

# Adaptive Semi-autonomous Robotic Neurocontroller

Chad Cox, John Edwards, Richard Saeks, Robert M. Pap

Accurate Automation Corporation  
7001 Shallowford Road  
Chattanooga, TN 37421

Karl Mathia

Department of Electrical Engineering  
Portland State University  
Portland, OR

## ABSTRACT

We have designed a neural network semiautonomous robotic arm controller. This controller performs end-effector path planning, inverse kinematics, and joint control to move the end-effector to a commanded position. We have tested the adaptive neural joint controller and inverse kinematics in simulation. The joint controller has been tested on two real arms. These real arms are the Extendable Stiff Arm Manipulator (ESAM) and the Proto-Flight Manipulator Arm (PFMA). Both of these arms are very different, yet the same unmodified joint controller software can control them both. The controller has also show tremendous adaptability to large payload variations. It has been shown to adapt to a 35 pound end-effector payload on the ESAM from a zeroed initial state. This ability to handle different arms and payloads is due to the fact that the controller makes no assumptions as to the arm's dynamics or payload. The same tests performed on a decentralized PD controller showed that the neural network controller is superior.

## 2. CONTROLLER DESIGN

Accurate Automation Corporation has been investigating the use of neural networks for robotic control applications<sup>1,2,3,4,5</sup>. This project addresses the issues in path planning, inverse kinematics, and joint (motor) controls together with the development of the neurocontrol hardware required to implement the resultant control system and interface it to the PFMA and ESAM robotic arms. Our objective under the NASA contract is to use neural networks in telerobotic operation of an arm like the Remote Manipulator System (RMS) on board the Space Shuttle.

This design is shown in Fig. 1. At the highest level is path planning. At the mid level is inverse kinematics. At the lowest functional level is decentralized adaptive joint control. We have developed neural networks for multiple joint controllers and inverse kinematics, and are developing a path planning network. Our design can be fed goals, either from a human, computer, or from other neural networks which may be developed in the future.

### 3. JOINT CONTROLLER

#### 3.1. Decentralized adaptive joint control

Our joint controller is based on a Decentralized Model Reference Adaptive Control (DMRAC) algorithm<sup>6</sup>. Neural networks are used to adapt the coefficients of conventional controllers. Our neurocontroller has the following advantages:

- o A complex model of the remote manipulator dynamics is not required;
- o Each joint can be controlled by a separate, independent controller;
- o Each joint controller depends only on the current angle and the reference angle;
- o The decentralized scheme lends itself to a parallel implementation;
- o Global asymptotic stability of the control system is assured.

The neural networks allow for greater parallelism and faster performance than the DMRAC algorithm with conventional integration techniques. Serial simulations have shown that, even without the parallelism advantage, our controller performs as fast as or faster than the DMRAC algorithm with Runge-Kutta integration. Our controller is as accurate as DMRAC the first time that the controller experiences a movement. Control becomes more accurate as the controller gains experience. We have tested the controller on two real robotic arms, the NASA Extendable Stiff Arm Manipulator (ESAM), and the NASA Proto-Flight Manipulator Arm (PFMA).

Although the primary motivation for the present project has been telerobotic control with the long term goal of controlling the NASA Remote Manipulator System, much of the underlying theory is applicable to any electromechanical system. Our joint controller assumes only that the plant:

- o is described by a second order dynamical system model with positive slowly varying mass and,
- o any higher order terms or coupling terms (from other joints or subsystems) are sufficiently small to be treated as disturbances.

A stable second order error model,  $e_r(t)$ , and a feedback controller are defined so that

$$e(t) - e_r(t)$$

is asymptotic to zero. In this case, since the error model is stable it is asymptotic to zero and hence  $e(t)$  will also be asymptotic to zero. The required feedback controller is designed by constructing a Lyapunov function for  $e(t) - e_r(t)$  and defining the feedback control coefficients to guarantee that the conditions of the Lyapunov Theorem are met. This then results in a system which is guaranteed to be stable, asymptotically track the desired trajectory, and be robust.

Under the limited assumptions that the plant is second order with positive slowly varying mass it is always possible to do this without a-priori knowledge of the plant parameters. This serves as the basis for our adaptive joint controller which assumes no a-priori knowledge of the joint kinematics automatically adapting to length, mass and load changes.

Since an electric motor can also be modeled by a second order equation with positive mass (modula small higher order terms which can be treated as disturbances) precisely the same adaptive control techniques developed for joint control can be applied to motor control. The ability of the algorithm to adapt to changing parameters on-line will permit one to design a motor controller which can adapt to changing loads on-line. When the adaptive control algorithm is augmented by our neural network training techniques for fine tuning the algorithm and implementing the controller, a real-time variable load adaptive motor controller can be developed.

