Proceedings of The
IEEE International Conference
on Industrial Technology, 1996

Hybrid Position/Force Controller of Robot Manipulators Based on CMAC Learning

Qing Wei, Wenseng Chang, and Peng Zhang
Department of Automatic Control
National University of Defense Technology
Changsha, China, 410073

Abstract—**In this paper, Based on CMAC learning algorithm we proposed a stable hybrid position and force control scheme. The controller is similar to Computed Torque controllers, with the robot dynamic model replaced by the CMAC module. A learning scheme is used to adjust the memory values in the CMAC module on-line based on observations of the robot input-output relationship, in order to form an approximate dynamic model of the robot in appropriate regions of the state space. The CMAC module is used to predict the actuator torque required to follow a desired trajectory(force), and these torque are used as feedforward terms in parallel with a fixed-gain, linear PD feedback controller. The parameters of the PD feedback controller are strictly selected to ensure system dynamic stability. Final simulation on a PUMA562 manipulator shows the based-CMAC hybrid force/position controller have get dynamic stable control, fast learning speed. It is not only suitable for repeated tasks, but also suitable for non-repeated tasks.**

I. INTRODUCTION

On the present day, most industry manipulators can only provide position control mode; they have not force control ability. However in some important industry automation application, especially in deburring, polishing, grinding, or automatic assembly, contact force control of manipulators is critical. Force control is going to be the ability of industry robots.

However robot manipulators in force control mode often become unstable during contact with high stiff environments. This is called dynamic instability. To ensure closed-loop stability during contacting high stiff environments, the feedback proportion gain of ordinary PID controllers can only be small. This leads a bad dynamic response and a low force control precision. An and Hollerback[3] point out the reason of dynamic instability is force sensor's information high gain feedback. Reberts[4] use a soft force sensor and gain a stable contract when manipulators contact with high stiffness environment. Eppinger and Seering[5] qualitative analysis instability problem of manipulator force control, they consider that the dynamic nature of manipulators and stimulate motors are one of main reasons to affect force control stability. However the contacting control problem of high stiffness force sensors and high stiffness environments is still unsolved. In most typical contact tasks, manipulators are unluckily contacting with high stiffness environment, such as some kind of metal surface. The final solution of this problem will lead the wide application of manipulators force control in industry produce.

In this paper, Based on CMAC learning algorithm we proposed a stable hybrid position/force control scheme. The controller is similar to Computed Torque controllers, with the robot dynamic model replaced by the CMAC module. To form an approximate contact dynamic model of the robot in appropriate operational regions, a learning scheme is used to adjust the memory values of CMAC module on-line based on observations of the robot input-output relationship. The CMAC module is used to predict the actuator torque required to follow a desired trajectory(force), and these torque are used as feedforward terms in parallel with a fixed-gain, linear PD feedback controller. The parameters of the PD feedback controller are strictly selected to ensure system dynamic stability.

II. CMAC LEARNING ALGORITHM

As shown in figure 1, CMAC learning methods in fact is associated memory system. CMAC divided input state space into many small pieces. Every piece assigns some storage units. A input vector of CMAC used as a address index of these storage units. The real output vectors of CMAC are distributed store in these memory units. The desired output are used to adjust values of these storage units in order to form an approximate dynamic model of the input-output relationship. The basic idea of CMAC learning algorithm is to learn the form of the input-output relationship function f from a knowledge of measurements of f at a set of train points, and subsequently to be able to reproduce with reasonable accuracy values of f at least at some neighboring regions of training points set. Clearly the learning system should have some capability of interpolating or generalizing for new points lying between the training points.

