term is not compensated. Similarly, Figure 14,15,16 show the trjectories of the system converging to reference trajectory as iteration number increases from 1 to 13.

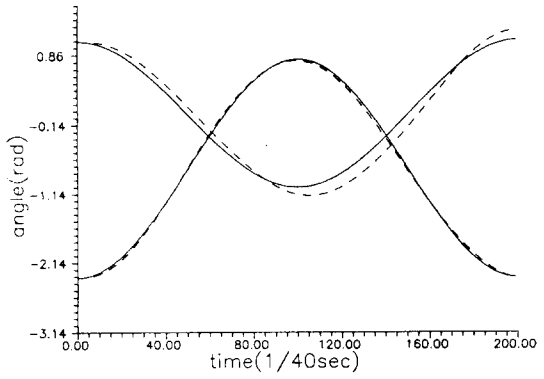

Figure 4. Trajectories of the system without a learning scheme (solid line: desired trajectory; dotted line: actual trajectory)

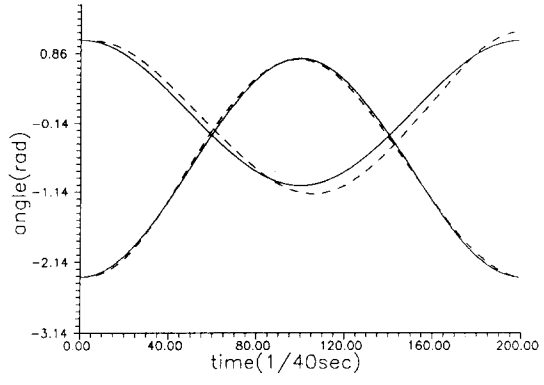

Figure 5. Trajectories of the system after the 1st iteration (solid line: desired trajectory; dotted line: actual trajectory)

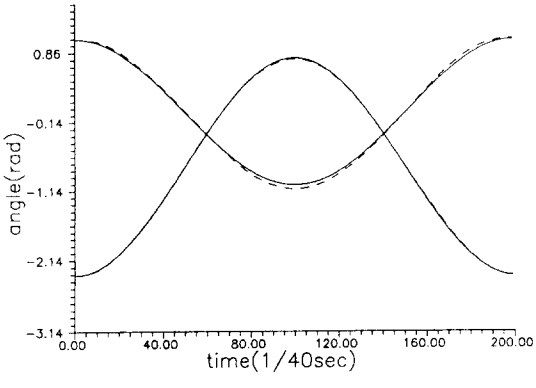

Figure 6. Trajectories of the system after the 3rd iteration (solid line: desired trajectory; dotted line: actual trajectory)

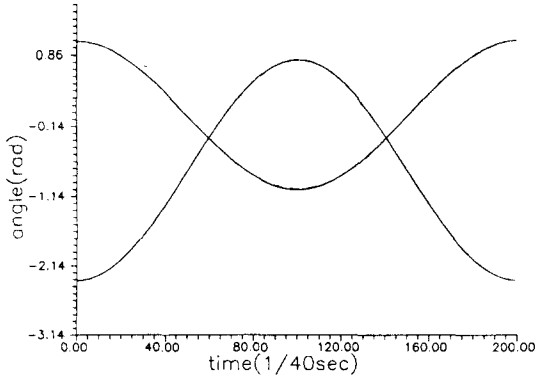

Figure 7. Trajectories of the system after the 9th iteration (solid line: desired trajectory; dotted line: actual trajectory)

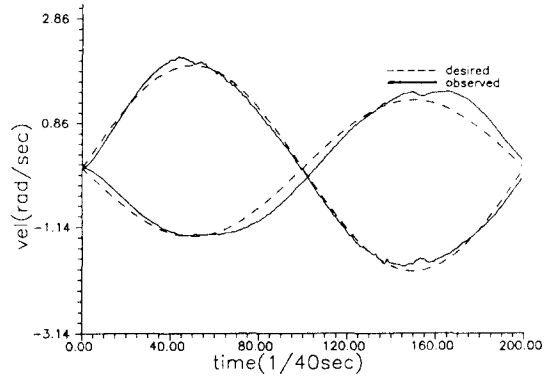

Figure 8. Velocity profile of the system after the 1st iteration (solid line: actual velocity; dotted line: desired velocity)