A single joint controller is shown in Figure 2. Our neural network implementation uses the state vector and information about the reference trajectory to adapt the coefficients of a PID/PVA controller. The PID/PVA controller, using tracking errors and reference joint angles, computes the torque or current to be applied to a joint or joint motor.

### 3.2. Experimental results

We have tested the controller on the NASA Extendable Stiff Arm Manipulator (ESAM) arm. This arm was built around 1975. It has two revolute joints near its base, the first yaw and the second pitch. It has a prismatic joint for extending its largest link, a wrist yaw joint, a wrist pitch joint, a wrist roll joint, and an Adam hand. When the largest link is unextended (as in the experiments below), the distance from the center of revolution of the first pitch joint and the tip of the hand is about 57 inches. The first two joints are driven by Inland Motors harmonic drive servomotors drawing 2 1/2 amperes. Documentation of many of the dynamical parameters of this old arm is not available. Since our controller is intended to work on any arm regardless of dynamical quantities, we made no attempt to measure any of these. We rewired much of ESAM and built entirely new servocontroller hardware.

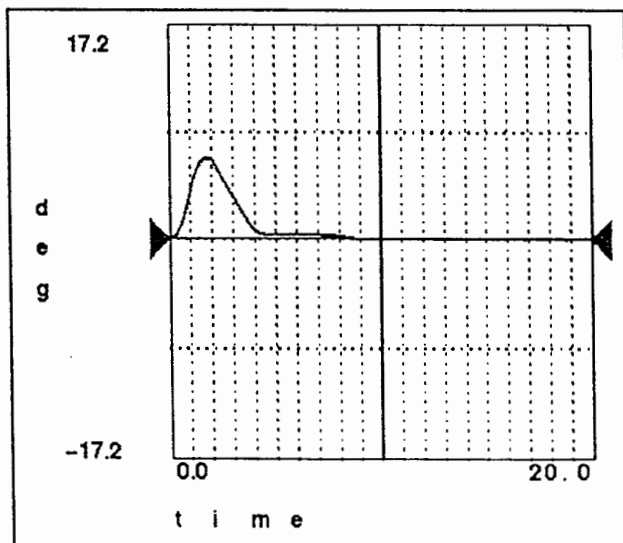


Fig. 3. Tracking error.

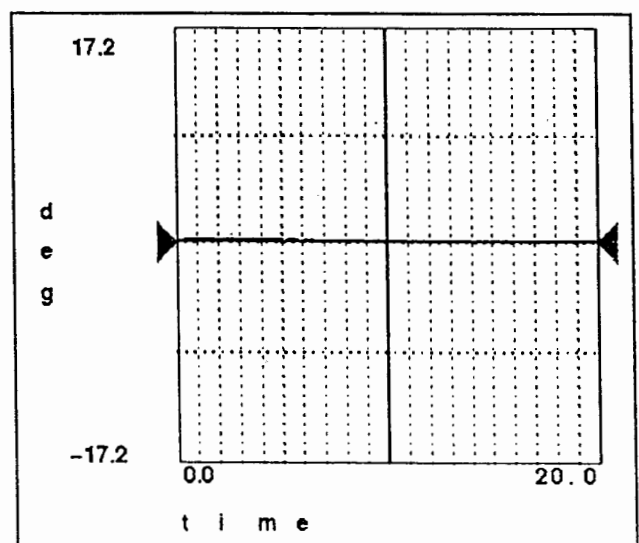


Fig. 4. Tracking error.

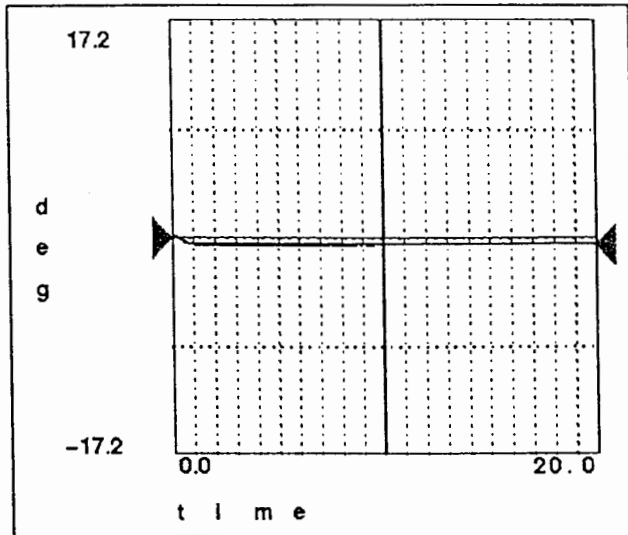


Fig. 5. Tracking error.