In figure 2, We show a detailed legend of CMAC learning algorithm. The basic input-output relationship function that the CMAC network system wants to learn is

$$Y = f(X) \tag{1}$$

Here $X \in S \subset R^n$ is the input vector, S is a space consisted of all possible input vector, Y is desired output of learning object. $(Y_i, X_i), i = 1,2,\cdots,N$ are training points set, N is number of

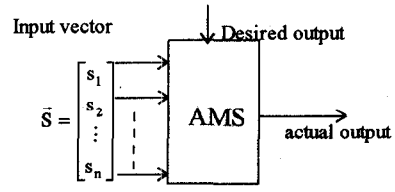


Fig.1 Associated memory system

0-7803-3104-4

training points. In fact every input X_i is nx1 vector, that is $X_i = [x_1^i, x_2^i, \cdots, x_n^i]^T$. For every input vector, CMAC has $R_j, j = 1,2,\cdots,n$ levels of quantization. So total input space are divided to $\prod_{j=1}^{n} R_j$ pieces. Every piece assigns ρ storage units. The output information of every piece are distributed store in these ρ storage units. ρ is called generalization coefficient. In input space S, the storage units of nearby input pieces are partly share. The number of input pieces are possible extreme huge in practical problem. For example if the input is 10x1 vector, every sub-input have 100 levels of quantification, then the number of input pieces are 10^{20}. We are unable to store the information of all these pieces. The number of actual memory units are more less than the number of all possible pieces. According to Albus' learning algorithm the number of required memory units are

$$P = \rho \prod_{j=1}^{n} \left(\frac{R_j}{\rho} + 1 \right) \tag{2}$$

It is transparent that when the dimension of input vecto and the number of quantification levels are more, the number of required memory units are impossible sufficed. So in many practical problem, we map all ideal required memory to the actual memory though a hash function. As a result of hash function's introduction, there are data collider exist in the CMAC memory. That is that irrelvant input pieces share the same storage units.

The concept of association vector is a good describe for above memory mapping pattern. Association vector $\vec{a}$ is a Px1 binary vector. P is actual memory numbers. Every input X_i is mapping a association $\vec{a}_i$. If the i-th bit of $\vec{a}_i$ is 1, its means that this input piece select the i-th memory units to store output information. On the contrary If the i-th bit of $\vec{a}_i$ is 0, its means that this input piece do not select the i-th memory units to store data. The association vector $\vec{a}_i$ usually have ρ bits is 1, others are 0. This means that the information of every input piece are only distributed store in these ρ memory units. Albus's learning Algorithm is

1) The k-th training point is $\left(Y_{(k)}, X_{(k)}\right), (k) \in [1,2,\cdots,N]$

then CMAC output forecast is

$$f_{(k)}(X_{(k)}) = \frac{1}{\rho} \sum_{i=1}^{\rho} m_{(i)} = \frac{1}{\rho} \vec{a}_{(k)}^T \cdot M_{k-1} \tag{3}$$

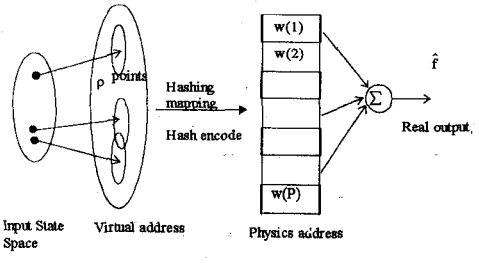


Fig 2. CMAC learning methods

2) System output errors is

$$\mu_{(k)} = Y_{(k)} - f_{(k-1)}\left(X_{(k)}\right) = Y_{(k)} - \frac{1}{\rho} \vec{a}_{(k)}^T \cdot M_{k-1} \tag{4}$$

3) CMAC memory adjustments are

$$M_k = M_{k-1} + \mu_k \vec{a}_{(k)} \tag{5}$$

CMAC learning algorithm essentially are to solve as follow linear matrix equation