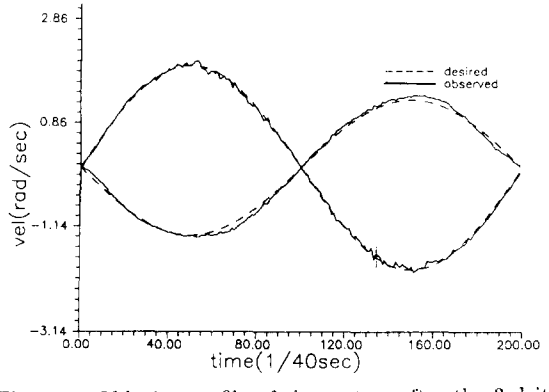

Figure 9. Velocity profile of the system after the 3rd iteration (solid line: actual velocity; dotted line: desired velocity)

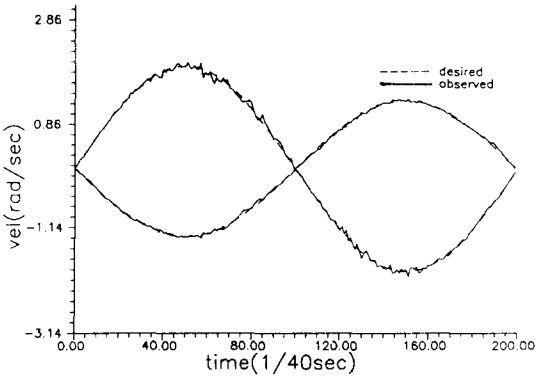

Figure 10. Velocity profile of the system after the 9th iteration (solid line: actual velocity; dotted line: desired velocity)

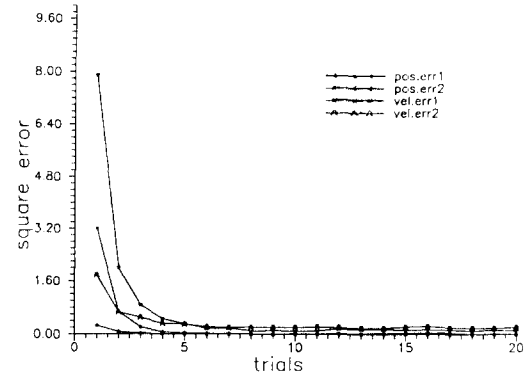

Figure 11. Plot of the squared sum of position and velocity errors versus iteration number

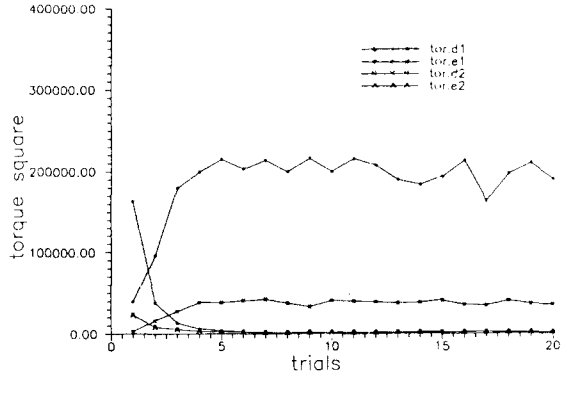

Figure 12. Plot of $\sum_k \{T_{ck}^n\}^2$ and $\sum_k \{T_{ck}^n\}^2$ for each joint versus iteration number

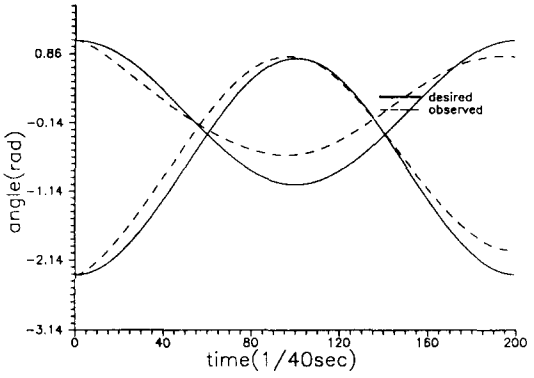

Figure 13. Trajectories of the system without gravity compensation and learning scheme (solid line: desired trajectory; dotted line: actual trajectory)