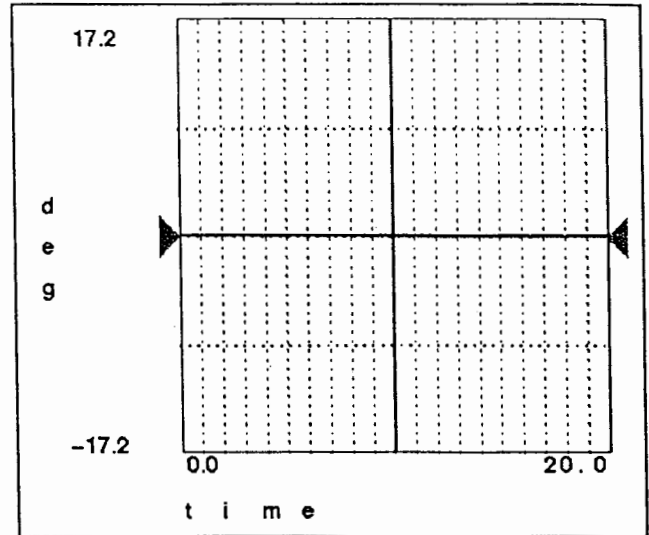


Fig. 6. Tracking error.

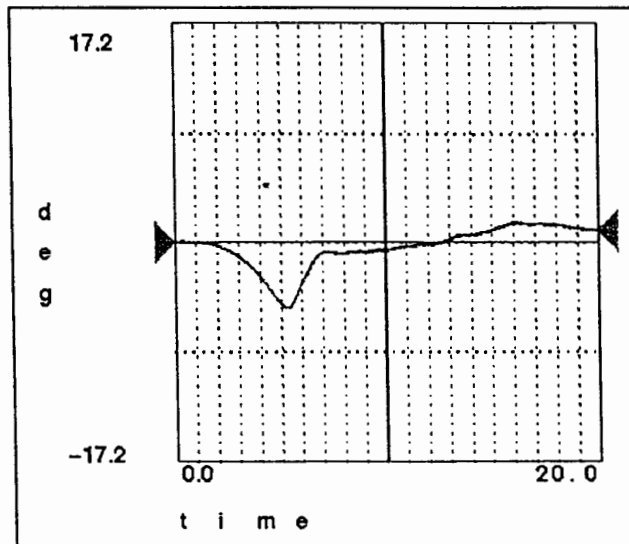


Fig. 7. Tracking error.

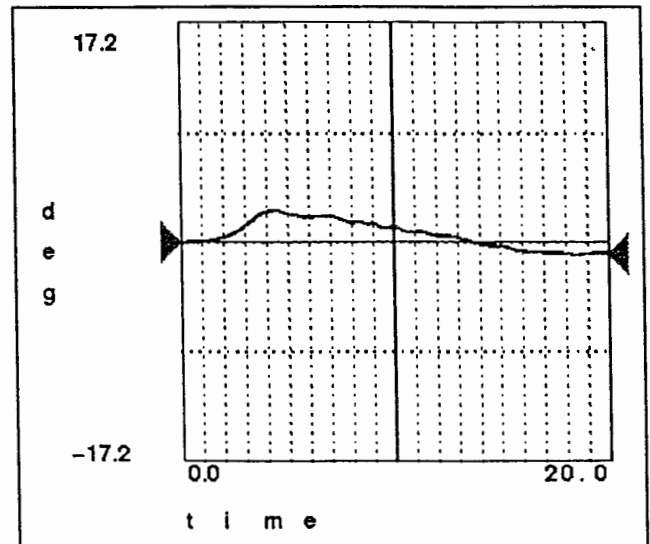


Fig. 8. Tracking error.

Figs. 3-10 were made from error data taken during tests of the controller on the ESAM arm. Each graph shows the difference between the reference trajectory and the actual trajectory of the shoulder joint. This joint produces motion perpendicular to the floor. During the experiment associated with figure 3, ESAM was commanded to hold a position such that its main link was roughly parallel to the ground. A 25 pound cinder block and 10 pounds of iron weight were hung approximately 32 inches from the center of revolution of the first pitch joint. The neurocontroller's memory was erased and the initial coefficient values were set to zero. The graph shows that the arm initially dropped. As the neurocontroller adapted, the arm returned to the commanded position. Figure 4 shows the same experiment repeated again. The controller held the arm steady despite the weight. Figure 5 shows the error after the weight was removed. Despite starting with large coefficients in expectation

of the large weight, the neurocontroller did not go unstable. A slight offset is all that is visible. Figure 6 shows this experiment repeated again. The neurocontroller held the arm exactly where commanded.

Similar experiments with a PD controller hand tuned to hold up the weight of the arm alone result in very good results without an additional load. As soon as the load is added, the arm drops until the error grows large enough that enough control voltage is generated to stop the fall of the arm. Thus, the arm sits at a large constant offset from the commanded position. Without being able to increase the magnitude of its coefficients, it cannot pull up the load.