$$A \cdot M^* = h \tag{6}$$

here $M^* = [m_1 \ m_2 \ \cdots \ m_P]^T$ is desired values of CMAC memories, p is number of CMAC memories. $A = [\vec{a}_1 \ \vec{a}_2 \ \cdots \ \vec{a}_N]^T$ is $N \times p$ matrix, N is number of train samples. $h = [Y_1 \ Y_2 \ \cdots \ Y_N]$ is $N \times 1$ vector. $\vec{a}_i$ is association vector of Y_i. It might seem that a solution for M^* is a trivial problem in linear algebra involving a matrix inversion or pseudo-inversion of A. Indeed the Moore-Penrose pseudo-inverse of A supplies an ideal least-squares solution for M^*. However in actual application, the size of A, which is Nxp with typical N~20,000 and p~5,000, rules out such a solution as the computational burden is far too great. The Albus algorithm, on the other hand, is extreme simple involving only very little scalar multiplication. It is this simplicity that has led to its widespread use in CMAC systems, as opposed to more sophisticated numerical techniques which have been developed to solve sparse matrix equations such as (6). The simplicity of the Albus algorithm is particularly advantageous in real-time computations with the comparatively simple computing hardware available for on-line use. In discussing convergence of Albus learning algorithm, the question of consistency or inconsistency of the equations (6) is very critical. In the case of inconsistency, which is quite likely to arise due to measurement noise distorting ideal input-output relationship. And the form of training procedure is also crucial. Parks and Militzer[8] shown in most actual application, that is the inconsistency equation (6) convergence to a 'limit cycle' or 'minimal capture zone'.

III. APPLICATION IN ROBOT FORCE CONTROL

Usually force control is divided to impedance control and hybrid position-force control. Impedance controls do not directly control contacting force itself, but according to contacting force adjusting position feedback error, or velocity feedback error, or adjusting system impedance to an appropriate mode. Hybrid position/force based on Mason's C-surface theory, decomposes robot's operational space into two orthogonal subspaces: force-free subspaces and position-free subspace. In force-free subspaces, robot carry on force control. On the contrary in orthogonal position-free subspace, manipulator carry on position control. If we only consider in single-axis force control case, a more reasonable division of force control is direct force feedback and impedance control. Direct force control has two kinds' applications: 1) Based on force servo control, 2) Based on inner position control. Force control methods based on force servo control usually apply

some kinds PID control method and/or a some simple filtering. Force control methods based on inner position control is that outer force feedback control loop provides position command to inner position control loop.

This paper discuss hybrid position/force control based on direct force control of mutil-axis robot. Robot dynamic equation is

$$M(x)\ddot{x} + H(x,\dot{x}) + G(x) = F_c + F_e \tag{7}$$

Operational space hybrid position/force control law is

$$\begin{aligned} F_c &= F_f + F_b,\ F_f = \left[H(x,\dot{x}) + G(x)\right] \\ F_b &= M^{-1}(x)S[\text{position control law}] + \bar{S}[\text{force control law}] \end{aligned} \tag{8}$$

F_f is the feedforword term. F_b is the feedback term. First there exists intrinsic dynamic coupling in hybrid position/force control law (8) of mutil-joint robots. There no simple manner to uncouple the dynamic move along all directions in operational space. Second There exists serious nonlinear dynamic in equation (7). This inevitably leads a backward - looking design of controller. Third when parameters in equation (7) change, The designer have to re-design the controller to keep a acceptable system performance. The dynamic nature beyond equation (7) also enormously affect the dynamic performance. Above all made system dynamic performance of force control are very low, So how to improve the dynamic force control performance is a critical to apply widely force control in industry production.

As shown in figure 3, The general CMAC controller is similar to computed torque controllers. The CMAC module is used to predict the actuator torque required to follow a desired trajectory(force), and these torque are used as feedforward terms in parallel with a fixed-gain, linear PD feedback controller. A learning scheme is used to adjust the memory values of the CMAC module on-line based on observations of the robot input-output relationship, in order to form an approximate dynamic model of the robot in appropriate regions of the state space. The parameters of the PD controller are strictly selected to ensure system dynamic stability.