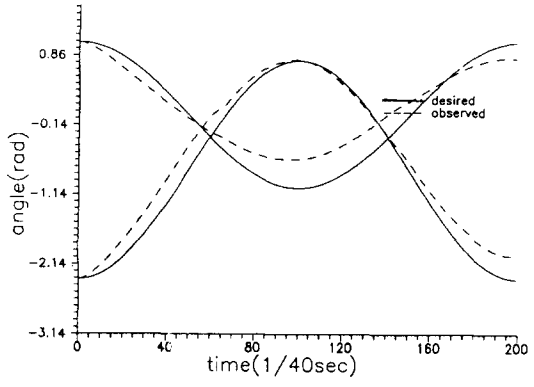

Figure 14. Trajectories of the system without gravity compensation after the 1st iteration (solid line: desired trajectory; dotted line: actual trajectory)

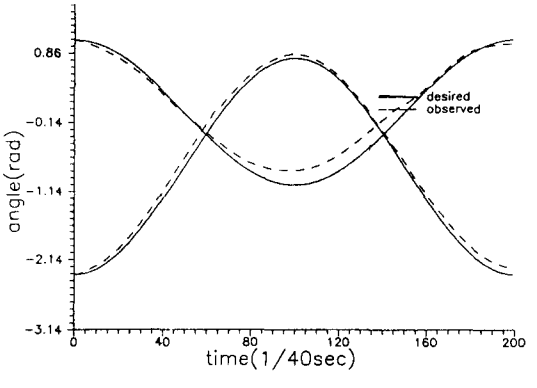

Figure 15. Trajectories of the system without gravity compensation after the 3rd iteration (solid line: desired trajectory; dotted line: actual trajectory)

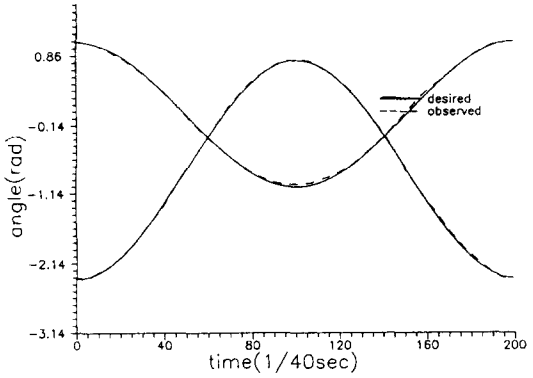

Figure 16. Trajectories of the system without gravity compensation after the 13th iteration (solid line: desired trajectory; dotted line: actual trajectory)

5 Conclusion

In this work, a new control method for robot tracking is suggested by modifying the learning control algorithm in [9]. The controller in this paper also utilizes the CMAC for learning as well as a linear feedback controller. But, comparing this control scheme with the original one, more emphasis is laid on the linear controller part. That is, through the addition of velocity feedback we makes the error equation stable. This control method, in turn, results in the elimination of the need for the acceleration which is necessary in obtaining the functional inverse in [9]. This not only reduces the memory size for learning but also makes the contoller more robust by avoiding the possible error caused by acceleration sensors. The simulation results for a two-axis manipulator prove strongly its performance.

References

[1] James S. Albus " Brains, Behavior, and Robotics, " N. H. Byte Books, 1981.

[2] S. Arimoto, S.Kawamura, and F. Miyasaki, "Bettering operation of robots by learning," J. Robotic Syst., pp 123-140. 1984.

[3] A. K. Bejezy, " Robot Arm Dynamics and Control," Technical Memo 33-669, Jet Propulsion Laboratory, Pasadena, CA, 1974.

[4] Poala Bondi, Giuseppe Casaline, and Lucia Gambardella, " On the iterative learning control theory for

robotic manipulators , " J.Robotics and Automation, Vol.4, No.1, Feb.1988.

[5] G. Casalino and G. Bartolini, "A learning procedure for the control of movements of robotic manipulators," presented at the IASTED Symp. Robotics and Automation, Amsterdam. The Netherlands. June 1984.

[6] J. J. Craig, "Adaptive control of manipulators through repeated trials," in Proc. 1984 Amer. Control Conf., San Diego, CA, June 1984.

[7] Sadao Kawamura, Fumio Miyazaki, and Suguru Arimoto, " Realization of robot motion based on a learning method ," Trans. Sys. Man.,Cyber., Vol.18, No.1, Jan./Feb.1988.

[8] C.S.G. Lee and M.J. Chung, "An adaptive control strategy for mechanical manipulators," IEEE Trans. Automatic Contr., AC-29, pp 837-840, 1984.

[9] W. Thomas Miller III, Filson H. Glanz, L. Gordon Kraft III, "Application of a general learning algorithm to the control of robotic manipulators, " Int.,J.Robotics R. Vol.6,No.2, Sum.1987.