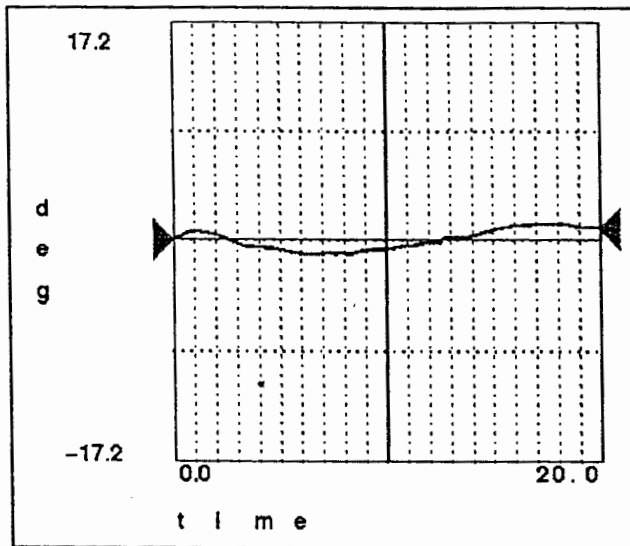


Fig. 9. Tracking error.

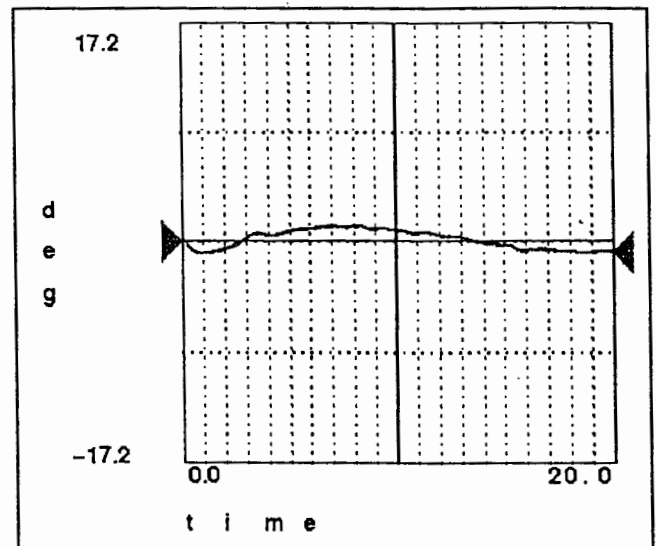


Fig. 10. Tracking error.

Figures 7-10 show the difference between cycloidal reference trajectories and actual trajectories for the base yaw joint. The base joint produces motion parallel to the floor. Figure 7 shows an experiment where the base joint was commanded to move from 150 degrees to 90 degrees. The initial coefficients were zero and the neural network's memory was erased before starting the experiment. This neurocontroller performs adequately from the start. Many motions of the arm are not required to insure an adequate performance, though repeated motions do result in improvement. After initial convergence, the neurocontroller is able to move the arm without the initial large hump on the error curve.

Figure 8 shows the following motion back from 90 degrees to 150 degrees. Figure 9 shows the motion after that from 150 degrees to 90 degrees. Figure 10 shows the next motion from 90 degrees to 150 degrees. The neurocontroller's performance improves with practice. The neurocontroller performs similarly on the PFMA, which has substantially differing kinematical and dynamical parameters.

## 4. INVERSE KINEMATICS

### 4.1. Inverse Kinematics with Neural Networks

Initially, we investigated a neural network inverse kinematics solution based on <sup>7</sup>. Like with this technique, our multi-layer perceptrons are trained to learn the inverse kinematic mapping of a robotic arm. Then during the operation of the neural network, the course solution of the neural network is refined using the Newton-Raphson method. This design provides a fast solution without compromising accuracy. Our inverse kinematics design has two major improvements over the original technique. Our neural network implementation is trained for each joint, providing greater fault tolerance and less training time than a single central network. Each network is provided with both a goal cartesian end effector position and the present one. This allows each neural network to optimize and constrain the solution based on such things as amount of joint motion, distance from a nominal value, and expected energy expenditure.

Still, we found that this method has a number of disadvantages that we wished to overcome. First, the neural networks require extensive training. The training involves many perturbations of either the real arm or an equivalent simulated one. Once the neural networks are trained on one arm, they are useless on an arm with different kinematics. Second, the neural networks require many arithmetic operations. While these can be parallelized, we would like our algorithms to be implementable on conventional computer hardware. Third, in an implementation requiring many discrete solutions along an end-effector trajectory (as apposed to only end-point), the neural networks are actually a hinderance since they provide worse initial conditions than the solutions for the previous time steps.