We rewrite the robot dynamic equation (7), its control torque are function of robot state, that is

$$\tau_c = f(\theta, \dot{\theta}, \ddot{\theta}, F_e) \tag{9}$$

if above function f is known, we can calculate the drive torque to follow a ideal trajectory. However the dynamic function (9) is usually unknown or can not be known accurately for a specific robot. To solve this problem, we lead CMAC network. We expect CMAC network to be able eventually improve

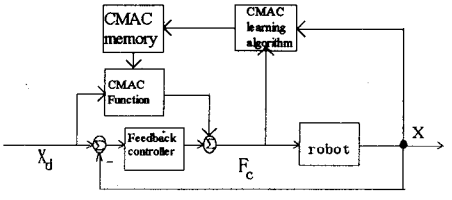


Fig 3. General CMAC controller

system control performance. The input states of CMAC network are $\theta, \dot{\theta}, \ddot{\theta}, F_e$. The output of CMAC network are forecast torque of function (9). In every control cycle, the desired robot state S_d as input vector are sent to CMAC network. CMAC network predict the estimate of control torque T_d to follows the desired robot states. In parallel with the output of the fixed-gain feedback PD controller, we get the actual control torque T. In the end of every control cycle, CMAC carry on a training. The robot state S_0 observed in last cycle is the input vector of CMAC network. The CMAC forecast output is $f(S_0)$. The actual control torque T_0 of this cycle are used as the desired output of CMAC network. So the memory weights adjustments of CMAC module are $\Delta w = \beta/\rho\left[T_0 - f(S_0)\right]$, here β is the training gain factor, $0 < \beta < 1$. In the beginning of CMAC learning, all memory weights are clear. So $f(S_d)$ is a zero vector. The control torque are equal the output of fixed-gain PD feedback control. After a series of control and learning of CMAC network, Most torque to control robot will supply by CMAC network. The contribution of the fixed-gain feedback PD control will less and less. It shows that CMAC network have successfully learned the input-output relationship of robot dynamic. If the manipulator moves in the region inside some neighboring of training regions, CMAC network will supply the enough control torque.

IV. SIMULATION

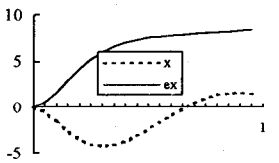
In order to test the performance of the based-CMAC hybrid position/force controller, we let the manipulator move along two trajectories. For the comparison, First the manipulator is controlled by pure PD controller to move along the two trajectories. Then we add CMAC network in parallel with the PD controller. Along the first trajectory, we carry on many times learning. Then Along the second we carry on one more time learning to verify the performance of the CMAC-based hybrid controller for the repeated control and non-repeated control.

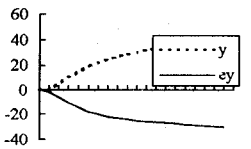
Trajectory 1:XY Axis position control mode. Z Axis: force control mode. desired trajectory: $x_d(0)=0$; $x_d(1.0)=0.01$; $y_d(0)=0$; $y_d(1.0)=0.01$; $fz_d(t)=-10.0$; $t \in [0\ 1.0]$

Trajectory 2:XY Axis position mode. Z Axis: force control mode. desired trajectory: $x_d(0)=0$; $x_d(1.0)=0.02$; $y_d(0)=0$; $y_d(1.0)=0.02$; $fz_d(t)=-10.0$; $t \in [0\ 1.0]$

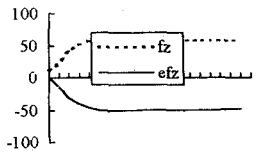
The parameters of the fixed-gain PD controller are kpx=5000, kdx=300, kpy=5000, kdy=300, kpz=1.0, kdz=300, The parameter of CMAC network are R=255, C=32, $\beta = 0.1$.

1. Pure PD control (Non CMAC learning)

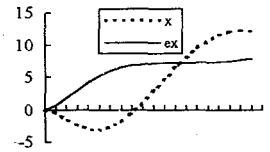




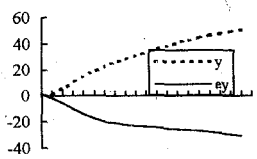
pure PD control trajectory 1: X



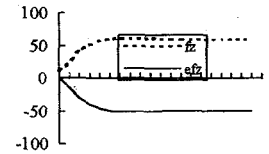
pure PD control trajectory 1: Y



pure PD control trajectory 1: :Z



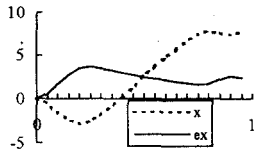
pure PD control trajectory 2: X

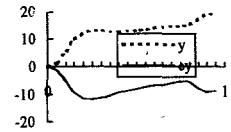


pure PD control trajectory 1: Y

pure PD control trajectory 2: Z

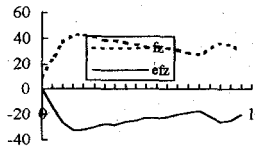
2.PD+CMAC Learning Control (Along trajectory 1)