So we have investigated an entirely new approach to the inverse kinematic problem. This approach involves a Linear Hopfield Network (LHN). The difficulty of complex inversion or no-invertability of robot inverse kinematics is circumvented by linearization of the forward kinematics of an appropriate energy function, which is then minimized by linear dynamic networks. The results of the dynamic optimization process are the 'best' joint angle rates which minimize the manipulator's position error. We call the network the Linear Hopfield Network because of its similarities with the original Continuous Hopfield Network <sup>8</sup>. A derivation as well as a further discussion and a convergence proof for the synchronous dynamics of the LHN are given in <sup>5</sup>. The simulated tracking control of a trajectory in 3-D Cartesian space with a three-joint robot manipulator demonstrates the performance of this approach.

The initial idea of using a neural network to circumvent a closed form solution to the inverse kinematics problem by linearizing the direct kinematics and then defining and minimizing an energy function was presented earlier<sup>9</sup>. To find the required joint angles for a desired manipulator position, the inverse kinematics problem is reduced to the numerical solution of a linear system. The linear system is specified and solved for each manipulator position, i.e. for each point on the manipulator's trajectory in Cartesian space. The required associated point in joint angle space is determined via an optimization process.

An appropriate energy function is represented in a form which can be minimized by a linear, fully connected, recurrent network. As opposed to the Continuous Hopfield Network, which has nonlinear transfer functions and is used as an autoassociator, the LHN has linear transfer functions and solves systems of linear equations. The LHN can be constructed to guarantee convergence to an optimal solution, here the 'best' joint angle velocity to minimize the manipulator's velocity error. The resulting discrete velocities are integrated to track the preplanned end-effector path. As an example, the simulated control of a three-joint manipulator in 3-D Cartesian space is presented.

#### 4.2. Experimental results

The LHN was used to generate joint trajectories for the simulated manipulator shown in figure 11. Each link is one meter long. One thousand discrete steps were used to generate the trajectories in figures 12-15. Figure 12 shows the three joint angles over time. The solid line represents joint  $q_1$ . The dashed line represents joint  $q_2$ . The other line represents joint  $q_3$ . All joint angles are in radians. Figures 13-15 show the three components of the end effector trajectory,  $x_1$ ,  $x_2$ , and  $x_3$ . All components are in meters. The dashed lines are the desired trajectories and the solid lines are the actual trajectories generated by the joint angles in figure 12. Note that in two of the three graphs, the trajectories coincide perfectly. The small error shown in figure 14 disappears when the number of discrete steps is increased to 1500.

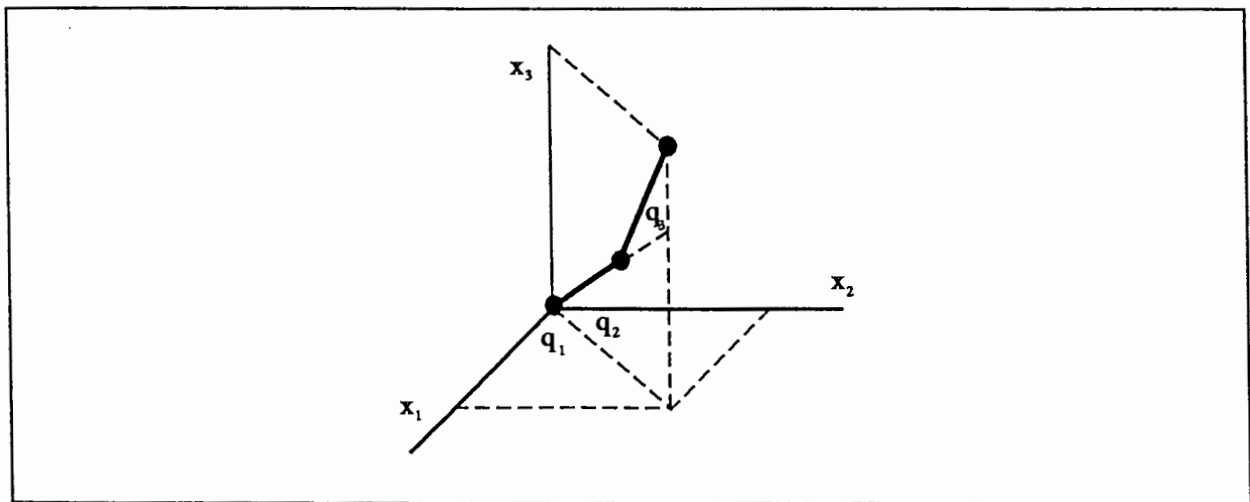


Fig. 11. 3-joint manipulator

### 5. CONCLUSION

Accurate Automation Corporation is designing a generic robot arm neurocontroller for use in space and underwater applications. Results have been obtained for the inverse kinematics and joint control portions. The inverse kinematic portion utilizes the speed of neural

networks without compromising accuracy. Simulation results show this to be a viable approach. Tests of the joint neurocontroller on real robotic arms have shown that the neurocontroller is fast and accurate. It is also robust in the presence of unknown dynamical parameters. The neurocontroller performs adequately on its first motion and improves with experience.

## 6. ACKNOWLEDGEMENTS

This work is sponsored by the National Aeronautics and Space Administration, George C. Marshall Space Flight Center under a Phase II Small Business Innovation Research contract NAS8-38967, The Department of the Navy, Office of Naval Research under contract N00014-92-C-0268 and OASN(RD&E), Product Assessment Division, and the National Science Foundation under Grants ECS-9015159 and ECS-9147774.

## 7. REFERENCES

1. Cox, C., J. Edwards, R. Saeks, M. Lothers, R. Pap, and C. Thomas. "A Neural Network for Joint and Motor Control", *World Conference on Neural Networks, Portland*. Vol. 3. pp. 350-353, Lawrence Erlbaum Associates: New Jersey, 1993.
2. Cox, C., R. Saeks, M. Lothers, R. Pap, and C. Thomas. "A Neurocontroller for Robotic Applications", *Proc. of the International Conference on Systems, Man, and Cybernetics, Chicago*. pp. 712-716. IEEE: Salem, 1992.
3. Parten, C.R., R.M. Pap, and M.D. Lothers. "Neural Network Robotics Control", *Proc. of WESCON/90*. Western Periodicals: North Hollywood, 1990.
4. Parten, C.R., R.M. Pap, and C.R. Thomas. "Neurocontrol Applied to Telerobotics for the Space Shuttle", *Proc. of the Inter. Neural Network Conf., Paris*. pp. 229-236. Kluwer Academic Publishers: Boston, 1990.
5. Mathia, K. and R. Saeks. "Inverse Kinematics Via Linear Dynamic Networks," Submitted to *World Conference on Neural Networks, San Diego*, 1994.
6. Seraji, H. "Decentralized Adaptive Control of Manipulators: Theory, Simulation, and Experimentation." In *IEEE Transactions on Robotics and Automation*. Vol 5. No. 2., pp. 183-201., 1989.
7. Guez, A. and Ahmad, Z. Improving the Solution of the Inverse Kinematics Problem Using Neural Networks. *Journal of Neural Network Computing*. Spring. pp. 21-32, 1990.
8. Hopfield, J. and Tank D. "Computing with Neural Circuits: A Model," *Science*, Vol. 233, August. pp. 625-633, 1986.
9. Guo, J. and Cherkassky, V. "A Solution to the Inverse Kinematics Problem in Robotics Using Neural Network Processing," *International Joint Conference on Neural Networks (IJCNN), Washington, D.C.*, Vol. II, pp. 299-304, IEEE: New York, 1989.