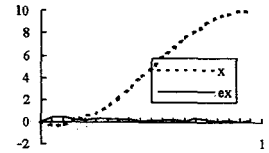




First Move: X

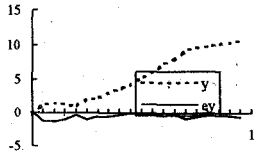
First Move: X

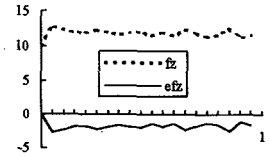




First Move :Z

Twentieth Move: X

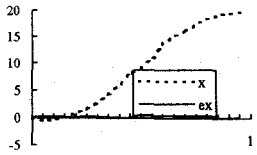


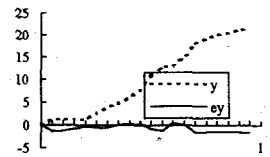


Twentieth Move: Y

Twentieth Move: Z

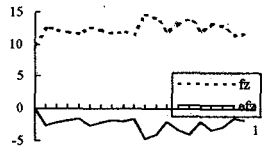
3.PD+CMAC (non-repeated control, Along trajectory 2)





trajectory 2: X

trajectory 2: Y



trajectory 2: Z

V. DISCUSSION AND CONCLUSION

The fixed-gain of PD feedback controller is small, the control errors of pure PD hybrid position/force controller are very huge. As a result of CMAC learning, the errors of position and force are smaller than pure PD control even if in the first learning. The simulation of CMAC learning algorithm in robot hybrid force/position control has shown CMAC learning controller will eventually improve the performance of robot force control. CMAC learning control are not only suitable for repetitive control, but also suitable for non-repetitive control mode. Moreover CMAC has a rapid learning capability, can rapid improve force control precision. However CMAC algorithm has a huge require for memory. The use of hash storage algorithm will lead so called data collision problem. CMAC will convergence to a limit cycle or the minimal capture zone. This will lead system control precision can not be reduced unlimited. The software realize of CMAC algorithm have a long control time, can not apply to actual real-time robot control. In our simulation, cycle time is 19ms for $\rho = 32$, cycle time is 41ms for $\rho = 64$. We have to wait for appearance of CMAC algorithm supported by the specific hardware.

REFERENCES

1. D.E. Whitney, Historical Perspective and State of the Art in Robot Force Control, Proc. IEEE Int. Conf. on Robotics and Automation pp.262,1985.
2. H. Asada, J.E. Slotine. Robot Analysis & Control. 1986.
3. C. An, J. M. Hollerback. Dynamic Stability Issues in Force Control of Manipulators, Proc. IEEE Int. Conf. on Robotics Automation. pp. 890-897. April 1987.
4. R.K. Roberts, R.P. Paul, E.M. Hillberry, The effect of wrist force sensor stiffness on the control of control manipulators, Proc. IEEE Conf. Robotics and Automation, St. Louis, MAr. 25-28, pp. 269-274. 1985.
5. S.D. Eppinger, W.P. Seering, Three Dynamic Problems in Robot Force Control, IEEE. International Conference on Robotics and Automation, 1989, pp. 392-397.
6. W.T. Miller, F.H. Glanz, L.G. Kraft, Application of a General Learning Algorithm to the Control of Robotic Manipulators, The International Journal of Robotic Research, vol.6, No.2, Summer 1987, pp84-98.
7. P.C. Parks, J. Militzer, A comparision of five algorithms for the training of CMAC memories for learning Control system. Automatica, vol. 28, No. 5, pp1027-1035, 1992.