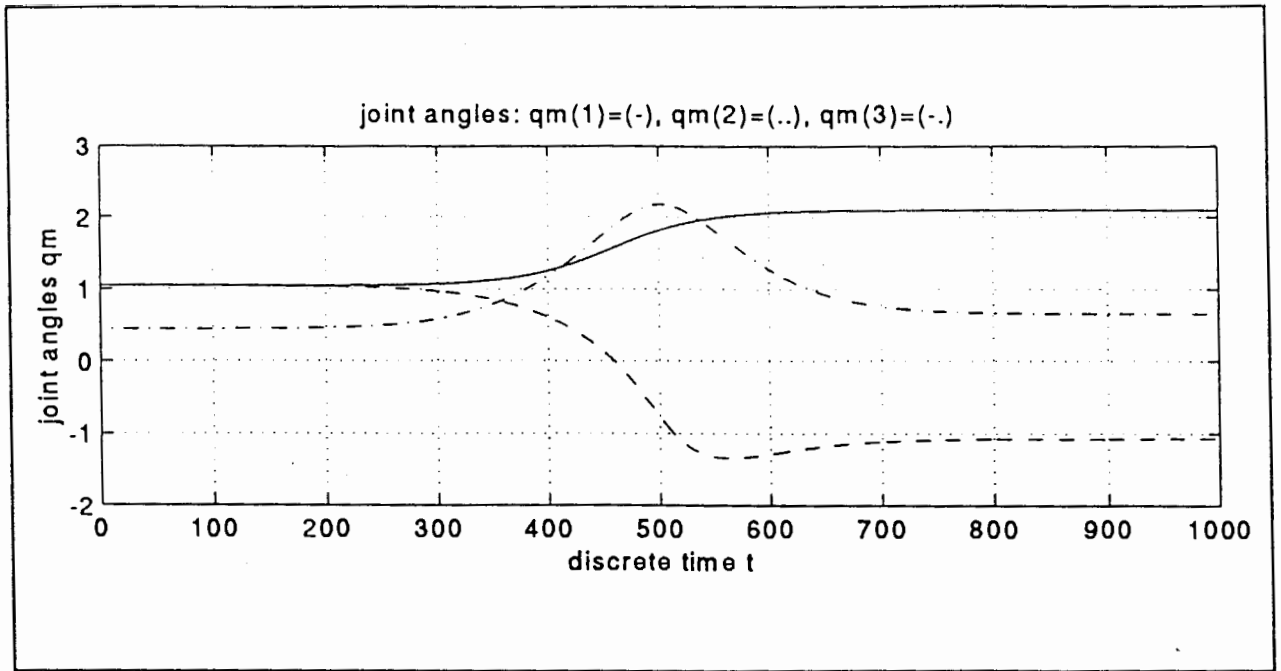


Fig. 12. Joint angles,  $q_1$ ,  $q_2$ , and  $q_3$  over time.

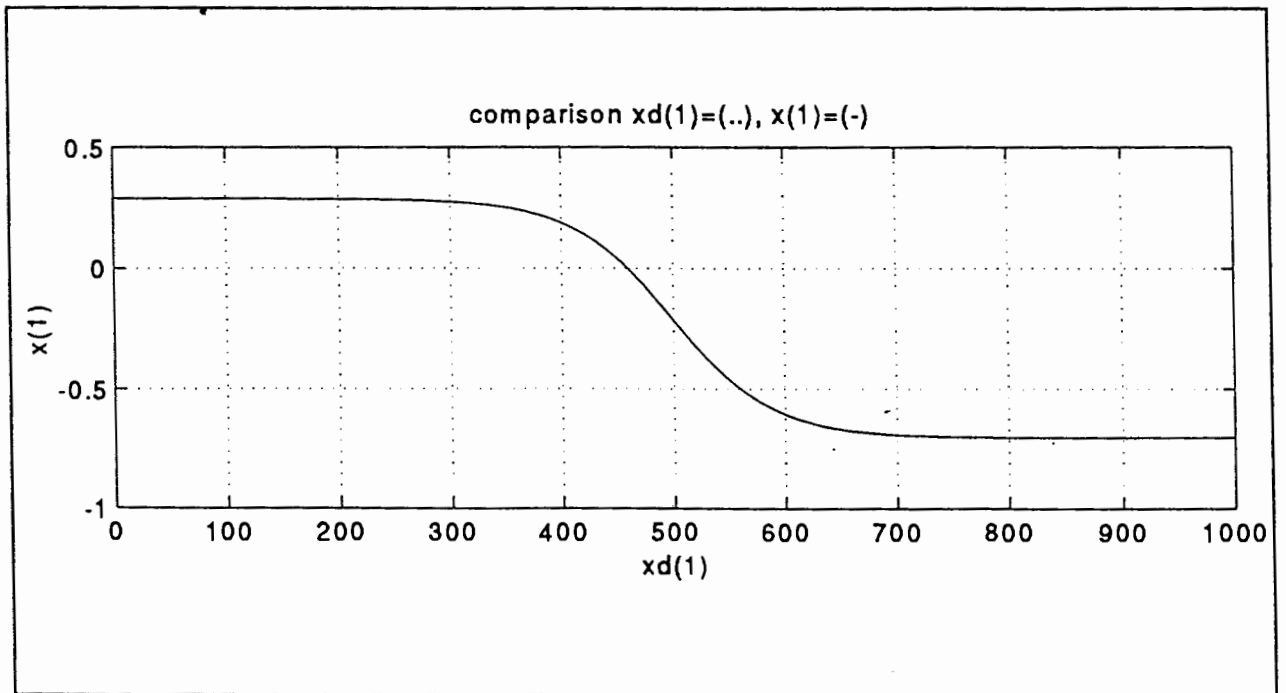


Fig. 13.  $x_1$  cartesian end-effector position component

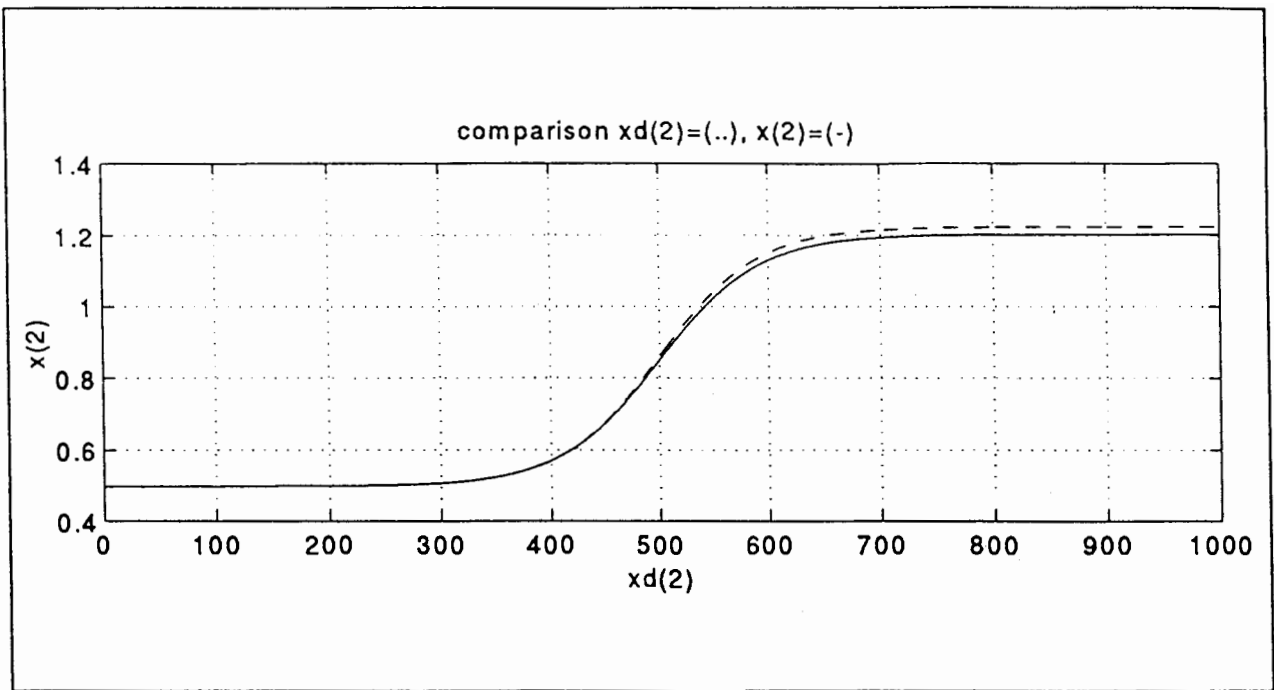


Fig. 14.  $x_2$  cartesian end-effector position component

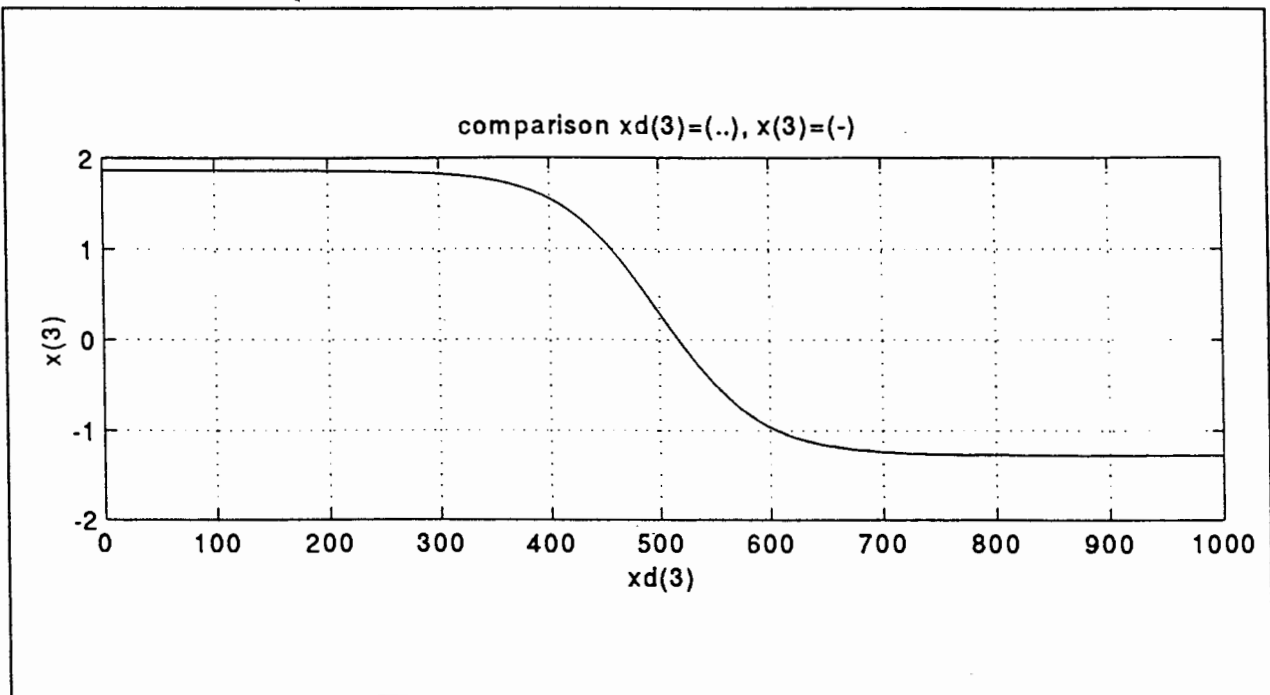


Fig. 15.  $x_3$  cartesian end-effector position component